



HAL
open science

Spatially-Distributed Parameter Identification by Physics-Informed Neural Networks illustrated on the Shallow-Water Equations

Hugo Boulenc, Robin Bouclier, Pierre-André Garambois, Jerome Monnier

► **To cite this version:**

Hugo Boulenc, Robin Bouclier, Pierre-André Garambois, Jerome Monnier. Spatially-Distributed Parameter Identification by Physics-Informed Neural Networks illustrated on the Shallow-Water Equations. 2024. hal-04721068

HAL Id: hal-04721068

<https://hal.science/hal-04721068v1>

Preprint submitted on 4 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Spatially-Distributed Parameter Identification by Physics-Informed Neural Networks illustrated on the Shallow-Water Equations

Hugo Boulenc^a, Robin Bouclier^{a,c,d}, Pierre-André Garambois^b and Jérôme Monnier^a

^aInstitut de Mathematiques de Toulouse (IMT), Univ Toulouse, UPS-INSA-CNRS UMR5219, 135 avenue de Rangueil, Toulouse, 31400, France

^bINRAE / UMR RECOVER, 3275 Rte Cézanne, Aix-en-Provence, 13100, France

^cInstitut Clement Ader (ICA), Univ Toulouse, INSA-ISAE-Mines Albi-UPS-CNRS UMR 5312, 3 rue Caroline Aigle, Toulouse, F-31400, France

^dInstitut Universitaire de France (IUF), , , , France

ARTICLE INFO

Keywords:

Data Assimilation
Inverse Problems
Parameter Identification
Physics-Informed Neural Networks
Hydraulics models
Shallow-Water Equations


ABSTRACT

Recent advancements in Physics-Informed Neural Networks (PINNs) offer promising opportunities for the identification of parameters of physical models based on ODEs and PDEs. This work revisits two representative PINN-based approaches for inverse problems, and apply them to fluid dynamics simulations. The first approach, here referred to as Fully-Parameterized PINN, develops a parameter-differentiable surrogate model through an offline training phase, followed by a rapid online parameter identification phase. This method treats physical parameters as Neural Network (NN) inputs, making it prone to the curse of dimensionality. Conversely, the second approach, termed Semi-Parameterized PINN (SP-PINN), integrates physical parameters as NN parameters, allowing for efficient inference regardless of dimension using automatic differentiation. The performance of these two methods is first assessed through numerical experiments using an ODE-based model (the Backwater Equation). Then, the SP-PINN is tested in a more representative scenario for identifying a $\mathcal{O}(10^3)$ -dimensional spatial friction parameter in a 2D Shallow-Water Equations model. Comparisons with the well-established and precision-validated Variational Data Assimilation (VDA) method are performed. The latter, even based on the adjoint technique, remains computationally expensive for high-dimensional parameter identification and can be complex to implement. This work highlights the attractiveness of SP-PINN in terms of simplicity and efficiency, thus establishing this strategy as a complementary approach or even a compelling alternative in parameter identification of real-world flow models.

1. Introduction

In recent years, the volume of data available for scientific research and engineering has grown tremendously. As a consequence, a considerable amount of research is currently devoted to combining these data with existing physical priors and mathematical models. This trend is particularly evident in environmental sciences such as weather forecasting (see, *e.g.*, [1, 2]), river hydraulics (see, *e.g.*, [3, 4]), and others, where the integration of observational data with complex models is crucial for understanding and predicting natural phenomena. From a methodological point of view, Data Assimilation (DA) has emerged as a powerful framework to achieve such a data-model coupling, in order, for instance, to update the state of a dynamical system or to estimate model parameters (see, *e.g.*, Asch et al. [5], Bocquet [6], Monnier [7] and references therein). Specifically, DA consists in solving the inverse problem of adjusting input parameters of a model so that its output aligns with observed data. DA techniques are generally divided into two major categories: filter methods (see, *e.g.*, [8, 9]) and variational methods (see, *e.g.*, [10, 11]), which are now combined for complex situations such as weather predictions for example.

In this work, we are interested in model parameter identification, especially when the parameter to be inferred is high-dimensional. In terms of application, we consider hydraulics modeling based on the Shallow-Water Equations (SWE, see, *e.g.*, [12, 13]). In this context, the parameters of interest can be spatially-distributed fields, such as friction or bathymetry fields, which are often inherently high-dimensional. For such problems, the conventional approach that has been developed over the years is Variational Data Assimilation (VDA, see [5, 6, 7, 11, 14] to name a few). VDA aims to reconcile observations and model outputs by minimizing a properly defined data-model discrepancy

 email address of the first author (H. Boulenc)

 URLofthefirstauthor (H. Boulenc)

ORCID(s):

functional using gradient descent. In high dimension, computing the gradient of the functional is tractable thanks to the introduction of the adjoint model technique, which requires solving only one additional direct model per iteration to obtain all the gradient components. However, the computational cost of solving an inverse problem using VDA is approximately $\mathcal{O}(10^2)$ times that of the direct model, which can be expensive for real-world applications. Nowadays, given the rapid advancements in the broad field of Artificial Intelligence, bringing Machine Learning techniques and VDA closer together may offer opportunities for faster computations, especially for high-dimensional parameters. This is what we propose to investigate in this paper by focusing on Physics-Informed Neural Networks (PINNs) methods to solve DA inverse problems.

To put it in a nutshell, PINNs enable the incorporation of physical constraints into the training of NNs in a weak sense, by adding a penalty term to the loss function. They were initially introduced by Psychogios and Ungar [15] and Lagaris et al. [16], and were later further developed and formalized using differentiable programming by Raissi et al. [17]. Recently, several libraries have been developed to facilitate their implementation (see, *e.g.*, [18, 19]). Historically, and still predominantly today, PINNs are rather used to solve direct problems (see, *e.g.*, [20, 21, 22]), yet they also appear well-suited for addressing inverse problems. For example, PINNs have been used to reconstruct dynamical states (see, *e.g.*, [23, 24, 25]), as well as to estimate physical model parameters from observational data (see, *e.g.*, [26, 27]). In the recently published review by Tanyu et al. [28], numerous NN-based methods for solving parameter identification problems are detailed and compared, particularly physics-informed methods. The latter work emphasizes two important approaches for parameter identification: the first one consists in approximating the parameter-to-state operator, while the second one focuses on approximating directly the state function. In addition to this investigation, a comparative analysis of the capacity of these two approaches to meticulously handle high-dimensional parameter identification problems would be of interest, as well as a comparison with more established and proven DA methods.

Therefore, the aim of the present work is to evaluate the capabilities of the two approaches described above for handling high-dimensional parameter identification problems and to compare their performance to the well-established VDA. Here, the approach consisting in the approximation of the parameter-to-state operator is naturally called Fully-Parameterized (FP), while the one focusing on the approximation of the state function is referred to as Semi-Parameterized (SP). More precisely, the first approach will be illustrated with the proposed Fully-Parameterized PINN (FP-PINN), which involves creating a NN-based parameter-differentiable surrogate model through a costly offline training phase to approximate the parameter-to-state operator, as in, *e.g.*, [29, 30], and then using this surrogate model in an online rapid parameter identification phase. The second approach, illustrated with what we call the Semi-Parameterized PINN (SP-PINN), consists in learning the physics and performing the parameter identification simultaneously by directly approximating the state function with the NN, as in, *e.g.*, Raissi et al. [17]. The FP-PINN introduced in the present work corresponds to the PI-DeepONet (see Wang et al. [30]), but with a simplified NN architecture, whereas the SP-PINN is related to the classical PINN introduced in Raissi et al. [17]. Furthermore, since in both of these approaches the NN training can be challenging, several best practices (Fourier features, pre-training, alternating minimization) that are crucial in the process are provided in this work.

Finally, further exploration is needed to better understand how these PINN-based methods perform relative to established techniques, particularly in high-dimensional parameter identification for real-world applications. Consequently, in this work, we also propose a comparison of the SP-PINN approach, which proves to be the most attractive among the two alternatives for high-dimensional parameter identification, with the well-established and accurate VDA method in a challenging fluid dynamics inverse problem scenario: a 2D SWE model applied to river flow, with a $\mathcal{O}(10^3)$ -dimensional spatial friction parameter to identify.

The paper is organized as follows. After this introduction, Section 2 introduces the general direct model, the inverse problem to be solved, and the flow models considered as applications. Section 3 presents, in a general manner, the studied PINN-based parameter identification approaches. It outlines the formalism and the differences between the two considered methods, as well as the algorithmic cares used to enable the training of such NNs. Then, Section 4 starts by presenting the numerical experiments conducted to evaluate the capacity of the two PINN-based methods to handle high-dimensional inverse problems using a simplified flow model. Thereafter, one of the two methods is tested on a more representative, high-dimensional inverse problem based on a 2D SWE model, and is compared with

conventional VDA. Eventually, Section 5 offers a brief summary of the key findings and main contributions of this study, and motivates future research based on these findings.

2. Problems and models setup

This section begins by introducing, in a general manner, the direct model and the inverse problem to be solved. Subsequently, the specific models used for application purposes are presented, consisting more precisely of the Backwater Equation and the SWE flow models. In each case, the Manning-Strickler friction coefficient is considered as the spatially-distributed parameter to be identified.

2.1. The direct model

Let us define a function \mathbf{k} on a spatial domain $\Omega \subset \mathbb{R}^d$ representing a spatially-distributed, potentially high-dimensional, parameter of the model, $\mathbf{k} : x \mapsto \mathbf{k}(x)$, $x = (x_1, \dots, x_d) \in \Omega$. In the sequel, we will seek to infer this parameter given some observations. Given a final time instant T , the general time-dependent model reads as:

$$\partial_t y(x, t) + A(\mathbf{k}; y)(x, t) = L(\mathbf{k})(x, t), \quad \forall (x, t) \in \Omega \times]0, T]. \quad (1)$$

Eq. (1) is accompanied by adequate Boundary Conditions (BC) on $\partial\Omega \times]0, T]$ and Initial Conditions (IC) on Ω . The unknown function $y(x, t)$ represents the state of the modeled system, A is a differential operator parameterized by \mathbf{k} , and L is a forcing term that may also be parameterized by \mathbf{k} . Typical particular cases of Eq. (1) are:

- Ordinary Differential Equations (ODE) in variable t (x is frozen) such as the Backwater Equation presented in Section 2.3.1.
- Partial Differential Equations (PDE), such as the Viscous Burgers' equation, $A(\mathbf{k}; y)(x, t) = -\mathbf{k} \partial_{xx}^2 y(x, t) + y \partial_x y(x, t)$, or the 2D SWE presented in Section 2.3.2.

Let \mathcal{K} and \mathcal{Y} be Banach spaces, representing the input parameter space and the model solution space, respectively. We introduce the parameter-to-state operator \mathcal{M} as:

$$\mathcal{M} : \mathbf{k} \mapsto y^{\mathbf{k}}, \quad \forall \mathbf{k} \in \mathcal{K}, \quad (2)$$

with $y^{\mathbf{k}} \in \mathcal{Y}$ the (unique) solution of Eq. (1) for a given parameter function \mathbf{k} .

2.2. The inverse problem

The inverse problem here consists in identifying the input parameter \mathbf{k} from given observations (dataset) \mathcal{Z}_{obs} , $\mathcal{Z}_{obs} = \{z_{obs}^{(i)}(x, t)_{obs}^{(i)}\}_{i=1, \dots, N_{obs}}$ of the physical system, given on a grid $\mathcal{X}_{obs} = \{(x, t)_{obs}^{(i)}\}_{i=1, \dots, N_{obs}}$ defined over $\Omega \times]0, T]$. As classically done in DA problems, we introduce the observation operator Z which maps from the physical states space \mathcal{Y} onto the observations space, where the finite-dimensional vector z_{obs} resides. This enables to compare model outputs with the given data. Following the VDA approach (see *e.g.*, [6, 7]), one identifies \mathbf{k} by minimizing a cost functional $j(\mathbf{k}) = J_{obs}(y^{\mathbf{k}})$, with:

$$J_{obs}(y) = \frac{1}{N_{obs}} \|Z(y) - z_{obs}\|_{R^{-1}}^2, \quad (3)$$

with R an error covariance matrix. The estimation is obtained by solving:

$$\mathbf{k}^* = \underset{\mathbf{k} \in \mathcal{K}}{\operatorname{argmin}} (j(\mathbf{k})), \quad (4)$$

using a numerical optimization method.

2.3. The considered flow models

As mentioned in the introduction, our interest in this work, in terms of application, lies in hydraulics models. We investigate two flow models:

1. the Backwater Equation, which is a fundamental 1D non-linear model in hydraulics,
2. the 2D SWE system as it is operationally employed, *e.g.* in complex flood dynamics (see, *e.g.*, [12, 13, 31]).

2.3.1. The Backwater Equation (1D ODE)

The general form of a \mathbf{k} -parameterized ODE may read as: $y'(x) = F(\mathbf{k}; y)(x)$. In the present case, the physical state y corresponds to the water height h in m , and $x \in \Omega \subset \mathbb{R}$ is the space variable in m . The parameter function \mathbf{k} represents the spatially-distributed Strickler friction coefficient \mathbf{K}_s , in $m^{1/3}/s$. This parameter is utilized in the Manning-Strickler formula, which embeds the forcing effect of the riverbed friction on flow dynamics (see, *e.g.*, [12, 13, 31]).

As a consequence, the considered ODE more specifically reads:

$$y'(x) = F(\mathbf{K}_s; y)(x), \quad \forall x \in \Omega,$$

$$\text{where } F(\mathbf{K}_s; y)(x) = -\frac{z'_b(x) + S_f(\mathbf{K}_s; y)(x)}{1 - Fr^2(y(x))}, \text{ with } S_f(\mathbf{K}_s; y)(x) = \frac{q^2}{\mathbf{K}_s^2(x)y^{10/3}(x)} \text{ and } Fr^2(y(x)) = \frac{q^2}{gy(x)^3}. \quad (5)$$

In the above equation, z'_b represents the bathymetry variation, S_f the friction term while Fr stands for the dimensionless Froude number. Additionally, the IC is expressed as $y(x_{IC}) = y_{IC}$, where x_{IC} depends on the flow regime. For further insights, interested readers are encouraged to refer to [12, 13].

2.3.2. The Shallow-Water Equations (2D hyperbolic PDE system)

For the 2D SWE, the general form of Eq. (1) needs to be considered, with the physical state y corresponding to the vector $(h, q)^T$ and $x = (x_1, x_2)^T \in \Omega \subset \mathbb{R}^2$. Again, h represents the water height in m . Then, $q = (q_1, q_2)^T$ stands for the flow rate per unit width in $m^3/s/m$ along x_1 and x_2 , respectively. As in Section 2.3.1, the parameter function \mathbf{k} represents the spatially-distributed Strickler friction coefficient \mathbf{K}_s , in $m^{1/3}/s$. For application of state-of-the-art VDA methods for identifying high-dimensional parameters on this hydraulics model, see, *e.g.*, [32, 33, 34].

In its conservative form, the 2D SWE reads as (see, *e.g.*, [12, 13, 35]):

$$\partial_t y + \partial_{x_1} \mathbf{F}(y) + \partial_{x_2} \mathbf{G}(y) = \mathbf{S}_g(y) + \mathbf{S}_f(\mathbf{K}_s; y), \quad \text{in } \Omega \times]0, T],$$

$$\text{where the physical state reads } y = \begin{pmatrix} h \\ q_1 \\ q_2 \end{pmatrix} \text{ and the fluxes } \mathbf{F}(y) = \begin{pmatrix} q_1 \\ \frac{q_1^2}{h} + \frac{gh^2}{2} \\ \frac{q_1 q_2}{h} \end{pmatrix} \text{ and } \mathbf{G}(y) = \begin{pmatrix} q_2 \\ \frac{q_1 q_2}{h} \\ \frac{q_2^2}{h} + \frac{gh^2}{2} \end{pmatrix}. \quad (6)$$

$$\text{The source terms are given by } \mathbf{S}_g(y) = \begin{pmatrix} 0 \\ -gh\partial_{x_1} z_b \\ -gh\partial_{x_2} z_b \end{pmatrix} \text{ for gravity and } \mathbf{S}_f(\mathbf{K}_s; y) = \begin{pmatrix} 0 \\ -g\frac{\|q\|}{\mathbf{K}_s^2 h^{7/3}} q_1 \\ -g\frac{\|q\|}{\mathbf{K}_s^2 h^{7/3}} q_2 \end{pmatrix} \text{ for friction.}$$

The BC and IC are prescribed, respectively, on $\partial\Omega \times]0, T]$ (with $\partial\Omega = \Gamma_{in} \cup \Gamma_{wall} \cup \Gamma_{out}$), and on Ω . For real-world applications, the BC have to be mixed. In the context of this work, they are as follows. At the inflow boundary Γ_{in} , a discharge (flow rate) time series is prescribed: $q(x, t) = Q_{in}(t)$ on $\Gamma_{in} \times]0, T]$, where $Q_{in}(t)$ represents the imposed discharge over time. At the outflow boundary Γ_{out} , a constant water height is enforced: $h(x, t) = h_{out}$ on $\Gamma_{out} \times]0, T]$, see, *e.g.*, Couderc et al. [35] for further details. At the remaining boundaries denoted here as Γ_{wall} , standard "wall conditions" are applied, see [36] for details.

3. Physics-Informed Neural Networks methods for parameter identification

In this section, the two PINN-based approaches investigated for parameter identification are described within a DA framework, with particular emphasis on their capacity to handle high-dimensional parameters. This section begins with a general and concise overview of the methods before delving into the detailed mathematical formalism. Next, the two

PINN approaches (Fully-Parameterized and Semi-Parameterized, respectively) are described in detail, highlighting the differences in how they account for the parameter to be inferred. Finally, algorithmic cares that enable the training of such NNs are presented.

3.1. The PINN-based approaches in a nutshell

We start here by presenting the key components of the aforementioned approaches. All over the section, we refer to Figs. 1 and 2 for an overview of the methods, and to Figs. 3 and 4 for detailed perspectives. From a global point of view, three tasks are necessary to derive the methods:

1. Definition of a loss function J_{res} based on the model residual $R(\mathbf{k}; y) = \partial_t y + A(\mathbf{k}; y) - L(\mathbf{k})$ (see Eq. (1)), and connection with J_{obs} measuring the discrepancy with the observations dataset \mathcal{Z}_{obs} (see Section 2.2).
2. Setting up a NN denoted by \mathcal{N} , $\mathcal{N} : I \mapsto O$, with I its input and O its output. The expressions of I and O depend on the choice of PINN approach:
 - The first approach corresponds to the FP-PINN, see Figs. 1 and 3. In this alternative, the primary objective is to build, in an *offline* phase, a \mathbf{k} -differentiable surrogate model that approximates the model output. This involves approximating the operator $\mathcal{M}_{FP} : (\mathbf{k}; (x, t)) \mapsto y^k(x, t)$ for a given set of parameter functions \mathbf{k} , with y^k the model solution defined in Eq. (2). Thus, $I = (\mathbf{k}; (x, t))$ and $O = \tilde{y}(\mathbf{k}; (x, t))$ with $\tilde{y}(\mathbf{k}; \cdot)$ approximating y^k for a given set of \mathbf{k} .
 - The second approach corresponds to the SP-PINN, see Figs. 2 and 4. Here, the aim is to learn the model behavior and identify the parameter function \mathbf{k} all at once during the NN training. This involves approximating the operator $\mathcal{M}_{SP} : (x, t) \mapsto y^k(x, t)$ for a given parameter function \mathbf{k} that is learned during the NN training. Thus, $I = (x, t)$ and $O = \tilde{y}_k(x, t)$ with \tilde{y}_k approximating the model solution y^k for a given learned parameter function \mathbf{k} . Note that here, \mathbf{k} is not an input of \mathcal{N} , contrary to the FP-PINN case.
3. Training of the NN, which also depends on the choice of PINN approach:
 - For the FP-PINN, the training is split into two phases. First, the NN is trained in a costly *offline* phase to build a \mathbf{k} -differentiable surrogate model that approximates the model output, by minimizing J_{res} for a set of parameter function \mathbf{k} . Then, a cheap *online* inference phase can be carried out, which consists in minimizing J_{obs} by gradient descent using the \mathbf{k} -differentiable surrogate model built in the offline phase. After the online phase, an approximation of the parameter function explaining at best the given observations, $\mathbf{k}^* \approx \mathbf{k}^{true}$, is found.
 - On the contrary, for the SP-PINN, only one training phase is required. There is therefore no longer a notion of offline and online phases. Here, \mathbf{k}^* and its corresponding solution $\tilde{y}_{\mathbf{k}^*} \approx y^{\mathbf{k}^{true}}$ are obtained on the fly during the NN training, which consists in minimizing J_{res} and J_{obs} all at once.

3.2. Formalism

We now detail the mathematical formalism behind these methods. To do so, we first introduce the used NN and then define the various physics-informed loss functions that complement the data-driven one.

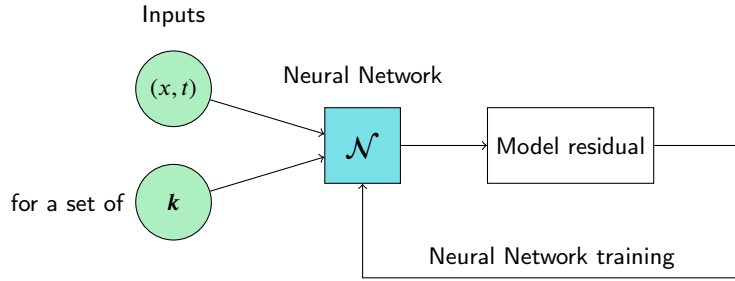
3.2.1. Fully-connected Neural Networks

Let us consider a fully-connected, dense NN composed of d hidden layers $\{f_i\}_{i=1, \dots, d}$ with σ_i the corresponding activation function and f_{d+1} the output layer. Let us denote by $\theta = (W, b) \in \mathbb{R}^{N_\theta}$ the set of its parameters (weights, biases), such that $W = \{W_i\}_{i=1, \dots, d}$ and $b = \{b_i\}_{i=1, \dots, d}$. This NN parameterized by θ is denoted by \mathcal{N}_θ and defined by:

$$\begin{aligned} \mathcal{N}_\theta : I \mapsto O &= (f_{d+1} \circ f_d \circ \dots \circ f_1)(\theta)(I) \\ &= W_{d+1} \cdot \underbrace{\sigma_d(W_d \cdot \dots \cdot \underbrace{\sigma_1(W_1 \cdot I + b_1)}_{\text{output of hidden layer } f_1} + \dots + b_d)}_{\text{output of hidden layer } f_d}. \end{aligned} \quad (7)$$

Note that for this simple architecture, hidden layers simply consist of affine transformations composed with an activation function. Additionally, the output layer is just a linear mapping without any bias nor activation function,

Offline phase:



Online phase:

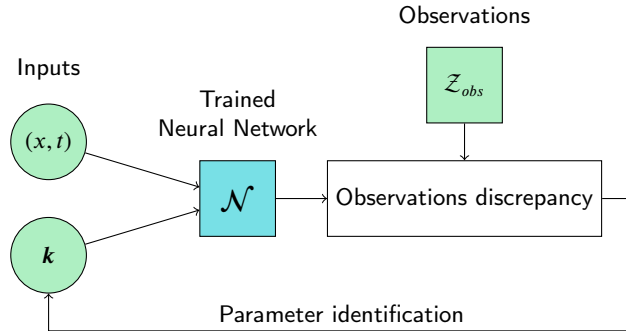


Figure 1: Offline and online phases for the here investigated FP-PINN. The offline phase consists in building a surrogate model by minimizing the model residual J_{res} for a set of parameter functions k . The online phase then consists in using this surrogate to perform the parameter identification, by minimizing the observations discrepancy J_{obs} . Here, k is considered as an input of \mathcal{N} .

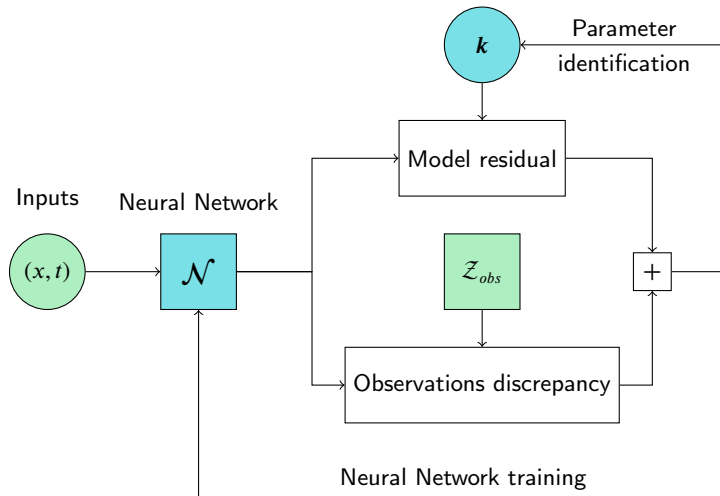


Figure 2: Training of the here investigated SP-PINN. The parameter identification is performed on the fly during the NN training, which consists in minimizing the model residual J_{res} and the observations discrepancy J_{obs} all at once. Here, k is considered as a parameter of \mathcal{N} .

as this last step represents a linear regression in the latent space generated by the penultimate hidden layer (see, *e.g.*, [37]).

3.2.2. Residual computation

The residual of the direct model defined in Eq. (1) reads as:

$$\mathbf{R}(\mathbf{k}; y)(x, t) = \partial_t y(x, t) + A(\mathbf{k}; y)(x, t) - L(\mathbf{k})(x, t), \quad \forall (x, t) \in \Omega \times]0, T]. \quad (8)$$

Then, the domain of interest is sampled through a grid of collocation points $\mathcal{I}_{col} = \{\mathcal{I}_{col}^{(i)}\}_{i=1, \dots, N_{col}}$, and the above residual is minimized on the latter. At this stage, a discretization strategy has to be chosen for the parameter function $\mathbf{k} \in \mathcal{K}$. This discretization step is necessary to approximate the parameter function \mathbf{k} as a function depending on a set of parameters gathered in a finite-dimensional vector, the latter being utilized as an input for the FP-PINN or as a NN parameter for the SP-PINN, see Figs. 3 and 4. Let \mathcal{K}_h be the space of finite-dimensional vectors containing the aforementioned finite set of parameters and $\dim(\mathcal{K}_h) = N_k$. Its elements, denoted by \mathbf{k}_h , consist of N_k components and are referred to as discretized parameter functions. With this in hand, we can define the collocation points \mathcal{I}_{col} depending on the PINN approach as follows:

- For the FP-PINN (see Fig. 3), the discretized parameter function \mathbf{k}_h is considered as an input of the NN, just like the physical variables (x, t) , therefore $\mathcal{I}_{col} = \mathcal{K}_{col} \times \mathcal{X}_{col}$. Here, $\mathcal{X}_{col} = \{(x, t)_{col}^{(i)}\}_{i=1, \dots, N_{col}^{(x,t)}}$ is constituted of values of (x, t) sampled regularly or not over $\Omega \times]0, T]$. Similarly, $\mathcal{K}_{col} = \{\mathbf{k}_{h,col}^{(i)}\}_{i=1, \dots, N_{col}^k}$ is constituted of values of \mathbf{k}_h sampled regularly or not over \mathcal{K}_h ,
- For the SP-PINN (see Fig. 4), the discretized parameter function \mathbf{k}_h is considered as a parameter of the NN, just like θ , therefore $\mathcal{I}_{col} = \mathcal{X}_{col}$.

We denote by $\|\cdot\|_{\mathcal{I}_{col}}$ the Euclidean norm evaluated on the grid \mathcal{I}_{col} . To embed the model knowledge (see Eq. (8)) into the training of the NN, the following residual loss function J_{res} is minimized:

$$J_{res}(\mathbf{k}; y) = \frac{1}{N_{col}} \|\mathbf{R}(\mathbf{k}; y)\|_{\mathcal{I}_{col}}^2 = \begin{cases} \frac{1}{N_{col}^k N_{col}^{(x,t)}} \sum_{\mathbf{k}_h \in \mathcal{K}_{col}} \sum_{(x,t) \in \mathcal{X}_{col}} \mathbf{R}(\mathbf{k}_h; y)(x, t) & \text{for the FP-PINN,} \\ \frac{1}{N_{col}^{(x,t)}} \sum_{(x,t) \in \mathcal{X}_{col}} \mathbf{R}(\mathbf{k}_h; y)(x, t) & \text{for the SP-PINN.} \end{cases} \quad (9)$$

Remark. It is worth emphasizing here that the evaluation of the J_{res} loss function involves computing partial derivatives of the NN output \tilde{y} with respect to the NN inputs (x, t) (through the term $\partial_t y(x, t)$ and the operator $A(\mathbf{k}; y)(x, t)$). Given the structure of a NN, that is composition (of a large number) of simple functions, this can be performed very efficiently by benefiting from automatic differentiation tools (see, e.g., Baydin et al. [38], and Raissi et al. [17] for its application to PINNs). However, it is crucial to ensure that the activation functions $\{\sigma_i\}_{i=1, \dots, d}$ are chosen to be sufficiently smooth to support multiple differentiations, ruling out the standard ReLU activation function. In this study, the hyperbolic tangent activation function is selected for its differentiability.

Remark. In the case of the SWE (see Eq. (6)), to minimize the number of backward passes during an evaluation of the functional J_{res} , the equations are reformulated in non-conservative form. To do so, the fluxes conservation term $(\partial_{x_1} \mathbf{F}(y) + \partial_{x_2} \mathbf{G}(y))$ is expressed in terms of $\bar{\nabla} h$, $\bar{\nabla} q_1$ and $\bar{\nabla} q_2$ as:

$$\partial_{x_1} \mathbf{F}(y) + \partial_{x_2} \mathbf{G}(y) = \begin{pmatrix} 0 & 0 \\ gh - \frac{q_1^2}{h^2} & \frac{q_1 q_2}{h^2} \\ \frac{q_1 q_2}{h^2} & gh - \frac{q_2^2}{h^2} \end{pmatrix} \cdot \bar{\nabla} h + \begin{pmatrix} 1 & 0 \\ 2\frac{q_1}{h} & \frac{q_2}{h} \\ \frac{q_2}{h} & 0 \end{pmatrix} \cdot \bar{\nabla} q_1 + \begin{pmatrix} 0 & 1 \\ 0 & \frac{q_1}{h} \\ \frac{q_1}{h} & 2\frac{q_2}{h} \end{pmatrix} \cdot \bar{\nabla} q_2. \quad (10)$$

This way, only three backward passes are needed for each evaluation of J_{res} , whereas seven backward passes would be required for the same evaluation with the conservative form of Eq. (6). This kind of reformulation could be useful for any conservative systems, since conservative forms generally require to compute the divergence of numerous terms.

3.2.3. Boundary and Initial Conditions

The boundary conditions and the initial conditions are sampled in a standard way, following e.g. Raissi et al. [17], as:

- $\mathcal{Z}_{BC} = \{(z_{BC}^{(i)}, I_{BC}^{(i)})\}_{i=1, \dots, N_{BC}}$, sampled regularly or not over $\mathcal{K} \times \partial\Omega \times]0, T]$ for the FP-PINN, and only over $\partial\Omega \times]0, T]$ for the SP-PINN,
- $\mathcal{Z}_{IC} = \{(z_{IC}^{(i)}, I_{IC}^{(i)})\}_{i=1, \dots, N_{IC}}$, sampled regularly or not over $\mathcal{K} \times \Omega$ for the FP-PINN, and only over Ω for the SP-PINN.

The following loss functions can then be minimized to ensure that the solution satisfies the boundary and initial conditions:

$$\begin{aligned} J_{BC}(y) &= \frac{1}{N_{BC}} \|Z^{BC}(y) - z_{BC}\|_2^2, \\ J_{IC}(y) &= \frac{1}{N_{IC}} \|Z^{IC}(y) - z_{IC}\|_2^2, \end{aligned} \quad (11)$$

with Z^{BC} and Z^{IC} the operators mapping from the states space \mathcal{Y} to the sampled boundary conditions space and to the sampled initial conditions space, respectively,.

3.2.4. Total loss function

To encapsulate both model constraints and observations discrepancy, the following total loss function can eventually be defined, where we recycle J_{obs} from Section 2.2:

$$\begin{aligned} J(\mathbf{k}; y) &= \lambda_{res} J_{res}(\mathbf{k}; y) + \lambda_{BC} J_{BC}(y) + \lambda_{IC} J_{IC}(y) + \lambda_{obs} J_{obs}(y) \\ &= \underbrace{\frac{\lambda_{res}}{N_{col}} \|\partial_t y + A(\mathbf{k}; y) - L(\mathbf{k})\|_{L_{col}}^2 + \frac{\lambda_{BC}}{N_{BC}} \|Z^{BC}(y) - z_{BC}\|_2^2 + \frac{\lambda_{IC}}{N_{IC}} \|Z^{IC}(y) - z_{IC}\|_2^2}_{\text{physics-informed}} \\ &\quad + \underbrace{\frac{\lambda_{obs}}{N_{obs}} \|Z(y) - z_{obs}\|_{R^{-1}}^2}_{\text{data-driven}} \end{aligned} \quad (12)$$

with the weight factors set $\lambda = (\lambda_{res}, \lambda_{BC}, \lambda_{IC}, \lambda_{obs}) \in (\mathbb{R}_+)^4$ given. In practice, we introduce the loss function J_{phy} as:

$$\lambda_{phy} J_{phy}(\mathbf{k}; y) = \lambda_{res} J_{res}(\mathbf{k}; y) + \lambda_{BC} J_{BC}(y) + \lambda_{IC} J_{IC}(y), \quad (13)$$

with $\lambda_{phy} \in \mathbb{R}_+$. The total loss function then becomes:

$$J(\mathbf{k}; y) = \lambda_{phy} J_{phy}(\mathbf{k}; y) + \lambda_{obs} J_{obs}(y). \quad (14)$$

The value of λ has a lot of influence on the training and is difficult to fix. In this work, its value is set by trial and error such that the loss functions defined earlier, after being normalized by their first iterate, have comparable magnitude during training.

3.3. Detailed methods

All the necessary ingredients have been detailed above, let us now detail the two PINN approaches: the FP-PINN and the SP-PINN.

3.3.1. Fully-Parameterized PINN

For the FP-PINN (Fig. 3), we recall that the primary objective is to train the NN in order to build a surrogate model associated to Eq. (1). Thus, \mathcal{N}_θ is expressed as follows:

$$\mathcal{N}_\theta : (\mathbf{k}_h; (x, t)) \mapsto \tilde{y}_\theta(\mathbf{k}_h; (x, t)), \quad (\mathbf{k}_h; (x, t)) \in \mathcal{K}_{col} \times \mathcal{X}_{col}. \quad (15)$$

The training phase consisting in the approximation of operator $\mathcal{M}_{FP} : (\mathbf{k}; (x, t)) \mapsto y^k(x, t)$ for a given set of \mathbf{k} is called the *offline* phase (Fig. 3) because it can be performed once and reused for multiple parameter identifications.

This phase involves solving the following optimization problem:

$$\text{offline: } \theta^* = \underset{\theta \in \mathbb{R}^{N_\theta}}{\operatorname{argmin}} \left(J_{phy}(\mathbf{k}_h; \tilde{y}_\theta(\mathbf{k}_h; \cdot)) \right), \quad \mathbf{k}_h \in \mathcal{K}_{col}. \quad (16)$$

Once the offline training phase is completed, the J_{obs} loss function can be minimized using \mathcal{N}_{θ^*} as a simple \mathbf{k}_h -differentiable surrogate model for operator \mathcal{M}_{FP} , as long as \mathbf{k}_h remains within the range values of \mathcal{K}_{col} to ensure that the surrogate model is reliable. By doing this, the discretized parameter function \mathbf{k}_h^{true} that best explains the given observations \mathcal{Z}_{obs} can be approximated by \mathbf{k}_h^* at a low computational cost:

$$\mathcal{Z}_{obs} \mapsto \mathbf{k}_h^* \approx \mathbf{k}_h^{true}. \quad (17)$$

This phase is called the *online* phase (Fig. 3) and involves solving the following optimization problem, with $\mathbf{k}_h^{min} = \inf(\mathcal{K}_{col})$ and $\mathbf{k}_h^{max} = \sup(\mathcal{K}_{col})$:

$$\text{online: } \mathbf{k}_h^* = \underset{\mathbf{k}_h \in [\mathbf{k}_h^{min}; \mathbf{k}_h^{max}]}{\operatorname{argmin}} \left(J_{obs}(\tilde{y}_{\theta^*}(\mathbf{k}_h; \cdot)) \right), \quad \text{for a given dataset } \mathcal{Z}_{obs}. \quad (18)$$

It should be noted that the output $\tilde{y}_\theta(\mathbf{k}_h; \cdot)$ has a double dependency on \mathbf{k}_h , a direct one through its inputs and an indirect one through the minimization of J_{phy} . Hence, the natural name Fully-Parameterized PINN proposed in this work. Additionally, let us stress that the offline/online strategy is possible here because \mathbf{k}_h is an input of the NN. Thus, the loss function J_{obs} depends directly on \mathbf{k}_h through \tilde{y}_θ , which makes the use of the derivative $\frac{\partial J_{obs}}{\partial \mathbf{k}_h}$ possible to minimize J_{obs} .

From a practical point of view, it is observed that minimizing J_{phy} alone, as required in the offline training phase, can be a difficult task. First, this loss function appears to be highly non-convex, and therefore very sensitive to the initialization of \mathcal{N}_θ . Then and more importantly, the space $\mathcal{K}_{col} \times \mathcal{X}_{col}$ can become huge as the dimension of \mathbf{k}_h (denoted N_k , see Section 3.2.2) increases, making the FP-PINN very prone to the curse of dimensionality. Indeed, the offline training phase of the FP-PINN entails sampling a hypercube of dimension $((d+1) + N_k)$, with d the spatial dimension of the problem, which quickly becomes intractable for large values of N_k . However, if these two issues are addressed, the result is a differentiable surrogate model that can be efficiently utilized for as many online phases as needed.

Remark. *The here investigated FP-PINN is connected to the so-called DeepONet (Deep Operator Network) (see Lu et al. [39] for the origin and [28] and references therein for applications in different fields). The DeepONet architecture is a NN architecture developed to approximate continuous non-linear operators. The concept has also been generalized to neural operators in [40]. This architecture is used by Wang et al. [30] to elaborate the Physics-Informed DeepONet (PI-DeepONet), see, e.g., [28] and references therein for applications. The goal of the PI-DeepONet is substantially the same as the offline phase of the FP-PINN proposed here (see Eq. (16)).*

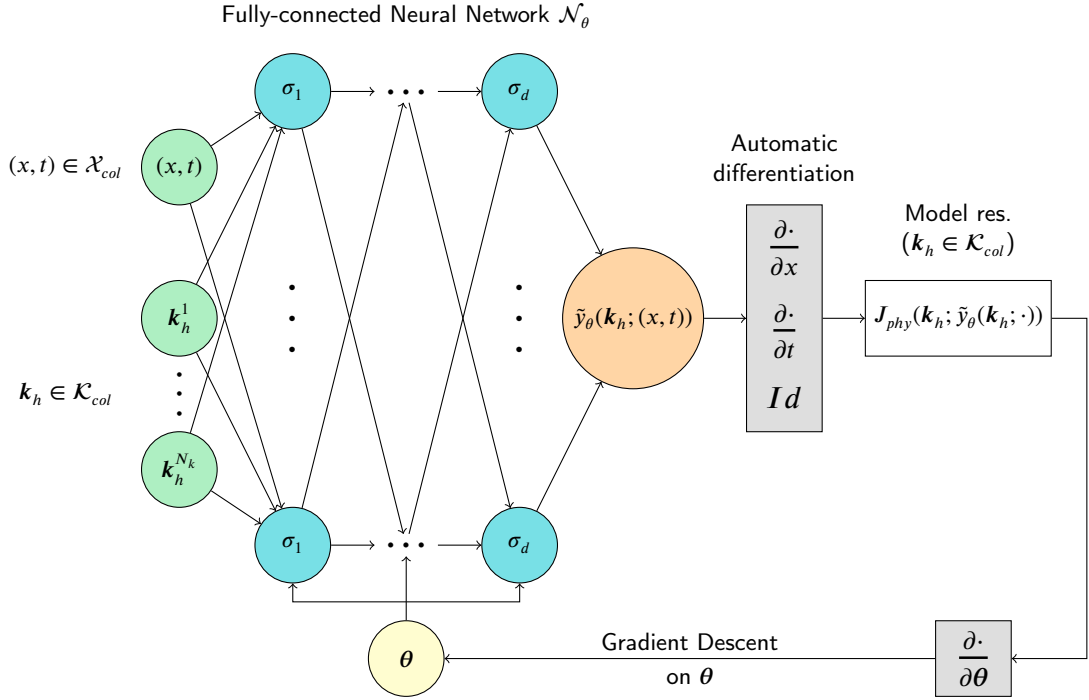
The difference between PI-DeepONet and the offline phase of the FP-PINN lies in the architecture of the NN. Where the FP-PINN is a classical fully-connected NN (also known as a multi-layer perceptron), the DeepONet architecture, elaborated from the universal approximation theorem for operators (see Chen and Chen [41]), expresses its output as the dot product of the outputs of two NNs: the branch and the trunk.

Given these differences and similarities, the offline phase of the FP-PINN introduced in this work appears as a brute force version of the PI-DeepONet. However, as both architectures consider the discretized parameter function as an input, they share the same advantages and disadvantages in terms of capacity to handle high-dimensional parameter in a DA context.

3.3.2. Semi-Parameterized PINN

Conversely, in the case of the SP-PINN (Fig. 4), let us remind that the objective is to learn the physics and the data at once during the training, thus approximating the operator $\mathcal{M}_{SP} : (x, t) \mapsto y_k(x, t)$ for a given value of \mathbf{k} which is

Offline phase:



Online phase:

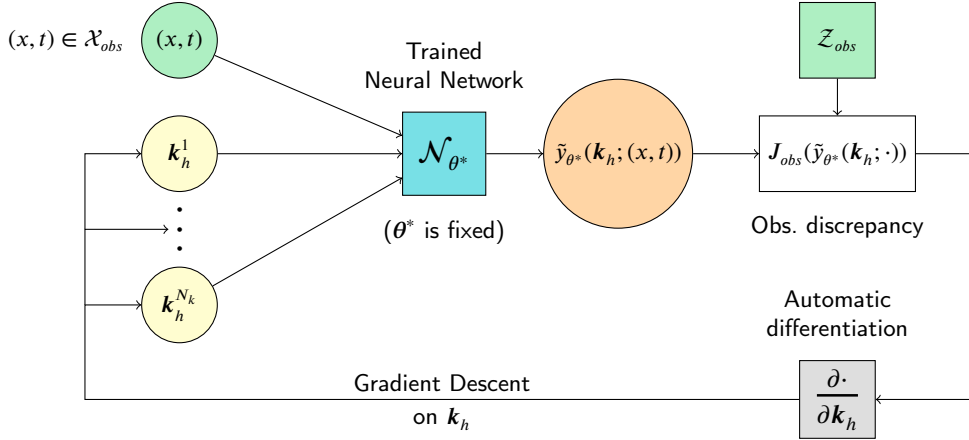


Figure 3: Offline and online phases for the here investigated FP-PINN. The offline phase consists in building a surrogate model for a set of discretized parameter function $\mathbf{k}_h \in \mathcal{K}_{col}$. It can be seen as a brute force counterpart of the PI-DeepONet (see Wang et al. [30], and, e.g., [28] and references therein for applications). The online phase then consists in using this surrogate to find the \mathbf{k}_h explaining at best the observations \mathcal{Z}_{obs} .

learned on the fly. In order to do so, the NN is now also parameterized by \mathbf{k}_h and therefore denoted by $\mathcal{N}_{\mathbf{k}_h, \theta}$:

$$\mathcal{N}_{\mathbf{k}_h, \theta} : (x, t) \mapsto \tilde{y}_{\mathbf{k}_h, \theta}(x, t), \quad \text{for a given value of } \mathbf{k}_h \in \mathcal{K}_{col}. \quad (19)$$

The training of the NN consists in solving the following optimization problem:

$$(\mathbf{k}_h^*, \boldsymbol{\theta}^*) = \underset{(\mathbf{k}_h, \boldsymbol{\theta}) \in \mathbb{R}^{N_k + N_\theta}}{\operatorname{argmin}} \left(\lambda_{phy} J_{phy}(\mathbf{k}_h; \tilde{\mathbf{y}}_{k_h, \boldsymbol{\theta}}) + \lambda_{obs} J_{obs}(\tilde{\mathbf{y}}_{k_h, \boldsymbol{\theta}}) \right), \quad \text{for a given dataset } \mathcal{Z}_{obs}. \quad (20)$$

Here, the discretized parameter function \mathbf{k}_h is directly considered as a parameter during the optimization, just like $\boldsymbol{\theta}$. Therefore, its value changes during the training such that after the optimization, \mathbf{k}_h^* is expected to be a good approximation of \mathbf{k}_h^{true} .

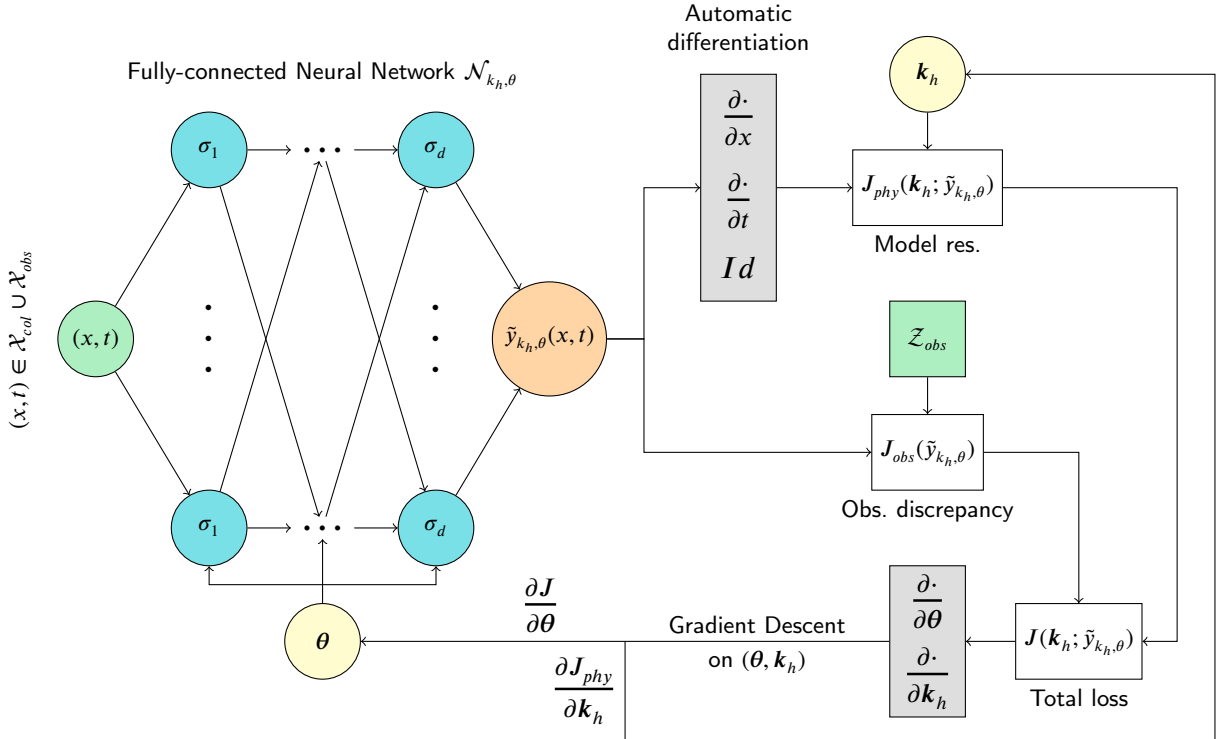


Figure 4: Training of the here investigated SP-PINN, which is equivalent to the original PINN for inverse problems introduced in Raissi et al. [17] and now widely used in different contexts (see, e.g., [28] and references therein). Here, the model and the observations are learned jointly and the parameter identification is conducted at once with the NN training, on the fly.

Contrary to the FP-PINN, the output $\tilde{\mathbf{y}}_{k_h, \boldsymbol{\theta}}$ has only one indirect dependency on \mathbf{k}_h , through the minimization of J_{phy} . Hence, the name Semi-Parameterized PINN proposed in this work. This method is equivalent to the original PINN method for inverse problems introduced by Raissi et al. [17], and widely applied since (see, e.g., [28] and references therein).

In practice, since the training of the SP-PINN involves learning the physics for only one discretized parameter function instead of an entire set as in the FP-PINN case, this method is much less prone to the curse of dimensionality. Indeed, increasing the dimension of \mathbf{k}_h increases the dimension of the parameter space for the SP-PINN training, which then becomes $(N_k + N_\theta)$, but it does not change the dimension of the hypercube for the training, which remains $(d + 1)$ (compared to $((d + 1 + N_k)$ for the FP-PINN). Increasing the number of parameters N_k is likely not problematic from a computational cost perspective, because N_k will a priori be low compared to N_θ that is usually very large (several thousands at least) and because NN technology is well-suited for handling large numbers of parameters. Another advantage of the SP-PINN over the FP-PINN is that the presence of observations during training can actually aid in minimizing J_{phy} (see, e.g., Gopakumar et al. [42]). However, it should be noted that we do not obtain a surrogate model

with the SP-PINN approach; the resulting NN after training reflects the physics for \mathbf{k}_h^* only.

3.4. Algorithmic cares

This sections emphasizes the algorithmic cares used in this work to make FP-PINN and SP-PINN training possible. Without these best practices, it has been numerically observed that training can often diverge or result in very poor parameter identification.

Training a NN essentially involves the minimization of a non-convex loss function in a high-dimensional parameter space using a (stochastic or deterministic) gradient descent method. Therefore, the initialization point chosen in the parameter space, *i.e.* the first parameter value θ_0 , is important in determining the value of θ^* . The so-called Xavier initialization [43] is known to produce good results for training NNs using the hyperbolic tangent activation function. We therefore use this initialization method in this work.

Even with the latter, it is numerically observed that the minimization of J_{res} can get stuck in bad local minima, resulting in poor learning of the model. To mitigate this issue, a pre-training phase is implemented, both for the FP-PINN and the SP-PINN, with different forms depending on the choice of method. These procedures are detailed hereafter. This ensures that the optimization of J_{res} starts at a θ value, called θ^{pre} , that corresponds to a NN output closer to the solution than after the initialization, making the minimization of J_{res} less prone to bad local minima. Without this pre-training phase, the non-convexity of J_{res} with respect to θ can make the optimization very sensitive to θ_0 , necessitating multiple training restarts before achieving proper minimization.

3.4.1. Fully-Parameterized PINN

For the pre-training phase of the FP-PINN, the idea is to set up a "good" first value of the state y , denoted by y^{1st} , and to perform a supervised training phase to make the output of the NN fit y^{1st} for all collocation points in $\mathcal{K}_{col} \times \mathcal{X}_{col}$. In this work, y^{1st} is set to equal BC values. During the offline and online phases, no special method is used other than classical training with a gradient descent method (*e.g.* L-BFGS or Adam). The entire training procedure for the FP-PINN is outlined in Algorithm 1:

Initialization

θ_0 obtained by Glorot and Bengio [43] method.

$\mathbf{k}_{h,0}$ chosen by the user.

Pre-training

Data: $\mathcal{X}_{col}; \mathcal{K}_{col}; \theta_0; y^{1st}$,

$$\theta^{pre} = \underset{\theta \in \mathbb{R}^{N_\theta}}{\operatorname{argmin}} \left(\|\tilde{y}_\theta - y^{1st}\|_{\mathcal{K}_{col} \times \mathcal{X}_{col}}^2 \right).$$

Offline phase

Data: $\mathcal{X}_{col}; \mathcal{K}_{col}; \mathcal{Z}_{BC}; \mathcal{Z}_{IC}; \theta_0 = \theta^{pre}; \lambda$,

$$\theta^* = \underset{\theta \in \mathbb{R}^{N_\theta}}{\operatorname{argmin}} \left(J_{phy}(\mathbf{k}_h; \tilde{y}_\theta(\mathbf{k}_h; \cdot)) \right), \quad \mathbf{k}_h \in \mathcal{K}_{col}.$$

Online phase

Data: $\tilde{y}_{\theta^*}; \mathcal{Z}_{obs}; \mathbf{k}_{h,0}; \mathbf{k}_h^{min}; \mathbf{k}_h^{max}$,

$$\mathbf{k}_h^* = \underset{\mathbf{k}_h \in [\mathbf{k}_h^{min}; \mathbf{k}_h^{max}]}{\operatorname{argmin}} \left(J_{obs}(\tilde{y}_{\theta^*}(\mathbf{k}_h; \cdot)) \right), \quad \text{for a given dataset } \mathcal{Z}_{obs}.$$

Algorithm 1: Initialization and training of the FP-PINN. The pre-training allows for a better minimization of J_{res} as the offline phase will start with an output $\tilde{y}_{\theta^{pre}}(\mathbf{k}_h; \cdot)$ closer to the solution y^k . The online phase then uses the surrogate model calibrated in the offline phase to perform parameter identification at a low computational cost.

3.4.2. Semi-Parameterized PINN

With the SP-PINN, the observations dataset \mathcal{Z}_{obs} can actually be used for the pre-training. Indeed, as the observations are expected to be close to the model solution, a simple supervised learning phase on J_{obs} allows to start the minimization of J_{res} with values of θ corresponding to an output close to the model solution, which shows better optimization of the residual. In simple cases, the functionals J_{BC} and J_{IC} can also be used during the pre-training phase, similarly to J_{obs} .

For the test cases studied in this article, and certainly in most cases, it is numerically observed that the training phase can be disrupted because the norm of the gradient of J with respect to k_h is negligible compared to the norm of the gradient of J compared to θ : $\|\frac{\partial J_{res}}{\partial k_h}\| \ll \|\frac{\partial J}{\partial \theta}\|$. This leads to almost no updates on k_h during gradient descent. To counter this effect, an alternating minimization strategy (fixed point algorithm) is used to alternate between minimizing J with respect to θ and with respect to k_h . During each alternating minimization step, a classical gradient descent method (e.g., L-BFGS or Adam) is employed. The entire procedure for the SP-PINN training is outlined in Algorithm 2:

Initialization

θ_0 obtained by Glorot and Bengio [43] method.
 $k_{h,0}$ chosen by the user.

Pre-training

Data: $\mathcal{Z}_{obs}; \mathcal{Z}_{BC}; \mathcal{Z}_{IC}; \theta_0; \lambda$,
 $\theta^{pre} = \underset{\theta \in \mathbb{R}^{N_\theta}}{\operatorname{argmin}} \left(\lambda_{obs} J_{obs}(\tilde{y}_{k_{h,0}, \theta}) + \lambda_{BC} J_{BC}(\tilde{y}_{k_{h,0}, \theta}) + \lambda_{IC} J_{IC}(\tilde{y}_{k_{h,0}, \theta}) \right)$, for a given dataset \mathcal{Z}_{obs} .

Alternating minimization

Data: $\mathcal{X}_{col}; \mathcal{Z}_{obs}; \mathcal{Z}_{BC}; \mathcal{Z}_{IC}; \theta^{(0)} = \theta^{pre}; k_h^{(0)} = k_{h,0}; N_{alter}; \lambda$,
for $i = 1, \dots, N_{alter}$ **do**
 $k_h^{(i)} = \underset{k_h \in \mathbb{R}^{N_k}}{\operatorname{argmin}} \left(J_{res}(k_h; \tilde{y}_{k_h, \theta^{(i-1)}}) \right)$,
 $\theta^{(i)} = \underset{\theta \in \mathbb{R}^{N_\theta}}{\operatorname{argmin}} \left(J(k_h^{(i)}; \tilde{y}_{k_h^{(i)}, \theta}) \right)$, for a given dataset \mathcal{Z}_{obs} .
end

Algorithm 2: Initialization and training of the SP-PINN. The pre-training allows for a better minimization of the J_{res} loss function as the alternating minimization will start with an output $\tilde{y}_{\theta^{pre}}(k_h; \cdot)$ closer to the solution y^k . To counteract the difference of influence on J between k_h and θ , an alternating minimization strategy is used to minimize J with respect to each parameter independently.

3.4.3. Fourier features

For the SP-PINN on the SWE test case, a learnable Fourier features embedding (see, e.g., [20, 44, 45]) is used to help the SP-PINN approximate the physical solution at all scales. More specifically, before the input $v = (x_1, x_2, t)^T$ is fed into the SP-PINN, an embedding of the following form:

$$\gamma(v) = (v, \sin(2\pi \xi_1 \odot v), \cos(2\pi \xi_1 \odot v), \dots, \sin(2\pi \xi_m \odot v), \cos(2\pi \xi_m \odot v))^T$$

is applied, where \odot denotes the element-wise product and $\xi_i = (\xi_i^{x_1}, \xi_i^{x_2}, \xi_i^t)^T \in \mathbb{R}^3$ is the frequency vector for the i -th Fourier feature. In this work, initial values are given for the frequencies $\xi = (\xi_1, \dots, \xi_m)$, but once the training begins they are considered as parameters of the NN. As here $\theta = (W, b, \xi)$, the frequencies are then optimized during the NN training along with weights and biases.

4. Numerical experiments: inverse problems in hydraulics

In the section below, two test cases based on the flow models introduced in Section 2.3 are presented and discussed. In Section 4.1, the two PINN approaches detailed in Section 3.3 are compared on the simple Backwater Equation model introduced in Section 2.3.1. Despite the model's simplicity, its complexity is sufficient to analyze the capacities of the different approaches to handle high-dimensional parameters. Based on these results, the SP-PINN method is then chosen in Section 4.2 to illustrate high-dimensional parameter identification on a model based on the 2D SWE.

In all the experiments below, the NNs are optimized using a L-BFGS optimizer (with Wolfe conditions for the linear search). The activation function is the hyperbolic tangent $\tanh()$ and θ is initialized using the so-called Xavier initialization [43]. The parameter \mathbf{K}_{sh} is initialized with a constant value of $40 \text{ m}^{1/3}/\text{s}$ throughout the domain. All observations are considered noise-free and the error covariance matrix for \mathbf{J}_{obs} is set to $\mathbf{R} = \mathbf{Id}$. The loss functions are normalized by their first iterate $\mathbf{J}_{\square}^{(0)}$. It should be noted that no training or test sets are used for collocation points, as the goal is to satisfy the model as much as possible. As the latter is considered perfect, the concept of overfitting does not apply here. Similarly, as the observations are considered noise-free, no training or test sets are used for observation points either. Numerical tests were conducted with test sets for both collocation points and observation points, and in each case, the inference was worse than without them.

Unless specified otherwise, all PINNs calculations are performed on a single NVIDIA RTX A3000 Laptop GPU with 6 GB memory and 4096 CUDA cores. The algorithms presented below are implemented using the Pytorch library [46] and are available on the following GitHub repository: DassHydro repository.

4.1. Backwater Equation

The aim of this subsection is to compare the two PINN approaches presented in Section 3.3 in terms of identification performance and capacity to handle high-dimensional parameter (here the Strickler friction coefficient \mathbf{K}_s). First, the results of both approaches are illustrated for the same test case to highlight the methodological differences.

4.1.1. Setup

The illustration test case, shown in Fig. 5, consists of a 1D spatial domain Ω representing a 1000 m -long reach. The bathymetry is a scaled so-called Mac Donald's type bathymetry (see, e.g., Delestre et al. [47]) and the discretized parameter function \mathbf{K}_{sh} is defined on 3 constant-value patches ($N_k = 3$ with a piecewise constant interpolation in Ω). The observations consist of 20 points ($N_{obs} = 20$) with a known water height, sampled randomly with a uniform distribution in the domain Ω . The reference solution from which the observations are drawn is obtained with a RK4 finite-difference scheme, obtained with friction values $\mathbf{K}_{sh}^{true} = (40, 30, 50) \text{ m}^{1/3}/\text{s}$. Since the flow is in subcritical regime, the BC is fixed downstream, at $h_{out} = 0.8 \text{ m}$ and the constant flow rate is set at $q = 1 \text{ m}^3/\text{s}/\text{m}$.

The results are presented below for the FP-PINN and the SP-PINN. Then, a comparison between the two approaches on inference accuracy and computation time when the dimension of \mathbf{K}_{sh} increases is proposed.

4.1.2. Results for the Fully-Parameterized PINN

For the FP-PINN, a fully-connected NN of the shape [4, 60, 60, 60, 1] ($N_{\theta} = 7680$) is used. This architecture, relatively wide and shallow, and other ones in the article have been determined through successive numerical experiments with the aim to maximize the physical consistency of the PINN compared to the reference flow while maintaining a reasonable NN size. The collocation points \mathcal{X}_{col} are sampled on a regular 1D grid with 100 vertices ($N_{col}^x = 100$) and those in the parameter domain \mathcal{K}_{col} are sampled using a regular 3D grid with 10 vertices in each dimension, so $N_{col}^k = 1000$ and $N_{col} = 10^5$. The value of λ is set to $\lambda_{res} = 10^3$ and to 1 for all other components. Then, a pre-training consisting of 100 L-BFGS iterations and an offline phase of 2000 L-BFGS iterations are performed. The representative value of the state y^{1st} used in the pre-training corresponds here to $h_{out} = 0.8 \text{ m}$. Once the offline phase is complete, parameter identification is carried out during the online phase, with a stopping criterion on the stability

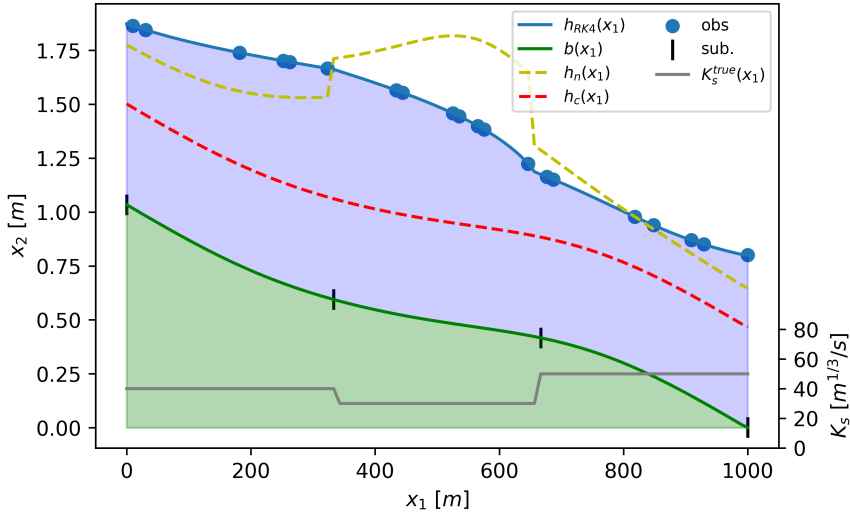


Figure 5: Backwater Equation test case to compare SP-PINN and FP-PINN. The bathymetry is shown in green, the reference flow line in blue and the reference spatially-distributed friction function parameter \mathbf{K}_s in grey. Observations points are represented by the blue dots and the subdomains where \mathbf{K}_s is discretized correspond to the vertical black dashes.

of the J_{obs} loss function.

The results of the offline training phase are presented in Fig. 6. The graphs show the minimization of the loss functions during pre-training and offline phase, as well as a projection of the trained surrogate model onto a subspace. The loss function represented in the left-hand graph, whose dominant term is the residual J_{res} , reaches a plateau after 1500 iterations, corresponding to a satisfactory training. The calibrated NN produces pertinent physical modeling compared to the reference solution, as shown in the graph on the right by the response surfaces represented in terms of longitudinal flow depth profiles $\tilde{h}_\theta(\mathbf{K}_{sh}, x_1)$ and $h_{RK4}(\mathbf{K}_{sh}, x_1)$ for different friction parameter patterns. These very similar response surfaces for a fairly wide range of friction parameters show that this surrogate is physically relevant.

Next, the calibrated FP-PINN is used to perform parameter identification from flow observations in the online phase, during which a calibrated flow model is also obtained. The results presented in the left-hand graph of Fig. 7 show that the calibrated flow model (black dashed line) fits well the flow observations (blue dots) from the reference flow line (blue line). Note that the simulated flow is fluvial ($Fr < 1$), as shown by the flow line above critical depth (red line, we refer the reader to [12, 13, 31] for the basic concepts in hydraulics). The normal depth (yellow dashed line), which corresponds to a local equilibrium between gravity and friction effects towards which the model would tend on long uniform reaches, shows the sharp friction variations. The spatial pattern and values of friction are well retrieved by resolution of the inverse problem using the FP-PINN as surrogate model. Note that the graph on the right shows the rapid convergence of the friction parameter identification during the online phase. Note also that a longer offline phase leads to a more accurate surrogate hence more accurate inversions in the tested configurations.

This highlighted the ability of the FP-PINN to identify a spatially-distributed friction parameter, by first building a surrogate of the parameter-to-state operator defined in Eq. (2), which is then used to perform the identification.

4.1.3. Results for the Semi-Parameterized PINN

For the SP-PINN, a fully-connected NN of the form [1, 60, 60, 60, 1] ($N_\theta = 7500$) is used. Collocation points \mathcal{X}_{col} are sampled on a regular 1D grid with 100 vertices ($N_{col} = 100$). The value of λ is set to $\lambda_{res} = 10^3$ and to 1 for all other components. Then, a pre-training consisting of 100 L-BFGS iterations and 4 alternating minimization steps of 50 L-BFGS iterations each (40 iterations on θ and 10 iterations on the parameter of interest \mathbf{K}_{sh}), representing 200

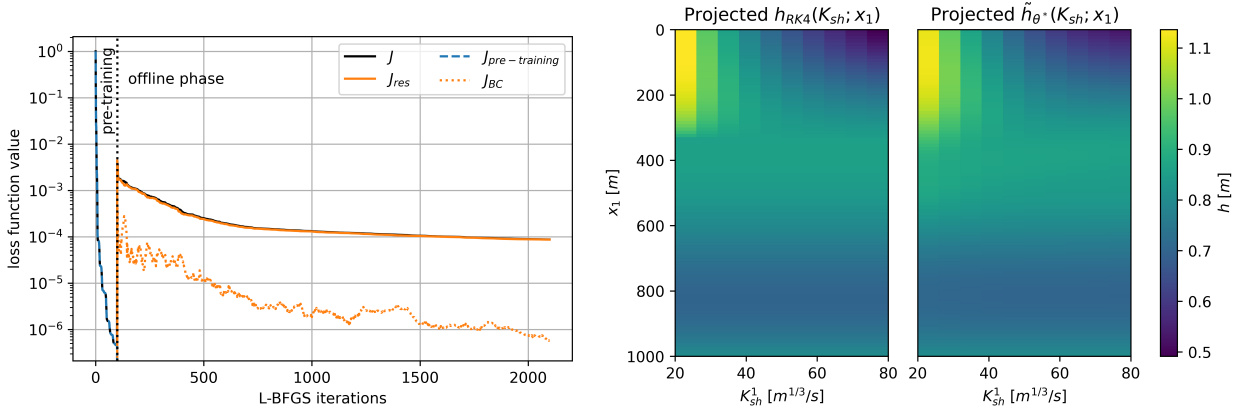


Figure 6: Results of the FP-PINN offline phase training (Backwater Equation). Left: loss functions minimization during pre-training and offline phase. Right: comparison between reference solution and trained model prediction projected on a subspace showing the dependency of the water height h on x_1 and on the first component of the parameter of interest K_{sh}^1 . For this projection, all other components of K_{sh} are set to $40 \text{ m}^{1/3}/s$.

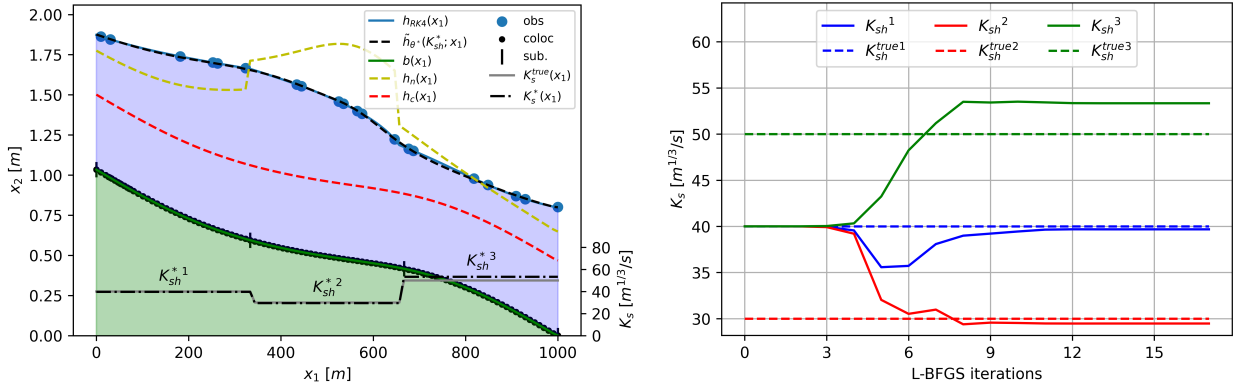


Figure 7: Results of the FP-PINN online phase training (Backwater Equation). Left: calibrated model after online phase, showing the inferred value of the parameter K_{sh}^* and the corresponding model solution $\tilde{h}_{\theta^*}(K_{sh}^*; x_1)$. Right: inference of K_{sh} components values during online phase, compared with K_{sh}^{true} components values.

L-BFGS iterations in total, are performed.

The training results are presented in Fig. 8. The graphs show the minimization of the loss functions during pre-training and alternating minimization phases, as well as the minimization of the relative RMSE. The loss function represented in the left-hand plot, whose dominant term is the observations discrepancy J_{obs} , reaches a plateau after 150 iterations, corresponding to a satisfactory training. The same plateau can be observed on the relative RMSE both on the model solution h and on the parameter K_{sh} , indicating that the training has led to satisfactory results.

The results represented in the left-hand graph of Fig. 9 show that the calibrated flow model (black dashed line) fits well to the flow observations (blue points) from reference flow line (blue line). Since the test case discussed here is identical to that of the previously presented FP-PINN, hydraulic quantities such as critical depth and normal depth exhibit the same behavior as described above. The spatial pattern and values of friction are well retrieved by resolution of the inverse problem during the SP-PINN training. Note that the graph on the right shows the rapid convergence of the friction parameter K_{sh} identification, where a plateau is reached almost right after the start of the alternating minimization phase.

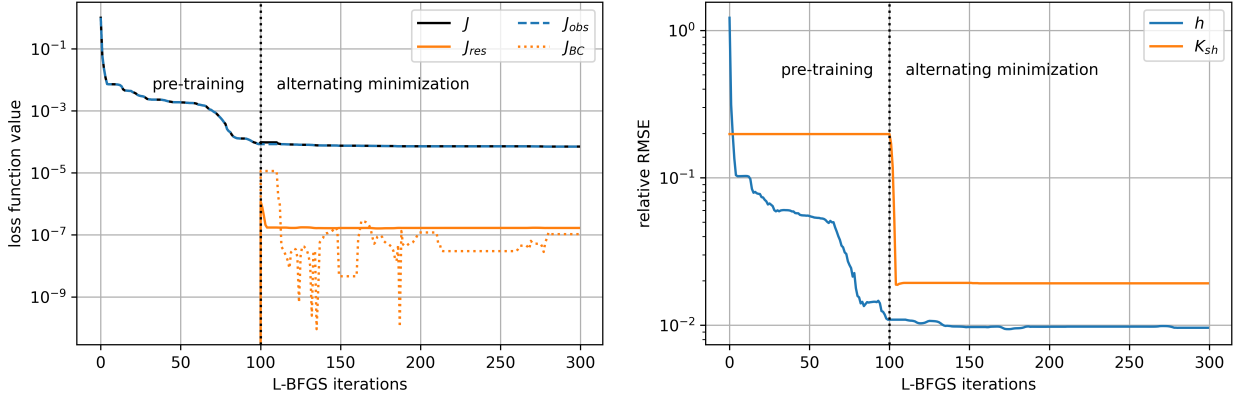


Figure 8: Results of the SP-PINN training (Backwater Equation). Left: loss functions minimization during pre-training and alternating minimization. Right: Relative RMSE minimization on h and on \mathbf{K}_{sh} during pre-training and alternating minimization phases.

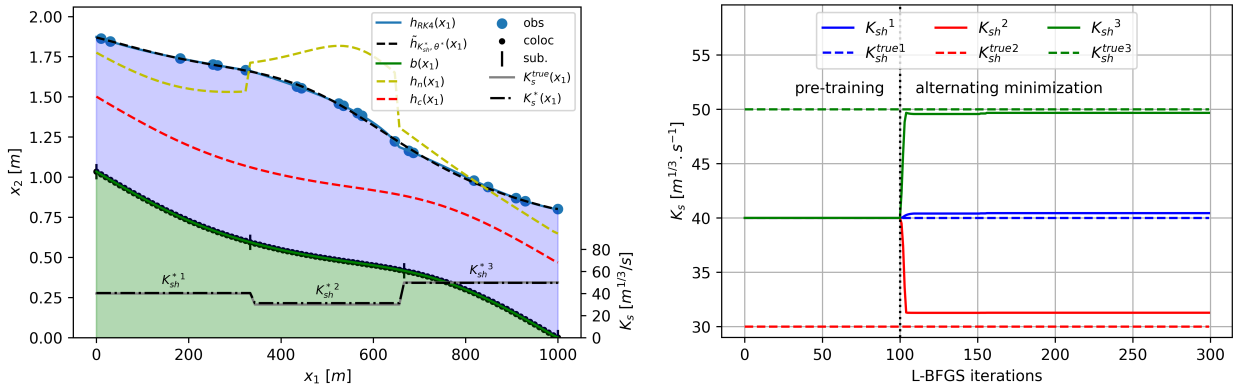


Figure 9: Results of the parameter identification with the SP-PINN (Backwater Equation). Left: calibrated model after training, showing the inferred value of the parameter \mathbf{K}_{sh}^* and the corresponding model solution $\tilde{h}_{\mathbf{K}_{sh}^*, \theta^*}(x_1)$. Right: inference of \mathbf{K}_{sh} components values during pre-training and alternating minimization, compared with \mathbf{K}_{sh}^{true} components values.

This showcased the capacity of the SP-PINN to perform an identification of a spatially-distributed friction parameter, by simultaneously learning the physics (represented by the model) and fitting to the observations during the NN training.

4.1.4. Methods comparison

The capacities of the two PINN approaches to handle a high-dimensional parameter of interest \mathbf{K}_{sh} is studied below. The test case is the same as the illustration test case presented above, but with increasing parameter dimension N_k . For each dimension, the relative RMSE on the inferred value of \mathbf{K}_{sh}^* and the computation time are monitored (for the FP-PINN, the offline and online computation times are specified). The friction is always distributed regularly in the domain, on N_k constant patches, similar to the illustration test case but with varying number of patches. Results of this comparison are presented in Table 1. The components values of \mathbf{K}_{sh}^{true} used are:

- $N_k = 1$: $\mathbf{K}_{sh}^{true} = 40 \text{ m}^{1/3}/\text{s}$
- $N_k = 2$: $\mathbf{K}_{sh}^{true} = (40, 30) \text{ m}^{1/3}/\text{s}$
- $N_k = 3$: $\mathbf{K}_{sh}^{true} = (40, 30, 50) \text{ m}^{1/3}/\text{s}$
- $N_k = 4$: $\mathbf{K}_{sh}^{true} = (40, 30, 35, 50) \text{ m}^{1/3}/\text{s}$

Table 1

Comparison of the SP-PINN and the FP-PINN performances on Backwater Equation with test cases of increasing parameter dimension. For the FP-PINN, the computation time is detailed under the form (offline computation time + online computation time) s.

		SP-PINN (Semi-Parameterized)	FP-PINN (Fully-Parameterized)
$N_k = 1$	Relative RMSE on \mathbf{K}_{sh}^*	1.44×10^{-3}	4.69×10^{-3}
	Computation time	9.21 s	(40.68 + 0.011) s
$N_k = 2$	Relative RMSE on \mathbf{K}_{sh}^*	3.61×10^{-3}	8.94×10^{-3}
	Computation time	8.50 s	(32.84 + 0.031) s
$N_k = 3$	Relative RMSE on \mathbf{K}_{sh}^*	1.94×10^{-2}	4.80×10^{-2}
	Computation time	8.42 s	(88.01 + 0.027) s
$N_k = 4$	Relative RMSE on \mathbf{K}_{sh}^*	3.82×10^{-2}	1.26×10^{-1}
	Computation time	4.55 s	(646.30 + 0.034) s

GPU memory exceeded, mini-batch descent (Adam)

This showcased the capacity of the SP-PINN to perform parameter identification without being affected by the dimension of the inverse problem (namely the value of N_k). Indeed, for the SP-PINN, the discretized parameter is used as a parameter of the NN, therefore increasing its size increases the dimension of the parameter space, but the NN already has on the order of $\sim 10^4$ parameters ($\dim(\theta) = \mathcal{O}(10^4)$). Therefore, adding a few parameters by increasing the dimension of \mathbf{K}_{sh} does not make much of a difference to the training, as one still has $N_k \ll N_\theta$. Note that the slight variations observed in computation time arise from the L-BFGS optimizer conducting additional function evaluations during gradient descent. Here, computation times for the SP-PINN remain consistently within the same order of magnitude, irrespective of the parameter dimension N_k .

On the other hand, the FP-PINN takes the parameter \mathbf{K}_{sh} as an input, therefore increasing its dimension drastically increases the complexity of the inverse problem because of the curse of dimensionality. It should be noted that the FP-PINN can handle low-dimensional parameter identification with reasonable computation time thanks to the parallelization capabilities of NNs. This is why the computation times for $N_k = 1, 2, 3$ remain within the same order of magnitude. Unfortunately, as soon as the cost of a gradient descent step (dominated by the cost of the numerous forward calls) exceeds the memory capacity of the GPU, the parallelization no longer applies. Then, the collocation points have to be divided into mini-batches and a stochastic gradient descent method (Adam) has to be used to perform the training (keeping the same number of iterations as before, *i.e.* 100 epochs for pre-training and 2000 epochs for the offline training). Then, the curse of dimensionality becomes very influential on the computation time, this is why for $N_k = 4$ the computation time grows by a factor of ~ 10 . This value corresponds to the 10 mini-batches that were used to perform the training in this configuration. Indeed, since the parameter domain \mathcal{K}_{col} has been sampled using a grid with 10 vertices in each dimension, going from $N_k = 3$ to $N_k = 4$ multiplied by 10 the number of collocation points, thus requiring to divide the dataset into 10 mini-batches to perform the training on the same GPU as before. For high-dimensional inverse problems, this method is out of reach, as the offline training phase requires to sample a hypercube of dimension $((d+1) + N_k)$ in order to train \mathcal{N}_θ . While d remains relatively small for most cases ($d \sim 1-3$), N_k can easily become huge, making the FP-PINN very prone to the curse of dimensionality.

4.2. Shallow-Water Equations

The objective here is to illustrate the capacity of the PINN-based approaches to perform identification of high-dimensional spatially-distributed parameter on a dynamic flow model, namely the non-linear hyperbolic system 2D SWE. As shown in Section 4.1, the FP-PINN is very prone to the curse of dimensionality and is therefore not a good candidate for identifying high-dimensional parameter on a more complex model, such as SWE. Thus, only the SP-PINN is used for inferring the spatially-distributed friction coefficient function $(x_1, x_2) \mapsto \mathbf{K}_s(x_1, x_2)$.

Moreover, we compare the performance of the SP-PINN with that of the well-established VDA method to better understand the advantages and disadvantages of each for solving high-dimensional inverse problems. Such identifiability problems in estimating parameters of a SWE flow model by VDA have been studied for example in [32, 48, 49, 50].

4.2.1. Setup

Initially, a test case is created in a 2D spatial domain Ω consisting in a 1000 m-long and 100 m-wide reach. The time domain is set to $[0 \text{ s}; 21600 \text{ s}]$ with 7 homogeneously distributed discrete time steps, one every 3600 s. The time interval $[0 \text{ s}; 7200 \text{ s}]$, here called the warm-up, is used to bring the flow at equilibrium. For the SP-PINN, the initial condition is set at 7200 s, therefore the time domain is set to $[7200 \text{ s}; 21600 \text{ s}]$ with 5 discrete time steps. The bathymetry is a scaled so-called Mac Donald's type bathymetry (see, *e.g.*, Delestre et al. [47]) along x_1 and a convex quadratic with 3 sinusoidal bumps along x_2 . The flow is in subcritical regime, therefore the downstream boundary condition Γ_{out} is set to $h_{out} = 0.7435 \text{ m} \quad \forall (x, t) \in \Gamma_{out} \times [0 \text{ s}; 21600 \text{ s}]$. The time-dependant flow rate $Q_{in}(t)$ at the input boundary Γ_{in} , shown in Fig. 10 is set to $100 \text{ m}^3/\text{s}/\text{m}$ between $t = 0 \text{ s}$ and $t = 7200 \text{ s}$ to set up the physical equilibrium, then to a ramp going linearly from $Q_{in}(t = 7200 \text{ s}) = 100 \text{ m}^3/\text{s}/\text{m}$ to $Q_{in}(t = 21600 \text{ s}) = 130 \text{ m}^3/\text{s}/\text{m}$:

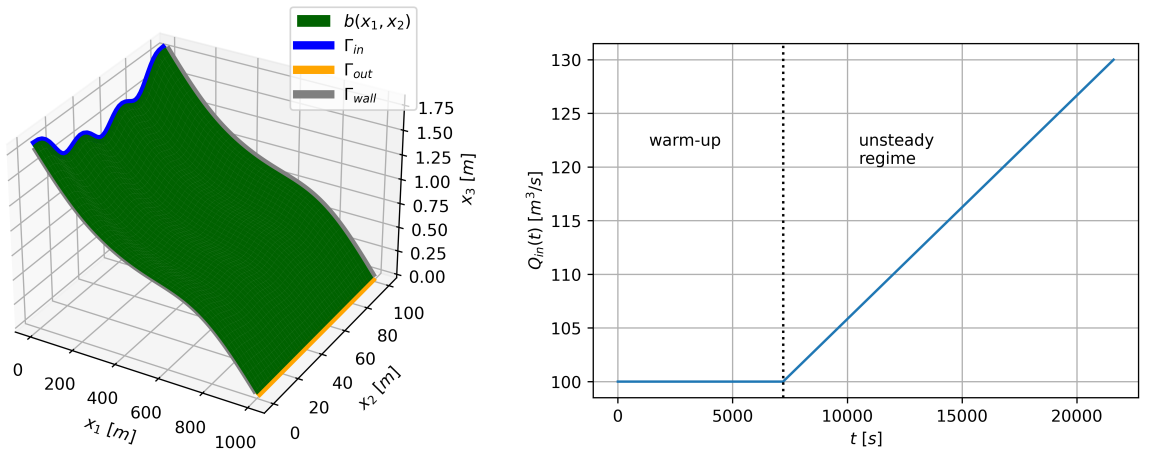


Figure 10: SWE test case. Left: bathymetry with boundary conditions, consisting in an imposed hydrograph at the input Γ_{in} , an imposed constant water height at the output Γ_{out} and wall conditions on the borders Γ_{wall} . Right: Hydrograph $Q_{in}(t)$ used at Γ_{in} .

The discretized parameter function \mathbf{K}_{sh} is defined on 1152 constant-value patches ($N_k = 1152$ with a piecewise constant interpolation in the domain Ω), on a (48×24) grid. The observations consist of water height values h_{obs} at 4608 points ($N_{obs} = 4608$), sampled on an uniform grid in the space-time domain (one observation per friction patch \times 4 time steps, not counting the SP-PINN initial condition at $t = 7200\text{s}$). Consequently, these observations do not provide direct information on the discharge q , making the inverse problem challenging since relying on partially observed state values only.

The reference solution from which observations are drawn is obtained using finite volumes method with a first-order Godunov spatial scheme and an explicit Euler time-stepping scheme (see Pujol et al. [34], Couderc et al. [35]).

4.2.2. Results

The inference of a friction parameter \mathbf{K}_{sh} of high dimension ($N_k = 1152$) from water height observations is studied here using the SP-PINN approach. The architecture consists of a fully-connected NN of shape [3, 60, 60, 60, 3] ($N_\theta = 9192$). The number of parameters is higher than one might expect for this architecture due to the presence of the Fourier features embedding. This embedding increases both the number of weights and biases and also adds various frequencies to the parameters. The Fourier features embedding is used with 4 features, whose initial values are set to (1, 2, 3, 4) for all physical variables. The value of λ is set to 1 for all components. The collocation points \mathcal{X}_{col} are sampled on a regular 3D grid with 50 vertices along x_1 , 30 vertices along x_2 and 5 vertices along t ($N_{col} = 7500$). Then, a pre-training consisting of 1000 L-BFGS iterations for the pre-training and 10 alternating minimization steps of 50 L-BFGS iterations each (40 iterations on θ and 10 on the parameter \mathbf{K}_{sh}), representing 500 L-BFGS iterations in total, are performed. The loss functions are re-normalized after the pre-training.

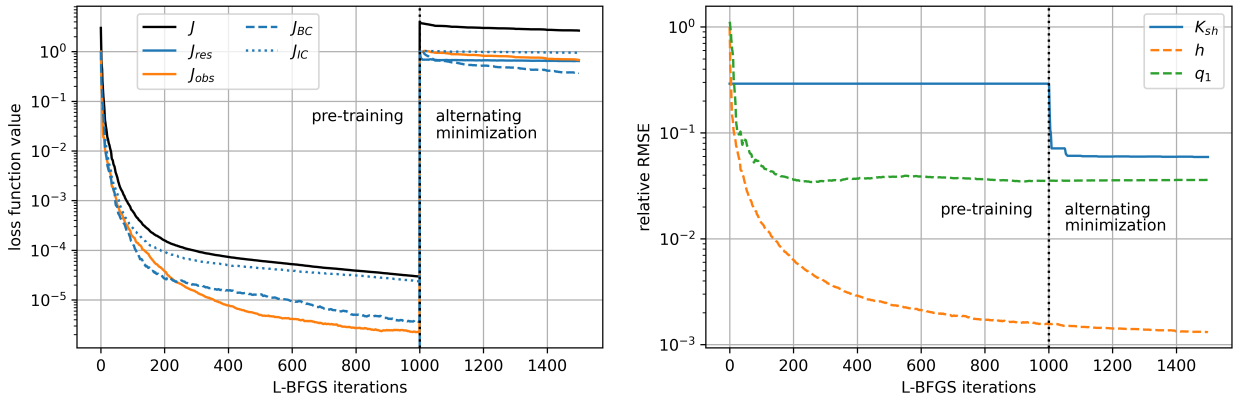


Figure 11: Results of the SP-PINN training (SWE) and high-dimensional parameter identification. Left: loss functions minimization during the pre-training and alternating minimization phases. Right: Relative RMSE minimization on h , q_1 and the spatially-distributed parameter \mathbf{K}_{sh} during the pre-training and alternating minimization phases. Results are not shown for q_2 for clarity sake.

Results for the loss minimization, during the pre-training and alternating minimization phases, as well as the relative RMSE minimization are presented in Fig. 11. The loss function shown in the graph on the left reaches a plateau after 1100 iterations, corresponding to a satisfying training. The same plateau can be observed on the relative RMSE both on the physical components of the solution h and q_1 , as well as on the parameter \mathbf{K}_{sh} . This indicates that the training has led to satisfying results on both optimization objectives, *i.e.* in term of physical quality of the surrogate flow model and in term of high-dimensional friction parameter identification. For q_2 , the relative RMSE is higher, which was expectable because of the smaller transversal flux q_2 with slight signature in the water surface, hence lower identifiability than q_1 on this mainly 1D velocity pattern. Therefore, results are not shown to avoid squeezing the vertical scale, but the same minimization pattern as q_1 is observed.

The results of water surface simulation and high-dimensional spatially-distributed friction parameter inference with the SP-PINN approach are presented in Fig. 12. The graph on the left shows that the calibrated flow model (grey surface) fits well the reference flow (blue surface) at time $t = 21600$ s, as they are indistinguishable from one another. As there are as many observations points in space as the number of friction patches, they are not shown in the figure for clarity sake. The graph on the right shows that the spatial pattern and values of friction are well retrieved by resolution of the inverse problem during the training of the SP-PINN. Note that the low frequency (large scale) patterns of the friction are identified (medium friction at the beginning of the domain, then high friction and low friction at the end). Some errors can be observed, where simulated water depth is less sensitive to friction and also in the influence zones of upstream and downstream boundaries of the domain. The latest is probably due to the fact that boundary conditions are taken into account in a naive way in the SP-PINN.

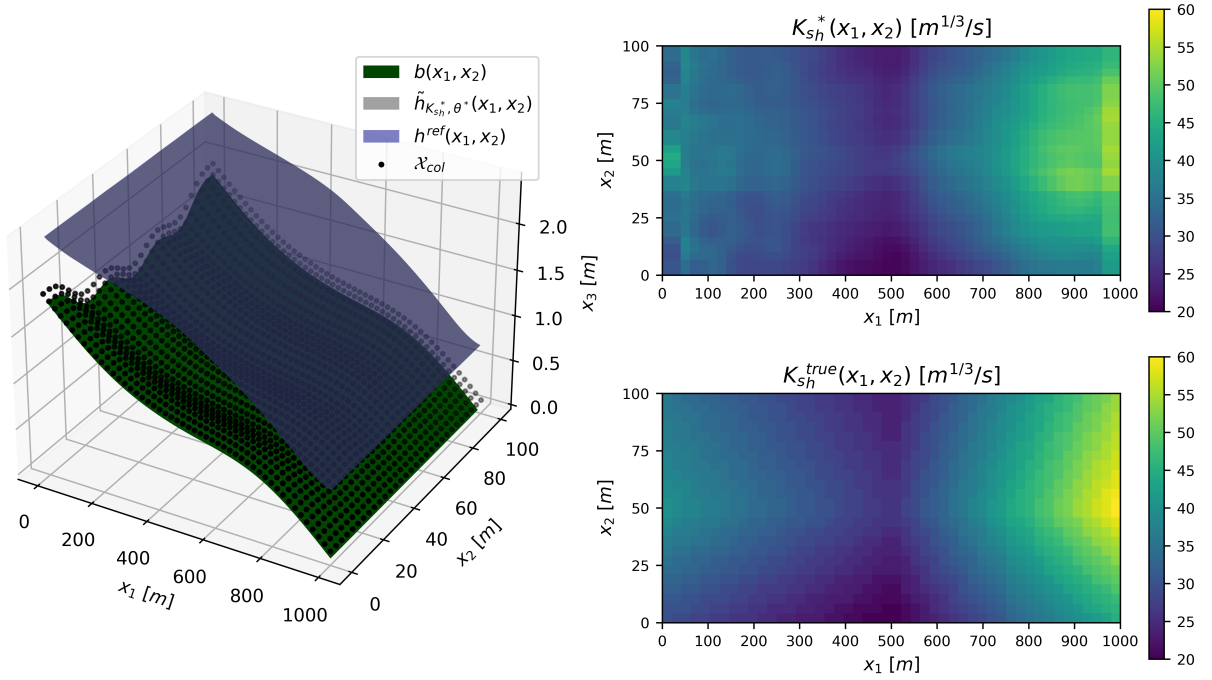


Figure 12: Results of the parameter identification with the SP-PINN (SWE). Left: calibrated model after training, showing the model solution $\tilde{h}_{\mathbf{K}_{sh}^*, \theta^*}(x_1, x_2)$ and the reference solution $h^{ref}(x_1, x_2)$ at time $t = 21600$ s, which are indistinguishable from one another. The collocation points and bathymetry are also shown. Right: inferred spatially-distributed parameter \mathbf{K}_{sh}^* patches values after 1500 L-BFGS iterations, compared with \mathbf{K}_{sh}^{true} patches values .

The error analysis is presented in Fig. 13. The graphs show the errors between the inferred value of \mathbf{K}_{sh}^* and the true value \mathbf{K}_{sh}^{true} by showing the signed error, the absolute error and the relative error. The friction field inference is satisfying with a global relative RMSE of 5.9 %. This relative RMSE goes to 8.4 % without Fourier features embedding (not shown for brevity). The error graphs on Fig. 13 highlight the higher inference errors near the boundary conditions, as explained before.

It is important to note that the SWE can be seen as a low-pass filter of the riverbed properties. More precisely, the higher frequencies of the friction and bathymetry patterns are filtered by the flow and are not discernable in the free surface signature (see [51, 52] and references therein). Therefore, as this method only uses water heights to retrieve the friction pattern, some information is lost because of the filtering properties of the flow.

Sensitivity of the simulated flow to the friction coefficient \mathbf{K}_{sh} . In the context of this test case, it is important to highlight the sensitivity of the water height h to the friction parameter \mathbf{K}_{sh} . Indeed, a global relative RMSE of about 10 % (upper bound) is observed on the inferred friction coefficient and it is important to understand how such a variation on the value of \mathbf{K}_{sh} affects the water height h (equivalently the free surface elevation H), as the observations consist of h values only. Therefore, the reference water height h of the SWE system (computed by the finite volume solver in DassFlow software [36]) is projected along x_1 and plotted in Fig. 14, showing for a constant friction coefficient value of $\mathbf{K}_{sh} = 20$ m^{1/3}/s and $\mathbf{K}_{sh} = 60$ m^{1/3}/s how a ± 10 % variation on \mathbf{K}_{sh} affects the free surface. The maximal deviation measured is 9 cm for $\mathbf{K}_{sh} = 20 \pm 10\%$ m^{1/3}/s and 6 cm $\mathbf{K}_{sh} = 60 \pm 10\%$ m^{1/3}/s. This corresponds to a sensitivity to the free surface with the SP-PINN method of 5 % on average in the domain and of 13 % at the maximum, on the case tested here. This sensitivity depends on flow physics, for example less sensitivity of flow depth to friction occurs *e.g.* in a channel with a pool triggering a significant flow deceleration and wetted section increase.

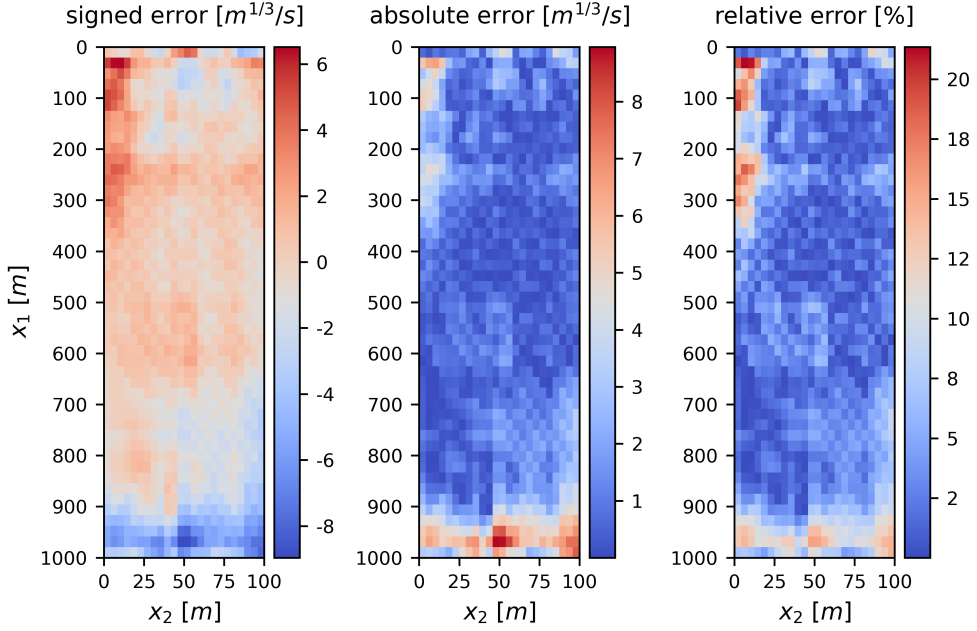


Figure 13: Error analysis between K_{sh}^* and K_{sh}^{true} . Left: Signed error. Middle: Absolute error. Right: Relative error with a median of 2.8 % and a standard deviation of 3.6 %. The errors near the boundaries are the result of the naive way in which boundary conditions are taken into account in the SP-PINN.

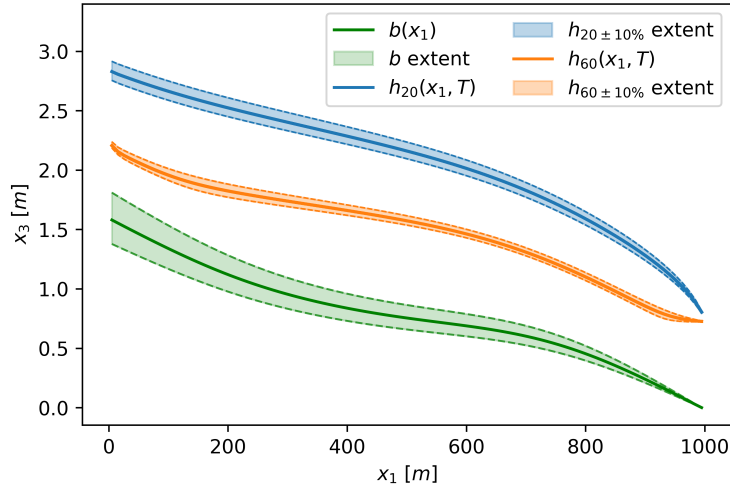


Figure 14: Free surface sensitivity to a $\pm 10\%$ increase in K_{sh} at $T = 21600$ s. The maximal deviation for the blue curve is 9 cm and 6 cm for the orange one. This corresponds to a sensitivity of the water height h to the friction parameter K_{sh} of 5 % on average in the domain and a maximum sensitivity of 13 % for this test case.

4.2.3. Comparison with Variational Data Assimilation

The results of the inference of a high-dimensional friction parameter in a SWE-based model with the SP-PINN method are compared to a state-of-the-art VDA algorithm, which has been demonstrated to be well-suited for the present inverse problem (see, e.g., [32, 34, 53]). A VDA algorithm implemented in the DassFlow software [36] and based on accurate cost gradients obtained by solving the adjoint model, is used to tackle the same spatially-distributed friction parameter identification problem as with the SP-PINN. A quasi Newton L-BFGS algorithm

(M1QN3 algorithm, see [54]), adapted to high-dimensional inverse problems, is used for the optimization without any regularization term. The convergence curves and the relative RMSE minimization between the inferred value of \mathbf{K}_{sh} and \mathbf{K}_{sh}^{true} are shown in Fig. 15. The present VDA computation is performed on an 11th Gen Intel Core i9-11950H CPU, which is more adapted to VDA than the GPU used for training the SP-PINN.

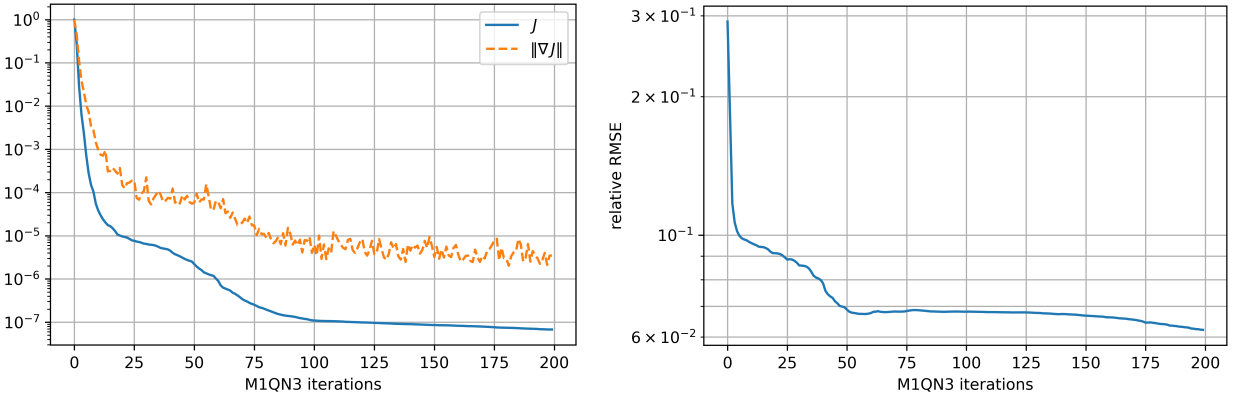


Figure 15: VDA convergence for the reference SWE-based model. Left: loss function and gradient norm minimization. Right: Relative RMSE between \mathbf{K}_{sh} and \mathbf{K}_{sh}^{true} minimization.

A direct comparison between the \mathbf{K}_{sh}^* inferred with VDA and the \mathbf{K}_{sh}^* inferred with the SP-PINN is shown in Fig. 16, as well as the differences in Fig. 17. It is important to note that boundary conditions are taken into account very differently between the SP-PINN method and the VDA method. More specifically, in this work, the wall conditions for the SP-PINN simply consist in imposing $q_2 = 0$ on the boundary, which seems very basic compared to what is done in VDA. This results in significant differences in the inferred parameter near the boundaries.

The global relative RMSE on the \mathbf{K}_{sh}^* is comparable with both methods: 6.2 % when inferred with VDA and 5.9% when inferred with the SP-PINN, demonstrating the performance of the latter. Nevertheless, the inference error with the SP-PINN stems from classical identifiability issues on physical parameters given the available observations but also depends on the quality of the learnt surrogate. Thus, it makes the interpretation of the SP-PINN inference error more difficult. It should also be noted that the computation time is very different from one method to the other. As a first comparison, both calculations have been performed on a single CPU thread and the results are: 30 hours of computation time for the VDA against 70 seconds for the SP-PINN, which represents a computational time factor of ~ 1500 between the two methods.

However for a fair comparison, the calculations have also been performed in parallel for the SP-PINN on the GPU (with 4096 CUDA cores as described earlier) and the computation time drops to 35 seconds. While for the VDA, a MPI computation on 6 CPU threads (for the 4608 mesh cells) provides about 5 hours of CPU-time computation for the standard laptop CPU aforementioned, bringing the computational time factor to ~ 500 .

5. Conclusion

This work aimed to evaluate the capabilities of two PINN-based approaches for handling high-dimensional parameter identification problems and compare their performance to established DA methods. To this end, a common formalism from the perspective of DA has been introduced, to revisit and compare classical inverse methods and new Machine Learning algorithms. Then, the two different PINN-based approaches were tested on academic test cases consisting in inferring a spatially-distributed friction coefficient used in an ODE-based hydraulics model (the Backwater Equation). Several best practices (Fourier features, pre-training, alternating minimization) that are crucial for the training of such methods have also been highlighted. The associated findings show that one of the two methods is capable of solving efficiently high-dimensional inverse problems: the here called Semi-Parameterized PINN (SP-PINN), actually corresponding to the original method introduced in Raissi et al. [17]. To illustrate its performance,

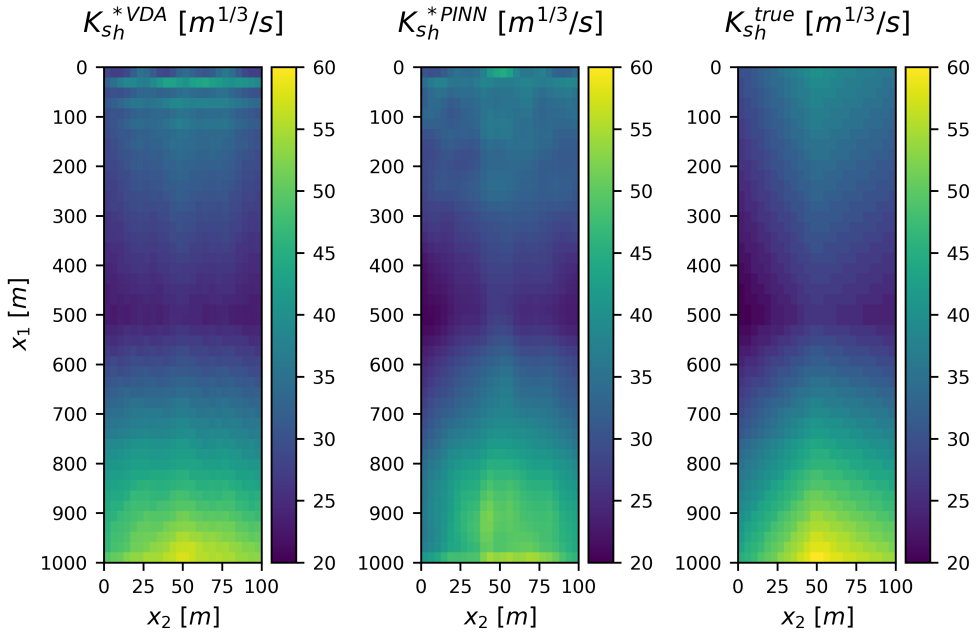


Figure 16: Comparison between the K_{sh}^* inferred with VDA and with the SP-PINN. Left: K_{sh}^* inferred with VDA after 200 M1QN3 iterations (5 h on 6 CPU threads). Middle: K_{sh}^* inferred with SP-PINN after 1500 L-BFGS iterations (1000 iterations for the pre-training and 500 iterations for the alternating minimization, 35 s in total on 1 GPU). Right: K_{sh}^{true} for comparison.

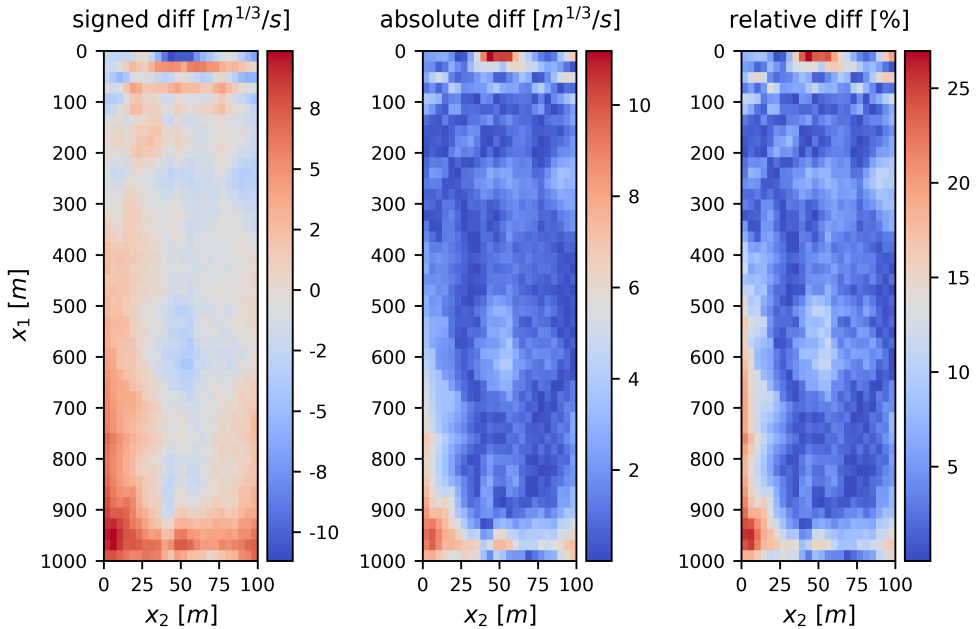


Figure 17: Difference between the K_{sh}^* inferred with VDA and with SP-PINN. Left: Signed difference. Middle: Absolute difference. Right: Relative difference, with a median of 4.1 % and a standard deviation of 4.8 %. The significant differences near the boundaries arise from the different ways in which boundary conditions are taken into account in the SP-PINN and VDA.

this method was then successfully used to identify a $\mathcal{O}(10^3)$ -dimensional spatially-distributed friction parameter on a SWE-based hydraulics modelling inverse problem. Another important contribution of this work lies in the comparison of the SP-PINN method with the well-established VDA method (based on the adjoint model technique), where the direct model is enforced pointwise as a strong constraint in the optimization process, unlike the SP-PINN where the direct model is enforced through its residual only. In the test case presented in this work, both the SP-PINN and the more traditional VDA provide comparable inferred parameter accuracy, however the SP-PINN requires $\mathcal{O}(10^2)$ – $\mathcal{O}(10^3)$ times less computational time than VDA.

At first glance, Fully-Parameterized PINNs (FP-PINNs) seems to be preferable for parameter identification because they directly approximate the parameter-to-state operator, allowing the NN to be trained only once, albeit at a significant cost, and then reused in multiple parameter identification phases. However, in high-dimensional inverse problems scenarios, this approach becomes impractical since FP-PINNs require feeding the parameters to be inferred as inputs to the NN. Therefore, the SP-PINN is preferable, where the parameters to be inferred are treated as parameters of the NN rather than inputs, making it practicable in high-dimensional inverse problems. Nevertheless, SP-PINNs require retraining each time a new inference is needed, unlike FP-PINNs.

The computational cost of training a SP-PINN is significantly lower than the cost of the well-established VDA method. However, as already well-known (see, *e.g.*, [55]), PINNs approaches require lots of trial and error to set the hyperparameters in order to achieve correct identification, such as the size and architecture of the NN, the activation functions, the number of collocation points and more critically, the weight factors in the loss function defined in Eq. (12). Indeed, the latter have a considerable impact on the inference accuracy. Therefore, under the assumption that the hyperparameters can be set efficiently, the very low computational cost and the ease of implementation and maintenance of the SP-PINN make it a very interesting approach for tackling high-dimensional inverse problems based on highly non-linear PDE systems.

Even when these methods are challenging to set up, such as in complex real-world scenarios (see, *e.g.*, [34] in the present river hydraulics context), they can still serve as initializers for conventional DA algorithms or act as surrogates. For these reasons, PINNs are a very promising method for the future of high-dimensional inverse problems resolution that deserve to be further investigated to fully leverage their computational cost advantage over conventional methods.

Acknowledgments

The PhD of the first author has been funded by the French national agency ANR, project "Multi-scale Flood Forecasting with INnovating Solutions" (MUFFINS), ANR-21-CE04-0021-01.

References

- [1] M. Bonavita. Overview of data assimilation methods, 2019. URL https://events.ecmwf.int/event/333/contributions/3863/attachments/2246/3965/2_DataAssim_Overview_Bonavita_2023.pdf. ECMWF course online.
- [2] A. Carrasi, M. Bocquet, L. Bertino, and G. Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.
- [3] M. Durand, C.J. Gleason, T.M. Pavelsky, R. Prata de Moraes Frasson, M. Turmon, C.H. David, E.H. Altenau, N. Tebaldi, K. Larnier, J. Monnier, et al. A framework for estimating global river discharge from the surface water and ocean topography satellite mission. *Water Resources Research*, 59(4):e2021WR031614, 2023.
- [4] K. Larnier and J. Monnier. Hybrid neural network-variational data assimilation algorithm to infer river discharges from swot-like data. *Computational Geosciences*, 27(5):853–877, 2023.
- [5] M. Asch, M. Bocquet, and M. Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.
- [6] M. Bocquet. Introduction to the principles and methods of data assimilation in geosciences, 2023. URL <https://citeserx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1eed296a1003aee50809969a22acb66479c72415>. Lectures notes, École des Ponts ParisTech.
- [7] J. Monnier. Data Assimilation. Inverse Problems, Assimilation, Control, Learning, 2024. URL <https://hal.science/hal-03040047v2/file/main.pdf>. Lectures notes, INSA Toulouse.
- [8] E. Calvello, S. Reich, and A. M. Stuart. Ensemble kalman methods: A mean field perspective. *arXiv preprint arXiv:2209.11371*, 2022.
- [9] G. Evensen. *Data assimilation: the ensemble Kalman filter*, volume 2. Springer, 2009.
- [10] P. Courtier and O. Talagrand. Variational assimilation of meteorological observations with the direct and adjoint shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, 42(5):531–549, 1990.

- [11] F.-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus A: Dynamic Meteorology and Oceanography*, 38(2):97–110, 1986.
- [12] O. Thual. *Hydrodynamique de l'environnement*. Editions Ecole Polytechnique, 2010.
- [13] C. Ancey. *Hydraulique à surface libre*, 2010. URL https://lhe.epfl.ch/cours/mastergc/cours-hydraulique_2010.pdf. Lecture notes, École Polytechnique Fédérale de Lausanne.
- [14] Y. Sasaki. An objective analysis based on the variational method. *Journal of the Meteorological Society of Japan. Ser. II*, 36(3):77–88, 1958.
- [15] D.C. Psychogios and L.H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.
- [16] 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.
- [17] 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.
- [18] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1): 208–228, 2021.
- [19] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni. Neurodifq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931, 2020.
- [20] D. Feng, Z. Tan, and Q. He. Physics-informed neural networks of the saint-venant equations for downscaling a large-scale river model. *Water Resources Research*, 59(2):e2022WR033168, 2023.
- [21] A. Bihlo and R. O. Popovych. Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, 456:111024, 2022.
- [22] R. Anelli and E. Miglio. Physics-Informed Neural Networks for Shallow Water Equations, 2022. URL https://www.politesi.polimi.it/bitstream/10589/195179/2/RA_Thesis.pdf. Master thesis, Politecnico di Milano.
- [23] R. Fablet, B. Chapron, L. Drumetz, E. Mémin, O. Pannekoucke, and F. Rousseau. Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10), 2021.
- [24] M. Beauchamp, Q. Febvre, H. Geogrentum, and R. Fablet. 4dvarnet-ssh: end-to-end learning of variational interpolation schemes for nadir and wide-swath satellite altimetry. *Geoscientific Model Development Discussions*, 2022:1–37, 2022.
- [25] 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.
- [26] S. Cedillo, A.-G. Núñez, E. Sánchez-Cordero, L. Timbe, E. Samaniego, and A. Alvarado. Physics-informed neural network water surface predictability for 1d steady-state open channel cases with different flow types and complex bed profile shapes. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):10, 2022.
- [27] A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, and D. Barajas-Solano. Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5), 2020.
- [28] D.N. Tanyu, J. Ning, T. Freudenberg, N. Heilenkötter, A. Rademacher, U. Iben, and P. Maass. Deep learning methods for partial differential equations and related parameter identification problems. *Inverse Problems*, 39(10):103001, 2023.
- [29] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of computational physics*, 446:110666, 2021.
- [30] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deep neural networks. *Science advances*, 7(40):eabi8605, 2021.
- [31] S.L. Dingman. *Fluvial hydraulics*. Oxford university press, 2009.
- [32] J. Monnier, F. Couderc, D. Dartus, K. Larnier, R. Madec, and J.-P. Vila. Inverse algorithms for 2D shallow water equations in presence of wet dry fronts: Application to flood plain dynamics. *Advances in Water Resources*, 97:11–24, 2016.
- [33] K. Larnier, J. Monnier, P.-A. Garambois, and J. Verley. River discharge and bathymetry estimation from swot altimetry measurements. *Inverse problems in science and engineering*, 29(6):759–789, 2021.
- [34] L. Pujol, P.-A. Garambois, and J. Monnier. Multi-dimensional hydrological–hydraulic model with variational data assimilation for river networks and floodplains. *Geoscientific Model Development*, 15(15):6085–6113, 2022.
- [35] F. Couderc, R. Madec, J. Monnier, and J.-P. Vila. Dassflow v2: user and developer guide. Technical report, Internal technical report CNRS/IMT/INSA Toulouse, 2015. URL https://hal.science/hal-01120285/file/doc_dassflow_v2.pdf.
- [36] DassFlow (Data Assimilation for Free Surface Flows), open-source computational software, since 2008. URL <https://github.com/DassHydro/dassflow2d>.
- [37] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [38] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.
- [39] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deep neural networks based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.
- [40] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.
- [41] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.
- [42] V. Gopakumar, S. Pamela, and D. Samaddar. Loss landscape engineering via data regulation on pinns. *Machine Learning with Applications*, 12:100464, 2023.
- [43] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

- [44] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.
- [45] Z. Aldirany, R. Cottreau, M. Laforest, and S. Prudhomme. Multi-level neural networks for accurate solutions of boundary-value problems. *Computer Methods in Applied Mechanics and Engineering*, 419:116666, 2024.
- [46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [47] O. Delestre, C. Lucas, P.-A. Ksinant, F. Darboux, C. Laguerre, T-N-T. Vo, F. James, and S. Cordier. Swashes: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, 72(3):269–300, 2013.
- [48] M. Honnorat, J. Monnier, N. Rivière, É. Huot, and F.-X. Le Dimet. Identification of equivalent topography in an open channel flow using lagrangian data assimilation. *Computing and visualization in science*, 13:111–119, 2010.
- [49] P. Brisset, J. Monnier, P.-A. Garambois, and H. Roux. On the assimilation of altimetric data in 1d saint-venant river flow models. *Advances in water resources*, 119:41–59, 2018.
- [50] L. Pujol, P.-A. Garambois, P. Finaud-Guyot, J. Monnier, K. Larnier, R. Mose, S. Biancamaria, H. Yesou, D. Moreira, A. Paris, et al. Estimation of multiple inflows and effective channel by assimilation of multi-satellite hydraulic signatures: The ungauged anabranching negro river. *Journal of Hydrology*, 591:125331, 2020.
- [51] J. Monnier and P.-E. des Bosc. Inference of the bottom properties in shallow ice approximation models. *Inverse Problems*, 33(11):115001, 2017.
- [52] N. Martin and J. Monnier. Inverse rheometry and basal properties inference for pseudoplastic geophysical flows. *European Journal of Mechanics-B/Fluids*, 50:110–126, 2015.
- [53] P.-A. Garambois, K. Larnier, J. Monnier, P. Finaud-Guyot, J. Verley, A.-S. Montazem, and S. Calmant. Variational estimation of effective channel and ungauged anabranching river discharge from multi-satellite water heights of different spatial sparsity. *Journal of Hydrology*, 581: 124409, 2020.
- [54] J.-C. Gilbert and C. Lemaréchal. Some numerical experiments with variable-storage quasi-newton algorithms. *Mathematical programming*, 45(1):407–435, 1989.
- [55] G. E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.