Physics informed neural networks for solving flow problems modeled by the Shallow Water Equations

Xin Qi[1], Gustavo Adolfo Mazza de Almeida[1], and Sergio Maldonado[1]

[1]University of Southampton

June 23, 2023

Abstract

This paper investigates the application of physics-informed neural networks (PINNs) to solve free-surface flow problems governed by the two-dimensional shallow water equations (SWEs). Two types of PINNs are developed and analysed: a physics-informed fully connected neural network (PIFCN) and a physics-informed convolutional neural network (PICN). The PINNs eliminate the need for labelled data for training by employing the SWEs, initial and boundary conditions as components of the loss function to be minimized. Solutions obtained by both PINNs are compared against those delivered by a finite volume (FV) solver for two idealized problems admitting analytical solutions, and one real-world flood event. The results of these tests show that the prediction accuracy and computation time (i.e., training time) of both PINNs may be less affected by the resolution of the domain discretization than the FV model. Overall, the PICN shows a better trade-off between computational speed and accuracy than the PIFCN. Also, our results for the two idealized problems indicated that PICN and PIFCN can provide more accurate predictions than the FV model, while the FV simulation with coarse resolution (e.g., 5 m and 10 m) outperformed PICN and PIFCN in terms of the speed-accuracy trade-off. Results from the real-world test showed the finely resolved (10 m resolution) FV simulation generally provided the most accurate approximations at flooding peaks. However, both PINNs showed better speed-accuracy trade-off than the FV model in terms of predicting the temporal distribution of water depth, while FV models outperformed the PINNs in their predictions of discharge.

Physics informed neural networks for solving flow problems modeled by the Shallow Water Equations

Xin Qi[1]

Gustavo A. M. de Almeida[1]

Sergio Maldonado[1]

[1]Faculty of Engineering and Physical Sciences, University of Southampton, Southampton SO16 7QF, UK

Key Points:

- Two PINNs are developed to find solutions to the 2D shallow water equations.
- In some scenarios, PINN accuracy is comparable to or surpasses finite-volume.
- Convolutional neural network outperformed fully connected neural network results.

Corresponding author: Xin Qi, `xq1n19@soton.ac.uk`

Abstract

This paper investigates the application of physics-informed neural networks (PINNs) to solve free-surface flow problems governed by the two-dimensional shallow water equations (SWEs). Two types of PINNs are developed and analysed: a physics-informed fully connected neural network (PIFCN) and a physics-informed convolutional neural network (PICN). The PINNs eliminate the need for labelled data for training by employing the SWEs, initial and boundary conditions as components of the loss function to be minimized. Solutions obtained by both PINNs are compared against those delivered by a finite volume (FV) solver for two idealized problems admitting analytical solutions, and one real-world flood event. The results of these tests show that the prediction accuracy and computation time (i.e., training time) of both PINNs may be less affected by the resolution of the domain discretization than the FV model. Overall, the PICN shows a better trade-off between computational speed and accuracy than the PIFCN. Also, our results for the two idealized problems indicated that PICN and PIFCN can provide more accurate predictions than the FV model, while the FV simulation with coarse resolution (e.g., 5 m and 10 m) outperformed PICN and PIFCN in terms of the speed-accuracy trade-off. Results from the real-world test showed the finely resolved (10 m resolution) FV simulation generally provided the most accurate approximations at flooding peaks. However, both PINNs showed better speed-accuracy trade-off than the FV model in terms of predicting the temporal distribution of water depth, while FV models outperformed the PINNs in their predictions of discharge.

1 Introduction

Free-surface flow phenomena are usually modeled by the shallow water equations (SWEs), a nonlinear system of partial differential equations (PDEs) governing the evolution of water depth and vertically averaged velocity in the two horizontal dimensions. Over the last decades, significant efforts have been made to approximate the solution to the SWEs in a discretized form through numerical methods, such as Finite Difference (FD) (e.g., Casulli, 1990; Molls & Chaudhry, 1995; Kurganov & Levy, 2002, and others), Finite Volume (FV) (e.g., Alcrudo & Garcia-Navarro, 1993; Bale et al., 2003; Botta et al., 2004; Yoon & Kang, 2004; Toro & Garcia-Navarro, 2007, and others), or Finite Element (FE) (e.g., Lynch & Gray, 1979; Hanert et al., 2005; Dawson et al., 2006; Marras et al., 2016, and others). These methods are now well-established and have been the object of extensive tests and validation (e.g., Toro & Garcia-Navarro, 2007; Wilson et al., 2007; Liang & Marche, 2009; LeVeque et al., 2011, and others). While the ability of these models to capture the main properties of free-surface flows has been widely verified, accurate solutions to real-world problems typically require the use of a finely resolved computational mesh, which tends to significantly increase the computational cost (Bernard et al., 2009; Liang, 2011; Juez et al., 2014). In explicit numerical schemes for the solution of the 2D SWEs, the computational cost C scales cubically with the size of the computational grid Δx (i.e., $C \sim \Delta x^{-3}$). As a result, important applications such as large-scale flood simulations are often beyond the capabilities of available numerical methods given existing computational resources. This is particularly challenging when simulations need to be performed in real-time, or as part of a probabilistic flood risk analysis (Leskens et al., 2014; Sanders & Schubert, 2019; Ferrari & Vacondio, 2022; J. Li et al., 2022). For example, flood risk management usually requires the simulation of a large number of inundation scenarios, which may increase the overall computational time by orders of magnitude. Although the computational speed of models can be improved by using state-of-the-art hardware and parallel computing algorithms (Leandro et al., 2014; Monnier et al., 2016), such techniques may still be insufficient to meet the computational requirements of many important applications (Kabir et al., 2020). Owing to these limitations, a cost-effective model for free-surface problems may offer an appealing alternative.

Over the last decades, Machine Learning (ML) models, and Artificial Neural Networks (ANNs) in particular, have found a wide range of applications, such as in financial prediction (Henrique et al., 2019), facial recognition (Sulavko, 2020), supply chain optimization (Carbonneau et al., 2008), radiation forecasting (El Naqa et al., 2015), and automatic cancer screening (William et al., 2018), to cite only a few. The extraordinary increase in applications of ML models is largely due to their ability to mathematically describe any nonlinear relationship between inputs and outputs according to the universal approximation theorem (Hornik et al., 1989), the increasing availability of data for training, and increasing computational power.

The first works using ML for the solution of problems governed by the SWEs are relatively recent and focused on the development of simple meta-models (e.g., Kabir et al., 2020; Liu & Pender, 2015; Bermúdez et al., 2019; Mahesh et al., 2022, and others). In this type of model, ML is used to build a prediction model to describe the input-output relationship previously obtained through the solution of the governing PDEs by another numerical approximation model; i.e. the ML model thus becomes a surrogate model. These surrogate models typically need to be trained using the results of a large number of numerical simulations conducted at fine resolution, which can be very computationally demanding.

Physics-Informed Neural Networks (PINNs), for which data (and therefore expensive numerical simulations) are not required for training, have gained increasing attention in recent years (Pang et al., 2019; Mao et al., 2020; Cai et al., 2021; Krishnapriyan et al., 2021; Jin et al., 2021). A PINN is essentially a ML algorithm which uses the information contained in the physical laws (such as the governing PDEs, boundary and initial conditions) to train the model. This eliminates the need for training data and thus, expensive numerical simulations. A data-free PINN is required to satisfy the governing PDEs, Initial Conditions (ICs) and Boundary Conditions (BCs) simultaneously. Recent applications of ML algorithms to solve complex physics phenomena have focused on the use of Deep Learning (DL) models (e.g., Sun et al., 2020; Zhang et al., 2020; Haghighat et al., 2020; Vlassis & Sun, 2021, and others). DL is a form of ANN with more than one hidden layer, which provides the complexity required to model intricate nonlinear relationships, such as those found in computer vision (Voulodimos et al., 2018), health management (Khan & Yairi, 2018), language translation (Rastgoo et al., 2021), and remote sensing (X. X. Zhu et al., 2017). In the past few years, a large number of data-free PINNs have been developed by employing DL techniques, such as the Fully Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs). For example, in Raissi et al. (2019) several FCNNs were trained to predict the solutions to various systems of PDEs, including Allen–Cahn, Schrödinger, Navier–Stokes, and Korteweg–de Vries equations. In the context of fluid dynamics, other implementations of FCNNs include those of Sun et al. (2020) and Mao et al. (2020), who used their DL models to find solutions to the Navier-Stokes (in steady state) and Euler equations (involving shock waves), respectively. The use of CNNs has also been explored, for instance, for problems governed by the Navier-Stokes (Cai et al., 2021) and Boltzmann transport equations (R. Li et al., 2021), or for predicting steady flow in random heterogeneous media (Y. Zhu et al., 2019). The success of these works shows that data-free PINNs should be considered as serious contenders for solving flow problems that are typically modelled by PDEs, and which have been traditionally solved using conventional numerical methods (FD, FV, etc.).

While ML trained from labeled data (i.e., mainly conventional numerical solutions) has been used to solve the SWEs (e.g., Mahesh et al., 2022; Ştefănescu et al., 2014; Yıldız et al., 2021; C. Li et al., 2023, and others), to the authors' knowledge only a very limited number of articles (e.g., Bihlo & Popovych, 2022) has been published so far on the use of a data-free PINNs for this purpose. In Bihlo and Popovych (2022), a PINN (specifically, based on a FCNN) was employed to find solutions to the SWEs on a spherical domain, and the focus was on idealized problems which may find applications in meteorology. Whether a similar DL technique may be used to accurately and efficiently solve challenging free-surface flow problems involving friction and complex boundary conditions, such as large-scale simulations of flow over complex topography in rivers and coastal areas, remains an open question.

The solution of PDEs, and in particular of the SWEs, using PINN algorithms is still in its infancy and further investigation is required to understand the main characteristics of solutions obtained by these methods. Firstly, the trainset for PINNs needs to be generated from a particular, discrete, set of points. It remains unclear how accuracy and computational performance (i.e., training speed) depend on the discretization of the domain. Additionally, both FCNNs and CNNs are commonly used DL models in the research field of PINN. However, in a specific problem governed by a system of PDEs, it is usually difficult to determine which one will deliver the best performance before carrying out tests.

The aim of this paper is to develop and test two different PINN models to approximate solutions to various free-surface flow problems governed by the 2D SWEs. The PINN models are based on the FCNN and CNN approaches, and are hereafter referred to as PIFCN (physics informed fully connected network) and PICN (physics informed convolutional network), respectively. These models are data-free in that they do not require data from separate numerical simulations, or laboratory/field measurements, to train the networks. In this paper both PIFCNs and PICNs are compared against the Finite Volume (FV) solver of the 2D SWE developed by de Almeida et al. (2016) through a set of test cases including two idealized flow problems and one real-world flood event. The rest of this paper is organized as follows. First, the governing equations and the framework of both PINNs are described in Section 2. This section also provides a concise review of FCNNs and CNNs. In Section 3, the accuracy and computational performance of the proposed physics-informed networks (PIFCN and PICN) are investigated for the three test cases. The main outcomes of the study are discussed and summarized in Section 4.

2 Methods

2.1 Overview

Most problems requiring the simulation of free-surface flows in the horizontal plane, such as flow in rivers and estuaries, dam-breaks, and flood wave propagation can be modeled by the SWEs. The 2D SWEs represent a system of nonlinear, hyperbolic PDEs describing the conservation of water mass and depth-average momentum, which can be expressed as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = \mathbf{S}(\mathbf{U}) \; ; \tag{1}$$

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \;\; \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \;\; \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}, \;\; \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ gh(s_{ox} - s_{fx}) \\ gh(s_{oy} - s_{fy}) \end{bmatrix}, \tag{2}$$

where x and y are the spatial (horizontal) coordinates; t is time; $h(x, y, t)$ is the water depth; $u(x, y, t)$ and $v(x, y, t)$ denote the x and y components of the depth-averaged flow velocity, respectively; $s_{ox} = -\partial z/\partial x$ and $s_{oy} = -\partial z/\partial y$ are the bed slopes in the x and y directions, respectively, and $z(x, y)$ is the terrain elevation (assumed constant in time); s_{fx} and s_{fy} denote the friction slopes in the x and y directions, respectively. The friction slopes can be modeled using Manning-Strickler's expression, $s_{fx} = n^2 u\sqrt{u^2 + v^2} h^{-4/3}$, $s_{fy} = n^2 v\sqrt{u^2 + v^2} h^{-4/3}$, where n is the Manning coefficient. Solutions $\mathbf{U} = (h, hu, hv)^T$ to this system (subject to well-posed boundary and initial conditions) can be computed by several numerical methods available, as discussed in the Introduction.

In this paper we propose a ML-based solution to this problem, whereby the input layer $\mathbf{x}$ represents the independent variables and parameters of the problem, $\mathbf{x} = (x, y, t, n, z)$, and the trained model $\aleph$ is expected to provide an approximate solution for $h(\mathbf{x})$, $hu(\mathbf{x})$ and $hv(\mathbf{x})$ in the corresponding domain; in other words:

$$\mathbf{U}(\mathbf{x}) \;\cong\; \tilde{\mathbf{U}}(\mathbf{x}) \;=\; \aleph(\mathbf{x}\;;\boldsymbol{\Gamma}), \tag{3}$$

where $\tilde{\mathbf{U}}(\mathbf{x})$ denotes the output from the PINN, which is in turn defined by the group of trainable parameters $\boldsymbol{\Gamma}$ (e.g., convolutional filter, weights and biases). The PINN models proposed in this paper are trained by minimizing the composite loss function, defined as:

$$\mathcal{L} = \boldsymbol{\lambda}_1 \cdot \boldsymbol{\mathcal{L}}_p + \boldsymbol{\lambda}_2 \cdot \boldsymbol{\mathcal{L}}_b + \boldsymbol{\lambda}_3 \cdot \boldsymbol{\mathcal{L}}_0, \tag{4}$$

where $\mathcal{L}$ (a scalar) is the loss function to be minimized, $\boldsymbol{\lambda}_{1-3}$ are the vectors of penalty coefficients for every specific loss term; namely, $\boldsymbol{\mathcal{L}}_p$ penalizes the residuals of the SWEs, $\boldsymbol{\mathcal{L}}_b$ and $\boldsymbol{\mathcal{L}}_0$ penalize the BCs (subscript b) and ICs (subscript 0) residuals, respectively. These loss terms are in turn given by:

$$\boldsymbol{\mathcal{L}}_p = \frac{1}{N}\sum_{i=1}^{N} |\partial_t \tilde{\mathbf{U}}_i + \partial_x \mathbf{F}(\tilde{\mathbf{U}}_i) + \partial_y \mathbf{G}(\tilde{\mathbf{U}}_i) - \mathbf{S}(\tilde{\mathbf{U}}_i)| \tag{5a}$$

$$\boldsymbol{\mathcal{L}}_b = \frac{1}{N_b}\sum_{i=1}^{N_b} |\tilde{\mathbf{U}}_{b,i} - \mathbf{U}_{b,i}| \tag{5b}$$

$$\boldsymbol{\mathcal{L}}_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |\tilde{\mathbf{U}}_{0,i} - \mathbf{U}_{0,i}| \tag{5c}$$

The tilde symbol ($\tilde{}$) denotes neuronal network output and the subscript $i \in [1, N]$ refers to the i^{th} collocation point. N represents the number of collocation points, which in this paper is defined from a uniformly discretized domain of the independent variables $N = n_x \times n_y \times n_t$, where n_x, n_y and n_t are the number of points used to discretize the domain along the x, y and t coordinates, respectively. The boundaries of the spatio-temporal domain are represented by a subset of N; in particular, the model will employ $N_b < N$ and $N_0 < N$ points to define the BCs and ICs, respectively.

For each approximate solution produced by the PINN, the partial derivatives in Eq. 5a are computed through the method of automatic differentiation (autodiff) (Paszke et al., 2017), which back-propagates derivatives from the outputs to the targeted inputs through the chain rule to compute the desired derivatives (Cai et al., 2021; Baydin et al., 2018). Thus, the partial derivatives of the approximate solution with respect to the independent variables can be computed without the errors common to numerical differentiation techniques. The loss function is minimised using the gradient descent method, with gradients of the loss function with respect to trainable parameters computed by back-propagation. These parameters can be updated either using all, or a subset (batch) of the collocation points.

One significant difficulty of solutions to flow problems modeled by the SWEs is the so-called wet-dry front issue (i.e., moving boundary). Physically, the value of the flow depth h cannot be negative. Areas of the domain where such solutions may be obtained correspond to dry areas, which are not governed by the SWEs. To overcome this problem, we set $\boldsymbol{\mathcal{L}}_p = 0$ if the predicted value of $\tilde{h}$ is negative. This ensures that the model does not penalize making predictions outside the wet domain.

Figure 1 shows a diagram illustrating the overall modeling framework proposed in this paper for solving the SWEs by a PINN method. Note that the collocation points can be chosen randomly in the space-time domain and their number prescribed. The general steps are outlined below.

1. Define the architecture of the PINN
2. Initialize the hyperparameters for the PINN
3. Compute the outputs from the PINN with given inputs
4. Compute the derivatives with respect to x, y, t and the corresponding loss $\mathcal{L}$
5. Update the PINN based on $\mathcal{L}$
6. Repeat steps 3 to 5 until the end of the user-prescribed number of training epochs

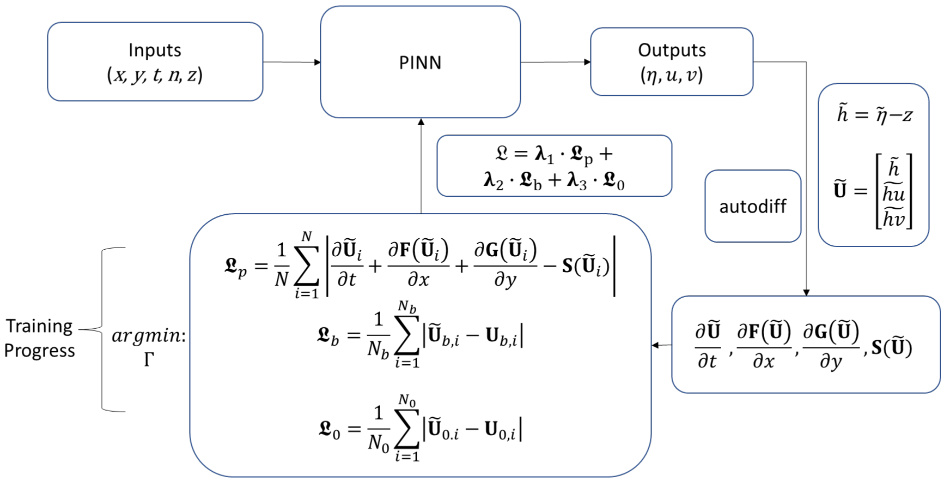

Figure 1. A schematic diagram of a physics informed neuronal network (PINN) for finding approximate solutions to the shallow water equations.

2.2 Fully Connected Neural Network

The FCNN is the most commonly applied ML model and often includes more than one hidden layer. Every hidden layer receives the signals from the previous layer, performs basic computations defined at each neuron, and passes the results to the next layer (Haykin, 2009). Figure 2 shows a diagram of a FCNN. Mathematically, the basic function of the output for the j^{th} hidden layer $\mathbf{y}_j$ is:

$$\mathbf{y}_j = \varphi\left(\mathbf{W}_j \mathbf{y}_{j-1} + \mathbf{b}_j\right) \tag{6}$$

where $\mathbf{W}$ is the matrix of weights, $\mathbf{b}$ is the vector of biases and $\varphi()$ is the activation function.

In the proposed method, solutions for each output variable $\eta(\mathbf{x}) = h(\mathbf{x}) + z$, $hu(\mathbf{x})$, $hv(\mathbf{x})$ are approximated by 3 separate FCNNs with the same structure, as illustrated in Figure 2. Every FCNN receives the same raw inputs. As a result, the trainable parameters of the solution for each output variable are decoupled. This can significantly improve the prediction accuracy in multivariate problems, especially when the distributions and magnitudes of the variables are significantly different (e.g., Sun et al., 2020; Gao et al., 2021; Guo et al., 2020, and others).

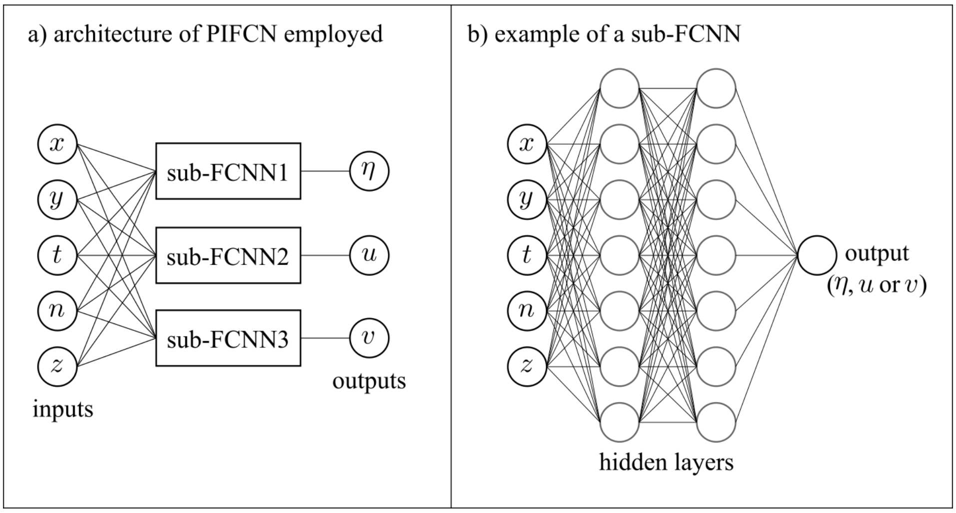

Figure 2. (a) The architecture of the physics-informed fully connected networks (PIFCNs) employed in this paper. (b) An example of a typical fully connected neuronal network (FCNN) which is employed as a sub-network within the PIFCN to predict each individual output; as illustration, 2 hidden layers with 7 neurons each are shown, but these hyperparameters are varied in this study.

2.3 Convolutional Neural Network

The CNN adds one or more convolutional layers that extract features of the raw training dataset before feeding this onto the typical hidden layers used to build FCNNs. The general expression for the convolution operator $\star$ with 1 stride is:

$$(\mathbf{s} \star \mathbf{k})_i = \sum_{j=1}^{n} k_j s_{i+j-1} \qquad i = 1, 2, \ldots, m - n + 1 \tag{7}$$

where $\mathbf{s}$ denotes the input signal vector of length m (in this paper, this is $\mathbf{x}$), and $\mathbf{k}$ denotes the trainable filter of length n. The convolution operation is to slide the preset convolutional filter over the signal input and output the signal with a shorter length (i.e., the input vector is shortened by $n-1$ elements). The shorter length of the convolved output signal allows the following typical hidden layer to have fewer neurons, facilitating the network's learning of large-scale problems with high complexity (Gao et al., 2021).

Figure 3 shows the structure of the CNN used in this paper. The trainset is generated from a number of points (i.e., collocation points) randomly sampled from a grid of equally spaced points. Each output variable, $h(\mathbf{x})$, $hu(\mathbf{x})$, $hv(\mathbf{x})$, is also predicted by a separate sub-CNN.

2.4 PINN design

The accuracy and computational performance of the PINNs described in the previous sections will be assessed and compared against the corresponding performance and solutions by a conventional FV model. There is currently no universal design approach to determine the optimal, or even appropriate, structure for a neural network (Bihlo & Popovych, 2022). The general selection rule for PINN design is to find a structure with the lowest possible complexity that achieves the desired accuracy of prediction. This rule can usually help provide an AI model with quick learning speed and improved predic-

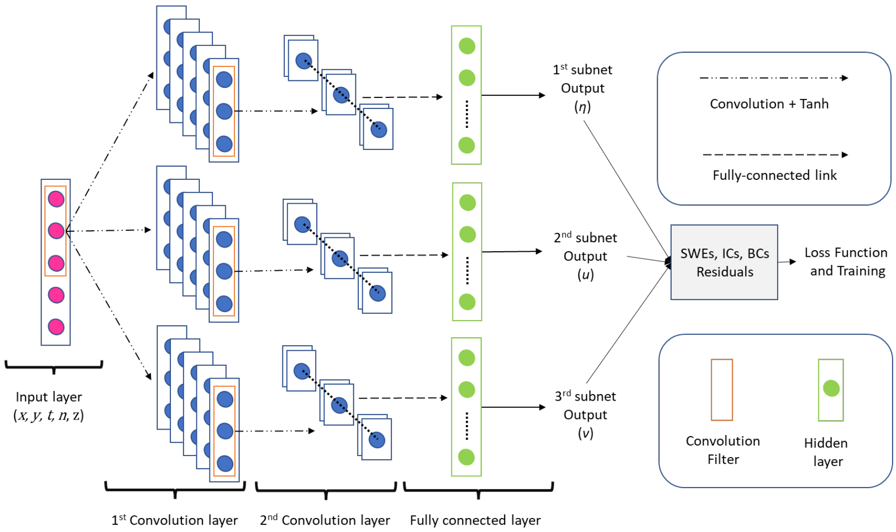

Figure 3. An example of the structure of a CNN-based model with 3 subnets for solving free-surface flow problems. Each output variable (η,u or v) is approximated by a separate CNN with the above structure; all sub-networks receive the same inputs. Each CNN has two convolutional layers and one hidden layer. The hyperparameters shown in the figure are discussed in Section 2.4.

tion capabilities while avoiding overfitting issues (Blumer et al., 1987). In this paper, the final decision for the model structure (i.e. hyperparameters such as the number of neurons, hidden layers, and convolutional layers and channels in the case of CNN) was made after many practical attempts (see Appendix A). As the evaluation of the PINNs performance in this paper consists of two, often competing, criteria (accuracy and computational cost), it may be difficult to find a single assessment metric to guide the PINNs design. Hence, we give priority to accuracy by gradually increasing the complexity of the PINNs until similar or higher accuracy than benchmark results (e.g. from an analytical solution or a finely resolved FV simulation) is attained. Generally, in our design iterations, the number of hidden layers and the corresponding neurons for building PINNs (i.e., PIFCN and PICN) started from 1 and 50, respectively. The number of convolutional layers and corresponding channels started from 1 and 5, respectively. For both PICN and PIFCN we use the hyperbolic tangent activation function (Tanh). Note that the PINN design may change significantly depending on domain and flow conditions; i.e., it can be very problem-specific. It is also important to recognize that the networks chosen do not represent the strictly optimal structure, but only the best out of the subset of structures that were tested.

For improving the learning speed and reducing the effect of parameter initialization, the Batch Normalization method of Ioffe and Szegedy (2015) was used, which normalizes the signals between adjacent convolutional or hidden layers. The Adam optimizer (Kingma & Ba, 2014), along with the '1-cycle' (Smith & Topin, 2019) strategy was used to control the training of the PINNs. The PINNs were implemented on the Pytorch platform Paszke et al. (2017). The FV simulation and the training of the PIFCNs and PICNs were performed using the University of Southampton's supercomputer Iridis 5 ensuring that the exact same hardware resources were employed (thus ensuring a fair comparison across all simulations performed).

3 Case studies

This section describes three case studies used to test the PINNs, comparing their results against analytical and numerical (Finite Volume) solutions. The first and second tests are idealized 1D (unsteady and steady, respectively) flow problems for which analytical solutions are available. However, simulations were performed on a 2D domain since the ultimate aim is to employ the PINNs developed here in 2D flow problems. The third test case is an unsteady two-dimensional simulation of a real-world flood event that took place in the Tiber river, Italy. This case study has been previously employed to evaluate the performance of other numerical models (e.g., Morales-Hernández et al., 2016; Shamkhalchian & de Almeida, 2021, and others).

Topographic data used in all tests are defined by square grids with different resolutions. The grid points are used to generate a triangular mesh for the FV model. These are also employed, along with defined temporal steps, as the collocation points for the PINNs training. The accuracy of the solutions will be assessed by the root mean square error, $\mathcal{R}$, of the outputs of each model relative to the benchmark solution. For example, in the evaluation of accuracy for the prediction of h with N_p output points, the performance metric is defined as $\mathcal{R}_h = \sqrt{\sum(h_i - \tilde{h}_i)^2/N_p}$, where h_i is the benchmark solution (i.e., the analytical solution when available, or the solution of the FV model at fine resolution). The second performance metric we employ is the computational cost, $\mathcal{T}_c$, which represents training time for the PINNs (PICN and PIFCN), and run time for the FV model. In the results presented in the following sections, predictions by the FV, PIFCN and PICN models are labelled with the different resolutions used. For example, FV (10) represents a 10 m resolved simulation using the FV hydraulic model, and PICN (50) refers to the prediction of the PICN trained from a 50 m resolved dataset.

3.1 Flood wave propagation over a horizontal plane

The first test case is a one-dimensional simulation of an inundation wave propagating over a horizontal bed. A time-dependent BC is imposed at $x = 0$. Under the idealized assumption of a flow velocity that is constant in space and time, the problem admits an analytical solution which can be expressed as (Hunter et al., 2005):

$$h_a(x,t) = \left\{ \frac{7}{3} \left(n^2 u^2 (x - ut) \right) \right\}^{3/7}, \tag{8}$$

where the subscript a is used to denote the analytical solution. The domain used is a 100 m wide, 1200 m long channel. The constant velocity is set as $u(x,t) = 0.29$ ms^{-1} and the boundary condition $h(x = 0, t)$ is given by Eq. 8. The domain is initially dry, i.e., $h(x, t = 0) = 0$. Manning's coefficient n is set to 0.03 sm$^{-1/3}$. The duration of the simulation is 3600 s. The FV model was run at resolutions of 1, 2, 5 and 10 m, while the PINN models were trained with datasets defined at resolutions of 10, 25, 50 and 100 m. While the time step of the explicit FV scheme is controlled by the Courant-Friedrichs-Lewy (CFL) stability condition, the regression approximation implemented by the PINN model is not limited by temporal resolution. However, the time step adopted to train the PINN model is a factor that clearly affects both accuracy and computational performance. For this test, we use a temporal resolution for the PINN of 300 s. The selected batch size is set as the full set of collocation points $n_x \times n_y \times n_t$.

The architecture of the PICN consists of 2 convolutional layers (the first and second layers have 5 and 20 channels, respectively) and 1 fully connected hidden layer with 50 neurons. The architecture for the PIFCN consists of 3 fully connected hidden layers, each of which has 1000 neurons.

Figures 4 and 5 illustrate the values of $h(x, y = 50\text{m})$ and $hu(x, y = 50\text{m})$ (left vertical axes), and the corresponding error (right vertical axes) $\epsilon_h(x, y = 50\text{m}) = \tilde{h}(x, y =$

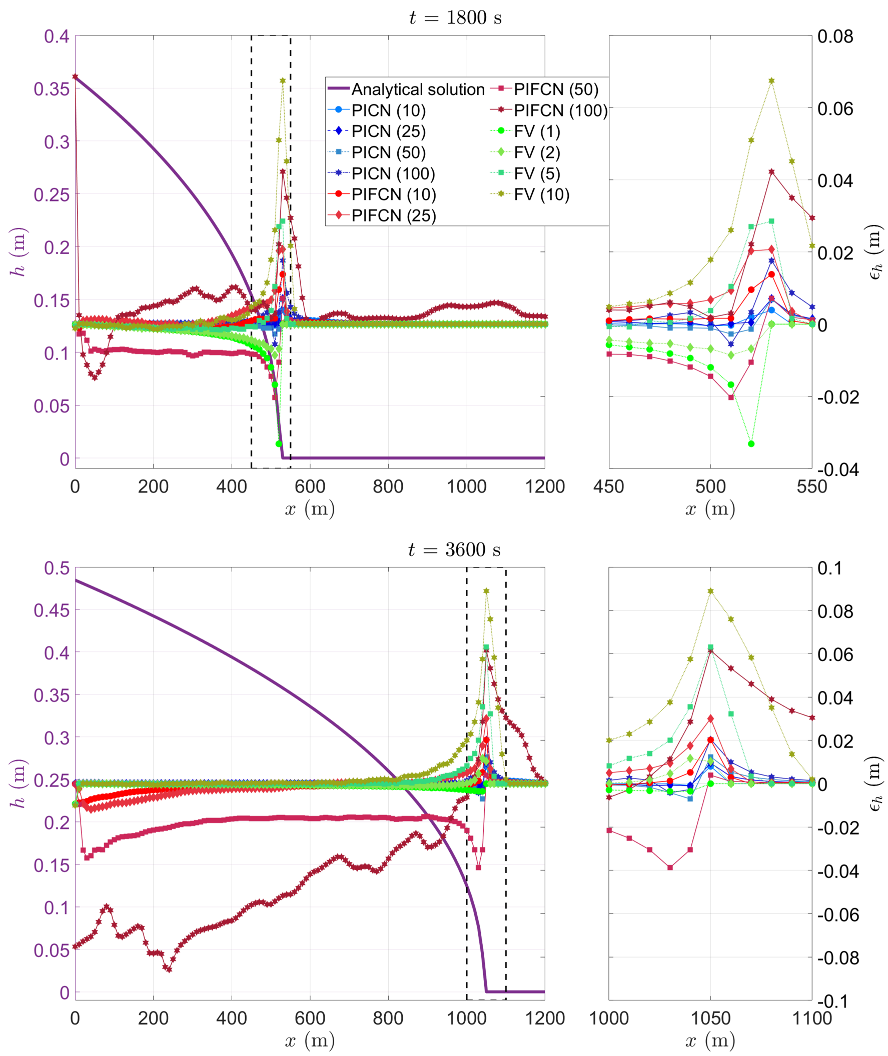

Figure 4. Test 1: Longitudinal profiles ($y = 50$ m) of water depth errors ϵ_h relative to the analytical solution obtained by each of the models at $t = 1800$ s (top) and $t = 3600$ s (bottom), shown against right $y-$axis. The analytical solution for h (purple line) is plotted against the left $y-$axis; note that the right-side figure is the enlarged version of the rectangular box in the left-side figure; both figures share the same right $y-$axis.

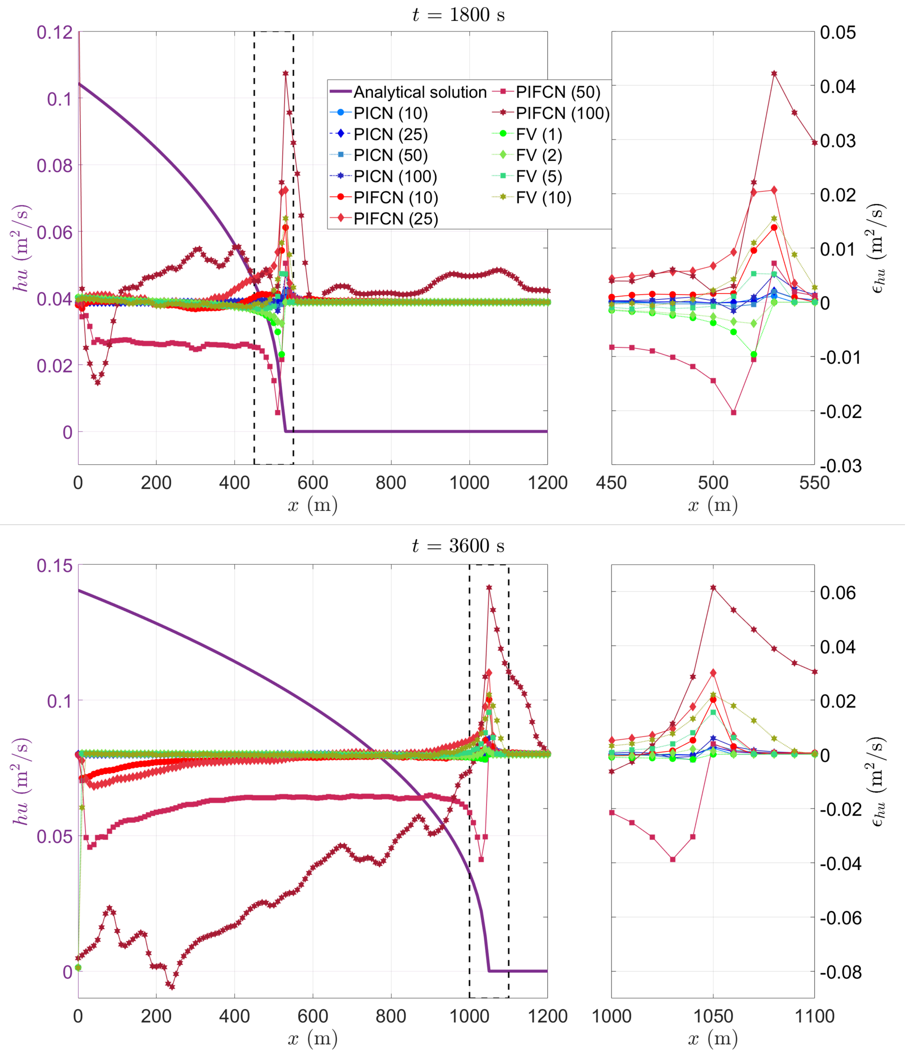

Figure 5. Test 1: Longitudinal profiles (y = 50 m) of water depth errors ϵ_{hu} relative to the analytical solution obtained by each of the models at t = 1800 s (top) and t = 3600 s (bottom). The analytical solution for hu (purple line) is plotted against the left $y-$axis; note that the right-side figure is the enlarged version of the rectangular box in the left-side figure; both figures share the same right $y-$axis.

50m) $- h_a(x, y = 50\text{m})$ and $\epsilon_{hu}(x, y = 50\text{m}) = \tilde{hu}(x, y = 50\text{m}) - (hu)_a(x, y = 50\text{m})$ computed by all three models at $t = 1800$ and 3600 s, respectively. Values of hv are not reported as the test case is fundamentally one-dimensional. Overall, all the water depth predictions, with the exception of PIFCN (100), show good agreement with the analytical solution (i.e. most results displaying $|\epsilon| < 0.01$m). As the position of the wet-dry front predicted by the models does not exactly match the analytical solution, and the front is steep at that point, errors are larger in this region. PICN and FV both show similar prediction accuracy of both h and hu, whereas PIFCNs with coarsely resolved trainsets (i.e., 50 m and 100 m) provide higher prediction errors of hu.

Figure 6 shows $\mathcal{R}_h$ (relative to the analytical solution h_a) for all results obtained with the PICN, PIFCN, and FV models as a function of the corresponding computational time $\mathcal{T}_c$. The sum to compute $\mathcal{R}_h$ is over all collocation points; i.e., spanning the whole spatio-temporal domain. In this figure, the various points (blue and red) presented for each PINN model represent solutions obtained at different epochs during the training of the networks, which correspond to different computation time and level of accuracy. The green cross points represent the simulation accuracy and computation time for the FV model. The results in this figure are based on model (i.e. PICN, PIFCN, and FV) outputs at the same grid points selected from the entire domain with a spatial and temporal resolution of 10 m and 360 s. Predictions of hu follow the general pattern observed for h on Figure 6 and are not reported here to avoid repetition. Figure 6 allows us to comparatively assess the performance of the models tested in terms of their speed-accuracy trade-off. Based on this criterion, a model performs better than another when it provides more accurate results under the same computational time, or vice-versa; in other words, the best results are those closest to the bottom left corner of the plot.

Figure 6 shows that FV (10) and FV (5) produce sub-centimetre $\mathcal{R}_h$ (which is usually considered a good level of accuracy for many applications) at least one order of magnitude faster than the PINN models, whereas FV (2) takes slightly longer than PICNs (for the same level of accuracy), and FV (1) only outperforms PIFCN (10) in terms of the speed-accuracy trade-off. All PINNs except PIFCN (100) show the potential to achieve better accuracy of prediction than the FV model at the highest resolution tested here (1 m), provided they are trained for long enough. PICNs provide a faster solution (for similar $\mathcal{R}_h$ values) than PIFCNs. Also, for PIFCN, the trainset size (which in this case is determined by the resolution) did not significantly affect its maximum accuracy at resolutions ≤ 50 m, whereas the accuracy of the FV model continues to improve as the mesh is refined below 10 m.

3.2 Subcritical steady flow over an undulating bed

The second test case represents a 1D, steady, non-uniform flow over an undulating bed, for which an analytical solution is available (see MacDonald, 1996; de Almeida & Bates, 2013; Delestre et al., 2013). This test case will be used to evaluate the solution obtained by the PICN and PIFCN in a problem with variable topography. The (rectangular) channel is 1000 m long, and Manning's coefficient n is set to 0.03 sm$^{-1/3}$. The constant inflow discharge per unit width of the channel is $q_x = uh = 2$ m^{2}s^{-1}, and the downstream water depth is $\frac{9}{8}$ m. We prescribe the following function representing the water depth $h(x)$ (which is the benchmark solution against which the PINN approximations will be compared):

$$h(x) = \frac{9}{8} + \frac{1}{4}\sin\left(\frac{\pi x}{500}\right). \tag{9}$$

We model this 1D problem in a 2D domain using a width of 50 m (and $q_y = 0$) for the reasons discussed previously. Also, although the solution sought is for a steady flow problem, the steady condition was reached via an unsteady flow simulation, as the

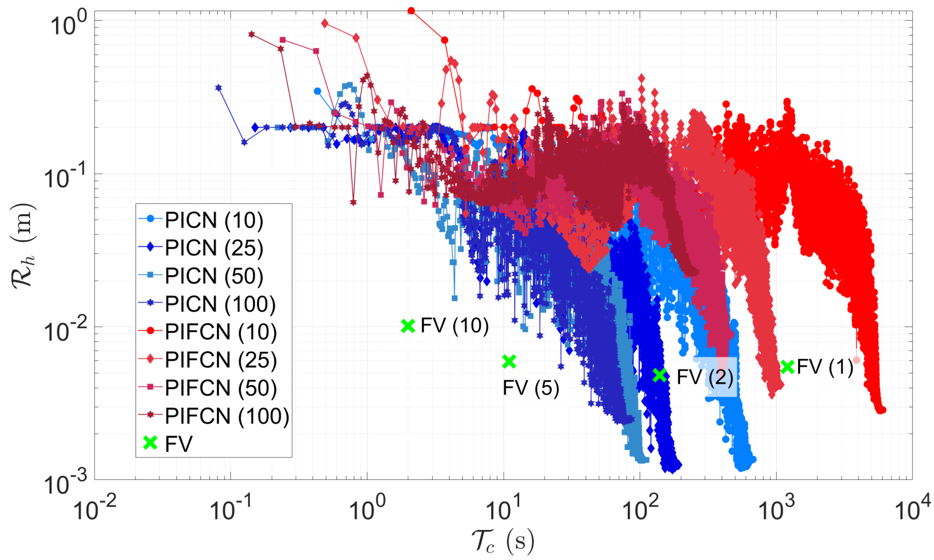

Figure 6. Test 1: Values of $\mathcal{R}_h$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV); note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model).

object of this paper is to test approximate methods to solve the time-dependent SWEs. The unsteady simulations were run from an initially dry domain over a period of 20 hours, whereby the upstream BCs increase linearly with time from zero to the aforementioned constant values over the first 10 hours of the simulation.

The training dataset for PICN and PIFCN was obtained from grids resolved at 5, 10, 25 and 50 m at the following times: 0, 1, 3, 5, 10, 15 and 20 hours. The selected batch size is $2/7 \times N$, where the value 2/7 comes from trial and error (larger batch sizes decreased the accuracy of the results). The FV model was run at resolutions of 2, 5 and 10 m.

For this case, the architecture of the PICN consists of 2 convolutional layers (the first and second layers have 5 and 20 channels, respectively) and 1 fully connected hidden layer with 50 neurons (same as in Test 1). The architecture of the PIFCN consists of 3 fully connected hidden layers, each of which has 1000 neurons (different from Test 1).

Figure 7 shows the analytical curve for depth profile at the centre of the channel $h(x, y = 30 \text{ m })$ (left axis) and the corresponding errors of each of the approximate solutions ϵ_h (right axis) predicted by the PICN (blue points), PIFCN (red points), and FV models (green points). Figure 8 presents similar results but for the variable hu. As the analytical solution is for the steady state, only the results at the end of the simulations are assessed. Overall, all models tested delivered results at sub-centimeter level of accuracy for h. The three PICNs showed the lowest errors of both h and hu, followed by FV (2).Values of ϵ_{hu} obtained from FV models display small (mostly within 1% of the actual value of hu) spatial variations, while they nearly are constant for both PIFCN and PICN.

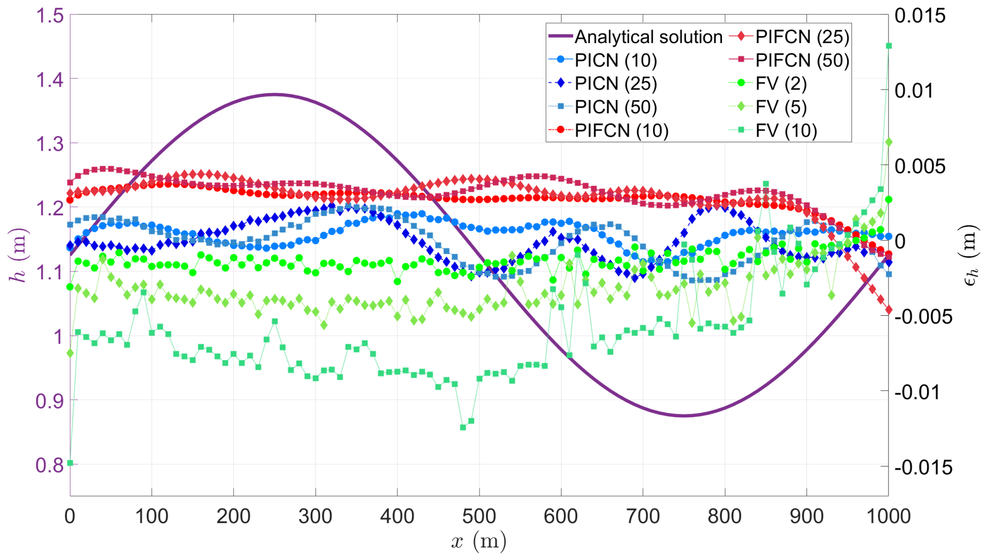

Figure 7. Test 2: Longitudinal profiles (y = 30 m) of water depth errors obtained by each of the models at the end of the simulation/training (right $y-$axis). The analytical solution h (purple line) is plotted against the left $y-$axis.

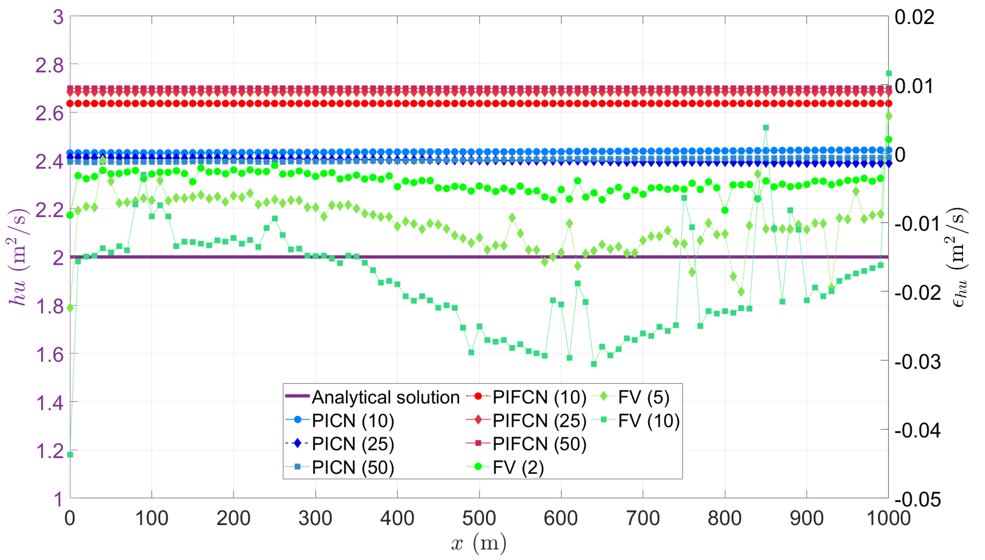

Figure 8. Test 2: Longitudinal profiles (y = 30 m) of water depth errors obtained by each of the models at the end of the simulation/training (right $y-$axis). The analytical solution hu (purple line) is plotted against the left $y-$axis.

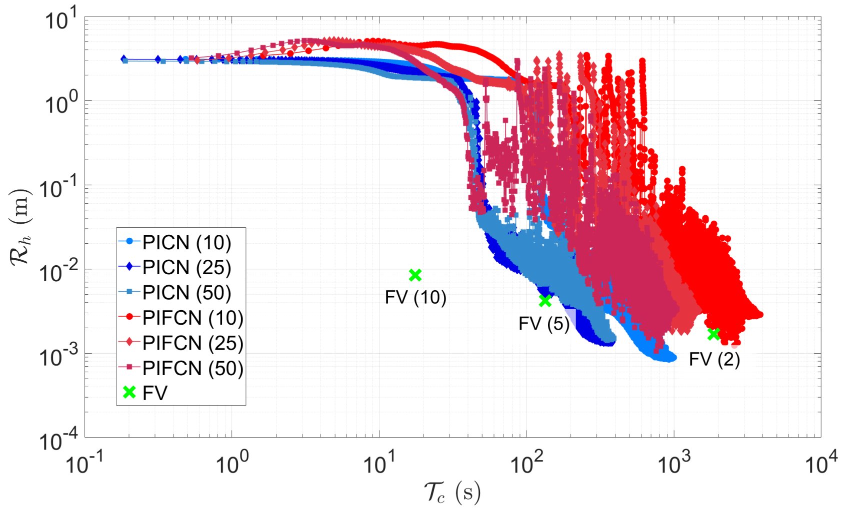

Figure 9. Test 2: Values of $\mathcal{R}_h$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV); note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model).

Figure 9 presents the values of $\mathcal{R}_h$ against the corresponding computational time taken to train the PICN (blue points), PIFCN (red points), and to run the FV model (green cross points) at different resolutions. The value of $\mathcal{R}_h$ of each model is calculated from its steady-state predictions of h; namely: $\mathcal{R}_h = \left.\left(\sqrt{\sum(h_i - \tilde{h}_i)^2/N_p}\right)\right|_{t=t_s}$, where t_s is the time after which a steady state is reached for each PINN or FV model. For the computation time of FV models described in Figure 9, the value of $\mathcal{T}_c$ is the time required for all FV models to reach steady state. The results for hu show a pattern similar to that in Figure 9 and are not presented for conciseness. All simulations achieve sub-centimetric $\mathcal{R}_h$, with FV (10) delivering the results at least one order of magnitude faster than the other solutions. PIFCN (10) was the slowest of all models. Figure 9 shows that the prediction of h from PICN (10) displays the highest accuracy, with an $\mathcal{R}_h$ of 0.85 mm, although this was obtained at a computation time that was 56 times longer than FV (10). All the PINN results also attain an accuracy higher than or similar to that of FV(2). In this test case, the relative differences in the prediction accuracy among the PICN models is less than the difference observed from FV (5) to FV (10). In terms of the influence of resolution on the computational speed, the PICN is also less sensitive than PIFCN in this problem.

3.3 Simulation of real-world river flooding

While Tests 1 and 2 have assessed the ability of PINN models to deal with important aspects of flow problems, such as unsteadiness and variable topography, both case studies represented idealized, one-dimensional problems. In order to investigate the performance of PINNs under more complex and realistic problems, this section presents the results of simulations of a real-world scenario. The scenario in question is a flood event that occurred between 27 November and 1 December 2005 in the Tiber river (Morales-Hernández et al., 2016), which flows from the Apennine Mountains to the Tyrrhenian Sea in Italy. The reach of river employed in this simulation is approximately 6 km long and is located near the city of Rome. In this region, the mean discharge of the Tiber river is 267 m^3s^{-1}, while its peak discharge for a 200-year return period is around 3200 m^3s^{-1}. The event modeled in this paper was also previously simulated in Morales-Hernández et al. (2016) and Shamkhalchian and de Almeida (2021). The domain comprises an area of 6 km $\times$ 2 km. The duration of the event simulated is 113 hours. The values of Manning's coefficient n used are the same as in Morales-Hernández et al. (2016) and Shamkhalchian and de Almeida (2021); namely, $n = 0.035$ $sm^{-1/3}$ for the main channel, and $n = 0.0446$ $sm^{-1/3}$ for the floodplains.

The boundary conditions were obtained from Morales-Hernández et al. (2016), and correspond to the time series of flow discharge and water surface elevation at the upstream and downstream sections of the river at the boundary of the computational domain. The initial conditions $\mathbf{U}(x, y, t = 0)$ were defined from the results of the FV model under steady-state conditions ($Q = 374$ m^3s^{-1}) performed at 5 m resolution. PINNs were trained from datasets resolved at 50, 100 and 200 m, while the FV model was run using meshes generated from gridded data at resolutions of 10, 25 and 50 m. The corresponding temporal resolution for the trainset for the PINNs is 4 hours. The batch size was set to one third of the total number of collocation points.

For this test case, the architecture of the PICN consists of 2 convolutional layers (the first and second layers have 10 and 40 channels, respectively) and 1 fully connected hidden layer with 100 neurons. The architecture of the PIFCN consists of 3 fully connected hidden layers, each having 2000 neurons. Our tests showed that further increasing the network complexity would not improve the model's prediction accuracy, and may substantially increase the training time and/or cause the program to exceed the memory capacity of the computer resources used.

Since an analytical solution is not available for this problem, the results of the FV simulation at fine resolution (5 m) were used as the benchmark. The accuracy of the solutions of the time-dependent variables is assessed at two cross-sections (located approximately at distances of 1/3 and 2/3 of the length of the river within the domain from the upstream boundary, and hereafter referred to as S1 and S2, respectively) at 1 hour temporal resolution.

Figures 10 and 11 illustrate the time series of prediction errors (right vertical axes), along with the actual predicted values of the flow depth h and flow discharge Q (left vertical axes) at cross-sections S1 and S2 for each PICN, PIFCN, and FV models. Figure 10 shows that the FV and PIFCN simulations consistently predict larger and lower depths than the benchmark solution, respectively, at both cross sections in the main channel, while PICN results display both positive and negative values of ϵ_h. Results from PICNs at S1 and S2 are markedly more accurate than those delivered by PIFCNs and the coarse-resolution FV models. For example, FV (50) and FV (25) produced results that deviate substantially (i.e. up to approximately 1.2 m and 2.5 m at S1 and S2, respectively) from the benchmark solution. On the other hand, FV (10) generally produced the most accurate depth predictions out of all models tested. The ability of the models to predict flow velocities (and therefore, the volumetric flow rate Q) is assessed by $\epsilon_Q = \tilde{Q} - |Q|$, where $Q = \int h\tilde{\mathbf{U}} \cdot \mathbf{n} dl$ is the total discharge; l is the length along the cross-sections (i.e., S1 and S2, which span across the whole domain) and $\mathbf{n}$ is the unit vector normal to the cross-section. Figure 11 shows the predicted errors ϵ_Q obtained by all models as a function of time. These results are markedly different from those previously presented for ϵ_h. Namely, all FV models display values of ϵ_Q that are substantially smaller than those predicted by PICN and PIFCN models. The maximum values of ϵ_Q for PICN and PIFCN are more than 50% and 70% of the benchmark (FV (5)) in S2, respectively. The possible reason behind these results might be that the water surface ($\eta = h + z$) presents much less spatial variation than Q in the domain. However, this hypothesis would need to be tested thoroughly in the future through a set of specifically designed case studies.

Figure 12 assesses the overall accuracy of temporal prediction for h of each model against the corresponding computational time, using the root-mean-square error metric $\mathcal{R}_h^t = \left.\left(\sqrt{\sum (h_i - \tilde{h}_i)^2 / N_p^t}\right)\right|_{(x,y)\in\mathcal{S}}$, where $\mathcal{S}$ denotes the corresponding cross-section and N_p^t is the number of collocation points in the testset. The best values of $\mathcal{R}_h^t$ (i.e., across all epochs) obtained from all PICN models are within the range of 0.22 m $< \mathcal{R}_h^t <$ 0.30 m (S1) and 0.26 m $< \mathcal{R}_h^t <$ 0.34 m (S2), while FV (10) delivered $\mathcal{R}_h^t = 0.29$ m (S1) and 0.35 m (S2), and results from FV (25) and FV (50) were substantially less accurate. It is interesting to note that PINN models trained with coarse datasets (e.g., 200 m) do not necessarily deliver poorer accuracy compared to their fine resolution counterparts; this contrasts with what is typically observed in simulations with traditional numerical methods such as FV. Figure 12 also indicates that PICN models may offer improved depth predictions at lower cost than a FV model. For example, the accuracy of depth predictions by PICN (200) is better than the accuracy delivered by FV (10), while the computational cost is more than one order of magnitude lower. Overall, the PICN shows better h prediction performance than PIFCN and FV in terms of the speed-accuracy trade-off.

Figure 13 shows examples of flood depth maps at $t = 32$ hours obtained by the FV model at resolutions of 5 m and 25 m, along with those produced by PICN and PIFCN at 100 m resolved trainsets. As expected from the results presented in Figure 10, FV (25) overestimates h during the peak time (which also translates into a larger flooded area), while the opposite is observed for PICN (100) and PIFCN (100). Further spatial analysis can be seen in Appendix B.

4 Concluding remarks

In this paper, two physics-informed neuronal networks (PINNs) were developed to predict the evolution of free-surface flows typically modeled by the shallow water equations (SWEs). The PINN formulation eliminates the need for labeled data, which is typically required in supervised learning. This is achieved by defining a loss function that combines the SWEs, the boundary conditions (BCs) and initial conditions (ICs), allowing the trained PINN to serve as an alternative method for solving the SWEs. The two PINNs developed and tested here vary in their architecture and main features. The first is based on the fully-connected neural network (PIFCN), and the second on the convolutional neural network (PICN) approach.

Three test cases were used to assess the accuracy and computational performance of each model, including two idealized flow problems for which analytical solutions are available, and one simulation of a real-world flood event over a relatively large-scale and complex topography domain. In the idealized problems, the PICN and PIFCN predictions achieved higher accuracy (lower $\mathcal{R}_h$) than the Finite Volume (FV) solver employed for comparison. However, in these problems, PINNs generally took longer to reach the same prediction accuracy as the coarsely resolved FV model. For the real-world flooding problem, in general, PINNs were able to yield similarly accurate predictions of flow depths compared to finely resolved FV simulations. However, all FV models show much higher accuracy in their predictions of Q. For the spatial analysis of flow depths at the peak of the flood event, PINNs were able to produce flood maps with accuracy (relative to the benchmark finely resolved FV simulation) that is comparable to the results of FV models run at intermediate resolution (e.g., 25 m). Some of the PINN models (e.g., PICN at 100 and 200 m resolution) achieved the same level of accuracy as the 25 m resolution FV model at least one order of magnitude faster. In addition, the prediction capability of PINNs may be less affected by changes in grid resolution than the FV solver, which may represent important advantages in real-world applications where finely resolved topographic data may not always be available. At the same resolution (e.g., 10 m in Tests 1 and 2, or 50 m in Test 3), the training process of PICNs and PIFCNs with random ini-

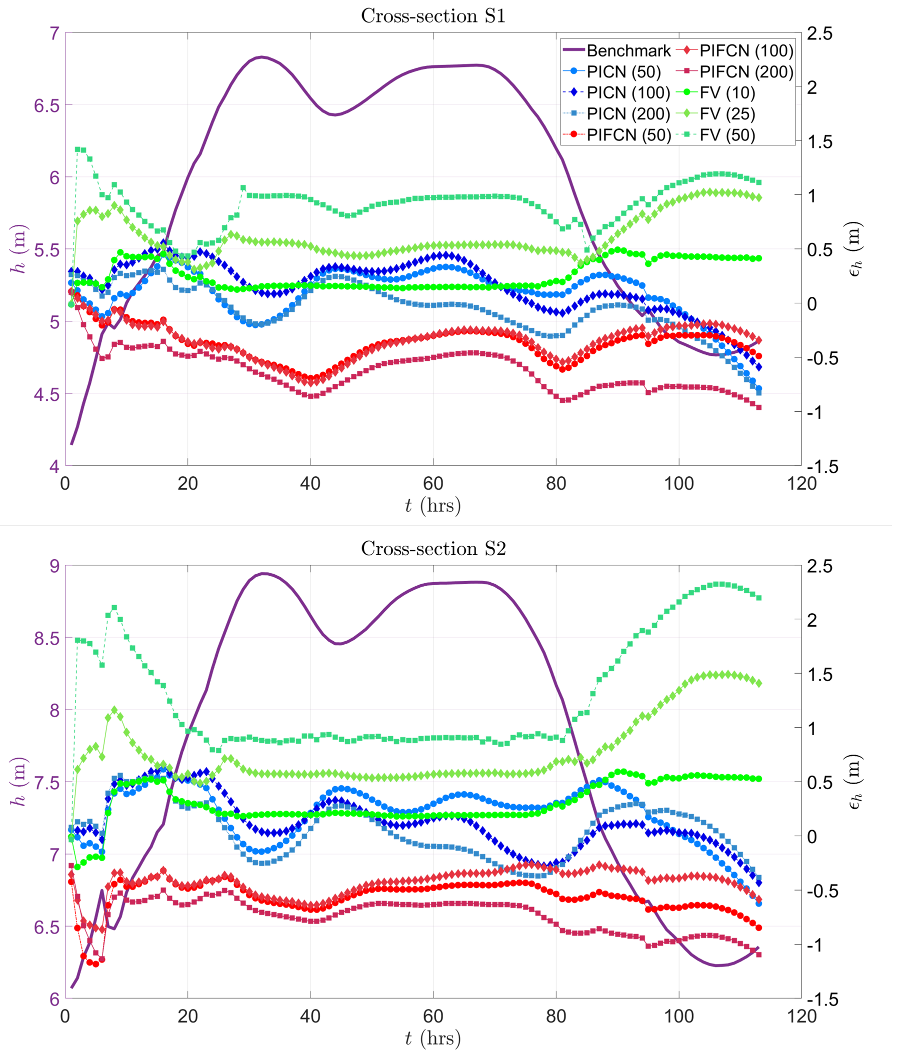

Figure 10. Test 3: Predicted water depths error ϵ_h (plotted against right $y-$axis) at cross-sections S1 (top) and S2 (bottom) of the main channel in the Tiber river. Benchmark solution (from a finely resolved FV simulation) shown by the purple line against the left $y-$axis.

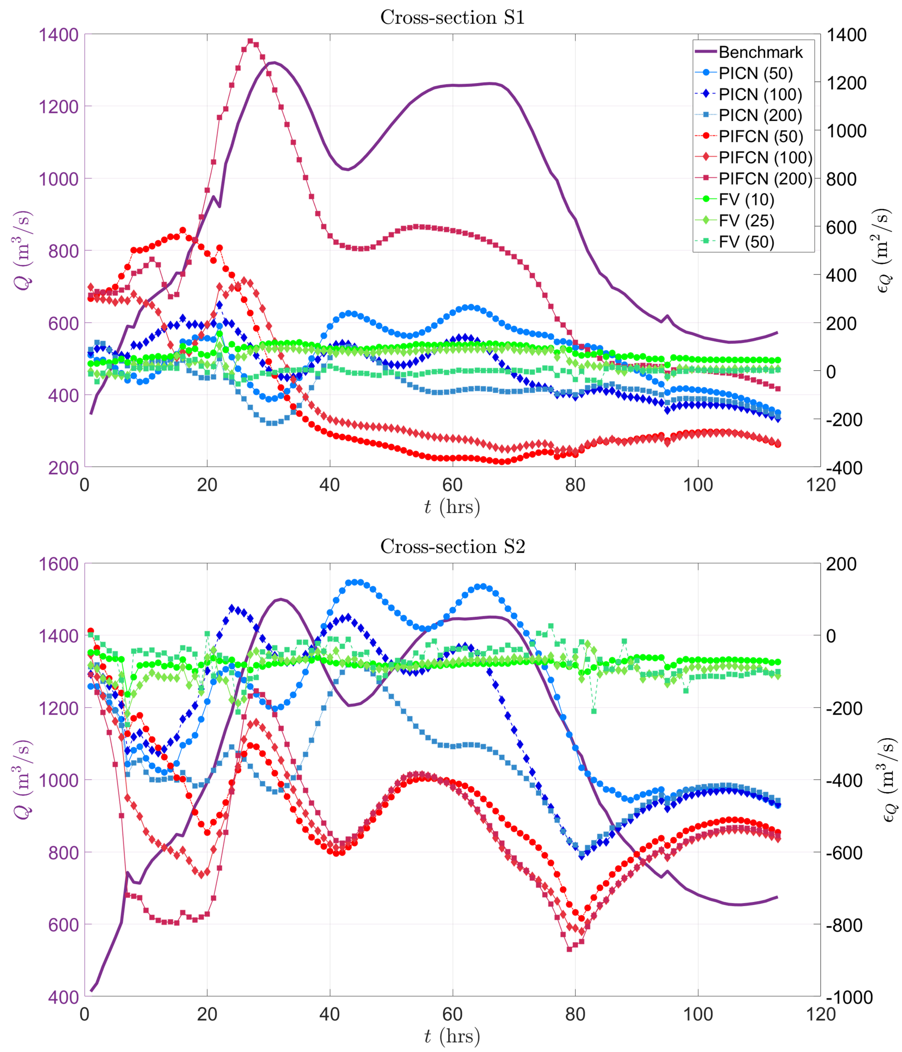

Figure 11. Test 3: Predicted water discharge error ϵ_Q (plotted against right $y-$axis) at cross-sections S1 (top) and S2 (bottom), which span across the whole domain. Benchmark solution (from a finely resolved FV simulation) shown by the purple line against the left $y-$axis.

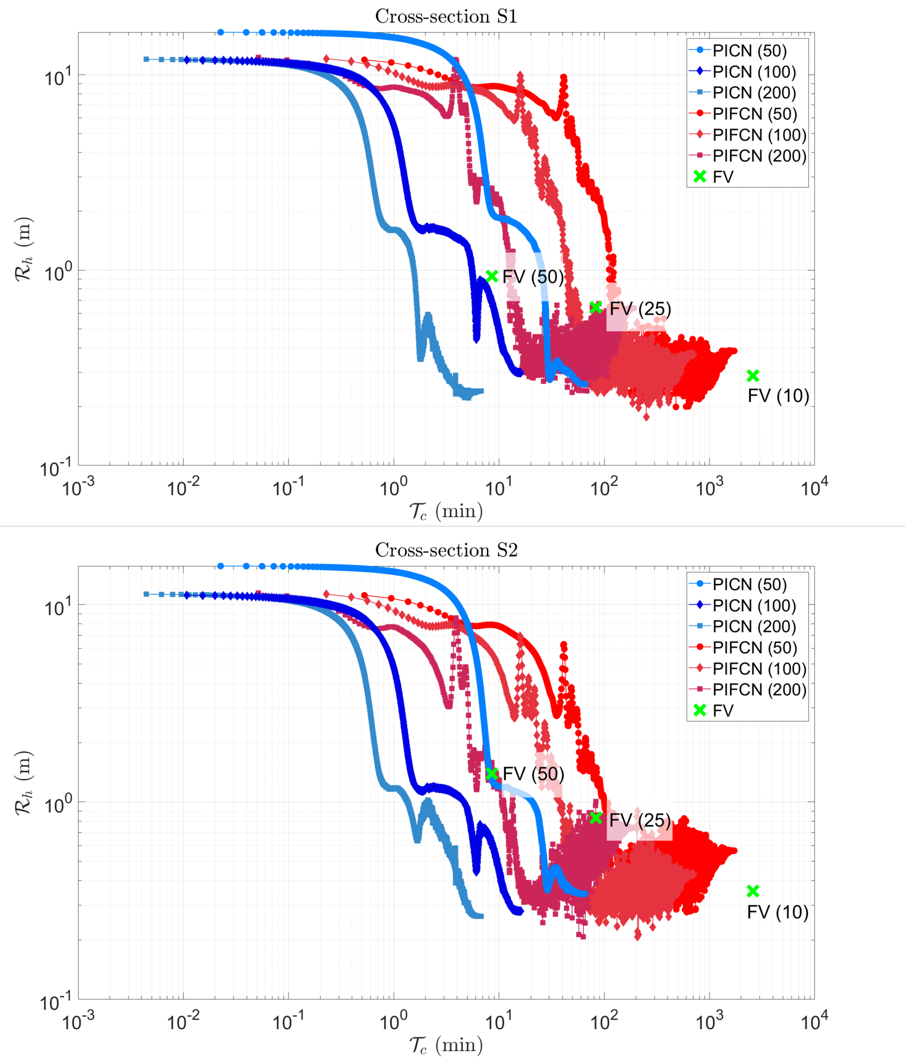

Figure 12. Test 3: $\mathcal{R}_h^t$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV) at cross-sections S1 (top) and S2 (bottom) of the Tiber river; note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model). The benchmark results are those from the FV (5) simulation.

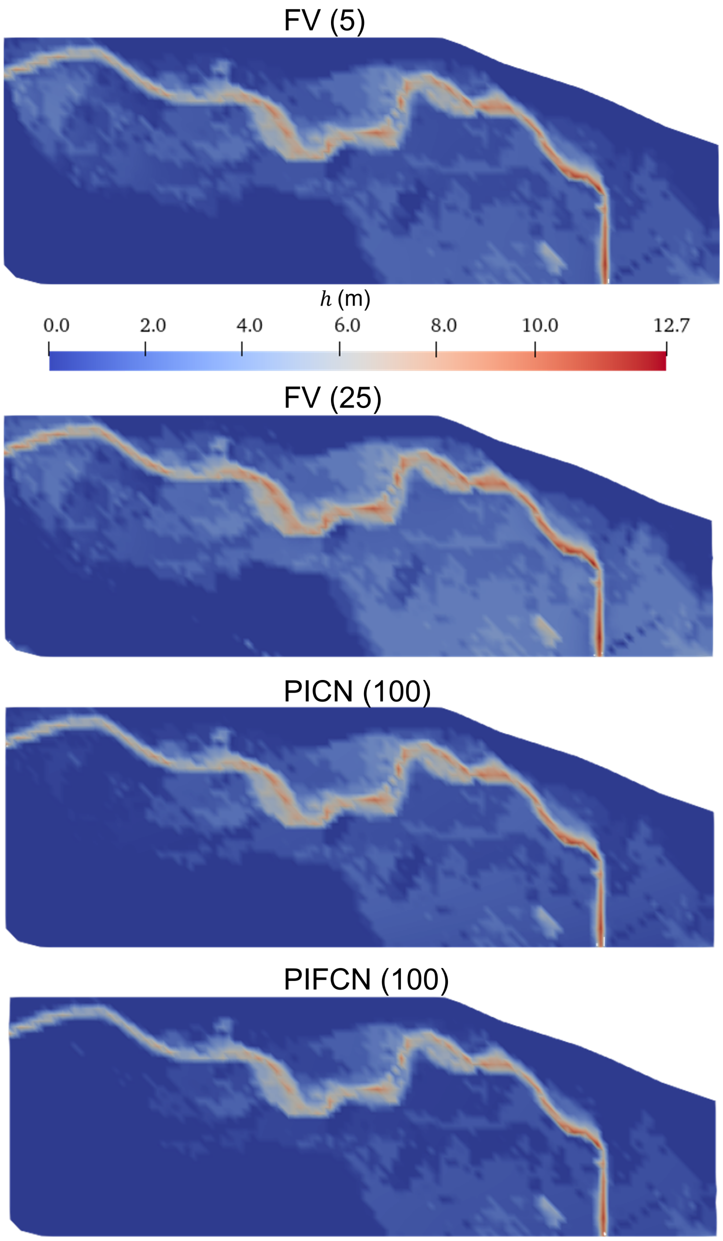

Figure 13. Examples of flood maps at time t = 32 hours produced by the FV model and PINNs. Note that FV (5) represents the benchmark results.

tialization of weights and biases takes longer than the run time of the FV model. Results show that, in most circumstances, PICNs usually exhibit better performance in terms of speed-accuracy trade-off than PIFCNs. However, more comparative tests between PICN and PIFCN are necessary before reaching general conclusions in this regard.

While the results in this paper may not suggest that PINNs can replace other well-established numerical techniques, they indicate that PINNs (and in particular PICNs) should be considered as an emerging technique that has the potential to deliver accurate and efficient solutions, and which should be further developed and assessed. Our results show that the approach might be particularly useful under certain circumstances which are challenging to conventional techniques. For example, in simulations performed at coarse resolutions (a typical case in real-world problems), PINN models may achieve a higher prediction accuracy with a lower computational cost than a FV solver. Since these techniques are still in their infancy, further research and development may enable PINNs to become a competitive alternative to simulate flow problems governed by the SWEs in the near future.

5 Open Research

The simulation data used for all three test cases in the study are available at the database from University of Southampton via `https://doi.org/10.5258/SOTON/D2645` with CC-BY license (Xin Qi, 2023).

Acknowledgments
The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

Alcrudo, F., & Garcia-Navarro, P. (1993). A high-resolution godunov-type scheme in finite volumes for the 2d shallow-water equations. *International Journal for Numerical Methods in Fluids*, *16*(6), 489–505.

Bale, D. S., Leveque, R. J., Mitran, S., & Rossmanith, J. A. (2003). A wave propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal on Scientific Computing*, *24*(3), 955–978.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, *18*, 1–43.

Bermúdez, M., Cea, L., & Puertas, J. (2019). A rapid flood inundation model for hazard mapping based on least squares support vector machine regression. *Journal of Flood Risk Management*, *12*(August 2018), 1–14. (Times cited: 6 Flooding papers) doi: 10.1111/jfr3.12522

Bernard, P.-E., Remacle, J.-F., Comblen, R., Legat, V., & Hillewaert, K. (2009). High-order discontinuous galerkin schemes on general 2d manifolds applied to the shallow water equations. *Journal of Computational Physics*, *228*(17), 6514–6535.

Bihlo, A., & Popovych, R. O. (2022). Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, *456*, 111024. doi: https://doi.org/10.1016/j.jcp.2022.111024

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*(6), 377-380. doi: https://doi.org/10.1016/0020-0190(87)90114-1

Botta, N., Klein, R., Langenberg, S., & Lützenkirchen, S. (2004). Well balanced finite volume methods for nearly hydrostatic flows. *Journal of Computational Physics*, *196*(2), 539–565.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, *143*(6).

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154.

Casulli, V. (1990). Semi-implicit finite difference methods for the two-dimensional shallow water equations. *Journal of Computational Physics*, *86*(1), 56–74.

Dawson, C., Westerink, J. J., Feyen, J. C., & Pothina, D. (2006). Continuous, discontinuous and coupled discontinuous–continuous galerkin finite element methods for the shallow water equations. *International Journal for Numerical Methods in Fluids*, *52*(1), 63–88.

de Almeida, G. A., & Bates, P. (2013). Applicability of the local inertial approximation of the shallow water equations to flood modeling. *Water Resources Research*, *49*(8), 4833–4844.

de Almeida, G. A., Bates, P., & Ozdemir, H. (2016). Modelling urban floods at submetre resolution: challenges or opportunities for flood risk management? *Journal of Flood Risk Management*, *11*, S855–S865.

Delestre, O., Lucas, C., Ksinant, P.-A., Darboux, F., Laguerre, C., Vo, T.-N.-T., ... Cordier, S. (2013). SWASHES: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, *72*(3), 269–300. doi: 10.1002/fld.3741

El Naqa, I., Li, R., & Murphy, M. J. (2015). *Machine learning in radiation oncology: theory and applications.* Springer.

Ferrari, A., & Vacondio, R. (2022). An augmented hllem ader numerical model parallel on gpu for the porous shallow water equations. *Computers & Fluids*, *238*, 105360.

Gao, H., Sun, L., & Wang, J.-X. (2021). Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, *428*, 110079.

Guo, L., Ye, S., Han, J., Zheng, H., Gao, H., Chen, D. Z., ... Wang, C. (2020). Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization. In *2020 ieee pacific visualization symposium (pacificvis)* (p. 71-80). doi: 10.1109/PacificVis48177.2020.8737

Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2020). A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751*.

Hanert, E., Le Roux, D. Y., Legat, V., & Deleersnijder, E. (2005). An efficient eulerian finite element method for the shallow water equations. *Ocean Modelling*, *10*(1-2), 115–136.

Haykin, S. (2009). *Neural networks and learning machines* (Third Edit ed.). Upper Saddle River: Prentice-Hall.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226–251.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hunter, N. M., Horritt, M. S., Bates, P. D., Wilson, M. D., & Werner, M. G. (2005). An adaptive time step solution for raster-based storage cell modelling of floodplain inundation. *Advances in water resources*, *28*(9), 975–991.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*.

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, *426*, 109951.

Juez, C., Murillo, J., & García-Navarro, P. (2014). A 2d weakly-coupled and efficient numerical model for transient shallow flow and movable bed. *Advances in Water Resources*, *71*, 93–109.

Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., & Pender, G. (2020). A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, *590*, 125481.

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, *34*.

Kurganov, A., & Levy, D. (2002). Central-upwind schemes for the saint-venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, *36*(3), 397–425.

Leandro, J., Chen, A., & Schumann, A. (2014). A 2d parallel diffusive wave model for floodplain inundation with variable time step (p-dwave). *Journal of Hydrology*, *517*, 250–259.

Leskens, J., Brugnach, M., Hoekstra, A. Y., & Schuurmans, W. (2014). Why are decisions in flood disaster management so poorly supported by information from flood models? *Environmental modelling & software*, *53*, 53–61.

LeVeque, R. J., George, D. L., & Berger, M. J. (2011). Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, *20*, 211–289.

Li, C., Han, Z., Li, Y., Li, M., Wang, W., Dou, J., . . . Chen, G. (2023). Physical information-fused deep learning model ensembled with a subregion-specific sampling method for predicting flood dynamics. *Journal of Hydrology*, *620*, 129465. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0022169423004079` doi: https://doi.org/10.1016/j.jhydrol.2023.129465

Li, J., Cao, Z., & Borthwick, A. G. (2022). Quantifying multiple uncertainties in modelling shallow water-sediment flows: A stochastic galerkin framework with haar wavelet expansion and an operator-splitting approach. *Applied Mathematical Modelling*, *106*, 259-275. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0307904X22000579` doi: https://doi.org/10.1016/j.apm.2022.01.032

Li, R., Lee, E., & Luo, T. (2021). Physics-informed neural networks for solving multiscale mode-resolved phonon boltzmann transport equation. *Materials Today Physics*, *19*, 100429.

Liang, Q. (2011). A structured but non-uniform cartesian grid-based model for the shallow water equations. *International Journal for Numerical Methods in Fluids*, *66*(5), 537–554.

Liang, Q., & Marche, F. (2009). Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in water resources*, *32*(6), 873–884.

Liu, Y., & Pender, G. (2015). A flood inundation modelling using v-support vector machine regression model. *Engineering Applications of Artificial Intelligence*, *46*, 223–231. (Times cited: 4 Flooding papers) doi: 10.1016/j.engappai.2015.09.014

Lynch, D. R., & Gray, W. G. (1979). A wave equation model for finite element tidal computations. *Computers & fluids*, *7*(3), 207–228.

MacDonald, I. (1996). *Analysis and computation of steady open channel flow* (Unpublished doctoral dissertation). Citeseer.

Mahesh, R. B., Leandro, J., & Lin, Q. (2022). Physics informed neural network for spatial-temporal flood forecasting. In *Climate change and water security* (pp. 77–91). Springer.

Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, *360*, 112789.

Marras, S., Kelly, J. F., Moragues, M., Müller, A., Kopera, M. A., Vázquez, M., ... Jorba, O. (2016). A review of element-based galerkin methods for numerical weather prediction: Finite elements, spectral elements, and discontinuous galerkin. *Archives of Computational Methods in Engineering*, *23*(4), 673–722.

Molls, T., & Chaudhry, M. H. (1995). Depth-averaged open-channel flow model. *Journal of Hydraulic Engineering*, *121*(6), 453–465.

Monnier, J., Couderc, F., Dartus, D., Larnier, K., Madec, R., & Vila, J.-P. (2016). 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.

Morales-Hernández, M., Petaccia, G., Brufau, P., & García-Navarro, P. (2016). Conservative 1d–2d coupled numerical strategies applied to river flooding: The tiber (rome). *Applied Mathematical Modelling*, *40*(3), 2087–2105.

Pang, G., Lu, L., & Karniadakis, G. E. (2019). fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, *41*(4), A2603–A2626.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.

Rastgoo, R., Kiani, K., Escalera, S., & Sabokrou, M. (2021). Sign language production: A review. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3451–3461).

Sanders, B. F., & Schubert, J. E. (2019). Primo: Parallel raster inundation model. *Advances in Water Resources*, *126*, 79–95.

Shamkhalchian, A., & de Almeida, G. A. (2021). Upscaling the shallow water equations for fast flood modelling. *Journal of Hydraulic Research*, *59*(5), 739–756.

Smith, L. N., & Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications* (Vol. 11006, p. 1100612).

Ştefănescu, R., Sandu, A., & Navon, I. M. (2014). Comparison of pod reduced order strategies for the nonlinear 2d shallow water equations. *International Journal for Numerical Methods in Fluids*, *76*(8), 497–521.

Sulavko, A. (2020). Bayes-minkowski measure and building on its basis immune machine learning algorithms for biometric facial identification. In *Journal of physics: Conference series* (Vol. 1546, p. 012103).

Sun, L., Gao, H., Pan, S., & Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, *361*, 112732.

Toro, E. F., & Garcia-Navarro, P. (2007). Godunov-type methods for free-surface shallow flows: A review. *Journal of Hydraulic Research*, *45*(6), 736–751.

Vlassis, N. N., & Sun, W. (2021). Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, *377*, 113695.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.

William, W., Ware, A., Basaza-Ejiri, A. H., & Obungoloch, J. (2018). A review of image analysis and machine learning techniques for automated cervical cancer screening from pap-smear images. *Computer methods and programs in biomedicine*, *164*, 15–22.

Wilson, M., Bates, P., Alsdorf, D., Forsberg, B., Horritt, M., Melack, J., ... Famiglietti, J. (2007). Modeling large-scale inundation of amazonian seasonally flooded wetlands. *Geophysical Research Letters*, *34*(15).

Xin Qi, S. M., Gustavo A. M. de Almeida. (2023). *Dataset supporting an article "physics informed neural networks for solving flow problems modeled by the shallow water equations"* [dataset]. University of SOuthampton. Retrieved from `https://doi.org/10.5258/SOTON/D2645` doi: 10.5258/SOTON/D2645

Yıldız, S., Goyal, P., Benner, P., & Karasözen, B. (2021). Learning reduced-order dynamics for parametrized shallow water equations from data. *International Journal for Numerical Methods in Fluids*, *93*(8), 2803–2821.

Yoon, T. H., & Kang, S.-K. (2004). Finite volume model for two-dimensional shallow water flows on unstructured grids. *Journal of Hydraulic Engineering*, *130*(7), 678–688.

Zhang, R., Liu, Y., & Sun, H. (2020). Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, *215*, 110704.

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, *394*, 56–81.

Appendix A PINN Design Experiments

This section illustrates the heuristic approach followed to determine the best possible design of the PINNs. We focus on Test 1, described in Section 3.1. All the PINNs shown in this section are trained from the same dataset resolved at 50 m resolution. Figures A1 and A2 show the accuracy ($\mathcal{R}_h$) of the PICNs and PIFCNs, respectively, as their architecture (number of layers and channels/neurons) is varied. In short, these figures show that it is difficult to conclude whether a single architecture can lead to significantly improved results, and we thus prioritize simplicity in our PINNs design. While this heuristic approach is, by definition, not guaranteed to find the optimal solution, it represents the summary of very many iterations. This holds for other tests and dataset resolutions considered in this study.

Similarly, we have tested three widely used activation functions: Relu, Sigmoid and Tanh (see Table A1). The chosen architecture for testing the PICN and PIFCN models is CNN-5-20 and FCNN-3(1000), respectively. For PICN, Sigmoid and Tanh display the same accuracy, while the result of the Relu-based PICN has higher errors. The PIFCN with Tanh yields better accuracy than using the other two activation functions. As a result, Tanh was chosen as the activation function to be employed in all PINNs discussed in this paper.

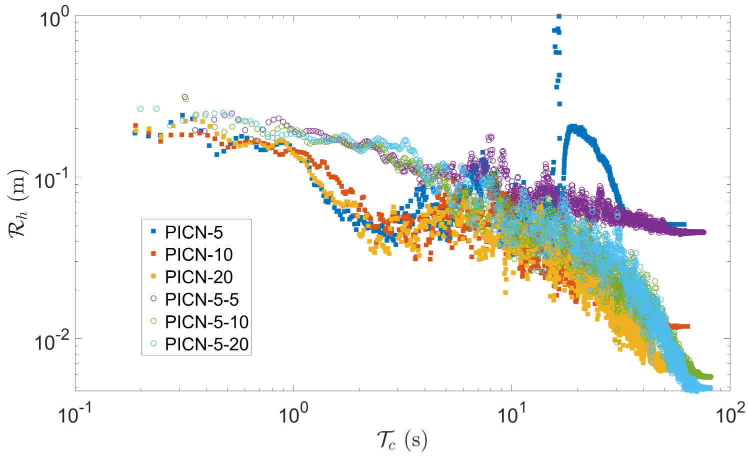

Figure A1. Comparison of PICNs with different architectures; the last hidden layer of all PICNs is one typical fully connected layer with 50 neurons. In the legend bar, the following format is adopted: PICN-X-Y, where the PICN has X channles in the first convolutional layer and Y channels in the second convolutional layer (thus, PICN-X denotes a network with one convolutional layer only).

Table A1. Results of water depth prediction by using Relu, Sigmoid and Tanh activation functions for PICN and PIFCN models. The trainset is a 50 m resolved dataset from Test 1; the evaluation metric is $\mathcal{R}_h$ and its unit is m.

Model	Relu	Sigmoid	Tanh
PICN	0.021	0.002	0.002
PIFCN	0.154	0.028	0.004

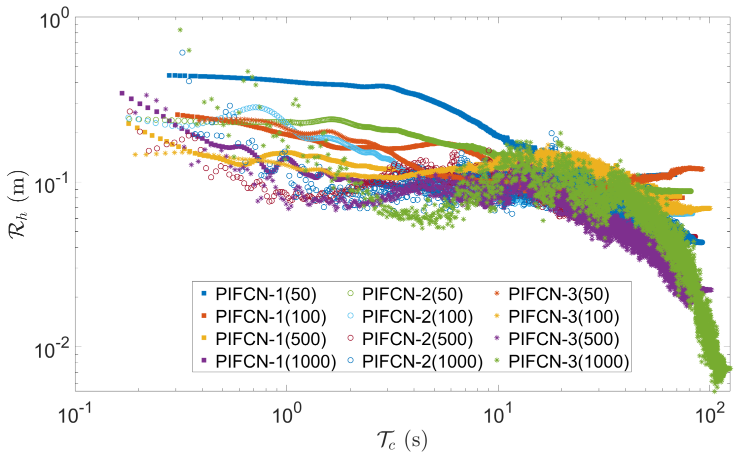

Figure A2. Comparison of PIFCNs with different architectures. In the legend bar, the following format is adopted: PIFCN-X(Y), where X denotes the number of hidden layers and Y is the number of neurons per layer.

Appendix B Further Spatial Analysis For Test 3

Table B1. Computation time and spatial prediction accuracy relative to benchmark simulation for the comparison. The unit for $\mathcal{R}_h^s$ is m, and the unit for $\mathcal{R}_{hu}^s$ and $\mathcal{R}_{hv}^s$ is m^2s^{-1}.

Model name	$\mathcal{T}_c$ (min)	t = 32 hours $\mathcal{R}_h^s$	$\mathcal{R}_{hu}^s$	$\mathcal{R}_{hv}^s$	t = 68 hours $\mathcal{R}_h^s$	$\mathcal{R}_{hu}^s$	$\mathcal{R}_{hv}^s$
PICN (50)	59.4	0.52	1.68	1.16	0.41	1.87	1.21
PICN (100)	15.3	0.40	1.72	1.18	0.37	1.92	1.20
PICN (200)	5.3	0.48	1.69	1.12	0.39	1.88	1.17
PIFCN (50)	504.9	0.59	2.21	1.32	0.52	2.21	1.28
PIFCN (100)	127.9	0.59	2.16	1.28	0.50	2.21	1.18
PIFCN (200)	30.2	0.63	2.27	1.21	0.47	2.16	1.15
FV (10)	2576.0	0.19	0.89	0.56	0.19	0.86	0.56
FV (25)	83.3	0.64	1.20	1.25	0.64	1.36	1.21
FV (50)	8.6	1.24	2.18	1.95	1.24	2.32	1.87

Table B1 summarizes the spatial prediction accuracy (i.e. $\mathcal{R}_h^s$, $\mathcal{R}_{hu}^s$, $\mathcal{R}_{hv}^s$) computed from a 50 m resolved set of points for each model at $t = 32$ and 68 hours, as well as their overall $\mathcal{T}_c$ (i.e. training time for PINN and computation time for FV). Among all the models, FV (10) and FV (50) achieve the highest and lowest accuracy, respectively. All PINNs present lower $\mathcal{R}_h^s$ than FV (25) and FV (50). On the other hand, FV (25) is more accurate than all PINNs in terms of hu prediction. PIFCN show a relatively similar value of $\mathcal{R}_{hu}^s$ to FV (50) at both time points. Moreover, the prediction accuracy of the PICNs and PIFCNs is less affected by the resolution of the input dataset than in the FV model. This last point may potentially be a main advantage of PINNs relative to conventional numerical methods in general, whose performance (numerical stability and accuracy) tends to be highly dependent on the mesh resolution.

Physics informed neural networks for solving flow problems modeled by the Shallow Water Equations

Xin Qi[1]

Gustavo A. M. de Almeida[1]

Sergio Maldonado[1]

[1]Faculty of Engineering and Physical Sciences, University of Southampton, Southampton SO16 7QF, UK

Key Points:

- Two PINNs are developed to find solutions to the 2D shallow water equations.
- In some scenarios, PINN accuracy is comparable to or surpasses finite-volume.
- Convolutional neural network outperformed fully connected neural network results.

Corresponding author: Xin Qi, `xq1n19@soton.ac.uk`

Abstract

This paper investigates the application of physics-informed neural networks (PINNs) to solve free-surface flow problems governed by the two-dimensional shallow water equations (SWEs). Two types of PINNs are developed and analysed: a physics-informed fully connected neural network (PIFCN) and a physics-informed convolutional neural network (PICN). The PINNs eliminate the need for labelled data for training by employing the SWEs, initial and boundary conditions as components of the loss function to be minimized. Solutions obtained by both PINNs are compared against those delivered by a finite volume (FV) solver for two idealized problems admitting analytical solutions, and one real-world flood event. The results of these tests show that the prediction accuracy and computation time (i.e., training time) of both PINNs may be less affected by the resolution of the domain discretization than the FV model. Overall, the PICN shows a better trade-off between computational speed and accuracy than the PIFCN. Also, our results for the two idealized problems indicated that PICN and PIFCN can provide more accurate predictions than the FV model, while the FV simulation with coarse resolution (e.g., 5 m and 10 m) outperformed PICN and PIFCN in terms of the speed-accuracy trade-off. Results from the real-world test showed the finely resolved (10 m resolution) FV simulation generally provided the most accurate approximations at flooding peaks. However, both PINNs showed better speed-accuracy trade-off than the FV model in terms of predicting the temporal distribution of water depth, while FV models outperformed the PINNs in their predictions of discharge.

1 Introduction

Free-surface flow phenomena are usually modeled by the shallow water equations (SWEs), a nonlinear system of partial differential equations (PDEs) governing the evolution of water depth and vertically averaged velocity in the two horizontal dimensions. Over the last decades, significant efforts have been made to approximate the solution to the SWEs in a discretized form through numerical methods, such as Finite Difference (FD) (e.g., Casulli, 1990; Molls & Chaudhry, 1995; Kurganov & Levy, 2002, and others), Finite Volume (FV) (e.g., Alcrudo & Garcia-Navarro, 1993; Bale et al., 2003; Botta et al., 2004; Yoon & Kang, 2004; Toro & Garcia-Navarro, 2007, and others), or Finite Element (FE) (e.g., Lynch & Gray, 1979; Hanert et al., 2005; Dawson et al., 2006; Marras et al., 2016, and others). These methods are now well-established and have been the object of extensive tests and validation (e.g., Toro & Garcia-Navarro, 2007; Wilson et al., 2007; Liang & Marche, 2009; LeVeque et al., 2011, and others). While the ability of these models to capture the main properties of free-surface flows has been widely verified, accurate solutions to real-world problems typically require the use of a finely resolved computational mesh, which tends to significantly increase the computational cost (Bernard et al., 2009; Liang, 2011; Juez et al., 2014). In explicit numerical schemes for the solution of the 2D SWEs, the computational cost C scales cubically with the size of the computational grid Δx (i.e., $C \sim \Delta x^{-3}$). As a result, important applications such as large-scale flood simulations are often beyond the capabilities of available numerical methods given existing computational resources. This is particularly challenging when simulations need to be performed in real-time, or as part of a probabilistic flood risk analysis (Leskens et al., 2014; Sanders & Schubert, 2019; Ferrari & Vacondio, 2022; J. Li et al., 2022). For example, flood risk management usually requires the simulation of a large number of inundation scenarios, which may increase the overall computational time by orders of magnitude. Although the computational speed of models can be improved by using state-of-the-art hardware and parallel computing algorithms (Leandro et al., 2014; Monnier et al., 2016), such techniques may still be insufficient to meet the computational requirements of many important applications (Kabir et al., 2020). Owing to these limitations, a cost-effective model for free-surface problems may offer an appealing alternative.

Over the last decades, Machine Learning (ML) models, and Artificial Neural Networks (ANNs) in particular, have found a wide range of applications, such as in financial prediction (Henrique et al., 2019), facial recognition (Sulavko, 2020), supply chain optimization (Carbonneau et al., 2008), radiation forecasting (El Naqa et al., 2015), and automatic cancer screening (William et al., 2018), to cite only a few. The extraordinary increase in applications of ML models is largely due to their ability to mathematically describe any nonlinear relationship between inputs and outputs according to the universal approximation theorem (Hornik et al., 1989), the increasing availability of data for training, and increasing computational power.

The first works using ML for the solution of problems governed by the SWEs are relatively recent and focused on the development of simple meta-models (e.g., Kabir et al., 2020; Liu & Pender, 2015; Bermúdez et al., 2019; Mahesh et al., 2022, and others). In this type of model, ML is used to build a prediction model to describe the input-output relationship previously obtained through the solution of the governing PDEs by another numerical approximation model; i.e. the ML model thus becomes a surrogate model. These surrogate models typically need to be trained using the results of a large number of numerical simulations conducted at fine resolution, which can be very computationally demanding.

Physics-Informed Neural Networks (PINNs), for which data (and therefore expensive numerical simulations) are not required for training, have gained increasing attention in recent years (Pang et al., 2019; Mao et al., 2020; Cai et al., 2021; Krishnapriyan et al., 2021; Jin et al., 2021). A PINN is essentially a ML algorithm which uses the information contained in the physical laws (such as the governing PDEs, boundary and initial conditions) to train the model. This eliminates the need for training data and thus, expensive numerical simulations. A data-free PINN is required to satisfy the governing PDEs, Initial Conditions (ICs) and Boundary Conditions (BCs) simultaneously. Recent applications of ML algorithms to solve complex physics phenomena have focused on the use of Deep Learning (DL) models (e.g., Sun et al., 2020; Zhang et al., 2020; Haghighat et al., 2020; Vlassis & Sun, 2021, and others). DL is a form of ANN with more than one hidden layer, which provides the complexity required to model intricate nonlinear relationships, such as those found in computer vision (Voulodimos et al., 2018), health management (Khan & Yairi, 2018), language translation (Rastgoo et al., 2021), and remote sensing (X. X. Zhu et al., 2017). In the past few years, a large number of data-free PINNs have been developed by employing DL techniques, such as the Fully Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs). For example, in Raissi et al. (2019) several FCNNs were trained to predict the solutions to various systems of PDEs, including Allen–Cahn, Schrödinger, Navier–Stokes, and Korteweg–de Vries equations. In the context of fluid dynamics, other implementations of FCNNs include those of Sun et al. (2020) and Mao et al. (2020), who used their DL models to find solutions to the Navier-Stokes (in steady state) and Euler equations (involving shock waves), respectively. The use of CNNs has also been explored, for instance, for problems governed by the Navier-Stokes (Cai et al., 2021) and Boltzmann transport equations (R. Li et al., 2021), or for predicting steady flow in random heterogeneous media (Y. Zhu et al., 2019). The success of these works shows that data-free PINNs should be considered as serious contenders for solving flow problems that are typically modelled by PDEs, and which have been traditionally solved using conventional numerical methods (FD, FV, etc.).

While ML trained from labeled data (i.e., mainly conventional numerical solutions) has been used to solve the SWEs (e.g., Mahesh et al., 2022; Ştefănescu et al., 2014; Yıldız et al., 2021; C. Li et al., 2023, and others), to the authors' knowledge only a very limited number of articles (e.g., Bihlo & Popovych, 2022) has been published so far on the use of a data-free PINNs for this purpose. In Bihlo and Popovych (2022), a PINN (specifically, based on a FCNN) was employed to find solutions to the SWEs on a spherical domain, and the focus was on idealized problems which may find applications in meteorology. Whether a similar DL technique may be used to accurately and efficiently solve challenging free-surface flow problems involving friction and complex boundary conditions, such as large-scale simulations of flow over complex topography in rivers and coastal areas, remains an open question.

The solution of PDEs, and in particular of the SWEs, using PINN algorithms is still in its infancy and further investigation is required to understand the main characteristics of solutions obtained by these methods. Firstly, the trainset for PINNs needs to be generated from a particular, discrete, set of points. It remains unclear how accuracy and computational performance (i.e., training speed) depend on the discretization of the domain. Additionally, both FCNNs and CNNs are commonly used DL models in the research field of PINN. However, in a specific problem governed by a system of PDEs, it is usually difficult to determine which one will deliver the best performance before carrying out tests.

The aim of this paper is to develop and test two different PINN models to approximate solutions to various free-surface flow problems governed by the 2D SWEs. The PINN models are based on the FCNN and CNN approaches, and are hereafter referred to as PIFCN (physics informed fully connected network) and PICN (physics informed convolutional network), respectively. These models are data-free in that they do not require data from separate numerical simulations, or laboratory/field measurements, to train the networks. In this paper both PIFCNs and PICNs are compared against the Finite Volume (FV) solver of the 2D SWE developed by de Almeida et al. (2016) through a set of test cases including two idealized flow problems and one real-world flood event. The rest of this paper is organized as follows. First, the governing equations and the framework of both PINNs are described in Section 2. This section also provides a concise review of FCNNs and CNNs. In Section 3, the accuracy and computational performance of the proposed physics-informed networks (PIFCN and PICN) are investigated for the three test cases. The main outcomes of the study are discussed and summarized in Section 4.

2 Methods

2.1 Overview

Most problems requiring the simulation of free-surface flows in the horizontal plane, such as flow in rivers and estuaries, dam-breaks, and flood wave propagation can be modeled by the SWEs. The 2D SWEs represent a system of nonlinear, hyperbolic PDEs describing the conservation of water mass and depth-average momentum, which can be expressed as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = \mathbf{S}(\mathbf{U})\ ; \tag{1}$$

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix},\ \ \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix},\ \ \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix},\ \ \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ gh(s_{ox} - s_{fx}) \\ gh(s_{oy} - s_{fy}) \end{bmatrix}, \tag{2}$$

where x and y are the spatial (horizontal) coordinates; t is time; $h(x, y, t)$ is the water depth; $u(x, y, t)$ and $v(x, y, t)$ denote the x and y components of the depth-averaged flow velocity, respectively; $s_{ox} = -\partial z/\partial x$ and $s_{oy} = -\partial z/\partial y$ are the bed slopes in the x and y directions, respectively, and $z(x, y)$ is the terrain elevation (assumed constant in time); s_{fx} and s_{fy} denote the friction slopes in the x and y directions, respectively. The friction slopes can be modeled using Manning-Strickler's expression, $s_{fx} = n^2 u\sqrt{u^2 + v^2} h^{-4/3}$, $s_{fy} = n^2 v\sqrt{u^2 + v^2} h^{-4/3}$, where n is the Manning coefficient. Solutions $\mathbf{U} = (h, hu, hv)^T$ to this system (subject to well-posed boundary and initial conditions) can be computed by several numerical methods available, as discussed in the Introduction.

In this paper we propose a ML-based solution to this problem, whereby the input layer $\mathbf{x}$ represents the independent variables and parameters of the problem, $\mathbf{x} = (x, y, t, n, z)$, and the trained model $\aleph$ is expected to provide an approximate solution for $h(\mathbf{x})$, $hu(\mathbf{x})$ and $hv(\mathbf{x})$ in the corresponding domain; in other words:

$$\mathbf{U}(\mathbf{x}) \;\cong\; \tilde{\mathbf{U}}(\mathbf{x}) \;=\; \aleph(\mathbf{x}\,;\boldsymbol{\Gamma}), \tag{3}$$

where $\tilde{\mathbf{U}}(\mathbf{x})$ denotes the output from the PINN, which is in turn defined by the group of trainable parameters $\boldsymbol{\Gamma}$ (e.g., convolutional filter, weights and biases). The PINN models proposed in this paper are trained by minimizing the composite loss function, defined as:

$$\mathcal{L} = \boldsymbol{\lambda}_1 \cdot \boldsymbol{\mathcal{L}}_p + \boldsymbol{\lambda}_2 \cdot \boldsymbol{\mathcal{L}}_b + \boldsymbol{\lambda}_3 \cdot \boldsymbol{\mathcal{L}}_0, \tag{4}$$

where $\mathcal{L}$ (a scalar) is the loss function to be minimized, $\boldsymbol{\lambda}_{1-3}$ are the vectors of penalty coefficients for every specific loss term; namely, $\boldsymbol{\mathcal{L}}_p$ penalizes the residuals of the SWEs, $\boldsymbol{\mathcal{L}}_b$ and $\boldsymbol{\mathcal{L}}_0$ penalize the BCs (subscript b) and ICs (subscript 0) residuals, respectively. These loss terms are in turn given by:

$$\boldsymbol{\mathcal{L}}_p = \frac{1}{N}\sum_{i=1}^{N} |\partial_t \tilde{\mathbf{U}}_i + \partial_x \mathbf{F}(\tilde{\mathbf{U}}_i) + \partial_y \mathbf{G}(\tilde{\mathbf{U}}_i) - \mathbf{S}(\tilde{\mathbf{U}}_i)| \tag{5a}$$

$$\boldsymbol{\mathcal{L}}_b = \frac{1}{N_b}\sum_{i=1}^{N_b} |\tilde{\mathbf{U}}_{b,i} - \mathbf{U}_{b,i}| \tag{5b}$$

$$\boldsymbol{\mathcal{L}}_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |\tilde{\mathbf{U}}_{0,i} - \mathbf{U}_{0,i}| \tag{5c}$$

The tilde symbol ($\tilde{}$) denotes neuronal network output and the subscript $i \in [1, N]$ refers to the i^{th} collocation point. N represents the number of collocation points, which in this paper is defined from a uniformly discretized domain of the independent variables $N = n_x \times n_y \times n_t$, where n_x, n_y and n_t are the number of points used to discretize the domain along the x, y and t coordinates, respectively. The boundaries of the spatio-temporal domain are represented by a subset of N; in particular, the model will employ $N_b < N$ and $N_0 < N$ points to define the BCs and ICs, respectively.

For each approximate solution produced by the PINN, the partial derivatives in Eq. 5a are computed through the method of automatic differentiation (autodiff) (Paszke et al., 2017), which back-propagates derivatives from the outputs to the targeted inputs through the chain rule to compute the desired derivatives (Cai et al., 2021; Baydin et al., 2018). Thus, the partial derivatives of the approximate solution with respect to the independent variables can be computed without the errors common to numerical differentiation techniques. The loss function is minimised using the gradient descent method, with gradients of the loss function with respect to trainable parameters computed by back-propagation. These parameters can be updated either using all, or a subset (batch) of the collocation points.

One significant difficulty of solutions to flow problems modeled by the SWEs is the so-called wet-dry front issue (i.e., moving boundary). Physically, the value of the flow depth h cannot be negative. Areas of the domain where such solutions may be obtained correspond to dry areas, which are not governed by the SWEs. To overcome this problem, we set $\boldsymbol{\mathcal{L}}_p = 0$ if the predicted value of $\tilde{h}$ is negative. This ensures that the model does not penalize making predictions outside the wet domain.

Figure 1 shows a diagram illustrating the overall modeling framework proposed in this paper for solving the SWEs by a PINN method. Note that the collocation points can be chosen randomly in the space-time domain and their number prescribed. The general steps are outlined below.

1. Define the architecture of the PINN
2. Initialize the hyperparameters for the PINN
3. Compute the outputs from the PINN with given inputs
4. Compute the derivatives with respect to x, y, t and the corresponding loss $\mathcal{L}$
5. Update the PINN based on $\mathcal{L}$
6. Repeat steps 3 to 5 until the end of the user-prescribed number of training epochs

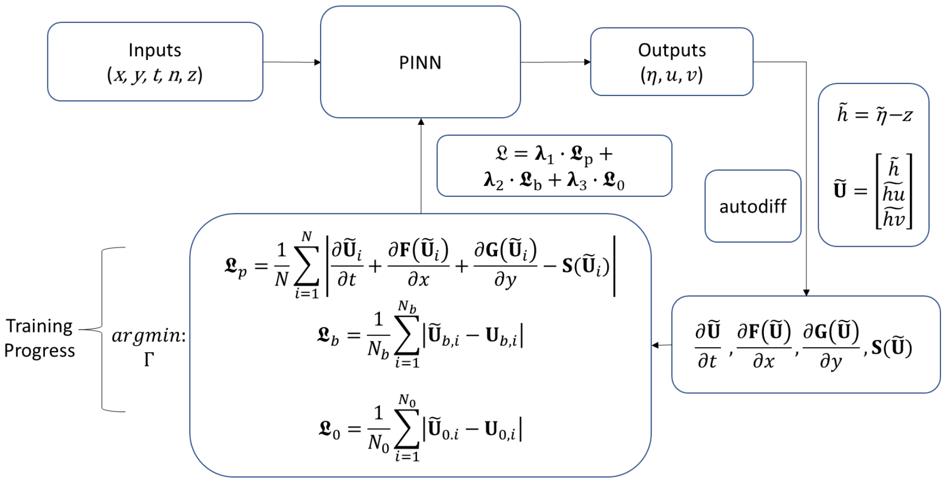

Figure 1. A schematic diagram of a physics informed neuronal network (PINN) for finding approximate solutions to the shallow water equations.

2.2 Fully Connected Neural Network

The FCNN is the most commonly applied ML model and often includes more than one hidden layer. Every hidden layer receives the signals from the previous layer, performs basic computations defined at each neuron, and passes the results to the next layer (Haykin, 2009). Figure 2 shows a diagram of a FCNN. Mathematically, the basic function of the output for the j^{th} hidden layer $\mathbf{y}_j$ is:

$$\mathbf{y}_j = \varphi\left(\mathbf{W}_j \mathbf{y}_{j-1} + \mathbf{b}_j\right) \tag{6}$$

where $\mathbf{W}$ is the matrix of weights, $\mathbf{b}$ is the vector of biases and $\varphi()$ is the activation function.

In the proposed method, solutions for each output variable $\eta(\mathbf{x}) = h(\mathbf{x}) + z$, $hu(\mathbf{x})$, $hv(\mathbf{x})$ are approximated by 3 separate FCNNs with the same structure, as illustrated in Figure 2. Every FCNN receives the same raw inputs. As a result, the trainable parameters of the solution for each output variable are decoupled. This can significantly improve the prediction accuracy in multivariate problems, especially when the distributions and magnitudes of the variables are significantly different (e.g., Sun et al., 2020; Gao et al., 2021; Guo et al., 2020, and others).

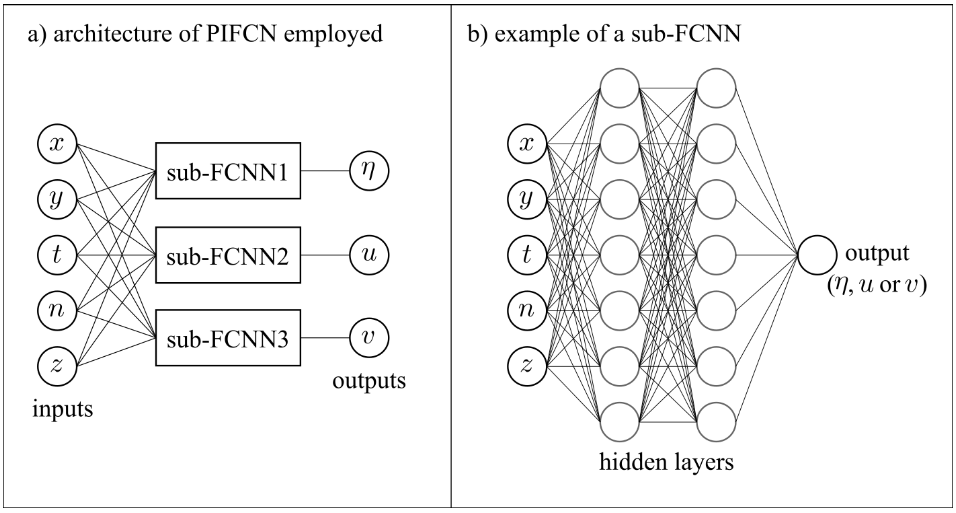

Figure 2. (a) The architecture of the physics-informed fully connected networks (PIFCNs) employed in this paper. (b) An example of a typical fully connected neuronal network (FCNN) which is employed as a sub-network within the PIFCN to predict each individual output; as illustration, 2 hidden layers with 7 neurons each are shown, but these hyperparameters are varied in this study.

2.3 Convolutional Neural Network

The CNN adds one or more convolutional layers that extract features of the raw training dataset before feeding this onto the typical hidden layers used to build FCNNs. The general expression for the convolution operator $\star$ with 1 stride is:

$$(\mathbf{s} \star \mathbf{k})_i = \sum_{j=1}^{n} k_j s_{i+j-1} \qquad i = 1, 2, \dots, m-n+1 \tag{7}$$

where $\mathbf{s}$ denotes the input signal vector of length m (in this paper, this is $\mathbf{x}$), and $\mathbf{k}$ denotes the trainable filter of length n. The convolution operation is to slide the preset convolutional filter over the signal input and output the signal with a shorter length (i.e., the input vector is shortened by $n-1$ elements). The shorter length of the convolved output signal allows the following typical hidden layer to have fewer neurons, facilitating the network's learning of large-scale problems with high complexity (Gao et al., 2021).

Figure 3 shows the structure of the CNN used in this paper. The trainset is generated from a number of points (i.e., collocation points) randomly sampled from a grid of equally spaced points. Each output variable, $h(\mathbf{x})$, $hu(\mathbf{x})$, $hv(\mathbf{x})$, is also predicted by a separate sub-CNN.

2.4 PINN design

The accuracy and computational performance of the PINNs described in the previous sections will be assessed and compared against the corresponding performance and solutions by a conventional FV model. There is currently no universal design approach to determine the optimal, or even appropriate, structure for a neural network (Bihlo & Popovych, 2022). The general selection rule for PINN design is to find a structure with the lowest possible complexity that achieves the desired accuracy of prediction. This rule can usually help provide an AI model with quick learning speed and improved predic-

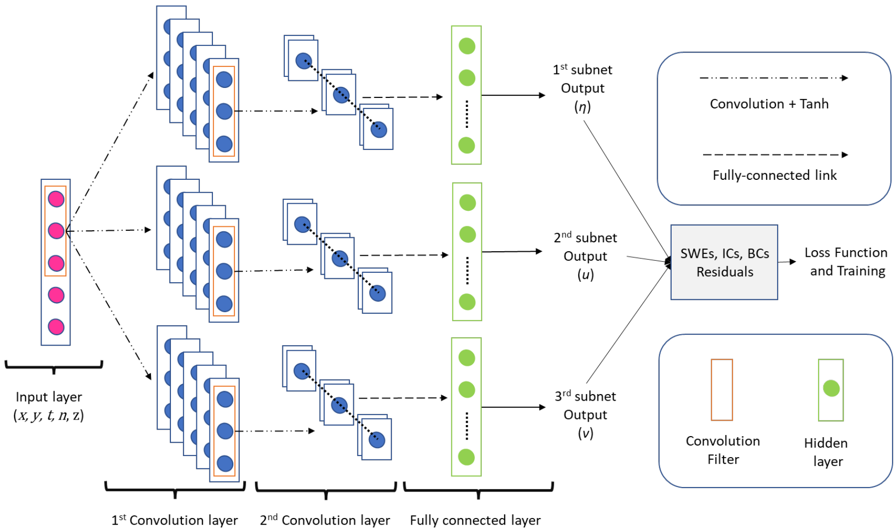

Figure 3. An example of the structure of a CNN-based model with 3 subnets for solving free-surface flow problems. Each output variable (η,u or v) is approximated by a separate CNN with the above structure; all sub-networks receive the same inputs. Each CNN has two convolutional layers and one hidden layer. The hyperparameters shown in the figure are discussed in Section 2.4.

tion capabilities while avoiding overfitting issues (Blumer et al., 1987). In this paper, the final decision for the model structure (i.e. hyperparameters such as the number of neurons, hidden layers, and convolutional layers and channels in the case of CNN) was made after many practical attempts (see Appendix A). As the evaluation of the PINNs performance in this paper consists of two, often competing, criteria (accuracy and computational cost), it may be difficult to find a single assessment metric to guide the PINNs design. Hence, we give priority to accuracy by gradually increasing the complexity of the PINNs until similar or higher accuracy than benchmark results (e.g. from an analytical solution or a finely resolved FV simulation) is attained. Generally, in our design iterations, the number of hidden layers and the corresponding neurons for building PINNs (i.e., PIFCN and PICN) started from 1 and 50, respectively. The number of convolutional layers and corresponding channels started from 1 and 5, respectively. For both PICN and PIFCN we use the hyperbolic tangent activation function (Tanh). Note that the PINN design may change significantly depending on domain and flow conditions; i.e., it can be very problem-specific. It is also important to recognize that the networks chosen do not represent the strictly optimal structure, but only the best out of the subset of structures that were tested.

For improving the learning speed and reducing the effect of parameter initialization, the Batch Normalization method of Ioffe and Szegedy (2015) was used, which normalizes the signals between adjacent convolutional or hidden layers. The Adam optimizer (Kingma & Ba, 2014), along with the '1-cycle' (Smith & Topin, 2019) strategy was used to control the training of the PINNs. The PINNs were implemented on the Pytorch platform Paszke et al. (2017). The FV simulation and the training of the PIFCNs and PICNs were performed using the University of Southampton's supercomputer Iridis 5 ensuring that the exact same hardware resources were employed (thus ensuring a fair comparison across all simulations performed).

3 Case studies

This section describes three case studies used to test the PINNs, comparing their results against analytical and numerical (Finite Volume) solutions. The first and second tests are idealized 1D (unsteady and steady, respectively) flow problems for which analytical solutions are available. However, simulations were performed on a 2D domain since the ultimate aim is to employ the PINNs developed here in 2D flow problems. The third test case is an unsteady two-dimensional simulation of a real-world flood event that took place in the Tiber river, Italy. This case study has been previously employed to evaluate the performance of other numerical models (e.g., Morales-Hernández et al., 2016; Shamkhalchian & de Almeida, 2021, and others).

Topographic data used in all tests are defined by square grids with different resolutions. The grid points are used to generate a triangular mesh for the FV model. These are also employed, along with defined temporal steps, as the collocation points for the PINNs training. The accuracy of the solutions will be assessed by the root mean square error, $\mathcal{R}$, of the outputs of each model relative to the benchmark solution. For example, in the evaluation of accuracy for the prediction of h with N_p output points, the performance metric is defined as $\mathcal{R}_h = \sqrt{\sum(h_i - \tilde{h}_i)^2/N_p}$, where h_i is the benchmark solution (i.e., the analytical solution when available, or the solution of the FV model at fine resolution). The second performance metric we employ is the computational cost, $\mathcal{T}_c$, which represents training time for the PINNs (PICN and PIFCN), and run time for the FV model. In the results presented in the following sections, predictions by the FV, PIFCN and PICN models are labelled with the different resolutions used. For example, FV (10) represents a 10 m resolved simulation using the FV hydraulic model, and PICN (50) refers to the prediction of the PICN trained from a 50 m resolved dataset.

3.1 Flood wave propagation over a horizontal plane

The first test case is a one-dimensional simulation of an inundation wave propagating over a horizontal bed. A time-dependent BC is imposed at $x = 0$. Under the idealized assumption of a flow velocity that is constant in space and time, the problem admits an analytical solution which can be expressed as (Hunter et al., 2005):

$$h_a(x,t) = \left\{ \frac{7}{3} \left(n^2 u^2 (x - ut) \right) \right\}^{3/7}, \tag{8}$$

where the subscript a is used to denote the analytical solution. The domain used is a 100 m wide, 1200 m long channel. The constant velocity is set as $u(x,t) = 0.29$ ms^{-1} and the boundary condition $h(x = 0, t)$ is given by Eq. 8. The domain is initially dry, i.e., $h(x, t = 0) = 0$. Manning's coefficient n is set to 0.03 sm$^{-1/3}$. The duration of the simulation is 3600 s. The FV model was run at resolutions of 1, 2, 5 and 10 m, while the PINN models were trained with datasets defined at resolutions of 10, 25, 50 and 100 m. While the time step of the explicit FV scheme is controlled by the Courant-Friedrichs-Lewy (CFL) stability condition, the regression approximation implemented by the PINN model is not limited by temporal resolution. However, the time step adopted to train the PINN model is a factor that clearly affects both accuracy and computational performance. For this test, we use a temporal resolution for the PINN of 300 s. The selected batch size is set as the full set of collocation points $n_x \times n_y \times n_t$.

The architecture of the PICN consists of 2 convolutional layers (the first and second layers have 5 and 20 channels, respectively) and 1 fully connected hidden layer with 50 neurons. The architecture for the PIFCN consists of 3 fully connected hidden layers, each of which has 1000 neurons.

Figures 4 and 5 illustrate the values of $h(x, y = 50\text{m})$ and $hu(x, y = 50\text{m})$ (left vertical axes), and the corresponding error (right vertical axes) $\epsilon_h(x, y = 50\text{m}) = \tilde{h}(x, y =$

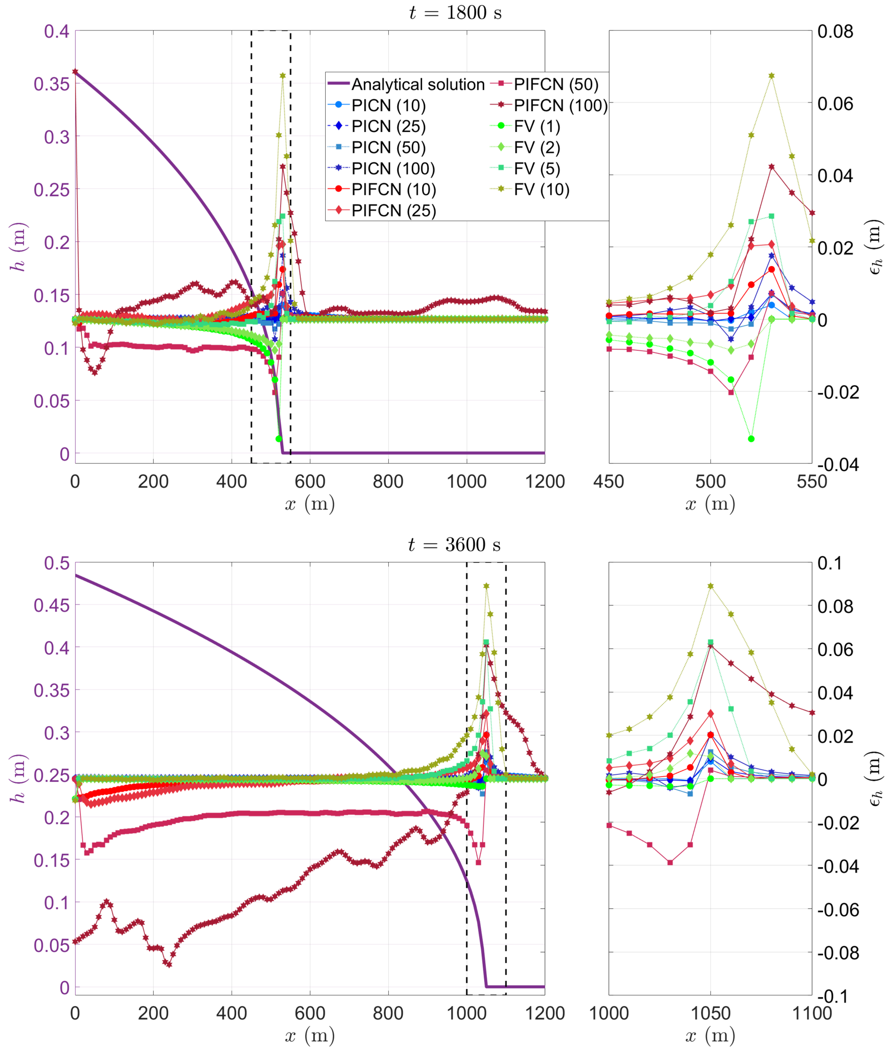

Figure 4. Test 1: Longitudinal profiles ($y = 50$ m) of water depth errors ϵ_h relative to the analytical solution obtained by each of the models at $t = 1800$ s (top) and $t = 3600$ s (bottom), shown against right $y-$axis. The analytical solution for h (purple line) is plotted against the left $y-$axis; note that the right-side figure is the enlarged version of the rectangular box in the left-side figure; both figures share the same right $y-$axis.

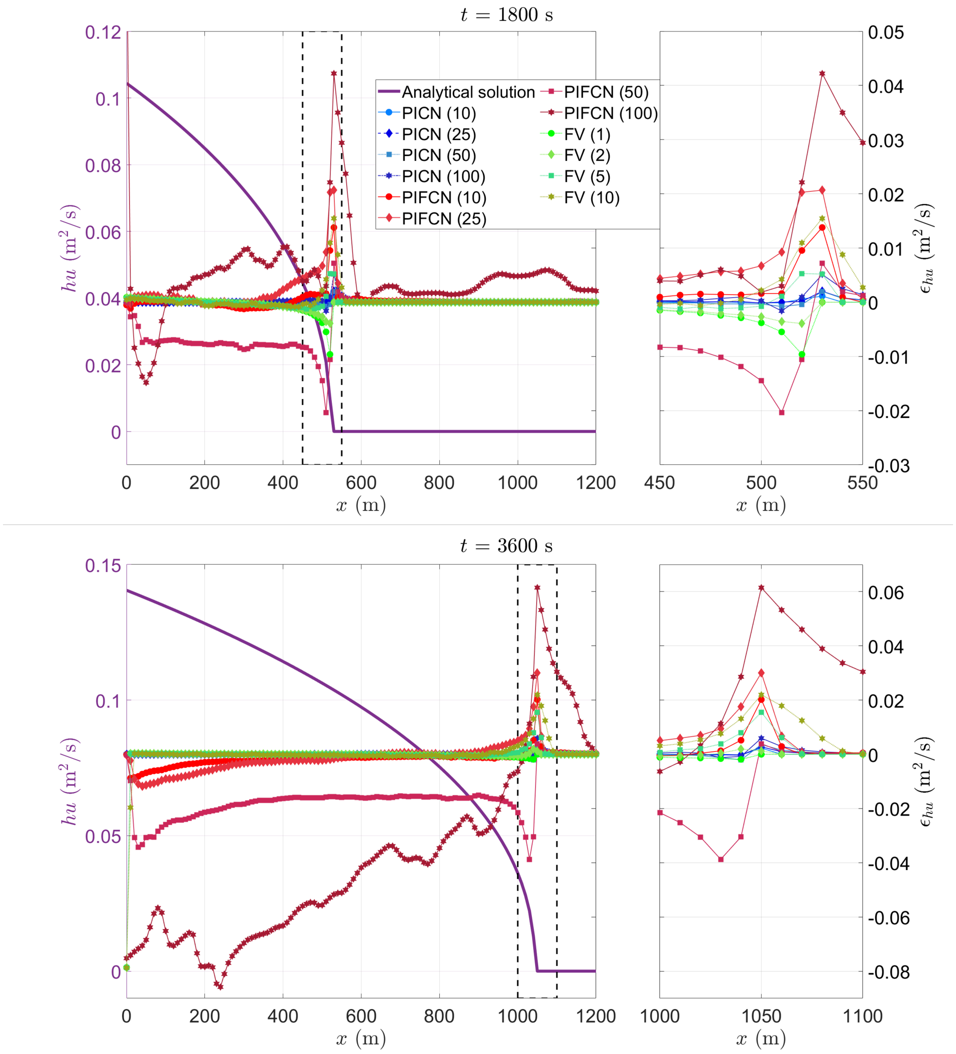

Figure 5. Test 1: Longitudinal profiles (y = 50 m) of water depth errors ϵ_{hu} relative to the analytical solution obtained by each of the models at t = 1800 s (top) and t = 3600 s (bottom). The analytical solution for hu (purple line) is plotted against the left $y-$axis; note that the right-side figure is the enlarged version of the rectangular box in the left-side figure; both figures share the same right $y-$axis.

50m) $- h_a(x, y = 50\text{m})$ and $\epsilon_{hu}(x, y = 50\text{m}) = \tilde{hu}(x, y = 50\text{m}) - (hu)_a(x, y = 50\text{m})$ computed by all three models at $t = 1800$ and 3600 s, respectively. Values of hv are not reported as the test case is fundamentally one-dimensional. Overall, all the water depth predictions, with the exception of PIFCN (100), show good agreement with the analytical solution (i.e. most results displaying $|\epsilon| < 0.01$m). As the position of the wet-dry front predicted by the models does not exactly match the analytical solution, and the front is steep at that point, errors are larger in this region. PICN and FV both show similar prediction accuracy of both h and hu, whereas PIFCNs with coarsely resolved trainsets (i.e., 50 m and 100 m) provide higher prediction errors of hu.

Figure 6 shows $\mathcal{R}_h$ (relative to the analytical solution h_a) for all results obtained with the PICN, PIFCN, and FV models as a function of the corresponding computational time $\mathcal{T}_c$. The sum to compute $\mathcal{R}_h$ is over all collocation points; i.e., spanning the whole spatio-temporal domain. In this figure, the various points (blue and red) presented for each PINN model represent solutions obtained at different epochs during the training of the networks, which correspond to different computation time and level of accuracy. The green cross points represent the simulation accuracy and computation time for the FV model. The results in this figure are based on model (i.e. PICN, PIFCN, and FV) outputs at the same grid points selected from the entire domain with a spatial and temporal resolution of 10 m and 360 s. Predictions of hu follow the general pattern observed for h on Figure 6 and are not reported here to avoid repetition. Figure 6 allows us to comparatively assess the performance of the models tested in terms of their speed-accuracy trade-off. Based on this criterion, a model performs better than another when it provides more accurate results under the same computational time, or vice-versa; in other words, the best results are those closest to the bottom left corner of the plot.

Figure 6 shows that FV (10) and FV (5) produce sub-centimetre $\mathcal{R}_h$ (which is usually considered a good level of accuracy for many applications) at least one order of magnitude faster than the PINN models, whereas FV (2) takes slightly longer than PICNs (for the same level of accuracy), and FV (1) only outperforms PIFCN (10) in terms of the speed-accuracy trade-off. All PINNs except PIFCN (100) show the potential to achieve better accuracy of prediction than the FV model at the highest resolution tested here (1 m), provided they are trained for long enough. PICNs provide a faster solution (for similar $\mathcal{R}_h$ values) than PIFCNs. Also, for PIFCN, the trainset size (which in this case is determined by the resolution) did not significantly affect its maximum accuracy at resolutions ≤ 50 m, whereas the accuracy of the FV model continues to improve as the mesh is refined below 10 m.

3.2 Subcritical steady flow over an undulating bed

The second test case represents a 1D, steady, non-uniform flow over an undulating bed, for which an analytical solution is available (see MacDonald, 1996; de Almeida & Bates, 2013; Delestre et al., 2013). This test case will be used to evaluate the solution obtained by the PICN and PIFCN in a problem with variable topography. The (rectangular) channel is 1000 m long, and Manning's coefficient n is set to 0.03 sm$^{-1/3}$. The constant inflow discharge per unit width of the channel is $q_x = uh = 2$ m^{2}s^{-1}, and the downstream water depth is $\frac{9}{8}$ m. We prescribe the following function representing the water depth $h(x)$ (which is the benchmark solution against which the PINN approximations will be compared):

$$h(x) = \frac{9}{8} + \frac{1}{4}\sin\left(\frac{\pi x}{500}\right). \tag{9}$$

We model this 1D problem in a 2D domain using a width of 50 m (and $q_y = 0$) for the reasons discussed previously. Also, although the solution sought is for a steady flow problem, the steady condition was reached via an unsteady flow simulation, as the

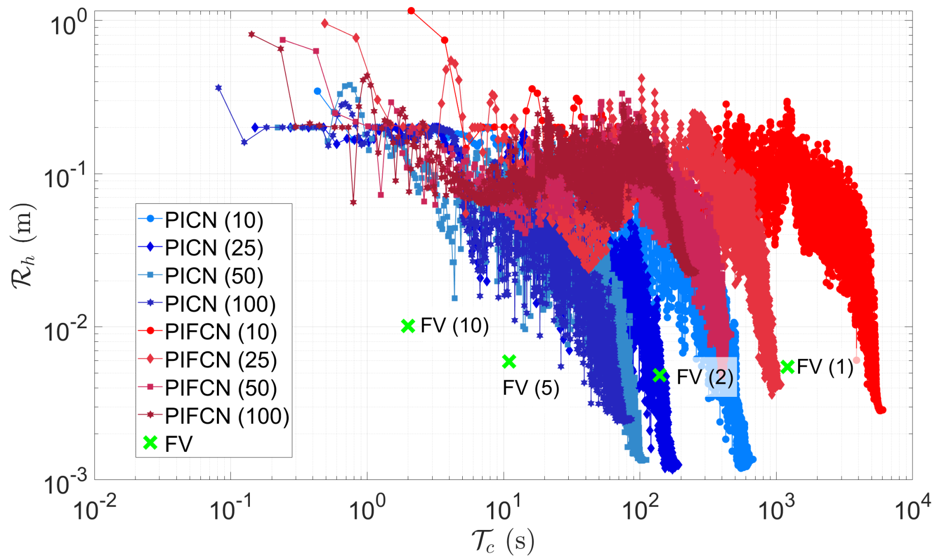

Figure 6. Test 1: Values of $\mathcal{R}_h$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV); note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model).

object of this paper is to test approximate methods to solve the time-dependent SWEs. The unsteady simulations were run from an initially dry domain over a period of 20 hours, whereby the upstream BCs increase linearly with time from zero to the aforementioned constant values over the first 10 hours of the simulation.

The training dataset for PICN and PIFCN was obtained from grids resolved at 5, 10, 25 and 50 m at the following times: 0, 1, 3, 5, 10, 15 and 20 hours. The selected batch size is $2/7 \times N$, where the value 2/7 comes from trial and error (larger batch sizes decreased the accuracy of the results). The FV model was run at resolutions of 2, 5 and 10 m.

For this case, the architecture of the PICN consists of 2 convolutional layers (the first and second layers have 5 and 20 channels, respectively) and 1 fully connected hidden layer with 50 neurons (same as in Test 1). The architecture of the PIFCN consists of 3 fully connected hidden layers, each of which has 1000 neurons (different from Test 1).

Figure 7 shows the analytical curve for depth profile at the centre of the channel $h(x, y = 30$ m $)$ (left axis) and the corresponding errors of each of the approximate solutions ϵ_h (right axis) predicted by the PICN (blue points), PIFCN (red points), and FV models (green points). Figure 8 presents similar results but for the variable hu. As the analytical solution is for the steady state, only the results at the end of the simulations are assessed. Overall, all models tested delivered results at sub-centimeter level of accuracy for h. The three PICNs showed the lowest errors of both h and hu, followed by FV (2).Values of ϵ_{hu} obtained from FV models display small (mostly within 1% of the actual value of hu) spatial variations, while they nearly are constant for both PIFCN and PICN.

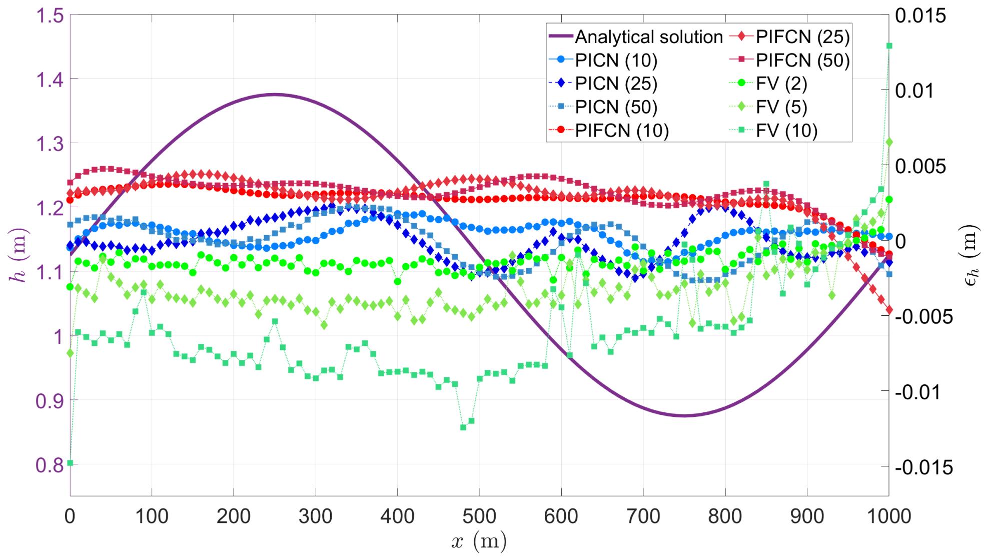

Figure 7. Test 2: Longitudinal profiles (y = 30 m) of water depth errors obtained by each of the models at the end of the simulation/training (right $y-$axis). The analytical solution h (purple line) is plotted against the left $y-$axis.

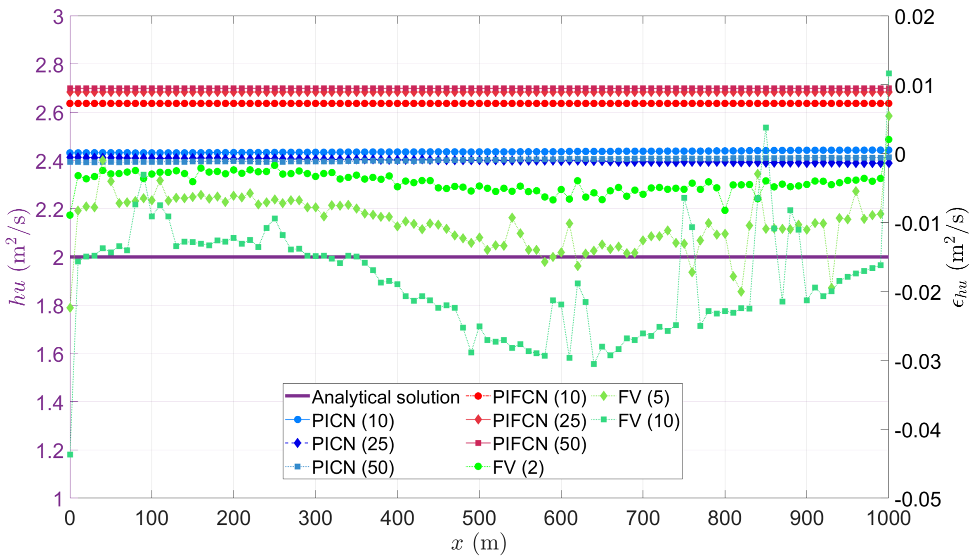

Figure 8. Test 2: Longitudinal profiles (y = 30 m) of water depth errors obtained by each of the models at the end of the simulation/training (right $y-$axis). The analytical solution hu (purple line) is plotted against the left $y-$axis.

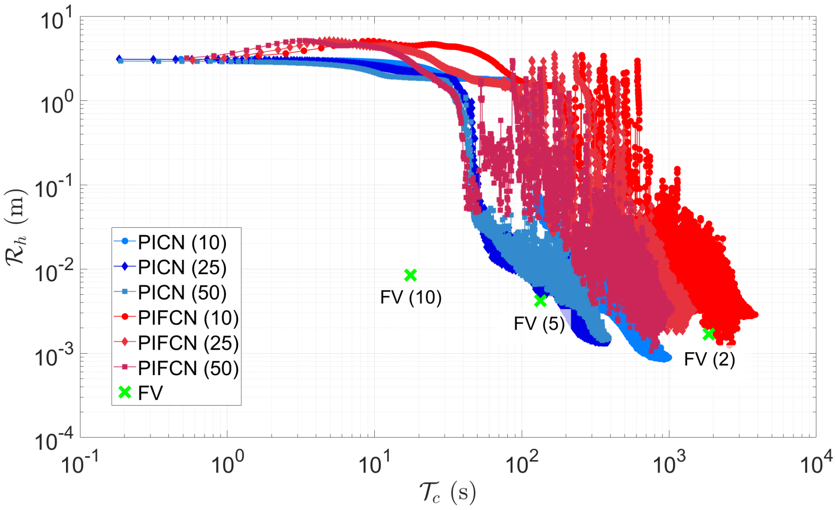

Figure 9. Test 2: Values of $\mathcal{R}_h$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV); note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model).

Figure 9 presents the values of $\mathcal{R}_h$ against the corresponding computational time taken to train the PICN (blue points), PIFCN (red points), and to run the FV model (green cross points) at different resolutions. The value of $\mathcal{R}_h$ of each model is calculated from its steady-state predictions of h; namely: $\mathcal{R}_h = \left.\left(\sqrt{\sum(h_i - \tilde{h}_i)^2/N_p}\right)\right|_{t=t_s}$, where t_s is the time after which a steady state is reached for each PINN or FV model. For the computation time of FV models described in Figure 9, the value of $\mathcal{T}_c$ is the time required for all FV models to reach steady state. The results for hu show a pattern similar to that in Figure 9 and are not presented for conciseness. All simulations achieve sub-centimetric $\mathcal{R}_h$, with FV (10) delivering the results at least one order of magnitude faster than the other solutions. PIFCN (10) was the slowest of all models. Figure 9 shows that the prediction of h from PICN (10) displays the highest accuracy, with an $\mathcal{R}_h$ of 0.85 mm, although this was obtained at a computation time that was 56 times longer than FV (10). All the PINN results also attain an accuracy higher than or similar to that of FV(2). In this test case, the relative differences in the prediction accuracy among the PICN models is less than the difference observed from FV (5) to FV (10). In terms of the influence of resolution on the computational speed, the PICN is also less sensitive than PIFCN in this problem.

3.3 Simulation of real-world river flooding

While Tests 1 and 2 have assessed the ability of PINN models to deal with important aspects of flow problems, such as unsteadiness and variable topography, both case studies represented idealized, one-dimensional problems. In order to investigate the performance of PINNs under more complex and realistic problems, this section presents the results of simulations of a real-world scenario. The scenario in question is a flood event that occurred between 27 November and 1 December 2005 in the Tiber river (Morales-Hernández et al., 2016), which flows from the Apennine Mountains to the Tyrrhenian Sea in Italy. The reach of river employed in this simulation is approximately 6 km long and is located near the city of Rome. In this region, the mean discharge of the Tiber river is 267 m^3s^{-1}, while its peak discharge for a 200-year return period is around 3200 m^3s^{-1}. The event modeled in this paper was also previously simulated in Morales-Hernández et al. (2016) and Shamkhalchian and de Almeida (2021). The domain comprises an area of 6 km × 2 km. The duration of the event simulated is 113 hours. The values of Manning's coefficient n used are the same as in Morales-Hernández et al. (2016) and Shamkhalchian and de Almeida (2021); namely, $n = 0.035$ $sm^{-1/3}$ for the main channel, and $n = 0.0446$ $sm^{-1/3}$ for the floodplains.

The boundary conditions were obtained from Morales-Hernández et al. (2016), and correspond to the time series of flow discharge and water surface elevation at the upstream and downstream sections of the river at the boundary of the computational domain. The initial conditions $\mathbf{U}(x, y, t = 0)$ were defined from the results of the FV model under steady-state conditions ($Q = 374$ m^3s^{-1}) performed at 5 m resolution. PINNs were trained from datasets resolved at 50, 100 and 200 m, while the FV model was run using meshes generated from gridded data at resolutions of 10, 25 and 50 m. The corresponding temporal resolution for the trainset for the PINNs is 4 hours. The batch size was set to one third of the total number of collocation points.

For this test case, the architecture of the PICN consists of 2 convolutional layers (the first and second layers have 10 and 40 channels, respectively) and 1 fully connected hidden layer with 100 neurons. The architecture of the PIFCN consists of 3 fully connected hidden layers, each having 2000 neurons. Our tests showed that further increasing the network complexity would not improve the model's prediction accuracy, and may substantially increase the training time and/or cause the program to exceed the memory capacity of the computer resources used.

Since an analytical solution is not available for this problem, the results of the FV simulation at fine resolution (5 m) were used as the benchmark. The accuracy of the solutions of the time-dependent variables is assessed at two cross-sections (located approximately at distances of 1/3 and 2/3 of the length of the river within the domain from the upstream boundary, and hereafter referred to as S1 and S2, respectively) at 1 hour temporal resolution.

Figures 10 and 11 illustrate the time series of prediction errors (right vertical axes), along with the actual predicted values of the flow depth h and flow discharge Q (left vertical axes) at cross-sections S1 and S2 for each PICN, PIFCN, and FV models. Figure 10 shows that the FV and PIFCN simulations consistently predict larger and lower depths than the benchmark solution, respectively, at both cross sections in the main channel, while PICN results display both positive and negative values of ϵ_h. Results from PICNs at S1 and S2 are markedly more accurate than those delivered by PIFCNs and the coarse-resolution FV models. For example, FV (50) and FV (25) produced results that deviate substantially (i.e. up to approximately 1.2 m and 2.5 m at S1 and S2, respectively) from the benchmark solution. On the other hand, FV (10) generally produced the most accurate depth predictions out of all models tested. The ability of the models to predict flow velocities (and therefore, the volumetric flow rate Q) is assessed by $\epsilon_Q = \tilde{Q} - |Q|$, where $Q = \int h\tilde{\mathbf{U}} \cdot \mathbf{n} dl$ is the total discharge; l is the length along the cross-sections (i.e., S1 and S2, which span across the whole domain) and $\mathbf{n}$ is the unit vector normal to the cross-section. Figure 11 shows the predicted errors ϵ_Q obtained by all models as a function of time. These results are markedly different from those previously presented for ϵ_h. Namely, all FV models display values of ϵ_Q that are substantially smaller than those predicted by PICN and PIFCN models. The maximum values of ϵ_Q for PICN and PIFCN are more than 50% and 70% of the benchmark (FV (5)) in S2, respectively. The possible reason behind these results might be that the water surface ($\eta = h + z$) presents much less spatial variation than Q in the domain. However, this hypothesis would need to be tested thoroughly in the future through a set of specifically designed case studies.

Figure 12 assesses the overall accuracy of temporal prediction for h of each model against the corresponding computational time, using the root-mean-square error metric $\mathcal{R}_h^t = \left(\sqrt{\sum (h_i - \tilde{h}_i)^2 / N_p^t}\right)\Big|_{(x,y)\in\mathcal{S}}$, where $\mathcal{S}$ denotes the corresponding cross-section and N_p^t is the number of collocation points in the testset. The best values of $\mathcal{R}_h^t$ (i.e., across all epochs) obtained from all PICN models are within the range of 0.22 m $< \mathcal{R}_h^t <$ 0.30 m (S1) and 0.26 m $< \mathcal{R}_h^t <$ 0.34 m (S2), while FV (10) delivered $\mathcal{R}_h^t$ = 0.29 m (S1) and 0.35 m (S2), and results from FV (25) and FV (50) were substantially less accurate. It is interesting to note that PINN models trained with coarse datasets (e.g., 200 m) do not necessarily deliver poorer accuracy compared to their fine resolution counterparts; this contrasts with what is typically observed in simulations with traditional numerical methods such as FV. Figure 12 also indicates that PICN models may offer improved depth predictions at lower cost than a FV model. For example, the accuracy of depth predictions by PICN (200) is better than the accuracy delivered by FV (10), while the computational cost is more than one order of magnitude lower. Overall, the PICN shows better h prediction performance than PIFCN and FV in terms of the speed-accuracy trade-off.

Figure 13 shows examples of flood depth maps at $t = 32$ hours obtained by the FV model at resolutions of 5 m and 25 m, along with those produced by PICN and PIFCN at 100 m resolved trainsets. As expected from the results presented in Figure 10, FV (25) overestimates h during the peak time (which also translates into a larger flooded area), while the opposite is observed for PICN (100) and PIFCN (100). Further spatial analysis can be seen in Appendix B.

4 Concluding remarks

In this paper, two physics-informed neuronal networks (PINNs) were developed to predict the evolution of free-surface flows typically modeled by the shallow water equations (SWEs). The PINN formulation eliminates the need for labeled data, which is typically required in supervised learning. This is achieved by defining a loss function that combines the SWEs, the boundary conditions (BCs) and initial conditions (ICs), allowing the trained PINN to serve as an alternative method for solving the SWEs. The two PINNs developed and tested here vary in their architecture and main features. The first is based on the fully-connected neural network (PIFCN), and the second on the convolutional neural network (PICN) approach.

Three test cases were used to assess the accuracy and computational performance of each model, including two idealized flow problems for which analytical solutions are available, and one simulation of a real-world flood event over a relatively large-scale and complex topography domain. In the idealized problems, the PICN and PIFCN predictions achieved higher accuracy (lower $\mathcal{R}_h$) than the Finite Volume (FV) solver employed for comparison. However, in these problems, PINNs generally took longer to reach the same prediction accuracy as the coarsely resolved FV model. For the real-world flooding problem, in general, PINNs were able to yield similarly accurate predictions of flow depths compared to finely resolved FV simulations. However, all FV models show much higher accuracy in their predictions of Q. For the spatial analysis of flow depths at the peak of the flood event, PINNs were able to produce flood maps with accuracy (relative to the benchmark finely resolved FV simulation) that is comparable to the results of FV models run at intermediate resolution (e.g., 25 m). Some of the PINN models (e.g., PICN at 100 and 200 m resolution) achieved the same level of accuracy as the 25 m resolution FV model at least one order of magnitude faster. In addition, the prediction capability of PINNs may be less affected by changes in grid resolution than the FV solver, which may represent important advantages in real-world applications where finely resolved topographic data may not always be available. At the same resolution (e.g., 10 m in Tests 1 and 2, or 50 m in Test 3), the training process of PICNs and PIFCNs with random ini-

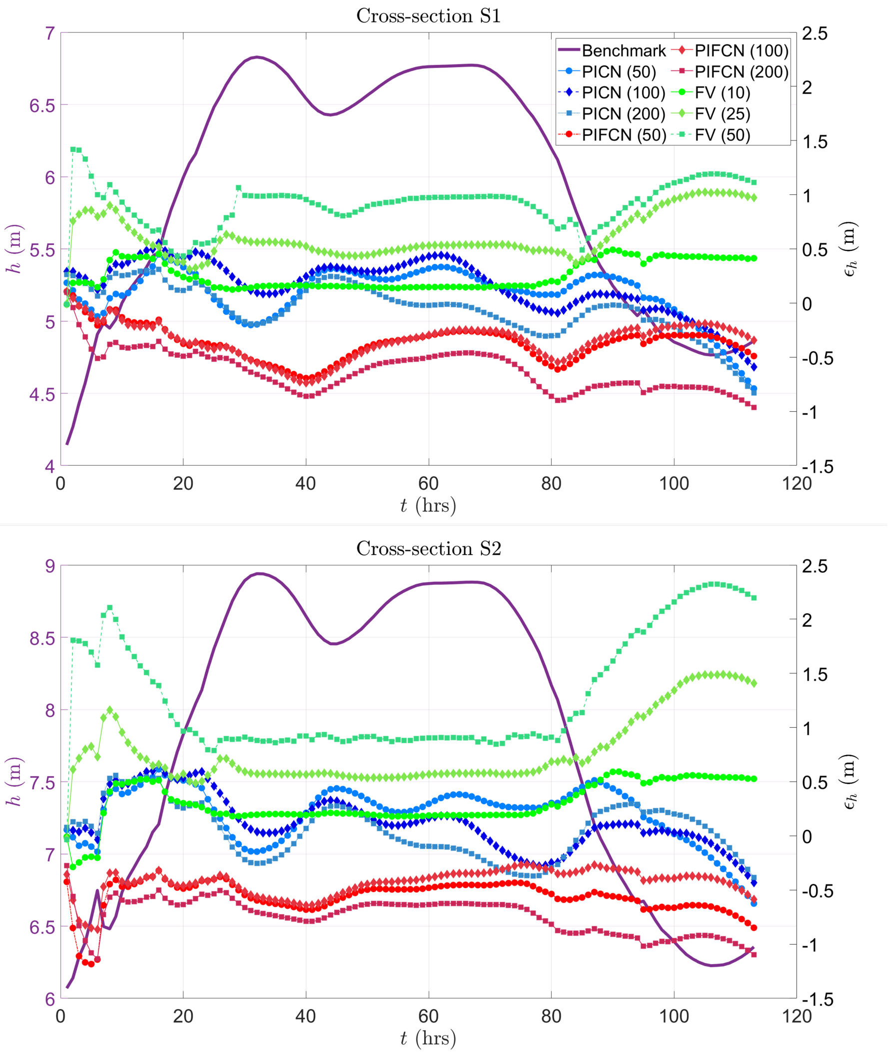

Figure 10. Test 3: Predicted water depths error ϵ_h (plotted against right $y-$axis) at cross-sections S1 (top) and S2 (bottom) of the main channel in the Tiber river. Benchmark solution (from a finely resolved FV simulation) shown by the purple line against the left $y-$axis.

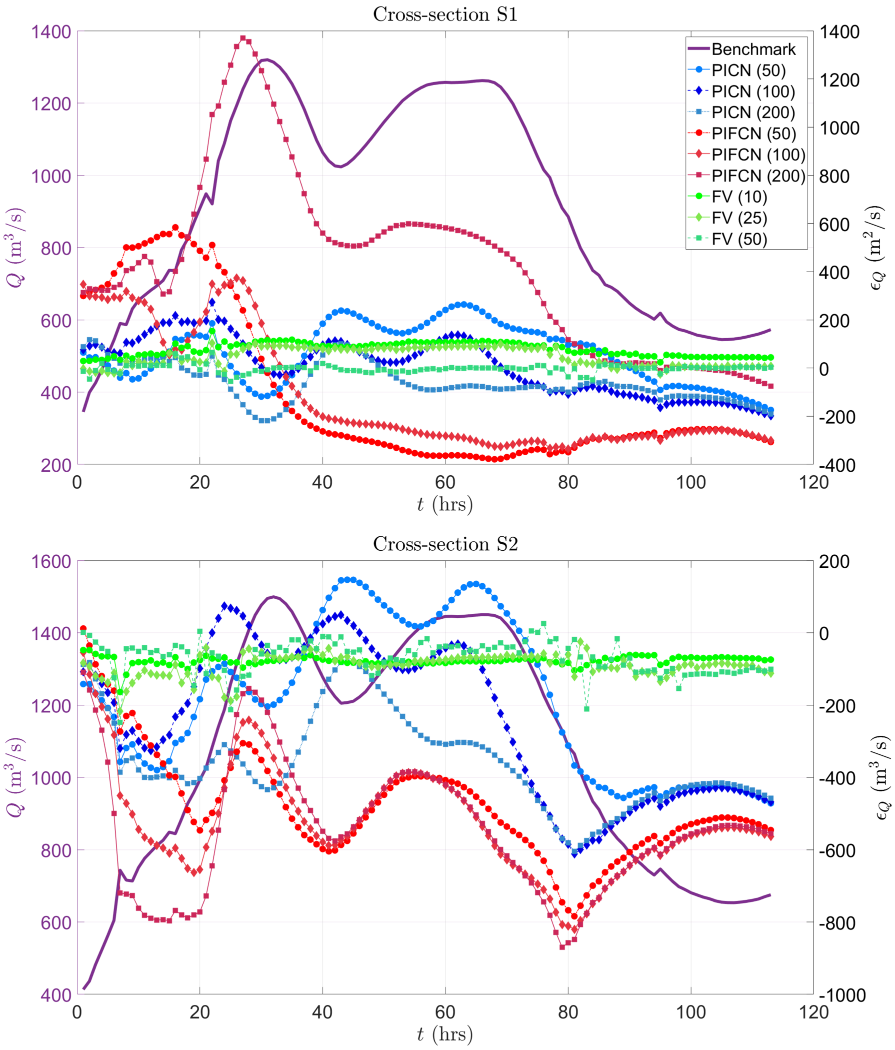

Figure 11. Test 3: Predicted water discharge error ϵ_Q (plotted against right $y-$axis) at cross-sections S1 (top) and S2 (bottom), which span across the whole domain. Benchmark solution (from a finely resolved FV simulation) shown by the purple line against the left $y-$axis.

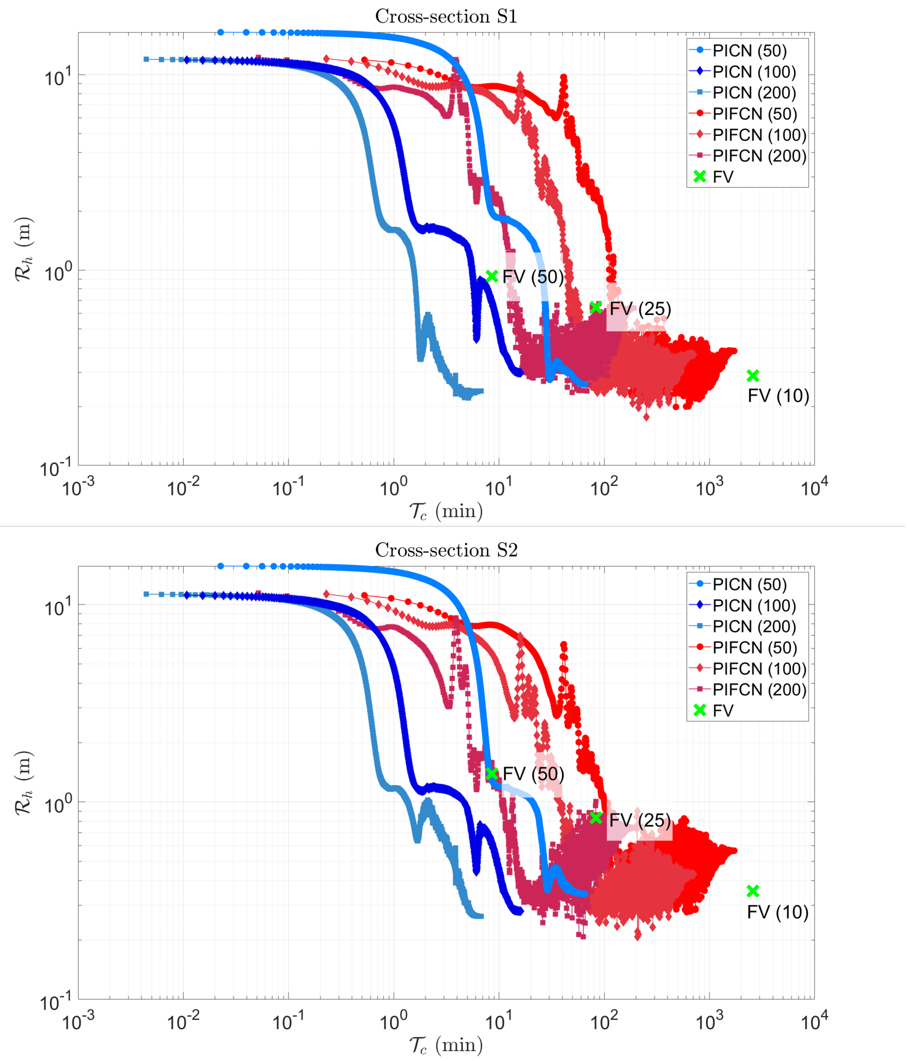

Figure 12. Test 3: $\mathcal{R}_h^t$ as a function of $\mathcal{T}_c$ (training time for PICN and PIFCN and run time for FV) at cross-sections S1 (top) and S2 (bottom) of the Tiber river; note that the right-most point of each cloud corresponds to the highest accuracy that any given PINN can achieve. The number in brackets represents the resolution (in meters) of the training data set (for PICN and PIFCN) or mesh (for the FV model). The benchmark results are those from the FV (5) simulation.

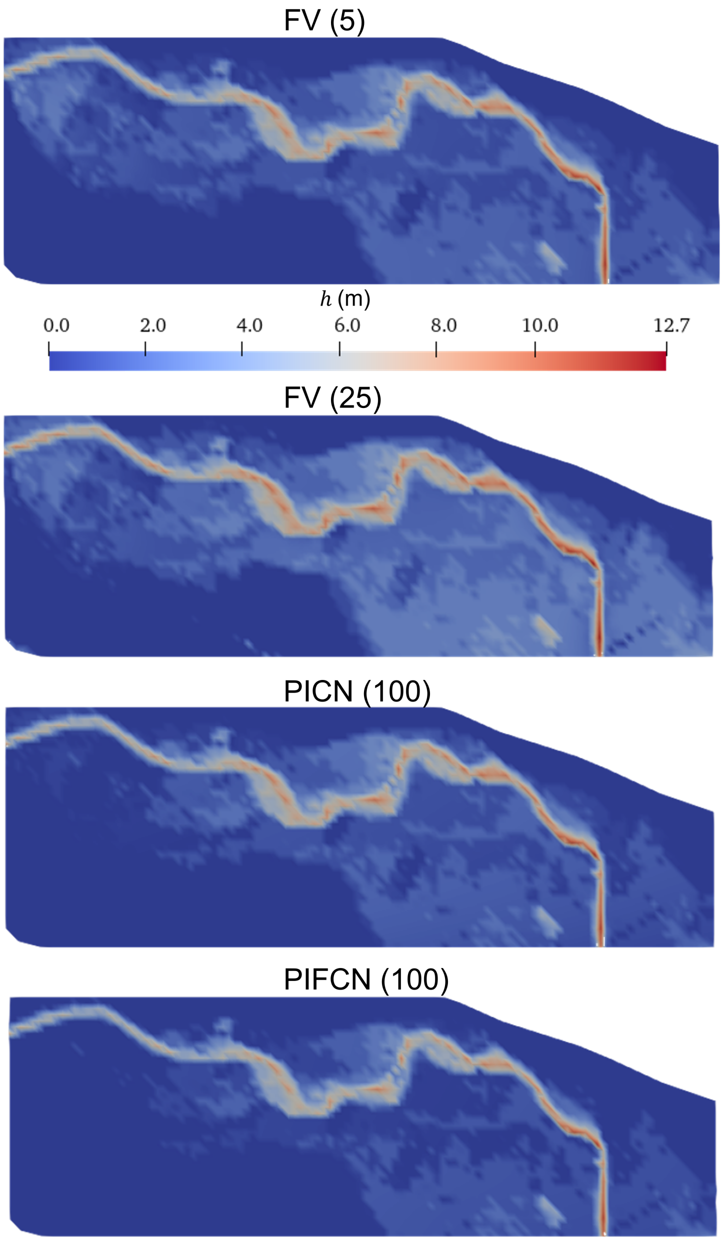

Figure 13. Examples of flood maps at time t = 32 hours produced by the FV model and PINNs. Note that FV (5) represents the benchmark results.

tialization of weights and biases takes longer than the run time of the FV model. Results show that, in most circumstances, PICNs usually exhibit better performance in terms of speed-accuracy trade-off than PIFCNs. However, more comparative tests between PICN and PIFCN are necessary before reaching general conclusions in this regard.

While the results in this paper may not suggest that PINNs can replace other well-established numerical techniques, they indicate that PINNs (and in particular PICNs) should be considered as an emerging technique that has the potential to deliver accurate and efficient solutions, and which should be further developed and assessed. Our results show that the approach might be particularly useful under certain circumstances which are challenging to conventional techniques. For example, in simulations performed at coarse resolutions (a typical case in real-world problems), PINN models may achieve a higher prediction accuracy with a lower computational cost than a FV solver. Since these techniques are still in their infancy, further research and development may enable PINNs to become a competitive alternative to simulate flow problems governed by the SWEs in the near future.

5 Open Research

The simulation data used for all three test cases in the study are available at the database from University of Southampton via `https://doi.org/10.5258/SOTON/D2645` with CC-BY license (Xin Qi, 2023).

Acknowledgments
The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

Alcrudo, F., & Garcia-Navarro, P. (1993). A high-resolution godunov-type scheme in finite volumes for the 2d shallow-water equations. *International Journal for Numerical Methods in Fluids*, *16*(6), 489–505.

Bale, D. S., Leveque, R. J., Mitran, S., & Rossmanith, J. A. (2003). A wave propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal on Scientific Computing*, *24*(3), 955–978.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, *18*, 1–43.

Bermúdez, M., Cea, L., & Puertas, J. (2019). A rapid flood inundation model for hazard mapping based on least squares support vector machine regression. *Journal of Flood Risk Management*, *12*(August 2018), 1–14. (Times cited: 6 Flooding papers) doi: 10.1111/jfr3.12522

Bernard, P.-E., Remacle, J.-F., Comblen, R., Legat, V., & Hillewaert, K. (2009). High-order discontinuous galerkin schemes on general 2d manifolds applied to the shallow water equations. *Journal of Computational Physics*, *228*(17), 6514–6535.

Bihlo, A., & Popovych, R. O. (2022). Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, *456*, 111024. doi: https://doi.org/10.1016/j.jcp.2022.111024

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*(6), 377-380. doi: https://doi.org/10.1016/0020-0190(87)90114-1

Botta, N., Klein, R., Langenberg, S., & Lützenkirchen, S. (2004). Well balanced finite volume methods for nearly hydrostatic flows. *Journal of Computational Physics*, *196*(2), 539–565.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, *143*(6).

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154.

Casulli, V. (1990). Semi-implicit finite difference methods for the two-dimensional shallow water equations. *Journal of Computational Physics*, *86*(1), 56–74.

Dawson, C., Westerink, J. J., Feyen, J. C., & Pothina, D. (2006). Continuous, discontinuous and coupled discontinuous–continuous galerkin finite element methods for the shallow water equations. *International Journal for Numerical Methods in Fluids*, *52*(1), 63–88.

de Almeida, G. A., & Bates, P. (2013). Applicability of the local inertial approximation of the shallow water equations to flood modeling. *Water Resources Research*, *49*(8), 4833–4844.

de Almeida, G. A., Bates, P., & Ozdemir, H. (2016). Modelling urban floods at submetre resolution: challenges or opportunities for flood risk management? *Journal of Flood Risk Management*, *11*, S855–S865.

Delestre, O., Lucas, C., Ksinant, P.-A., Darboux, F., Laguerre, C., Vo, T.-N.-T., ... Cordier, S. (2013). SWASHES: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, *72*(3), 269–300. doi: 10.1002/fld.3741

El Naqa, I., Li, R., & Murphy, M. J. (2015). *Machine learning in radiation oncology: theory and applications.* Springer.

Ferrari, A., & Vacondio, R. (2022). An augmented hllem ader numerical model parallel on gpu for the porous shallow water equations. *Computers & Fluids*, *238*, 105360.

Gao, H., Sun, L., & Wang, J.-X. (2021). Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, *428*, 110079.

Guo, L., Ye, S., Han, J., Zheng, H., Gao, H., Chen, D. Z., ... Wang, C. (2020). Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization. In *2020 ieee pacific visualization symposium (pacificvis)* (p. 71-80). doi: 10.1109/PacificVis48177.2020.8737

Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2020). A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751*.

Hanert, E., Le Roux, D. Y., Legat, V., & Deleersnijder, E. (2005). An efficient eulerian finite element method for the shallow water equations. *Ocean Modelling*, *10*(1-2), 115–136.

Haykin, S. (2009). *Neural networks and learning machines* (Third Edit ed.). Upper Saddle River: Prentice-Hall.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226–251.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hunter, N. M., Horritt, M. S., Bates, P. D., Wilson, M. D., & Werner, M. G. (2005). An adaptive time step solution for raster-based storage cell modelling of floodplain inundation. *Advances in water resources*, *28*(9), 975–991.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*.

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, *426*, 109951.

Juez, C., Murillo, J., & García-Navarro, P. (2014). A 2d weakly-coupled and efficient numerical model for transient shallow flow and movable bed. *Advances in Water Resources*, *71*, 93–109.

Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., & Pender, G. (2020). A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, *590*, 125481.

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, *34*.

Kurganov, A., & Levy, D. (2002). Central-upwind schemes for the saint-venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, *36*(3), 397–425.

Leandro, J., Chen, A., & Schumann, A. (2014). A 2d parallel diffusive wave model for floodplain inundation with variable time step (p-dwave). *Journal of Hydrology*, *517*, 250–259.

Leskens, J., Brugnach, M., Hoekstra, A. Y., & Schuurmans, W. (2014). Why are decisions in flood disaster management so poorly supported by information from flood models? *Environmental modelling & software*, *53*, 53–61.

LeVeque, R. J., George, D. L., & Berger, M. J. (2011). Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, *20*, 211–289.

Li, C., Han, Z., Li, Y., Li, M., Wang, W., Dou, J., … Chen, G. (2023). Physical information-fused deep learning model ensembled with a subregion-specific sampling method for predicting flood dynamics. *Journal of Hydrology*, *620*, 129465. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0022169423004079` doi: https://doi.org/10.1016/j.jhydrol.2023.129465

Li, J., Cao, Z., & Borthwick, A. G. (2022). Quantifying multiple uncertainties in modelling shallow water-sediment flows: A stochastic galerkin framework with haar wavelet expansion and an operator-splitting approach. *Applied Mathematical Modelling*, *106*, 259-275. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0307904X22000579` doi: https://doi.org/10.1016/j.apm.2022.01.032

Li, R., Lee, E., & Luo, T. (2021). Physics-informed neural networks for solving multiscale mode-resolved phonon boltzmann transport equation. *Materials Today Physics*, *19*, 100429.

Liang, Q. (2011). A structured but non-uniform cartesian grid-based model for the shallow water equations. *International Journal for Numerical Methods in Fluids*, *66*(5), 537–554.

Liang, Q., & Marche, F. (2009). Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in water resources*, *32*(6), 873–884.

Liu, Y., & Pender, G. (2015). A flood inundation modelling using v-support vector machine regression model. *Engineering Applications of Artificial Intelligence*, *46*, 223–231. (Times cited: 4 Flooding papers) doi: 10.1016/j.engappai.2015.09.014

Lynch, D. R., & Gray, W. G. (1979). A wave equation model for finite element tidal computations. *Computers & fluids*, *7*(3), 207–228.

MacDonald, I. (1996). *Analysis and computation of steady open channel flow* (Unpublished doctoral dissertation). Citeseer.

Mahesh, R. B., Leandro, J., & Lin, Q. (2022). Physics informed neural network for spatial-temporal flood forecasting. In *Climate change and water security* (pp. 77–91). Springer.

Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, *360*, 112789.

Marras, S., Kelly, J. F., Moragues, M., Müller, A., Kopera, M. A., Vázquez, M., ... Jorba, O. (2016). A review of element-based galerkin methods for numerical weather prediction: Finite elements, spectral elements, and discontinuous galerkin. *Archives of Computational Methods in Engineering*, *23*(4), 673–722.

Molls, T., & Chaudhry, M. H. (1995). Depth-averaged open-channel flow model. *Journal of Hydraulic Engineering*, *121*(6), 453–465.

Monnier, J., Couderc, F., Dartus, D., Larnier, K., Madec, R., & Vila, J.-P. (2016). 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.

Morales-Hernández, M., Petaccia, G., Brufau, P., & García-Navarro, P. (2016). Conservative 1d–2d coupled numerical strategies applied to river flooding: The tiber (rome). *Applied Mathematical Modelling*, *40*(3), 2087–2105.

Pang, G., Lu, L., & Karniadakis, G. E. (2019). fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, *41*(4), A2603–A2626.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.

Rastgoo, R., Kiani, K., Escalera, S., & Sabokrou, M. (2021). Sign language production: A review. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3451–3461).

Sanders, B. F., & Schubert, J. E. (2019). Primo: Parallel raster inundation model. *Advances in Water Resources*, *126*, 79–95.

Shamkhalchian, A., & de Almeida, G. A. (2021). Upscaling the shallow water equations for fast flood modelling. *Journal of Hydraulic Research*, *59*(5), 739–756.

Smith, L. N., & Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications* (Vol. 11006, p. 1100612).

Ştefănescu, R., Sandu, A., & Navon, I. M. (2014). Comparison of pod reduced order strategies for the nonlinear 2d shallow water equations. *International Journal for Numerical Methods in Fluids*, *76*(8), 497–521.

Sulavko, A. (2020). Bayes-minkowski measure and building on its basis immune machine learning algorithms for biometric facial identification. In *Journal of physics: Conference series* (Vol. 1546, p. 012103).

Sun, L., Gao, H., Pan, S., & Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, *361*, 112732.

Toro, E. F., & Garcia-Navarro, P. (2007). Godunov-type methods for free-surface shallow flows: A review. *Journal of Hydraulic Research*, *45*(6), 736–751.

Vlassis, N. N., & Sun, W. (2021). Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, *377*, 113695.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.

William, W., Ware, A., Basaza-Ejiri, A. H., & Obungoloch, J. (2018). A review of image analysis and machine learning techniques for automated cervical cancer screening from pap-smear images. *Computer methods and programs in biomedicine*, *164*, 15–22.

Wilson, M., Bates, P., Alsdorf, D., Forsberg, B., Horritt, M., Melack, J., ... Famiglietti, J. (2007). Modeling large-scale inundation of amazonian seasonally flooded wetlands. *Geophysical Research Letters*, *34*(15).

Xin Qi, S. M., Gustavo A. M. de Almeida. (2023). *Dataset supporting an article "physics informed neural networks for solving flow problems modeled by the shallow water equations"* [dataset]. University of SOuthampton. Retrieved from `https://doi.org/10.5258/SOTON/D2645` doi: 10.5258/SOTON/D2645

Yıldız, S., Goyal, P., Benner, P., & Karasözen, B. (2021). Learning reduced-order dynamics for parametrized shallow water equations from data. *International Journal for Numerical Methods in Fluids*, *93*(8), 2803–2821.

Yoon, T. H., & Kang, S.-K. (2004). Finite volume model for two-dimensional shallow water flows on unstructured grids. *Journal of Hydraulic Engineering*, *130*(7), 678–688.

Zhang, R., Liu, Y., & Sun, H. (2020). Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, *215*, 110704.

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, *394*, 56–81.

Appendix A PINN Design Experiments

This section illustrates the heuristic approach followed to determine the best possible design of the PINNs. We focus on Test 1, described in Section 3.1. All the PINNs shown in this section are trained from the same dataset resolved at 50 m resolution. Figures A1 and A2 show the accuracy ($\mathcal{R}_h$) of the PICNs and PIFCNs, respectively, as their architecture (number of layers and channels/neurons) is varied. In short, these figures show that it is difficult to conclude whether a single architecture can lead to significantly improved results, and we thus prioritize simplicity in our PINNs design. While this heuristic approach is, by definition, not guaranteed to find the optimal solution, it represents the summary of very many iterations. This holds for other tests and dataset resolutions considered in this study.

Similarly, we have tested three widely used activation functions: Relu, Sigmoid and Tanh (see Table A1). The chosen architecture for testing the PICN and PIFCN models is CNN-5-20 and FCNN-3(1000), respectively. For PICN, Sigmoid and Tanh display the same accuracy, while the result of the Relu-based PICN has higher errors. The PIFCN with Tanh yields better accuracy than using the other two activation functions. As a result, Tanh was chosen as the activation function to be employed in all PINNs discussed in this paper.

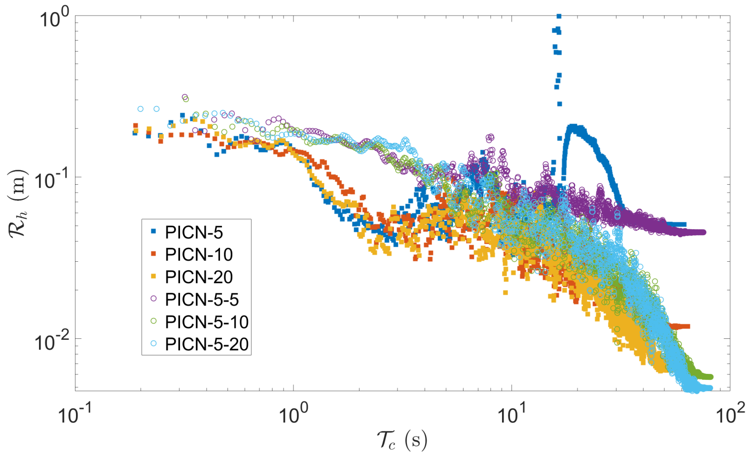

Figure A1. Comparison of PICNs with different architectures; the last hidden layer of all PICNs is one typical fully connected layer with 50 neurons. In the legend bar, the following format is adopted: PICN-X-Y, where the PICN has X channles in the first convolutional layer and Y channels in the second convolutional layer (thus, PICN-X denotes a network with one convolutional layer only).

Table A1. Results of water depth prediction by using Relu, Sigmoid and Tanh activation functions for PICN and PIFCN models. The trainset is a 50 m resolved dataset from Test 1; the evaluation metric is $\mathcal{R}_h$ and its unit is m.

Model	Relu	Sigmoid	Tanh
PICN	0.021	0.002	0.002
PIFCN	0.154	0.028	0.004

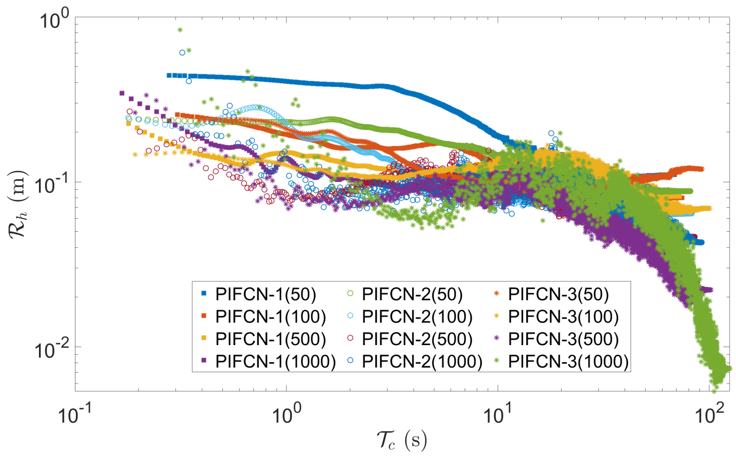

Figure A2. Comparison of PIFCNs with different architectures. In the legend bar, the following format is adopted: PIFCN-X(Y), where X denotes the number of hidden layers and Y is the number of neurons per layer.

Appendix B Further Spatial Analysis For Test 3

Table B1. Computation time and spatial prediction accuracy relative to benchmark simulation for the comparison. The unit for $\mathcal{R}_h^s$ is m, and the unit for $\mathcal{R}_{hu}^s$ and $\mathcal{R}_{hv}^s$ is m^{2}s^{-1}.

Model name	$\mathcal{T}_c$ (min)	t = 32 hours $\mathcal{R}_h^s$	$\mathcal{R}_{hu}^s$	$\mathcal{R}_{hv}^s$	t = 68 hours $\mathcal{R}_h^s$	$\mathcal{R}_{hu}^s$	$\mathcal{R}_{hv}^s$
PICN (50)	59.4	0.52	1.68	1.16	0.41	1.87	1.21
PICN (100)	15.3	0.40	1.72	1.18	0.37	1.92	1.20
PICN (200)	5.3	0.48	1.69	1.12	0.39	1.88	1.17
PIFCN (50)	504.9	0.59	2.21	1.32	0.52	2.21	1.28
PIFCN (100)	127.9	0.59	2.16	1.28	0.50	2.21	1.18
PIFCN (200)	30.2	0.63	2.27	1.21	0.47	2.16	1.15
FV (10)	2576.0	0.19	0.89	0.56	0.19	0.86	0.56
FV (25)	83.3	0.64	1.20	1.25	0.64	1.36	1.21
FV (50)	8.6	1.24	2.18	1.95	1.24	2.32	1.87

Table B1 summarizes the spatial prediction accuracy (i.e. $\mathcal{R}_h^s$, $\mathcal{R}_{hu}^s$, $\mathcal{R}_{hv}^s$) computed from a 50 m resolved set of points for each model at $t = 32$ and 68 hours, as well as their overall $\mathcal{T}_c$ (i.e. training time for PINN and computation time for FV). Among all the models, FV (10) and FV (50) achieve the highest and lowest accuracy, respectively. All PINNs present lower $\mathcal{R}_h^s$ than FV (25) and FV (50). On the other hand, FV (25) is more accurate than all PINNs in terms of hu prediction. PIFCN show a relatively similar value of $\mathcal{R}_{hu}^s$ to FV (50) at both time points. Moreover, the prediction accuracy of the PICNs and PIFCNs is less affected by the resolution of the input dataset than in the FV model. This last point may potentially be a main advantage of PINNs relative to conventional numerical methods in general, whose performance (numerical stability and accuracy) tends to be highly dependent on the mesh resolution.