

A Physics-Informed Machine Learning Framework for Solid Boundary Treatment in Meshfree Particle Methods

Nariman Mehranfar^a, Ahmad Shakibaeinia^a

*^aDepartment of Civil, Geological, and Mining Engineering, Polytechnique
Montreal, , Montreal, Canada*

Abstract

Meshfree particle methods such as Smoothed Particle Hydrodynamics (SPH) and the Moving Particle Semi-Implicit (MPS) method are widely used to simulate complex free-surface and multiphase flows. A key challenge in these methods is the treatment of solid boundaries, where kernel truncation causes errors and instabilities. Traditional treatments, such as ghost particles and semi-analytical wall corrections, restore kernel completeness but add significant computational cost and complexity, especially for irregular geometries. We propose a physics-informed machine learning (ML) framework that directly predicts boundary correction terms for particle approximations, eliminating the need for ghost particles or analytical corrections. The framework is based on a hybrid convolutional neural network–multilayer perceptron (CNN–MLP) trained on physics-informed features that capture local geometry, particle states, and kernel properties. Once trained, it provides consistent boundary contributions across all spatial differential operators, including gradients, divergences, and Laplacians. The approach is demonstrated with MPS but is readily extensible to other particle methods such as SPH. Tests with predefined fields, unsteady diffusion, and incompressible Navier–Stokes flows show an accuracy comparable with ghost-particle methods while reducing computational overhead. The model generalizes well to unseen geometries, flow conditions, and particle distributions, including dynamically evolving domains. This work establishes a flexible, physics-informed ML paradigm for boundary treatment in particle-based PDE solvers, improving both accuracy and scalability across a broad class of meshfree methods.

Keywords: Meshfree particle methods, MPS method, Solid boundary treatment, Physics-informed machine learning (ML), Boundary correction

1. Introduction

Meshfree Lagrangian (particle) methods such as Smoothed Particle Hydrodynamics (SPH) [1, 2], and Moving Particle Semi-implicit (MPS) [3] are well-suited for fluid simulations, especially in problems involving large deformations and interface fragmentation. These methods represent the domain with freely moving particles, where the governing equations are approximated through a kernel smoothing process. SPH and MPS share many similarities, but they differ primarily in how spatial derivatives are approximated. In SPH, derivatives are computed by differentiating the kernel function, whereas in MPS, the kernel itself is used as a weight function, and the derivative operator is applied directly to the field variable.

A major challenge in both SPH and MPS is the treatment of solid boundaries, which strongly affects their accuracy, stability, and overall reliability. Near solid boundaries, the kernel support domain is truncated, leading to incomplete summation of particle contributions. This causes density deficiencies and spurious pressure gradients toward the boundary, which can lead to particle penetration to the boundary. Kernel truncation also increases approximation errors, especially for derivative estimates. In SPH, neglected surface-integral terms in the approximation of derivatives further contribute to errors near boundaries. While in MPS, derivatives approximation using directionally weighted differences mitigates this error, it still suffers from incomplete or asymmetric neighbor contributions. These issues directly impact the consistency and conservation properties of both methods. To address boundary-related errors, many approaches have been developed in the past, which broadly fall into three categories [4–12]:

1. Repulsive force methods that introduce short-range repulsive forces (e.g., Lennard–Jones-type) to prevent fluid particles from crossing the wall [13–15]. They are simple to implement, but often suffer from poor physical fidelity and weak conservation

properties [16].

2. Ghost and dummy particle methods, which are among the most widely used approaches, populate the region outside the boundary with fictitious particles to restore kernel support. Particles can be placed in fixed predefined positions (commonly referred to as dummy particles) [3, 17–20], or dynamically mirrored from nearby fluid particles (often called ghost particles) [21–27]. As the literature uses these terms interchangeably, in this paper, we refer to all such particles as ghost particles for simplicity. While effective and physically consistent, these methods can be computationally expensive [28]. The number of ghost particles can exceed that of fluid particles in large domains, significantly increasing memory and run-time costs. Defining and updating them is also challenging for complex geometries.
3. Semi-analytical and hybrid approaches that explicitly correct for kernel truncation without relying on extra particles. Examples include semi-analytical wall boundary conditions (SAW) [29, 30], and methods like the Unified Semi-Analytical Wall (USAW) [5] and Polygon Wall (PW) boundary conditions [6, 31, 31–37]. While more direct and often accurate, they can also be computationally demanding, especially when complex analytical corrections are required [31]. Approximations for complex geometries are also not always straightforward.

Artificial intelligence (AI) and machine learning (ML) are increasingly being applied in computational science and engineering, including fluid mechanics, to complement traditional physics-based numerical models [38–46]. Recent studies show that machine learning (ML) is being used in numerical modeling of PDEs in various ways. At one end of the spectrum, fully data-driven surrogate models can replace an entire solver [38, 47]. In other cases, ML works alongside conventional numerical methods, with the goal of acceleration, enhancement, and optimization. For instance, ML can help approximate certain components of a PDE, like spatial derivatives or source terms, while the numerical solver handles the rest, like in neural ODEs and discrete-time

PINNs [40, 41]. Another approach is to use ML selectively for specific tasks (often computationally expensive or physically uncertain ones), such as learning turbulence closures, constitutive laws, or solving auxiliary algebraic equations like the pressure Poisson equation [38, 39]. ML is also commonly applied for parameter estimation, calibration, and model closure in physics-based simulations [39, 41]. Success of these examples illustrates how ML can help traditional physics-based modeling.

In recent years, particle methods have also increasingly benefited from machine learning (ML) techniques. Early work by Ladický et al. [48] showed that particle positions and velocities could be learned using regression forests, while Marinho [49] employed k-nearest neighbors (kNN) to design an anisotropic SPH kernel. Bai et al. [50] proposed a chained hashing algorithm for data-driven constitutive modeling in SPH. Neural networks have also been applied in different contexts—for example, Xiaoxing et al. [51] used them to compute interface curvature in surface-tension models. More recently, physics-informed approaches have emerged, such as the Lagrangian formulations of physics-informed neural networks (PINNs) developed by Wessels et al. [52] and Bai et al. [50].

Alongside these efforts, researchers have explored how particle neighbor lists can be leveraged as physics-informed aggregations of local information. Woodward et al. [53] and Alexiadis [54] drew parallels between neighbor lists in particle-based systems and convolutional layers in grid-based data, highlighting their role in reducing computational complexity. Woodward et al. [53] also developed reduced Lagrangian models for turbulence with varying levels of ML involvement, while Tian et al. [55] extended ML applications to turbulence characterization. More recently, Zhang et al. [56] replaced the pressure Poisson equation (PPE) in incompressible SPH (ISPH) with a CNN-based surrogate, achieving faster simulations without loss of accuracy. Despite this growing body of work, to the best of our knowledge, no ML framework has yet been developed specifically for boundary treatments in particle methods.

Building on recent progress in applying ML to numerical methods, this paper introduces a physics-informed, data-driven framework for solid boundary treatment in par-

ticle methods, demonstrated here for the MPS scheme. The goal is to develop a flexible and efficient alternative that avoids the complexity and cost of conventional approaches, such as ghost particles. The proposed model learns from ghost-particle data to predict boundary correction terms for each MPS operator, thereby replacing contributions previously computed by traditional treatments. To the best of our knowledge, this is the first ML framework specifically designed for boundary corrections in particle discretizations. The architecture employs a hybrid convolutional neural network–multilayer perceptron (CNN–MLP), which processes physics-inspired features—including geometric descriptors, field variables, and kernel/boundary properties, to predict boundary contributions. Separate models are trained for number density, gradient, divergence, and Laplacian operators. The training datasets cover diverse geometries and field conditions to promote generalization. The models can be considered physics-informed, as they learn from features and datasets derived from established physics-based methods, and they predict boundary contributions that remain consistent with the underlying physics of MPS.

The framework’s performance and generalization are assessed on three test cases with unseen geometries: (1) a spatially varying prescribed field with static particles; (2) a spatio-temporally varying field obtained from the solution of a pure diffusion PDE with static particles; and (3) a spatio-temporally varying field with dynamic particles governed by the Navier–Stokes and transport equations. In all cases, the ghost-particle approach provides the ground truth data. This study demonstrates the potential of physics-informed ML to replace conventional boundary treatments in MPS, offering a foundation for future extensions to three-dimensional problems and other particle-based methods.

2. Methodology

2.1. MPS Particle Method

The Moving Particle Semi-Implicit (MPS) method is a widely used meshfree particle scheme for incompressible flows, originally introduced by Koshizuka and Oka [3].

Here, we briefly summarize its standard formulations to establish the baseline for the proposed ML-based boundary treatment.

2.1.1. Particle approximations

In MPS, similar to other particle methods, the continuum is represented by a set of freely moving nodes, referred to as particles, which carry physical quantities. The field variables and their derivatives are approximated through kernel-weighted interactions between each particle target i , with position vector $\mathbf{r}_i$, and its neighboring particles j , with position vector at $\mathbf{r}_j$. The interaction weighted by a kernel function $W_{ij} = W(r_{ij}, r_e)$, where $r_{ij} = \|\mathbf{r}_{ij}\| = \|\mathbf{r}_j - \mathbf{r}_i\|$ is the relative position vector (distance), and r_e is the kernel support radius (Fig. 1) [3, 57].

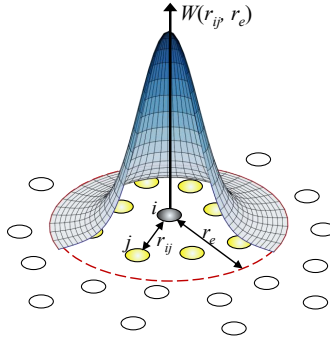


Figure 1: Kernel function in the 2-Dimensional MPS method

The kernel-weighted convolution of a scalar field ϕ is given by :

$$\langle \phi(\mathbf{r}) \rangle = \frac{\int_v \phi(\mathbf{r}') W_{ij} dv'}{\int_v W_{ij} dv'} \quad (1)$$

which in discrete form gives the MPS interpolation formula as [3]:

$$\langle \phi(\mathbf{r}) \rangle_i = \frac{\sum_{j \neq i} (\phi_j W_{ij}) V_j}{\int_v W_{ij} dv} = \frac{\sum_{j \neq i} (\phi_j W_{ij})}{\sum_{j \neq i} W_{ij}}, V_j = \frac{m_j}{\rho_j} \quad (2)$$

Here, m_j is the mass of the particle j , and ρ_j is its density. Unlike in the SPH method, in the MPS, the approximations are normalized, so there is no requirement

for a partition of unity, and the kernel function is dimensionless. The normalization factor, known as the particle number density, is defined as $\langle n \rangle_i = \sum_{j \neq i} W_{ij}$, which serves as a dimensionless measure of local particle concentration. When the material density is constant (e.g., in incompressible flows), the particle number density can be replaced by a constant initial value n_0 . Therefore, the interpolation operator can be written as:

$$\langle \phi(\mathbf{r}) \rangle_i = \frac{1}{n_0} \sum_{j \neq i} (\phi_j W_{ij}) \quad (3)$$

The standard MPS approximation for spatial derivatives is based on kernel-weighted pairwise interactions between a target particle i and each of its neighbors j . The MPS formulations for the gradient, divergence, and Laplacian operators (for scalar field ϕ and vector field $\boldsymbol{\Phi}$) are given by Koshizuka and Oka [3]:

$$\langle \nabla \phi \rangle_i = \frac{d}{n_0} \sum_{j \neq i} \left(\frac{\phi_{ij}}{r_{ij}} \mathbf{e}_{ij} W_{ij} \right) \quad (4)$$

$$\langle \nabla \cdot \boldsymbol{\Phi} \rangle_i = \frac{d}{n_0} \sum_{j \neq i} \left(\frac{\boldsymbol{\Phi}_{ij}}{r_{ij}} \cdot \mathbf{e}_{ij} W_{ij} \right) \quad (5)$$

$$\langle \Delta \phi \rangle_i = \langle \nabla^2 \phi \rangle_i = \frac{2d}{n_0 \lambda} \sum_{j \neq i} (\phi_{ij}) W_{ij} \quad (6)$$

$$\langle \Delta \boldsymbol{\Phi} \rangle_i = \langle \nabla^2 \boldsymbol{\Phi} \rangle_i = \frac{2d}{n_0 \lambda} \sum_{j \neq i} (\boldsymbol{\Phi}_{ij}) W_{ij} \quad (7)$$

where $\phi_{ij} = \phi_j - \phi_i$ in scalar field and $\boldsymbol{\Phi}_{ij} = \boldsymbol{\Phi}_j - \boldsymbol{\Phi}_i$ in vector field, d is the number of space dimensions, $\mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij}$ is the unit direction vector between particles i and j , and λ is a normalization factor defined as:

$$\lambda = \langle r_{ij}^2 \rangle_i = \frac{\sum_{j \neq i} r_{ij}^3 W_{ij}}{\sum_{j \neq i} r_{ij} W_{ij}} \quad (8)$$

Throughout this work, we adopt the third-order polynomial spiky function proposed by Shakibaeinia and Jin [19] as the kernel function:

$$W_{ij} = \begin{cases} \left(1 - \frac{r_{ij}}{r_e}\right)^3, & \text{if } r_{ij} \leq r_e \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

Note that the formulations presented above correspond to the standard MPS approximations. Over the years, various alternative formulations have been developed to improve conservation properties (e.g., [23, 58–60]). A comprehensive review of these developments can be found in [61]. In this study, we primarily follow the original formulation to maintain consistency and facilitate comparison. The only exception is the adoption of the improved gradient formulation proposed by Jandaghian and Shakibaeinia [60], which is necessary for one of our test cases. It is given by:

$$\langle \tilde{\nabla} \phi \rangle_i = \frac{d}{n_0} \sum_{j \neq i} \left(\left(n_i \frac{\phi_j}{n_j} + n_j \frac{\phi_i}{n_i} \right) \mathbf{e}_{ij} W_{ij} \right) \quad (10)$$

2.1.2. MPS for flow and transport simulation

Here, we briefly describe the application of the basic MPS formulations for simulating incompressible flow and heat transport, which is the phenomenon modeled in one of the test cases in this study. The governing equations consist of the mass and momentum conservation equations (Navier–Stokes), the advection-diffusion equation for heat transport. In the Lagrangian framework, advection terms in all equations are naturally absorbed, and a separate equation of motion is added. The governing equations are therefore given by [62]:

$$\frac{1}{\rho} \frac{D\rho}{Dt} + \nabla \cdot \mathbf{u} = 0 \quad (11)$$

$$\frac{D\rho \mathbf{u}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}_b \quad (12)$$

$$\frac{D\rho C_p T}{Dt} = k \nabla^2 T \quad (13)$$

$$\frac{D\mathbf{r}}{Dt} = \mathbf{u} \quad (14)$$

where $\mathbf{u}$ is the velocity vector, t is time, p is pressure, ρ is density, μ is the kinematic viscosity, T is temperature, k is the thermal conductivity, C_p is the specific heat capacity, and $\mathbf{f}_b$ represents the body force vector. Pressure is computed using a weakly compressible approach (WC-MPS) [19], which employs an explicit equation of state (EoS) to relate pressure to density, i.e., $p = f(\rho)$. The MPS discretization of the spatial derivatives in the governing equations is given by:

$$\langle \nabla p \rangle_i = \frac{d}{n_0} \sum_{j \neq i} \left(\frac{p_{ij}}{r_{ij}} \mathbf{e}_{ij} W_{ij} \right) \quad (15)$$

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \frac{d}{n_0} \sum_{j \neq i} \left(\frac{\mathbf{u}_{ij}}{r_{ij}} \cdot \mathbf{e}_{ij} W_{ij} \right) \quad (16)$$

$$\langle \nabla^2 \mathbf{u} \rangle_i = \frac{2d}{n_0 \lambda} \sum_{j \neq i} \mathbf{u}_{ij} W_{ij} \quad (17)$$

$$\langle \nabla^2 T \rangle_i = \frac{2d}{n_0 \lambda} \sum_{j \neq i} T_{ij} W_{ij} \quad (18)$$

For the pressure gradient, one can use Jandaghian and Shakibaeinia [60] alternative formulation:

$$\langle \tilde{\nabla} p \rangle_i = \frac{d}{n_0} \sum_{\substack{j=1 \\ j \neq i}}^N \left(n_i \frac{p_j}{n_j} + n_j \frac{p_i}{n_i} \right) \frac{\mathbf{e}_{ij}}{r_{ij}} W_{ij}. \quad (19)$$

Time integration is performed using a predictor–corrector scheme. The predictor step calculates the predicted velocity, $\mathbf{u}_i^*$, and predicted position, $\mathbf{r}_i^*$, as:

$$\mathbf{u}_i^* = \mathbf{u}_i^k + \frac{\Delta t}{\rho_i} (\mathbf{f}_b + \mu \nabla^2 \mathbf{u}_i^k) \quad (20)$$

$$\mathbf{r}_i^* = \mathbf{r}_i + \mathbf{u}_i^* \Delta t \quad (21)$$

The corrected velocity, $\mathbf{u}'$, is calculated in the correction step as:

$$\mathbf{u}' = -\frac{\Delta t}{\rho_i} \nabla p^{k+1} \quad (22)$$

Here, the superscript k denotes the current time step. The updated velocity at the new time step is then given by:

$$\mathbf{u}_i^{k+1} = \mathbf{u}_i^* + \mathbf{u}'$$

The pressure is calculated using the equation of state as:

$$p_i^{k+1} = \frac{\rho c_0^2}{\gamma} \left[\left(\frac{\langle n^* \rangle_i}{n_0} \right)^\gamma - 1 \right] \quad (23)$$

In this expression, γ is a constant (typically taken as 7), and c_0 is an artificial speed of sound. The predicted particle number density $\langle n^* \rangle_i$ is calculated based on the predicted positions $\mathbf{r}_i^*$.

To improve the stability and accuracy of the method, we use an artificial density diffusion term in the continuity equation following the DMPS technique [60] and employ a particle regularization approach known as dynamic pairwise collision (DPC) [63].

2.1.3. MPS Boundary treatment

As mentioned, boundaries pose challenges to the MPS method (similar to SPH) because the kernel support is truncated when a target particle is closer than r_e to a boundary (Fig. 2a). Several boundary treatment techniques have been developed to address kernel truncation and density deficiencies near boundaries in MPS (and similarly in SPH). The most widely used approach employs ghost/dummy particles, which populate the compact support across the boundary to restore kernel completeness and improve accuracy (Fig. 2b). Their physical properties (e.g., velocity, pressure) are either prescribed directly or extrapolated from neighboring interior particles, depending on the boundary condition type. Although ghost particle methods are effective, they struggle with treating complex boundaries and can have significant computational and memory overhead.

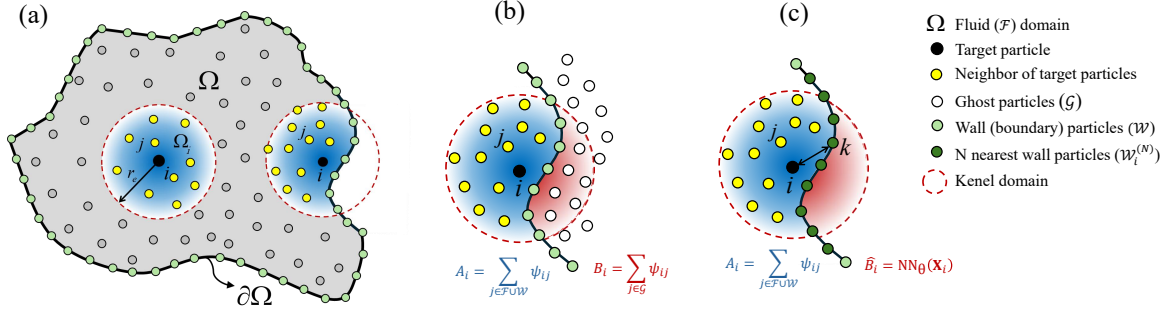


Figure 2: Boundary treatment strategies in the MPS method. (a) Kernel truncation near boundaries. (b) Ghost particle method, which introduces artificial particles to compensate for the truncated kernel. (c) Proposed ML-based boundary treatment that predicts the missing boundary contribution.

2.2. Machine Learning for Boundary Conditions

2.2.1. Machine Learning Contribution in MPS

Using machine learning instead of ghost particle methods in MPS boundary conditions offers several key benefits. It reduces the overall number of particles by eliminating the need for ghost particles, which leads to lower memory usage. Additionally, it simplifies the complexity of implementing boundary conditions by bypassing the need for complex ghost particle setups, making the method more flexible and easier to apply.

To develop the method, let's first express the MPS approximation of various operators ($\mathcal{L}$) applied to a scalar or vector field ϕ over a target particle i in a general form as:

$$C_i = \langle \mathcal{L}\phi \rangle_i = \sum_{j \neq i} \psi_{ij}, \quad (24)$$

where ψ_{ij} defines the interaction between the particle i and each of its neighbors j . Table 1 provides the expression ψ_{ij} for the standard MPS approximation of various operators ($\mathcal{L}$), defined in section 2.1.1, including for Number Density (n), Interpolation ($\mathcal{I}$), Gradient (∇ and $\tilde{\nabla}$), Divergence ($\nabla \cdot$), and Laplacian (Δ).

Table 1: Expression ψ_{ij} based on the standard MPS approximation of operators $\mathcal{L}$

$\mathcal{L}$	n	$\mathcal{I}$	∇	$\tilde{\nabla}$	$\nabla \cdot$	∇^2
ψ_{ij}	W_{ij}	$\frac{1}{n_0} \phi_j W_{ij}$	$\frac{d}{n_0} \left(\frac{\phi_{ij}}{r_{ij}} e_{ij} \right) W_{ij}$	$\frac{d}{n_0} \left(n_i \frac{\phi_j}{n_j} + n_j \frac{\phi_i}{n_i} \right) \frac{e_{ij}}{r_{ij}} W_{ij}$	$\frac{d}{n_0} \left(\frac{\phi_{ij}}{r_{ij}} \cdot e_{ij} \right) W_{ij}$	$\frac{2d}{n_0 \lambda} \phi_{ij} W_{ij}$

For the ghost particle approach, the formulation can be split into two components of A_i the contribution of the fluid ($\mathcal{F}$) and wall ($\mathcal{W}$) particles, and B_i the contribution of the ghost particles ($\mathcal{G}$), representing the boundary impact, expressed as:

$$C_i = \langle \mathcal{L}\phi \rangle_i = A_i + \underbrace{B_i}_{\text{boundary impact}} \quad \text{where} \quad A_i = \sum_{j \in \mathcal{F} \cup \mathcal{W}} \psi_{ij}, \quad \text{and} \quad B_i = \sum_{j \in \mathcal{G}} \psi_{ij} \quad (25)$$

The goal of the machine learning here will be to eliminate the ghost particles, and use a neural network $\text{NN}_\theta(\mathbf{X}_i)$, with parameters θ and input features $\mathbf{X}_i$ to predict a learnable correction term as boundary impact $\hat{B}_i$ (previously given by ghost particles contribution) as:

$$\hat{C}_i = \langle \mathcal{L}\phi \rangle_i = A_i + \underbrace{\hat{B}_i}_{\text{boundary impact}} \quad \text{where} \quad A_i = \sum_{j \in \mathcal{F} \cup \mathcal{W}} \psi_{ij}, \quad \text{and} \quad \hat{B}_i = \text{NN}_\theta(\mathbf{X}_i) \quad (26)$$

Note that we keep the wall particles as they represent the boundary (see Fig. 2c). The input feature vector $\mathbf{X}_i$ is given by:

$$\mathbf{X}_i = \left\{ \left(\mathbf{r}_i, \mathbf{r}_k, \mathbf{r}_{ik}, \|\mathbf{r}_{ik}\|, \|\mathbf{r}_{ik}\|^2, Nw_i, A_i, \phi_i, \phi_k, \phi_{ik}, \phi_{ik} \mathbf{e}_{ik}, d_p, \frac{r_e}{d_p}, n_i, \lambda_i, \text{BC}_k \right) \mid k \in \mathcal{W}_i^{(N)} \right\} \quad (27)$$

and is defined to contain the following physics-inspired components:

1. **Geometrical characteristics**, including: the position of the target particle, $\mathbf{r}_i$, the positions of the N nearest wall particles $\mathbf{r}_k$ (for all $k \in \mathcal{W}_i^{(N)}$, where $\mathcal{W}_i^{(N)} \subset \mathcal{W}$), the relative position of the target particle and its N nearest wall particles, $\mathbf{r}_{ik} = \mathbf{r}_k - \mathbf{r}_i$, and its magnitude (i.e., distance) $\|\mathbf{r}_{ik}\|$ and squared magnitude $\|\mathbf{r}_{ik}\|^2$, as well as the number of wall particles in the neighborhood (within support area) of i , given by $Nw_i = \sum_{j \in \mathcal{W}} \mathbb{I}(\|\mathbf{r}_{ij}\|^2 < r_e)$ where $\mathbb{I}(\cdot)$ is the indicator function. Note that while $\mathbf{r}_{ik}$ already contains the information to derive $\|\mathbf{r}_{ik}\|$ and $\|\mathbf{r}_{ik}\|^2$, in practice this redundancy can help learning by explicitly providing the model with useful nonlinearities included in the physics.

2. **Field variable characteristics**, including: contributions of fluid and wall particles, A_i , field variable values at the target particle, ϕ_i , field variable values at the N nearest wall particles, ϕ_k , the difference $\phi_{ik} = \phi_k - \phi_i$, and the direction product $\phi_{ik}\mathbf{e}_{ik}$. Note that in the case of boundary contribution to particle number density (n), only A_i is required.
3. **Kernel and boundary characteristics**, including: particle size d_p , normalized effective radius r_e/d_p , particle number density n_i , the coefficient in the Laplacian approximation formula λ_i , and the boundary condition type BC_k (Dirichlet or Neumann). Through this study, we keep the kernel function type constant.

2.2.2. Machine Learning Architecture

The machine learning model of this study includes a hybrid deep learning architecture that combines a feature extractor to capture local dependencies with a convolutional neural network (CNN) and a predictor (of boundary contribution $\hat{B}_i$) with a multilayer perceptron (MLP). Fig. 3 shows the architecture of this hybrid CNN-MLP network. Importantly, we found that MLP-only networks—even when made several times deeper than the hybrid design—failed to generalize beyond the training set, underscoring the need for the CNN-based feature extraction.

First, a subset of the input feature vector $\mathbf{X}i$ is extracted, corresponding to the N nearest wall particles ($N = 9$ in this study, Fig. 2c). This subset, denoted as $\mathbf{X}_{w,i}$, is used as input to the CNN to learn the geometric relationship between the wall particles and characterize the shape of the local boundary. CNN is particularly well-suited for this task due to its inherent ability to efficiently capture local patterns and spatial dependencies (defining the local boundary shape). Note that N nearest wall particles are reordered so that they follow the natural progression along the boundary, required for CNN. The feature vector is defined as:

$$\mathbf{X}_{w,i} = \left\{ (\mathbf{r}_k, \mathbf{r}_{ik}, \|\mathbf{r}_{ik}\|, \|\mathbf{r}_{ik}\|^2, \phi_k, \phi_{ik}, \phi_{ik}\mathbf{e}_{ik}, \text{BC}_k) \mid k \in \mathcal{W}_i^{(N)} \right\} \quad (28)$$

For scalar fields ϕ , this results in 11 features (f) per wall neighbor; for vector

fields $\boldsymbol{\phi}$, there are 12 features per neighbor. The CNN input is therefore a 1D vector $\mathbf{X}_{w,i} \in \mathbb{R}^{fN}$. With $N = 9$, this gives $fN = 99$ for scalars and $fN = 108$ for vectors. The CNN then processes this input by treating each feature, collected across the N wall neighbors, as a separate 1D sequence $\mathbf{X}_{w,i}^{(f)} \in \mathbb{R}^N$. These sequences are passed through three 1D convolutional layers, each with three filters of size 3. The outputs are then flattened, resulting in a vector with the same length as $\mathbf{X}_{w,i}$.

This vector is further processed by two fully connected layers with 64 and 32 units, respectively. The resulting vector, denoted by $\mathbf{X}'_i$, is then concatenated with the full input feature vector $\mathbf{X}_i$ (a fusion of physics-inspired and data-driven inputs) and passed through a four-layer MLP with 128, 64, 32, and 16 units to predict the boundary contribution $\hat{B}_i$. The length of $\mathbf{X}_i$ and $\mathbf{X}_{w,i}$ for each operator is summarized in Table 2. As mentioned, for particle number density n , all features that containing field variable ϕ (or $\boldsymbol{\phi}$) are excluded. The output layer of the network can be a vector or scalar, depending on the operation.

Table 2: Lengths of input feature vectors used in the network

Differential Operator $\mathcal{L}\phi$	Length of $\mathbf{X}_i$	Length of $\mathbf{X}_{w,i}$
n	61	54
$\langle \nabla \phi \rangle_i$	109	99
$\langle \tilde{\nabla} \phi \rangle_i$	109	99
$\langle \nabla \cdot \boldsymbol{\phi} \rangle_i$	118	108
$\langle \nabla^2 \phi \rangle_i$	108	99
$\langle \nabla^2 \boldsymbol{\phi} \rangle_i$	119	108

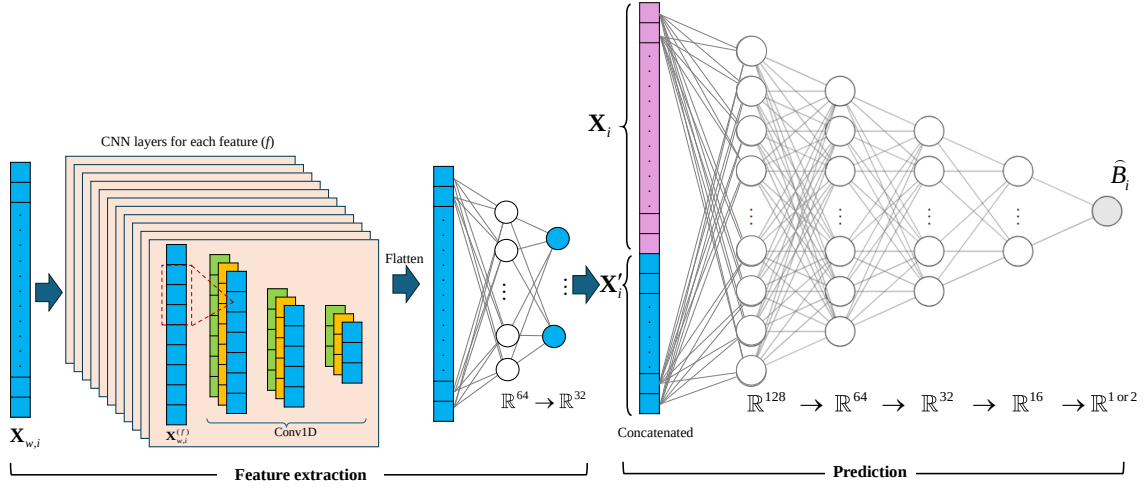


Figure 3: Schematic of Neural Networks Architecture

The activation function used in the feature extractor block (comprising the CNN layers and the first MLP) is ELU, while the predictor network (the second MLP) uses ReLU. All kernels are initialized using the **He normal** initializer. Biases are initialized with a **random normal** distribution having a mean of 0 and a standard deviation of $\sqrt{2/\sigma}$, where σ is the number of input units to the layer (see Table 2). The output layer uses a **linear** activation function, with no special initialization for its kernel or bias.

The input features $\mathbf{X}_i$ and $\mathbf{X}_{w,i}$ are normalized using the mean and standard deviation computed from the training dataset. The network is trained using the **AdamW** optimizer with a scheduled learning rate and weight decay. The learning rate starts at 10^{-3} and decays to 10^{-5} over 1000 steps. The weight decay follows the same schedule, decreasing from 10^{-3} to 10^{-5} over the same period.

A custom loss function is employed to improve model convergence:

$$\text{Loss} = \text{MAE} + (1 - \max(R, 0)) \quad (29)$$

where MAE is the mean absolute error, and R is the Pearson correlation coefficient given by:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |\hat{B}_i - B_i| \quad (30)$$

$$R = \frac{\sum_{i=1}^N (B_i - \bar{B})(\hat{B}_i - \bar{\hat{B}})}{\sqrt{\sum_{i=1}^N (B_i - \bar{B})^2} \sqrt{\sum_{i=1}^N (\hat{B}_i - \bar{\hat{B}})^2}} \quad (31)$$

Training is conducted over 2000 epochs, and the model achieving the best validation performance is selected as the final model, representing the cross-validation trade-off. This architecture was determined by testing various configurations of layer/unit counts for the two MLPs, and different activation functions including **ELU**, **ReLU**, **tanh**, **sigmoid**, **leaky ReLU**, and **swish**. The number and size of convolutional kernels were also optimized through experimentation. Training is done using a batch size of 16,384 on a PC with an NVIDIA Tesla P40 (24 GB) GPU, AMD Ryzen 9 5950x (16 cores, 32 threads) CPU, and 32 GB DDR4-3200 MHz RAM.

2.3. Training dataset

To train the hybrid CNN-MLP network for each MPS operator, we construct a dataset of boundary contributions from diverse 2D test cases with the following characteristics:

- **Geometry:** Four complex geometries featuring various shapes, including convex and concave polygons as well as curved boundaries (Fig. 4).
- **Particle and kernel size:** Three particle diameters ($d_p = 0.01, 0.02, 0.03$, m) and five kernel radii ($r_e = 2.1d_p, 2.6d_p, 3.1d_p, 3.6d_p, 4.1d_p$) are considered.
- **Variable field:** For scalar fields, 62 predefined functions are used (summarized in Eq. (32)), capturing combinations of linear, polynomial, or periodic variations (with various frequencies). For vector fields, we define $\boldsymbol{\phi}_i = (\phi_i, -\phi_i)$.

$$\begin{aligned}\phi(x, y) &= \pm f_1(2\alpha\pi x) \cdot f_2(2\beta\pi y), & f_1, f_2 &\in \{\sin, \cos\}, & \alpha, \beta &\in \{1, 2\} \\ \phi(x, y) &= \pm x^m y^n, & m, n &\in \{0, 1, 2, 3\}\end{aligned}\tag{32}$$

- **Boundary conditions:** Each geometry is simulated under eight boundary condition configurations, with half of the boundary length assigned a fixed value (Dirichlet condition) and the other half a zero-gradient (Neumann condition) (Fig. 5).



Figure 4: Geometries used for making the training set

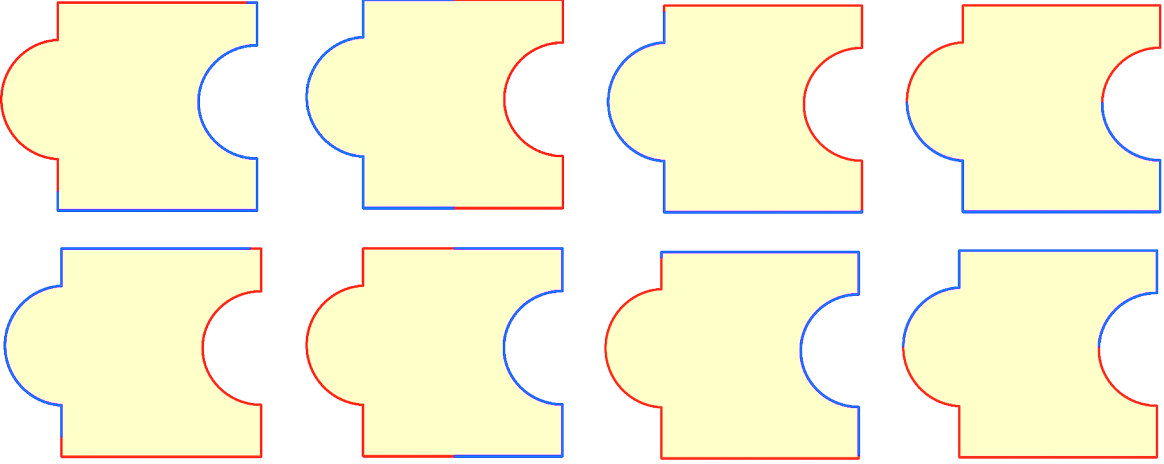


Figure 5: Example of boundary condition configurations used in the training set. Dirichlet boundaries are shown in red, and Neumann boundaries are shown in blue.

For each MPS operation, this setup yields approximately 37,000 test cases and around 24 million data rows, based on the number of particles per case. However,

most particles are unaffected by boundaries (i.e., they have no wall particles within their kernel support), which reduces training efficiency and accuracy. Therefore, these particles are filtered out, leaving a refined dataset of approximately 7.75 million rows. Of this, 80% (≈ 6.2 millions) is used for training, 10% ($\approx 775,000$) for validation, and 10% for testing. The labels are generated using MPS with the ghost particle approach.

2.4. Test cases

We evaluate the model’s generalizability on three test cases of increasing complexity, all featuring unseen conditions. Each case has a geometry that differs from both the training data and the other test cases. A key distinction lies in how the field variable is defined: in **Case 1**, it is predefined using an analytical function; in **Case 2**, it is derived from the solution of an unsteady diffusion PDE; and in **Case 3**, it results from solving the unsteady Navier–Stokes equations coupled with advection–diffusion (representing flow and heat transfer). Figure 6 illustrates the geometry, boundary conditions, and particle distribution for the test cases. Note that ghost particles are employed only in the ghost particle approach used to evaluate the developed model. Table 3 summarizes characteristics of these three test cases.

Case 1 (Predefined Function): Figure 6 shows the geometry, boundary conditions, and particle representation of this case. The geometry includes complex convex and concave curves and corners, distinct from those in the training dataset. Boundary conditions are shown in are zero-gradient (Neumann) and fixed-value (Dirichlet) boundaries. The case uses a particle diameter of 0.02 m and a kernel radius of $3.1d_p$. A new set of analytical functions, not included in the training set, is used to define the scalar field as:

$$\phi = \sin(2\pi\lambda x) \sin(2\pi\lambda y)xy, \quad \lambda \in \{2, 3, 4\} \quad (33)$$

The functions are periodic with variable spatial frequencies, including values higher than those present in the training data. Both the geometry and the field functions in this case are more complex than those used in training. The trained hybrid CNN-MLP

model predicts the boundary contributions to all MPS operators, including particle density, and scalar/vector gradients and Laplacians.

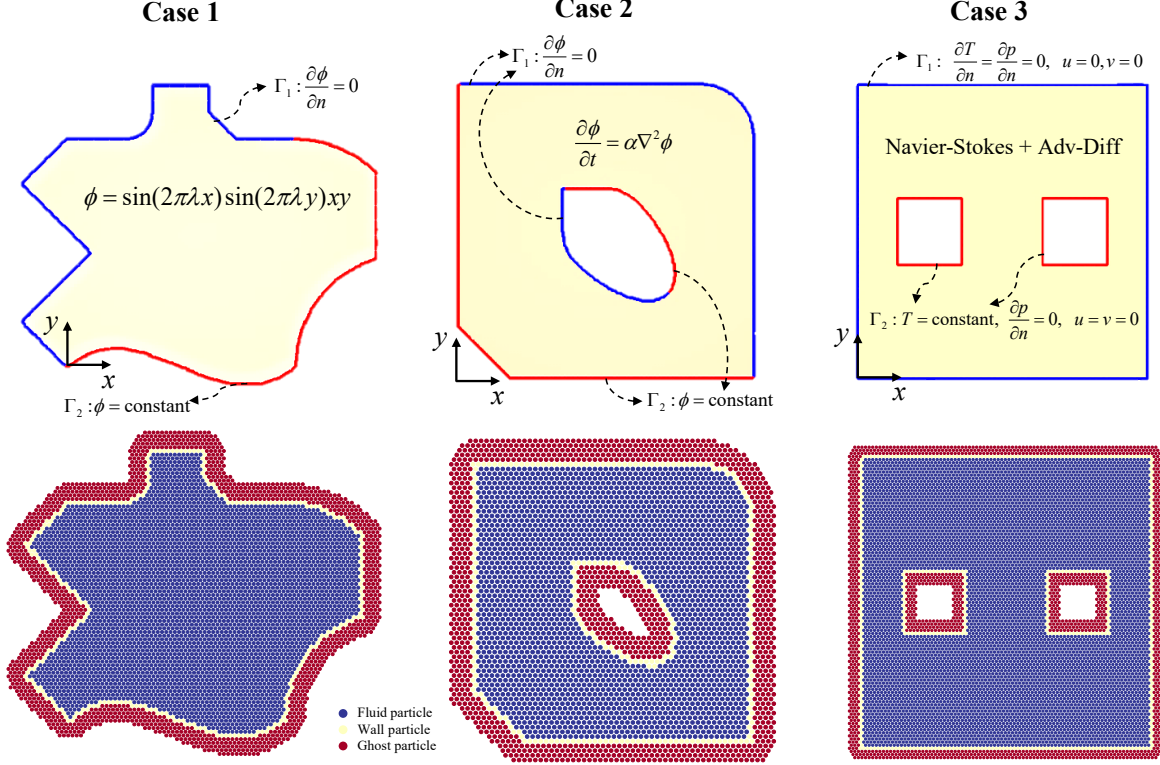


Figure 6: Geometry and boundary conditions (top row), and the particle representation and types (bottom row) for three test cases of this study.

Case 2 (Unsteady pure diffusion): Boundary conditions again include the Dirichlet and Neumann types. The case uses a particle diameter of 0.02 m and a kernel radius of $3.1d_p$. In this case, the variable field is given by the solution of a PDE, i.e., 2D unsteady pure diffusion as:

$$\frac{\partial \phi}{\partial t} = \alpha \nabla^2 \phi \quad (34)$$

with diffusion coefficient $\alpha = 1 \text{ m}^2/\text{s}$. The only spatial derivative in this case is the scalar Laplacian, which is approximated using the MPS formulation. The trained model predicts the boundary contribution to the Laplacian operator without relying on ghost particles, and the results are compared with those obtained using the ghost

particle method. Time integration is performed using the explicit Euler method with a time step $\Delta t = 0.001\text{s}$.

Case 3 (Navier–Stokes with advection–diffusion): This test case involves coupled flow and heat transfer, governed by the unsteady incompressible Navier–Stokes equations with an advection–diffusion equation for temperature (see Section 2.1.2 for governing equations and MPS numerical solution). The geometry is shown in Figure 6 and includes a square domain with two square cavities. The particle diameter is $d_p = 0.0125\text{m}$, and the kernel radius is set to $3.2d_p$.

The temperature boundary conditions are defined with a fixed value of 1 (hot) on the left square cavity, a fixed value of -1 (cold) on the right square cavity, and zero-gradient (Neumann) conditions on all external boundaries. For this case, the density $\rho = 1 \text{ kg m}^{-3}$, dynamic viscosity $\mu = 0.004 \text{ kg m}^{-1} \text{ s}^{-1}$, and the Rayleigh number $Ra = 1.6 \times 10^6$ (see Section Appendix A). This test case introduces additional complexity due to fluid motion, resulting in a dynamic particle distribution that is not encountered during training. The trained model predicts boundary contributions to the MPS operators in this unsteady, multi-physics scenario without relying on ghost particles.

Table 3: Summary of test case characteristics

Feature	Case 1	Case 2	Case 3
Physics	None (predefined field)	Pure diffusion	Flow and heat transfer
Governing Equation	Analytical function: $\phi = \sin(2\pi\lambda x) \sin(2\pi\lambda y)xy$	PDE: $\frac{\partial\phi}{\partial t} = \alpha \nabla^2 \phi$	PDEs: (N.S. & A.D.)
Particle Distribution	Static	Static	Dynamic
Boundary Conditions	Dirichlet, Neumann	Dirichlet, Neumann	Dirichlet, Neumann
Similarity to Training	Unseen geometry, similar physics	Unseen physics & geometry	Unseen physics & geometry
Particle Diameter (d_p)	0.02 m	0.02 m	0.0125 m
Kernel Radius (r)	$3.1d_p$	$3.1d_p$	$3.2d_p$
Examined operators	$n, \nabla\phi, \tilde{\nabla}\phi, \nabla \cdot \boldsymbol{\Phi}, \nabla^2\phi, \nabla^2\boldsymbol{\Phi}$	$n, \nabla^2\phi$	$n, \tilde{\nabla}p, \nabla \cdot \mathbf{u}, \nabla^2\mathbf{u}, \nabla^2T$

3. Results and discussion

3.1. Training evaluation

As previously mentioned, the dataset consists of 7.75 million data points, of which 80% were used for training, 10% for validation, and the remaining 10% for testing. The model was trained using a cross-validation trade-off to determine the optimal weights of the CNN-MLP (see Appendix B). To evaluate the accuracy of the trained hybrid CNN-MLP model, in addition to the Pearson correlation coefficient (R) discussed earlier, we employ two additional metrics: the normalized mean absolute error (NMAE) and the normalized root mean square error (NRMSE), defined as:

$$\text{NMAE}(\%) = \frac{1}{N} \sum_{i=1}^N \frac{|\hat{B}_i - B_i|}{\max(B) - \min(B)} \times 100, \quad (35)$$

$$\text{NRMSE}(\%) = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{B}_i - B_i)^2}}{\max(B) - \min(B)} \times 100 \quad (36)$$

where $\hat{B}$ and B are predicted and true (from ghost particle method) boundary contributions as in equations 25, and 26, respectively. The model's performance is evaluated for particle number density n and five differential operators (n , $\nabla\phi$, $\tilde{\nabla}\phi$, $\nabla \cdot \boldsymbol{\phi}$, $\nabla^2\phi$, and $\nabla^2\boldsymbol{\phi}$) as summarized in Table 4. The results show consistently low errors and high accuracy across all datasets, indicating excellent model generalization, robustness, and no signs of overfitting. Slightly higher errors are observed for the second-order operators, particularly the vector Laplacian $\nabla^2\boldsymbol{\phi}$, likely due to their greater sensitivity to local variations and noise in the data.

Table 4: Model Performance for training dataset

Case	Metrics	Targets								
		n	$\nabla\phi_x$	$\nabla\phi_y$	$\tilde{\nabla}\phi_x$	$\tilde{\nabla}\phi_y$	$\nabla^2\phi$	$\nabla \cdot \Phi$	$\nabla^2\Phi_x$	$\nabla^2\Phi_y$
Training	R	1.000	0.984	0.989	1.000	1.000	0.980	0.997	0.999	0.999
	NMAE (%)	0.253	0.369	0.338	0.067	0.075	0.385	0.170	0.074	0.079
	NRMSE (%)	0.390	0.965	0.767	0.176	0.171	1.611	0.500	0.265	0.280
Validation	R	1.000	0.979	0.987	1.000	1.000	0.972	0.996	0.999	0.999
	NMAE (%)	0.254	0.390	0.353	0.070	0.078	0.429	0.186	0.078	0.082
	NRMSE (%)	0.390	1.099	0.852	0.189	0.183	1.914	0.584	0.272	0.273
Testing	R	1.000	0.979	0.987	1.000	1.000	0.972	0.996	0.998	0.998
	NMSE(%)	0.253	0.389	0.353	0.071	0.078	0.426	0.185	0.078	0.082
	NRMSE (%)	0.391	1.109	0.846	0.194	0.181	1.888	0.579	0.278	0.296

3.2. Generalizability beyond training conditions

3.2.1. Case 1: Predefined Function

Figures 7 and 8 present the MPS calculated particle number density, first-order derivatives, and second-order derivatives, where the boundary contributions are predicted by the developed ML model, alongside the corresponding ground truth values computed using the ghost particle method. These figures also display the difference between the predicted and reference fields, enabling a visual assessment of the model’s accuracy.

Overall, excellent agreement is observed between the ML predictions and the ground truth for all operators. The predicted fields closely match the true values, demonstrating the model’s capability to generalize to new, unseen test cases. Minor discrepancies are observed near a few sharp boundary corners, and this behavior appears consistently across all operators, including the particle number density. Since the boundary contribution of the particle number density depends only on geometry (with no dependence on field variables), these deviations can be attributed to geometrical complexities not captured during training.

To further quantify the model’s performance, Figures 9 and 10 show the parity plots (predicted vs. ground truth) and corresponding error histograms for each predicted quantity. The parity plots show a strong alignment along the ideal $R = 1$ line, indicating high accuracy of the ML predictions. The error histograms show that most error values are tightly clustered around zero, confirming the model’s reliability and precision.

To evaluate the model’s generalization capacity, Table 5 compares the correlation coefficient R , NMSE, and NRMSE of the ML predictions for different spatial frequency parameters λ , including values beyond the training range (up to three times higher). Additionally, a case with artificially introduced non-uniformity in the particle distribution is considered, achieved by randomly perturbing particle positions using Gaussian noise with a standard deviation of $\pm 2\% d_p$. This aims to account for the physical and numerical fluctuations that commonly occur during flow simulations in particle methods.

Among the differential operators evaluated, the ML model consistently shows the highest prediction accuracy for the Laplacian of vector components ($\nabla^2 \boldsymbol{\phi}_x, \nabla^2 \boldsymbol{\phi}_y$) and the divergence operator ($\nabla \cdot \boldsymbol{\phi}$). The scalar Laplacian ($\nabla^2 \phi$) is also predicted with good accuracy, exhibiting slightly higher errors than the vector components but remaining robust across all conditions. The model’s performance is slightly less for the gradient components ($\nabla \phi_x, \nabla \phi_y$). Prediction of the particle number density n remains consistently accurate across all cases.

The results demonstrate that the developed model generalizes well across a range of spatial frequencies, maintaining consistently high correlation coefficients and NMSE and NRMSE for most differential operators. The model also shows strong robustness to moderate particle distribution non-uniformity. Prediction accuracy remains reliable for both first- and second-order derivatives, as well as for the particle number density. Overall, only a modest decline in performance is observed as spatial frequency and non-uniformity increase, confirming the model’s robustness and generalization capacity for practical applications.

Table 5: Model performance for test case 1 for various particle distributions and frequency coefficients (λ)

Condition	Metrics	Targets								
		n	$\nabla\phi_x$	$\nabla\phi_y$	$\tilde{\nabla}\phi_x$	$\tilde{\nabla}\phi_y$	$\nabla^2\phi$	$\nabla \cdot \Phi$	$\nabla^2\Phi_x$	$\nabla^2\Phi_y$
$\lambda = 2$, Uniform	R	0.996	0.860	0.899	0.993	0.985	0.967	0.976	0.991	0.991
	NMAE(%)	0.350	1.146	1.010	0.220	0.259	0.653	0.321	0.243	0.244
	NRMSE (%)	1.874	3.141	2.681	0.765	1.065	1.909	1.243	0.959	0.955
$\lambda = 3$, Uniform	R	0.996	0.895	0.923	0.974	0.983	0.947	0.987	0.994	0.994
	NMAE(%)	0.350	1.070	1.047	0.471	0.350	0.961	0.282	0.221	0.220
	NRMSE (%)	1.874	3.160	2.941	1.842	1.269	2.826	0.812	0.762	0.758
$\lambda = 4$, Uniform	R	0.996	0.882	0.908	0.980	0.979	0.956	0.965	0.983	0.982
	NMAE(%)	0.350	1.186	0.979	0.481	0.458	0.578	0.504	0.302	0.301
	NRMSE (%)	1.874	2.965	2.581	1.468	1.409	1.723	1.675	1.126	1.135
$\lambda = 2$, Nonuniform	R	0.996	0.890	0.918	0.978	0.973	0.950	0.980	0.988	0.988
	NMAE(%)	0.350	1.004	0.896	0.396	0.396	0.752	0.313	0.276	0.278
	NRMSE (%)	1.874	2.831	2.470	1.394	1.515	2.318	1.123	1.111	1.108

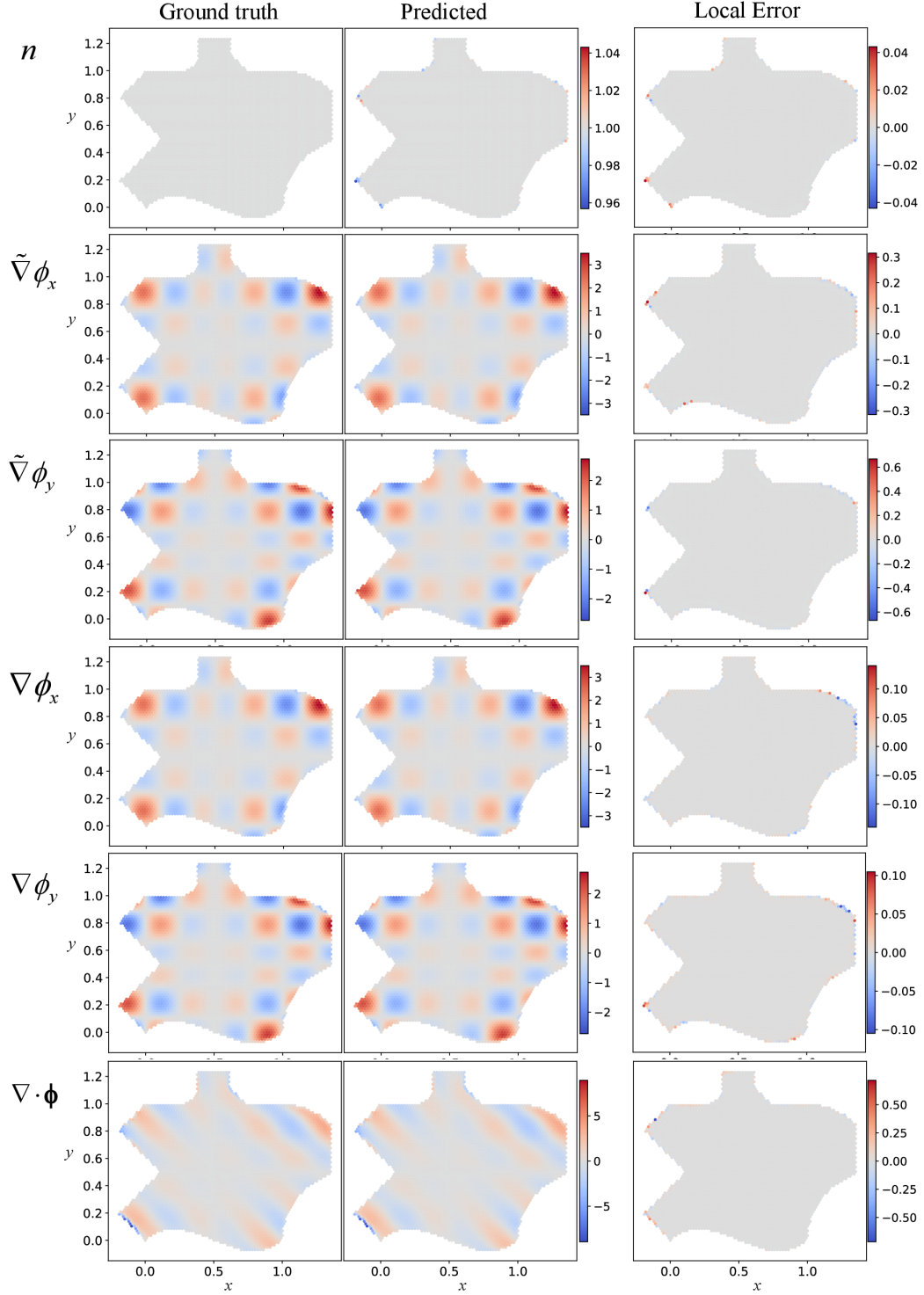


Figure 7: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for particle number density (n) and first-order derivatives ($\nabla\phi$, $\tilde{\nabla}\phi$, and $\nabla \cdot \phi$) in Case 1 ($\lambda = 2$)

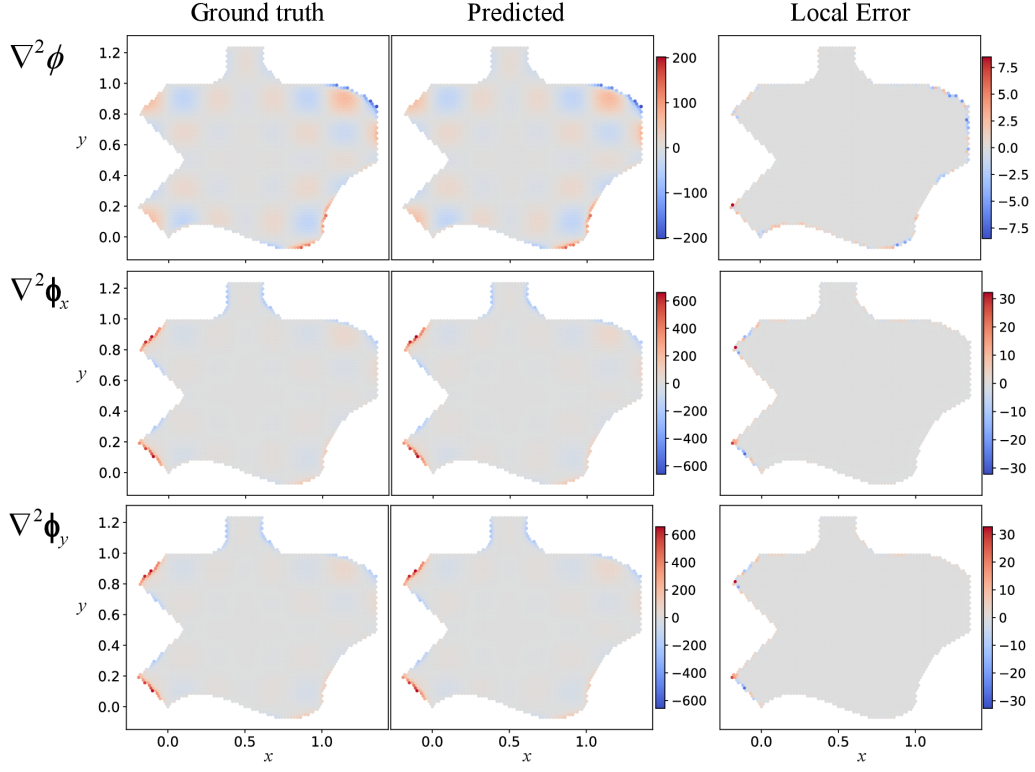


Figure 8: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for second-order derivatives ($\nabla^2\phi$, $\nabla^2\phi_x$, and $\nabla^2\phi_y$) in Case 1 ($\lambda = 2$)

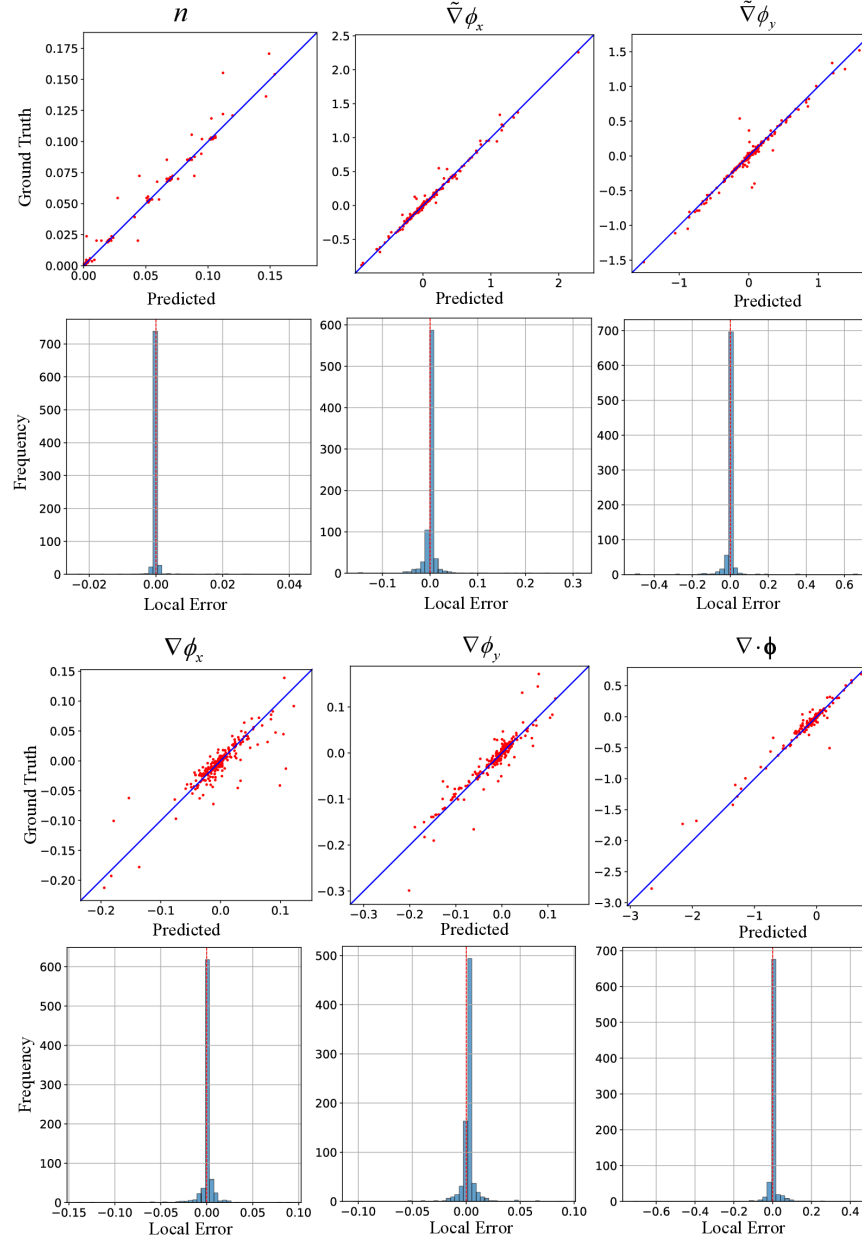


Figure 9: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for particle number density (n) and first-order derivatives ($\nabla\phi$, $\tilde{\nabla}\phi$, and $\nabla\cdot\phi$) in Case 1 ($\lambda = 2$)

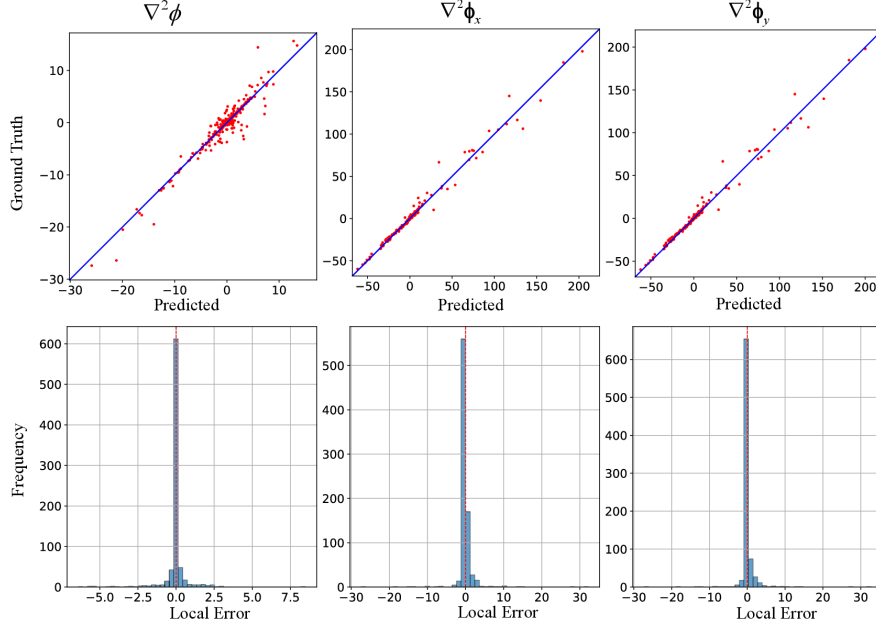


Figure 10: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for second-order derivatives ($\nabla^2\phi$, $\nabla^2\phi_x$, and $\nabla^2\phi_y$) in Case 1 ($\lambda = 2$).

3.2.2. Case 2: Unsteady pure diffusion

Figure 11 illustrates a comparison of MPS results for the only spatial derivative in this case, $\nabla^2\phi$, where the boundary contributions are provided by the ghost particle method (ground truth) and the predictions from the developed ML model. The difference between the two methods has also been provided. Overall, the ML model demonstrates excellent agreement with the ground truth at all simulation times. Aside from minor discrepancies near sharp corners, the predicted fields closely match the reference values. This highlights the model's strong generalization capability to handle cases with non-predefined, spatially variable fields and an unseen domain featuring a central hole. The parity plots and error histograms in Figure 12 further support these findings, showing a high correlation between predicted and true boundary contributions, with most errors tightly clustered around zero. Table 6 confirms that the model consistently maintains low error levels in this test, with performance metrics even better than those observed in Case 1.

Table 6: Performance metrics for Case 2.

Metrics	$\nabla^2\phi$				n
	$t = 25$	$t = 50$	$t = 75$	$t = 100$	
R	0.944	0.946	0.942	0.938	0.999
NMAE (%)	1.220	1.284	1.160	1.403	0.394
NRMSE (%)	4.894	4.673	4.212	4.600	1.345

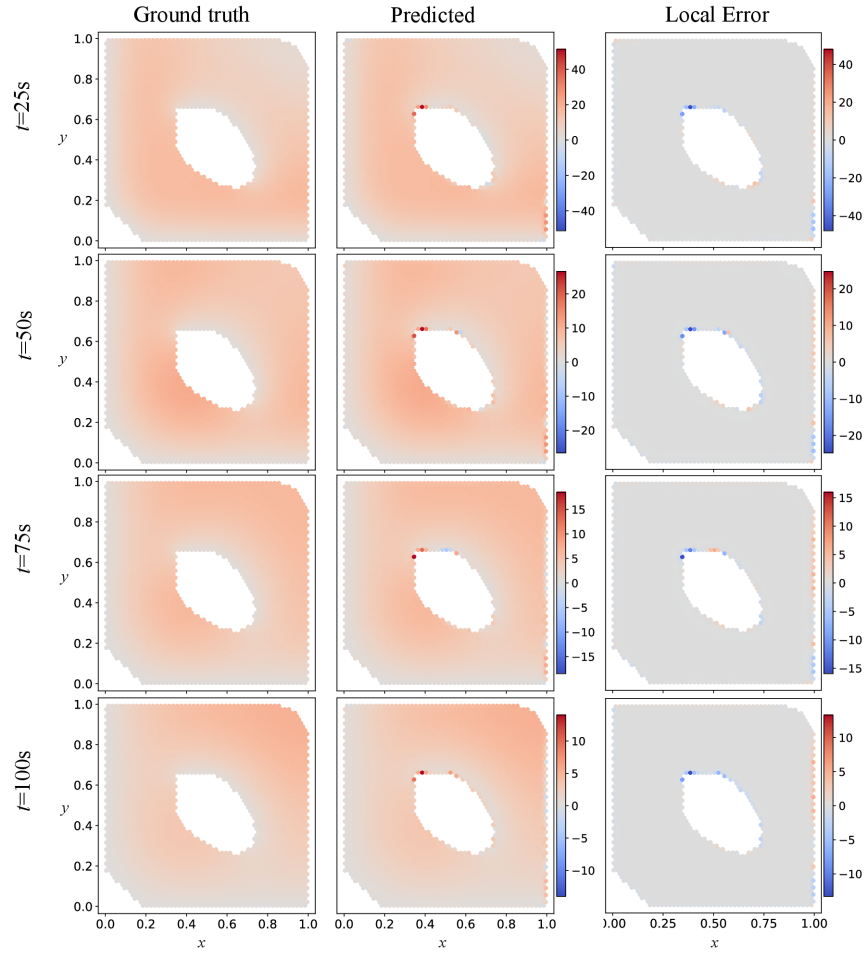


Figure 11: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for the laplacian of the scalar field ($\nabla^2\phi$) in Case 2

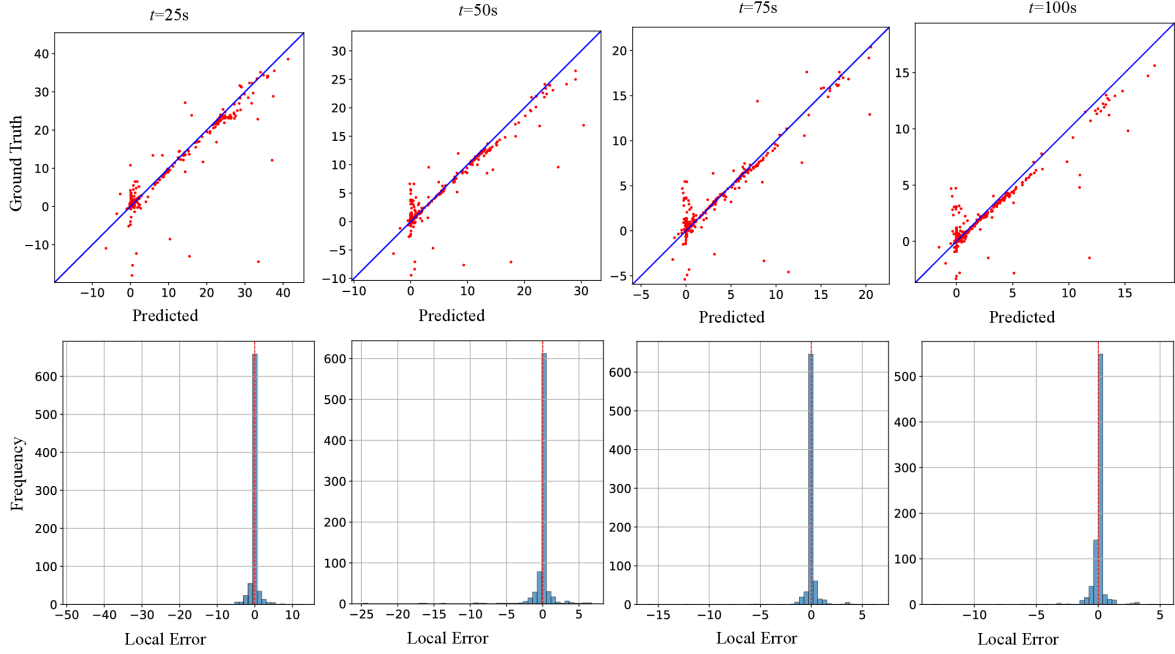


Figure 12: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for particle number density (n) and laplacian of the scalar field ($\nabla^2\phi$) in Case 2 .

3.2.3. Case 3: Unsteady Navier-Stokes with advection-diffusion

This test case introduces new generalization challenges due to its dynamic nature, characterized by spatial and temporal variability in both the fields and particle distributions, conditions not encountered during training. Figures 13, 14, 15, 16, 17 compare MPS results for different spatial derivatives in case 3, where boundary contributions are provided by the ghost particle method (ground truth) and the ML model predictions. Across all time steps, the predicted fields closely match the ground truth, with errors remaining relatively low and mostly confined to regions near the boundaries of the internal holes, particularly around sharp corners.

The error metrics in Table 7 show that the trained hybrid CNN-MLP model generalizes well to this unseen scenario. Compared with the training dataset performance (Table 4), the correlation coefficients remain high ($R > 0.94$ for all derivatives), demonstrating that the model continues to capture the underlying relationships between

boundary values and spatial derivatives. Although normalized errors increase relative to the training case, particularly for second-order terms such as $\nabla^2 T$ and $\nabla^2 \mathbf{u}$, the overall accuracy remains satisfactory, with NRMSE values typically below 4%. This suggests that the model is reliable for predicting both primary and higher-order derivative quantities, even though the latter are more sensitive to generalization errors due to their dependence on local variations.

The parity plots in Figures 18, 19, 20, 21, 22 support these observations. Predictions for first-order quantities, such as the pressure gradient, align almost perfectly with the ground truth, showing minimal scatter. In contrast, second-order quantities—especially the velocity Laplacians, display a wider spread and systematic deviations at larger magnitudes, consistent with the higher NRMSE values reported in Table 7. Nonetheless, the strong clustering of points along the diagonal demonstrates that the model generalizes effectively and maintains accuracy even for complex derivative quantities.

Table 7: Performance metrics: only n is available for $t = 0.0$ s; other variables are for $t = 0.5$ s, and $t = 1.0$ s.

	$t = 0.0$ s		$t = 0.5$ s							$t = 1.0$ s					
	n	n	$\nabla^2 T$	$\tilde{\nabla} p_x$	$\tilde{\nabla} p_y$	$\nabla \cdot \mathbf{u}$	$\nabla^2 \mathbf{u}_x$	$\nabla^2 \mathbf{u}_y$	n	$\nabla^2 T$	$\tilde{\nabla} p_x$	$\tilde{\nabla} p_y$	$\nabla \cdot \mathbf{u}$	$\nabla^2 \mathbf{u}_x$	$\nabla^2 \mathbf{u}_y$
R	1.000	0.993	0.987	0.997	0.997	0.942	0.952	0.956	0.993	0.987	0.989	0.990	0.927	0.950	0.941
NMSE(%)	0.192	2.299	0.584	0.358	0.407	0.779	0.431	0.469	1.180	0.474	0.525	0.673	1.516	0.787	0.733
NRMSE (%)	0.556	4.167	1.910	0.974	1.088	2.923	1.376	1.568	2.651	1.581	1.398	1.774	3.763	2.229	2.242

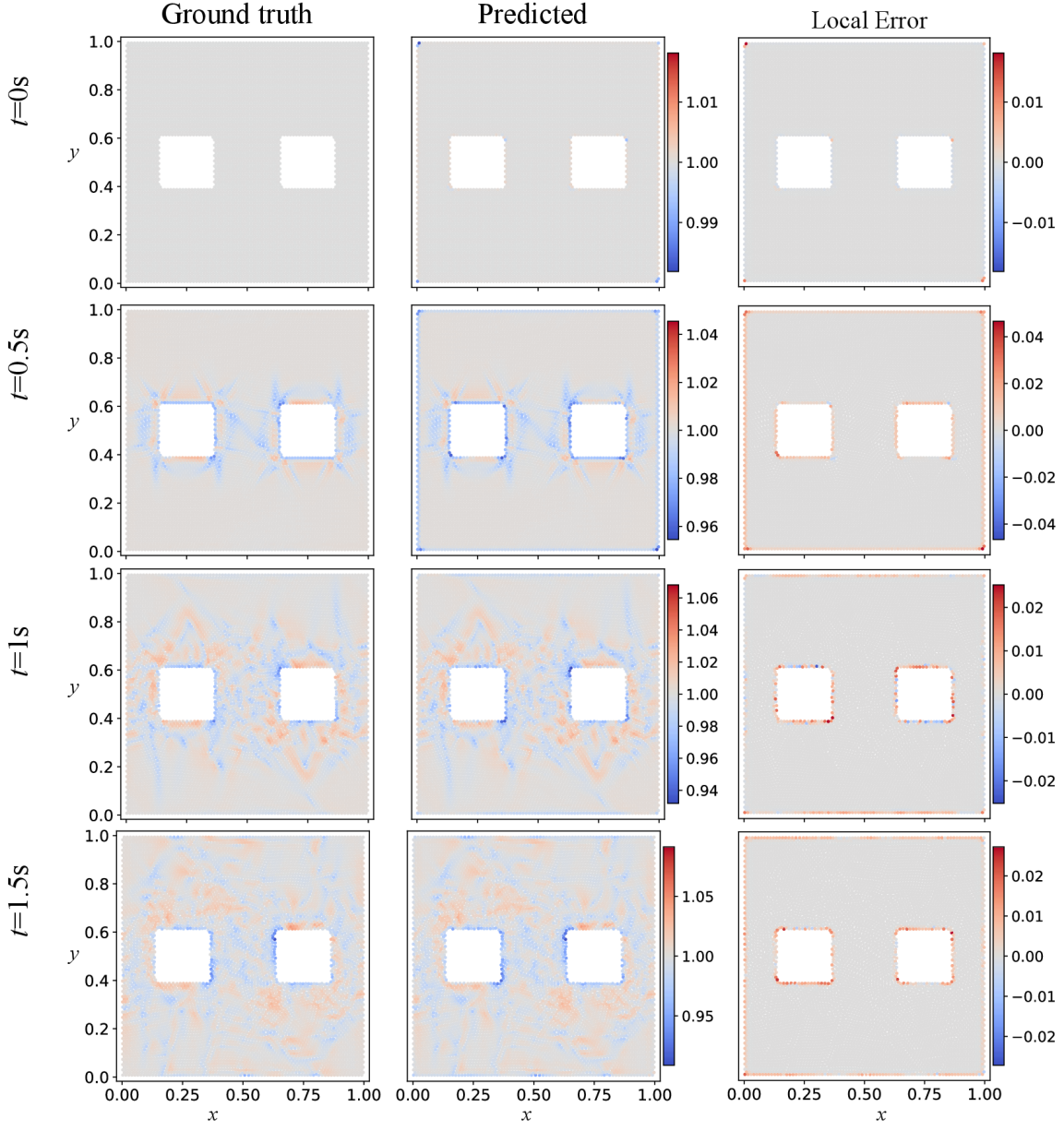


Figure 13: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for particle number density (n) in Case 3

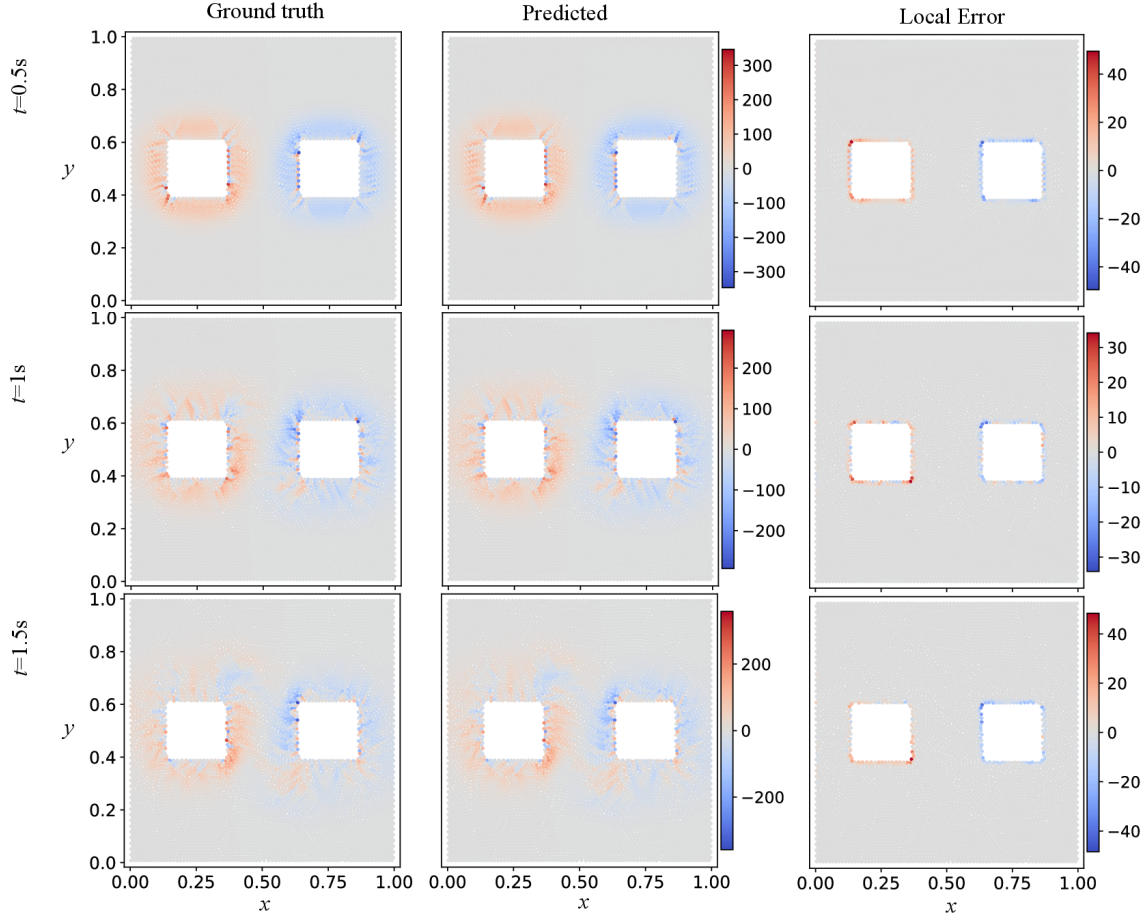


Figure 14: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for laplacian of the temperature field ($\nabla^2 T$) in Case 3

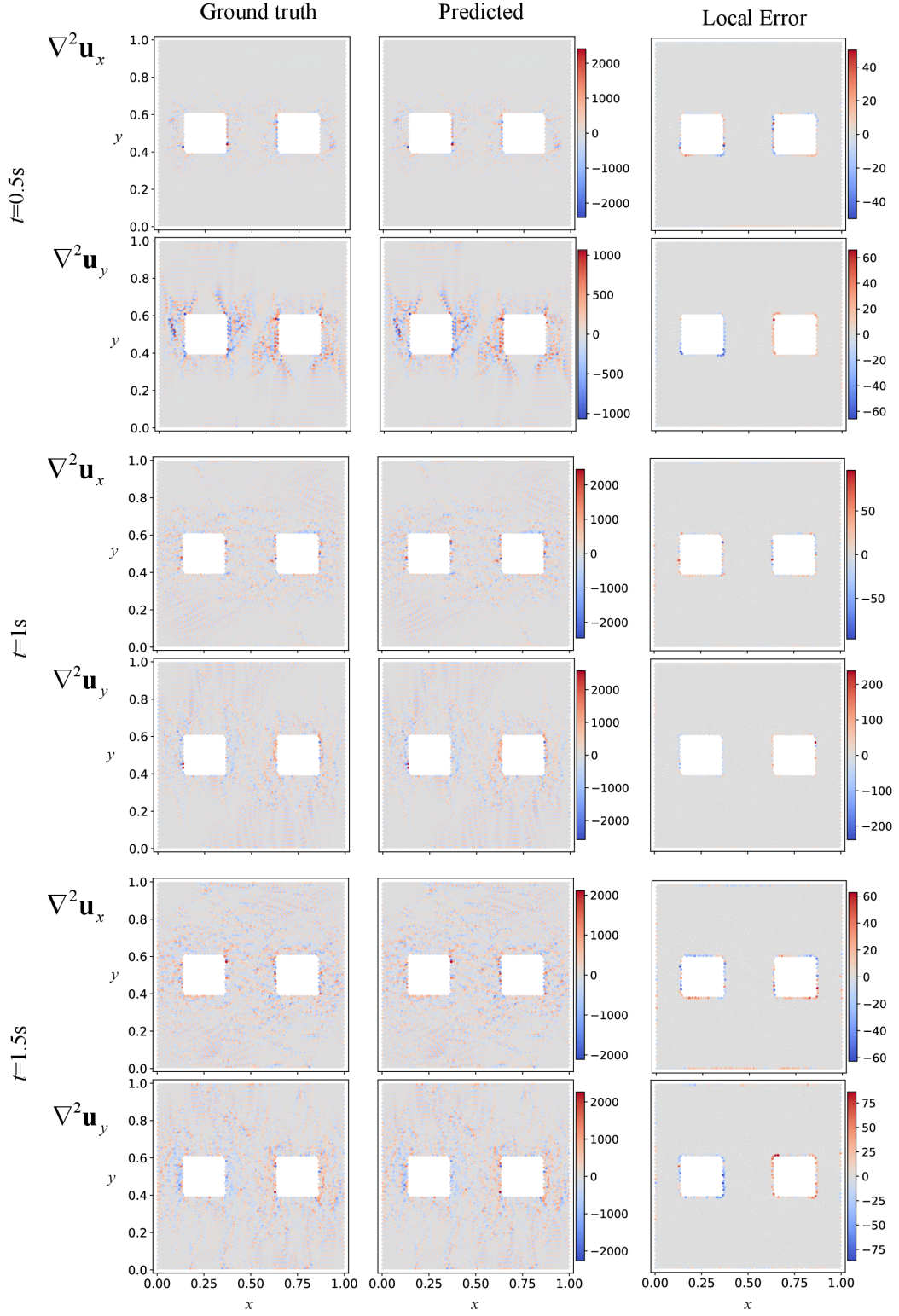


Figure 15: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for the laplacian of velocity ($\nabla^2 \mathbf{u}_x, \nabla^2 \mathbf{u}_y$) in Case 3

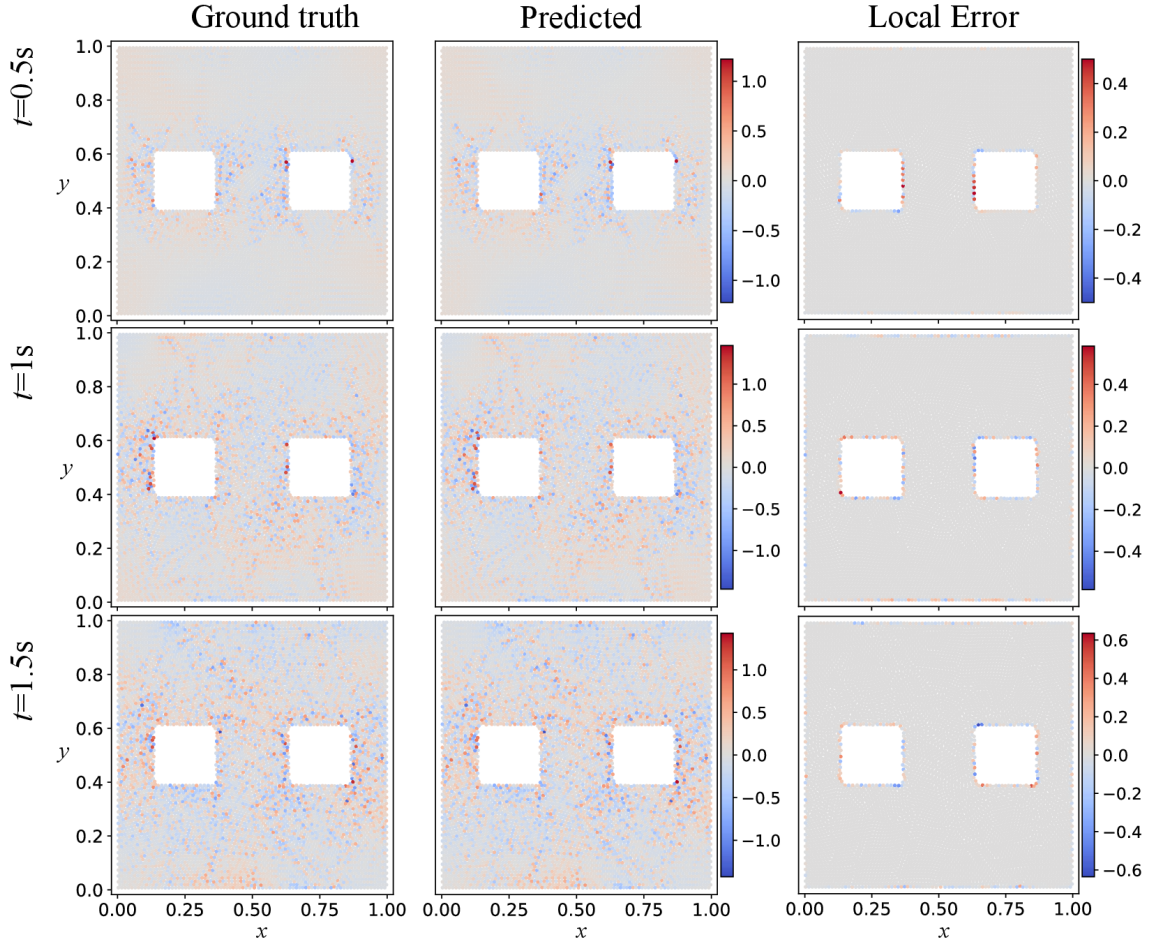


Figure 16: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for particle number density (n) and first-order derivatives ($\nabla \cdot \mathbf{u}$) in Case 3

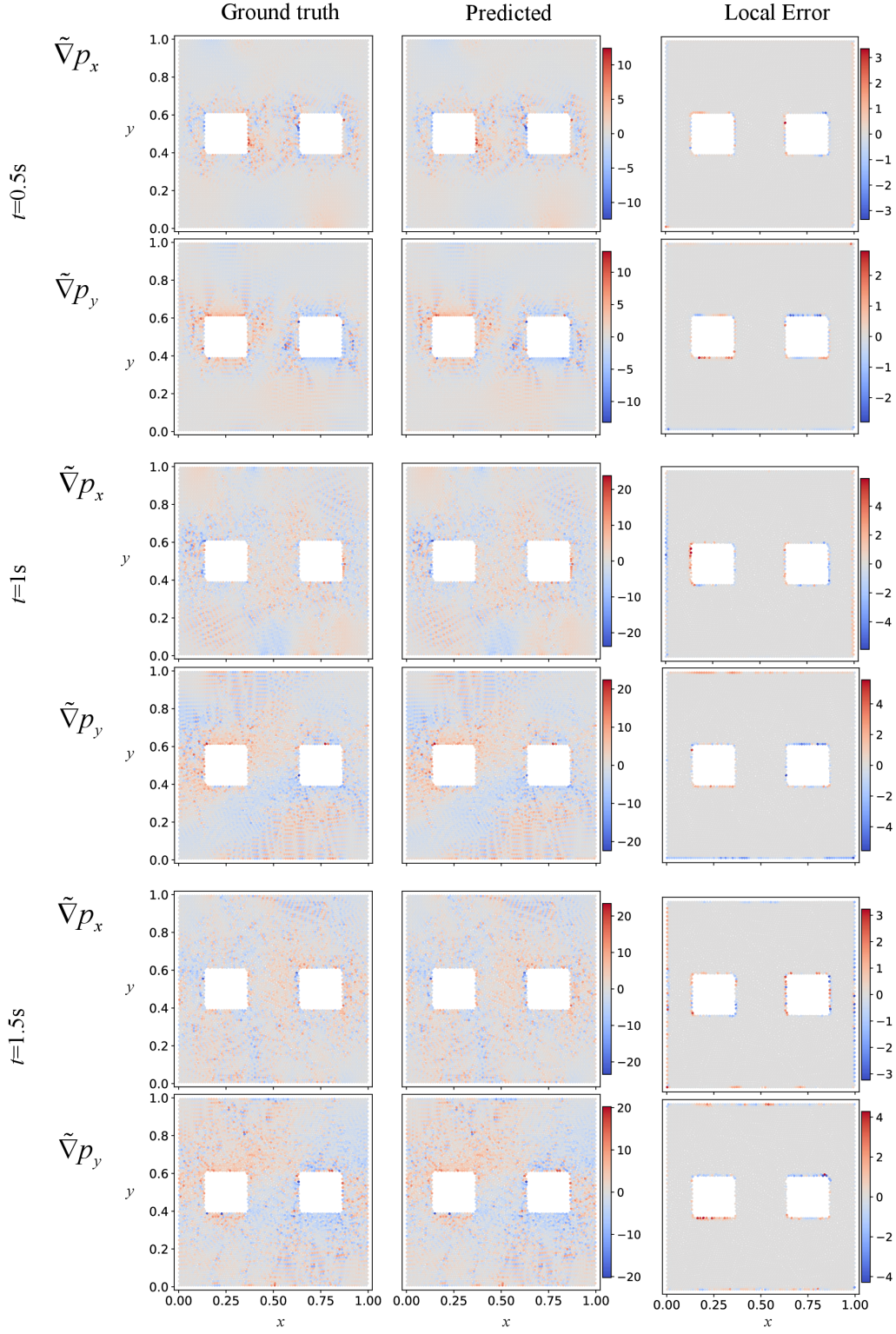


Figure 17: MPS-predicted fields with ML-based boundary treatment ($\hat{C}_i$), ground-truth MPS fields with ghost-particle boundary treatment (C_i), and local error ($C_i - \hat{C}_i$) for pressure gradient (∇p_x , ∇p_y) in Case 3

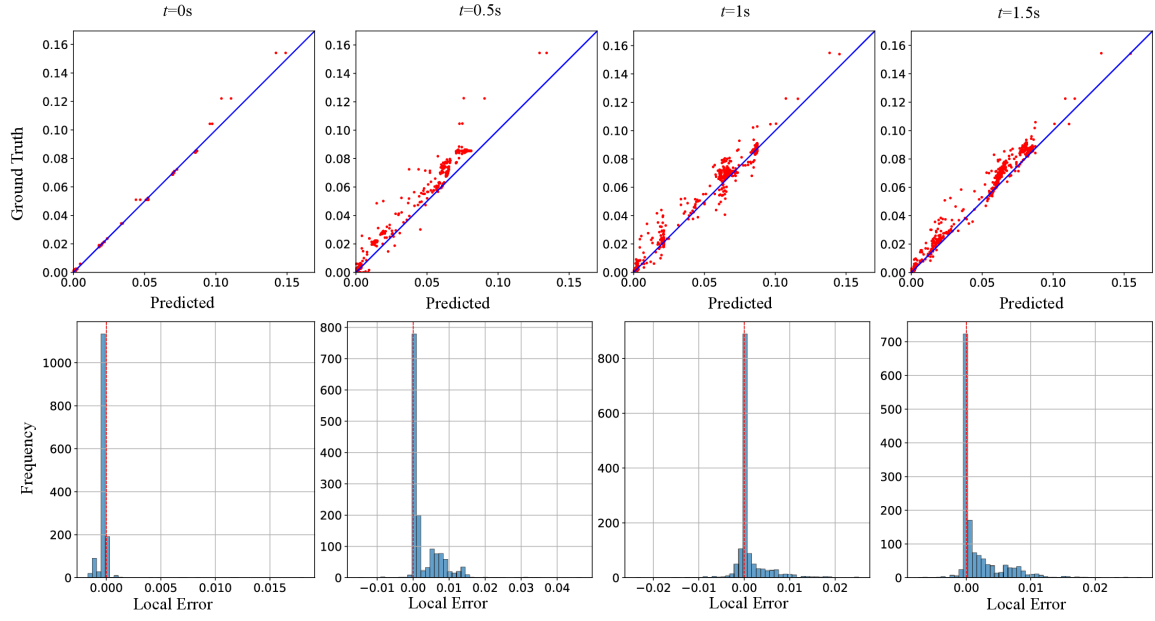


Figure 18: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for particle number density (n) in Case 3

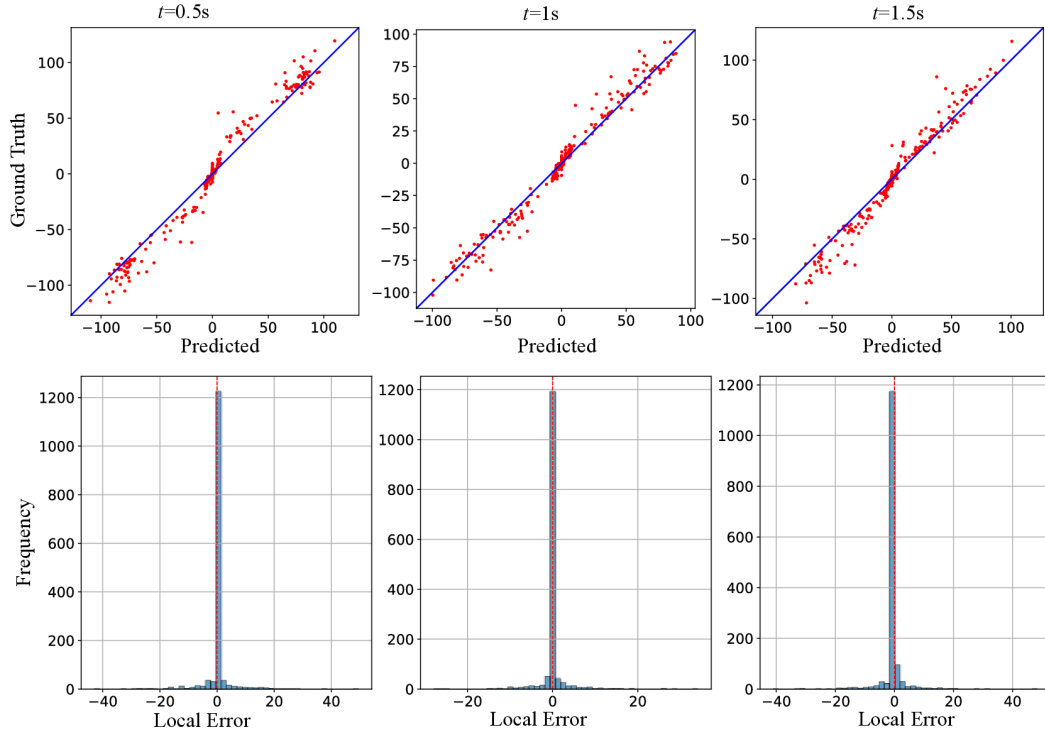


Figure 19: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for Laplacian of temperature ($\nabla^2 T$) in Case 3

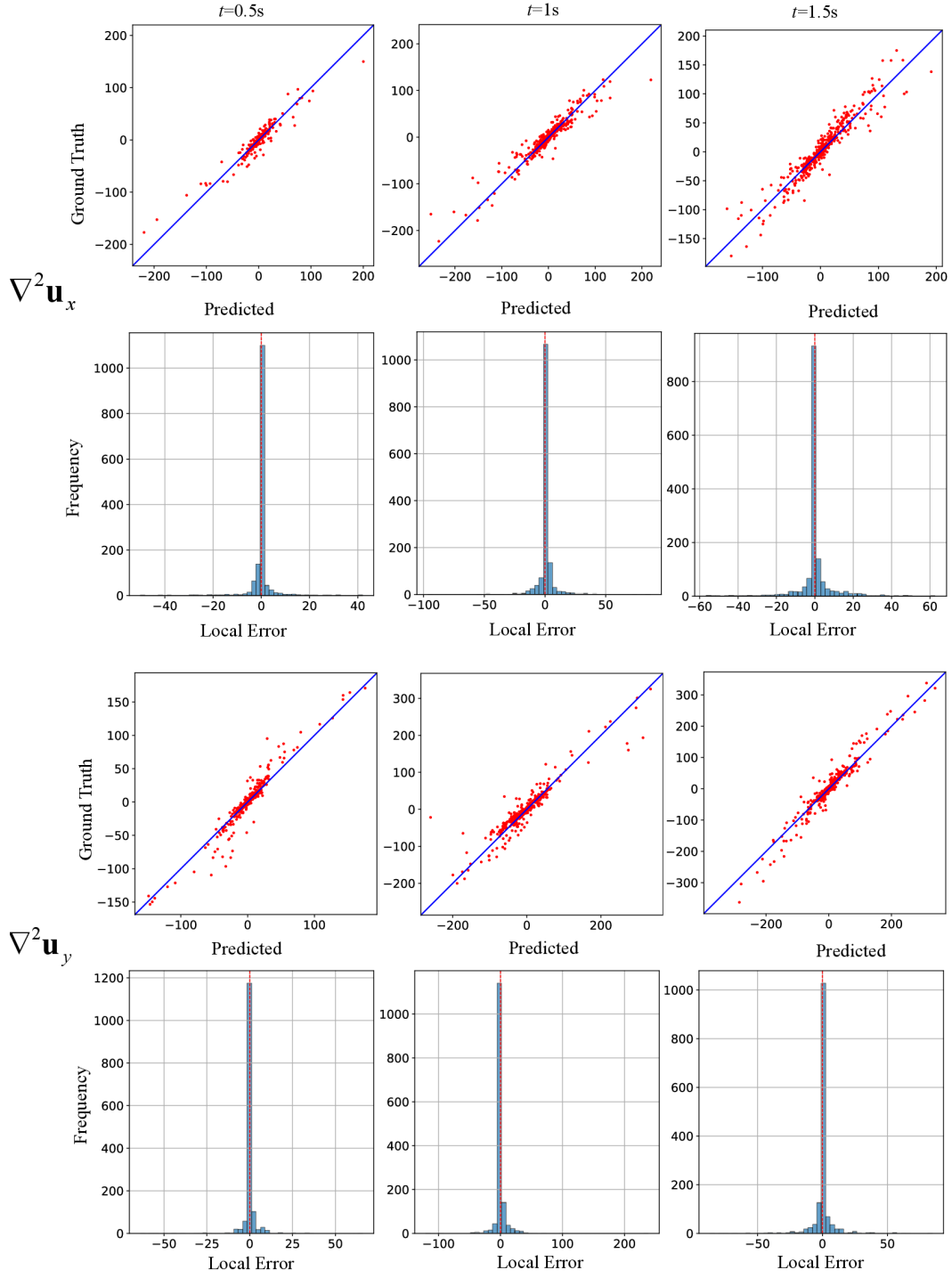


Figure 20: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for laplacian of velocity ($\nabla^2 \mathbf{u}_x$, $\nabla^2 \mathbf{u}_y$) in Case 3

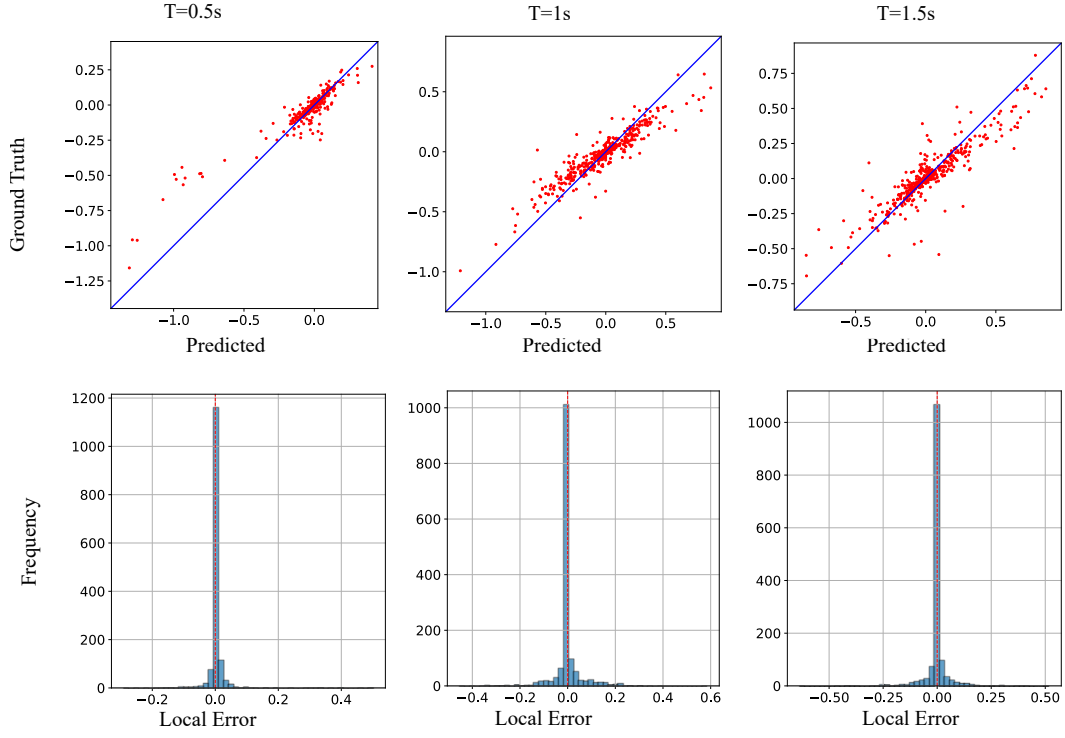


Figure 21: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for velocity divergence ($\nabla \cdot \mathbf{u}$) in Case 3

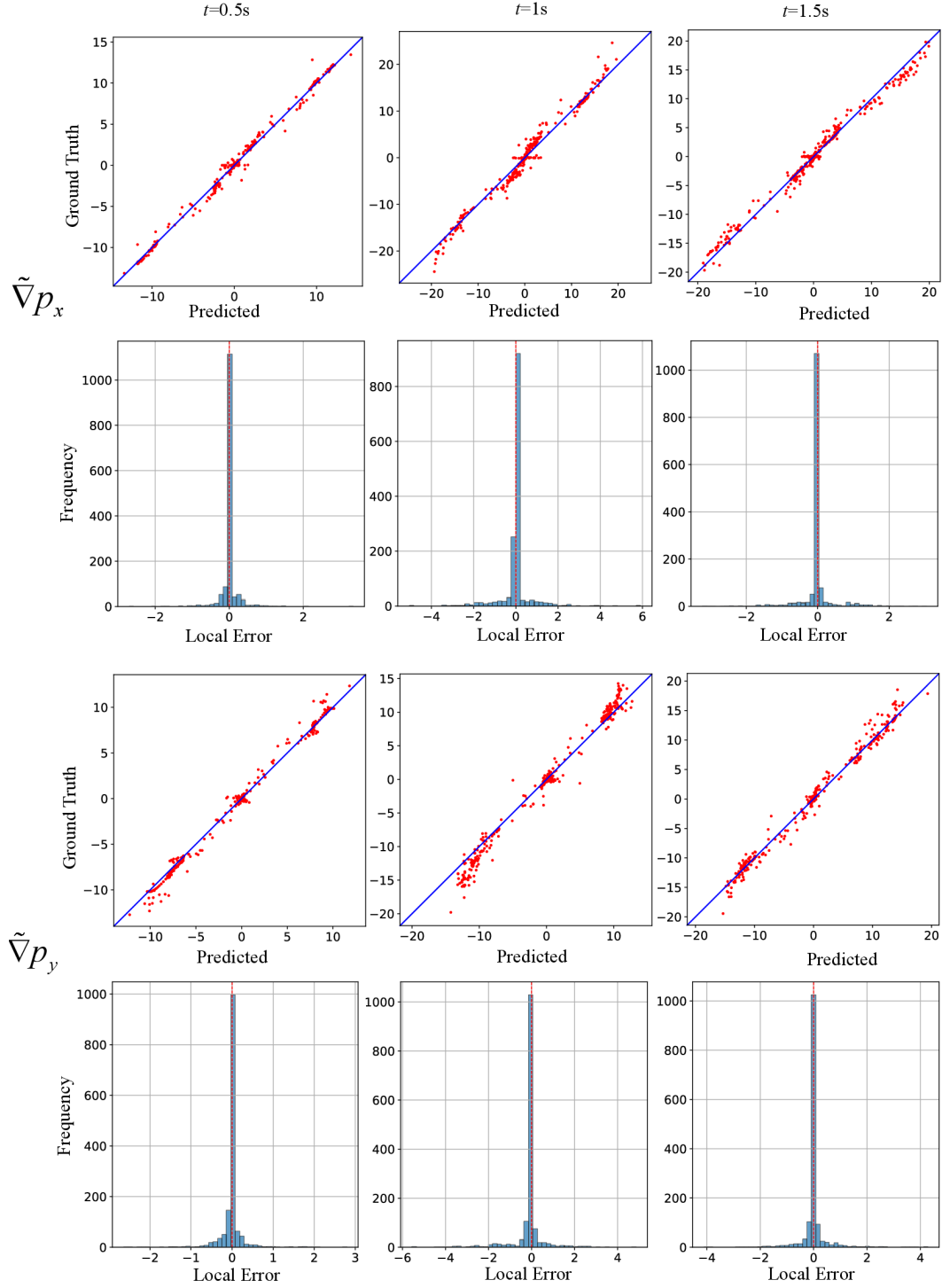


Figure 22: Parity plots (top row) and error histograms (bottom row) comparing predicted boundary contributions $\hat{B}_i$ with ground truth values B_i for pressure gradient $(\nabla p_x, \nabla p_y)$ in Case 1

4. Conclusion

We presented a physics-informed machine learning (ML) framework for computing solid boundary contributions in particle methods. The approach learns from the ghost-particle method to predict correction terms in each MPS approximation operator, thereby eliminating the need for explicit ghost particles. The architecture combines convolutional and fully connected layers to process physics-inspired features, including geometry, field variables, and kernel properties, into accurate boundary predictions. Training on datasets with diverse geometries and field conditions enabled the model to generalize across a wide range of scenarios.

The results demonstrate that the hybrid CNN–MLP models are accurate, robust, and broadly applicable for boundary treatment in particle discretizations. They consistently achieved near-perfect correlations with ghost-particle results, while avoiding overfitting and maintaining high accuracy ($R > 0.94$, $NRMSE < 4\%$) even for challenging second-order derivatives. Importantly, the models generalized well to unseen geometries, varying spatial frequencies, non-uniform particle distributions, and unsteady Navier–Stokes flows. Errors were largely confined to geometric singularities, and performance remained stable across both static and dynamic conditions, suggesting strong potential for use in other particle-based solvers without retraining.

This study focused on two-dimensional problems, where the computational cost of ML and ghost-particle methods is comparable. Future work will extend the framework to three dimensions, where eliminating ghost particles can offer substantial efficiency gains. We also plan to explore adaptive boundary representations using nodes or polygonal meshes (instead of wall particles), enabling local resolution control and further reductions in computational cost. Beyond MPS, the framework can be directly applied to SPH and related meshfree methods, where boundary treatment often dominates accuracy. Finally, incorporating a richer set of dimensionless, scale-independent features may enhance generalization to flows at very different scales.

5. CRediT authorship contribution statement

Nariman Mehranfar: Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Ahmad Shakibaeinia:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

6. 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.

7. Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used Grammarly and ChatGPT to check grammatical correctness and to improve the readability and clarity of the text. After using these tools, the authors reviewed and edited the content as necessary and take full responsibility for the final content of the published article.

8. Acknowledgments

This work was funded by the Canada Research Chair (CRC) Program. Computational infrastructure was provided through the John R. Evans Leaders Fund (JELF) of the Canada Foundation for Innovation (CFI).

Appendix A. Example simulation results of Case 3

Figure (A-1) presents snapshots of the simulated flow-field variables for Case 3, as obtained from the MPS solution.

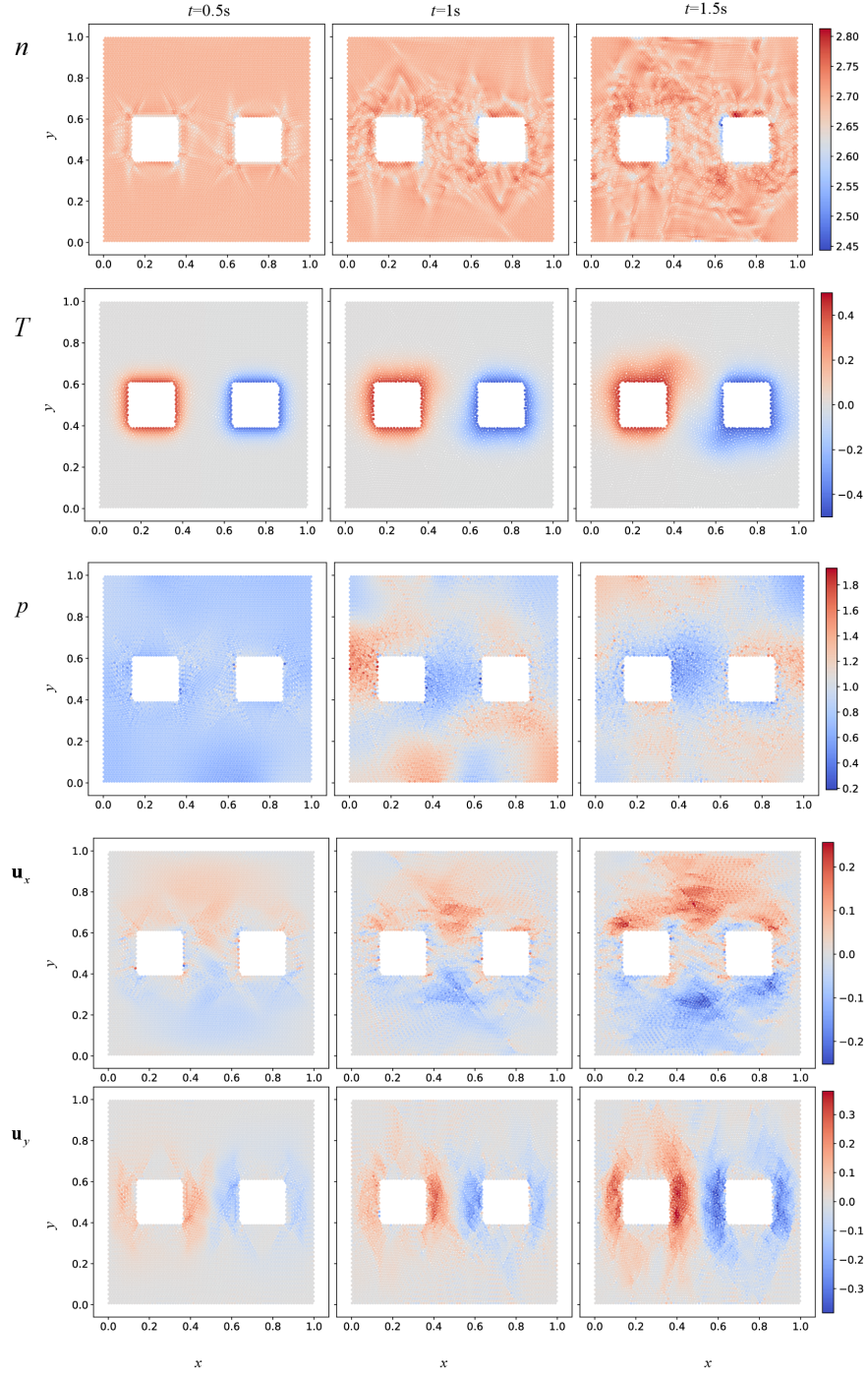


Figure (A-1): MPS-simulated particle number density (n), temperature (T), pressure (p), and velocity field $\mathbf{u} = (u_x, u_y)$ at different time steps for Case 3.

Appendix B. Cross-validation trade-off for training NN

The validation and training loss values are illustrated in Fig. (B-1).

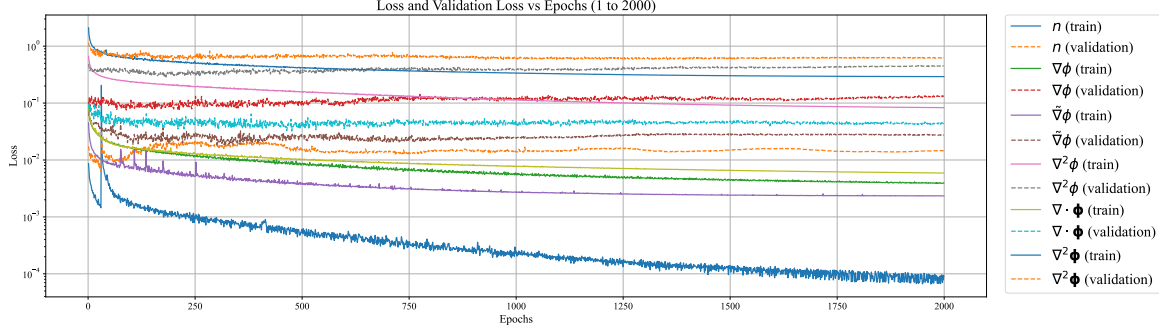


Figure (B-1): Evolution of the loss function for training and validation datasets over 2000 epochs.

References

- [1] R. A. Gingold, J. J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, *Monthly Notices of the Royal Astronomical Society* 181 (1997) 375–389. doi:10.1093/mnras/181.3.375.
- [2] L. B. Lucy, A numerical approach to the testing of the fission hypothesis, *The Astronomical Journal* 82 (1977) 1013–1024. doi:10.1086/112164.
- [3] S. Koshizuka, Y. Oka, Moving-particle semi-implicit method for fragmentation of incompressible fluid, *Nuclear Science and Engineering* 123 (1996) 421–434. doi:10.13182/NSE96-A24205.
- [4] J. L. Cercos-Pita, D. Duque, P. E. Merino-Alonso, J. Calderon-Sanchez, Boundary conditions for sph through energy conservation, *Computers & Fluids* 285 (2024) 106454. doi:https://doi.org/10.1016/j.compfluid.2024.106454.
- [5] M. Ferrand, D. R. Laurence, B. D. Rogers, D. Violeau, C. Kassiotis, Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless sph method, *International Journal for Numerical Methods in Fluids* 71 (2013) 446–472. doi:https://doi.org/10.1002/fld.3666.

- [6] T. Zhang, S. Koshizuka, P. Xuan, J. Li, C. Gong, Enhancement of stabilization of mps to arbitrary geometries with a generic wall boundary condition, *Computers & Fluids* 178 (2019) 88–112. doi:<https://doi.org/10.1016/j.compfluid.2018.09.008>.
- [7] A. C. Crespo, J. M. Dominguez, A. Barreiro, M. Gómez-Gesteira, B. D. Rogers, Gpus, a new tool of acceleration in cfd: efficiency and reliability on smoothed particle hydrodynamics methods, *PLoS One* 6 (2011) e20685. doi:[10.1371/journal.pone.0020685](https://doi.org/10.1371/journal.pone.0020685).
- [8] T. Harada, S. Koshizuka, Y. Kawaguchi, Smoothed particle hydrodynamics on gpus, *Computer Graphics International* (2007).
- [9] D. J. Price, Smoothed particle hydrodynamics: things i wish my mother taught me, *arXiv preprint arXiv:1111.1259* (2011).
- [10] M. S. Shadloo, G. Oger, D. Le Touzé, Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: Motivations, current state, and challenges, *Computers & Fluids* 136 (2016) 11–34. doi:<https://doi.org/10.1016/j.compfluid.2016.05.029>.
- [11] C. Zhang, Y.-j. Zhu, D. Wu, N. A. Adams, X. Hu, Smoothed particle hydrodynamics: Methodology development and recent achievement, *Journal of Hydrodynamics* 34 (2022) 767–805. doi:[10.1007/s42241-022-0052-1](https://doi.org/10.1007/s42241-022-0052-1).
- [12] F. Xie, W. Zhao, D. Wan, Overview of moving particle semi-implicit techniques for hydrodynamic problems in ocean engineering, *Journal of Marine Science and Application* 21 (2022) 1–22. doi:[10.1007/s11804-022-00284-9](https://doi.org/10.1007/s11804-022-00284-9).
- [13] J. Monaghan, J. Kajtar, Sph particle boundary forces for arbitrary boundaries, *Computer Physics Communications* 180 (2009) 1811–1820. doi:<https://doi.org/10.1016/j.cpc.2009.05.008>.

- [14] R. A. D. A. J. C. Crespo, M. Gómez-Gesteira, Boundary conditions generated by dynamic particles in sph methods, *Computers, Materials & Continua* 5 (2007) 173–184. doi:10.3970/cmc.2007.005.173.
- [15] D. Violeau, B. D. Rogers, Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future, *Journal of Hydraulic Research* 54 (2016) 1–26. doi:10.1080/00221686.2015.1119209.
- [16] J. Wu, G. Zhang, Z. Sun, H. Yan, B. Zhou, An improved mps method for simulating multiphase flows characterized by high-density ratios and violent deformation of interface, *Computer Methods in Applied Mechanics and Engineering* 412 (2023) 116103. doi:https://doi.org/10.1016/j.cma.2023.116103.
- [17] J. J. Monaghan, Simulating free surface flows with sph, *Journal of Computational Physics* 110 (1994) 399–406. doi:https://doi.org/10.1006/jcph.1994.1034.
- [18] J. P. Morris, P. J. Fox, Y. Zhu, Modeling low reynolds number incompressible flows using sph, *Journal of Computational Physics* 136 (1997) 214–226. doi:https://doi.org/10.1006/jcph.1997.5776.
- [19] A. Shakibaeinia, Y.-C. Jin, A weakly compressible mps method for modeling of open-boundary free-surface flow, *International Journal for Numerical Methods in Fluids* 63 (2010) 1208–1232. doi:https://doi.org/10.1002/fld.2132.
- [20] L. Vela Vela, J. M. Reynolds-Barredo, R. Sánchez, A positioning algorithm for sph ghost particles in smoothly curved geometries, *Journal of Computational and Applied Mathematics* 353 (2019) 140–153. doi:https://doi.org/10.1016/j.cam.2018.12.021.
- [21] T. Harada, S. Koshizuka, K. Shimazaki, Improvement of wall boundary calculation model for mps method, *Transactions of the Japan Society for Computational Engineering and Science* 2008 (2008) 20080006–20080006. doi:10.11421/jscs.2008.20080006.

- [22] S. Adami, X. Hu, N. Adams, A generalized wall boundary condition for smoothed particle hydrodynamics, *Journal of Computational Physics* 231 (2012) 7057–7075. doi:<https://doi.org/10.1016/j.jcp.2012.05.005>.
- [23] B.-H. Lee, J.-C. Park, M.-H. Kim, S.-C. Hwang, Step-by-step improvement of mps method in simulating violent free-surface motions and impact-loads, *Computer Methods in Applied Mechanics and Engineering* 200 (2011) 1113–1125. doi:<https://doi.org/10.1016/j.cma.2010.12.001>.
- [24] G. Duan, A. Yamaji, M. Sakai, An incompressible–compressible lagrangian particle method for bubble flows with a sharp density jump and boiling phase change, *Computer Methods in Applied Mechanics and Engineering* 372 (2020) 113425. doi:<https://doi.org/10.1016/j.cma.2020.113425>.
- [25] A. Colagrossi, M. Landrini, Numerical simulation of interfacial flows by smoothed particle hydrodynamics, *Journal of computational physics* 191 (2003) 448–475.
- [26] S. Park, G. Jeun, Coupling of rigid body dynamics and moving particle semi-implicit method for simulating isothermal multi-phase fluid interactions, *Computer Methods in Applied Mechanics and Engineering* 200 (2011) 130–140. doi:<https://doi.org/10.1016/j.cma.2010.08.001>.
- [27] H. Akimoto, Numerical simulation of the flow around a planing body by mps method, *J Ocean engineering* 64 (2013) 72–79.
- [28] A. English, J. Domínguez, R. Vacondio, A. Crespo, P. Stansby, S. Lind, L. Chiapponi, M. Gesteira, Modified dynamic boundary conditions (mdbc) for general-purpose smoothed particle hydrodynamics (sph): application to tank sloshing, dam break and fish pass problems, *Computational Particle Mechanics* 9 (2021). doi:[10.1007/s40571-021-00403-3](https://doi.org/10.1007/s40571-021-00403-3).
- [29] A. Mayrhofer, B. D. Rogers, D. Violeau, M. Ferrand, Investigation of wall bounded flows using sph and the unified semi-analytical wall bound-

- ary conditions, *Computer Physics Communications* 184 (2013) 2515–2527. doi:<https://doi.org/10.1016/j.cpc.2013.07.004>.
- [30] W. Kostorz, A. Esmail-Yakas, A semi-analytical boundary integral method for radial functions with application to smoothed particle hydrodynamics, *Journal of Computational Physics* 417 (2020) 109565. doi:<https://doi.org/10.1016/j.jcp.2020.109565>.
- [31] N. Mitsume, S. Yoshimura, K. Murotani, T. Yamada, Explicitly represented polygon wall boundary model for the explicit mps method, *Computational Particle Mechanics* 2 (2015) 73–89. doi:10.1007/s40571-015-0037-8.
- [32] M. He, X. Gao, W. Xu, B. Ren, H. Wang, Potential application of submerged horizontal plate as a wave energy breakwater: A 2d study using the wcsph method, *Ocean Engineering* 185 (2019) 27–46. doi:<https://doi.org/10.1016/j.oceaneng.2019.05.034>.
- [33] M. He, W. Xu, X. Gao, B. Ren, The layout of submerged horizontal plate breakwater (shpb) with respect to the tidal-level variation, *Coastal Engineering Journal* 60 (2018) 280–298. doi:10.1080/21664250.2018.1514758.
- [34] R. Amaro Junior, L.-Y. Cheng, P. Osello, An improvement of rigid bodies contact for particle-based non-smooth walls modeling, *Computational Particle Mechanics* (2019). doi:10.1007/s40571-019-00233-4.
- [35] T. Zhang, S. Koshizuka, K. Murotani, K. Shibata, E. Ishii, Improvement of pressure distribution to arbitrary geometry with boundary condition represented by polygons in particle method, *International Journal for Numerical Methods in Engineering* 112 (2017) 685–710. doi:<https://doi.org/10.1002/nme.5520>.
- [36] Y.-x. Zhang, D.-c. Wan, T. Hino, Comparative study of mps method and level-set method for sloshing flows, *Journal of Hydrodynamics, Ser. B* 26 (2014) 577–585. doi:[https://doi.org/10.1016/S1001-6058\(14\)60065-2](https://doi.org/10.1016/S1001-6058(14)60065-2).

- [37] T. Zhang, S. Koshizuka, K. Murotani, K. Shibata, E. Ishii, M. Ishikawa, Improvement of boundary conditions for non-planar boundaries represented by polygons with an initial particle arrangement technique, *Int. J. Comput. Fluid Dyn.* 30 (2016). doi:10.1080/10618562.2016.1167194.
- [38] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, *Annual Review of Fluid Mechanics* 52 (2020) 477–508. doi:10.1146/annurev-fluid-010719-060214.
- [39] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, *Annual Review of Fluid Mechanics* 51 (2019) 357–377. doi:10.1146/annurev-fluid-010518-040547.
- [40] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.
- [41] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (2021) 422–440. doi:10.1038/s42254-021-00314-5.
- [42] M. P. Brenner, J. D. Eldredge, J. B. Freund, Perspective on machine learning for advancing fluid mechanics, *Phys. Rev. Fluids* 4 (2019) 100501. doi:10.1103/PhysRevFluids.4.100501.
- [43] B. Mahesh, Machine Learning Algorithms -A Review, 2019. doi:10.21275/ART20203995.
- [44] A. L. Samuel, Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development* 3 (1959) 210–229. doi:10.1147/rd.33.0210.

- [45] J. Alzubi, A. Nayyar, A. Kumar, Machine learning from theory to algorithms: an overview, in: *Journal of physics: conference series*, volume 1142, IOP Publishing, 2018, p. 012012.
- [46] R. T. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, *J Advances in neural information processing systems* 31 (2018).
- [47] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A survey, *arXiv preprint arXiv:2003.04919* 1 (2020) 1–34.
- [48] L. Ladický, S. Jeong, B. Solenthaler, M. Pollefeys, M. Gross, Data-driven fluid simulations using regression forests, *ACM Transactions on Graphics* 34 (2015) 1–9. doi:10.1145/2816795.2818129.
- [49] E. P. Marinho, Covariance-based smoothed particle hydrodynamics. a machine-learning application to simulating disc fragmentation, 2021. URL: <https://arxiv.org/abs/2106.08870>. arXiv:2106.08870.
- [50] J. Bai, Y. Zhou, C. M. Rathnayaka, H. Zhan, E. Sauret, Y. Gu, A data-driven smoothed particle hydrodynamics method for fluids, *Engineering Analysis with Boundary Elements* 132 (2021) 12–32. doi:10.1016/j.enganabound.2021.06.029.
- [51] L. Xiaoxing, K. Morita, Z. Shuai, Machine-learning-based surface tension model for multiphase flow simulation using particle method, *International Journal for Numerical Methods in Fluids* 93 (2021) 356–68. doi:10.1002/fld.4886.
- [52] H. Wessels, C. Weißenfels, P. Wriggers, The neural particle method – an updated lagrangian physics informed neural network for computational fluid dynamics, *Computer Methods in Applied Mechanics and Engineering* 368 (2020) 113127–113127. doi:10.1016/j.cma.2020.113127.

- [53] M. Woodward, Y. Tian, C. Hyett, C. Fryer, D. Livescu, M. Stepanov, M. Chertkov, Physics informed machine learning of sph: Machine learning lagrangian turbulence 12 (2021).
- [54] A. Alexiadis, A minimalistic approach to physics-informed machine learning using neighbour lists as physics-optimized convolutions for inverse problems involving particle systems, *Journal of Computational Physics* 473 (2023) 111750. doi:<https://doi.org/10.1016/j.jcp.2022.111750>.
- [55] Y. Tian, M. Woodward, M. Stepanov, C. Fryer, C. Hyett, D. Livescu, M. Chertkov, Lagrangian large eddy simulations via physics informed machine learning (2022) 1–31.
- [56] N. Zhang, S. Yan, Q. Ma, X. Guo, Z. Xie, X. Zheng, A cnn-supported lagrangian isph model for free surface flow, *Applied Ocean Research* 136 (2023) 103587. doi:<https://doi.org/10.1016/j.apor.2023.103587>.
- [57] A. Shakibaeinia, Y.-C. Jin, Mps mesh-free particle method for multiphase flows, *Computer Methods in Applied Mechanics and Engineering* 229-232 (2012) 13–26. doi:<https://doi.org/10.1016/j.cma.2012.03.013>.
- [58] A. Khayyer, H. Gotoh, Enhancement of stability and accuracy of the moving particle semi-implicit method, *J. Comput. Phys.* 230 (2011) 3093–3118.
- [59] T. Tamai, S. Koshizuka, Least squares moving particle semi-implicit method: An arbitrary high order accurate meshfree lagrangian approach for incompressible flow with free surfaces, *Comput. Part. Mech.* 1 (2014) 277–305.
- [60] M. Jandaghian, A. Shakibaeinia, An enhanced weakly-compressible mps method for free-surface flows, *Computer Methods in Applied Mechanics and Engineering* 360 (2020) 112771. doi:<https://doi.org/10.1016/j.cma.2019.112771>.

- [61] J. Wu, B. Yang, Z. Sun, G. Zhang, A. Shakibaeinia, Research advances in moving particle semi-implicit method and applications in ocean engineering, *Physics of Fluids* 37 (2025).
- [62] F. Garoosi, A. Shakibaeinia, An improved high-order isph method for simulation of free-surface flows and convection heat transfer, *Powder Technology* 376 (2020) 668–696. doi:<https://doi.org/10.1016/j.powtec.2020.08.074>.
- [63] M. Jandaghian, H. M. Siaben, A. Shakibaeinia, Stability and accuracy of the weakly compressible sph with particle regularization techniques, *European Journal of Mechanics - B/Fluids* 94 (2022) 314–333. doi:<https://doi.org/10.1016/j.euromechflu.2022.03.007>.