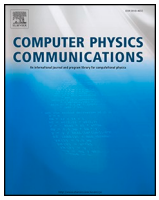
Computer Physics Communications 301 (2024) 109220

Contents lists available at ScienceDirect

Computer Physics Communications

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



Computational Physics

A hybrid method combining ISPH with graph neural network for simulating free-surface flows

Ningbo Zhang, Shiqiang Yan, Qingwei Ma *, Qian Li

School of Science & Technology, City, University of London, London, United Kingdom

ARTICLE INFO

Keywords:
ISPH
PPE
Graph neural network (GNN)
Free-surface flows
Generalization

ABSTRACT

The incompressible Smoothed Particle Hydrodynamics (ISPH) is a popular Lagrangian Particle method. In the conventional ISPH method for simulating free-surface flows, the pressure-projection phase, which solves the pressure Poisson's equation (PPE), is the most time-consuming. In this paper, we propose a novel hybrid method by combining the graph neural network (GNN) with the ISPH for modelling the free-surface flows. In the new hybrid method, the graph neural network (GNN) is employed to replace solving the PPE for pressure in the conventional ISPH. To the best of knowledge of the authors, this is the first attempt to combine the GNN with ISPH model in a Lagrangian formulation. The performance of the hybrid method will be evaluated by comparing its results with experimental data, analytical solution or numerical results from other methods for three benchmark test cases: dam breaking, sloshing wave and solitary wave propagation. In addition, the potential of generalization of the hybrid method will be studied by applying it with the GNN model trained on data for relatively simple cases to simulate more complex cases. It will be demonstrated that the hybrid method does not only give satisfactory results, but also shows good potential of generalization. In addition, the new method will be demonstrated to require the computation time which can be 80 times less than the conventional ISPH for estimating pressure for cases with a large number of particles that is usually needed in the practical free-surface flows simulation using ISPH.

1. Introduction

The Smoothed Particle Hydrodynamics (SPH) [1,2] is a Lagrangian meshless method that discretizes the computational domain by particles that carry physical field variables (e.g. the pressure, density and velocity) and move at their material velocity. The incompressible SPH (ISPH) is an important stream of the SPH. It solves the incompressible Navier-Stokes (NS) equation and continuity equation using the projection method, where the pressure is evaluated by solving a pressure Poisson Equation (PPE). Recently, ISPH has been widely applied to modelling the free-surface flows [3–9], producing stable and accurate pressure fields and securing good volume conservation properties [10]. Its promising superiority has been demonstrated over the conventional mesh-based method for modelling the free-surface flows, especially the breaking wave impact on structures. The accuracy and convergence of the ISPH are largely affected by the numerical schemes to solve the PPE. The reviews about this issue can be found in [8,11]. Despite intensive research and a significant progress on developing new numerical schemes in order to improve the consistency, convergence and adaptivity of the ISPH, solving the PPE remains to be the most time-consuming part in the ISPH.

Recently, the machine learning (ML) techniques have been widely used in the fluid dynamics. For example, instead of the conventional computational fluid dynamics (CFD) approaches, which solve the governing equation directly, the data-driven approach using ML is emerging in the fluid simulation [12], e.g. learning the pressure and velocity field from data [13]. The application of ML in another direction, which is more relevant to this paper, is to use the ML techniques to accelerate the numerical process through replacing the challenging and/or time-consuming execution part in the conventional procedure by appropriate ML models. One typical example is the ML accelerated turbulence modelling, in which an appropriate ML model is trained using the data produced from the large eddy simulation (LES) or the direct numerical simulation (DNS) and adopted to replace the traditional turbulence models for predicting the turbulent production term [14] or Reynolds stress anisotropy tensor [15]. According to their comparative study with the DNS or LES, the ML supported RANS simulation can, not only accelerate computational process significantly,

* Corresponding author.
E-mail address: q.ma@city.ac.uk (Q. Ma).

https://doi.org/10.1016/j.cpc.2024.109220
Received 29 July 2023; Received in revised form 18 April 2024; Accepted 24 April 2024
Available online 28 April 2024

but also largely reproduce the turbulence feature.

Attempts have also been made to use the artificial neural network to predict the fluid pressure to replace solving the PPE in the Navier-Stokes [16,17] or Euler equations [18,19] using the projection method. In these applications, the most widely used approach is to use the convolutional neural networks (CNNs) to replace solving the PPE. In this regard, Tompson et al. [18] proposed the CNN architecture to handle the pressure projection in Eulerian-based solvers for accelerating Eulerian fluid simulation; Xiao et al. [17] developed a CNN-based deep learning model, which uses the discretization structure and the intermediate velocities as the inputs, to predict the solutions of the large linear equations resulted from the discretization of the PPE; Dong et al. [19] proposed a CNN-based framework (Smartfluidnet) that automatically uses multiple neural network models to accelerate Eulerian fluid simulation.

While CNNs have achieved numerous successes in applications, they have inherent limitations and pose challenges in dealing with physical problems involving data represented on a set of nodes (particles) which are irregularly distributed. That is because the convolutions in CNNs have to be performed on a uniform Cartesian grid and, therefore, require input (such as the domain geometry, field data) to be expressed in this way. Taking the Lagrangian ISPH simulation of water waves as the example, particles are moving following their material velocities and may be artificially shifted for achieving desired particle distribution [20,21]. Therefore, the particle distribution will become irregular even if it is regular at the start of the simulation. To deal with such data associated with particles which are irregularly distributed, CNN cannot be used directly. To use it, the data must be mapped on regular grids before used for training CNN models. During the prediction stage using the trained CNN modes, all the inputs must be transferred onto regular grids and the predicted data by CNN need to be transferred back to the irregular distributed particles. For example, Zhang et al. [22] used a structured background mesh to convert the field data corresponding to irregularly distributed particles in ISPH into the structured data at the node of the background mesh by interpolation. These structured data are then utilized for training the CNN. When the trained CNN model is implemented for predicting the pressure, the input data, such as velocity fields, from ISPH are also needed to be mapped on the background grids; after the pressure at all nodes of the background grid are predicted using the trained CNN model, the interpolation is employed again to give the pressure at the ISPH particles. Although the combination of CNN with ISPH method (ISPH_CNN) can be achieved through the above approach, two interpolations take additional computational costs and have adverse effects on the accuracy of the simulation results. In addition, it is challenging to implement structured grids and the interpolations for the cases with boundaries and/or objects of complex geometries.

Graph neural networks (GNNs) are based on graphs which are formed by information at nodes which can be distributed in any way, regularly or irregularly. Therefore, GNNs do not bear the limitations of CNNs discussed above. As such, GNNs can offer key advantages for particle-based simulation, e.g., Lagrangian ISPH simulation, in which the data are usually represented at irregularly-distributed particles.

The GNNs [23,24] were developed and applied for network analysis, especially on internet data, and were trained not with the back-propagation algorithm, but with fixed point iteration in earlier stage. The gated graph sequence neural networks in [25] helped integrate more recent deep learning innovations into GNNs. Modern GNN models share fundamental properties with CNN models such as spatial invariance and locality. In recent years, the application of GNNs has grown very rapidly to the fields of science and engineering. For example, the data-driven approaches using the GNNs have been increasingly applied to the fluid simulation. Belbute-Peres et al. [26] developed a hybrid graph neural network that combines a traditional graph convolutional network with an embedded differentiable fluid dynamics simulator for fluid flow prediction. Pfaff et al. [27] proposed a mesh-based GNN to simulate incompressible flow around cylinders and compressible flow around airfoils, in which the fluid domain is discretized into an unstructured mesh using traditional meshing techniques. Chen et al. [28] employed a similar GNN as a surrogate model to infer the velocity and pressure fields for a steady incompressible flow around a bluff body. Gao and Jaiman [29] presented a quasi-monolithic GNN-based framework for data-driven reduced-order modeling of fluid-structure interaction, where a multi-layer perceptron was used to evolve mesh displacements. A data-driven framework including the multi-scale rotation-equivariant graph neural networks for learning the inference of unsteady Eulerian fluid dynamics defined on a fluid domain discretized using an unstructured set of fixed nodes was proposed in [30].

It is pointed out that there have been some works that use GNNs to model fluid dynamics simulations in Lagrangian particle (meshfree) methods. Sanchez-Gonzalez et al. [31] presented a machine learning framework with graph neural network for simulating fluids. Kumar and Vantassel [32] developed a Graph Network Simulator (GNS) to learn physics and predict the flow behavior of particulate and fluid systems, in which GNS discretizes the domain with nodes representing a collection of particles and the links connecting the nodes representing the local interaction between particles. Li and Farimani [33] proposed a data-driven model for fluid simulation under Lagrangian particle representation based on GNN. They mainly focused on the development of the GNN to speed up the simulations of flow of different materials. To the best of our knowledge, there is no work that combines the GNN with the projection based Lagrangian particle method, ISPH, for modelling the free-surface flows.

In this paper, we adopt the GNN model to replace solving the PPE in the conventional ISPH method. To this end, we combine the GNN model developed in [33] and built on the open-access library of PyTorch [34] with our in-house ISPH solver [7] to form a hybrid method, referred to as ISPH_GNN hereafter. To the best of knowledge of the authors, it is the first hybrid method combining ISPH with GNN. In this method, the GNN model is trained and implemented with appropriate selections of the input data that represent the typical features of the ISPH model, such as gravitational and viscous effects, boundary conditions and the Lagrangian nature of the particle movement, in order to achieve a satisfactory accuracy for free-surface flows. We apply the ISPH_GNN to three benchmark cases, i.e., the dam breaking, sloshing wave and solitary wave propagation, to demonstrate its accuracy and convergence performance by comparing its numerical results with the experimental data, analytical solution and results from other SPH simulations.

Another contribution of this paper is to demonstrate that the ISPH_GNN trained on data for relatively simple cases can be implemented for relatively complex cases with satisfactory outputs. To be convenient and to follow the convention of machine learning community, e.g. [35,36], this is phrased as ‘generalization’ in this paper. Such generalization capability is an important aspect of any ML models. There are some limited investigations on the generalization performance of GNN models based on data-driven approaches. For example, Sanchez-Gonzalez et al. [31] conducted the generalization experiments to evaluate their model on several conditions different from those for training, including different number of particles, different object shapes, different number of objects, different initial positions and velocities and longer trajectories. Li and Farimani [33] investigate the generalization capability of their data-driven model and showed that its performance under some cases beyond its training distributions was satisfactory. In these studies, only the visualization results for the generalization examination are demonstrated. In this paper, after the GNN in the hybrid method ISPH_GNN is trained on simple cases, the hybrid method will be employed to simulate the cases with more complex geometries and configurations. The assessments of its generalization performance in these cases will be carried out by comparing both the pressure field and wave surface profile with the corresponding results from other two ISPH models. Such comparison fills the gap of lack of quantitative assessments of the generalization performance for GNN methods in the cited studies

and will enrich the understanding on the generalization of the GNN. In addition, the computational efficiency of ISPH_GNN is also investigated.

2. Mathematical and numerical formulation of ISPH_GNN

2.1. Lagrangian ISPH

In the present ISPH, the fluid is assumed to be incompressible and is governed by the Navier-Stokes equation and the continuity equation that are respectively given as,

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} + \nu\nabla^2\mathbf{u} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}$, p and ρ are, respectively, the velocity, pressure and density of the fluid, $\mathbf{g}$ is the gravitational acceleration and ν is the kinematic viscosity of the fluid. Eq. (1) is written in a Lagrangian frame and D/Dt is the material derivative following the motion of fluid particles.

On the solid boundaries, the following Neumann boundary conditions for velocity and pressure, respectively, are imposed,

$$\mathbf{u}\cdot\mathbf{n} = \mathbf{U}\cdot\mathbf{n} \tag{3}$$

$$\mathbf{n}\cdot\nabla p = \rho(\mathbf{n}\cdot\mathbf{g} - \mathbf{n}\cdot\ddot{\mathbf{U}}), \tag{4}$$

in which $\mathbf{n}$ is the unit normal vector of the solid boundary; $\mathbf{U}$ and $\ddot{\mathbf{U}}$ are the velocity and acceleration of the solid boundary.

On the free surface, the following condition is imposed,

$$p = 0 \tag{5}$$

For completeness, the details of identifying the free-surface particles are given in Appendix A.

A projection method is used to solve the governing equations and boundary conditions, which is composed of three main steps:

(1) Prediction step: an intermediate temporal velocity and position of particle are predicted using

$$\mathbf{u}^* = \mathbf{u}_t + \Delta\mathbf{u}^* \tag{6}$$

$$\Delta\mathbf{u}^* = \left(\mathbf{g} + \nu\nabla^2\mathbf{u}\right)\Delta t \tag{7}$$

$$\mathbf{r}^* = \mathbf{r}_t + \mathbf{u}^*\Delta t \tag{8}$$

where $\mathbf{u}_t$ and $\mathbf{r}_t$ are the velocity and position of particle at the time t, respectively; Δt is the time step size; $\mathbf{u}^*$ and $\mathbf{r}^*$ denote the intermediate velocity and position of particle.

(2) Pressure projection: By substituting Eqs. (6) and (7) to Eq. (1) and (2), it is not difficult to derive the PPE, $\nabla^2 p_{t+\Delta t} = \frac{\rho\nabla\cdot\mathbf{u}^*}{\Delta t}$. Following the study in [5,37], the following alternative form [7–9] is applied to all internal fluid particles,

$$\nabla^2 p_{t+\Delta t} = \frac{\psi}{\Delta t} \tag{9}$$

where $\psi = \alpha\frac{\rho-\rho^*}{\Delta t} + (1-\alpha)\rho\nabla\cdot\mathbf{u}^*$ and α is the blending coefficient, which is taken as a small value such as 0.01. ρ^* is the particle density at the intermediate time step estimated by $\sum_{j=1}^{N} m_j W(\mathbf{r}_{ij}^*)$, where N is the number of the particles in the influence domain of the particle i, m_j is the particle mass of the local particle j and $W(\mathbf{r}_{ij}^*)$ is a kernel function. In this paper, the cubic B-spline kernel proposed by Monaghan and Lattanzio [38] is used for the kernel function.

(3) Correction step: After the pressure $p_{t+\Delta t}$ at time $t+\Delta t$ is predicted, the velocity change during the correction step is estimated by

$$\mathbf{u}^{**} = -\frac{\Delta t}{\rho}\nabla p_{t+\Delta t} \tag{10}$$

The velocity and position of particle at $t+\Delta t$ are then corrected using,

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}^* + \mathbf{u}^{**} \tag{11}$$

$$\mathbf{r}_{t+\Delta t} = \mathbf{r}_t + \frac{\mathbf{u}_t + \mathbf{u}_{t+\Delta t}}{2}\Delta t \tag{12}$$

Further details of the numerical implementations including the numerical discretisation of the PPE, the numerical schemes to deal with the boundary conditions, the gradient and the divergence operator in the conventional ISPH can be found in [8].

2.2. GNN model

According to the comprehensive survey on graph neural networks in [39], the graph neural network developed by Li and Farimani [33] and used in this paper can be classified as the Spatial Convolutional Graph Neural Networks (ConvGNNs). Analogous to the convolutional operation of CNN on an image defined on structured grids, the ConvGNNs perform convolutions on graphs defined by nodes and their spatial relations, which may be irregularly distributed spatially. In the following sections, we use GNN to refer to the graph neural network used in this paper for simplicity.

The GNN (Fig. 1) used in this paper includes three blocks: encoder, message-passing, and decoder, as described in [29–31]. In the GNN, graph convolutions can be considered as a message passing process in which information is passed from one node to another along edges directly.

The encoder is to convert the inputs (such as the velocity, pressure and the term ψ in PPE in this paper) into the node embeddings. These node embeddings can be used for graph convolutions in the next message passing step. Specifically, considering a particle i in Fig. 1, the input $\mathbf{f}_i^0$ is encoded into $\mathbf{h}_i^0$ via a learnable encoder *en()* based on the multilayer perceptron (MLP) with n layers. The MLP is the non-trivial neural network. Mathematically, the encoder *en()* and the corresponding node embedding can be expressed as:

$$en\left(\mathbf{f}_i^l\right) = \sigma\left(\mathbf{w}_l\mathbf{f}_i^{l-1} + b_l\right) \quad with\ l = 1, 2, 3...., n \tag{13a}$$

$$\mathbf{h}_i^0 = en\left(\mathbf{f}_i^n\right) \tag{13b}$$

where $\sigma()$ is a non-linear activation function, $\mathbf{w}_l$ and b_l are the learnable weight vector and bias term of the l-th layer in MLP; $\mathbf{f}_i^l$ is the node

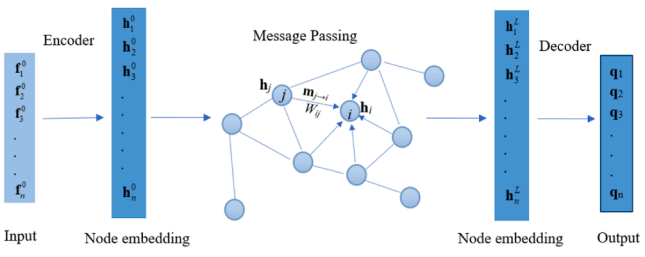


Fig. 1. Diagram of the GNN network architecture.

feature at the l-th layer in MLP. The node embedding, $\mathbf{h}_i^0$, is equal to the output of MLP as given in Eq. (13b), i.e., $\mathbf{f}_i^n$, which will be used as the input of graph convolution described below.

In the message passing block, the resulted node embedding is updated by using a recursive scheme. In this stage, the influence of a neighbouring node j on the node i, $\mathbf{m}_{j\to i}$, is firstly evaluated. Considering that the neighbouring influence $\mathbf{m}_{j\to i}$ depends on the relative location between the nodes in reality, a weight is applied. Following Li and Farimani [33], the neighbouring influence in the l-th layer convolution is given by

$$\mathbf{m}_{j\to i}^l = \mathbf{h}_j^{l-1} W\left(r_{ij}, r_0\right) \tag{14}$$

where $\mathbf{h}_j^{l-1}$ denotes the node embedding of neighbor node j in the l-th layer of convolution, and the weight is given by $W(r_{ij}, r_0) = \begin{cases} r_0/r_{ij} - 1 & 0 < r_{ij} \le r_0 \\ 0 & r_{ij} > r_0 \end{cases}$. r_0 is the radius of the influence domain, which is taken as a value of $1.7dx$ and dx is the initial particle spacing. The influence from all the neighbouring nodes is aggregated and given by

$$\mathbf{M}_i^l = \frac{\sum_{j=1}^N \mathbf{m}_{j\to i}^l}{\sum_{j=1}^N W\left(r_{ij}, r_0\right)} \tag{15}$$

In the l-th layer of message passing, the node embedding of the node i is calculated by

$$\mathbf{h}_i^l = \sigma\left(\mathbf{w}_l^{'} \mathbf{M}_i^l + b_l^{'} \mathbf{h}_i^{l-1}\right) \quad with\ l = 1, 2, ..., L \tag{16}$$

where $\mathbf{w}_l^{'}$ and $\boldsymbol{b}_l^{'}$ are the learnable weight vector and bias term in the l-th

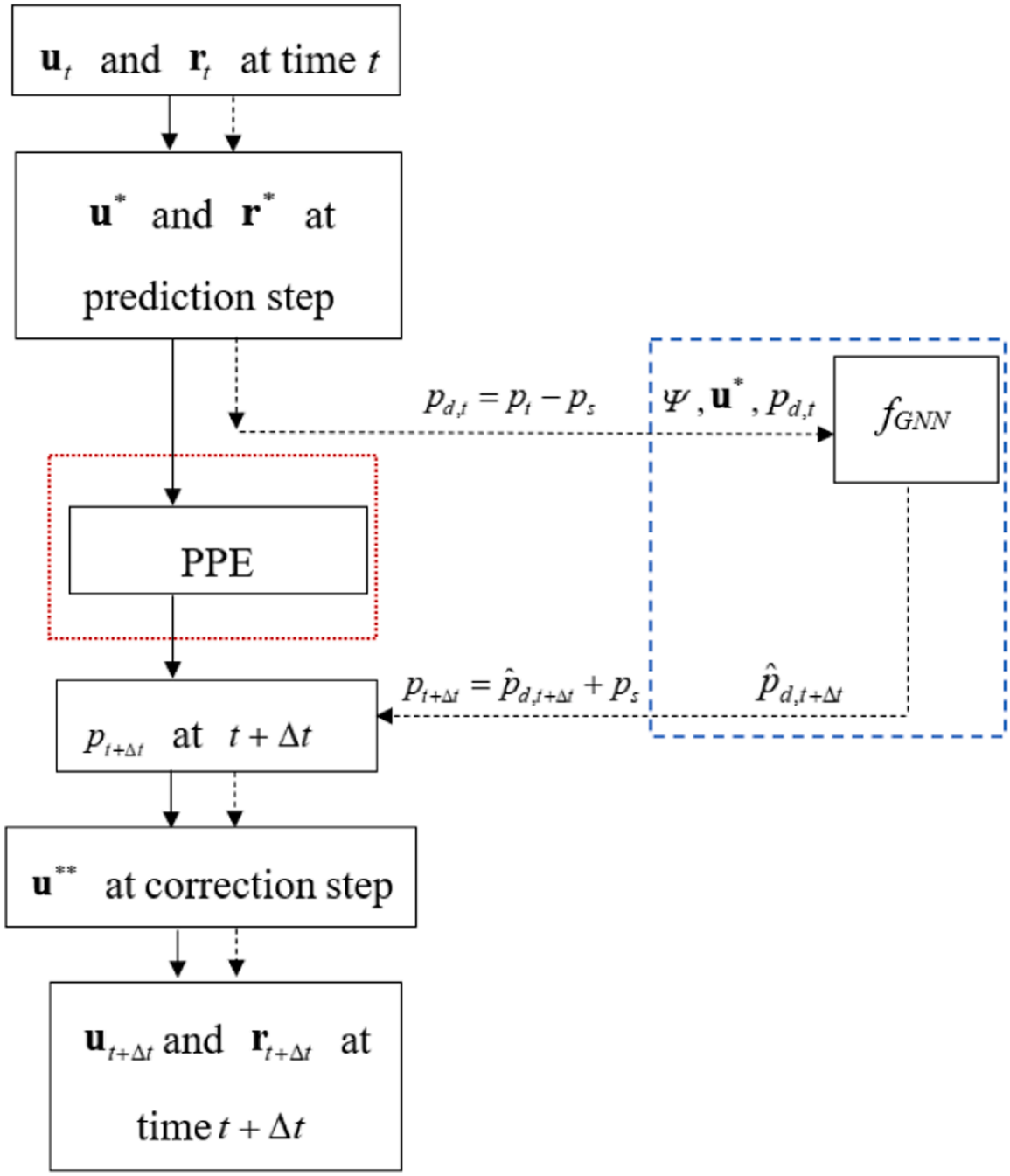


Fig. 2. Flowchart of ISPH incorporating the GNN (flowcharts of the conventional ISPH (solid arrows) and the ISPH_GNN (dashed arrows)).

layer of message passing.

The decoder aims to establish the relation between the node embeddings at the final layer of message passing and the output. More specifically, after L layers of recursive message passing, the output $\mathbf{q}_i$ (the pressure in this work) is generated by decoding the node embedding $\mathbf{h}_i^L$ of the final layer of message passing via a learnable decoder $em()$. The decoder is also based on the multi-layer perceptron (MLP) similar to that (Eq. (13) a) used in encoder but with different learnable weights and bias:

$$\mathbf{q}_i = em\left(\mathbf{h}_i^L\right) \tag{17}$$

where $\mathbf{q}_i$ represents the relation between the inputs and the pressure of each particle, which will be denoted by f_{GNN} below.

In this work, three layers in all the encoder/decoder MLP are used while the number of recursive message passing layers is chosen to be 2 based on the study of Li and Farimani [33] and our numerical tests to be shown in Section 3 and 4. We adopt Gaussian Error Linear UnitS (GeLUs) [40] as the non-linear activation function for all the layers.

2.3. Incorporating GNN with ISPH

As indicated above, in the ISPH_GNN, it is proposed to use the trained GNN model to replace the PPE for solving the pressure in the conventional ISPH. The flowchart of ISPH_GNN is demonstrated in Fig. 2. Some important issues related to combing ISPH with the GNN as well as training and implementing the GNN will be discussed in this section.

2.3.1. Input parameters

In order to achieve a satisfactory prediction accuracy, the inputs of the GNN model need to include the variables, which reflect the hydrodynamic characteristics. In this paper, a blended form (Eq. (9)) including the right hand side $\frac{\psi}{\Delta t}$ is used in the PPE of the ISPH model. Therefore, the ψ is adopted as one of the input parameters. In addition, the intermediate velocity, $\mathbf{u}^*$ can reflect the gravity and the viscous effects on the water wave. According to the study of combining ISPH with CNN model (ISPH_CNN) for free-surface flows modelling in [22], the better prediction performance can be obtained by including $\mathbf{u}^*$ as one of input parameters than without considering it. Therefore, we also use $\mathbf{u}^*$ as an input parameter for the GNN model here.

In addition, the study of Zhang et al. [22] has demonstrated that adoption of the pressure at the previous time step in the inputs can improve the prediction accuracy for the pressure at current step. Therefore, we also consider the pressure at the previous time step in the input parameters. However, it should be noted that in both the training and predictions of GNN, the total pressure p is split into two parts: the reference pressure, p_s, and the rest part of pressure, p_d, i.e., $p = p_d + p_s$. The reference pressure p_s can be directly calculated by $\rho g\tilde{z}$, where $\tilde{z}$ is the vertical distance from the fluid particle to its nearest free-surface particle, as explained in Appendix A. Based on the splitting, the GNN is trained using the data for p_d and also predicts the corresponding part of pressure $\widehat{p}_d$ after training. Based on our numerical tests, the overall results with the pressure predicted by the GNN trained in this way are significantly better than those obtained by using GNN to directly handle total pressure.

According to the above discussions, the field data of ψ, $\mathbf{u}^*$ and $p_{d,t} = p_t - p_s$ are adopted as the input parameters to train the GNN model for predicting the pressure $\widehat{p}_{d,t+\Delta t}$. The function f_{GNN} from input to output by the GNN model can be expressed as:

$$\widehat{p}_{d,t+\Delta t} = f_{GNN,IC}\left(\psi, \mathbf{u}^*, p_{d,t}, c_p\right) \tag{18a}$$

where c_p are the network coefficients, i.e., the weights and bias involved in Eqs. (13), (16) and (17), which are optimized during the training process as discussed in Section 2.2. In Eq. (18), IC is just used as the label representing the specific class of physical problems, such as IC = '*dam*' denoting dam breaking cases, and is not a parameter to be determined by the training.

In general, after the model is trained, the pressure $\widehat{p}_{d,t+\Delta t}$ can be predicted based on the input data of ψ, $\mathbf{u}^*$ and $p_{d,t}$ by the GNN model $f_{GNN,IC}$ with the optimized network coefficients c_p. After the prediction by the GNN model $f_{GNN,IC}$, the total pressure of fluid particles used in Eq. (1) for updating the velocity field can be obtained by

$$p_{t+\Delta t} = \widehat{p}_{d,t+\Delta t} + p_s \tag{18b}$$

It is noted that as the reference pressure is subtracted before training the GNN model and added back after prediction by the GNN model. The overall results should not be very sensitive to how it or $\tilde{z}$ is estimated. Other methods would be studied in future work.

2.3.2. Training data

The overall accuracy of f_{GNN} is also affected by the trained database. Similar to the study in ISPH_CNN [22] and inspired by the turbulence modelling [14,15] that use the higher-fidelity modelling solutions to build a model used by lower-fidelity RANS simulation, we also use the consistent second order ISPH (ISPH_CQ) [9] developed by the authors of this paper to produce the training database with higher accuracy than that from the conventional ISPH [7]. In the ISPH_CQ, the governing equations and boundary conditions are the same as the conventional ISPH, but the derivatives for Laplacian operator, velocity divergence and corresponding Neumann boundary condition and the pressure gradient are numerically approximated by the quadric semi-analytical finite difference interpolation scheme (QSFDI) [41]. More details about the ISPH_CQ can be found in [9,41].

For each numerical case used for generating the training database, a random selection of the initial test conditions will be used. For example, in the dam breaking problem, different values of the water column height and width will be selected randomly in a practical range to form a series of cases for the database. Each case will be run by using the ISPH_CQ and the required variables, such as velocity and pressure, will be calculated and recorded for the database.

2.3.3. Training and implementing the GNN

The GNN is trained on the training datasets described in above section to minimize the loss function. The loss function f_{obj} used in this work is the sum of squared L-2 norm of the error between the prediction and actual values of the pressure, which can be expressed as:

$$f_{obj} = \sum_i^N s_i\left(\widehat{p}_{d,i} - p_{d,i}\right)^2 \tag{19}$$

where $\widehat{p}_{d,i}$ and $p_{d,i}$ are the predicted and actual pressure from the training data at particle i, respectively.

In Eq. (19), s_i is a weight for considering the effects of boundaries. The problems dealt with in this paper usually have two kinds of boundary conditions: i.e. the free-surface and solid boundary conditions. The difference between the predicted and actual pressure on the free surface is not taken into account during the training process. When implementing the pressure prediction by GNN model, the pressure of the free-surface particle is enforced to be zero, ensuring the satisfaction of the free-surface boundary condition Eq. (5).

When considering the solid boundary condition which involves the normal derivative of pressure, i.e., Eq. (4), we found that the larger error would be introduced by directly imposing the conditions in the training and testing stages. To avoid this, we adopt the following strategy. During the training and testing stages of the GNN, only inner fluid particles are considered, ignoring the particles representing solid boundaries. The trained GNN model in such a way is only applied to the prediction of the pressures of the fluid particles, excluding the boundary particles. The pressure p_b of the boundary particle is obtained directly from the neighbouring fluid particles by [42],

$$p_b = \frac{\sum_{f\in fluid}^{M} p_f W_{bf} + (\mathbf{g} - \dot{\mathbf{U}})\cdot\sum_{f\in fluid}^{M}\rho_f(\mathbf{r}_f - \mathbf{r}_b)W_{bf}}{\sum_{f\in fluid}^{M} W_{bf}} \tag{20}$$

where p_f is the pressure of the fluid particle, ρ_f is the density of fluid, W_{bf} is a kernel weight function, M is the total number of fluid particles in the neighbourhood of the boundary particle, $\mathbf{r}_f$ and $\mathbf{r}_b$ are the position vector of fluid and boundary particles. However, the pressure near the boundaries may not be consistent because the different methods are employed for estimating the pressure on solid boundaries and on the fluid particles near the boundaries. In order to overcome this problem, more weight is applied to the fluid particles near the solid boundaries in Eq. (19). The weight term considering this and the free surface is proposed as

$$s_i = \begin{cases} 3 & d_b \le 3.0{\cdot}dx \\ 2 & 3.0{\cdot}dx < d_b \le 6.0{\cdot}dx \\ 1 & d_b > 6.0{\cdot}dx \\ 0 & \text{free surface} \end{cases} \tag{21}$$

which d_b is the distance from a particle to a boundary.

The treatment on the boundary conditions may not be perfect. However, it is quite effective as the results demonstrate below in Section 3 and 4. Other alternatives for dealing with the boundary conditions may be explored in our future work. After the model is trained, it will be implemented in the numerical practice of the ISPH_GNN as illustrated in Fig. 2.

3. Validation and convergence

In this section, three benchmark tests are used to evaluate the performance of the new method in terms of the prediction accuracy, convergence and mass (volume) and energy conservation. For the purpose of comparison, the conventional ISPH [7] and the ISPH_CQ [9] are also employed.

3.1. Dam-break flow

As a classical validation case for Lagrangian fluid simulations, the dam breaking is first considered, in which a rectangular column of water with the width l and the height h is confined in a water tank with a horizontal length of L, as shown in Fig. 3. A pressure sensor P_1 with a vertical distance h_1 from the bottom is located on the right wall of the tank.

A series of ISPH_CQ simulations for the dam breaking cases as shown in Fig. 3 have been carried out to generate the training/testing data including a training set of 50 cases and a test set of 30 cases to train the GNN model $f_{GNN,dam}(\psi,\mathbf{u}^*,p_{d,t},c_p)$. In these cases, the lengths and heights of the water column are randomly chosen in the ranges of $h/L = 0.2\sim0.5$ and $l/L = 0.2\sim0.5$ with $L = 2.0$ m. In the ISPH_CQ simulations, the initial particle size of 0.01 m and a time step size of 0.001 s are adopted.

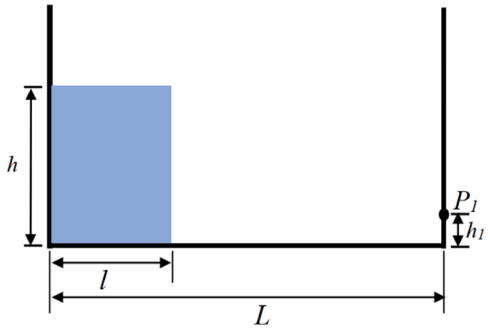


Fig. 3. Schematic sketch of dam break flow.

For each case/scene, the total 128 frames with 0.006 s apart, which can cover the entire process from the initial condition to the impact of dam breaking flow on the right side of the tank, are produced for the training data.

After the GNN model is trained, the ISPH_GNN is applied to the dam breaking case in [43], where $h = 0.37L$, $l = 0.37L$ and $L = 1.61$ m, which is different from any cases used for training. The initial particle spacing dx is taken as 0.01 m, i.e. $h/dx = 60$. The radius of the influence domain is taken as $r_0 = 1.5dx$. For this case and all other cases studied in this paper, the time step size dt is determined by $(dt/dx) = 0.1$, which has been tested to be appropriate in the conventional ISPH and ISPH_CQ. It is also worth noting that the trained $f_{GNN,dam}(\psi,\mathbf{u}^*,p_{d,t},c_p)$may not work well at the initial stage due to involvement of $\mathbf{u}^*$ and the pressure at the previous time steps. Therefore, the PPE solver is run during a short period from the beginning of the dam breaking until $\widetilde{t} = 0.8$ ($\widetilde{t} = t\sqrt{\frac{g}{h}}$) and the GNN model is switched on for predicting pressure in the rest of the simulation. The particle distributions with the pressure contour by using different ISPH methods at two instants are given in Fig. 4. One can see that the pressure distribution predicted by the ISPH_GNN is very similar to that from ISPH_CQ. The free-surface profile and the formation of the breaking jet in the ISPH_GNN simulation are close to the corresponding results from the ISPH and ISPH_CQ. The comparisons of the pressure at P_1 with the height $h_1 = 0.05L$ from different ISPH methods and the experimental data of Lobovský et al. [43] are illustrated in Fig. 5. The result of the ISPH_GNN is very close to the result of the ISPH. Taking the experimental peak pressure as reference, the relative error of peak pressure defined by $Err_m = |p_e - p_n|)/p_e$ where p_n is the peak pressure from numerical result and p_e is the experimental peak pressure, is about 4.9% for ISPH_GNN, 4.7% for ISPH and 4.1% for ISPH_CQ.

3.2. Sloshing waves in a moving tank

The next cases considered are about liquid sloshing in an oscillating tank. The schematic diagram of the tank is illustrated in Fig. 6, where L is the tank length, D is its height, and d is the mean water depth. The tank is subjected to a periodic sway motion with its displacement being specified by $X_s = a\sin(\omega t)$, where a and ω are the amplitude and the frequency of the motion, respectively.

For these cases, the ISPH_CQ is also used to generate a training set of 40 cases and a test set of 20 cases for training the GNN model $f_{GNN,slo}(\psi,\mathbf{u}^*,p_{d,t},c_p)$. In these cases, $d = 0.5L$, $D = 1.0L$ and $L = 1.0$ m are selected. The amplitude a are randomly selected ranging from $0.3L$ to $0.7L$, whereas different values of $\omega = 0.6\omega_1$, $\omega = 0.8\omega_1$ and $\omega = 0.9\omega_1$ are assigned for$\omega$with $\omega_1$being the first resonant frequency of the sloshing. These conditions fall into the valid range of the analytical solution given by Faltinsen [44] and Wu et, al. [45]:

$$\zeta = \frac{a}{g}\left(x\omega^2 + \sum_{n=0}^{\infty} C_n\omega\sin k_n x\right)\sin\omega t - \frac{a}{g}\sum_{n=0}^{\infty}\omega_n\left(C_n + \frac{H_n}{\omega^2}\right)\sin k_n x\sin\omega t \tag{22}$$

where ζ is the free-surface elevation measured from the mean water surface, $k_n = \frac{2n+1}{L}\pi, \omega_n = \sqrt{gk_n\tanh k_n d}, H_n = \omega^3\frac{1}{L}\frac{4(-1)^n}{k_n^2}andC_n = \frac{H^n}{\omega_n^2-\omega^2}$. The initial particle size of 0.01 m and a time step size of 0.001 s are adopted in the ISPH_CQ simulations for generating the training datasets. For each case/scene, the total 700 frames of data with the time interval of 0.01 s are used in training and testing.

The ISPH_GNN with the trained model $f_{GNN,slo}(\psi,\mathbf{u}^*,p_{d,t},c_p)$ is used to study the case, where $L = 1.0$ m, $a = 0.01L$ and $\omega = 0.8\omega_1$. The particle spacing $dx = 0.01$ m, with the time step determined in the same way as mentioned above, i.e., $dt/dx = 0.1$. Fig. 7 illustrates the particle distribution and the pressure contour from different ISPH methods at two different instants $\widetilde{t}$ ($\widetilde{t} = t\sqrt{\frac{g}{L}}$) corresponding to the occurrences of the highest water surface elevations on the right and left sides of the tank,

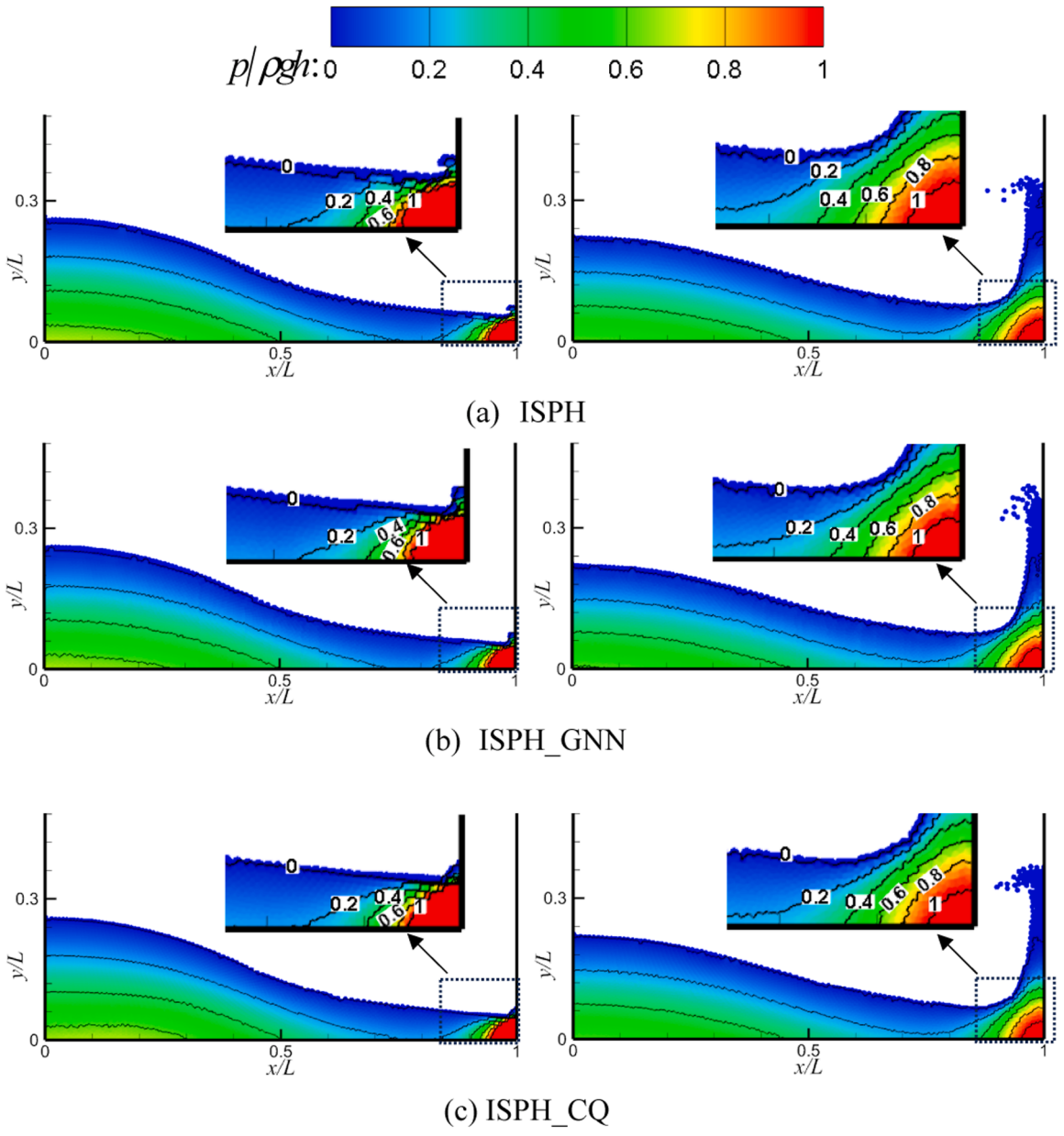


Fig. 4. Particle distributions with pressure contours at $\tilde{t} = 1.62$ (left column) and $\tilde{t} = 2.02$ (right column) obtained from different ISPH methods ($h = 0.37L$, $l = 0.37L$ and $L = 1.61$ m).

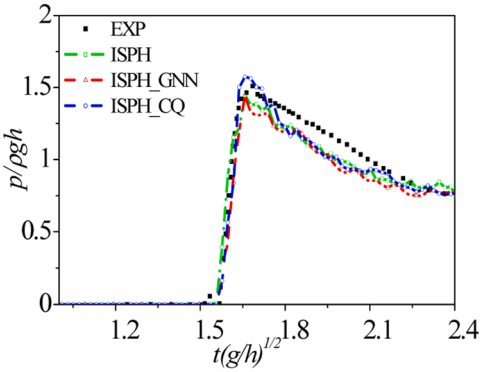


Fig. 5. Time histories of the pressure at P_1 from different ISPH models ($h = 0.37L$, $l = 0.37L$ and $L = 1.61$ m; $h_1 = 0.05L$).

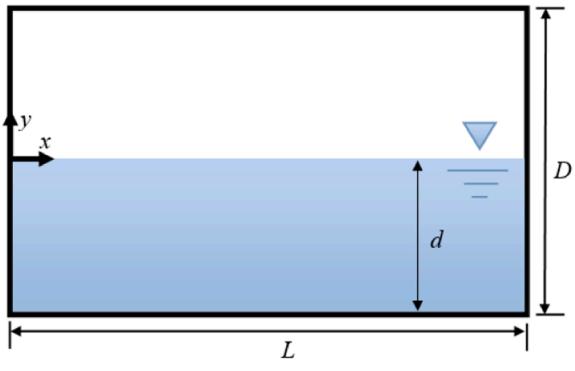


Fig. 6. Sketch of the sloshing tank.

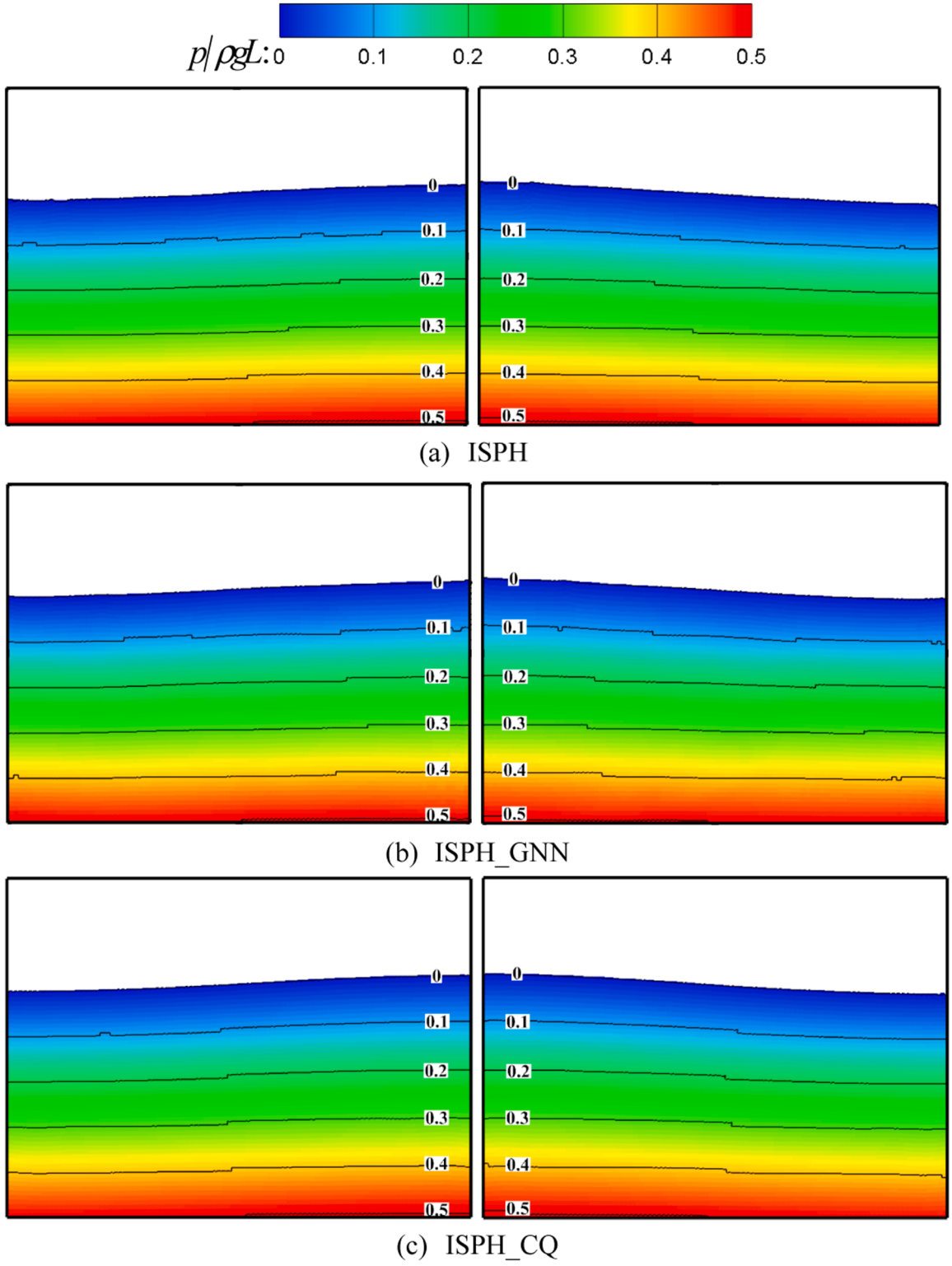


Fig. 7. The particle distributions and pressure contour from different ISPH models at different time instants: $\widetilde{t}$= 25.06 (left column) and $\widetilde{t}$= 26.62 (right column).

respectively in this case. As can be seen that the ISPH_GNN can produce a smooth pressure field similar to that from ISPH_CQ.

Fig. 8 compares the time histories of the surface elevations recorded at the left wall of the sloshing tank in the case corresponding to Fig. 7. The result predicted by ISPH_GNN shows good agreement with that from the ISPH_CQ and the analytical solution, although some differences are visible from the analytical solution, particularly during $\widetilde{t}$= 16 to 22 and 33 to 37. In order to investigate the convergence behaviours of the ISPH_GNN model, Fig. 9 gives the corresponding surface elevations obtained by ISPH_GNN model with different initial particle spacings ranging from 0.01 m to 0.02 m, i.e. $L/dx = 100$ to 50. It can be seen that the numerical results become closer to the analytical solution with reducing the initial particle spacing. The errors of numerical results are shown in Fig. 10, in which the error is computed by

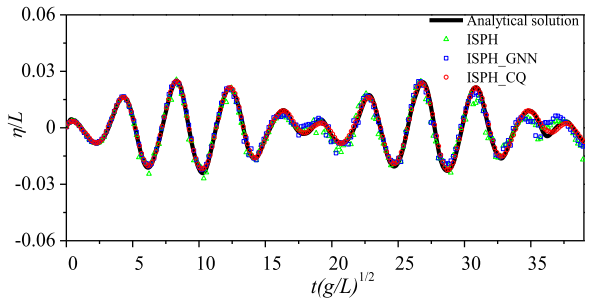


Fig. 8. The comparison of time histories of the surface elevations recorded at the left wall of the sloshing tank ($L = 1.0$ m, $a = 0.01L$ and $\omega = 0.8\omega_1$).

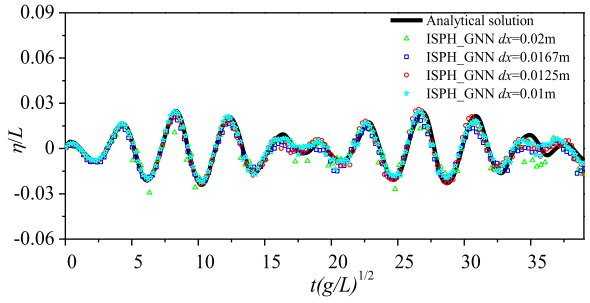


Fig. 9. The comparison of time histories of the surface elevations recorded at the left wall of the sloshing tank with different particle resolutions ($L = 1.0$ m, $a = 0.01L$ and $\omega = 0.8\omega_1$).

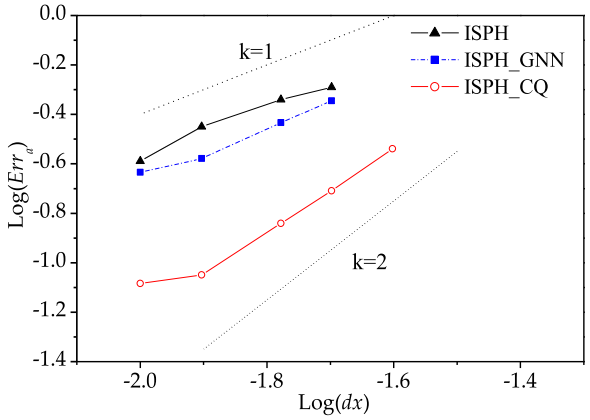


Fig. 10. Errors of numerical results in the sloshing case with different particle resolutions.

$$Err_a = \frac{\sqrt{\sum_{i=1}^{N}\left(\eta_{i,n} - \eta_{i,f}\right)^2}}{\sqrt{\sum_{i=1}^{N}{\eta_{i,f}}^2}} \tag{23}$$

where $\eta_{i,n}$ is the surface elevation at different time steps from the ISPH_GNN and the $\eta_{i,f}$ is the corresponding result from the analytical solution, N denotes the total number of time steps in the duration of simulation from $\widetilde{t} = 0.0$ to 39.2. For convenience, two dashed lines representing first-order ($k = 1$) and second-order ($k = 2$) convergence rates are included in the figure. It can be seen in Fig. 10 that the error of the ISPH_GNN reduces linearly as the particle size decreases, which demonstrates the linear convergence similar to that in the conventional ISPH. One may also observe from Fig. 10 that the error of the ISPH_GNN is lower than the conventional ISPH, implying that the present ISPH_GNN is slightly more accurate than the conventional ISPH if the same particle spacing is used.

It should be noted that the GNN model $f_{GNN,slo}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$ is trained only on data for small amplitude sloshing cases. Correspondingly, a small amplitude sloshing example is used for the verification of ISPH_GNN. The capacity of ISPH_GNN with the trained model for simulating violent sloshing problems need to be confirmed using more complex violent cases. In future work, more violent sloshing cases, including broken sloshing waves may be used to generate training data to be added to the current training datasets for training the GNN model, which may improve the generalization and accuracy of the trained model for simulating different violent sloshing problems.

3.3. Solitary wave propagation

Further attention is paid to the solitary wave propagation as sketched in Fig. 11, where h, d and L are the solitary wave height, the mean water depth and the length of the tank, respectively. The solitary wave is generated by the wavemaker using the approach in Ma and Zhou [37].

In these cases, another training set of 40 cases and a test set of 20 cases generated by the ISPH_CQ are used to train the GNN model $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$. The cases are randomly selected with the wave height h ranging from $0.2d$ to $0.4d$, whereas water depth d is adopted to be 0.25 m, 0.275 m and 0.3 m, respectively. The length of the tank $L = 40d$ is used in these cases. For each case/scene, the total 256 frames with the time interval of 0.0125 s are generated. The initial particle size of 0.0125 m and the time step size of 0.00125 s are used in ISPH_CQ to produce the training dataset.

To validate the prediction accuracy, we use the ISPH_GNN model incorporating with the trained $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$ to model the solitary wave propagation with $h = 0.28d$. For comparison, the corresponding free-surface profiles at $\widetilde{t} = 20$ ($\widetilde{t} = t\sqrt{\frac{g}{d}}$) from different ISPH results including the ISPH_CNN [22] with initial particle spacing 0.01 m, i.e. $dx = d/25$, are shown in Fig. 12. It is noted that the ISPH_CQ result agrees well with the Boussinesq solution [46] as confirmed by Zhang et al. [9]. This figure also confirms a good agreement between the ISPH_GNN result and the Boussinesq solution and the ISPH_CQ result. In contrast, the ISPH_CNN result exhibits larger difference from the Boussinesq and ISPH_CQ result, which is also reflected in the following error analysis in Fig. 14.

To further show the convergence behaviours of the ISPH_GNN model, Fig. 13 gives the wave profiles at $\widetilde{t} = 20$ obtained by ISPH_GNN model with different initial particle spacings ranging from 0.01 m to 0.025 m i.e. $dx = d/25$ to $d/10$. As can be found in Fig. 13 again that the numerical results become closer to the analytical solution with reducing the initial particle spacing. The corresponding errors of results from all the methods mentioned in the paper for this case with different particle spacing are plotted in Fig. 14. The error is computed in the same way as that used in Eq. (23) but $\eta_{i,n}$ and $\eta_{i,f}$ are replaced by the wave elevation at i-th particle obtained from the ISPH_GNN simulation and the analytical solution for this case; N is the total number of particles in the subdomain from $x/d = 8.8$ to $x/d = 20.0$ in Fig. 12. Overall, the ISPH_GNN has a similar convergence property to the conventional ISPH and ISPH_CNN, but it has a slightly higher accuracy than the conventional ISPH and ISPH_CNN, and its accuracy is closer to that of ISPH_CQ.

To further assess the performance of the numerical method, the

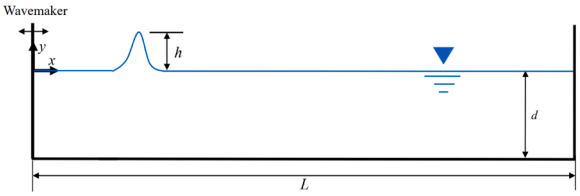


Fig. 11. Schematic wave tank for solitary wave.

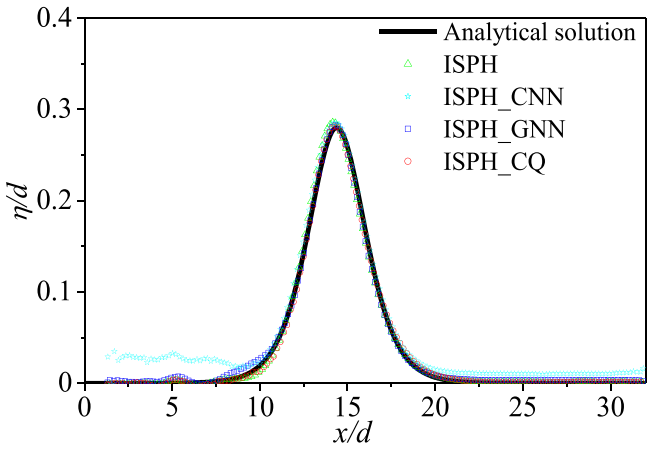


Fig. 12. Comparisons of solitary wave profiles between analytical solution and different numerical results at $\widetilde{t}$= 20 in the case with $h = 0.28d$.

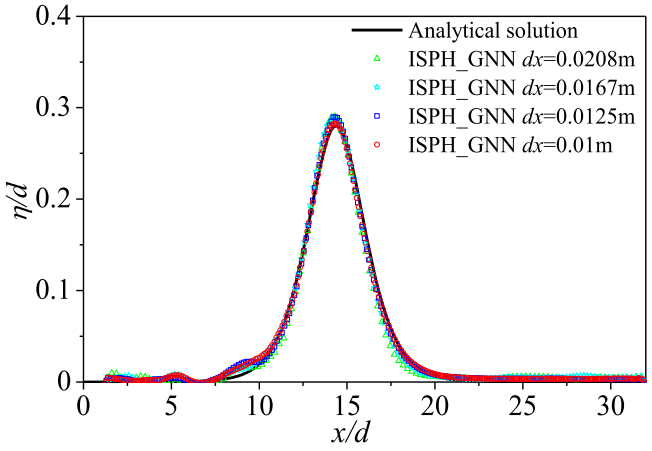


Fig. 13. Comparisons of wave profiles at $\widetilde{t}$= 20 between analytical solution and ISPH_GNN results corresponding to different initial particle spacings in the case with $h = 0.28d$.

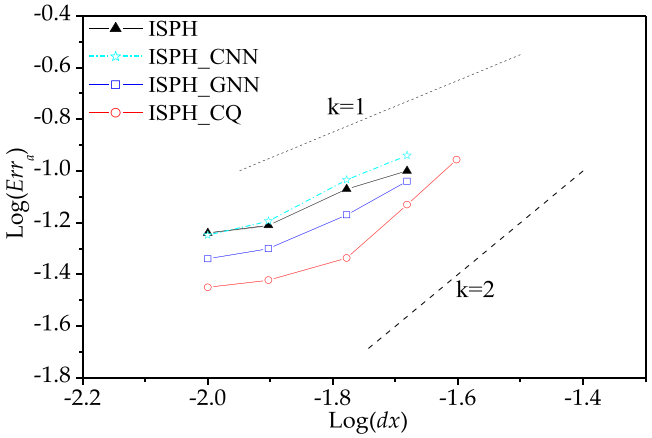


Fig. 14. Averaged errors Err_a of numerical results corresponding to different particle spacing in the solitary wave propagation ($h = 0.28d$).

mass/volume and energy conservation are also studied. Fig. 15 gives the time histories of the fluid volume in the case corresponding to Fig. 12. It can be seen that the ISPH_GNN result agrees well with the theoretical value. Compared with the theoretical value from $\widetilde{t} = 0.0$ to $\widetilde{t} = 31.3$, the errors of the fluid volume are about 0.2% for ISPH_GNN, 0.16% for ISPH

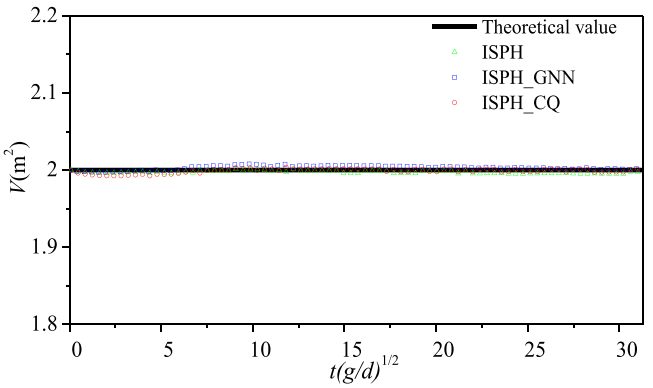


Fig. 15. Time histories of the volume of the fluid in the case of solitary wave ($h = 0.28d$ and $dx = 0.0125$ m, i.e. $dx = d/20$).

and 0.14% for ISPH_CQ in Fig 15. In addition, Fig. 16 depicts the time histories of the total energy in the ISPH_GNN simulation for the same case, in which the good reservation of the total energy of fluid after the wavemaker stops can be observed. Compared with the ISPH_CQ result from $\widetilde{t} = 12.5$ to $\widetilde{t} = 31.3$ in Fig 16, the errors of the total energy are about 0.048% for ISPH_GNN and 0.046% for ISPH, respectively.

4. Demonstration of generalization and computational efficiency of ISPH_GNN

4.1. Generalization

The generalization capability is a notable issue in assessing the performance of the machine learning model. After the validation for the ISPH_GNN, we further investigate the potential of generalization of the model by using the trained model to study more complex scenarios, beyond the scenarios of training and testing data. The results from ISPH_CNN [22] will be employed for comparison in some cases.

Firstly, to demonstrate the generalization performance of the numerical model ISPH_GNN with $f_{GNN,dam}(\psi, \mathbf{u}^*, p_{d.t}, c_p)$, the case with the configuration similar to Fig. 3 but with a semicircle located in the middle of bottom is considered. In this case, the distance from the center of semicircle to the left wall is $L = 2.0$ m while the water column hgiht $h = 0.5L$ and its length $l = 0.25L$ are used. The initial particle size dx is taken as $h/dx = 100$, and the time step is still determined by dt/dx=0.1. Fig. 17 shows the spatial distribution of pressure obtained from different ISPH models at two instants, corresponding to the occurrences of the violent wave impacts on the semicircle. It can be clearly seen that there is unreasonable pressure distribution before the semicircle in ISPH_CNN results. In contrast, the pressure distribution from the ISPH_GNN is very

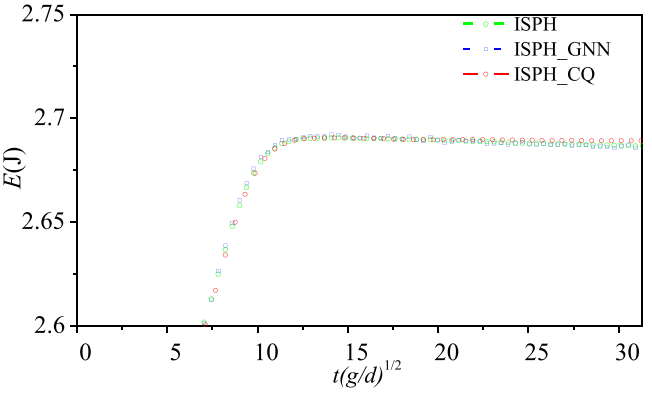


Fig. 16. Time histories of the total energy in the case with solitary wave propagation ($h = 0.28d$).

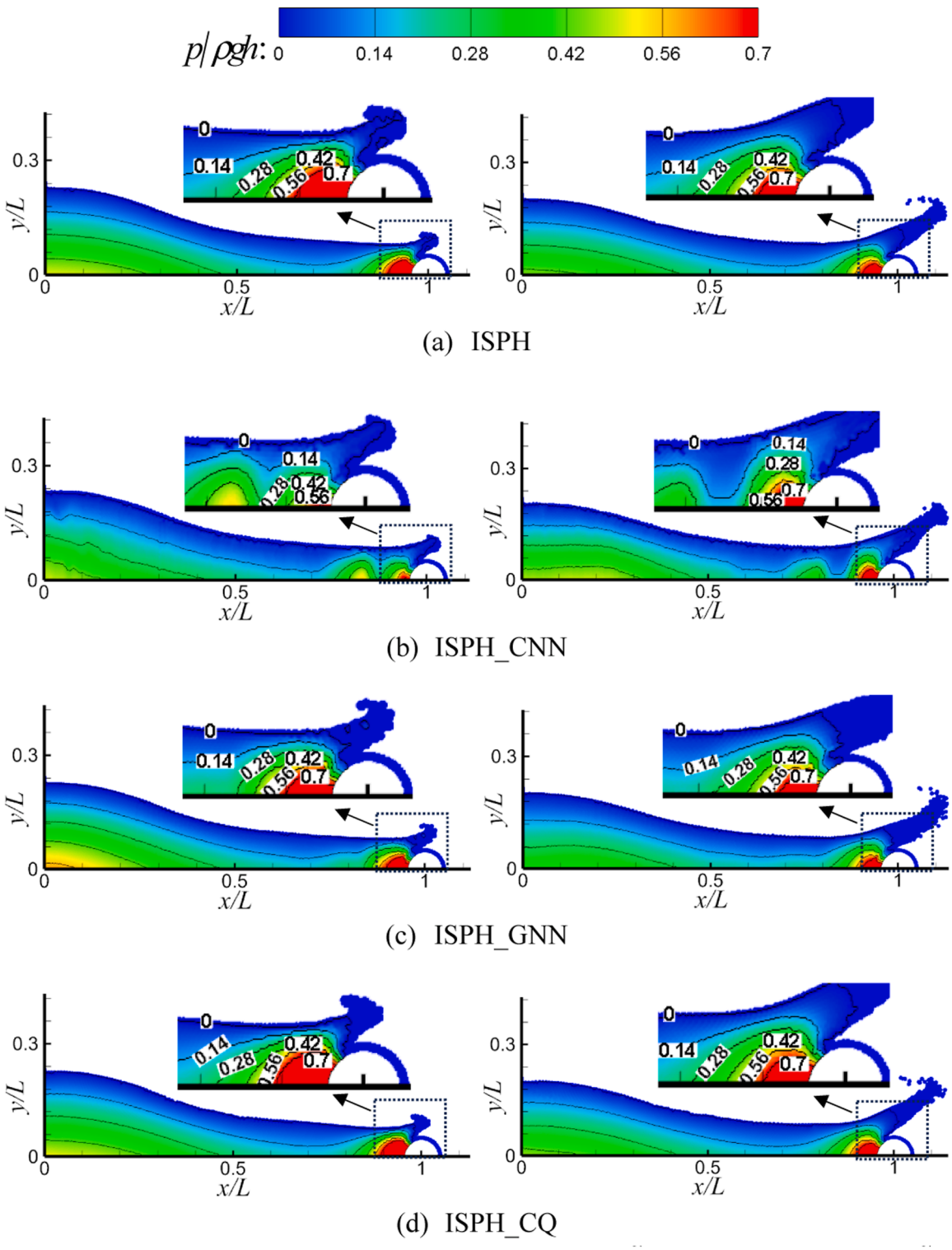


Fig. 17. Particle distributions with pressure contours at $\tilde{t}$= 1.0 (left column) and $\tilde{t}$ = 1.1 (right column) obtained by using different ISPH models in the case of dam breaking over a semicircle.

close to these of ISPH and ISPH_CQ results. This comparison implies that the ISPH_GNN has better potential of generalization than the ISPH_CNN. One of the possible reasons is that the CNN model needs structural background mesh, which could not conform to the semicircle surface. Fig. 18 displays the comparison of the wave front and water column height of dam breaking flow computed by different ISPH methods, in which the wave front is represented by the coordinate value in the *x*-direction of the particle at the forefront of the water flow and the water column height is represented by the coordinate value in the *y*-direction of the particle at the highest position of the water flow. It is observed that the ISPH_GNN results are almost identical to the corresponding ISPH_CQ results. Taking the ISPH_CQ result from $\tilde{t}$= 0.0 to $\tilde{t}$ = 1.3 in Fig 18 as the reference, the errors of the wave front in Fig. 18(a) are about 0.77%, 0.59% and 0.58% for ISPH_CNN, ISPH_GNN and ISPH,

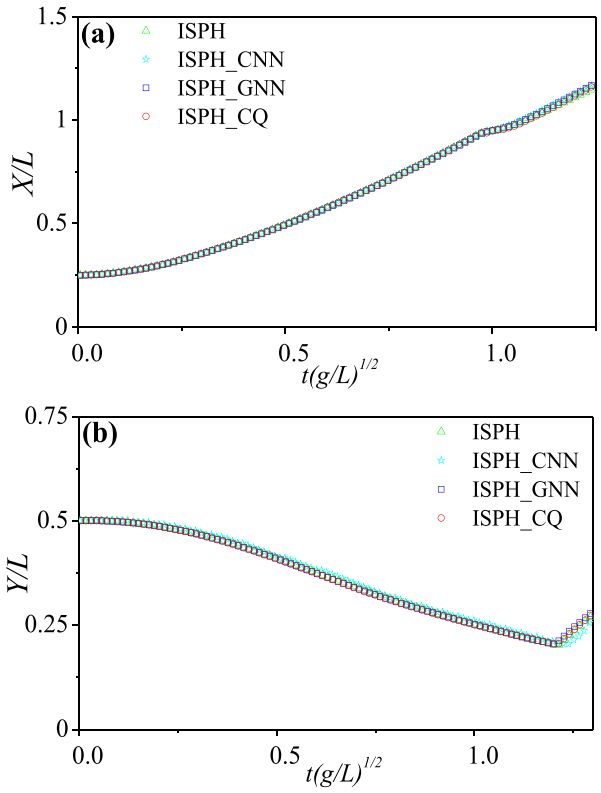


Fig. 18. Comparisons of dam breaking flow (a) water front; and (b) water column height.

respectively, while the errors of the water column height are 2.3% for ISPH_CNN, 0.41% for ISPH_GNN, and 0.43% for ISPH in Fig. 18(b). It can be seen that the error of ISPH_CNN are larger than that of ISPH_GNN, especially for the water column height in Fig. 18(b).

Secondly, the case of sloshing with a vertical baffle in the middle of the tank shown in Fig. 6 is used here to demonstrate the generalization performance of the ISPH_GNN with $f_{GNN,slo}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$. The flow condition is same to that in Fig. 10 with $a = 0.01L$ and $\omega = 0.8\omega_1$. The height and width of the baffle are $0.2L$ and $0.04L$, respectively. Fig 19 denotes the snapshots of the liquid motion including the particle distribution and the pressure contour at different instants, where the particle spacing is $L/dx = 100$ with the time step determined in the same above. From Fig. 19, the ISPH_GNN method can provide quite smooth and consistent pressure fields particularly near the baffle with those of ISPH and ISPH_CQ. Correspondingly, the free surface of the whole sloshing tank at different instants calculated by different ISPH methods in this case are given in Fig. 20. Based on the reference data from the ISPH_CQ result in Fig 20, the errors of the free surface are about 4.6% for ISPH_GNN and 6.7% for ISPH in Fig. 20(a), while the errors are about 4.4% for ISPH_GNN, and 6.9% for ISPH in Fig. 20(b).

Further, the ISPH_GNN with $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$ is used to simulate the solitary wave propagation on a slope with angle of 150°. For this purpose, a solitary wave with wave amplitude $h = 0.24d$ and the water depth $d = 0.25$ m is studied. The initial particle spacing is taken as 0.01 m, i.e. $dx = d/25$. Fig. 21 illustrates the process of solitary wave running up and down the slope at different times computed by ISPH_GNN. The wave profiles predicted using the ISPH_CQ are also plotted together (orange line on the free surface). It can be seen from Fig. 21(a) that the wave front reaches its maximum climbing point. Then the run-down process starts and the main flow retreats from the slope as shown in Fig. 21(c). Generally, the agreement of free surfaces between ISPH_GNN and ISPH_CQ is satisfactory. Fig. 22 gives the comparisons of wave profiles at $\tilde{t}$= 32.6 from different ISPH models. Taking the ISPH_CQ result from $x/d = 0$ to $x/d = 26$ in Fig. 22 as the reference, the errors are about 2.9% for ISPH_GNN and 2.6% for ISPH.

To further explore the generalization performance of the trained $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$, a case of solitary wave propagation over a vertical baffle is also studied by ISPH_GNN with $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$ here. The same conditions as in Fig. 21 are considered here. The snapshots for the pressure field during the wave propagation over the vertical baffle computed by ISPH_GNN are given in Fig. 23. It can be seen that the ISPH_GNN with $f_{GNN,sol}(\psi, \mathbf{u}^*, p_{d,t}, c_p)$ can provide quite smooth and consistent pressure fields near the baffle. In addition, the comparisons of wave profiles from different ISPH models are also shown in Fig. 24 and the agreement is generally acceptable. Compared with the corresponding ISPH_CQ result from $x/d = 8$ to $x/d = 18$ in Fig 24, the errors of the wave profile for ISPH_GNN and ISPH are about 3.5% and 2.4% respectively.

Overall, the above investigations show that the ISPH_GNN with the GNN model trained on the data for simply configurations can be applied to simulate the cases with more complex configurations and produce satisfactory results. Although the more extensive studies are needed, this paper shed some light on the good potential of generalization of the present ISPH_GNN.

4.2. Computational efficiency

In addition to the accuracy and generalization potential demonstrated above, it may be worth to show the potential computational efficiency of ISPH_GNN. The dominated computational time used by ISPH is that taken by finding the solution of pressure. Therefore, in this section, the computational efficiency is demonstrated by comparing the computational time of ISPH_GNN with that of ISPH for finding the solution of pressure. Fig. 25(a) plots the averaged computational time taken for pressure prediction per time step by the ISPH_GNN with that in the ISPH model for the case of solitary wave with different computational parameters given in Table 1. The speedup ratio, which is obtained by dividing the corresponding averaged computational time per step required for solving PPE in the conventional ISPH by the averaged computational time of the GNN model for predicting the pressure, is given in Fig. 25(b). All these cases are run on a workstation with the main specification as follows: Intel Xeon Platinum 8268 CPU at 2.9 GHz, 256 GB RAM and NVIDIA GeForce RTX 3090 with 24 GB RAM. As can be seen, the pressure prediction in the ISPH can be significantly speeded up if using the trained GNN model to replace the solution of the PPE. With the growth of particle numbers, the time savings or speed up ratio increase strikingly. For example, at 800k particles, the ISPH_GNN can achieve about 85 times faster for the pressure prediction than the conventional ISPH. This indicates that the ISPH_GNN method has potential to be much more efficient than the traditional ISPH. For the purpose of comparison, the corresponding speedup ratio for the ISPH_CNN, taken from [22], is also plotted together in Fig. 25(b), for which all the cases for solving PPE and implementing ISPH_CNN used for the calculation of speedup ratio were run on an older machine with the specification of Intel Xeon E5–2667 CPU at 3.3 GHz, 16.0 GB RAM and NVIDIA GeForce GTX 1070 with 8 GB RAM. The ISPH_CNN can also be much faster than the conventional ISPH, but the speedup ratio is just at about 8 times at 800k particles. Although these two models are run on different workstations, it is reasonable to conclude that the ISPH_GNN method perform better in computational efficiency than the ISPH_CNN. That is because the ISPH_CNN method requires two interpolation operations which are necessary to achieve the conversion between particle data in ISPH and grid-based data in CNN.

It should be noted that the computational time used by solving the PPE depends on which solver for linear systems is used. In this paper, the biconjugate gradient stabilized (BCGSTAB) method [47], one of popular linear solvers for linear systems, is employed. The speedup ratios presented in Fig. 25 can be different if other solvers would be employed. It is

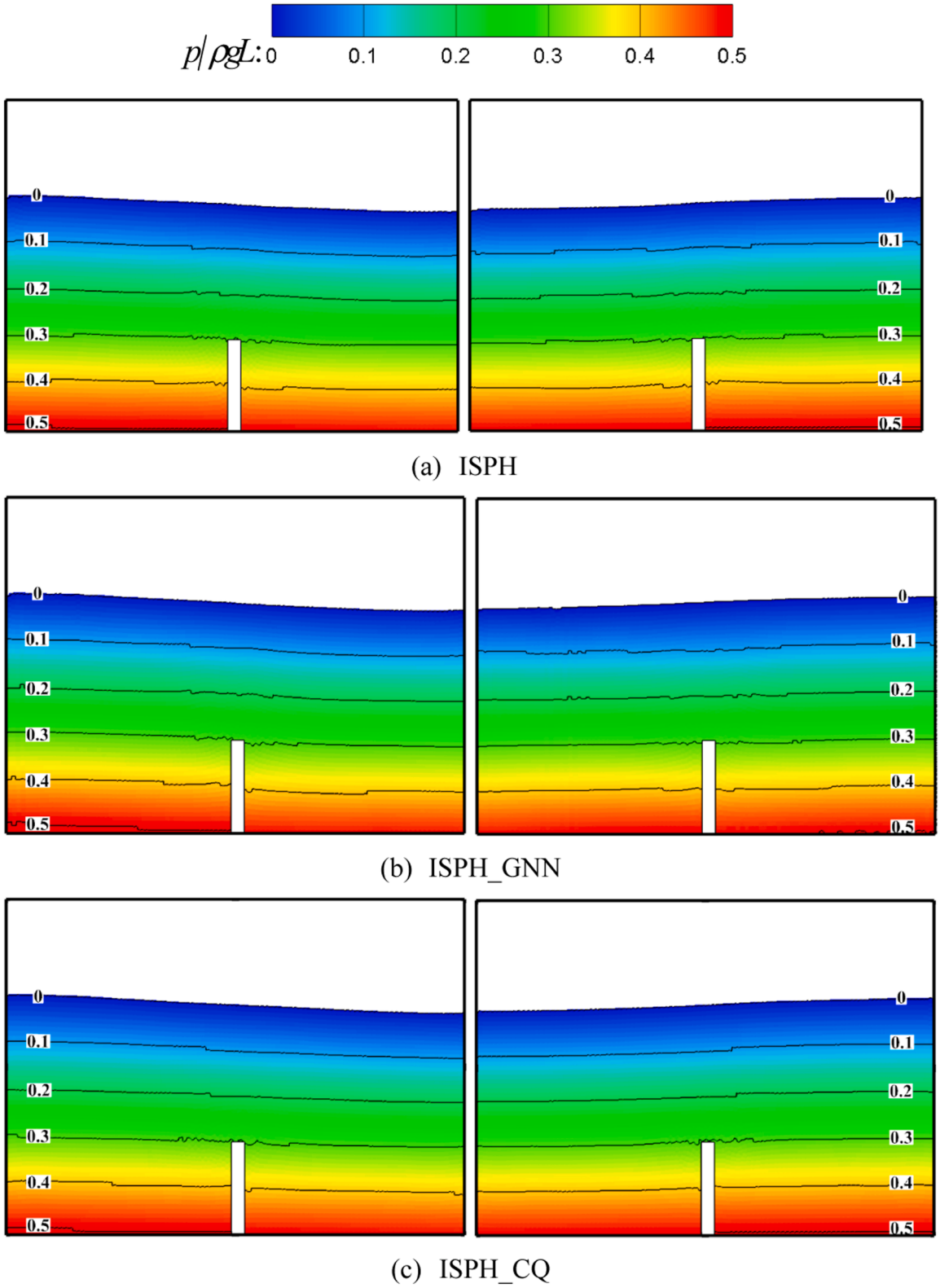


Fig. 19. The particle distribution and the pressure contour in the case of sloshing with a vertical baffle at different instants: $\widetilde{t}$= 17.23 (left column) and $\widetilde{t}$ = 18.79 (right column).

also noted that the computational time of the GNN model for predicting the pressure does not include the time for data generation and training. The main reason for doing so is that the GNN model does not need to be trained for each specific case and can become mature after being cumulatively trained by a sufficiently large number of cases. As a result, including the time for data generation and training for each specific case may not reflect the real efficiency of ISPH_GNN.

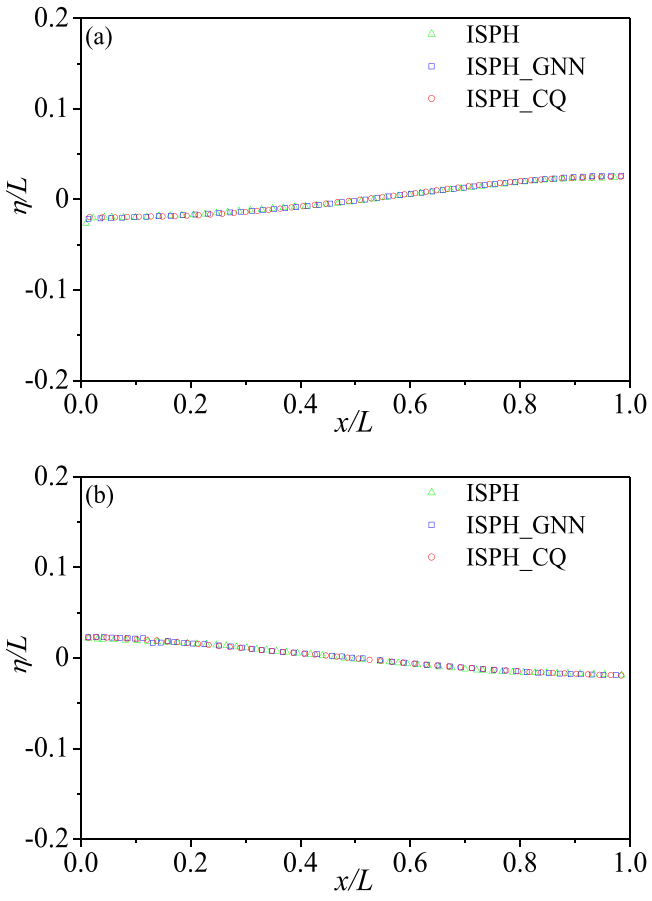


Fig. 20. The comparison of the surface elevations between different ISPH results in the case of sloshing with a vertical baffle at two different instants:(a) $\widetilde{t}$= 15 and (b) $\widetilde{t}$= 36.

5. Conclusion

The present study proposes a new hybrid method (ISPH_GNN) for simulating free-surface flows by combing the Graph Neural Network (GNN) with the Lagrangian ISPH method. In this method, the trained GNN model is used to replace solving the PPE in the conventional ISPH. Some important issues related to the method, including the selection of the input parameters, the satisfaction of the boundary conditions and the formulation of the objective functions, have been addressed by considering the underlying features dominating the free-surface flows, such as the physical quantities involved in the PPE (velocity divergence and particle density at the intermedia step), viscous and gravitational effects, particle movement in the Lagrangian frame etc. The ISPH_GNN is then applied to three benchmark tests, i.e. the dam breaking, sloshing wave and the solitary wave propagation. The performance including the convergence, accuracy and the computational efficiency of the present ISPH_GNN are investigated in detail. It has been observed that:

(1) The present ISPH_GNN can produce reasonable results which are visually very similar to the corresponding results obtained by the conventional ISPH and ISPH_CQ.
(2) The ISPH_GNN is shown to be linearly convergent, the same as the conventional ISPH but the error of ISPH_GNN is visibly smaller than that of the conventional ISPH if their particle spacing is the same.
(3) The ISPH_GNN can be potentially much more efficient than the conventional ISPH at similar level of accuracy when the number of particles used is large.

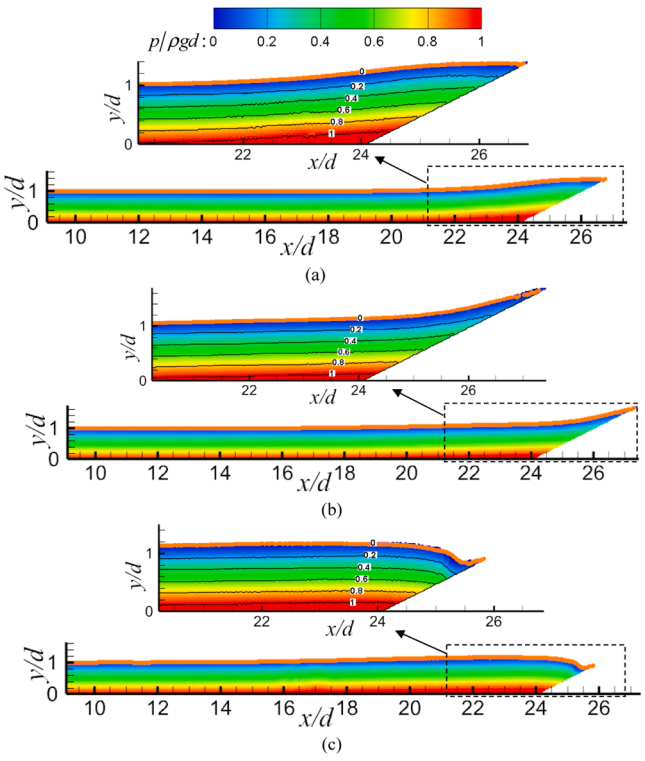


Fig. 21. Snapshots illustrating distribution of particles along with pressure field from the ISPH_GNN in the case of solitary wave propagation on a slope with $h = 0.24d$ at (a) $\widetilde{t}$= 31.32; (b) $\widetilde{t}$= 34.45 and (c) $\widetilde{t}$= 34.45 (the orange line denotes the wave profile from the ISPH_CQ). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

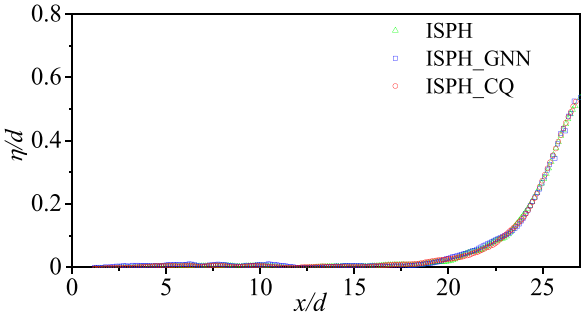


Fig. 22. Comparisons of wave profiles between different ISPH results in the case of solitary wave propagation on a slope ($h = 0.24d$).

The paper also investigates the potential of generalization of the ISPH_GNN by applying the method with a GNN model trained on the data for relatively simple cases to simulating the cases with more complex geometries and configurations. It is shown that the ISPH_GNN method can yield satisfactory results for the more complex cases beyond the cases used for generating the training data. This sheds some light on the potential of the method.

It should be noted that this is the first paper proposing the method, which has some limitations and needs to be improved. Our future work will address the limitations, such as adopting a wider range of training data for the training, exploring better way to implement the solid boundary conditions, and/or testing alternative objective functions, e. g., including the satisfaction of the physical law. In the future work, we may also attempt to train one generalized GNN model by using the mixed training dataset generated from different physical problems and

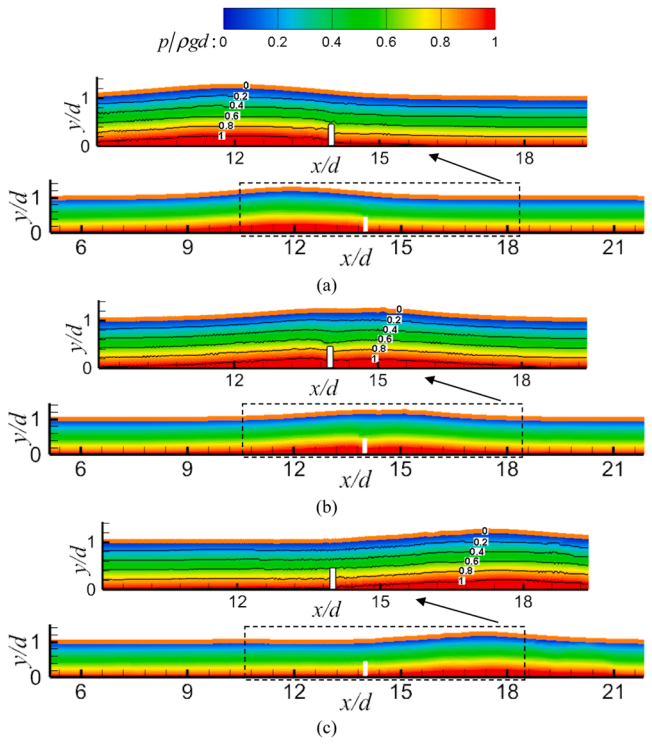


Fig. 23. Snapshots illustrating distribution of particles along with pressure field from the ISPH_GNN in the case of solitary wave over a vertical baffle with $h = 0.24d$ at (a) $\tilde{t}$= 18.79; (b) $\tilde{t}$= 21.3 and (c) $\tilde{t}$= 23.8 (the orange line denotes the wave profile from the ISPH_CQ). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

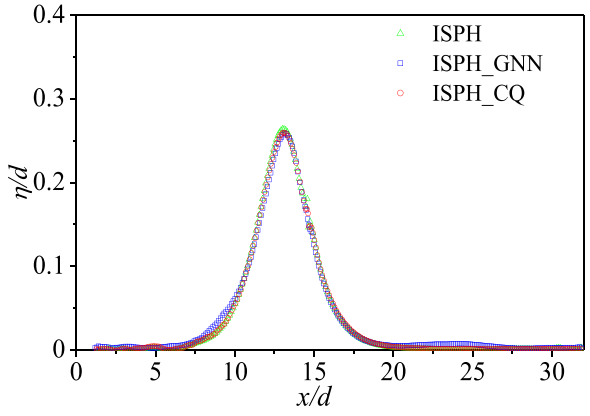


Fig. 24. Comparisons of wave profiles at $\tilde{t}$= 20 obtained by different ISPH models in the case of solitary wave over a vertical baffle with $h = 0.24d$.

then apply the ISPH_GNN with this generalized model to simulate different class of physical problems.

In addition, only single-phase flows have been simulated by the ISPH_GNN in this paper. The application of the ISPH_GNN in multi-phase flows simulation is worth to be investigated in future work. The investigation can be conducted from two aspects. One is the application of the ISPH_GNN trained on data generated from single-phase flows to the simulation of multi-phase flows. The other is the attempt to train the ISPH_GNN using training data generated from a certain type of multi-phase flow physics problem, e.g., the gas–liquid two-phase flow, and then extend the trained ISPH_GNN to study other types of multi-phase flow problems. Furthermore, only two-dimensional problems have been considered in this paper. The capacity of ISPH_GNN for simulating

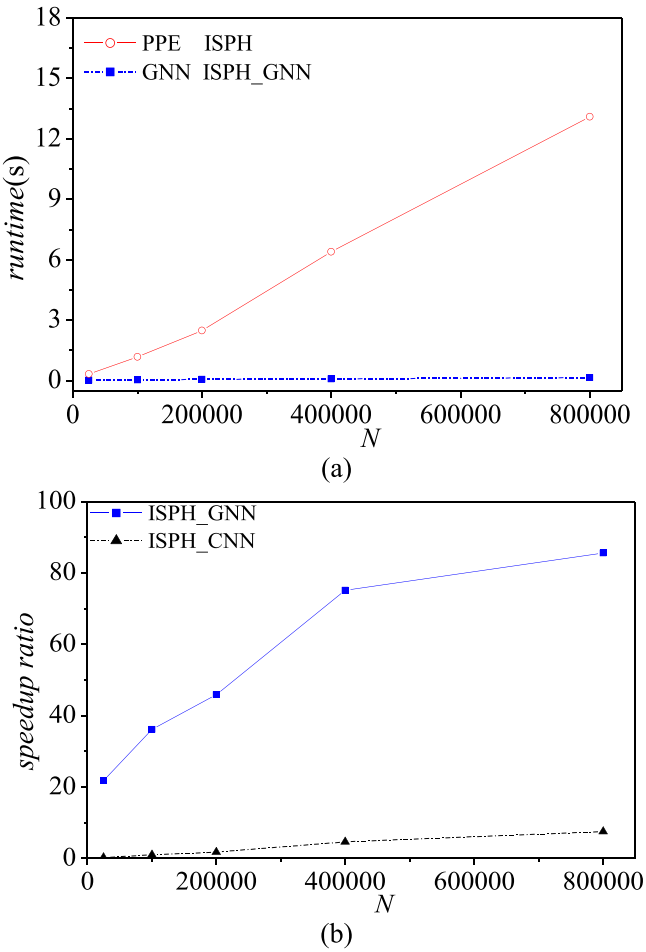


Fig. 25. Comparison of the computational efficiency for the case of solitary wave propagation: (a) averaged computational time used for pressure prediction per step and (b) the speedup ratio for the averaged computational time used for pressure prediction per step.

Table 1
Case configurations for the solitary wave propagation.

Case	D (m)	L (m)	dx (m)	N
1	0.25	10	0.01	25,000
2	0.25	40	0.01	100,000
3	0.25	80	0.01	200,000
4	0.5	80	0.01	400,000
5	0.5	160	0.01	800,000

three-dimensional problems needs to be investigated in future work.

CRediT authorship contribution statement

Ningbo Zhang: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Shiqiang Yan:** Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Qingwei Ma:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Qian Li:** Visualization, Software, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The authors gratefully acknowledge the financial support of EPSRC projects (EP/V040235, EP/T026782 and EP/T00424X). The authors greatly appreciate the reviewers for their dedication in conducting thorough and professional reviews, which significantly enhanced the quality of this paper.

The implementation and corresponding instructions of this hybrid method (ISPH_GNN) for predicting the results presented in this paper are publicly available at: https://github.com/cityhydro/ISPH_GNN.

Appendix A. Identifying of the free-surface particles and determining the vertical distance between the inner particle to free surface

For modelling the free-surface flows, it is critical to identify the free-surface particles. In this paper, an effective approach based on the particle density ratio and the particle distribution developed in Zheng et al. [5] is used to identify the free-surface particles. In this approach, three auxiliary functions relevant to the particle distribution are defined. The first one $f_{spa} = 1$ if there are more than one free-surface particle in the influence domain of the particle i, and zero otherwise. The 2nd one $f_{spb} = 1$ if particles are detected in all quadrants of the particle i defined using the local x_i and y_i axis (see Fig. A.1(a)), and $f_{spb} = 0$ otherwise. The third one $f_{spc} = 1$ if particles are detected in all quadrants of the particle i divided by $y_i = x_i$ and $y_i = -x_i$ (see Fig. A.1(b)), and $f_{spc} = 0$ otherwise. The particle i can be identified as a free-surface particle if (1) no inner particles are observed in its influence domain; or (2) if the particle density ratio is not higher than 0.9, either $f_{spa} = 1$ or particles are observed in no more than 2 quadrants of the particle i defined for both f_{spb} and f_{spc}; or (3) if the particle density ratio is higher than 0.9 and $f_{spa} = 1$, either f_{spb} or $f_{spc} = 0$.

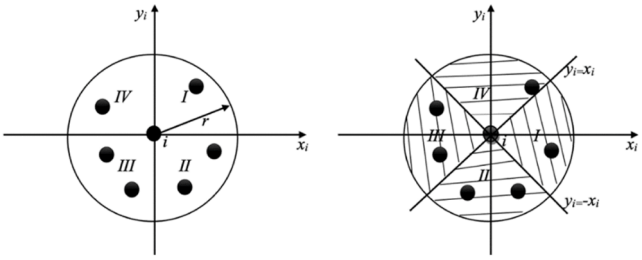


Fig. A.1. Illustration of definitions of auxiliary functions for the free-surface particle identification.

Based on the free-surface particles identified in this way, the reference pressure related to Eq. (18) is estimated by $p_s = \rho g \tilde{z}$ with $\tilde{z}$ being the vertical distance from the inner fluid particle i to its nearest free-surface particle j, as shown in Fig. A.2.

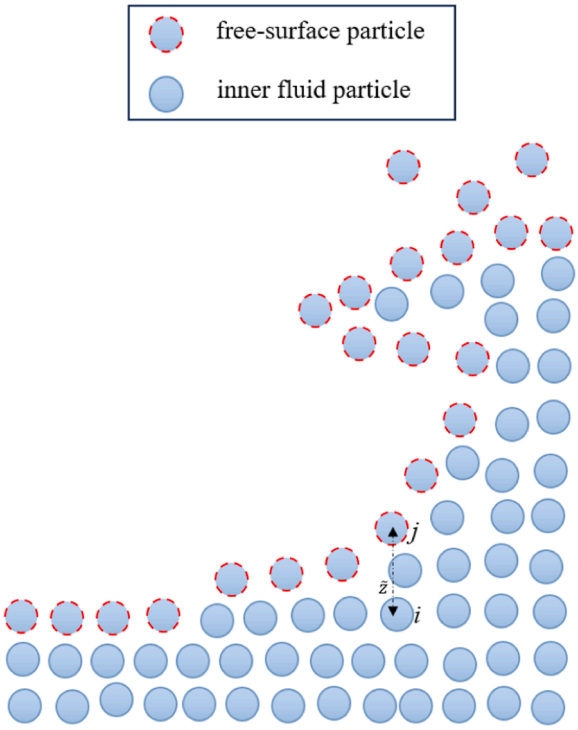


Fig. A.2. The free-surface and inner fluid particles.

References

[1] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, Astrophys. J. 8 (12) (1977) 1013–1024.
[2] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics – theory and application to non-spherical stars, Mon. Not. R. Astro. Soc. 181 (3) (1977) 375–389.
[3] S.D. Shao, E.Y.M. Lo, Incompressible SPH method for simulating newtonian and non-newtonian flows with a free surface, Adv. Water Resour. 26 (7) (2003) 787–800.
[4] G. Fourtakas, B.D. Rogers, A.M.A. Nasar, Towards pseudo-spectral incompressible smoothed particle hydrodynamics (ISPH), Comput. Phys. Commun. 226 (2021) 108028.
[5] X. Zheng, Q.W. Ma, W.Y. Duan, Incompressible SPH method based on Rankine source solution for violent water wave simulation, J. Comput. Phys. 276 (2014) 291–314.
[6] A. Khayyer, H. Gotoh, H. Falahaty, Y. Shimizu, An enhanced ISPH–SPH coupled method for simulation of incompressible fluid–elastic structure interactions, Comput. Phys. Commun. 232 (2018) 139–164.
[7] N.B. Zhang, X. Zheng, Q.W. Ma, W.Y. Duan, A. Khayyer, X.P. Lv, S.D. Shao, A hybrid stabilization technique for simulating water wave–structure interaction by incompressible smoothed particle hydrodynamics (ISPH) method, J. Hydro-environ. Res. 18 (2018) 77–94.
[8] N.B. Zhang, S.Q. Yan, Q.W. Ma, X. Zheng, A QSFDI based Laplacian discretisation for modelling wave-structure interaction using ISPH, Appl. Ocean Res. 117 (2021) 102954.
[9] N.B. Zhang, S.Q. Yan, Q.W. Ma, A. Khayyer, X.H. Guo, X. Zheng, A consistent second order ISPH for free surface flow, Comput. Fluids 274 (2024) 106224.
[10] H. Gotoh, A. Khayyer, H. Ikari, T. Arikawa, K. Shimosako, On enhancement of incompressible SPH method for simulation of violent sloshing flows, Appl. Ocean Res 46 (11) (2014) 104–115.
[11] Q.W. Ma, Y. Zhou, S.Q. Yan, A review on approaches to solving Poisson's equation in projection-based meshless methods for modelling strongly nonlinear water waves, J. Ocean. Eng. Mar. Energy 2 (3) (2016) 279–299.
[12] J.N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. 814 (2017) 1–4.
[13] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science (1979) 367 (6481) (2020) 1026–1030.
[14] Z.J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2460.
[15] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.
[16] C. Yang, X.B. Yang, X.Y. Xiao, Data driven projection method in fluid simulation, Comput. Animat. Virt. W. 27 (3-4) (2016) 415–424.
[17] X.Y. Xiao, Y. Zhou, H. Wang, X.B. Yang, A novel CNN-based poisson solver for fluid simulation, IEEE T. Vis. Comput. Gr. 26 (3) (2018) 1454–1465.
[18] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating eulerian fluid simulation with convolutional networks, in: International Conference on Machine Learning. PMLR, 2017, pp. 3424–3433.
[19] W. Dong, J. Liu, Z. Xie, D. Li, Adaptive neural network-based approximation to accelerate eulerian fluid simulation, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–22.
[20] A. Khayyer, H. Gotoh, Y. Shimizu, Comparative study on accuracy and conservation properties of two particle regularization schemes and proposal of an optimized particle shifting scheme in ISPH context, J. Comput. Phys. 332 (2017) 236–256.
[21] A. Skillen, S. Lind, P.K. Stansby, B.D. Rogers, Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothing applied to body–water slam and efficient wave–body interaction, Comput. Method Appl. M. 265 (2013) 163–173.
[22] N.B. Zhang, S.Q. Yan, Q.W. Ma, X.H. Guo, Z.H. Xie, X. Zheng, A CNN-supported Lagrangian ISPH model for free surface flow, Appl. Ocean Res. 136 (2023) 103587.
[23] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. International Joint Conference on Neural Networks, IEEE, 2005, pp. 729–734.
[24] F. Scarselli, M. Gori, A.C.M. Tsoi, Hagenbuchner, G. Monfardini, The graph neural network model, IEEe Trans. Neural Netw. 20 (1) (2008) 61–80.
[25] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, 2015, arXiv preprint.
[26] F.D.A. Belbute-Peres, T. Economon, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: In international conference on machine learning, PMLR, 2020, pp. 2402–2411.
[27] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, 2020, arXiv preprint.
[28] J. Chen, E. Hachem, J. Viquerat, Graph neural networks for laminar flow prediction around random two-dimensional shapes, Phys. Fluids. 33 (12) (2021).
[29] R. Gao, R. Jaiman, Quasi-Monolithic Graph Neural Network for Fluid-Structure Interaction, 2022, arXiv preprint.
[30] M. Lino, S. Fotiadis, A.A. Bharath, C.D. Cantwell, Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics, Phys. Fluids. 34 (8) (2022) 087110.
[31] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International conference on machine learning, PMLR, 2020, pp. 8459–8468.
[32] K. Kumar, J. Vantassel, GNS: A generalizable Graph Neural Network-based simulator for particulate and fluid modeling. arXiv preprint , (2022).
[33] Z.J. Li, A.B. Farimani, Graph neural network – accelerated Lagrangian fluid simulation, Comput. Graph. 103 (2022) 201–211.
[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).
[35] Y. Chung, P.J. Haas, E. Upfal, T. Kraska, Unknown examples & machine learning model generalization. arXiv preprint , (2018).
[36] M. Emami, M. Sahraee-Ardakan, P. Pandit, S. Rangan, A. Fletcher, Generalization error of generalized linear models in high dimensions, in: International Conference on Machine Learning, PMLR, 2020, pp. 2892–2901.
[37] Q.W. Ma, J.T. Zhou, MLPG_R method for numerical simulation of 2D breaking waves, CMES-Comp. Model Eng. 43 (3) (2009) 277–303.
[38] J.J. Monaghan, J.C. Lattanzio, A refined method for astrophysical problems, Astron. Astrophys. 149 (1) (1985) 135–143.
[39] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE T. Neur. Net. Lear. 32 (1) (2020) 4–24.
[40] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2016, arXiv preprint.
[41] S.Q. Yan, Q.W. Ma, J.H. Wang, Quadric SFDI for Laplacian Discretisation in Lagrangian Meshless Methods, J. Mar. Sci. Appl. 19 (3) (2020) 362–380.
[42] S. Adami, X.Y. Hu, N.A. Adams, A generialised wall boundary condition for smoothed particle hydrodynamics, J. Comput. Phys. 231 (2012) 7057–7075.
[43] L. Lobovský, E. Botia-Vera, F. Castellana, J. Mas-Soler, A. Souto-Iglesias, Experimental investigation of dynamic pressure loads during dam break, J. Fluid Struct. 48 (2014) 407–434.
[44] O.M. Faltinsen, A numerical nonlinear method of sloshing in tanks with two-dimensional flow, J. Ship Res. 22 (03) (1978) 193–202.
[45] G.X. Wu, Q.W. Ma, R.E. Taylor, Numerical simulation of sloshing waves in a 3D tank based on a finite element method, Appl. Ocean Res. 20 (6) (1998) 337–355.
[46] J.J. Lee, J.E. Skjelbreia, F. Raichlen, Measurement of velocities in solitary waves, J. Waterw. Port Coast. 108 (2) (1982) 200–218.
[47] H.A. Van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear system, SIAM Journal on scientific and Statistical Computing 13 (1992) 631–644.