Reduction of the Shallow Water System by an Error Aware POD-Neural Network Method: Application to Floodplain Dynamics

Mustapha Allabou, Robin Bouclier, Pierre-André Garambois, Jerome Monnier

Reduction of the Shallow Water System by an Error Aware POD-Neural Network Method: Application to Floodplain Dynamics.

M. Allabou[a], R. Bouclier[a,b], P.A. Garambois[c] and J. Monnier[a]

[a]*Institut de Mathématiques de Toulouse (IMT), Universiteé de Toulouse, CNRS-INSA-UT1-UT2-UPS, 1 R.3, Université Paul Sabatier, 118 Rte de Narbonne, Toulouse, 31400, France*

[b]*Institut Clément Ader (ICA), Universiteé de Toulouse, CNRS-INSA-ISAE-Mines Albi-UPS, Toulouse, France*

[c]*INRAE, UMR Recover, Aix-Marseille Université, Aix-en-Provence, France*

ARTICLE INFO

Keywords:
Non-intrusive reduced order models
Shallow water equations
Proper orthogonal decomposition
Artificial neural network
Error learning
River hydraulics

Abstract

In this study, we elaborate on and evaluate a new reduced basis method for model reduction of the shallow water equations using Proper Orthogonal Decomposition (POD) and artificial Neural Networks (NNs). The method begins with the POD technique to construct reduced bases from high-resolution solutions, followed by training two deep NNs to learn associated coefficients in the reduced bases. The approach follows an offline-online strategy: the POD reduced basis, along with the training of the NNs, is performed in an offline stage, and then the surrogate model can be used in an online stage for real-time predictions. The method takes into account the POD-based projection error, enabling the attainment of higher accuracy while preserving a limited number of POD modes, even in the delicate situation of convection-dominated flow problems. This point is crucial in our approach since it enables to limit the output dimension of the NNs, thus providing the opportunity to employ smaller NNs (with less parameters). This may lead to the utilization of potentially smaller datasets (i.e., the snapshots), better generalization of the obtained model, and simpler explainability. The process is non-intrusive: it does not require opening the high-resolution model code. The method is evaluated on a real-world test case aimed at simulating inundation of the Aude river (Southern France). The results show that the proposed method provides satisfying accuracy for the hydraulic variables (water elevation, discharge) compared to the reference high-resolution 2D shallow water model, with quite small dimension NNs. Overall, the method is promising, particularly for performing real-time simulations of large floodplains hydrodynamics.

1. Introduction

The Reduced Order Model (ROM) theory (see, *e.g.*, [24, 17, 5]) has gained significant attention in the field of computational sciences, offering techniques to tackle the computational challenges posed by complex systems. The goal is to simplify high-dimensional models while preserving their essential features. Model reduction methods can be broadly categorized into intrusive and non-intrusive approaches, each with its advantages and limitations, see e.g. [24, 17, 5] and references therein. In this paper, we adopt an offline-online non-intrusive strategy, applied to non-linear hyperbolic problems, specifically to the 2D Shallow Water Equations (SWEs). This PDEs system describes well the behavior of flood plain dynamics. Capturing complex features such as non linear travelling waves, over long time

*Corresponding author:
E-mail address: monnier@insa-toulouse.fr (J. Monnier)
allaboumustapha96@gmail.com (M. Allabou); bouclier@insa-toulouse.fr (R. Bouclier); pierre-andre.Garambois@inrae.fr (P.A. Garambois); monnier@insa-toulouse.fr (J. Monnier)
https://www.math.univ-toulouse.fr/~jmonnie/ (J. Monnier)
ORCID(s):

integration periods, using *e.g.* Finite Volume (FV) schemes is CPU-time consuming, which is impractical for real-time simulations or optimization and uncertainty quantification. To overcome these challenges, model reduction techniques offer an alternative approach by reducing the dimensionality of the problem while preserving the crucial dynamics. However, ROMs are particularly challenging in this situation of convection-dominated problems, see e.g. [24, 1, 4] and references therein. Applying ROM techniques aims to strike a balance between computational efficiency and accuracy. The reduction of these complex systems enables faster simulations, facilitates real-time decision-making processes, and opens avenues for optimization and uncertainty quantification tasks.

The literature addressing ROMs for non linear hyperbolic systems like SWEs is already quite large. For intrusive ROM approaches, let us mention for example [7, 28, 12, 29] and for non-intrusive ones [11, 30, 16], see Table 1. To reduce the 2D SWEs model (considered as non-parametrized), [28, 7] propose POD-DEIM (Proper Orthogonal Decomposition - Discrete Empirical Interpolation Method) approaches which are intrusive. [11] proposes a non-intrusive method based on POD and Radial Basis Functions (RBF) interpolation. Considering the 2D SWEs as a parametrized model, [**?** 29] propose intrusive approaches based on generalized polynomial chaos and POD-Galerkin, respectively. [30, 16] address non-intrusive POD-NN (Neural Network) based reduction methods to reduce parametrized SWEs. These POD-NN based methods rely on the usual offline-online strategy: a computationally expensive offline phase is performed beforehand to construct the reduced-order model, while the online phase efficiently computes the reduced system's response given a new parameter value, without further involving the full-order model (non-intrusive aspect). In these POD-NN methods, the POD enables to identify the dominant modes of the system, capturing the most significant features, while the NN serves as approximating the coefficients in the POD-based Reduced Basis (RB). The ROMs techniques are often limited to Galerkin numerical approaches, therefore in practice Finite Elements based numerical codes, see e.g. [24] and references therein. On the contrary, one of the extra interest of the POD-NN methods is to be applicable to Finite Volume based codes. Such a combination of POD and NN has been first introduced in [18], and later in [30]. In these pioneering studies, the method showed promising efficiency for relatively simple PDE-based models, but also for the steady incompressible Navier-Stokes equations modeling an academic driven cavity viscous flows. However, the application of the POD-NN method does not appear to have been explored extensively on complex real-like cases modeled by non-linear hyperbolic systems like the one addressed in the present study (a real-life flooding event), moreover solved by a finite volume solver. In such situations, the wave propagation dominant feature is challenging in terms of reduction, therefore requiring an important number of modes to accurately approximate the original system solutions.

Paper	Math. model, num. scheme	ROM method	Input variable/dimension	Test case	Non-intrusive?
Stefanescu et al., IJNMF 2014 [28]	2D SWEs (h,u,v), FD	Tensorial POD-DEIM	time / 1	Analytical solution	Intrusive
Steinstraesser et al., JCM 2021 [7]	2D SWEs (h,u,v), FV	POD-DEIM	time / 1	Analytical solution	Intrusive
Dutta et al., JCP 2021 [11]	2D SWEs (h,Q), FE	RBF-POD	time / 1	Kissimmee river (USA)	✓
El Moçayd et al., EMA 2017 [**?**]	1D SWEs (h,Q), FV	Generalized polynomial chaos	(Q,K_s) / 2 or 4	Garonne river (FR)	Intrusive
Strazzullo et al., JNMA 2022 [29]	2D SWEs, FE	Galerkin POD	Affinely parametrized (optimal control context) / 3	Analytical solution	Intrusive
Wang et al., JCP 2019 [30]	1D Euler equations, FV	POD-NN	$(t;\delta_1,\delta_2)$ (IC param.) / 3	Shock tube	✓
Ghorbanidehno et al., AWR 2020 [16]	2D SWEs (h,u,v), FE	POD-NN	Velocity data (Inverse pb context) / 11	Real rivers portions (USA)	✓

Table 1
A few references of ROM applied to SWEs (or Euler equations). (FV) Finite Volumes, (FE) Finite Elements, (FD) Finite Differences.

In this work, we extend the aforementioned POD-NN method by learning, in addition to the reduced (projected) solution, the projection error, through an additional NN. For non-linear convection-dominated or hyperbolic systems such as the one considered here, this enhancement turns out to be beneficial for constructing an accurate, robust and light surrogate model. Indeed, it allows for the correction of the reduced solution obtained from the standard POD-NN method while limiting the output dimension of the NNs, *i.e.*, the dimension of the reduced spaces. Obtaining smaller latent spaces, and therefore smaller dimensions for regression within the context of NNs is of great interest for several reasons. In particular, it prevents over-fitting during the learning phase, thereby promoting better generalization of the resulting model, and it facilitates explainability, see *e.g.*, [6, 22, 3]. Indeed, whenever possible, lower-dimensional and sparse representation systems are preferred for these reasons [22]. Ultimately, the proposed method preserves twofold advantages of the original POD-NN method: (i) it is generic, meaning it can be applied to any PDEs based problems solved by any discretization schemes (FV or FE for example); (ii) it is non-intrusive as it only requires to run, in a black-box fashion, the high-resolution model code multiple times in the offline phase. Due to the incorporation of this error-learning step, we refer to our approach as an Error-Aware POD-NN (EA-POD-NN) reduction method.

The paper is organized as follows. After this introduction, Section 2 introduces the considered parametrized 2D SWEs model, and its numerical resolution. Then, Section 3 presents, in a general manner, the developed EA-POD-NN methodology. It outlines the steps involved in the offline and online phases of the method and describes the

corresponding algorithms. Particular care is given in this section to highlight the limitations of the existing POD-NN approach, thereby motivating the integration of the projection error within the process. Section 4 presents the numerical experiments conducted to evaluate the performance and accuracy of the EA-POD-NN method. It discusses the selected SWEs problems, the setup of the simulations, and the comparison of results between the initial high-resolution model and the ROM using the proposed EA-POD-NN approach. Eventually, Section 5 provides a brief conclusion, summarizing the key findings and main contributions of this study, and identifying potential areas for future research based on the proposed approach.

2. The μ-parametrized model and the reference high-resolution solutions

In this section, first, the general concept of High-Resolution (HR) solutions corresponding to a given set of input parameters is recalled. Next, the flow model considered in our study is presented, followed by a discussion of the numerical scheme and computational software employed to solve it.

2.1. Basic principle of the high-resolution solutions generation

The reduction method developed in this study can be formally applied to any μ-parametrized non-linear PDE-based model with the input parameter $\mu = (\mu_1, \dots, \mu_{N_\mu}) \in \mathbb{R}^{N_\mu}$, which typically represents a set of physical parameters in the bulk and/or boundary condition parameters. From a reduction point of view, the important characteristic of μ is its dimension N_μ: N_μ has to be quite small, say $N_\mu = \mathcal{O}(10)$ at most, otherwise an upstream reduction of μ should be considered in a pre-processing step *e.g.*, based on the use of autoencoder [16].

More precisely, we set $\mathcal{H}$ such that $\mu \in \mathcal{H}$ with $\dim(\mathcal{H}) = N_\mu$. A set of N_s parameters are fixed by sampling in some way the parameter space $\mathcal{H}$. We obtain the reference parameter set $P_s = \{\mu_s\}_{s=1}^{N_s}$, $P_s \in (\mathcal{H})^{N_s}$. Next, the μ-parametrized model, denoted as $\mathcal{M}_\mu$, is employed to generate the N_s corresponding vector solutions $\mathbf{u}_h$: this is the classically called HR solutions set, also called the *snapshots* set. These snapshots $\mathbf{u}_h$ are stored in the so-called snapshot matrix:

$$\mathbf{S} = \left[\mathbf{u}_h(\mu_1) | \dots | \mathbf{u}_h(\mu_{N_s})\right]. \tag{1}$$

2.2. The μ-parametrized mathematical and numerical flow model

In the developed application, the considered model $\mathcal{M}_\mu$ relies on the 2D SWEs, employed in particular to simulate river and floodplain flows dynamics. In this context, the parameter μ could be related to boundary conditions, to the initial conditions, or to the friction coefficient, etc.

We denote by $h(x,t)$ (m) the water depth and by $\mathbf{q}(x,t)$ (m^2s^{-1}) the discharge. We have $\mathbf{q} = h\mathbf{U}$, where $\mathbf{U} = (v_x, v_y)^T$ (ms^{-1}) denotes the depth-averaged velocity. For a given computational domain $\Omega \subset \mathbb{R}^2$ and a time interval $[0,T]$, the 2D SWEs model is considered in its conservative form as:

$$(\mathcal{M}_\mu)\begin{cases} \partial_t h + div(\mathbf{q}) & = 0 & \text{in } \Omega \times]0,T] \\ \partial_t \mathbf{q} + div\left(\dfrac{\mathbf{q}\otimes\mathbf{q}}{h} + g\dfrac{h^2}{2}Id\right) & = S_g(\mu;h) + S_f(\mu;h,\mathbf{q}) & \text{in } \Omega \times]0,T] \\ \text{plus Initial Conditions,} & \text{plus Boundary Conditions (B.C.)}(\mu) & \end{cases} \tag{2}$$

In the above equation, $S_g(\mu;h)$ is the gravity source term: $S_g(\mu;h) = -gh\nabla z_b$, with g the gravity magnitude and z_b the bed elevation; and $S_f(\mu;h,\mathbf{q})$ is the friction source term: $S_f(\mu;h,\mathbf{q}) = -g\dfrac{n^2\,\|\mathbf{U}\|}{h^{7/3}}\mathbf{q}$, with n the Manning-Strickler friction coefficient. The B.C. is a mix of conditions necessary for real-world applications. The domain boundary is decomposed as $\partial\Omega = \Gamma_{in} \cup \Gamma_{wall} \cup \Gamma_{out}$. At the N_{in} inflow boundaries ($\Gamma_{in} = \cup_k^{Nin}\Gamma_{in,k}$), a discharge time series $Q_{in,k}(\mu;t)$ is imposed on $\Gamma_{in,k}$ for $t \in]0,T]$. On Γ_{wall}, the standard non-penetration conditions are imposed, i.e. $\boldsymbol{q}.\boldsymbol{n} = 0$ with $\boldsymbol{n}$ the normal vector to the boundary. At the outflow boundary Γ_{out}, a water elevation H_{out} is prescribed. The initial conditions (IC) consists in a steady state solution obtained for the inflow discharge value corresponding to the begining of the hydrograph that will be used for the flood simulation, starting from no water over Ω. We refer to [27] for more technical details.

For the forthcoming presentation of our reduction method in a general manner in Section 3, we denote by $u_h = (h,\mathbf{q}) = (h,h\mathbf{U}) = (h,q_x,q_y)$ the HR solution field of (2).

In this work, our primary focus is on the most important input parameter for operational users; that is, the inflow discharge signal at upstream (inflow B.C.). Consequently, in the numerical applications, the μ parameter will be related to the discharge functions $Q_{in,k}(\mu,t)$. Moreover, as we are dealing with a time-dependent problem, we include the time as a parameter (i.e., into μ) as it represents a dimension in the space $\mathcal{H}$. We will therefore introduce t_{sn} that refers to the time sampling of the snapshots.

Given a parameter value μ, the 2D SW system $\mathcal{M}_\mu$ is here numerically solved by the Finite Volume (FV) method implemented into the open-source computational software DassFlow 2D [27, 21]. The solver relies on a well-balanced Godunov-type scheme, using the hydrostatic reconstruction proposed by [2], and the explicit Euler time scheme with adaptative time step to satisfy a target CFL (0.5 here). The mesh can consist in a mix of triangles and quadrangles. The DassFlow 2D kernel code is written in Fortran 90, using MPI library. Moreover, the Fortran computational kernel is wrapped in Python. This allows for seamless integration of these physics-based computations with other Python libraries. These libraries can serve various purposes, in particular Python libraries like PyTorch for deep learning.

3. The Error-Aware POD-NN reduction method

Let us now detail the proposed reduction method that allows to construct an efficient and accurate surrogate model, here applied to the 2D SWEs-based model $\mathcal{M}_\mu$. The presentation in this section is crafted in a general manner as the proposed approach is generic and, consequently, can be applied in various contexts. Specifically, the presentation follows the different aspects of the method, namely (i) the projection onto a RB, (ii) the POD technique for constructing the RB, (iii) the learning of coefficients of the reduced solution using a deep NN, and (iv) the involvement of the projection error.

3.1. Reduced basis and projection

Our method being based on a RB approach, let us start by properly defining the complete space, the reduced space through its RB, and the relations between both. For illustration purpose, one can refer to Fig. 1 which will be detailed in the following.

Let us denote V_h as the initial complete functional space that contains the discrete solution computed using the HR model. Additionally, we introduce the basis of shape functions $\Phi(x) = \{\varphi_i(x)\}_{i=1}^{N_h}$ that generates V_h, meaning that $V_h = \text{span}\ \Phi(x)$ with $dim(V_h) = N_h$. Then, let us define the reduced space $V_{rb} \subset V_h$ such that $V_{rb} = \text{span}\ \Xi(x)$, where $\Xi(x) = \{\xi_n(x)\}_{n=1}^{N_{rb}}$ constitutes the RB. Therefore, we have $dim(V_{rb}) = N_{rb}$, and we expect $N_{rb} << N_h$. Ultimately, the initial complete solution $u_h(\mu; x)$ and the reduced one $u_{rb}(\mu; x)$ can be written in algebraic form as follows:

$$u_h(\mu; x) = \mathbf{\Phi}(x)^T \mathbf{u}_h(\mu) \quad ; \quad u_{rb}(\mu; x) = \mathbf{\Xi}(x)^T \mathbf{u}_{rb}(\mu). \tag{3}$$

Here, $\mathbf{\Phi}(x)$ (respectively, $\mathbf{\Xi}(x)$) and $\mathbf{u}_h(\mu)$ (respectively, $\mathbf{u}_{rb}(\mu)$) are vectors in $\mathbb{R}^{N_h}$ (resp. in $\mathbb{R}^{N_{rb}}$) that collect the shape functions and associated coefficients. In the remainder of this section, since the spaces V_h and V_{rb} remain constant for all values of $\mu \in \mathcal{H}$, we will omit the dependence of the solutions on μ for more simplicity in the notations.

Remark 1. Note that the definition of V_h encompasses all common discretization methods encountered in scientific computing, such as FE, FV, Discontinuous Galerkin, and so on. For instance, $\Phi(x)$ simply represents the nodal Lagrange shape functions in the FE context or corresponds to constant functions per cell in a FV framework. In our application, we will focus on the FV method, but it is worth noting that the proposed approach is applicable to other contexts, as will be underlined later.

As a first relation between the two spaces, we can construct the matrix $\mathbf{B}_{rb} \in \mathbb{R}^{N_h \times N_{rb}}$, which encodes the change of variables from the RB $\Xi(x)$ to the complete one $\Phi(x)$. More precisely, it is defined as:

$$\mathbf{B}_{rb} = \left[\xi_1 | \dots | \xi_{N_{rb}}\right], \tag{4}$$

where $\boldsymbol{\xi}_n$ denotes the vector gathering the coefficients of function $\xi_n(x)$ in the complete basis $\Phi(x)$. Thus, we can write:

$$\boldsymbol{\Xi}(x) = \mathbf{B}_{rb}^T \boldsymbol{\Phi}(x), \tag{5}$$

and it follows that any reduced solution $u_{rb}(x) \in V_{rb}$ can be represented in the complete basis $\Phi(x)$ as:

$$u_{rb}(x) = \boldsymbol{\Xi}(x)^T \mathbf{u}_{rb} = \boldsymbol{\Phi}(x)^T \mathbf{B}_{rb} \mathbf{u}_{rb} = \boldsymbol{\Phi}(x)^T \mathbf{u}_{rb}^{N_h}. \tag{6}$$

where

$$\mathbf{u}_{rb}^{N_h} = \mathbf{B}_{rb} \mathbf{u}_{rb} \in \mathbb{R}^{N_h} \tag{7}$$

refers to the vector that contains the coefficients of the reduced solution $u_{rb}(x)$ in the complete basis $\Phi(x)$.

Next, as a second relation, we can express the orthogonal projection from the complete space V_h onto the reduced space V_{rb}. In our application, since the numerical solution is discretized using piecewise constant functions (see Remark 1), we will simply consider the L^2-scalar product. With this in mind, the projection matrix $\mathbf{P}_{rb} \in \mathbb{R}^{N_h \times N_h}$ is such that:

$$\forall \mathbf{u}_h \in \mathbb{R}^{N_h}, \quad \mathbf{P}_{rb} \mathbf{u}_h = \mathbf{B}_{rb}\, \mathbf{B}_{rb}^T \mathbf{u}_h = \mathbf{B}_{rb} \mathbf{u}_h^{N_{rb}}, \tag{8}$$

where

$$\mathbf{u}_h^{N_{rb}} = \mathbf{B}_{rb}^T \mathbf{u}_h \in \mathbb{R}^{N_{rb}} \tag{9}$$

refers to the vector that collects the coefficients in the RB $\Xi(x)$ of the projection of $u_h(x)$ onto the reduced space V_{rb} (see Fig. 1). With all this in hand, we can eventually define the projection error of $u_h(x)$ onto V_{rb} in the algebraic form as follows:

$$\mathbf{e}_h = \mathbf{u}_h - \mathbf{P}_{rb} \mathbf{u}_h = \mathbf{u}_h - \mathbf{B}_{rb} \mathbf{u}_h^{N_{rb}} = \mathbf{u}_h - \mathbf{B}_{rb} \mathbf{B}_{rb}^T \mathbf{u}_h, \tag{10}$$

where $\mathbf{e}_h \in \mathbb{R}^{N_h}$ (see Fig. 1 again).

The key point now is to build a RB, or equivalently the matrix $\mathbf{B}_{rb}$, such that N_{rb} is small and $\mathbf{P}_{rb} \mathbf{u}_h \approx \mathbf{u}_h$, meaning $\mathbf{e}_h \approx 0$ for any $\mu \in \mathcal{H}$. This will be performed with the classical POD method, which is outlined in next Section.

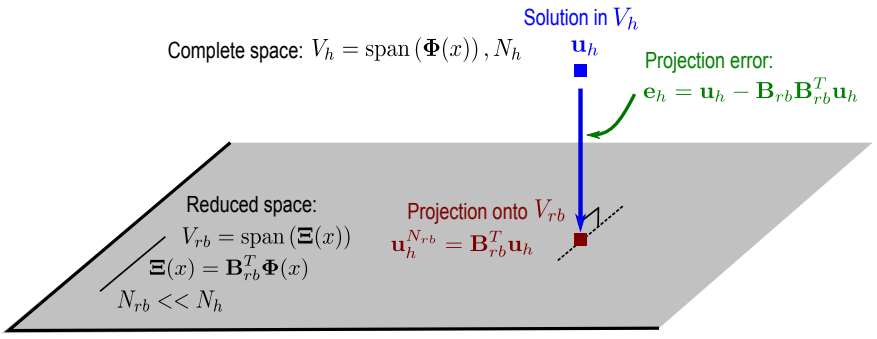

Figure 1: Schematic representation of the different spaces: V_h is the initial complete space (associated basis $\mathbf{\Phi}(x)$); $V_{rb} \subset V_h$ is the reduced space (associated basis $\mathbf{\Xi}(x)$); N_h and N_{rb} are the dimensions of V_h and V_{rb}, respectively; B_{rb} is the change of variables matrix; $\mathbf{u}_h^{N_{rb}}$ is the projection vector in $\mathbf{\Xi}(x)$ of $\mathbf{u}_h$ onto V_{rb}; and $\mathbf{e}_h$ is the corresponding projection error (expressed in basis $\mathbf{\Phi}(x)$).

3.2. The POD reduced basis

The POD method is far from recent (see, *e.g.*, [20]) and is now widely used in many fields of scientific computing (refer to the following books [24, 17, 5] to name a few). The interest of the POD strategy is that it leads to a RB that is optimal in a chosen norm with respect to a collection of complete solutions.

Starting with the snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$ (See Eq. (1)), the principle of the POD method is to perform a Singular Value Decomposition (SVD) and to retain the most significant left singular vectors. These retained singular vectors constitute the POD modes that form the RB. For the SVD, we build the correlation matrix $\mathbf{C}$ as follows:

$$\mathbf{C} = \mathbf{S}\,\mathbf{S}^T. \tag{11}$$

Note that, again, we use here the L^2-scalar product due to the regularity of our discrete fields (see Remark 1). The next step is to compute the eigenvalues $\{\lambda_i\}_{i=1}^{N_h}$ ($\lambda_i \in \mathbb{R}+$) and eigenvectors $\{\boldsymbol{\xi}_i\}_{i=1}^{N_h}$ ($\boldsymbol{\xi}_i \in \mathbb{R}^{N_h}$ and $\|\boldsymbol{\xi}_i\|_{L^2} = 1$) of $\mathbf{C}$, which correspond, respectively, to the squares of the singular values and the left singular vectors of $\mathbf{S}$: $\mathbf{C}\,\boldsymbol{\xi}_i = \lambda_i\,\boldsymbol{\xi}_i, \quad 1 \leq i \leq N_h$. Finally, $\mathbf{B}_{rb}$ (introduced in Eq. (4)) is simply constructed with the N_{rb} vectors $\{\boldsymbol{\xi}_i\}_{i=1}^{N_{rb}}$ associated with the N_{rb} largest $\{\lambda_i\}_{i=1}^{N_{rb}}$.

Let us observe at this stage that $\mathbf{B}_{rb}\mathbf{B}_{rb}^T \neq \mathbf{I}_{N_h}$ since $\mathbf{P}_{rb}\mathbf{u}_h = \mathbf{B}_{rb}\mathbf{B}_{rb}^T\mathbf{u}_h$ only approximates $\mathbf{u}_h$ due to the fact that $V_{rb} \subset V_h$ (remind Eq. (8) and Fig. 1). An error estimation is actually available in this context. More precisely, let us define the space of semi-orthonormal matrices of dimension N_{rb}: $\mathcal{B}_{N_{rb}}^{\perp} = \{\mathbf{B} \in \mathbb{R}^{N_h \times N_{rb}},\ \mathbf{B}^T\mathbf{B} = \mathbf{I}_{N_{rb}}\}$. The interest of the POD matrix $\mathbf{B}_{rb}$ is that it minimizes, over all possible N_{rb}-dimensional semi-orthonormal matrices $\mathbf{B} \in \mathcal{B}_{N_{rb}}^{\perp}$, the projection error (10) onto the whole set of snapshots. Mathematically, this reads:

$$\sum_{s=1}^{N_s} \|\mathbf{u}_h(\mu_s) - \mathbf{B}_{rb}\mathbf{B}_{rb}^T\mathbf{u}_h(\mu_s)\|_{L^2}^2 = \min_{\mathbf{B}\in\mathcal{B}_{N_{rb}}^{\perp}} \sum_{s=1}^{N_s} \|\mathbf{u}_h(\mu_s) - \mathbf{B}^T\mathbf{B}\mathbf{u}_h(\mu_s)\|_{L^2}^2 = \sum_{s=N_{rb}+1}^{N_h} \lambda_s, \tag{12}$$

which result from the Schmidt-Eckart-Young theorem, see, *e.g.*, [24].

Benefiting from the above error estimation, the dimension of the RB can be fairly defined. In practice, N_{rb} is defined as the minimal value such that:

$$R(N_{rb}) = \frac{\sum_{i=1}^{N_{rb}} \lambda_i}{\sum_{i=1}^{N_h} \lambda_i} \geq \left(1 - \epsilon_{POD}^2\right), \tag{13}$$

meaning that the signal of the snapshots captured by the most significant N_{rb} POD modes is conserved (in the L^2-norm here) within $(100 - \epsilon_{POD})\%$. The chosen value for ϵ_{POD} will be specified and accounted for in Section 4.

The construction of the RB is obviously performed during the offline phase. This actually constitutes the initial step of the present reduction method (refer to Algorithm 1 for an overview of the complete offline phase carried out in this work). Once again, it is worth noting that the strategy employed so far can be applied to any μ-parametrized model, solved by any discretization method. It only requires the N_s HR solutions to form the snapshot matrix $\mathbf{S}$.

3.3. The POD-NN method

The goal now is to compute, for a new set of parameters $\mu_{new} \in \mathcal{H}$ which does not belong to P_s, a good approximation of the complete solution $u_h(\mu_{new}; x)$ (or equivalently of the complete solution vector $\mathbf{u}_h(\mu_{new})$) in the reduced space V_{rb}. To achieve this, the novel method proposed in the present study relies on an enriched version of the POD-NN method first introduced in [18]. In the present section, we detail the original POD-NN method.

The POD-NN method basic idea is to start from the projection defined in Eq. (8) (see also Fig. 1) and to learn, using a deep NN, the coefficients in the RB $\Xi(x)$ of the projection of $\mathbf{u}_h(\mu_{new})$ onto V_{rb}; that is, to learn $\mathbf{u}_h^{Nrb}(\mu_{new}) \in \mathbb{R}^{N_{rb}}$, see Eq. (9). More precisely, let us introduce the following mapping:

$$\begin{array}{rccl} F_1 & : & \mathcal{H} \subset \mathbb{R}^{N_\mu} & \rightarrow & \mathbb{R}^{N_{rb}} \\ & & \mu & \mapsto & \mathbf{u}_h^{N_{rb}}(\mu) = \mathbf{B}_{rb}^T \mathbf{u}_h(\mu). \end{array} \tag{14}$$

It maps each set of input parameters $\mu \in \mathcal{H}$ to the coefficients $\mathbf{B}_{rb}^T \mathbf{u}_h(\mu)$ in the RB $\Xi(x)$ of the projection of the complete solution $\mathbf{u}_h(\mu)$ onto V_{rb}.

In the offline phase, this non-linear mapping F_1 is approximated by employing an artificial NN, particularly a multi-layer perceptron (see Fig. 2). In practice, training the aforementioned NN can be performed using supervised learning based on input-output pairs obtained from the snapshots $\mathbf{S}$ (see Eq. (1)). Denoting by $\mathcal{N}_1(\boldsymbol{\theta}_1)(\mu)$ the considered NN with associated parameters (weights and biases) gathered in vector $\boldsymbol{\theta}_1$, this leads to minimizing the following loss

function for the training:

$$\mathcal{J}_1(\boldsymbol{\theta}_1) = \frac{1}{N_s}\sum_{s=1}^{N_s} \|\mathcal{N}_1(\boldsymbol{\theta}_1)(\mu_s) - \mathbf{B}_{rb}^T \mathbf{u}_h(\mu_s)\|_{L^2}^2, \tag{15}$$

Training $\mathcal{N}_1$ constitutes Step 8 of our offline phase, see Algorithm 1.

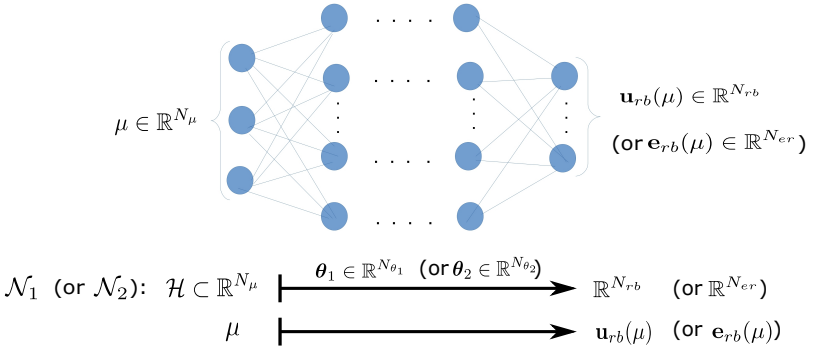

Figure 2: NN approximation of the map F_1, see Eq. (14) (or F_2, see Eq. (19)). Note that N_μ, N_{rb} and N_{er} are on the order of 10. In practice the hidden layers are taken with the same number of neurons and the latter is larger than the numbers of input and output neurons (see Section 4 for the exact chosen values).

Remark 2. It is important to notice at this stage that the reduction step, which transforms $\mathbf{u}_h \in \mathbb{R}^{N_h}$ into $\mathbf{u}_h^{N_{rb}} \in \mathbb{R}^{N_{rb}}$ with $N_{rb} << N_h$ is crucial in the process. This step allows to obtain only a few ouput neurons for the NN (in our application $N_{rb} = \mathcal{O}(10)$, see Section 4), making it possible to consider a deep NN with a number of parameters N_{θ_1} (*i.e.*, $\boldsymbol{\theta}_1 \in \mathbb{R}^{N_{\theta_1}}$) reasonable compared to the number of data (*i.e.*, the N_s snapshots) during training. Of course, for the same reason, the number of input neurons of the network must be low, *i.e.*, the number of physical parameters N_μ ($\mu \in \mathbb{R}^{N_\mu}$) must be small, which is also a limitation of more conventional reduced methods that require the computation of snapshots beforehand (the larger N_μ, the larger the number of snapshots).

Next, during the online phase, all that is required is to perform a forward pass of the NN $\mathcal{N}_1$ given a new (unseen) set of physical parameters μ_{new} and the optimized $\boldsymbol{\theta}_1$ obtained after training. This process computes the reduced coefficients $\mathbf{u}_{rb}(\mu_{new})$ and is known to be very fast, as the forward evaluation of a NN simply involves affine and element-wise activation transformations. This makes the online phase achievable in real-time. Finally, the reduced solution can be written in the original basis $\Phi(x)$ using Eq. (7): $\mathbf{u}_{rb}^{N_h}(\mu_{new}) = \mathbf{B}_{rb}\mathbf{u}_{rb}(\mu_{new}) \in \mathbb{R}^{N_h}$ (which is also very fast to perform). In particular, this allows the utilization of existing subroutines from the initial computational code for post-processing. The entire online phase performed in this work is summarized in Algorithm 2. The procedures mentioned above correspond to Steps 1 and 2 of the online phase.

Remark 3. In the proposed method, we ultimately only need to utilize the initial computational code associated with the HR model $\mathcal{M}_\mu$ in a black-box fashion to run it multiple times (N_s times) with different parameters. Subsequently, the method is entirely independent of the considered HR model $\mathcal{M}_\mu$ and underlying code since, in particular, the online phase is decoupled from $\mathcal{M}_\mu$. Consequently, the method has the twofold advantages of being generic (it can be applied to any type of problem solved by any discretization method), and non-intrusive (we never need to delve into the code itself). Here, we leverage on these properties to apply the strategy to a non-linear problem with a non-affine dependence on the parameters and where a Galerkin approach is not used at the discretization level, which is a well-known scenario in which traditional reduced-order methods fail.

3.4. The Error-Aware POD-NN method

The POD-NN method considered so far requires the ability to strongly reduce the dimension of the problem. More precisely, we need to have $N_{rb} = \mathcal{O}(10)$, so that it is possible to employ a deep NN with a number of parameters N_{θ_1} reasonable compared to the number snapshots N_s (see Fig. 3 that illustrates the different constraints). Therefore, in some situations, especially in the case of convection-dominated flow problems, it is requested to further enhance the accuracy of the method to mitigate the Kolmogorov barrier to reducibility (see, *e.g.*, [4] for the concept of reduction complexity). To achieve this in a straightforward manner with minimal computational overhead, we propose to integrate the ROM error estimation. Note that this generic idea has been developed, *e.g.*, in [31] within the framework of Gaussian processes. The idea is here to learn, in addition to the projection of $\mathbf{u}_h(\mu)$ onto V_{rb}, the projection error introduced in Eq. (10) (see also Fig. 1):

$$\mathbf{e}_h(\mu) = \mathbf{u}_h(\mu) - \mathbf{B}_{rb}\mathbf{B}_{rb}^T\mathbf{u}_h(\mu), \tag{16}$$

and to benefit from the latter to correct the reduced solution obtained so far.

For this purpose, we follow the same procedure used to construct the reduced solution, but this time starting with the snapshot error matrix $\mathbf{E} \in \mathbb{R}^{N_h \times N_s}$:

$$\mathbf{E} = \left[\mathbf{e}_h(\mu_1) | \dots | \mathbf{e}_h(\mu_{N_s})\right], \tag{17}$$

which collects the projection errors of the already computed snapshots $\{\mathbf{u}_h(\mu_s)\}_{s=1}^{N_s}$ (see Eq. (1)). Consequently, we first apply the POD strategy (same operations as in Section 3) to compute the corresponding error RB $\{\zeta_n(x)\}_{n=1}^{N_{er}}$ (where we again anticipate $N_{er} << N_h$) or, in practice, the error RB matrix $\mathbf{B}_{er} \in \mathbb{R}^{N_h \times N_{er}}$:

$$\mathbf{B}_{er} = \left[\zeta_1 | \dots | \zeta_{N_{er}}\right]. \tag{18}$$

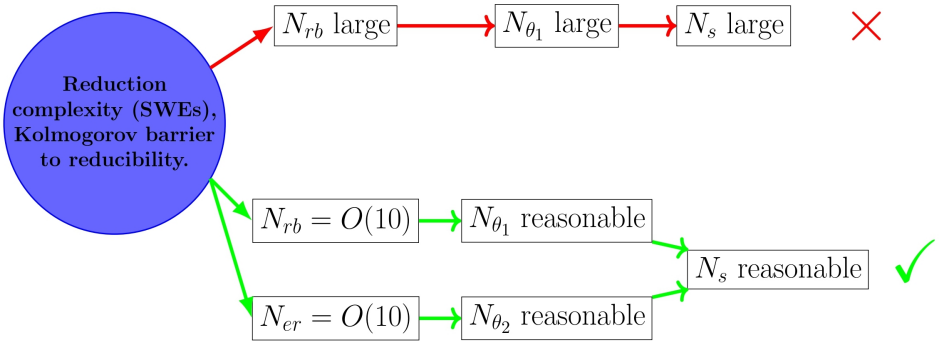

Figure 3: Due to a high reduction complexity for convection-dominated problems, (see, *e.g.*, [4] for the concept of the Kolmogorov barrier to reducibility), it appears beneficial to complement the basic POD-NN method with the projection error learning to achieve an accurate solution. Indeed, without it (see red path), a large N_{rb} is needed, which implies a large number of output neurons and therefore a large N_{θ_1}, requiring potentially a large amount of data for training, i.e., a large N_s. Conversely, with the projection error learning (see green path), N_{rb} et N_{er} can be limited (here $N_{rb} = \mathcal{O}(10)$ and $N_{er} = \mathcal{O}(10)$), resulting in two ANNs of reasonable size (N_{θ_1} and N_{θ_2} reasonable), and thus a potential reasonable number of snapshots for training.

Note that $\mathbf{B}_{er}$ also serves as the change-of-variable matrix from basis $\{\zeta_n(x)\}_{n=1}^{N_{er}}$ to the initial complete one $\Phi(x)$. At this stage, similarly as in Eq. (14), we introduce the following mapping:

$$\begin{array}{rccl} F_2 & : & \mathcal{H} \subset \mathbb{R}^{N_\mu} & \to & \mathbb{R}^{N_{er}} \\ & & \mu & \mapsto & \mathbf{e}_h^{N_{er}}(\mu) = \mathbf{B}_{er}^T \mathbf{e}_h(\mu), \end{array} \tag{19}$$

which maps each set of input parameters $\mu \in \mathcal{H}$ to the coefficients $\mathbf{B}_{er}^T \mathbf{e}_h(\mu)$ of the projection of the error $\mathbf{e}_h(\mu)$ onto the subspace spanned by $\{\zeta_n(x)\}_{n=1}^{N_{er}}$. The remaining task of the extended offline phase is therefore to train a new artificial NN, denoted by $\mathcal{N}_2(\theta_2)(\mu)$, to approximate the non-linear mapping F_2 (refer to Fig. 2 once again). As for the reduced solution, the latter is performed by minimizing the following loss function:

$$\mathcal{J}_2(\theta_2) = \frac{1}{N_s} \sum_{s=1}^{N_s} \| \mathcal{N}_2(\theta_2)(\mu_s) - \mathbf{B}_{er}^T \mathbf{e}_h(\mu_s) \|_{L^2}^2. \tag{20}$$

For a better understanding and conciseness, the additional operations to learn the projection error in the offline phase are outlined in Algorithm 1 (see Steps 6, 7 and 9).

Remark 4. It is important to notice at this stage that the reduction step, which transforms $\mathbf{u}_h \in \mathbb{R}^{N_h}$ into $\mathbf{u}_h^{N_{rb}} \in \mathbb{R}^{N_{rb}}$ with $N_{rb} << N_h$ is crucial in the process. This step allows to obtain only a few ouput neurons for the NN (in our application $N_{rb} = \mathcal{O}(10)$, see Section 4), making it possible to consider a deep NN with a number of parameters N_{θ_1}

(*i.e.*, $\boldsymbol{\theta}_1 \in \mathbb{R}^{N_{\theta_1}}$) reasonable compared to the number of data (*i.e.*, the N_s snapshots) during training. Of course, for the same reason, the number of input neurons of the network must be low, *i.e.*, the number of physical parameters N_μ ($\mu \in \mathbb{R}^{N_\mu}$) must be small, which is also a limitation of more conventional reduced methods based on POD.

Next, during the online phase, it is possible to estimate the projection error at a new set of physical parameters μ_{new} by simply performing a forward pass of the NN $\mathcal{N}_2$ using the optimized $\boldsymbol{\theta}_2$ obtained after training. The reduced projection error can also be written in the original basis $\Phi(x)$ such as: $\mathbf{e}_{rb}^{N_h}(\mu_{new}) = \mathbf{B}_{er}\mathbf{e}_{rb}(\mu_{new}) \in \mathbb{R}^{N_h}$. In this basis, we eventually complement the former reduced solution $\mathbf{u}_{rb}^{N_h}(\mu_{new})$ of Section 3 with the present projection error, which yields the final corrected reduced solution as follows:

$$\left(\mathbf{u}_{rb}^{N_h}(\mu_{new}) + \boldsymbol{e}_{rb}^{N_h}(\mu_{new})\right) = \tilde{\boldsymbol{u}}_{rb}^{N_h}(\mu_{new}) \approx \mathbf{u}_h(\mu_{new}). \tag{21}$$

The overall online phase is outlined in Algorithm 2; more precisely, Steps 3 to 5 constitute the additional operations to perform to correct the initial reduced solution with a learning of the projection error. This phase can be carried out very quickly since it merely consists in performing forward passes of trained NNs.

Remark 5. Let us note here that the computational overhead of the proposed methodology compared to the standard POD-NN method is negligible. Indeed, the training of the second NN can be conducted in parallel with the first during the offline phase (steps 8 and 9 of Algorithm 1 can be executed concurrently), and similarly, the evaluation of the reduced solution and its correction can be performed in parallel (steps 1-2 and 3-4 can be carried out concurrently in Algorithm 2).

As mentioned in the introduction, because our approach incorporates NNs with POD and takes into account the projection error, we refer to it as an *Error-Aware POD-NN reduction method*. Complementing the POD-NN strategy with the projection error enables to attain higher accuracy while preserving a limited number of POD modes, thus providing the opportunity to employ smaller NNs therefore with much less parameters. This may lead to the utilization

of less data (the snapshots) for training, better generalization of the model, and simpler explainability [6, 22, 3].

Algorithm 1: The Error-Aware POD-NN reduction method: the offline phase

function $[\mathbf{B}_{rb}, \mathcal{N}_1(\theta_1), \mathbf{B}_{er}, \mathcal{N}_2(\theta_2)\,] = \textit{POD-NN-OFFLINE}(N_s, \varepsilon_{POD})$.

1. Generate the reference parameters values by sampling the hypercube $\mathcal{H}$: $P_s = \{\mu_s\}_{s=1}^{N_s} \in (\mathcal{H})^{N_s}$.
2. Compute the associated complete numerical solutions with the HR model and form the snapshot matrix: $\mathbf{S} = \left[\mathbf{u}_h(\mu_1)| \ldots |\mathbf{u}_h(\mu_{N_s})\right] \in \mathbb{R}^{N_h \times N_s}$.
3. Build up the correlation matrix $\mathbf{C} = \mathbf{S}\,\mathbf{S}^T \in \mathbb{R}^{N_h \times N_h}$.
4. Compute its eigenelements $(\lambda_i, \boldsymbol{\xi}_i)_{i=1}^{N_h}$, with $\lambda_i \in \mathbb{R}_+$ and $\boldsymbol{\xi}_i \in \mathbb{R}^{N_h}$ $\left(\|\boldsymbol{\xi}_i\|_{L^2} = 1\right)$: $\mathbf{C}\,\boldsymbol{\xi}_i = \lambda_i\,\boldsymbol{\xi}_i, \quad 1 \leq i \leq N_h$.
5. Take the N_{rb} vectors $\{\boldsymbol{\xi}_i\}_{i=1}^{N_{rb}}$ associated with the N_{rb} largest $\{\lambda_i\}_{i=1}^{N_{rb}}$ (by using criterion (13) based on ε_{POD}) to form the RB matrix: $\mathbf{B}_{rb} = \left[\boldsymbol{\xi}_1| \cdots |\boldsymbol{\xi}_{N_{rb}}\right] \in \mathbb{R}^{N_h \times N_{rb}}$ with $N_{rb} << N_h$.
6. By making use of the POD projection error defined as $\mathbf{e}_h(\mu) = \mathbf{u}_h(\mu) - \mathbf{B}_{rb}\mathbf{B}_{rb}^T\mathbf{u}_h(\mu)$, form the snapshot error matrix: $\mathbf{E} = \left[\mathbf{e}_h(\mu_1)| \ldots |\mathbf{e}_h(\mu_{N_s})\right] \in \mathbb{R}^{N_h \times N_s}$.
7. Apply the POD procedure to $\mathbf{E}$ (see Steps 3 to 5 above) to compute the error RB matrix: $\mathbf{B}_{er} = \left[\boldsymbol{\zeta}_1| \ldots |\boldsymbol{\zeta}_{N_{er}}\right] \in \mathbb{R}^{N_h \times N_{er}}$ with $N_{er} << N_h$.
8. Train artificial NN $\mathcal{N}_1$ which approximates mapping F_1 defined in (14), by minimizing the loss function $\mathcal{J}_1(\theta_1) = \frac{1}{N_s}\sum_{s=1}^{N_s} \|\mathcal{N}_1(\theta_1)(\mu_s) - \mathbf{B}_{rb}^T\mathbf{u}_h(\mu_s)\|_{L^2}^2$.
9. Train artificial NN $\mathcal{N}_2$ which approximates mapping F_2 defined in (19), by minimizing the loss function $\mathcal{J}_2(\theta_2) = \frac{1}{N_s}\sum_{s=1}^{N_s} \|\mathcal{N}_2(\theta_2)(\mu_s) - \mathbf{B}_{er}^T\mathbf{e}_h(\mu_s)\|_{L^2}^2$.

Algorithm 2: The Error-Aware POD-NN reduction method: the online phase (real-time computations)

function $[\mathbf{u}_{rb}^{N_h}(\mu_{new})] = \textit{POD-NN-ONLINE}(\mu_{new}; \mathbf{B}_{rb}, \mathcal{N}_1(\theta_1), \mathbf{B}_{er}, \mathcal{N}_1(\theta_2))$.

1. Given a new set of input physical parameters gathered in μ_{new}, and taking the optimal θ_1 obtained from Algorithm 1 (Step 9), evaluate the output $\mathbf{u}_{rb}(\mu_{new})$ of artificial NN $\mathcal{N}_1$.
2. Deduce the reduced solution in basis $\Phi(x)$: $\mathbf{u}_{rb}^{N_h}(\mu_{new}) = \mathbf{B}_{rb}\mathbf{u}_{rb}(\mu_{new}) \in \mathbb{R}^{N_h}$.
3. Given the same new set of input parameters μ_{new}, and taking the optimal θ_2 obtained from Algorithm 1 (Step 10), evaluate the output $\mathbf{e}_{rb}(\mu_{new})$ of artificial NN $\mathcal{N}_2$.
4. Deduce the reduced projection error in basis $\Phi(x)$: $\boldsymbol{e}_{rb}^{N_h}(\mu_{new}) = \mathbf{B}_{er}\boldsymbol{e}_{rb}(\mu_{new}) \in \mathbb{R}^{N_h}$.
5. Deduce the final corrected reduced solution: $\tilde{\boldsymbol{u}}_{rb}^{N_h}(\mu_{new}) = \mathbf{u}_{rb}^{N_h}(\mu_{new}) + \boldsymbol{e}_{rb}^{N_h}(\mu_{new})$.

4. Numerical experiments and discussions

In this section, we numerically evaluate the proposed EA-POD-NN method on a real-world flooding event of relatively high magnitude occurred on the Aude River in the southeastern region of France in 2018. The flood hydraulic model was built in [14] based on HR terrain elevation data provided by IGN (French geographical national institute) and hydrometeorological data from SCHAPI (French national flood forecasting center) and Météo France. The DassFlow 2D software [27, 21] is employed for numerically solving the 2D SWEs following the FV numerical scheme described at the end of Section 2.2. The spatial domain Ω represents a floodplain located at the confluence of the Aude and Fresquel rivers. This domain is discretized using a triangular unstructured mesh (with 7 267 elements), refined along the rivers and generated using Gmsh [15]. It is inflowed by two real hydrographs $Q_{in,k}(\mu, t)$ at 2 upstream points on $\partial\Omega$ the border of Ω (see Fig. 4). The open-source software DassFlow 2D is available on GitHub (https://dasshydro.github.io/codes_presentation/pres_dassflow/). Additionally, the datasets can be obtained upon request for traceability purposes.

Again, the primary focus is on the μ-parameter related to the inflow discharge functions which is the most important factor controlling flooding dynamics, particularly crucial in operational context where fast and accurate hydraulic simulations of flooding are needed after hydro-meteorological forecasts of discharge. In this context, two test cases are carried out, representative of real complexity of concomitant or not flood inflows (therefore different intensities of non-linear waves interactions):

1. Case A) The two inflows are simultaneous.

   In this case, the classical POD-NN method with $N_{rb} = 5$ modes only turns out to be sufficient to satisfactorily reduce the flow model. $N_{rb} = 5$ corresponds to the tolerance $\epsilon^2 = 10^{-2}$ in Eq. (13).

2. Case B) The two inflows are non-simultaneous therefore involving more complex non-linear waves interactions.

   In this case, the classical POD-NN method with $N_{rb} = 5$ modes only struggles to accurately reproduce the solution. On the contrary, the proposed EA-POD-NN method enables to provide a good accuracy with $N_{rb} = 5$ and $N_{er} = 16$ (corresponding to $\epsilon^2 = 10^{-1}$ in Eq. (13).

Moreover, a last section proposes a comparison of the two methods in particular in terms of NNs dimensions.

Let us introduce at this stage some notations for the physical and numerical quantities that will be used throughout this section, especially in the majority of the figures:

- $U = \sqrt{v_x^2 + v_y^2}$ denotes the velocity norm in (ms^{-1}).
- h_{ref} in (m) (resp. U_{ref} in (ms^{-1})) denotes the reference 2D hydraulic model water depth (resp. the reference 2D hydraulic model velocity norm).

- $h_{\text{POD-NN}}$ in (m) (resp. $U_{\text{POD-NN}}$ in (ms^{-1})) denotes the approximated water depth (resp. the approximated velocity norm) with the POD-NN method. Similarly, $h_{\text{EA-POD-NN}}$ in (m) (resp. $U_{\text{EA-POD-NN}}$ in (ms^{-1})) denotes the approximated water depth (resp. the approximated velocity norm) with the EA-POD-NN method.

- $\varepsilon_h(x,t) = (h_{ref} - h_{\text{POD-NN}})$ or $\varepsilon_h(x,t) = (h_{ref} - h_{\text{EA-POD-NN}})$, resp. $\varepsilon_U(x,t) = (U_{ref} - U_{\text{POD-NN}})$ or $\varepsilon_U(x,t) = (U_{ref} - U_{\text{EA-POD-NN}})$, denotes the difference between the reference solution h_{ref} (resp. U_{ref}) and the approximated solution $h_{\text{POD-NN}}$ or $h_{\text{EA-POD-NN}}$ (resp. $U_{\text{POD-NN}}$ or $U_{\text{EA-POD-NN}}$).

- Finally, $\overline{\varepsilon_h}(t)$ (resp. $\overline{\varepsilon_U}(t)$) denotes the spatial average over Ω of the error on water depth (resp. of the error on velocity norm).

4.1. General design of the numerical experiments

To start with, we outline here the design of the numerical experiments considered to reduce the 2D hydraulic simulation of the 2018 flood event that occurred in the southeastern region of France. This event was characterized by intense rainfall on upstream catchments, river overflow, and extensive flooding over the studied domain Ω. The maximal submersion depths and flow velocity norm over the floodplain simulated by the reference 2D hydraulic model are shown in Fig 4(Top), along with the real hydrographs from gauging stations that are used as inflows. A $T = 60$h simulation time period is chosen for the reference 2D SW model numerical solution, to enable sufficient time for flood propagation and recession over the domain Ω.

4.1.1. Representation of the inflows signals

A necessary condition to apply the reduction method without additional pre-processing is to have an input parameter μ of quite small dimension: $N_\mu = \mathcal{O}(10)$ (see sections 2 and 3). Hence, an effective representation of inflows is performed using a minimal number of key parameters: the maximum peak flow discharge $Q_{\max}$ $(m^3 s^{-1})$, and the corresponding time of peak t_{peak} (h), defined as $t_{\text{peak}} = (t_{\max} - t_0)$ the duration between t_0 the onset of rising water levels and $t_{\max}$ the corresponding attainment of $Q_{\max}$, see Fig. 4 (Top right) for the first inflow (red curve). In this manner, hydrographs are approximated using triangular signals, see Fig. 6(bottom): we will henceforth refer to them as input triangular hydrographs in comparison to real input hydrographs. Therefore, in the present case, we have two input signals, each dependent on 2 parameters: $Q_{in,k}(t_{\text{peak,k}}, Q_{\text{max,k}}; t)$ for $k = 1, 2$. The concept of utilizing $Q_{\max}$ and t_{peak} as representative parameters is rooted in their ability to characterize the fundamental aspects of flood initiation. $Q_{\max}$ quantifies the peak discharge intensity, reflecting the maximum flow rate reached during the flood. On the other hand, t_{peak} encapsulates the time it takes for the river to transition from its initial state to the point of maximum discharge, offering insights into the flood's temporal evolution. This simple parameterization enables to represent the inflow hydrographs while preserving their essential features, *i.e.* peak flow magnitude and timing, therefore ensuring

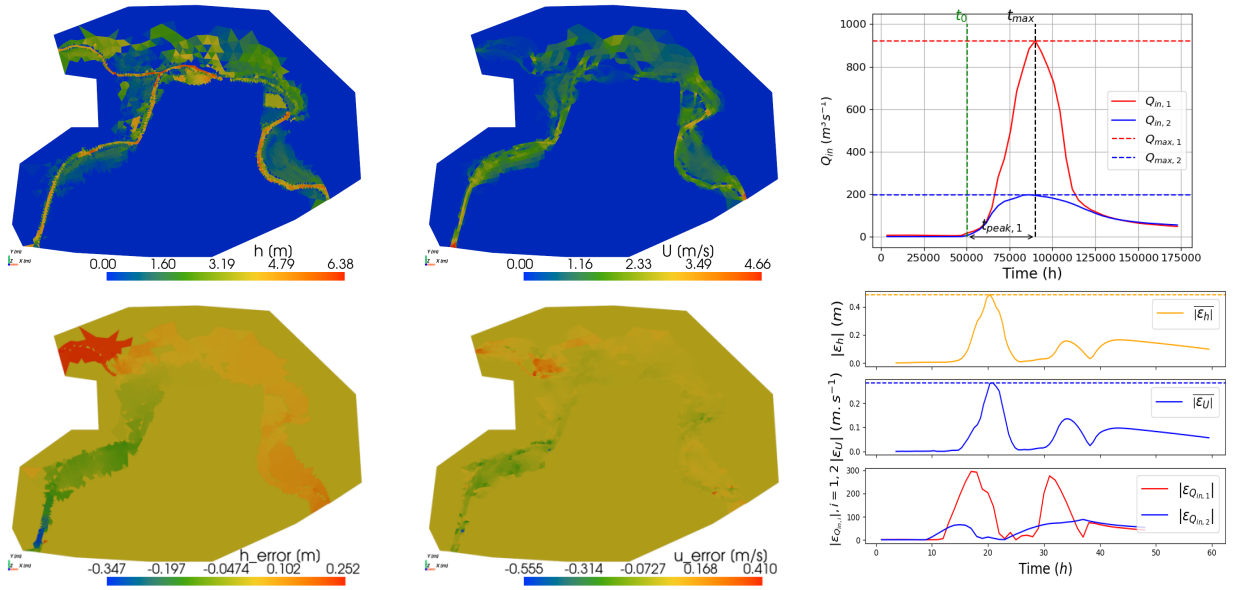

Figure 4: (Top) The reference 2D hydraulic model of the 2018 flood on the Aude River domain Ω at $t = 25.2$h around maximum flooding. (Left) The water depth h_{ref} in (m). (Middle) The velocity norm U_{ref} in (ms^{-1}). (Right) The real input hydrographs from gauging stations; (Red) for the Aude river $Q_{in,1}$ with $t_{peak,1} = 11.94$h and $Q_{max,1} = 922\text{m}^3\text{s}^{-1}$, (Blue) for the Fresquel river $Q_{in,2}$ in m^3s^{-1} with $t_{peak,2} = 9.16$h and $Q_{max,1} = 198\text{m}^3\text{s}^{-1}$; the black line corresponds to maximal flooding time at which spatial maps are plotted. (Bottom) Comparison between the reference 2D hydraulic model inflowed by the real or triangular hydrographs. We recall that the triangular hydrographs are characterized by a piecewise linear approximation from the flood start time t_0 to the peak time t_{peak} and to the recession end, see also Fig. 6(bottom). (Left) difference in terms of ε_h on water depth h, (Middle) difference in terms of ε_U on velocity norm U. (Right) Temporal variation of the spatially averaged absolute difference $\overline{|\varepsilon_h|}$ for h, $\overline{|\varepsilon_U|}$ for U, and the errors $\varepsilon_{Q_{in,1}}$ and $\varepsilon_{Q_{in,2}}$ of reproduction of real inflow hydrographs with triangular ones.

consistent flood inundation modeling as shown by the limited errors in space and time on submersion depth and velocity norm between the reference 2D SW model run with real or triangular hydrographs, see Fig. 4(Bottom). In particular, the errors are relatively lower at peak time and maximum flooding rather than during more dynamic phases of submersion and recessions i.e. corresponding to hydrograph rising limb and also during recession.

As mentioned previously, two test cases are considered with different input parameters, corresponding to increasingly complex inflow signals and resulting SW model responses over the floodplain:

1. Case A): simultaneous inflow hydrographs: $\mu = (t_{\text{peak}}, Q_{\text{max},1}, Q_{\text{max},2}; t_{sn})$, so $N_\mu = 4$.
2. Case B): non-simultaneous inflow hydrographs: $\mu = (t_{\text{peak},1}, Q_{\text{max},1}, t_{\text{peak},2}, Q_{\text{max},2}; t_{sn})$, so $N_\mu = 5$.

For both cases, the different steps of Algorithm 1 are performed to achieve the offline phase of the reduction method (see Section 3). Case-specific parameters values are detailed for each experiment in the following paragraphs.

Note that considering more complex inputs such as more inflow BCs with temporal variations and/or spatially distributed forcing term such as rain signal (i.e. a source term into the mass equation), that is an input dimension $N_\mu > \mathcal{O}(10)$, an upstream reduction (for example using an AutoEncoder) would be needed as remarked in Section 2.1.

4.1.2. Snapshots matrix construction

Here, the array $P_s = \{\mu_s\}_{s=1}^{N_s}$ (see Section 2.1) of N_s parameters sets is obtained by uniformly sampling M times the discharge related parameters in μ considering upper and lower bounds around the reference values of the 2018 real flood hydrographs (the detailed values will be specified later). The last parameter t_{sn} defines the N_t physical times at which the snapshots of the reference 2D SW model are computed. It is fixed *a priori* in function of the response dynamics of the physical model to reduce. Overall, we have $N_s = M \times N_t$.

Remark 6. It is important to note that the snapshot time parameter t_{sn} conditions the capture of information in the reduction process of the physical model. Here hydraulic simulations were performed over a span of 60 hours. However, in the context of SW model reduction, we select the time interval starting from the point corresponding to the onset of rising water levels for the inflows. This selection allows us to concentrate on the period where the flood dynamics exhibit significant changes, aligning with the goals of SW model reduction. Also, this approach ensures that the NNs focus on capturing meaningful flow variations that occur starting from the rising phase.

Next, for each pair of inflow hydrographs $Q_{in;k=1,2}(\mu_s, t)$, we compute the model output vectors $\mathbf{u}_h(\mu_s) = [\mathbf{h}^s,\ \mathbf{v}_x^s,\ \mathbf{v}_y^s]^T$ which is of dimension $N_h = 3 \times N_x = 21\ 801$, where $N_x = 7\ 267$ counts the number of cells of the mesh covering Ω. Note eventually that each model output $\mathbf{h}$, $\mathbf{v}_x$ or $\mathbf{v}_y$ consists in N_t temporal snapshots of model outputs spatial fields. This enables to obtain the $(N_h \times N_s)$ snapshot matrix $\mathbf{S}$ as follows:

$$\mathbf{S} = [\mathbf{h},\ \mathbf{v}_x,\ \mathbf{v}_y]^T = \begin{bmatrix} \mathbf{h}(\mu_1; x)| & \cdots & |\mathbf{h}(\mu_{N_s}; x)| \\ \mathbf{v}_x(\mu_1; x)| & \cdots & |\mathbf{v}_x(\mu_{N_s}; x) \\ \mathbf{v}_y(\mu_1; x)| & \cdots & |\mathbf{v}_y(\mu_{N_s}; x)| \end{bmatrix}.$$

Once $\mathbf{S}$ is built, one can perform the other steps of the offline phase for the POD-NN and EA-POD-NN algorithms (see Algorithm 1).

4.1.3. RB matrix construction

The second step involves constructing the reduced matrix $\mathbf{B}_{rb} \in \mathbf{R}^{N_{rb} \times N_h}$ by applying the POD method to the snapshots matrix $\mathbf{S}$. In the two cases A) and B) previously defined, we conserve 99.9% of the total energy of the system that is $\epsilon^2_{POD} = 10^{-2}$ in Eq. (13). Then, we obtain $N_{rb} = 5$. Concerning the error matrix $\mathbf{E}$, we set $\epsilon^2_{er} = 10^{-1}$, this provides $N_{er} = 17$ for Case A) and $N_{er} = 16$ for Case B). The corresponding eigenvalues of the correlation matrix $\mathbf{C}$ are presented in Fig. 5. Note that the behaviour of the eigenvalues of the correlation matrix of the errors ($\mathbf{C}_{er} = \mathbf{E} \cdot \mathbf{E}^T$) is similar for both cases A) and B).

Constraints	
Case 1: $N_s < N_h$	Correlation matrix $\mathbf{C} = \mathbf{S}^T\mathbf{S}$ of size $N_s \times N_s$
Case 2: $N_s > N_h$	Correlation matrix $\mathbf{C} = \mathbf{S}\mathbf{S}^T$ of size $N_h \times N_h$
Last layer size:	$N_L > N_{rb}$
Over-fitting or not:	N_θ vs N_s

Table 2
The conditions that must be adhered to in the the offline phase for both numerical test cases. These conditions concern the following parameters: N_s representing the number of snapshots, N_θ denoting the number of parameters of the NN, and N_L indicating the number of neurons in the last layer of the NN.

Remark 7. The Kolmogorov N_{rb}-width [10, 24] enables to measure the degree of reductibility of model. A link between the Kolmogorov N_{rb}-width and the POD model construction is presented in [24]. This link relies on the decay of the eigenvalues. In Fig. 5, we observe a slow decay of the eigenvalues, which is typical in the case of non-linear convection-dominated problems [4]. That is, the error in the right hand side of Eq. (12) is large. This turns the SWEs reduction problem to be challenging.

Remark 8. For the EA-/POD-NN approach, certain numerical constraints must be adhered to. These constraints are summarized in Tab 2.

4.2. Case A): simultaneous inflows

We present here the results obtained in Case A) involving simultaneous inflow hydrographs, and ROMs with $N_{rb} = 5$ and $N_{er} = 17$.

4.2.1. The parameter space

As mentioned above, the same t_{peak} for both inflows is considered here, so that the parameter reads $\mu = (t_{\text{peak}}, Q_{\text{max},1}, Q_{\text{max},2}, t_{sn})$ and is of dimension $N_\mu = 4$. Following previous discussions, the parameters ranges are defined as follows: $t_{\text{peak}} \in [11.2, 12.69]$, $Q_{\text{max},1} \in [600, 1200]$, and $Q_{\text{max},2} \in [100, 400]$. Then, the snapshots time input parameter is chosen as $t_{sn} \in [3.6, 60]$. Recapitulating, one has $\mathcal{H} = [11.2, 12.69] \times [600, 1200] \times [100, 400] \times [3.6, 60]$. A visual representation of the parameter set sample $P_s = \{\mu_s\}_{s=1}^{N_s}$ with a focus around t_{peak}, $Q_{\text{max},1}$, and $Q_{\text{max},2}$ is given in Fig. 6.

The snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$ is constructed with a number of snapshots fixed as $N_s = M \times N_t = 189\,000$ (with $M = 1\,000$ simulations using the DassFlow software and $N_t = 189$ writing times, *i.e.* a $dt_w = 18$min writing time step). Note that quasi similar results could be obtained with $dt_w = 36$min, as will be chosen for the second case with non-simultaneous hydrographs because of memory limitations. Note finally that such writing time steps remain significantly lower than the flood propagation time that is around $3h$ over the studied zone.

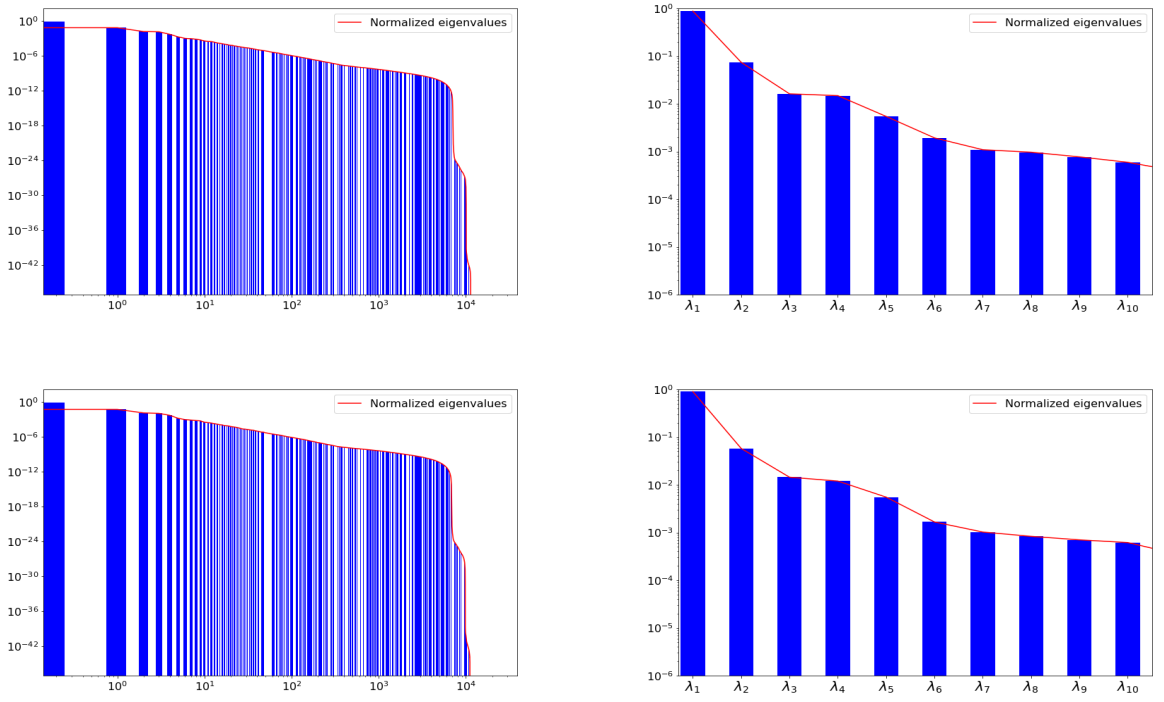

Figure 5: (Left) The decay of eigenvalues of the correlation matrix $\mathbf{C}$ (log-scales). (Right) The zoom on the first 10 eigenvalues. (Top) For the case with simultaneous hydrographs ($N_\mu = 4$). (Bottom) For the case with non-simultaneous hydrographs ($N_\mu = 5$).

4.2.2. The training phase (during the offline phase)

For the training of the NNs, the data of size $N_s = 189\,000$ is classically partitioned into 80% for the actual training, 10% for the test and 10% for the validation. To ensure that the number of NNs parameters remains small relative to the data dimension, we determine the number of parameters of the NN $\mathcal{N}_1$ as 3% (resp. of the NN $\mathcal{N}_2$ as 5%) of the training data dimension N_s, yielding $N_{\theta_1} = 4\,405$ for $\mathcal{N}_1$ with $N_{rb} = 5$ for the output layer, and $N_{\theta_2} = 9\,977$ with $N_{er} = 17$ for the output layer of $\mathcal{N}_2$ in the case of simultaneous inflows (resp. $N_{er} = 16$ in the case of non simultaneous inflows). The number of hidden layers for $\mathcal{N}_1$ is 10 with 20 neurons per layer and for $\mathcal{N}_2$ it is 10 with 30 neurons per layer. The chosen activation function is '*RELU*'. We train the NNs over 10 000 epochs which leads to $\varepsilon_{NN} = 10^{-7}$ in terms of loss function. The loss function on the validation data set is observed to well decrease too. An *Adam* optimizer [19] is used for the gradient descent to minimize the loss function. The latter computes the loss between the HR projected output $\mathbf{u}_h^{N_{rb}}(\mu_s) = \mathbf{B}_{rb}^T \mathbf{u}_h(\mu_s)$ (resp. $\mathbf{e}_h^{N_{er}}(\mu_s) = \mathbf{B}_{er}^T \mathbf{e}_h(\mu_s)$), $1 \leq s \leq N_s$, and the NN output $\mathcal{N}_1(\theta_1)(\mu_s)$ (resp. $\mathcal{N}_2(\theta_2)(\mu_s)$) using the Mean Square Error (MSE), in line with Eqs. (15) and (20).

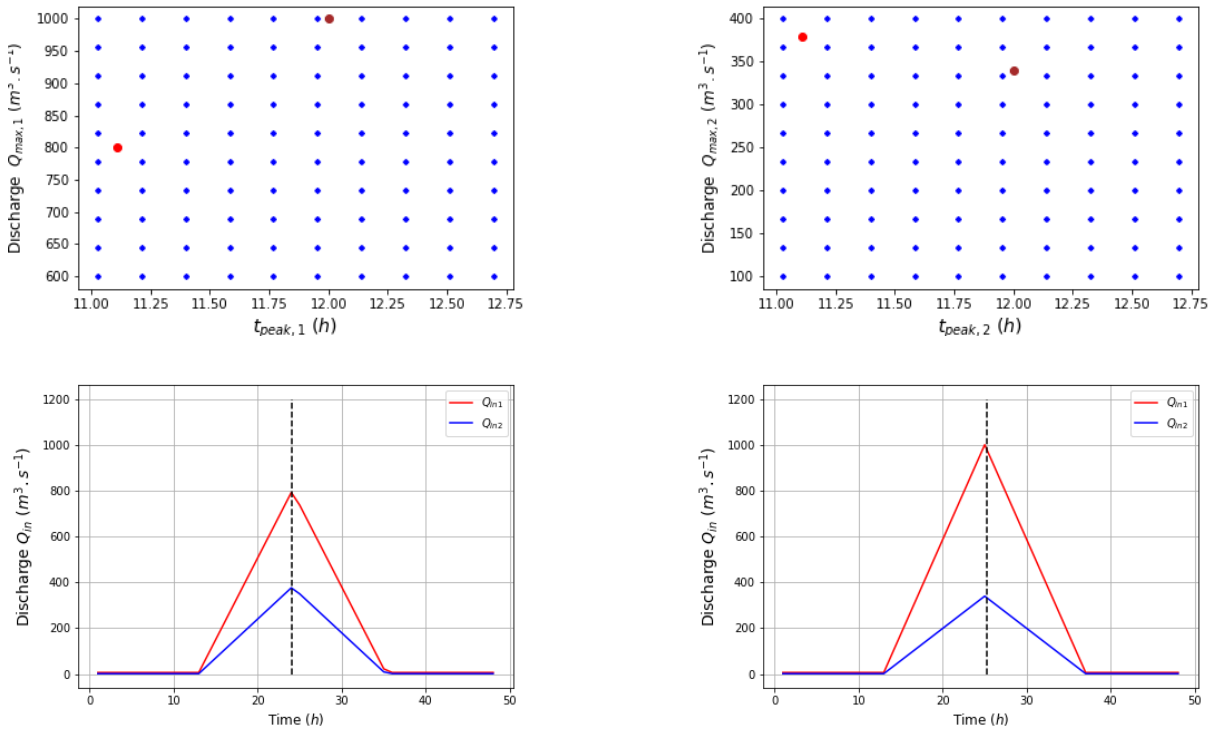

Figure 6: Simultaneous inflows case. (Top) The sampled parameter sets P_s, blue points for offline phase, red and brown points being random parameter sets that will be used for online prediction; (left) For the first hydrograph $Q_{in,1}$ on the Aude River and (right) for the second hydrograph $Q_{in,2}$ on the Fresquel River (corresponding to red and blue hydrographs, respectively, in Fig. 4). (Bottom) Input inflow hydrographs for (left) red and (right) brown points that will be used for online prediction (vertical dashed lines correspond to the time of maximum flooding at which spatial error analysis will be performed).

4.2.3. The online prediction phase

After successfully completing the training phase and optimizing the NNs parameters, we move to the prediction stage, where the model demonstrates its ability to rapidly forecast hydraulic variables $(h, v_x, v_y)(x, t)$ for new parameter values, denoted as μ_{new}. These new parameter values are in the parameter space range ($\mu_{\text{new}} \in \mathcal{H}$) but distinct from all the points in P_s (let us recall that these new parameter values correspond to the red and brown points in Fig. 6 (Top), which are different from the blue points that form P_s). In contrast, the chosen parameter t_{sn} defines the times at which the snapshots are performed in the offline phase and the prediction time in online phase. These two points μ_{new} are carefully selected: the first (brown point in Fig. 6(Top)) is chosen to be in close proximity to the training phase data points, while the second (red point in Fig. 6(Top)) is taken at a considerable distance from the training data points. This deliberate selection allows us to rigorously evaluate the predictive capabilities and precision of our approximation model. The performances obtained in reducing the 2D SW model are depicted in their spatio-temporal dimensions with

several graphes (see Figs. 7 to 10), especially at maximal flooding which is a crucial instant for operational forecasting, with related spatial statistics of $\overline{\varepsilon_h}(x, t_{peak})$ and $\overline{\varepsilon_U}(x, t_{peak})$, but also in terms of temporal variability of $\overline{\varepsilon_h}(t)$ and $\overline{\varepsilon_U}(t)$.

More precisely, spatial performances of the online solution $\boldsymbol{u}_{\square}(\mu_{new})$ at maximum flooding are first shown in Figs. 7 and 8 for $\mu_{new,1} = (11.11, 800, 380; 24)$ (red point in Fig. 6) and $\mu_{new,2} = (12, 1000, 340; 25.2)$ (brown point in Fig. 6). Then, the spatial values of misfit to the reference 2D SW simulation $\boldsymbol{u_h}(\mu_{new})$ are depicted at peak time in histograms in Fig 9 and in terms of Cumulative Distribution Function (CDF) of their spatial variability in Fig. 10. In particular, the median value of $|\varepsilon_h(x, t_{peak})|$ (or $|\varepsilon_U(x, t_{peak})|$) is 10^{-3} m (or 0.7×10^{-3} ms^{-1}) with EA-POD-NN and 3×10^{-3} m (or 10^{-3} ms^{-1}) with POD-NN for $\mu_{new,1}$. Also, the median value of $|\varepsilon_h(x, t_{peak})|$ (or $|\varepsilon_U(x, t_{peak})|$) is 2×10^{-3} m (or 10^{-3} ms^{-1}) with EA-POD-NN and 5×10^{-3} m (or 4×10^{-3} ms^{-1}) with POD-NN for $\mu_{new,2}$.

Next, the temporal variation of errors spatial averages $\overline{|\varepsilon_h|}(t)$ (or $\overline{|\varepsilon_U|}(t)$) in Fig. 10 shows that submersion depth (or flow velocity magnitude) stays below $0.1m$ (or $0.1ms^{-1}$) over the whole spatio-temporal domain for both reduction approaches. For instance, the median error value is 0.006 m (or 0.005 ms^{-1}) with EA-POD-NN and 0.0252 m (or 0.016 ms^{-1}) with POD-NN for $\mu_{new,1}$ and 0.007 m (or 0.007 ms^{-1}) with EA-POD-NN and 0.025 m (or 0.019 $ms^{-}1$) with POD-NN for $\mu_{new,2}$. Also, the maximum value of the temporal variation of errors spatial averaged $\overline{|\varepsilon_h|}(t)$ (or $\overline{|\varepsilon_U|}(t)$) is 0.04 m (or 0.03 ms^{-1}) with EA-POD-NN and 0.12 m (or 0.08 ms^{-1}) with POD-NN for $\mu_{new,1}$ and 0.04 m (or 0.03 ms^{-1}) with EA-POD-NN and 0.10 m (or 0.08 ms^{-1}) with POD-NN for $\mu_{new,2}$.

Overall, both the classic POD-NN and proposed EA-POD-NN demonstrate good performance in reproducing flow depth and velocity in this moderately complex dynamic flooding scenario. The accuracy achieved in both spatial and temporal aspects of flow variables highlights the applicability of both methods for reducing the 2D SW model in this case with simultaneous inflows. Nevertheless, a slight improvement in accuracy is already noticeable with the proposed EA-POD-NN, suggesting that this approach may also prove high-performance in more complex flooding scenarios, as demonstrated below.

4.3. Case B): non-simultaneous inflows

We present here the results obtained in Case B) involving non-simultaneous inflow hydrographs, and ROMs with $N_{rb} = 5$ and $N_{er} = 16$.

4.3.1. The parameter space and training phase

This case is for non-simultaneous inflow hydrographs and the parameter reads $\mu = (t_{\text{peak},1}, Q_{\max,1}, t_{\text{peak},2}, Q_{\max,2}; t_{sn})$ and is of dimension $N_\mu = 5$. Again, parameters ranges are defined around the values of the reference hydrograph and are as follows: $t_{\text{peak},1} \in [7.83, 14.77]$, $Q_{\max,1} \in [600, 1200]$, $t_{\text{peak},2} \in [7.44, 9.22]$, and $Q_{\max,2} \in [100, 400]$.
In summary, one has $\mathcal{H} = [7.83, 14.77] \times [600, 1200] \times [7.44, 9.22] \times [100, 400] \times [7.2, 60]$. A visual representation of P_s, with a focus on $t_{\text{peak},1}$, $Q_{\max,1}$, $t_{\text{peak},2}$, and $Q_{\max,2}$ is given in Fig. 11.

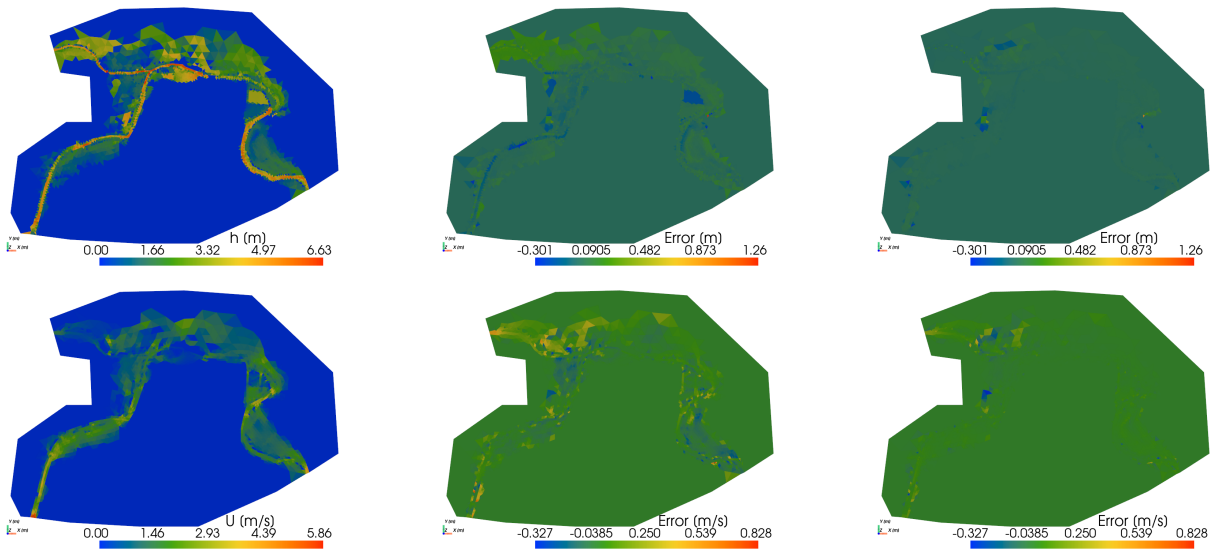

Figure 7: Simultaneous inflows case. Comparison of the model outputs at maximum flooding time for $\mu_{new} = (11.11, 800, 380, 24.0)$ (red point in Fig. 6 (Top)), (Top) For the water depth h. (Bottom) For the velocity norm U. (Left) The reference solution: 2D SW model with triangular inflow, i.e. $\boldsymbol{u}_h(\mu_{new})$. (Middle) The difference ε_h or ε_U for the calssic POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$. (Right) The difference ε_h or ε_U for the proposed EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$.

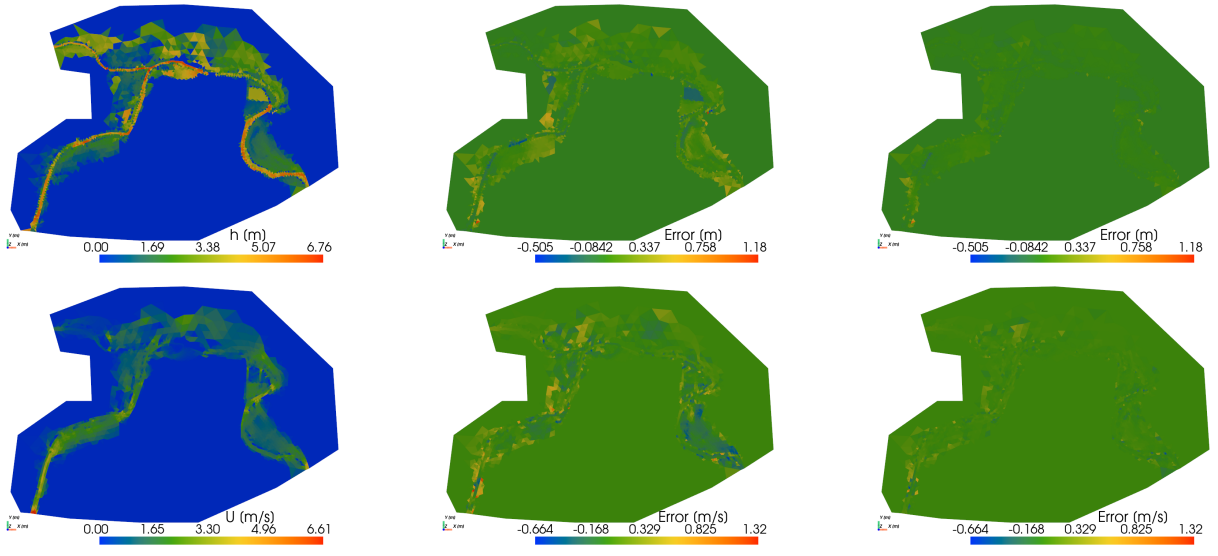

Figure 8: Simultaneous inflows case. Comparison of the model outputs at maximum flooding time for $\mu_{new} = (12, 1000, 340, 25.2)$ (brown point in Fig. 6 (Top)), (Top) For the water depth h. (Bottom) For the velocity norm U. (Left) The reference solution: 2D SW model with triangular inflow, i.e. $\boldsymbol{u}_h(\mu_{new})$. (Middle) The difference ε_h or ε_U for the classic POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$. (Right) The difference ε_h or ε_U for the proposed EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$.

The snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$, with a number of snapshots fixed as $N_s = M \times N_t = 180\,225$, with $M = 2\,025$ simulations using DassFlow software and $N_t = 89$ writing times, i.e. $dt_w = 36$min writing time steps - chosen due to numerical memory capacity constraints but this dt_w remains significantly lower than the flood propagation time that is around $3h$ over the studied zone. Note that with $dt_w = 18$min as in the previous case (results not presented): (i) the

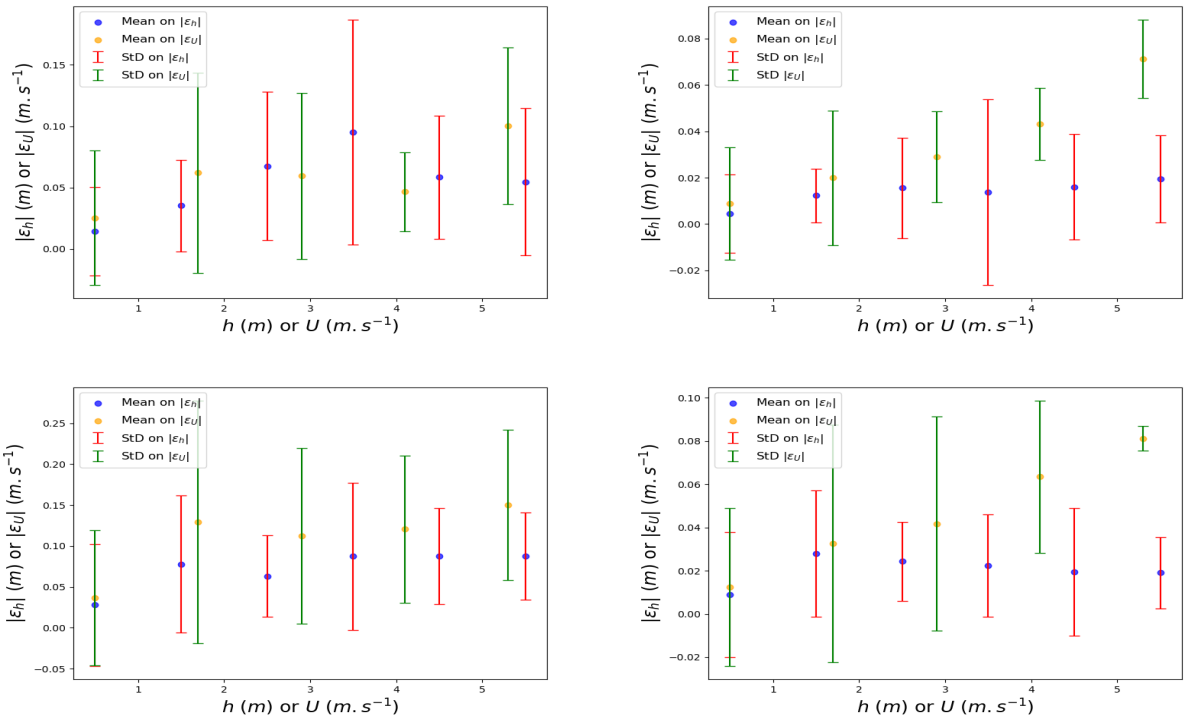

Figure 9: Simultaneous inflows case. The histogram of the mean and the StD of spatial values of $|\varepsilon_h|$ and $|\varepsilon_U|$ at maximum flooding time. (Left) For the classic POD-NN method. (Right) For the proposed EA-POD-NN method. (Top) For $\mu_{new} = (11.11, 800, 380, 24.0)$ (red point in Fig. 6. (Bottom) For $\mu_{new} = (12, 1000, 340, 25.2)$ (brown point in Fig. 6).

POD-NN method leads to an inaccurate reduction in terms of predicted physical variables, (ii) the EA-POD-NN gives model reduction performances that are comparable to those obtained with $dt_w = 36$min.

Note that the training phase is performed in this case with the same tolerances and reduced dimension $N_{rb} = 5$ as in first case (cf. section 4.1.3). The only difference is that the reduced dimension for errors reduced matrix $\mathbf{B}_{er} \in \mathbb{R}^{N_h \times N_{er}}$ is $N_{er} = 16$. The number of NNs parameters are $N_{\theta_1} = 4\,405$ for $\mathcal{N}_1$ with $N_{rb} = 5$ for the output layer, and $N_{\theta_2} = 9\,977$ with $N_{er} = 16$ for the output layer of $\mathcal{N}_2$. The number of hidden layers for $\mathcal{N}_1$ is 10 with 20 neurons per layer and for $\mathcal{N}_2$ it is 10 with 30 neurons per layer.

4.3.2. The prediction phase (online phase)

Evaluation against reference SW model with triangular inflows. For the prediction stage, again, we perform a prediction for a new parameter value, denoted as μ_{new}. This new parameter value is in the parameter space range, i.e. $\mu_{\text{new}} \in \mathcal{H}$, but is distinct from all the calibration parameter sets that compose P_s. This new value μ_{new} corresponds to the red point in Fig. 11 and is selected at a considerable distance from the training data points. Fig. 12 shows a comparison between the reference HR solution (DassFlow solution) and the predicted solution by the EA-POD-NN method.

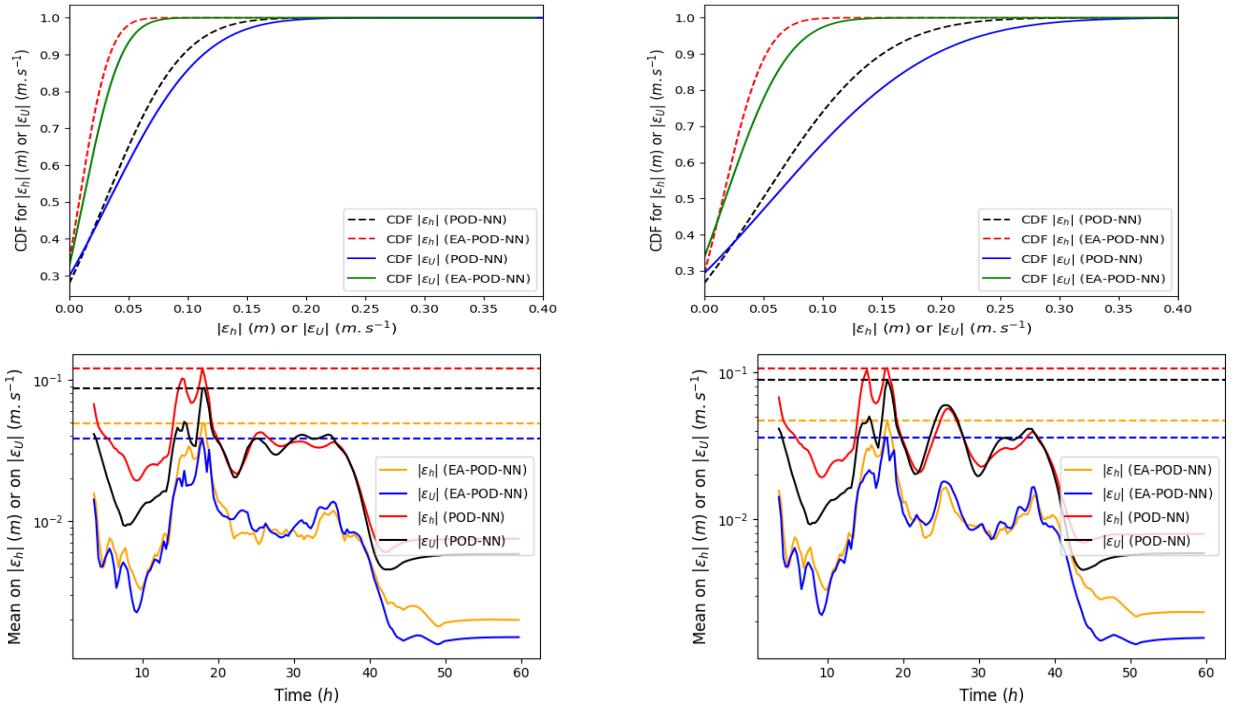

Figure 10: Simultaneous inflows case. Comparison between the classic POD-NN and the proposed EA-POD-NN methods performances. (Top) The CDF of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$. (Bottom) The mean of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$ with respect to the real time. (Left) For $\mu_{new} = (11.11, 800, 380, 24.0)$ (red point in Fig. 6. (Right) For $\mu_{new} = (12, 1000, 340, 25.2)$ (brown point in Fig. 6).

The prediction performances with this online phase are analyzed, in terms of $\varepsilon_h(x,t)$ and $\varepsilon_U(x,t)$, for both POD-NN and EA-POD-NN methods, with similar graphes as before: spatial fields of errors at maximum flooding are shown in Fig. 12 and the corresponding values are depicted by histograms of spatial values of ε at maximum flooding time in function of physical variable values in Fig. 13. Again, reduction error is significant with the proposed EA-POD-NN method over this more complex case with non-simultaneous hydrographs. This is confirmed by the temporal variation and the Cumulative Distribution Function (CDF) of absolute error spatial average shown in Fig. 13 which clearly shows that the error with EA-POD-NN is 10 times lower than with POD-NN. Namely, the median value of error on h (or U) is 0.02 m (or 0.012 ms^-1) with POD-NN and 1.6×10^{-3} m (or 9×10^{-4} ms^-1) with EA-POD-NN. Also, the temporal variation of errors spatial averages $\overline{|\varepsilon_h|}(t)$ (or $\overline{|\varepsilon_U|}(t)$) are shown in Fig. 14. In the latter, the maximum values of the temporal variation of errors spatial averaged is 0.002 m (or 0.002 ms^-1) with EA-POD-NN and 0.44 m (or 0.27 ms^-1) with POD-NN. The maximum value of the temporal variation of errors spatial averaged $\overline{|\varepsilon_h|}(t)$ (or $\overline{|\varepsilon_U|}(t)$) is 0.02 m (or 0.02 ms^-1) with EA-POD-NN and 1.11 m (or 0.79 ms^-1) with POD-NN.

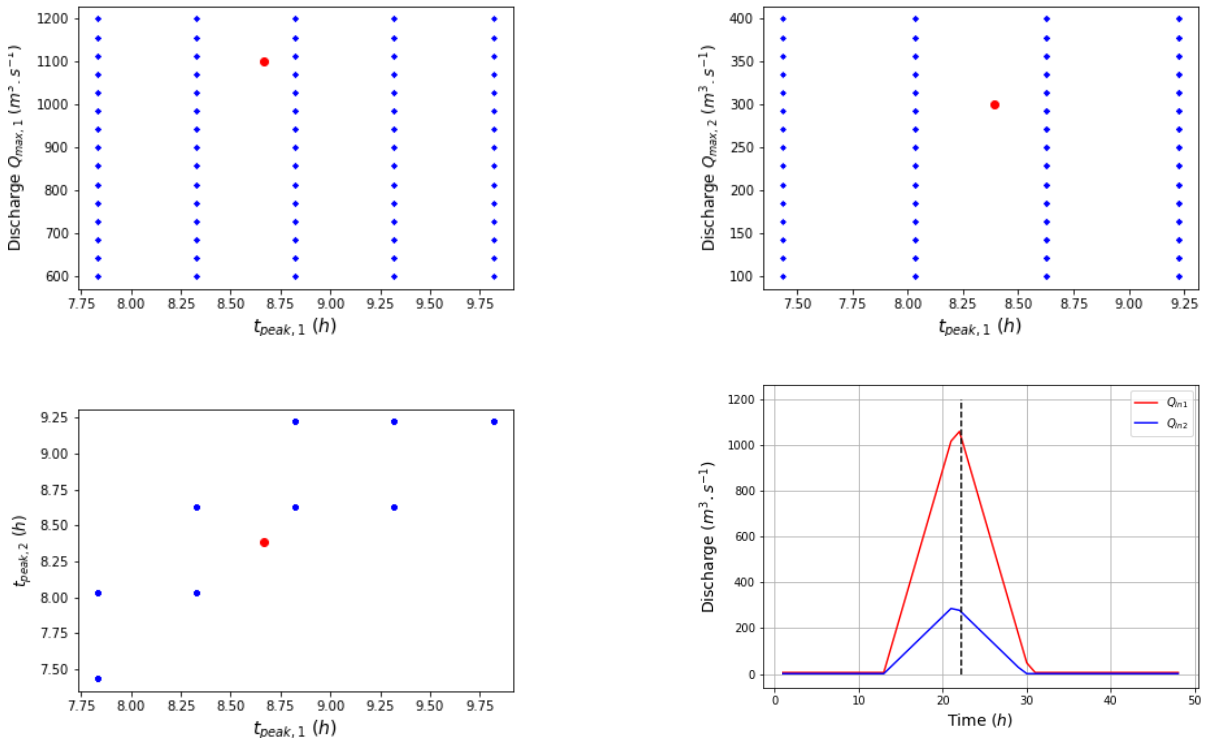

Figure 11: Non-simultaneous inflows case. (Top) The sampled parameter sets P_s, blue points for offline phase, red point being random parameter sets for online prediction; (left) For the first hydrograph $Q_{in,1}$ on the Aude River and (right) for the second hydrograph $Q_{in,2}$ on the Fresquel River (red and blue hydrographs in Fig. 4). (Bottom left) The sampled parameter sets P_s with focus on time of peak $t_{\text{peak},1}$ for the first hydrograph on the Aude River and $t_{\text{peak},2}$ for the second hydrograph on the Fresquel River. (Bottom right) Input inflow hydrographs for red point used for online prediction - vertical dashed lines correspond to the time of maximum flooding at which spatial error analysis are performed in each case.

These relatively good performances show the capability of EA-POD-NN method to accurately reduce the 2D SW model in this case with non-simultaneous inflows. Results also show the limitations of the classical POD-NN to reduce of the SW model for this more complex case with non-simultaneous hydrographs.

Evaluation against reference SW model with real inflows. Now let us go further and test the predictability of our model first when taking the real hydrographs in Fig. 4 (right) as input inflows for the reference 2D HR hydraulic model; second when taking triangular hydrographs corresponding to a parameter μ. The latter is given by $\mu_{new} = (11.94, 922, 9.16, 198, 25)$ and it is out of the range of the space parameter points P_s given in Fig. 11. The predicted hydraulic variables at peak flow are comparable to those of the reference HR hydraulic model as shown in spatial maps of Fig. 15 with real like hydrographs (resp. in 18 with triangular hydrographs corresponding to the paramerter $\mu_{new} = (11.94, 922, 9.16, 198, 25)$). The associated error histogram is shown in Fig. 16 with real like hydrographs (resp. in 19 with triangular hydrographs). The temporal variation of spatially averaged error analysis is presented in Fig. 17 with real like hydrographs (resp. in 20 with triangular hydrographs). These results show the

relatively good accuracy of the reduction compared to the reference model. For instance, in the case of real input hydrographs, the maximum value of the temporal variation of errors spatial averaged with POD-NN on h (or U) is 0.99 m (or 0.73 $ms^{-}1$) and equal to 3 times those one obtained by the EA-POD-NN (0.34 m (or 0.22 $ms^{-}1$)). Moreover, with triangular hydrographs corresponding to the parameter $\mu_{new} = (11.94, 922, 9.16, 198, 25)$, the maximum value of the temporal variation of errors spatial averaged with POD-NN on h (or U) is 1.03 m (or 0.74 $ms^{-}1$) and equal to 5 times those one obtained by the EA-POD-NN (0.20 m (or 0.11 $ms^{-}1$)). Again, the EA-POD-NN is able to reduced the SW model in this complex case with non-simultaneous inflows where the parameter $\mu_{new} = (11.94, 922, 9.16, 198, 25)$ is taken out of the learning set ($\mu_{new} \notin P_s$). While, the POD-NN is no more applicable to reduce the SW model.

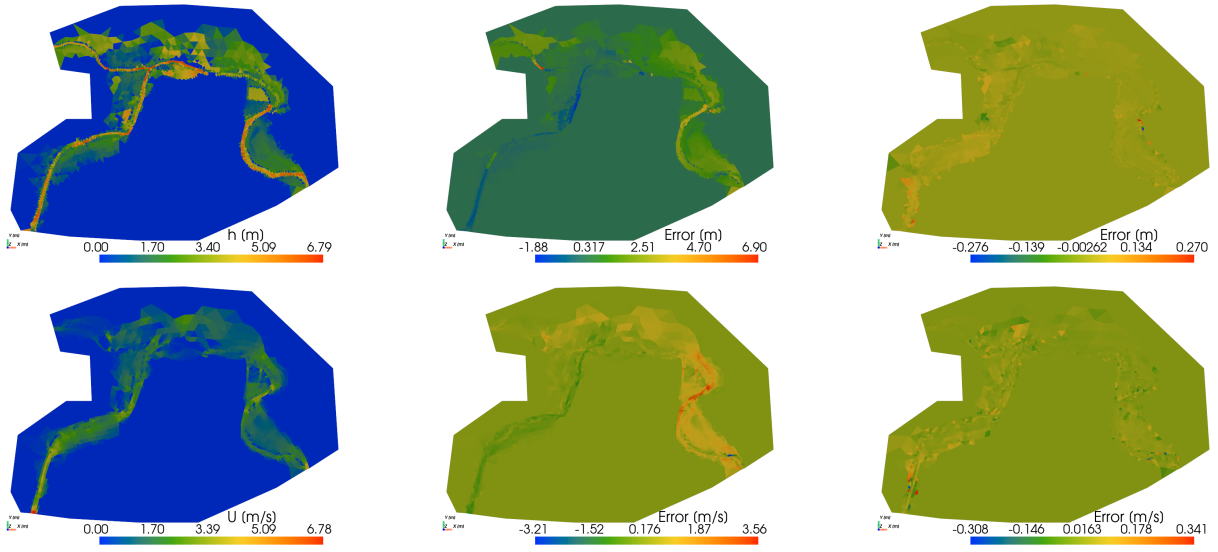

Figure 12: Non-Simultaneous inflows case. Comparison of the model outputs at maximum flooding time for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 11), (Top) For the water depth h. (Bottom) For the velocity norm U. (Left) The reference solution: 2D SW model with triangular inflow, i.e. $\boldsymbol{u}_h(\mu_{new})$. (Middle) The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$. (Right) The difference ε_h or ε_U for EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$.

4.4. Computational costs, complexity and comparison of methods

To provide further insight into the developed approach, we now investigate more precisely the NNs complexity which is a critical aspect in the explainability of machine learning based models. We compare the proposed EA-POD-NN method to the standard POD-NN method for similar complexities, as well as to the POD-NN method with much higher complexity. The complexity of the ROM approximator is determined by the depth of the NN(s) and the number of neurons per layer, the latter depending on the number of modes N_{rb}, plus N_{er} in the case of EA-POD-NN. The experiments are performed on the most challenging case with non-simultaneous inflows (Case B) described in Section 4.3), studying the impact of the NNs complexity and a smaller amount of training data (i.e. with a lower

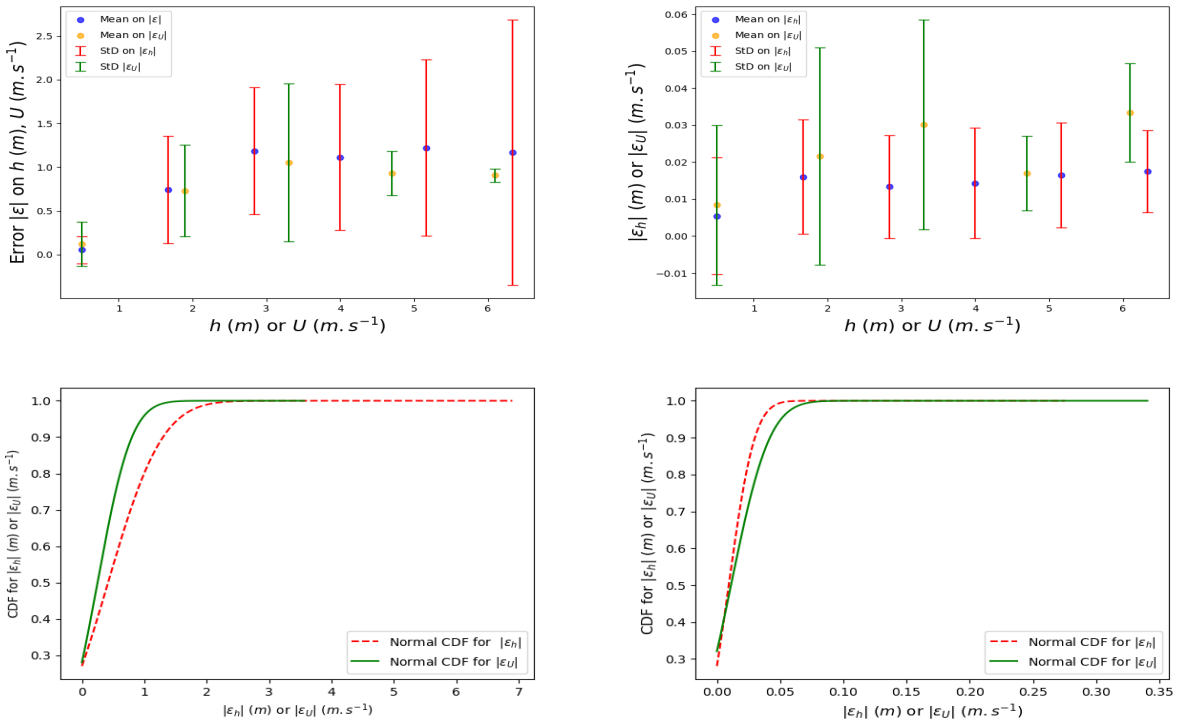

Figure 13: Non-simultaneous inflows case. (Top) The histogram of the mean and the StD of spatial values of the absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$ at maximum flooding time. (Left) with POD-NN method. (Right) with EA-POD-NN method for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 11. (Bottom) The CDF of the absolute absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$. (Left) with POD-NN method. (Right) with EA-POD-NN method for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 11.

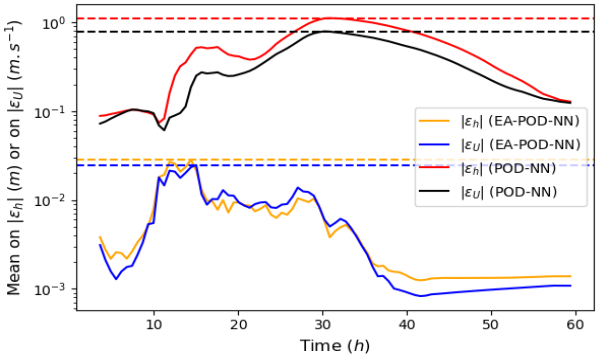

Figure 14: Non-simultaneous inflows case. The mean on the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$ with respect to the real time with POD-NN and EA-POD-NN methods for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 6. The horizontal dashed lines correspond to to the maximum value of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$.

number of snapshots N_s). For all comparisons presented here, we use $N_s = 20\,025$. The setup for the various numerical experiments designed for comparison purposes is as follows.

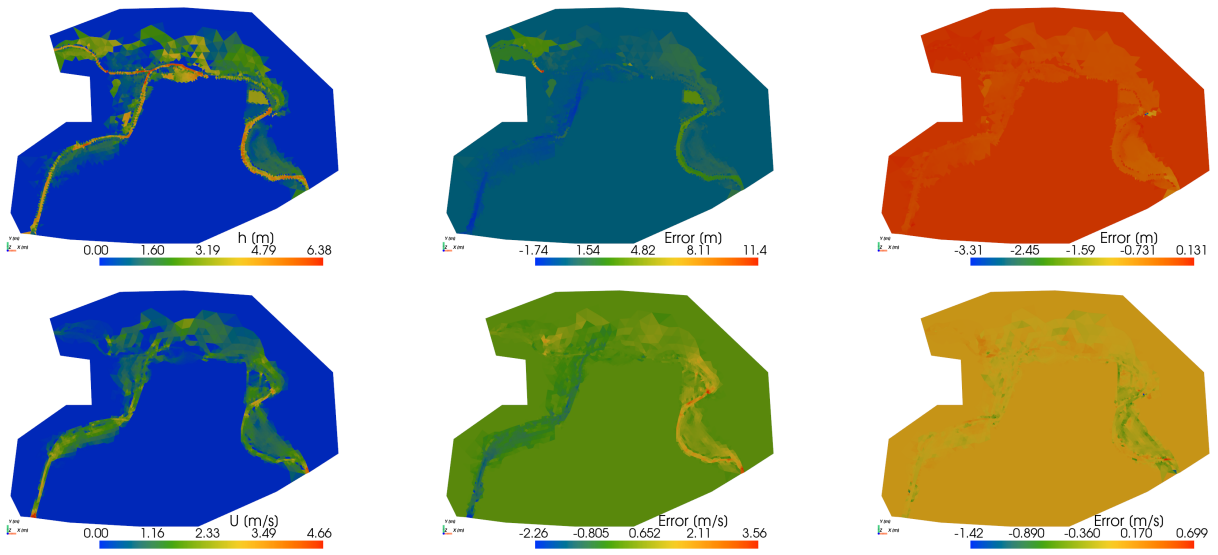

Figure 15: Non-simultaneous inflows case (real hydrographs). Comparison of the model outputs at maximum flooding time. (Top) For the water depth h. (Bottom) For the velocity norm U. (Left) The reference solution: 2D SW model with real inflow hydrographs (Fig. 4 right), i.e. $\boldsymbol{u}_h(\mu_{new})$. (Middle) The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$. (Right) The difference ε_h or ε_U for EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$.

Setup 1: the EA-POD-NN method (see Section 3) with $N_{rb} = 5$, $N_{er} = 10$ and $N_{\theta_1} + N_{\theta_2} = 11\,335$. The number of error modes has voluntarily been slightly reduced compared to $N_{er} = 16$ in the previous numerical experiments (Case B in Sections 4.3), while it still provides a good ROM accuracy, as shown after.

Setup 2: the POD-NN method as in [18] with $N_{rb} = 10$ and $N_\theta = 11\,085$, i.e. with an equivalent complexity as the one of EA-POD-NN of Setup 1 in terms of CPU time and of memory storage (about the same number of parameters), see Tab. 3.

Setup 3: the POD-NN method with a much greater NN complexity: $N_{rb} = 63$ and $N_\theta = 54\,593$.

This last experiment is the only one which does not satisfy the non-over fitting constraint $N_\theta < N_s$, see Tab. 3.

Firstly, as already indicated in Remark 5, it is important to note that adding an additional NN does not increase CPU time in the online phase because the two NNs can be evaluated in parallel. In other words, steps 1-2 and 3-4 of Algorithm 2 can be executed in parallel, see a synthesis in Tab. 3. Similarly, this applies to the NNs training in the offline phase but with an additional step in the proposed EA-POD-NN method that requires to compute $\boldsymbol{B}_{er}$ in addition to $\boldsymbol{B}_{rb}$ (steps 6 and 7 in Algorithm 1). This additional cost is paid during the offline phase: it does not impact the real-time aspect during the online phase.

Furthermore, the results of the 3 setups, comparing the proposed EA-POD-NN method and the standard POD-NN method, are presented in a similar manner as previously, in terms of errors flow depth and velocity in the online phase

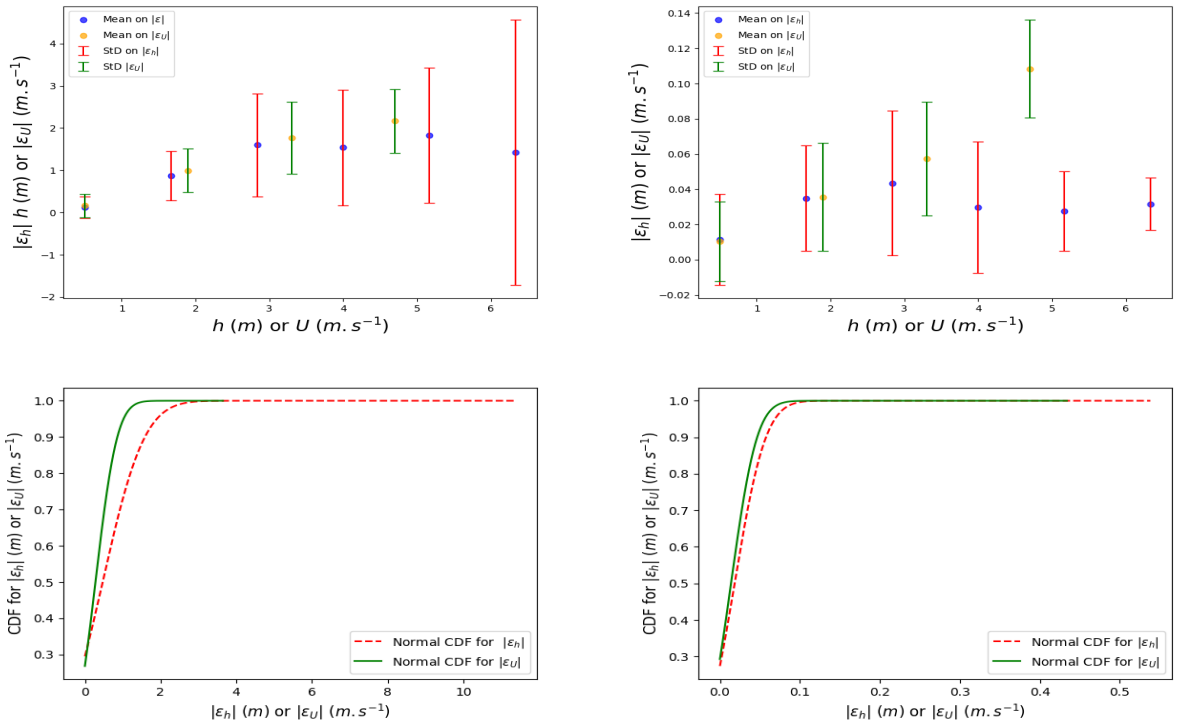

Figure 16: Non-simultaneous inflows case (real hydrographs). (Top) The histogram of the mean and the StD of spatial values of the absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$ at maximum flooding time with real inflow hydrographs (Fig. 4 right). (Left) with POD-NN method. (Right) with EA-POD-NN method. (Bottom) The CDF of the absolute absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$. (Left) with POD-NN method. (Right) with EA-POD-NN method

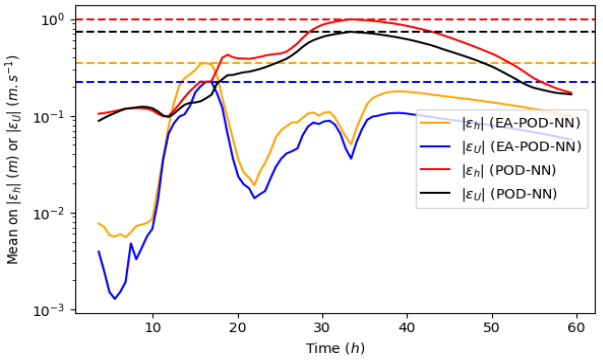

Figure 17: Non-simultaneous inflows case (real hydrographs). The mean on the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$ with respect to the real time with POD-NN and EA-POD-NN methods with real inflow hydrographs (Fig. 4 right). The horizontal dashed lines correspond to to the maximum value of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$.

with spatial patterns of errors at maximum flooding time (Fig. 21), error histograms on those spatial errors on predicted flow at max flooding time (Fig. 22), temporal variations of average spatial error (Fig. 23).

Using the same complexity (number of NNs parameters and CPU time), a comparison between Setup 1 and Setup 2 shows that the EA-POD-NN method ($N_{rb} = 5$ and $N_{er} = 10$) is more accurate than the classical POD-NN ($N_{rb} = 10$).

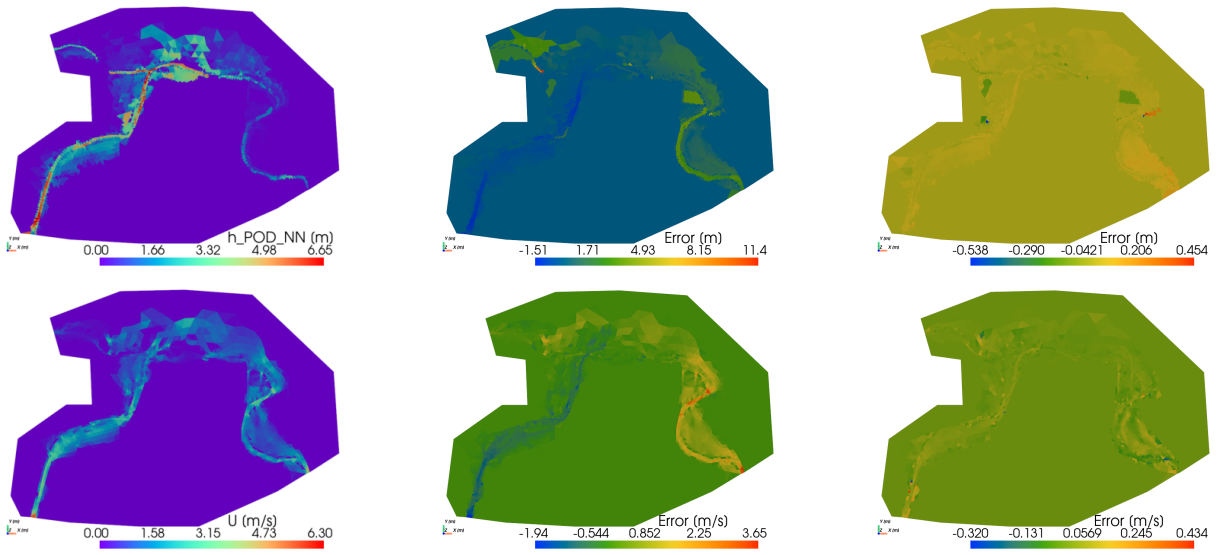

Figure 18: Non-simultaneous inflows case (triangle hydrographs). Comparison of the model outputs at maximum flooding time. (Top) For the water depth h. (Bottom) For the velocity norm U. (Left) The reference solution: 2D SW model with triangle inflow hydrographs for $\mu_{new} = (11.94, 922, 9.22, 198, 25)$, i.e. $\boldsymbol{u}_h(\mu_{new})$. (Middle) The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$. (Right) The difference ε_h or ε_U for EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$.

This is borne out by the spatial error patterns, which are lower for EA-POD-NN with Setup 1 than the standard POD-NN with Setup 2. Also, the histogram of the mean and the standard deviation of spatial values of $|\varepsilon_h|$ and $|\varepsilon_U|$ for the EA-POD-NN method are almost below 0.5m compared with those of Setup 2 (between 1m and 2m). The CDF of absolute error spatial average shows that the error with Setup 1 is lower than the one with Setup 2. Namely, the median value of error on h (or U) is 0.23 m (or 0.02 ms^{-1}) with Setup 2 and 0.028m (or 0.010) with Setup 1. The average of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$ with respect to the real time for Setup 1 are better compared to the ones obtained with Setup 2. Finally, Setup 3 of higher complexity with $N_{rb} = 63$, shows that POD-NN provides equivalent precision results as EA-POD-NN with Setup 1, but for a much larger NN, which here corresponds to a higher number of parameters than the number of data (snapshots), potentially resulting in over-fitting and thus hindering model generalization.
The results obtained through these experiments show that:

- The proposed EA-POD-NN method (Setup 1) is more accurate in prediction compared to the POD-NN method (Setup 2) for equivalent CPU time and memory cost (in the online phase).
- To reach results of similar accuracy, the POD-NN method requires a larger number of modes (here e.g. $N_{rb} = 63$), so a larger NN that is more memory-intensive. Additionally, this can easily lead to a larger number of parameters in the NN than the number of snapshots, potentially resulting in over-fitting and thus hindering model generalization.

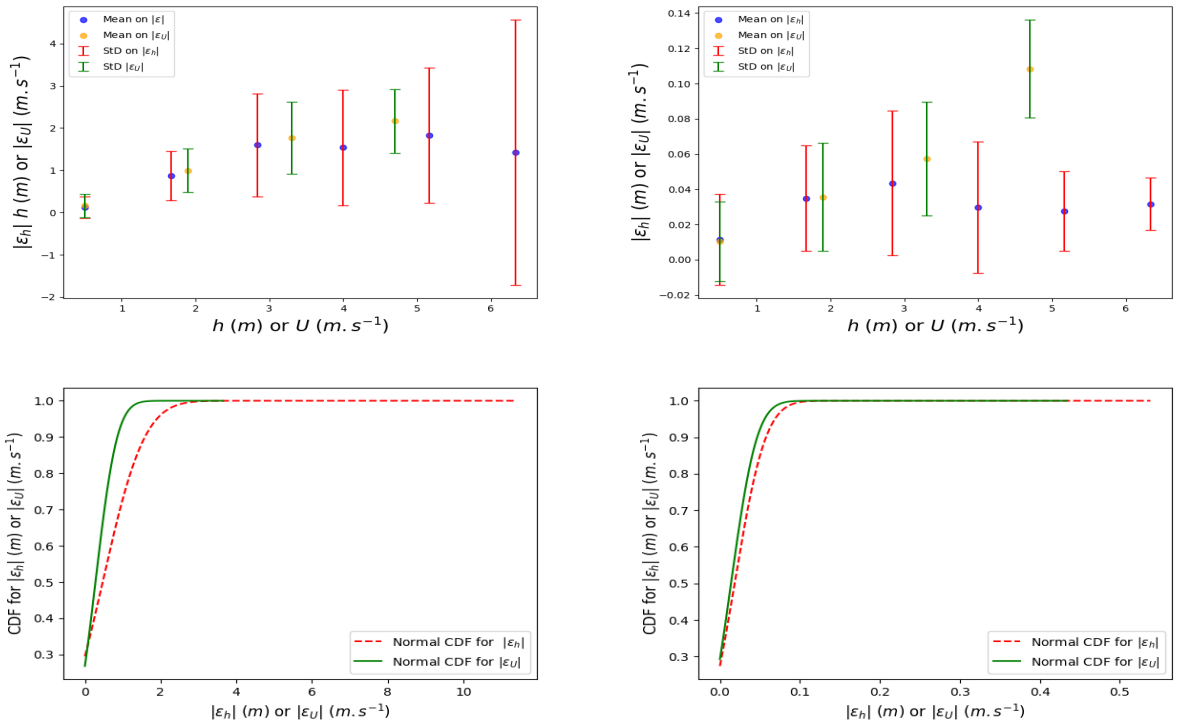

Figure 19: Non-simultaneous case (triangle hydrographs). (Top) The histogram of the mean and the StD of spatial values of the absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$ at maximum flooding time with triangle inflow hydrographs for $\mu_{new} = (11.94, 922, 9.22, 198, 25)$. (Left) with POD-NN method. (Right) with EA-POD-NN method. (Bottom) The CDF of the absolute absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$. (Left) with POD-NN method. (Right) with EA-POD-NN method

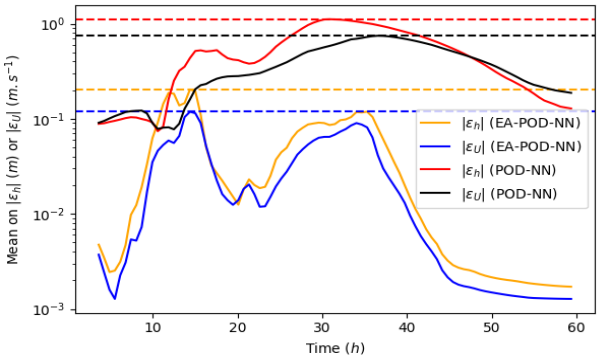

Figure 20: Non-simultaneous inflows case (triangle hydrographs). The mean on the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$ with respect to the real time with POD-NN and EA-POD-NN methods with triangle inflow hydrographs for $\mu_{new} = (11.94, 922, 9.22, 198, 25)$. The horizontal dashed lines correspond to to the maximum value of the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$.

Finally, the long-term advantage of having smaller latent spaces and therefore smaller NNs with the EA-POD-NN method will be to facilitate the explainability of the method. This makes the proposed EA-POD-NN method better suited for large applications e.g. over larger computational domains.

	Offline Phase: Algorithm 1				Online Phase: Algorithm 2		N_θ	Non over-fitting: $N_\theta < N_s$
Algorithm steps	**Step 1-2**	**Step 3-5**	**Step 6-7**	**Step 8//9**	**Step 1-2//3-4** for flood duration	**Memory (Mo)**	-	-
DassFlow	-	-	-	-	118.21 s	1287.34	-	-
EA-POD-NN $N_{rb} = 5$ **&** $N_{er} = 10$	1h10min	1h33min	1h31min	18min	5.46 s	126.96	$N_{\theta_1} = 4425$ & $N_{\theta_2} = 6910$ Total: $N_{\theta_1} + N_{\theta_2} = 11\,335$	✓
POD-NN $N_{rb} = 10$	1h10min	1h33min	-	18min	5.40 s	123.91	11 082	✓
POD-NN $N_{rb} = 63$	1h10min	1h33min	-	18min	5.62 s	165.63	54 593	✗

Table 3
CPU-time of different steps of the offline and online phases given in Algorithm 1, 2 of **Setup 1**: EA-POD-NN method with $N_{rb} = 5$, $N_{er} = 10$, **Setup 2**: POD-NN with $N_{rb} = 10$ and **Setup 3**: POD-NN with $N_{rb} = 63$ in the case where $N_s = 20\,025$. The CPU time (resp. memory cost) is indicated in the sixth (resp. seventh) column over the flood time window for (i) the FV simulation with DassFlow and for (ii) the online prediction at multiple times with the NNs based approaches. Computations were performed in 11th Gen Intel(R) Core(TM) i9-11950H, 2.60GHz with 32Gb of RAM capacity.

5. Conclusion

From the obtained results, and in line with the objectives of this research, the following few conclusions can be made. The basic POD-NN method as proposed in [18, 30], see also Section 3, enables to reduce the 2D SW model on a moderately complex case in the sense of the tested dynamics with simultaneous inflow hydrographs, parameterized in low dimension. However for more complex non-linear waves interactions like those produced by the two non-simultaneous inflows hydrographs (Section 4.3), the basic POD-NN method requires to be performed on much larger NNs therefore more CPU-time consuming, more memory requirements and larger datasets (equivalently greater number of snapshots). The proposed EA-POD-NN method, that additionally learns the projection error onto the RB, enables to reduce accurately the 2D SW model (comparisons based on the same parameter sampling) in actual low dimension bases (here $O(10)$ number of modes). In the present application, this enables to well simulate the complex flood case with the two non simultaneous hydrographs. The spatio-temporal hydraulic variables are sufficiently accurately predicted. Higher accuracy on simulated flow is obtained with the proposed EA-POD-NN method than with the standard POD-NN method when considering the same complexity, for equivalent CPU and memory cost in online phase. Achieving a specific level of accuracy with a smaller NN is of great interest for

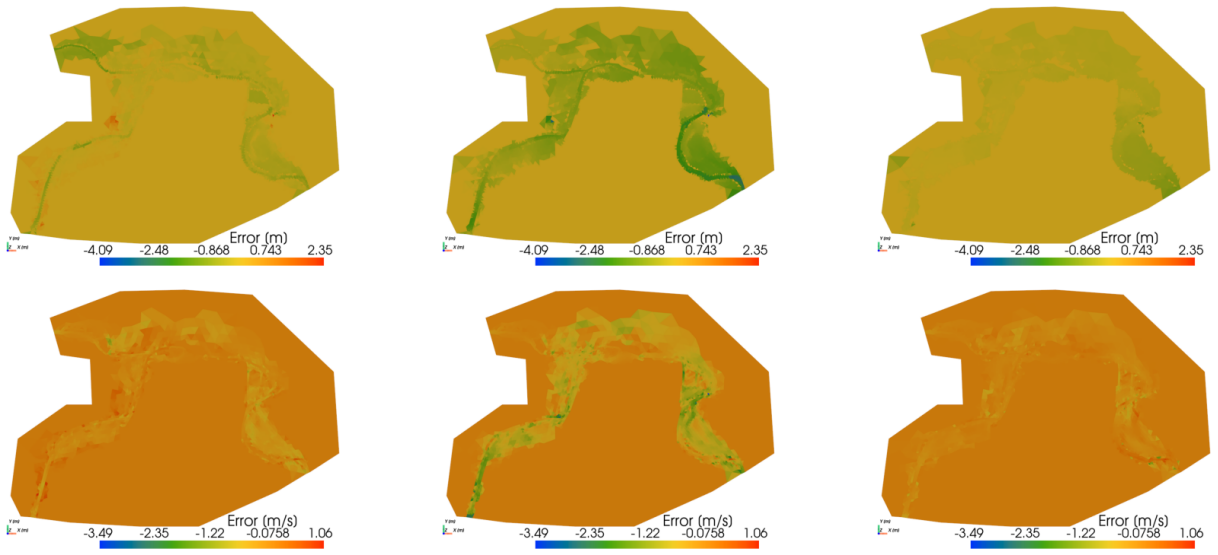

Figure 21: Spatial error patterns on predicted flow variables in the online phase. Comparison of complexity between (left) **Setup 1**: EA-POD-NN with $N_{rb} = 5$ and $N_{er} = 10$, (middle) **Setup 2**: POD-NN with $N_{rb} = 10$ and (right) **Setup 3**: POD-NN with $N_{rb} = 63$ in the non-simultaneous inflows case at maximum flooding time for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 11), (Top) Water depth h. (Bottom) Velocity norm U. (Left) **Setup 1** The difference ε_h or ε_U for EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$ with $N_{rb} = 5$ and $N_{er} = 10$. (Middle) **Setup 2** The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$ with $N_{rb} = 10$. (Right) **Setup 3** The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$ with $N_{rb} = 63$.

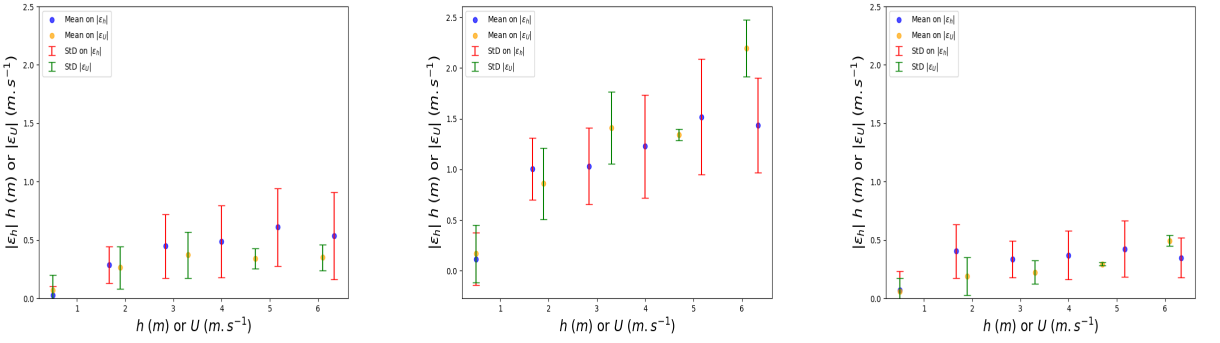

Figure 22: Error metrics in space and time on predicted flow variables in the online phase. Comparison of complexity between (left) **Setup 1**: EA-POD-NN with $N_{rb} = 5$ and $N_{er} = 10$, (middle) **Setup 2**: POD-NN with $N_{rb} = 10$ and (right) **Setup 3**: POD-NN with $N_{rb} = 63$ in the non-simultaneous inflows case at maximum flooding time for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 11). The histogram of the mean and the StD of spatial values of the absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$ at maximum flooding time. (Left) **Setup 1** The difference ε_h or ε_U for EA-POD-NN $\boldsymbol{u}_{h,EA-POD-NN}(\mu_{new})$ with $N_{rb} = 5$ and $N_{er} = 10$. (Middle) **Setup 2** The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$ with $N_{rb} = 10$. (Right) **Setup 3** The difference ε_h or ε_U for POD-NN $\boldsymbol{u}_{h,POD-NN}(\mu_{new})$ with $N_{rb} = 63$.

several reasons. This approach prevents over-fitting during the learning phase, thereby favoring better generalization of the obtained model and improved interpretability, see e.g. [22, 6]. When feasible, lower-dimensional and sparse representation systems are always preferable for these reasons [6]. The EA-POD-NN method, which we present, offers such advantages compared to the simpler POD-NN method, and the learning of projection error in the EA-POD-NN

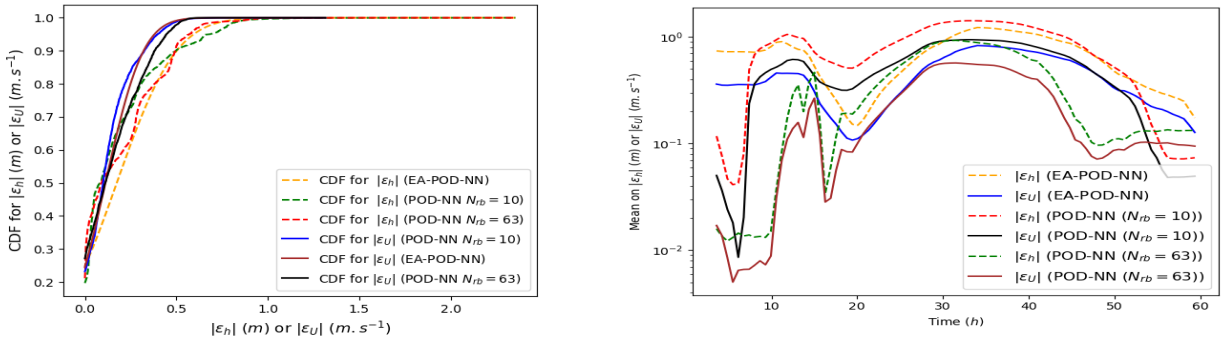

Figure 23: Comparison of complexity between **Setup 1**: EA-POD-NN with $N_{rb} = 5$ and $N_{er} = 10$, **Setup 2**: POD-NN with $N_{rb} = 10$ and **Setup 3**: EA-POD-NN with $N_{rb} = 63$ for $\mu_{new} = (8.66, 1100, 8.38, 300, 22.2)$ (red point in Fig. 6. (Left) The CDF of the absolute absolute difference $|\varepsilon_h|$ and $|\varepsilon_U|$. (Right) The mean on the absolute difference $|\varepsilon_h|$ or $|\varepsilon_U|$.

improves the predictive capability of the NN.

The proposed EA-POD-NN method is able to predict with a good accuracy the ROM solution of non linear hyperbolic systems, here the 2D SW model, solved by Finite Volumes, in complex real-world situations. Its interesting precision-to-cost ratio would enable tackling large dimensional real cases over larger computational domains. In the river hydraulics context, it could be of interest for tackling large floodplains at river networks scale (see e.g. [13] with simplified flow model or e.g. [23] with complete flow models). The obtained surrogate model could then be used as a digital twin [9, 26], allowing for rapid flood forecasts given the registered scenario. Immediate research work will pertain in improving the capability of the method to be applicable in higher dimensional input parameters. In a flood modeling context, this may be more complex inflow hydrographs or spatially distributed friction fields. This should enable to generalize its applicability to the variety of real hydrological signals and rivers-floodplains connectivities, as well as to optimization or uncertainty quantification tasks. Moreover, the required snapshots number may be reduced by enforcing the model residual to vanish at some points, as in the Physics Informed NNs methods for instance [25, 8].

Acknowledgments

The application of this computational sciences study has been inspired by scientific discussions led in the MUFFINS ANR project, ”MUlti- scale Flood Forecasting with INnovating Solutions”, ANR-21-CE04-0021-01. The authors acknowledge SCHAPI-DGPR, Météo-France and IGN for providing data used in this work.

References

[1] Shady E Ahmed, Omer San, Adil Rasheed, and Traian Iliescu. Nonlinear proper orthogonal decomposition for convection-dominated flows. *Physics of Fluids*, 33(12), 2021.

[2] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sci. Comput.*, 25(6):2050–2065, June 2004.

[3] François Bachoc, Fabrice Gamboa, Max Halford, Jean-Michel Loubes, and Laurent Risser. Explaining machine learning models using entropic variable projection. *Information and Inference: A Journal of the IMA*, 12(3):1686–1715, 2023.

[4] Joshua Barnett, Charbel Farhat, and Yvon Maday. Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility. *Journal of Computational Physics*, 492:112420, 2023.

[5] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. Miguel Silveira. *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms.* Model Order Reduction ; Volume 2. De Gruyter., Berlin ;, 2020-2021.

[6] MP Brenner, JD Eldredge, and JB Freund. Perspective on machine learning for advancing fluid mechanics. *Physical Review Fluids*, 4(10):100501, 2019.

[7] Joao G Caldas Steinstraesser, Vincent Guinot, and Antoine Rousseau. Modified parareal method for solving the two-dimensional nonlinear shallow water equations using finite volumes. *The SMAI journal of computational mathematics*, 7:159–184, 2021.

[8] Wenqian Chen, Qian Wang, Jan S Hesthaven, and Chuhua Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of computational physics*, 446:110666, 2021.

[9] Francisco Chinesta, Elias Cueto, Emmanuelle Abisset-Chavanne, Jean Louis Duval, and Fouad El Khaldi. Virtual, digital and hybrid twins: a new paradigm in data-based engineering and engineered data. *Archives of computational methods in engineering*, 27:105–134, 2020.

[10] Albert Cohen and Ronald DeVore. Approximation of high-dimensional parametric pdes. *Acta Numerica*, 24:1–159, 2015.

[11] Sourav Dutta, Matthew W Farthing, Emma Perracchione, Gaurav Savant, and Mario Putti. A greedy non-intrusive reduced order model for shallow water equations. *Journal of Computational Physics*, 439:110378, 2021.

[12] Nabil El Moçayd, Sophie Ricci, Nicole Goutal, Mélanie C Rochoux, Sébastien Boyaval, Cédric Goeury, Didier Lucor, and Olivier Thual. Polynomial surrogates for open-channel flows in random steady state. *Environmental Modeling & Assessment*, 23:309–331, 2018.

[13] A.S. Fleischmann, R.C.D. Paiva, W. Collischonn, V.A. Siqueira, A. Paris, D.M. Moreira, F. Papa, A.A. Bitar, M. Parrens, F. Aires, et al. Trade-offs between 1-D and 2-D regional river hydrodynamic models. *Water Resources Research*, 56(8), 2020.

[14] Pierre-André Garambois, Jérôme Monnier, and Villenave Lilian. Coupled 2D hydrologic-hydraulic catchment scale flood modeling with data assimilation capabilities: the DassHydro platform. In *Colloque SHF - "Prévision des crues et inondations, avancées, valorisations et perspectives"*, Toulouse,, France, 2023.

[15] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[16] Hojat Ghorbanidehno, Jonghyun Lee, Matthew Farthing, Tyler Hesser, Eric F Darve, and Peter K Kitanidis. Deep learning technique for fast inference of large-scale riverine bathymetry. *Advances in Water Resources*, 147:103715, 2021.

[17] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

[18] Jan S Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] John Leask Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pages 166–178, 1967.

[21] Jerome Monnier, Frédéric Couderc, Denis Dartus, Kévin Larnier, Ronan Madec, and J-P Vila. Inverse algorithms for 2D shallow water equations in presence of wet dry fronts: Application to flood plain dynamics. *Advances in Water Resources*, 97:11–24, 2016.

[22] Thomas Peters. *Data-driven science and engineering: machine learning, dynamical systems, and control: by SL Brunton and JN Kutz, 2019, Cambridge, Cambridge University Press, 472 pp.,£ 49.99 (hardback), ISBN 9781108422093. Level: postgraduate. Scope: textbook.*, volume 60. Taylor & Francis, 2019.

[23] L. Pujol, P.-A. Garambois, and J. Monnier. Multi-dimensional hydrological-hydraulic model with variational data assimilation for river networks and floodplains. *EGUsphere*, 2022:1–44, 2022.

[24] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

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

[26] Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *Ieee Access*, 8:21980–22012, 2020.

[27] Open source computational software DassFlow (Data Assimilation for Free Surface Flows). Math. institute of toulouse (imt) and inrae and insa and icube strasbourg. https://github.com/dasshydro, 2023.

[28] Răzvan Ştefănescu, Adrian Sandu, and Ionel M Navon. Comparison of POD reduced order strategies for the nonlinear 2D shallow water equations. *International Journal for Numerical Methods in Fluids*, 76(8):497–521, 2014.

[29] Maria Strazzullo, Francesco Ballarin, and Gianluigi Rozza. POD-Galerkin model order reduction for parametrized nonlinear time-dependent optimal flow control: an application to shallow water equations. *Journal of Numerical Mathematics*, 30(1):63–84, 2022.

[30] Qian Wang, Jan S Hesthaven, and Deep Ray. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *Journal of computational physics*, 384:289–307, 2019.

[31] Dunhui Xiao. Error estimation of the parametric non-intrusive reduced order model using machine learning. *Computer Methods in Applied Mechanics and Engineering*, 355:513–534, 2019.