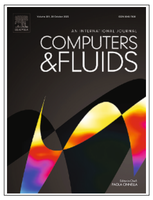
Computers and Fluids 304 (2026) 106884

Contents lists available at ScienceDirect

Computers and Fluids

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



Calibration of Manning's roughness coefficients for shallow-water flows on complex bathymetries using optimization algorithms and surrogate neural network models

Igor Gildas Metcheka Kengne[a], Vincent Delmas[b,c], Azzeddine Soulaïmani[a,*]

[a] Department of Mechanical Engineering, École de Technologie Supérieure (ÉTS), 1100 Notre-Dame Ouest, Montréal, H3C 1K3, QC, Canada
[b] Institut de Mathématiques de Bordeaux (IMB), Université de Bordeaux, CNRS, Bordeaux INP, F33400, Talence, France
[c] CEA Cesta, 15 avenue des sablières, Le Barp, France

ARTICLE INFO

Keywords:
Manning roughness coefficient
Ensemble model
Neural networks
Calibration
Metaheuristic optimization algorithms
Shallow-water equations
Multi-GPU
CUDA
Finite volumes

ABSTRACT

This paper presents an effective methodology for the automatic calibration of Manning's roughness coefficients, which are crucial parameters for modeling shallow free-surface flows. Traditionally determined through empirical methods, these coefficients are subject to significant variability, making their determination challenging, especially in flow areas with complex bathymetry. The conventional trial-and-error approach, widely used to select these coefficients, is often tedious and time-consuming, particularly in applications constrained by time and data availability. The proposed methodology aims to determine the optimal values of Manning's coefficients distributed over the flow domain while minimizing global discrepancies between simulations and field measurements. The calibration approach is formulated as an inverse optimization problem and addressed using metaheuristic optimization algorithms such as the Genetic Algorithm or Particle Swarm Optimization, combined with an ensemble model of deep neural networks. The database for training the neural networks is obtained using a newly developed finite volume-based shallow-water equations solver, parallelized on multiple GPUs, to generate large datasets of solutions for machine learning purposes. The performance of this approach is evaluated through various flow scenarios. Compared to conventional techniques, this methodology stands out for its simplicity, computational efficiency, and robustness. Additionally, Hybrid Particle Swarm Optimization (HPSO) proves to be particularly effective, notably for its speed. The developed codes are available at: https://github.com/ETS-GRANIT/CuteFlow.

1. Introduction

The prediction of free-surface flows, especially during high discharge periods, is crucial for developing effective water resource management policies and flood prevention strategies. Accurate predictions rely on computational models that simulate the behavior of free-surface flows. Over the years, numerous high-fidelity computational codes have been developed to model river flows by solving shallow-water equations (SWEs). Among the various parameters required for this modeling, Manning's roughness coefficient is particularly important for accurate prediction.

Generally denoted in the SWEs by the variable n, Manning's coefficient is an empirical parameter that allows hydraulic engineers to characterize the effect of bed topography roughness on river hydrodynamics. This coefficient varies considerably depending on factors such as river geomorphology and flow conditions [1,2]. This variability complicates its selection and can significantly impact the prediction accuracy of numerical models, leading to discrepancies between simulated and field-measured data.

Several approaches to identifying Manning's coefficients have been developed to improve simulation accuracy and ensure reliable forecasts. In [3], these approaches are categorized into three groups: identification through measurements, estimation of values, and value adjustment. The first category, which is very costly, involves indirect identification of Manning's coefficients in water bodies based on field measurements. To mitigate the time and resource costs associated with this approach, alternative procedures have been developed. Estimation methods, such as visual comparison with reference sections and the use of reference tables, have been adopted to facilitate this identification (Chow (1959); Henderson (1966); Barnes (1967); Urquhart (1975)). In 1967, Barnes identified the roughness characteristics of natural channels and provided photographs as well as typical river cross-sections with

* Corresponding author.
E-mail address: azzeddine.soulaimani@etsmtl.ca (A. Soulaïmani).

https://doi.org/10.1016/j.compfluid.2025.106884
Received 31 March 2025; Received in revised form 3 September 2025; Accepted 21 October 2025
Available online 25 October 2025

associated values of Manning's coefficients n. These photographs served as reference sources, allowing modelers to select appropriate values for this empirical parameter in their models. However, due to the diversity in river morphology, these values cannot be directly transferred solely based on photographs and visual comparisons. Therefore, various empirical formulations have been proposed for more practical applications [4]; Lane and Carlson (1953); Meyer-Peter and Muller (1948). Cowan (1956) proposed a formulation based on a detailed analysis of the effects of shape factors influencing roughness, formulating the coefficient n as an algebraic sum of these factors. While useful, these formulations have limitations due to their variation across sites and inadequacy for unstable flows. The third approach category, based on optimization search methods, has emerged to reduce uncertainty and enable the modeling of complex large-scale flow problems. It includes two methods: the traditional trial-and-error method, which is subjective and time-consuming [5–7], and methods based on numerical optimization. These methods combine hydrodynamic simulation models and optimization algorithms to reduce discrepancies between measured and predicted values of hydraulic variables at specified observation locations by adjusting Manning coefficients. Identifying Manning coefficients then becomes an inverse optimization problem to determine the optimal parameter n^* that minimizes the objective function, L, such that:

$$L(n^*) = \min_{n\in I} ||\Psi(n) - z||^2 \tag{1}$$

where n is constrained to belong to an interval of realistic values $I = [n_{\min}, n_{\max}]$, $\Psi(n)$ is the predicted hydrodynamic variable of interest (i.e. water level or velocity), z is the field-measured value, and $||\cdot||$ is the norm in the observation space [8].

The literature on parameter identification methods through optimization is highly diversified. Two main approaches for identifying roughness coefficients have been followed: gradient-based search methods [1,8–12] and metaheuristic algorithms [2,13–18]. Gradient-based methods rely on finding the optimum by following the gradients of the objective function. However, their sensitivity to initial predictions can lead to convergence towards local minima, depending on the complexity of the solution space. Additionally, they can be very intrusive, sometimes requiring specific and complex numerical formulations to compute the gradient [8]. In contrast, metaheuristic algorithms offer an alternative by exploring the search space more globally. However, they can be sensitive to tuning parameters and do not always guarantee convergence to the optimal solution.

More recently, the increasing integration of deep artificial neural networks with classical optimization algorithms has opened new perspectives in the field of hydraulics. The methodology proposed in this article capitalizes on the advantages of both approaches, offering a robust, efficient, and better-suited solution for managing uncertainties and outliers in data. This method proceeds in two phases. First, a numerical mapping between Manning's coefficients and hydraulic variables is established using a surrogate model, for which a set of high-fidelity solutions are initially obtained using a fast parallel numerical solver. Subsequently, this surrogate model, also called the reduced-order model, is integrated into an optimization process where various heuristics, classical, and hybrid algorithms, such as the Genetic Algorithm (GA & HGA) and Particle Swarm Optimization (PSO & HPSO), are used to determine the optimal values of Manning coefficients.

The rest of the paper is organized as follows. In Section 2, we start by describing our in-house multi-CPU/multi-GPU shallow-water equations solver, CuteFlow [19–22], which has been used to generate the large databases needed for machine learning purposes. The capabilities of CuteFlow are crucial to our work, enabling simulations that would otherwise be unfeasible. We then describe the methodology of our approach in Section 3, first presenting the surrogate modeling based on an ensemble of deep neural networks, and then the optimization algorithms and their features. Next, in Section 4, the applicability of this methodology is validated through three benchmark tests. Two of these are based on representative simple geometries and bathymetries, while the third is conducted on a real portion of a river with complex bathymetry. These tests include flow along a divergent channel, a free-surface flow in a river with non-submersible piers, and a calibration test on a hypothetical flow in the Mille-Îles River in Quebec. We summarize our findings in Section 5, where we also mention a performance test, detailed in an appendix.

2. A high-fidelity parallel solver for the shallow-water equations

The generation and preparation of a sufficiently large database of high-fidelity simulation solutions, needed to build a surrogate model (a neural network model in our case) of the full-order numerical solver, which will be used in the optimization stage, can be a very time-consuming task and is of10 a major factor in the convergence and quality of the resulting surrogate. In this section, we start by recalling the Shallow-water equations before briefly describing the multi-GPU parallel solver used to generated the large databases needed by the machine learning models.

2.1. Finite volume method for the shallow-water equations

We begin by recalling the two-dimensional Shallow-water equations

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot\mathbb{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U}), \tag{2}$$

with

$$\mathbf{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \quad \mathbb{F}(\mathbf{U}) = \begin{pmatrix} hu & hv \\ hu^2 + \frac{1}{2}gh^2 & huv \\ huv & hv^2 + \frac{1}{2}gh^2 \end{pmatrix}, \tag{3}$$

in which h is the depth of the water such that $h = \eta - b$ where η is the elevation of the free surface and b the bathymetry (see Fig. 1), u and v are the components of the depth-averaged velocity along the x and y directions, respectively, and g is the acceleration of gravity set to 9.81 m/s^2.

On the right-hand side, the source term $\mathbf{S}(\mathbf{U})$ contains both the bathymetric and the friction source terms and is written as

$$\mathbf{S}(\mathbf{U}) = \begin{pmatrix} 0 \\ gh(S_{b_x} - S_{f_x}) \\ gh(S_{b_y} - S_{f_y}) \end{pmatrix}, \quad \begin{pmatrix} S_{b_x} \\ S_{b_y} \end{pmatrix} = -\nabla b, \quad \begin{pmatrix} S_{f_x} \\ S_{f_y} \end{pmatrix} = \begin{pmatrix} \frac{n^2 u\sqrt{u^2+v^2}}{h^{4/3}} \\ \frac{n^2 v\sqrt{u^2+v^2}}{h^{4/3}} \end{pmatrix}, \tag{4}$$

where S_{b_x} and S_{b_y} are the slopes of the bathymetry function $b(x,y)$ in the x and y directions, respectively, S_{f_x} and S_{f_y} are the friction terms in the x and y directions, respectively, and n is the Manning roughness coefficient. The finite-volume formulation is then found by splitting the spatial domain Ω into N volumes ω_i such that $\bigcup_{i=1}^N \omega_i = \Omega$ and $\omega_i \cap \omega_j = \emptyset, \;\; \forall i \neq j$, and by integrating (2) over ω_i to get the discrete form.

$$|\omega_i|\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} + \sum_{e\in\mathcal{E}(i)} l_e \mathbf{F}_{ie} = |\omega_i|\mathbf{S}_i, \tag{5}$$

where $\mathbf{U}_i$ is the cell-averaged solution vector, $\mathcal{E}(i)$ is the set of edges of the cell i, l_e is the length of edge e, and $\mathbf{F}_{ie}$ is the discrete flux that passes through the edge e of cell i in the normal direction of the edge $\mathbf{n}_{ie}$. The discrete flux $\mathbf{F}_{ie}$ is found by solving a one-dimensional Riemann problem in the direction $\mathbf{n}_{ie}$. We use the classical HLLC Riemann solver [24–26] to compute the intercell flux. The friction source term contained in the right-hand side of $\mathbf{S}_i$ is computed in a semi-implicit manner, see [22]. For the bathymetric source term, we use the hydrostatic reconstruction from [27] to have a well-balanced discretization. Alternative approaches exist in the literature, including [28–31], which achieve high-order accuracy while preserving equilibrium states in finite-volume shallow water solvers. The second-order extension is performed in a fully discrete manner using the MUSCL-Hancock procedure with limited reconstructed polynomials as presented in [21].

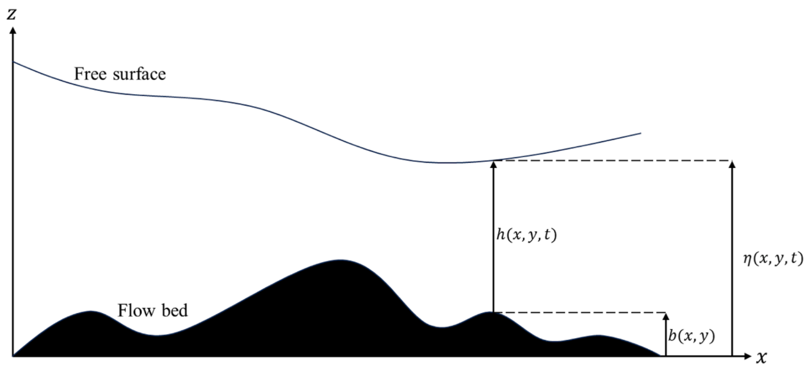


Fig. 1. Description of the variables adapted from [23].

2.2. Multi-GPU parallel solver

To accelerate the computations, we leverage a hybrid MPI/CUDA implementation to utilize multiple GPUs on the computer clusters. The development of this parallel solver has been described in previous papers, refer to [20,21,32] and we only recall the key aspects hereafter. The computational domain is divided into subdomains using the METIS library [33], and a single MPI process that controls its own dedicated GPU is used to compute the solution on each subdomain. Because the solution vector is stored inside the GPU's memory, we use a CUDA-*aware* version of OpenMPI [34] to minimize CPU-GPU memory exchanges and improve communication efficiency, as demonstrated in [21]. The partitioned subdomains are also renumbered to minimize slower internode communications and benefit from faster intranode communications. In order to improve file handling, a method using the Hierarchical Data Format (HDF) through the use of the CFD General Notation System (CGNS) library [35] was developed in [20] in order to store the entire domain decomposition including the overlap information and the solution data inside a single CGNS file. We finally leverage the *array jobs* from the Slurm workload manager on our computer clusters to launch multiple simulations simultaneously. In this manner, we can generate solution databases in just a few hours of real-time. This process was used in [36] and is also used here.

3. Methodology

The calibration methodology outlined in this paper involves two main stages: the development of a surrogate model for the high-performance computing code and the calibration of Manning's coefficients using numerical optimization algorithms.

3.1. Surrogate modeling

Surrogate models are simplified mathematical and computational frameworks that replicate the input-output relationships among parameters within a complex high-fidelity model. In the realm of numerical simulations, these models play a crucial role in swiftly predicting outputs for new, unseen input values without necessitating the full original computational model. In this study, deep neural networks (DNNs) are used to build our surrogate model in place of the high-fidelity solver. To mitigate the uncertainties associated with neural network approximations, this surrogate model is based on an ensemble of DNNs.

3.1.1. Deep neural networks

Deep neural networks are a generalization of artificial neural networks (ANNs), inspired by the intricate workings of the human brain. At the core of an ANN lies the neuron, also known as a node. Neurons receive input signals from other neurons or external sources, undergo computations on these inputs, and generate an output signal. Neurons are organized into layers, forming a hierarchical structure, and are interconnected in a feed-forward manner. The initial and final layers are called the input and output layers, respectively, while those in between are termed hidden layers. The diagram in Fig. 2 below provides a visual representation of a single-layer artificial neural network.

DNNs are characterized by multiple hidden layers. The output response of a DNN is computed through a process known as forward propagation, wherein input data progresses through the network layer by layer, undergoing computations at each stage. Values from each layer are transmitted to nodes in successive hidden layers by being multiplied by specific weights. Ultimately, the final output is derived from the output layer after passing through all the hidden layers. The new value in each node is determined by

$$a_j^{(k)} = \sigma(z_j^{(k)}), \quad \forall j \in \{1, \dots, n_k\}, \tag{6}$$

with

$$z_j^{(k)} = \sum_{i=1}^{n_{k-1}} w_{ji}^{(k)} a_i^{(k-1)} + b_j^{(k)}, \quad \forall j \in \{1, \dots, n_k\}, \tag{7}$$

where

- k represents the layer index and n_k the size of the activation vector $\mathbf{a^k}$,
- $a_j^{(k)}$ is the activation unit of the node j,
- $w_{ji}^{(k)}$ are the weight parameters associated with each node i whose state is given by $a_i^{(k-1)}$ with $a_i^{(0)} = x_i^{(in)}$,
- $z_j^{(k)}$ is the net input value at node j,
- $\sigma(\cdot)$ is the activation function that helps to introduce the non-linearity and has to be derivable. The ReLU function and hyperbolic tangent are the most widely used activation functions for the hidden layers, while the linear function is used for the output layer in regression problems.

The number of nodes in the input and output layers is equal to the number of input and output variables, respectively. Eqs. (6) and (7) can be rewritten in the following vector relationship:

$$\mathbf{a}^{(k)} = \sigma(\mathbf{z}^{(k)}), \tag{8}$$

with

$$\mathbf{z}^{(k)} = \mathbf{W}^{(k)}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k)}, \tag{9}$$

where

- $\mathbf{z}^{(k)}$ is the net input vector of the kth layer,
- $\mathbf{W}^{(k)}$ is the weight matrix connecting layer $k-1$ and k,
- $\mathbf{b}^{(k)}$ is the bias vector (composed of one bias unit per hidden node).

In the same way, the output layer is calculated by the relationship below, in which L represents the number of hidden layers:

$$\mathbf{z}^{(out)} = \mathbf{W}^{(out)}\mathbf{z}^{(L)} + \mathbf{b}^{(out)}, \tag{10}$$

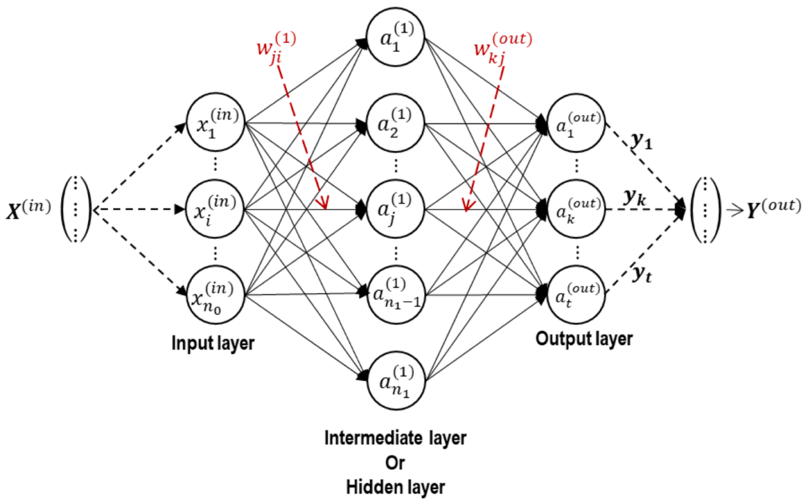


Fig. 2. Illustration of a single layer feed-forward ANN.

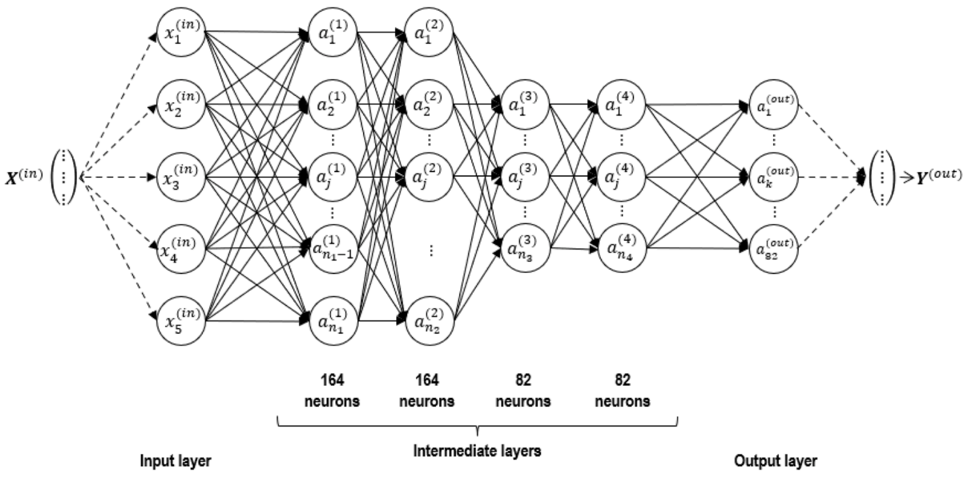


Fig. 3. Example of a four-layer deep neural network used in our study.

which is then passed to the activation function ϕ of the output layer (usually linear) to calculate the model response as follows:

$$\mathbf{Y}^{(out)} = \phi(\mathbf{z}^{(out)}). \tag{11}$$

In this study, after experimenting with several architectures, neural networks with four hidden layers were considered. Each neural network is developed using a dataset of high-fidelity flow solutions corresponding to a sample of Manning's coefficients generated randomly within an interval of realistic values. More precisely, the flow domain is subdivided into m zones, where each zone i potentially has a distinct Manning's value n_i $(i = 1, ., m)$ to be identified, constrained to $n_{\min} \leq n_i \leq n_{\max}$. A vector of Manning's coefficients is denoted by $\mathbf{n} = (n_1, \dots, n_m)$ and constitutes an example of the input parameters to the neural networks. A dataset of Manning's vectors is generated randomly (for instance, using a uniform distribution and a sampling algorithm, such as Sobol's [37]). The flow solver is then used to compute the corresponding flow solutions. The output dataset is constructed by extracting the hydraulic variables of interest (water level or velocity) at the location points where measurements are taken.
Fig. 3 shows an example of a four-layer deep neural network. In this configuration, the input layer consists of $m = 5$ neurons, corresponding to the number of distinct Manning's zones, while the output layer encompasses $t = 82$ nodes, representing the hydraulic variable of interest at specified locations in the domain. The neural network takes Manning's roughness vectors as parameters, passes them through the hidden layer, and predicts the hydraulic variables. The selection of the number of hidden layers and the number of neurons within each layer was refined through several tests guided by the observation of validation and test errors. To determine the weights and bias parameters, the network undergoes training on the dataset to minimize the loss (error) function, formulated as a mean square error [38]. In this process, 70 % of the data was allocated for training the neural network, 15 % was reserved for model validation, and the remaining 15 % was used to test the efficiency of the neural model. To build our neural network model, we used Matlab's Deep Learning Toolbox User's Guide [39].

3.1.2. *Ensemble of models*

The integration of multiple predictions from various neural network models has proven to be an effective strategy for enhancing performance over a single network [40]. Known as ensemble modeling in the field of machine learning, this approach involves merging predictions from multiple models to generate a more reliable final prediction. With each model associated with a prediction error, the objective of ensemble modeling is to reduce this error by independently training several networks, fed with the same data, and then combining their results. When handling datasets, different configurations of neural networks can be obtained by adjusting various parameters, such as the number of layers, the number of neurons per layer, and the learning optimization algorithm.A simple yet effective method involves utilizing multiple random initial weights [36,38,41]. Given p (typically between 5 and 12) neural networks, the prediction of the ensemble model is characterized by the Gaussian probability distribution with the mean $\mathbf{Y}_{ens}^{(out)}$, such that:

$$\mathbf{Y}_{ens}^{(out)} = \frac{1}{p}\sum_{i=1}^{p}\mathbf{Y}_{i}^{(out)} \tag{12}$$

with $\mathbf{Y}_{i}^{(out)}$ the response vector generated by the ith network. In our study, ensemble models were constructed by merging 10 distinct neural networks, each created with randomly varying weight initializations.

3.1.3. Dataset preparation

The first step in high-fidelity modeling involves creating an appropriate data sample. In this work, samples of Manning's coefficients of size N were generated from Sobol sequences. Each example in the sample consists of a vector (m-tuples) which is subsequently utilized to solve the shallow-water equations using the high-fidelity solver. The results of interest are extracted to serve as the output data for the neural network training and the construction of the ensemble surrogate model. Finally, optimization algorithms use this surrogate model for the calibration process, i.e., to determine the optimal values of the Manning coefficients by minimizing the objective function. Algorithm 1 summarizes the various steps of the proposed dataset preparation and Manning's roughness coefficients' calibration process.

Algorithm 1 General algorithm.

```
1. Stage 1: Generate a dataset of $N$ high fidelity solutions
    (a) Generate a sample of $N$ Manning's vectors $\mathbf{n} : (\mathbf{n_1}, \mathbf{n_2}, \dots, \mathbf{n_N})$.
    (b) For each sample, run the flow solver.
    (c) Extract from the $N$ solutions the set of flow variables (water, height, velocity, etc.) of interest, denoted as $\mathbf{\Psi} = (\psi_1, \psi_2, \dots, \psi_M)$, at $M$ predefined locations: $(\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_M})$.
2. Stage 2: Construct the surrogate model
    (a) Build a neural network architecture to approximate the mapping $\mathbf{n} \to \mathbf{\Psi}$.
    (b) Train the network $p$ times with different initializations using $\mathbf{n}$ as input and $\mathbf{\Psi}$ as output.
    (c) Get the $p$ predictions $\mathbf{Y}_i^{(out)}$.
    (d) Compute the ensemble prediction $\mathbf{Y}_{ens}^{(out)} = \frac{1}{p}\sum_{i=1}^{p}\mathbf{Y}_i^{(out)}$.
3. Stage 3: Optimization using surrogate model
    (a) Define the objective function (Eq. 18) as the discrepancy between the surrogate model prediction and reference/observed data.
    (b) Choose and configure an optimization algorithm (e.g., GA, PSO, or hybrid with fmincon), including population size, iteration limit, and convergence criteria.
    (c) Run the optimization algorithm.
        At each iteration, the optimizer:
            • generates a candidate Manning's vector $\mathbf{n}$.
            • Feeds $\mathbf{n}$ into the surrogate model.
            • Computes the predicted hydrological response $\mathbf{\Psi}$.
            • Calculates the objective function value based on prediction error.
    (d) Repeat until the termination criteria of the optimizer are met (e.g., convergence tolerance or maximum number of iterations).
    (e) Get the calibrated Manning's coefficient vector $\mathbf{n}^*$ that minimizes the objective under the defined constraints (Eq. 19).
```

3.2. Optimization algorithms

To calibrate Manning's roughness coefficients, we studied two optimization algorithms and their hybrid forms: the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO) algorithm. These algorithms, implemented using Matlab's optimization toolbox, are briefly described in the following subsections.

3.2.1. Genetic algorithm (GA)

Genetic Algorithms (GAs) are heuristic and adaptive search algorithms inspired by natural selection. They were first proposed by John Holland (Holland, J.H. (1992), as cited in [41]), and they rely on the principle of Darwinian evolution by natural selection. GAs mimic the process of human genetics, where hereditary traits are passed from parent to offspring, with the basic unit being the gene. Classified as a guided random search method, GAs are among the most prevalent optimization techniques, effectively leveraging randomness to address problems with robustness.

A GA consists of five key components working together iteratively, as illustrated in Fig. 4: initialization, selection, crossover, mutation, and termination. Initially, a population ($\mathbf{N_p}$) of individuals, typically encoded in binary strings, is randomly generated while adhering to boundary constraints. This population serves as the first generation and undergoes an eligibility test, in which a fitness function evaluates each individual. Individuals with higher scores are selected and undergo various transformations, including crossover and mutation, to produce improved offspring. Once these transformations are completed, the variable values are computed by decoding the binary strings and then evaluated using the fitness function. The process repeats until a termination criterion is reached, which could be reaching a maximum number of iterations, finding a satisfactory solution, or exceeding a predefined computational cost.

As in natural processes, the crossover step involves exchanging a sequence between two selected chromosomes in the population based on a crossover probability (P_{cr}) to then generate two descendants. Mutation, on the other hand, randomly alters specific parts of an individual, with the mutation frequency determined by the mutation probability (P_{mut}). In this work, the genetic algorithm is implemented using functions integrated into the MATLAB software toolbox. MATLAB's GA supports various data types for representing individuals, including binary strings and floating-point (double) values. In our implementation,

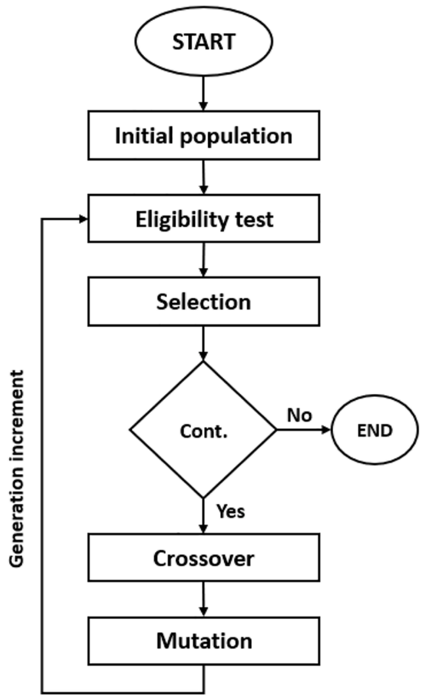


Fig. 4. Genetic algorithm flowchart.

Table 1
Implementation parameters of the genetic algorithm.

Population size (N_p)	25
Maximum number of generations	50
Selection method	Roulette wheel
Crossover probability (P_{cr})	0.8
Mutation probability (P_{mu})	0.1
Tolerance	1×10^{-6}

Table 2
Implementation parameters of the particle swarm optimization.

Swarm size (N_p)	25
Maximum number of iterations	50
Best personal position weighting coefficient (c_1)	1.5
Best-known position of the global swarm weighting coefficient (c_2)	1.5
Function tolerance	1×10^{-6}

instead of the classical binary encoding, individuals are represented using decimal (double precision) values. This choice allows for a more accurate representation of real-valued parameters, such as Manning's coefficient, by avoiding quantization errors associated with binary encoding. Moreover, MATLAB is optimized for operations on floating-point arrays, so this representation improves computational efficiency by eliminating the need for binary-to-decimal conversions and enabling faster, native vectorized processing. The parameters considered in our study are grouped in Table 1 below:

3.2.2. Particle swarm optimisation (PSO)

The Particle Swarm Optimization (PSO) technique is a method inspired by the social behavior of groups of animals or insects, such as bird flocks or fish schools. Initially proposed in 1995 by Eberhart and Kennedy [41], this approach also serves as an evolutionary method to solve global optimization problems. It begins by generating a random population, where each particle represents a candidate solution - in our case, a set of Manning's roughness coefficients - and then iteratively searches for the optimal solution by evaluating each solution's quality using a fitness function. Unlike genetic algorithms, PSO is distinguished by the simplicity of its rules and the absence of transformation steps such as crossover or mutation. In this animal behavior-based approach, each individual in the crowd is termed a particle and is characterized by its position and velocity in a multidimensional search space. The position corresponds to specific values of the calibrated parameter (i.e., the Manning's coefficient), and the velocity governs how these values are ajusted from one iteration to the next.After evaluating each solution's fitness using the surrogate model, which rapidly estimates the hydrological response (such as water depth or flow velocity) for each candidate Manning's coefficient, the positions and velocities of each particle in the global search space are updated based on their personal best (p_{best}) and the global best (g_{best}) solutions found so far, using the following two equations: [42]:

$$v_i^{k+1} = v_i^k + c_1 r_1 (p_i^k - x_i^k) + c_2 r_2 (g_{best} - x_i^k), \tag{13}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \tag{14}$$

where v_i^{k+1} is the new velocity of particle i at time $k+1$, x_i^k is the former position of particle i, r_1 and r_2 are random variables taken in $[0,1]$, c_1 and c_2 are weighting coefficients for the best personal position p_i^k and the best-known position of the global swarm g_{best}, respectively, such that

$$g_{best} = \min(p_i^k) \tag{15}$$

For our study, the values of c_1 and c_2 have been fixed both at 1.5. Fig. 5 shows the flowchart of the particle swarm optimization algorithm and Table 2 summarizes the implementation parameters.

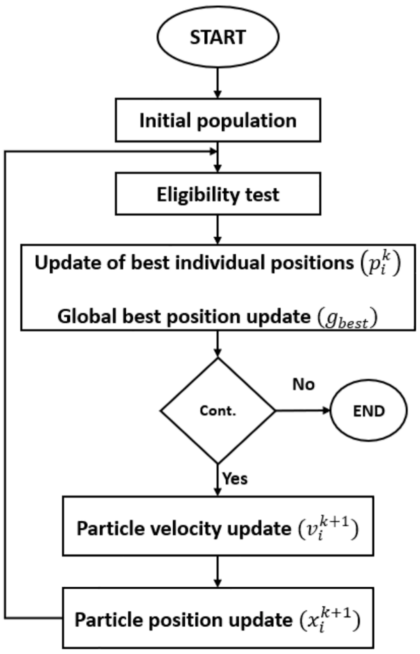

START
Initial population
Eligibility test
Update of best individual positions (p_i^k)
Global best position update (g_{best})
Cont.
No
END
Yes
Particle velocity update (v_i^{k+1})
Particle position update (x_i^{k+1})

Fig. 5. Particle swarm optimization flowchart.

3.3. Hybrid algorithms

In optimization, hybridization involves combining two or more algorithms that solve the same problem, capitalizing on their complementary strengths and mitigating individual weaknesses. Metaheuristic algorithms, such as GA and PSO, are useful optimization methods for exploring a search space but often face challenges in exploitation and in converging to the optimal solution. These methods may require numerous function evaluations to achieve convergence. To accelerate this process, a hybrid approach has been developed by integrating a gradient-based method, specifically Matlab's *fmincon* function, into these evolutionary algorithms. This synergistic combination leverages the ability of metaheuristics to navigate complex, high-dimensional spaces while utilizing the efficiency and precision of gradient-based optimization for local searches.

There are various methods to integrate a local search strategy into optimization heuristics like GA or PSO [43]. The approach proposed in this study is outlined in two steps below:

1. Exploration phase: GA or PSO is used to explore the search space and identify an initial set of potential solutions (e.g., Manning vector $\mathbf{n_0}$). The metaheuristic search is intentionally limited to a small number of iterations and a small population size to allow for quick exploration without incurring excessive computational cost. In this study, the number of iterations and the population size are set to 5 and 10, respectively.
2. Explotation phase: The best candidate solution found in the first phase is then passed to the *fmincon* function as a starting point for local optimization. *fmincon* is a gradient-based algorithm that by default, uses the interior-point method; an approach well-suited for constrained nonlinear problems and effective in handling both small, dense problems and large, sparse ones. The algorithm estimates gradients using finite differences and is capable of converging reliably even from distant starting points; however, providing a good initial guess, as done here, significantly improves its efficiency and convergence rate. In this study, *fmincon* refines the candidate solution by minimizing the objective function until it meets the termination criteria. This criterion is defined by a convergence threshold (e.g. changes in the objective function below a tolerance of 1^{-8}).

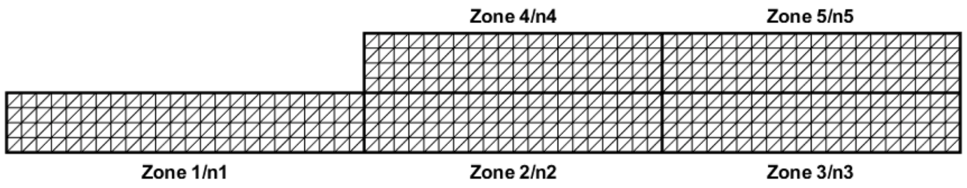


Fig. 6. Flow in a divergent canal mesh, segmented into 5 subdomains.

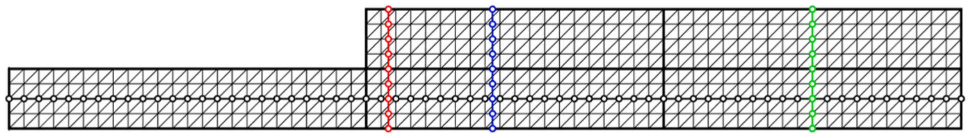


Fig. 7. Flow in a divergent canal: location of data measurement stations.

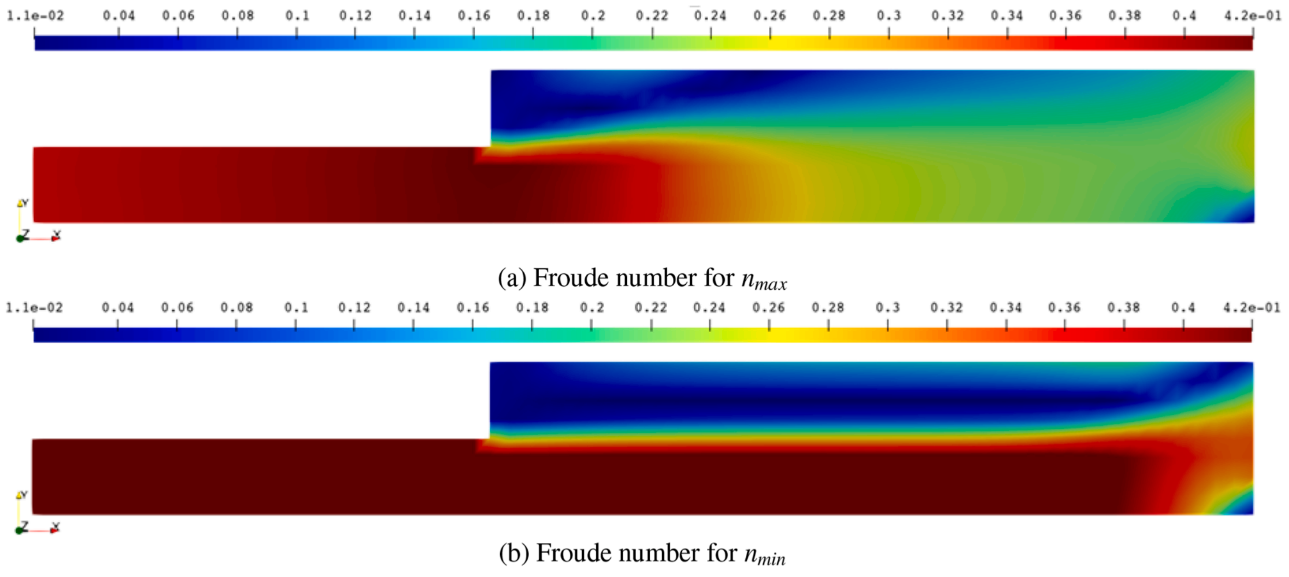


Fig. 8. Numerical results of the preliminary flow study: Froude number- divergent channel for ($n_{\min} = 0.01$; $n_{\max} = 0.05$, ($F_r < 1$)).

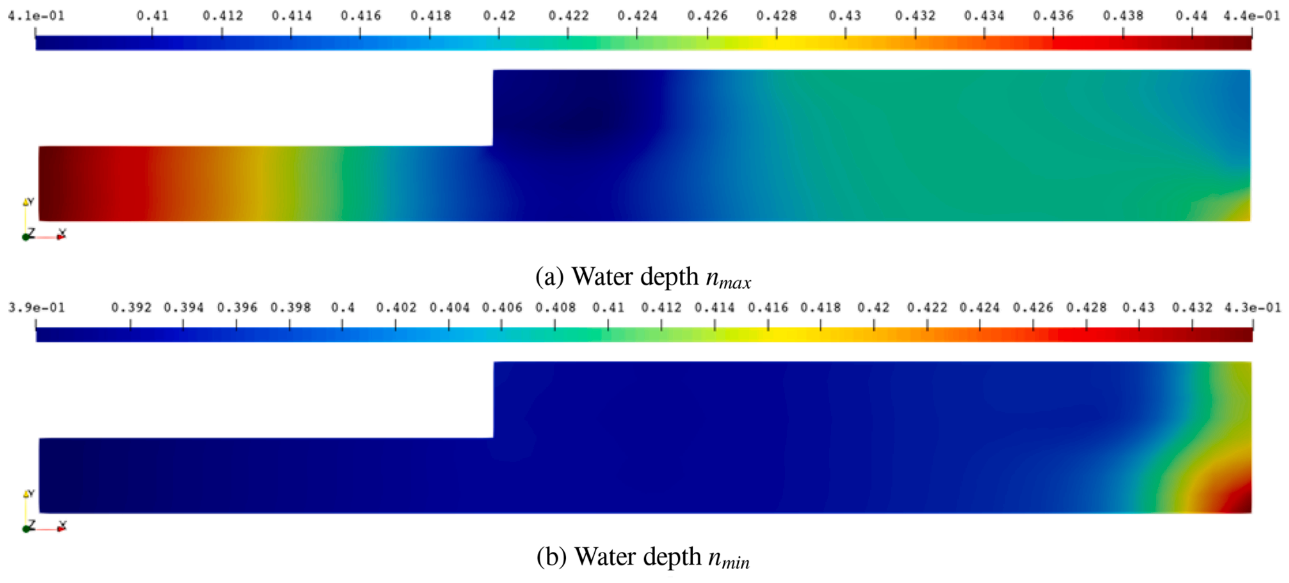


Fig. 9. Numerical results of the preliminary flow study: Water depth - diverging channel for ($n_{\min} = 0.01$; $n_{\max} = 0.05$).

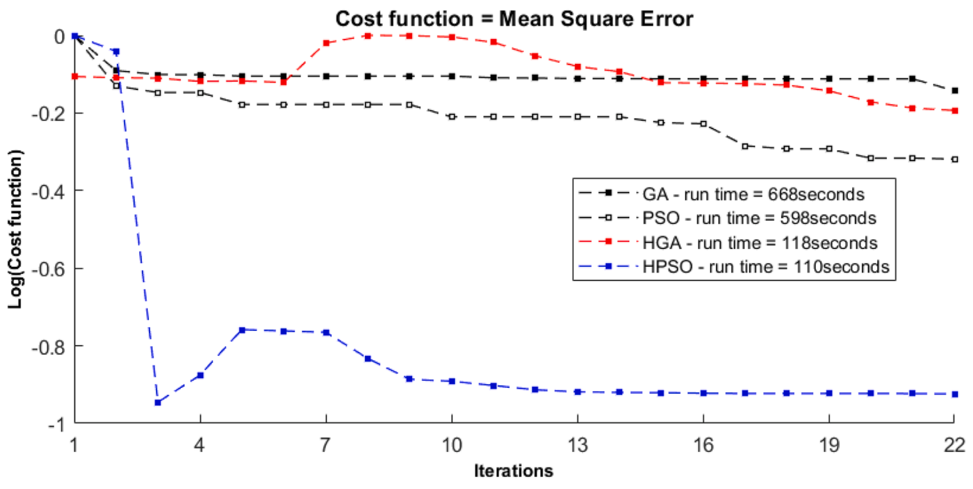


Fig. 10. Flow in a divergent channel. Convergence history for the GA, HGA, PSO and HPSO.

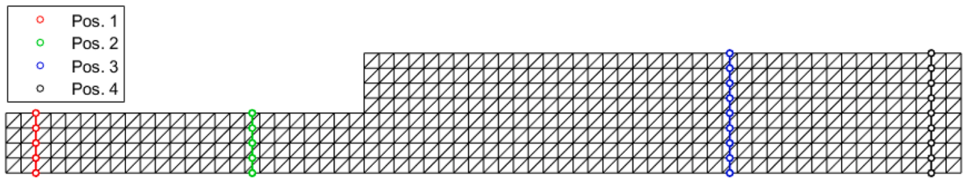


Fig. 11. Different measurement line location for the second transversal line.

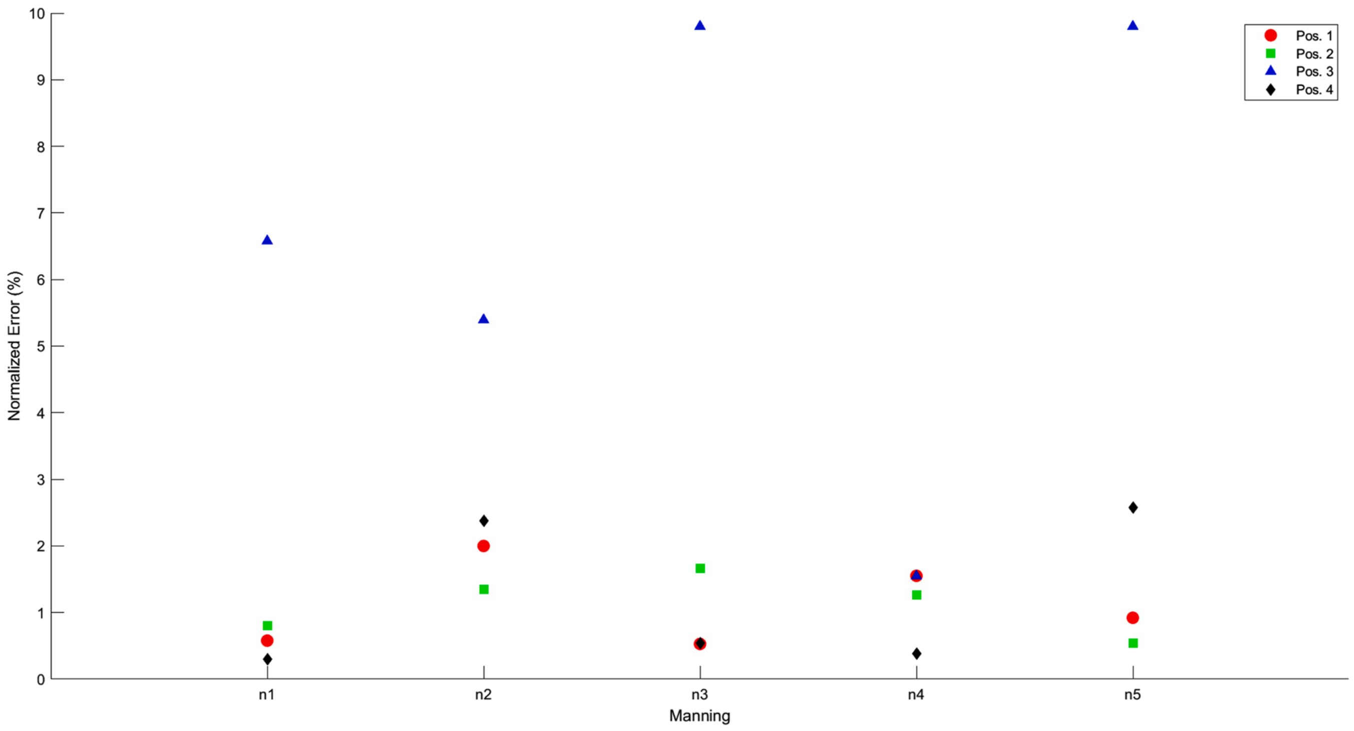


Fig. 12. Normalized error on Manning parameters with respect to measurement points location.

This hybrid optimization framework enhances both speed and precision, resulting in faster convergence, more accurate solutions, and improved robustness across various optimization problems.

3.3.1. Objective function

Considering the predicted variables $\mathbf{Y}^{(out)}_{ens}$ and the observed variables $\mathbf{Y}^{(obs)}$, the objective function can be defined as a mean squared error function (MSE). The solution to the calibration problem is a real vector $\mathbf{n}^* \in \mathbb{R}^{\mathbf{m}}$, and each component of this vector is constrained to an appropriate interval. Optimization aims at determining the $\mathbf{n}^*$ that minimizes the objective function:

$$f^{obj}(\mathbf{n}) = \frac{\mathbf{1}}{\mathbf{N_p}} \sum_{\mathbf{i=1}}^{\mathbf{N_p}} (\mathbf{Y}^{(\mathbf{out})}_{\mathbf{ens}} - \mathbf{Y}^{(\mathbf{obs})})^{\mathbf{2}}, \tag{16}$$

under constraints

$$n_i \in [0.01, 0.05] \quad \forall\, 1 \leq i \leq m, \tag{17}$$

Table 3
Flow in a divergent channel: Calibration parameters.

Number of mesh elements	832
Number of mesh nodes	4893
Number of zones	5
Sample size	256
Ensemble model size	10
Number of measurement stations	108
Coefficient of variation **CV**	5 %

Table 4
Flow in a divergent channel: Roughness coefficients in each zone.

Zones	Bathymetry	Search space	Ground truth
n_1	Concrete	$[1.26 - 1.54] \times 10^{-2}$	1.40×10^{-2}
n_2	Sand	$[1.53 - 1.87] \times 10^{-2}$	1.70×10^{-2}
n_3	Rock cuts	$[2.26 - 2.75] \times 10^{-2}$	2.50×10^{-2}
n_4	Cobble	$[3.16 - 3.84] \times 10^{-2}$	3.50×10^{-2}
n_5	Gravel	$[2.53 - 3.07] \times 10^{-2}$	2.80×10^{-2}

Table 5
Flow in a divergent channel: Results from the Genetic algorithm, Hybrid genetic algorithm, Particle swarm optimization and Hybrid particle swarm optimization.

Zones	n_1	n_2	n_3	n_4	n_5
Ground truth ($\times 10^{-2}$)	1.40	1.70	2.50	3.50	2.80
Genetic Algorithm (GA)					
Predictions ($\times 10^{-2}$)	1.533	1.650	2.432	3.516	2.747
Relative error (%)	9.50	2,94	2.72	0.46	1.9
Hybrid Genetic Algorithm (HGA)					
Predictions ($\times 10^{-2}$)	1.406	1.686	2.486	3.505	2.806
Relative error (%)	0.43	0.82	0.56	0.14	0.21
Particle Swarm Optimization (PSO)					
Predictions ($\times 10^{-2}$)	1.396	1.619	2.459	3.478	2.921
Relative error (%)	0.29	4.76	1.64	0.63	4.32
Hybrid Particle Swarm Optimization (HPSO)					
Predictions ($\times 10^{-2}$)	1.407	1.698	2.488	3.505	2.805
Relative error (%)	0.5	0.12	0.48	0.14	0.18

with N_p representing the number of observation points, $\mathbf{Y}^{(out)}_{ens}$ the vector of values predicted by the surrogate model, and $\mathbf{Y}^{(obs)}$ the values of the ground solution.

4. Numerical experiments

Three benchmark tests are proposed to evaluate the performance and robustness of the developed methodology and algorithms for the efficient calibration of Manning's coefficients, specifically within the context of sub-critical flows. For each test, we generate a *ground-truth* flow solution in a domain divided into m sub-domains, each with assigned *ground-truth* values n_i. The dataset for training the ensemble model is generated as described in Section 3.1.3. The input Manning's samples were generated using Sobol's algorithm, assuming a uniform distribution around the reference values and a coefficient of variation (CV) set at 5 % or 10 %. The optimization algorithms use the established Manning's intervals for their search domains and start the iterative process from arbitrary values.

For each of the given reference tests, results were obtained from an ensemble model based on the water depth at the measurement points. Simulations were carried out using 4 GPUs on the GRAHAM cluster of the Canadian Digital Research Alliance, where Cuteflow was hosted. Simulation results were then visualized and extracted using ParaView, version 5.11.2. In addition, neural networks and optimization algorithms were developed using Matlab libraries and built-in functions. Finally, the calibration tests were performed on a computer equipped with an Intel(R) Xeon(R) E3-1225 v5 @ 3.30 GHz CPU, a 9.9 GB NVIDIA Quadro K420 graphics card, and 16 GB of RAM.

4.1. *Flow in a divergent channel*

The aim is to evaluate the proposed calibration methodology on a simple, variably-shaped domain. The selected channel, previously studied in [8], spans 16 m, comprising a 6-meter long, 1-meter wide narrow section and a 10-meter long, 2-meter wide broader section. The bottom slope is 10^{-3}, with an exit depth of 0.2 m. The domain is discretized using a structured triangular mesh of 832 elements and 489 nodes, as shown in Fig. 6.

Before calibrating Manning coefficients, a crucial preliminary step is to thoroughly analyze the flow model to establish a robust knowledge base, including understanding the flow hydrodynamics and identifying areas for data comparison. In this study, these analyses considered extreme values (min and max) of the Manning roughness coefficient. The results for the divergent channel case, shown in Figs. 8 and 9, provide essential insights for effective calibration and help to identify suitable measurement locations (Fig. 7).

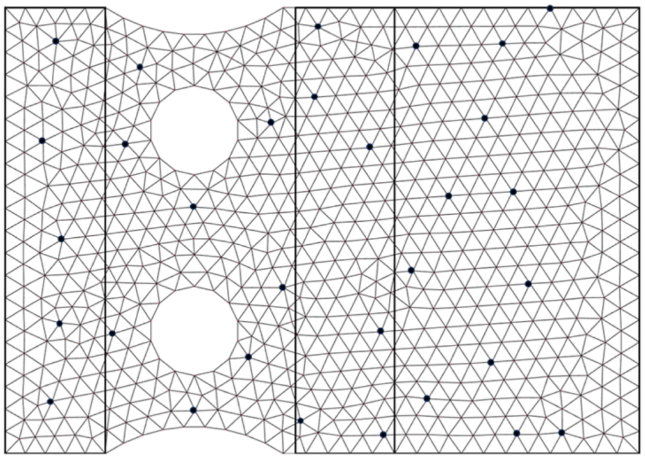


Fig. 13. Flow in a river with piers. Mesh decomposition divided into 4 subdomains; locations of data measurement stations.

As input parameters and starting values for this test, a flow rate of 0.50 m^3/s is imposed at the inlet, and a water level of $h = 0.2$ m is fixed at the exit. Slip conditions are applied along the solid walls. This test considers five different parameter zones presented in Table 4. Observations are taken at 108 points as shown in Fig. 7), and the coefficient of variation is fixed at $CV = 5\,\%$. The calibration parameters used in this test are recorded in Table 3.
Table 5 presents the values of the Manning coefficients obtained from the calibration. The results are remarkably satisfactory, with maximum relative errors below 10 %. Fig. 10 compares the convergence of the four different algorithms considered (GA, HGA, PSO, and HPSO) for a fixed maximum number of iterations. The graph confirms the observations made from Table 5 and reveals that the HPSO achieves the highest precision, while the GA and the HGA show a noticeable gap in precision compared to the top-performing HPSO. This gap is due to the limited number of iterations and would be much less for a larger number of iterations. Furthermore, it is important to observe that the hybrid algorithms demonstrate faster convergence with higher precision than their basic forms. These results demonstrate the efficiency and inherent speed of the proposed hybrid methodology for accurately identifying Manning's coefficients. Additionally, it is essential to emphasize the

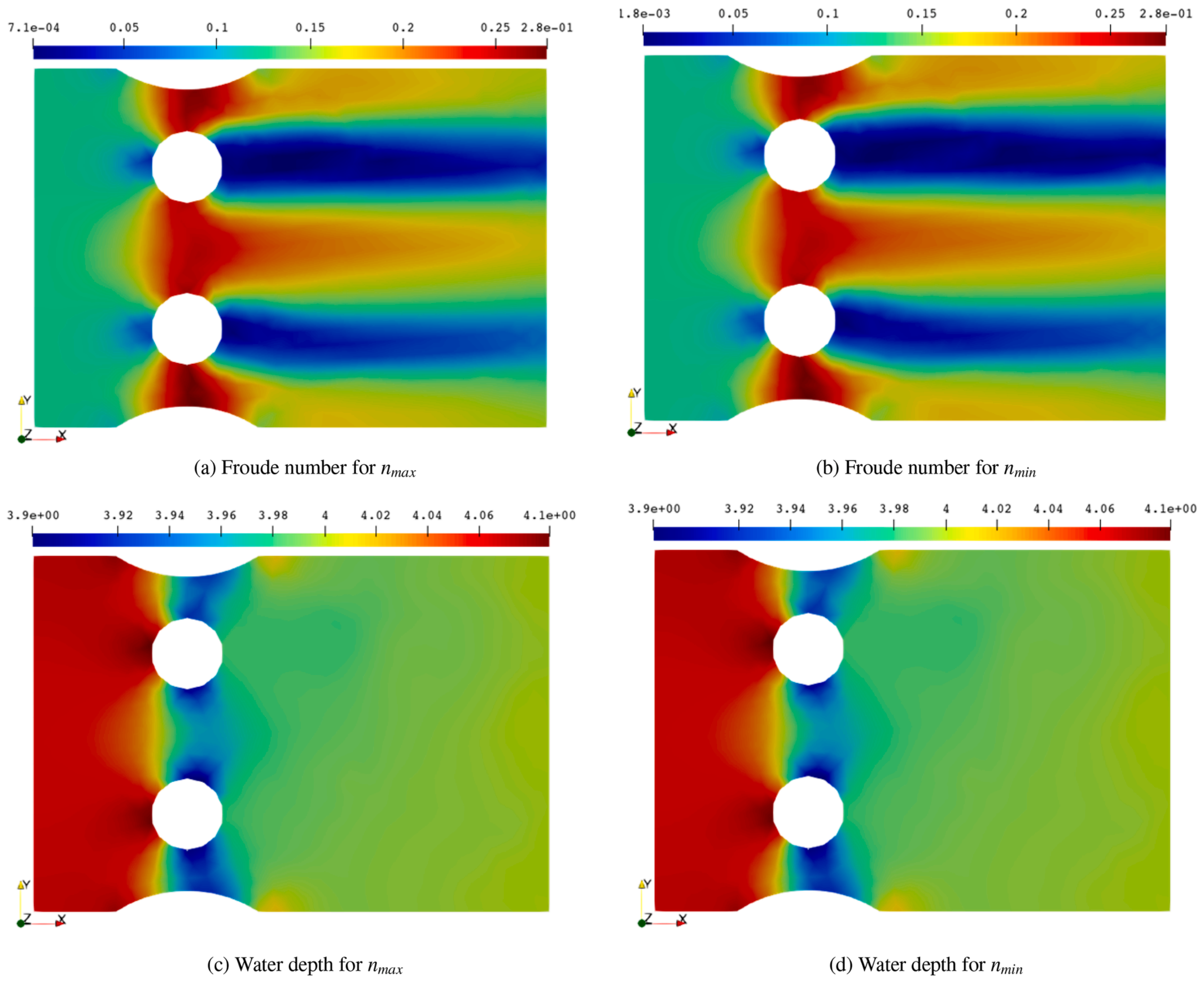


Fig. 14. Numerical results of the preliminary flow study: Froude number and water depth - river with piers ($n_{\text{min}} = 0.01$; $n_{\text{max}} = 0.05$).

importance of selecting appropriate measurement stations during calibration. Hydraulic engineers must have a thorough understanding of the watercourse under study and make informed decisions regarding field measurement points.

4.2. Sensitivity analysis to measurement points location

In Section 4.1, we emphasized the importance of carefully selecting measurement stations during the calibration process. Here, we extend this analysis by investigating the sensitivity of the calibrated results to the location measurement points. In hydrology, such testing is often part of a measurement campaign, where the position of observation lines across the riverbed is systematically varied to capture different aspects of the flow field.

To illustrate this approach, we revisit the divergent channel case study. All measurement lines remain fixed at their initial locations except for the second line, which is progressively displaced within the channel.The initial position of this second line (shown in blue in Fig. 7) is progressively shifted to four alternative positions denoted as positions 1–4 in Fig. 11. For each configuration, the model is recalibrated using the updated set of observation points, and the robustness of the calibration is assessed by comparing the resulting Manning's roughness parameters.

Table 6
Flow in a River with non-submersible piers: calibration parameters.

Number of mesh elements	1408
Number of mesh nodes	767
Number of zones	4
Sample size	256
Ensemble model size	10
Number of measurement stations	28
Coefficient of variation **CV**	5 %

Table 7
Flow in a River with non-submersible piers: Roughness coefficients of each zone.

Zones	Bathymetry	Search space	Ground truth
n_1	Fine gravel	$[2.16 - 2.64] \times 10^{-2}$	2.40×10^{-2}
n_2	Gravel	$[3.16 - 3.84] \times 10^{-2}$	3.50×10^{-2}
n_3	Sand	$[1.80 - 2.20] \times 10^{-2}$	2.00×10^{-2}
n_4	Cobble	$[2.71 - 3.29] \times 10^{-2}$	3.00×10^{-2}

Fig. 12 shows the normalized error values obtained for the calibrated Manning parameters as a function of the displaced line position. This provides direct insight into how the spatial distribution of measurement stations affects the stability of the calibration.

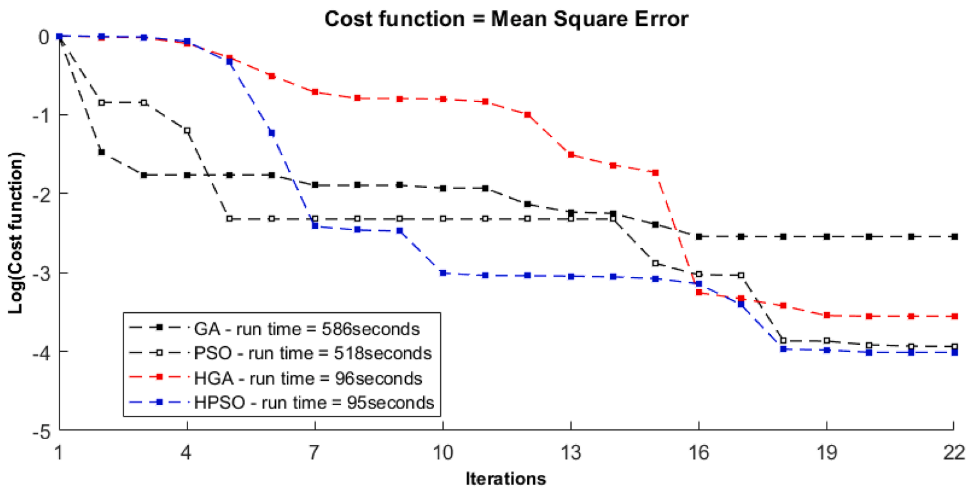


Fig. 15. Flow in a River with non-submersible piers. Convergence history for the GA, HGA, PSO and HPSO.

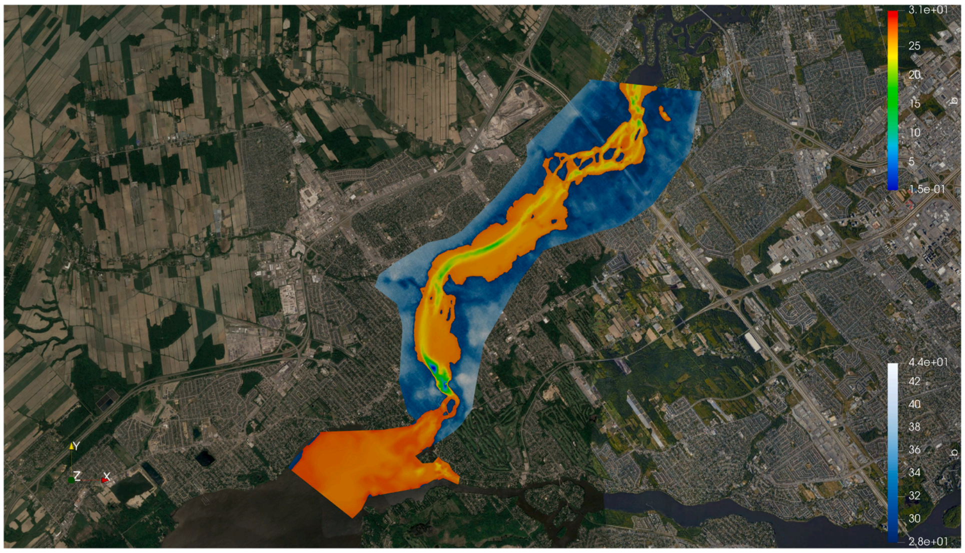


Fig. 16. The domain of the Mille_îles river overlaid on a satellite image. Adapted from [19].

The results reveal that not all sections contribute equally to a robust calibration. Positions 1 (upstream) and 4 (downstream) yield parameters closer to the ground truth, whereas intermediate positions, particularly after the expansion zone, lead to larger deviations. This suggests that upstream and downstream sections capture the global momentum distribution and boundary-layer development more effectively, while midstream sections are dominated by local flow features that provide less representative information for Manning's parameter estimation.

These findings reinforce the importance of strategically placing measurement stations to ensure robust parameter calibration in river flow modeling.

4.3. *Flow in a river with non-submersible piers*

The second benchmark test was conducted in a river domain with pier obstacles, modeled after a similar setup examined by [8]. This experimental scenario aims to assess the methodology's robustness in a complex geometry context, mimicking real-world conditions while remaining relatively simple to replicate.

The channel measures 28.5 m by 20 m, with a constant depth of 4 m and two 4-meter diameter cylindrical obstacles. Fig. 13 shows the domain divided into 4 subdomains, with 28 strategically positioned measurement points. These points were chosen both before and after the obstacles, as well as around them, following the preliminary study detailed in Fig. 14. The domain's unstructured mesh comprises 767 nodes and 1408 elements.

A flow rate of 62 m^3/s is imposed at the inlet, with the water level fixed at $h = 0.0$ m at the outlet. The flow simulation was initialized from rest and a horizontal plane, and then computed over six min. Boundary conditions remain consistent with those of the previous reference test. Table 6 summarizes the initial calibration parameters, while Table 7 presents the Manning's coefficients of the four different zones considered.

The results obtained after calibration are shown in Table 8, which displays the ground truth and calibrated distributions of Manning's coefficients. The results reveal a high level of satisfaction, with relative errors of less than 10 % overall. Fig. 15 illustrates the convergence history using the four algorithms. It can be observed, similar to the previous case, that the HPSO converges rapidly and with greater precision compared to the others, followed by PSO.

4.4. *Flow in the mille Îles river*

The last benchmark test was conducted in the domain of the Mille-Îles River, a 42-kilometer watercourse situated in the Montreal region, integral to the Saint Lawrence River hydrological system. Known for

Table 8
Flow in a River with non-submersible piers: Results from the Genetic algorithm, Hybrid genetic algorithm, Particle swarm optimization and the Hybrid particle swarm optimization.

Zones	n_1	n_2	n_3	n_4
Ground truth ($\times 10^{-2}$)	2.40	3.50	2.00	3.00
Genetic Algorithm (GA)				
Predictions ($\times 10^{-2}$)	2.254	3.518	1.956	3.012
Relative error (%)	6.08	0.51	2.20	0.40
Hybrid Genetic Algorithm (HGA)				
Predictions ($\times 10^{-2}$)	2.384	3.506	2.016	2.992
Relative error (%)	0.67	0.17	0.8	0.27
Particle Swarm Optimization (PSO)				
Predictions ($\times 10^{-2}$)	2.306	3.511	1.966	3.015
Relative error (%)	3.92	0.31	1.70	0.50
Hybrid Particle Swarm Optimization (HPSO)				
Predictions ($\times 10^{-2}$)	2.395	3.508	1.998	3.007
Relative error (%)	0.21	0.23	0.10	0.23

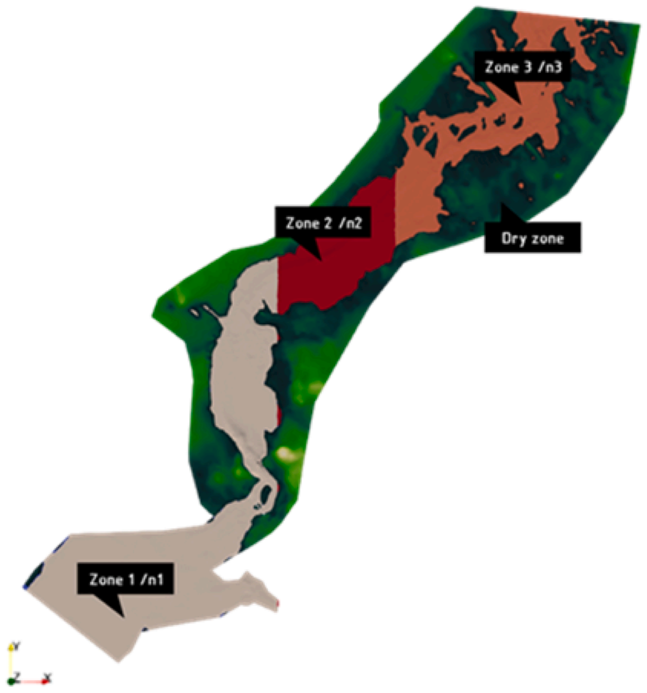
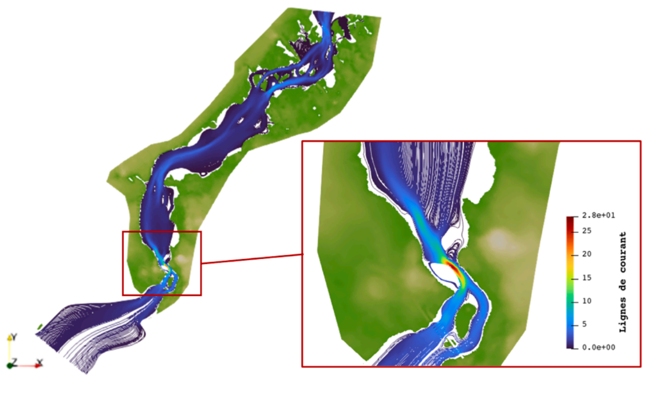
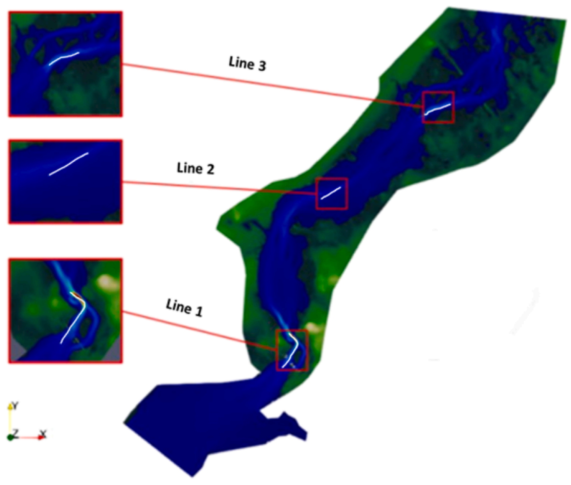


Fig. 17. Mille_îles River channel divided into 3 subdomains.



(a)



(b)

Fig. 18. (a) Numerical results of the preliminary flow study: Streamlines with momentum-based coloring - Mille_Îles River, (b) Location of Data Measurements stations on Mille_îles River.

Table 9
Flow in the Mille Îles River: calibration parameters.

Number of mesh elements	481930
Number of mesh nodes	243,161
Number of zones	3
Sample size	128
Ensemble model size	10
Number of measurement stations	420
Coefficient of variation *CV*	5 % & 10 %

Table 10
Flow in the Mille Îles River: Roughness coefficients and search space in each zone.

Zones	Bathymetry	Search space (CV = 10 %)	Search space (CV = 5 %)	Ground truth
n_1	Sand	$[1.77 - 2.63] \times 10^{-2}$	$[1.98 - 2.42] \times 10^{-2}$	2.20×10^{-2}
n_2	Gravel	$[2.25 - 3.35] \times 10^{-2}$	$[2.53 - 3.07] \times 10^{-2}$	2.80×10^{-2}
n_3	Rock cuts	$[2.09 - 3.11] \times 10^{-2}$	$[1.26 - 1.54] \times 10^{-2}$	2.60×10^{-2}

its scenic islands, rich biodiversity, and recreational opportunities, this natural environment serves as the real-world setting to assess the robustness of the proposed calibration methodology. Fig. 16 provides an overview of the simulation domain overlaid on a satellite image from Google Earth. The domain is oriented with the y-axis pointing northward, featuring an inlet at the bottom left and an outlet at the top right. For this test, utilizing real bathymetry data, the area is subdivided into four subdomains, as depicted in Fig. 17. The focus is on the riverbed flow, characterized by depths greater than 0.1 m, with each subdomain assigned a specific Manning's coefficient: zones 1, 2, and 3.

A preliminary study, shown in Fig. 18a, identified key zones suitable for placing measurement points. Flow parameters, particularly water depths, were gathered and compared at 420 points distributed along three strategically positioned lines within the channel. Fig. 18b depicts the locations of these observation lines on the numerical model.

For this test, two scenarios were considered: firstly, with a coefficient of variation set at 5 %, consistent with previous cases; and secondly, with a coefficient of variation set at 10 %. Given the dense mesh (over 400,000 elements), the number of samples was fixed at 128. Tables 9 and 10 show the conditions and parameters used for the calibration.

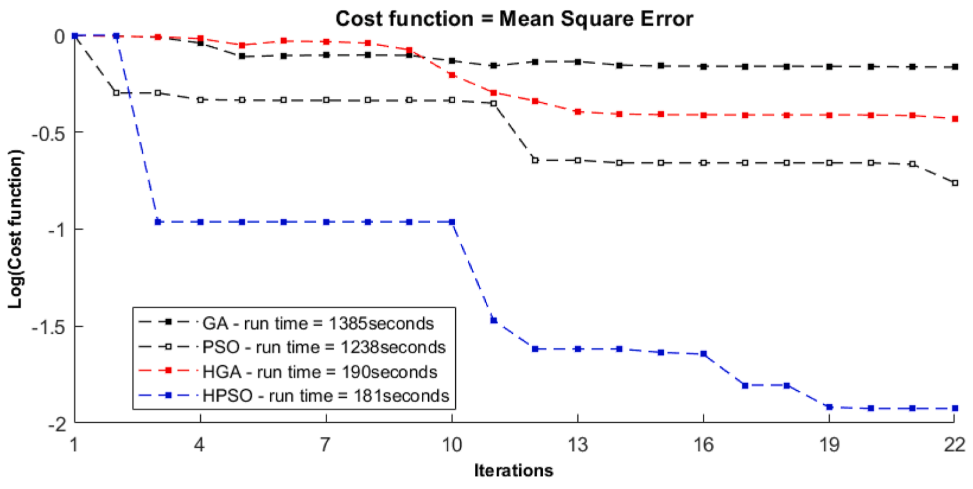


Fig. 19. Flow in the Mille Îles River. Convergence history for the GA, HGA, PSO and HPSO with $CV = 10\,\%$.

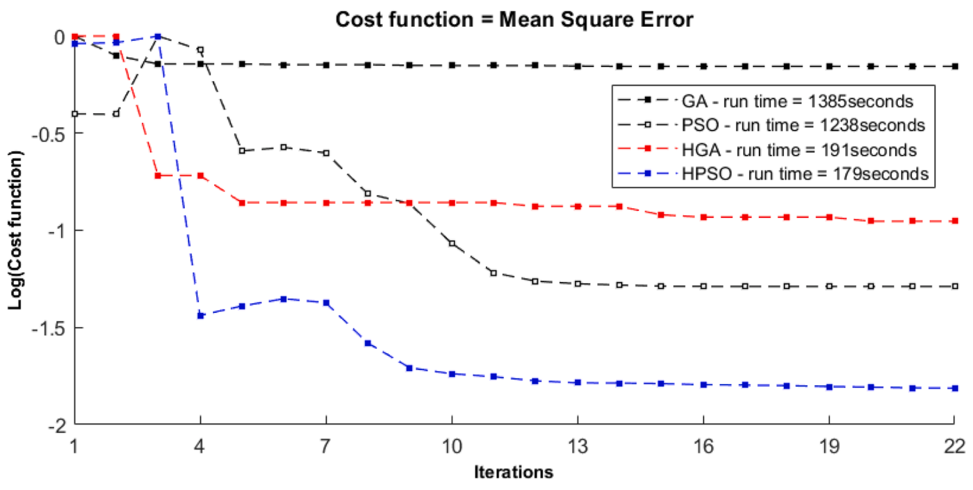


Fig. 20. Flow in the Mille Îles River. Convergence history for the GA, HGA, PSO and HPSO with $CV = 5\,\%$.

Table 11
Flow in the Mille Iles River: Results from the Genetic algorithm, the Hybrid genetic algorithm, Particle swarm optimization, and the Hybrid particle swarm optimization.

	CV = 10 %			CV = 5 %		
Zones	n_1	n_2	n_3	n_1	n_2	n_3
Ground truth ($\times 10^{-2}$)	2.20	2.80	2.60	2.20	2.80	2.60
Genetic Algorithm (GA)						
Predictions ($\times 10^{-2}$)	2.171	3.057	2.605	2.177	2.923	2.569
Relative error (%)	1.32	9.18	0.19	1.04	4.39	1.19
Hybrid Genetic Algorithm (HGA)						
Predictions ($\times 10^{-2}$)	2.204	2.749	2.591	2.177	2.796	2.589
Relative error (%)	0.18	1.82	0.35	1.04	0.14	0.38
Particle Swarm Optimization (PSO)						
Predictions ($\times 10^{-2}$)	2.215	2.498	2.571	2.159	2.826	2.586
Relative error (%)	0.68	3.92	1.12	1.86	0.92	0.54
Hybrid Particle Swarm Optimization (HPSO)						
Predictions ($\times 10^{-2}$)	2.203	2.762	2.591	2.183	2.790	2.593
Relative error (%)	0.16	1.36	0.35	0.77	0.36	0.27

The results are illustrated in Table 11. Again, the results were highly satisfactory in both scenarios. The convergence of the Manning's coefficient was obtained after a few iterations and with great precision for the HPSO (see Figs. 19 and 20). For large search spaces, PSO and the GA have larger relative errors, but the hybrid forms of both algorithms offer good performance.

5. Conclusion

In this paper, we introduced a novel methodology for calibrating Manning's friction coefficients in shallow water flows by combining optimization algorithms with an ensemble model of deep neural networks. The proposed approach involves minimizing a cost function that quantifies the discrepancies between the numerical model solution and the given data. By leveraging deep neural networks and ensemble modeling, we constructed a surrogate model to replace the high-fidelity solver, thereby dramatically reducing the computational cost and convergence issues associated with optimization when using high-fidelity solvers directly. Four different optimization algorithms were then employed to calibrate the friction coefficients distributed over the riverbed.

Numerical tests on hypothetical and real bathymetry setups demonstrated that the developed methodology is stable and converges with high precision within a short time interval. This attests to the robustness of the methodology and its applicability to real flow scenarios, particularly in contexts with limited time or data. The proposed techniques produced calibrated Manning values with relatively small deviations from the ground truth values. Despite general satisfaction with the performance of all the proposed algorithms, HPSO stood out due to its superior precision and efficiency. Additionally, a performance test of the in-house multi-CPU/multi-GPU shallow-water equations high-fidelity solver, CuteFlow, was conducted to evaluate its capabilities. A detailed description of this test is provided in the Appendix.

The proposed methodology leverages advanced computational methodologies, including a multi-GPU solver to build a large dataset of high-fidelity solutions, a surrogate model based on neural networks for fast inference, and hybrid algorithms for robust and accurate optimization. This combination of techniques not only enhances the efficiency and accuracy of the calibration process but also demonstrates the potential for solving a wide range of inverse problems. Through extensive numerical experiments and a real-world application, we demonstrated the effectiveness of the proposed methodology.

Overall, this methodology represents a significant advancement in the field of hydraulic modeling and optimization. Its ability to integrate state-of-the-art computational techniques ensures superior performance and robustness. The versatility of this approach makes it applicable to various domains, offering a powerful tool for researchers and practitioners facing complex inverse problems. Future work will further broaden the scope of applications by reformulating the inverse problem within a probabilistic framework, thereby explicitly incorporating field uncertainties. We believe this direction will provide the additional level of realism required for many practical scenarios, reinforcing the methodology's potential across diverse domains.

CRediT authorship contribution statement

Igor Gildas Metcheka Kengne: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation; **Vincent Delmas:** Writing – original draft, Visualization, Software, Investigation, Formal analysis, Conceptualization; **Azzeddine Soulaïmani:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by the Natural Sciences and Engineering Research Council of Canada (Grant number RGPIN/2693-2021). The computing resources were provided by the Digital Research Alliance of Canada. The river bathymetry data was provided by Pierre Dupuis of the Communaute Metropolitaine de Montreal.

Appendix A. Evaluation of the parallel high calculation code CuteFlow

The aim of this section is to evaluate the CUTEFLOW code and demonstrate its functionality through a practical example. Extensive performance tests have been conducted in previous studies [19] and [21]. In this study, we use a numerical test to simulate a complex dam break scenario. This type of test, commonly employed in laboratory tsunami modeling, provides a concrete illustration of the code's operational capabilities.

Specifically, we replicate an experimental complex dam break flow case over a dry bed using CUTEFLOW. This experiment was originally conducted at the Hydraulics Laboratory of the University of Parma by [44] and was later studied by [45].

A.1. Experimental setup

Fig. A.1 shows the top view configuration of the experimental system. The experimental setup consists of a rectangular container with a constant bathymetry, measuring 260 cm in length and 120 cm in width. This container is divided into two sections: one serving as a reservoir and the other as a dry area representing the flood zone. A barrier acting as a dam is positioned in the middle of the dividing wall.

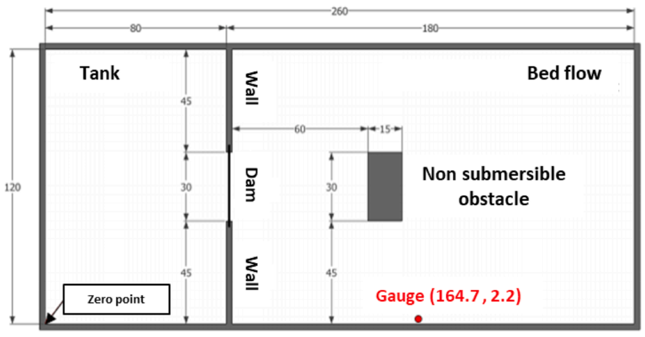


Fig. A.1. Dimensions (in cm) of the experimental complex dam-break experimental system. Taken from [45].

The water level in the reservoir is 0.13 m. Removing the barrier will cause water to flow into the flood zone, where a non-submersible obstacle creates disturbances in the flow field.

A.2. Initial conditions

This type of flow scenario is treated by CuteFlow as a Riemann one-dimensional problem. Initial values selected for this test are as follows:

$$\begin{cases} \text{h} = \begin{cases} 0.13 & if \quad x > 0.8m \\ 0.00 & if \quad x < 0.8m \end{cases} \\ \bar{u} = 0 \quad [m/s] \\ \bar{v} = 0 \quad [m/s] \end{cases}$$

The simulation domain was discretized using unstructured triangular elements generated by the GMSH software (https://gmsh.info/) and subsequently imported into the computational code. For this study, the Manning roughness coefficient was fixed at 0.007 $s.m^{-1/3}$, as per the calibration by [44]. The Courant number CFL, ensuring numerical stability, was set to 0.2. To evaluate the computational efficiency of the CuteFlow code, various tests were conducted by varying the mesh element sizes from $5mm$ to $0.5mm$. Table A.1 presents the mesh sizes used in these tests, while A.2 illustrates an example of the coarse and unstructured mesh of the domain, comprising 3456 elements.

Table A.1
Caption.

	Test 1	Test 2	Test 3	Test 4
Mesh type	Coarse	Medium	Fine	benchmark solution
Element size (mm)	5	2.5	1.25	0.5
Number of mesh elements	287,195	1,137,841	4,530,572	15690153

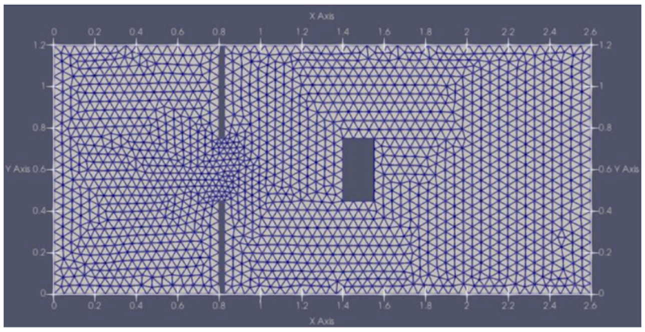


Fig. A.2. Coarse mesh of 3456 elements for the case of a complex dam break.

Table A.2
Complex dam break test run time (Ts = 10s).

	Test 1	Test 2	Test 3	Test 4
Mesh type	Coarse	Medium	Fine	benchmark solution
Element size (mm)	5	2.5	1.25	0.5
Number of mesh elements	287,195	1,137,841	4,530,572	15690153
Recorded Run Time				
Computational resources	4GPUs - 8GB			16GPUs - 32GB
1st order - HLLC	24s	3min30s	9min26s	/
2nd order - MUSCL	30s	3min37s	16min14s	15h27min15s

A.3. Results

We presented the solutions obtained at different time steps over the different mesh types, using two of the numerical schemes integrated into CuteFlow: the 1storder HLLC scheme and the 2nd order MUSCL scheme. Solutions were generated considering a Riemann problem at the dam.

A.3.1. Comparison to experimental data

We assess CuteFlow's capability to simulate a real-world complex flow scenario. Fig. A.3 illustrates the comparison between experimental and numerical results of water height for two numerical schemes. It presents multiple snapshots of the water height in plan view. At the initial time ($t = 0.4s$), the initial water flow is observed following the barrier opening. Discrepancies between experimental and numerical results may stem from the three-dimensional effect of water falling at the opening during the initial phase, a feature not accurately captured by CuteFlow's two-dimensional Saint-Venant equation solver.

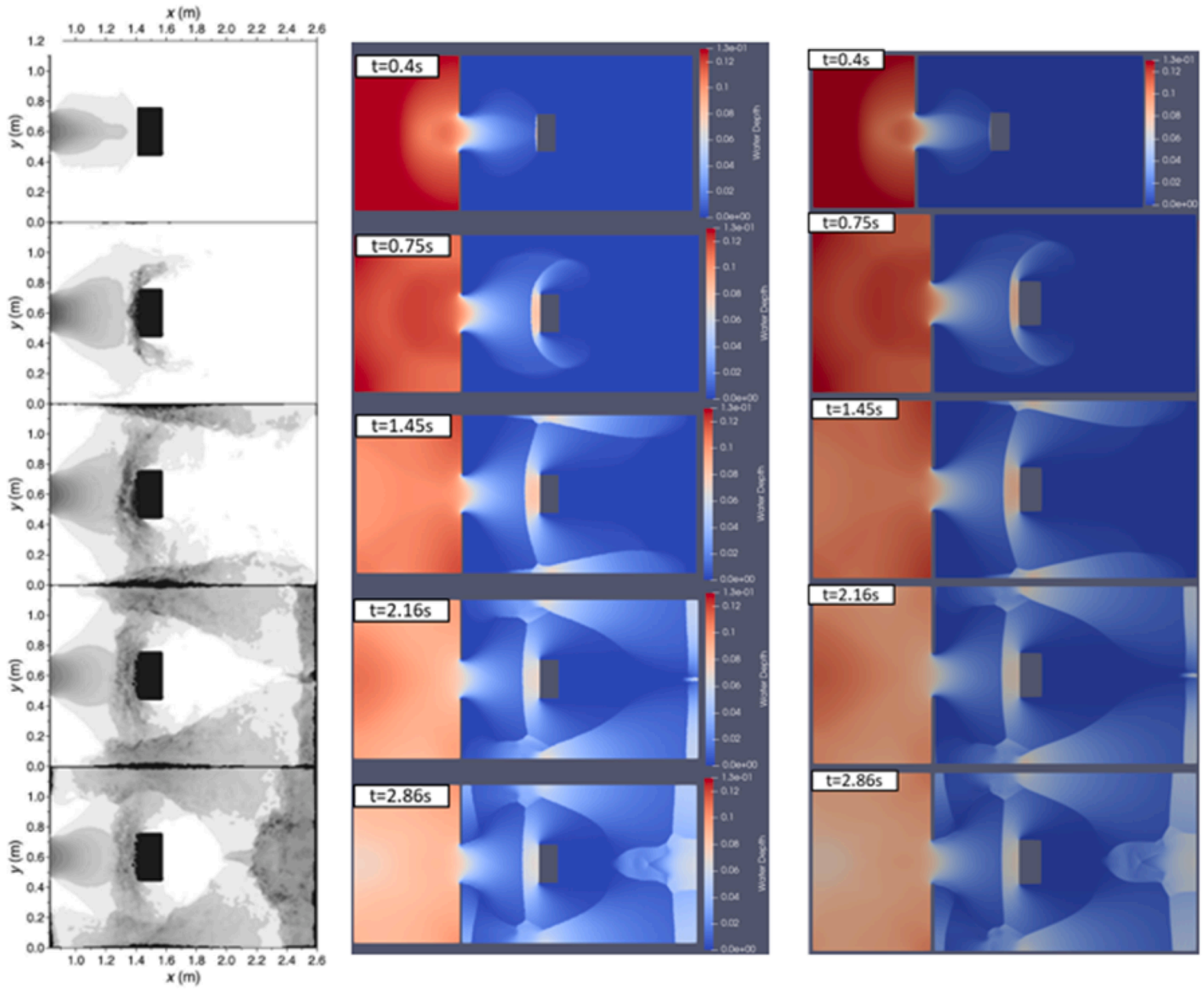
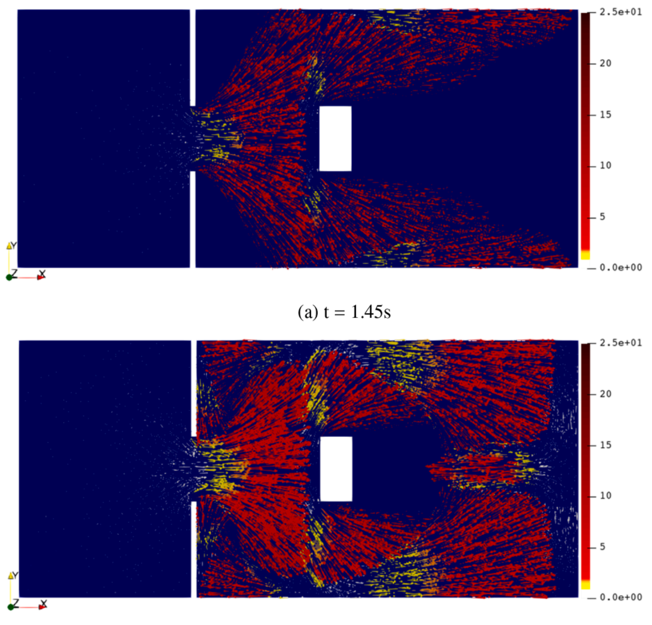


Fig. A.3. Plan snapshots, from left to right, of experimental [44] and numerical (1st- and 2nd-order) water levels at different times ($t = 0.4s$; $t = 0.75s$; $t = 1.45s$; $t = 2.16s$ and $t = 2.86s$) on a 287,000 elements - mesh.



(a) t = 1.45s

(b) t = 2.86s

Fig. A.4. Velocity field plot of numerical results with coloring based on Froude number, at instants ($t = 1.45s$ and $t = 2.86s$).

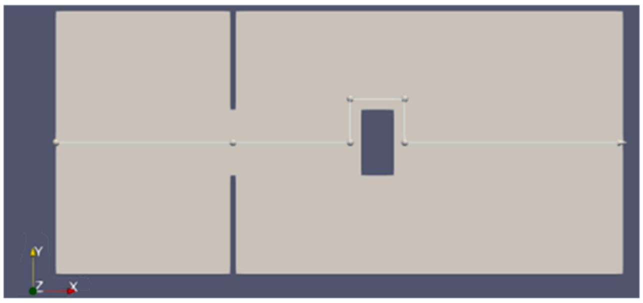
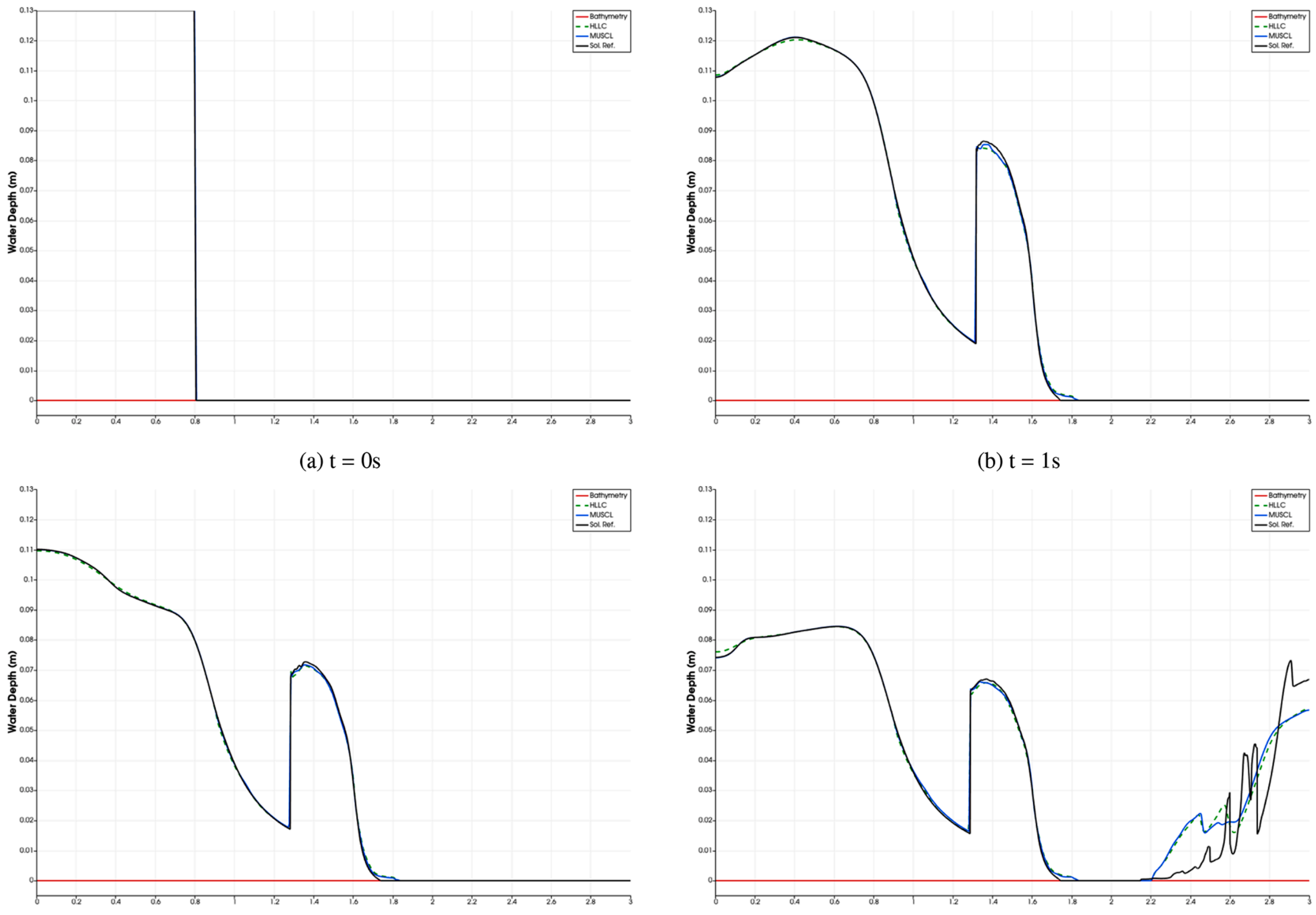


Fig. A.5. Solution visualization polyline - case of a complex dam break.

Furthermore, contrary to real-world conditions, the model assumes a spontaneous gate break.

By $t = 0,75s$, a shock wave emerges due to the collision between water and a non-submersible obstacle. While various numerical models of CuteFlow successfully reproduce the hydraulic jump observed in experimental snapshots, indicating the transition from supercritical ($F_r > 1$) to subcritical flow ($F_r < 1$), the chaotic effects observed in laboratory experiments appear subdued in the numerical model. At subsequent times ($t = 1.45s$; $t = 2.16s$ and $t = 2.86s$), the numerical model accurately replicates upstream propagation of the hydraulic jump and the emergence of new shock waves resulting from flow interactions with side walls.



(a) t = 0s

(b) t = 1s

(c) t = 2s

(d) t=3s

Fig. A.6. Water level evolution along the viewing line.

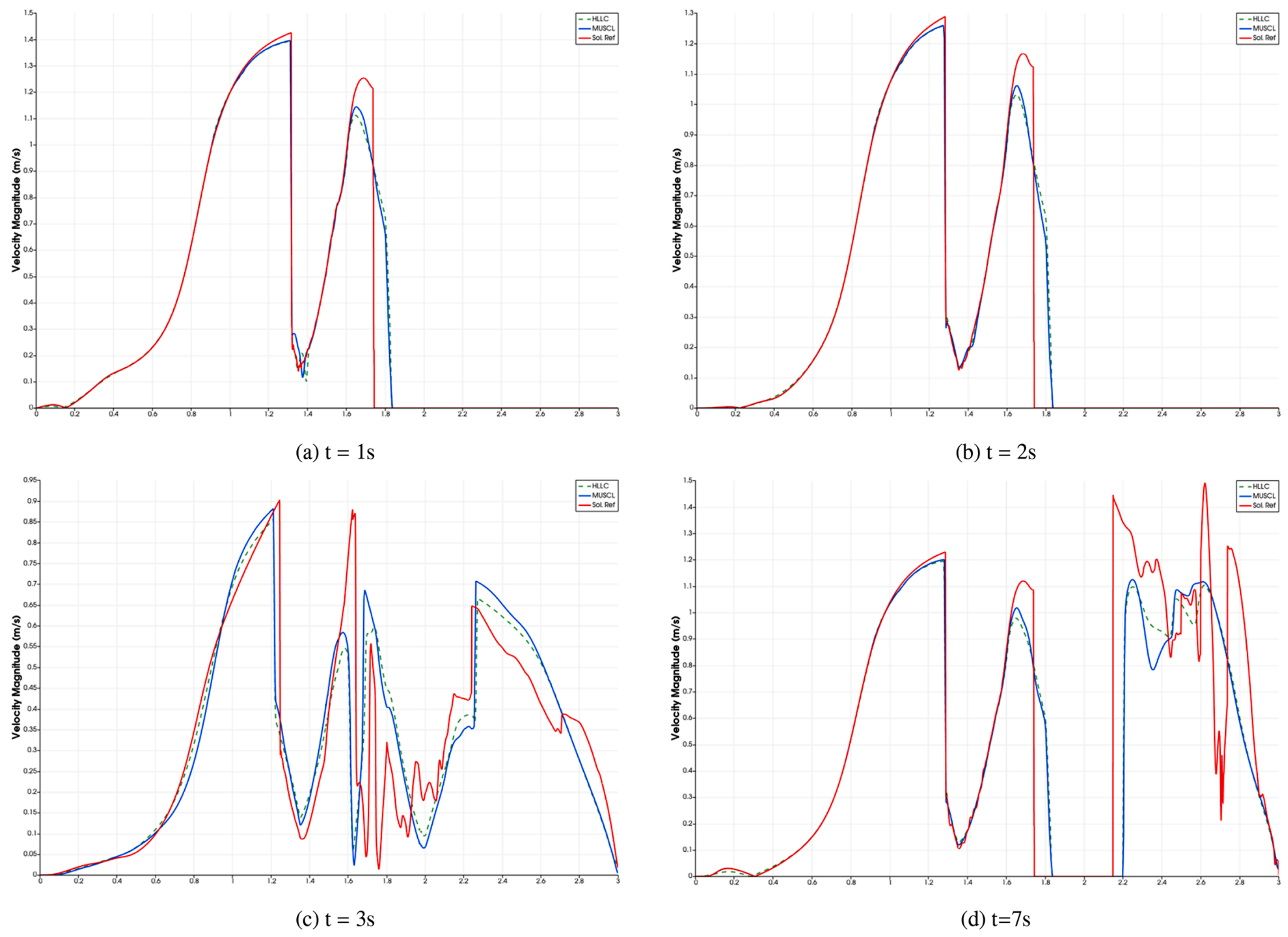


Fig. A.7. evolution of $\bar{u}$ velocity component along the viewing line.

Fig. A.4 depicts a velocity field plot with color-coded Froude numbers ranging from 0 (white) to 25 (dark red), delineated by $F_r = 1$ (yellow). These visualizations identify flow transitions, from supercritical to subcritical, characterized by hydraulic jump development.

CuteFlow's different numerical schemes effectively simulate the complex dam break phenomenon, with exceptions noted in areas dominated by three-dimensional effects.

A.3.2. *Comparison between the numerical and reference solutions*

The reference solution, used as a benchmark, is computed on an adapted mesh consisting of 1,569,0153 elements using a second-order numerical scheme. Here, we compare the water level and velocity results between the reference solution and the first- and second-order solutions on a medium mesh. The solutions are plotted along the line defined in Fig. A.5.

Fig. A.8 compares the momentum magnitudes obtained using the first-order HLLC and second-order MUSCL methods on a 15-million-element mesh serving as the reference. This illustration provides insight into the accuracy and robustness of the different methods.

The obtained results are illustrated in Figs. A.6 and A.7.

The primary distinction between the two schemes is observed in Zone A, near the dam break, where a hydraulic jump occurs. The propagation of the shock wave is more pronounced and clear with the second-order MUSCL method. Additionally, fine vortex structures in the water flow movement within Zone A are also noticeable with this second-order scheme. In Zone B, the appearance of a reddish area, absent in the first-order scheme, demonstrates the increased accuracy of the second-order method. Moreover, this method better reproduces flow regime transition zones and the propagation of shock waves following various impacts with the walls.

In its pursuit of closer alignment with experimental results, Cuteflow, leveraging its multi-GPU version, also offers the ability to further refine the mesh for significantly enhanced precision within a reasonable time frame, even with modest computational resources. However, it should be noted that calculation time largely depends on the level of mesh refinement. Table A.2 summarizes the computational resources used for these simulations, along with the recorded execution times.

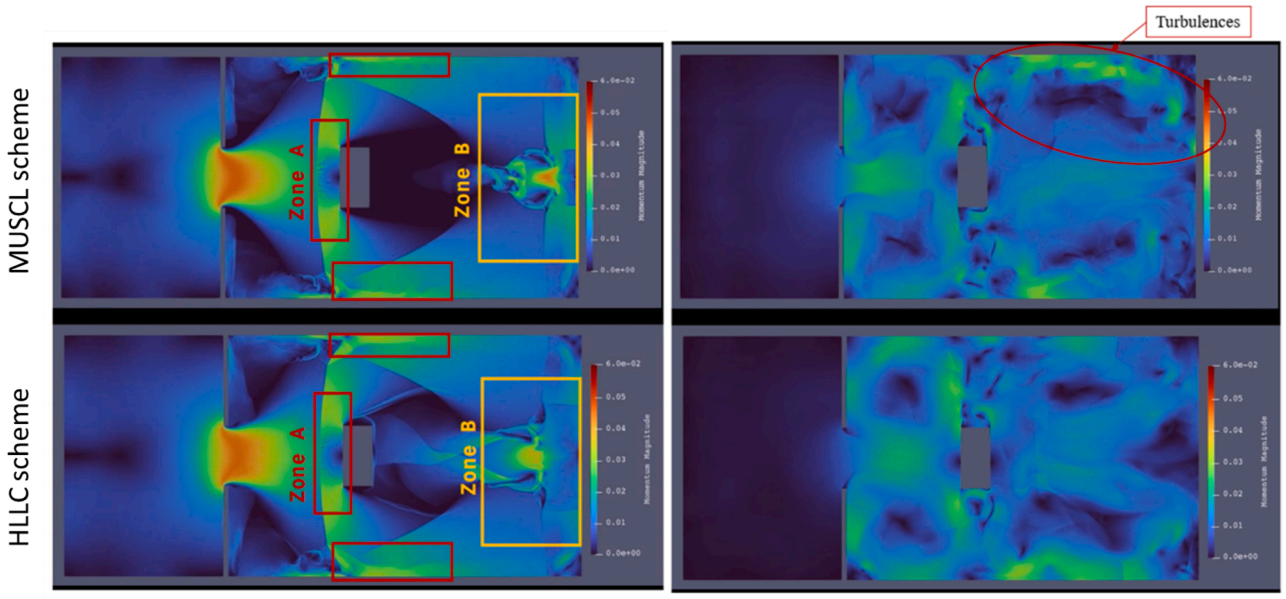


Fig. A.8. Comparison of the first-order HLLC and second-order MUSCL methods on a 15-million-element mesh for a complex dam break scenario, computed using 16 GPUs.

References

[1] Ramesh R, Datta B, Bhallamudi M, Narayana A. Optimal estimation of roughness in open-channel flows. J Hydraul Eng-asce - J HYDRAUL ENG-ASCE 2000;126. https://doi.org/10.1061/(ASCE)0733-9429(2000)126:4(299).

[2] Yao L, Peng Y, Yu X, Zhang Z, Luo S. Optimal inversion of Manning's roughness in unsteady open flow simulations using adaptive parallel genetic algorithm. Water Resour Manage 2022;37. https://doi.org/10.1007/s11269-022-03411-x.

[3] Vidal J-P, Moisan S, Faure J-B, Dartus D. River model calibration, from guidelines to operational support tools. Environmental Modelling and Software 2007;22:1628–40. https://doi.org/10.1016/j.envsoft.2006.12.003.

[4] Limerinos JT. Determination of the manning coefficient for measured bed roughness in natural channels. United States government printing office; 1970.

[5] Hameed Alwaeli L, Ali S. Estimating of Manning's roughness coefficient for hilla river through calibration using HEC-RAS model 2013;7:44–53.

[6] Boulomytis V TG, Zuffo AC, Dalfré Filho JG, Imteaz MA. Estimation and calibration of Manning's roughness coefficients for ungauged watersheds on coastal floodplains. Int J River Basin Manage 2017;15(2):199–206. https://doi.org/10.1080/15715124.2017.1298605.

[7] Li Y, Geng Y, Mao L. Calibration method for Manning's roughness coefficient for a river flume model. Water Supply 2020;20. https://doi.org/10.2166/ws.2020.235.

[8] Soulaimani A, Idrissi M. Identification of the friction coefficient in shallow-water flows using optimal control theory. Int J Comut Fluid Dyn 1999;12:29–48. https://doi.org/10.1080/10618569908940814.

[9] Becker L, Yeh W WG. Identification of parameters in unsteady open channel flows. Water Resour Res 1972;8(4):956–65. https://doi.org/10.1029/WR008i004p00956.

[10] Lal W. Calibration of riverbed roughness. J Hydraul Eng -asce - J HYDRAUL ENG-ASCE 1995;121. https://doi.org/10.1061/(ASCE)0733-9429(1995)121:9(664).

[11] Nguyen TH, Fenton DJ. Identification of roughness in open channels. In: Proc. of 6th international conference on hydro-science and engineering, Brisbane, Australia. 2004,.

[12] Ding Y, Wang S. Identification of Manning's roughness coefficients in channel network using adjoint analysis. Int J Comut Fluid Dyn 2005;19:3–13. https://doi.org/10.1080/10618560410001710496.

[13] Yu Z, Tian Y, Zheng Y, Zhao X. Calibration of pipe roughness coefficient based on manning formula and genetic algorithm. Trans Tianjin Univ 2009;15:452–6. https://doi.org/10.1007/s12209-009-0078-2.

[14] Tang H-w, Xin X-k, Dai W-h, Xiao Y. Parameter identification for modeling river network using a genetic algorithm. J Hydrodyn 2010;22(2):246–53. https://doi.org/10.1016/S1001-6058(09)60051-2.

[15] Yang T-H, Wang Y-c, Tsung S-C, Guo W. Applying micro-genetic algorithm in the one-dimensional unsteady hydraulic model for parameter optimization. J Hydroinf 2014;16:772–83. https://api.semanticscholar.org/CorpusID:62164120.

[16] Reshma T, Reddy KV, Pratap D, Agilan V. Single- and two- step optimization of infiltration parameters and manning's roughness coefficients for a watershed using a multi-objective genetic algorithm. ISH J Hydraul Eng 2017;24(1):53–60.

[17] Jayakumar D, Kumar D. Automated calibration of a two-dimensional overland flow model by estimating manning's roughness coefficient using genetic algorithm. J Hydroinf 2017;20:jh2017110. https://doi.org/10.2166/hydro.2017.110.

[18] Agresta A, Baioletti M, Biscarini C, Milani A, Santucci V. Evolutionary algorithms for roughness coefficient estimation in river flow analyses. In: Applications of evolutionary computation. Springer International Publishing; 2021, p. 795–811.

[19] Delmas V. Implémentation multi-GPU d'un code en volumes finis pour le calcul de haute per formance des écoulements à sur face libre. Master's thesis; École de Technologie Supérieure, Montréal, QC; 2020.

[20] Delmas V, Soulaïmani A. Parallel high-order resolution of the shallow-water equations on real large-scale meshes with complex bathymetries. J Comput Phys 2022;471:111629. https://doi.org/10.1016/j.jcp.2022.111629.

[21] Delmas V, Soulaïmani A. Multi-GPU implementation of a time-explicit finite volume solver using CUDA and a CUDA-aware version of openMPI with application to shallow water flows. Comput Phys Commun 2022;271:108190. https://doi.org/10.1016/j.cpc.2021.108190.

[22] Loukili Y, Soulaimani A. Numerical tracking of shallow water waves by theunstructured finite volume WAF approximation. Int J Comput Methods Eng Mech 2007;8:1–14.

[23] Zokagoa J-M. Modélisation numérique des écoulements à surface libre avec bancs couvrants-découvrants par les volumes finis et la décomposition orthogonale aux valeurs propres. Ph.D. thesis; École de technologie supérieure, Montréal, QC; 2011.

[24] Toro E. Riemann solvers and numerical methods for fluid dynamics: a practical introduction. 2009, https://doi.org/10.1007/b79761.

[25] Ata R, Pavan S, Khelladi S, Toro EF. A weighted average flux (WAF) scheme applied to shallow water equations for real-life applications. Adv Water Resour 2013;62:155–72. http://www.sciencedirect.com/science/article/pii/S0309170813001802.

[26] Ata R, Soulaimani A. A stabilized SPH method for inviscid shallow water flows. Int J Numer Methods Fluids 2005;47:139–59. https://doi.org/10.1002/fld.801.

[27] Audusse E, Bouchut F, Bristeau M-O, Klein R, Perthame B. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM J Sci Comput 2004;25(6):2050–65. https://doi.org/10.1137/S1064827503431090.

[28] Martínez-Aranda S, Fernández-Pato J, Echeverribar I, Navas-Montilla A, Morales-Hernández M, Brufau P, et al. Finite volume models and efficient simulation tools (EST) for shallow flows. Singapore: Springer Nature Singapore. ISBN 978-981-19-1438-6; 2022, p. 67–137. https://doi.org/10.1007/978-981-19-1438-6_3

[29] Ziggaf M, Boubekeur M, kissami I, Benkhaldoun F, Mahi IE. The FVC scheme on unstructured meshes for the two-dimensional shallow water equations. In: Klöfkorn R, Keilegavlen E, Radu FA, Fuhrmann J, editors. Finite volumes for complex applications IX - Methods, theoretical aspects, examples. Cham: Springer International Publishing. ISBN 978-3-030-43651-3; 2020, p. 455–65.

[30] Benkhaldoun F, Elmahi I, Seaïd M. A new finite volume method for flux-gradient and source-term balancing in shallow water equations. Comput Methods Appl Mech Eng 2010;199(49):3324–35. https://doi.org/10.1016/j.cma.2010.07.003.

[31] Noelle S, Xing Y, Shu C-W. High-order well-balanced finite volume WENO schemes for shallow water equation with moving water. J Comput Phys 2007;226:29–58. https://doi.org/10.1016/j.jcp.2007.03.031.

[32] Suthar A, Soulaimani A. Parallelization of shallow water equations solver cuteflow. Tech. Rep.; École de technologie supérieure, Montréal, QC; 2018.

[33] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 1998;20(1):359–92. https://doi.org/10.1137/S1064827595287997.

[34] Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, et al. Open MPI: goals, concept, and design of a next generation MPI implementation. In: Proceedings, 11th European PVM/MPI Users' group meeting. Budapest, Hungary; 2004, p. 97–104.

[35] Poirier D, Allmaras SR, McCarthy DR, Smith MF, Enomoto FY. The CGNS system. AIAA Paper 1998; 39:98–3007.

[36] Jacquier P, Abdedou A, Delmas V, Soulaïmani A. Non-intrusive reduced-order modeling using uncertainty-aware deep neural networks and proper orthogonal decomposition: application to flood modeling. J Comput Phys 2021;424:109854. https://doi.org/10.1016/j.jcp.2020.109854.

[37] Joe S, Kuo FY. Constructing sobol sequences with better two-dimensional projections. SIAM J Sci Comput 2008;30(5):2635–54. https://doi.org/10.1137/070709359.

[38] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016. http://www.deeplearningbook.org.

[39] Hagan MT, Demuth HB, Beale MH, De Jesús O. Neural network design. Martin Hagan; 2014.

[40] Ganaie MA, Hu M, Malik AK, Tanveer M, Suganthan PN. Ensemble deep learning: a review. Eng Appl Artif Intell 2022;115:105151. https://doi.org/10.1016/j.engappai.2022.105151.

[41] Shahzadi G, Soulaïmani A. Deep neural network and polynomial chaos expansion-based surrogate models for sensitivity and uncertainty propagation: an application to a rockfill dam. Water 2021;13(13). https://doi.org/10.3390/w13131830.

[42] Kumar A. Optimization algorithms and applications. Chapman and Hall/CRC, Boca Raton, U.S.A; 2015.

[43] Xu YG, Li GR, Wu ZP. A novel hybrid genetic algorithm using local optimizer based on heuristic pattern move. Appl Artif Intell 2001;15(6):601–31. https://doi.org/10.1080/088395101750363966.

[44] Aureli F, Dazzi S, Maranzoni A, Mignosa P, Vacondio R. Experimental and numerical evaluation of the force due to the impact of a dam-break wave on a structure. Adv Water Resour 2015;76:29–42. https://doi.org/10.1016/j.advwatres.2014.11.009.

[45] Aslami MH, Rogers B, Stansby PK, Bottacin-Busolin A. Complex dam break simulation using the 2-D depth-averaged SPH flow model: a validation for tsunami application. IOP Conf Series: Earth and Environ Sci 2023;1169:012026. https://doi.org/10.1088/1755-1315/1169/1/012026.