



Reconstruction of 3D flow field around a building model in wind tunnel: a novel physics-informed neural network framework adopting dynamic prioritization self-adaptive loss balance strategy

En-Ze Rui, Zheng-Wei Chen, Yi-Qing Ni, Lei Yuan & Guang-Zhi Zeng

To cite this article: En-Ze Rui, Zheng-Wei Chen, Yi-Qing Ni, Lei Yuan & Guang-Zhi Zeng (2023) Reconstruction of 3D flow field around a building model in wind tunnel: a novel physics-informed neural network framework adopting dynamic prioritization self-adaptive loss balance strategy, *Engineering Applications of Computational Fluid Mechanics*, 17:1, 2238849, DOI: [10.1080/19942060.2023.2238849](https://doi.org/10.1080/19942060.2023.2238849)

To link to this article: <https://doi.org/10.1080/19942060.2023.2238849>



© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 25 Jul 2023.



[Submit your article to this journal](#)



Article views: 1983



[View related articles](#)



[View Crossmark data](#)



Citing articles: 1 [View citing articles](#)

Reconstruction of 3D flow field around a building model in wind tunnel: a novel physics-informed neural network framework adopting dynamic prioritization self-adaptive loss balance strategy

En-Ze Rui^{a,b}, Zheng-Wei Chen^{a,b}, Yi-Qing Ni^{a,b}, Lei Yuan^{a,b} and Guang-Zhi Zeng^{a,b}

^aDepartment of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong, People's Republic of China; ^bNational Rail Transit Electrification and Automation Engineering Technology Research Center (Hong Kong Branch), Hong Kong, People's Republic of China

ABSTRACT

Physics-informed neural networks (PINNs) have recently emerged and attracted extensive attention as an alternative approach to computational fluid dynamics (CFD) methods, which can provide competitive solutions to a variety of forward and inverse fluid problems. In this study, we reconstruct a 3D wind field around a building model in wind tunnel test with a Reynolds number of 2.4×10^4 by formulating a novel PINN framework, which is the first exploration of PINNs for building wind engineering problems. To surmount the hurdle in multi-objective optimization for PINN training, a dynamic prioritization (dp) self-adaptive loss balance strategy is proposed (termed dpPINN), which adaptively reconciles the loss terms of distinct scales to facilitate convergence in PINN training. A zero-equation turbulence model and the wind velocity data collected in near-wall regions are embedded in dpPINN training. Comparison results indicate that dpPINN predictions show good consistency with observation data, which is superior to two current PINN paradigms in prediction accuracy. Furthermore, the influence of neural network configurations, turbulence models, and the layout arrangements of training points on the dpPINN prediction is investigated. It is demonstrated that the dpPINN could be a powerful auxiliary means for airflow simulation and reconstruction in wind engineering applications.

ARTICLE HISTORY

Received 18 January 2023
Accepted 14 July 2023


KEYWORDS

Buildings; Flow fields; Reconstruction; Physics-informed neural network (PINN); Wind tunnel test

1. Introduction

The acquaintance of outdoor airflow characteristics is essential for building simulations since these characteristics directly influence the thermal environment, air pollutant diffusion, and other relevant effects surrounding a building. The outdoor airflow characteristics of the flow field around a building can be comprehended through solution to a system of equations that is composed of Navier-Stokes (NS) equations, boundary conditions, and initial conditions. Over the past decades, computational fluid dynamics (CFD) has been the mainstream technique for solving NS equations and simulating flow characteristics on account of its versatility and easy implementation (Guo et al., 2020; Liu et al., 2020; Chen et al., 2021; He et al., 2021; Chen et al., 2022; Song et al., 2022; Zhang et al., 2022). Additionally, the successive accumulation of field-measured flow data has offered a good opportunity to conduct data-based analysis on flow field reconstruction, especially

with the advent of machine learning algorithms in recent years (Milano & Koumoutsakos, 2002; Bhatnagar et al., 2019; Chen et al., 2019; Pache & Rung, 2022). However, both approaches have their own shortcomings. The accuracy of a physics-based simulation, i.e. a CFD simulation, is always restricted by various factors such as grid resolution and model applicability (Kataoka et al., 2020). The identification of specific parameters in the physical model is also a difficult issue. For data-based methods, the demand for massive data hinders their wide application. In most cases, the amount of labelled data that can be obtained from experiments or field measurements is far from enough to fully train a machine learning model and further reconstruct an entire flow field. Under these circumstances, physics-aided data-driven procedures which leverage both physics information and measurement data are highly desired for solving NS equations and simulating the flow characteristics around buildings.

CONTACT Yi-Qing Ni  ceyqni@polyu.edu.hk  Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR, People's Republic of China;  National Rail Transit Electrification and Automation Engineering Technology Research Center (Hong Kong Branch), Hung Hom, Kowloon, Hong Kong SAR, People's Republic of China

© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

The newly emerging physics-informed neural network (PINN) (Raissi et al., 2019) has received extensive attention in diverse research fields in recent years (Sahli Costabal et al., 2020; Cai et al., 2022). By embedding the residuals of physical governing equations, initial conditions, and boundary conditions into the total loss function of a neural network, PINN becomes a reliable physics-informed tool for solving various types of partial differential equations (PDEs) such as Schrodinger equations and linear Poisson problems (Meng et al., 2020; Sun et al., 2021; Yuan et al., 2022). More specifically, the sum of the residuals of these physical constraints tends to converge toward zero during the PINN training process with the aid of gradient descent algorithms such as the stochastic gradient descent (SGD) and adaptive moment estimation (Adam) algorithms, which help to achieve the target of providing approximate solutions to PDEs. Investigations on the application of PINN to fluid mechanics have been conducted in recent years, on the ground that the core of fluid mechanics is to solve second-order nonlinear PDEs, i.e. NS equations (Wessels et al., 2020; Eivazi & Vinuesa, 2022). As an alternative solver for NS equations, PINN promises several advantages. First, it is a meshless method that does not suffer from grid resolution problems. Second, the versatility of the physical model can be greatly extended with a data-driven approach. Like other neural networks, PINN can embed labelled data (on-site information) into the model training process (Riel et al., 2021; Choi et al., 2022). The residuals between the PINN prediction and the labelled data, along with the residuals of NS equations at collocation points, are calculated at each iteration and embedded in the total loss. In this way, the final solution to NS equations by the PINN method can be thought of as a comprehensive result that combines both physical laws and on-site information. Nevertheless, research on PINN in the field of fluid mechanics is still in its infancy. A pioneering work on laminar flows was conducted by Raissi et al. (2019) where they addressed an inverse problem on identifying the unknown parameters in NS equations by PINN. Rao et al. (2020) solved a forward fluid problem that simulates a laminar cylinder flow in a mixed-variable scheme without embedding any training data. As for turbulent flows, Reynolds-Averaged Navier-Stokes (RANS) equations are the commonly used physical governing equations that have been embedded into the PINN framework. Luo et al. (2020) formulated a PINN model for fluid parameter identification that precisely identifies the five parameters in RANS equations based on direct numerical simulation (DNS) data. Eivazi et al. (2022) simulated a variety of turbulent flows by incorporating flow characteristics on the domain boundaries into a

PINN which also uses RANS equations as the governing functions.

Premised on the above investigations, this study aims to take a further step in the field of PINN-aided fluid mechanics to explore building airflow simulation. An attempt is made to reconstruct the entire 3D flow field around a scale building model in wind tunnel by using only a small amount of wind characteristic data collected near the building model and near the ground (no-slip wall boundary) within the PINN framework. To this end, a PINN model is formulated to simulate the flow field in wind tunnel test using sparse experimental wind velocity data in the near-wall regions for supervised learning. To the best of the authors' knowledge, this is the first study on PINN applications for 3D building airflow simulation with a Reynolds number of higher than 2.4×10^4 , and it manages to embed experimental data instead of simulation data in configuring PINN. Keeping in mind of limited accessibility in real practice, only the wind characteristics collected in the near-wall regions of wind tunnel are embedded in PINN, in conjunction with the constraints of physical equations, to guide the training of a deep neural network. The formulated PINN model is validated through comparing its predictions with experimental results obtained from the wind tunnel test, and then is applied to reconstruct the entire flow field of wind tunnel including the zones without measurement data and far away from the near-wall regions.

In addition, a novel strategy is proposed in this study to tackle the loss balance problem stemming from multiple-objective optimization in PINN training, mainly due to distinctly different scales of the flow components in three directions. Similar to the circumstance in multi-task learning, the total loss function in PINN training for the present problem involves a number of individual components, which include three components regarding the three measurement horizons and those associated with the governing equations, boundary conditions and initial conditions. In principle, faster computational rate and more accurate solution can be achieved if the values of these loss components can be balanced at each training iteration instead of keeping them at a fixed ratio. Various self-adaptive loss balance strategies can be used for this purpose. For instance, the task relatedness has been well formalized in multi-task learning (Kendall et al., 2018), and this formalism has recently been incorporated into PINN training (Xiang et al., 2022). In the present study, a dynamic prioritization (dp) loss balance strategy is proposed to adaptively adjust the scale of each loss component (e.g. the wind velocity component in each direction) in accordance with its relative error at each training iteration. We also reconfigure the

total loss function to facilitate the realization of the proposed strategy, which leads to a new PINN paradigm referred to as dpPINN. The performance of dpPINN will be evaluated through comparison with other loss balance strategies.

The rest of the paper is organized as follows. Section 2 provides a briefing to the wind tunnel test and the computational domain considered for PINN formulation. In Section 3, we explore a zero-equation RANS turbulence model that will be encoded into PINN and the formation of the dpPINN paradigm. In Section 4, we discuss the validation results, the performance of dpPINN, and the influence of various factors on dpPINN prediction results. Finally, conclusions are drawn in Section 5.

2. Statements of the problem

2.1. Brief description of the wind tunnel test

The data from wind tunnel test conducted by the Shimizu Corporation Institute of Technology (Meng & Hibi, 1998) are used in this study. A scale model of building was positioned in the center location of the wind tunnel, which is 0.08 m in length and width and 0.16 m in height. During the wind tunnel test, the maximum wind speed at the inlet reached 6.75 m/s. This results in a Reynolds number of up to 2.4×10^4 , which is elicited using the building width and the wind velocity at the building height. A total of 186 measurement points were sprinkled throughout the wind tunnel. The sensors deployed collected real-time wind velocity components in the x , y , and z directions at the measurement points. The mean values and the standard deviations of the measured wind velocity in sixty seconds during the test were also provided serving as a public dataset for benchmark study.

2.2. Boundary conditions of the computational domain

The computational domain considered here is a cuboid with a size of $1.4 \text{ m} \times 0.7 \text{ m} \times 0.7 \text{ m}$ (length \times width \times height), as shown in Figure 1. The details about the position of the scale building model and the boundary conditions are provided in the figure as well.

In this study, the air density and the kinematic viscosity are set to be 1.225 kg/m^3 and $1.497 \times 10^{-5} \text{ m}^2/\text{s}$, respectively. The vertical surface OGDA is defined as an initial speed boundary and the x -direction velocity u_{isb} is defined as follows:

$$u_{isb} = u_{initial} \quad (1)$$

where $u_{initial}$ denotes the velocity distribution, which is shown in Figure 2. For more details about the wind tunnel test, the reader is referred to Meng and Hibi (1998).

The y -direction velocity component v_{isb} and the z -direction velocity component w_{isb} in the initial speed boundary are subject to the following constraints:

$$v_{isb} = 0 \quad (2)$$

$$w_{isb} = 0 \quad (3)$$

The horizontal surface ABED is defined as a symmetry wall boundary in which the velocity components are subject to the following constraints:

$$\frac{\partial u_{swb1}}{\partial z} = 0 \quad (4)$$

$$\frac{\partial v_{swb1}}{\partial z} = 0 \quad (5)$$

$$w_{swb1} = 0 \quad (6)$$

The vertical surfaces ABCO and DEFG are also defined as symmetry wall boundaries in which the velocity components are subject to the following constraints:

$$\frac{\partial u_{swb2}}{\partial y} = 0 \quad (7)$$

$$v_{swb2} = 0 \quad (8)$$

$$\frac{\partial w_{swb2}}{\partial y} = 0 \quad (9)$$

The vertical surface CFEB takes the form of a zero-pressure outlet boundary. The pressure on the boundary is described as follows:

$$p_{zpb} = 0 \quad (10)$$

Finally, the horizontal surface OCFG and the building surfaces are defined as no-slip wall boundaries, where

$$u_{wb} = 0 \quad (11)$$

$$v_{wb} = 0 \quad (12)$$

$$w_{wb} = 0 \quad (13)$$

In the above expressions, u , v , and w denote the x , y , and z -direction velocity components, respectively; p denotes the pressure; the subscripts isb , $swb1$, $swb2$, zpb , and wb denote the initial speed boundary, symmetry wall boundary of the first kind (surface ABED), symmetry wall boundary of the second kind (surfaces ABCO and DEFG), zero-pressure outlet boundary (surface CFEB), and wall boundary, respectively.

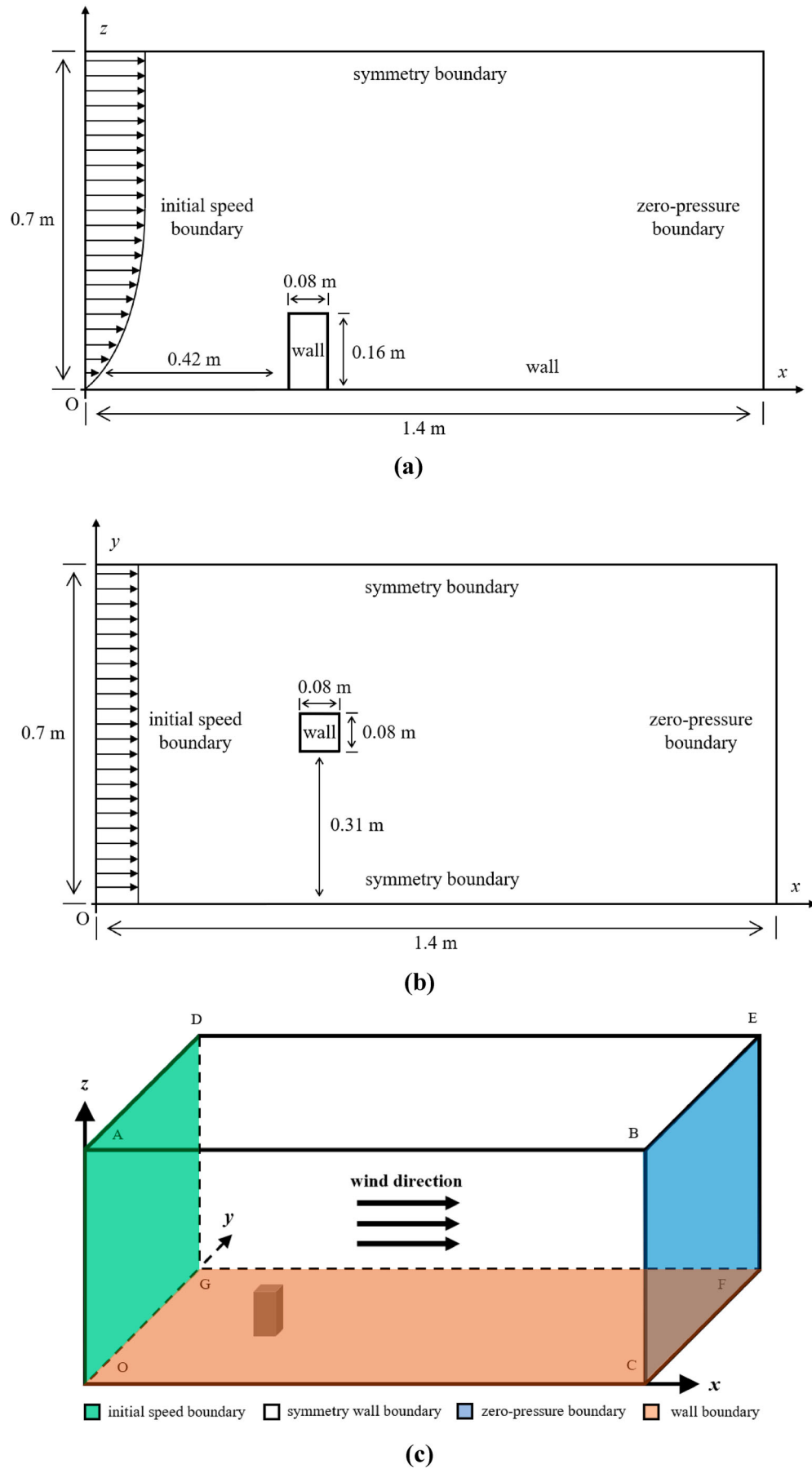


Figure 1. Computational domain and boundary conditions for 3D airflow simulation: (a) side view, (b) top view, and (c) general view.

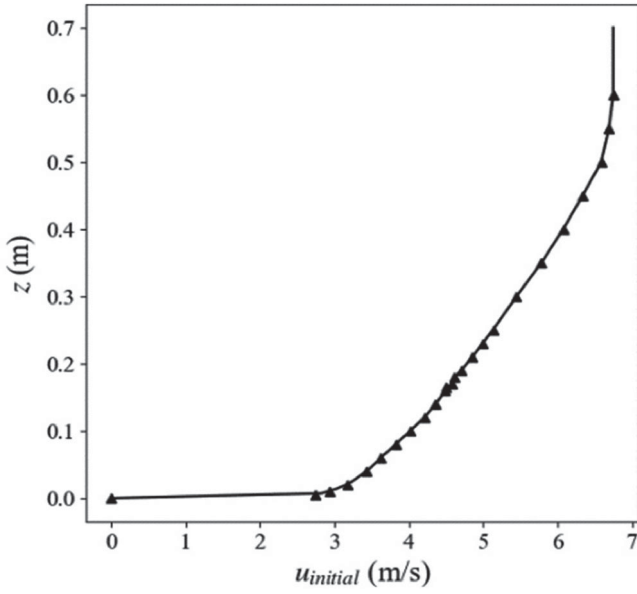


Figure 2. Distribution of velocity component $u_{initial}$ on the initial speed boundary.

2.3. Data constraints of the computational domain

As aforementioned, 186 measurement points are dispersed throughout the wind tunnel (also within the computational domain), as shown in Figure 3. Among these measurement points, 66 points are in the cross-section $y = 0.35$ m (the middle plane in the spanwise direction of the computational domain), and the remaining 120 points are distributed in the cross-sections $z = 0.01$ m and $z = 0.10$ m. Among the 186 measurement points, 93 near-wall points which are either close to the surface of the building model or near the ground will be utilized for supervised learning of PINN. More specifically, 21 points are in the cross-section $y = 0.35$ m, 60 in the cross-section $z = 0.01$ m, and 12 in the cross-section $z = 0.10$ m, as marked by red circles in Figure 3. The selection of training points ensures easy accessibility of the data even in real practice. At the training points, the PINN prediction of the three wind velocity components should satisfy the following constraints:

$$u_p = u_m \quad (14)$$

$$v_p = v_m \quad (15)$$

$$w_p = w_m \quad (16)$$

where the subscripts p and m denote the PINN predictions and the measured results at the training points, respectively.

3. Methodology

3.1. Zero-equation model of RANS equations

The RANS equations are an unclosed system of equations that incorporate a new quantity, i.e. the turbulent viscosity μ_t , in NS equations. To make the RANS equations tractable when CFD methods are executed, different models of the turbulent viscosity have been proposed in the past decades. Li model is one of the zero-equation models of turbulent viscosity (Li et al., 2013). To explain the Li model, the RANS equations for turbulent flow are expressed as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u_i) = 0 \quad (17)$$

$$\begin{aligned} \frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) \\ = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\mu_{eff} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] \end{aligned} \quad (18)$$

where ρ is the density of the fluid, u_i is the velocity component in the x_i direction, p is the pressure, and μ_{eff} is the effective viscosity of the fluid. The effective viscosity is the sum of the turbulent viscosity μ_t and the laminar viscosity μ , which is expressed as

$$\mu_{eff} = \mu_t + \mu \quad (19)$$

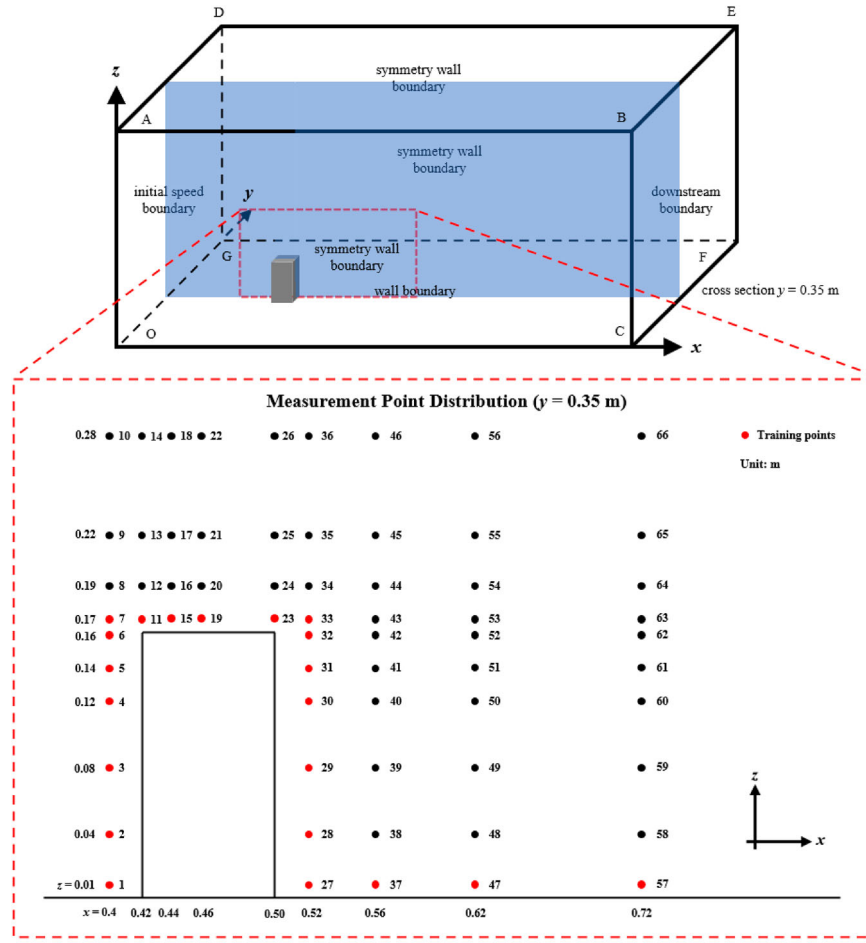
It should be noted that the introduction of the term μ_t makes the RANS equations no longer a closed-form system of equations. The Li model surmounts this difficulty by adopting a mixing-length strategy to represent the characteristics of air turbulence near buildings. In the Li model, the turbulent viscosity is defined as

$$\mu_t = \max(\mu_{in}, \mu_{out}) \quad (20)$$

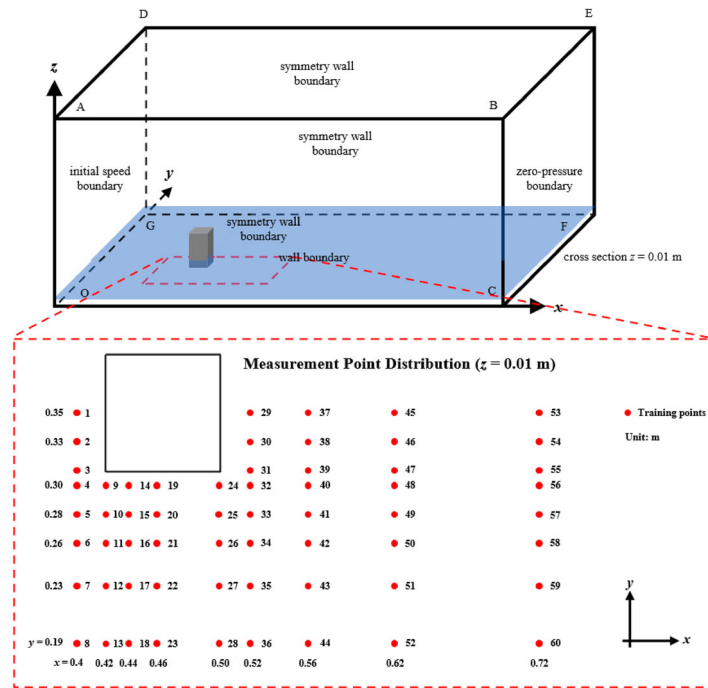
where μ_{in} and μ_{out} denote the viscosities of the inner layer flow (also known as boundary layer flow) and the outer layer flow, respectively. In the simulation of the inner layer flow, the Li model takes the form of a Prandtl mixing-length model (Prandtl, 1925), which can be expressed as

$$\mu_{in} = (C_{in} l)^2 S \quad (21)$$

where $C_{in} = 1.8 \times \left\{ 1 - \exp \left[-0.645 \left(\frac{C}{H} \right)^{0.8} \right] \right\} \times \exp \left[-2 \times \min \left(\frac{z}{H}, 1 \right) \right]$, l represents the distance from the nearest wall, and $S = \sqrt{\frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)^2}$. C and H are the width and height of the windward side of the building, respectively. z is the height of the grid points. For the airflow away from a building, the turbulent viscosity can



(a)



(b)

Figure 3. Distribution of measurement points in the cross-sections (a) $y = 0.35$ m, (b) $z = 0.01$ m, and (c) $z = 0.10$ m.

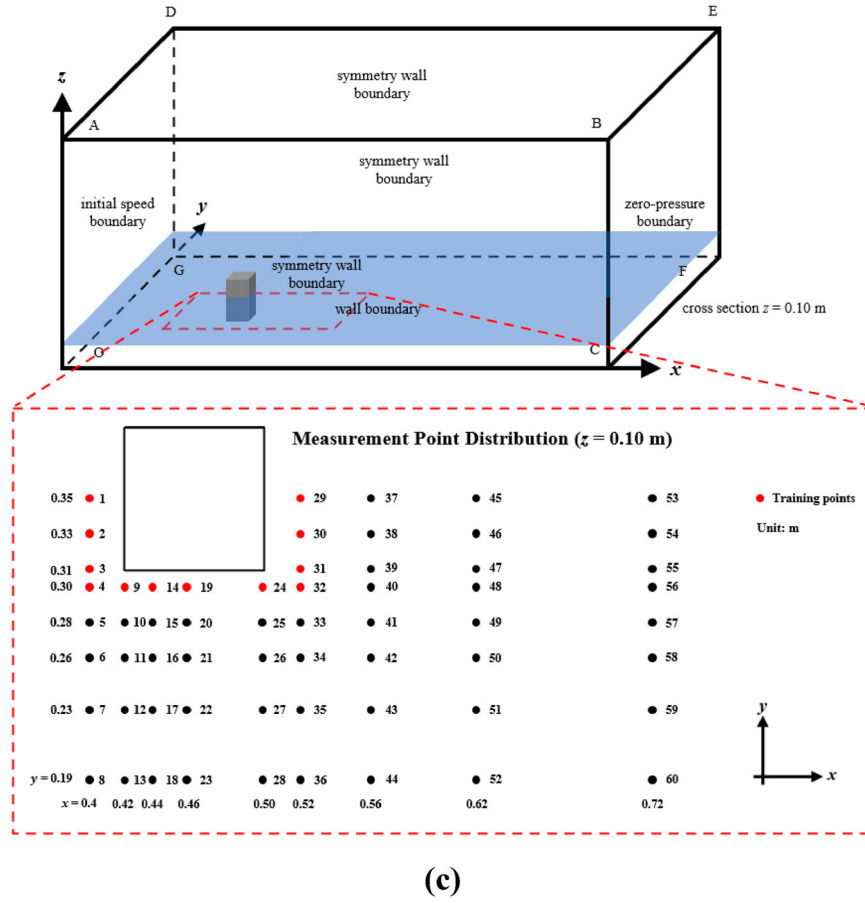


Figure 3. Continued

be expressed as the product of the coefficient C_{out} , local velocity V , and length l .

$$\mu_{out} = C_{out} V l \quad (22)$$

where the coefficient C_{out} is expressed as

$$C_{out} = \frac{C_{\mu}^{0.5} I_G^2 z_G^{2\alpha+0.1} z^{0.9-2\alpha}}{\alpha} \quad (23)$$

where $C_{\mu} = 0.09$, $I_G = 0.1$, $z_G = 350$, and $\alpha = 0.22$. When the Li model is embedded in RANS equations, the governing differential equations of the turbulent flow become tractable.

3.2. Basic framework of PINN

By directly embedding physical equations, initial and boundary conditions, and measurement data into the total loss function for training a neural network, the PINN is proven to be a charming physics-informed data-driven approach for solving various types of PDE problems. The residuals of these physical constraints tend to converge toward zero during the training process with the aid of an optimizer. In this way, the PINN is configured to achieve its function to approximate solutions to

PDEs. The schematic diagram of the PINN targeting to achieve the 3D airflow simulation is depicted in Figure 4. The left part of the PINN is a fully connected neural network that maps the relationship between the spatial coordinates (x, y, z) and flow characteristics of the wind field (u, v, w, p) , here u , v , and w represent the velocity components in the x , y , and z directions, respectively, and p represents the pressure. In the middle part of the PINN, automatic differentiation (AD) is applied to calculate the gradients of the outputs with respect to the inputs, which plays a key role in the neural network training (Baydin et al., 2018). The right part of the PINN is the total loss, which is calculated by

$$L = w_f L_f + w_b L_b + w_d L_d \quad (24)$$

where

$$L_f = \frac{1}{N_f} \sum_{n=1}^{N_f} \sum_{i=1}^4 |f_i^n|^2 \quad (25)$$

$$L_b = \frac{1}{N_{nb}} \sum_{i=1}^{N_{nb}} |r_{nb}^i|^2 + \frac{1}{N_{db}} \sum_{i=1}^{N_{db}} |r_{db}^i|^2 \quad (26)$$

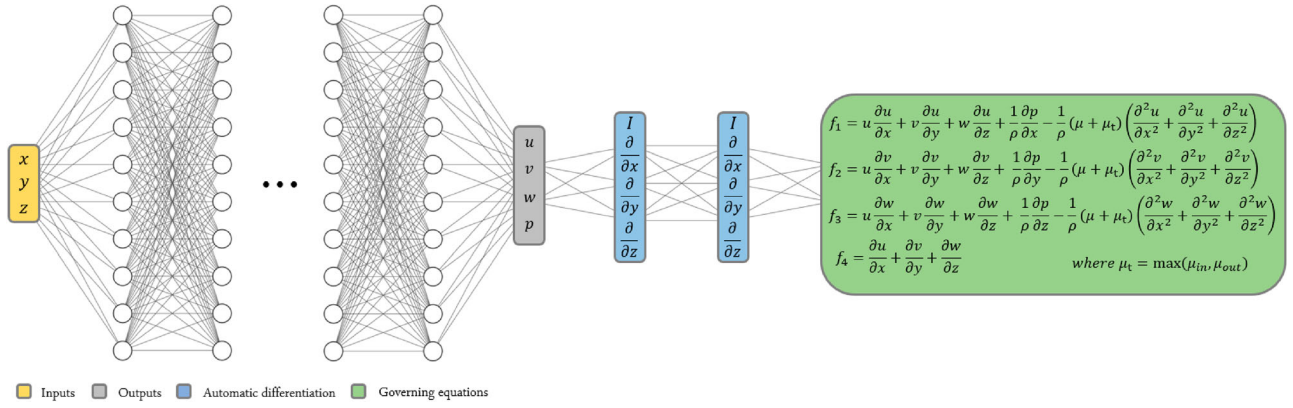


Figure 4. Schematic diagram of PINN for 3D airflow simulation.

$$L_d = \frac{1}{N_d} \sum_{i=1}^{N_d} |r_d^i|^2 \quad (27)$$

In the above expressions, L_f , L_b , and L_d denote the loss components associated with the residuals of the governing equations, boundary conditions, and training data, respectively; w_f , w_b , and w_d denote the weighting coefficients of the corresponding loss terms; f_i^n is the residual of the i th governing equation in Figure 4; r_{nb}^i , r_{db}^i , and r_d^i are the residuals for the Neumann boundary, Dirichlet boundary, and training data, respectively. N_f is the number of collocation points used to calculate the residual of the governing functions, while N_{nb} , N_{db} , and N_d are the numbers of points used to calculate the residuals for the Neumann and Dirichlet boundaries, and for the training data, respectively.

3.3. Dynamic prioritization loss balance strategy for PINN (dpPINN)

How to balance the loss terms in the total loss function during the training process of PINN is a challenging issue (Xiang et al., 2022). Imbalance between the loss terms may significantly diminish the convergence rate and computational efficiency in the training of PINN. To alleviate this problem, we rewrite the total loss in the following form:

$$L = L_f + w_u L_u + w_v L_v + w_w L_w + w_p L_p \quad (28)$$

where

$$L_u = \sum_{n=1,4,7,11,14} \frac{1}{N_n} \sum_{i=1}^{N_n} |r_n^i|^2 \quad (29)$$

$$L_v = \sum_{n=2,5,8,12,15} \frac{1}{N_n} \sum_{i=1}^{N_n} |r_n^i|^2 \quad (30)$$

$$L_w = \sum_{n=3,6,9,13,16} \frac{1}{N_n} \sum_{i=1}^{N_n} |r_n^i|^2 \quad (31)$$

$$L_p = \frac{1}{N_{10}} \sum_{i=1}^{N_{10}} |r_{10}^i|^2 \quad (32)$$

Unlike the original expression in Eq. (24), the total loss function is now reshaped to consist of five separate components, i.e. L_f , L_u , L_v , L_w , and L_p . L_f has the same meaning as given before; L_u , L_v , and L_w denote the loss terms directly related to the velocity components u , v , and w , respectively; L_p denotes the loss term related to the pressure p ; w_u , w_v , w_w , and w_p denote the weighting coefficients of the corresponding loss terms. r_n^i is the residual of Eq. (n), and N_n is the number of points used to calculate the residual of Eq. (n). It can be seen that the residuals of Eqs. (1) to (16) form the loss terms L_b and L_d in Eq. (24) in the original PINN configuration. They are reorganized in our approach to form the u , v , w , and p -related loss terms. For instance, the residuals of Eqs. (1) and (14) belong to different loss terms in the original PINN configuration, i.e. L_b and L_d , respectively. However, in the new configuration, both belong to the loss term L_u in Eq. (28). Such a reconfiguration enables us to balance each loss term in the following way:

$$w_i = \left[\frac{k_i}{\min_j(k_j)} \right]^\gamma, \quad i, j = u, v, w, p \quad (33)$$

$$k_i = \frac{\|U_{T_i} - \tilde{U}_{T_i}\|_2}{\|U_{T_i}\|_2}, \quad i = u, v, w, p \quad (34)$$

where γ is a newly introduced hyperparameter that affects the weight balance between different loss terms; $\|\cdot\|_2$ denotes the ℓ_2 -norm; U_{T_i} denotes the vector of the reference data on the training points; and \tilde{U}_{T_i} denotes the vector of PINN predictions on the training points. Such a setting means that we calculate the relative errors of u ,

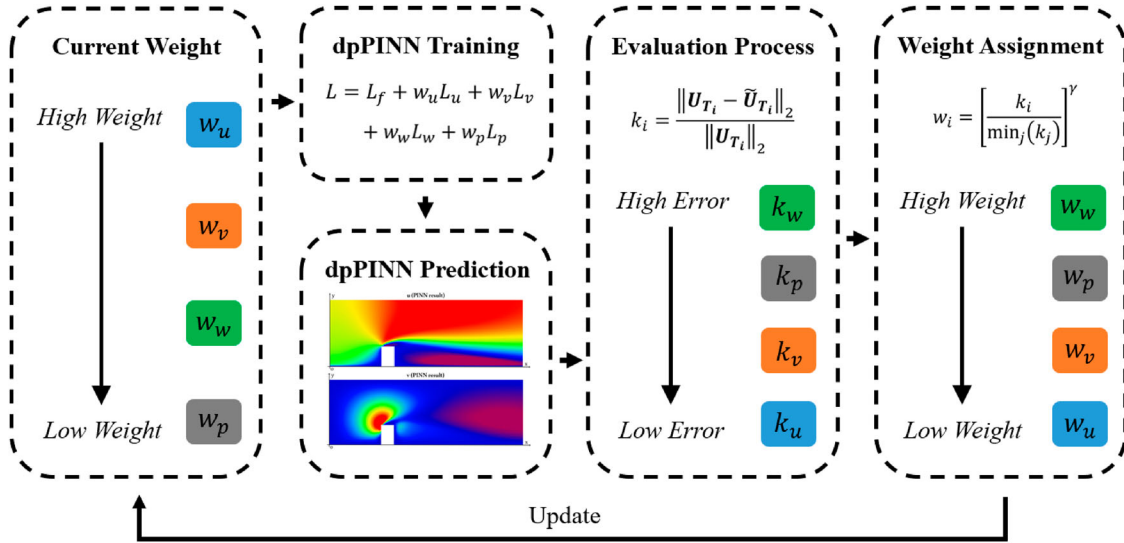


Figure 5. Updating mechanism of the dpPINN weighting coefficients.

v , w , and p between the PINN predictions and the reference data at the training points after certain iterations. A flowchart is depicted to further illustrate the updating mechanism of the dpPINN weighting coefficients, as shown in Figure 5. In the first step, current weights will be assigned to the loss function of a dpPINN and the dpPINN will make its prediction as a normal neural network model. What follows is an evaluation process, where the errors between dpPINN prediction and the reference data on the training points, i.e. Eq. (34), are calculated. For example, in Figure 5, the velocity component w is found to possess the highest relative error during the evaluation process. Then it will be given the highest weight using Eq. (33) in the following training process, as a punishment for the low prediction accuracy in the past training process. The assigned weight will be used to form the total loss to update the neural network parameters.

The core idea of the dpPINN is that we desire to automatically prioritize the terms with larger relative errors in the subsequent iteration steps. Similar ideas of dynamic prioritization loss balance strategies can also be found in the field of computer vision and multi-task learning (Lin et al., 2017; Guo et al., 2018). It should be mentioned that we do not explicitly define a weighting coefficient w_f for L_f in Eq. (28); instead, we indirectly adjust the weight of L_f in the total loss by tuning the value of the coefficient γ . The influence of the coefficient γ on the prediction accuracy will be discussed in detail later. The reconfigured PINN, referred to as dpPINN, encompasses a new total loss defined in Eq. (28) and the dynamic prioritization loss balance strategy. The implementation of the dpPINN paradigm for simulating the 3D airflow around a building is presented in Algorithm 1.

Algorithm 1: dpPINN for 3D airflow simulation using RANS equations

Require: Training dataset, training iteration, learning rate, initial values of weighting coefficients, and the value of the coefficient γ .

Target: Find the best prediction model with appropriate neural network parameters.

- Step 1:** Construct a deep neural network with the initial neural network parameters.
- Step 2:** Specify the collocation points in the computational domain.
- Step 3:** Calculate L_f , L_u , L_v , L_w , and L_p at the collocation and measurement points using AD.
- Step 4:** Use gradient descent algorithms to update the neural network parameters as:

for each iteration:

- (a) Calculate the total loss function Eq. (28) using the values of the weighting coefficients from the previous iteration.
- (b) Update the neural network parameters using the optimizer with a fixed learning rate by minimizing the total loss function.
- (c) Update the weighting coefficients according to Eq. (33) and (34).

end for

4. Results

4.1. Preliminary validation results

As mentioned before, this study aims to reconstruct the flow field around the scale model of a building in wind

tunnel by using sparse near-wall velocity data within the dpPINN framework. In this section, we validate the dpPINN predictions with the measured wind velocity components at the measurement points from which the data were not used in dpPINN training. All simulations are conducted on the platform equipped with PyTorch v1.9.0 and NVIDIA A100 graphics processing units.

The value of the coefficient γ is preliminarily set as 2. Because no pressure data is embedded in the dpPINN training, the weighting coefficient w_p is set to 1 in this case. A dpPINN with six hidden layers, each containing 40 neurons, is formulated for outdoor airflow simulation. *Tanh* is adopted as the activation function in the neural network. The *Adam* optimizer with 1×10^5 iterations is used to train the dpPINN. The learning rate is set as 3×10^{-4} , and 26,944 collocation points are uniformly distributed within the cubic computational domain. These collocation points are used to compute the residuals of the governing equations. Additionally, 9900 collocation points distributed on the domain boundaries are used to compute the residuals of the boundary conditions. In this case study, one training iteration takes about

0.318 s. It takes about 53.6% of the duration for AD to calculate the physical residuals and form the total loss, and the remaining 46.4% is for error backpropagation to update the neural network parameters.

As indicated previously, among the total 186 measurement points, we select 93 near-wall points for training the dpPINN (21 in the cross-section $y = 0.35$ m, 60 in the cross-section $z = 0.01$ m, and 12 in the cross-section $z = 0.10$ m), as shown in Figure 3. The velocity data at the 93 training points are used for supervised learning and updating of the weighting coefficients. Excluding the training points, the remaining measurement points (the data from which are referred to as reference data) are used to verify the proposed approach for outdoor airflow simulation. After training the dpPINN model, its prediction results are obtained as shown in Figures 6 and 7, which are compared with the reference data from wind tunnel measurements. The black solid lines, red dotted lines, and blue dash-dot lines represent the dpPINN predictions of the velocity components u , v , and w , respectively. The black dots, red squares, and blue diamonds denote the experimental mean values of the velocity components u , v , and w , respectively, at the measurement points. From

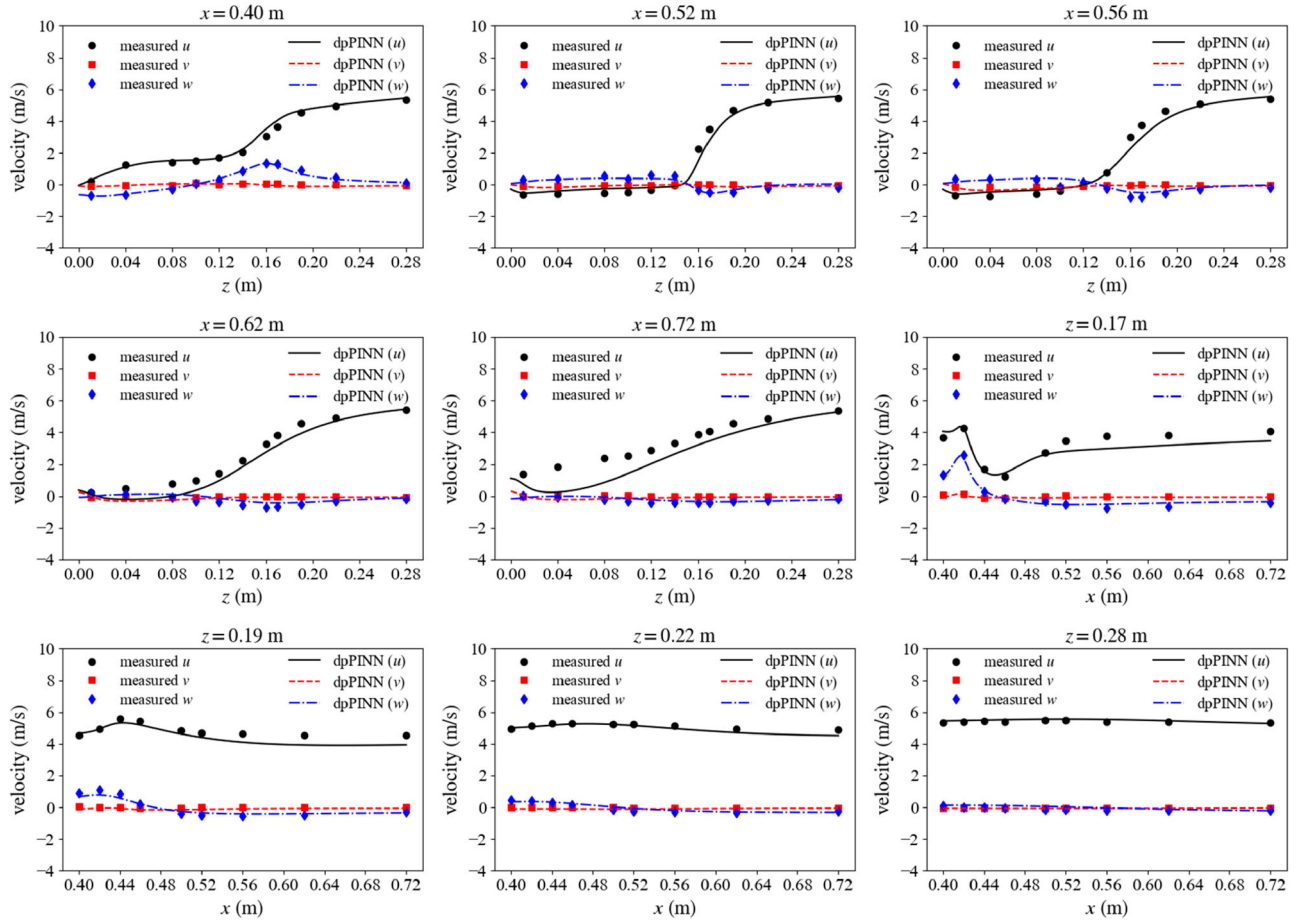


Figure 6. Comparison of results between dpPINN predictions and wind tunnel measurements in the cross-section $y = 0.35$ m.

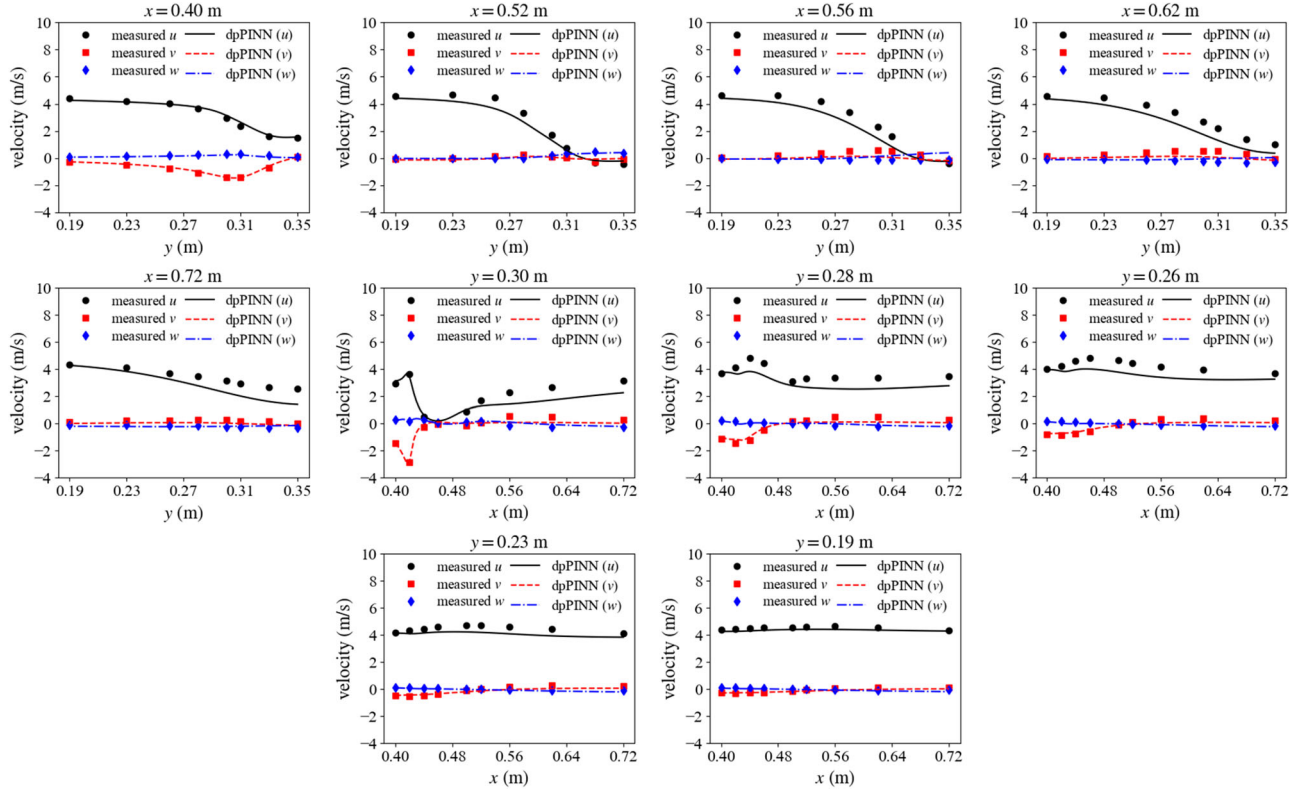


Figure 7. Comparison of results between dpPINN predictions and wind tunnel measurements in the cross-section $z = 0.10$ m.

the results, it can be found that the dpPINN predictions show a good agreement with the experimental data even when the experimental data used for verification were not included in training the dpPINN.

We take the top left panel of Figure 6 as an example for further exploration. Eight training points (points 27–33 in Figure 3(a) and point 29 in Figure 3(c)) and three testing points (points 34–36 in Figure 3(a)) are scattered on the line $x = 0.40$ m in the cross-section $y = 0.35$ m. The dpPINN not only generates agreeable simulation results of the velocity component u when z (height) is less than 0.17 m but also provides good predictions in the areas without training data ($z > 0.17$ m). As another example, on the line $x = 0.56$ m, only one training data point (point 37 in Figure 3(a)) is included for dpPINN training. However, the dpPINN predictions still show a good agreement with the field-measured results for the other 10 testing points (points 38–46 in Figure 3(a) and point 37 in Figure 3(c)). After examining all the results, it is found that dpPINN demonstrates good prediction accuracy in other regions as well. One may easily come to the first conclusion about the advantage of this physics-informed data-driven approach for outdoor airflow simulation around buildings: Compared with pure data-driven methods, the embedding of physical laws in the dpPINN framework enables us to use few datasets to generate a model with strong generalization

and forecasting capability, which is of great practical significance.

4.2. Comparison of different loss balance strategies

In the following, we use ℓ_2 error to quantitatively evaluate the accuracy of the dpPINN prediction, which is defined as.

$$l_2error = \frac{U_i - \tilde{U}_{i2}}{U_{i2}} \times 100\% \quad (35)$$

where $\|\cdot\|_2$ denotes the ℓ_2 -norm, U_i denotes the vector of the reference data, and \tilde{U}_i denotes the vector of the dpPINN predictions. The ℓ_2 -norm of a vector $x = (x_1, x_2, x_3, \dots, x_n)$ is calculated as $\sqrt{\sum_{i=1}^n x_i^2}$. To quantify the relative error of the dpPINN prediction, the ℓ_2 errors of the velocity component u and the total velocity are recorded during the training process, as depicted in Figure 8. The value of the coefficient γ is still set as 2, same as in Section 4.1. It is seen that, after 1×10^5 iterations, the ℓ_2 errors of u and the total velocity are reduced to 0.125 and 0.123. Eivazi and Vinuesa (2022) recently investigated the influence of observation noise on PINN prediction accuracy. Their results indicated that noisy training data would interfere with the PINN prediction and make it less accurate. In view of this, the above

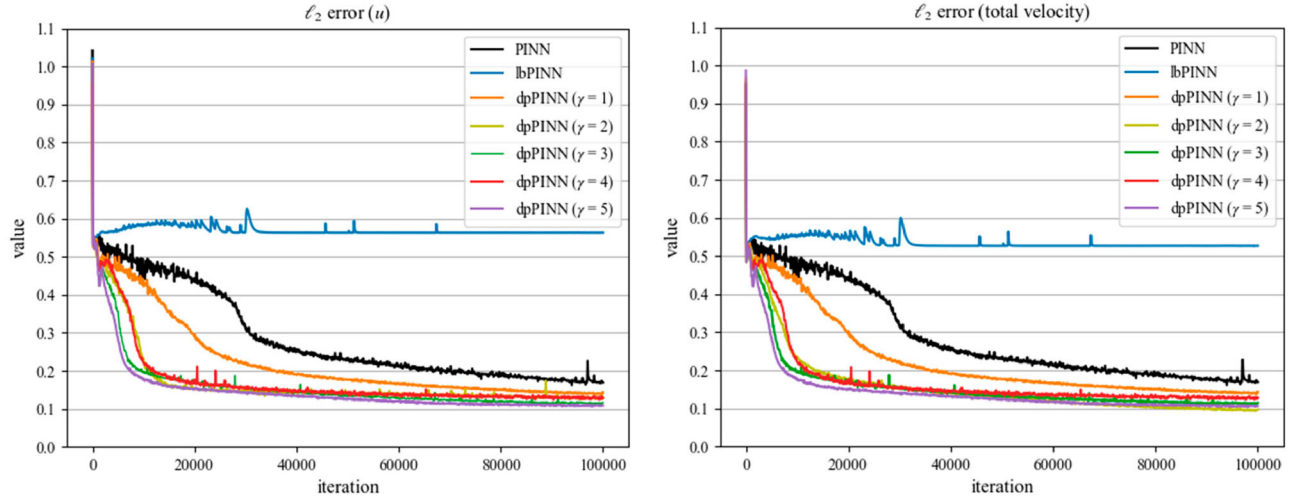


Figure 8. ℓ_2 errors of PINN predictions using different loss balance strategies (left: u ; right: total velocity).

results would be acceptable in consideration of the existence of measurement noise during the wind tunnel test. Also, we compare the prediction accuracy between the dpPINN and other PINNs adopting different loss balance strategies, as shown in Figure 8, where the influence of the coefficient γ is investigated as well. In Figure 8, the black lines represent the original PINN, which uses Eq. (24) as the total loss. The values of the weighting coefficients w_f , w_b , and w_d are set as 1 during the training process. The blue lines represent the loss balance strategy ‘lbPINN’ proposed by Xiang et al. (2022), which updates the weighting coefficients based on the maximum likelihood estimation. The other lines represent the results of the proposed dpPINN with different values of the coefficient γ .

Unfortunately, the lbPINN seems inapplicable in this case. The ℓ_2 errors of the velocity components u and the total velocity reach more than 0.5 when the lbPINN is adopted, which means that the total loss of the neural network is unable to converge toward zero in this case. It is probably because the lbPINN is inapplicable for solving 3D incompressible turbulent flow problems. Better solutions can be achieved by the original PINN. The ℓ_2 errors of the velocity component u and the total velocity drop eventually to 0.169 after 1×10^5 iterations. By contrast, the dpPINN shows much better performance under the same training iteration; in particular, the final ℓ_2 error is not significantly affected by the value of the coefficient γ . Compared with the original PINN, the ℓ_2 errors of the velocity component u and the total velocity are reduced by roughly 20% to 30% when γ ranges from 1 to 5 after 1×10^5 iterations. However, we cannot currently draw the conclusion that dpPINN outperforms PINN in prediction accuracy because the slow convergence rate seems to limit the potential of the PINN, so we extend the training process for additional 8×10^4 iterations to a

Table 1. ℓ_2 errors of the original PINN, lbPINN, and dpPINN predictions after 1×10^5 iterations.

Type	ℓ_2 error of u	ℓ_2 error of v	ℓ_2 error of w	ℓ_2 error of total velocity
PINN	0.169	0.271	0.528	0.169
lbPINN	0.562	0.947	1.010	0.526
dpPINN ($\gamma = 1$)	0.138	0.283	0.437	0.138
dpPINN ($\gamma = 2$)	0.125	0.248	0.314	0.123
dpPINN ($\gamma = 3$)	0.112	0.238	0.323	0.110
dpPINN ($\gamma = 4$)	0.124	0.258	0.357	0.123
dpPINN ($\gamma = 5$)	0.107	0.224	0.328	0.105

total of 1.8×10^5 iterations. The findings reveal that the two approaches estimate u and v with similar accuracy, while the error of w predicted by dpPINN ($\gamma = 2$) is still 4% lower than that predicted by PINN.

A more detailed comparison is tabulated in Table 1. In addition, Figure 8 indicates that a rough solution can be quickly achieved after only about 3×10^4 iterations, demonstrating that the computational efficiency is greatly enhanced when adopting the dpPINN. It should be indicated that the ℓ_2 errors of the velocity components v and w are not discussed here because they are near-zero and very small compared to u , so they are assumed to be less important in this case; nevertheless, the results of these two components are still tabulated in Table 1. The dynamic balance process of weighting coefficients of the dpPINN ($\gamma = 2$) for 1×10^5 iterations is also monitored, as shown in Figure 9. Since u demonstrates the minimum relative error among those of all velocity components, it possesses the lowest weight, which is stabilized at 1, in its training process. In addition, the relative error of the velocity component w is always greater than that of v , thus it is penalized with a higher weight. dpPINN appears to be striving for the best balance in the first 4×10^4 iterations, while in the last 6×10^4 iterations, w_u , w_v and w_w gradually balance at the ratio of 1.0: 2.4: 5.2.

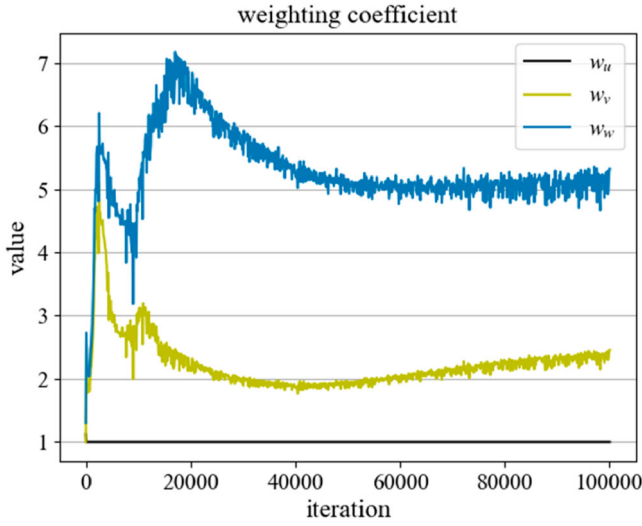


Figure 9. The dynamic balance of weighting coefficients in the dpPINN.

Table 2. Relative ℓ_2 errors of dpPINN predictions with different neural network configurations.

Depth Width	4		6		8	
	u	total velocity	u	total velocity	u	total velocity
20	0.251	0.253	0.185	0.184	0.170	0.167
40	0.130	0.129	0.125	0.123	0.116	0.119
60	0.131	0.128	0.109	0.108	0.143	0.143

4.3. Influence of the neural network configuration

This section discusses the influence of neural network configuration on the dpPINN prediction accuracy. The value of the coefficient γ is set as 2. Nine different neural network configurations are considered, with three different depths and three different widths. The relevant results are displayed in Table 2. In general, a broader width of the neural network is beneficial to the overall performance of the dpPINN, but this involves more computational expense as a side effect. In comparison, the depth of the neural network has a less positive influence on prediction accuracy. To summarize, the neural network containing six hidden layers, each with 60 neurons, is found to possess the best performance among the nine configurations.

4.4. Investigation on the embedded turbulence model

Next, we perform a further analysis to identify the impact when different turbulence models are encoded in the dpPINN. The Chen model is another widely used zero-equation turbulence model for airflow simulation (Chen & Xu, 1998). It assumes that the turbulent viscosity is

the product of air density, the local velocity V , and the distance from the nearest wall l , which is calculated by

$$\mu_t = 0.03874\rho V l \quad (36)$$

We attempt to embed the Chen model instead of the Li model into dpPINN for the airflow simulation. The dpPINNs using the Li model and the Chen model are termed Li-dpPINN and Chen-dpPINN, respectively. The neural network configuration keeps the same as before (6 hidden layers, each with 40 neurons). The value of the coefficient γ is still set as 2. The corresponding results are depicted in Figure 10. Figure 10(a) shows the contours of the velocity component u in the cross-section $y = 0.35$ m predicted by Li-dpPINN and Chen-dpPINN. From the velocity contours, it can be found that when we employ the Chen model in lieu of the Li model for airflow simulation by dpPINN, a bigger clockwise-rotating vortex tends to appear near the leeward side of the building, which contradicts the measurement results in the near-ground region. This is probably due to the applicability of the encoded turbulence model, i.e. the Chen model, which is more appropriate for indoor airflow simulation than for outdoor airflow simulation.

Under the same circumstances, we try to include more measurement points in dpPINN training. The number of training data is increased from the original 93–100 by adding points 48–53 shown in Figure 3(a) and point 45 shown in Figure 3(c). The addition of velocity data at these measurement points helps to provide proper guidance for the dpPINN solution in the leeward side of the building, which is also the region with the maximum relative error when different turbulence models are encoded. The results after adopting more training data are provided in Figure 10(b) and Figure 11. Figure 10(b) shows the contours of the velocity component u using different turbulence models when more labelled data are incorporated. Figure 11 compares the Chen-dpPINN predictions of the velocity component u in the cross-section $y = 0.35$ m with the Li-dpPINN predictions at the same locations. The ℓ_2 errors of the velocity component u drop to 0.089 and 0.128 when seven more measurement points are used for training the Li-dpPINN and the Chen-dpPINN, both of which are acceptable in engineering practice. It is observed that even when more training data are used for supervised learning, the Li-dpPINN model still performs better than the Chen-dpPINN model in prediction. Additionally, it is found that even when a less appropriate turbulence model, i.e. the Chen model, is encoded in dpPINN for airflow simulation, the solution is still valid with satisfactory prediction accuracy, due to the existence of field-measured training data which help rectify the size of the vortex near the leeward side of the building. The Li-dpPINN and Chen-dpPINN

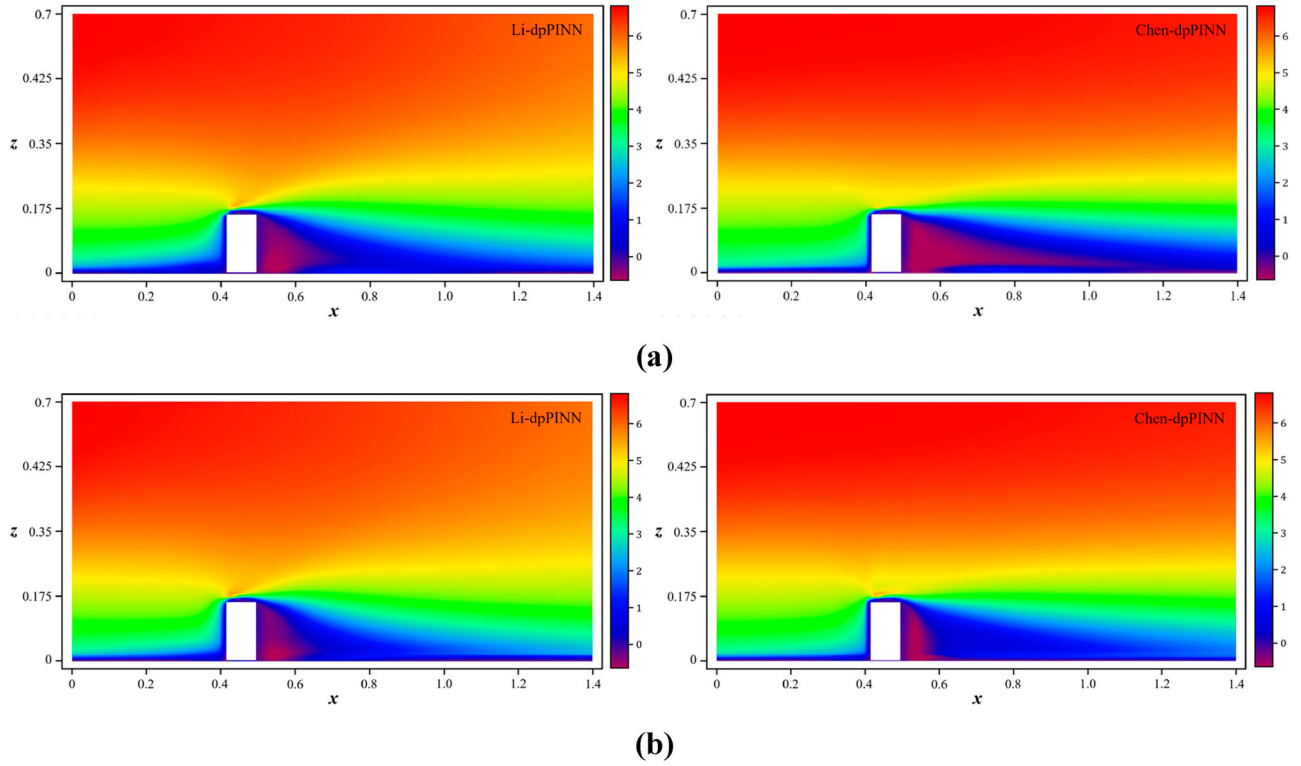


Figure 10. Contours of u in the cross-section $y = 0.35$ m: (a) 93 measurement points are used to train the dpPINN model; (b) 100 measurement points are used to train the dpPINN model.

predictions of the velocity component u in the cross-section $y = 0.35$ m show good consistency with the measurement results, although the former is slightly better than the latter. This manifests another advantage of the physics-informed data-driven approach for outdoor airflow simulation: The PINN efficiently leverages the information of physical laws and observation data in the same network configuration, which is difficult to realize when adopting traditional physics-based methods. This also broadens the applicability of physical models, which in turn makes the PINN a robust framework for airflow simulation and others.

Furthermore, two well-known two-equation RANS turbulence models, i.e. the standard $k-\varepsilon$ model and the Re-Normalization Group (RNG) $k-\varepsilon$ model, are also encoded in the dpPINN model. Figure 12 depicts the flow streamlines in the building's leeward recirculation zone obtained by various RANS turbulence models. Here, we focus on exploring the streamline distribution in an area with a size of 0.2 meters by 0.2 meters which is on the leeward side of the building and near the ground. The results indicate that when using the zero-equation RANS turbulence models, the airflow recirculation zone cannot be well portrayed; but when the two-equation models are adopted, appropriately sized refluxes are generated in the leeward side. The reason might be that dpPINN fails to effectively limit the residuals of the

physical governing equations when solving the zero-equation model-embedded RANS equations. Anyway, the constraints of physical conditions in dpPINN are relatively flexible, i.e. by adding penalty terms in the loss function. As a result, it is not surprising that its solution sometimes deviates. We believe that further in-depth research is needed to specifically address this issue.

4.5. Influence of the layout arrangement of the training points

This section discusses the influence of the layout arrangement of the training points on the dpPINN prediction accuracy. Here, we maintain the consistency of neural network hyperparameters. The test embraces three distinct training point layout arrangements, which are named Arrangements 1, 2, and 3, respectively. These layout arrangements only differ on the cross-section $z = 0.01$ m, as depicted in Figure 13. In Arrangements 1, 2, and 3, the cross-section $z = 0.01$ m contains 60, 20, and 12 training points, leading to a total of 93, 53, and 45 training points within the entire computational domain respectively. Table 3 presents the results. When contrasting Arrangements 1 and 2, it may be discovered that the number of training points appears to have a slight effect on the precision of predictions for the velocity

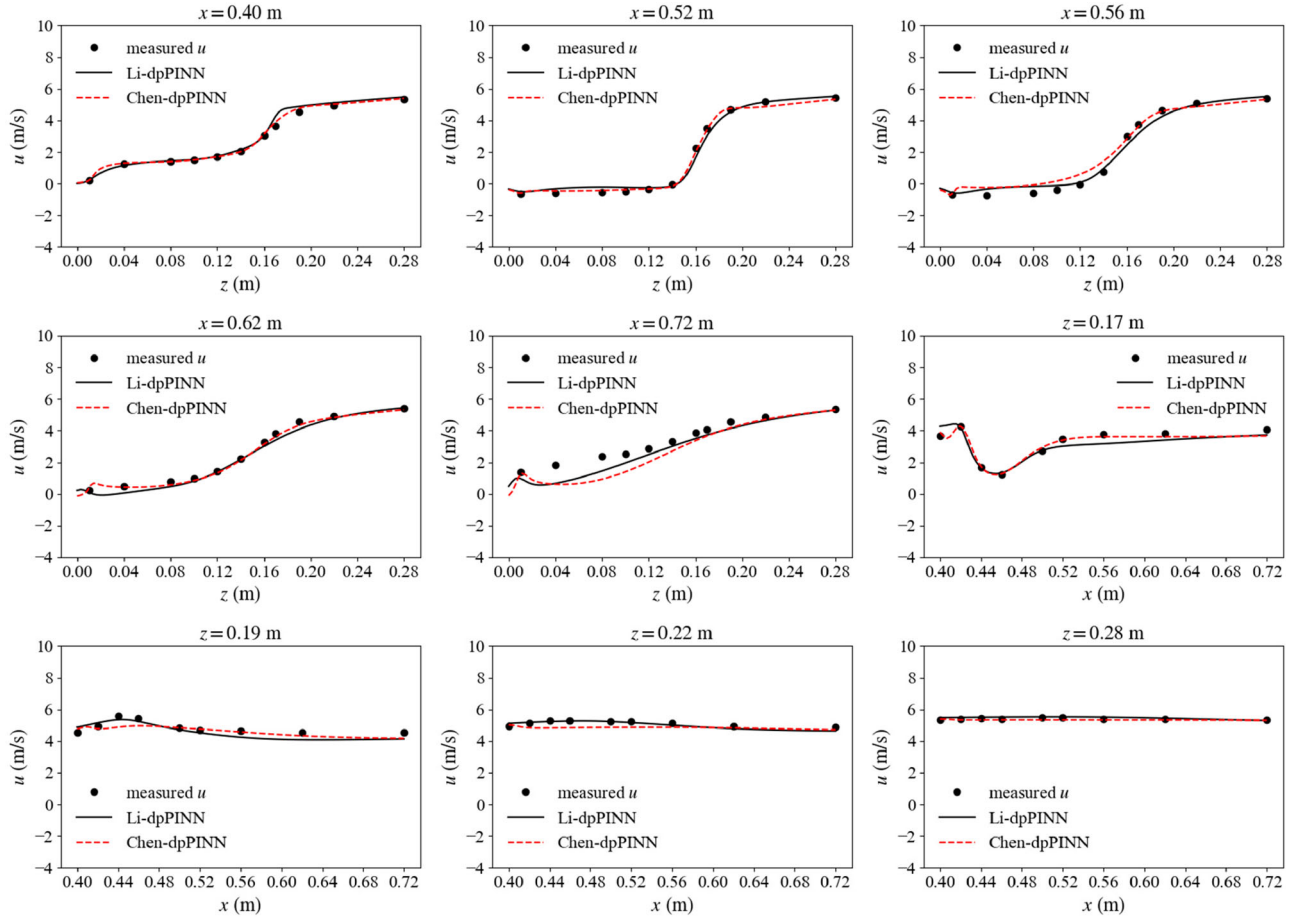


Figure 11. Comparison of Li-dpPINN predictions with Chen-dpPINN predictions of the velocity component u in the cross-section $y = 0.35$ m.

Table 3. l_2 errors when different training points layout arrangements are adopted.

Arrangement No.	1	2	3
l_2 error of u	0.125	0.126	0.317
l_2 error of v	0.248	0.444	0.536
l_2 error of w	0.314	0.478	0.530
l_2 error of total velocity	0.123	0.128	0.314

component u and total velocity. Its range of distribution, by contrast, is more significant. The accuracy of the dpPINN model is considerably lowered in Arrangement 3 when the points away from the building (i.e. No. 6, 8, 34, 36, 53, 56, 58, and 60) are excluded from training. Therefore, a decentralized layout of training points improves dpPINN's prediction accuracy. This is because training data can directly influence how dpPINN is trained in a local region, increasing its prediction accuracy at a global scale.

5. Conclusions

Aiming to precisely reconstruct the entire flow field around the scale model of a building in a wind tunnel, a

novel PINN framework encoding a zero-equation RANS turbulence model and sparse near-wall velocity measurement data was proposed in this study. The proposed PINN framework, termed dpPINN, is configured to make its total loss function in compliance with the dynamic prioritization self-adaptive loss balance strategy in PINN training. The performance and prediction accuracy of the dpPINN framework were verified by using measurement data from the wind tunnel test and comparing it with other PINN paradigms adopting different loss balance strategies. The influence of different neural network configurations, adopting different turbulence models, and different layout arrangements of training points on the dpPINN prediction accuracy are also investigated. The results indicate that the dpPINN framework could be a powerful means for the prediction of spatial flow fields around a building in wind tunnel tests and in wind engineering applications. Conclusions are drawn as follows:

- When only a small amount of sparse near-wall data is accessible for reconstructing the 3D flow field around a building model in a wind tunnel test, the missing

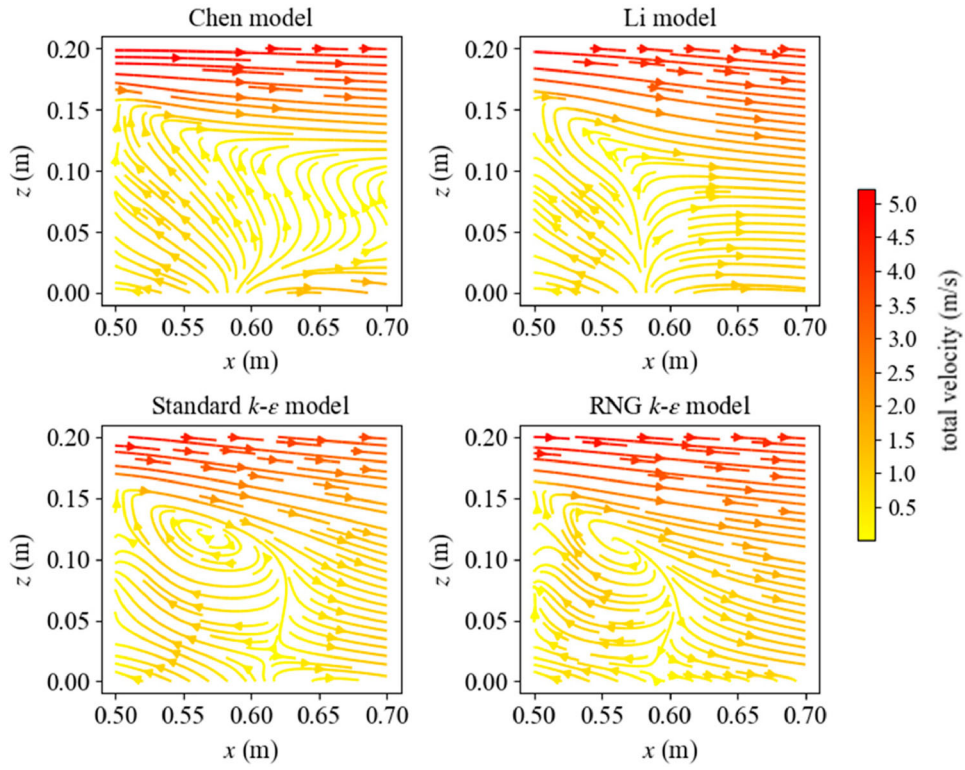


Figure 12. The flow streamlines in the building's leeward recirculation zone.

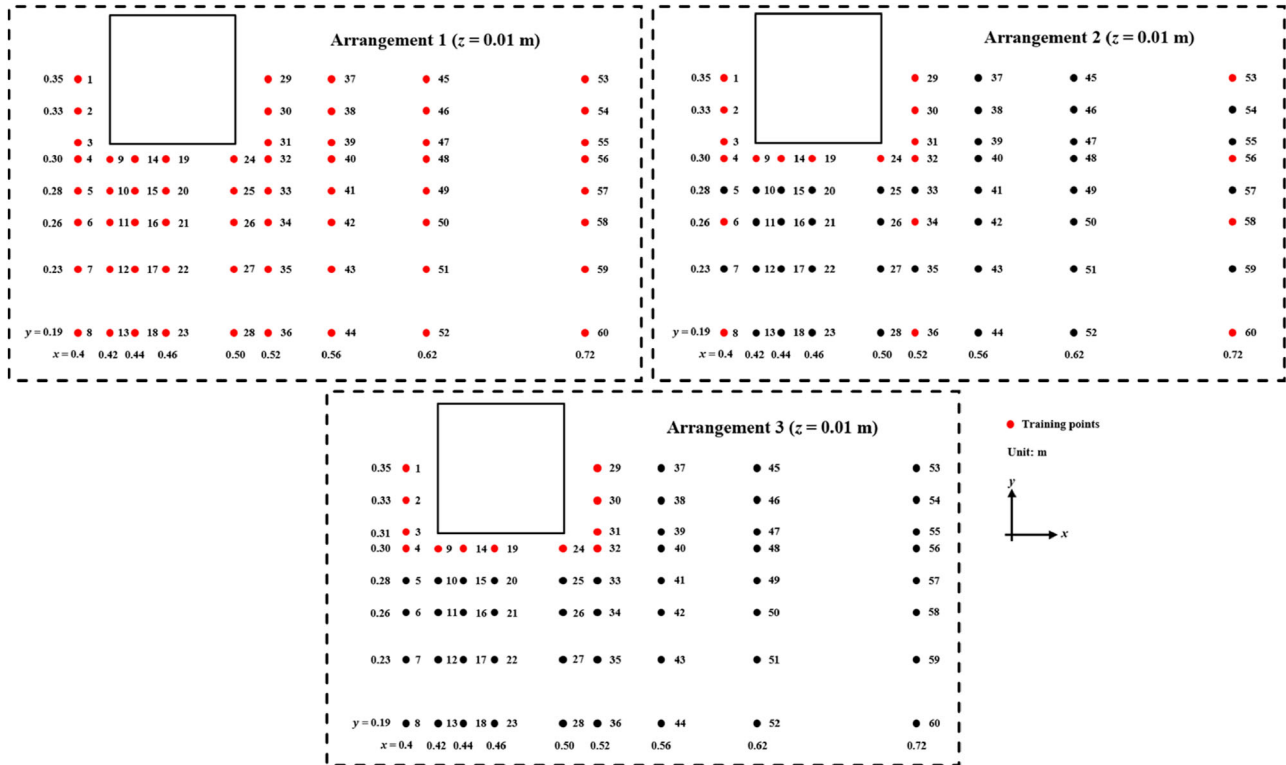


Figure 13. Three layout arrangements of training points on the cross-section $z = 0.01$ m.

airflow information within the whole computational domain can be well recovered by encoding the physical laws and measurement data into PINN;

- The dpPINN framework proposed in this study outperforms the original PINN and lbPINN for the airflow simulation in terms of prediction accuracy and

computing speed. In the proposed approach, the loss terms with higher relative errors are prioritized during training, which leads to a faster convergence rate and more accurate prediction results;

- Compared with the depth, the width of the neural network is in general more influential on the prediction accuracy. A broader width helps improve the dpPINN performance, but at the cost of higher computational expense. The configuration of six layers, each with 60 neurons, is found to be the best among the nine configurations considered in this study;
- Using different turbulence models of RANS equations as embedded physics directly affects the dpPINN solution. A bigger clockwise-rotating vortex tends to appear near the leeward side of the building when the Chen model in lieu of the Li model is encoded into dpPINN. However, when more training data are available, predictions by either Li-dpPINN or Chen-dpPINN become more consistent with the experimental results;
- The distribution range of training points, as opposed to their quantity, is more crucial to the dpPINN model accuracy. The decentralized layout of training points is beneficial for the training of dpPINN;
- In regard to the flow simulation around a building, the PINN models (including dpPINN) show stronger generalization and forecasting capability than pure physics-based methods when few labelled training data are available. They are also less data-demanding than pure data-driven methods since only sparse data in the near-wall regions are required.

Acknowledgement

The work described in this paper was supported by a grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region (SAR), China (grant number PolyU 152308/22E) and a grant from The Hong Kong Polytechnic University (grant number 1-WZ0C). The authors also appreciate the funding support by the Innovation and Technology Commission of Hong Kong SAR Government to the Hong Kong Branch of National Engineering Research Center on Rail Transit Electrification and Automation (grant number K-BBY1), the National Natural Science Foundation of China (Grant No. 52202426), Start-up Fund for RAPs under the Strategic Hiring Scheme of The Hong Kong Polytechnic University (Grant No. 1-BD23), and grants from Wuyi University Hong Kong-Macao Joint R&D Fund (Grants No. 2019WGALH15, 2019WGALH17, and 2021WGALH15).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

- Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18, 1–43. <https://www.jmlr.org/papers/volume18/17-468/17-468.pdf>
- Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., & Kaushik, S. (2019). Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2), 525–545. <https://doi.org/10.1007/s00466-019-01740-0>
- Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2022). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12), 1727–1738. <https://doi.org/10.1007/s10409-021-01148-1>
- Chen, Q., & Xu, W. (1998). A zero-equation turbulence model for indoor airflow simulation. *Energy and Buildings*, 28(2), 137–144. [https://doi.org/10.1016/S0378-7788\(98\)00020-6](https://doi.org/10.1016/S0378-7788(98)00020-6)
- Chen, Z., Liu, T., Li, M., Yu, M., Lu, Z., & Liu, D. (2019). Dynamic response of railway vehicles under unsteady aerodynamic forces caused by local landforms. *Wind and Structures*, 29(3), 149–161. <https://doi.org/10.12989/was.2019.29.3.149>
- Chen, Z., Liu, T., Li, W., Guo, Z., & Xia, Y. (2021). Aerodynamic performance and dynamic behaviors of a train passing through an elongated hillock region beside a wind-break under crosswinds and corresponding flow mitigation measures. *Journal of Wind Engineering and Industrial Aerodynamics*, 208, 104434. <https://doi.org/10.1016/j.jweia.2020.104434>
- Chen, Z. W., Ni, Y. Q., Wang, Y. W., Wang, S. M., & Liu, T. (2022). Mitigating crosswind effect on high-speed trains by active blowing method: A comparative study. *Engineering Applications of Computational Fluid Mechanics*, 16(1), 1064–1081. <https://doi.org/10.1080/19942060.2022.2064921>
- Choi, S., Jung, I., Kim, H., Na, J., & Lee, J. M. (2022). Physics-informed deep learning for data-driven solutions of computational fluid dynamics. *Korean Journal of Chemical Engineering*, 39(3), 515–528. <https://doi.org/10.1007/s11814-021-0979-x>
- Eivazi, H., Tahani, M., Schlatter, P., & Vinuesa, R. (2022). Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. *Physics of Fluids*, 34(7), 065129. <https://doi.org/10.1063/5.0095270>
- Eivazi, H., & Vinuesa, R. (2022). Physics-informed deep-learning applications to experimental fluid mechanics. *arXiv preprint arXiv:2203.15402*.
- Guo, M., Haque, A., Huang, D., Yeung, S., & Li, F. (2018). *Dynamic task prioritization for multitask learning*. 15th European Conference on Computer Vision (ECCV 2018), Munich, Germany, September 8–14.
- Guo, Z., Liu, T., Chen, Z., Xia, Y., Li, W., & Li, L. (2020). Aerodynamic influences of bogie's geometric complexity on high-speed trains under crosswind. *Journal of Wind Engineering and Industrial Aerodynamics*, 196, 104053. <https://doi.org/10.1016/j.jweia.2019.104053>
- He, K., Minelli, G., Su, X., Gao, G., & Krajnović, S. (2021). Influence of the rounded rear edge on wake bi-stability of

- a notchback bluff body. *Physics of Fluids*, 33(11), 115107. <https://doi.org/10.1063/5.0071925>
- Kataoka, H., Ono, Y., & Enoki, K. (2020). Applications and prospects of CFD for wind engineering fields. *Journal of Wind Engineering and Industrial Aerodynamics*, 205, 104310. <https://doi.org/10.1016/j.jweia.2020.104310>
- Kendall, A., Gal, Y., & Cipolla, R. (2018). *Multi-task learning using uncertainty to weigh losses for scene geometry and semantics*. Ieee Conference on Computer Vision and Pattern Recognition (CVPR 2018), Salt Lake City, Utah, USA, June 18–22.
- Li, C., Li, X., Su, Y., & Zhu, Y. (2013). Zero-equation turbulence model for outdoor airflow simulation. *Journal of Tsinghua University (Science and Technology)*, 53, 589–594. <http://jst.tsinghuajournals.com/CN/Y2013/V53/I5/589>
- Lin, T., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal loss for dense object detection*. Ieee International Conference on Computer Vision (ICCV 2017), Venice, Italy, October 22–29.
- Liu, T., Chen, Z., Guo, Z., & Krajnović, S. (2020). Reasonable pressure tap layout to measure the aerodynamic forces of a train at different yaw angles. *Measurement*, 166, 108255. <https://doi.org/10.1016/j.measurement.2020.108255>
- Luo, S., Vellakal, M., Koric, S., Kindratenko, V., & Cui, J. (2020). *Parameter identification of RANS turbulence model using physics-embedded neural network*. International Conference on High Performance Computing, Frankfurt, Germany, June 21–25.
- Meng, X., Li, Z., Zhang, D., & Karniadakis, G. E. (2020). PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering*, 370, 113250. <https://doi.org/10.1016/j.cma.2020.113250>
- Meng, Y., & Hibi, K. (1998). Turbulent measurements of the flow field around a high-rise building. *Wind Engineers, JAWE*, 1998(76), 55–64. https://doi.org/10.5359/jawe.1998.76_55
- Milano, M., & Koumoutsakos, P. (2002). Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1), 1–26. <https://doi.org/10.1006/jcph.2002.7146>
- Pache, R., & Rung, T. (2022). Data-driven surrogate modeling of aerodynamic forces on the superstructure of container vessels. *Engineering Applications of Computational Fluid Mechanics*, 16(1), 746–763. <https://doi.org/10.1080/19942060.2022.2044383>
- Prandtl, L. (1925). 7. Bericht über Untersuchungen zur ausgebildeten Turbulenz. *Journal of Applied Mathematics and Mechanics*, 5(2), 136–139. <https://doi.org/10.1002/zamm.19250050212>
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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. <https://doi.org/10.1016/j.jcp.2018.10.045>
- Rao, C., Sun, H., & Liu, Y. (2020). Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3), 207–212. <https://doi.org/10.1016/j.taml.2020.01.039>
- Riel, B., Minchew, B., & Bischoff, T. (2021). Data-driven inference of the mechanics of slip along glacier beds using physics-informed neural networks: Case study on rutford Ice stream, antarctica. *Journal of Advances in Modeling Earth Systems*, 13(11), e2021MS002621. <https://doi.org/10.1029/2021MS002621>
- Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., & Kuhl, E. (2020). Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8, 42. <https://doi.org/10.3389/fphy.2020.00042>
- Song, J.-L., Li, J.-W., Xu, R.-Z., & Flay, R. G. (2022). Field measurements and CFD simulations of wind characteristics at the Yellow River bridge site in a converging-channel terrain. *Engineering Applications of Computational Fluid Mechanics*, 16(1), 58–72. <https://doi.org/10.1080/19942060.2021.2004240>
- Sun, Y., Sun, Q., & Qin, K. (2021). Physics-based deep learning for flow problems. *Energies*, 14(22), 7760. <https://doi.org/10.3390/en14227760>
- Wessels, H., Weißenfels, C., & Wriggers, P. (2020). The neural particle method – An updated Lagrangian physics informed neural network for computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 368, 113127. <https://doi.org/10.1016/j.cma.2020.113127>
- Xiang, Z., Peng, W., Liu, X., & Yao, W. (2022). Self-adaptive loss balanced Physics-informed neural networks. *Neurocomputing*, 496, 11–34. <https://doi.org/10.1016/j.neucom.2022.05.015>
- Yuan, L., Ni, Y., Deng, X., & Hao, S. (2022). A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462, 111260. <https://doi.org/10.1016/j.jcp.2022.111260>
- Zhang, J., Wang, F., Guo, Z., Han, S., Gao, G., & Wang, J. (2022). Investigation of the wake flow of a simplified heavy vehicle with different aspect ratios. *Physics of Fluids*, 34(6), 065135. doi:10.1063/5.0094534