Oral presentation | Multi-phase flow
High performance computing-II
Wed. Jul 17, 2024 10:45 AM - 12:15 PM Room D

[7-D-03] Evaluate Computational Efficiency of Adaptive Particle Neural Particle Method (NPM-A) for Modeling Fluid Flow Problems

Pei-Hsin Pai[1], *Heng-Chuan Kan[2], Yih-Chin Tai[1] （1. Department of Hydraulic and Ocean Engineering, National Cheng Kung University, 2. National Center for High-performance Computing, National Applied Research Laboratories>）

Keywords: Adaptive Neural Particle Method (NPM-A), Transfer Learning, Physics-Informed Neural Networks (PINNs)

Twelfth International Conference on Computational Fluid Dynamics (ICCFD12), Kobe, Japan, July 14-19, 2024

ICCFD12-2024-xxxx

Evaluate Computational Efficiency of Adaptive Particle Neural Particle Method (NPM-A) for Modeling Fluid Flow Problems

Pei-Hsin Pai*, Heng-Chuan Kan** and Yih-Chin Tai*
Corresponding author: hckan@narlabs.org.tw
* Department of Hydraulic and Ocean Engineering, National Cheng Kung University, Tainan, Taiwan.
** National Center for High-performance Computing, National Applied Research Laboratories, Tainan, Taiwan.

1 Introduction

In recent years, the utilization of physics-informed neural networks (PINNs) [1], a physics-based machine-learning method, has emerged as a powerful approach in the field of computational fluid dynamics (CFD) owing to its capability and generality in solving non-linear partial differential equations [2, 3, 4]. Unlike purely data-driven machine learning methods, PINNs integrate the training process of neural networks with governing physics equations, facilitating the modeling of diverse simulations even when data is limited. However, a significant limitation of many existing PINN formulations lies in their development within the Eulerian framework, which falls short in addressing the challenges posed by free surface flows and moving boundary problems inherent in fluid dynamics [5, 6].

The Neural Particle Method (NPM) stands as a notable advancement in Lagrange-based PINNs tailored specifically for modeling free surface flows [7]. Unlike Eulerian methods, NPM exhibits the capability to accurately describe flow surfaces without succumbing to issues such as particle splashing commonly observed in other conventional particle-based methods like Smooth Particle Hydrodynamics (SPH) [8, 9], Moving-particle semi-implicit methods (MPS) [10, 11], and Discrete Element methods (DEM) [12]. The effectiveness of NPM has prompted numerous research endeavors aimed at enhancing its functionalities and addressing existing challenges. However, NPM's limitation to simulating inviscid flow led to the introduction of the general Neural Particle Method (gNPM) by Bai et al. [13], which extends its applicability to simulate viscous flow through adjustments in differentiation across the spatial domain. Despite these advancements, Lagrange-based PINNs encounter challenges, particularly in accurately delineating complex flow geometries due to distorted fluid particle patterns. To address this, Shao et al. [14] proposed the Improved Neural Particle Method (INPM), aiming to rectify insufficient particle distributions along boundaries. However, INPM remains confined to inviscid flow simulations, failing to address a broader spectrum of fluid dynamics scenarios. Moreover, distorted particle patterns compromise computational resolution in specific regions, thereby impacting result accuracy. In response, Pai et al. [15] introduced an adaptive Neural Particle Method (NPM-A), integrating numerical techniques such as interface tracking and adaptive particle refinement to ensure uniform particle patterns across both interface and interior regions. Leveraging inference, a unique feature of machine learning modeling, NPM-A offers flexibility in fluid particle reassignment without introducing additional numerical errors. Consequently, NPM-A sustains regular particle patterns while enhancing its capacity to capture complex fluid behaviors. Despite the competitive advantages of NPM-A, a notable drawback pertains to their computational inefficiency compared to traditional numerical approaches. While accelerated methods for deep learning have undergone significant development in both hardware and software realms [16], their application to purely physics-driven machine learning approaches remains hindered by the absence of accessible validation data.

Automatic differentiation (AD) stands out as a primary contributor to computational time consumption in PINNs. This technique, distinct from conventional numerical approaches, facilitates the computation of higher-order derivatives crucial for solving complex physical equations. Yet, the computation of these derivatives poses challenges, particularly in large-scale simulations and training processes of neural network frameworks. Recent advancements have proposed alternative strategies such as CAN-PINNs [17], which substitute automatic differentiation with traditional numerical differentiation techniques to reduce computational costs without sacrificing spatial resolution. Additionally, first-order formulation techniques, as exemplified by FO-PINNs [18], aim to enhance the computational efficiency of

standard PINNs by reformulating governing equations into a first-order framework. These innovations, coupled with techniques like automatic mixed precision (AMP), further accelerate computational time by utilizing lower precision numerical formats, thereby minimizing memory usage and computational overhead.

Furthermore, the discrete time model (DTM) employed in NPM-A for time marching entails individual training for each time step, resulting in prolonged computational time and resources. To mitigate these limitations, we introduce the transfer learning strategies [19] for the Neural Particle Method (TL-NPM). By leveraging knowledge from related tasks, transfer learning significantly reduces training time and resources, particularly in scenarios with limited data or high correlation between tasks [20, 21, 22]. For Lagrange-based PINNs, the gradual evolution of physical quantities and coordinates across time-steps suggests a high degree of correlation, rendering transfer learning a feasible avenue for enhancing efficiency. In present study, we investigate methods for enhancing the computational efficiency of NPM-A through proposed transfer learning strategies (TL-NPM) with first-order formulation techniques (FO-NPM) to accelerate the training process. We employ these techniques to simulate two distinct fluid problems: (i) a rotating square fluid patch and (ii) wave breaking. Our results demonstrate significant improvements in computational efficiency, achieving speed-ups ranging from 3.97 to 21.47 times for the rotational square fluid patch case. Furthermore, even more substantial computational performance improvements up to 7.59 times for the wave breaking case without sacrificing solution accuracy.

The remainder of this paper is organized as follows. In Section 2, we first describe the model equations, the adaptive Neural Particle Method (NPM-A), and the proposed TL-NPM, which integrate transfer learning and the first-order Neural Particle Method (FO-NPM). The efficiency of TL-NPM is examined through two numerical examples in Section 3. Finally, concluding remarks are discussed in Section 4.

2 Neural Particle Method and Transfer Learning

2.1 Model Equations

We consider a two-dimensional incompressible Newtonian fluid, for which the continuity equation reads

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0\,, \tag{1}$$

and the momentum equations (Navier-Stokes equations) are given by

$$\begin{aligned} \frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 + p\right) + \frac{\partial}{\partial y}\left(\rho u v\right) &= \rho g_x + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ \frac{\partial(\rho v)}{\partial t} + \frac{\partial}{\partial x}\left(\rho u v\right) + \frac{\partial}{\partial y}\left(\rho v^2 + p\right) &= \rho g_y + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \end{aligned} \tag{2}$$

In (1) and (2), x and y denote the Cartesian coordinates, u and v represent the corresponding components of velocity, respectively, t denotes time, ρ represents the fluid density, p is the pressure, μ means the (dynamic) viscosity, and g_x and g_y are the gravitational acceleration in x- and y-direction, respectively. Together with the continuity equation (1), the momentum equations (2) can be transferred to be in Lagrangian form,

$$\begin{aligned} a_x = \frac{\mathrm{d}u}{\mathrm{d}t} &= g_x - \frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ a_y = \frac{\mathrm{d}v}{\mathrm{d}t} &= g_y - \frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \tag{3}$$

which is the acceleration of the tracked particle used in the Neural Particle Methods (NPM), see [7, 13, 15].

2.2 Adaptive Neural Particle Method (NPM-A)

For a highly deforming flow body, the distribution of fluid particle may become sparsely scattered at some portions, inducing a poor resolution locally. The adaptive neural particle method (NPM-A) is therefore introduced in Pai et al. [15], in which the inference technique is employed to determine the physical quantities of the added fluid particles at new positions. The flowchart of NPM-A is depicted in Fig. 1. The complete process of a single time step is given in the light-gray-shaded region, where the marker ★ indicates the starting point of training process. At the initial time level, the input data for

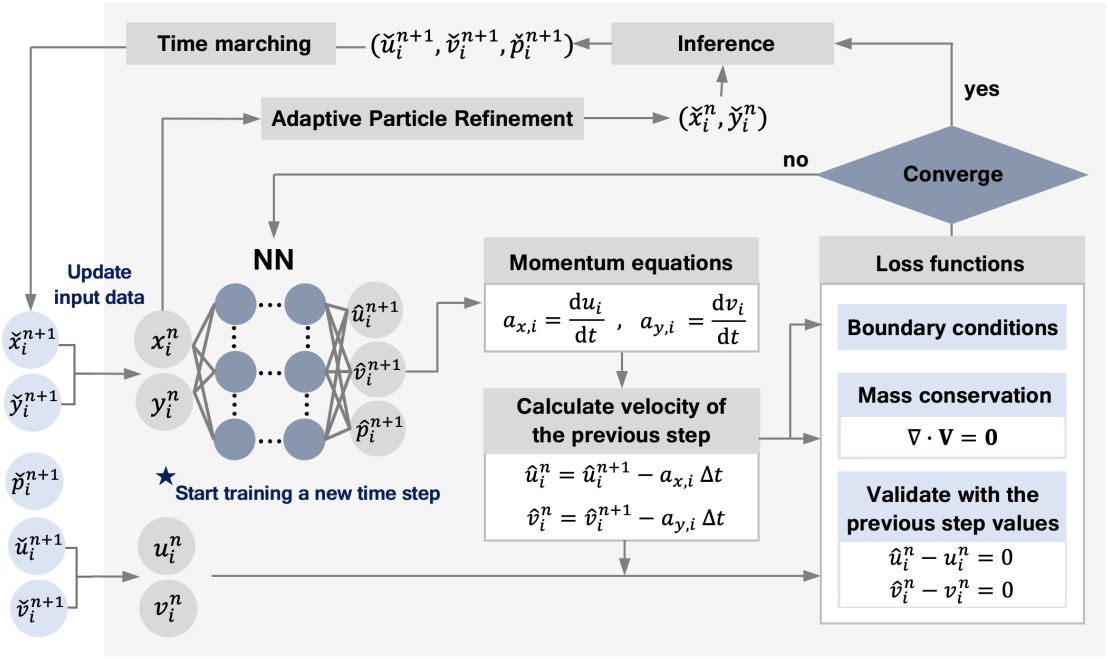

Figure 1: Flowchart of NPM-A.

the neural networks (NN) are the positions of the fluid particles, e.g., (x_i^n, y_i^n) for particle i at time level n. Although the corresponding velocities (u_i^n, v_i^n) are not listed in the input data, they are needed in evaluating the deviations through the associated loss functions. Through the forward propagation of the assigned multilayer perceptron (MLP), the output data are the corresponding velocities and pressures for the input positions at the new time level $n+1$, i.e., $(\hat{u}_i^{n+1}, \hat{v}_i^{n+1}, \hat{p}_i^{n+1})$. With the help of the Automatic Differentiation (AD), the back-propagation process in NN-training, the spatial derivatives in the model equations (1) and (3) can be obtained. The continuity equation (1) guarantees the incompressibility, and the momentum equations in (3) yield the accelerations, i.e., $(a_{x,i}, a_{y,i})$ and they are given by

$$\begin{aligned} a_{x,i} &= g_x - \frac{1}{\rho}\frac{\partial}{\partial x} p_i^{n+1} + \frac{\mu}{\rho}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u_i^{n+1}, \\ a_{y,i} &= g_y - \frac{1}{\rho}\frac{\partial}{\partial y} p_i^{n+1} + \frac{\mu}{\rho}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) v_i^{n+1}. \end{aligned} \tag{4}$$

The NN-predicted velocity $(\hat{u}_i^n, \hat{v}_i^n)$ at the previous time level n can therefore be calculated by

$$\begin{cases} \hat{u}_i^n = \hat{u}_i^{n+1} - a_{x,i}\,\Delta t \\ \hat{v}_i^n = \hat{v}_i^{n+1} - a_{y,i}\,\Delta t \end{cases}, \tag{5}$$

in which Δt means the time interval. The loss functions consist of the model equations, and the associated boundary conditions. The continuity equation (1) means the vanishing velocity gradient, makes sure the incompressibility. The momentum equations turn to be the difference between the velocity (u_i^n, v_i^n) at t^n and the NN-predicted velocity $(\hat{u}_i^n, \hat{v}_i^n)$ given in (5), where the magnitude of the differences should vanish. At each single time step, the trained NN-model can directly predict the physical quantities for any positions in the computational domain, known as *inference*. In the present study, we utilize the inference, instead of interpolation, to determine the physical quantities of the rearranged particles, because they are supposed to fulfill the model equations in the deep-learning process. Leveraging this advantage, NPM-A can adaptively adjust the positions of computing particles to meet the assigned distribution pattern, such as an even distribution or local refinement. The Adaptive Particle Refinement (APR) in NPM-A is applied for both the interior region and interfaces. Through APR, the physical quantities $(\check{u}_i^{n+1}, \check{v}_i^{n+1}, \check{p}_i^{n+1})$ at the rearranged positions $(\check{x}_i^{n+1}, \check{y}_i^{n+1})$, are predicted by inference, which is trained

with respect to the original positions (x_i^n, y_i^n). Then, we can update the new positions $(\check{x}_i^{n+1}, \check{y}_i^{n+1})$ with

$$\begin{cases} \check{x}_i^{n+1} = \check{x}_i^n + \check{u}_i^{n+1}\Delta t \\ \check{y}_i^{n+1} = \check{y}_i^n + \check{v}_i^{n+1}\Delta t \end{cases} \tag{6}$$

for time marching. The input data (x_i^n, y_i^n) is updated by the new positions $(\check{x}_i^{n+1}, \check{y}_i^{n+1})$ and the corresponding velocities $(\check{u}_i^{n+1}, \check{v}_i^{n+1})$ for the next training process at next time step. By repeating the proceeding process, the dynamical fluid simulation is achieved.

Following [1, 13, 15], a 2-stage fourth-order implicit Runge Kutta (IRK) method is utilized for higher accuracy. That is, we have three predicted velocities at t^n based on the physical quantities at three time levels $(t^{n+c_1}, t^{n+c_2}, t^{n+1})$, respectively, for each computational particle, where $c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}$ and $c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. Hence, the output of the NN-process reads

$$\left(\hat{u}^{n+c_1},\, \hat{u}^{n+c_2},\, \hat{u}^{n+1},\, \hat{v}^{n+c_1},\, \hat{v}^{n+c_2},\, \hat{v}^{n+1},\, \hat{p}^{n+c_1},\, \hat{p}^{n+c_2}\right) \quad \text{and} \quad \hat{p}^{n+1} = \tfrac{1}{2}\left(\hat{p}^{n+c_1} + \hat{p}^{n+c_2}\right).$$

For details on the high-order approach, we refer readers to [15] or [1].

2.3 First-Order Formulation for Neural Particle Method (FO-NPM)

To enhance the efficiency of NPM, Gladstone et al., [18] suggested a reformulation of the model equations into a type of first-order for the physics-informed neural networks (FO-PINNs). Here we refer this method as the FO-NPM. In FO-NPM, the output of the neural networks is expanded from $(\hat{u}, \hat{v}, \hat{p})$ to $(\hat{u}, \hat{v}, \hat{p}, \hat{u}_{,x}, \hat{u}_{,y}, \hat{v}_{,x}, \hat{v}_{,y})$, with $\hat{u}_{,x} = \frac{\partial \hat{u}}{\partial x}$, $\hat{u}_{,y} = \frac{\partial \hat{u}}{\partial y}$, $\hat{v}_x = \frac{\partial \hat{v}}{\partial x}$, and $\hat{v}_{,y} = \frac{\partial \hat{v}}{\partial y}$ the first-order derivatives. Hence, the momentum equations (3) are recast by

$$\begin{aligned} a_x^{\mathrm{F}} &= \frac{\mathrm{d}u}{\mathrm{d}t} = g_x - \frac{1}{\rho}\frac{\partial \hat{p}}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial \hat{u}_{,x}}{\partial x} + \frac{\partial \hat{u}_{,y}}{\partial y}\right), \\ a_y^{\mathrm{F}} &= \frac{\mathrm{d}v}{\mathrm{d}t} = g_y - \frac{1}{\rho}\frac{\partial \hat{p}}{\partial y} + \frac{\mu}{\rho}\left(\frac{\partial \hat{v}_{,x}}{\partial x} + \frac{\partial \hat{v}_{,y}}{\partial y}\right). \end{aligned} \tag{7}$$

Despite of the additional output of the first-order derivatives $(\hat{u}_{,x}, \hat{u}_{,y}, \hat{v}_{,x}, \hat{v}_{,y})$, the efficiency of FO-NPM can be highly enhanced because the back-propagation steps for calculating the second-order derivatives are reduced. FO-NPM can also integrate with the treatment of automatic mixed precision (AMP) [23] to reduces the runtime and memory footprint by adjusting the datatype during the training process. Although AMP is widely used in machine learning, one should notice that it can cause instability for high-order differentiation in PINNs [18]. With the intrinsic first-order formulation, the FO-NPM can benefit of further accelerating the training process by using the AMP. In addition, the FO-NPM may substantially reduce memory requirements, thereby increasing the potential for large-scale simulations.

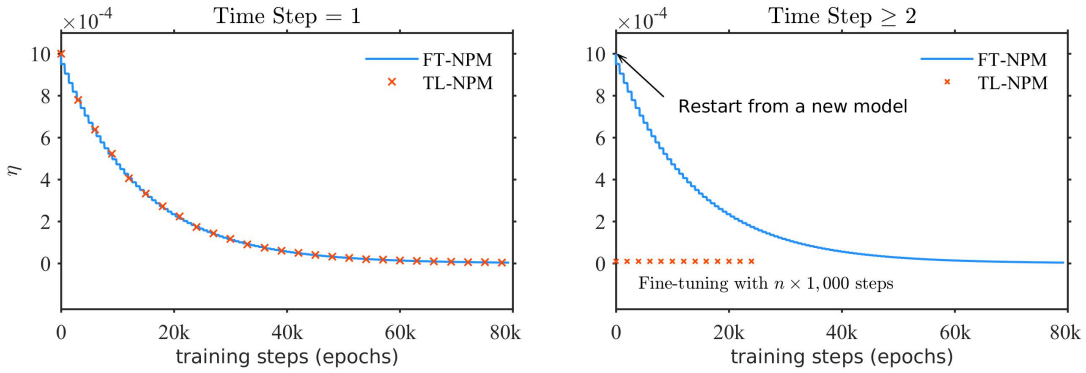

Figure 2: Schematic of the learning rate scheduler with time step progression, where the blue line denotes the standard training method and the red-cross line represents the proposed transfer learning strategy.

Table 1: Hyper parameters of the proposed transfer learning strategies.

	NN sizes	**Optimizer**	**Activation function**	**Epochs**	**Learning rate**	**Learning rate scheduler**	
						Decay rate α	**Decay steps n_d**
FT-40k	6×512	Adam	SiLU	$40,000$	1e-3	0.95	350
FT-80k	6×512	Adam	SiLU	$80,000$	1e-3	0.95	700
TL-NPM (time step =1)	6×512	Adam	SiLU	$80,000$	1e-3	0.95	700
TL-NPM (time step >1)	6×512	Adam	SiLU	$n \times 1,000$ ($n = 5, 10, \ldots, 20$)	1e-5	n.a.	n.a.
TL-ADA-NPM (time step =1)	6×512	Adam	adaptive SiLU	$80,000$	1e-3	0.95	700
TL-ADA-NPM (time step >1)	6×512	Adam	adaptive SiLU	$n \times 1,000$ ($n = 5, 10, \ldots, 30$)	1e-5	n.a.	n.a.

2.4 Implementation of Transfer Learning

Taking advantage of the capability to accelerate the training process, the NPM-A with transfer learning strategy, referred to as TL-NPM, is implemented based on the FO-NPM formulation. Without the mechanism of learning transfer, the assigned training steps (epochs) at each time step are 80 k, which is named "full-training" and indexed by "FT-NPM" in the present study. To evaluate the effectiveness of the imposed transfer learning strategy, the concept of *fine-tuning* is employed. That is, at the first time step (time step =1 in Fig. 2), the training steps (epochs) remains as in the full-training approach (80 k), where the learning rate η reduces exponentially from 10^{-3} to 10^{-5} with the increasing training steps, cf. Fig. 2. In the subsequent time steps, the NN-model trained at the previous time step is reused, the total training steps reduce to the assigned $n \times 1,000$ epochs ($n = 5, 10, \ldots,$ or 30), and the learning rate remains the asymptotic value 10^{-5}, see the right panel of Fig. 2 and Table 1. Figure 2 illustrates the evolution of learning rate scheduler with time steps, where the blue line denotes the standard full training method (referred to as FT-NPM), and the red-cross line represents TL-NPM. Because the hyper-parameters are adjustable with abundant possibilities, we fix most of the parameters to simplify the problem and focus on demonstrating the required number of epochs for fine-tuning, where the parameter values used in the testing cases are collected and listed in Table 1.

Similar to [15], a multilayer perceptron (MLP) composed of 6 hidden layers with 512 neurons in each layer is used, where the optimizer Adam [24] and the activation function SiLU [25] are adopted. As will be detailed in Section 3.2, the activation function "adaptive SiLU" facilitates the improvement of training convergence. At the initial stage (time step = 1), an exponential decay learning rate scheduler [26] is utilized in both methods (FT-NPM and TL-NPM), viz.,

$$\eta = \eta_0 \times \exp\left((\alpha - 1)\left\lfloor \frac{k}{n_d} \right\rfloor\right), \tag{8}$$

where η is the current learning rate with η_0 its initial value, $\alpha = 0.95$ denotes the decay rate, n_d stands for the decay steps, and k represents the current number of epochs (cf. Table 1). In (8), $\left\lfloor \frac{k}{n_d} \right\rfloor$ stands for the largest integer that is smaller than or equal to k/n_d. That is, the learning rate will decay once every n_d training steps. Similar to [15], two types of full training strategies are employed, respectively referred to as FT-40k and FT-80k, where, as listed in Table 1, FT-40k involves 40,000 epochs and 350 decay steps, and FT-80k consists of 80,000 epochs and 700 decay steps. Both FT-40k and FT-80k have the same initial learning rate of 10^{-3} and a decay rate of 0.95. As indicated in [15], the training with FT-80k shows better convergence. Hence, we choose FT-80k to ensure the convergence at the initial time step for our proposed transfer learning strategies, as the following time steps training rely on the model from the initial time step. For the subsequent time steps, FT-NPM will restart from a new NN-model, maintaining the same learning rate strategy. For the method with transfer learning TL-NPM, the trained

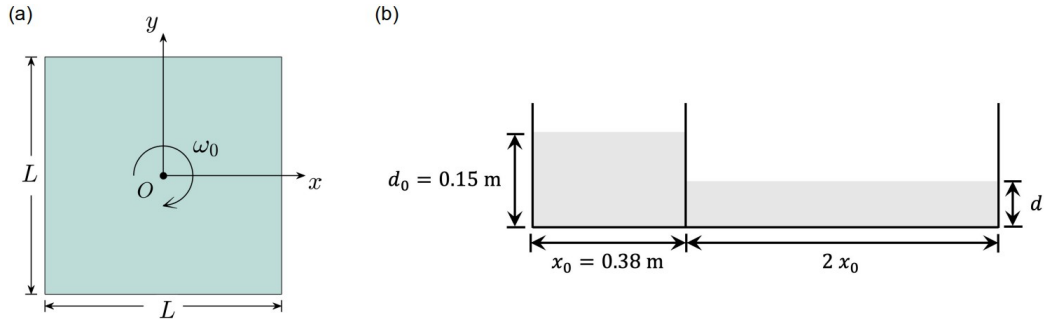

Figure 3: Initial configurations of the two testing examples. (a) The rotating square fluid patch; (b) The wave breaking problem, where the total length of the channel is three times that of the initial water column.

model continues fine-tuning with the tiny learning rate 10^{-5}, which is close to the final learning rate of the initial time step, as illustrated by the red-cross line in Figure 2. Here, the primary effect of TL-NPM on acceleration is the reduction in the number of required epochs. The performances for various number of fine-tuning epochs will be discussed in Section 3.

3 Numerical Investigation

Two numerical examples are examined in terms of efficiency, one involves a rotating square fluid patch and the other concerns the wave breaking phenomenon. Their initial configurations are given in Fig. 3, which are same as in [15]. For the fluid properties, in the testing case of rotating square fluid patch the fluid is inviscid, while the wave breaking is simulated with the viscosity of water. Hence, their derivatives for AD are of different orders, which means more back-propagation steps are needed for the wave breaking problem due to the viscous drags. We first explore the feasibility of FO-NPM with the first-order formulation (7). The transfer learning is then applied to FO-NPM, called TL-NPM. The practice of fine-tuning with various hyper-parameters is made in the following numerical examples.

3.1 Rotating square fluid patch

The first testing case is a classical benchmark example for examining the performance of a Lagrangian method. As shown in Fig. 3(a), a 2D square inviscid fluid patch, centered at $(0, 0)$ with both initial length and width $L = 1.0$ m, rotates at a constant angular speed $\omega_0 = 10.0$ s^{-1}. The the fluid density reads $\rho_0 = 1,000$ kg/m^3 and no gravity is taken into account, i.e., $(g_x = g_y = 0)$. Hence, the fluid patch starts with the velocity

$$\begin{cases} u(t=0) = +\omega_0\, y \\ v(t=0) = -\omega_0\, x \end{cases} . \tag{9}$$

The initial fluid particles are uniformly assigned with $\mathrm{d}x = \mathrm{d}y = 0.005$ m, including 2,401 interior particles and 200 boundary particles. For the time marching, a time interval of $\Delta t = 0.002$ s is used.

Driven by the angular speed ω_0, the square patch deforms in a spreading manner due to centripetal acceleration. Since the flow body is assumed to be inviscid, the vertices of the four corners move in straight lines and stretch with their initial velocities. Simultaneously, the fluid patch rotates, with the four corners forming slim arms. That is, the vertices should be located on the four straight lines and at the positions indicated by the cyan lines, as shown in Fig. 4. Hence, the performance of the numerical approach can be evaluated by comparing the exact positions of the vertices of four corners. The deviation $\Delta\ell$ infers the distance between the theoretical position and the position in simulation. In present study, the performance of the proposed TL-NPM are examined by the dimensionless deviation $\Delta\ell/L$ at dimensionless time level $t\omega_0 = 4.0$. To investigate the effectiveness between full training (FT) and transfer leaning (TL), and the impacts of AMP and number of epochs, a campaign is designed. The campaign consists of three groups. In the first group, we examine the accuracy and efficiency of the full-training method with 40,000 epochs (FT-40k), with 40,000 epochs and automatic mixed precision (FT-40k-AMP), with 80,000 epochs (FT-80k), and with 80,000 epochs and automatic mixed precision (FT-80k-AMP). The effects of the transfer learning are explored with respect to various epochs in the

second group, viz. 5,000 epochs (TL-05k), 10,000 epochs (TL-10k), 15,000 epochs (TL-15k), and 20,000 epochs (TL-20k). In the third group, the automatic mixed precision is employed together with the TL, 5,000 epochs (TL-05k-AMP), 10,000 epochs (TL-10k-AMP), 15,000 epochs (TL-15k-AMP), and 20,000 epochs (TL-20k-AMP) are used. The modeled results at $t = 4.0/\omega_0$ s are depicted in Figure 4, in which the color scale represents the dimensionless pressure field $p/(\rho_0\omega_0^2 L^2)$. In Fig. 4, the four corner vertices should theoretically align with the black dashed-lines and intersect with the cyan lines. The associated speed-up ratios, indicated within the brackets, are given in the corresponding panels, where the magnitudes are in terms of the training strategy with 80,000 epochs (FT-80k), the longest elapsed time. For the evolution of the rotating patch at various time levels, we refer the readers to [15].

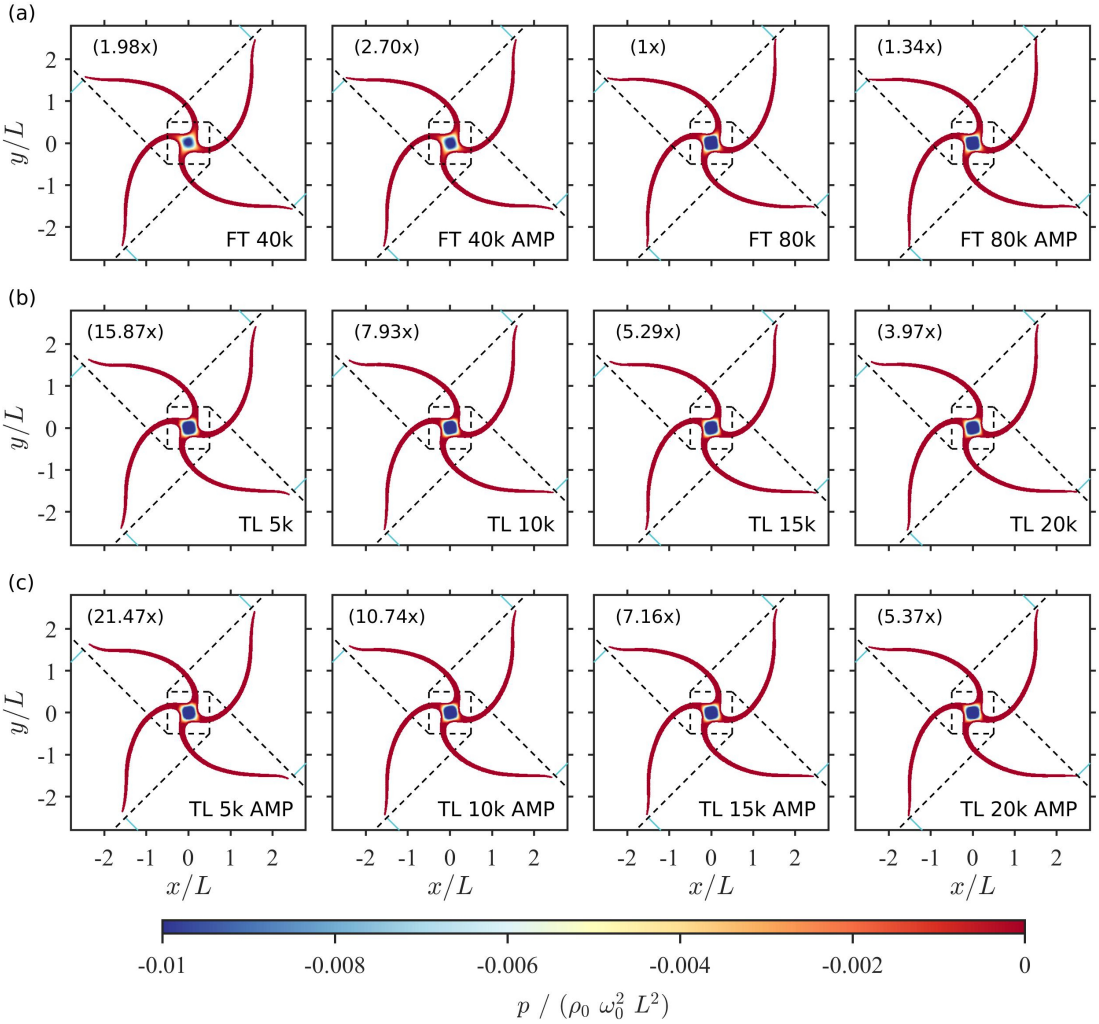

Figure 4: Snapshots of the shapes and the corresponding pressure fields at time level $t = 4.0/\omega_0$ s, where the results are modeled by (a) full training strategies (FT-NPM); (b) transfer learning strategies (TL-NPM); and (c) transfer learning strategies with automatic mixed precision (TL-AMP-NPM), respectively. In addition, the speed-up ratios, indicated within brackets, are based on the elapsed time of FT-80k.

Figure 5 visualizes the performance in terms of acceleration and accuracy with bar charts. The left panel demonstrates the speed up effect, where the reference elapsed time given by FT-80k is indicated by the cyan line. The dark blue and light blue bars in the right panel of Fig. 5 represent the average deviation and the associated standard deviation of the four corner vertices at $t = 4.0/\omega_0$ s, respectively, for evaluating the corresponding training strategies. Although FT-80k provides the highest accuracy, the results from FT-40k are acceptably accurate and offer significantly greater efficiency. Hence, we take the averaged deviation of FT-40k, depicted by the red line in Fig. 5, as the reference value for appraising the performance of the proposed training strategies. As already elaborated, there are three groups to explore the effectiveness of the proposed training strategies: the full training strategies (FT-NPM), the transfer learning strategies (TL-NPM), and the transfer learning strategies with automatic mixed precision (TL-

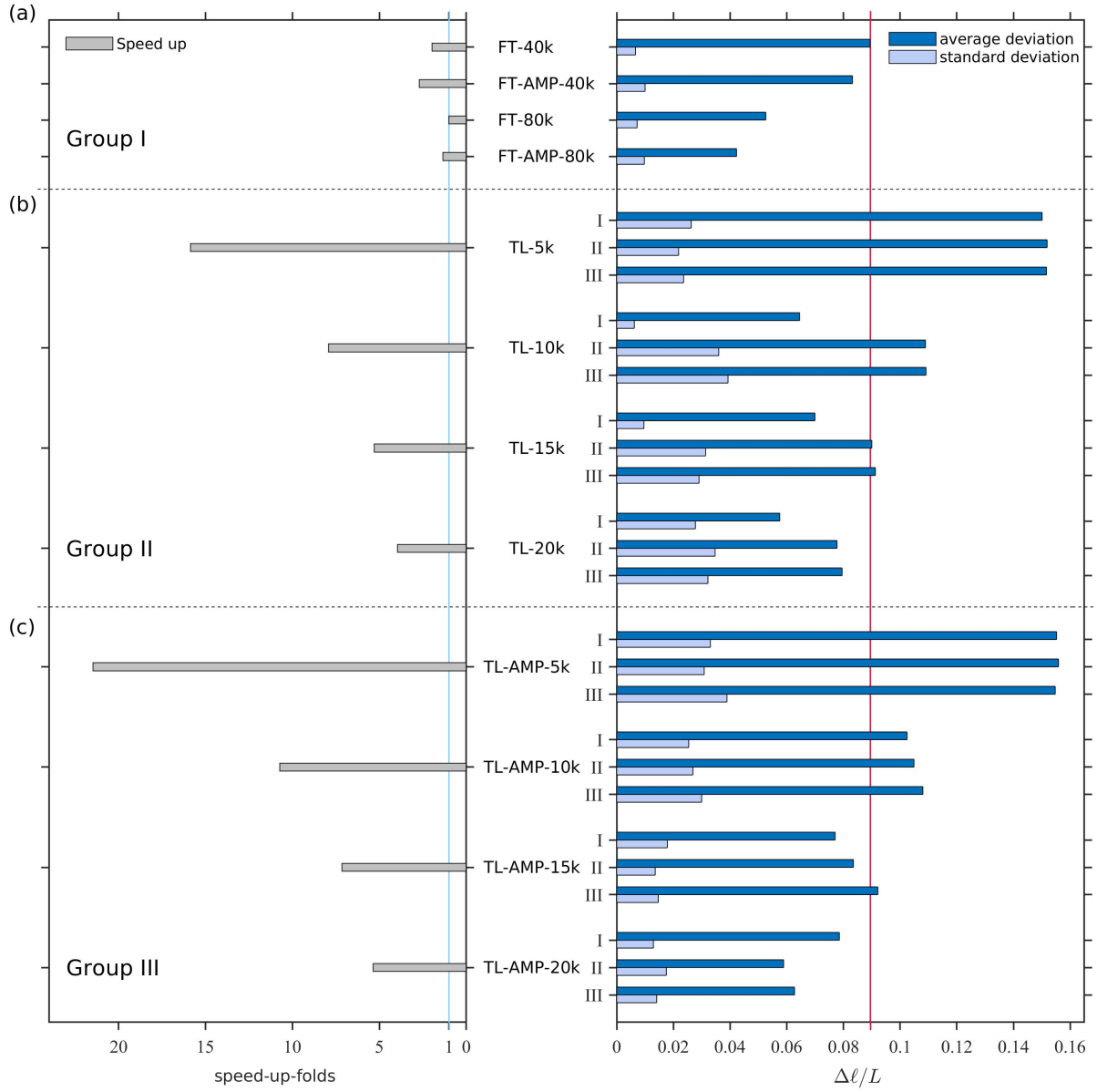

Figure 5: Speed-up ratios (left) and the corresponding accuracies (right) in modeling the rotating patch problem. Panel (a) illustrates the results modeled by the full training strategies (FT-NPM, Group-I), panel (b) represents the results modeled by the transfer learning strategies (TL-NPM, Group-II), and panel (c) shows the results modeled by strategies of transfer learning with automatic mixed precision (TL-AMP-NPM, Group-III).

AMP-NPM). The performances of the three groups are depicted in the right panel of Fig. 5. The detailed information of elapsed time and deviation are listed in Table 2.

FT-NPM

Since the rotating patch is composed of an inviscid fluid and therefore given in first-order formulation intrinsically, we can examine the influence of AMP without any additional effect caused by high order derivatives. Panels in row (a) of Fig. 4 show the results modeled by full training strategies, referred to as FT-NPM, with 40,000 epochs and 80,000 epochs at $t = 4.0/\omega_0$ s. For 40,000 epochs, an asymmetrical pressure field at the center region was found either with AMP (FT-AMP-40k) or without AMP (FT-40k). We can also notice the discrepancy in the corner vertices between the theoretical positions. For the results with 80,000 epochs, the symmetry of the pressure field and the position deviations of the corner vertices are both improved, demonstrating the benefits of increasing the number of epochs. As shown

Table 2: Elapsed time and deviation in numerical examples with various training strategies. The employed GPU device is NVIDIA RTX A5000.

		FO	AMP	ADA	Elapsed time (min) for one time step	Speed up	Deviation
Rotating square fluid patch (2,602 particles)	**FT-40k**	■			18.4	1.98×	0.089
	FT-40k	■	■		13.5	2.70×	0.083
	FT-80k	■			36.5	1.0 ×	0.053
	FT-80k	■	■		27.2	1.34×	0.042
	TL-05k	■			2.3	15.87×	0.152
	TL-10k	■			4.6	7.93×	0.109
	TL-15k	■			6.9	5.29×	0.090
	TL-20k	■			9.2	3.97×	0.078
	TL-05k	■	■		1.7	21.47×	0.156
	TL-10k	■	■		3.4	10.74×	0.105
	TL-15k	■	■		5.1	7.16×	0.084
	TL-20k	■	■		6.8	5.37×	0.059
Wave breaking (6,350 particles)	**FT-80k**				208.8	1.0 ×	–
	FT-80k	■			115.2	1.82×	0.0189
	FT-80k	■	■		70.4	2.97×	0.0234
	TL-10k	■			14.4	14.50×	0.030
	TL-15k	■			21.7	9.62×	0.0321
	TL-10k	■		■	18.4	11.35×	0.0210
	TL-15k	■		■	27.5	7.59×	0.0171
	TL-20k	■		■	37.0	5.64×	0.0178
	TL 25k	■		■	46.2	4.52×	0.0181
	TL 30k	■		■	55.4	3.77×	0.0196

in Fig. 5(a), training with AMP may reduce the averaged deviations in both cases of FT-AMP-40k and FT-AMP-80k. However, the use of AMP may lead to lower precision (cf. the light blue bars in the right panel), which is suspected to be induced by the instability of mixed precision. In terms of computational efficiency, training with AMP significantly reduces the run time, with a total elapsed time approximately 1.3 times faster compared to the standard training, i.e., FT-AMP-80k vs FT-80k, cf. Fig. 5(a). Hence, training with AMP for model euqations with first-order derivatives is a feasible option.

TL-NPM

In the campaign with TL-NPM, fine-tuning with various epoch number is used to examine the effectiveness of the proposed transfer learning strategies. The results of FT-NPM reveal that an increased number of epochs may enhance the stability of convergence, improve the symmetry of pressure field, and reduce the deviation of the trajectory of the corner vertices. The epoch number for fine-tuning processes starts from 5,000 to 20,000 with an increment of 5,000. For each epoch number, the whole training process is repeated for three times, indexed by run-I, run-II, and run-III in the right panel of Fig. 5, for a concise illustration of the sensitivity by transfer learning. The panels of row (b) in Fig. 4 illustrate the pressure fields of run-II with different fine-tuning epochs at $t = 4.0/\omega_0$ s. Comparing the results between FT-40k and TL-NPM, the transfer learning strategies (TL-05k, TL-10k, TL-15k and TL-20k) have improved the symmetry of the pressure field. It is also noted that larger fine-tuning epochs lead to more accurate results. That is, the positions of the corner vertices gradually approach the corresponding cross-points with the black dashed lines, inferring a positive correlation between the epoch number of fine-tuning and the accuracy of the associated results. The corresponding position deviations of the

corner vertices by the TL-NPMs are given in Fig. 5(b), where the average deviations become smaller in general when more fine-tuning epochs are used. As the fine-tuning epochs increase to 15,000, the average deviations become comparable to those modeled by FT-40k, of which the performance is taken as the judgement criteria in the present study, see the red line in the right panel of Fig. 5. Besides, the TL-15k enhanced the efficiency by about 5.3 times when compared to FT-40k, demonstrating the effectiveness of the proposed TL-NPM. Nevertheless, TL-NPM exhibits a random characteristic with respect to the position deviations (both for the averaged value and standard deviation) in the three runs (run-I, run-II, and run-III). This finding implies a slight instability in the training process compared to the results modeled by FT-NPM.

TL-AMP-NPM

Since the application of automatic mixed precision (AMP) can substantially reduces memory requirements [23], the efficiency and performance of fine-tuning with AMP was examined. As shown in the left panel of Fig. 5, AMP has slightly improved the efficiency in the four cases of epoch number (TL-AMP-05k, TL-AMP-10k, TL-AMP-15k and TL-AMP-20k). However, the position deviations depicted in Figs. 4(c) (and 5(c) for the averaged ones) do not show remarkable improvement compared to those of trained without AMP. The right panel of Fig. 5(c) reveals that, among the three runs, the average position deviations modeled by TL-NPM-AMP follow a similar trend to those modeled by TL-NPM, regardless of the number of fine-tuning epochs. Here, the random characteristics can also be observed in the results modeled by TL-NPM-AMP. Regarding efficiency, the elapsed time of running TL-NPM-AMP can be reduced by more than 1.3 times compared to running TL-NPM, while maintaining similar accuracy. Therefore, the proposed transfer learning strategies (with or without AMP) significantly highlight the acceleration effect while ensuring satisfactory performance levels, using the results from FT-40k as the reference criteria.

3.2 Wave breaking

The wave breaking problem is concerning a water column released by opening the gate suddenly, where a thin water layer exists in the channel. Once the water column is released, the flow pushes the thin water layer, forming a mushroom-shaped surge of waves. As the wave steepness increases, it becomes unstable and breaks. The process from formation to breaking dynamically exhibits a complex flow surface. Hence, we take the wave breaking problem as the second testing case to rigorously examine the proposed transfer learning strategies. Figure 3(b) illustrates the initial configuration of wave breaking, which follows the experiment setup of [27]. The initially stagnant water body has a depth of $d_0 = 0.15$ m and a length of $x_0 = 0.38$ m, with a shallow ambient layer of depth $d = 0.018$ m. In the modeling process, the fluid is set to have a density of $\rho = 1{,}000$ kg/m^3 with a viscosity of $\mu = 0.001$ Pa·s, and the time step $\Delta t = 0.005$ s is used for time marching. Additionally, because of the good agreement (in terms of the propagation of the wave front) between the experimental images and the results modeled with the non-linear opening gate opening speed [27, 15], we apply the same gate speed from [15] in the present wave breaking problem. Similar to Section 3.1, three groups of various training strategies are arranged, (a) Group A: full training strategies in first-order formulation (denoted by FO-NPM); (b) Group B: transfer learning strategies in first-order formulation (denoted by TL-FO-NPM); and (c) Group C: transfer learning strategies in first-order formulation with adaptive activation function (denoted by TL-FO-ADA-NPM).

In Figs. 6 and 7, the light-blue-shaded areas outlined by blue lines depict the results of wave breaking modeled by FT-NPM with 80,000 epochs (referred to as FT-80k) at time levels $t = 0.220$, 0.280, and 0.345 s. The red dots in panels of row (a) represent the outlines of the wave in experiment in [27] at approximately the same time levels, viz. $t = 0.219$, 0.281, and 0.343 s. In the present study, we use the results of FT-80k as the baseline for evaluating the performances for the various training strategies in modeling this wave breaking problem, including the first-order formulation (FO), transfer learning (TL), adaptive mixed precision (AMP), and adaptive activation function (ADA). The speed-up ratios of the various training strategies relative to the FT-80k strategy are given in the brackets of the right panels in Figs. 6 and 7. In order to quantitatively assess the accuracy of different training strategies, the deviation index suggested in [28] is employed, viz.

$$\Lambda_{\text{area}} = \frac{|A_{\text{FT}} - A_{\text{FT}\cap\text{TS}}| + |A_{\text{TS}} - A_{\text{FT}\cap\text{TS}}|}{|A_{\text{FT}\cap\text{TS}}|}, \tag{10}$$

which represents the area deviation of the specific training strategy from FT-80k. In (10), A_{FT} denotes

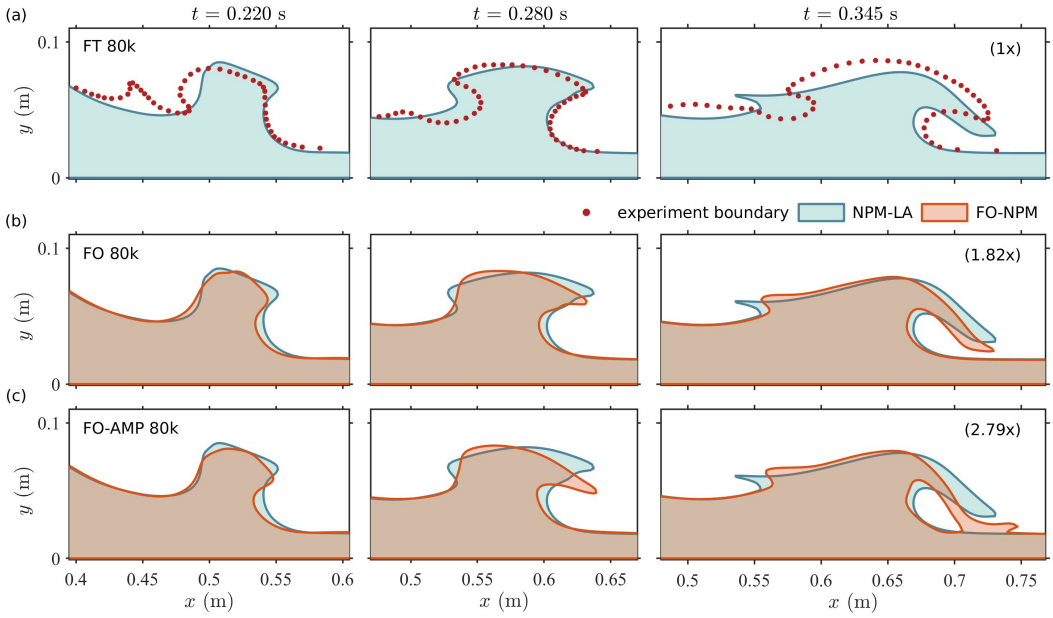

Figure 6: Results of shape evolution modeled by methods of Group A at time levels $t = 0.220,\ 0.280$, and 0.345 s. (a) The red dots outline the flow body in experiments in [27], while the light-blue shaded areas with blue outlines represent the results modeled by FT-80k. Panels (b) and (c) depict results modeled by FT-FO-80k and FT-FO-AMP-80k, respectively, with light-red shaded areas and red outlines. The speed-up ratio, shown in brackets, is calculated relative to the elapsed time of FT-80k.

the area modeled by FT-80k, A_{TS} is the area modeled by target training strategy (referred to as TS), and $A_{\mathrm{FT}\cap\mathrm{TS}}$ indicates their intersection. In other words, Λ_{area} is the ratio of the total area difference to the area of intersection modeled by FT-80k and the target training strategy.

Figure 8 visualizes the enhanced efficiencies (performance of acceleration, given by the speed-up ratio with respect to FT-80k) and the corresponding accuracies of the target training strategies. The speed-up ratios, calculated with respect to FT-80k, are indicated by the sky-blue line and depicted in the left section of Fig. 8. The right section of Fig. 8 illustrates the area deviations by light blue bars at three time levels, viz. $t = 0.220$, 0.280, and 0.345 s, and indexed by T1, T2, and T3, respectively. The dark blue bars indicate the associated average area deviations across the three time levels. In the right section of Fig. 8, the area deviation of FT-80k serves as the reference model and is marked by "norm". The efficiency and performance assessment are investigated in terms of the three assigned groups, i.e., FT-NPM, TL-FO-NPM, and TL-FO-ADA-NPM, respectively. The detailed information about the elapsed time and average area deviation are collected and listed in Table 2.

FT-NPM

In addition to the results of full training strategy (FT-80k) in Fig. 6(a) We illustrate the results of wave breaking problem trained by FT-FO-NPM and FT-FO-AMP-NPM in panels (b) and (c), respectively, where the flow body is depicted by the light-red shaded area with red outlines. The shaded areas are colored with transparency to clearly present the discrepancy with the shape modeled by FT-80k. Examining the results at time level $t = 0.220$ s in Fig. 6(b)(c), no significant discrepancy between FT-FO-NPM and FT-FO-AMP-NPM is identified, with the surge wave developing at approximately the same position. When time marches to $t = 0.280$ and 0.345 s, the mushroom-jet develops and the front edge tends to touch down the water level. Although the surges in the three strategies are located at approximately the same position, it is noted that the introduction of the first-order formulation (FO) leads to an earlier bending of the front edge (either by FT-FO-NPM or FT-FO-AMP-NPM). At $t = 0.345$ s, the front edge modeled by FT-FO-AMP-NPM has already touched the water level. On the other hand, the first-order formulation (FO) has postponed the development of the rear edge of the mushroom jet, where the rear edge did not appear in both results by FT-FO-NPM and FT-FO-AMP-NPM until $t = 0.345$ s. In addition, as shown in Fig. 8, the area deviations of FT-FO-AMP-NPM is more significant than those of FT-FO-NPM, and the gap increases as time marches. In terms of the runtime, both FO and AMP can

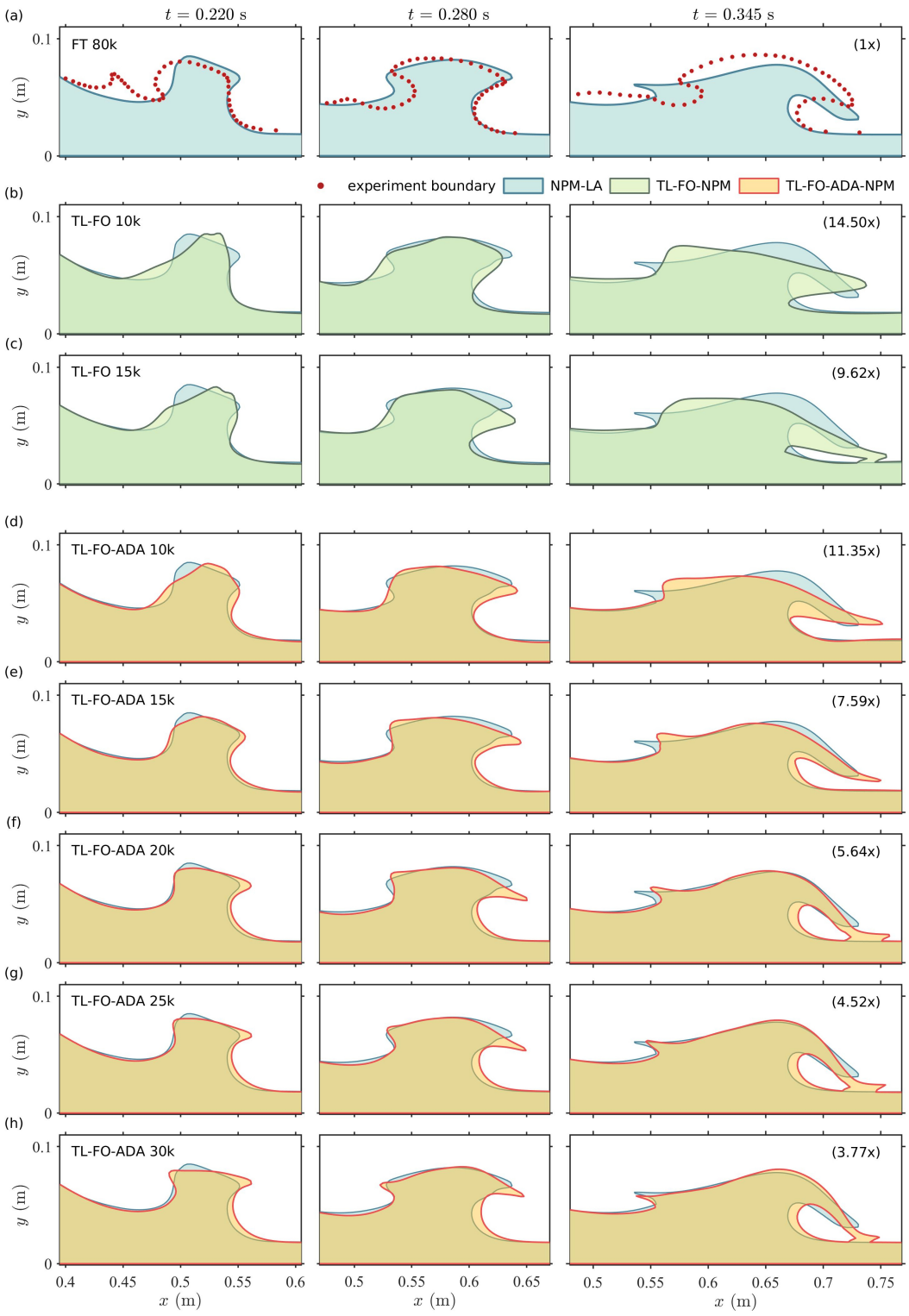

Figure 7: Results of shape evolution in modeling the wave breaking at time levels $t = 0.220,\ 0.280$, and 0.345 s. The red dots outline the flow body in experiments in [27], while the light-blue shaded areas with blue outlines are the results modeled by FT-80k. The light-green and light-yellow shaded areas illustrate the modeled by flow body in TL-FO-NPMs and TL-FO-ADA-NPMs, respectively. The speed-up ratio, with respect to the runtime by FT-80k, is given in brackets.

enhance the efficiency. Training with FT-FO-NPM can significantly accelerate the training process by about 2 times compared to FT-80k, while the acceleration can be up to 3 times for FT-FO-AMP-NPM. These findings reveal that, for equations consisting of second-order derivatives, the fist-order formulation

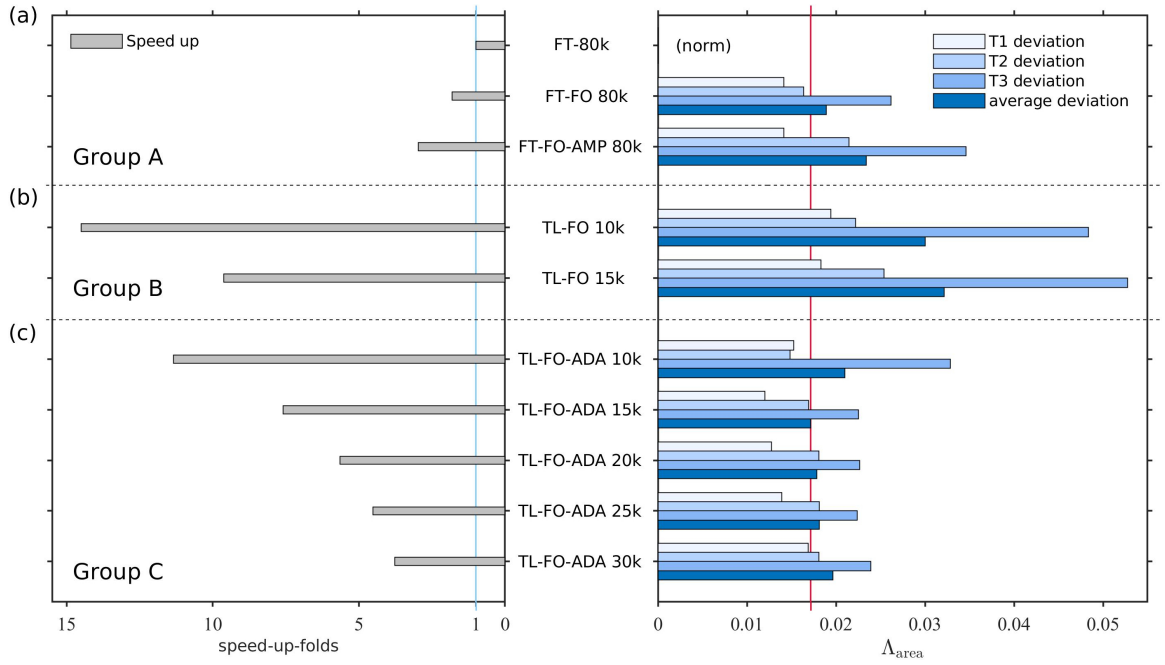

Figure 8: Performances of acceleration and accuracy in wave breaking with different training strategies. The left side and right side respectively present the speed-up effect and the deviation. On the right side, the light blue in gradient bar denotes the area deviation at the three time levels, while the dark blue bar represents the average area deviation across these time level. The sky-blue line on the left side serves as the baseline for FT-80k, and the red line on the right side is based on the average deviation of TL-15k. Panel (a) illustrates the full training strategies (FT-NPM, Group A), panel (b) represents the transfer learning strategies reformulated in first-order type (TL-FO-NPM, Group B), and panel (c) shows transfer learning strategies reformulated in first-order type with automatic mixed precision (TL-FO-ADA-NPM, Group C).

(FO-NPM) may serve as an alternative option for obtaining a rough result within limited computational resources and time constraints.

TL-FO-NPM

As the feasibility of FO-NPM has been confirmed in the previous paragraph, the transfer learning strategies, referred to as TL-FO-NPM, are examined in the same way, with $10,000$ and $15,000$ fine-tuning epochs (indexed as TL-FO-10k and TL-FO-15k, respectively) taken into account. The results of TL-FO-NPM at three time levels ($t = 0.220$, 0.280, and 0.345 s) are illustrated in panels (b) and (c) of Fig. 7 for TL-FO-10k and TL-FO-15k, respectively. The flow bodies are depicted by the light-green shaded area with green outlines, using a transparent illustration is used for ease of comparison with the results trained by FT-80k. However, it seems that the transfer learning strategy cannot improve the accuracy in this wave breaking problem, although efficiency has been significantly enhanced (cf. the right section of Fig. 8). That is, with both $10,000$ and $15,000$ fine-tuning epochs, the shapes of the flow body modeled by TL-FO-NPM exhibit substantial discrepancies to those trained by FT-80k, failing to precisely capture the physical characteristics of wave breaking. This discrepancy is suspected to be caused by insufficient convergence in the training process; therefore, the adaptive activation function proposed in [29] is therefore adopted.

TL-FO-ADA-NPM

To enhance the training convergence in TL-FO-NPM, we employ an adaptive activation function [29] into the transfer learning strategy, referred to as TL-FO-ADA-NPM. That is, trainable parameters are introduced in the activation function, and their values are adjusted based on the current training status. In TL-FO-ADA-NPM, the adaptive activation function is used for both the initial time step and the subsequent time steps, as the trained model is sequentially reused. As shown in panels from row (d) to

row (h) in Fig. 7, the transparent light-yellow shaded areas represent the results modeled by various fine-tuning epochs at the three time levels. At $t = 0.220$ s, more fine-tuning epochs increases the sharpness the front and rear edges of the mushroom jet. It is also noted that, as shown in Fig. 8, the use of more fine-tuning epochs does not essure better convergence in terms of area deviation, whereas the results trained by TL-FO-ADA-15k show the best agreement with FT-80k. At $t = 0.280$ s, fine-tuning with more than 15,000 epochs can precisely depict the rear edge of the mushroom, where the results trained by TL-FO-ADA 25k exhibit the best agreement with those by FT-80k. Similar to the results by TL-FO-NPM, the first-order formulation (FO) causes the earlier bending of the front edge. At $t = 0.345$ s, fine-tuning from 15,000 to 25,000 epochs give the similar performance in terms of area deviation. However, the front edge of the mushroom jet has touched the water level when the epoch number exceeds 20,000. This early touchdown is inconsistent with the experiment image in [27], and also yields larger area deviations with respect to the results trained by FT-80k. Consequently, the results trained by TL-FO-ADA-15k at this time level is supposed to be in a better agreement with the propagation of the water bore. In terms of the average area deviation across the three time levels, it is interesting that TL-FO-ADA-15k yields the best performance, rather than models with a higher number of fine-tuning epochs. Although it seems reminiscent of overfitting in machine learning, further evidence and validation are necessary to support this assertion. Regarding the runtime acceleration, TL-NPM-ADA can significantly reduce the elapsed time due to efficient computation for terms with second-order derivatives. The elapsed time of TL-FO-ADA-15k speeds up by over 7 times compared to FT-80k, demonstrating the high efficiency of TL-NPM-ADA. Therefore, the TL-NPM-ADA strategy showcases its ability to enhance efficiency while preserving acceptable accuracy for equations with second order derivatives.

4 Concluding remarks

In this study, we introduce transfer learning strategies for the Neural Particle Method (TL-NPM), a common technique in machine learning for accelerating training tasks by utilizing knowledge acquired from similar tasks. As the coordinates and physical parameters in NPM evolve over time, we employ the trained model from the initial time step to fine-tune the subsequent steps, a fundamental principle of transfer learning. By fine-tuning the model with several epochs, TL-NPM remarkably accelerates the training process while effectively capturing the evolving physical properties. Leveraging the first-order formulation of equations (FO-NPM) [18], TL-NPM is well-suited for implementing automatic mixed precision (AMP) to enhance training efficiency. Moreover, training with FO-NPM reduces the number of back-propagations, resulting in decreased memory footprint requirements and addressing hardware limitations for large-scale simulations. The proposed transfer learning strategies offer a promising solution for NPM to overcome the challenges of high computational resource demands and long runtime.

The performance of TL-NPM is first examined using the benchmark problem of a rotating square fluid patch, where the model equations are intrinsically formulated in first order due to the inviscid flow. The results indicate that increasing the number of fine-tuning epochs may enhance the accuracy, as evaluated by the average deviation of the four corner vertices. Taking the accuracy of the results trained with 40,000 epochs (FT-40k) as the norm, fine-tuning with more than 10,000 epochs can deliver the comparable performance. In terms of training efficiency, TL-NPM significantly accelerates the elapsed time. TL-NPM with 10,000 epochs (TL-NPM-10k) speeds up the training process almost 8 times compared to standard training with 80,000 epochs (FT-80k). Additionally, fine-tuning with AMP (TL-AMP-NPM) further enhances efficiency, resulting in a speed-up effect of over 10 times for TL-AMP 10k. Consequently, acceleration with AMP is approximately 1.3 times faster than TL-NPM, while maintaining the same level of accuracy. However, both TL-NPM and TL-AMP-NPM lead to higher standard deviations compared with FT-NPM, indicating a relatively unstable convergence in the training process. Despite this slight instability, the proposed transfer learning strategies offer remarkable acceleration without sacrificing accuracy, demonstrating their promising feasibility.

The capability of TL-NPM in describing complex flow behaviors is examined and demonstrated using the wave breaking problem. Since second-order derivatives appear in the model equations, we recast the model equations using the first-order (FO-NPM) formulation to enhance efficiency. Training with FO-NPM can mimic the shape modeled by standard full training with 80,000 epochs (FT-80k) and speed up the training process by 2 times, while training with FO-AMP-NPM leads to an early bending of the front edge, causing it to touch down to the water level. However, it is found that TL-NPM alone cannot precisely capture the phenomenon of wave breaking without additional treatment. With the integration of an adaptive activation function [29], the performance can be significantly improved. The shape of the rear edge of the mushroom jet becomes sharper with increasing fine-tuning epochs, indicating a

better performance of capturing the complex flow behavior. However, with TL-FO-ADA-NPM, higher fine-tuning epochs lead to an early bending and faster propagation of the front, which is inconsistent with the experimental results in [27]. As a result, these findings indicate that fine-tuning with 15,000 epochs (TL-FO-ADA-15k) presents the best agreement with FT-80k, and the efficiency is enhanced by up to 7.59 times, see Table 2.

In the present work, the proposed transfer learning strategies for NPM have demonstrated the capability to accelerate training processes while effectively describing various types of flow behaviors. Despite the performance of TL-NPM not being entirely comparable to FT-NPM due to the instability of convergence and the loss of some physical details, its practicality should not be underestimated, particularly for large scale applications. We demonstrate the feasibility of accelerating NPM through fine-tuning, showcasing the potential of basic transfer learning. Fine-tuning with 15,000 epochs has proven to be adequate strategy in the two numerical examples. However, further investigations on related hyperparameter and their associated calibrations are still needed, such as exploring the impacts of the number for fluid particles, or the effectiveness when model equations with terms of higher-order derivatives. The successful integration of transfer learning strategies in NPM marks a significant advancement in addressing computational efficiency. Further enhancing accuracy requires exploration of advanced transfer learning techniques, which we are currently working on and will report in due time.

Acknowledgments

The financial support of the National Science and Technology Council, Taiwan (NSTC 112-2221-E-006 -051-MY3) is sincerely acknowledged. We also thank to National Center for High-performance Computing (NCHC) in Taiwan for providing computational and storage resources.

References

[1] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[2] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[3] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[4] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. *Physics of Fluids*, 34(7), 2022.

[5] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[6] Ehsan Haghighat and Ruben Juanes. SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 373:113552, 2021.

[7] Henning Wessels, Christian Weißenfels, and Peter Wriggers. The neural particle method–an updated Lagrangian physics informed neural network for computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 368:113127, 2020.

[8] Gui-Rong Liu and Moubin B Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World scientific, 2003.

[9] Joe J Monaghan. Smoothed particle hydrodynamics. *In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574.*, 30:543–574, 1992.

[10] Seiichi Koshizuka and Yoshiaki Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear science and engineering*, 123(3):421–434, 1996.

[11] Seiichi Koshizuka, Atsushi Nobe, and Yoshiaki Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *International journal for numerical methods in fluids*, 26(7):751–769, 1998.

[12] Corné J Coetzee. Calibration of the discrete element method. *Powder Technology*, 310:104–142, 2017.

[13] Jinshuai Bai, Ying Zhou, Yuwei Ma, Hyogu Jeong, Haifei Zhan, Charith Rathnayaka, Emilie Sauret, and Yuantong Gu. A general neural particle method for hydrodynamics modeling. *Computer Methods in Applied Mechanics and Engineering*, 393:114740, 2022.

[14] Kaixuan Shao, Yinghan Wu, and Suizi Jia. An improved neural particle method for complex free surface flow simulation using physics-informed neural networks. *Mathematics*, 11(8):1805, 2023.

[15] Pei-Hsin Pai, Heng-Chuan Kan, Hock-Kiet Wong, and Yih-Chin Tai. A neural particle method with interface tracking and adaptive particle refinement for free surface flows. *Communications in Computational Physics*, page (accepted), 2024.

[16] Paul Escapil-Inchauspé and Gonzalo A Ruz. h-Analysis and data-parallel physics-informed neural networks. *Scientific Reports*, 13(1):17562, 2023.

[17] Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

[18] Rini J Gladstone, Mohammad A Nabian, and Hadi Meidani. FO-PINNs: A first-order formulation for physics informed neural networks. *arXiv preprint arXiv:2210.14320*, 2022.

[19] Thommen George Karimpanal and Roland Bouffanais. Self-organizing maps for storage and transfer of knowledge in reinforcement learning. *Adaptive Behavior*, 27(2):111–126, 2019.

[20] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[21] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[22] Zhipeng Wang, Xuejun Liu, Jian Yu, Haizhou Wu, and Hongqiang Lyu. A general deep transfer learning framework for predicting the flow field of airfoils with small data. *Computers & Fluids*, 251:105738, 2023.

[23] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[26] Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019.

[27] Imre Jánosi, Dominique Jan, K Gábor Szabó, and Tamás Tél. Turbulent drag reduction in dam-break flows. *Experiments in Fluids*, 37:219–229, 2004.

[28] Chih-Ling Wang, Chi-Jyun Ko, Hock-Kiet Wong, Pei-Hsin Pai, and Yih-Chin Tai. An approach for preliminary landslide scarp assessment with genetic algorithm (ga). *Water*, 14(15):2400, 2022.

[29] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.