Research Article

Eleonora Amato, Vito Zago*, and Ciro Del Negro

A physically consistent AI-based SPH emulator for computational fluid dynamics

https://doi.org/10.1515/nleng-2022-0359
received September 11, 2023; accepted December 1, 2023

Abstract: The integration of artificial intelligence (AI) into computational fluid dynamics (CFD) has significantly expanded the scope of fluid modeling, allowing enhanced analysis capabilities and improved simulation performance. While Eulerian methods already benefit extensively from AI, notably in reliable weather prediction, the application of AI to Lagrangian methods remains less consolidated. Smoothed particle hydrodynamics (SPH) is a Lagrangian mesh-less numerical method for CFD with well-established advantages for the simulation of highly dynamic free-surface flows. Here, we explore an application of AI to SPH simulations, utilizing an artificial neural network (ANN) to estimate hydrodynamic forces between particle pairs, learning from SPH-simulated results. A model of this nature, which emulates the mathematical representation of physics, is termed an *emulator*. We examine the physical significance of the emulator, presenting its applications in benchmark tests, assessing its faithfulness to traditional SPH simulations, and highlighting its ability to generalize and simulate test cases with varying levels of complexity beyond its training data.

Keywords: emulator, SPH, CFD, machine learning, ANN

1 Introduction

There is a growing interest in leveraging artificial intelligence (AI) to push forward the frontiers of computational fluid dynamics (CFD) [1], enabling new forms of analysis [2–4] and enhancing simulation performances [5,6]. A common approach in this direction is the use of emulators, where AI models are trained over simulated data to learn the behavior of traditional CFD models and replace them, offering better performances [7,8]. The use of AI-based emulators opens to new levels of analysis [9,10], allowing a better description of physical phenomena [11,12] and an efficient simulation of large-scale problems [8,13,14]. However, the increasing widespread use of AI-based models raises the issue of rigorously validating the outputs, necessitating a deeper analysis of their reliability. Eulerian formulations can already provide high-fidelity and reliable results, with clear examples being the Deep Emulator Network SEarch (DENSE) model for weather predictions [8] or MeshGraphNet for fluid dynamics simulations [15]. On the other hand, Lagrangian applications, especially in their mesh-free variants, are less consolidated, requiring a significant part of the design effort already to handle the absence of pre-established connections between nodes [16]. This aspect is also reflected in the generated results. While some of these approaches are based on simplified CFD models with little physical meaning, many others only show qualitative comparisons to the reference CFD models, with quantitative analysis typically being out of their scope. It is, therefore, important to explore the real-world relevance and accuracy of mesh-free Lagrangian emulators compared to traditional equation-based methods. We will focus on smoothed particle hydrodynamics (SPH) [17], a mesh-free numerical method that allows to obtain high-fidelity simulations of complex fluids [18], specifically for free surface and multi-phase flows subject to large deformations and interacting with complex geometries [19].

Among the major groups of approaches that can be found in the literature for Lagrangian emulators, one is based on the use of graph neural networks (GNNs), where the graph is a direct way to compensate for the lack of a spatial structure in the dataset. GNNs are built so that particles are treated as the nodes and the interactions between them are represented as the edges. Specifically for the context of fluid dynamics, Lagrangian fluid graph networks [20] have been developed [21]. Despite this approach can effectively cope with unstructured datasets, it affects the mesh-free nature of the reference SPH method. It introduces

*** Corresponding author: Vito Zago,** Osservatorio Etneo, Istituto Nazionale di Geofisica e Vulcanologia, Catania, Italy,
e-mail: vito.zago@ingv.it
Eleonora Amato: Osservatorio Etneo, Istituto Nazionale di Geofisica e Vulcanologia, Catania, Italy; Dipartimento di Matematica e Informatica, Università di Palermo, Palermo, Italy
Ciro Del Negro: Osservatorio Etneo, Istituto Nazionale di Geofisica e Vulcanologia, Catania, Italy

the need for re-meshing after large deformations, a computationally expensive job that goes against some of the possible applications of emulators.

Another major kind of approach for Lagrangian emulators uses convolutional neural networks (CNNs). These take into account the spatial distribution of the Lagrangian nodes by performing spatial convolutions over restricted neighborhoods using predefined masking functions. Originally, CNNs are designed to work with structured grids of data, as would be the pixels of an image or the nodes of an Eulerian numerical method. Therefore, their application to Lagrangian mesh-free methods is not straightforward and needs specific designs [22,23]. These approaches allow a high-level emulation and can replace the process of iterating over the neighboring particles, one of the fundamental tasks in SPH simulations, altogether. However, these models show a strong dependence on the spatial features of the training data (*e.g.*, global position of the particles), which can affect the quality of the results and the generalization capabilities. This is the case when some particles' configurations are not well represented in the training dataset.

A third kind of approach treats the emulation locally, at a particle level or particle-pairs interactions. In this case, the spatial distribution of the particles is handled with iterations over neighbors, as in the original SPH approach. Ladickỳ *et al.* [7] presents an application of AI to speed-up SPH simulations, where a random forest (RF) model learns and reproduces the behavior of each particle over time steps longer than those utilized in simulation. The RF takes as input features the SPH viscous and pressure forces (computed online with the classical SPH equations), as well as some particle state physical quantities, and returns an update of the particles state relative to the time step presented at training that can be longer than those used for the reference simulation. The emulator delivers good speed-up of the method with real-time performances. However, one of the main limits of RF models for regression tasks is the inability to extrapolate output values outside of the data ranges observed during training, which is a limitation to the consistency of the method and to generalization possibilities. A different solution is presented by Alexiadis [16], where an artificial neural network (ANN) is used to estimate the force between particle pairs. The inputs to the network are the relative particle distance, relative velocity, and the two particles densities. Because the network works on particle pairs, in this approach, the global spatial distribution of the particles in the training dataset is not as substantial as it is for CNN. This is a case where the concept of emulator is applied in a strong sense, as the network plays the same role of the reference equations. In the work by Alexiadis [16], different CFD methods are emulated (SPH, molecular dynamics, and discrete element method), showing good fidelity of the results to those of the reference models and good generalization capabilities. The paper also shows an example of model inversion, as the ANN is trained to estimate particle forces while only knowing total forces per particle. However, the study mainly covers the technical aspects of the development and testing of the emulator, whereas the adopted reference models only take into account little physical meaning (they are so-called "vanilla" models). This does not make possible a direct applicability of the emulator for practical use.

Here, we present an SPH emulator based on ANN, developed for physically consistent simulations, where the reference model is in line with SPH formulations adopted in practical applications. This study emphasizes the design aspects that have a strong impact on the physical fidelity of the reference SPH model. The design of the ANN recalls the idea proposed in the study by Alexiadis [16] and extends the overall structure of the emulator to handle the more complex and realistic physics. The presented design enforces symmetry in the predicted forces by only using particle-pair properties as inputs to the network. Specifically, we feed the network the product of the particle densities as a unique feature, as opposed to giving them individually. This also lightens the computational management of the network, which has a reduced number of inputs. To test the generalization capabilities, we train the emulator over a test case with intermediate complexity, represented by a dam break flow, and then we reproduce test cases with both lower and higher complexities.

In the following, we will begin by introducing the SPH model that we use as a reference for the emulator. In Section 3, we will discuss the design and training of the emulator, and Section 4 shows how the emulator reproduces the same problem used for training and how it generalizes to new cases. Finally, we will conclude with remarks in Section 5.

2 SPH reference model

SPH [17] is a particle-based Lagrangian mesh-free numerical method with growing applications to fluid dynamics. The Lagrangian nature of the method allows an accurate treatment of flow interfaces, including free surfaces and interactions with complex geometries [24]. The absence of any predefined interconnection between nodes (*i.e.*, meshes) allows efficient management of highly dynamic flows undergoing strong mixing [25]. This model has shown to be very flexible in its application to diverse scientific and technological fields. As a part of its flexibility, SPH comes in many

formulations [26,27], including numerical corrections [19,28,29], which can be chosen according to the characteristics of the simulated problem and the level of abstraction that one wants to adopt with respect to the simulated physics.

2.1 SPH formulation

The SPH model that we will take as reference is based on a quite general, weakly compressible formulation for inviscid fluids (as water is typically approximated), which is used in many validated applications [19,26,28,30,31]. We will consider the Navier–Stokes equations of momentum conservation, to model the dynamics of the particles, and mass conservation, to model the evolution of particles density. Our model will be developed in 2D. In this formulation, the SPH discretization of the equation for mass conservation is written as follows:

$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = \sum_j \mathbf{u}_{ij} \cdot \mathbf{x}_{ij} F_{ij} m_j + \xi h c_0 \sum_j \Psi_{ij} F_{ij} m_j, \tag{1}$$

where the second term on the right-hand side stands for the density diffusion correction, presented in [32], and it is written as follows:

$$\Psi_{ij} = \begin{cases} 2\left(\dfrac{\rho_j}{\rho_i} - 1\right), & \text{if} \quad \dfrac{|P_i - P_j|}{\rho_i \mathbf{g} |y_i - y_j|} > 1, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

with $\varepsilon_h = 0.01$.

The 2D equations of momentum conservation are written as follows:

$$\frac{\mathrm{D}\mathbf{u}_i}{\mathrm{D}t} = \sum_j -\left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij}\right) \mathbf{x}_{ij} F_{ij} m_j + \mathbf{g}, \tag{3}$$

where $\mathbf{g} = (0, -9.81)^T$ is the gravity vector and Π_{ij} is the artificial viscosity term, which helps reducing numerical high-frequency components of the velocity field, defined as follows [17]:

$$\Pi_{ij} = \begin{cases} -\dfrac{\alpha h c_0}{\rho_j} \left(\dfrac{\mathbf{u}_{ij} \cdot \mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2 + \varepsilon h^2} \right), & \text{if } (\mathbf{u}_{ij} \cdot \mathbf{x}_{ij}) > 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In order to reproduce an equivalent kinematic viscosity, ν, we define the artificial viscosity coefficient α as in [17], using the relationship

$$\alpha = \frac{8\nu}{h c_0}. \tag{5}$$

The pressure P is obtained from the density using Cole's equation of state [33]:

$$P(\rho) = c_0^2 \frac{\rho_0}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right], \tag{6}$$

where ρ_0 is the reference density of the fluid and γ is the polytropic constant that we take equal to 1. The speed of sound c_0 is taken 20 times higher than the hydrostatic velocity, *i.e.*, $c_0 = 20\sqrt{2|\mathbf{g}|h}$, with h as the maximum height of the fluid [34].

We use the fifth order Wendland smoothing kernel [35], which has been observed to be beneficial for free-surface simulations [36]. This kernel has a smoothing radius $2h$, where h is the smoothing length, defined using the smoothing factor as $\alpha_s = h/\Delta p$, and we take $\alpha_s = 1.33$.

2.2 Integration scheme

We adopt a second-order predictor–corrector integration scheme described by the following steps:

(1) Compute accelerations and density derivatives at instant n:
   (a) $\mathbf{a}^{(n)} = \mathbf{a}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)})$,
   (b) $\dot{\rho}^{(n)} = \dot{\rho}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)})$.
(2) Compute half-step intermediate positions, velocities, and densities (predictor):
   (a) $\mathbf{x}^{(n\star)} = \mathbf{x}^{(n)} + \mathbf{u}^{(n)} \frac{\Delta t}{2}$,
   (b) $\mathbf{u}^{(n\star)} = \mathbf{u}^{(n)} + \mathbf{a}^{(n)} \frac{\Delta t}{2}$,
   (c) $\rho^{(n\star)} = \rho^{(n)} + \dot{\rho}^{(n)} \frac{\Delta t}{2}$.
(3) Compute corrected accelerations and density derivatives:
   (a) $\mathbf{a}^{(n\star)} = \mathbf{a}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)})$,
   (b) $\dot{\rho}^{(n\star)} = \dot{\rho}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)})$.
(4) Compute new positions, velocities, and densities (corrector):
   (a) $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + (\mathbf{u}^{(n)} + \mathbf{a}^{(n\star)} \frac{\Delta t}{2}) \Delta t$,
   (b) $\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \mathbf{a}^{(n\star)} \Delta t$,
   (c) $\rho^{(n+1)} = \rho^{(n)} + \dot{\rho}^{(n\star)} \Delta t$.

The time step Δt is computed for each particle i, and it is required to fulfill CFL-like (Courant-Friedrichs-Lewy) stability conditions [37] determined by the acceleration magnitude and speed of sound:

$$\Delta t_i \leq \min \left\{ 0.3 \sqrt{\frac{h}{||\mathbf{a}_i||}}, 0.3 \frac{h}{c_0} \right\}. \tag{7}$$

The resulting overall time step for the model is chosen as the minimum particle time step.

2.3 Boundary model

We adopt the dynamic boundary model [38,39], for which some parallel layers of extra particles, referred to as boundary particles, are placed along analytical boundaries. These layers are spaced at intervals of Δp from one another, originating at the analytical boundary position. Fluid particles are thus positioned, commencing from a distance of Δp away from the analytical boundary. These extra layers of particles should be enough to complete the support of the smoothing kernel when it is centered on the outmost fluid particle. Boundary particles are assigned a density, which is treated equally to fluid particles. The velocity of these particles is set to zero (unless a moving boundary needs to be modeled).

3 AI-based emulators for CFD

An AI-based CFD (AI-CFD) emulator is a model that combines the principles of CFD simulations with AI techniques to enhance fluid simulations, either in terms of performances or generated information. The AI algorithms can join an equation-based CFD model or replace parts of it and *emulate* the respective behavior, learning from a dataset of simulated input–output values. Incidentally, emulators are trained to mimic CFD models with specified mathematical laws, which represent the emulated entity. This is in contrast to generic data-driven approaches, where the data can be experimental and the physical laws unknown [40,41].

Applications of AI can be divided into two main tasks [42]: classification and regression. The first task aims at dividing the data into classes, or clusters, connected to each other by intraclasses similarities, which measure the homogeneity of a cluster, and interclasses dissimilarities between different clusters, which measure the distance between two clusters [43–47]. Therefore, the output of the model is a discrete number linked to the class membership. For the regression task, the purpose is to evince the relationships between input and output data and to make it applicable to new datasets [48]. In this case, outputs are given as continuous values. For emulators, regression applications of AI are generally used, as the aim is to generate continuous simulation quantities compatible with training data. Examples of AI architectures that can be used in Lagrangian emulators for regression tasks are RFs [7] and ANNs [16,49]. The first one is known for the inference speed, and the second one for better generalization ability. The two architectures are largely used for nonlinear regression tasks.

3.1 ANN-based SPH emulator

We here design an emulator for the weakly compressible SPH scheme shown in Section 2.1, where the momentum equation is replaced by an ANN, thus emulating the forces between particle pairs.

The ANN that we use is a multilayer perceptron (MLP). This is a fully connected class of feedforward ANN composed of layers of nodes (usually called neurons) with weighted interconnections. Each node computes its output from the inputs by using so-called *activation functions* [50]. A network is typically composed of at least three layers: the first one is called input layer, the last one is called output layer, and any other layers in between are called hidden layers. The number of nodes in the input and output layers determines, respectively, the number of inputs and outputs of the network. The number of hidden layers and their nodes determines the complexity of the network and, therefore, the capability to catch details of the dataset. It is demonstrable that the standard multilayer feed-forward networks, with at least one sufficiently wide hidden layer of neurons, can approximate any kind of functions with any accuracy, and so they can be considered *universal approximators* [51]. The process of extrapolating the relationships between input and output from a representative dataset is called training, during which the weights of the connections are set.

While the outputs of a network are clearly defined by the quantity that one is looking for, in our case, the particles interaction forces, the set of input variables provided to the ANN must be designed in order to contain all the information needed to infer input–output relationships within the dataset. This design process is called *features extraction*, and it is proper of AI approaches classified as machine learning techniques [42,52], as is the one presented here. These are in contrast to deep learning techniques, where features are automatically extracted from the provided input dataset by the model during the training phase [50,53].

For the design of an emulator, the relationships between inputs and output data are typically known in analytical form (the set of equations of the reference numerical model), which is a great advantage for the process of feature extraction. Looking at our equations of interest, Eq. (3), which in turn recalls Eqs (4) and (6), it is clear that the inputs to the network will include particle positions, velocities, and densities. Still, the design of the MLP inputs requires additional effort, primarily because velocities and positions are 2D vectors, while input variables of the MLP should be scalars. We, therefore, derive scalar quantities from the variables of interest, combining them in a way suggested by how

these variables figure within the reference equations. We use three input variables, which are the relative distance between the particles, the normal component of the relative velocity, and the product of the two densities. While the first two quantities are identical to those used in [16], individual densities were used in that case as two independent inputs. Our design choice keeps the number of inputs smaller, with consequent computational advantages during training and execution of the network. Furthermore, our inputs only refer to particle-pairs quantities, forcing the network to provide symmetric interaction forces. As for the particle forces estimated by the network, they need to be vectors with the component of acceleration in the two spatial dimensions. However, considering Eq. (3), these vectors are, by construction, oriented as the vector of relative position between the interacting particles, which is known. Therefore, the network is designed to produce in output the module of the acceleration vector, which we then multiply by the unit vector of the relative position. The structure of the network was defined empirically during training, including three hidden layers with the following number of neurons: 9, 27, and 9. We note that this network is larger than the one used by Alexiadis [16], likely because of the higher complexity of the physics represented in the dataset. The activation function adopted for the nodes is the exponential linear unit (ELU) function [54].

As a measure to reduce the execution time of the emulator, during runtime, we substitute the use of the ANN with a lookup table that maps its input–output relationship [55], as done in the study by Alexiadis [16]. This table is constructed once at the conclusion of the training phase, by sweeping the inputs of the ANN across a predefined set of values and storing the corresponding outputs. During runtime, the output values of the ANN are reconstructed from this table using linear interpolation. Although this process introduces some errors in the estimation of the output, it significantly reduces the emulator's runtime and maintains its independence from the complexity of the MLP.

3.2 Training the emulator

The dataset for the training is generated by simulating a dam break flow, with the SPH formulation presented in Section 2.1. Other than being a classical benchmark case for SPH, a dam break presents a wide range of flow conditions, evolving from the initially violent flow of the transient, toward a static steady state. Therefore, the training dataset will be representative for the different flow conditions. The fluid that we model has density $\rho = 1.00\ \mathrm{kg/m^3}$, artificial speed of sound $c_0 = 560.29\ \mathrm{m/s}$, and artificial viscosity coefficient $\alpha = 0.01$. The tank is 90 m long, and the fluid is initially confined within a rectangular shape of width 30 m and height 40 m, as shown in Figure 1. We will consider three spatial resolutions, for which the whole particle set is discretized by using a spatial step of $\Delta p = 0.5$ m, $\Delta p = 0.75$, and $\Delta p = 1.0$ m, resulting in 6,681, 3,423, and 2,151 particles, respectively. The initial density is set by using an inversion of the state equation (Eq. (6)) and considering a hydrostatic pressure profile.

In order to incorporate meaningful phases of the flow into the training dataset, we simulate 50.0 s of evolution, at

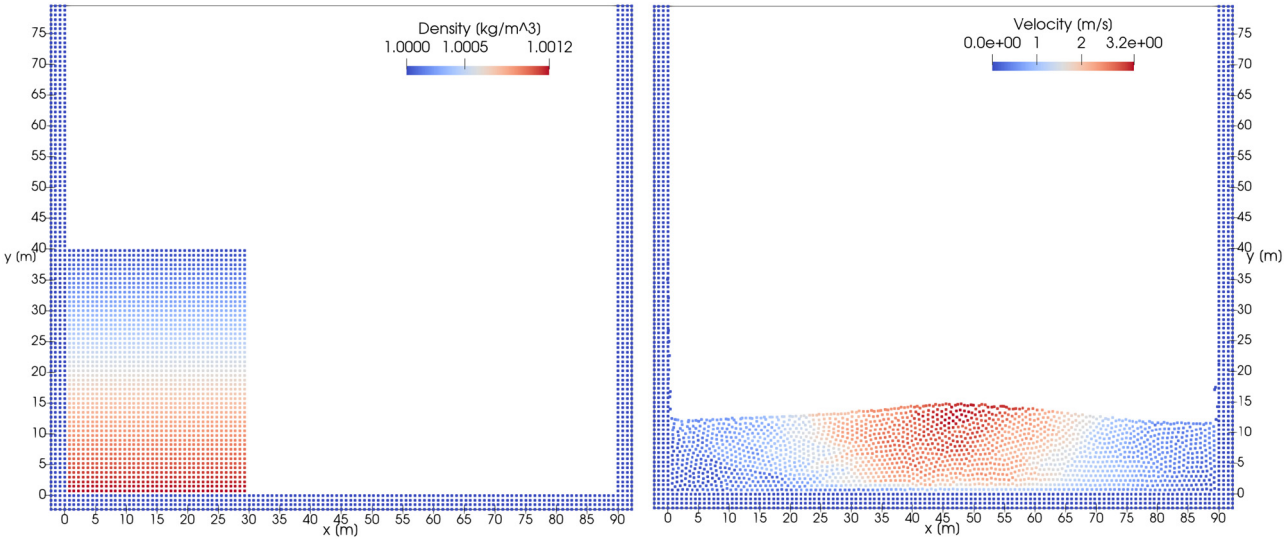

Figure 1: Dam break test case used to generate training data. On the left is the initial simulation setup with particles colored by density [$\mathrm{kg/m^3}$]. On the right is the final frame of the dataset with particles colored by velocity magnitude [m/s].

the end of which the flow reaches the state shown in Figure 1. We sample the whole particle set, including boundary particles, every 0.5 s. This results into a dataset including 3.42 million of particle samples for the intermediate spatial resolution. About 80% of the total dataset is used for the training phase, while the remaining part is used for the validation phase. We found that including boundary particles within the dataset improved the quality of the results in terms of particle disorder. This is likely due to a better representativity of surface particles.

Despite the output of the MLP represents particle interaction forces, following [16], we write the loss function L_i taking into account the total forces acting on particles. Specifically, the network is trained to minimize a total loss function, composed by the sum over all the neighbor lists for all the particles and time steps, namely of all the single L_i, which consider the difference between the total force F_i^{target} given by the SPH model and the sum of the particle pair interaction forces f_{ij}^{ANN} predicted by the MLP:

$$L_i = \left\| F_i^{\text{target}} - \sum_j f_{ij}^{\text{ANN}} \frac{\mathbf{r}_{ij}}{r_{ij}} \right\|^2 . \tag{8}$$

In this way, the sum of the estimated forces has to approximate the sum of the computed force, without knowing them individually. This approach allows to reduce the size of the training dataset, as it overcomes the need to store individual samples of the forces, in favor of storing only total forces. Total forces are computed and added to the training set for boundary particles.

Training is run for 10,000 epochs, with a learning rate of $\alpha = 5 \cdot 10^{-4}$ [50]. The Adam optimizer is used, typically reaching a final training and validation loss in the order of 10^{-3}.

We note that emulating the momentum equation means that we also emulate the computation of the kernel and the equation of state. Therefore, the parameters adopted in these expressions, like the Δp, the smoothing length, or the speed of sound, are learnt during training and will not need to be specified for the emulator. On the other hand, this means that these values are fixed for the emulator, and a new model (by means of a new training) needs to be created when one needs to use different values.

4 Results and discussion

To test the emulator, we initially replicate the dam break problem used to generate the training dataset. Figure 2 shows a comparison of three different phases of the dam break flow, and a good agreement can be appreciated between the results obtained with the equation-based SPH model (hereafter simulations) and the AI-based SPH emulator (hereafter emulations).

Some discrepancies can be seen at the level of particle configurations, especially in the regions where the details of the flow are comparable to the particle size or where the fluid behaves in a highly chaotic way (see the splashes shown in the middle and bottom frames of Figure 2). In both cases, the local flow is highly sensitive to particle positions and velocities, and even small discrepancies can generate a different content of flow features in the resulting simulations. We observed these phenomena by also reproducing the experiments, which include the three steps, *i.e.*, equation-based SPH simulation, training, and AI-based SPH emulation, with different spatial resolutions (one lower resolution with $\Delta p = 1.0$ m and one higher resolution with $\Delta p = 0.5$ m). For both lower resolutions, where the particle size becomes more important within the domain, and higher resolutions, where more sensitive flow details are generated, different forms of discrepancies emerged during the comparison of SPH-emulator results. Appendix 1 shows more details about the results obtained with the three resolutions. However, in most practical applications, this level of comparison can be disregarded, as the discrepancies can be considered compatible with simulation tolerances. On the other hand, a robust matching can be observed in terms of major flow features, like the initial advancement of the fluid front, and the interaction with the walls, including the formation of a swirl in proximity of the left wall, visible at the bottom of Figure 2.

To quantitatively compare these results, Figure 3 shows the position of the waterfront over time in the two cases, measured as the position of the rightmost particle of fluid over time. Taking into account the scale of the fluid domain and the adopted spatial resolution, the two fronts advance with a strong matching, with a little divergence developing after 2 s of evolution. By looking at the particles configurations in Figure 4, we can see that the discrepancy in the advancement of the front is generated by a slightly different particle configuration at the front of the flows. The difference is compatible with the scale of a particle size and, therefore, is comparable to the simulation tolerance. On the other hand, a very strong agreement can be observed for the overall profile of the flow.

For the emulator to be used in a scientific context, it is also important to assess the ability to generalize and correctly predict the behavior of the fluid in conditions that have not been presented during training. To this aim, we simulate two other problems that involve the same physics of the dam break but introduce different levels of complexity.

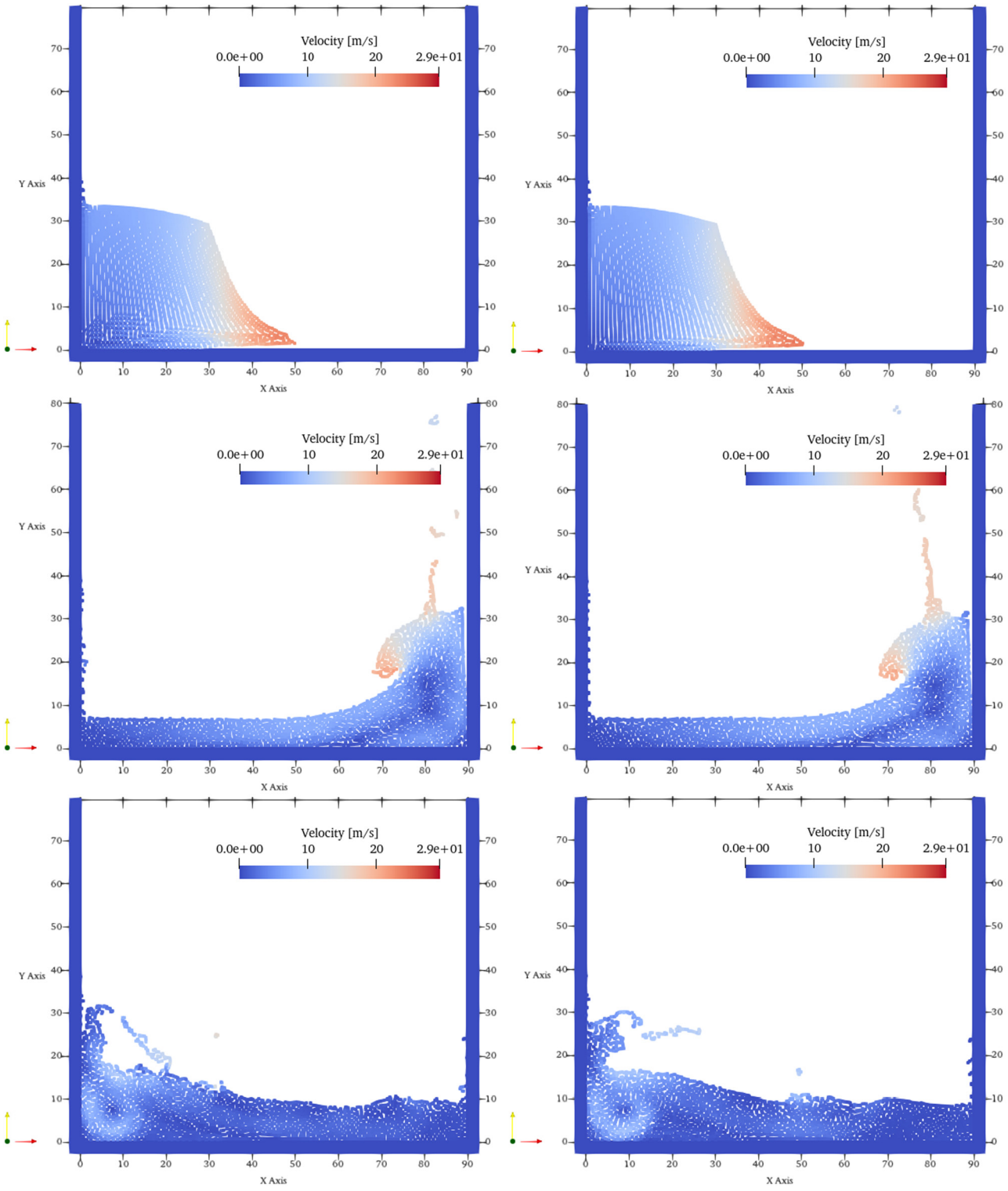

Figure 2: Simulation (left) and emulation (right) of a dam break with an artificial viscous term. The time frames per line are related to 1.5, 8.5, and 15.0 s of simulation. Particles are colored by velocity magnitude (m/s), with a minimum of 0.0 m/s and a maximum of 29.0 m/s.

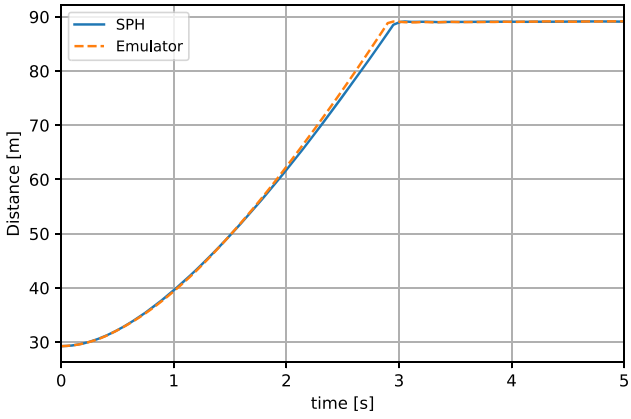

Figure 3: Dam break front over time. Solid line: SPH model; dashed line: emulator.

We study a flow at lower complexity by examining the behavior of a still volume of fluid in a tank. This is a first generalization test of the emulator, which, being trained over fluid in motion, should also be able to deal with fluid at rest. We used a 90 m by 15 m domain of fluid, with the same physical and simulation properties of the training fluid, and we used the emulator previously trained over the dam break to reproduce 200.0 s of evolution. We then simulated this problem with the original equation-based SPH model and compared the results. Although in this case the individual particles assume a different configuration, the overall behavior of the fluid was consistent with the expectations for the test case and with the respective SPH-based simulation.

To test the ability of the emulator to deal with flows with higher complexity than the training case, we include the presence of waterfalls, obtained by starting the flow from an elevated shell, again using the model trained over the dam break. Figure 5 shows the evolution of the flow, simulated with both the emulator and the reference SPH model. Similar to what observed for the dam break, some differences arise over simulated time, due to the discrepancies between the computed and estimated accelerations that gather during the simulation. However, the sequence of major flow dynamics is preserved.

As a remark about generalization capabilities, we recall the peculiarity of using an emulator that reproduces particle interaction quantities rather than directly modeling the spatial behavior of the fluid. In this case, the emulator learns to reproduce the numerical relationships between the variables in the reference model. Therefore, when the emulator is presented with a different problem to simulate than what was seen during training, as long as the values of the variables are in ranges where they hold the same physics learnt from the reference simulation, the emulator should be able to produce physically meaningful results. From the generalization tests we saw previously, even though some differences appeared in particle distribution with respect to the reference simulations, the fluid consistently exhibited a realistic behavior across all examined cases.

In addition, for the generalization tests, we reproduce the experiments (SPH simulation, training, and emulation) with the two different spatial resolutions (Δp = 1.0 m and Δp = 0.5 m), obtaining results consistent with the respective SPH-based simulations. The only limitation that we have encountered occurs when increasing flow complexity and spatial resolution, with some instabilities happening at the impact of jets or drops. However, these limitations are inherently conservative, as they primarily influence the numerical execution of the simulation and do not lead to physically unrealistic results. We have noted that training

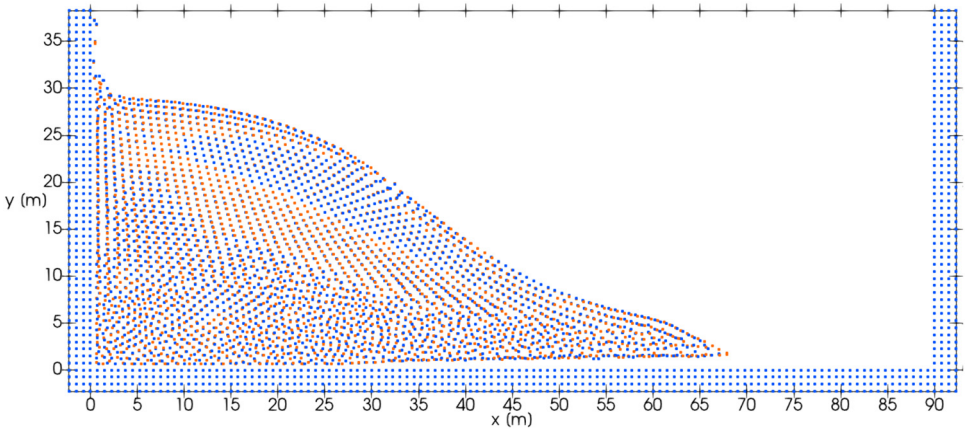

Figure 4: Dam break simulation at t = 2.2 s. Blue particles: SPH; orange particles: emulator.

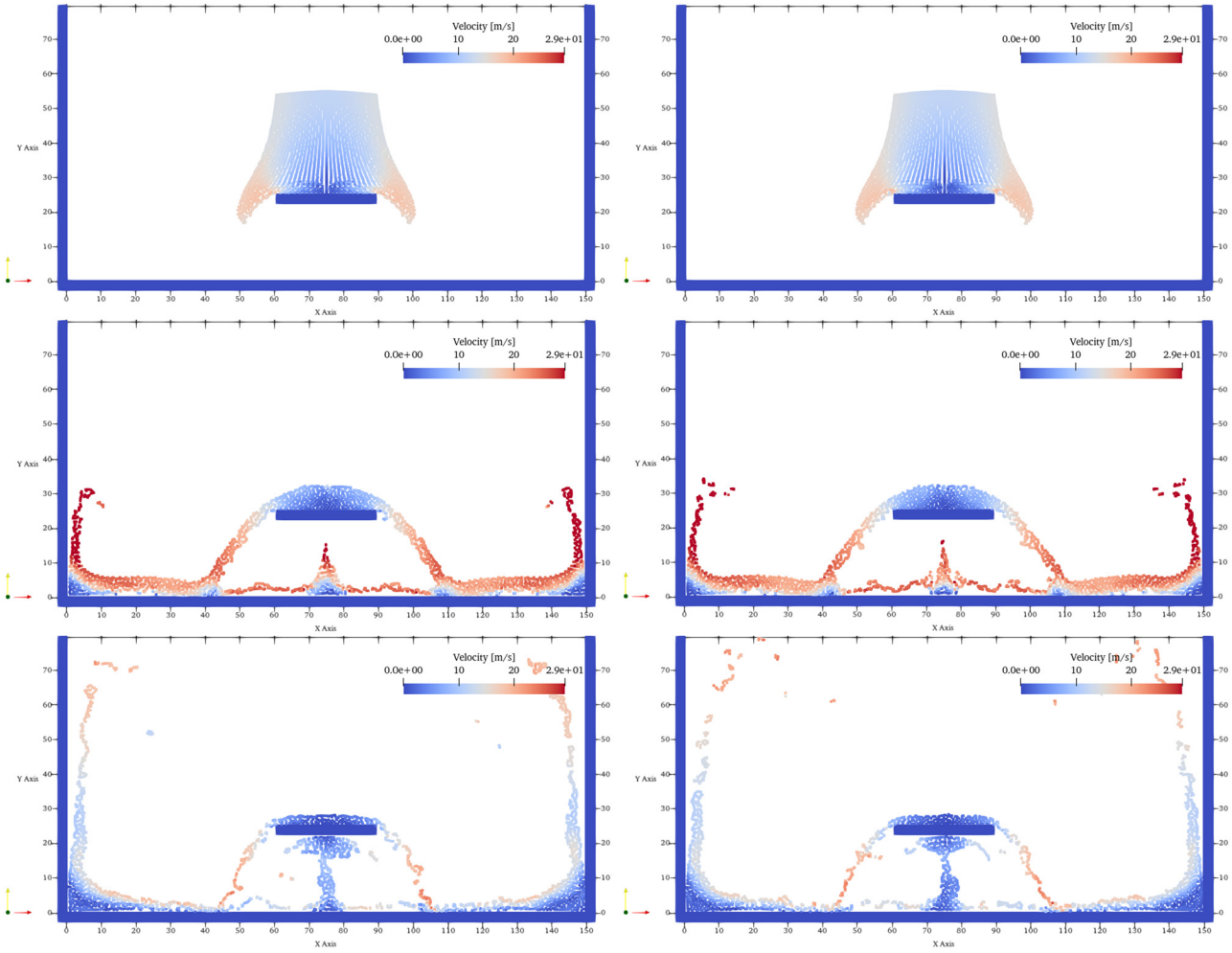

Figure 5: Simulation (left) and emulation (right) of the waterfalls problem with artificial viscous term. The time-frames per line are related to 7.5, 22.0, and 30.0 s of simulation. Particles are colored by velocity magnitude (m/s), with a minimum of 0.0 m/s and a maximum of 29.0 m/s.

the network with input SPH quantities sampled during the predictor phase of the integration (see Section 2.2) and SPH reference outputs sampled during the corrector phase (see Section 2.2), improves the stability of the emulator. This could be explained considering the similarity of this approach to an implicit integration scheme. Although this procedure introduces a deviation from the reference integration scheme, we deem this deviation to be negligible with respect to the achieved benefit in stability.

5 Conclusions

We presented an AI-CFD emulator based on the use of an ANN to emulate the SPH momentum equation and produce forces between particles starting from quantities representative of the particle state.

This emulator is meant to take into account the physical meaningfulness of the results, so it was developed by referring to SPH formulations adopted in real-world applications. We, therefore, adopt a second-order predictor–corrector integration scheme, where the density is integrated by means of a continuity equation. We adopted the dynamic boundary model and studied the effect of their presence within the training set, finding that the results of the emulator were more disordered when not including the boundary particles data. Moreover, we formed all the input features from pair-related quantities, enforcing reciprocity in the interaction, which is a requirement for energy conservation.

Simulations of a dam break problem showed that the results of the emulator are faithful to those of the equation-based simulation. To validate the model, from a qualitative point of view, we analyzed the presence of major flow features in the generated results, and from a quantitative

point of view, we studied the position over time of a dam break front, finding good levels of matching between the results.

We then studied the ability of the emulator to generalize for conditions never seen during training phase. We used the model trained over dam break data to different settings with more or less complexity with respect to the reference case. We have compared the emulations with the corresponding simulations, showing similar levels of agreement to what obtained for the validation. In addition, we also have changed the spatial resolution of the problems and repeated the experiments, to verify the robustness of this approach also for changes of this type. Even if particles distribution is not reproduced exactly, due to the differences in the accelerations estimation, these tests show how this emulator is able to reproduce the main features of the fluid, with a faithful appearance of jets, vortexes, and droplets.

Therefore, the emulator here presented is capable of faithfully reproducing traditional SPH simulations and generalizing over test cases with varying levels of complexity.

Acknowledgments: This work was developed within the framework of the Laboratory of Technologies for Volcanology (TechnoLab) at the Istituto Nazionale di Geofisica e Vulcanologia (INGV) in Catania (Italy).

Funding information: This research was funded by the ATHOS Research Program (INGV OB.FU. 0867.010), by the 2019 Strategic Project FIRST – ForecastIng eRuptive activity at Stromboli volcano: timing, eruptive style, size, intensity, and duration – of the INGV Volcanoes Department (Delibera n. 144/2020), and by Project INGV Pianeta Dinamico VT_ORME 2023-2025 (INGV OB.FU. 1020.010).

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.

Conflict of interest: Ciro del Negro, who is the co-author of this article, is a current Editorial Board member of *Nonlinear Engineering*. This fact did not affect the peer-review process. The authors declare no other conflict of interest.

References

[1] Sofos F, Stavrogiannis C, Exarchou-Kouveli KK, Akabua D, Charilas G, Karakasidis TE. Current trends in fluid research in the era of artificial intelligence: a review. Fluids. 2022;7(3):116. https://www.mdpi.com/2311-5521/7/3/116.

[2] Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE. Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mech Sin. 2021;37(12):1727–38.

[3] Li X, Feng M, Ran Y, Su Y, Liu F, Huang C, et al. Big data in Earth system science and progress towards a digital twin. Nature Rev Earth Environ. 2023;4:1–14.

[4] Morita Y, Rezaeiravesh S, Tabatabaei N, Vinuesa R, Fukagata K, Schlatter P. Applying Bayesian optimization with Gaussian process regression to computational fluid dynamics problems. J Comput Phys. 2022;449:110788. https://www.sciencedirect.com/science/article/pii/S0021999121006835.

[5] Vinuesa R, Brunton S. Enhancing computational fluid dynamics with machine learning. Nature Comput Sci. 2022;2:358–66.

[6] Dar Z, Baiges J, Codina R. Artificial neural network based correction for reduced order models in computational fluid mechanics. Comput Meth Appl Mech Eng. 2023;415:116232. https://www.sciencedirect.com/science/article/abs/pii/S0045782523003560.

[7] Ladickỳ L, Jeong S, Solenthaler B, Pollefeys M, Gross M. Data-driven fluid simulations using regression forests. ACM Trans Graphics (TOG). 2015;34(6):1–9.

[8] Kasim M, Watson-Parris D, Deaconu L, Oliver S, Hatfield P, Froula D, et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. Machine Learn Sci Technol. 2021;3(1):015013.

[9] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[10] Ummenhofer B, Prantl L, Thuerey N, Koltun V. Lagrangian fluid simulation with continuous convolutions. International Conference on Learning Representations; 2019 May 6–9; New Orleans (LA), USA.

[11] Bortnik J, Camporeale E. Ten ways to apply machine learning in the Earth and space sciences. AGU Fall Meeting Abstracts. New Orleans. Vol. 102; 2021. doi: 10.1029/2021EO160257.

[12] Mohaghegh SD, Aboaba A, Martinez Y, Shahnam M, Guenther C, Liu Y. Chapter 11 - Application of artificial intelligence to computational fluid dynamics. In: Bhattacharya S, Di H, editors. Advances in subsurface data analytics. Amsterdam, The Netherlands: Elsevier; 2022. p. 281–352. https://www.sciencedirect.com/science/article/pii/B9780128222959000017.

[13] Kim B, Azevedo VC, Thuerey N, Kim T, Gross M, Solenthaler B. Deep fluids: A generative network for parameterized fluid simulations. Comput Graph Forum. 2019;38(2):59–70.

[14] Kochkov D, Smith JA, Alieva A, Wang Q, Brenner MP, Hoyer S. Machine learning accelerated computational fluid dynamics. Proc Nat Acad Sci. 2021;118(21):e2101784118. doi: 10.1073/pnas.2101784118.

[15] Tobias P, Meire F, Alvaro SG, Battaglia PW. Learning mesh-based simulation with graph networks. arXiv: https://arxiv.org/abs/2010.03409.

[16] Alexiadis A. A minimalistic approach to physics-informed machine learning using neighbour lists as physics-optimized convolutions for inverse problems involving particle systems. J Comput Phys. 2023;473:111750.

[17] Monaghan JJ. Smoothed particle hydrodynamics. Reports Progress Phys. 2005;68:1703–59.

[18] Zago V, Bilotta G, Cappello A, Dalrymple RA, Fortuna L, Ganci G, et al. Simulating complex fluids with smoothed particle hydrodynamics. Ann Geophys. 2017;60(6):1–11.

[19] Zago V, Schulze LJ, Bilotta G, Almashan N, Dalrymple R. Overcoming excessive numerical dissipation in SPH modeling of water waves. Coastal Eng. 2021;170:104018.

[20] Li Z, Farimani AB. Graph neural network-accelerated Lagrangian fluid simulation. Comput Graph. 2022;103:201–11.
[21] Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia P. Learning to simulate complex physics with graph networks. International Conference on Machine Learning; 2020 Jul 12–17; virtual. PMLR, 2020. p. 8459–68.
[22] Ummenhofer B, Prantl L, Thuerey N, Koltun V. Lagrangian fluid simulation with continuous convolutions. In: International Conference on Learning Representations; 2020. https://openreview.net/forum?id=B1lDoJSYDH.
[23] Yan B, Harp DR, Chen B, Pawar R. A physics-constrained deep learning model for simulating multiphase flow in 3D heterogeneous porous media. Fuel. 2022;313:122693. https://www.sciencedirect.com/science/article/pii/S001623612102559X.
[24] Zago V, Bilotta G, Cappello A, Dalrymple R, Fortuna L, Ganci G, et al. Preliminary validation of lava benchmark tests on the GPUSPH particle engine. Ann Geophys. 2019;62(2).
[25] Zago V, Bilotta G, Hérault A, Dalrymple RA, Fortuna L, Cappello A, et al., Semi-implicit 3D SPH on GPU for lava flows. J Comput Phys. 2018;375:854–70. https://www.sciencedirect.com/science/article/pii/S0021999911830593X.
[26] Bilotta G, Zago V, Hérault A, van Ettinger HD, Dalrymple RA. Fast, feature-rich weakly-compressible SPH on GPU: coding strategies and compiler choices. 2022. arXiv: http://arXiv.org/abs/arXiv:220711328.
[27] Bilotta G, Zago V, Centorrino V, Dalrymple RA, Hérault A, Del Negro C, et al. A numerically robust, parallel-friendly variant of BiCGSTAB for the semi-implicit integration of the viscous term in Smoothed Particle Hydrodynamics. J Comput Phys. 2022;466:111413.
[28] Molteni D, Colagrossi A. A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. Comput Phys Commun. 2009;180:861–72.
[29] Saikali E, Bilotta G, Hérault A, Zago V. Accuracy improvements for single precision implementations of the SPH method. Int J Comput Fluid Dyn. 2020;34(10):774–87.
[30] Rogers BD, Fourtakas G, Stansby PK, Domiiinguez Alonso JM, Crespo AJ, Gómez Gesteira M, et al. DualSPHysics: an open-source code for engineering purposes. In: 2020 SPHERIC Harbin International Workshop; 2020 Jan 13–16; Harbin, China.
[31] Zago V, Dalrymple R, Almashan N, Bilotta G, Al-Houti D, Neelamani S. Characterization and modeling of greenwater overtopping of a sea-level deck. Ocean Eng. 2023;275:114131.
[32] Antuono M, Colagrossi A, Marrone S. Numerical diffusive terms in weakly-compressible SPH schemes. Comput Phys Commun. 2012;183(12):2570–80. https://www.sciencedirect.com/science/article/pii/S0010465512002342.
[33] Cole RH. Underwater explosion. Princeton, (NJ), USA: Princeton University Press; 1948.
[34] Torricelli E. Opera geometrica Evangelistae Torricellii. Florence, Italy: Masse and de Landis; 1644. http://eudml.org/doc/203881.
[35] Wendland H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv Comput Math. 1995;4:389–96.
[36] Macià F, Colagrossi A, Antuono M, Souto-Iglesias A. Benefits of using a Wendland kernel for free-surface flows. 6th ERCOFTAC SPHERIC Workshop on SPH Applications; 2011 Jun 8–10; Hamburg, Germany. p. 30–7.
[37] Monaghan JJ. Smoothed particle hydrodynamics. Ann Rev Astron Astrophys. 1992;30(1):543–74.
[38] Crespo AAJC, Gomez-Gesteira M, Dalrymple RA. Boundary conditions generated by dynamic particles in SPH methods. Comput Mater Contin. 2007 June;5:173–84.
[39] Amato E. How a CFD emulator can resolve the boundary conditions in a viscous flow. IEICE Proc Series. 2023;76(B4L-42):383. https://www.ieice.org/publications/proceedings/summary.php?expandable=13&iconf=NOLTA&session_num=B4L-4&number=B4L-42&year=2023.
[40] Torrisi F, Amato E, Corradino C, Del Negro C. The FastVRP automatic platform for the thermal monitoring of volcanic activity using VIIRS and SLSTR sensors: FastFRP to monitor volcanic radiative power. Ann Geophys. 2022;65:6, RS642.
[41] Amato E, Corradino C, Torrisi F, Del Negro C. Spectral analysis of lava flows: temporal and physicochemical effects. Il Nuovo Cimento C. 2023;46:1–4. https://www.sif.it/riviste/sif/ncc/econtents/2023/046/05/article/0.
[42] Bonaccorso G. Machine learning algorithms. Birmingham, UK: Packt Publishing Ltd; 2017.
[43] Corradino C, Amato E, Torrisi F, Del Negro C. Towards an automatic generalized machine learning approach to map lava flows. 2021 17th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA); 2021 29 Sep-1 Oct; Catania, Italy. IEEE, 2021. p. 1–4.
[44] Amato E, Corradino C, Torrisi F, Del Negro C. Mapping lava flows at Etna Volcano using Google Earth Engine, open-access satellite data, and machine learning. In: 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME); 2021 Oct 7–8; Mauritius, Mauritius. IEEE, 2021. p. 1–6. https://ieeexplore.ieee.org/abstract/document/9591110.
[45] Torrisi F, Amato E, Corradino C, Mangiagli S, Del Negro C. Characterization of volcanic cloud components using machine learning techniques and SEVIRI infrared images. Sensors. 2022;22(20):7712.
[46] Corradino C, Amato E, Torrisi F, Del Negro C. Data-driven random forest models for detecting volcanic hot spots in Sentinel-2 MSI images. Remote Sens. 2022;14(17):4370.
[47] Cariello S, Corradino C, Torrisi F, Del Negro C. Cascading machine learning to monitor volcanic thermal activity using orbital infrared data: From detection to quantitative evaluation. Remote Sens. 2023;16:1–171.
[48] Malik A, Kumar A, Rai P, Kuriqi A. Prediction of multi-scalar standardized precipitation index by using artificial intelligence and regression models. Climate. 2021;9(2):28.
[49] Zago V, Amato E, Cariello S, Corradino C, Torrisi F, Del Negro C. On Artificial Intelligence-based emulators of physical models to forecast the evolution of lava flows. EGU23 General Assembly; 2023 Apr 21–23; Vienna, Austria. Copernicus, 2023.
[50] Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA: MIT Press; 2016.
[51] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks. 1989;2(5):359–66.
[52] Amato E. Machine learning and best fit approach to map lava flows from space. Il Nuovo Cimento C. 2022. https://www.sif.it/riviste/sif/ncc/econtents/2022/045/04/article/15.
[53] Amato E, Corradino C, Torrisi F, Del Negro C. A Deep convolutional neural network for detecting volcanic thermal anomalies from satellite images. Remote Sens. 2023;15(15):3718.
[54] Clevert DA, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). 2015. arXiv: http://arXiv.org/abs/arXiv:151107289.
[55] Groeneveld D, Shan J, Vasić A, Leung L, Durmayaz A, Yang J, et al. The 2006 CHF look-up table. Nuclear Eng Design. 2007;237(15–17):1909–22.

Appendix

A Effect of resolution

The following Figures A1–A3 show a comparison of the simulations obtained with the original SPH model and the emulator. The columns of each picture show the three significant times: t_1 = 9.5, t_2 = 10.0, and t_1 = 17.5. After the dam break reaches the right wall, the returning flow tends to create an impinging jet at t_1, and some discrepancies are visible in the minor flow features between the results produced by the SPH model (shown in upper rows of the figures) and emulator (shown in lower rows of the figures). For the reasons explained in Section 4, the discrepancies degenerate in more apparent differences in the minor flow features, as visible right after at t_2. However, at t_3 the effect of the minor flow features have dissipated, and the matching between the major evolution of the flow is again visible.

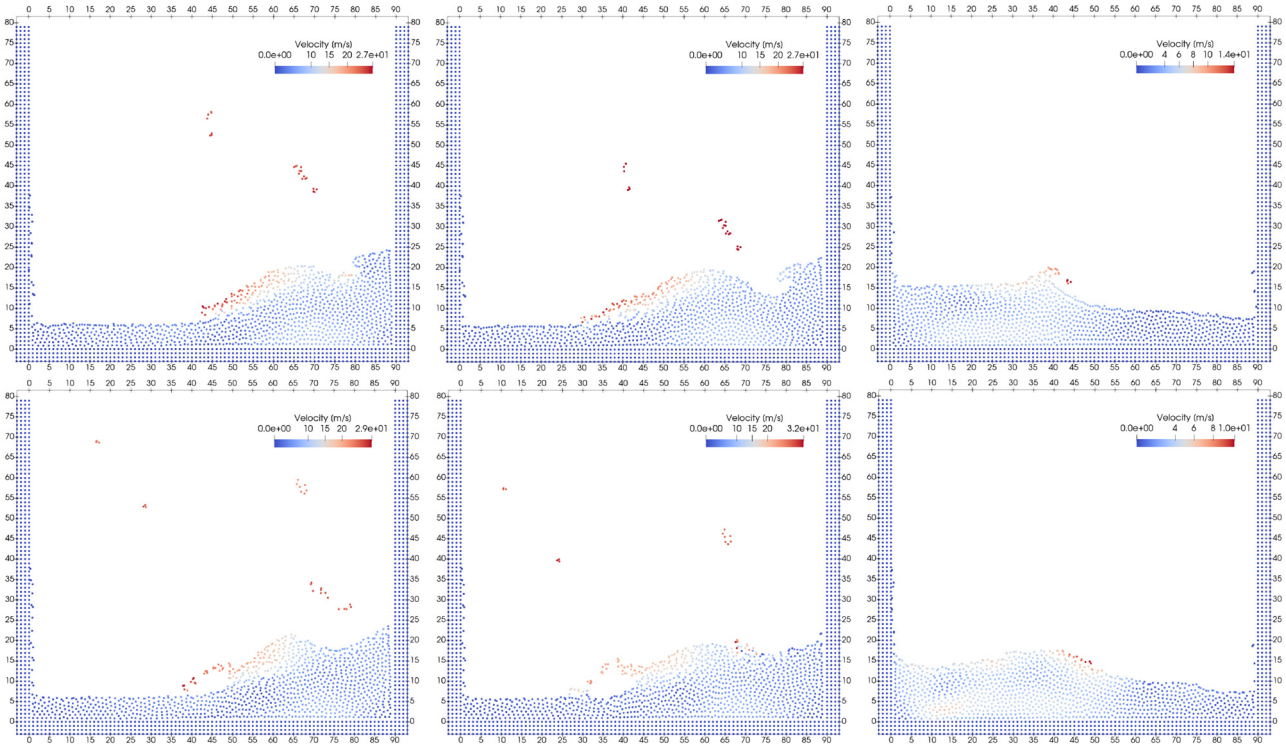

Figure A1: Δp = 1.0 m.

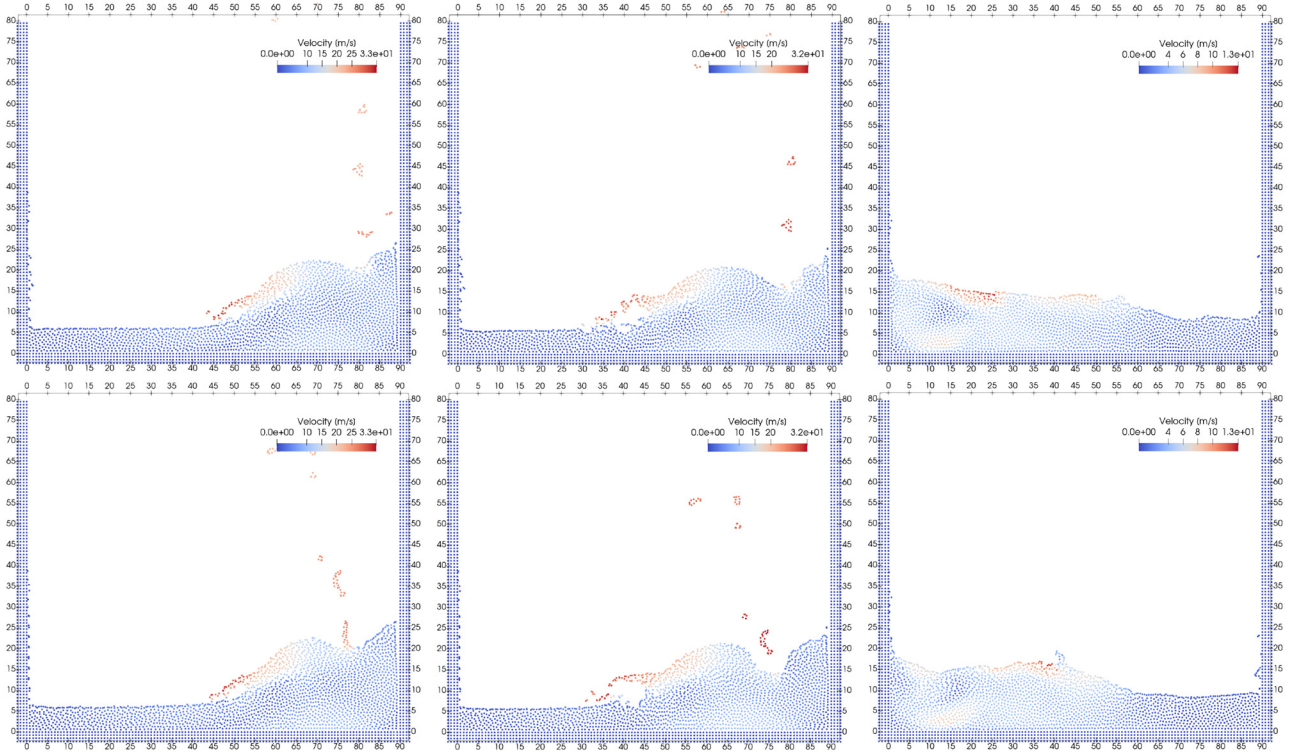

Figure A2: $\Delta p = 0.75$ m.

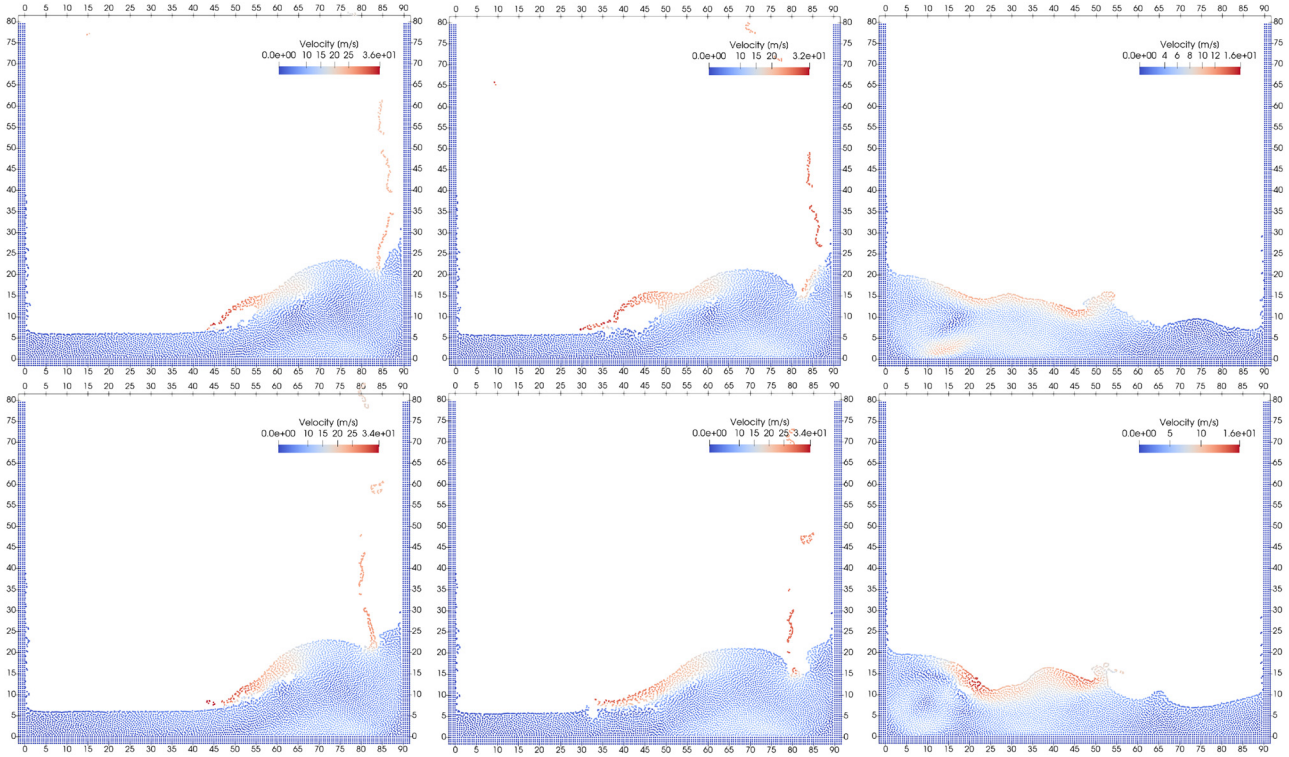

Figure A3: $\Delta p = 0.5$ m.