A General Neural Particle Method for Hydrodynamics Modeling

Jinshuai Bai[a], Ying Zhou[a,b], Yuwei Ma[c], Hyogu Jeong[a], Haifei Zhan[d,a], Charith Rathnayaka[a,e], Emilie Sauret[a], Yuantong Gu[a,*]

[a]School of Mechanical, Medical and Process Engineering, Queensland University of Technology, Brisbane, QLD 4000, Australia

[b]State IJR Center of Aerospace Design and Additive Manufacturing, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

[c]State Key Laboratory for Turbulence and Complex System, Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China

[d]Department of Civil Engineering, Zhejiang University, Zhejiang, Hangzhou 310058, China

[e]School of Science, Technology and Engineering, University of the Sunshine Coast, Petrie, QLD 4502, Australia

* Corresponding authors.

E-mail addresses: yuantong.gu@qut.edu.au (Y. Gu)

Abstract

Neural Particle Method (NPM) is a newly proposed Physics-Informed Neural Network (PINN) based, truly meshfree method for hydrodynamics modeling. In the NPM, PINN is applied to globally approximate field variables, and the high-order Implicit Runge-Kutta (IRK) method is used to treat the time integration. The NPM can easily achieve incompressibility and deal with the free-surface problem. However, the NPM is currently limited to inviscid hydrodynamics problems and is computationally expensive. In this work, we developed a general NPM (gNPM) for viscous hydrodynamics modeling. In the gNPM, a single pressure is output as the predicted pressure field rather than the multiple pressures in the original NPM. Thus, the size of the neural network is greatly reduced, making the gNPM computationally more efficient. Besides, the spatial derivatives in the governing equations are calculated with respect to the current spatial coordinates rather than the predicted space. In this manner, the gNPM is more straightforward to be implemented. Furthermore, by considering the viscous term in the conservation of momentum, the gNPM can be applied for viscous hydrodynamics modeling. The effectiveness and robustness of the gNPM have been demonstrated through several hydrodynamics benchmark cases with different boundary conditions. We highlight that

the proposed gNPM is able to cope with highly uneven particle distributions, while the traditional meshfree method can produce severe failure.

Keywords: Neural Particle Method; Physics-informed neural network; Machine learning; Deep learning; Hydrodynamics modeling, Meshfree method

1. Introduction

Computational Fluid Dynamics (CFD) is an essential way to study hydrodynamics problems [1]. In CFD, numerical and computational methods are applied to deal with the governing equations of fluid dynamics and boundary conditions [2]. These numerical methods can provide favorable approximate solutions to complex hydrodynamics problems, which are challenging to be solved analytically [3].

In the past few decades, great progress has been witnessed in the CFD field, and various CFD methods have been proposed [4]. In the early methods, meshes or grids were applied to discretize the spatial domain, and the terms in governing equations were approximated through connected mesh nodes [5]. Typical examples of these so-called mesh-based methods include the Finite Difference Method (FDM) [6], Finite Volume Method (FVM) [7], and Finite Element Method (FEM) [8]. To date, mesh-based methods remain the most prevailing type of numerical techniques for hydrodynamics modeling. However, mesh-based methods suffer significant difficulties, including the high cost of generating the mesh for complex geometries, mesh distortion in large deformation problems, and boundary tracking in free-surface problems [5]. Hence, to avoid these difficulties, a class of new methods, namely the meshfree methods, were proposed. The meshfree methods use nodes or particles in replacement of meshes. Compared to mesh-based methods, meshfree methods offer flexibility for hydrodynamics modeling, which can effectively alleviate the difficulties of the aforementioned mesh-based methods [2]. Thus, meshfree methods have attracted more attention in recent decades [1]. The typical meshfree methods for hydrodynamics modeling include Smoothed Particle Hydrodynamics (SPH) [9-11], Moving Particle Semi-implicit (MPS) [12] method, and Meshless Local Petrov-Galerkin (MLPG) [13].

Deep learning, an important branch of the Machine Learning method, is based on the multi-layer artificial neural network, which is a biomimetic computing system inspired by biological neural networks [14]. Deep learning automatically extracts the inherent features and

representations from raw data [15]. Deep learning techniques have successfully been applied in various areas [16], including bioinformatics [17], image recognition [18], and financial prediction [19].

Deep learning techniques have attracted attention in hydrodynamics modeling as well [20-22]. It has opened a new avenue for solving hydrodynamics problems [20]. Works have been published to apply various deep learning techniques in predicting hydrodynamics problems with massive observation data, which can be classified as the pure data-driven method [23-27]. However, pure data-driven deep learning methods for hydrodynamics modeling still have some shortcomings. Firstly, the pure data-driven deep learning hydrodynamics methods can be regarded as interpolation methods [32]. Deep learning techniques are trained to fit the given data and then used for predictions. When applications of the trained neural networks fall out of the span of the given data, the results from the neural network may not be reliable anymore. To alleviate this issue, a tremendously large, completed, and labeled database is required to cover the possible problem domain. However, it is currently difficult or even impossible for the hydrodynamics field to obtain such a well-developed database [20]. Secondly, the accuracy of pure data-driven deep learning techniques for hydrodynamic modeling is not ideal enough [26-28]. Accordingly, those applications are currently confined to the areas of animations and game engines [26, 27, 29, 30].

Recently, Raissi et al. [31-33] introduced the Physics-Informed Neural Networks (PINNs), which can integrate physics information to train neural networks. With embedded physics information, the neural network can achieve favorable accuracy with less training data. Besides, PINNs also contribute to the inverse problem, such as extracting parameters from observations [34, 35]. Up to now, PINNs have attracted increased attention and great efforts have been made to further improve the performance of PINNs framework [36, 37]. Wang et al. [38] studied the convergence of PINNs and leveraged the neural tangent kernel algorithm to alleviate the scale differences in the loss function. Liu and Wang [39] introduced an adaptive learning algorithm for PINNs to normalize the loss terms. Nguyen-Thanh et al. [40] proposed a parametric training approach that improves the performance of PINNs for complex geometry domains. Sukumar and Srivastava [41] proposed the distance functions to exactly impose the boundary conditions. Kharazmi et al. [42] tailored a decomposition approach for PINNs to deal with variational problems. Shukla et al. [43] further incorporated parallel computing to improve the computational efficiency of PINNs for decomposed domains. Haghighat et al. [44] proposed a non-local scheme to greatly improve the performance of PINNs with respect to sharp gradient

problems. In addition, various kinds of neural networks have been applied to explore further possibilities of the PINN [45, 46]. Currently, PINNs have become one of the most popular deep learning techniques in certain areas of research and have already made significant contributions [47-49].

PINNs have also been applied to solve direct hydrodynamics problems as an alternative to traditional CFD approaches [21, 47, 50]. In most PINN-based hydrodynamics modeling methods, the spatiotemporal coordinates are equally treated as the inputs of neural networks. Jin et al. [51] introduced the Navier-Stokes Flow nets (NSFnets) for incompressible hydrodynamics modeling. In their work, PINN is applied to solve two formulations of the Navier-Stokes equations, i.e., the velocity-pressure and vorticity-velocity formulations. The convergence and accuracy of the NSFnets have been discussed, and it has been employed for numerical examples in terms of laminar and turbulent flows to prove its effectiveness. Xiang et al. [52] proposed a self-adaptive loss balanced Physics-Informed Neural Network (lbPINN) for incompressible Navier-Stokes equations. By adding weights in front of loss terms, the lbPINN greatly improves the accuracy of PINN in solving incompressible hydrodynamics problems.

Another attempt of using PINNs to the direct hydrodynamics problems separately treats the spatial and temporal coordinates through the Lagrangian point of view. Wessels et al. [53] have explored the applicability of PINNs for inviscid hydrodynamics modeling and proposed Neural Particle Method (NPM). In their method, neural networks have been applied to approximate the velocity and pressure in the spatial fields, while the Implicit Runge-Kutta (IRK) method has been used to achieve temporal discretization. Particles are used to represent the fluid substance, and the positions of the particles are updated after each timestep. In this manner, the NPM is able to deal with free surface hydrodynamics problems while achieving incompressibility. Furthermore, numerical techniques in terms of interpolation [54] and penalty approach are introduced to strictly fulfill the Dirichlet boundary and solid rigid wall, respectively. Several free-surface examples have been conducted to test the conservation properties of the NPM, and close agreements have been observed with respect to experimental data. Accordingly, the NPM opens up a new avenue for hydrodynamics modeling while possessing favorable potential to become an even more powerful tool in this context [53].

Nevertheless, the applications of the NPM are currently confined to inviscid problems. Only limited results and discussion of the NPM are provided. Besides, in the NPM, multiple pressures are output at each timestep to fit the IRK method. We notice that no time-dependent

evolution equation is given for the pressure field. Therefore, the predicted pressure is updated through the weighted average of the multiple pressure outputs rather than using the IRK method, resulting in redundant pressure information and increasing the computational expense. Furthermore, the spatial derivatives in the NPM are obtained with respect to the predicted configuration. A linear push-forward operation is required to transfer the derivatives from the current space to the predicted space. The application of the push-forward operator results in additional calculations and brings difficulties in taking the viscous force into account, which includes a second-order spatial derivative. Accordingly, in this paper, we develop a general Neural Particle Method (gNPM) for viscous hydrodynamics modeling. In the gNPM, a single pressure is output as the predicted pressure field. In this manner, the size of the neural network is significantly reduced, and the gNPM becomes computationally more efficient. All the spatial derivatives are obtained regarding the current space, making the gNPM more straightforward to be implemented than the original NPM. In addition, by considering the viscous term in the conservation of momentum, the gNPM can be applied for viscous hydrodynamics modeling. We study the convergence of the gNPM and test its performance towards static, moving, periodic, and free surface boundary conditions. Moreover, we demonstrate that the gNPM brings more flexibility in spatial discretization to the CFD field. Comparing with the SPH method, a typical meshfree particle method, we find that the gNPM is generally more robust for hydrodynamics modeling than traditional meshfree methods.

We organize this paper as follows. In Section 2, the fundamentals of hydrodynamics in terms of the governing equations and the boundary conditions are given, respectively. In Section 3, we elucidate the methodology of the gNPM and discuss its numerical implementation. In Section 4, numerical cases with different boundaries are conducted to assess the convergence and robustness of the gNPM. Finally, conclusions and future perspectives are summarized in Section 5.

2. Physics laws of hydrodynamics problems

2.1. Conservation laws of momentum and mass

The governing equations for hydrodynamics problems, in terms of conservation of mass and momentum, can be written in the form of Lagrangian description as

$$\frac{\partial \rho(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \left(\rho(\boldsymbol{x},t) \boldsymbol{v}(\boldsymbol{x},t) \right) = 0 \tag{1}$$

$$\boldsymbol{a}(x,t) = -\frac{1}{\rho(\boldsymbol{x},t)}\nabla p(\boldsymbol{x},t) + \frac{1}{\rho(\boldsymbol{x},t)}\nabla\cdot\boldsymbol{\tau}(\boldsymbol{x},t) + \boldsymbol{g}(\boldsymbol{x},t) \tag{2}$$

where $\boldsymbol{a}(\boldsymbol{x}, t)$ is the acceleration of the fluid. $\boldsymbol{v}(\boldsymbol{x}, t)$ and $p(\boldsymbol{x}, t)$ represent the fluid velocity and pressure, respectively. $\rho(\boldsymbol{x}, t)$ is the fluid density. $\boldsymbol{g}(\boldsymbol{x}, t)$ refers to the body force per unit mass. $\boldsymbol{\tau}(\boldsymbol{x}, t)$ is the shear stress tensor related to the rheological properties of the concerned fluid.

For incompressible Newtonian fluids, ρ remains constant and $\boldsymbol{\tau}$ can be formulated via Newton's law as $\boldsymbol{\tau} = \mu\nabla\boldsymbol{v}$ where μ denotes the fluid viscosity. Accordingly, Eq. (1) and Eq. (2) can be simplified as

$$\nabla\cdot\boldsymbol{v}(\boldsymbol{x},t) = 0 \tag{3}$$

$$\boldsymbol{a}(\boldsymbol{x},t) = -\frac{1}{\rho}\nabla p(\boldsymbol{x},t) + \frac{1}{\rho}\nabla\cdot\left(\mu\nabla\boldsymbol{v}(\boldsymbol{x},t)\right) + \boldsymbol{g}(\boldsymbol{x},t) \tag{4}$$

2.2. Initial and boundary conditions

The governing equations are closed by the boundary conditions. Here, we introduce three common boundary conditions, namely the solid boundary, the periodic boundary, and the free surface. First of all, the no-slip solid boundary condition can be written as:

$$\boldsymbol{v}_{\Gamma_s} = \boldsymbol{v}_{\mathrm{f}}, \tag{5}$$

where $\boldsymbol{v}_{\Gamma_s}$ represents the solid boundary velocity and $\boldsymbol{v}_{\mathrm{f}}$ is the fluid velocity at the solid boundary. For the periodic boundary, let us assume we have two boundaries that are connected periodically. Respectively, they are denoted as Γ_1 and Γ_2. In this manner, the periodic boundary condition can be written as:

$$\begin{cases} \boldsymbol{v}_{\Gamma_1} = \boldsymbol{v}_{\Gamma_2}, \\ p_{\Gamma_1} = p_{\Gamma_2}, \end{cases} \tag{6}$$

where $\boldsymbol{v}_{\Gamma_1}$, $\boldsymbol{v}_{\Gamma_2}$, p_{Γ_1}, p_{Γ_2} are respectively the velocity and pressure on Γ_1 and Γ_2. Finally, the free-surface boundary condition is achieved via setting the pressure on the boundary line, Γ_{fs}, to be zero, which reads

$$p_{\Gamma_{\mathrm{fs}}} = 0. \tag{7}$$

3. General Neural Particle Method (gNPM)

The basic idea of the NPM [53] is using an Feedforward Neural Network (FNN) [55] to approximate the field variables $\boldsymbol{v}$ and p at each timestep. The FNN is trained with the inviscid

governing equations and boundary conditions. Besides, particles are used to represent both the fluids and the boundaries. The positions of the particles are updated after each timestep so that the morphology of the fluid and free surface can be clearly captured. Based on the idea of the NPM, in this section, we propose the methodology of the general Neural Particle Method (gNPM) for viscous hydrodynamics modeling. It should be noted that, although the loss function of the well-trained FNN should ideally reach zero, the actual loss function is generally a small value because the FNN is trained through optimizers (such as SGD [56], Adam [57], Adagrad [58], and Adadelta [59]) to reach predefined convergence criteria. Thus, the satisfaction of governing equations and initial/boundary conditions is in the *"weak"* sense.

3.1. FNN for physical field modeling

Without loss of generality, we consider a 2D hydrodynamics problem. The velocity and pressure fields in Eqs. (4) and (3) are functions of temporal variable t and spatial coordinates (x, y). In this work, we focus on the temporal-discretized model where the time domain is discretized into timesteps t_0, $t_0+\Delta t$, $t_0+2\Delta t$, ⋯. Note that Δt is the time increment. The velocity and pressure fields at timestep t are then modeled as functions of spatial coordinates (x, y) through an FNN [60] as

$$v_x^t, v_y^t, p^t = F(x, y) \tag{8}$$

where v_x^t and v_y^t denote the x- and y-component of $\boldsymbol{v}^t$. $F(x, y)$ is the FNN as shown in Fig. 1. The FNN consists of an input layer, single or multiple hidden layers, and an output layer. Each layer has a number of neurons [61]. The number of neurons in the input layer n_{input} equals the dimension of input features (in our case $n_{\text{input}}=2$). The Fully Connected Neural Network (FCNN) is obtained if all neurons in the previous layer are fully connected to the neurons in the adjacent layer. The $(l+1)$-layer feature $\boldsymbol{h}^{l+1}$ is mathematically expressed in terms of the l-layer feature $\boldsymbol{h}^l$ as

$$\boldsymbol{h}^{l+1} = \vartheta(\boldsymbol{w}^l \cdot \boldsymbol{h}^l + \boldsymbol{b}^l) \tag{9}$$

where $\boldsymbol{w}^l$ and $\boldsymbol{b}^l$ are the weights and biases of l-layer. The activation function ϑ is a nonlinear operator which endows the FNN with the ability to model nonlinear functions. Actually, it has been demonstrated that the FNN with a single hidden layer can model arbitrarily complex functions if the number of neurons is large enough [62]. For a more detailed introduction of the FNN, readers are referred to [55]. The FNN in Fig. 1 can be obtained through sequence operations of Eq. (9)

$$F(x,y)=\mathcal{G}(\boldsymbol{w}^{n+1}\cdots\underbrace{\mathcal{G}(\boldsymbol{w}^{2}\underbrace{\mathcal{G}(\boldsymbol{w}^{1}\mathbf{x}+\boldsymbol{b}^{1})}_{\text{layer 1}}+\boldsymbol{b}^{2})}_{\text{layer 2}}+\cdots+\boldsymbol{b}^{n+1})\quad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{layer } n+1}\quad\text{(10)}$$

A distinguished advantage of using the FNN for modeling physical fields is that the partial differentials of field variables with respect to spatial coordinates can be easily calculated via automatic differentiation [63]. As is known, traditional CFD methods rely on the surrounding mesh nodes or particles to obtain the spatial differential terms. For instance, the FDM uses the Taylor expansion with connected mesh nodes to calculate the gradient terms [64], and the SPH interpolates the spatial differential terms by summing up the field variables of the surrounding particles with weights (kernel functions) [4]. Because the traditional CFD methods approximate the differential terms through the nearby nodes and particles, it unconsciously brings restrictions in terms of the mesh quality and particle distribution to the conventional CFD methods. Researchers have pointed out that, for mesh-based methods, the highly distorted or irregular meshes can greatly affect the accuracy of modeling [2]. For meshfree methods, the applications with highly uneven particle distribution can fail to satisfy the completeness condition [65]. Local particle clustering and vacancy can respectively induce local information redundancy and deficiency, resulting in large departures of the interpolated values on the particles [66]. Therefore, some additional algorithms or techniques are required to force the nodes or particles to be evenly distributed [4], e.g., the particle shifting algorithm that slightly modifies positions of the particles from the high-dense region to the low-dense region after each timestep [67, 68] and the artificial stress that introduces a force between every two particles to avoid particles clustering [69]. The neural network-based method can obtain the spatial differential terms directly from the automatic differentiation rather than nearby nodes or particles. Hence, the neural network-based method can cope with highly uneven particle distributions and is more robust than traditional CFD methods. To further elaborate this, a detailed comparison between the gNPM and a widely used meshfree method, namely the SPH method, with respect to particle distributions are provided in the Section 4.

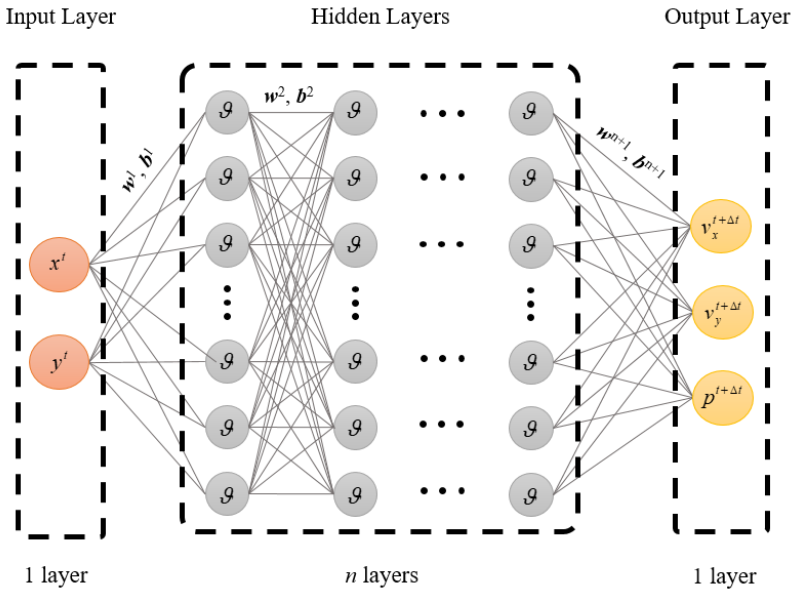

Fig. 1. An illustration of the FNN structure (with an input layer, n hidden layers, and an output layer) for modeling 2D fluid velocity and pressure fields. The spatial coordinates (x, y) are fed into the FNN as input. The FNN outputs the velocity and pressure fields as a black-box function of (x, y). $\boldsymbol{w}^l$ and $\boldsymbol{b}^l$ are the weights and biases of l-layer (l = 1, …, n+1). ϑ denotes the activation function.

3.2. Training of the FNN with physics laws

Training of the FNN aims at finding the optimal weights $\boldsymbol{w}^l$ and biases $\boldsymbol{b}^l$ that approximate the outputs. In the training process, the backpropagation algorithms [70] are applied to minimize the selected loss functions, which can qualify how good the neural network is in approximating the outputs [71]. In the original NPM, the loss function is formulated by the inviscid conservation of mass and momentum, and boundary conditions. Multiple pressure predictions are required to fit the selected IRK order. Besides, the spatial derivatives in the loss function are calculated with respect to the predicted space through a linear push-forward operator. In the gNPM, we include the viscous term in the loss function and simplify the pressure output of the FNN. In addition, all the spatial derivatives are calculated based on the current space rather than the predicted space.

In incompressible hydrodynamics problems, since the velocity and the pressure fields satisfy both the governing equations and the boundary conditions at every timestep, we formulate the loss function, $\mathcal{L}$, in the following manner,

$$\mathcal{L} = \mathcal{L}_{\mathrm{BC}} + \mathcal{L}_{GE}, \tag{11}$$

where $\mathcal{L}_{\mathrm{BC}}$ and $\mathcal{L}_{\mathrm{GE}}$ are the loss terms from the boundary conditions and the governing equations, respectively. $\mathcal{L}_{\mathrm{BC}}$ can be straightforwardly formulated by calculating the Mean

Squared Error (MSE) of the predictions on the boundaries comparing to the corresponding boundary conditions, which reads

$$\mathcal{L}_{\mathrm{BC}} = \mathrm{MSE}_{\mathrm{p}} + \mathrm{MSE}_{\mathrm{s}} + \mathrm{MSE}_{\mathrm{fs}}, \tag{12}$$

where $\mathrm{MSE}_{\mathrm{p}}$, $\mathrm{MSE}_{\mathrm{s}}$, and $\mathrm{MSE}_{\mathrm{fs}}$ are the MSE for the periodic boundary, solid boundary, and the free surface, respectively. Hence, at timestep t, they can be calculated through the predicted field variables

$$\begin{aligned}
\mathrm{MSE}_{\mathrm{p}} &= \frac{1}{N_{\mathrm{p}}}\sum_{i=1}^{N_{\mathrm{p}}}\left|\bar{\boldsymbol{v}}_{\Gamma_1,i}^{t+\Delta t} - \bar{\boldsymbol{v}}_{\Gamma_2,i}^{t+\Delta t}\right|^2 + \frac{1}{N_{\mathrm{p}}}\sum_{i=1}^{N_{\mathrm{p}}}\left(\bar{p}_{\Gamma_1,i}^{t+\Delta t} - \bar{p}_{\Gamma_2,i}^{t+\Delta t}\right)^2, \\
\mathrm{MSE}_{\mathrm{s}} &= \frac{1}{N_{\mathrm{s}}}\sum_{i=1}^{N_{\mathrm{s}}}\left|\bar{\boldsymbol{v}}_{\Gamma_{\mathrm{s}},i}^{t+\Delta t} - \boldsymbol{v}_{\Gamma_{\mathrm{s}},i}^{t+\Delta t}\right|^2, \\
\mathrm{MSE}_{\mathrm{fs}} &= \frac{1}{N_{\mathrm{fs}}}\sum_{i=1}^{N_{\mathrm{fs}}}\left(\bar{p}_{\Gamma_{\mathrm{fs}},i}^{t+\Delta t}\right)^2,
\end{aligned} \tag{13}$$

where N_{p}, N_{s} and N_{fs} are the number of the particles on the periodic boundary, the solid boundary and free surface. We note that those residuals will vanish to zero when the boundary conditions are strictly satisfied.

In addition, the conservation of mass and momentum are used to formulate the other loss term, $\mathcal{L}_{\mathrm{GE}}$, which reads

$$\mathcal{L}_{GE} = \mathrm{MSE}_{\mathrm{m}} + \mathrm{MSE}_{\mathrm{v}}, \tag{14}$$

where $\mathrm{MSE}_{\mathrm{m}}$ and $\mathrm{MSE}_{\mathrm{v}}$ are respectively the MSE from the conservation of mass and momentum. Based on the conservation of mass, in which the predicted field variables should satisfy Eq. (3), $\mathrm{MSE}_{\mathrm{m}}$ can be therefore written as

$$\mathrm{MSE}_{\mathrm{m}} = \frac{1}{N_{\mathrm{f}}}\sum_{i=1}^{N_{\mathrm{f}}}\left|\nabla\cdot\bar{\boldsymbol{v}}_i^{t+\Delta t}\right|^2, \tag{15}$$

where N_{f} is the number of the fluid particles, and the differential term in the equation is gained through automatic differentiation.

However, unlike the conservation of mass, when applying the conservation of momentum given as Eq. (4), there is an unknown variable, i.e., the predicted acceleration $\boldsymbol{a}^{t+\Delta t}$, on the right-hand side and no ground truth acceleration data is available to compare with it. Thus, the conservation of momentum cannot be directly used in the loss function. Here, with the predicted velocity and acceleration term, we apply both implicit and explicit k^{th} order Runge-Kutta (RK) method [31] to obtain the velocity, $\bar{\boldsymbol{v}}^t$, at the timestep t. In this manner, we can

formulate $\mathrm{MSE_v}$ by comparing $\bar{\boldsymbol{v}}^t$ with the input velocity field, $\boldsymbol{v}^t$. The implicit and explicit k^{th} order Runge-Kutta methods are respectively given as follows:

$$\begin{aligned}\bar{\boldsymbol{v}}_i^t &= \bar{\boldsymbol{v}}^{t+\Delta t} - \Delta t\sum_{j=1}^{k} d_{ij}\bar{\boldsymbol{a}}^{c_j}, \quad i=1,...,k,\\ \bar{\boldsymbol{v}}_{k+1}^t &= \bar{\boldsymbol{v}}^{t+\Delta t} - \Delta t\sum_{j=1}^{k} e_j\bar{\boldsymbol{a}}^{c_j}.\end{aligned} \tag{16}$$

where c_j, d_{ij}, e_j are the coefficients in the RK method [72], and $\bar{\boldsymbol{a}}^{c_j}$ is the predicted acceleration calculated from Eq. (4),

$$\bar{a}^{c_j} = -\frac{1}{\rho}\nabla\bar{p}^{t+\Delta t} + \frac{1}{\rho}\nabla\cdot\left(\mu\nabla\bar{v}^{c_j}\right) + g, \tag{17}$$

where $\bar{\boldsymbol{v}}^{c_j} = \bar{\boldsymbol{v}}^{t+c_j\Delta t}$ for $i = 1, \ldots, k$, are the predicted velocities from the FNN. In order to apply the k^{th} order RK method, the FNN should output k+1 terms, , as the predicted velocities rather than a single velocity output, $\bar{\boldsymbol{v}}^{t+\Delta t}$. It is worth noting that, in the NPM, the number of pressure outputs should fit the IRK order. However, since no time-dependent pressure equation is available for incompressible hydrodynamics, the predicted pressure at the next timestep is approximated through the weighted average of all the pressure outputs rather than using the IRK method [53]. The multiple pressure outputs were found not essential for the calculations but more time-consuming. Herein, we directly apply $\bar{p}^{t+\Delta t}\left[\bar{\boldsymbol{v}}^{c_1}, \ldots, \bar{\boldsymbol{v}}^{c_k}, \bar{\boldsymbol{v}}^{t+\Delta t}\right]$ as the pressure in all RK stages calculations. Excellent pressure contours are obtained and detailed in the Section 4. Accordingly, the loss function term from the conservation of momentum, $\mathrm{MSE_v}$, can be written as

$$\mathrm{MSE_v} = \frac{1}{N_\mathrm{f}}\sum_{i=1}^{N_t}\sum_{j=1}^{k+1}\left|\bar{\boldsymbol{v}}_{i,j}^t - \boldsymbol{v}_i^t\right|^2. \tag{18}$$

3.3 Time integration

Once the training process converged, we use the trained FNN to predict both the velocity and the pressure fields of the particles. The positions and the velocities of those fluid particles are updated after each timestep as follows:

$$\begin{aligned}\boldsymbol{x}^{t+\Delta t} &= \boldsymbol{x}^t + \frac{1}{2}\left(\boldsymbol{v}^t + \bar{\boldsymbol{v}}^{t+\Delta t}\right),\\ \boldsymbol{v}^{t+\Delta t} &= \bar{\boldsymbol{v}}^{t+\Delta t}.\end{aligned} \tag{19}$$

In this manner, the updated information, $\boldsymbol{x}^{t+\Delta t}$ and $\boldsymbol{v}^{t+\Delta t}$, of the particles can be fed into the next timestep as input to continue the modeling.

3.4 Numerical Implementation

To implement the gNPM, at each timestep, we build up a physics-informed FNN through the TensorFlow [73] package in python. The initial weights and biases in the FNN are randomly generated by the TensorFlow package. Besides, we apply the Mish function [74] as the activation function in the neural network. As for the optimizer, the Adam and the L-BFGS-B optimizers are the commonly used optimizers for PINNs [52]. The L-BFGS-B optimizer performs better in terms of finding the optimal, while the Adam optimizer has the advantage of handling problems with a large volume of sample data [75]. Considering that the sizes of the sample data in our numerical examples are relatively small, we select the L-BFGS-B optimizer provided by the SciPy package [76] to train the FNN. The parameter '*factr*' in the L-BFGS-B optimizer is 10, representing the extremely high accuracy convergence condition [76]. In all cases, the 4th order Runge-Kutta method is applied for time integration. In addition, we note that all the parameters for the FNN and the optimizer used in Section 4 are settled as default if not explicitly mentioned.

The detailed flowchart of the gNPM is given in Fig. 2.

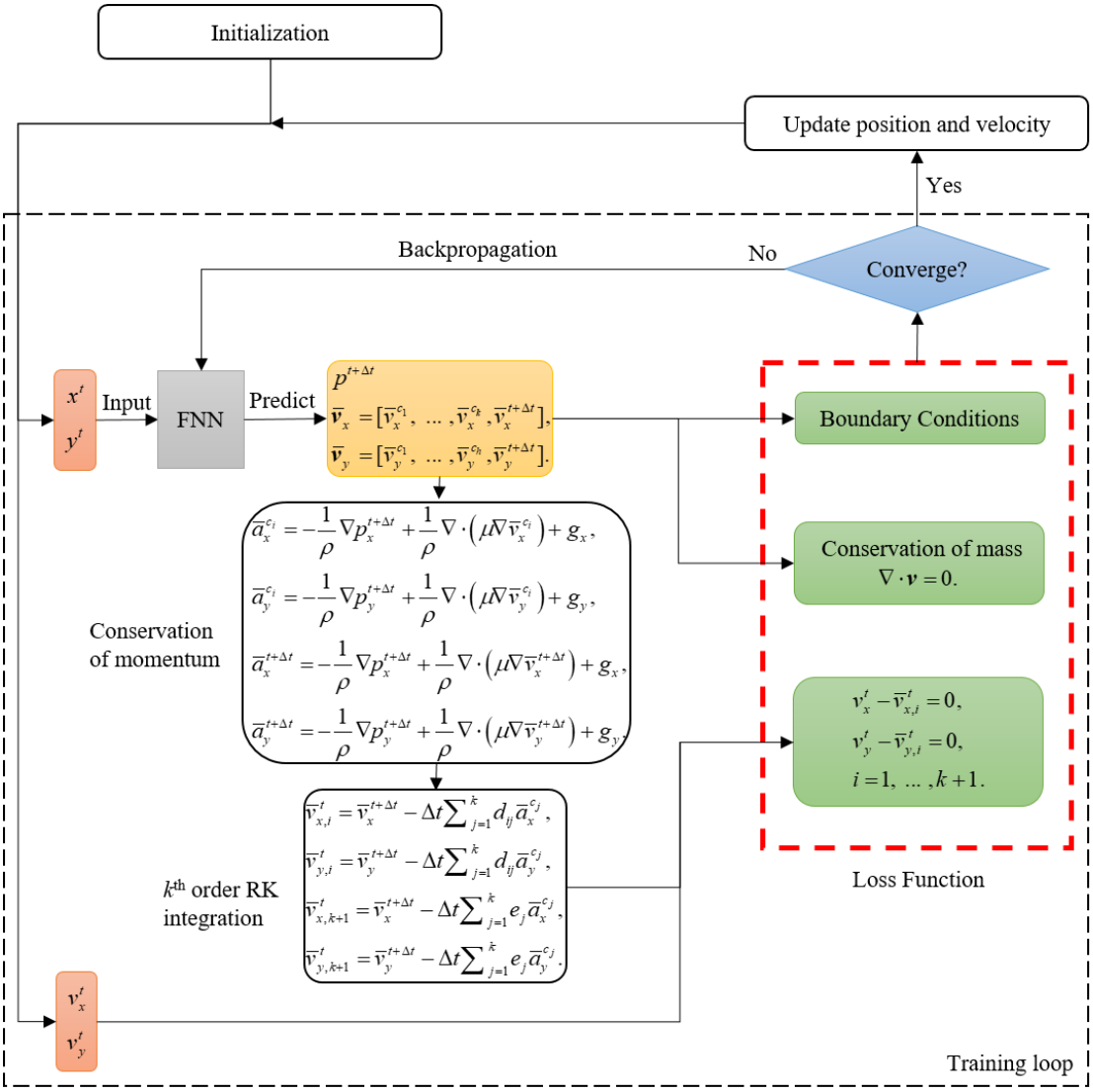

Fig. 2. Flowchart of the gNPM for a 2D hydrodynamics problem. In each timestep, the positions, x^t and y^t, from the previous timestep are fed into an FNN as inputs. The predicted outputs are two sets of velocities with respect to x and y directions and the pressure. The loss function comprises three terms: the MSEs obtained through the conservation of momentum, conservation of mass, and boundary conditions. The FNN is trained via the backpropagation algorithm. Once the training is converged, positions and velocities of particles are updated and passed forwardly to the next timestep.

4. Numerical examples

In this section, numerical examples are conducted to test the performance of the gNPM. All the numerical examples are implemented on a 64-bit Linux system with an Intel(R) Xeon(R) Gold 6140 CPU (2.3GHz). We study the convergence and robustness of the gNPM through the Poiseuille flow case with periodic boundary and static solid boundary. The errors of the gNPM with respect to different numbers of particles, neurons, layers, and particle distributions are discussed in detail. Then, we further test the effectiveness of the gNPM towards the periodic boundary and moving solid boundary through the Couette flow. Finally, a dam breaking case

is conducted to show the effectiveness of the gNPM towards free surface problems and its ability to predict the incompressible pressure field. Comparisons are also given to show the excellent robustness of the gNPM with respect to highly uneven particle distributions compared to the SPH method.

4.1. Poiseuille flow

The plane Poiseuille flow is a well-known benchmark case for various CFD methods. In this problem, the fluid between two infinite parallel walls is driven by a constant body force, g_x = 1 m/s^2. The corresponding configuration is shown in Fig. 3. The timestep is Δt = 0.01 s. The distance between the two parallel planes is L = 1 m. The density of the fluid is 1 kg/m^3, and the viscosity of the fluid is 1 kg·s/m. Due to the viscosity, the velocity in the channel direction, v_x, varies with t and y, while the vertical velocity, v_y, equals zero. Further, as the conservation of mass is naturally satisfied here, we do not apply it in the loss function. Consequently, the output of the neural network is the multiple predicted velocities in the x direction. The governing equation of this problem is given as:

$$a_x = \frac{1}{\rho}\frac{\partial^2 v_x}{\partial y^2} - g_x. \tag{20}$$

Furthermore, the no-slip boundary condition is applied to both the solid walls, which reads,

$$v_x\left(y = L/2\right) = 0,\ v_x\left(y = -L/2\right) = 0. \tag{21}$$

In the numerical implementation, we set a 1×1 m^2 computational domain and apply the periodic boundaries respectively at the left and right of the computational domain to achieve infinite length. The periodic boundary condition reads,

$$v_x\left(x = 0\right) = v_x\left(x = L\right). \tag{22}$$

The analytical solution for this problem can be found in [77].

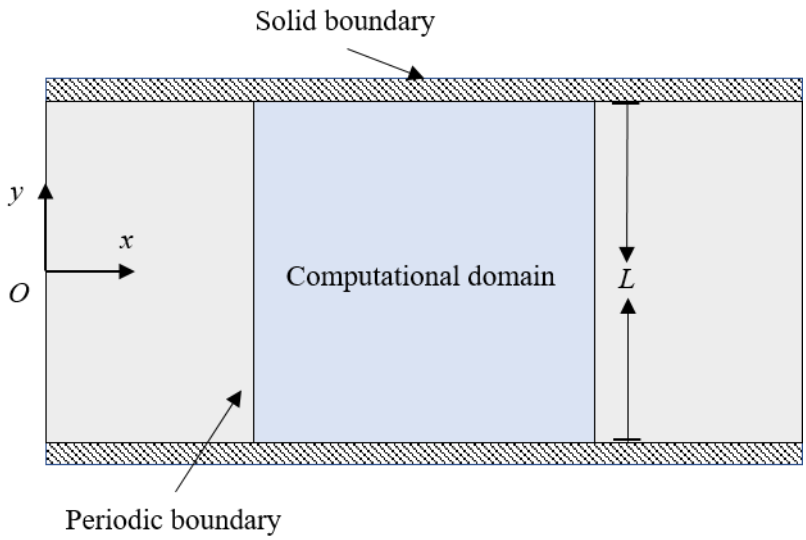

Fig. 3. Configuration of the Poiseuille and Couette flow problems.

The performance of the gNPM with respect to the number of particles used in modeling is firstly studied. Four uniform particle distributions, i.e., 25, 100, 225, and 400 particles, are used respectively to represent the fluid domain. 16 uniform particles are used to represent each solid boundary, and 14 particles are placed on each periodic boundary. The FNN is fixed to have 4 layers and 50 neurons per layer. Fig. 4 shows the velocity profiles at the cross-section of the channel at t = 0.05, 0.08, 0.12, 0.2, and 1 s. We observe that the velocity results from the gNPM are in good agreement with the analytical results. The accuracy of the gNPM is further validated through Table 1, which lists the velocity L_2 error and the overall CPU time for training with respect to the number of particles. As expected, the gNPM can achieve higher accuracy with more particles in the problem domain. It is reasonable to conclude that using more particles in the domain brings sufficient data to train the neural network, resulting in higher accuracy. Besides, we notice that the overall CPU time for training increases with the increasing number of particles, indicating that using more particles makes training more time-consuming. In addition, referring to our numerical experiments, the computational efficiency of the gNPM is more than 20 times lower than the traditional SPH method. The major CPU time is consumed in training the neural networks. Hence, more efforts should be paid for training algorithms to improve the computational efficiency of the gNPM in future research endeavors.

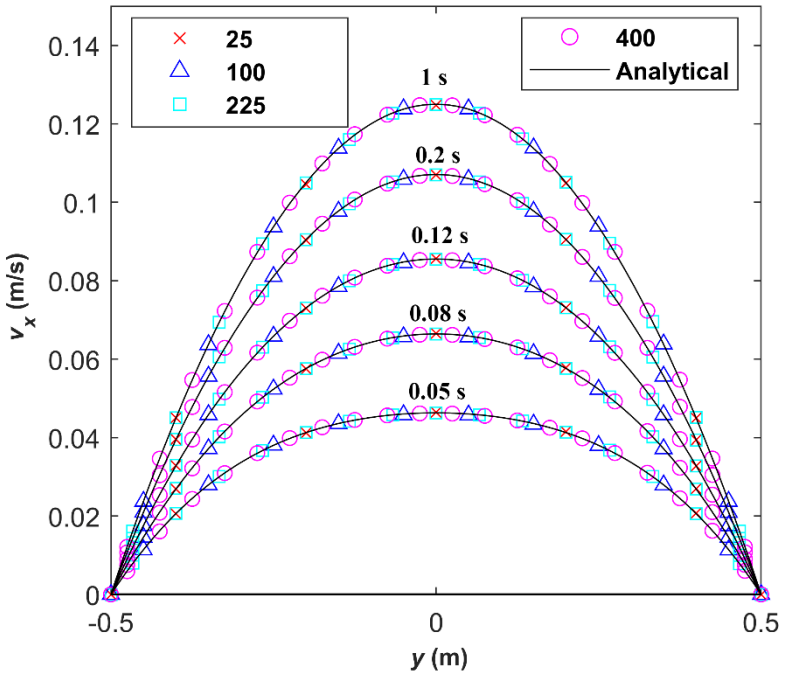

Fig. 4. Velocity profiles of the Poiseuille flow problem with respect to the different numbers of particles at t = 0.05, 0.08, 0.12, 0.2, 1 s.

Table 1
Velocity L_2 error and overall CPU time for training with respect to different numbers of particles, N_f. All results are the mean value from 10 cases with different weights and bias initialization.

N_f	25	100	225	400
Velocity L_2 Error	4.27×10^{-6}	1.11×10^{-6}	5.86×10^{-7}	4.32×10^{-7}
CPU time (s)	897.12	1128.7	1480.7	1848.1

Next, we investigate the robustness of the gNPM with respect to unevenly particle distributions. Here, we generate 255 fluid particles inside the computational domain with uniform and random distributions. Fig. 5 shows the comparison of the velocity contour between uniform and random distributions, and the analytical result at t = 1 s. It is clear that the velocity contours from both particle distributions are in good agreement with the analytical solutions (see Supplementary Video 1 and 2 for more details). Fig. 6 shows the comparison of the histogram of the velocity L_2 error with random and uniform particle distributions. For each particle distribution, 100 cases with different weights and bias initialization are used. We find that the distributions of the velocity L_2 error from both cases show approximately normal distribution. The mean values of the velocity L2 error with respect to random and uniform particle

distributions are 6.93×10^{-7} and 6.07×10^{-7}, respectively, indicating that uneven particle distributions can induce a small decrease in the modeling accuracy. However, both the values and the distribution of the errors are quite close to each other, demonstrating that uneven particle distributions have a small influence on the gNPM, and that the method is robust enough with respect to unevenly distributed particles.

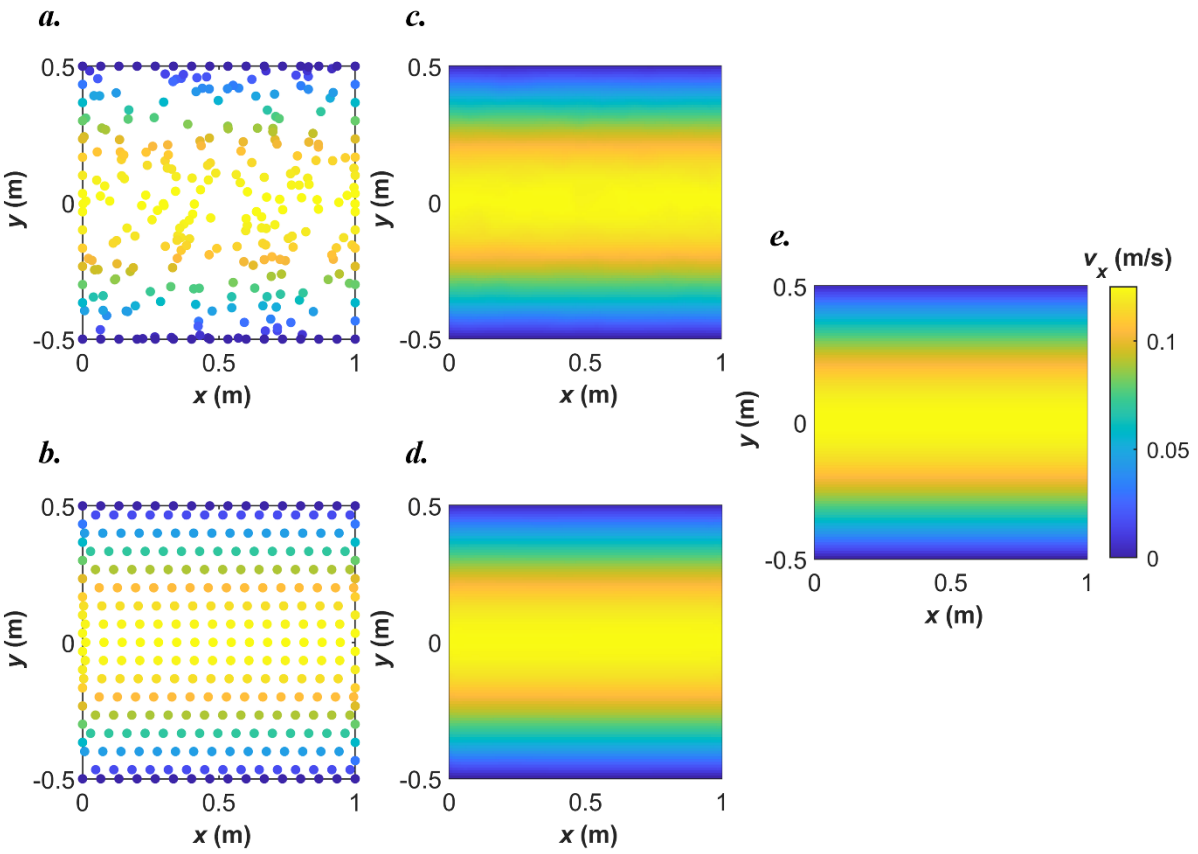

Fig. 5. Velocity contour and interpolated velocity contour of the Poiseuille flow problem at t = 1 s. (a) The velocity contour result from gNPM with randomly generated particles; (b) The velocity contour result from gNPM with uniformly generated particles; (c) The interpolated velocity contour from (a); (d) The interpolated velocity contour from (b); (e) The analytical solution.

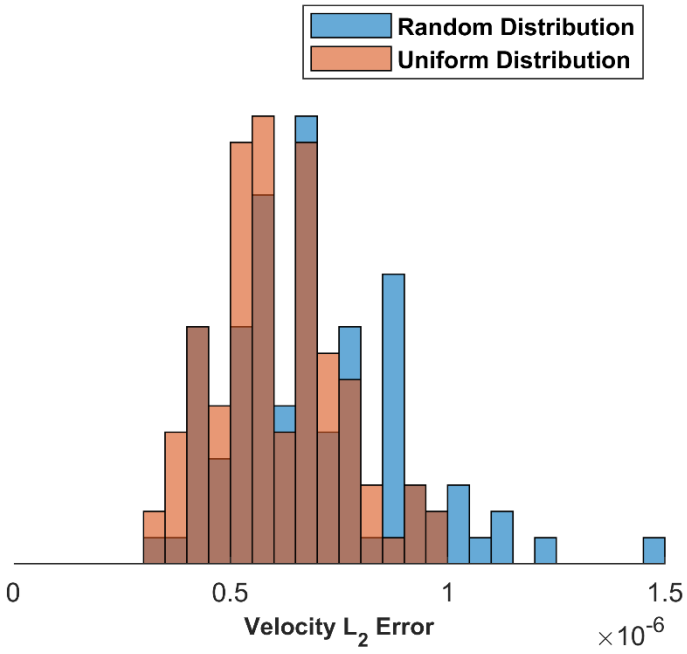

Fig. 6. Histogram of the velocity L_2 error of random and uniform particle distributions. Total 100 cases are studied for each particle distribution with different weights and bias initializations.

In addition, we revisit the problem to test the performance of the gNPM towards different numbers of layers and neurons. Table 2 lists the velocity L_2 error with different settings. It can be seen that the velocity L_2 error in all the cases remains at a low level. Besides, the velocity L_2 error decreases with the increasing number of layers. However, we notice that the error exhibit fluctuations with the increasing number of neurons per layer. In fact, the same issues also appear in other neural network applications [32, 51]. We need to note that, for a given problem, more layers with more neurons do not guarantee a better result. Researchers have pointed out that a neural network with more layers and neurons is more capable of approximating more complex functions, but for a given simple problem, using a too complex neural network may induce overfitting [14]. Besides, larger neural networks also bring difficulties to the training algorithms and are more time-consuming in the training process. Thus, the number of layers and neurons for the applications should be carefully determined. Fortunately, some criteria to guide this decision of the number of layers and neurons for various problems have been given in [14, 78]. However, further investigations for more accurate criteria for the neural network are still required.

Table 2
Velocity L_2 error with respect to different layers and neurons. All the results are the mean value from 10 cases with different weights and biases initializations.

Layers \ Neurons	20	30	40	50
1	1.08×10^{-6}	1.34×10^{-6}	1.57×10^{-6}	1.45×10^{-6}
2	5.89×10^{-7}	7.64×10^{-7}	7.12×10^{-7}	6.94×10^{-7}
3	6.20×10^{-7}	4.12×10^{-7}	4.49×10^{-7}	4.35×10^{-7}
4	6.35×10^{-7}	4.69×10^{-7}	4.02×10^{-7}	4.28×10^{-7}

4.2. Couette flow

Next, we present the plane Couette flow simulated with the gNPM. In this problem, the fluid between two infinite parallel walls is driven by the upper moving boundary with constant velocity, $v_x = 1$ m/s. The configuration of the problem is same as the Poiseuille flow case, which is shown in Fig. 3. The timestep $\Delta t = 2\times10^{-3}$ s. The density of the fluid is 1 kg/m^3, and the viscosity of the fluid is 1 kg·s/m. The FNN is fixed to have 4 layers and 50 neurons per layer. Here, we use the same geometry settings and fluid properties as the Poiseuille case. In this manner, the governing equation can be written as:

$$a_x = \frac{1}{\rho}\frac{\partial^2 v_x}{\partial y^2}. \tag{23}$$

The corresponding boundary conditions can be written as:

$$\begin{aligned} &v_x\left(y = L/2\right) = 1 \text{ m/s}, \\ &v_x\left(y = -L/2\right) = 0, \\ &v_x\left(x = 0\right) = v_x\left(x = L\right). \end{aligned} \tag{24}$$

Fig. 7 shows the velocity profile of the Couette flow from both random and uniform particle distributions at t = 0.01, 0.03, 0.06, 0.13 and 0.5 s. It is observed that both results align well with the analytical solutions [77]. The velocity L_2 error from random and uniform particle distributions are 9.50×10^{-5} and 1.42×10^{-4}, indicating that the gNPM can achieve high accuracy towards both particle distributions and is robust with respect to uneven particle distributions. Besides, we also observe that velocity of the particles near the moving boundary remains a high level of accuracy throughout modeling, despite the existence of high non-linearity at the beginning stage t = 0.01 s. Fig. 8 shows the velocity contour and interpolated velocity contour

of the Couette flow problem at t = 0.5 s. It can be observed that the velocity contours from both the particle distributions agree well with the analytical solutions (see Supplementary Video 3 and 4 for more details).

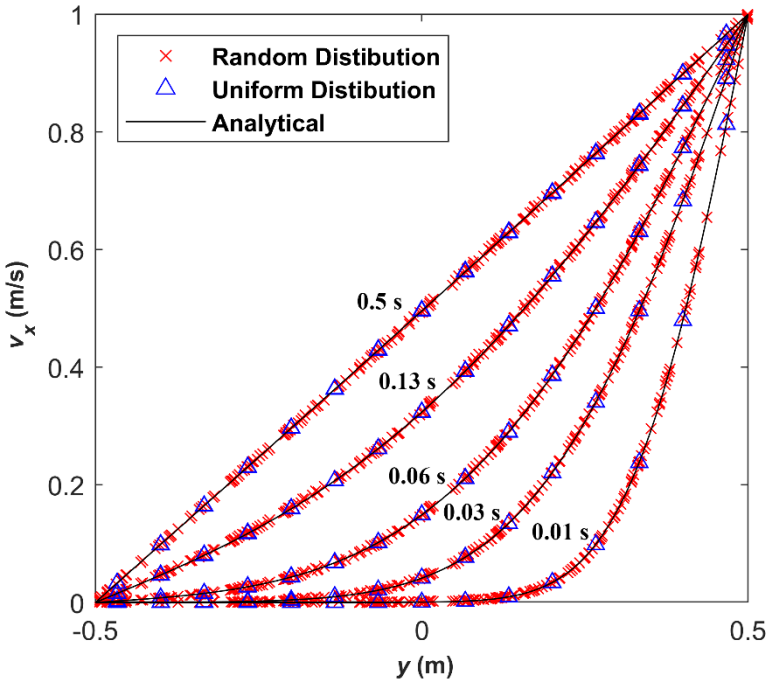

Fig. 7. Velocity profiles of the Couette flow problem with respect to two different particle distributions at t = 0.01, 0.03, 0.06, 0.13, and 0.5 s.

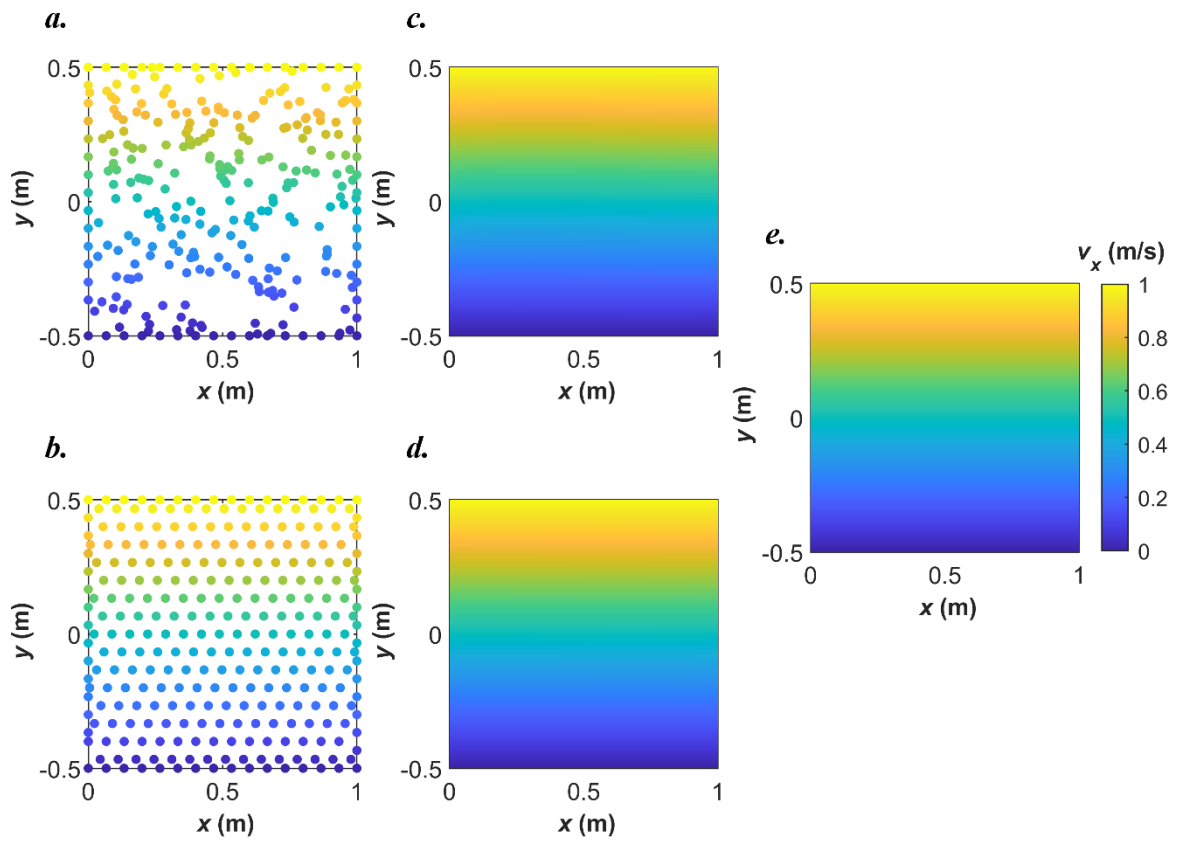

Fig. 8. Velocity contour and interpolated velocity contour of the Couette flow problem at t = 0.5 s. (a) The velocity contour result from gNPM with randomly generated particles; (b) The velocity contour result from gNPM with uniformly generated particles; (c) The interpolated velocity contour from (a); (d) The interpolated velocity contour from (b); (e) The analytical solution.

4.3. Breaking dam

Finally, a breaking dam case is studied to test the performance of the gNPM with respect to the free surface problem and show the ability of the gNPM in predicting the incompressible pressure field. Besides, we also conduct some comparisons between the gNPM and NPM. The configuration for the breaking dam case is shown in Fig. 9. The initial dam length L and height H are respectively 0.1 m and 0.2 m. Water is used as the fluid whose viscosity and density are respectively 1×10^{-3} N·s/m^2 and 1×10^{3} kg/m^3. The gravitational acceleration is 9.81 m/s^2. Here, we use 741 particles to represent the fluid domain. As for the solid boundary condition, 121 fixed wall particles are uniformly placed on the solid boundary with 0.005 m intervals. Meanwhile, 59 fluid particles are placed on the initial free surface with the same interval as the fixed wall particles. In modeling, the neural network is fixed to have 4 layers with 50 neurons per layer, and the timestep is $\Delta t = 2\times10^{-3}$ s. It is worth noting that, during modeling, the scale of the output pressure field is far larger than the velocity field, which can result in low

efficiency and non-convergence in the training process [79]. Hence, we apply the normalized pressure, $P = p/\rho$, so that the scale of the pressure output is the same as the velocity outputs. In this manner, the conservation of momentum used in the training process can be rewritten as:

$$a = -\nabla P + \frac{1}{\rho}\nabla \cdot (\mu \nabla v) + g. \tag{25}$$

In addition, we note that the free surface particles along the modeling will remain unchanged. Accordingly, we can easily impose the free surface boundary condition on those particles. For more complex problems, additional boundary tracking algorithms, such as the alpha-shape technique [80], are required to identify the free surface particles at every timestep. However, this is beyond the current scope of this paper and will be covered in our future works.

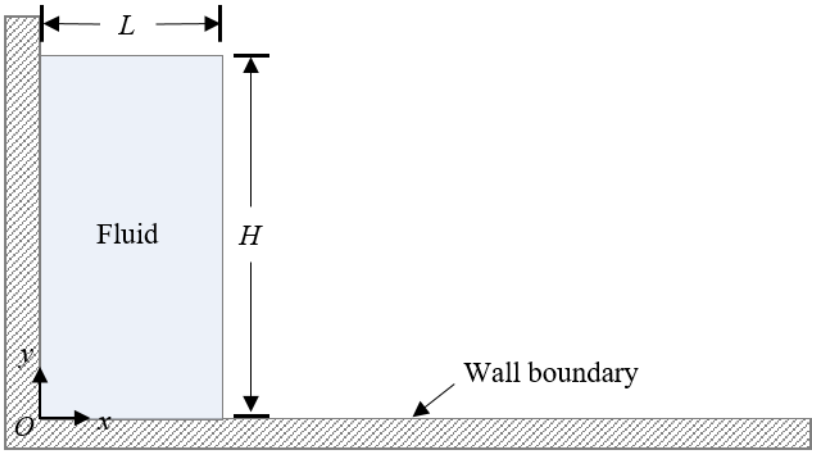

Fig. 9. Configuration of the breaking dam problem.

We test the gNPM with uniformly and randomly distributed particles. Fig. 10 shows the configurations and the pressure contours of the dam at t = 0, 0.09, and 0.18 s. The results from the Incompressible SPH (ISPH) method [67] with the shifting particle algorithm [66] are also provided for comparison (see Supplementary Video 5 for more details). We note that the number of fluid particles in ISPH is the same as in the gNPM. From Fig. 10, it can be observed that the free-surface of the dam at the given times can be easily recognized with both particle distributions, and the shapes of the dam from both the methods greatly align with each other. This can be further demonstrated through Fig. 11, where the outlines from the gNPM, ISPH method, and VOF results [81] are compared at t = 0.09, 0.14, and 0.2 s. It is worth noting that, in Fig. 10, at the leading edge from the ISPH method, there are particles far away from the continuous fluid domain, while smoother leading edge outlines are obtained through our method. Meanwhile, the pressure contours from different methods are in good agreement. Due

to the existence of the gravitational acceleration, pressure near the bottom-left solid boundary is relatively high to prevent the fluid from penetrating the wall. Further, we observe that the gNPM is robust in dealing with unevenly distributed particles, as shown in the middle column of Fig. 10, where favorable pressure results can be obtained. With the same randomly distributed particles, the ISPH method results in severe failure, as shown in Fig. 12, even though the particle shifting algorithm is applied to re-distribute the irregular particles to be more regular [4]. Fig. 13 plots the non-dimensional leading edge position, $X = x/H$, versus the non-dimensional time, $T = t(g/H)^{0.5}$. The VOF and the experimental results from references [81] are also provided for the sake of a more detailed comparison. The results from our method agree very closely with the experimental data and other methods, for both uniformly and randomly distributed particles.

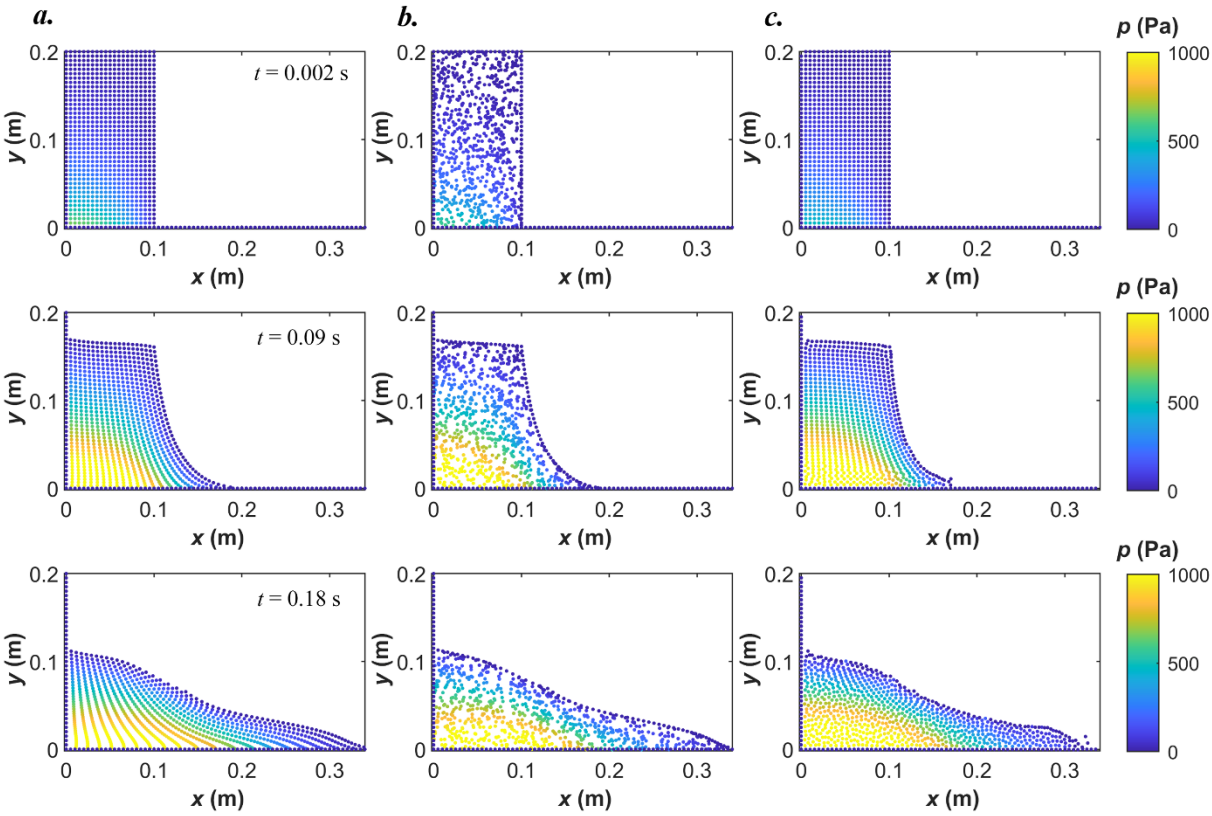

Fig. 10. Configurations of the fluid and pressure field at t = 0, 0.09, and 0.18 s from (a) gNPM with uniformly distributed particles, (b) gNPM with randomly distributed particles, and (c) ISPH with uniformly distributed particles and particle shifting algorithm [66].

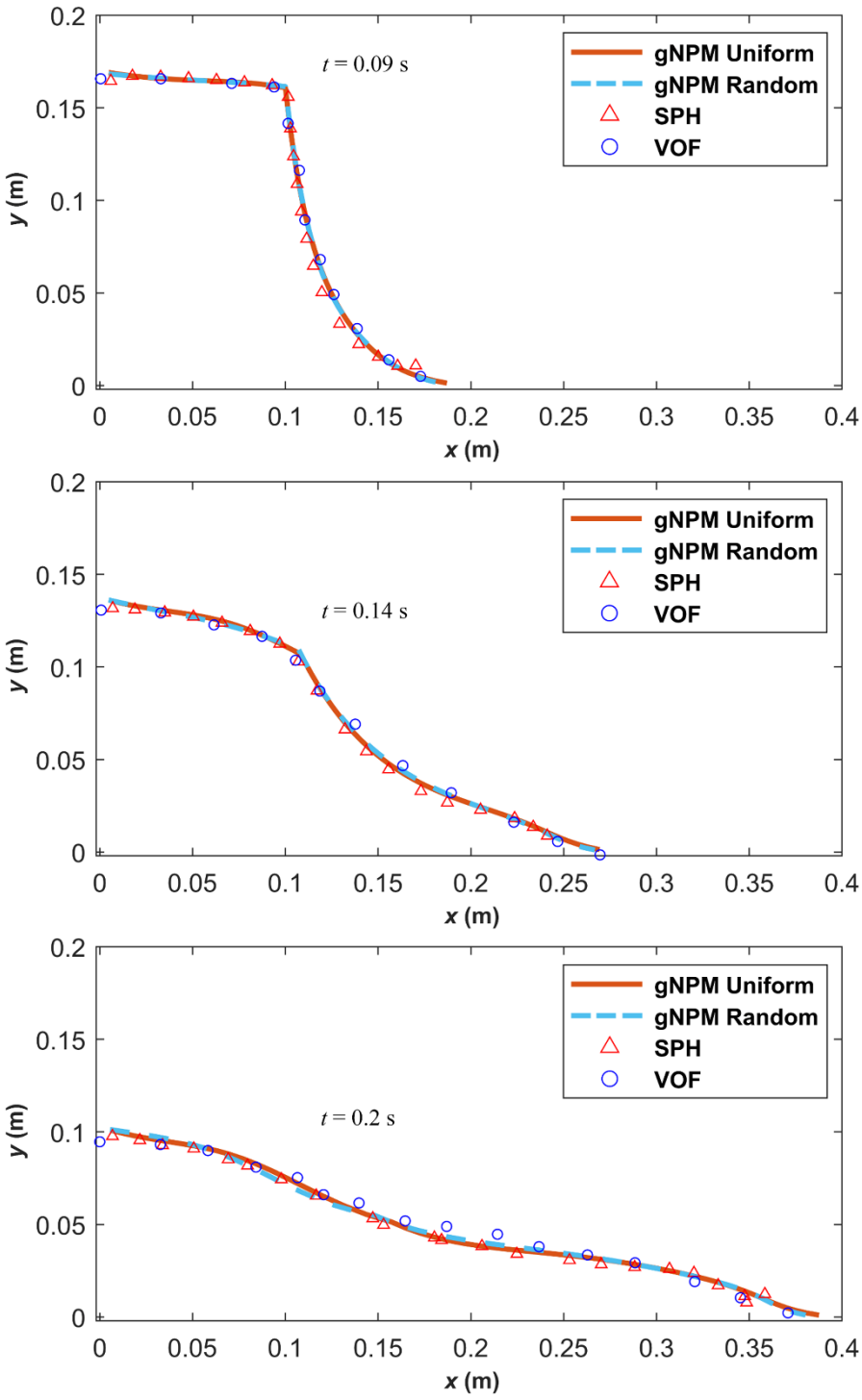

Fig. 11. Comparison of the free surface line from gNPM with different particle distributions, ISPH, and VOF [81] at t = 0.09, 0.14, and 0.2 s.

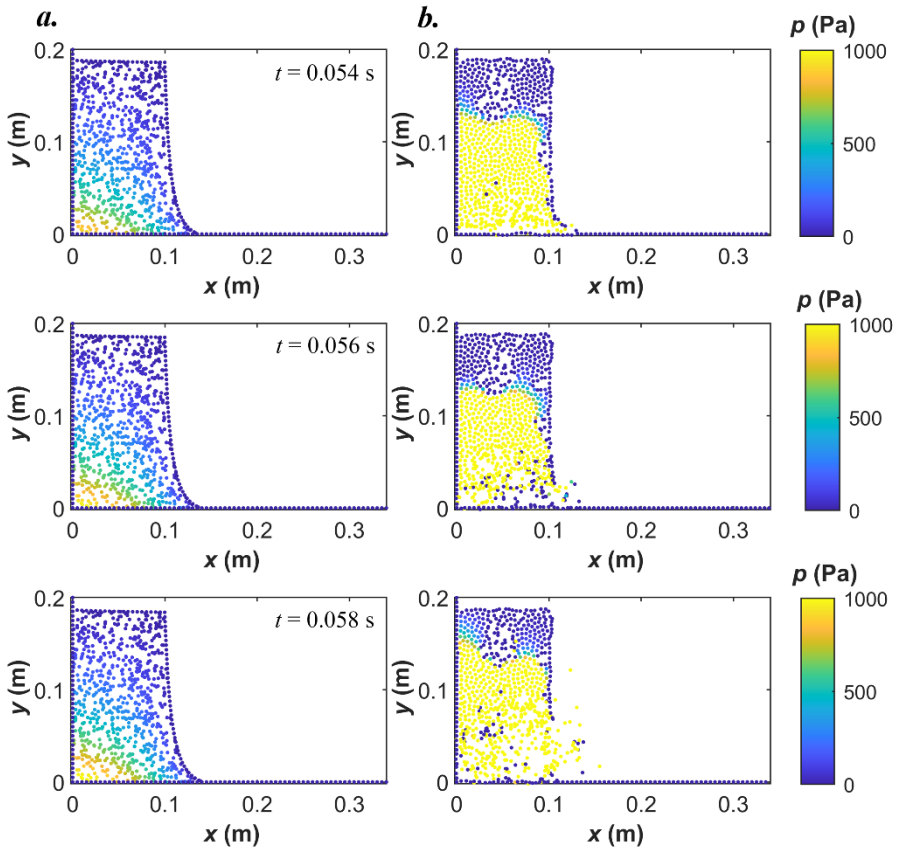

Fig. 12. Comparison of the results obtained by gNPM and ISPH with the same random particle distribution at t = 0.054, 0.056, and 0.058 s. The ISPH results explode at the right bottom of the dam due to the uneven particle distribution. (a) Results from gNPM; (b) Results from ISPH method.

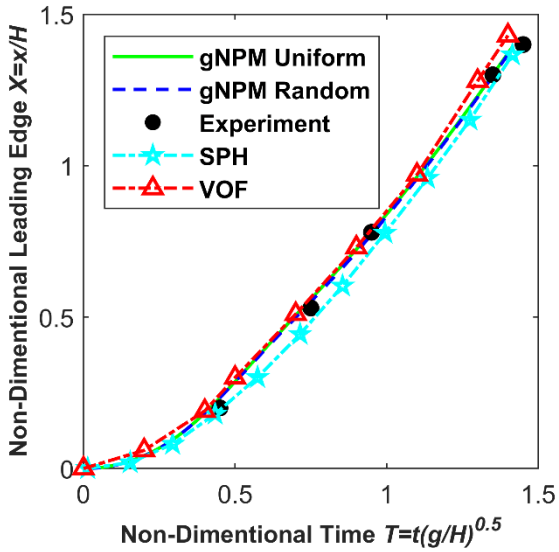

Fig. 13. Comparison of the non-dimensional leading-edge position versus the non-dimensional time plot from gNPM with different particle distributions, experimental data, ISPH, and VOF [81].

In addition, we conduct comparisons between the proposed gNPM and the original NPM. Fig. 14 shows the comparisons of the pressure fields along with time at three different dam positions. As shown in the figure, the pressure fields from both the gNPM and the NPM are quite close to the SPH results at positions (0,0) and (0.1,0), indicating that the simplified pressure scheme in the gNPM can be as effective as the NPM with multiple pressure outputs. Besides, the pressure curve from the gNPM still aligns well with the SPH result at the position (0,0.1), while the pressure curve from the NPM generates a departure after t = 0.1 s. Meanwhile, we also compare the gNPM with the NPM regarding the final loss, CPU time, and iterations for one timestep training, which are given in Fig. 15. We studied 100 cases with random weights and biases initializations for both the gNPM and NPM. As shown in Fig. 15 (a), the final losses of the gNPM and NPM can converge to the same level (the mean values of the final losses of the gNPM and NPM are respectively 4.29×10^{-3} and 4.75×10^{-3}). However, to achieve the same level of final loss, the NPM requires more CPU time and iterations than the gNPM, as shown in Fig. 15 (b) and (c). The gNPM can take an average of 9945 iterations (corresponding to 2256.4 s) to converge, while the NPM requires an average of 12165 iterations (corresponding to 2747.3 s). We note that the multiple pressure outputs in the NPM are related to the Runge-Kutta stages, which may ensure the accuracies of pressure fields for intermediate Runge-Kutta stages. However, the NPM updates the next timestep pressure field by using the weighted average of the multiple intermediate pressure fields. This simple approximation will bring errors for the predicted pressure field at the next timestep. On the contrary, the gNPM directly applies the pressure at the next timestep in all Runge-Kutta calculations. Although it is difficult to analytically quantify errors of the pressure caused by the approximations in these two methods, nevertheless, based on the numerical examples, we can conclude that the single pressure output in the gNPM can improve the computational efficiency compared to the original NPM, while the pressure field from the gNPM can maintain a favorable level of accuracy as the NPM and ISPH.

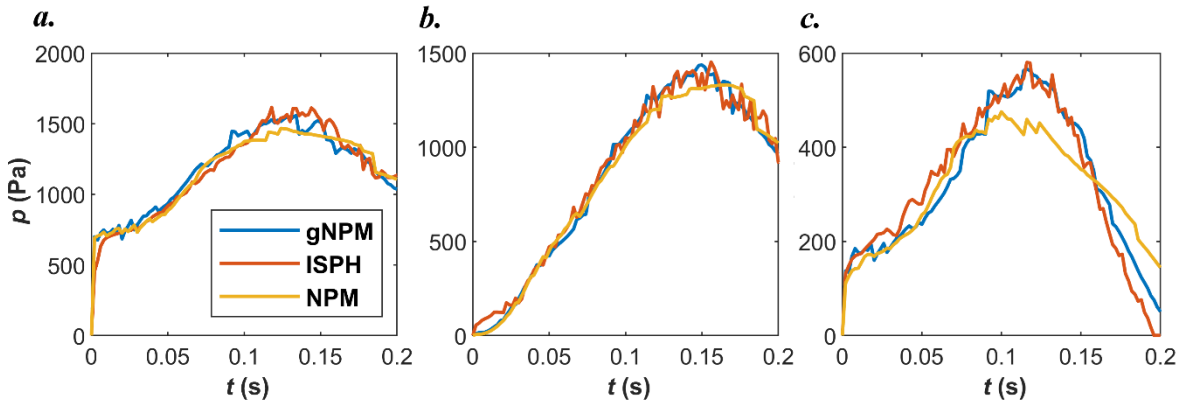

Fig. 14. Comparisons of the pressure values along with time at (a) position (0,0); (b) position (0.1,0); (c) position (0,0.1). The results from gNPM, ISPH and NPM are plotted in blue, red and yellow, respectively.

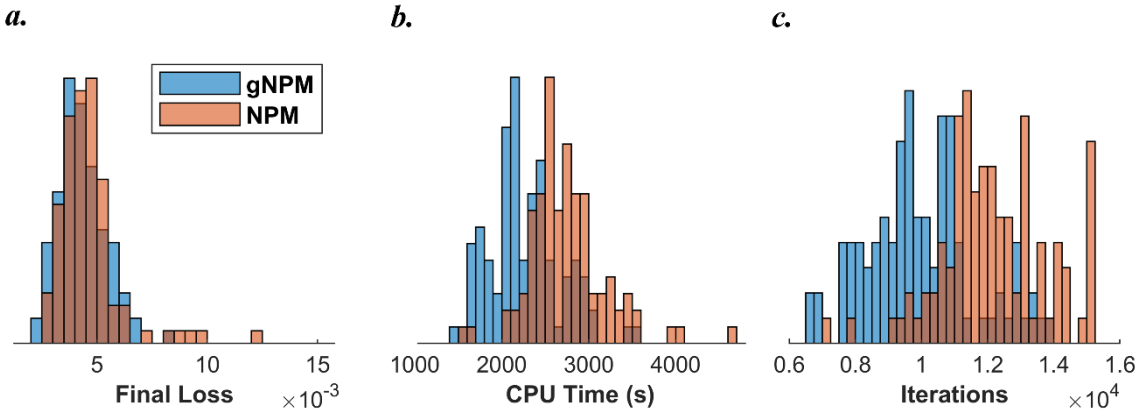

Fig. 15. Histograms of (a) the final loss, (b) CPU time, and (c) iterations for one timestep training. For each method, 100 cases with different weights and biases initializations are studied.

5. Summary

In this work, we have developed a general Neural Particle Method (gNPM), a physics-informed neural network (PINN)-based truly meshfree method, for viscous hydrodynamics modeling. In gNPM, the field variables in terms of multiple velocities relating to the selected order of the RK method and the single pressure at the next timestep are approximated through an FNN. The neural network is trained through the physical information in terms of the boundary conditions and governing equations of hydrodynamics including the viscous term. The spatial derivatives in the governing equations are obtained through the automatic differentiation at the current configuration. Particles are used to discretize the computational domain and boundaries. In addition, the particles move with the fluid, and the positions of particles are updated after each timestep. In this manner, the free surfaces and moving boundaries can be easily tracked at each timestep.

Through numerical validations, we found that the gNPM shows good convergence property for the number of particles used in modeling. Further, the gNPM can achieve high levels of accuracy with different sizes of neural networks. Both velocity and pressure fields can be accurately predicted for incompressible hydrodynamics problems. The effectiveness of the gNPM towards static, moving, periodic, and free-surface boundary conditions are also demonstrated. Specifically, the gNPM can produce smoother free-surface boundary lines than the SPH method. In addition, the proposed gNPM is proven to be more robust than the SPH method when facing highly uneven particle distributions.

Compared to the original NPM, the proposed gNPM is a more generalized method for viscous hydrodynamics applications. Unlike the NPM that calculates the spatial derivatives with respect to the predicted configuration, the gNPM computes spatial derivatives within the current configuration, making the gNPM more straightforward to implement the viscous term in modeling. The capability for the viscous modeling is a significant advantage of gNPM to solve problems in the real world. Besides, the single pressure output simplification in the gNPM has been demonstrated to be as effective as the NPM and ISPH. Moreover, the simplified pressure output reduces the required iterations for convergence compared to the original NPM. Hence, the gNPM is computationally more efficient than the original NPM.

It is worth noting that, unlike other physics-informed deep learning hydrodynamics modeling techniques that apply neural networks to study the whole spatiotemporal space [50-52], the gNPM separates spatial and temporal coordinates. Normal time integration is used to handle the temporal space, and only the spatial space is studied through neural networks. In this manner, users can customize and control neural networks at different timesteps or periods to generate more accurate results. Further, the well-trained neural network for a given timestep can be applied in the next timestep training through transfer learning [82]. Transfer learning can greatly improve the efficiency of neural network training [82]. Moreover, the proposed gNPM can be extended to broader applications, including the dynamics of solids. This is one of the avenues we intend to explore in our future work.

Nevertheless, we still need to note that there are certain limitations of the studies in this work. The proposed gNPM relies on neural networks, and more investigations regarding neural networks should be thoroughly conducted. Terms in the loss function can face severe scale differences for complex problems. Hence, more advanced techniques, e.g., the adaptive learning [39] and the neural tangent kernel algorithm [38], should be incorporated to balance the scales of the loss terms. In addition, as the developed gNPM is also computationally more

expensive than traditional CFD methods, further numerical techniques should be explored to improve the computational efficiency of gNPM, which will broaden the applicability of the proposed gNPM.

Author Contributions

J. Bai: Conceptualization, Methodology, Coding, Formal analysis, Writing-Original draft. **Y. Zhou:** Conceptualization, Methodology, Formal analysis, Writing-Reviewing and Editing. **Y. Ma:** Coding, Formal analysis. **H. Jeong:** Coding. **H. Zhan**: Writing-Reviewing and Editing. **C. Rathnayaka:** Writing-Reviewing and Editing. **E. Sauret:** Writing-Reviewing and Editing. **Y. Gu:** Conceptualization, Formal analysis, Writing-Reviewing and Editing, Supervision.

Declaration of Competing Interest

The authors declare no competing interests.

Acknowledgements

Support from the ARC Discovery Project (DP200102546) and the High-Performance Computing (HPC) resources provided by the Queensland University of Technology (QUT) are gratefully acknowledged.

References

[1] S.J. Lind, B.D. Rogers, P.K. Stansby, Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling, Proceedings of the Royal Society A, 476 (2020) 20190801.

[2] G.-R. Liu, M.B. Liu, Smoothed Particle Hydrodynamics: A Meshfree Particle Method, World Scientific, Singapore, SINGAPORE, 2003.

[3] M.S. Shadloo, G. Oger, D. Le Touzé, Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: Motivations, current state, and challenges, Computers & Fluids, 136 (2016) 11-34.

[4] T. Ye, D.Y. Pan, C. Huang, M.B. Liu, Smoothed particle hydrodynamics (SPH) for complex fluid flows: Recent developments in methodology and applications, Physics of Fluids, 31 (2019).

[5] G.-R. Liu, Y.T. Gu, An introduction to meshfree methods and their programming, Springer Science & Business Media, 2005.

[6] J.W. Thomas, Numerical partial differential equations: finite difference methods, Springer Science & Business Media, 2013.

[7] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, Handbook of numerical analysis, 7 (2000) 713-1018.

[8] E. Oñate, A stabilized finite element method for incompressible viscous flows using a finite increment calculus formulation, Computer Methods in Applied Mechanics and Engineering, 182 (2000) 355-370.

[9] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Monthly notices of the royal astronomical society, 181 (1977) 375-389.

[10] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, Astronomical Journal, 82 (1977) 1013-1024.

[11] C.M. Rathnayaka, H.C.P. Karunasena, W.D.C.C. Wijerathne, W. Senadeera, Y.T. Gu, A three-dimensional (3-D) meshfree-based computational model to investigate stress-strain-time relationships of plant cells during drying, PLOS ONE, 15 (2020) e0235712.

[12] S. Koshizuka, Y. Oka, Moving-particle semi-implicit method for fragmentation of incompressible fluid, Nuclear Science Engineering, 123 (1996) 421-434.

[13] S.N. Atluri, T. Zhu, A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics, Computational Mechanics, 22 (1998) 117-127.

[14] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 521 (2015) 436-444.

[15] I. Goodfellow, Y. Bengio, A. Courville, Deep learning (adaptive computation and machine learning series), Cambridge Massachusetts, (2017) 321-359.

[16] Y. Bengio, I. Goodfellow, A. Courville, Deep learning, MIT press Massachusetts, USA:, 2017.

[17] D. Chicco, P. Sadowski, P. Baldi, Deep autoencoder neural networks for gene ontology annotation predictions, in: Proceedings of the 5th ACM conference on bioinformatics, computational biology, and health informatics, 2014, pp. 533-540.

[18] D. CireAan, U. Meier, J. Masci, J. Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural networks, 32 (2012) 333-338.

[19] C. Kleanthous, S. Chatzis, Gated mixture variational autoencoders for value added tax audit case selection, Knowledge-Based Systems, 188 (2020) 105048.

[20] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics, 52 (2020) 477-508.

[21] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, arXiv preprint arXiv:2105.09506, (2021).

[22] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics, 3 (2021) 422-440.

[23] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of Fluid Mechanics, 656 (2010) 5-28.

[24] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic mode decomposition: data-driven modeling of complex systems, SIAM, 2016.

[25] M.O. Williams, I.G. Kevrekidis, C.W. Rowley, A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition, Journal of Nonlinear Science, 25 (2015) 1307-1346.

[26] B. Moya, D. Gonzalez, I. Alfaro, F. Chinesta, E. Cueto, Learning slosh dynamics by means of data, Computational Mechanics, 64 (2019) 511-523.

[27] L. Ladický, S.H. Jeong, B. Solenthaler, M. Pollefeys, M. Gross, Data-driven fluid simulations using regression forests, ACM Transactions on Graphics, 34 (2015) 199.

[28] J. Bai, Y. Zhou, C.M. Rathnayaka, H. Zhan, E. Sauret, Y. Gu, A data-driven smoothed particle hydrodynamics method for fluids, Engineering Analysis with Boundary Elements, 132 (2021) 12-32.

[29] Y. Xie, E. Franz, M.Y. Chu, N. Thuerey, tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow, ACM Transactions on Graphics, 37 (2018) 95.

[30] B. Kim, V.C. Azevedo, N. Thuerey, T. Kim, M. Gross, B. Solenthaler, Deep fluids: A generative network for parameterized fluid simulations, Computer Graphics Forum, 38 (2019) 59-70.

[31] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019) 686-707.

[32] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering, 365 (2020).

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

[34] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science, 367 (2020) 1026-1030.

[35] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Computer Methods in Applied Mechanics and Engineering, 379 (2021).

[36] S. Wang, P. Perdikaris, Deep learning of free boundary and Stefan problems, Journal of Computational Physics, 428 (2021).

[37] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, Journal of Computational Physics, 394 (2019) 136-152.

[38] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, Journal of Computational Physics, (2021).

[39] D. Liu, Y. Wang, Multi-fidelity physics-constrained neural network and its application in materials modeling, Journal of Mechanical Design, 141 (2019).

[40] V.M. Nguyen-Thanh, C. Anitescu, N. Alajlan, T. Rabczuk, X. Zhuang, Parametric deep energy approach for elasticity accounting for strain gradient effects, Computer Methods in Applied Mechanics and Engineering, 386 (2021).

[41] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Computer Methods in Applied Mechanics and Engineering, (2021).

[42] E. Kharazmi, Z. Zhang, G.E.M. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Computer Methods in Applied Mechanics and Engineering, 374 (2021).

[43] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, Journal of Computational Physics, 447 (2021).

[44] E. Haghighat, A.C. Bekar, E. Madenci, R. Juanes, A nonlocal physics-informed deep learning framework using the peridynamic differential operator, Computer Methods in Applied Mechanics and Engineering, 385 (2021).

[45] X. Zhuang, H. Guo, N. Alajlan, H. Zhu, T. Rabczuk, Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning, European Journal of Mechanics - A/Solids, 87 (2021).

[46] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, Journal of Computational Physics, 425 (2021).

[47] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering, 358 (2020).

[48] C. P. Batuwatta-Gamage, C.M. Rathnayaka, H.C.P. Karunasena, W.D.C.C. Wijerathne , H. Jeong, Z. G. Welsh, M.A. Karim, Y.T. Gu, A physics-informed neural network-based surrogate framework to predict moisture concentration and shrinkage of a plant cell during drying, Journal of Food Engineering, (Forthcoming).

[49] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theoretical and Applied Fracture Mechanics, 106 (2020).

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

[51] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, Journal of Computational Physics, 426 (2021).

[52] Z. Xiang, W. Peng, X. Zheng, X. Zhao, W. Yao, Self-adaptive loss balanced Physics-informed neural networks for the incompressible Navier-Stokes equations, arXiv preprint arXiv:2104.06217, (2021).

[53] H. Wessels, C. Weißenfels, P. Wriggers, The neural particle method – An updated Lagrangian physics informed neural network for computational fluid dynamics, Computer Methods in Applied Mechanics and Engineering, 368 (2020).

[54] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing, 317 (2018) 28-41.

[55] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks, 61 (2015) 85-117.

[56] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: International conference on machine learning, PMLR, 2013, pp. 1139-1147.

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

[58] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of machine learning research, 12 (2011).

[59] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701, (2012).

[60] R. Reed, R.J. MarksII, Neural smithing: supervised learning in feedforward artificial neural networks, Mit Press, 1999.

[61] H. Guo, X. Zhuang, T. Rabczuk, A deep collocation method for the bending analysis of Kirchhoff plate, Computers, Materials & Continua, 59 (2019) 433--456.

[62] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Computer Methods in Applied Mechanics and Engineering, 362 (2020).

[63] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, Journal of machine learning research, 18 (2018).

[64] J.C. Strikwerda, Finite difference schemes and partial differential equations, SIAM, 2004.

[65] T. Belytschko, Y. Krongauz, J. Dolbow, C. Gerlach, On the completeness of meshfree particle methods, International Journal for Numerical Methods in Engineering, 43 (1998) 785-819.

[66] S.J. Lind, R. Xu, P.K. Stansby, B.D. Rogers, Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves, Journal of Computational Physics, 231 (2012) 1499-1523.

[67] R. Xu, P. Stansby, D. Laurence, Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach, Journal of Computational Physics, 228 (2009) 6703-6725.

[68] Z.L. Zhang, M.B. Liu, A decoupled finite particle method for modeling incompressible flows with free surfaces, Applied Mathematical Modelling, 60 (2018) 606-633.

[69] J.J. Monaghan, Simulating free surface flows with SPH, Journal of Computational Physics, 110 (1994) 399-406.

[70] M.A. Nielsen, Neural networks and deep learning, Determination press San Francisco, CA, 2015.

[71] K. Janocha, W.M. Czarnecki, On loss functions for deep neural networks in classification, arXiv preprint arXiv:1702.05659, (2017).

[72] A. Iserles, A first course in the numerical analysis of differential equations, Cambridge university press, 2009.

[73] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467, (2016).

[74] D. Misra, Mish: A self regularized non-monotonic neural activation function, arXiv preprint arXiv:1908.08681, 4 (2019) 2.

[75] S. Markidis, The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?, Front Big Data, 4 (2021) 669097.

[76] J.L. Morales, J. Nocedal, Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization", ACM Transactions on Mathematical Software, 38 (2011) 1-4.

[77] J.P. Morris, P.J. Fox, Y. Zhu, Modeling low Reynolds number incompressible flows using SPH, Journal of Computational Physics, 136 (1997) 214-226.

[78] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153-160.

[79] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448-456.

[80] M. Fayed, H.T. Mouftah, Localised alpha-shape computations for boundary recognition in sensor networks, Ad Hoc Networks, 7 (2009) 1259-1269.

[81] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics, 39 (1981) 201-225.

[82] T. George Karimpanal, R. Bouffanais, Self-organizing maps for storage and transfer of knowledge in reinforcement learning, Adaptive Behavior, 27 (2019) 111-126.