ANNALS OF GEOPHYSICS, 69, 1, V107, 2026; doi:10.4401/ag-9447
OPEN ACCESS

Simulations of Complex Visco-Thermal Fluids with an AI-based CFD Emulator

Eleonora Amato[*], Vito Zago, Ciro Del Negro

Istituto Nazionale di Geofisica e Vulcanologia, Sezione Osservatorio Etneo, Catania, Italy

Article history: received October 09, 2025; accepted January 20, 2026

Abstract

Physical phenomena evolve in space and time following governing laws with a high level of complexity and mathematical models can help to make accurate predictions of their behavior, describing complex fluids with a good balance between accuracy and computational costs. We have long used detailed Computational Fluid Dynamics (CFD) models to simulate complex dynamics with high accuracy, but these simulations typically entail high computational costs, resulting in long execution times and the use of expensive computational resources. To overcome these limitations, we have recently integrated CFD with Artificial Intelligence (AI), in the so-called Emulators, to expand the scope of fluid modeling, improving its performance. Here, we present an AI-based CFD emulator for Smoothed Particle Hydrodynamics (SPH) simulations, which uses an Artificial Neural Network (ANN) to enhance simulations of complex fluids with viscous and thermal components. We show the model capability to reproduce the spatio-temporal evolution of natural visco-thermal fluids. In addition, we demonstrate the emulator capacity to generalize its applicability to problems not encountered during the training phase. We also conduct a detailed error evaluation, showing that the minimal observed discrepancies do not compromise model accuracy and robustness, especially given the theoretical and computational advantages. These key points open this emulator to practical applications for natural fluids, e.g., oil, honey, or geophysical fluids such as lava, enhancing fluid modeling performance and extending functionalities. The innovation of this method improves studies in the field of numerical simulations, for example in its use as a digital Twin of a physical phenomenon, for the study of the dynamics of the system without the need of large costs for field analysis or laboratory experiments.

Keywords: Computational Fluid Dynamics (CFD); Smoothed Particle Hydrodynamics (SPH); Lagrangian method; Artificial Intelligence (AI); Artificial Neural Network (ANN); Emulator

1. Introduction

Computational Fluid Dynamics (CFD) models are crucial for studying the evolution of physical systems over space and time (Anderson and Wendt, 1995; Chung, 2002). These models numerically solve the fluid dynamics constitutive equations, producing accurate simulations of fluid behavior both in terms of phenomena modeling and final results (Zago et al., 2017). Due to their reliability and advanced calculation techniques, CFD models require high computational costs, including long execution times and substantial computational resources, which can limit modeling efficiency. The balance between accuracy and computational resources must be selected according to the purpose of the study.

One highly detailed CFD method is Smoothed Particle Hydrodynamics (SPH), a Lagrangian mesh-free particle method capable of simulating complex fluids, with applications ranging from fluid simulations with simplified physics to complex fluid simulations with detailed physical characteristics (Monaghan, 2005; Zago et al., 2019). In brief, in Eulerian methods the observed object is a specific part of the domain, fixed in space, through which the system evolves; in Lagrangian one the observed object is a single portion of the fluid, so the nodes of the discretization move in space over time (Batchelor, 1967). For these reasons, mesh-free Lagrangian methods (as SPH) allow an accurate treatment of flow interfaces, including free surfaces, and interactions with complex geometries, and the absence of grid allows efficient management of highly dynamics flows undergoing strong mixing (Zago et al., 2018, 2019). Initially SPH was introduced in the field of astrophysics independently by Gingold and Monaghan (Gingold and Monaghan, 1977), and Lucy (Lucy, 1977). Since then, it has been widely adopted in the field of fluid dynamics (Monaghan, 2005). SPH is based on a discrete approximation of governing equations, using smoothed kernels to weigh interactions between particles (Monaghan, 1992). It is particularly useful for simulating complex fluids and free-surface flows, such as water waves or lava flows (Zago et al., 2017, 2021; Zago et al., 2023), helping in monitoring natural phenomena without direct hazardous observations.

The SPH properties allow it to effectively manage irregular flow shapes, interfaces, free surfaces, and large fluid deformations. However, SPH simulations are computationally expensive, requiring long execution times and substantial computational resources, especially when modeling fluids with complex physical characteristics. For example, in some cases, weeks of computation are needed to just simulate a few minutes of physical time. Integrating CFD models with Artificial Intelligence (AI) (Bonaccorso, 2017; Goodfellow et al., 2016; Kochkov et al., 2021; Sofos et al., 2022) can enhance fluid modeling performance, addressing this limitation and extending the model's functionalities (Bortnik and Camporeale, 2021; Vinuesa and Brunton, 2022).

As shown in Amato et al. (2024a), different approaches can be used to combine CFD and AI, obtaining hybrid methods and emulators. An emulator is an AI-based model that can learn from physics-based ones how to reproduce physical behavior (Alexiadis, 2023; Amato et al., 2024a; Kasim et al., 2021). It learns from CFD simulations the behavior of reference CFD models and becomes capable to replicate it, solving fluid dynamics problems with better performance. This approach also can broaden the traditional scope of numerical models and, through inversion studies, it can reveal hidden properties of natural phenomena (Bortnik and Camporeale, 2021). These advancements bridge the gap between Earth observations and numerical modeling, leading to a deeper understanding of natural fluids and multi-scale Earth's dynamics, accelerating computational tasks and refining the interpretation of Earth observation data.

Emulators are trained to reproduce specific mathematical laws discretized via CFD models, through a physics-based approach, rather than extrapolating information from experimental data with unknown physical laws (Amato et al., 2023a; Amato et al., 2023b; Cariello et al., 2023; Corradino et al., 2024; Ramsey et al., 2023; Torrisi et al., 2022a; Torrisi et al., 2024), i.e., data-driven approaches (Amato et al., 2021; Amato, 2022; Corradino et al., 2022; Torrisi et al., 2022b).

Starting from the main distinction between Eulerian and Lagrangian numerical methods, Eulerian emulators have already been extensively studied, providing reliable and advanced results in different applications. Examples can be found in the Deep Emulator Network SEarch (DENSE) model for weather predictions (Kasim et al., 2021) or in MeshGraphNet for fluid dynamics simulations (Pfaff et al., 2020). In contrast, Lagrangian emulators are less developed, and no standard strategy has emerged. The Graph Neural Networks (GNNs) (Li and Farimani, 2022; Sanchez-Gonzalez et al., 2020; Choi and Kumar, 2024) can handle unstructured data, but require a constant re-meshing after large deformations, Convolutional Neural Networks (CNNs) (Ummenhofer et al., 2019; Yan et al., 2022), represent another option, but have limitations for application to Lagrangian mesh-free methods, because of their strong dependence on the spatial features of the training data. Another approach involves local emulators, that model the particle-particle interactions, avoiding the need of pre-established connections between nodes. This model has been applied to accelerate SPH simulations (Ladický et al., 2015) (by reproducing the behavior of each particle over longer time steps than those used in simulations), and to replicate various CFD methods with good fidelity and generalization capability (Alexiadis, 2023).

Models that combine SPH and AI to speed up simulations and enhance functionalities are also shown in Gao et al. (2023), that successfully reduced the number of simulations required for early warning of natural disasters, or in Toshev et al. (2024), in which GNNs are compared with Neural SPH-enhanced simulators for physics modeling. Other examples can be found in Higaki et al. (2025) and Zhang et al. (2023, 2024) works, that combine CFD (e.g., Incompressible SPH) with AI (CNN or GNN) for free-surface simulations, with different wall boundaries and flow dynamics.

Another class of numerical models coupled with AI is composed by the Physics-Informed Neural Networks (PINNs – Cai et al., 2021), that differ from the emulator for the way in which they are built. In particular, in a PINN the physical knowledge that governs a phenomenon is considered directly in the learning process, adding some constraint in the function to be minimized (the Loss function) or in some hidden layers of the network, to obtain the final result.

In Amato et al. (2024a), we have already presented a local emulator. It is a Lagrangian AI-based CFD emulator for inviscid fluids, that uses the SPH method and an Artificial Neural Network (ANN) to emulate the computation of hydrodynamic forces between SPH particles. The SPH method is based on a discrete approximation of the Navier-Stokes equations, and we developed a framework in which the momentum equation is replaced by a multilayer perceptron (i.e., the ANN) that learns to emulate the reference CFD model (i.e., SPH), reducing the computational costs while maintaining the good fidelity of the simulations. In this previous work, we chose this AI architecture because of its capability to approximate any kind of function with any kind of accuracy, as is demonstrated by the universal approximators theorem (Hornik et al., 1989). In the same work (Amato et al., 2024a), we verified the correctness of the physics of the emulator, demonstrating its ability to faithfully reproduce traditional SPH simulations, to remain robust across different spatial resolutions, and to generalize over test cases with varying levels of complexity. This emulator accurately predicts fluid behaviors under flow conditions not encountered during the training phase (Amato et al., 2024b; Amato, 2024). It ensures the symmetry of particle interactions, it remains independent from local particles configurations, and it is consistent with classical CFD formulations (e.g., boundary models, Amato, 2023; Amato et al., 2024b).

All the contributions introduced in Amato et al. (2024a) make it possible to apply this emulator to several practical cases, involving inviscid fluids. Natural fluids have a non-negligible viscous component which determines their evolution, and a thermal component, which is linked to the viscous term behavior.

To address this gap in possible natural fluid applications, here we present an extended enhanced version of our previous model, that includes viscous and thermal effects. Specifically, we progressively increase the physical characterization and complexity of the model by first adding the viscous component of the fluid and then incorporating the thermal model. In natural fluids, viscosity is linked to temperature: as temperature increases, viscosity decreases and vice versa (e.g., oil or lava). Therefore, we integrate this temperature-dependent viscous behavior into the model, by modifying the network to include this physical characteristic.

We test the model by comparing the simulations (results obtained with the equation-based SPH model) and emulations (results obtained with the emulator) for complex visco-thermal fluids, demonstrating also the emulator's ability to generalize across different scenarios. Our work focuses on three main objectives for CFD emulators: ensuring correctness in reproducing the physics observed in the reference CFD simulations, reducing simulation times by leveraging computational efficiency, and enhancing capabilities beyond traditional numerical models. Accordingly, we have developed this AI-based CFD emulator specifically tailored for complex visco-thermal fluids, with all the methodological aspects and details presented here. Moreover, an application of this model has been presented with a specific test case on the lava flow (Amato, 2025b). These geophysical flows have already been widely studied in Amato (2025a), from a purely numerical perspective, given us a strong interest in 2D numerical simulations of lava flow evolution in space and time, under varying simulation parameters, such as the effusion rate.

In that previous work (Amato, 2025b), the AI-based CFD emulator here presented was applied and tested over a visco-thermal fluid that presents phase transition and solidification processes and that undergoes air effects with thermal radiation and air convection (i.e., lava flows). This work was conducted to demonstrate the robustness of the emulator (AI architecture), and to reproduce and generalize it under different physical conditions, enhancing the model applicability and extending classic functionalities.

Here, formal and methodological aspects of the AI-based CFD emulator are presented, providing a detailed study of the definition and construction of the model.

2. Method: an AI-based SPH emulator

2.1 SPH reference formulation

The SPH method allows for various formulations (Bilotta et al., 2022a, 2022b), with several numerical corrections (Molteni and Colagrossi, 2009; Saikali et al., 2020; Zago et al., 2021), chosen according to the characteristics of the problem under analysis and the desired level of abstraction. As in Amato et al. (2024a), we use a 2D quite general Weakly Compressible SPH formulation, following Zago et al. (2018).

Briefly, we outline the governing equations of fluid dynamics. The continuity equation, which expresses the law of mass conservation, is written as in the following Eq. (1),

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\vec{u}, \tag{1}$$

where ρ is the density, t is the time, and $\vec{u}$ the velocity. $\frac{D}{Dt}$ is the Lagrangian or material (total) derivative,

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u}\cdot\nabla.$$

The conservation of momentum equation is expressed by the Navier-Stokes equation which, for incompressible fluids ($\frac{D\rho}{Dt} = 0$), is given by Eq. (2),

$$\rho\frac{D\vec{u}}{Dt} = -\nabla P + \mu\nabla^2\vec{u} + \vec{G}, \tag{2}$$

where P is the pressure, μ is the viscosity, and $\vec{G}$ is the external force (gravity). It can also be written as:

$$\rho\frac{D\vec{u}}{Dt} = -\nabla P + \nabla\cdot(\mu\nabla\vec{u}) + \vec{G}, \tag{3}$$

when the viscosity is not homogeneous, and it can spatially variate.

The SPH discretization of the mass conservation equation (equation of continuity) is written as

$$\frac{D\rho_i}{Dt} = -\sum_j \vec{u}_{ij}\cdot\vec{x}_{ij}F_{ij}m_j + \xi h c_0\sum_j \Psi_{ij}\,F_{ij}m_j, \tag{4}$$

for a particle i,where $\vec{x}_i$ and $\vec{y}_i$ are position coordinates, $\vec{u}_i$ is the velocity, m_i is the fixed mass, ρ_i is the density, $V_i = \frac{m_i}{\rho_i}$ is the volume, and P_i is the pressure. We can also define $\vec{x}_{ij} = \vec{x}_i - \vec{x}_j$ and $\vec{u}_{ij} = \vec{u}_i - \vec{u}_j$, $F(r) = \frac{1}{r}\frac{\partial W}{\partial r}$, with $r = |\vec{x}_{ij}|$ the distance, then $F_{ij} = F(r)$, and $\nabla_j W_{ij} = -\vec{x}_{ij}F_{ij}$. Moreover, c_0 is the speed of sound, h is the smoothing length, defined using the smoothing factor as $s_f = \frac{h}{\Delta p}$, with Δp being the spatial resolution of the approximation (particle size) and, typically, $s_f = 1.33$. Ψ_{ij} is the density diffusion term, following the formulation of Antuono et al. (2012), with ξ as the density diffusion coefficient, typically $\xi = 0.1$:

$$\Psi_{ij} = \begin{cases} 2\left(\frac{\rho_j}{\rho_i} - 1\right) & \text{if} \quad \frac{|P_i - P_j|}{\rho_i\vec{g}|\vec{y}_i - \vec{y}_j|} > 1 \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

The discretization for the viscous equation of momentum conservation (with physical viscosity), i.e., the Navier-Stokes equation, is written as

$$\frac{D\vec{u}_i}{Dt} = +\sum_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \vec{x}_{ij} F_{ij} m_i - \sum_j \frac{2\mu_{ij}}{\rho_i \rho_j} \vec{u}_{ij} F_{ij} m_i + \vec{G}, \tag{6}$$

where $\vec{G} = (0, -9.81)^T$ is the gravity vector and μ_{ij} is a parameter for the viscosity. For the case in which viscous term is independent of temperature, the physical viscosity is set as $\mu = 15.0$ Pa s (otherwise, for temperature-dependent viscous case, $\mu_i(T_i)$ is defined as in Eq. (15) in paragraph 2.3).

The pressure P is obtained from the density (and not from the Poisson equation), so it derives from the Cole's state equation (Cole, 1948), and it can be expressed as Eq. (7)

$$P(\rho) = c_0^2 \frac{\rho_0}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right], \tag{7}$$

where ρ_0 is the reference density of the fluid at rest, and γ is the polytropic constant. In the simulations, the physical parameters used are density $\rho_0 = 1.00$ kg/m^3, polytropic constant $\gamma = 1$, and the speed of sound c_0 is taken as $c_0 = 20\sqrt{2||\vec{g}||d}$, with d being the maximum height of the fluid (Torricelli, 1644).

The fluid thermal evolution is described by the heat equation, as in Eq. (8)

$$\frac{DT}{Dt} = \frac{1}{c_p \rho} \nabla(\kappa \nabla T), \tag{8}$$

where T is the temperature, c_p the specific heat at constant pressure, and κ the thermal conductivity.

The SPH discretization of the thermal equation is as the Eq. (9):

$$\frac{DT_i}{Dt} = -\frac{1}{c_p \rho_i} \sum_j \frac{2 m_j \kappa_{ij} T_{ij}}{\rho_j} F_{ij}, \tag{9}$$

with T_{ij} being the relative temperature between particles. In this case, the following physical parameters are used: $c_p = 1000.0$ J kg^{-1} K^{-1}, $\kappa_{ij} = 500.0$ W m^{-1} K^{-1} (value chosen to accentuate the heating dynamics of a highly conductive fluid), $T_{fluid} = 293.15$ K as the temperature of the fluid, and $T_{wall} = 273.15$ K as the temperature of the boundaries.

We adopt the quintic Wendland Smoothing kernel with $h = 1.33\ \Delta p$ as the smoothing length (Wendland, 1995), which has advantages when applied to free-surface simulations (Macia Lang et al., 2011). The simulation domain has dimensions 90×80 m, and the initial fluid shape is of 30×40 m.

2.2 Integration scheme

Once the spatial derivatives have been discretized, it is necessary to integrate the resulting equations over time. Therefore, we adopt a second order predictor-corrector integration scheme (Zago et al., 2021), that follows these steps. Firstly, we calculate accelerations and density derivatives for particles at time n as in Eqs. (10),

$$\begin{aligned} \vec{a}^{(n)} &= \vec{a}\left(\vec{x}^{(n)}, \vec{u}^{(n)}, \rho^{(n)}\right) \\ \dot{\rho}^{(n)} &= \dot{\rho}\left(\vec{x}^{(n)}, \vec{u}^{(n)}, \rho^{(n)}\right) \end{aligned} \tag{10}$$

and we calculate an half-step for the intermediate state of the particles (composed by position, velocity and density), namely the predictor step (Eqs. 11),

$$\begin{aligned} \vec{x}^{(n*)} &= \vec{x}^{(n)} + \vec{u}^{(n)}\frac{\Delta t}{2} \\ \vec{u}^{(n*)} &= \vec{u}^{(n)} + \vec{a}^{(n)}\frac{\Delta t}{2} \\ \rho^{(n*)} &= \rho^{(n)} + \dot{\rho}^{(n)}\frac{\Delta t}{2} \end{aligned}. \tag{11}$$

Subsequently, we calculate the corrected accelerations and density derivatives for particles (Eqs. 12),

$$\begin{aligned} \vec{a}^{(n*)} &= \vec{a}\left(\vec{x}^{(n*)}, \vec{u}^{(n*)}, \rho^{(n*)}\right) \\ \dot{\rho}^{(n*)} &= \dot{\rho}\left(\vec{x}^{(n*)}, \vec{u}^{(n*)}, \rho^{(n*)}\right) \end{aligned} \tag{12}$$

and we calculate the new particles state, in the corrector step (Eqs. 13),

$$\begin{aligned} \vec{x}^{(n+1)} &= \vec{x}^{(n)} + \left(\vec{u}^{(n)} + \vec{a}^{(n*)}\frac{\Delta t}{2}\right)\Delta t \\ \vec{u}^{(n+1)} &= \vec{u}^{(n)} + \vec{a}^{(n*)}\Delta t \\ \rho^{(n+1)} &= \rho^{(n)} + \dot{\rho}^{(n*)}\Delta t \end{aligned}. \tag{13}$$

Simulations advance with time steps of Δt, repeating the computation of derivatives and integration. Δt is calculated starting for each particle i (Δt_i) and it is needed to fulfill the CFL-like (Courant-Friedrichs-Lewy) stability conditions (Monaghan, 1992; Zago et al., 2018). The relative equation involves an acceleration magnitude component ($\sqrt{\frac{h}{||\vec{a}_i||}}$), a speed of sound component ($\frac{h}{c_0}$), a viscosity component ($\frac{\rho_0 h^2}{\mu_i}$), and a forth condition linked to the thermal diffusivity ($\frac{\rho_0 c_p\, h^2}{\kappa}$) when the thermal model is present in the simulation. Using these conditions, we choose the Δt_i as in Eq. (14),

$$\Delta t_i \leq \min\left\{ c_1\sqrt{\frac{h}{||\vec{a}_i||}}, c_2\frac{h}{c_0}, c_3\frac{\rho_0 h^2}{\mu_i}, c_4\frac{\rho_0 c_p\, h^2}{\kappa} \right\} \tag{14}$$

where $c_1 = 0.3$, $c_2 = 0.3$, $c_3 = 0.125$, and $c_4 = 0.1$.

Therefore, the time step used for simulation and emulation is $\Delta t = \min \Delta t_i$. It is evident that Δt directly affects the duration of a simulation, the smaller it is, the more steps are needed, and the longer the simulation will take. Here, we reach a Δt with an order of magnitude of 10^{-5}, resulting in slow simulations.

2.3 Training the ANN

We use the framework outlined in references Amato et al. (2024a), Amato (2024), where the momentum equation is emulated using a multilayer perceptron (ANN) (Alexiadis, 2023), following the universal approximators theorem (Hornik et al., 1989). The network is trained to replicate individual particle-particle interactions, minimizing the *loss function L*. L is obtained as the difference between the sum of particle-particle interaction forces for a particle and the reference (target) total force on the particle, for all the particles and time steps. We explore two cases, starting with only the viscous term and subsequently adding the thermal model, coupled with the viscous one.

In the temperature-independent viscous case (case one), the network takes as input at each iteration the relative position between two particles (x_{ij}), the normal component of relative velocity ($u_{n_{ij}}$, although the original choice of the normal component was made to remain consistent with Amato et al. (2024a), in the case one of this work we also tested the use of the tangential component or the complete vector, not obtaining a great improvement compared to the case presented here), and the product of the two densities ($\rho_i \cdot \rho_j$). In the visco-thermal case (case two), we introduce the harmonic mean of the two neighboring particles viscosities ($\bar{\mu}_{ij}$), as a fourth feature, and retrain the emulator accordingly. In both cases, we select the input feature vector based on the conservation of momentum equation, focusing on the independent variables therein. The ANN output for the two cases is the magnitude (module) of the acceleration vector, subsequently multiplied by the unit vector of the relative position to have the relative vector.

For both case studies, we train the ANN using the dam break problem, chosen for its diverse flow conditions over space and time, representative of the physics the emulator will handle. We sample 50.0 seconds of evolution every 0.5 seconds, discretizing the fluid with around 3400 particles, at a spatial resolution of $\Delta p = 0.75$.

For the case one, following Amato et al. (2024a), the network comprises a three-dimensional input layer, three hidden layers with 9, 27, and 9 neurons respectively, and a one-dimensional output layer. This network has been empirically found to provide a good balance between computational load of the network itself and training accuracy (increasing the number of neurons or layers does not significantly change the quality of the model at the expense of computational costs, while reducing it leads to a loss of accuracy). In case two, the network consists of a four-dimensional input layer, three hidden layers with 12, 48, and 12 neurons, and a one-dimensional output layer. This architecture has been selected for the same reasons discussed for the case one, and in particular also to maintain a mathematical symmetry between the ratio of the hidden layers neurons and the input dimension, which changes according to the selected input features (i.e., case one: $27/9 = 3 =$ input layer dimension; case two: $48/12 = 4 =$ input layer dimension). A deeper architecture (e.g., 16, 64, and 16 neurons) would introduce a significant higher computational cost without providing substantial performance improvements.

In case two, we define a functional form (Eq. 15) to couple temperature and viscosity numerically. This temperature-dependent viscosity is used in the conservation of momentum equation, to compute particle velocity.

$$\mu_i(T_i) = c\ e^{\frac{1}{T_i}}, \tag{15}$$

where $c = 438.41$ is a constant of proportionality derived from the initial viscosity at temperature $T_0 = 293.15$ K using the exponential functional form $\mu_0 = e^{1/T_0}$ and the linear coupling $\mu_0 = 1.5 \cdot T_0$.

As in Amato et al. (2024a), we train the network for 10000 epochs in the temperature-independent viscous case, and we use 5000 epochs in the temperature-dependent viscous case. In the first case, this number of epochs is a good trade-off between performance and complexity (we tried to vary this value, but this number of epochs balances the two aspects well). For the second case, although physics is more complex, the additional input features can help the model represent the data more efficiently. As a result, the training required epochs could be reduced. We use a learning rate of $\alpha = 5 \cdot 10^{-5}$ (Goodfellow et al., 2016), looking at any fluctuations in the cost function and convergence to the minimum. The dataset is split into training and validation sets. To split the data in this way, we define a random permutation of the data and select the two sets, with 80% of the data allocated for training, and the remaining 20% for validation. To test the model, we use it after the training, comparing the simulations (results obtained with the equation-based SPH model) and emulations (results obtained with the emulator). The Exponential Linear Unit (ELU) function (Clevert et al., 2015) is chosen as activation function for the nodes. All the code is written in Python using PyTorch libraries and Numba compiler. Training is performed on a GPU (Graphics Processing Unit) NVIDIA RTX4090, while simulations and emulations are run on a CPU (Central Processing Unit) INTEL i9 (Bilotta et al., 2016, 2024).

To optimize performance, similar to references (Alexiadis, 2023; Amato et al., 2024a), we substitute the ANN with a lookup table, to efficiently map its input-output relationship (Groeneveld et al., 2007).

3. Results

We have trained and verified the model reliability and its generalization capacity, also reaching a global speed-up of the simulation times.

We have used a dam break scenario for training, initially with temperature-independent viscosity, and subsequently incorporating a thermal model coupled with the viscous one. Following a standard setup (also detailed in Amato et al., 2024a, Amato, 2024), we simulate the dam break using an equation-based SPH model. We sample the resulting dataset to train the ANN of the AI-based CFD emulator, which is then used to emulate the same problem. Furthermore, we assess the trained emulator ability to generalize, by applying it to different problems than the one used during training. For each case study, we compare the simulation (equation-based SPH simulation target) with the emulation (AI-emulated SPH ANN output); both of them have been executed with identical problem setting (same number of particles, spatial resolution, and simulation duration).

3.1 Reliability and generalization capacity of the emulator

Starting with the temperature-independent viscous case, Fig. 1 compares the simulation and emulation of a viscous fluid within a simulation domain consisting of a box with flat topography at time $t = 2.5$ seconds. The particles are color-coded by velocity (m/s). The comparison shows a close match between simulation (Fig. 1a) and emulation (Fig. 1b), in terms of flow shape, extent, and overall fluid behavior, demonstrating the effectiveness of the emulator in handling viscous fluids.

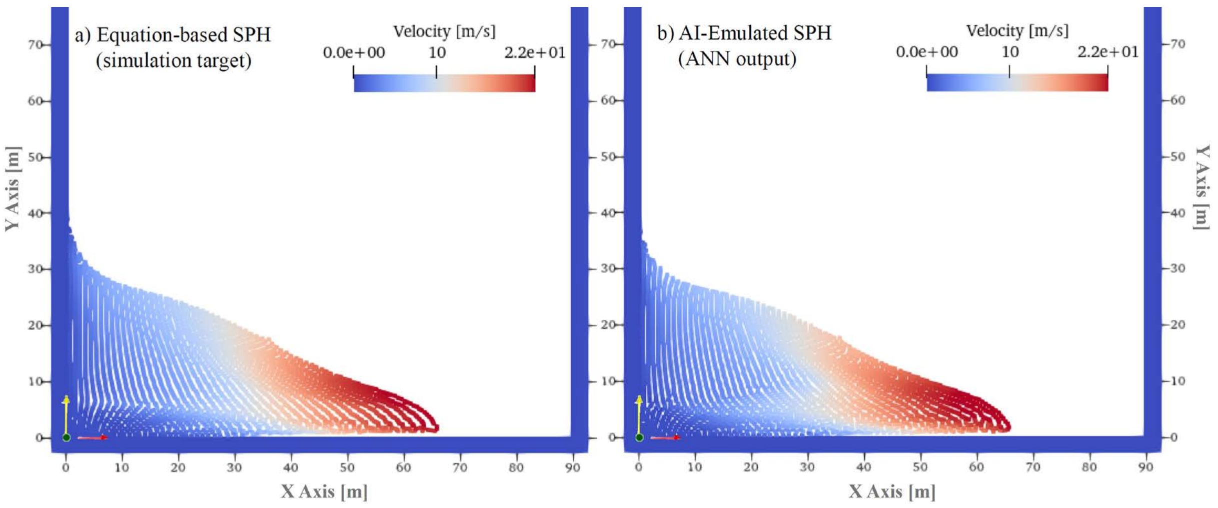

Figure 1. Simulation (a) and emulation (b) of a viscous fluid in a domain with flat topography at time $t = 2.5$ s. The particles are colored by velocity magnitude [m/s].

In order to quantitatively assess the correctness of the results, we show in Fig. 2 a comparison between the simulated and emulated dam break fronts over time, obtained measuring the position of the rightmost fluid particle over time. There is a perfect match between the two fronts, highlighted also by Fig. 3, in which we manually overlapped the two fluids at time $t = 2.5$, with blue particles for simulation and orange particles for emulation.

Figure 4 depicts the results for the generalized case. We altered the topography by adding bumps to simulate an irregular inclined plane, using the emulator trained with the dam break scenario in flat topography (all other parameters remain the same). Comparing emulator results (Fig. 4b) with the respective simulations (obtained using the same topography and parameters and only the equation-based SPH model, Fig. 4a), it demonstrates its ability to generalize and perform well on problems not encountered during the training phase, well reproducing the shape of the flow, only with a little overestimation of the fluid extension. In particular, simulation front travels a distance of 136.7 m, emulation front of 145.7 m, therefore the second flow covered 106.6% of the distance of the first one at the same time (relative percent error: 6.6%). The same particle analyzed (the rightmost particle of the simulated and emulated fluid) has a X-axis component velocity of 1.31 m/s for the simulation and 1.36 m/s for the emulation. This suggests that the emulator accurately reproduces the momentum transfer but tends to slightly under-predict the dissipative effects (e.g., viscous friction or topographic obstacles), leading to a cumulative overestimation of the flow front.

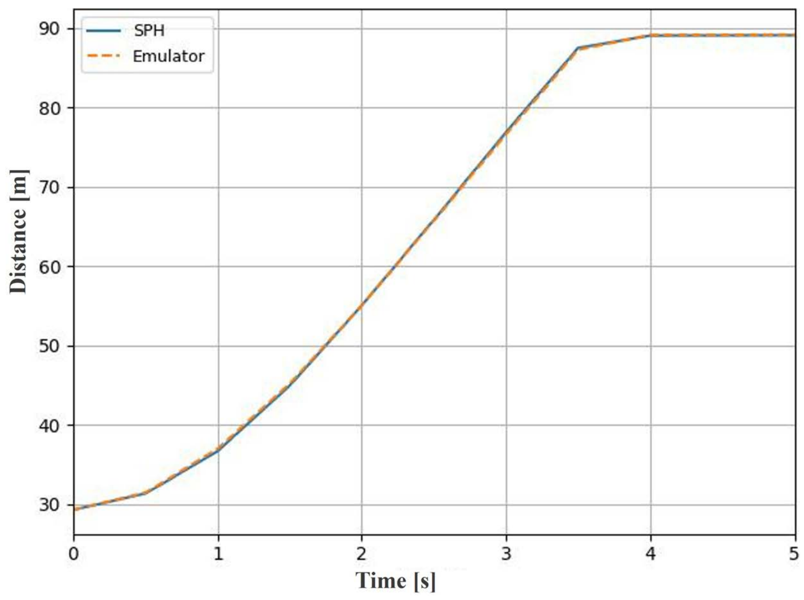

Figure 2. Viscous simulated and emulated dam break fronts over time, measuring the position of the rightmost particle of fluid over time. Note that the front reach and hit the right wall of the box already after 4 seconds.

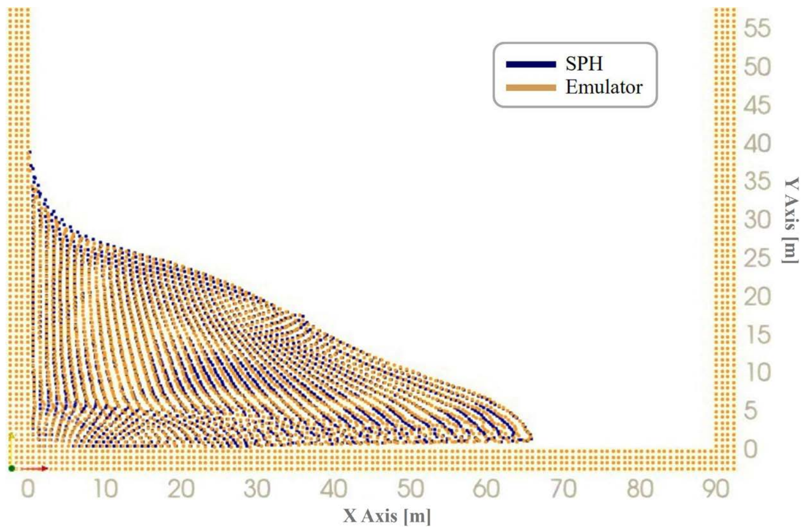

Figure 3. Relative particles configuration of the overlapped simulated and emulated viscous dam break fronts at time $t = 2.5$ s, with blue particles for equation-based SPH simulation and orange particles for AI-based emulation.

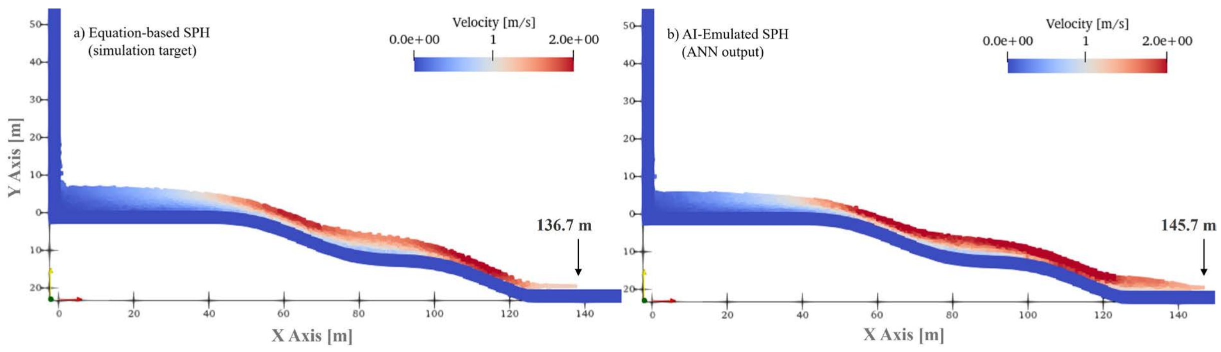

Figure 4. Simulation (a) and emulation (b) of a viscous fluid over a topography with bumps at time $t = 20.0$ s. The model used was trained over an SPH viscous simulation with a flat topography domain. The particles are colored by velocity magnitude [m/s].

Moving to the temperature-dependent viscous case, Fig. 5 presents a comparison between the simulation and emulation of the visco-thermal dam break within the simulation domain with flat topography at $t = 25.0$ seconds. Particles are color-coded by temperature (K). The results show consistent agreement between simulation (Fig. 5a) and emulation (Fig. 5b), in terms of shape and trend, affirming the ability of the emulator to also reproduce this visco-thermal coupled behavior. Also in this case a slight overestimation of the emulator front is present, reaching a distance of 67.6 m and 69.7 m in the case of simulation and emulation front, respectively. The second flow covered 103.1% of the distance of the first one at the same time (relative percent error: 3.1%, at time $t = 25.0$ s). The same particle analyzed has an X-axis component velocity of 0.68 m/s and a temperature of 292.49 K (viscosity: 439.91 Pa s) for the simulation, and an X-axis component velocity of 0.65 m/s and a temperature of 291.60 K (viscosity: 439.92 Pa s) for the emulation.

Also in this case, to quantitatively assess the results correctness, we show in Fig. 6 a comparison between all the simulated and emulated dam break fronts over time. The graph presents some discrepancies between

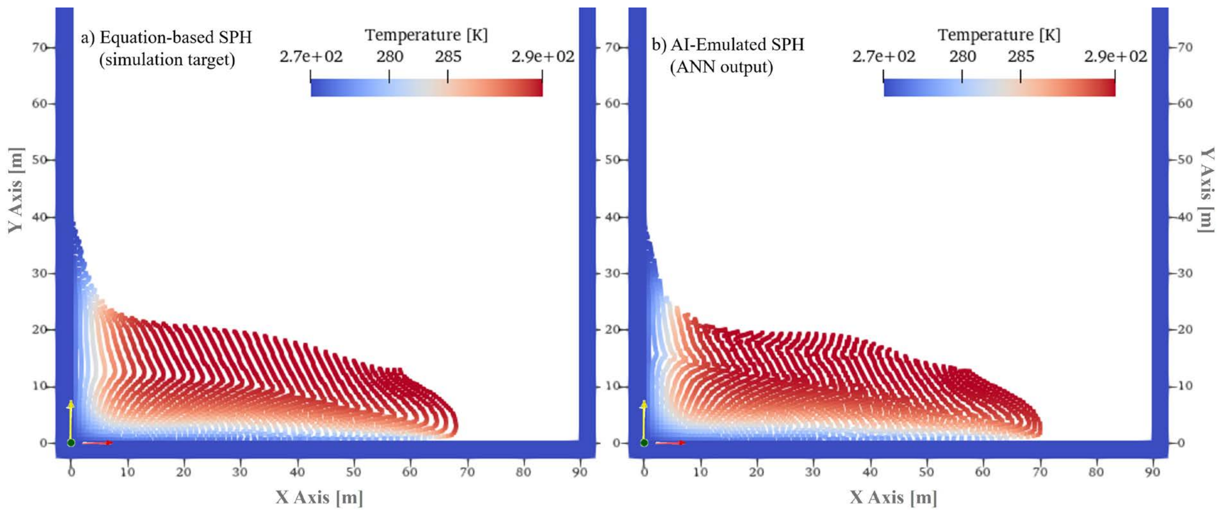

Figure 5. Simulation (a) and emulation (b) of a visco-thermal fluid in a domain with flat topography at time $t = 25.0$ s. The particles are colored by temperature [K].

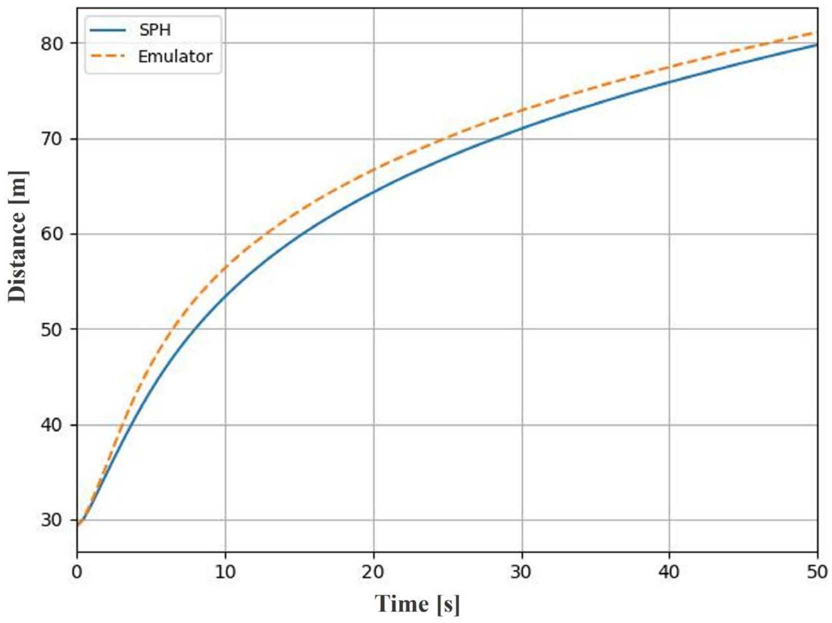

Figure 6. Visco-thermal simulated and emulated dam break fronts over time, measuring the position of the rightmost particle of fluid over time. Full duration of simulation and emulation results are reported, noting that high fluid viscosity precludes contact with the right wall.

the two fronts. In particular, if we set a given distance (e.g., 50 m), the emulation front slightly precedes the simulation front (emulation front reaches that distance after around 6 s, simulation front after around 7.5 s). Similarly, if we instead set a specific time step (e.g., $t = 50$ s), the emulation front has traveled a greater distance than the simulation front (around 80.9 m and 79.6 m, respectively, the emulator covered 101.6% of the simulator distance).

These differences in Fig. 6 (with the following statistics: maximum of around 3 m, mean of around 2 m, standard deviation of 0.63 m) can be tolerated and considered in the numerical error of the simulation. This discrepancy is also shown in Fig. 7, in which we show simulated and emulated fluid front overlapped at time $t = 25.0$, with blue particles for simulation and orange particles for emulation.

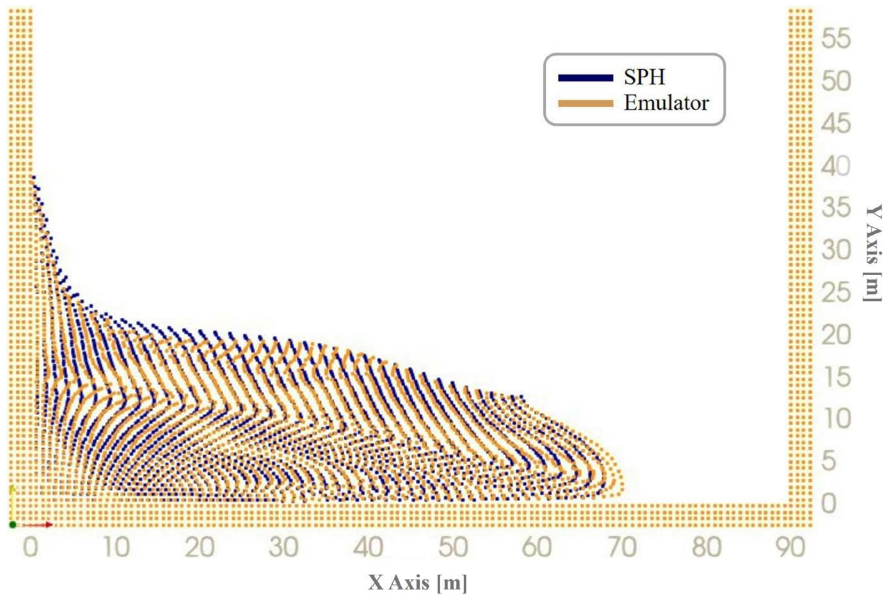

Figure 7. Relative particles configuration of the overlapped simulated and emulated visco-thermal dam break fronts at time $t = 25.0$ s, with blue particles for equation-based SPH simulation and orange particles for AI-based emulation.

Figure 8 shows the results for the generalized case. In this case too, the model trained with the flat topography scenario (Fig. 5) is applied to a setting with bumps. Comparing the emulation results (Fig. 8b) with the corresponding simulations (same setting and only the equation-based SPH model, Fig. 8a), we observe that the emulator is able to effectively generalize to scenarios not encountered during the training phase, with the same slight overestimation of the fluid extension of the precedent case. The simulated front travels a distance of 83.0 m (temperature: 289.07 K, viscosity: 433.6 Pa s) and the emulated front of 85.8 m (temperature: 287.5 K and viscosity 439.94 Pa s), thus the emulator covered 103.4% of the simulation distance with an overestimation of 3.4%. From a physical point of view, it is noteworthy that the emulator predicts a slightly lower temperature and, consequently, a higher viscosity compared to the simulator, as it is right given the laws of physics. Despite this increased flow "resistance" (due to higher viscosity), the emulated front still travels further; therefore, the overestimation seems not to be linked to a thermal-rheological misestimation, but rather to the emulator's difficulty to model all the physical detailed dynamics in this exemplified model (precisely because it is still a model) albeit accurate (e.g., necessary dissipative action).

Here, we have also investigated another generalization case, with an obstacle in the topography. Figure 9 shows an example of this application, using the same model trained on the flat topography scenario (Fig. 5) and this new setting (all other parameters remain the same). Comparing the emulation results (Fig. 9b) with the corresponding simulations (equation-based SPH model, Fig. 9a), we observe that the emulator is also able to replicate the fluid behavior to overcome barriers, typical of real applications. In this case, the simulated front travels a distance of 74.5 m (temperature: 279.61 K, viscosity: 438.81 Pa s) and the emulated front of 74.9 m (temperature: 282.32 K and viscosity 439.97 Pa s), with only a 0.5% of overestimation.

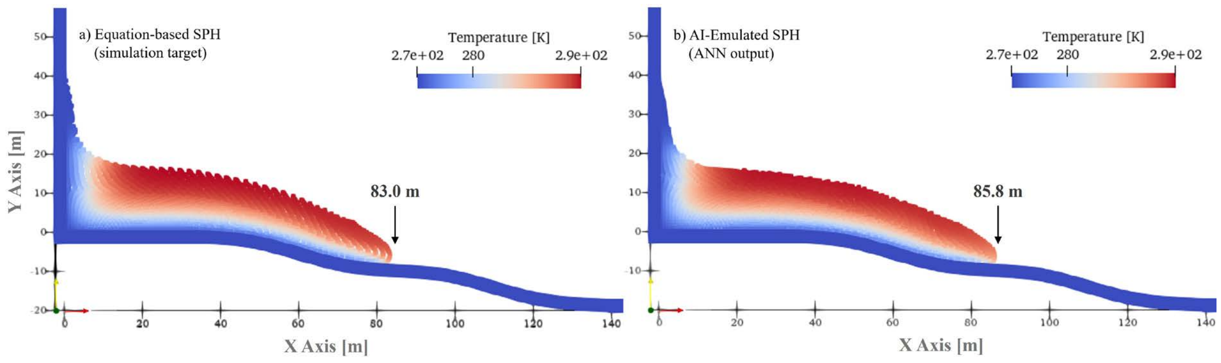

Figure 8. Simulation (a) and emulation (b) of a visco-thermal fluid over a topography with bumps at time $t = 45.0$ s. The model used was trained over an SPH viscous simulation with a flat topography domain and generalized to this case study. The particles are colored by temperature [K].

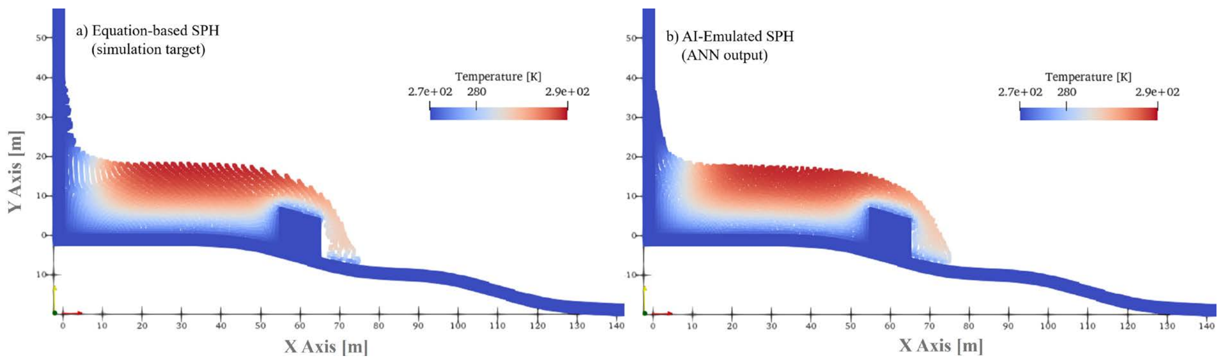

Figure 9. Simulation (a) and emulation (b) of a visco-thermal fluid over a topography with bumps and an obstacle, at time $t = 70.0$ s. The model used was trained over an SPH viscous simulation with a flat topography domain and generalized to this case study. The particles are colored by temperature [K].

3.2 Speed-up of the simulation times

In (Amato, 2024), we demonstrated that the AI-CFD emulator significantly speeds up the estimation of accelerations for inviscid fluids, by replacing numerical equations with a fitting process and using a look-up table. This substitution effectively doubles the acceleration estimations speed (i.e., the total execution time of the emulated block is halved). Furthermore, for inviscid scenarios, the emulator can be trained to use longer time steps compared to those used in reference simulations, thereby reducing the computational load over time. Specifically, by giving the features at time t as input to the network, and training it to predict outputs at a longer time step ($t + k\Delta t$, $k \in \mathbb{N}$), the ANN can be trained to predict accelerations k time steps ahead of the reference (i.e., prediction at time $t + k\Delta t$). It is possible to reach and implement this larger step due to the implementation of the two-step predictor-corrector integration scheme. Note that the continuity equation is integrated with the time step of the original numerical model, using the velocity updated by the latest available acceleration. The mathematical formulation of this concept is the following (Eq. 16):

$$a_{ij}^{t+k\Delta t} = ANN\left(x_{ij}^{t},\ u_{n_{ij}}^{t},\ \rho_i^t\ \cdot \rho_j^t\right). \tag{16}$$

Using a different integration scheme (i.e., hindsight), we applied this concept to inviscid simulations, detailed in Amato et al. (2024a), Amato (2024), achieving stable results up to $k = 5$. This approach mimics a backward integration, enhancing stability and significantly accelerating simulation runtimes. Combining both speed-up factors, we achieved an order of magnitude improvements in the speed of the emulated block for inviscid fluids.

The doubled speed-up observed in the inviscid case is also achievable for viscous and thermal fluid applications discussed here. Additionally, efforts are underway to extend the second speed-up approach, which involves larger time steps, to visco-thermal fluids, trying to further improve emulation speed.

Overall simulation speed-up depends on the proportion of computational load handled by the emulated part relative to the entire model and on device-specific technical characteristics.

4. Discussion

We applied this emulator across various types of complex fluids, each representative of different natural phenomena. In this context, validation tests evaluate the emulator's ability to replicate the numerical model behavior. We compare the AI-based model's results with those of the already validated numerical one, which we take as the reference. These verification studies confirmed the emulator ability to faithfully replicate traditional CFD simulations and its capability to generalize across different test cases.

Previously (Amato et al., 2024a), we demonstrated the emulator's efficiency in replicating inviscid and isothermal fluids. We illustrated its potential to accurately reproduce these fluids, to generalize across problems of different complexity, and to remain robust under changes in spatial resolution. Expanding the scope of application to a broader class of natural fluid, here we have studied the emulator's applicability to visco-thermal fluids. Progressing step-by-step, we initially examined the performance of the emulator with temperature-independent viscous fluids. Subsequently, we incorporated a thermal model to develop an emulator capable of reproducing complex visco-thermal fluids behavior. Applications of these models can be found in the work of Amato (2025b); instead, a purely numerical treatment of simulations of visco-thermal fluids in Amato (2025a).

Figures 1 and 5 illustrate the comparison between simulations and emulation, demonstrating agreement and showing the ability of the emulator to faithfully reproduce the behavior observed in the reference numerical model for viscous and thermal fluids. Figures 4, 8, and 9 further showcase the emulator's robustness in generalizing its applicability beyond the training scenarios, effectively emulating problems not encountered during the training phase. It is noteworthy that the emulator performs well when variables fall within the ranges of physics presented during training, ensuring physically meaningful results (Amato et al., 2024a). In these cases, variations in topography and the presence of obstacles changed the emulation setting (while maintaining the same physical parameters), still obtaining good matches between simulations and emulations, confirming the model generalizability. Thus, the emulator successfully replicates temperature-independent and temperature-dependent viscous behaviors, while preserving physics accuracy, reducing simulation times, and enhancing modeling capabilities.

Notably, the network estimates the magnitude of the acceleration vector, which is subsequently scaled by the unit vector of the relative position. In contrast, Eq. (6) expresses acceleration components parallel to the relative position and velocity vectors. Despite the emulator's output depends solely on relative position, it successfully replicates fluid behavior even in viscous scenarios. Figures 4 and 8 (confirmed also by Figs. 6 and 7) show a little overestimation of the fluid extension, and this may reflect the model's difficulty in capturing the full physics of the system, owing to its initial structural assumptions. Despite these limitations, the emulator well reproduces the overall fluid behavior across the tested scenarios. This effectiveness may stem from the dominant influence of the position component or an implicit method of merging both outputs into a unified estimation.

To evaluate the overestimation error of the models, we thus consider these two cases under the same conditions, so that they can be compared. Specifically, we compare results shown in Figs. 4 and 8, which analyze a case of generalization of the models (trained on flat topography and then tested on bumpy topography). The first result is related to the temperature-independent viscous case, the second to the temperature-dependent viscous case. In the first case, the overestimation is 6.6%, in the second case it is 3.4%. We can consider these results from different points of view.

Firstly, looking for example at Fig. 6 (for the case with a flat topography), it is also possible to notice that as the observed time instant varies, the difference between the emulated and simulated fronts varies slightly. These error values here found may therefore also depend on the fluid's emplacement dynamics, which in turn depend on various factors such as topography (e.g., bump height) or the physical parameters used.

Mathematically speaking, we can see that the relative error between the two cases is nearly halved. This may be a consequence of the physical formulation used. In the first case, constant viscosity leads to a "homogeneous" fluid dynamic, while in the second case, as the temperature decreases, the viscosity increases, preventing the fluid from "flowing" for too long a distance. Furthermore, the fact that the emulator's output depends solely on relative position may influence this aspect. In particular, due to this simplified representation of the parameters, the emulator fails to capture (and model) the pure shear contribution of viscous friction forces. Particularly referring to Fig. 8, as we have already seen, this suggests that the overestimation is not linked to a thermal-rheological miscalculation or misunderstanding, but rather to the emulator's inherent difficulty in modeling the pure shear friction forces against the bumps, which would otherwise provide the necessary dissipative action to halt the flow earlier. For this reason, the fluid may extend further.

In addition, comparing our results with similar works, in which similar studies have been conducted with different techniques, our result is coherent with other cases. The work of Sanchez-Gonzalez et al. (2020) shows the problems linked to cumulative errors during rollout (multi-step prediction). In another work, as Choi and Kumar (2024), the model reaches a runout prediction with a maximum error between around 5% of the reference runout estimation method. Thus, our error values remain below or comparable to those obtained by them with different approaches, also using a simpler ML architecture (Alexiadis, 2023), respect to more complex approaches, such as Graph neural network-based simulator (GNS), used by Choi and Kumar (2024) to model granular flow behavior. Always as in this last work, we also here show the distance of the flow front over time (in Figs. 2 and 6), with similar trends. Comparing our work with those reported here, possible causes of data error could be a numerical error propagation of the model over time, especially for long-term simulations, and specific settings of physical parameters.

Regarding the physical consistency of our results, our models are characterized by a step-by-step increase in physical significance. A model with a more robust description of dynamics (like the visco-thermal one here presented) provides a more representative physics to the AI model. This allows the emulator to better learn the physical laws, and therefore to better replicate the system dynamics, also in a case not seen during training phase (thus reducing the error).

Moving to a more context-specific perspective, this fluid could be for example approximated as a representation of a lava field. Volcanologically speaking, when the mean discharge rate is also considered (not considered here, but studied in detail in Amato, 2025a), it is possible to relate the length of the front to this parameter using an empirical model. In particular, the maximum distance of a lava flow can be written as $L = 3.11\ E^{0.47}$, with L the final length of the single channel-fed flow [m] and E the mean discharge rate [$m^3\ s^{-1}$] averaged for the entire time of an individual active flow (Calvari and Pinkerton, 1998). This law was obtained using data related only to the Mt. Etna volcano (Sicily, Italy). Typically, in cases of mean discharge rate estimations, a range of $\pm 50\%$ is considered for error estimation (Harris et al., 2007). So, for example, considering a hypothetical value of 10 $m^3\ s^{-1}$ for the mean discharge rate, we would obtain a maximum distance value equal to $L = 9.18$ m. Therefore, the range of variability of L would be set equal to [6.63, 11.11] m (corresponding to the range of $\pm 50\%$ for the mean discharge rate). Looking at the first extreme of the variability range, the final length estimated covers 72.2% of the maximum distance (underestimation of 27.8%), while, looking at the second extreme, it covers 121.0% of the maximum distance obtained by the formula without error estimation (overestimation of 21.0%). The error values obtained here (6.6% and 3.4% of overestimation) are significantly lower than the type of estimates that can typically be obtained.

Looking at the context of monitoring active volcanic areas, the models presented here overestimate the simulation by 9 m over approximately 137 m and by approximately 3 m over 83 m, respectively. However, the model offers a gain in obtaining results (in computational costs for the emulator), enhancing its functionalities. For short-term monitoring cases, this is therefore an excellent compromise. Finally, in both cases, the emulator returned an overestimation, not an underestimation, of the front distance compared to the one given by the simulator. In a context like this, an overestimation is a more "conservative" error (it is better to evacuate a larger area near to a volcano than a smaller one). Although an error of 6.6% could lead to excessive false alarms, the value of 3.4% for the more physically complete model indicates an optimization that maintains safety while increasing model precision.

We are already working on overcoming these limits, improving the models' performance. We are currently better investigating all these aspects here described, in particular introducing a detailed sensitivity analysis on the parameter settings used (Amato, 2025a) and implementing an emulator for the inviscid case that considers the estimation of the two acceleration components, parallel to the relative position and to the velocity vectors (Zago et al., 2025). These studies will then be integrated also into the visco-thermal fluid emulators, to overcome these possible limits, reducing the error estimations and increasing the accuracy of the outcomes.

5. Conclusion

We have introduced an emulator designed to enhance visco-thermal fluid simulations, by integrating CFD models with AI techniques. While traditional CFD numerical models provide high-level results with strong physical reliability, their computational costs, including high resources requirements and long simulation times, present significant challenges for practical applications. The aims to overcome these limitations and delve deeper into physical phenomena have motivated us to develop this emulator, which incorporates the use of AI, in particular ANN, into SPH method to improve CFD simulation performance.

This emulator is able to well-reproduce the physical characteristics of a fluid and its spatio-temporal evolution, reducing the computational costs and enhancing its functionalities. Here, we have trained the emulator firstly over a temperature-independent viscous fluid, subsequently over a visco-thermal case, obtaining good reproduction of the considered fluids, and a good ability to generalize for cases not seen during the training phase. While the current emulator shows a slight overestimation of the fluid extension, the model well reproduces the overall fluid behavior in space and time for all the tested scenarios, and future refinements in visco-thermal coupling will improve these results.

In particular, future developments of the work will focus on refining the emulator to estimate the two acceleration components (parallel to position and velocity, as in Eq. 6) independently. Improvements can also target the integration of additional physical processes (after a detailed sensitivity analysis of the parameters that can influence the models' outcomes), to reduce the small inconsistencies observed in some of the results presented here.

Future advancements will incorporate spatially and temporally distant input data into the network, enabling better tracking of ongoing fluid dynamics and further improving simulation stability with larger time steps.

Efforts are underway to transition all codes to GPU, for enhanced efficiency, and extend capability of the emulator to 3D applications. This expansion includes all the integrations seen here for viscous and thermal terms in both simulation and emulation phases, ensuring accurate simulations with reduced computational overhead.

In addition, further enhancements and speed-up could be obtained by combining AI advantages with the new frontiers of Quantum Computing (Steane, 1998), to potentiate the simulations, exploiting the various levels of parallelization that such technologies can offer, and the advantages of intrinsic properties of quantum mechanics such as superposition and entanglement.

Finally, this approach opens to improvements in studies in the field of numerical simulations, thus it could extend its applicability to geophysical flows of different nature, such as lava flows, ash clouds, tephra fallout or gases dispersing into the atmosphere (Amadio et al., 2024; Cappello et al., 2011; Corradino et al., 2024; Del Negro et al., 2016, 2020; Macedonio et al., 2005), with impacts in environment, humans and climate changes (Mauro et al., 2024; Sebastianelli et al., 2023, 2025; Van Der Meer et al., 2023; Watt-Meyer et al., 2023). The combination of analytical and numerical models with enhanced technologies, such as Artificial Intelligence and Quantum Computing, can produce models able to replicate physical dynamics. These hybrid approaches open to direct applications, for example as reliable reference tools for emergency response and as digital Twins of Earth's physical systems.

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

Data availability statement. The dataset used in this work is freely available upon request to the authors.

Acknowledgements. 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). We thank Stephen Conway for revising the English of the manuscript.

Funding Information. This research was supported by the DEMETRA research line within the ROSE (Reinforcement of the Observational Systems of the Earth) infrastructural project of INGV (OB.FU.: 1215.010), funded by the Italian Ministry of University and Research, and by the Project INGV Pianeta Dinamico VT_ORME 2023-2025 (INGV OB.FU. 1020.010).

Conflict of interest. Authors state no conflict of interest.

References

Alexiadis, A. (2023). A minimalistic approach to physics-informed machine learning using neighbour lists as physics-optimized convolutions for inverse problems involving particle systems, J. Comput. Phys., 473, 111750.

Amadio, F., L. Pioli and S. Scollo (2024). Constraining proximal grainsize distribution of tephra from paroxysmal eruptions at etna volcano, J. Volcanol. Geother. Res., 454, 108164.

Amato, E., C. Corradino, F. Torrisi and C. Del Negro (2021). 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) 1-6 p., IEEE.

Amato, E. (2022). Machine learning and best fit approach to map lava flows from space. Il Nuovo Cimento C, 45, 1-12.

Amato, E. (2023). How a CFD emulator can resolve the boundary conditions in a viscous flow. IEICE Proceedings Series, 76(B4L-42), 383.

Amato, E., C. Corradino, F. Torrisi and C. Del Negro (2023a). A deep convolutional neural network for detecting volcanic thermal anomalies from satellite images, Remote Sensing, 15, 15, 3718.

Amato, E., C. Corradino, F. Torrisi and C. Del Negro (2023b). Spectral analysis of lava flows: Temporal and physicochemical effects, Il Nuovo Cimento C, 46, 1-4.

Amato, E., V. Zago and C. Del Negro (2024a). A physically consistent AI-based SPH emulator for computational fluid dynamics, Nonlin. Engin., 13, 1, 20220359.

Amato, E., V. Zago and C. Del Negro (2024b). Generalizability of AI-based emulators for CFD lagrangian methods. Copernicus Meetings, Sci Profiles, doi:10.5194/egusphere-egu24-563.

Amato, E. (2024). Enhancing computational fluid dynamics with artificial intelligence: An AI-based smoothed particle hydrodynamics (SPH) emulator for lava flow modeling, PHD Thesis.

Amato, E. (2025a). A study on the effects of effusion rate on 2D numerical simulations of lava flow evolution, Annals of Geophysics, 68, 2, doi:10.4401/ag-9188.

Amato, E. (2025b). An AI-powered CFD emulator for lava flow modeling, Il Nuovo Cimento C, 48, 1-8.

Anderson, J. D. and J. Wendt (1995). Computational fluid dynamics, Springer, 206, 344 p.

Antuono, M., A. Colagrossi and S. Marrone (2012). Numerical diffusive terms in weakly-compressible SPH schemes, Compu. Phys. Commu., 183, 12, 2570-2580.

Batchelor, G. K. (1967). An introduction to fluid dynamics. Cambridge university press, 634 p.

Bilotta, G., A. Hérault, A. Cappello, G. Ganci and C. Del Negro (2016). GPUSPH: A smoothed particle hydrodynamics model for the thermal and rheological evolution of lava flows. Geological Society, London, Special Publications, 426, 1, 387-408.

Bilotta, G., V. Zago, V. Centorrino, R. A. Dalrymple et al. (2022a). A numerically robust, parallel-friendly variant of BiCGSTAB for the semi-implicit integration of the viscous term in smoothed particle hydrodynamics, J. Comput. Phys., 466, 111413.

Bilotta, G., V. Zago, A. Hérault, H. D. van Ettinger and R. A. Dalrymple (2022b). Fast, feature-rich weakly-compressible SPH on GPU: Coding strategies and compiler choices, arXiv Preprint arXiv:2207.11328.

Bilotta, G., V. Zago, A. Hérault, A. Cappello, G. Ganci et al. (2024). Optimization of flexible neighbors lists in smoothed particle hydrodynamics on GPU, Adv. Engin. Softw., 196, 103711.

Bonaccorso, G. (2017). Machine learning algorithms. Packt Publishing Ltd, 360 p.

Bortnik, J. and E. Camporeale (2021). Ten ways to apply machine learning in the earth and space sciences. In AGU fall meeting abstract, IN12A-06, New Orleans.

Cai, S., Z. Mao, Z. Wang, M. Yin and G. E. Karniadakis (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mechanica Sinica, 37, 12, 1727-1738.

Calvari, S. and H. Pinkerton (1998). Formation of lava tubes and extensive flow field during the 1991-1993 eruption of Mount Etna, J. Geophys. Res.: Solid Earth, 103, B11, 27291-27301.

Cappello, A., A. Vicari and C. Del Negro (2011). Assessment and modeling of lava flow hazard on mt. Etna volcano, Bollettino Di Geofisica Teorica e Applicata, 2, 52, 299-308, doi:10.4430/bgta0003.

Cariello, S., C. Corradino, F. Torrisi and C. Del Negro (2023). Cascading machine learning to monitor volcanic thermal activity using orbital infrared data: From detection to quantitative evaluation, Remote Sensing, 16, 1, 171.

Choi, Y. and K. Kumar, (2024). Graph neural network-based surrogate model for granular flows. Computers and Geotechnics, 166, 106015.

Chung, T. J. (2002). Computational fluid dynamics. Cambridge University Press, 1058 p.

Clevert, D.-A., T. Unterthiner and S. Hochreiter (2015). Fast and accurate deep network learning by exponential linear units (elus), arXiv Preprint arXiv:1511.07289.

Cole, R. H. (1948). Underwater explosion. Princeton Univ. Press, 437 p.

Corradino, C., E. Amato, F. Torrisi and C. Del Negro (2022). Data-driven random forest models for detecting volcanic hot spots in sentinel-2 MSI images, Remote Sensing, 14, 17, 4370.

Corradino, C., P. Jouve, A. La Spina and C. Del Negro (2024). Monitoring earth's atmosphere with sentinel-5 TROPOMI and artificial intelligence: Quantifying volcanic SO_2 emissions, Remote Sensing Environ., 315, 114463.

Corradino, C., A. B. Malaguti, M. S. Ramsey and C. Del Negro (2024). Quantitative assessment of volcanic thermal activity from space using an isolation forest machine learning algorithm, Remote Sensing, 16, 11, 2001.

Del Negro, C., A. Cappello and G. Ganci (2016). Quantifying lava flow hazards in response to effusive eruption, GSA Bulletin, 128, 5-6, 752-763.

Del Negro, C., A. Cappello, G. Bilotta, G. Ganci et al. (2020). Living at the edge of an active volcano: Risk from lava flows on Mt. Etna, GSA Bulletin, 132, 7-8, 1615-1625.

Gao, L., M. Pastor, T. Li, S. Moussavi Tayyebi et al. (2023). A framework coupled neural networks and SPH depth integrated model for landslide propagation warning, Acta Geotechnica, 18, 7, 3863-3888.

Gingold, R. A. and J. J. Monaghan (1977). Smoothed particle hydrodynamics: Theory and application to non-spherical stars, Month. Not. Royal Astron. Soc., 181, 3, 375-389.

Goodfellow, I., Y. Bengio and A. Courville (2016). Deep learning. MIT press, 800 p.

Groeneveld, D., J. Shan, A. Vasić, L. Leung et al. (2007). The 2006 CHF look-up table, Nuclear Engin. Des., 237, 15-17, 1909-1922.

Harris, A. J., J. Dehn and S. Calvary (2007). Lava effusion rate definition and measurement: a review, Bull. Volcanol., 70, 1, 1-22.

Higaki, T., Y. Tanabe, H. Hashimoto and T. Iida (2025). Step-by-step enhancement of a graph neural network-based surrogate model for lagrangian fluid simulations with flexible time step sizes, Appl. Ocean Res., 154, 104424.

Hornik, K., M. Stinchcombe and H. White (1989). Multilayer feedforward networks are universal approximators, Neural Net., 2, 5, 359-366.

Kasim, M. F., D. Watson-Parris, L. Deaconu, S. Oliver et al. (2021). Building high accuracy emulators for scientific simulations with deep neural architecture search, Machine Learning: Sci. Technol., 3, 1, 015013.

Kochkov, D., J. A. Smith, A. Alieva, Q. Wang et al. (2021). Machine learning-accelerated computational fluid dynamics in Proceedings of the National Academy of Sciences, 118, 21, e2101784118.

Ladickỳ, L., S. Jeong, B. Solenthaler, M. Pollefeys and M. Gross (2015). Data-driven fluid simulations using regression forests, ACM Trans. Graphics, 34, 6, 1-9.

Li, Z. and A. B. Farimani (2022). Graph neural network-accelerated lagrangian fluid simulation, Compu. Graph., 103, 201-211.

Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis, Astronon. J., 82, 12, 1013-1024

Macedonio, G., A. Costa and A. Longo (2005). A computer model for volcanic ash fallout and assessment of subsequent hazard, Compu. Geosci., 31, 7, 837-845.

Macia Lang, F., A. Souto Iglesias, M. Antuono and A. Colagrossi (2011). Benefits of using a wendland kernel for free-surface flows in 6th international SPHERIC workshop, Hamburg, Germany, https://files01.core.ac.uk/download/pdf/148662105.pdf.

Mauro, F., A. Sebastianelli, B. Le Saux, P. Gamba and S. L. Ullo (2024). A hybrid MLP-quantum approach in graph convolutional neural networks for oceanic niño index (ONI) prediction, In IGARSS 2024-2024 IEEE international geoscience and remote sensing symposium, 812-816, IEEE.

Molteni, D. and A. Colagrossi (2009). A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH, Compu. Phys. Commu., 180, 6, 861-872.

Monaghan, J. J. (1992). Smoothed particle hydrodynamics, Annual Rev. Astron. Astrophys., 30, 1, 543-574.

Monaghan, J. J. (2005). Smoothed particle hydrodynamics. Rep.Progress Phys., 68, 8, 1703.

Pfaff, T., M. Fortunato, A. Sanchez-Gonzalez and P. W. Battaglia (2020). Learning mesh-based simulation with graph networks, arXiv Preprint arXiv:2010.03409.

Ramsey, M. S., J. O. Thompson and C. Corradino (2023). Surface biology and geology (SBG) observing terrestrial thermal emission radiometer (OTTER), NASA, D-1000792, https://sbg.jpl.nasa.gov/downloads/documents/SBG-TIR_PSD_L3_ETF_20231107.pdf.

Saikali, E., G. Bilotta, A. Hérault and V. Zago (2020). Accuracy improvements for single precision implementations of the SPH method. Int. J. Comput. Fluid Dyn., 34, 10, 774-787.

Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying et al. (2020). Learning to simulate complex physics with graph networks. In International conference on machine learning, 8459-8468, PMLR.

Sebastianelli, A., M. P. Del Rosso, S. L. Ullo and P. Gamba (2023). On quantum hyperparameters selection in hybrid classifiers for earth observation data, IEEE Geosci Remote Sensing Lett., 20, 1-5.

Sebastianelli, A., F. Mauro, G. Ciabatti, D. Spiller, B. Le Saux et al. (2025). Quanv4eo: Empowering earth observation by means of quanvolutional neural networks, IEEE Trans.Geosci. Remote Sensing, 63, doi:10.1109/TGRS.2025.3556335.

Sofos, F., C. Stavrogiannis, K. K. Exarchou-Kouveli, D. Akabua et al. (2022). Current trends in fluid research in the era of artificial intelligence: A review, Fluids, 7, 3, 116.

Steane, A. (1998). Quantum computing. Reports on Progress in Physics, 61, 2, 117.

Torricelli, E. (1644). Opera geometrica evangelistae torricellii. Florentiae: Masse; de Landis.

Torrisi, F., E. Amato, C. Corradino, S. Mangiagli and C. Del Negro (2022a). Characterization of volcanic cloud components using machine learning techniques and SEVIRI infrared images, Sensors, 22, 20, 7712.

Torrisi, F., E. Amato, C. Corradino and C. Del Negro (2022b). The FastVRP automatic platform for the thermal monitoring of volcanic activity using VIIRS and SLSTR sensors: FastFRP to monitor volcanic radiative power, Ann. Geophys., 65, 6, doi:10.4401/ag-8823.

Torrisi, F., C. Corradino, S. Cariello and C. Del Negro (2024). Enhancing detection of volcanic ash clouds from space with convolutional neural networks, J. Volcanol. Geoth. Res., 448, 108046.

Toshev, A. P., J. A. Erbesdobler, N. A. Adams and J. Brandstetter (2024). Neural SPH: Improved neural modeling of lagrangian fluid dynamics, arXiv Preprint arXiv:2402.06275.

Ummenhofer, B., L. Prantl, N. Thuerey and V. Koltun (2019). Lagrangian fluid simulation with continuous convolutions, In International conference on learning representations. New Orleans.

Van Der Meer, M., S. de Roda Husman and S. Lhermitte. (2023). Deep learning regional climate model emulators: A comparison of two downscaling training frameworks, J. Ad. Mod. Earth Syst., 15, 6, e2022MS003593.

Vinuesa, R. and S. L. Brunton (2022). Enhancing computational fluid dynamics with machine learning. Nature Comput. Sci., 2, 6, 358-366.

Watt-Meyer, O., G. Dresdner, J. McGibbon, S. Clark et al. (2023). ACE: A fast, skillful learned global atmospheric model for climate prediction, arXiv preprint arXiv:2310.02074.

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Computat. Math., 4, 389-396.

Yan, B., D. R. Harp, B. Chen and R. Pawar (2022). A physics-constrained deep learning model for simulating multiphase flow in 3D heterogeneous porous media, Fuel, 313, 122693.

Zago, V., G. Bilotta, A. Cappello, R. A. Dalrympl et al. (2017). Simulating complex fluids with smoothed particle hydrodynamics. Ann. Geophys., 60, 6 suppl., doi:10.4401/ag-7362.

Zago, V., G. Bilotta, A. Hérault, R. A. Dalrymple et al. (2018). Semi-implicit 3D SPH on GPU for lava flows, J. Computational Phys., 375, 854-870.

Zago, V., G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, et al. (2019). Preliminary validation of lava benchmark tests on the GPUSPH particle engine. Ann. Geophys., 62, 2, doi:10.4401/ag-7870.

Zago, V., L. J. Schulze, G. Bilotta, N. Almashan and R. A. Dalrymple (2021). Overcoming excessive numerical dissipation in SPH modeling of water waves, Coastal Engin., 170, 104018.

Zago, V., R. A. Dalrymple, N. Almashan, G. Bilotta et al. (2023). Characterization and modeling of greenwater overtopping of a sea-level deck, Ocean Engin., 275, 114131.

Zago, V., E. Amato and C. Del Negro (2025). A Lagrangian AI-based CFD emulator with reconstruction of hydrodynamic forces components, submitted to Computers and Fluids.

Zhang, N., S. Yan, Q. Ma, X. Guo et al. (2023). A CNN-supported lagrangian ISPH model for free surface flow, Appl. Ocean Res., 136, 103587.

Zhang, N., S. Yan, Q. Ma and Q. Li (2024). A hybrid method combining ISPH with graph neural network for simulating free-surface flows, Compu. Phys. Commu., 301, 109220.

***CORRESPONDING AUTHOR: Eleonora AMATO**,
Istituto Nazionale di Geofisica e Vulcanologia, Sezione Osservatorio Etneo, Catania, Italy
e-mail: eleonora.amato@ingv.it