DEVELOPMENT OF AMELIORATIVE METHODS FOR REYNOLDS-AVERAGED NAVIER-STOKES SIMULATIONS BASED ON PHYSICS-INFORMED MACHINE LEARNING

RUI ENZE

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Civil and Environmental Engineering

Development of Ameliorative Methods for Reynolds-Averaged Navier-Stokes Simulations Based on Physics-Informed Machine Learning

Rui Enze

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

December 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

RUI Enze (Name of student)

Dedicated to my family

for their love and support

ABSTRACT

Reynolds-averaged Navier-Stokes (RANS) simulation is a widely employed numerical approach for turbulence modeling. The computational fluid dynamics (CFD) method has long been the predominant approach in RANS simulation, owing to its intuitiveness, ease of implementation, and high accuracy in modeling. Some well-known CFD methods such as the finite element method and finite volume method have been adopted in RANS simulation in past decades. However, as research progresses, the shortcomings of the CFD-based RANS simulations are gradually becoming apparent. Several consensus problems are enumerated here. For instance, the CFD methods require manual meshing, which may induce various kinds of grid generation issues. In addition, the RANS equations incorporate the Reynolds stress terms as additional unknown variables during the averaging process. The Reynolds stress modeling resulting from the action of Reynolds averaging contributes to the non-universality of RANS simulations.

In recent years, a novel machine learning-based solver for partial differential equations (PDEs), i.e., physics-informed neural network (PINN), has emerged. Since its inception, it has shown a considerable impact in the field of fluid mechanics. Researchers have also been identifying the potential of PINN's applications in RANS simulations. The advantages of this method over CFD methods are evident. Firstly, being a meshless approach, it does not encounter any grid-related issues. For instance, the partial derivative terms in the PDEs are calculated using the automatic differentiation function in a PINN, eliminating any truncation error that may arise from grid methods. In addition, PINN has a neural network foundation,

while it can integrate data information into its simulation, and thus derive physics-based data-driven solutions.

Nevertheless, it must be acknowledged that while PINN possesses certain advantages over conventional CFD methods when solving the RANS equations, this emerging machine learning solver for PDEs still faces several challenges. First and foremost, the convergence performance of a PDE solver is a crucial indication for assessing its effectiveness in solving equations. However, studies have shown that minimizing the loss of a PINN may sometimes be challenging using these gradient-based methods. Secondly, there is the issue of the limitation of the nonlinear expression and feature learning capabilities for PINN-based RANS simulations. Learning high-frequency features using neural networks has been found to be difficult in previous research. Furthermore, although grid issues do not plague PINNs, the limited applicability of the RANS turbulence models under various flow conditions still exists. One of the most difficult issues to solve is still the search for a universal turbulence simulation approach in PINN-based RANS simulations.

The entire thesis is divided into seven chapters, with the first chapter being an introduction to the entire thesis and the second chapter being a review of existing methods. Based on the discussions in Chapters 1 and 2, this thesis proposes four amelioration measures grounded in the PINN framework, namely Dynamic Prioritization in Chapter 3, Multifidelity Modeling in Chapter 4, Quantum Layer Integration in Chapter 5, and Weighted Sum Turbulence Model in Chapter 6. These amelioration measures are dedicated to alleviating the aforementioned key issues in PINN-based RANS simulations. Some are used to improve the convergence performance of a PINN, some can accelerate the learning ability of a PINN for extracting high-frequency features, and some can alleviate the issue of poor applicability of RANS turbulence models. These proposed methods have been validated by using experiments and achieved

satisfactory results, promoting the further developments of physics-informed machine learning methods in turbulence modeling. At the end of this research, the future work is prospected, and the future research direction is pointed out.

ACKNOWLEDGEMENTS

In the first place, I would like to express my gratitude to Prof. Yi-Qing Ni, who serves as my chief supervisor during my PhD study at the Hong Kong Polytechnic University. Prof. Ni helped me a lot through my step-by-step transition process from an undergraduate graduate who had no prior knowledge of scientific research to an experienced researcher. Prof. Ni has been fully engaged in the area for decades, possessing a thorough and one-of-a-kind grasp of scientific research. Prof. Ni never loses touch with the cutting edge of the research, despite the fact that he holds a prominent position among peers and is always thrown into a packed schedule. I hold his personality and his approach to work in the highest regard. He shows me that nothing is impossible to a willing heart.

I would also like to take this opportunity to express my appreciation to my PhD co-supervisor, Dr. Zheng-Wei Chen, for his meticulous guidance in every stage of my research study, from guiding me in formula manipulation to teaching me how to prepare figures in research papers. The completion of this thesis would not have been possible for me without his careful guidance.

I want to especially express my appreciation to Dr. Xiang-Yun Deng and Ms. Wendy So for their assistance in both my personal life and academic pursuits at the beginning of my five-year stay in Hong Kong.

I would like to offer my sincere gratitude to the National Rail Transit Electrification and Automation Engineering Technology Research Center (Hong Kong Branch). I sincerely thank Mr. Wing-Hong Kwan, Mr. Tai-Tung Wai, Ms. Freda Chow, Mr. Chen-Xing Zhang, Ms. Lin-

Lin Cai, Ms. Josephine Lui, Mr. Han-Zhan Lu, and Ms. Karina Leung for their help during my work time. I would also like to express my thanks to the administrative staff in the CEE department for their assistance in the past four years of my PhD study.

I want to express my thanks to Dr. You-Wu Wang, Dr. Su-Mei Wang, Dr. Wai-Kei Ao, Dr. Hong-Wei Li, Dr. Jason Lin, Dr. Wen-Qiang Liu, Dr. Duo Zhang, Dr. Zi-Jian Guo, Dr. Xiao-Ming Tan, Dr. Yun-Fan Yang, Dr. Xiu-Yu Chen, Dr. Wen-Bo Hu, Dr. Yuan-Hao Wei, Dr. Yun-Ke Luo, Dr. Si-Yi Chen, Dr. Xin Ye, Mr. Chao Zhang, Mr. Yang Lu, Mr. Guang Zhou, Mr. Gao-Feng Jiang, Mr. Shuo Hao, Mr. Zhen Lin, Mr. Lei Yuan, Mr. You-Liang Zheng, Mr. Yu-Ling Wang, Mr. Qi-Fan Zhou, Mr. Xiang-Xiong Li, Mr. Wei-Jia Zhang, Ms. Qi Zhu, Mr. Da-Zhi Dang, Mr. Yan-Ke Tan, Mr. Yu-Xuan Liang, Mr. Guang-Zhi Zeng, Mr. Zhan-Hao Guo, Mr. Zheng-Xin Che, Mr. Sheng-Yuan Liu, Mr. Zhen-Bin Zhou, Mr. Jia-Hao Lu, Ms. Xin-Yue Xu, Mr. Yuan-Jiang Zeng, Mr. Qing-Chen Tang, Mr. Liang Lyu, Mr. Gang Zeng, Mr. Cheng Peng, Mr. Yu-Hang Lu, Ms. Xin-Ge Zhao, Ms. Shu Li, Ms. Xuan Zhao, Mr. Zi-Rui Zuo, and other members in our research group for their support and friendship. Sincerely wish everyone a bright future!

I would like to express my special thanks once again to Dr. Xin Ye, Mr. Lei Yuan, Mr. Wei-Jia Zhang, Mr. Guang-Zhi Zeng, and Mr. Zhan-Hao Guo for their academic advising that has greatly benefited me in my PhD study.

I would like to express my deepest gratitude to Ms. Yi-Min Zheng for being the one who offered me the most profound spiritual support when I was working on preparing this thesis for graduation. The inspiration to keep pushing forward and the will to keep working hard were both gifts that she bestowed upon me. I am indebted to her for her unwavering support and concern throughout the entire process.

Lastly and most importantly, I would like to express my gratitude to my parents and families for their long-term endless support, care, and raising. I appreciate all the little things they do for me from my birth to now. Wish them all the best in their work and life!

LIST OF PUBLICATIONS

Journal Papers:

Chen, Z. W., **Rui, E. Z.**, Liu, T. H. *, Ni, Y. Q., Huo, X. S., Xia, Y. T., Li, H. W., Guo, Z. J. & Zhou, L. (2022). Unsteady aerodynamic characteristics of a high-speed train induced by the sudden change of windbreak wall structure: A case study of the Xinjiang railway. *Applied Sciences*, **12**, 7217.

Liu, W. Q. †, **Rui, E. Z.** †, Yuan, L., Chen, S. Y., Zheng, Y. L. & Ni, Y. Q. * (2023). A novel computer vision-based vibration measurement and coarse-to-fine damage assessment method for truss bridges. *Smart Structures and Systems*, **31**, 393-407.

Rui, E. Z., Chen, Z. W., Ni, Y. Q. *, Yuan, L. & Zeng, G. Z. (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**, 2238849.

Rui, E. Z., Zeng, G. Z., Ni, Y. Q. *, Chen, Z. W. & Hao, S. (2024). Time-averaged flow field reconstruction based on a multifidelity model using physics-informed neural network (PINN) and nonlinear information fusion. *International Journal of Numerical Methods for Heat & Fluid Flow*, **34**, 131-149.

Li, H. W., Ni, Y. Q. *, Wang, Y. W., Chen, Z. W., **Rui, E. Z.** & Xu, Z. D. (2024). Modeling of forced-vibration systems using continuous-time state-space neural network. *Engineering Structures*, 302, 117329.

Zeng, G. Z., Chen, Z. W. *, Ni, Y. Q. & **Rui, E. Z.** (2024). Investigating embedded data distribution strategy on reconstruction accuracy of flow field around the crosswind-affected train based on physics-informed neural networks. *International Journal of Numerical Methods for Heat & Fluid Flow*, **34**, 2963-2985.

Rui, E. Z., Chen, Z. W. *, & Ni, Y. Q. (2024). Integrating a quantum layer into a physics-informed neural network for solving forward and inverse problems involving differential equations. *Neurocomputing*, under review.

Conference Papers:

Rui, E. Z., Chen, Z. W., Ni, Y. Q. * & Yuan, L. Full domain flow information recognition around buildings with sparse near-wall data through a physics-informed data-driven approach. *Proceedings of the 8th World Conference on Structural Control and Monitoring*, 5-8 Jun 2022, Orlando, Florida, United States of America.

Li, H. W. *, Ni, Y. Q., Wang, Y. W., Chen, Z. W. & **Rui, E. Z.** Continuous-time state-space neural network and its application in modeling of forced-vibration systems. *Structural Health Monitoring 2023: Proceedings of the 14th International Workshop on Structural Health Monitoring*, 12-14 Sep 2023, Stanford, United States.

Li, H. W. *, Ni, Y. Q., Wang, Y. W., Chen, Z. W., **Rui, E. Z.** & Xu, Z. D. State-integration neural network for modeling of forced-vibration systems. *Proceedings of the 29th International Conference on Computational & Experimental Engineering and Sciences*, 26-29 May 2023, Shenzhen, China.

Yuan, L. [*], Ni, Y. Q., **Rui, E. Z.** & Zhang, W. J. Structural damage inverse detection from noisy vibration measurement with physics-informed neural networks. *Proceedings of the XII International Conference on Structural Dynamics*, 3-5 Jul 2023, Delft, Netherlands.

Rui, E. Z. *, Chen, Z. W., Li, H. W. & Ni, Y. Q. Integration of a weak-form RANS turbulence model into PINN-based fluid simulations. *Proceedings of the 10th Asia Conference on Mechanical Engineering and Aerospace Engineering*, 18-20 Oct 2024, Taicang, China.

Note: [*] Corresponding author; [†] Equal contribution.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

The main symbols used in this thesis are listed below (duplicate symbols are excluded).

Chapter 2	**Description**
x_i	Spatial coordinate
u_i	Velocity component in the x_i direction
t	Time
ρ	Density
p	Pressure
ν	Kinematic viscosity
f_i	External force in the x_i direction
$\emptyset$	Physical quantity
φ	Physical quantity
T	Length of the time range
μ	Dynamic viscosity

R_{ij}	Reynolds stress term
μ_t	Turbulent viscosity
δ_{ij}	Kronecker delta function
k	Turbulent kinetic energy
μ_{eff}	Effective viscosity
$C_{p\mu}$	Problem-dependent constant
l_{p0}	Length scale
u_t	Characteristic fluid velocity
l_{p1}	Turbulent length scale
C_D	Constant
σ_k	Constant
ε	Turbulent dissipation rate
S_{ij}	Mean strain rate tensor
C_μ	Constant
$C_{1\varepsilon}$	Constant

$C_{2\varepsilon}$	Constant
σ_{ε}	Constant
η_0	Constant
β	Constant
ψ	Physical quantity of diffusion
K	Diffusion coefficient
Q	Rate of heat generation per unit volume
Q_c	Constant component
a_E	Intermediate variable
a_W	Intermediate variable
a_P	Intermediate variable
b	Intermediate variable
δx	Length of the control volume
f	Scalar function of the independent variables
R	Residual value
W_i	Weight function

D	Computational domain
m_D	Number of elements
$\boldsymbol{N}$	Shape function
$\boldsymbol{\psi^e}$	Unknowns to be solved on nodes
R_c	Residual of the continuity equation
R_m	Residual of the momentum equation

Chapter 3	**Description**
L_f	Governing equation loss
L_b	Boundary condition loss
L_d	Data loss
w_f	Weight coefficient for L_f
w_b	Weight coefficient for L_b
w_d	Weight coefficient for L_d
f_i^n	Residual of the governing equation

r_{nb}^{i}	Residual of the Neumann boundary
r_{db}^{i}	Residual of the Dirichlet boundary
r_{d}^{i}	Residual of the data constraints
N_f	Number of the collocation points for calculating f_i^n
N_{nb}	Number of the collocation points for calculating r_{nb}^{i}
N_{db}	Number of the collocation points for calculating r_{db}^{i}
N_d	Number of the collocation points for calculating r_{d}^{i}
L_u	Loss component related to u
L_v	Loss component related to v
L_w	Loss component related to w
L_p	Loss component related to p
w_u	Weight coefficient for L_u
w_v	Weight coefficient for L_v
w_w	Weight coefficient for L_w
w_p	Weight coefficient for L_p

$\mathbf{U}_r$	Set of the equation numbers related to u
$\mathbf{V}_r$	Set of the equation numbers related to v
$\mathbf{W}_r$	Set of the equation numbers related to w
$\mathbf{P}_r$	Set of the equation numbers related to p
r_n^i	Residual of Eq. (3-n)
N_n	Number of the collocation points for calculating r_n^i
γ	Hyperparameter in dpPINN
$\boldsymbol{U}_{\boldsymbol{T}_i}$	Vector of the labeled training data
$\widetilde{\boldsymbol{U}}_{\boldsymbol{T}_i}$	Vector of PINN predictions on the training points
k_i	Intermediate variable
μ_{in}	Boundary layer's turbulent viscosity
μ_{out}	Outer layer's turbulent viscosity
C_b	Building width
H_b	Building height
I_G	Constant

z_G	Constant
α	Constant
$\boldsymbol{U_i}$	Vector of the reference data
$\widetilde{\boldsymbol{U}}_{\boldsymbol{i}}$	Vector of the dpPINN predictions

Chapter 4	**Description**
f_h	High-fidelity model
f_l	Low-fidelity model
θ_h	Hyperparameter
k_h	Kernel function
k_{h_ρ}	Kernel function
k_{h_f}	Kernel function
k_{h_δ}	Kernel function
θ_{h_ρ}	Hyperparameter
θ_{h_f}	Hyperparameter

θ_{h_δ}	Hyperparameter
$\boldsymbol{K}$	Kernel function
$\boldsymbol{y}$	Training target
n_d	Dimension of the input space

Chapter 5	**Description**
X	NOT gate
H	Hardamard gate
ψ_0	Initial state of a qubit
V	Unitary operator
U	Parameterized VQC
θ	Trainable parameter of a VQC
Y	The observable
$\boldsymbol{\theta}$	Rotation angle vector
σ_y	Pauli-Y operator

L_i	Initial condition loss
w_i	Weight coefficient for L_i
n	Repetition number
E	Young's modulus
A	Cross-sectional area
p_b	Body force

Chapter 6	**Description**
C_s	Empirical constant
V_s	Turbulent velocity scale
l_s	Turbulent length scale
G	Mean strain rate
l_m	Mixing length
l	Nearest distance from the wall
l_{max}	Maximum value of l
C_c	Constant

n_c	Number of the candidate turbulence models
r_i	Weight coefficient for the candidate model
N_R	Number of training points
r_{Rxx}^i	Residual of the Reynolds stress component
r_{Rxy}^i	Residual of the Reynolds stress component
r_{Ryy}^i	Residual of the Reynolds stress component
L_R	Reynolds stress loss

LIST OF ABBREVIATIONS

NS	Navier-Stokes
PDE	Partial differential equation
RANS	Reynolds-averaged Navier-Stokes
CFD	Computational fluid dynamics
DNS	Direct numerical simulation
PINN	Physics-informed neural network
AD	Automatic differentiation
FCNN	Fully connected neural network
RSM	Reynolds stress model
RNG	Re-Normalization Group
FVM	Finite volume method
FEM	Finite element method
DGM	Deep Galerkin method

PIV	Particle image velocimetry
LDA	Laser Doppler anemometry
SGD	Stochastic gradient descent
Adam	Adaptive moment estimation
Tanh	Hyperbolic tangent function
WSS	Wall shear stress
LES	Large eddy simulation
DES	Detached-eddy simulation
dpPINN	Dynamic prioritization PINN
GP	Gaussian Process
NIF	Nonlinear information fusion
NLML	Negative log marginal likelihood
VQC	Variational quantum circuit
NISQ	Noisy intermediate-scale quantum
QAOA	Quantum approximate optimization algorithm

PIQC	Physics-informed quantum circuit
ODE	Ordinary differential equation
PIHCQNN	Physics-informed hybrid classical-quantum neural network
ENN	Embedding neural network
CNOT	Controlled NOT
RoNN	Readout neural network
MSE	Mean-squared error

CHAPTER 1

INTRODUCTION

1.1 Background

The Navier-Stokes (NS) equations, as a set of partial differential equations (PDEs), are commonly used to describe the movements of viscous fluid substances, which accurately depict the momentum and mass conservations in the motions of Newtonian fluids. However, only in some laminar flow cases with low Reynolds numbers can we see the presence of the NS equations as the fluid governing equations (Wissink *et al.* 2003, Lu *et al.* 2006). In contrast to the laminar flow, the turbulent flow exhibits a more disorderly pattern of velocity and pressure distributions in both spatial and temporal dimensions. The NS equations are proficient in laminar modeling, but they are not suitable for simulating turbulence due to the challenges posed by the computational resources required for turbulence modeling (Mikulevicius and Rozovskii 2004, Wong 2020). Naturally, various kinds of simplified forms for NS equations have been put forward to solve turbulence modeling (Boris *et al.* 1992, Iaccarino *et al.* 2003, Mikulevicius and Rozovskii 2004). For instance, the Reynolds-averaged Navier-Stokes (RANS) equations are one of its variants, which is the time-averaged form of the NS equations. The fluid properties within the flow field are simplified into time-averaged variables, which constitute the parameters to be solved in the RANS equations, while the turbulence effect is manifested in the RANS turbulence models (Mikulevicius and Rozovskii 2004). Such a simplified form of the NS equations provides an alternative approach for turbulent flow

simulation at the theoretical level. Large Eddy Simulation (LES) is another high-precision numerical simulation method for turbulence investigation. The core idea of LES is to directly analyze large-scale turbulent structures while modelling small-scale eddies using subgrid scale models. Compared to RANS, large eddy simulation (LES) can capture unsteady turbulent characteristics such as the separated flows and vortex evolution, making it suitable for problems sensitive to transient turbulent characteristics such as the aerodynamic noise. Considering that this thesis mainly focuses on fast and high-precision prediction of industrial-scale problems, and RANS simulation should be a better carrier. This is because by filtering out turbulent fluctuations through time averaging, only the averaged fluid equation is solved in RANS simulations. All turbulence scales are enclosed by models, without the need to directly analyze transient vortex structures.

Computational fluid dynamics (CFD) methods have long been used to solve the RANS equations. Some well-known CFD methods such as the finite element method, finite volume method, and discrete element method have been adopted in flow simulation in the past decade (Fasel 1976, Jones and Titi 1992, Tobiska and Verfürth 1996). The finite volume method is one of the most mature and widely adopted approaches, which discretizes the computational domain into a set of non-overlapping control volumes and then involves the derivation of discrete equations in each individual control volume (Eymard *et al.* 2000). The finite volume method has been widely adopted by commercial software like *Ansys Fluent* and *OpenFOAM* due to its excellent ability to formulate unstructured meshes and handle complicated geometries while solving PDEs such as the RANS equations.

Since the flow field has been averaged and there is no need to analyze vortices at all scales, the grid resolution required for solving the RANS equations when using CFD methods can be far less precise compared to directly solving the NS equation (also known as Direct Numerical

Simulation, i.e., DNS). This significantly reduces the computer resources needed. The CFD simulation, which relies on the RANS equations, offers the benefits of reduced computational complexity and superior time efficiency. Despite the loss of turbulent characteristics due to averaging in the RANS equations, there are innumerable cases where the analysis of time-averaged properties of the flow should be prioritized, such as the vehicle wind resistance problem (Ashton *et al.* 2016). As a result, CFD-based RANS simulation has become one of the methods commonly adopted for fluid simulation in engineering problems (Coroneo *et al.* 2011, Hertwig *et al.* 2012). A large number of research studies have emerged to solve practical engineering problems involving turbulence in the past period of time based on CFD-based RANS simulations (Peltier and Hambric 2007, Tominaga and Stathopoulos 2011). For example, train aerodynamics has become a research focus of great significance in the modern high-speed railway system. Massive train-air models have been established to study the flow field around the high-speed train and further reduce the wind resistance during its daily operation (Chen *et al.* 2017, Deng *et al.* 2019, Chen *et al.* 2020, Deng *et al.* 2020, Chen *et al.* 2022).

As research progresses, the shortcomings of the CFD-based RANS simulations are gradually becoming apparent. Firstly, the CFD methods require manual meshing, while the results are greatly affected by grid resolution (Katz and Sankaran 2011). Improper grids may hinder the analysis of fine flow details, leading to imprecise results. Secondly, discretization in CFD simulations brings new problems such as numerical diffusion and dissipation, which may have a significant impact on the simulation results (Ekaterinaris 2005). Thirdly, compared to the NS equations, the RANS equations incorporate the Reynolds stress terms as additional unknown variables during the averaging process. To fully define the RANS equations, the Reynolds stress terms should be incorporated by the time-averaged flow characteristics, which is known as RANS turbulence modeling. A number of turbulence models have been proposed

in recent years. However, these are semi-empirical models, and there is no consensus on which specific turbulence model should be selected in different flow conditions (Zhang *et al.* 2007). As a result, the accuracy of the RANS simulation largely depends on the applicability of the selected turbulence model (Farhadi *et al.* 2018).

In recent years, a novel machine learning-based PDE solver, i.e., physics-informed neural network (PINN), has emerged (Raissi *et al.* 2019). Since its inception, it has shown considerable impact in the field of fluid mechanics (Choi *et al.* 2022). The advantages of this method over CFD methods are evident. Firstly, being a meshless approach, it does not encounter any grid-related issues. For instance, the partial derivative terms in the PDEs are calculated using the automatic differentiation (AD) function in a PINN, eliminating any truncation error that may arise from grid methods. In addition, PINN has a neural network foundation, while it can integrate data information into its simulation, and thus derive physics-based data-driven solutions (de la Mata *et al.* 2023). Finally, with sufficient data, PINN is also a powerful tool for solving inverse problems in fluid dynamics, such as parameter identification issues and the discovery of underlying physical relationships of fluid parameters (Luo *et al.* 2020, Ngo and Lim 2021, Guo and Fang 2023, Yang *et al.* 2023).

In the PINN framework, the problem of the NS equation's incapacity to explicitly model turbulent flows with high Reynolds numbers persists (Patel *et al.* 2024). Consequently, researchers have been identifying the potential of PINN’s applications in RANS simulations (Hanrahan *et al.* 2023, Pioch *et al.* 2023). To fully define the system of PDEs, RANS turbulence modeling is still indispensable in the PINN framework. In the PINN framework, the approach of turbulence modeling, i.e., introducing extra PDEs to solve Reynolds stress terms, is still applicable, as it is in CFD methods. However, it should be noted that, compared to CFD, PINNs have the ability to integrate data for turbulence simulation. Therefore, with sufficient data

support, the turbulence modeling task may be substituted with the embedded training data, where the Reynolds stress terms are directly incorporated into the computational domain, eliminating the need for collateral modeling (Eivazi *et al.* 2022). In addition, PINN has shown exceptional performance not just in solving forward problems but also in addressing inverse problems involving the RANS equations (Luo *et al.* 2020). The RANS turbulence models are filled with a multitude of empirical constants that lack universal applicability to all flow conditions (Geng and Escaler 2020). Deriving empirical constants from data under different flow conditions can also contribute to the further development of the RANS turbulence models. Specifically, PINN completes this task by converting empirical constants into neural network parameters to be optimized (Raissi *et al.* 2019).

Nevertheless, it must be acknowledged that while PINN possesses certain advantages over conventional CFD methods when solving the RANS equations, this emerging machine learning solver for PDEs still faces several challenges. Some are chronic illnesses even in the CFD framework, while others are newly introduced. These challenges compel researchers to delve deeper into this research topic, thereby laying the foundation for this thesis.

1.2 Research Motivation

While researchers have successfully utilized PINN as a machine learning solver for the RANS equations in flow field simulation tasks and have gotten favorable outcomes, there are still some unresolved issues that need to be addressed in PINN-RANS simulations. To begin with, let us review the limitations of using CFD methods to solve the RANS equations, which were previously described. Although grid issues do not exist in PINNs, the limited applicability of the RANS turbulence models under various flow conditions still exists (Pioch *et al.* 2023,

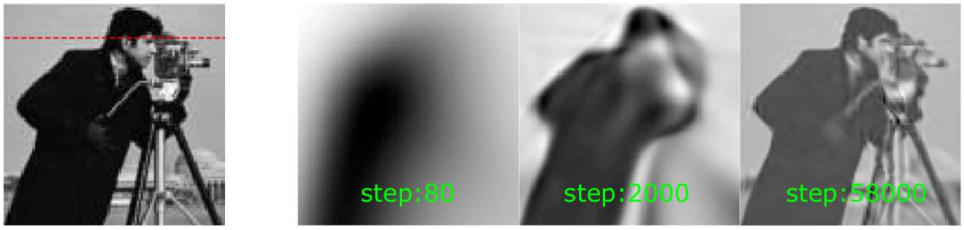
Rui *et al.* 2023). The problem is not induced by the PDE solver, but rather an inevitable result of the simplification process from the NS equations to the RANS equations. The RANS equations transform the unsteady turbulence problem into a steady-state problem, and the information inherent in the time-accurate NS equations will be lost. In addition, it is well-known that the RANS turbulence model should be introduced in order to close the system of PDEs. RANS turbulence models can be categorized into various groups depending on the number of extra PDEs incorporated in turbulence modeling. These groups include zero-equation models, one-equation models, two-equation models, and so on (Yusuf *et al.* 2020). Some of these models are suited for simulating indoor airflows (Chen and Xu 1998). Some are for outdoor flow fields around building structures (Li *et al.* 2012), while others are for free shear flows (Lakshminarayana 1986, Yakhot *et al.* 1992). Different RANS turbulence models describe the Reynolds stress terms in totally different ways due to the close relationships between these terms and flow boundaries, making it challenging to find a universally applicable model under various flow conditions. Nevertheless, it is important to note that the PINN has the capability to mitigate this issue by embedding pre-existing data in their training to aid in the flow simulations (Eivazi *et al.* 2022, Hanrahan *et al.* 2023). The capacity of machine learning algorithms to generalize from training data is a distinctive feature in the age of big data, and this ability is inherently passed down to the PINN (Li and Chitre 2023). Continuously correcting the solution of PDEs using measurement data during the equation solving process is a solution to overcome the limited universality of such physical models (de la Mata *et al.* 2023).

Furthermore, PINN, being a PDE solver, has some fundamental limitations that restrict its potential applications in RANS simulations. First and foremost, the problem at hand is convergence. The convergence performance of a PDE solver is a crucial indication for assessing its effectiveness in solving equations (Pantidis *et al.* 2023). The governing equation

loss, the boundary condition loss, and the labeled data loss which is likely to be present, make up the loss function of a PINN while solving the RANS equation. Out of these, the governing equation loss and the labeled data loss usually contain around three to four loss components apiece. However, the number of loss components in the boundary condition loss alone usually reaches two digits, depending on whether it is a two-dimensional or three-dimensional problem. Therefore, the total loss of a PINN in solving the RANS equations is generally accumulated by more than ten to twenty independent loss components. This may lead to several issues, such as the imbalance between various loss components, namely, the challenge of determining the weight coefficients for each component in the total loss (Li and Feng 2022, Liu *et al.* 2022, Heldmann *et al.* 2023, Hou *et al.* 2024). Should the values of these weight coefficients remain consistent throughout the training process? If their values stay constant, how should they be taken? If the values do not remain constant, how should they be altered throughout the training process? The answers to these questions are currently undisclosed, and these also constitute the current focus of many studies. Moreover, as is well known, the training of a neural network is generally carried out using gradient descent methods. However, studies have shown that minimizing the loss of a PINN may sometimes be challenging using these gradient-based methods (Krishnapriyan *et al.* 2021, Wang *et al.* 2021, Rathore *et al.* 2024). These studies generally assume that the presence of the differential operators in a PINN will result in an ill-conditioned loss function.

Secondly, there is the issue of the limitation of the nonlinear expression and feature learning capabilities of a PINN. It is widely acknowledged that in the vast majority of current research, the main body of a PINN, namely the function fitting module, is a fully connected neural network (FCNN). Nevertheless, FCNNs have some established deficiencies in the function fitting procedure such as the frequency principle observed in its training process. More

specifically, the frequency principle refers to the common phenomenon that an FCNN tends to learn low-frequency information in the signal first, and then slowly learn high-frequency information when fed with training data to comprehend the distribution of the signal (Xu *et al.* 2020). This phenomenon is detrimental to the utilization of a PINN for solving the RANS equations (Sallam and Fürth 2023, Ye *et al.* 2024, Zhang *et al.* 2024). Although the solution of the RANS equations does not contain a time term and traditional high-frequency and low-frequency components, similar to image reconstruction tasks (for details, see Figure 1-1), the PINN prioritizes providing a crude drawing of the time-averaged solution when solving the RANS equations, which hinders the acquisition of the accurate solutions for areas with substantial variations in gradients.



(a) True image (b) DNN output

Figure 1-1. The frequency principle is observed in a two-dimensional image reconstruction task. Reprinted from (Xu et al. 2020).

To summarize, there are challenges reflected in three aspects when using the PINN to solve the RANS equations. Firstly, the inclusion of the Reynolds stress terms in the fluid governing equations is necessary due to the trade-off made in simplifying the NS equations. Nevertheless, the numerical modeling for the Reynolds stress terms would unavoidably give rise to the applicability issue between various turbulence models, leading to disparities between the simulation results and the real flow fields. The second issue is the convergence difficulty when training a PINN. This involves the loss balance issue and the ill-conditioned loss function

induced by differential operators in governing equations. Finally, there are issues with the nonlinear expression and feature learning capabilities of a PINN. The frequency principle implies that for an FCNN, high-frequency features are difficult to capture, which is often manifested as the difficulty in solving flow characteristics in areas with sharp gradients in fluid simulation tasks. This study seeks to partially resolve these three difficulties, therefore advancing the use of PINN in solving the RANS equations.

1.3 Research Methodology and Tasks

Based on Sections 1.1 and 1.2, the rest of this study will be organized into six chapters. In addition to the literature review in Chapter 2 and the summary in Chapter 7, the main research will be divided into four sections, namely Dynamic Prioritization in Chapter 3, Multifidelity Modeling in Chapter 4, Quantum Layer Integration in Chapter 5, and Weighted Sum Turbulence Model in Chapter 6, which will be explained sequentially as follows.

In Chapter 2, a comprehensive review is conducted, concentrating on the origin as well as the mathematical form of the RANS equations. This chapter reviews the conventional numerical methods for solving the RANS equations, the concept of PINN, and its developments in recent years. At the end of this chapter, the research gap in this thesis is clarified.

In Chapter 3, a novel self-adaptive dynamic prioritization loss balance strategy is proposed to partially alleviate the loss balance issue to improve the convergence performance of the PINN in the RANS simulations. This approach first reorganizes the loss function in the PINN-RANS simulations and then borrows the experience from the loss balancing strategy in multi-task learning. The weight coefficients in the PINN's loss function are determined by the relative errors between its prediction values and the training data. Greater weights are assigned to the

loss components with larger relative errors as a punishment, wherein these loss components can be emphasized in the next stages of the training process.

In Chapter 4, a Gaussian process-based multifidelity modeling algorithm is adopted as a post-processing step in the PINN-RANS simulation. In this research, the solution of the PINN-RANS simulation is defined as the low-fidelity data, while the measurement data is regarded as the high-fidelity data. The algorithm captures the nonlinear non-functional space-dependent cross-correlations between the low-fidelity and high-fidelity data sets. It utilizes the trend of the low-fidelity PINN-RANS simulation results to fit the scattered data points on the high-fidelity level. This approach significantly alleviates the problem of the inconsistency between PINN-RANS simulations and the measurement data.

In Chapter 5, in order to improve the nonlinear expression and feature learning capabilities of the PINN, the structure of the FCNN, which is the function fitting module, has been the focus of innovation. By incorporating a quantum layer into the FCNN structure, the PINN is transformed into a hybrid classical-quantum model. Empirical evidence has shown that this model successfully mitigates the problems arising from the frequency principle, therefore enhancing the expressive capability of the model in the PINN-RANS simulations.

In Chapter 6, a weighted sum RANS turbulence model is proposed under the PINN framework, which is based on the linear superposition of the existing zero-equation RANS turbulence models. A novel PINN structure is designed for calculating the Reynolds stress loss and other physics-based losses in the meantime. The weight for each base model is automatically optimized through minimizing the Reynolds stress loss. Results demonstrate that the proposed method significantly alleviates the poor applicability of the RANS turbulence model.

Chapter 7 provides a comprehensive summary of the thesis. Furthermore, in this chapter, the drawbacks of the proposed methods are laid out and the issues that still need attention are identified. At the end of the entire thesis, the potential research directions are outlined, which are the further extensions of existing works.

Figure 1-2 illustrates the technical route of this thesis as a summary for this chapter. The figure provides a concise overview of the key issues identified in the current research, as well as the key technologies to be used and research content in the following chapters. The research objectives are also outlined in the figure.

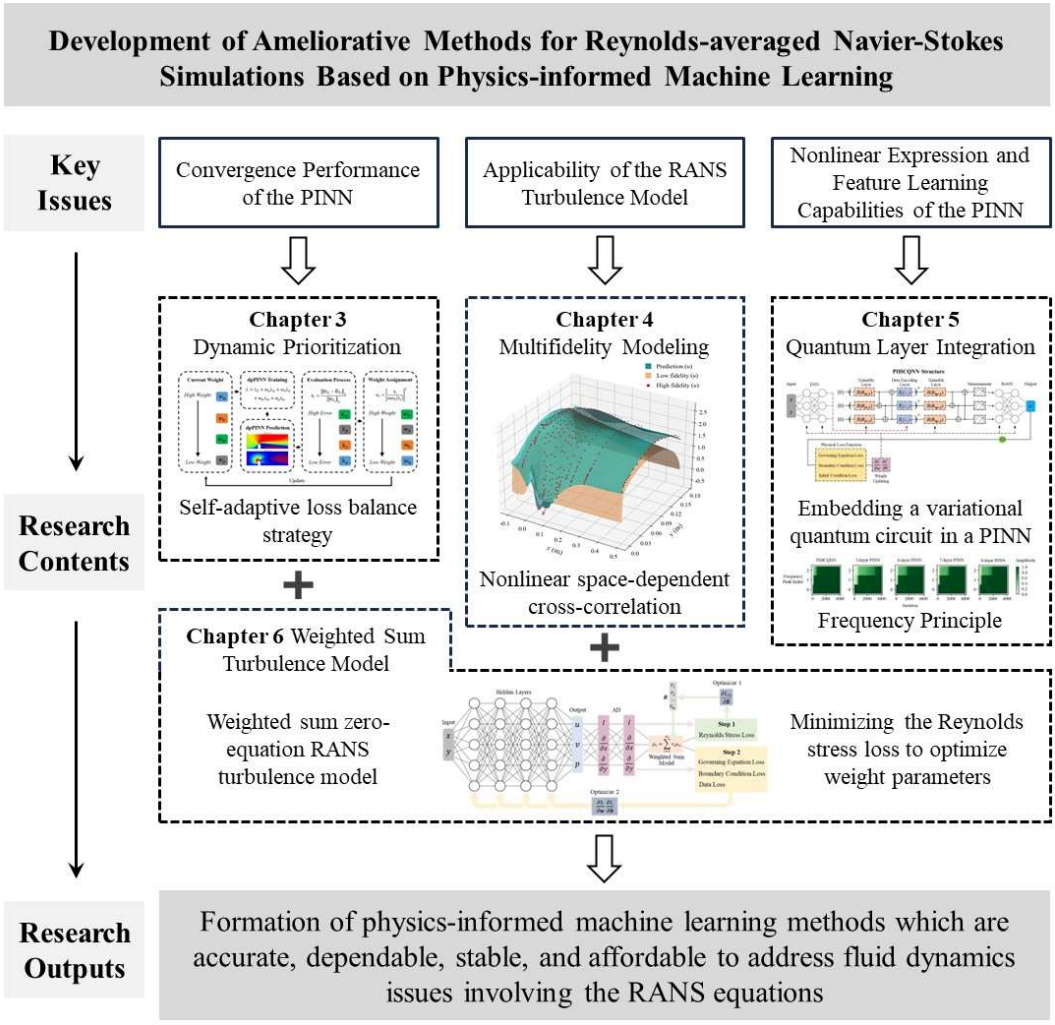


Figure 1-2. Technical route of this thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 An Overview of the RANS Equations

2.1.1 Beginning with the NS equations

Since the RANS equations are derived by averaging the NS equations, a thorough comprehension of the latter is essential before delving into the RANS equations. The NS equations is a collective name, covering the fluid continuity and momentum equations. For the incompressible flow, the fluid continuity equation takes the following form. Note that the subscripts in this thesis follow the Einstein summation convention.

$$\frac{\partial u_i}{\partial x_i} = 0 \qquad (2-1)$$

where x_i is the spatial coordinate, and u_i is the velocity component in the x_i direction. The continuity equation is the specific manifestation of the mass conservation law in fluid flows. For the incompressible flow, the momentum equation takes the following form

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial (u_i)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] + f_i \qquad (2-2)$$

where t stands for the time. ρ denotes the fluid density. p denotes the pressure. ν stands for the kinematic viscosity of the fluid. f_i is the external force. The momentum equation can be split into five distinct terms. First comes the time derivative term $\frac{\partial u_i}{\partial t}$, which describes the

variation of the flow rate over time. Then comes the inertial force term $\frac{\partial(u_i u_j)}{\partial x_j}$, which reflects the inertial response of the fluid flow. $\frac{1}{\rho}\frac{\partial p}{\partial x_i}$ denotes the pressure force, followed by the viscous force term $\frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right]$, which represents the effect of smoothing flow distribution due to viscosity. The rightmost side of the equation is the external forces applied to the fluid f_i. If there is no external force, this term equals zero.

2.1.2 Reynolds averaging

Reynolds averaging refers to the process of dividing an instantaneous physical quantity $\emptyset(\boldsymbol{x},t)$, which is within the time range of $[0,T]$, into a mean quantity $\overline{\emptyset}(\boldsymbol{x},t)$ and a fluctuating quantity $\emptyset'(\boldsymbol{x},t)$. For steady flows, the mean quantity $\overline{\emptyset}(\boldsymbol{x},t)$ is independent of time. Thus, it is also written as $\overline{\emptyset}(\boldsymbol{x})$. This thesis discusses steady flows, therefore in the following text, $\overline{\emptyset}$ refers to $\overline{\emptyset}(\boldsymbol{x})$. Mathematically, the quantity $\emptyset$ can be written as follows

$$\emptyset = \overline{\emptyset} + \emptyset' \qquad (2-3)$$

where

$$\overline{\emptyset} = \frac{1}{T}\int_0^T \emptyset\, d\tau \qquad (2-4)$$

Here, T represents the length of the time range, which should exceed the fluctuation period of turbulence by a substantial margin. Now, it is necessary to clarify some rules of Reynolds averaging before performing Reynolds averaging on the NS equations. These rules are listed as follows

$$\overline{\overline{\emptyset}} = \frac{1}{T}\int_0^T \overline{\emptyset}\, d\tau = \frac{\overline{\emptyset}}{T}\int_0^T d\tau = \overline{\emptyset} \qquad (2-5)$$

$$\overline{\emptyset \pm \varphi} = \frac{1}{T}\int_0^T (\emptyset \pm \varphi)\, d\tau = \frac{1}{T}\int_0^T \emptyset\, d\tau \pm \frac{1}{T}\int_0^T \varphi\, d\tau = \bar{\emptyset} \pm \bar{\varphi} \qquad (2-6)$$

$$\bar{\emptyset}' = \overline{\emptyset - \bar{\emptyset}} = \bar{\emptyset} - \bar{\bar{\emptyset}} = 0 \qquad (2-7)$$

$$\overline{\bar{\emptyset}\varphi} = \frac{1}{T}\int_0^T \bar{\emptyset}\,\varphi d\tau = \bar{\emptyset}\frac{1}{T}\int_0^T \varphi\, d\tau = \bar{\emptyset}\bar{\varphi} \qquad (2-8)$$

$$\overline{\emptyset\varphi} = \overline{(\bar{\emptyset} + \emptyset')(\bar{\varphi} + \varphi')} = \overline{\bar{\emptyset}\bar{\varphi}} + \overline{\bar{\emptyset}\varphi'} + \overline{\emptyset'\bar{\varphi}} + \overline{\emptyset'\varphi'} = \bar{\emptyset}\bar{\varphi} + \overline{\emptyset'\varphi'} \qquad (2-9)$$

$$\overline{\frac{\partial \emptyset}{\partial t}} = \frac{1}{T}\int_0^T \frac{\partial \emptyset}{\partial t} d\tau = \frac{1}{T}[\emptyset(\boldsymbol{x}, T) - \emptyset(\boldsymbol{x}, 0)] = \frac{1}{T}\frac{\partial}{\partial t}\left(\int_0^T \emptyset d\tau\right) = \frac{\partial \bar{\emptyset}}{\partial t} = 0 \qquad (2-10)$$

$$\overline{\frac{\partial \emptyset}{\partial \boldsymbol{x}}} = \frac{1}{T}\int_0^T \frac{\partial \emptyset}{\partial \boldsymbol{x}} d\tau = \frac{1}{T}\frac{\partial}{\partial \boldsymbol{x}}\left(\int_0^T \emptyset d\tau\right) = \frac{\partial \bar{\emptyset}}{\partial \boldsymbol{x}} \qquad (2-11)$$

Based on the above rules, the continuity and momentum equations in the NS equations can be Reynolds averaged, thus transforming the NS equations into the RANS equations.

2.1.3 The RANS equations

First, the continuity equation in the incompressible NS equations, namely Eq.(2-1), undergoes Reynolds averaging on both sides of the equal sign, which is as follows

$$\overline{\frac{\partial u_\iota}{\partial x_\iota}} = \bar{0} \qquad (2-12)$$

Then, based on Eq. (2-11), we may get

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \qquad (2-13)$$

Here, $\bar{u}_i$ denotes the mean quantity of instantaneous fluid velocity u_i. Or, to put it more bluntly, $\bar{u}_i$ is the time-averaged velocity components in the x_i direction. Similarly, Reynolds

averaging is applied to both sides of the equal sign in the momentum equation, i.e., Eq. (2-2). Then, we may get

$$\overline{\frac{\partial u_\iota}{\partial t}} + \overline{u_\jmath \frac{\partial (u_\iota)}{\partial x_\jmath}} = -\frac{1}{\rho}\overline{\frac{\partial p}{\partial x_\iota}} + \overline{\frac{\partial}{\partial x_\jmath}\left[\nu\left(\frac{\partial u_\iota}{\partial x_\jmath} + \frac{\partial u_\jmath}{\partial x_\iota}\right)\right]} + \overline{f_\iota} \qquad (2-14)$$

Handle the terms on the left side of the equal sign individually. For steady flows, there are

$$\begin{aligned}\overline{\frac{\partial u_\iota}{\partial t}} + \overline{u_\jmath \frac{\partial (u_\iota)}{\partial x_\jmath}} &= \overline{(\overline{u_\jmath} + u_\jmath{}')\frac{\partial(\overline{u_\iota} + u_\iota{}')}{\partial x_\jmath}} = \overline{\overline{u_\jmath}\frac{\partial \overline{u_\iota}}{\partial x_\jmath} + \overline{u_\jmath}\frac{\partial u_\iota{}'}{\partial x_\jmath} + u_\jmath{}'\frac{\partial \overline{u_\iota}}{\partial x_\jmath} + u_\jmath{}'\frac{\partial u_\iota{}'}{\partial x_\jmath}} \\ &= \overline{\overline{u_\jmath}}\frac{\partial \overline{u_\iota}}{\partial x_\jmath} + \overline{u_\jmath{}'\frac{\partial u_\iota{}'}{\partial x_\jmath}} = \overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right) \qquad (2-15)\end{aligned}$$

Then, handle the terms on the right side of the equal sign individually. For steady flows, if there is no external force, it will be as follows

$$-\frac{1}{\rho}\overline{\frac{\partial p}{\partial x_\iota}} + \overline{\frac{\partial}{\partial x_\jmath}\left[\nu\left(\frac{\partial u_\iota}{\partial x_\jmath} + \frac{\partial u_\jmath}{\partial x_\iota}\right)\right]} + \overline{f_\iota} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial}{\partial x_\jmath}\overline{\left(\frac{\partial u_\iota}{\partial x_\jmath}\right)} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial^2 \overline{u}_i}{\partial {x_\jmath}^2} \qquad (2-16)$$

Based on Eq. (2-14), (2-15) and (2-16), we may get

$$\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu\frac{\partial \overline{u}_i}{\partial x_j} - \overline{u_i' u_j'}\right) \qquad (2-17)$$

If the density of the fluid is multiplied on both sides of the equal sign simultaneously, then this equation may be expressed as follows

$$\rho\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\frac{\partial \overline{u}_i}{\partial x_j} - \rho\overline{u_i' u_j'}\right) \qquad (2-18)$$

where μ is the dynamic viscosity, and $\mu = \rho\nu$. Eq. (2-18) is the momentum equation after Reynolds averaging, which forms the RANS equations together with Eq. (2-13). $-\rho\overline{u_i' u_j'}$ is an additional mathematical term introduced in the process of Reynolds averaging, which is the

well-known Reynolds stress term. For convenience, $-\rho\overline{u_i'u_j'}$ is substituted by R_{ij} in the following text. The additional introduction of the Reynolds stress term R_{ij} has led to the non-closure problem of the system of equations. Handling the Reynolds stress term to achieve the closure of the RANS equations is also known as RANS turbulence modeling. Various treatments of this term have led to the appearances of different RANS turbulence models.

2.1.4 RANS Turbulence modeling

The treatment methods of the Reynolds stress term in the RANS equations can be mainly divided into two categories. One is to use the Reynolds stress model (RSM) to describe the Reynolds stress term, that is, to establish the transport equation for each element in the Reynolds stress term. RSMs are believed to have the capability to provide accurate predictions for complex flows. Nevertheless, the precision of RSM predictions is inherently limited by the closure assumptions of several terms in the RSM transport equations, such as the pressure-strain and dissipation rate terms. Furthermore, the introduction of a substantial number of transport equations (five in two-dimensional cases and seven in three-dimensional cases) results in quite high computational expense for RSM. Consequently, RSM is not the preferred method in RANS turbulence modeling.

In fact, currently, the dominant approaches to RANS turbulence modelling are based on the Boussinesq eddy viscosity assumption, which presupposes a linear correlation between the Reynolds stress and the mean rate of strain tensor. Based on the Boussinesq eddy viscosity assumption, the Reynolds stress can be expressed as follows

$$R_{ij} = \mu_t\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho\delta_{ij}k \qquad (2-19)$$

where μ_t is the turbulent viscosity. δ_{ij} is the Kronecker delta function. $k = \frac{1}{2}\overline{u_i' u_i'}$ is the turbulent kinetic energy. Substitute the Reynolds stress term in Eq. (2-18) with Eq. (2-19), and then we may get

$$\begin{aligned}\rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} &= -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial \bar{u}_i}{\partial x_j} + \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - c\rho\delta_{ij}k\right) \\ &= -\frac{\partial\left(\bar{p} + \frac{2}{3}\rho k\right)}{\partial x_i} + \frac{\partial}{\partial x_j}\left((\mu + \mu_t)\frac{\partial \bar{u}_i}{\partial x_j}\right) \end{aligned} \quad (2-20)$$

In some literature, authors may use μ_{eff} to refer to the sum of the dynamic viscosity μ and the turbulent viscosity μ_t, which is also known as the effective viscosity. Therefore, it is evident from the above equation that the problem of determining the Reynolds stress term in the RANS equations has been transformed into determining the turbulent viscosity μ_t, which is the only unknown brought by Reynolds averaging in Eq. (2-20). To determine the turbulent viscosity μ_t, the general approach is to introduce new PDEs to solve it. Depending on the number of additional equations introduced, these methods are categorized as zero-equation models, one-equation models, two-equation models, and so on.

The zero-equation model, also known as the algebraic model, has the simplest form among various turbulence models. The essence of the zero-equation model is to describe turbulent viscosity using the averaged characteristics of the flow. For instance, the Prandtl mixing-length model is one of the most well-known zero-equation RANS turbulence models (Prandtl 1925). The turbulent viscosity μ_t is described as follows

$$\mu_t = C_{p\mu}\rho u_t l_{p0} \quad (2-21)$$

where $C_{p\mu}$ is a problem-dependent constant and l_{p0} is a prescribed length based on the location of the grid. u_t is the characteristic velocity of the fluid. In the two-dimensional shear

flows, there are $u_t = L\left|\frac{\partial \bar{u}}{\partial y}\right|$. Here, $\bar{u}$ denotes the time-averaged mainstream velocity and y is the perpendicular direction. Based on Eq. (2-21), it is evident that the Prandtl mixing-length model couples the turbulent viscosity with the gradient of the time-averaged flow velocity without introducing new PDEs in equation solving. Consequently, it is referred to as the zero-equation RANS turbulence model. The Prandtl mixing-length model is demonstrated to have good precision in modeling shear flows, waves, and so on.

The Baldwin-Lomax model is another commonly used zero-equation model for RANS turbulence enclosure (Baldwin and Lomax 1978). The model decomposes the flow field into two layers, namely the inner and outer layers. Each layer employs a distinct calculation method to determine turbulent viscosity u_t. It is worth noting that when using the Baldwin-Lomax model, the value of k in Eq. (2-19) is set to zero, which is a prevalent approach when zero-equation models are adopted. Other commonly used zero-equation models include the Chen model for indoor airflow simulation (Chen and Xu 1998), and the Li model for modeling external airflow over buildings (Li *et al.* 2012).

Despite their significant computational efficiency, zero-equation models generally lack universality and are unable to accurately represent flows in diverse intricate geometries. Moreover, these zero-equation models may lack the capability to adequately include the historical influences of the turbulence, such as the convection and diffusion of the turbulent energy. Lastly, when dealing with complicated turbulent flows, it is challenging to identify the turbulent length scale which always appears in various zero-equation models. These factors all make it difficult for the zero-equation RANS turbulence models to be widely applied in dealing with practical problems.

The one-equation model refers to the RANS turbulence model that incorporates one extra

PDE into Reynolds stress modeling. Prandtl was also the first to propose a one-equation RANS turbulence model (Prandtl 1945). In this model, an equation describing turbulent kinetic energy k is introduced, which takes the following form

$$\frac{\partial \rho k}{\partial t} + \bar{u}_j \frac{\partial \rho k}{\partial x_j} = R_{ij} \frac{\partial \bar{u}_i}{\partial x_j} - C_D \frac{\rho k^{\frac{3}{2}}}{l_{p1}} + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \qquad (2-22)$$

where l_{p1} is the turbulent length scale. To be noted, the approach for estimating the turbulent length scale is similar to the one used in the Prandtl mixing-length model. In addition, there is the following relationship between the turbulent kinetic energy k and the turbulent viscosity μ_t

$$\mu_t = \rho k^{\frac{1}{2}} l_{p1} \qquad (2-23)$$

It is not difficult to discover that there are some empirical constants in the Prandtl's one-equation model, and the values of these empirical constants directly affect the accuracy of the RANS turbulence model. The typical values of the empirical constants in the Prandtl's one-equation model are now tabulated in Table 2-1 for readers' reference.

Table 2-1. The typical values of the empirical constants in the Prandtl's one-equation model.

Coefficient	C_D	σ_k
Value	0.08	1

Other well-known one-equation RANS turbulence models include the Spalart-Allmaras model (Spalart and Allmaras 1992), the Baldwin-Barth model (Baldwin and Barth 1991), and so on. In general, the one-equation models outperform the zero-equation models for simulating separated flows, which is due to the calculation of the convection and diffusion effects of

turbulence. Nevertheless, similar to the zero-equation model, the one-equation model still requires empirical estimation of turbulent length scales. As a result, its overall generality remains limited and it is only applicable in certain cases, such as the hypersonic flows (Paciorri *et al.* 1997).

Among all the RANS turbulence models, the two-equation model is the prevailing form. The plenty of engineering cases adopting two-equation models in RANS turbulence modeling are enough to prove its core position in this field. As its name suggests, a two-equation model includes two additional transport equations to characterize turbulent properties, in addition to the continuity and momentum equations. Basically, there will be an equation used to describe the turbulent kinetic energy k. As for the second additional equation introduced, it depends on the specific model.

For example, the standard *k*-*ε* model is one of the well-known two-equation turbulence models, which has been widely adopted in practical engineering (Launder and Sharma 1974). Apart from the continuity and the momentum equations, the kinetic energy equation and the dissipation equation (i.e., *k*-equation and *ε*-equation) are additionally introduced in the standard *k*-*ε* turbulence model to simulate turbulent behaviors. *k*-equation and *ε*-equation can be described as follows (buoyancy is neglected)

$$\frac{\partial}{\partial t}(\rho k)+\frac{\partial}{\partial x_i}(\rho k\bar{u}_i)=\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right]+\mu_t S^2+R_{ij}\frac{\partial \bar{u}_j}{\partial x_i}-\rho\varepsilon \qquad (2-24)$$

$$\frac{\partial}{\partial t}(\rho \varepsilon)+\frac{\partial}{\partial x_i}(\rho \varepsilon\bar{u}_i)=\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right]+C_{1\varepsilon}\frac{\varepsilon}{k}R_{ij}\frac{\partial \bar{u}_j}{\partial x_i}-C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \qquad (2-25)$$

Here, $S=\sqrt{2S_{ij}S_{ij}}$. $S_{ij}=\frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j}+\frac{\partial \bar{u}_j}{\partial x_i}\right)$, which is the mean strain rate tensor. ε is the turbulent dissipation rate. k, ε, and μ_t have the following relationship

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \qquad (2-26)$$

The typical values of the empirical constants in the standard *k*-ε model are now tabulated in Table 2-2 for readers' reference.

Table 2-2. The typical values of the empirical constants in the standard *k*-ε model.

Coefficient	C_μ	$C_{1\varepsilon}$	$C_{2\varepsilon}$	σ_k	σ_ε
Value	0.09	1.44	1.92	1.0	1.3

As another well-known two-equation RANS turbulence model, the Re-Normalization Group (RNG) *k*-ε model is the renormalized form of the standard *k*-ε model, which takes the smaller scales of turbulent fluctuation into consideration (Smith and Woodruff 1998). More specifically, in the RNG *k*-ε model, the *k*-equation has the same mathematical expression as that in the standard *k*-ε model. The only difference is the ε-equation, which takes the following form at present

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \frac{\partial}{\partial x_i}(\rho\varepsilon\bar{u}_i) = \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}R_{ij}\frac{\partial \bar{u}_j}{\partial x_i} - C_{2\varepsilon}^*\rho\frac{\varepsilon^2}{k} \qquad (2-27)$$

where

$$C_{2\varepsilon}^* = C_{2\varepsilon} + \frac{C_\mu \eta^3 \left(1 - \frac{\eta}{\eta_0}\right)}{1 + \beta\eta^3} \qquad (2-28)$$

Here, $\eta = \frac{Sk}{\varepsilon}$. The typical values of the empirical constants in the RNG *k*-ε model are now tabulated in Table 2-3 for readers' reference.

Table 2-3. The typical values of the empirical constants in the RNG *k*-ε model.

Coefficient	C_μ	$C_{1\varepsilon}$	$C_{2\varepsilon}$	σ_k	σ_ε	η_0	β
Value	0.0845	1.42	1.68	0.7194	0.7194	4.38	0.012

In contrast to the zero-equation and one-equation models, the two-equation model essentially introduces additional equations to estimate the turbulence length scale using turbulence kinetic energy and turbulence dissipation rate, thereby enclosing the RANS equations. An advantage of the two-equation RANS turbulence model is its enhanced applicability, making it suitable for a broader spectrum of flow conditions. Nevertheless, the drawback lies in the potential increase in computational expenses and the comparatively low computational efficiency.

Besides the aforementioned models, there are more RANS turbulence models such as the three-equation models (Walters and Cokljat 2008, Li *et al.* 2020), the four-equation models (Chitta *et al.* 2013, Grunloh 2019), and so on. Moreover, some models have challenged the Boussinesq eddy viscosity assumption by assuming a nonlinear correlation between the Reynolds stress term and the time-averaged turbulent properties (Craft *et al.* 1997, Bauer *et al.* 2000). While these models will not be enumerated, their fundamental nature lies in different approaches for the characterization of the Reynolds stress term.

2.2 Conventional Methods for Solving the RANS Equations

Once the physical modeling is completed, namely the closure of the RANS equations, it becomes imperative to address the subsequent challenge of using numerical methods for approximating the solutions of these physical equations. One may doubt the need to pursue an

analytical solution. While it is indeed feasible to explore the possibility of obtaining an analytical solution, as the problem becomes more complex, the task of finding such a solution becomes challenging. Thus, the use of other approximation approaches, i.e., numerical methods, is necessitated. More specifically, the primary objective of such numerical methods is to develop effective, resilient, and dependable numerical algorithms for obtaining an approximate solution of the governing PDEs.

As mentioned in Section 1.1, the CFD methods, which are the numerical methods involving the discretization of the governing PDEs within the computational domain, have been predominantly reigning in the realm of RANS turbulence simulations. CFD methods serve as an umbrella term, encompassing two main approaches in RANS simulations. One faction employs the finite volume method (FVM) as its foundation for solving the RANS equations, while the other relies on a finite element method (FEM), i.e., the Discontinuous Galerkin method (DGM), for tackling these equations. The following discussion will delve into these two methods.

2.2.1 FVM

FVM is based on the concept of partitioning the computational domain into a number of grids and aiming to find the approximate solutions for the PDEs on the grid nodes. FVM generates a non-overlapping control volume around each grid point. It then integrates the PDEs to be solved for each control volume to obtain a set of discrete equations. In order to calculate the integral, it is necessary to assume the variation law of the solution of the PDEs between grid nodes, that is, to set the piecewise interpolation of the solution function between grid nodes. The advantage of FVM lies in the fact that the discrete equations satisfy conservation in each control volume and also in the entire computational domain.

Taking the steady one-dimensional diffusion equation as an example, the discretization method of FVM will be discussed. The steady one-dimensional diffusion equation can be expressed as follows

$$\frac{d}{dx}\left(K\frac{d\psi}{dx}\right)+Q=0 \qquad (2-29)$$

where ψ denotes the physical quantity of diffusion. K denotes the diffusion coefficient. Q is the rate of heat generation per unit volume. Figure 2-1 depicts the control volume and grid points when solving the steady one-dimensional diffusion equation. As shown in the figure, the grid point P is sandwiched between adjacent grid points W and E. The dashed lines represent the boundary surfaces w and e.

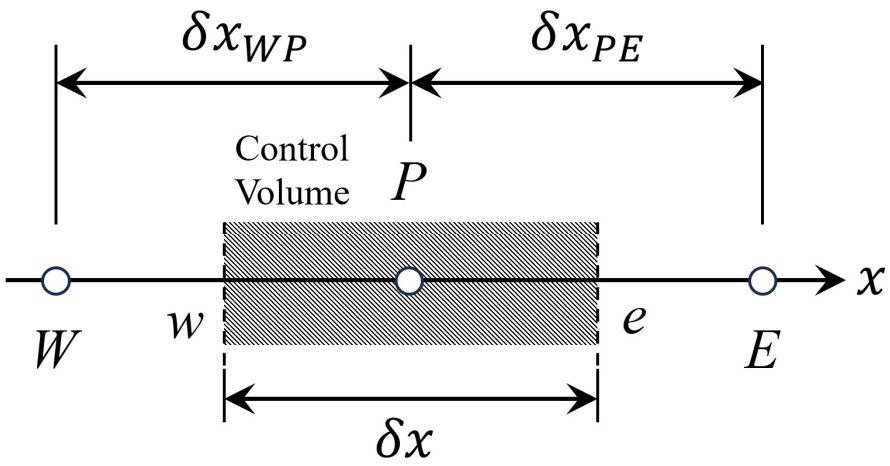


Figure 2-1. The control volume and grid points when solving the steady one-dimensional diffusion equation.

By integrating Eq. (2-29) with the control volume around the grid point P, we may obtain the following equation

$$\left(K\frac{d\psi}{dx}\right)_e-\left(K\frac{d\psi}{dx}\right)_w+\int_w^e Q dx=0 \qquad (2-30)$$

In order to solve the above equation, it is important to approximate the derivative terms in the equation. We can now assume that the solution function is linearly interpolated between grid points. Then, we may get

$$\frac{K_e(\psi_E - \psi_P)}{\delta x_{PE}} - \frac{K_w(\psi_P - \psi_W)}{\delta x_{PE}} + \bar{Q}_P \delta x = 0 \qquad (2-31)$$

where $\bar{Q}_P$ denotes the averaged value of the source term Q within the control volume. In most cases, the source term Q is a function of the variable ψ. Now assume that $\bar{Q}_P$ takes the following form

$$\bar{Q}_P = Q_c + a\psi_P \qquad (2-32)$$

where Q_c is the constant component of the $\bar{Q}_P$, and a is the coefficient of ψ_P. Then by substituting the $\bar{Q}_P$ in Eq. (2-31) with Eq. (2-32), we may get the following equation

$$\frac{K_e(\psi_E - \psi_P)}{\delta x_{PE}} - \frac{K_w(\psi_P - \psi_W)}{\delta x_{PE}} + (Q_c + a\psi_P)\delta x = 0 \qquad (2-33)$$

If we denote $\frac{K_e}{\delta x_{PE}}$ as a_E and $\frac{K_w}{\delta x_{PE}}$ as a_W, then we may get

$$a_E(\psi_E - \psi_P) - a_W(\psi_P - \psi_W) + Q_c\delta x + a\psi_P\delta x = 0 \qquad (2-34)$$

That is

$$a_E\psi_E + a_W\psi_W + Q_c\delta x = (a_E + a_W - a\delta x)\psi_P \qquad (2-35)$$

Then, we may find that Eq. (2-35) equals to the following form

$$a_E\psi_E + a_W\psi_W + b = a_P\psi_P \qquad (2-36)$$

where $b = Q_c\delta x$, and $a_P = a_E + a_W - a\delta x$. Eq. (2-36) is the discretization form of Eq. (2-29) for use in further numerical computations. Notably, a linear interpolation is employed to

calculate the derivative, which is not the mandatory choice. Alternative methods of interpolation can be employed to calculate the derivative, potentially yielding different outcomes. From Eq. (2-30) it can be discoverd that FVM is directly based on conservation laws, i.e., mass, momentum, energy, etc. FVM ensures the conservation of these physical quantities by integrating over each control volume. This is particularly important for RANS turbulence simulations, as turbulence involves complex momentum and energy exchanges. FVM's capability of ensuring physical conservation laws can improve the physical modeling accuracy of the RANS turbulence simulations. The preceding analysis also clearly shows that FVM can handle complicated geometries and boundary constraints. Turbulence typically occurs in complex engineering environments such as the regions around aircraft wings and car bodies, and the FVM's flexibility allows it to better adapt to these complex geometries.

The result is that many well-known commercial CFD software, such as *Ansys Fluent*, *Star CCM+*, and *OpenFOAM*, have all chosen FVM as the computational method for fluid flow simulation, including RANS turbulence simulation. Researchers have carried out many meaningful studies in a variety of domains based on mature commercial software, for example, in the fields of building's indoor and outdoor flow field simulation and high-speed train aerodynamics.

In the field of indoor airflow simulation, for instance, numerous research studies have been conducted, that integrated the RANS turbulence models with the FVM approach to simulate the building's indoor environment (Durrani *et al.* 2015). Stamou and Katsiris (2006) verified the applicability of the Shear Stress Transport (SST) k-ω RANS turbulence model, which was proposed by Menter (1993), in an office environment. The work was carried out using the FVM-based CFD commercial software *CFX*. This work also compared the simulation results of the standard k-ε model and the RNG k-ε model are also with experimental data. The

results in this study indicated that the SST k-ω RANS turbulence model is the model highly recommended for practical applications in indoor airflow simulations due to its consistency with the experimental data.

Posner *et al.* (2003) investigated the influence of obstructions on the indoor flow field. Initially, they employed commercial software *Ansys Fluent*, which is based on the FVM approach, to conduct RANS simulations using various turbulence models including the standard k-ε model and the RNG k-ε model. Subsequently, they implemented experimental measurements employing the particle image velocimetry (PIV) and laser Doppler anemometry (LDA) technologies to verify the CFD simulation results. Their findings suggested that the RNG k-ε model may be the most effective in simulating indoor airflow when there exist obstructions indoors.

Considering the substantial computing expense induced by the two-equation RANS turbulence models in indoor airflow simulation, Chen and Xu (1998) proposed a simple zero-equation RANS turbulence model as an alternative. Figure 2-2 depicts the comparison between simulated wind speed and measured results when using the proposed zero-equation model and the standard k-ε model. The findings suggested that while the proposed zero-equation model has enhanced computing efficiency, the simulated wind velocities in regions with significant gradient variations are inferior to those obtained using the conventional two-equation RANS turbulence models. It is worth mentioning that this study was carried out using the commercial FVM-based software *PHOENICS*.

Furthermore, apart from the indoor flow field, there are also many research cases of FVM-based RANS simulations for building's outdoor flow (Van Hooff *et al.* 2017, Vita *et al.* 2020, Zheng *et al.* 2020). Acquiring outdoor airflow characteristics is crucial in building simulations

since they have a direct impact on the thermal environment and air pollution issues around a building (Mfula *et al.* 2005, Challoner and Gill 2014, Yang *et al.* 2021).

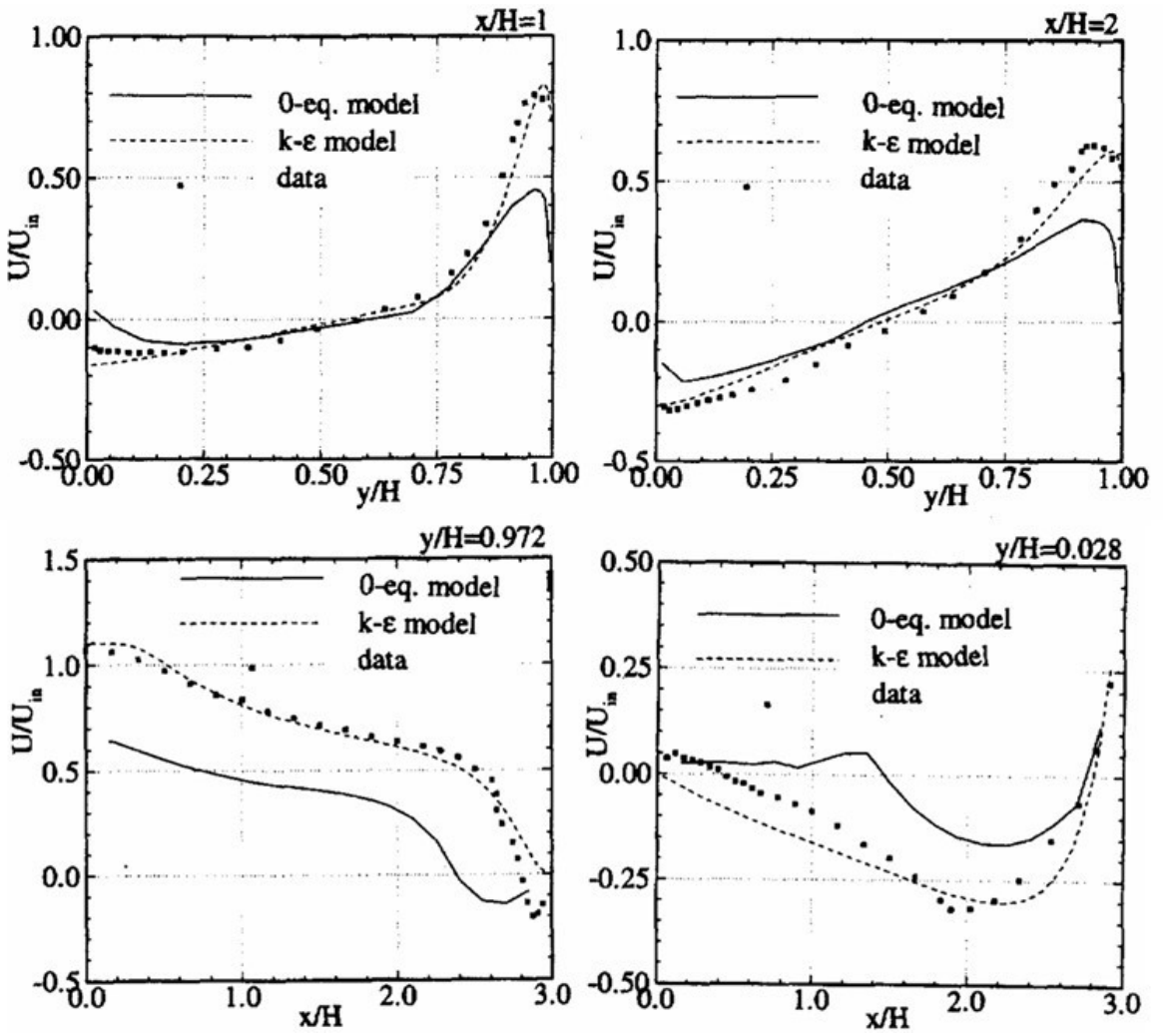


Figure 2-2. A comparison between the RANS simulated wind speed and measurement results carried out by Chen and Xu (1998).

Jędrzejewski *et al.* (2017) conducted research on investigating the airflow characteristics around building clusters using *Ansys Fluent*. Aside from utilizing the widely used realizable *k-ε* model, which was proposed by Shih *et al.* (1995), to close the RANS equations, the aforementioned RSM was also adopted in the simulation. Figure 2-3 illustrates the disparity between the results obtained from the wind tunnel test and FVM-based numerical results. The

cross represents the discrete data from the experiment, the red line represents the results of the realizable *k*-ε model, and the blue line represents the results of the RSM. Such a finding suggests that the FVM-based RANS simulation has the capability to accurately present the actual flow field around building structures, provided that the turbulence model is chosen properly.

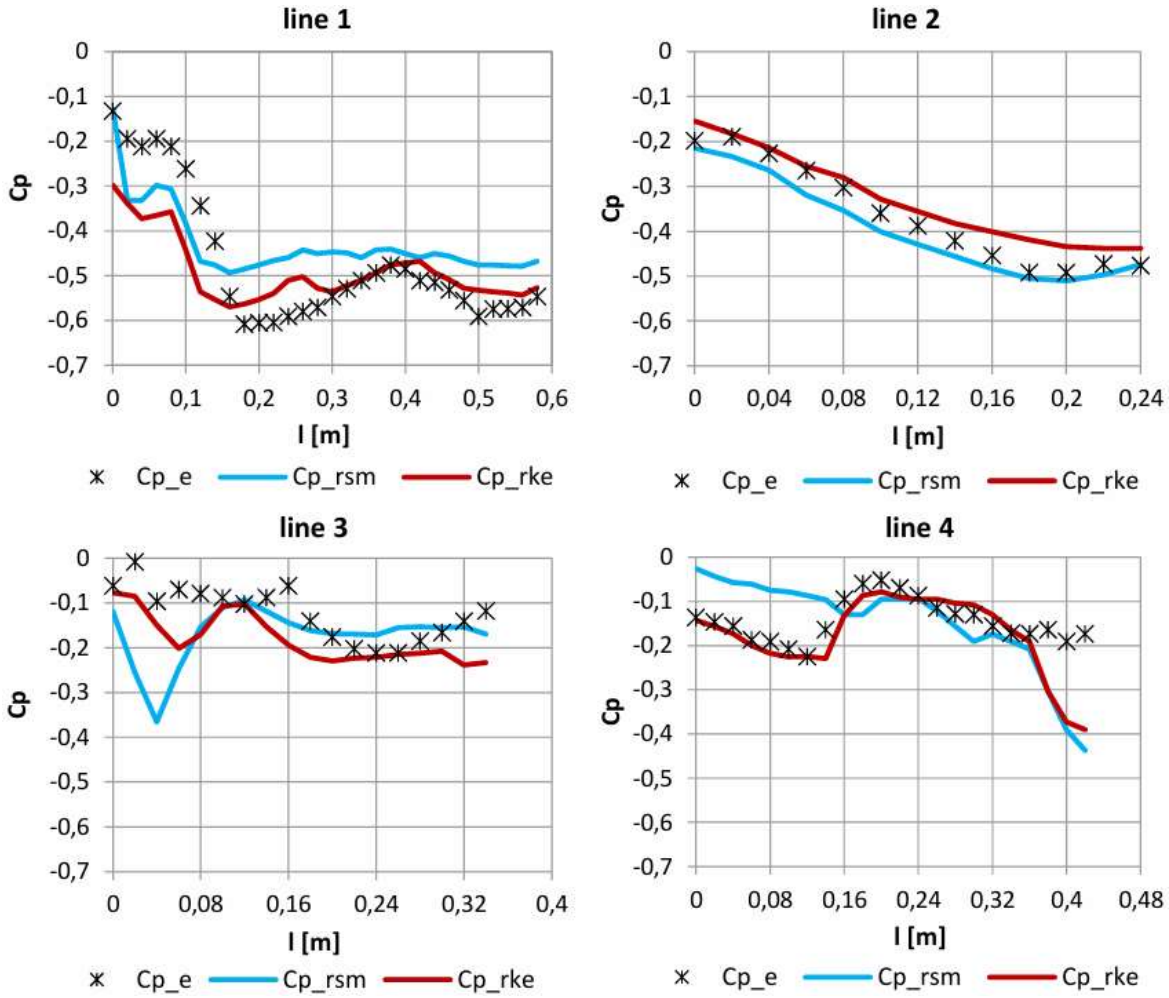


Figure 2-3. Comparison between the pressure coefficients obtained from RANS simulations and wind tunnel experiments by Jędrzejewski *et al.* (2017).

In the realm of high-speed train aerodynamics, researchers have also carried out a number of meaningful works using FVM-based RANS simulations (Diedrichs 2010, Rezvani and

Mohebbi 2014, Munoz-Paniagua *et al.* 2017). Issues including a sudden rise in the air resistance and a higher risk of overturning induced by crosswinds when trains operate at high speeds have emerged as critical challenges in the development of high-speed rail (Brockie and Baker 1990, Raghunathan *et al.* 2002, Baker *et al.* 2011). Real vehicle testing opportunities are rare, leading to the difficulty for researchers to obtain data in real environments (Gallagher *et al.* 2018, Misu and Ishihara 2018). Owing to its inherent ground effects and other factors that may influence the experimental results, the wind tunnel test is not the mainstream method in this research field (Baker and Brockie 1991). Thus, numerical simulation has become a fundamental tool in the discipline of high-speed train aerodynamics to search for strategies for reducing air resistance and preventing train overturning. Given the complex geometry of the train, the steady RANS simulation is a commonly used numerical method to minimize the number of grids and computational expenses without compromising accuracy.

For instance, Li *et al.* (2019) applied FVM-based RANS simulations to the study of investigating the train aerodynamics under crosswinds, which determined the optimal RANS turbulence model for modeling the aerodynamic characteristics of high-speed trains. Comparative analysis was conducted on six RANS turbulence models, including the aforementioned standard k-ε model, RNG k-ε model, Realizable k-ε model, SST k-ω model, and Spalart–Allmaras model. Figure 2-4 shows the analysis of train surface pressure distribution using various RANS turbulence models. The results in the figure indicated that the choice of the RANS turbulence model will influence the surface pressure distribution significantly. The authors also suggested that the SST k-ω model might be the best choice for FVM-based RANS simulations of train aerodynamics under crosswind conditions considering its extremely high accuracy.

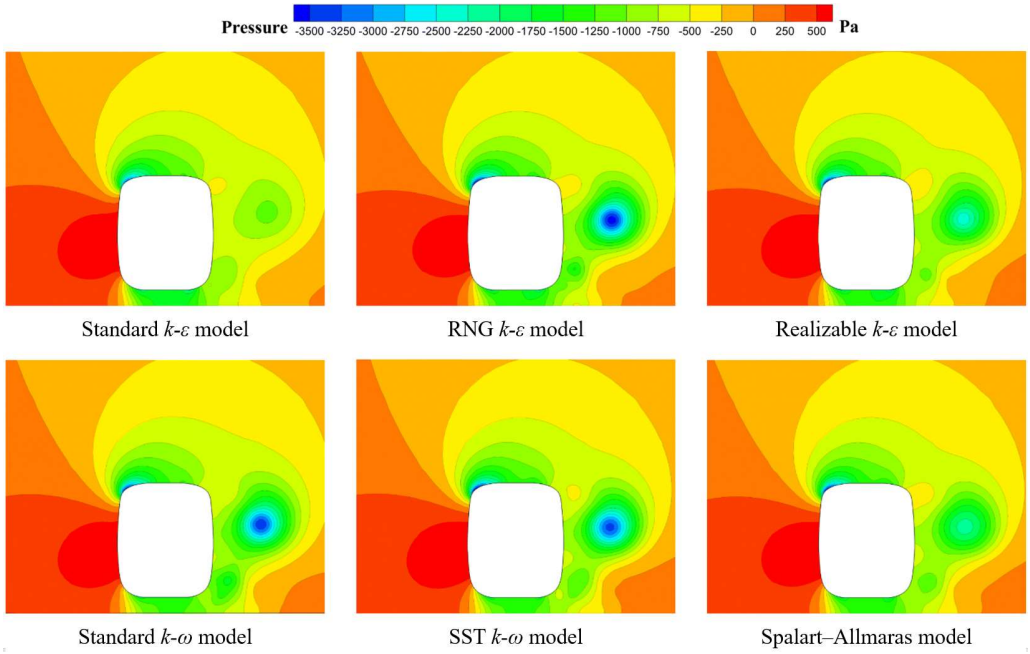


Figure 2-4. Analysis of train surface pressure distribution using various RANS turbulence models carried out by Li *et al.* (2019).

2.2.2 FEM

FEM is a widely employed numerical method for solving fluid-related problems, which is mainly based on the variational principle or the weighted residual method. On this basis, the FEM also borrows the concept of discretization from the finite difference method. The core idea of the FEM is to approximate the solution of PDEs by using simple equations inside a small element. The use of the finite element method for solving partial differential equations may be generally divided into the following steps: (a) formulating the mathematical model for the physical problem, identifying the computational domain, and determining the initial and boundary conditions; (b) utilizing the principle of virtual work to construct the weak-form physical governing equations; (c) discretization of the computational domain as well as the PDEs; (d) unit analysis and establishing the unit stiffness matrix; (e) establishing the global stiffness matrix; (f) treatment of the boundary conditions, and (g) solving the linear equations.

More specifically, physical governing PDEs, such as Eq. (2-29), can be expressed in the following form

$$L(\psi) = f \tag{2-37}$$

where $L(\cdot)$ is the differential operator, and f is the scalar function of the independent variables. By drawing inspiration from the weighted residual method, we may assume ψ' as the approximate solution. By substituting the approximate solution ψ' into the above equation, a residual value R can be obtained

$$R = L(\psi') - f \neq 0 \tag{2-38}$$

Based on the weighted residual method, the weight function W_i is chosen to ensure that the weighted integral of the residual R throughout the entire computational domain is equal to zero, which is as follows

$$\iint_D RW_i dD = \iint_D (L(\psi') - f)W_i dD = 0 \tag{2-39}$$

where D is the computational domain. In FEM, the computational domain is partitioned into a finite number of elements. Mathematically, it can be expressed as follows

$$D = \sum_{i=1}^{m_D} \Delta D_i \tag{2-40}$$

where m_D is the number of elements. In each element, we may assume that the approximate solution of the PDEs takes the following form

$$\psi' = \boldsymbol{N}\boldsymbol{\psi}^{\boldsymbol{e}} \tag{2-41}$$

where $\boldsymbol{N} = [N_i\ N_j\ N_k\ \cdots]$ denotes the shape function, and $\boldsymbol{\psi}^{\boldsymbol{e}} = [\psi_i\ \psi_j\ \psi_k\ \cdots]^T$ denotes

the unknowns to be solved on nodes i, j, k, ⋯. By substituting the computational domain D and the approximate solution ψ' in Eq. (2-39) with Eq. (2-40) and Eq. (2-39), we may obtain

$$\sum_{i=1}^{m_D} \iint_{\Delta D_i} (L(\boldsymbol{N}\boldsymbol{\psi}^{\boldsymbol{e}}) - f) W_i dD = 0 \qquad (2-42)$$

By solving Eq. (2-42) to determine the unknowns on the nodes $\boldsymbol{\psi}^{\boldsymbol{e}}$, one may get the approximate numerical solution for the PDEs. The DGM, which evolved from the FEM, is commonly used in solving fluid mechanics problems.

The difference between the conventional FEM and the DGM is that the FEM requires the shape function $\boldsymbol{N}$ to be continuous between the elements, while the DGM does not. Instead, an additional term called 'numerical flux' is introduced to handle the discontinuities of shape functions at the boundary of the unit in the DGM. However, the principle of DGM will not be elaborated here, which is beyond the scope of the thesis.

DGM is often utilized in fluid mechanics, particularly in RANS simulations, due to its satisfying performance for handling complex boundary conditions (Crivellini *et al.* 2013, Bassi *et al.* 2014, Li *et al.* 2015, Lorini *et al.* 2021).

Crivellini *et al.* (2013) utilized the high-order DGM for solving three-dimensional incompressible RANS equations in simple and complex geometries. Several flow cases were studied, including the flow over a three-dimensional sinusoidal bump in a channel, as shown in Figure 2-5, and the flow past a sphere. The one-equation Spalart-Allmaras model was adopted in the RANS simulation in this work. They also emphasized the difficulties that still exist when using DGM in RANS modeling, such as the requirement for the special treatment of wall boundary conditions.

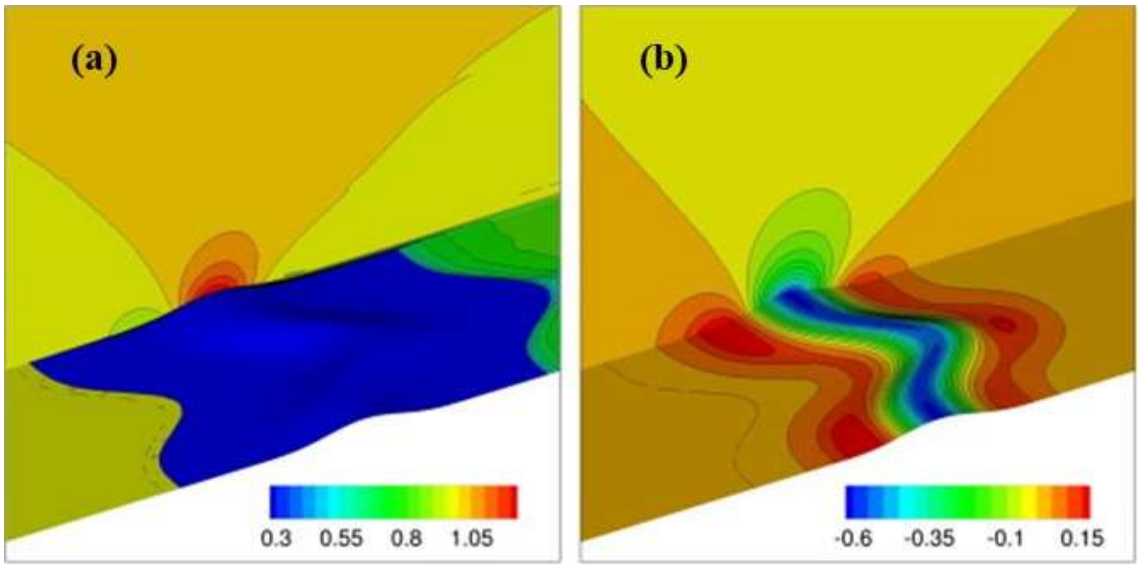


Figure 2-5. The DGM-based simulation of flow over a three-dimensional sinusoidal bump in a channel: **(a)** the velocity contour, and **(b)** the pressure contour. Reprinted from (Crivellini *et al.* 2013).

Jiang *et al.* (2015) discussed the implementation and performance of a p-multigrid DGM solver for steady RANS simulations. This research focused on the practical aspects of the solver, including algorithmic details and computational efficiency. The proposed DGM solver was demonstrated to possess improved convergence capability and accuracy for steady flow problems. Tiberga *et al.* (2020) proposed a novel DGM solver for the incompressible RANS equations coupled with the standard k-ε RANS turbulence model. The solver used an algebraic pressure correction scheme, implicit backward differentiation formulae for time discretization, and the symmetric interior penalty method for diffusive terms.

Currently, however, DGM is not the preferred solution approach for solving the RANS equations because it requires costly high-order polynomial computations inside each unit, resulting in significant computing and storage demands. In contrast, the FVM has superior computing efficiency and is well-suited for large-scale use in practical engineering applications.

2.3 PINN

2.3.1 Overall description

PINN has recently emerged as a new branch of scientific machine learning that solves governing PDEs of physical problems by leveraging the universal approximation and nonlinear expression capabilities of the deep learning algorithm (Scarselli and Tsoi 1998). It is a new approach to solving physical PDEs, which combines one of the hottest research topics today, i.e., deep learning, with classical physics.

According to our traditional understanding of deep learning, which is typically represented by the deep neural network, it ought to be an algorithm that is entirely data-driven. That is, for the majority of deep learning models, data and its corresponding label are essential elements in their training process (Bishop 1994). From this perspective, solving PDEs is not the task in which deep learning algorithms should be skilled, as there is no or only a very small amount of sparse training data available for them to learn from. In other words, the solutions of the physical PDEs in most regions are unknown. However, the primary objective of equation solving is to obtain unknown solutions within the computational domain, thereby leaving deep learning models without any viable labels for training and learning. So, the question is, how does a PINN solve PDEs?

Before answering this question, it is necessary to first give an introduction to the loss function of a neural network. As mentioned earlier, data labels are crucial in the training process of deep learning models. In deep learning models, the loss function is derived from the discrepancies between the model prediction values and the label values. By minimizing the loss function to converge to an infinitesimal value, regression or clustering tasks can be

achieved. Now let us return to that question: how does a PINN solve equations? The answer actually lies in the PINN's utilization of the residuals of the equations as the loss function to supervise model training. The primary issue is how we define the residual of an equation. In this thesis, we define the residual of an equation as a function derived from the equation, which is obtained by subtracting the terms on the right side of the equal sign from the term on the left side. Let us take the continuity equation in the RANS equations, i.e., Eq. (2-13), as an example, its residual shall be

$$R_c = \frac{\partial \bar{u}_i}{\partial x_i} - 0 = \frac{\partial \bar{u}_i}{\partial x_i} \tag{2-43}$$

If we take the momentum equation in the RANS equations, i.e., Eq. (2-18), as another example, its residual shall be

$$R_m = \rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{p}}{\partial x_i} - \frac{\partial}{\partial x_j}\left(\mu \frac{\partial \bar{u}_i}{\partial x_j} - \rho \overline{u_i' u_j'}\right) \tag{2-44}$$

Based on the above equations, it is evident that the residual is a function that depends on temporal and spatial coordinates. For deep neural networks, as long as a point (namely, collocation point) is taken in the computational domain, the prediction value of the neural network at that point can be outputted. If necessary, the partial derivatives at the collocation point can also be calculated based on AD or numerical differentiation (Baydin *et al.* 2018, Chiu *et al.* 2022). The prediction values of the physical quantities and their partial derivatives at these collocation points constitute the residuals of PDEs, which serve as the loss function of the PINN.

By employing the gradient descent methods to minimize the loss function, the residuals of the PDEs are minimized and approximated to zero, which is equivalent to the roughly equal

relationship between the left and right sides of the equal sign.

In the steady RANS simulations, the physical governing PDEs within the computational domain are the Reynolds-averaged continuity and momentum equations, while the physical governing PDEs on the domain boundary are the boundary conditions. Minimizing the loss function results in weak satisfaction of the Reynolds-averaged continuity and momentum equations at the collocation points within the computational domain. In addition, the boundary conditions are also weakly satisfied at the collocation points on the domain boundaries. Equivalently, this yields an approximate solution to the RANS equations.

One benefit of PINN as a PDE solver should be highlighted in comparison to conventional methods such as the FVM and FEM. Given that PINN remains fundamentally a deep neural network, it is still possible to use the conventional data-driven supervised learning strategy to assist in PINN model training.

This is crucial because, in engineering practices, the solutions within the domain are not completely unknown since sensors are often deployed to measure real-time data on sparse measurement points. Taking the RANS simulation of the building outdoor wind field as an example, the time-averaged wind velocity and pressure data obtained from the sensors installed on the building surfaces can be fully embedded as label data in the training process of a PINN, thereby giving guidance to the solution in the entire domain (Pu *et al.* 2021, Qin *et al.* 2023, Rui *et al.* 2023, Liu *et al.* 2024).

Briefly, PINNs have the capability to act as a general PDE solution approximator, which can embed the knowledge of physical laws in the model training process. The process of utilizing a PINN to solve the RANS equations can be summarized as follows: (a) utilizing a deep neural network to establish the relationship between spatial coordinates (input) and the

flow characteristics to be solved (output); (b) computing the residuals of the continuity and momentum equations using forward propagation and AD for the collocation points inside the domain; (c) computing the residuals of the Dirichlet and Neumann boundary conditions using forward propagation and AD for the collocation points on the domain boundaries; (d) computing the labeled data loss if applicable; (e) forming the loss function using the weighted sum of various equation residuals, and data loss, if any; (f) error backpropagation and model parameter optimization; and (g) error evaluation.

2.3.2 Origins of PINN in fluid mechanics

The PINN framework was first conceptualized by Raissi *et al.* (2019), which has aroused great interest recently across multiple research areas (Fang and Zhan 2019, Liu *et al.* 2020, Yucesan and Viana 2020, Chen *et al.* 2022). PINN has been used to solve PDEs (Mao *et al.* 2020), including Schrodinger equations and linear Poisson problems, by embedding the residuals of physical governing equations, initial conditions, and boundary conditions into the total loss function of the neural network (Pang *et al.* 2019, Yuan *et al.* 2022). More specifically, the approximation of the PDE solution is achieved by summing the residuals of these physical constraints and then shrinking it towards 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 (Robbins and Monro 1951, Kingma 2014).

Fluid mechanics is one of the research directions where PINN has been able to showcase its capabilities since the motion of fluids can be described by governing PDEs. In other words, since the main task of fluid mechanics is to solve the second-order nonlinear PDE, i.e., the NS equations, or its variants, various research investigations on the PINN applications to fluid mechanics have been carried out in recent years (Sun *et al.* 2021, Cai *et al.* 2022). Research on

PINNs in fluid mechanics is definitely still in its early stages. Following is a concise review of the most recent cutting-edge research on the PINN application in fluid mechanics.

In fact, a case study of PINN's application to fluid mechanics was included in the paper where it was first presented. Raissi *et al.* (2019) managed to solve an inverse problem of identifying unknown parameters in the NS equations using a PINN, as shown in Figure 2-6.

The definitions of forward and inverse problems in this thesis need to be clarified here. The forward problem is one in which we need to solve for the system state or evolution given the knowledge of the physical laws governing a specific system, while the process of determining the system's unknown parameters or initial conditions from observations is known as the inverse problem. The values of two parameters in the NS equations were estimated by Raissi *et al.* (2019) based on massive observation data. Two tests were carried out, one with noisy observation data and the other without, and both achieved good estimates of the values of the unknown parameters. The results demonstrated that the PINN can accurately identify the unknown parameters with extremely high accuracy even if the training data is interfered with by noise.

Actually, PINN rarely involved turbulence problems in its early development. Instead, researchers mostly employed it to simulate laminar flows with relatively low Reynolds numbers (Bai *et al.* 2020, Rao *et al.* 2020, Arzani *et al.* 2021, Biswas and Anand 2023, Hu and McDaniel 2023). This is due to the fact that laminar flows are more stable and orderly than turbulent flows, which just necessitate solving the NS equations and do not involve the process of turbulence modeling. Rao *et al.* (2020) solved a forward laminar flow problem using a PINN under the mixed-variable scheme they proposed in their paper. The flow cases this research investigated were two cylinder flows, with Reynolds numbers of 5 and 20 respectively. A

mixed-variable scheme for PINN was proposed in their research, which greatly improved the performance of the PINN in laminar flow simulation. In contrast to previous studies, this approach reduced the training difficulty of a PINN by splitting the momentum equation in the NS equations into two PDEs containing only first-order derivatives. Figure 2-7 compares the velocity and pressure contours in a steady cylinder flow when different PDE solvers are adopted. The results indicated the feasibility of the PINN framework in solving forward fluid problems and established a solid foundation for fluid dynamitists to use the PINN to address their problems.

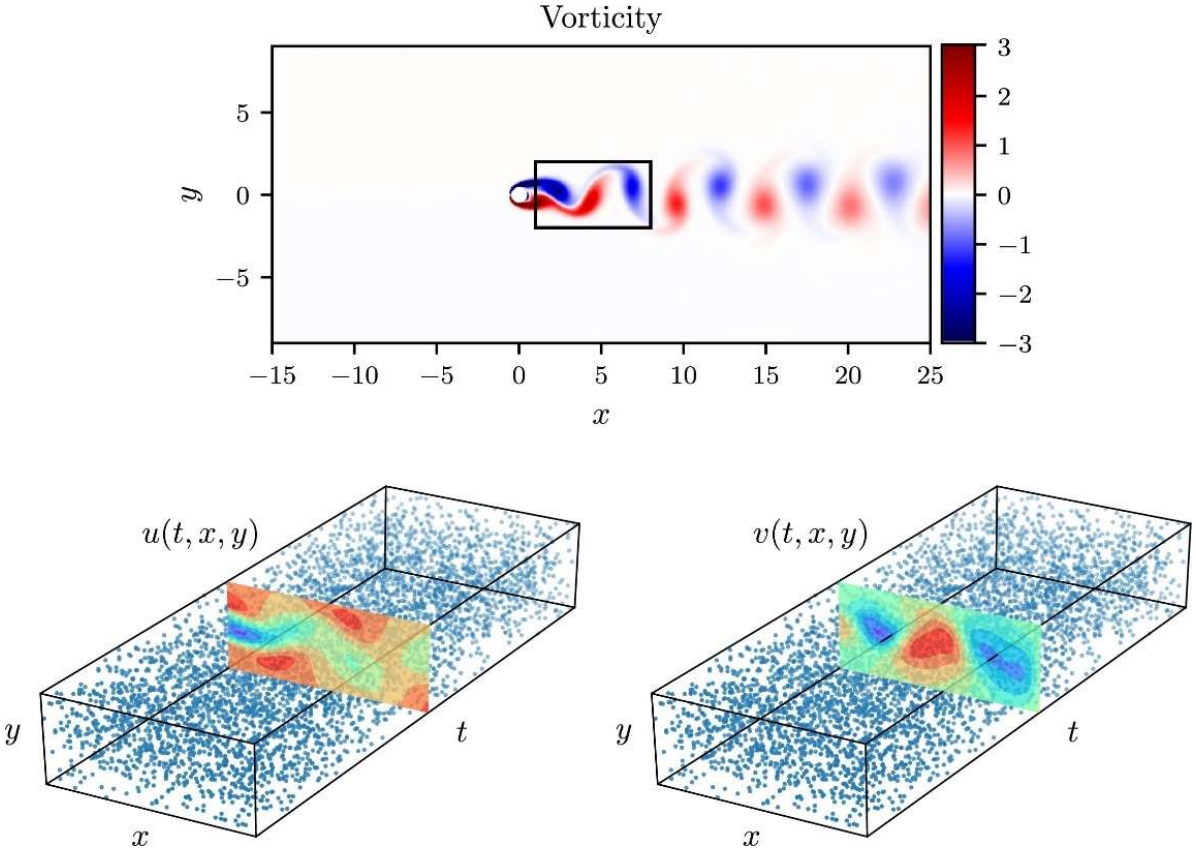


Figure 2-6. Identification of unknown parameters in the NS equations using a PINN by Raissi *et al.* (2019). The figures show the detailed spatiotemporal distributions of the data points used for training.

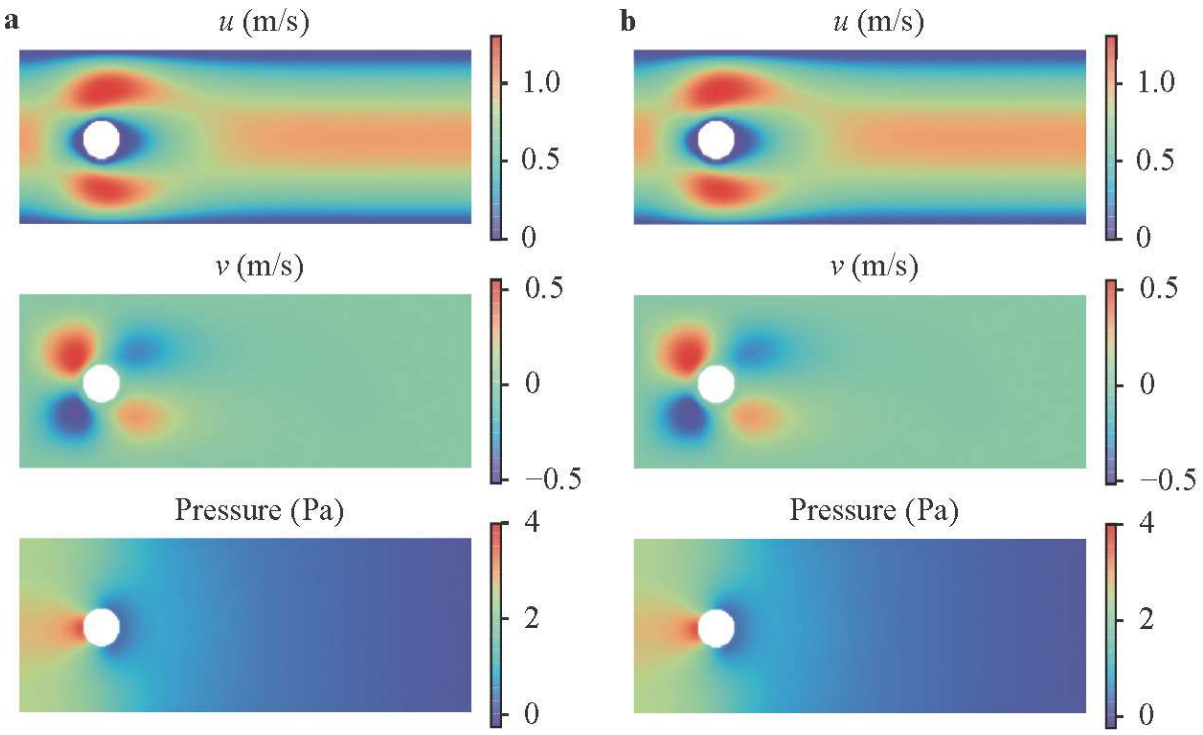


Figure 2-7. Comparison of the velocity and pressure contours in a steady cylinder flow when different PDE solvers are adopted: **(a)** *Ansys Fluent*, and **(b)** PINN adopting the mixed-variable scheme. Reprinted from (Rao *et al.* 2020).

Arzani *et al.* (2021) simulated three blood flows with Reynolds numbers of 150, 320, and 320 respectively, which adopted the steady incompressible NS equations as the fluid governing equations. PINN was used as the PDE solver for fluid simulation in this research. Since the boundary conditions remain uncertain when simulating blood flow, the research team included sparse measurement data points in the PINN's training process to guide the PDE solving process. This study leveraged the NS equations and sparse velocity measurements to accurately quantify wall shear stress (WSS), even without full knowledge of boundary conditions. The results in Figure 2-8 confirm the above advantage of utilizing PINN to solve PDEs. That is, conventional wisdom holds that solving PDEs without boundary conditions is nearly impossible. However, PINN offers an alternative approach, which makes use of sparse

solutions inside the domain to guide equation solving and compensate for the absence of physical information.

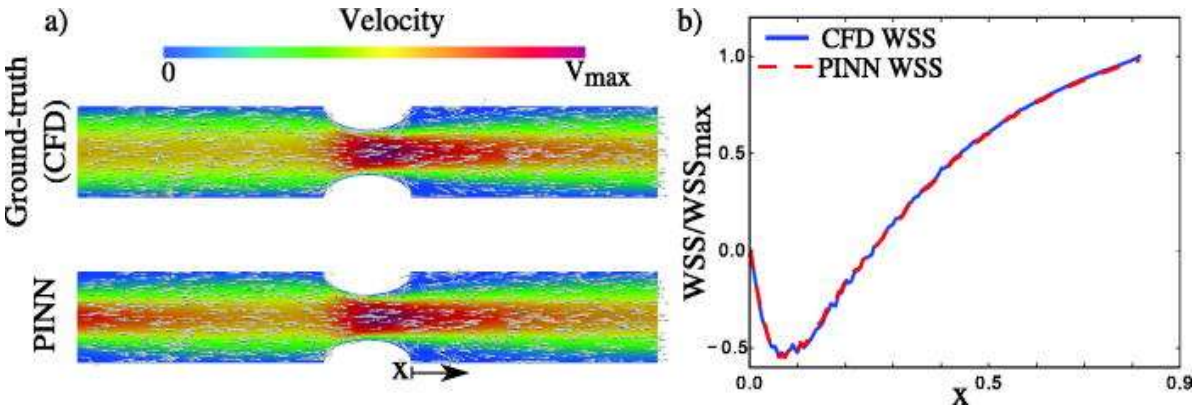


Figure 2-8. Comparison of the CFD results with the PINN results in a two-dimensional blood flow: **(a)** the velocity contour, and **(b)** the WSS. Reprinted from (Arzani *et al.* 2021).

Sun *et al.* (2021) proposed a physics-informed deep learning framework for simulating a stationary, inviscid, and incompressible cylinder flow using a PINN. Instead of feeding the neural network a number of labeled training data, they embedded the known physical knowledge in the neural network training process to increase prediction accuracy and relieve the stringent constraints on massive data. The NS equations were used as the physical governing equations. Their results, as illustrated in Figure 2-9, also demonstrated the feasibility of applying the PINN framework to solving fluid-related problems. The Taylor–Green vortex problem was also used to demonstrate the robustness and efficiency of the PINN.

Huang *et al.* (2022) proposed an immersed boundary method for PINN-based laminar flow simulation, as shown in Figure 2-10. The proposed method introduced two additional loss penalties in the total loss of a PINN to simulate the no-slip condition at the fluid–solid interface. The author verified the effectiveness of the proposed immersed boundary method using a case study of circular cylinder flow. The proposed method was highly successful in capturing the

vortices on the leeward side of the cylinder, and the results of the proposed method demonstrated good consistency with CFD results.

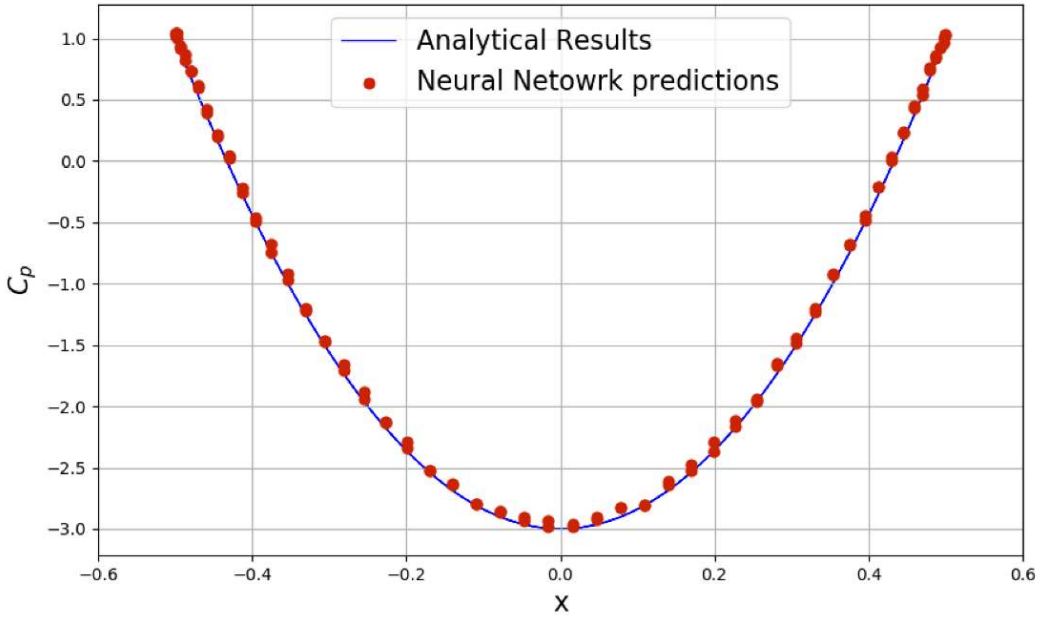


Figure 2-9. Comparison of the pressure coefficients on the cylinder surface between the PINN's results and the analytical results. Reprinted from (Sun *et al.* 2021).

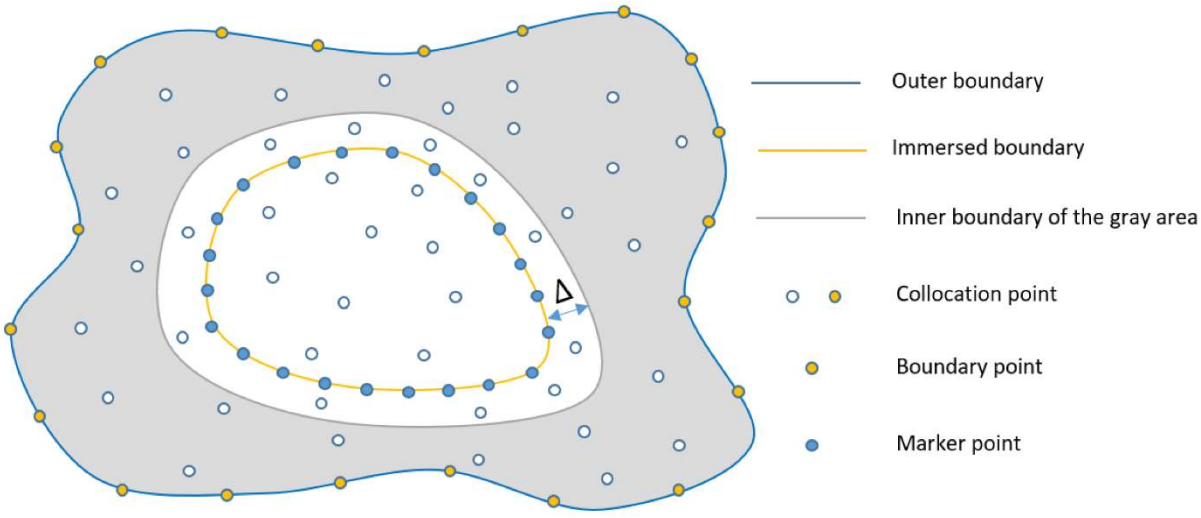


Figure 2-10. The immersed boundary method for PINN-based laminar flow simulation proposed by Huang *et al.* (2022).

Almajid and Abu-Al-Saud (2022) simulated a porous media flow under the PINN framework, as shown in Figure 2-11. They used a PINN to solve the Buckley-Leverett problem in this study, which took into account both labeled observation data and physical knowledge from the fluid flow. According to their findings, the PINN could capture the solution's general trend even in the absence of observation data, but on the other hand, observation data could greatly increase the solution's precision and accuracy.

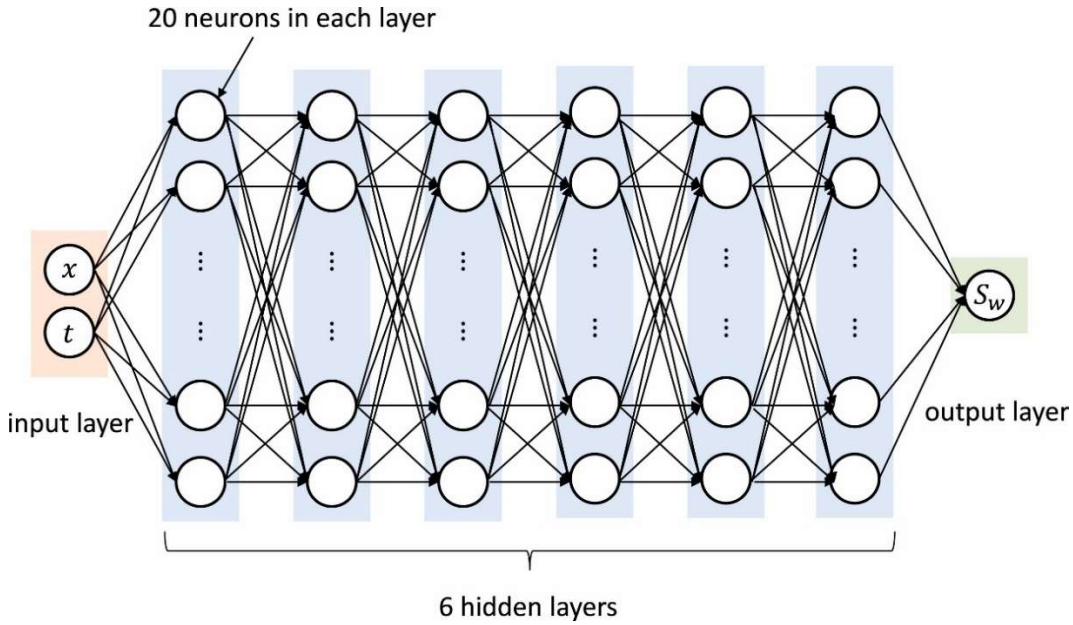


Figure 2-11. The FCNN structure adopted by Almajid and Abu-Al-Saud (2022) for porous media flow simulation.

The above research mainly focuses on laminar flow with low Reynolds numbers. In contrast to turbulent flows, the laminar flow has a more straightforward structure, and even if there are eddies in the flow field, the scale of the eddies will not change significantly. Therefore, the NS equations are the dominant governing PDEs in PINN-based laminar flow simulation. The following benefits of using PINN as a brand-new machine learning solver for solving the NS equations are concluded here. Firstly, it is a meshless approach, therefore no grid-related

issues will arise.. Secondly, a data-driven strategy can significantly increase the accuracy of solutions owing to the PINN's special ability to incorporate labeled data (on-site measurement) in the model-training process (Riel *et al.* 2021, Choi *et al.* 2022). That is, similar to physical governing equations, the residuals between the PINN prediction and the labeled data can be incorporated into the total loss. As a result, the final solution to the NS equations derived from a data-embedded PINN may be viewed as a comprehensive outcome that incorporates both physical and data information.

2.3.3 PINN in RANS simulations

Due to the complexity of turbulence, it was not the initial focus of PINN's research in fluid mechanics. Numerous CFD numerical methods are available to simulate turbulent behaviors. In addition to RANS, popular methods include LES and detached eddy simulation (DES). Studies have shown that some of these methods can be implemented within the PINN framework as well and they have achieved satisfying results (Yang *et al.* 2019, von Saldern *et al.* 2022, Tian *et al.* 2023, Maejima *et al.* 2024). However, considering that the research focus of this thesis is on the RANS method, methods other than RANS will not be elaborated here. Interested readers may refer to the aforementioned citations.

To the best of the author's knowledge, the earliest use of PINN in RANS-related research can be traced back to 2020, which was carried out by Luo *et al.* (2020). The main objective of this work was to solve an inverse problem. More specifically, the five empirical constants in the standard k-ε turbulence model were to be identified, given that the fluid velocity and pressure were known. The study also pointed out that the value of empirical constants in turbulence models will to some extent affect the accuracy of RANS simulation results, and different values are applicable under different flow conditions. This is essentially in line with

how the model applicability problem is described in Chapter 1. The DNS data is used to validate the results of the PINN simulation. The results, as illustrated in Figure 2-12, proved the reliability and accuracy of utilizing the PINN framework in solving inverse problems in RANS simulations.

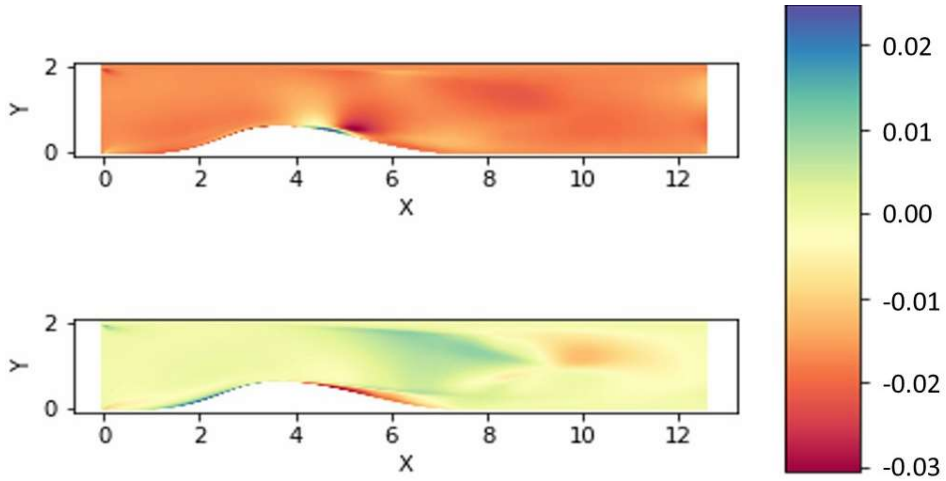


Figure 2-12. Error maps for the *y*-direction fluid velocity: **(top)** errors between the DNS and RANS (default values) results, and **(bottom)** errors between the DNS and RANS (PINN-inferred values) results. Reprinted from (Luo *et al.* 2020).

One of the most representative research works on utilizing a PINN to solve the forward problem involving the RANS equations was carried out by Eivazi *et al.* (2022). It is worth mentioning that, no additional PDEs were required to describe the Reynolds stress terms to close the RANS equations when PINN was adopted as the PDE solver in their work, which was in contrast to turbulence modelling in conventional CFD methods. Instead, the fluid velocity, pressure, and Reynolds stresses on the domain boundaries were used as the labeled data to assist in the PINN's training. Or in other words, the data information on the boundaries as well as the physical equations were utilized to reconstruct the turbulence characteristics inside the domain. When compared to conventional methods, this approach offers both benefits

and drawbacks. It is important to recognize that the task of turbulence modeling has been eliminated, which can help to enhance computing efficiency and partially alleviate the issue of poor universality of turbulence models. The drawback is that this approach requires an increasing amount of information on the domain boundaries, and the choice of the computational domain has an immediate impact on the simulation result. How to delimit the size of the computational domain has become another challenge. Four case studies were used to demonstrate the feasibility of the proposed strategy, including the well-known case of the NACA4412 airfoil and the flow past periodic hills, as shown in Figure 2-13.

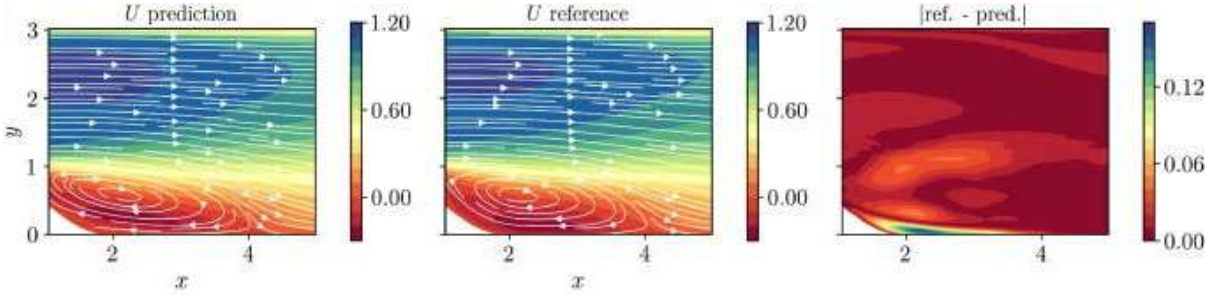


Figure 2-13. RANS simulation of the flow past periodic hills using PINN carried out by Eivazi *et al.* (2022).

Harmening *et al.* (2024) investigated the influence of the FCNN configuration on the performance of a PINN when it is used to simulate circular and square cylinder flows. In their research, the mixed-variable scheme proposed by Rao *et al.* (2020) was adopted to modify the RANS equations to facilitate the training and convergence of the PINN models. The aforementioned Prandtl mixing-length model was adopted as the RANS turbulence model to describe Reynolds stress. In addition, PINN's training was not supported by any known labeled data, so, in this research, PINN turns into a PDE solver with the same functionality as CFD methods. Their results, as illustrated in Figure 2-14, demonstrated that the PINN simulation results are highly sensitive to the structure of the FCNN. Based on their findings, it can be

concluded that FCNNs with too deep or too wide structures are not conducive to PINN-based RANS simulations.

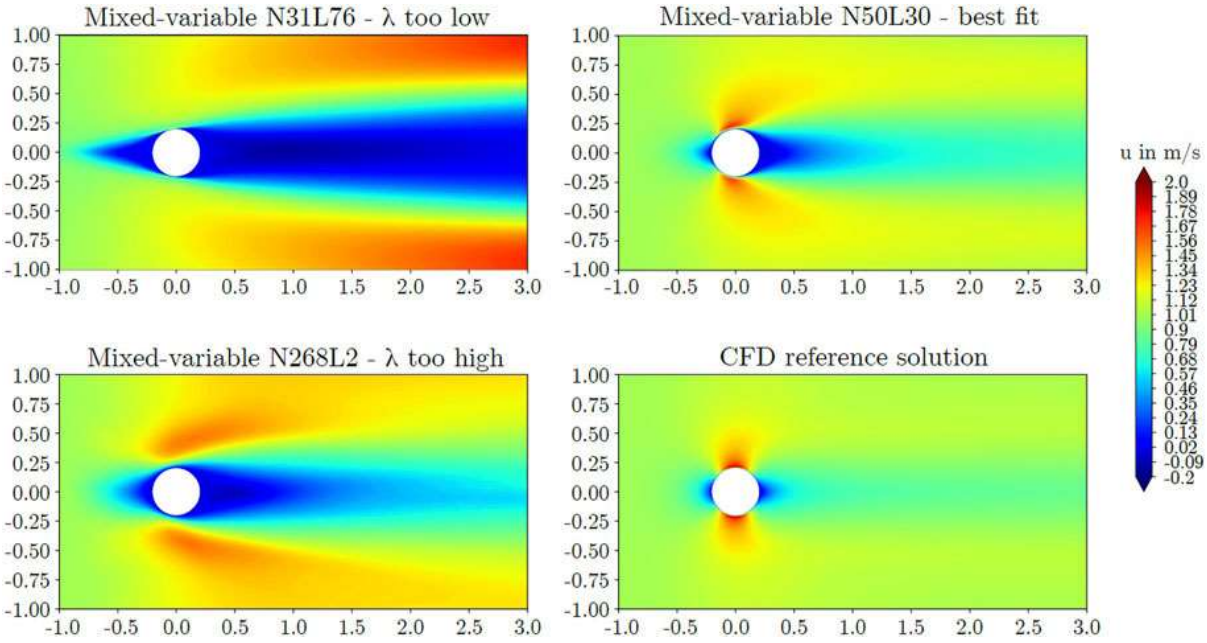


Figure 2-14. Comparison of the PINN-based RANS simulation results when different FCNN structures were adopted. Reprinted from (Harmening *et al.* 2024).

In conclusion, the basic principle of the PINN-based RANS simulation is to approximate the solution of the RANS equations by training a neural network while minimizing its physics-based loss function, which includes the residuals of RANS equations and boundary conditions at the collocation points. In the above-mentioned cases, PINNs show satisfying performances when dealing with the forward problems involving the RANS equations. Generally speaking, the grid-based CFD methods such as the FVM and FEM are more exposed in the textbooks of numerical analysis. What sets it apart from conventional CFD methods is that PINN can serve as a physics-informed data-driven solver for fluid simulation since it can integrate not only the physical knowledge but also the measurement information into the neural network training process, making it a fusion method in a more profound sense.

2.3.4 Limitations of PINN-based RANS simulations

As summarized in Figure 1-2, currently, PINN-based RANS simulations mainly face three key challenges, which are the limited applicability of the RANS turbulence model, the weak convergence performance of the PINN, as well as the inadequate nonlinear expression and feature learning capabilities of the PINN.

As mentioned before, the applicability issue is not a new problem, which also appears in CFD-based RANS simulations. While PINN has made great achievements in the field of fluid mechanics, researchers have also reported this issue when they use PINN as the PDE solver. For example, Pioch *et al.* (2023) validated the applicability of four different RANS turbulence models under the PINN framework using a backward-facing step flow. The turbulence model they adopted included the standard k-ω model (Wilcox 1988) and the Prandtl mixing-length model. In their case study, the Reynolds number reached up to 5100. Figure 2-14 compares the time-averaged velocity contours of the backward-facing step flow when the standard k-ω model and the Prandtl mixing-length model are adopted in RANS simulations, respectively. From the figure, it can be observed that there are significant differences between the results of the two in the location of the reattached flow and the sizes of the recirculation vortex and the corner vortex. The results show that, as with CFD methods, when PINN is utilized to solve the RANS equations, the turbulence model will significantly influence the simulation results. How to determine the turbulence model that is applicable to the flow conditions before simulation remains a challenge in PINN-based RANS simulations. However, this research also mentioned that after embedding adequate sparse labeled data for supervised learning, the applicability issue can be effectively alleviated.

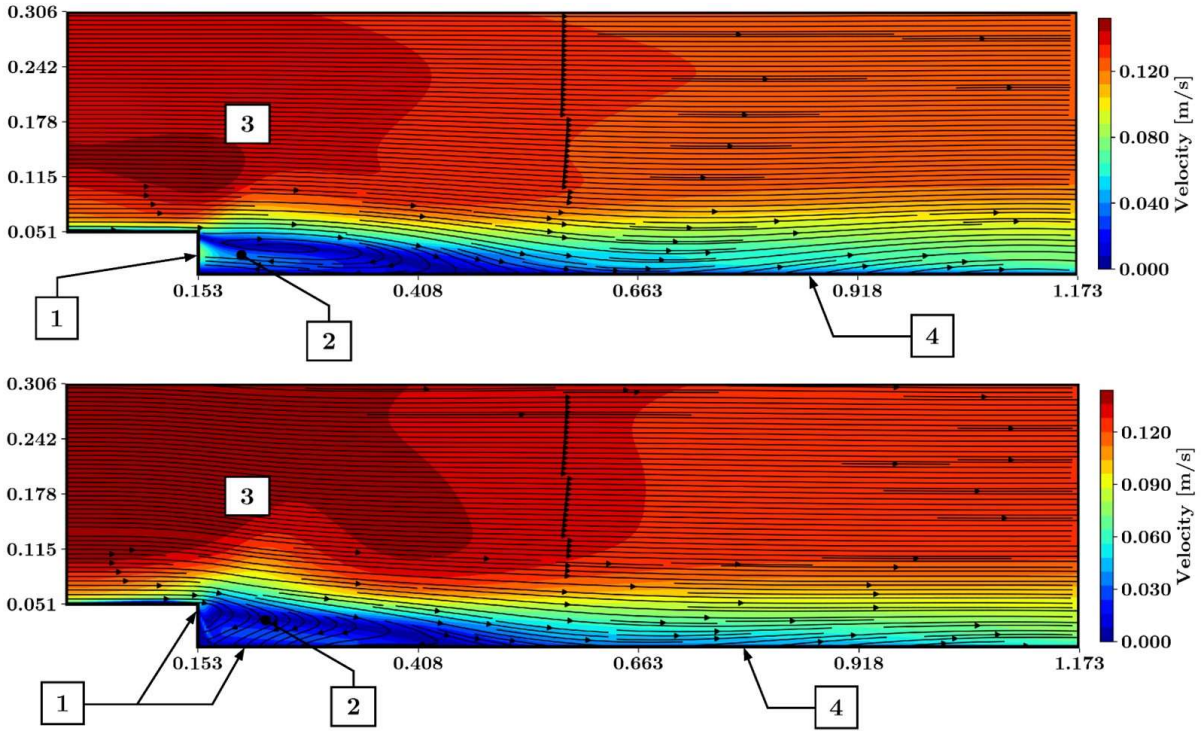


Figure 2-15. Comparison of the velocity contours of the backward-facing step flow when different RANS turbulence models are adopted: **(top)** the standard k-ω model, and **(bottom)** the Prandtl mixing-length model. Reprinted from (Pioch *et al.* 2023).

The weak convergence performance of the PINN has long been criticized by researchers. There are multiple factors contributing to its poor convergence performance. On the one hand, the loss balance issue may lead to an unreasonable emphasis on certain loss components in PINN's training process (Bischof and Kraus 2021, Yang and Wang 2022, Heldmann *et al.* 2023, Wang *et al.* 2024). On the other hand, using AD to calculate high-order differential terms is a time-consuming procedure, which further affects its convergence performance (Sharma and Shankar 2022, Yuan *et al.* 2022). Undoubtedly, there are many other factors that could impact PINN's convergence performance, and interested readers are welcome to conduct further investigation on this issue. Xiang *et al.* (2022) proposed a self-adaptive loss balanced strategy for training PINNs, which is named lbPINN. Instead of treating the weight coefficient before

each loss component as a constant value, lbPINN updates these weight coefficients based on maximum likelihood estimation after each training iteration. Results showed that lbPINN performs better than the default PINN when solving forward problems involving the Poisson equation, as shown in Figure 2-15.

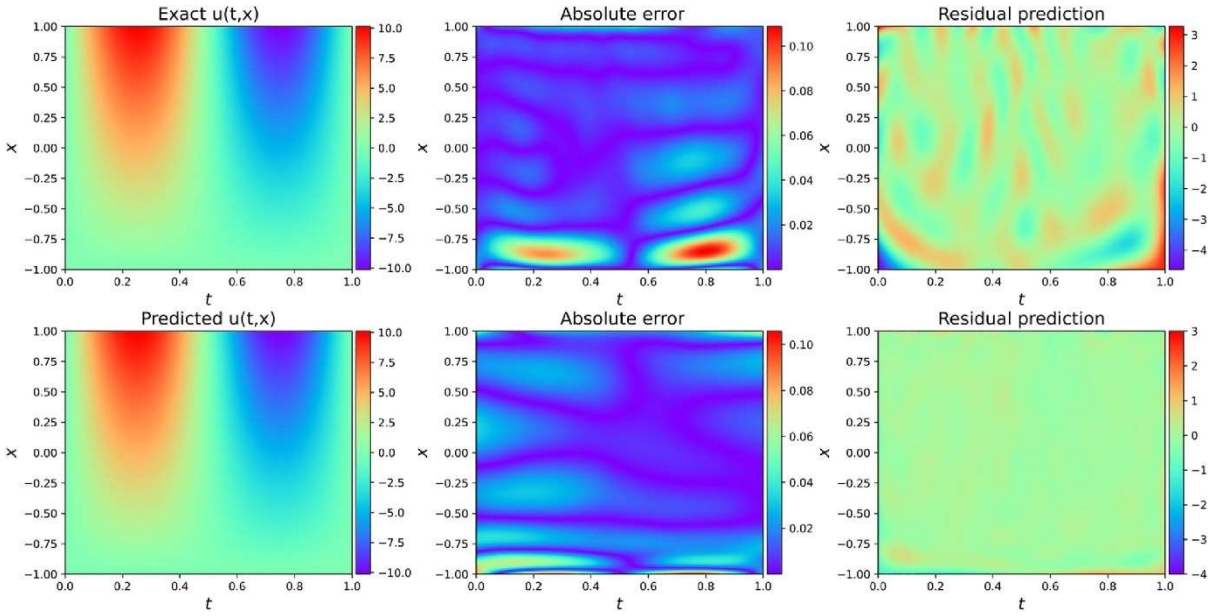


Figure 2-16. Comparison of the PINN's solutions to the time-dependent Poisson equation when different loss balance strategies are adopted: **(top)** the default PINN, and **(bottom)** the proposed lbPINN.

The nonlinear expression and feature learning capabilities of a PINN are limited by its neural network backbone. Although the universal approximation theorem tells us that theoretically, as long as there are enough hidden neurons, an FCNN can learn to fit any nonlinear function, in practical operation, many research studies have reported the frequency principle phenomenon when training an FCNN (Cao *et al.* 2019, Rahaman *et al.* 2019, Xu *et al.* 2024). That is, compared to the low-frequency components, the high-frequency ones are more difficult for a neural network to learn, which is also reflected in PINNs. Ye *et al.* (2024)

introduced the Fourier features embedding strategy in PINN's training process, which significantly reduced the adverse effects caused by the frequency principle. Although RANS simulations do not involve frequency issues on a time scale, similar to the training process in Figure 1-1, PINN prefers to prioritize providing a crude drawing of the time-averaged solution when solving the RANS equations, which hinders the acquisition of the accurate solutions for areas with substantial variations in gradients. Therefore, the limited nonlinear expression and feature learning capabilities of a PINN have become one of the obstacles that restrict the further developments of PINN-based RANS simulations.

2.4 Summary

This chapter begins by introducing the NS equations and explains how Reynolds averaging can be performed on the NS equations to derive the RANS equations. Subsequently, the RANS turbulence models are introduced, and the conventional methods for solving the RANS equations are presented. Next, this chapter introduces PINN as a novel PDE solver and elaborates on the origins of PINN in fluid mechanics, while also reviewing its applications in RANS simulations. The end of this chapter summarizes the factors that may limit the further developments of PINN-based RANS simulation.

To conclude, turbulence simulation is mainly based on CFD methods nowadays. At the same time, a new force emerges, which is the machine learning-based PDE solvers represented by PINN. They can not only concern physical constraints in the modeling process but also fuse labeled data information, which shows the potential to become one of the mainstream methods for solving forward and inverse problems involving the RANS equations. However, such technology is still in its infancy right now, and there are issues that urgently need to be solved,

which calls for further research. Accordingly, this thesis is conducted on this basis, and the uses of various physics-based machine learning algorithms in turbulence simulation are examined, which aims to work for the advancement of the research on PINN-based RANS simulations.

CHAPTER 3

DYNAMIC PRIORITIZATION

3.1 Foreword

Convergence is a key issue in training a neural network, which refers to the process in which a model gradually reduces the errors between the predictions and target values, and stabilizes its performance by learning from training data. As is well known, the total loss of a neural network represents the mismatch between its prediction and the target, which needs to be converted to the minimum with the aid of gradient descent algorithms (such as Adam and SGD) to reach the goal of classification or regression. Similarly, the convergence performance of the neural network backbone has a significant impact on the accuracy of the results of the PINN-based RANS simulation. However, there exists a common problem when PINN is utilized for RANS simulations. That is, the total loss of a PINN used to solve the RANS equations is generally accumulated by more than ten to twenty independent loss components, including the residuals of the physical governing equations and various kinds of boundary conditions, which may lead to the imbalance issue between each loss component.

In fact, the loss balance issue had already shown up prior to the birth of PINN in a branch of machine learning, i.e., multitask learning. Dealing with the conflict and balance issue between the loss components of different tasks during the training process is crucial in multitask learning. The gradient of parameter optimization varies during error backpropagation, which may lead to different convergence speeds for different tasks. Some tasks may have been

well-trained, while others are still far from being well-optimized. Therefore, it is believed that there exists the theoretical possibility to achieve a faster convergence speed and more accurate predictions by dynamically balancing these tasks throughout the training process instead of simply assigning them with fixed weights. It should be explained that the weight here refers to the weight coefficient before each loss component, and the total loss of the neural network is the weighted sum of each loss component. Some have been trying to employ self-adaptive loss balance strategies in order to balance these loss components. A number of correlative studies have been carried out in the field of multitask learning (Kendall *et al.* 2018, Sener and Koltun 2018).

PINN has been trying to incorporate some of these ideas from multitask learning (Xiang *et al.* 2022). A dynamic prioritization loss balance strategy for the PINN framework will be proposed in this chapter, which disjoins the total loss of a PINN and recombines the loss components to form physics-based tasks. On this basis, an evaluation index will be established for each physics-based task to assess its past performance, and tasks with lagging performance will be assigned with high weights to be emphasized in the subsequent training process. Meanwhile, tasks that perform well will be given less attention in the ensuing training process. It is also worth mentioning that the proposed dynamic prioritization physics-informed neural network (dpPINN) incorporates the labeled training data into the dynamic loss balance process.

Two forward problems involving the RANS equations will be solved using the proposed dpPINN models in this chapter. One is the square cylinder flow, while the other is the outdoor flow around a building. Based on the sparse experimental measurement (i.e., fluid velocity data), dpPINN models will be fully trained to simulate the fluid flow. Other experimental data will be used to validate the feasibility of the proposed model.

3.2 Basic Principles

3.2.1 Structure of PINN

The basic structure and principle of the PINN model adopted in this chapter will be described in detail here. By directly embedding physical equations, initial and boundary conditions, and measurement data into the total loss function for training a neural network, PINN is proven to be a charming physics-informed data-driven approach for solving forward and inverse physical problems involving differential equations. The residuals of the physical constraints tend to converge toward zero during PINN's 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 RANS simulation of the three-dimensional fluid flow is depicted in Figure 3-1. The left part of the PINN is an FCNN which 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, AD is applied to calculate the gradients of the outputs with respect to the inputs, which also plays a key role in training the neural network (Baydin *et al.* 2018). The right part of the PINN is the total loss, which takes the following form

$$L = w_f L_f + w_b L_b + w_d L_d \tag{3-1}$$

where

$$L_f = \frac{1}{N_f} \sum_{n=1}^{N_f} \sum_{i=1}^{4} |f_i^n|^2 \tag{3-2}$$

$$L_b = \frac{1}{N_{nb}} \sum_{i=1}^{N_{nb}} \left| r_{nb}^i \right|^2 + \frac{1}{N_{db}} \sum_{i=1}^{N_{db}} \left| r_{db}^i \right|^2 \qquad (3-3)$$

$$L_d = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| r_d^i \right|^2 \qquad (3-4)$$

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 data constraints, 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 ith governing equation in Figure 3-1; r_{nb}^i, r_{db}^i, and r_d^i are the residuals for the Neumann boundary, Dirichlet boundary, and data constraints, 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 data constraints, respectively. Once the total loss is formed, the error backpropagation algorithm is then employed for calculating the gradients of the parameters to be optimized and updating their values.

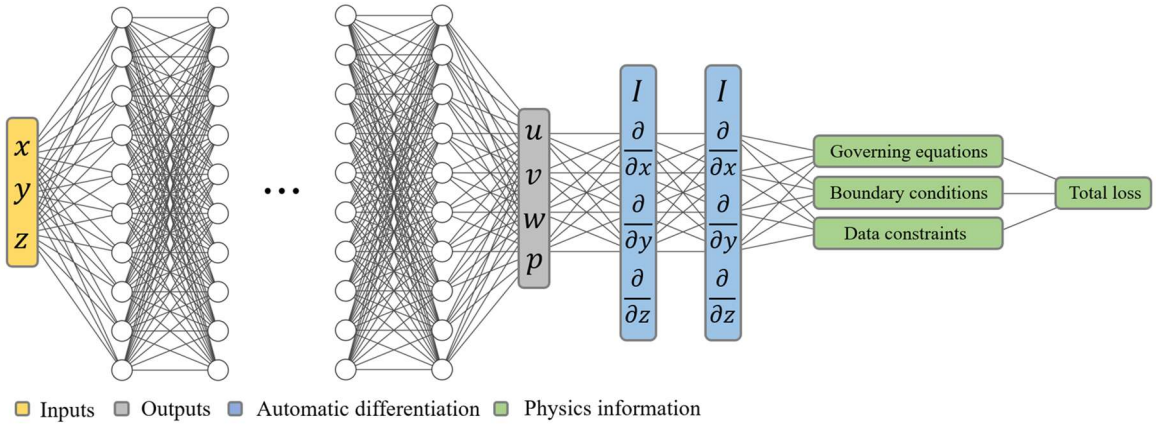


Figure 3-1. Schematic diagram of PINN for the RANS simulation of the three-dimensional fluid flow.

3.2.2 Dynamic prioritization loss balance strategy

How to balance the loss terms in the total loss function during the training process of PINN has become a challenging issue, as illustrated by Xiang *et al.* (2022). The imbalance between different loss terms may significantly diminish the convergence rate and computational efficiency in the training of a PINN. To alleviate this problem, we rewrite the total loss in the following form:

$$L = L_f + w_u L_u + w_v L_v + w_w L_w + w_p L_p \tag{3-5}$$

where

$$L_u = \sum_{n \in \mathbf{U}_r} \frac{1}{N_n} \sum_{i=1}^{N_n} \left| r_n^i \right|^2 \tag{3-6}$$

$$L_v = \sum_{n \in \mathbf{V}_r} \frac{1}{N_n} \sum_{i=1}^{N_n} \left| r_n^i \right|^2 \tag{3-7}$$

$$L_w = \sum_{n \in \mathbf{W}_r} \frac{1}{N_n} \sum_{i=1}^{N_n} \left| r_n^i \right|^2 \tag{3-8}$$

$$L_p = \sum_{n \in \mathbf{P}_r} \frac{1}{N_n} \sum_{i=1}^{N_n} \left| r_n^i \right|^2 \tag{3-9}$$

Unlike the original expression in Eq. (3-1), 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. $\mathbf{U}_r$, $\mathbf{V}_r$,

$\mathbf{W}_r$, and $\mathbf{P}_r$ represent the sets which are composed of formula numbers related to u, v, w, and p. The details will be further explained in the following validation case. r_n^i is the residual of Eq. (3-n) on the ith collocation point, and N_n is the number of collocation points used to calculate the residual of Eq. (3-n). It can be seen that the residuals of various boundary conditions and data constraints form the loss terms L_b and L_d in Eq. (3-1) in the default PINN. However, they are reorganized in the proposed dpPINN to form the u, v, w, and p-related loss terms. Such a reconfiguration enables us to balance each loss term in the following way:

$$w_i = \left[\frac{k_i}{\min_j(k_j)}\right]^{\gamma}, i, j = u, v, w, p \tag{3-10}$$

$$k_i = \frac{\left\|\boldsymbol{U}_{\boldsymbol{T}_i} - \widetilde{\boldsymbol{U}}_{\boldsymbol{T}_i}\right\|_2}{\left\|\boldsymbol{U}_{\boldsymbol{T}_i}\right\|_2}, \qquad i = u, v, w, p \tag{3-11}$$

where γ is a newly introduced hyperparameter that affects the weight balance between different loss terms; $\|\cdot\|_2$ denotes the ℓ_2-norm; $\boldsymbol{U}_{\boldsymbol{T}_i}$ denotes the vector of the labelled data on the training points; and $\widetilde{\boldsymbol{U}}_{\boldsymbol{T}_i}$ denotes the vector of PINN predictions on the training points. Such a setting means that the relative errors of u, v, w, and p between the PINN predictions and the reference data are calculated at the training points after certain iterations. A flowchart is depicted to further illustrate the updating mechanism of the dpPINN's weighting coefficients, as shown in Figure 3-2. 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 the dpPINN predictions and the reference data on the training points, i.e., Eq. (3-11), are calculated. For example, in Figure 3-2, 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. (3-10) 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.

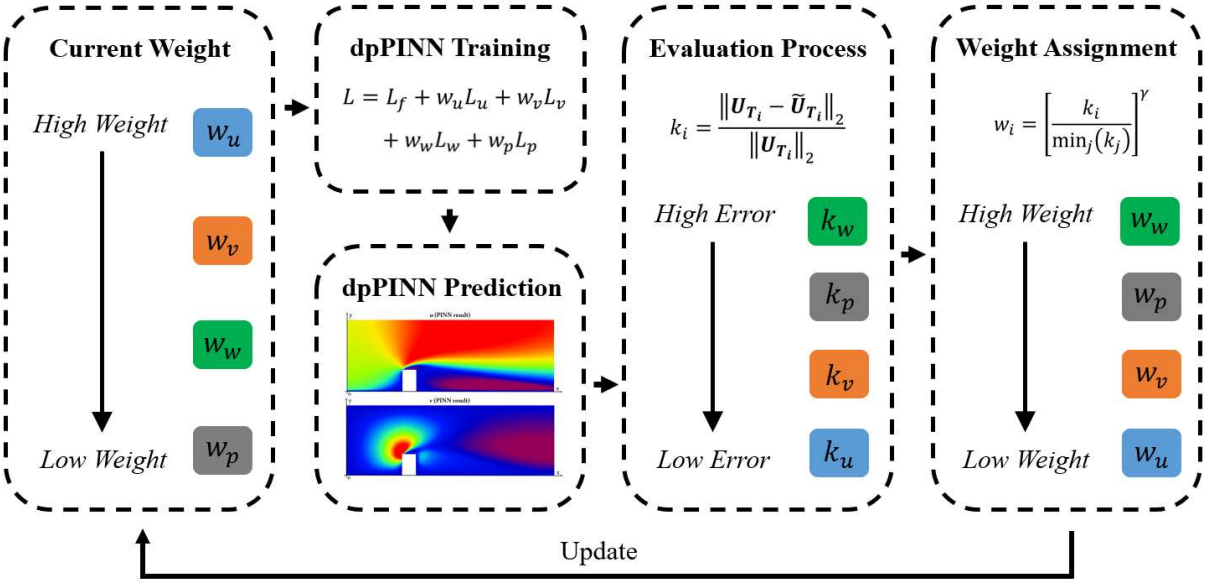


Figure 3-2. Updating mechanism of the dpPINN weighting coefficients.

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 the weighting coefficient w_f for L_f in Eq. (3-5) is not explicitly defined; instead, the weight of L_f in the total loss is indirectly adjusted 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. (3-5) and the dynamic prioritization loss balance strategy. The implementation of the dpPINN paradigm for simulating the three-dimensional flow field simulation is presented in Algorithm 3-1.

Algorithm 3-1 dpPINN for three-dimensional flow simulation using the RANS equations

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

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

Step 1: Determine the physical problem by specifying the computational domain, boundary conditions, and RANS turbulence model.

Step 2: Construct a deep neural network with the specified hyperparameters and initial neural network parameters.

Step 3: Specify the collocation points in the computational domain.

Step 4: Calculate loss components L_f, L_u, L_v, L_w, and L_p at the collocation points and data points using AD.

Step 5: Use the gradient descent algorithm to update the neural network parameters as follows:

for each iteration:

(a) Calculate the total loss function Eq. (3-5) 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 Eqs. (3-10) and (3-11).

end for

3.3 Validation Case: Building Outdoor Wind Field

3.3.1 Brief description of the wind tunnel test

In this section, utilize a building outflow wind field simulation will be utilized to verify the feasibility of the proposed dpPINN. 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 data from the wind tunnel test conducted by the Shimizu Corporation Institute of Technology (Meng and Hibi 1998) will be used in this study. A scale model of a building was positioned in the center location of the wind tunnel, which was 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 resulted in a Reynolds number of up to 2.4×10^4, which was 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.

3.3.2 Boundary conditions of the computational domain

The computational domain considered here is a cuboid with a size of 1.4 m × 0.7 m × 0.7 m (length × width × height), as shown in Figure 3-3. 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 1.497×10^{-5} m^2/s,

respectively. The vertical surface OGDA is defined as an initial speed boundary and the x-direction velocity u_{isb} is defined as follows:

$$u_{isb} = u_{initial} \tag{3-12}$$

where $u_{initial}$ denotes the velocity distribution, which is shown in Figure 3-4. For more details about the wind tunnel test, readers may refer 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:

$$v_{isb} = 0 \tag{3-13}$$

$$w_{isb} = 0 \tag{3-14}$$

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

$$\frac{\partial u_{swb}}{\partial z} = 0 \tag{3-15}$$

$$\frac{\partial v_{swb1}}{\partial z} = 0 \tag{3-16}$$

$$w_{swb1} = 0 \tag{3-17}$$

$$\frac{\partial p_{swb1}}{\partial z} = 0 \tag{3-18}$$

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

$$\frac{\partial u_{swb2}}{\partial y} = 0 \tag{3-19}$$

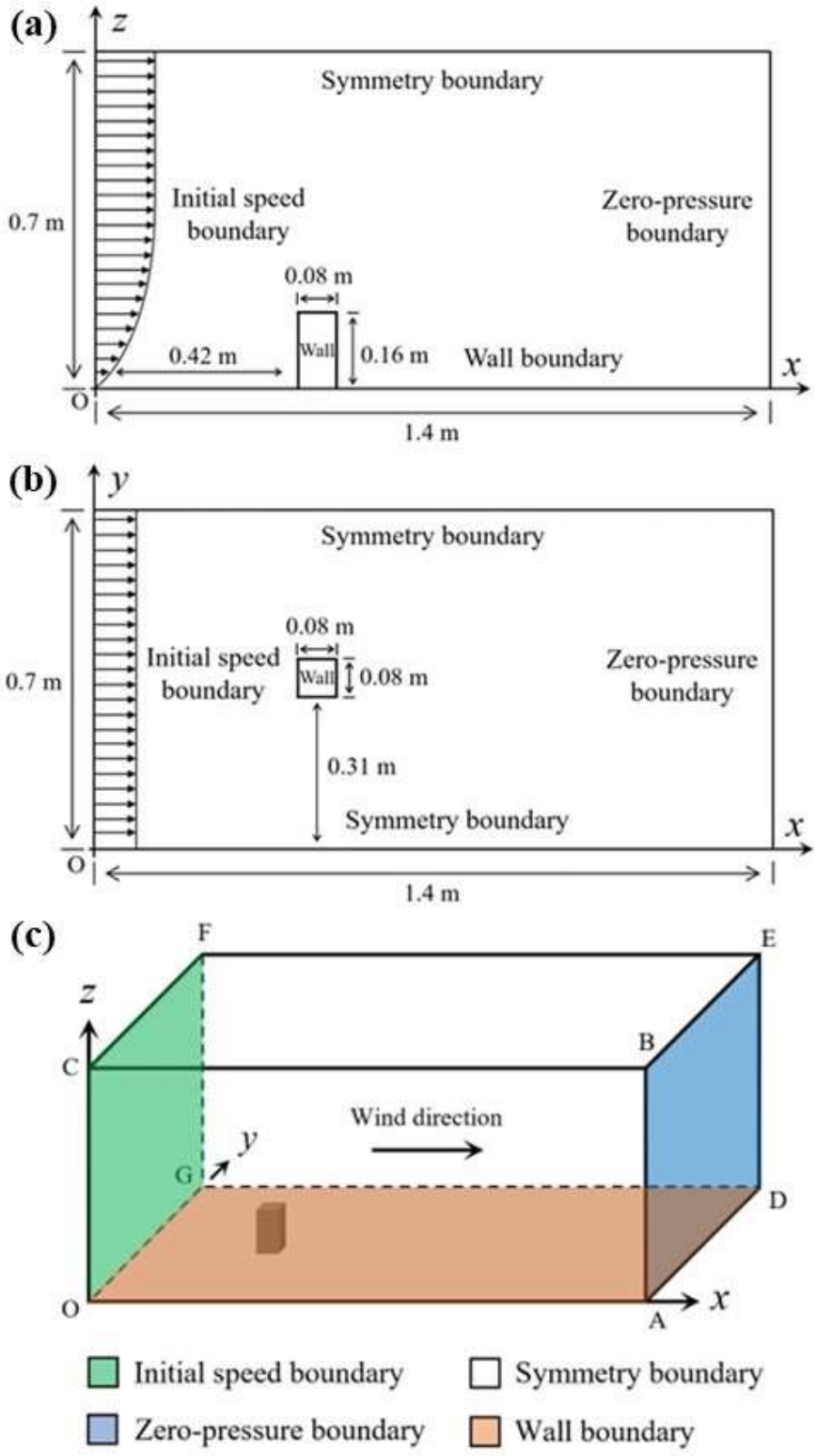


Figure 3-3. Computational domain for outdoor airflow simulation: **(a)** side view, **(b)** top view, and **(c)** general view.

$$v_{swb2} = 0 \tag{3-20}$$

$$\frac{\partial w_{swb2}}{\partial y} = 0 \tag{3-21}$$

$$\frac{\partial p_{swb2}}{\partial y} = 0 \tag{3-22}$$

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

$$p_{zpb} = 0 \tag{3-23}$$

$$\frac{\partial u_{zpb}}{\partial x} = 0 \tag{3-24}$$

$$\frac{\partial v_{zpb}}{\partial x} = 0 \tag{3-25}$$

$$\frac{\partial w_{zpb}}{\partial x} = 0 \tag{3-26}$$

$$\frac{\partial p_{zpb}}{\partial x} = 0 \tag{3-27}$$

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

$$u_{wb} = 0 \tag{3-28}$$

$$v_{wb} = 0 \tag{3-29}$$

$$w_{wb} = 0 \tag{3-30}$$

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.

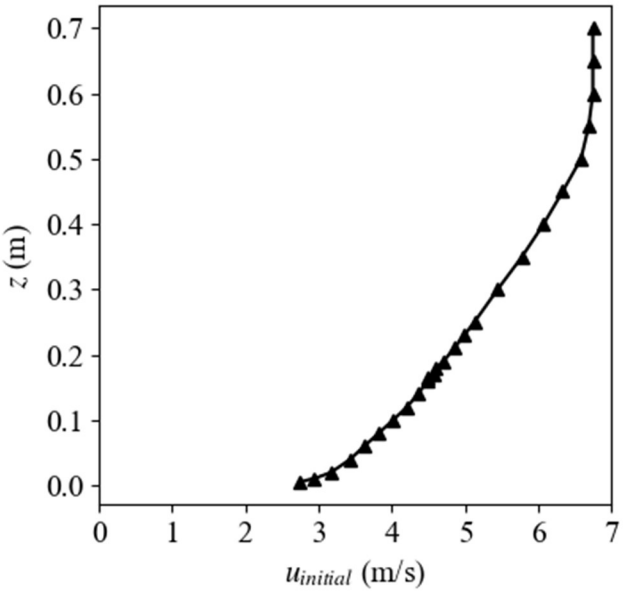


Figure 3-4. Distribution of velocity component $u_{initial}$ on the initial speed boundary.

3.3.3 Data constraints of the computational domain

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, Tang *et al.* 2022, Xie *et al.* 2022). The residuals between the PINN prediction and the labelled data, along with the residuals of RANS equations at collocation points, are calculated at each iteration and embedded in the total loss. In this way, the final solution to RANS equations by the PINN method can be thought of as a comprehensive result that combines both physical laws and on-site information (Guo *et al.* 2020). So this research is to make an attempt 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.

As aforementioned, 186 measurement points are dispersed throughout the wind tunnel, as shown in Figure 3-5. 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 dpPINN. 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 3-5. The selection of training points ensures easy accessibility of the data even in real practice. At the training points, the dpPINN predictions of the wind velocity components should satisfy the following constraints:

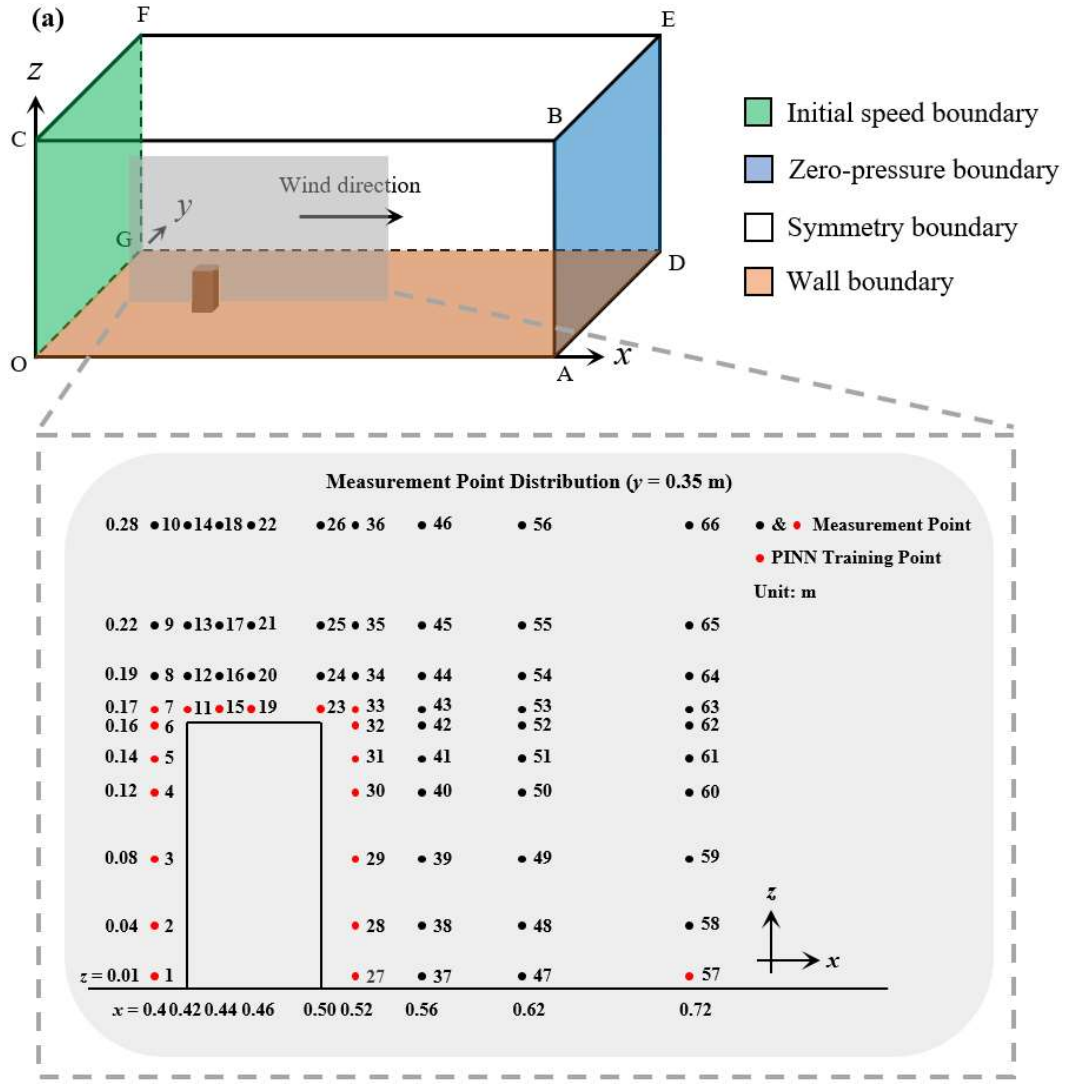
$$u_p = u_m \tag{3-31}$$

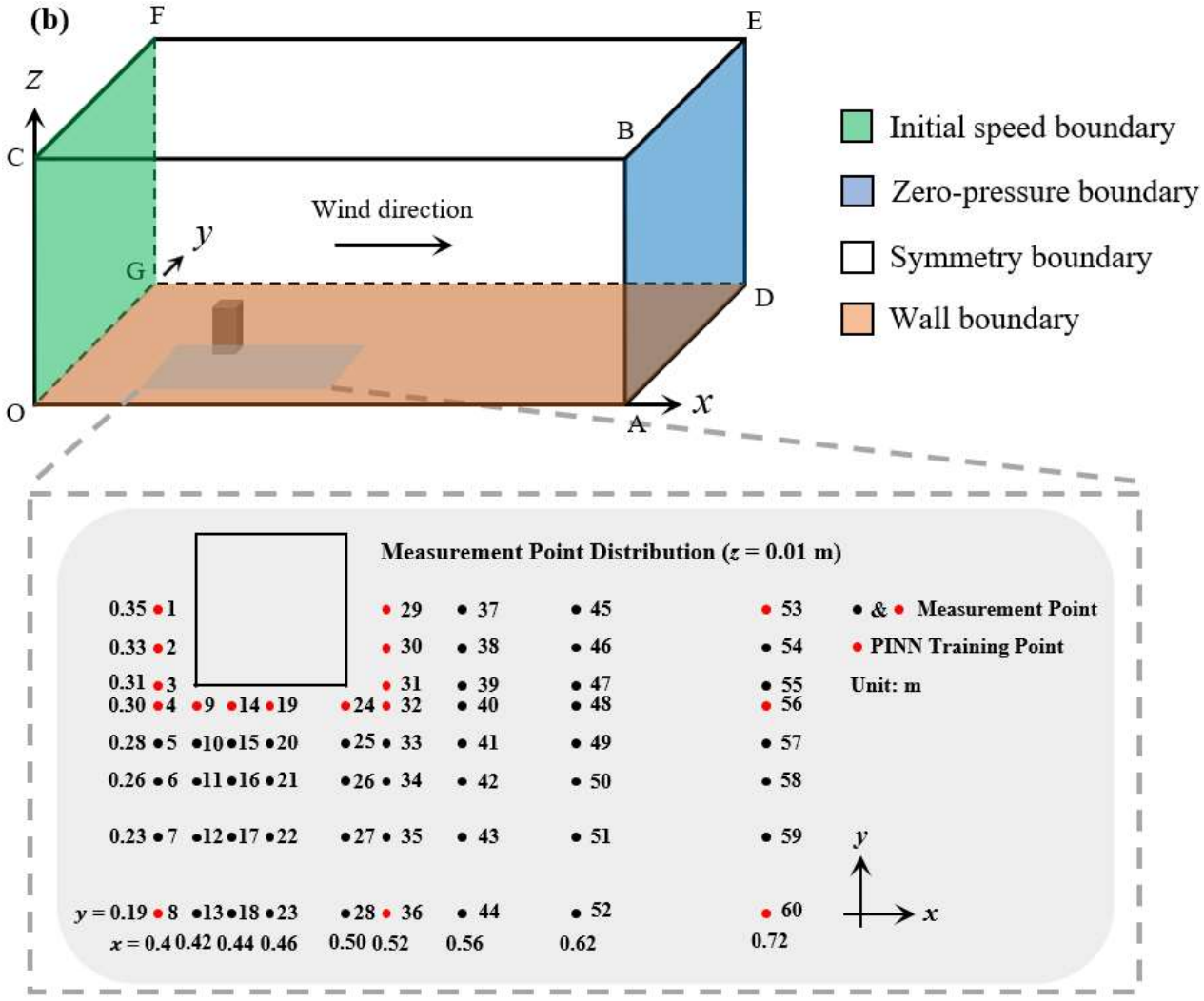
$$v_p = v_m \tag{3-32}$$

$$w_p = w_m \tag{3-33}$$

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

Then, a dpPINN 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. Keeping in mind limited accessibility in real practice, only the wind characteristics collected in the near-wall regions of wind tunnel are embedded, in conjunction with the constraints of physical equations, to guide the training of the dpPINN.


(a)
F
E
Z
C
B
Wind direction
y
G
D
O
A
x
Initial speed boundary
Zero-pressure boundary
Symmetry boundary
Wall boundary
Measurement Point Distribution (y = 0.35 m)
0.28 • 10 • 14 • 18 • 22 • 26 • 36 • 46 • 56 • 66
0.22 • 9 • 13 • 17 • 21 • 25 • 35 • 45 • 55 • 65
0.19 • 8 • 12 • 16 • 20 • 24 • 34 • 44 • 54 • 64
0.17 • 7 • 11 • 15 • 19 • 23 • 33 • 43 • 53 • 63
0.16 • 6 • 32 • 42 • 52 • 62
0.14 • 5 • 31 • 41 • 51 • 61
0.12 • 4 • 30 • 40 • 50 • 60
0.08 • 3 • 29 • 39 • 49 • 59
0.04 • 2 • 28 • 38 • 48 • 58
z = 0.01 • 1 • 27 • 37 • 47 • 57
x = 0.4 0.42 0.44 0.46 0.50 0.52 0.56 0.62 0.72
• & • Measurement Point
• PINN Training Point
Unit: m
z
x


(b)
F
E
Z
C
B
Wind direction
y
G
D
O
x
A
Initial speed boundary
Zero-pressure boundary
Symmetry boundary
Wall boundary
Measurement Point Distribution (z = 0.01 m)
0.35 1 29 37 45 53
0.33 2 30 38 46 54
0.31 3 31 39 47 55
0.30 4 9 14 19 24 32 40 48 56
0.28 5 10 15 20 25 33 41 49 57
0.26 6 11 16 21 26 34 42 50 58
0.23 7 12 17 22 27 35 43 51 59
y = 0.19 8 13 18 23 28 36 44 52 60
x = 0.4 0.42 0.44 0.46 0.50 0.52 0.56 0.62 0.72
& Measurement Point
PINN Training Point
Unit: m
y
x

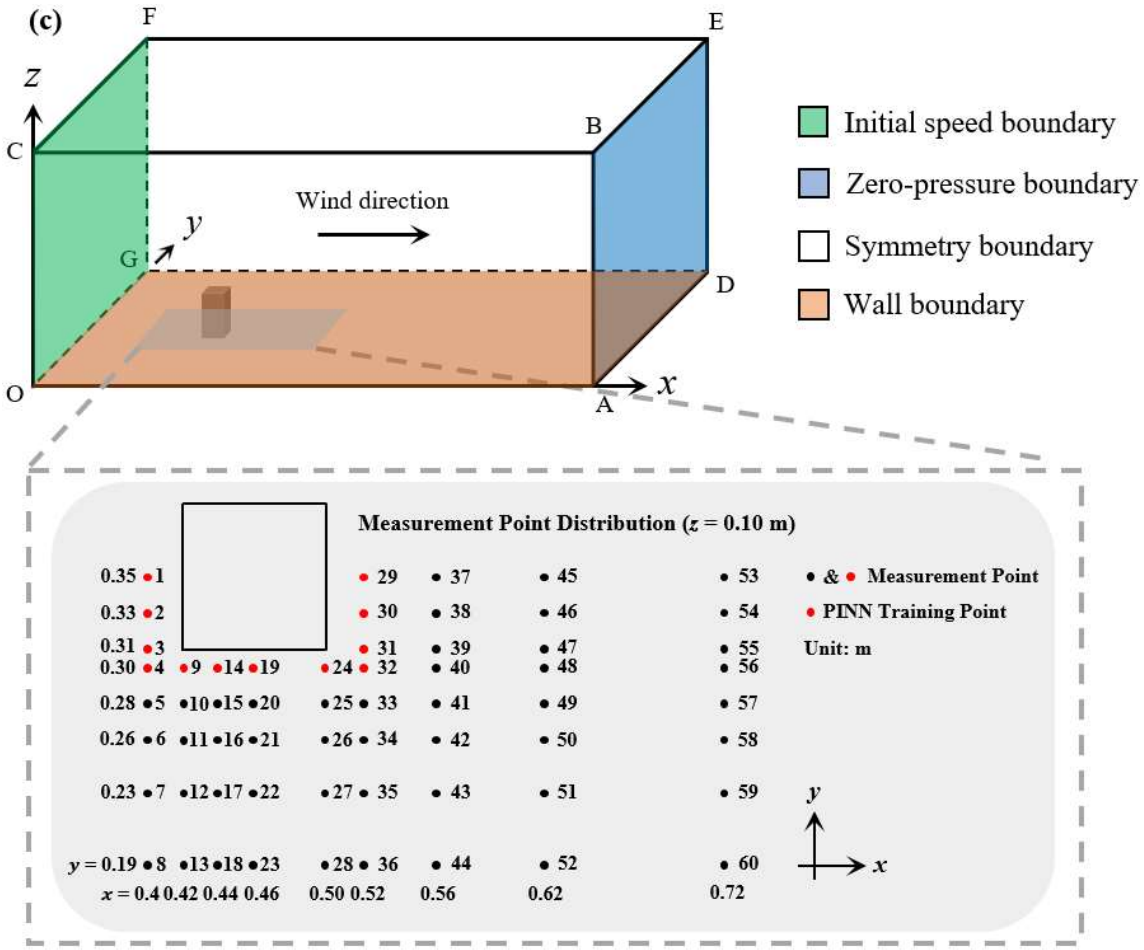


Figure 3-5. Distribution of measurement points in the cross-sections **(a)** y = 0.35 m, **(b)** z = 0.01 m, and **(c)** z = 0.10 m.

3.3.4 Physical governing equations

Although the instantaneous NS equation can describe turbulent behaviors, its nonlinearity makes the computation extremely complex, and it is rarely used in engineering practices. Instead, solving the RANS equations has evolved into an alternative effective approach to addressing such engineering problems. The NS equations are averaged in the time domain in the RANS framework to solve a time-averaged flow field rather than an instantaneous flow field. The computational burden can be significantly lessened since the turbulent fluctuation on each scale is no longer calculated. The steady RANS equations for a three-dimensional flow are shown as follows:

$$\frac{\partial}{\partial x}(\rho \bar{u}_i) = 0 \tag{3-34}$$

$$\rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial \bar{u}_i}{\partial x_j} - \rho \overline{u_i' u_j'}\right) \tag{3-35}$$

where ρ is the fluid density, $\bar{u}_i$ is the velocity component in x_i-direction, $\bar{p}$ is the pressure, μ is the laminar viscosity, and $-\rho \overline{u_i' u_j'}$ is the Reynolds stress. Turbulence models are adopted to describe the influence of the Reynolds stress terms in the momentum equations. Later in this chapter, the performances of four different turbulence models on the wind field simulation around a building will be evaluated. Among them, two are zero-equation models, while the other two are two-equation models. In zero-equation RANS turbulence models, the Reynolds stress is described as follows:

$$-\rho \overline{u_i' u_j'} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3-36}$$

where μ_t is the turbulent viscosity. Li model is a zero-equation RANS model for simulating

outdoor flow (Li *et al.* 2013). Li model adopts a mixing-length strategy to describe turbulent viscosity μ_t, which can be expressed as:

$$\mu_t = \max(\mu_{in}, \mu_{out}) \tag{3-37}$$

where μ_{in} is the boundary layer's turbulent viscosity, and μ_{out} is the outer layer's turbulent viscosity. Furthermore, μ_{in} takes the following form:

$$\mu_{in} = (C_{in} l)^2 S \tag{3-38}$$

where

$$C_{in} = 1.8 \times \left(1 - \exp\left(-0.645 \left(\frac{C_b}{H_b}\right)^{0.8}\right)\right) \times \exp\left(-2 \times \min\left(\frac{y}{H_b}, 1\right)\right) \tag{3-39}$$

$$S = \sqrt{\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)^2} \tag{3-40}$$

where C_b and H_b denote building width and height in its windward side. Outer layer's flow viscosity takes the following form:

$$\mu_{out} = C_{out} V l \tag{3-41}$$

where V is the local resultant velocity of the fluid flow. The coefficient C_{out} is expressed as:

$$C_{out} = \frac{C_\mu^{0.5} I_G^2 z_G^{2\alpha+0.1} z^{0.9-2\alpha}}{\alpha} \tag{3-42}$$

where $C_\mu = 0.09$, $I_G = 0.1$, $z_G = 350$, and $\alpha = 0.22$.

3.3.5 Implementation of dpPINN in the case study

Now, the sets $\mathbf{U}_r$, $\mathbf{V}_r$, $\mathbf{W}_r$, and $\mathbf{P}_r$ can be figured out based on the above information.

That is, $\mathbf{U}_r = \{12, 15, 19, 24, 28, 31\}$, $\mathbf{V}_r = \{13, 16, 20, 25, 29, 32\}$, $\mathbf{W}_r = \{14, 17, 21, 26, 30, 33\}$, and $\mathbf{P}_r = \{18, 22, 23, 27\}$. Please note that the chapter numbers within the formula numbers have been omitted here for clarity. It can be clearly observed that the original boundary condition loss and training data loss have been broken down, replaced by task losses related to *u*, *v*, *w*, and *p*.

For instance, the residuals of Eqs. (3-12) and (3-31) belong to different loss components in the default PINN, i.e., L_b and L_d, respectively. However, in the proposed dpPINN, both belong to the loss term L_u in Eq. (3-5). For another instance, the residuals of Eqs. (3-12) and (3-13) belong to the same loss components in the default PINN, i.e., L_b. However, in the proposed dpPINN, the former belongs to the loss term L_u while the latter belongs to the loss term L_v.

The proposed dynamic prioritization loss balance strategy has the benefit of allowing us to assess the performance of each task throughout the training process and allocate weight coefficients according to how well it performed in earlier training iterations. Tasks that perform exceptionally well will generally have a smaller weight in subsequent training, whereas tasks that perform just moderately well will be assigned with a larger weight so that they can be emphasized in the next stage of training.

3.3.5 dpPINN results compared with experimental data

As mentioned before, this study simulates the flow field around the scale model of a building in a wind tunnel by using sparse near-wall velocity data under the dpPINN framework. In this section, the dpPINN predictions are validated 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. The hyperbolic tangent function (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 26944 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 roughly about 0.318 second. It takes about 53.6% of the duration for AD to calculate the physical residuals and form the components of 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, 93 near-wall points are selected 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 3-5. 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 Figure 3-

6 and Figure 3-7, 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. The red and black dots are training and testing points respectively, which is consistent with Figure 3-5.

From 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. The top left panel of Figure 3-6 can be taken as an example for further exploration. Eight training points (points 27 to 33 in Figure 3-5(a) and point 29 in Figure 3-5(c)) and three testing points (points 34 to 36 in Figure 3-5(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 $x = 0.72$ m, only one training data point (point 57 in Figure 3-5(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 58 to 66 in Figure 3-5(a) and point 53 in Figure 3-5(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.

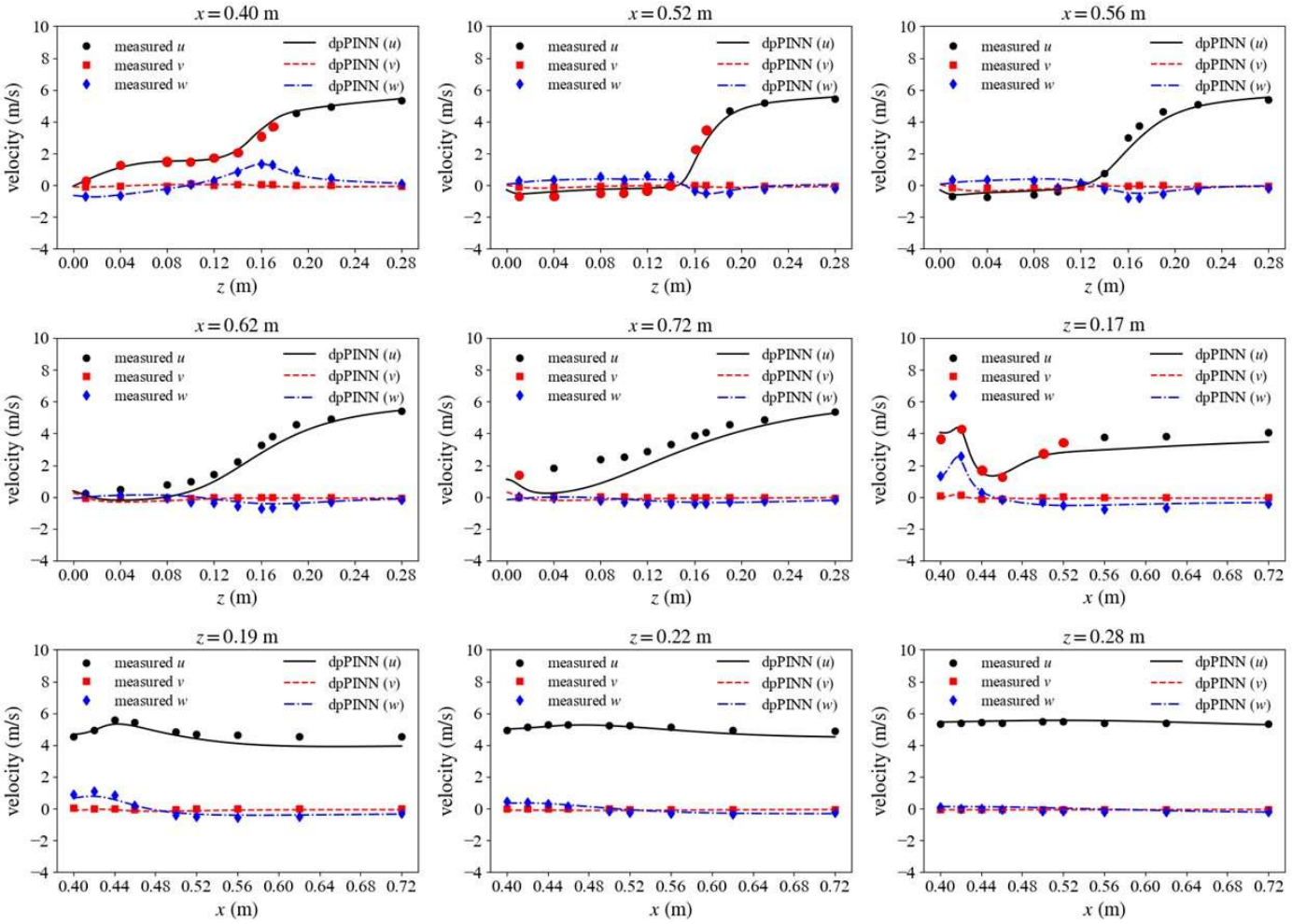


Figure 3-6. Comparison of results between dpPINN predictions and wind tunnel measurements in the cross-section y = 0.35 m.

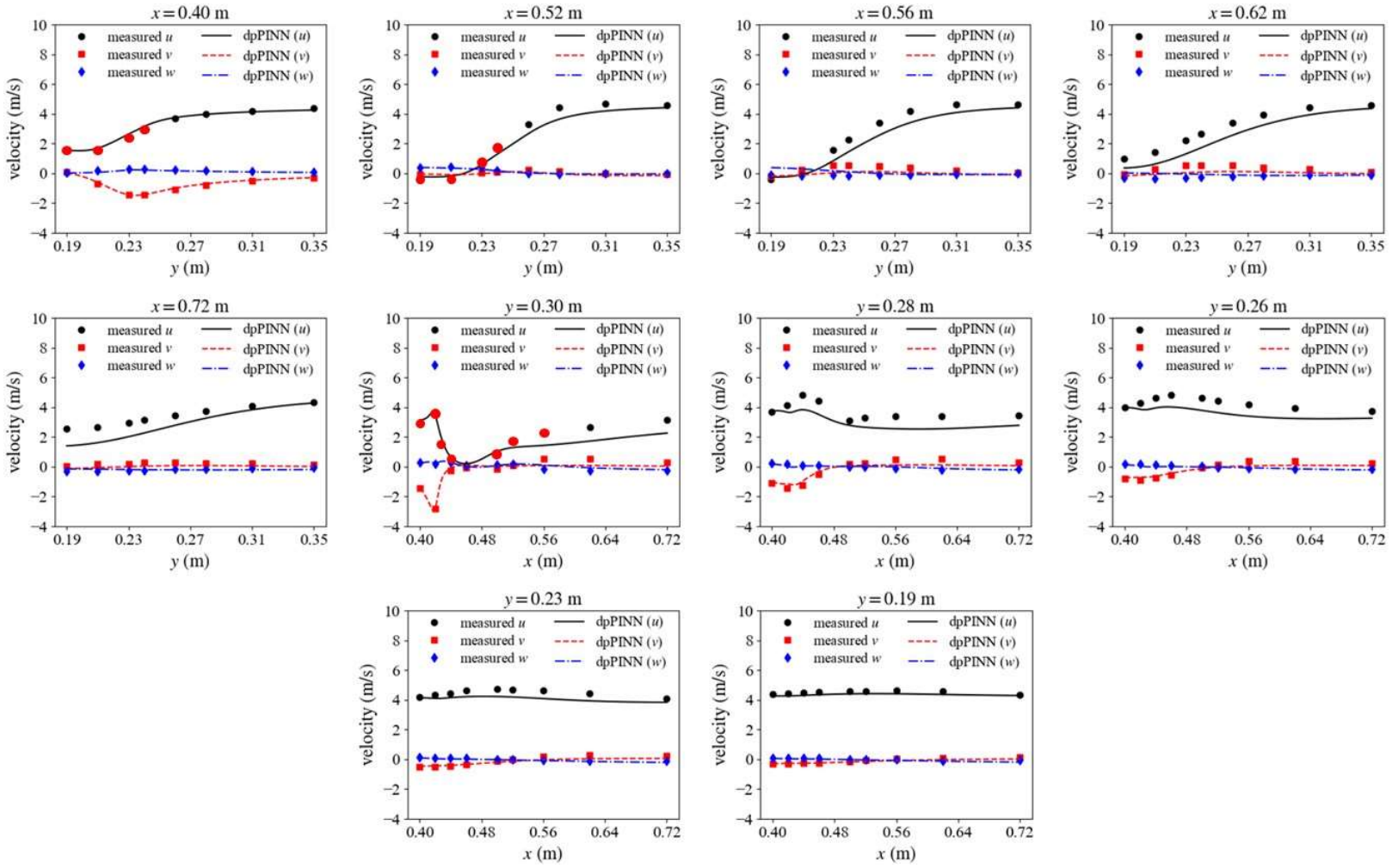


Figure 3-7. Comparison of results between dpPINN predictions and wind tunnel measurements in the cross-section z = 0.10 m.

3.3.6 Comparison of different loss balance strategies

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

$$l_2\ error = \frac{\left\| \boldsymbol{U_i} - \widetilde{\boldsymbol{U}}_{\boldsymbol{i}} \right\|_2}{\left\| \boldsymbol{U_i} \right\|_2} \times 100\% \qquad (3-43)$$

where $\|\cdot\|_2$ denotes the ℓ_2-norm, $\boldsymbol{U_i}$ denotes the vector of the reference data, and $\widetilde{\boldsymbol{U}}_{\boldsymbol{i}}$ denotes the vector of the dpPINN predictions. The ℓ_2-norm of a vector $\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_n)$ 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 resultant velocity are recorded during the training process, as depicted in Figure 3-8. The value of the coefficient γ is still set as 2, the same as in Section 4.1. It is seen that, after 1×10^5 iterations, the ℓ_2 errors of u and the resultant 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 results would be acceptable in consideration of the existence of measurement noise during the wind tunnel test. Also, the prediction accuracy between the dpPINN and other PINNs adopting different loss balance strategies is compared, as shown in Figure 3-8, where the influence of the coefficient γ is investigated as well. In Figure 3-8, the black lines represent the original PINN, which uses Eq. (3-1) 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.

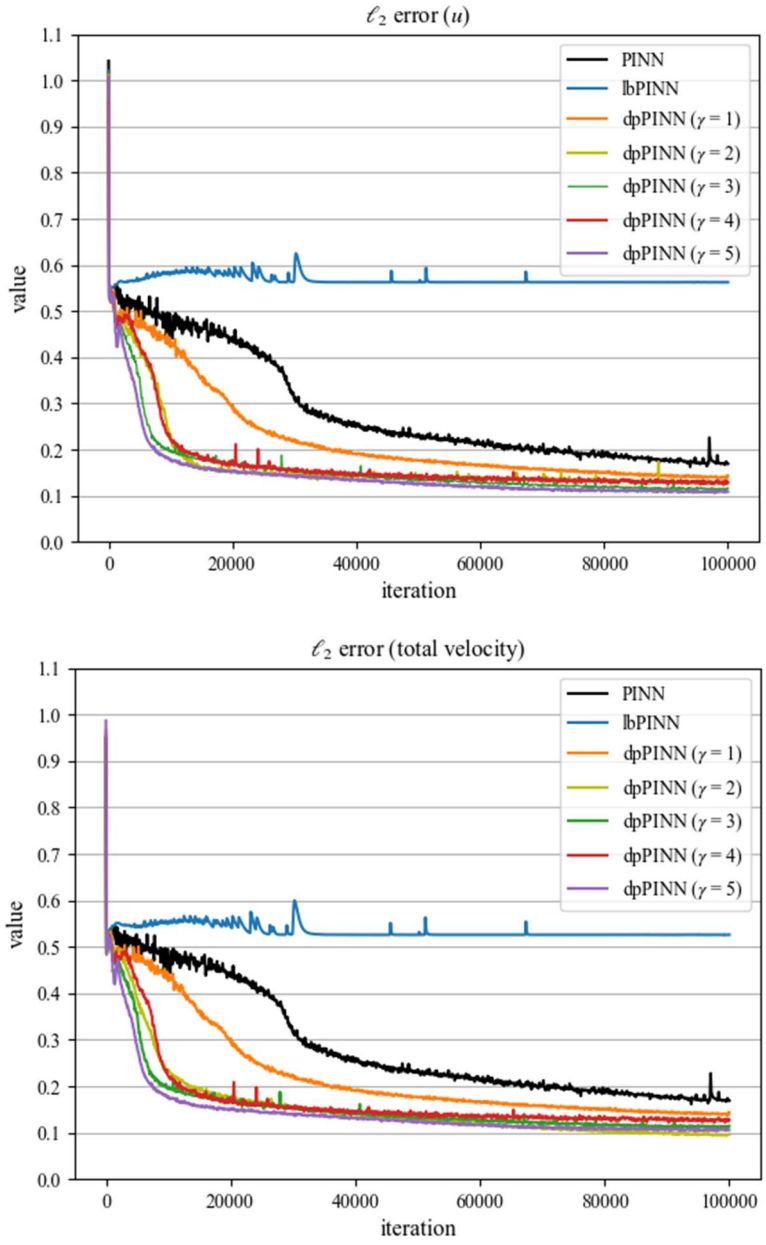


Figure 3-8. ℓ_2 errors of the PINN predictions using different loss balance strategies (top: u; bottom: resultant velocity).

Unfortunately, the lbPINN seems inapplicable in this case. The ℓ_2 errors of the velocity components u and the resultant 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 resultant 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 resultant velocity are reduced by roughly 20% to 30% when γ ranges from 1 to 5 after 1×10^5 iterations. However, the conclusion that dpPINN outperforms PINN in prediction accuracy because the slow convergence rate seems to limit the potential of the PINN cannot be currently drawn, so the training process for additional 8×10^4 iterations to a total of 1.8×10^5 iterations is extended. The findings reveal that the two approaches estimate u and v with similar accuracy, while the error of w predicted by dpPINN (γ=2) is still 4% lower than that predicted by PINN. A more detailed comparison is tabulated in Table 3-1.

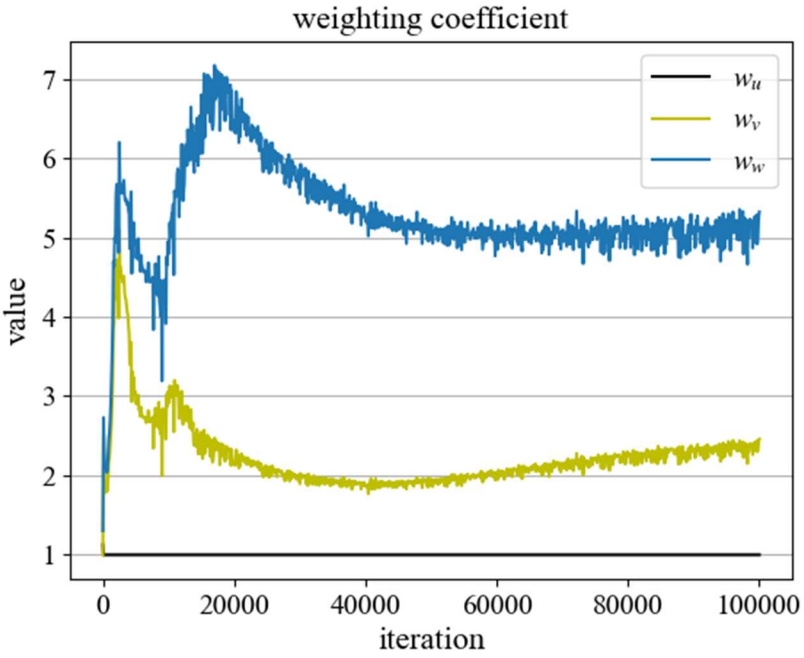


Figure 3-9. The dynamic balance of weighting coefficients in the dpPINN.

In addition, Figure 3-8 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 noted that we mainly focus on the l_2 errors of the mainstream velocity and resultant velocity in this figure, while the l_2 errors of the other two velocity components are summarized in Table 3-1.

The dynamic balance process of weighting coefficients of the dpPINN (γ=2) is also monitored, as shown in Figure 3-9. 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.

Table 3-1. ℓ_2 errors of the PINN, lbPINN, and dpPINN predictions after 1×10^5 iterations.

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

3.3.7 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 3-2. 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.

Table 3-2. ℓ_2 errors of the dpPINN predictions with different neural network configurations.

Width \ Depth	4		6		8	
	u	V	u	V	u	V
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

3.3.8 Influence of turbulence model

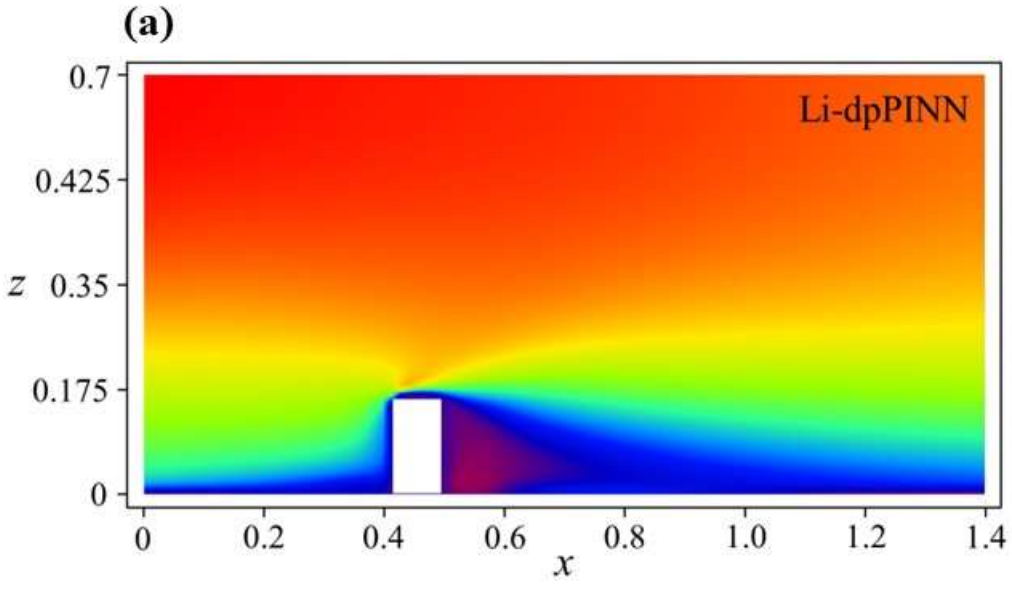
To determine the impact of embedding different turbulence models in the PINN framework, a further analysis is carried out. Another widely used zero-equation turbulence model is the Chen model (Chen and Xu 1998). It is assumed that the turbulent viscosity is

determined by the air density, the local velocity V, and the distance l from the nearest wall, which can be calculated as follows:

$$\mu_t = 0.03874\rho V l \qquad (3-44)$$

For this simulation, an attempt is made to embed the Chen model rather than the Li model. Li-dpPINN and Chen-dpPINN are dpPINNs based on the Li and Chen models, respectively. As before, the neural network consists of six hidden layers with 40 neurons each. Coefficient γ is still set to 2.

Figure 3-10 shows the results. This figure shows the contours of the velocity component u predicted by the Li-dpPINN and the Chen-dpPINN at the cross-section y = 0.35 m respectively. The velocity contours show that when the Chen model is used in the wind field simulation instead of the Li model, there is a greater clockwise rotation of the vortex near the leeward side of the building, contrary to the measured results near the ground. Chen model, which is widely used in indoor airflow simulation rather than outdoor airflow simulation, may explain this phenomenon.



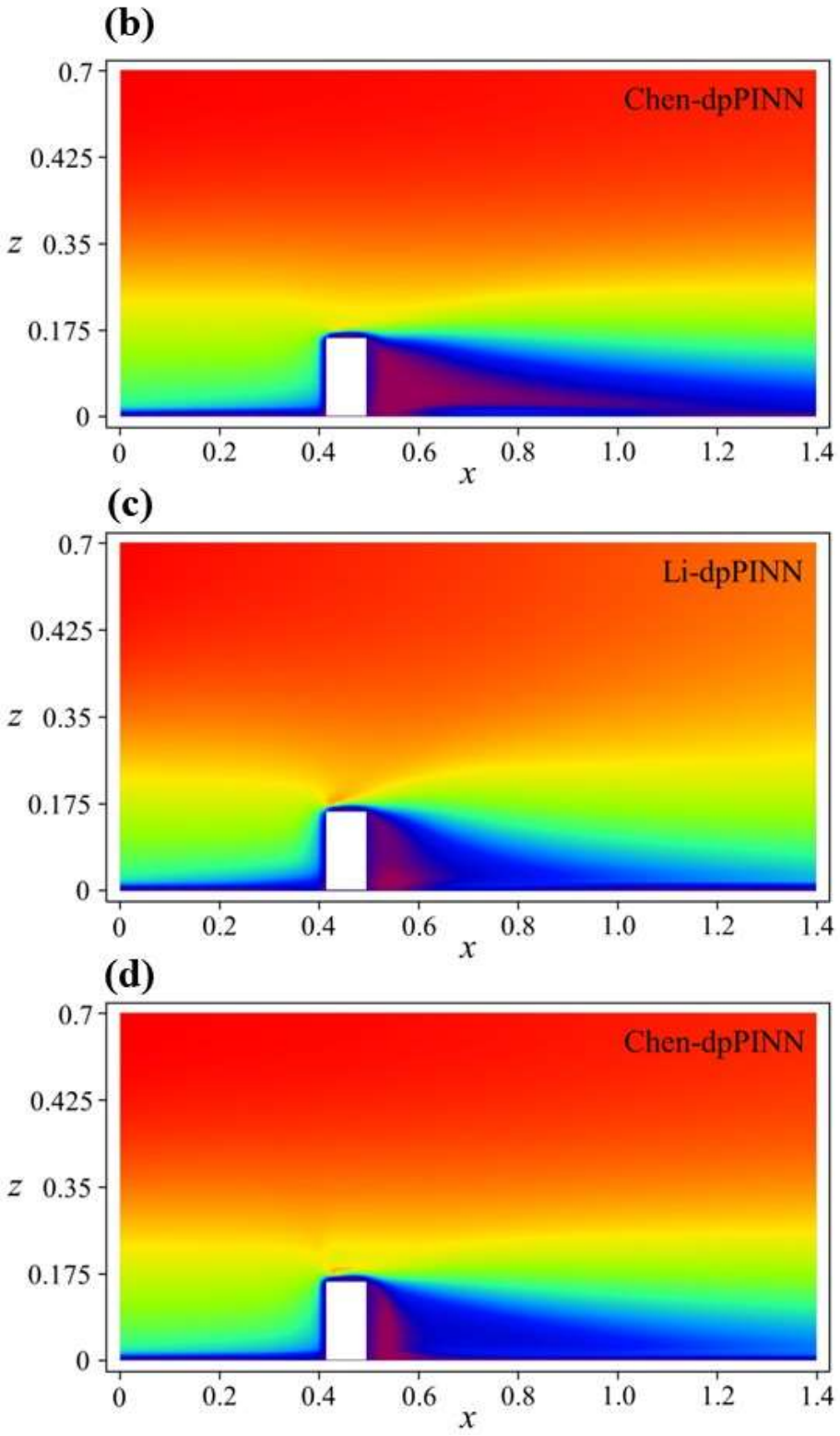


Figure 3-10. u contours at y = 0.35 m: **(a)** and **(b)** 93 measurement points are used to train the dpPINN model; **(c)** and **(d)** 100 measurement points are used to train the dpPINN model.

To overcome these challenges, an attempt is made to include velocity data from seven additional measurement points in the dpPINN training process. These are points 48–53 in Figure 3-5(a) and point 45 in Figure 3-5(c). With velocity data from these measurement points, the leeward side of the scale building model can be guided properly for the dpPINN solution. To be noted, when different turbulence models are embedded for airflow simulation, this region also has the maximum relative error. Figure 3-10(c), Figure 3-10(d) and Figure 3-11 show the modified results. As more labelled data are embedded, Figures 3-10(c) and (d) illustrates the contours of the velocity component u using different turbulence models.

In Figure 3-11, the Chen-dpPINN prediction results for the velocity component u at the cross-section $y = 0.35$ m are compared to the Li-dpPINN prediction results. When seven more measurement points are used to train the Li-dpPINN and Chen-dpPINN, the ℓ_2 errors of the velocity component u drop to 0.089 and 0.128, respectively, which are acceptable in engineering practice. As a result, even when more training data are used for supervised learning, the Li-dpPINN still performs better than the Chen-dpPINN.

Although an inapplicable turbulence model, namely the Chen model, is embedded in the dpPINN for airflow simulation, the solution does not seem totally ineffective because the size of the vortex near the leeward side of the building is corrected by sufficient training data. In the cross-section $y = 0.35$ m, Li-dpPINN and Chen-dpPINN predictions of the velocity component u are well congruent with the field-measured results, although the former is slightly better. Here is the second benefit of using this physics-based data-driven approach for outdoor airflow simulation. PINN integrates measurement data into physical models, which is difficult when using traditional physics-based methods. Furthermore, this extends the applicability of these physical models, making the PINN framework robust for airflow simulation.

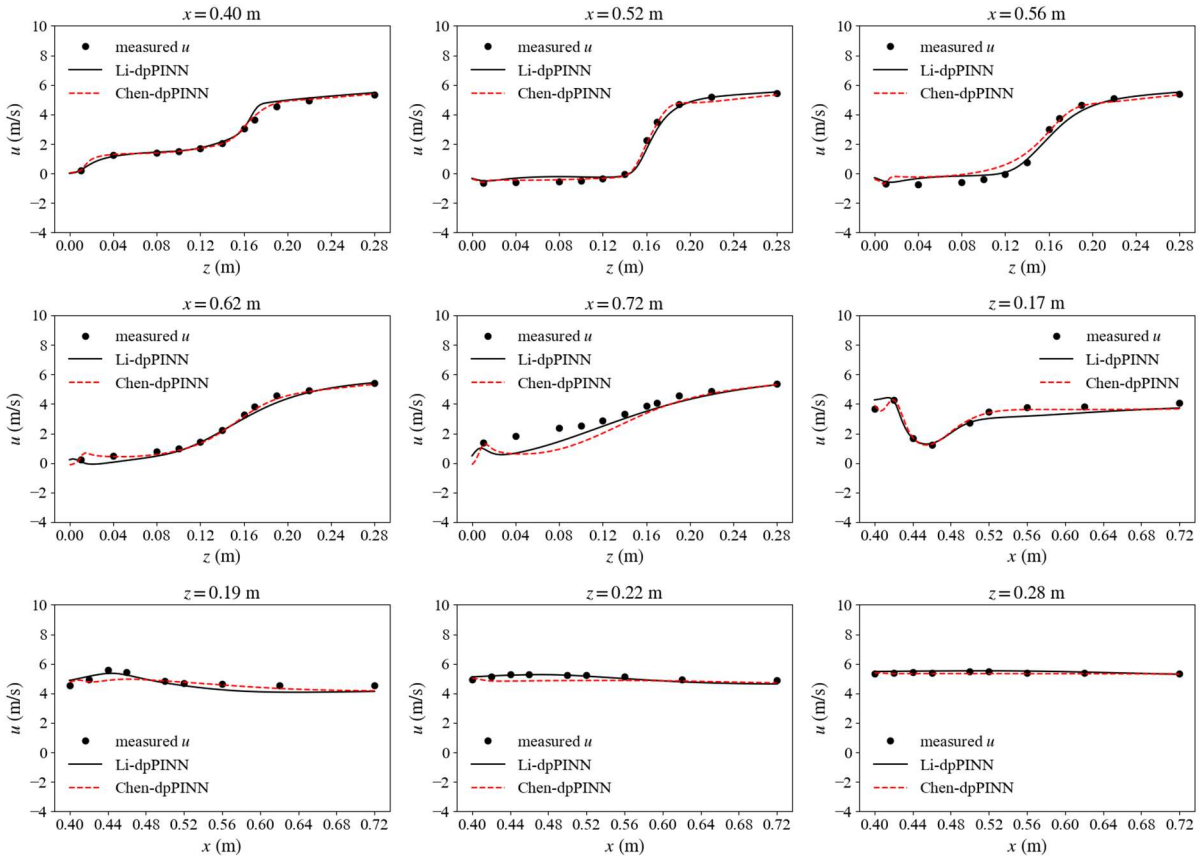


Figure 3-11. Comparison of Li-dpPINN predictions with the Chen-dpPINN predictions of the velocity component u at the cross-section $y = 0.35$ m.`

Furthermore, two well-known two-equation RANS turbulence models, i.e., the standard k-ε model and RNG k-ε model, are also encoded in the dpPINN model. Figure 3-12 depicts the flow streamlines in the building's leeward recirculation zone when various RANS turbulence models are utilized. Here, 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 becomes the focus. The result indicates that when using the zero-equation RANS turbulence models, the airflow recirculation zone cannot be well reflected, but when the two-equation models are adopted, appropriately sized refluxes are generated in its 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 dpPINN on physical conditions are relatively flexible, that is, only by adding penalty terms in the loss function, so it is not surprising that its solution sometimes deviates. It is believed that further in-depth research is needed to specifically address this issue.

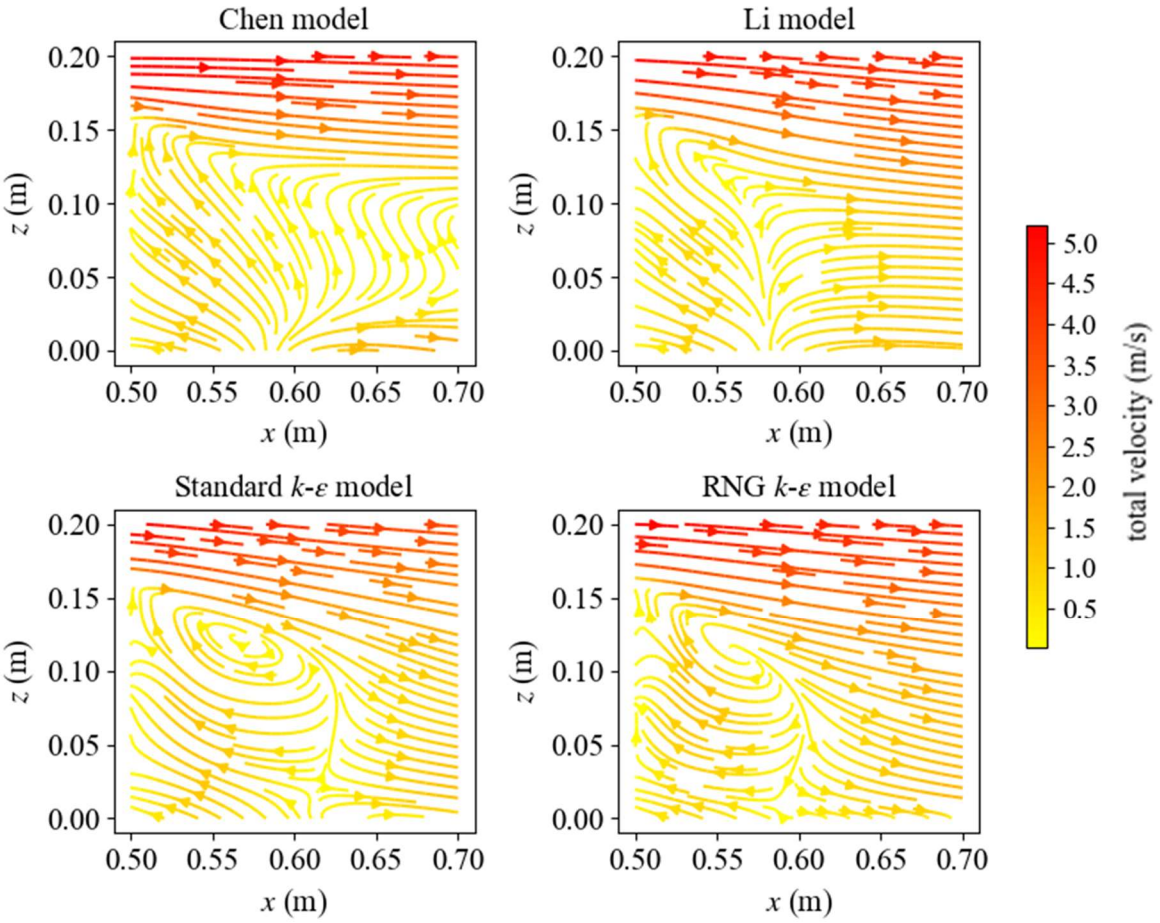


Figure 3-12. The flow streamlines in the building's leeward recirculation zone.

3.4 Conclusions

By rewriting the form of the total loss in the original PINN, a novel self-adaptive loss balance strategy, i.e., dpPINN, is proposed in this chapter. A zero-equation RANS turbulence

model is used to reconstruct the entire flow field around a scale model of a building in a wind tunnel. For supervised learning, sparse near-wall velocity data is used. Furthermore, the wind tunnel experiment provides verification that the dpPINN framework is feasible. The impact of different neural network configurations and embedded turbulence models on dpPINN prediction is also investigated. It appears that the dpPINN can provide an auxiliary means to predict spatial flow fields around a building based on the results. Here are some conclusions:

- Even though only a small portion of the sparse near-wall data can be used to reconstruct the three-dimensional flow field around the scale building model, embedding the measurements and laws into the neural network can still provide the missing airflow information across the whole computational domain.
- This chapter proposes a dpPINN model frame that outperforms the original PINN and lbPINN in prediction accuracy. In the dpPINN framework, a higher relative error loss term is prioritized after certain iterations, speeding up the training process and resulting in better performance.
- In general, the width of the neural network has a greater influence on prediction accuracy than the depth of the network. The dpPINN performance improves with a wider width, but the computational cost increases as well. Nine test configurations were investigated for outdoor airflow simulation, and the configuration of six hidden layers, each with 60 neurons, was found to be optimal.
- It is important to note that turbulence models embedded in the RANS equations directly affect the dpPINN solution. There is a tendency for a bigger clockwise-rotating vortex to occur near the leeward side of the building when the Chen model instead of the Li model is embedded in the RANS equations. When more training data is used for supervised

learning in the PINN framework, however, Li-dpPINN and Chen-dpPINN predictions are both consistent with experimental results, although the former is still slightly better.

- When there are labeled training data available for supervised learning in flow simulations around buildings, the PINN framework shows a higher universality compared with pure physics-based methods. Due to the fact that only the near-wall data is used for training the neural network, it is also less data-demanding than pure data-driven methods, and yet still demonstrates competitive results.

CHAPTER 4

MULTIFIDELITY MODELING

4.1 Foreword

In the multifidelity modeling framework, accurate but expensive high-fidelity data is scarce, while cheap and inaccurate low-fidelity data is abundant. In many cases, low-fidelity data can provide useful information such as trends for high-fidelity modeling. Therefore, multifidelity modeling based on a small amount of high-fidelity data can greatly improve the prediction accuracy of single-fidelity modeling. As a result, multifidelity modeling has been proven to be efficient and effective, by using low-fidelity and high-fidelity data to achieve high accuracy in different applications.

Building a link between low-fidelity and high-fidelity data in multifidelity modeling is crucial. Among the current methods, Gaussian Process (GP) regression with a linear autoregressive framework has attracted the most interest, even though only the linear association between low-fidelity and high-fidelity data may be captured under this approach (Le Gratiet and Garnier 2014). This approach has been enhanced by some such that it can recognize intricate nonlinear relationships (Perdikaris *et al.* 2017). The multifidelity approach based on GP regression has made considerable strides, but there are still certain drawbacks, including the approximation of discontinuous functions, high-dimensional issues, and strongly nonlinear inverse problems. Nevertheless, multifidelity modeling is still an indispensable

means to integrate physical information and data information which has been widely adopted in different research fields.

For instance, Perdikaris *et al.* (2017) proposed a nonlinear fusion algorithm for multifidelity modeling based on the GP regression. In their algorithm, multifidelity modeling is based on the GP regression and is achieved by combining low-fidelity models with a small number of high-fidelity observations, which saves much computational cost. Their scheme was verified by different cases and the results indicated its feasibility when sufficient low-fidelity and high-fidelity data was available for modeling. In addition, Meng and Karniadakis (2020) proposed a method which was named multifidelity PINN, as shown in Figure 4-1 in the case that only a small set of high-fidelity data was available for multifidelity modeling. In their opinion, implementing multifidelity modeling based on GP regression optimization is fairly challenging, so multifidelity approaches are therefore urgently required to address these flaws. Their results also demonstrated that the proposed multifidelity PINN could be a powerful means of multifidelity modeling.

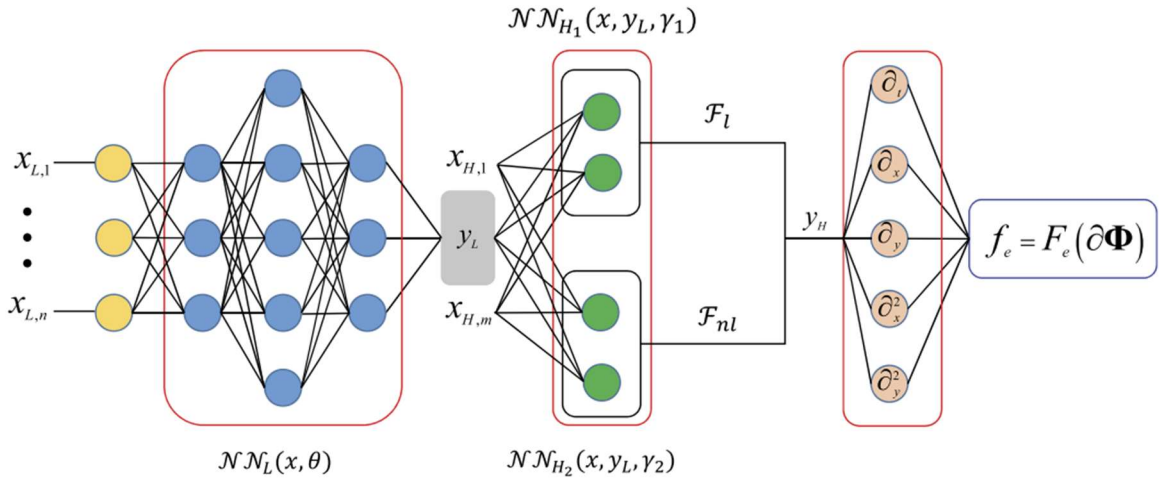


Figure 4-1. The structure of the multifidelity PINN proposed by Meng and Karniadakis (2020).

It has been proposed that a small sampling of high-fidelity data can be used to reconstruct the flow field within the entire computational domain in order to achieve this balance. For instance, using direct numerical simulation data, Abrahamson and Lonnes (1995) reconstructed vorticity fields using the least-squares method. However, it ignores the details of local flow features, so a local characteristic analysis cannot be conducted based on the least-squares method, even though it approaches the averaged field well. Also, to reconstruct a flow field with this method, thousands of high-fidelity data points are required, which remains a heavy burden for engineers.

This chapter proposes a multifidelity, physics-informed data-driven strategy for time-averaged turbulent flow field simulation. First, PINN prediction is used to estimate low-fidelity flow fields based on RANS equations. Since no training data is necessary in low-fidelity modeling, it is purely physics-based. Second, sparse field or laboratory measurements are regarded as high-fidelity observations. To reconstruct flow fields, a multifidelity GP model is established using the nonlinear information fusion (NIF) algorithm proposed by Perdikaris *et al.* (2017). By extracting nonlinear cross-correlations between low-fidelity approximations and high-fidelity observations, the multifidelity GP model can be trained and high-fidelity predictions can be carried out using the NIF algorithm. A flow past a hill and a flow past a square cylinder are presented in this chapter to demonstrate the feasibility of our proposed method. Furthermore, the proposed strategy is compared with two other common methods. According to the results, the multifidelity model demonstrates superior accuracy when approximating measurement data for these two flow cases.

Firstly, as the PINN model is used only for low-fidelity, less accurate modeling, the proposed strategy significantly increases its applicability. In addition, embedded physical information, however, provides significant guidance in multifidelity modeling, resulting in less

training data being needed in flow field reconstruction compared with other pure data-driven methods.

4.2 Methodology

4.2.1 PINN structure

In Chapter 4, the RANS equations using the Chen model are adopted for time-averaged flow field simulation. The PINN framework is depicted in Figure 4-2. As shown in the figure, the framework of the PINN adopted here is similar to that in Section 3.2.1, except that no measurement data is embedded in the total loss for training the model. The total loss of the neural network is written as follows:

$$L = w_f L_f + w_b L_b \tag{4-1}$$

where

$$L_f = \frac{1}{N_f}\sum_{n=1}^{N_f}\sum_{i=1}^{2}|f_i^n|^2 \tag{4-2}$$

$$L_b = \frac{1}{N_{nb}}\sum_{i=1}^{N_{nb}}\left|r_{nb}^i\right|^2 + \frac{1}{N_{db}}\sum_{i=1}^{N_{db}}\left|r_{db}^i\right|^2 \tag{4-3}$$

In the expression, L_f and L_b still denote the loss components corresponding to the residuals of the governing equations and boundary conditions, respectively. w_f and w_b denote the weighting coefficients of the corresponding loss terms. f_i^n is the residual of the *i*th governing equation in Fig. 5-1. r_{nb}^i and r_{db}^i are the residuals for the Neumann boundary and Dirichlet boundary, respectively. N_f is the number of points used to calculate the residuals of

the governing function, while N_{nb} and N_{db} are the numbers of points used to calculate the residuals for the Neumann boundary and Dirichlet boundary, respectively. It is worth noting that in two-dimensional flow cases, the velocity components are replaced with a stream function φ in the neural network outputs to ensure that the continuity condition is strictly satisfied.

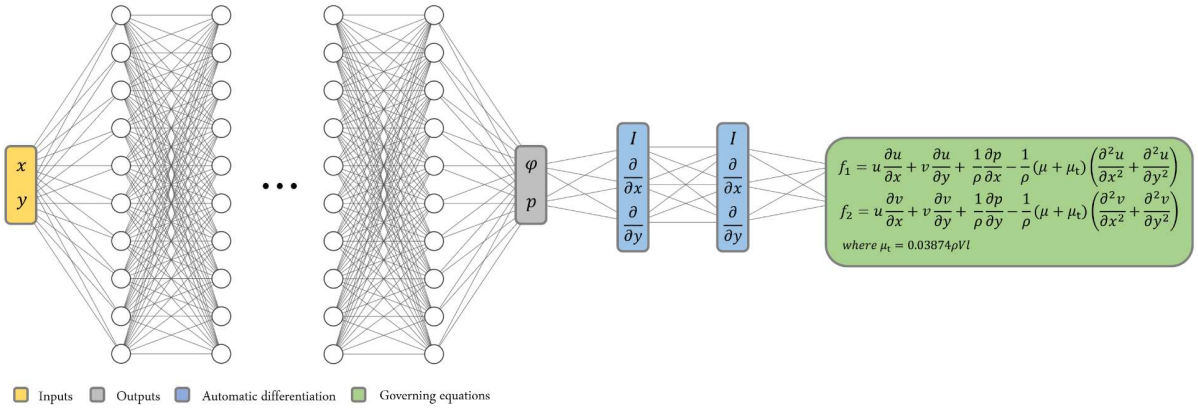


Figure 4-2. PINN framework for RANS simulation when Chen model is adopted.

4.2.2 NIF algorithm

In the NIF algorithm, multifidelity modeling is based on the GP algorithm and is achieved by combining low-fidelity models with a small number of high-fidelity observations. Now, the GPs f_h and f_l represent high-fidelity and low-fidelity models, respectively. The NIF algorithm expresses f_h as follows:

$$f_h(\boldsymbol{x}) = g_h(\boldsymbol{x}, f_{*l}(\boldsymbol{x})) \qquad (4-4)$$

where $g_h \sim \mathcal{GP}(\boldsymbol{f_h}|\boldsymbol{0}, k_h((\boldsymbol{x}, f_{*l}(\boldsymbol{x})), (\boldsymbol{x}', f_{*l}(\boldsymbol{x}')); \theta_h))$. A GP posterior at the low-fidelity level is given by $f_{*l}(\boldsymbol{x})$. θ_h is the hyperparameter. As a covariance kernel, k_h can be decomposed into:

$$k_h = k_{h_\rho}\left(\boldsymbol{x}, \boldsymbol{x}'; \theta_{h_\rho}\right) \times k_{h_f}\left(f_{*l}(\boldsymbol{x}), f_{*l}(\boldsymbol{x}'); \theta_{h_f}\right) + k_{h_\delta}\left(\boldsymbol{x}, \boldsymbol{x}'; \theta_{h_\delta}\right) \quad (4-5)$$

A squared exponential form covariance function is used here, with ARD weights applied to k_{h_ρ}, k_{h_f}, and k_{h_δ} (Rasmussen 2003). θ_{h_ρ}, θ_{h_f}, and θ_{h_δ} are hyperparameters. Based on the NIF algorithm, the high-fidelity model may be derived from the input coordinates $\boldsymbol{x}$ and the output of the low-fidelity model $f_{*l}(\boldsymbol{x})$. Therefore, it is a joint representation of the low-fidelity model's input space and its posterior prediction. Eq. (4-4) also incorporates both $\boldsymbol{x}$ and $f_{*l}(\boldsymbol{x})$ into its covariance kernel, which captures nonlinear nonfunctional cross-correlations of space-dependent nonlinearity. Optimization of the hyperparameters of the GP model is based on minimizing the negative log marginal likelihood (NLML), which can be described as:

$$NLML = \frac{1}{2} log|\boldsymbol{K}| + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}^{-1}\boldsymbol{y} + \frac{n_d}{2} log 2\pi \quad (4-6)$$

where $\boldsymbol{K}$ represents the kernel function, $\boldsymbol{y}$ represents the training target, and n_d represents the dimension of the input space. Based on the optimized hyperparameters, the posterior distribution of the high-fidelity GP model at a test point $(\boldsymbol{x}_*, f_{*l}(\boldsymbol{x}_*))$ can be calculated as follows:

$$p(f_{*h}(\boldsymbol{x}_*)) = \int p\left(f_h\left(\boldsymbol{x}_*, f_{*l}(\boldsymbol{x}_*)\right)|\boldsymbol{y}_h, \boldsymbol{x}_h, \boldsymbol{x}_*\right) p(f_{*l}(\boldsymbol{x}_*))\mathrm{d}\boldsymbol{x}_* \quad (4-7)$$

To simulate the posterior distribution of the high-fidelity model, Monte Carlo simulation is used. In the low-fidelity model, the posterior prediction follows a Gaussian distribution because it is a standard GP regression with parametric input data points. High-fidelity models, however, are GP regression models with the input of the posterior prediction from the low-fidelity model. This results in a non-Gaussian posterior distribution for the high-fidelity model. Therefore, Monte Carlo integration of Eq. (4-7) is applied here to calculate the high-fidelity

model's posterior mean and variance.

4.2.3 Workflow of the multifidelity strategy

As a summary of the multifidelity flow field reconstruction strategy, which is proposed in the chapter, the following workflow is given (also shown in Figure 4-3):

Step 1: A PINN model is used to generate massive quantities of low-fidelity continuous data. In the training process of neural networks, residuals of governing equations and boundary conditions are incorporated.

Step 2: A sample of training data is obtained from the PINN results in the computational domain for the low-fidelity GP modeling. Then minimize the NLML of Eq. (4-6) to optimize the hyperparameters of the low-fidelity GP model. Calculate the posterior mean and variance of the low-fidelity standard GP regression model.

Step 4: Using the posterior prediction of the low-fidelity model and a small number of high-fidelity observations, the high-fidelity GP regression model of Eq. (4-4) is constructed. In the high-fidelity GP model, the hyperparameters are optimized by minimizing the NLML of Eq. (4-6), using the kernel function from Eq. (4-5).

Step 5: According to the Monte Carlo integration of Eq. (4-7), the posterior mean and variance for the high-fidelity GP model can be outputted. In this step, the posterior mean and variance of the low-fidelity standard GP regression model in *Step 2* will be used.

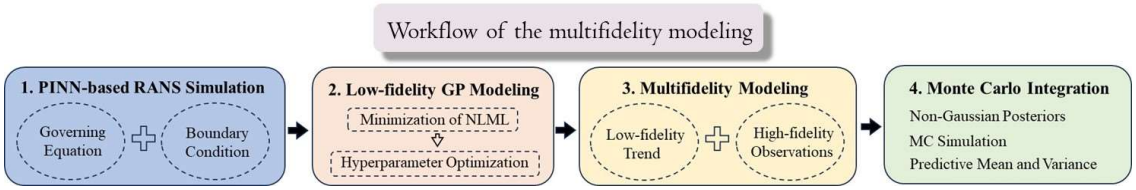


Figure 4-3. Workflow of the multifidelity strategy.

4.3 Results and Discussions

4.3.1 Case 1: Flow past a single hill (Reynolds number: 60000)

This case study uses data from a fluid dynamic experiment that is publicly available (Almeida *et al.* 1993). A fully developed channel flow passed through a single hill which was located 6 meters downstream from the inlet of the water channel in the experiment.

Around the hill, time-averaged flow velocities were measured and recorded in both horizontal and vertical directions, which will later serve as a standard test database to verify the feasibility of our proposed multifidelity flow field reconstruction strategy. In the experiment, the Reynolds number reached 6×10^4. This case study will use the low-fidelity PINN predictions and the high-fidelity measurements to reconstruct the mainstream flow velocity under the multifidelity strategy.

Making use of the RANS equations and the boundary conditions, a PINN is first formulated, which can offer a solution to the two-dimensional time-averaged flow field around the hill. As aforementioned in Chapter 2, the introduction of the Reynolds stress terms makes the RANS equations no longer a closed-form system of equations.

To close RANS equations, the Chen model proposed by Chen and Xu (1998) is adopted which takes the form as that in Eq. (3-44). Under the PINN framework, an approximate solution to the time-averaged flow field around the hill is provided first. As described by Casey and Wintergerste (2000), the recommended computational domain configuration, except that the downstream boundary is defined as a zero-pressure outlet, is adopted in this case study. A more detailed description on the configuration of computational domain for PINN calculations can be found in Figure 4-4.

In order to map the relationship between spatial coordinates and flow characteristics, a deep neural network that contains six hidden layers, each with forty neurons, is employed. The Adam optimizer with a steady learning rate of 3×10^{-4} as well as the Tanh activation function are adopted in this case study to simulate the two-dimensional flow field.

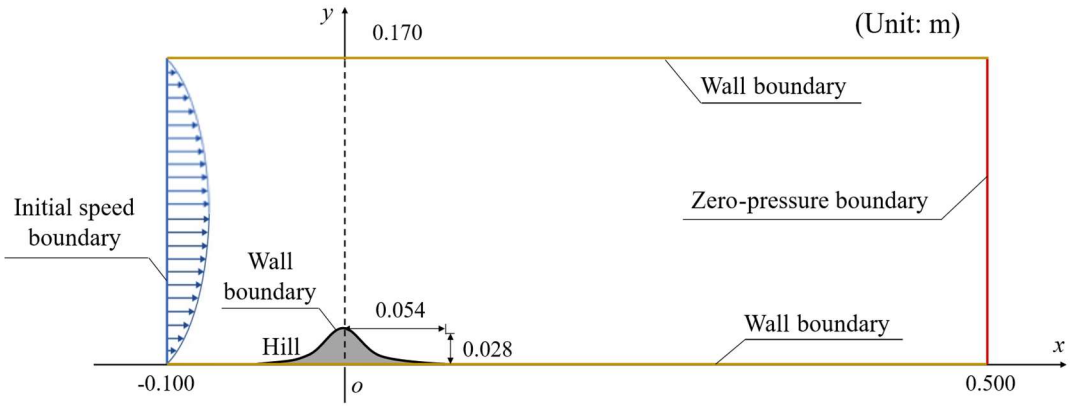


Figure 4-4. Computational domain of the flow past a two-dimensional hill.

Inside the domain, 100 equally-spaced collocation points are sampled along the x-axis and y-axis respectively. A lattice of collocation points with a 100 × 100 size is thus generated. Among these points, 254 points are located inside the two-dimensional hill, so they have been excluded, which results in the number of collocation points reducing to 9766 inside the computational domain. Meanwhile, there are four distinct boundaries in this case, which are an inlet boundary, an outlet boundary, a symmetry boundary and a wall boundary (both the hill surface and the ground). On each boundary, 500 equally-spaced collocation points are sampled (on the wall boundary, the projections of the distances between collocation points on the x-axis, instead of the distances themselves, are equal). Thus, there are 2000 collocation points on the domain boundaries to assist in the calculation the residuals of the boundary conditions.

More specifically, the residuals of the Chen model-based RANS equations calculated at the 9766 domain collocation points form the loss term L_f defined in Eq. (4-2), and the

residuals of the boundary conditions calculated at the 2000 boundary collocation points form the loss term L_b defined in Eq. (4-3). By minimizing these physical constraints, the configured PINN realizes its function of offering approximate solutions to this flow problem.

The PINN predictions of the mainstream velocity component u after 1×10^5 training iterations are shown in Figure 4-5(a), which are depicted as an orange curved surface, compared with the red dots that represent the experiment measurements. Figure 4-5(b) depicts the contour of the velocity component u based on the PINN prediction, which is also compared with the reference data from the experiment. As can be seen in Figure 4-5, there exists a significant difference between the PINN predictions and the experimental results.

A conclusion can be easily drawn from Figure 4-5 that, without incorporating measurement data to train the PINN, its solution can only be viewed as low-fidelity approximation. To establish the proposed multifidelity model for predicting the flow field around the two-dimensional hill, then 900 uniformly distributed low-fidelity sampling points (a 30 × 30 lattice) generated by the PINN prediction are selected.

Meanwhile, among the total 325 experimental measurement points scattered in the computational domain, 35 points are picked out and the measured mainstream velocities at these points are considered as high-fidelity training data. The spatial coordinates of the 35 high-fidelity training points are tabulated in Table 4-1. In selecting the training points, the principle of distributing the training points over the whole computational domain as evenly as possible is abided by. A multifidelity model is then established using the NIF algorithm. To obtain the Gaussian predictive posterior distribution of the mainstream velocity component u, a GP regression model is first trained based on the low-fidelity data. With the randomized restart strategy, the marginal log-likelihood to seek optimal hyperparameters is maximized.

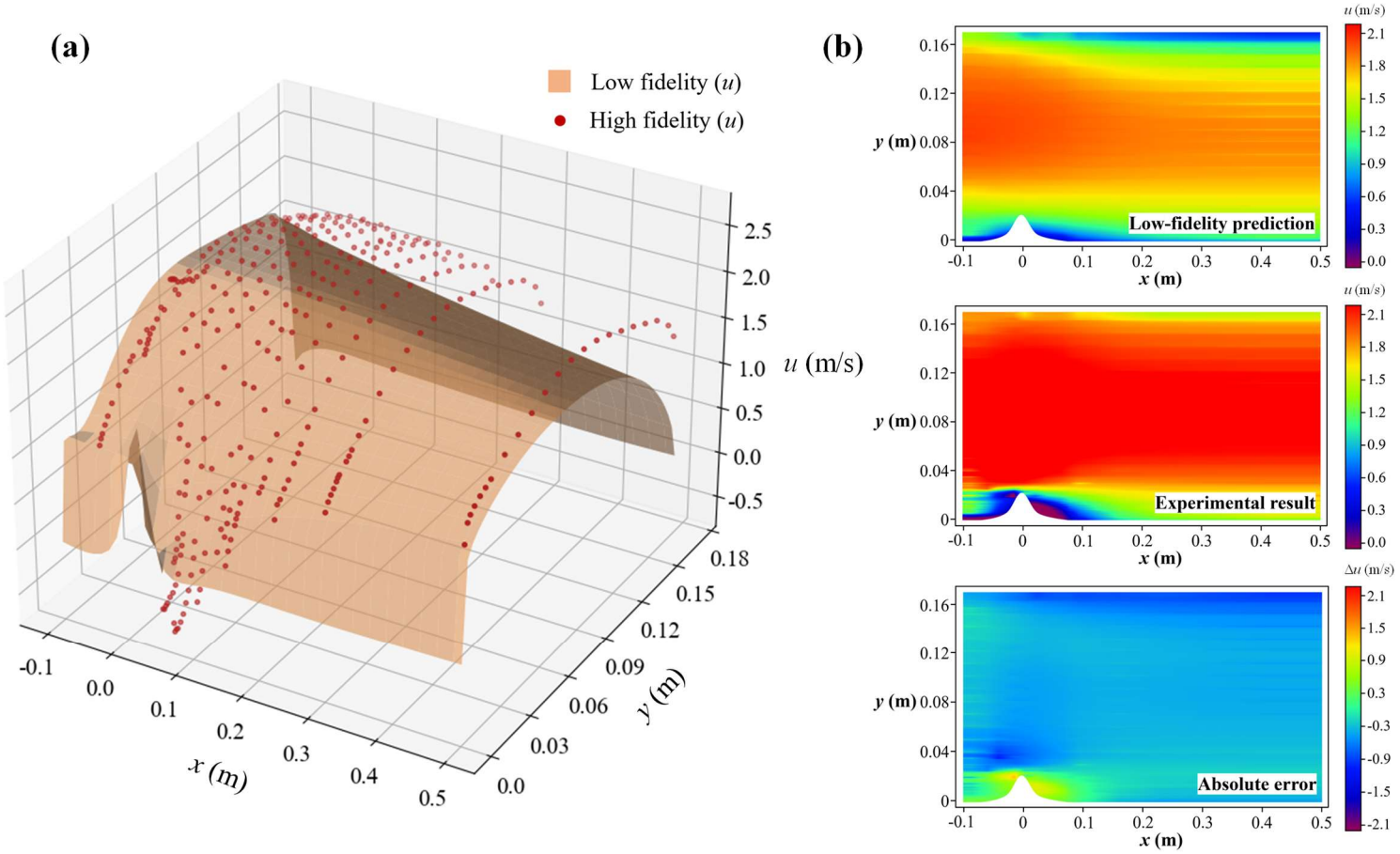


Figure 4-5. Prediction of *u* using the low-fidelity model in Case 1: **(a)** general view; **(b)** the velocity contour compared with the experimental counterpart (absolute error = prediction – experimental result).

Table 4-1. Spatial coordinates of the high-fidelity points used for multifidelity modeling.

***x* coordinate (m)**	-0.050	0.050	0.150	0.300	0.500
	0.006	0.002	0.001	0.001	0.001
	0.015	0.015	0.015	0.016	0.016
	0.030	0.030	0.030	0.030	0.030
***y* coordinate (m)**	0.070	0.070	0.070	0.070	0.070
	0.100	0.100	0.100	0.100	0.100
	0.130	0.130	0.130	0.130	0.130
	0.160	0.165	0.165	0.165	0.165

With the Gaussian posterior distribution acquired on the low-fidelity level, the training data for the high-fidelity GP regression model can be generated. Hyperparameters are optimized again by maximizing marginal log-likelihood using the kernel function in Eq. (4-6). With Monte Carlo integrations, the posterior distribution of mainstream velocity component u on a high-fidelity level can be obtained after the model has been fully trained. In Figure 4-6, the prediction results of the velocity component u using the proposed multifidelity model are depicted. The multifidelity model's prediction of the mainstream velocity component u is represented by the green curved surface. The yellow curved surface represents low-fidelity data based on PINN predictions. These red dots correspond to high-fidelity experimental results on a total of 325 measurement points for the mainstream velocity component u. The low-fidelity data (PINN predictions) and multifidelity model predictions show a similar trend across the whole computational domain. However, the latter is much closer to high-fidelity measurement

data. The reason for this is that our proposed multifidelity model can precisely capture the nonlinear nonfunctional space-dependent cross-correlations between the low-fidelity and high-fidelity data sets. Hence, the multifidelity model can fit the scattered data points on the high-fidelity level based on the trend of the low-fidelity PINN predictions. To put it another way, the scattered high-fidelity data points are used to correct the low-fidelity prediction surface while maintaining the trend as much as possible.

Figure 4-7 compares the high/low fidelity data with the multifidelity model prediction on twelve vertical lines within the computational domain. Across the entire computational domain, the multifidelity model predictions are consistent with the high-fidelity data (experiment measurements). Due to the 35 high-fidelity training data points scattered along lines x = -0.050 m, x = 0.050 m, x = 0.150 m, x = 0.300 m, and x = 0.500 m, the results are even better on those lines.

The multifidelity model, however, still shows competitive results in comparison to PINN, especially when it comes to prediction accuracy. In addition, the performance of the proposed multifidelity model and the other two widely used strategies are compared in Table 4-2. In addition to embedding the physical governing equations and boundary conditions into the total loss, the 35 high-fidelity training data in Table 4-1 are also embedded into the neural network training process when the data-driven PINN strategy is adopted. The same configuration is adopted for the data-driven PINN as that in the low-fidelity modeling. A regression task based on the 35 high-fidelity training data points is all that is required in the neural network strategy. The L-BFGS optimizer with a learning rate of 5×10^{-4} is used to train a neural network with only one hidden layer containing 10 neurons. The ℓ_2 error is still used here for the evaluation of prediction accuracy on a quantitative level, and the results are shown in Table 4-2.

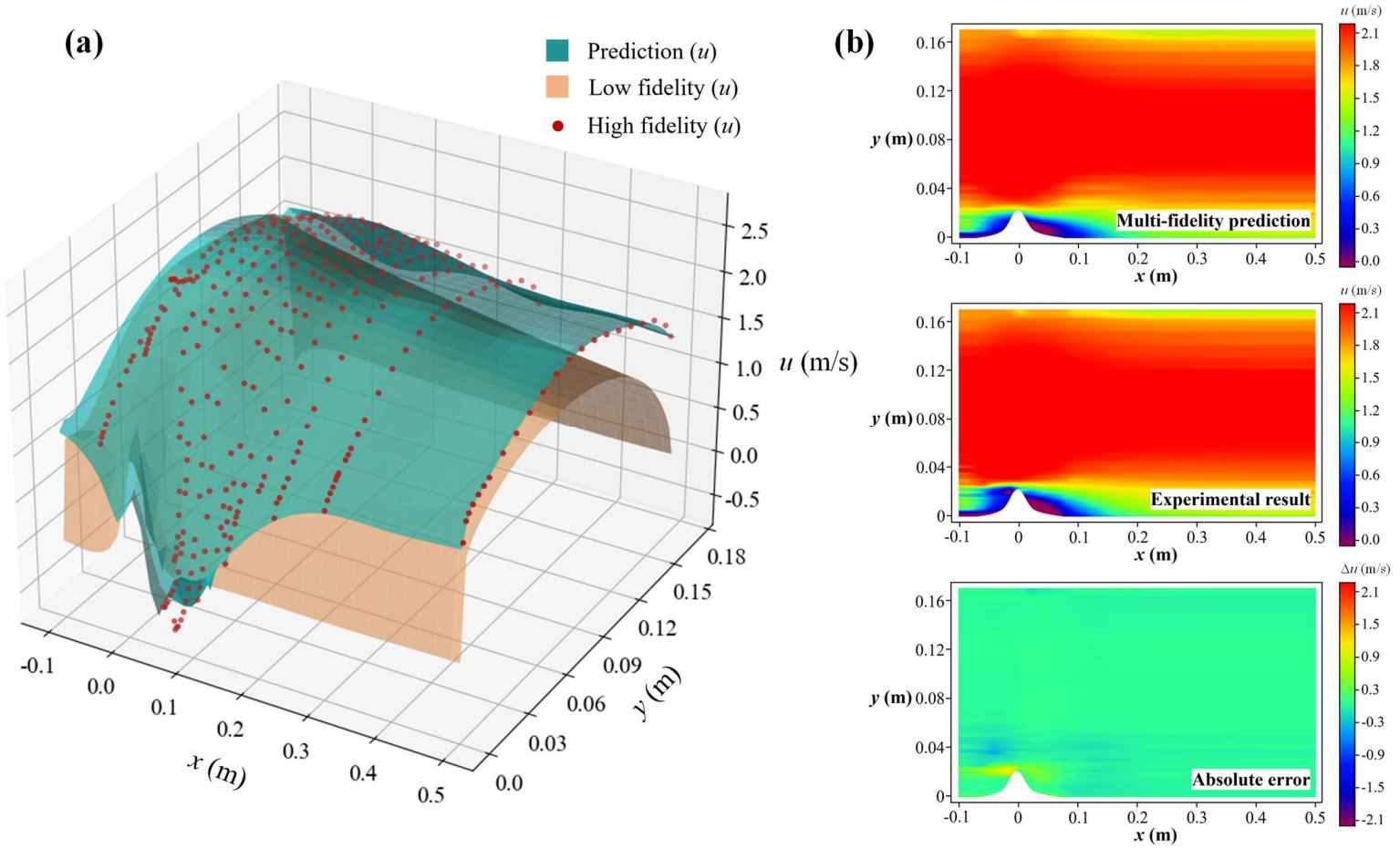


Figure 4-6. Prediction of u using the multifidelity model in Case 1: **(a)** general view; **(b)** the velocity contour compared with the experimental counterpart (absolute error = prediction – experimental result).

Considering the low-fidelity PINN model can be fully trained and well-prepared in advance since no measurement data is required, multifidelity modeling takes around 53 seconds in the training process. An ℓ_2 error of 24.6% is obtained with the data-driven PINN strategy, in comparison. PINN may also be hampered by its high computational cost, which may prevent its widespread adoption. When using the Adam optimizer, a data-driven PINN with 6 hidden layers, each with 40 neurons, usually takes around 2.3×10^4 seconds to train with 1×10^5 iterations. This can be a heavy burden in practical use. Reconstructing the flow field from the neural network strategy is faster, but when its predictions are compared to experiments, its relative error becomes unbearable. The ℓ_2 error reaches 55.3% after 41 seconds of computing when the flow field is reconstructed using the neural network strategy using 35 high-fidelity training data points distributed throughout the computational domain.

It must be admitted that the computational cost of the PINN-related strategy may become a stumbling block to its wide application. Considering a PINN with 6 hidden layers, each with 40 neurons, it usually takes around 2.3×10^4 seconds for its training process with 1×10^5 iterations when using the Adam optimizer, which would be a heavy burden in engineering applications. For the multifidelity model, it takes an additional 53 seconds for multifidelity modeling in considering that the low-fidelity PINN model can be fully trained off-line because no measurement data is needed in this process. Based on the above comparisons, it can be concluded that our proposed multifidelity model demonstrates the most competitive performance for reconstructing the flow field around the two-dimensional single hill without considering computing resources. It is worth noting that the experimental data of v is unevenly distributed and insufficient to support multifidelity modeling in this case, so the issues regarding the velocity component in y-direction are not considered here.

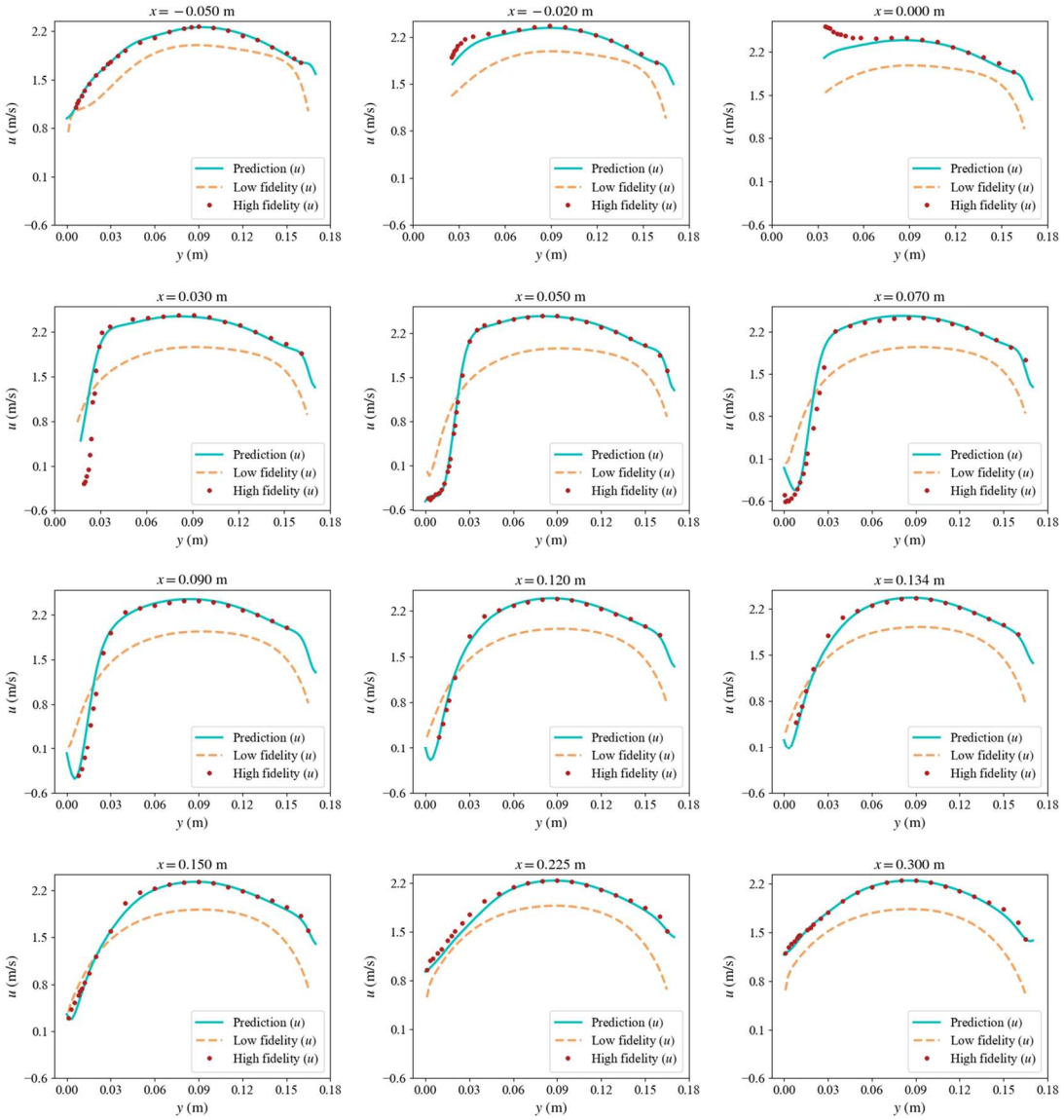


Figure 4-7. Comparison of the results between the multifidelity model prediction and the high/low-fidelity data on twelve vertical lines.

Table 4-2. Performance of different flow field reconstruction strategies.

	Multifidelity Model	**Data-driven PINN**	**FCNN**
ℓ_2 error	9.8%	24.6%	55.3%
Computing Time (s)	5.3×10^{1}	2.3×10^{4}	4.1×10^{1}

4.3.2 Case 2: Square cylinder flow (Reynolds number: 21400)

Based on the experiment conducted by Lyn and Rodi (1994), this case study describes a turbulent flow around a two-dimensional square cylinder, as shown in Figure 4-8. In the middle of the computational domain, which measures 0.44 m in length and 0.32 m in width., a 0.04 m × 0.04 m square cylinder is located at the left center.

According to Figure 4-8, point A is located at the bottom left corner of the square cylinder at a coordinate of (0.10, 0.14). There is an initial speed boundary on the left boundary of the computational domain, where the fluid velocity stabilizes at 0.535 m/s. The right boundary of the computational domain is a zero-pressure outlet, while upper and lower boundaries are defined as symmetry wall boundaries. As the fluid velocity equals zero at the surfaces of the square cylinder, they are considered to be wall boundaries. In the experiment, there were 517 experimental measurement points capturing the time-averaged flow velocity components *u* and *v* inside the flow field.

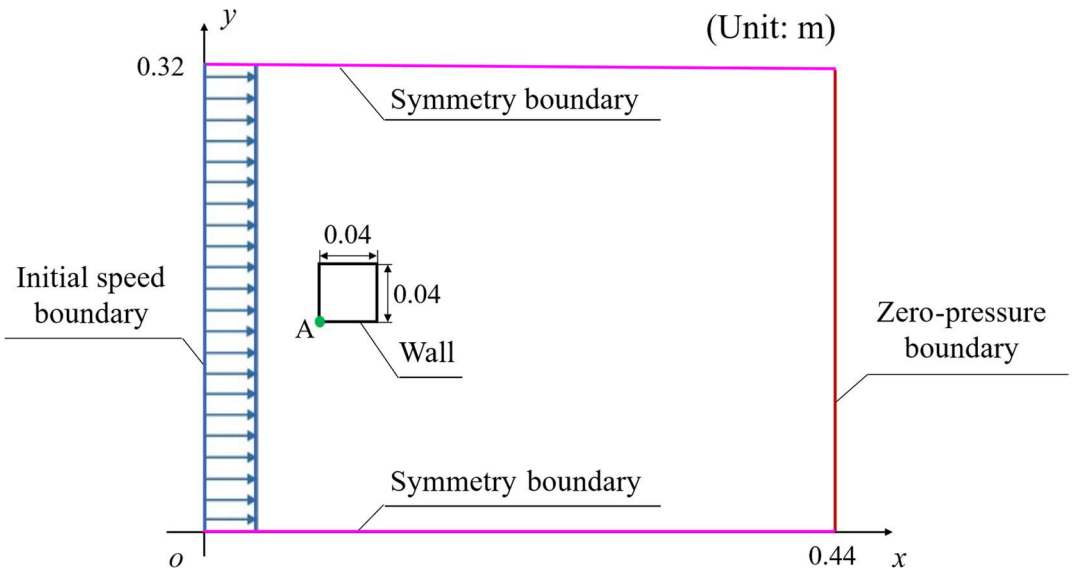


Figure 4-8. Computational domain for the flow past a two-dimensional square cylinder.

Generally, Case 2 and Case 1 differ significantly in two aspects. First, the most intuitive aspect is the different geometric appearances. Second, the velocity component v, in addition to u, is assumed to be known on the high-fidelity points that are utilized to establish the multifidelity model. From the 517 high-fidelity measurement points, 36 points scattered along nine vertical lines are selected for training the multifidelity model. Table 4-3 tabulates their spatial coordinates. In selecting training points, the principle of distributing the training points over the whole computational domain as evenly as possible is still abided by.

Table 4-3. Spatial coordinates of the high-fidelity points used for multifidelity modeling.

***x* coordinate (m)**	0.000	0.100	0.155	0.200	0.250	0.300	0.350	0.392	0.400
	0.200	0.200	0.160	0.160	0.160	0.160	0.160	0.160	0.160
	0.240	0.240	0.180	0.200	0.200	0.200	0.200	0.200	0.200
***y* coordinate (m)**	0.280	0.280	0.200	0.240	0.240	0.240	0.240	0.240	0.240
	0.320	0.320	0.220	-	-	0.280	-	0.280	0.280
	-	-	-	-	-	0.320	-	0.320	0.320

The objective of Case 2 remains to apply all available physical restrictions and sparse measurement information to reconstruct the high-fidelity mainstream velocity u within the entire computational domain. As the data of v is also included in the training process (already known), the number of schemes has increased from three to five. In Case 2, the performances of five distinct schemes are horizontally compared to achieve this objective.

For the first three schemes, the NIF algorithm and sparse u measurements are utilized for establishing the multifidelity model. The only difference between these three schemes is the

low-fidelity data source. A PINN without training data (purely physics-based), a PINN with v embedded in its training process, and CFD are used as the low-fidelity data sources, respectively. The fourth scheme is a data-driven PINN which both u and v are embedded in to train the neural network parameters, and the NIF algorithm is not engaged in Scheme 4. The fifth scheme is an FCNN which is similar to that mentioned in Case 1. For the sake of explanation, the five strategies are now clarified in Table 4-4 as follows.

Table 4-4. Five flow field simulation strategies in Case 2.

Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5
PINN+NIF	v-embedded PINN+NIF	CFD+NIF	u v-embedded PINN	FCNN

It is meaningful to go into further detail on the PINN and CFD frameworks separately due to the multiple sources of the low-fidelity data in this case study. For the PINN framework, its configuration remains the same as that in the previous case except that there are only 108 collocation points to be excluded inside the two-dimensional square cylinder, which ultimately leads to a total of 9892 internal collocation points. In addition, there are 50 boundary equally-spaced collocation points on the initial speed boundary, zero-pressure outlet, upper symmetry boundary, lower symmetry boundary, and each of the four side surfaces of the two-dimensional square cylinder, respectively. Thus, a total of 400 boundary collocation points is used to calculate the physical residuals in this case. For the CFD framework, the simulation of the time-averaged flow field is performed based on the commercial software *Star CCM+* in this study, and the mesh inside the computational domain consists of more than 36484 cells. For both the PINN and CFD frameworks, the standard k-ε model is adopted in RANS turbulence modeling, which has already been introduced in detail in Chapter 2.

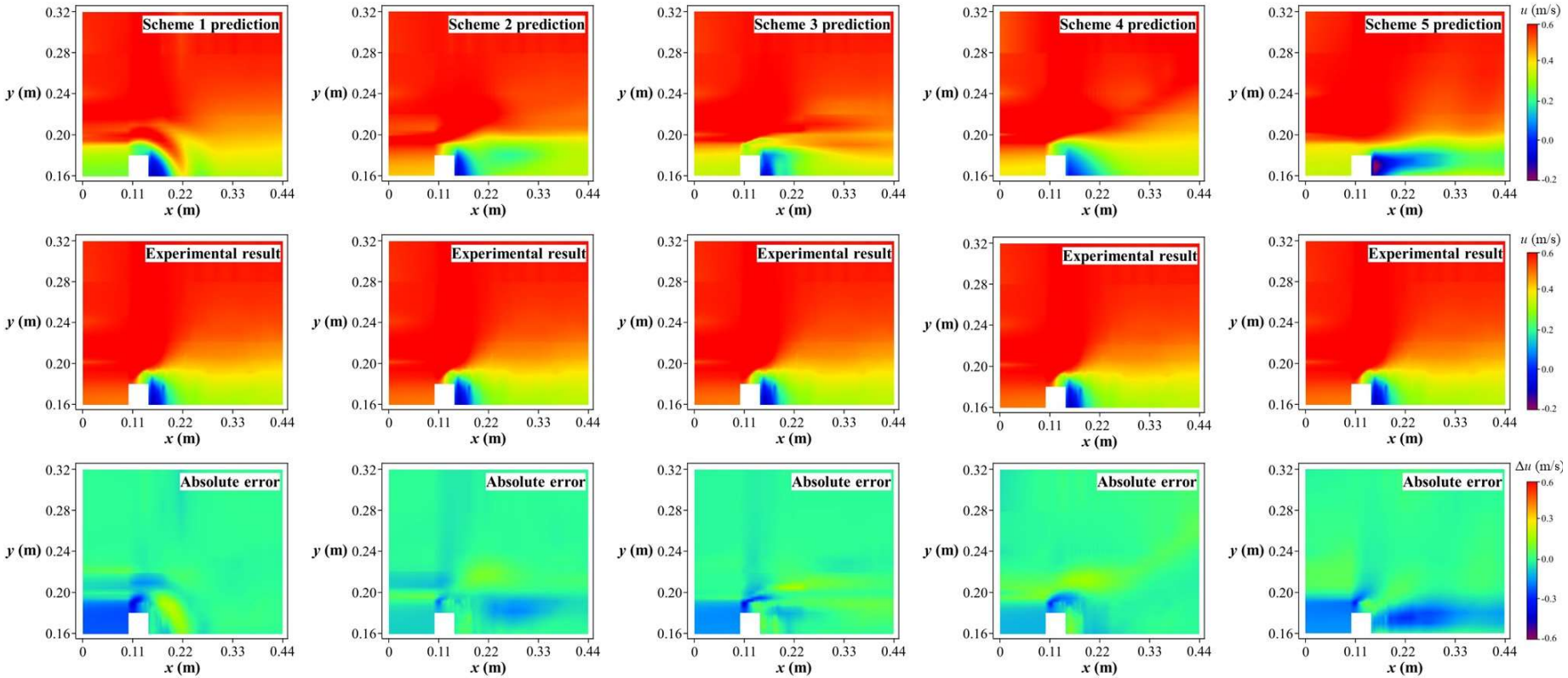


Figure 4-9. Prediction of *u* using five distinct schemes: (top) prediction; (middle) experimental result; (bottom) absolute error.

The prediction results of the mainstream velocity u from different schemes are depicted in Figure 4-9, which are also compared with the experiment results. These velocity contours show that Scheme 2, notably in the upstream region of the square cylinder, reconstructs the fluid features most accurately and efficiently. This may be due to the fusion of measured fluid features from upstream regions in Scheme 2 in low-fidelity modeling. Again, ℓ_2 error is used to quantitatively evaluate the accuracy of predictions from different models. As tabulated in Table 4-5, the ℓ_2 error of Scheme 2 is the lowest among those of all schemes, which is only 8.8% compared with the experimental results.

In particular, when v-embedded PINN, instead of CFD, is used as the low-fidelity data source, the ℓ_2 error has decreased by 6.6%. This is because measurement information has been included, and compared to CFD, the simulation results from v-embedded PINN are more accurate. However, when field information is unavailable, the precision of the PINN's prediction drops dramatically. This resulted in the ℓ_2 error of Scheme 1 being 18.1%, which is the worst among those of the multifidelity models.

The significance of physical information fusion in neural network modeling should be emphasized, nevertheless, as it is clear that Scheme 1 has an improved accuracy of 10.2% over Scheme 5. By the way, it can be observed that all strategies involving PINN, i.e., Scheme 1, 2, and 4, are time-consuming, compared to CFD. Undeniably, this is one of the main drawbacks of PINN, which urgently needs to be addressed by further research. Figure 4-10 also illustrates the verification results. Across the entire computational domain, the multifidelity model predictions (Scheme 2) show good agreement with high-fidelity data (experiment measurements). The accuracy of the multifidelity model is still evaluated quantitatively by the ℓ_2 error.

Table 4-5. Performance of different flow field reconstruction strategies.

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5
ℓ_2 error	18.1%	8.8%	15.4%	15.5%	29.1%
Computing time (s)	2.8×10^4	2.8×10^4	5.4×10^2	2.8×10^4	2.1×10^1

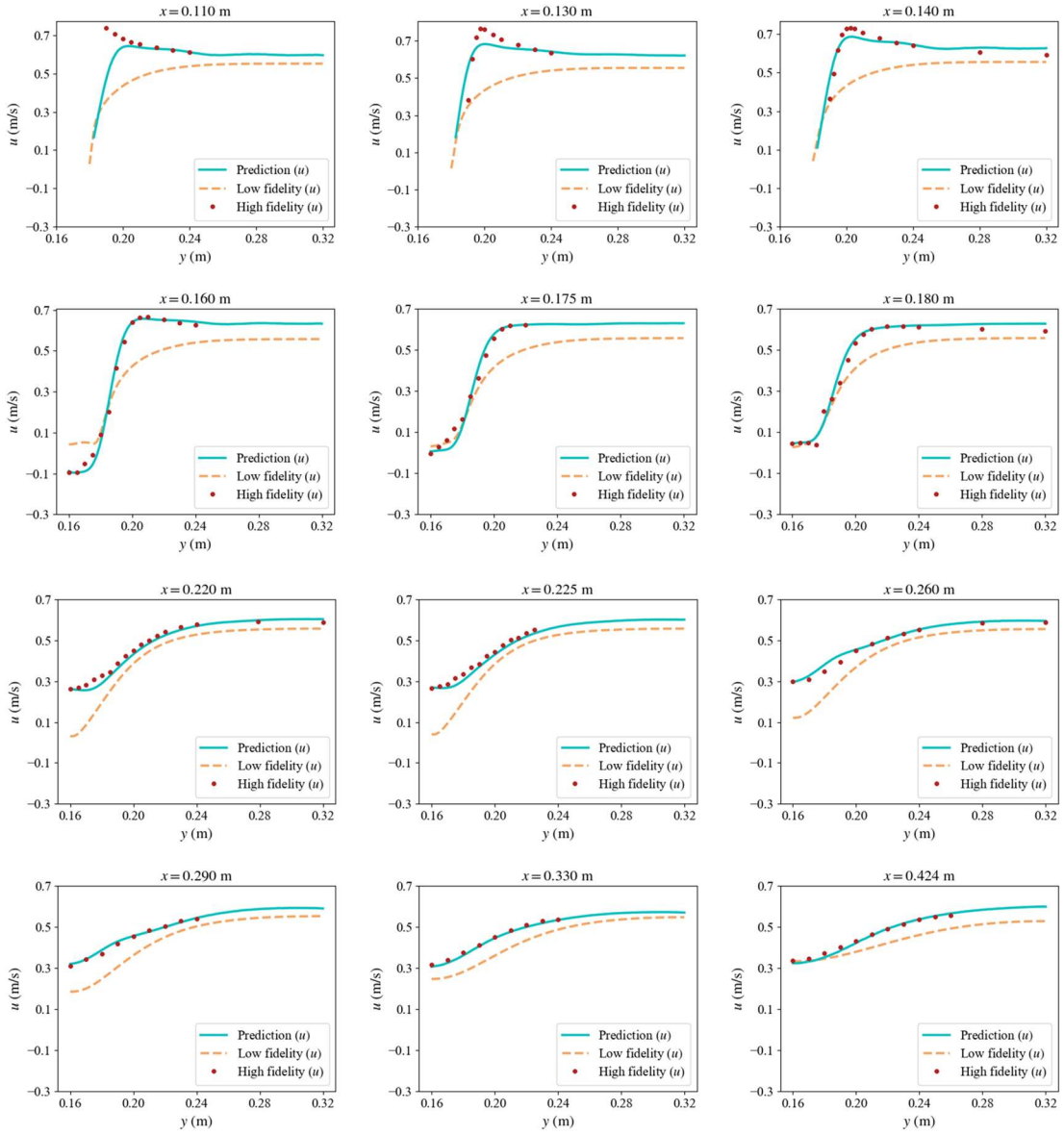


Figure 4-10. Comparison of the results between the multifidelity model prediction (Scheme 2) and the high/low-fidelity data on twelve vertical lines.

4.4 Conclusions

Based on PINN and the NIF algorithm, a novel time-averaged turbulent flow field reconstruction strategy is proposed in this chapter. In conclusion, this strategy is a two-step process where low-fidelity data are first generated by the PINN framework without measurement data embedded. By using the NIF algorithm, the sparse experimental or field measurements are regarded as high-fidelity data, which can be used to build a multifidelity model. Our proposed multifidelity model is verified using flows past a hill and a square cylinder, and their results indicate that the strategy is feasible. The following preliminary conclusions can also be drawn:

- Even though only a small portion of data can be used to reconstruct the time-averaged turbulent flow field, embedding the measurements and physical laws into the neural network can still provide the missing airflow information across the whole computational domain.
- Since both PINN predictions are poor in accuracy, they are considered low-fidelity data. The low-fidelity NIF-based prediction surface can then be corrected using high-fidelity data. The multifidelity model can learn from the trend of the low-fidelity PINN predictions, and then fit the dispersed data points on the high-fidelity level and achieve strong agreement with the test data.
- The method proposed in this chapter for flow field reconstruction exhibits the most competitive outcomes when compared to other reconstruction methods. When compared to the experimental observations, the mainstream velocity component's relative errors from the multifidelity prediction are less than 10% in both cases. Considering that the low-

fidelity PINN model is fully trained in advance, it often takes less than one minute to train the multifidelity model.

- Compared with other flow field reconstruction strategies, the proposed strategy demonstrates the most competitive results. The relative errors of the mainstream velocity component from the multifidelity prediction are less than 10% in both cases relative to the experimental measurements. However, all strategies involving PINN are time-consuming, and it usually takes additional time to establish the multifidelity model, which is a challenging issue to urgently resolve.

CHAPTER 5

QUANTUM LAYER INTEGRATION

5.1 Foreword

Quantum machine learning is an intersection of two cutting-edge scientific frontiers: quantum computing and machine learning (Schuld *et al.* 2015, Schuld and Killoran 2019). It leverages the principles of quantum mechanics to improve the algorithms used in machine learning, potentially leading to faster processing and the ability to easily handle complex datasets (Ciliberto *et al.* 2018, Kavitha and Kaulgud 2024). This field is still in its infancy, but it holds promise for significant advancements in areas where classical machine learning algorithms struggle, such as optimization problems and pattern recognition in vast amounts of data (Khan and Robles-Kelly 2020, Abbas 2024).

Variational quantum circuits (VQCs) are a class of quantum algorithms designed for optimization problems on quantum computers (Cerezo *et al.* 2021, Griol-Barres *et al.* 2021). They leverage a hybrid approach, combining classical optimization techniques with quantum computation to find solutions to complex problems. These algorithms are particularly suited for Noisy intermediate-scale quantum (NISQ) computers, which are the current generation of quantum devices (Huembeli and Dauphin 2021, Bharti *et al.* 2022). They use a parametrized quantum circuit, which is adjusted iteratively by a classical optimizer to minimize a cost function. This method has been applied in various algorithms, such as the Quantum approximate optimization algorithm (QAOA), which are promising for solving real-world

problems using quantum computing (Guerreschi and Matsuura 2019, Nakhl *et al.* 2024). The structure of a VQC is pivotal in quantum computing. A well-designed circuit can efficiently simulate quantum systems and solve complex problems (Ostaszewski *et al.* 2021, Du *et al.* 2022). Typically, a VQC comprises two layers: the initialization layer and the variational layer. The initialization layer prepares a quantum state that approximates the problem's solution, while the variational layer consists of parametrized gates that are tuned to minimize a cost function, often related to the problem's Hamiltonian. This structure allows for the adjustment of parameters to find the lowest energy state of the system, which is essential in many quantum algorithms.

Physics-informed machine learning is a prominent research area at the moment (Raissi *et al.* 2019, Yuan *et al.* 2022, Rui *et al.* 2023, Rui *et al.* 2024, Zeng *et al.* 2024), and the field of quantum machine learning is no exception (Lloyd *et al.* 2020, Lubasch *et al.* 2020, Kyriienko *et al.* 2021). Some studies convert differential equations into Ising models and use QAOA methods to solve them (Albino *et al.* 2022).

However, although these methods are ingenious, their accuracy is generally not high due to the limitation of the number of qubits. Thus, the prevailing approach is to utilize VQCs to approximate the solution of the physical equations and optimize the trainable parameters in VQCs using physical residuals as costs. For instance, Siegl *et al.* (2023) propose the concept of physics-informed variational quantum circuits (PIQC), which investigates the use of quantum circuits for solving differential equations. They compare a classical approach, i.e., PINN, with its quantum counterpart, i.e., PIQC, and discuss their performances and convergence properties. Their results indicate that PIQC has demonstrated superior convergence speed and accuracy compared to PINN in certain situations. However, it should also be noted that PIQC currently falls short of achieving performance similar to PINN in

complex problems. This is due in part to the belief that PIQC struggles to accurately approximate nonlinear functions (Schuld *et al.* 2021). Additionally, the increase of qubits in quantum circuits poses challenges in terms of computational costs.

Hence, the author believes that physics-informed quantum machine learning necessitates utilizing a more intricate classical-quantum hybrid model, rather than relying on a single VQC when resolving intricate issues like high-order ordinary differential equations (ODEs) and PDEs. Some studies attempt to integrate classical neural networks with VQCs to obtain more expressive models, such as the dressed quantum circuit in (Mari *et al.* 2020), the quantum Helmholtz machine in (Benedetti *et al.* 2018), and the variational quantum classifier in (Adhikary *et al.* 2020).

In comparison with a VQC, these hybrid models offer notable benefits. For example, the neural network-based pre-processing and post-processing modules in a dressed quantum circuit enable itself to adapt to different input and output formats, which significantly improves the model's flexibility and nonlinear expression capability. Given both classical neural networks and VQCs are inherently differentiable, these hybrid models also possess inherent differentiability. This property allows for the seamless utilization of the crucial AD function in classical physics-informed machine learning (Baydin *et al.* 2018, Margossian 2019). Naturally, some physics-informed hybrid models have emerged, which have preliminarily solved some issues such as laminar flow simulation (Dehaghani *et al.* 2024, Sedykh *et al.* 2024). This study takes into account the nonlinear expression capability and generalization performance of the dressed quantum circuit and uses it as the function-fitting module of the physics-informed hybrid model.

A novel physics-informed hybrid classical-quantum neural network (PIHCQNN) is

therefore proposed to solve ODE and PDE problems. In addition, the initial stage of addressing the inverse problem is also explored in this research. In the following text, the basic principles of the variational quantum algorithm, the dressed quantum circuit, the physics-informed machine learning, and the proposed PIHCQNN will be explained to readers in Section 5.2. The performance of PIHCQNN in addressing three forward PDE problems and an inverse ODE problem will be demonstrated in Section 5.3, which is also contrasted to that of classical PINNs. In Section 5.4, the phenomenon that PIHCQNN can potentially accelerate the learning of high-frequency features will be discussed. In addition, a counterexample as well as the factors that may influence the PIHCQNN's performance will also be discussed. Lastly, this research will be comprehensively summarized in Section 5.5, which also identifies the model's current deficiencies and future research directions.

5.2 Methodology

5.2.1 Quantum computing and variational quantum algorithms

In the same way that classical computers are composed of numerous classical bits, quantum computers are composed of quantum bits, namely, qubits. As with bits, qubits possess their own states. More specifically, the state of a bit is either 0 or 1, while the quantum state of a qubit is represented by a vector that resides in a two-dimensional complex vector space. For instance, the set of computational basis states that is most often used may be expressed in the following manner

$$|0\rangle \coloneqq \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad (5-1)$$

$$|1\rangle \coloneqq \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5-2}$$

where $|\cdot\rangle$ denote the ket symbol by using the Dirac notation, which is widely used in quantum mechanics. Similarly, there is also the bra symbol $\langle\cdot|$, which is the conjugate transpose matrix of the ket matrix. The inner product of two distinct quantum state vectors $|\gamma\rangle$ and $|\varphi\rangle$ can be denoted by a parenthesis $\langle\gamma|\varphi\rangle$. For instance, the inner product of the two computational basis states listed above in Eq. (1) and (2) can be expressed as

$$\langle 0|1\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \tag{5-3}$$

Another important concept in quantum computing is quantum gates, which are the foundation of quantum computing and are similar to the logic gates in classical computers. They operate on the fundamental unit qubits of quantum information. Quantum gates manipulate these qubits through operations represented by unitary matrices, altering quantum states to perform complex quantum computations. Common quantum gates include the NOT gate and the Hadamard gate. For example, if we apply a NOT gate *X* and a Hardamard gate *H* to a qubit $|0\rangle$, then we can get

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{5-4}$$

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \tag{5-5}$$

It is worth emphasizing that there are also some parameterized quantum gates in quantum computing, such as the rotation gates which will be mentioned below. These parameterized quantum gates play a crucial role in embedding classical data in quantum circuits and parameter optimization in quantum machine learning. Finally, measurement operations are essential procedures in quantum computing that collapse the superposition of states into a single state

by determining the state of a qubit. This process is essential for the operation of quantum computers, as it is the final stage in the extraction of classical information from quantum systems. The well-known quantum circuit is a graphical representation of the application of quantum gates and the measurement operations to qubits.

Variational quantum algorithm is an analogy of machine learning in the context of quantum computing (Huang *et al.* 2022). In essence, variational quantum algorithms involve the optimization of parameterized quantum circuits, i.e., VQCs, using classical optimizers such as Adam and SGD (Moll *et al.* 2018, Ma *et al.* 2019). For a quantum classifier (Chen *et al.* 2021), if there exists a classical labeled dataset (x_k, y_k) , the cost function of the quantum classifier can be written as

$$C(\theta) = \sum\nolimits_k \left[y_k - \langle\psi_0|V^\dagger(x_k)U^\dagger(\theta)YU(\theta)V(x_k)|\psi_0\rangle\right]^2 \qquad (5-6)$$

where ψ_0 is the initial state of the qubits in the VQC. $V(\cdot)$ denotes the unitary operator in the data embedding module. $U(\theta)$ represents the parameterized VQCs with parameter θ. Y is the observable. $X^\dagger$ is the conjugate transpose matrix of X.

The main idea of a variational quantum algorithm is to encapsulate the objective problem as an optimization task that reduces the cost function of a parameterized quantum circuit by optimizing the learnable parameter θ. It is worth noting that the optimization of variational quantum circuits are generally solved by classical computers, so VQC itself can be regarded as a hybrid classical-quantum algorithm. To conclude, VQC combines classical and quantum computing, using a classical optimizer to optimize the parameters of a quantum circuit to minimize a cost function. This hybrid approach leverages the advantages of both classical and quantum computing algorithms.

5.2.2 Dressed quantum circuit

As one of the embodiments of hybrid classical-quantum algorithms, the dressed quantum circuit fuses VQCs with conventional NNs, which was first proposed by Mari and his colleagues in 2020 (Mari *et al.* 2020). A dressed quantum circuit generally comprises two classical NN layers at either end of a VQC to enable flexible data embedding and readout, as shown in the upper part of Figure 5-1. As seen in Figure 5-1, on the left-hand side of the dressed quantum circuit, a fully connected neural network (FCNN) accepts independent variables in physical equations as its inputs, which is called an embedding neural network (ENN). Similar to a classical FCNN, an ENN may consist of a few hidden layers, but the number of neurons in its last layer should always be consistent with the number of qubits in the ensuing VQC. This is because, as the red dashed line in the figure illustrates, an ENN's output will be delivered, one by one, into the data encoding layers of the VQC. In the VQC, a reuploading strategy is adopted for data embedding, where n denotes the cycling number. $R(\boldsymbol{\theta})$ in the trainable layer can be expressed as

$$R(\boldsymbol{\theta}) = Rot(\varphi,\theta,\omega) = RZ(\omega)RY(\theta)RX(\varphi) = \begin{bmatrix} e^{-\frac{i(\varphi+\omega)}{2}}\cos\left(\frac{\theta}{2}\right) & -e^{-\frac{i(\varphi-\omega)}{2}}\sin\left(\frac{\theta}{2}\right) \\ e^{-\frac{i(\varphi-\omega)}{2}}\sin\left(\frac{\theta}{2}\right) & e^{\frac{i(\varphi+\omega)}{2}}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (5-7)$$

where $\boldsymbol{\theta}$ is the rotation angle vector to be optimized in PIHCQNN's training process. Following the trainable layer are the Controlled NOT (CNOT) gates for quantum entanglement. In the data encoding layer, R_y gates are used for data embedding, which can be expressed as

$$Ry(x) = e^{-\frac{i\varphi\sigma_y}{2}} = \begin{bmatrix} \cos\left(\frac{x}{2}\right) & -\sin\left(\frac{x}{2}\right) \\ \sin\left(\frac{x}{2}\right) & \cos\left(\frac{x}{2}\right) \end{bmatrix} \quad (5-8)$$

where σ_y denotes the Pauli-Y operator, which allows real numbers to be encoded in each qubit and quantum gate. An additional entanglement module, i.e., a single trainable layer followed by CNOT gates between each two qubits, comes after the n repetitions.

The measurement module is the last stage of the VQC. The Hamiltonian adopted in this research is a Pauli-Z operator, which ensures the differentiability of the VQC. A readout neural network (RoNN) comes after the sandwiched VQC. An FCNN structure is still preserved in a RoNN, where the number of neurons in the first layer corresponds to the qubit number, and the number of neurons in the last layer, i.e., the output layer in Figure 5-1, corresponds to the number of unknown parameters to be solved in the physical equations.

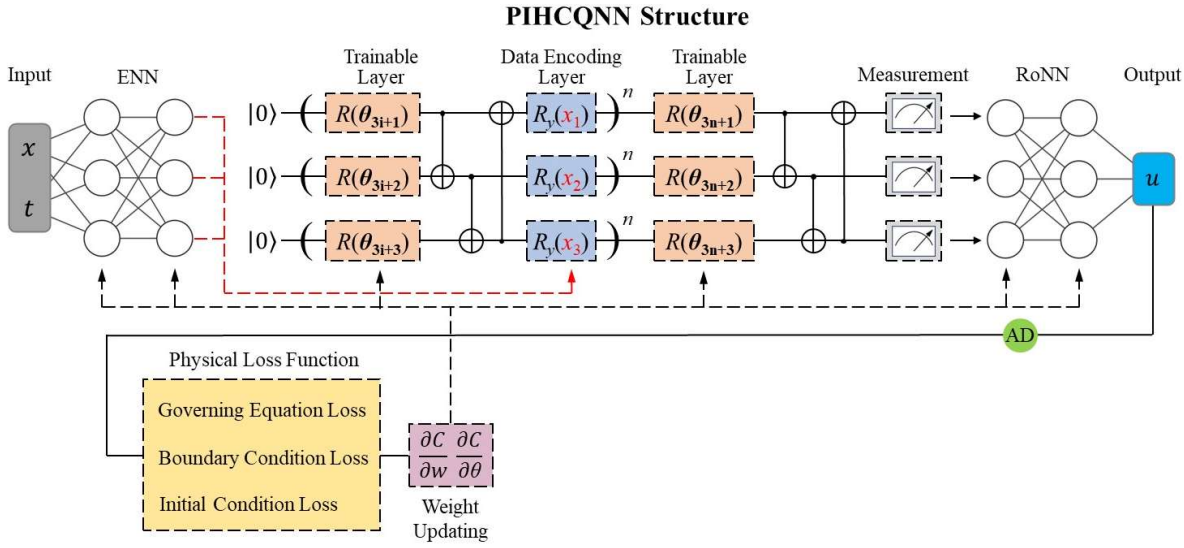


Figure 5-1. The structure of a PIHCQNN.

5.2.3 Physics-based loss function

The loss function drives the learning process of a neural network since training a neural network is to adjust the neural network's parameters, i.e., weights and biases, to minimize its loss function, effectively improving the model's predictions. One of the most significant

contributions of PINN is the use of AD to construct residuals for partial differential equations and to treat these physics-based residuals as loss components in its loss function to optimize model parameters (Raissi *et al.* 2019). In this study, analogously, the proposed PIHCQNN takes advantage of the physics-based loss function module in a PINN to approach the approximate solution to PDEs. Since VQC is also differentiable, AD still works in the PIHCQNN framework. Figure 5-1 illustrates that AD is initially applied to the RoNN outputs in order to form the partial differential terms in the physical equations. Similarly, the total residuals of physical governing equations, boundary conditions, and initial conditions will serve as the loss of a PIHCQNN, which can be expressed as follows

$$C = w_g L_g + w_b L_b + w_i L_i + w_d L_d \qquad (5-9)$$

where C represents the total loss. L_g, L_b, L_i, and L_d denote the governing equation loss component, the boundary condition loss component, the initial condition loss component, and the data loss component respectively. w_g, w_b, w_i, and w_d are weight coefficients of corresponding loss components. In this work, these loss components take a mean-squared error (MSE) form. The neural network parameters of the ENN and RoNN, along with the rotation angle vector $\boldsymbol{\theta}$ in the VQC, will be optimized by using gradient descent algorithms. Then, the objective of approximating the PDE solution may be accomplished in a PIHCQNN by minimizing the physics-based loss function.

5.3 Results and Analysis

In this chapter, a classical simulator of quantum devices running on classical computers, i.e., Pennylane, undertakes the computational tasks of the quantum circuit in the algorithm.

Pytorch undertakes the construction of neural networks, the implementation of AD, and the formation of physics-based loss functions. The NVIDIA A100 GPU device performed the computations that are detailed in this chapter. In Section 5.3.1, a PIHCQNN will be initially employed to conduct a RANS simulation using a square cylinder flow. Then, in Section 5.3.2, PIHCQNN will be used to resolve a forward PDE problem in thermodynamics. Subsequently, in Section 5.3.3, another forward problem involving the Poisson equation will be used to evaluate the performance of PIHCQNN. Finally, PIHCQNN will be employed to resolve an inverse ODE problem in order to determine material parameters in Section 5.3.4.

5.3.1 Forward problem: RANS equations

The cylinder flow described in Section 4.3.2 is adopted here again to validate the feasibility of the proposed PIHCQNN. All settings for calculation remain the same, except for a slight expansion of the calculation domain. In this case, the computational domain has a length of 0.8 m and a width of 0.56 m, which is shown in Figure 5-2. There is also a little variation in the quantity of collocation points.

In this case, the collocation points inside the domain are arranged in a 50×50 matrix, additionally with 50 on each boundary, which results in a total of 2492 collocation points for calculating the residuals of governing equations (after removing the collocation points inside the square cylinder) and 240 for calculating the residuals of boundary conditions. As shown in Figure 5-2, point A is the bottom left corner of the square cylinder, whose spatial coordinate is (0.18, 0.26). The training data on the points tabulated in Table 4-3 are used for supervised learning in this validation case to execute a data-embedding strategy, assisting in the training of the PIHCQNN and PINN. The Chen model remains the turbulence model to close the RANS equations in this validation case.

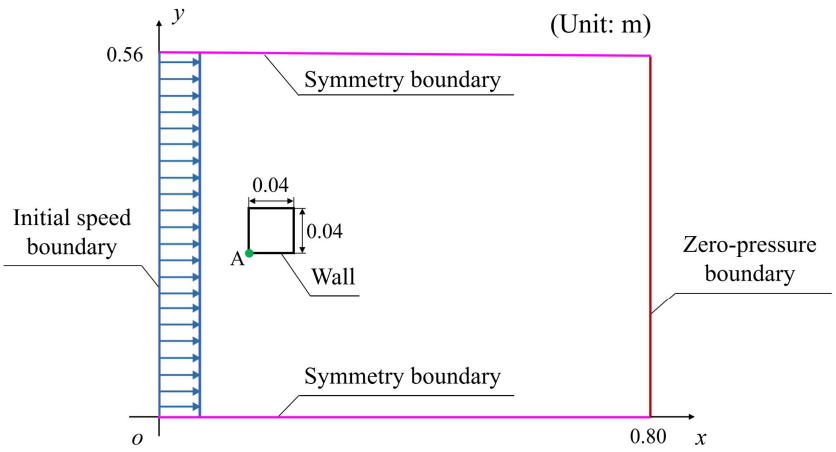


Figure 5-2. Computational domain for the two-dimensional square cylinder flow.

An experiment is conducted to evaluate the accuracy of PIHCQNN and PINN in solving such a PDE problem. Being a part of the PIHCQNN structure, ENN is an FCNN consisting of three layers. The input layer (first layer) of the ENN has two neurons that represent the spatial coordinate x and y respectively. The ENN also consists of a hidden layer (second layer) and an output layer (third layer), comprising 10 and 5 neurons respectively. RoNN, like the ENN, is also an FCNN consisting of three layers: an input layer, a hidden layer, and an output layer. However, in contrast to ENN, there are 5 neurons in the RoNN input layer, 10 neurons in its hidden layer, and three neurons in the output layer, which denotes velocity components u, v, and pressure p. For the sandwiched VQC structure, a quantum circuit consisting of five qubits is simulated, with a cycling number n of 1.

Two structures are adopted for the PINN used in this chapter for comparison, which are depicted in Figure 5-3 below. One is to substitute the VQC in the PIHCQNN with an identity matrix, that is, to directly transmit the output of each neuron in the output layer of the ENN to those in the input layer of the RoNN one by one. To be precise, this equals a 5-layer FCNN structure with three hidden layers, with a width of [2, 10, 5, 10, 3]. The other one is to substitute the VQC with a linear layer (for example, torch.nn.Linear in *Pytorch*) and a nonlinear

transformation using an activation function, which equals a 6-layer FCNN structure with four hidden layers, with a width of [2, 10, 5, 5, 10, 3]. Here, a 6-layer FCNN has 30 additional trainable parameters compared to a 5-layer structure, but an additional quantum layer entails just 10 more trainable parameters. The quantity of trainable parameters in quantum layers is far less than that in conventional fully connected layers; hence, we refrained from comparing quantum layers with deeper FCNN architectures for the sake of fairness in the subsequent discussion. The activation function is Tanh. Adam optimizer is adopted with a learning rate of 2×10^{-3}. In this case, the number of training iterations is set to 5×10^{4}. w_g is set to 1, while the other two weight coefficients w_b and w_d are set to 1 and 100. Since the RANS simulation is a steady-state process, there is no initial condition loss component in this case.

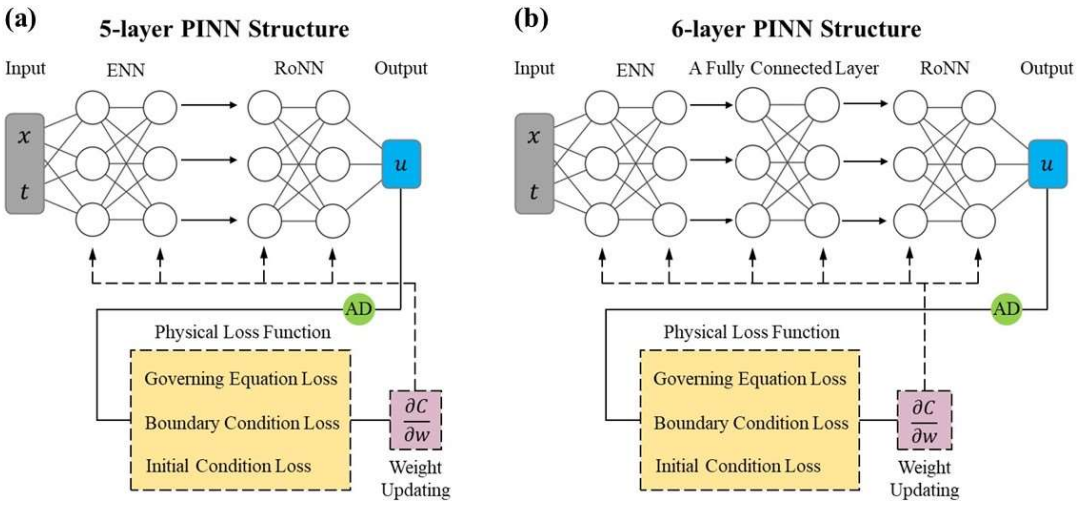


Figure 5-3. The PINN structures adopted for comparison study: **(a)** 5-layer PINN, and **(b)** 6-layer PINN.

The results are shown in Figure 5-4, which depicts the attenuation curves of the l_2 errors of velocity components u and v during the training process. In the figure, the blue curve represents the results of PIHCQNN, while the orange and green curves represent the results of 5-layer PINN and 6-layer PINN, respectively. The findings in the figure demonstrate that, at

least when examining the relative error of the fluid velocities, adding a quantum layer to the FCNN structure can slightly improve the accuracy of PINN-based RANS simulation.

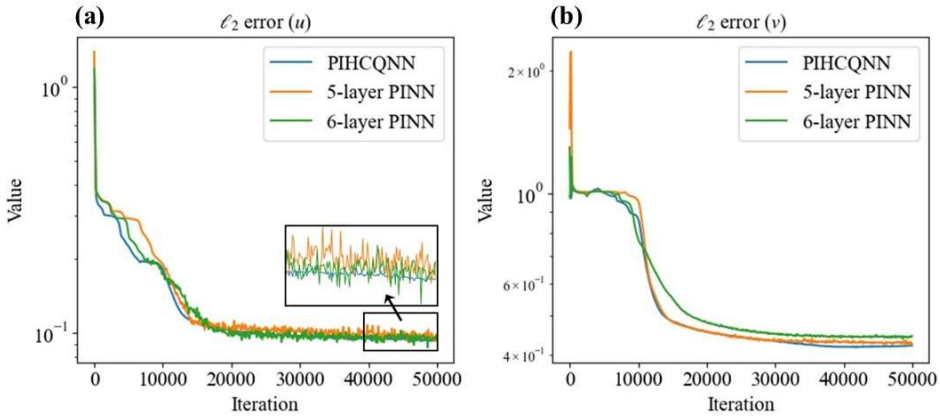


Figure 5-4. The l_2 errors of **(a)** u, and **(b)** v.

To mitigate the impact of random error resulting from a single experiment, the experiment is repeated five times, and the mean values of the l_2 errors are now tabulated in Table 5-1. Compared to the PINN models, the PIHCQNN model has a roughly 0.4% improvement in accuracy for u and a 1.3% -2.1% improvement in accuracy for v. In addition, as seen by Figure 5-4, another advantage of PIHCQNN, which is not easily noticed, is that it consistently exhibits the fastest convergence speed in the early phase of training. Overall, PIHCQNN has demonstrated a more stable and reliable performance in RANS simulations compared to PINN based on the above findings.

Table 5-1. Averaged l_2 errors of the different model's results when solving the RANS equations.

	PIHCQNN	**5-layer PINN**	**6-layer PINN**
Averaged l_2 error (u)	9.2%	9.6%	9.6%
Averaged l_2 error (v)	42.2%	43.5%	44.3%

5.3.2 Forward problem: Heat equation

Now consider a transient problem to showcase the capability of the proposed PIHCQNN to solve PDEs. The temperature $u(t, x)$ can be interpreted as a function that depends on both time t and spatial position x, which is governed by the following equations

$$c\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\kappa\frac{\partial u}{\partial x}\right) - s = 0 \qquad \text{on } T \times \Omega \tag{5-10}$$

where $c(u) = \frac{1}{2000}u^2 + 500$, and $\kappa(u) = \frac{1}{100}u + 7$. For the computational domain $T \times \Omega$, time t is within the range of [0, 0.5], while spatial coordinate x is within the range of [0, 1]. In Eq. (5-10), s takes the following form

$$s(u,t) = \frac{\kappa u}{\sigma^2} + u\frac{x-p}{\sigma^2}\left[c\frac{\partial p}{\partial t} - \frac{x-p}{\sigma^2}\left(\kappa + u\frac{\partial \kappa}{\partial x}\right)\right] \tag{5-11}$$

where $p(t) = \frac{1}{4}\cos(4\pi t) + \frac{1}{2}$, and $\sigma = 0.02$. The temperature $u(t, x)$ should also meet the following physical constraints on the domain boundaries and the initial state.

$$\frac{\partial u(t,0)}{\partial x} = \frac{\partial u(t,1)}{\partial x} = 0 \tag{5-12}$$

$$u(0,x) = \exp\left(-\frac{(x-0.75)^2}{2\sigma^2}\right) \tag{5-13}$$

The analytical solution of the heat equation is as follows

$$u(t,x) = \exp\left(-\frac{(x-(\frac{1}{4}\cos(4\pi t)+\frac{1}{2}))^2}{2\sigma^2}\right) \tag{5-14}$$

Now, an experiment is conducted to evaluate the accuracy of PIHCQNN and PINN in solving such a PDE problem. Within the structure of the PIHCQNN, ENN is an FCNN

consisting of three layers. The input layer (first layer) of the ENN has two neurons that represent the temporal variable t and the spatial coordinate x respectively. The ENN also consists of a hidden layer (second layer) and an output layer (third layer), both comprising 10 neurons. RoNN, like the ENN, is also an FCNN consisting of three layers: an input layer, a hidden layer, and an output layer. However, in contrast to ENN, there are ten neurons in the RoNN input layer, 10 neurons in its hidden layer, and merely one neuron in the output layer, which denotes temperature u. For the sandwiched VQC structure, a quantum circuit consisting of ten qubits is simulated, with a cycling number n of 2. The activation function is Tanh. Adam optimizer is adopted with a learning rate of 2×10^{-3}. In this case, the number of training iterations is set to 5×10^{4}. w_g is set to 3×10^{-7}, while the other two weight coefficients w_b and w_i are set to 1. 50 collocation points are sampled at equal intervals along the spatial and temporal axes, forming a collocation point lattice of 50 by 50.

Figures 5-5 (a) and (b) show the convergence curves of the loss functions and the l_2 errors of the temperature $u(t, x)$ concerning the analytical solution throughout the training phase, for both PIHCQNN and PINN results. After 5×10^{4} training iterations, the total loss of PIHCQNN is observed to be the smallest at 0.006. On the contrary, the ultimate total loss of the 5-layer PINN is 0.024, while that of the 6-layer PINN is 0.007. In terms of the l_2 error compared to the analytical solution, PIHCQNN yields the most negligible result of 0.046. Comparatively, the l_2 error of the 6-layer PINN result is 0.057, whereas that of the 5-layer PINN yields 0.064. To mitigate the impact of random error resulting from a single experiment, the experiment is repeated five times, and the mean values of the l_2 errors are now tabulated in Table 5-2. The table still shows that the PIHCQNN yields the closest results to the analytical solution, with an error of only 4.9%. In contrast, both the 5-layer PINN and 6-layer PINN produce greater errors. These findings demonstrate that both the 5-layer and 6-layer PINNs do not exhibit the

equivalent superiority to the PIHCQNN in terms of loss convergence. In addition, the PIHCQNN results also exhibit a higher degree of proximity to the analytical solution compared to those obtained from PINNs.

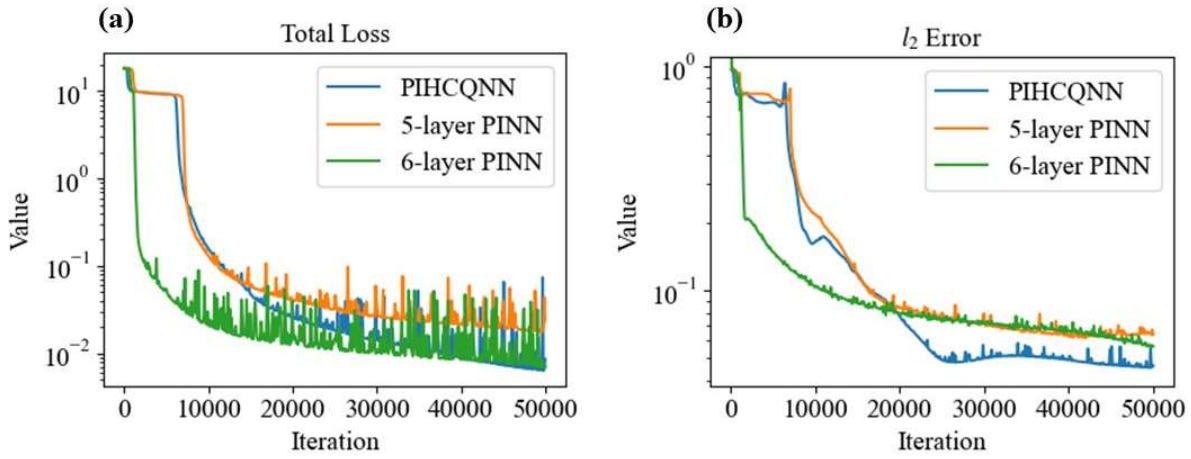


Figure 5-5. Convergence curves of **(a)** the total loss, and **(b)** the l_2 error of $u(t, x)$.

Table 5-2. Averaged l_2 errors of different model results in solving the heat equation.

	PIHCQNN	**5-layer PINN**	**6-layer PINN**
Averaged l_2 error	4.9%	7.9%	5.1%

To enhance the comprehensibility of the findings, Figure 5-6 illustrates the transient temperature distributions by PIHCQNN, 5-layer PINN, and 6-layer PINN respectively. From the figure, it is evident that all three models effectively simulate the temperature distribution over time. However, when we zoom into the errors, it becomes apparent that there are discrepancies. The absolute error of the 5-layer PINN is most significant within the entire computational domain, while the 6-layer PINN shows a better result. However, in regions where $x < 0.5$, the error of the 6-layer PINN exhibits substantial negative values. In comparison, PIHCQNN demonstrates a distinct advantage in this aspect. The error across the computational

domain is negligible, except for that when $t = 0.5$, which is also evident in other error graphs. And since this is an initial value problem, there are no constraints except for the two boundary points during the evolution process, so it is expected that the error will gradually increase over time.

The VQC structure in a PIHCQNN can be abstracted as a single quantum layer, rather than a classical layer composed of neurons. The findings suggest that the inclusion of such a single quantum layer in a neural network can greatly improve the model's capacity to express nonlinearity, and may even yield more advantages than merely inserting a single fully connected layer.

As a result, the introduction of the quantum layer has enabled neural networks to have stronger expression and feature learning capabilities. The repeatable data embedding structure in a VQC could potentially account for this. The features acquired in the preceding layer are transmitted directly to the subsequent layer in conventional feedforward neural networks. However, PIHCQNN, on the other hand, permits VQC to process the features learned in the final layer of ENN iteratively and transfer them to RoNN for analysis following complex feature processing and measurement operations, which may be the advantage of the PIHCQNN structure.

It is worth mentioning that the addition of a single quantum layer to a neural network invariably results in a significant escalation in computational expenses. In this case, the execution time for PIHCQNN was approximately 1.1×10^5 seconds for 5×10^4 iterations, while PINN (regardless of 5 or 6 layers) requires less than 1.2×10^3 seconds on the same device, which is due to the fact that the quantum circuits simulated by classical computers are very time-consuming.

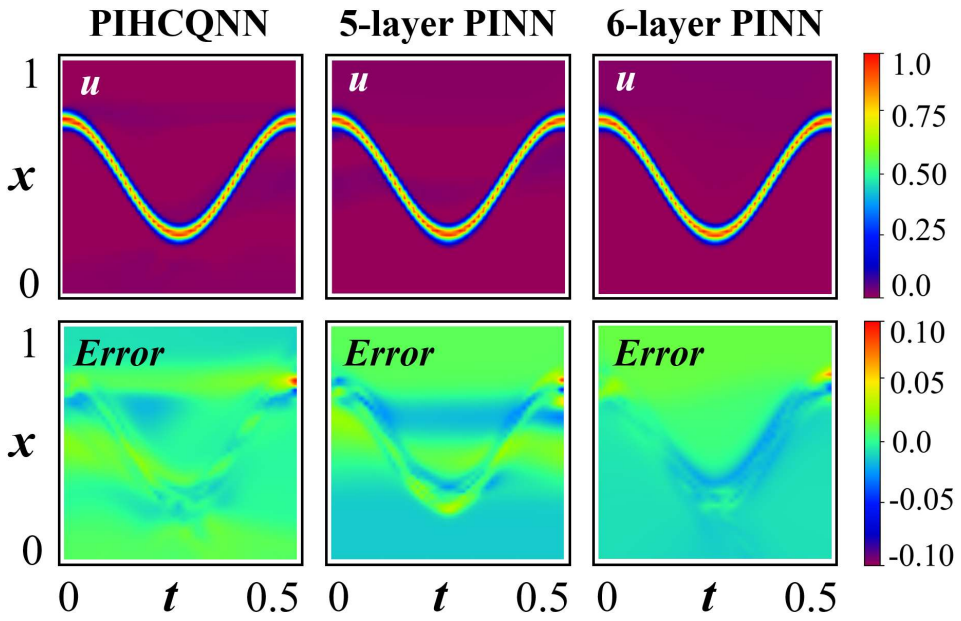


Figure 5-6. Comparison of the transient temperature distributions and absolute errors (error = prediction – ground truth) by PIHCQNN, 5-layer PINN, and 6-layer PINN.

5.3.3 Forward problem: The Poisson equation

The Poisson equation is a fundamental PDE, which is used in electrostatics to describe the electric field generated by the distribution of charges. In fluid mechanics, the Poisson equation describes the relationship between pressure and velocity in incompressible flows. This study focuses on solving a specific instance of the Poisson equation, as described below.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = x^2 + y^2 \qquad (5-15)$$

where independent variables x and y are within the range of [-1, 1] × [-1, 1]. In this case study, boundary conditions on the domain boundaries are as follows

$$u(x,-1) = u(x,1) = \frac{1}{2}x^2 \qquad (5-16)$$

$$u(-1,y) = u(1,y) = \frac{1}{2}y^2 \qquad (5-17)$$

The analytical solution of u in this case can be expressed as follows

$$u = \frac{1}{2}x^2y^2 \qquad (5-18)$$

PIHCQNN and PINN models are utilized, respectively, for solving the Poisson equation. The structures of both models remain consistent with the previous example, except that the widths of the ENN and RoNN, as well as the number of qubits in the sandwiched VQC, have been changed from 10 to 5. The number of training iterations has been changed to 5×10^3. In this case, the Poisson equation can be classified as a steady-state problem, meaning that no initial condition component is included in the loss function. The weight coefficients w_g and w_b are each assigned a value of 1. The collection points have been decreased from a lattice of 50 by 50 to a lattice of 40 by 40 in order to increase computational efficiency. Additionally, there are forty equally-spaced collection points on each domain boundary.

Figure 5-7 displays the solutions obtained from PIHCQNN and PINNs respectively, along with the corresponding absolute errors. While the three models yield similar results based on the figures in the first row, the figures in the second row clearly show that the absolute error disparity among the three is substantial. These error graphs indicate that in this case study, the 5-layer PINN still exhibits the lowest accuracy in solving the Poisson equation, while the PIHCQNN still performs the best. Similar to the previous case, the training process is repeated five times to obtain the mean l_2 errors of different models, which are summarized in Table 5-3. In comparison to the 5-layer PINN and the 6-layer PINN, the l_2 error of the PIHCQNN is reduced by seven thousandth and one thousandth, respectively. This is consistent with the results when solving the heat equation.

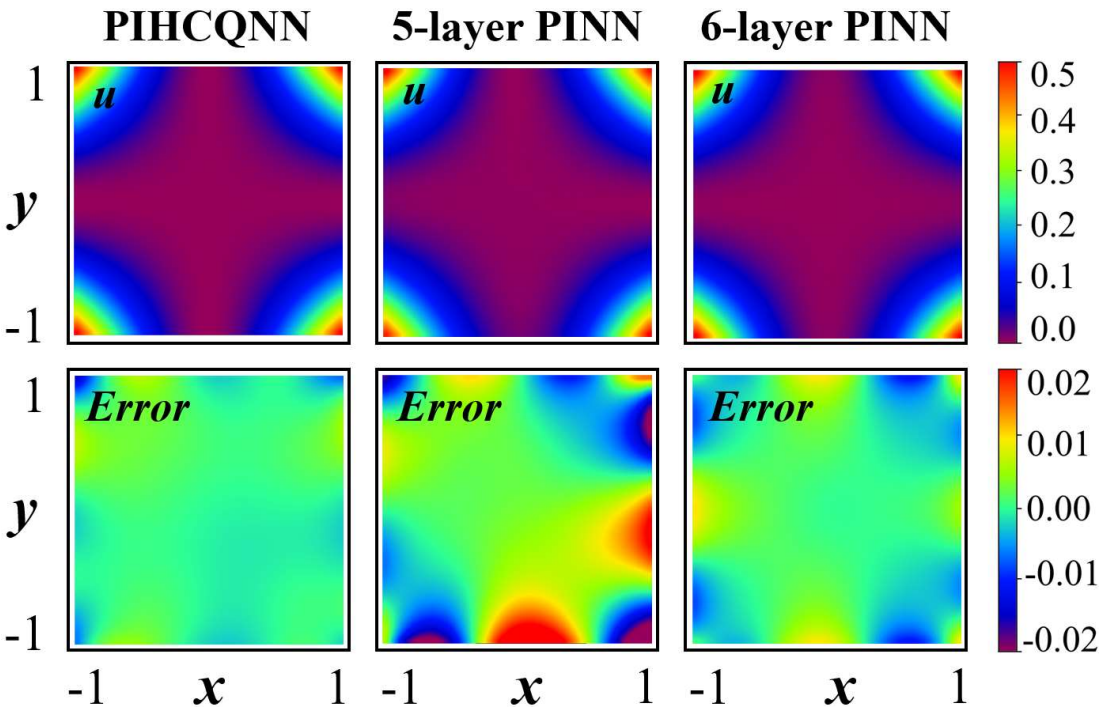


Figure 5-7. Comparison of the solutions of the Poisson equation and corresponding absolute errors (error = prediction – ground truth) obtained from PIHCQNN, 5-layer PINN, and 6-layer PINN.

Table 5-3. Averaged l_2 errors of different results in solving the Poisson equation.

	PIHCQNN	**5-layer PINN**	**6-layer PINN**
Averaged l_2 error	1.9%	2.6%	2.0%

In Figure 5-8, the results of one of the five independent experiments are depicted, which are essentially consistent with the previous situation. Following 5000 training iterations, the final total loss and l_2 error of the PIHCQNN are the lowest of all three models, while those of the 5-layer PINN are the highest. Such a result indicates that PIHCQNN has better convergence performance in solving PDEs. The expressive capability of the model is considerably enhanced by the addition of a quantum layer in comparison to a fully connected classical layer.

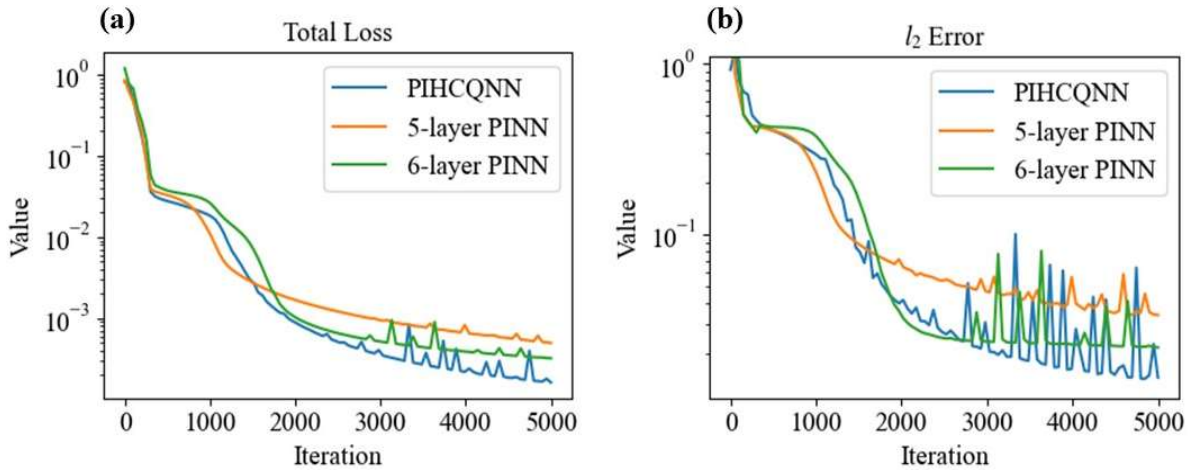


Figure 5-8. Convergence curves of **(a)** the total loss of different models, and **(b)** the l_2 error of $u(x, y)$.

5.3.4 Inverse problem: One-dimensional elastostatics equation

In this section, an inverse elastostatics problem will be solved by PIHCQNN and PINN simultaneously. The problem is governed by an ODE equation and its corresponding boundary conditions, which describe the static displacement of a one-dimensional linear elastic bar. The ODE is a second-order differential equation which is as follows

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) + p_b = 0 \qquad (5-19)$$

Spatial coordinate x is within the range of [0, 1]. In Eq. (5-19), $E(x)$ is Young's modulus and $A(x)$ is the cross-sectional area. $u(x)$ denotes the displacement. $p_b(x)$ is the body force. Here, we define p_b as

$$p_b(x) = 4\pi^2 \sin(2\pi x) \qquad (5-20)$$

Boundary conditions are also considered at the two ends of the one-dimensional linear elastic bar, which can be expressed as

$$u(0) = u(1) = 0 \qquad (5-21)$$

For forward problems, the value of $EA(x)$ is known, and $u(x)$ is the unknown of the problem. For example, we may set the value of $EA(x)$ to a constant of 1. As a result, the analytical solution of the bar equation is as follows

$$u(x) = \sin(2\pi x) \qquad (5-22)$$

This time, slightly different is that an attempt is made to solve an inverse problem. Assuming that the displacement of the linear elastic bar across the entire computational domain is known, then the value of $EA(x)$ can be solved. 50 labeled data points are equidistantly sampled in the computational domain as determined by Eq. (5-22) to guide the model training process.

In order to resolve the inverse problem, it is necessary to incorporate an additional data constraint into the physics-based loss function. During the resolution of the inverse problem, the undetermined parameter $EA(x)$ is optimized simultaneously with the neural network parameters and VQC parameters by the optimizer.

Assuming that this is a steady-state problem, the weight coefficient of the initial condition loss (i.e., w_i) should be set to zero. Additionally, the weight coefficient of the boundary condition loss (i.e., w_b) is also set to zero to prevent redundant calculations, as the first and last labeled data perfectly coincide with the boundary conditions. This results in the loss function containing only governing equation loss and data loss, and the loss coefficients of these two are set to 1 and 10, respectively.

The ENN employs a two-layer FCNN structure here. The spatial coordinate x is represented by a single neuron in the first layer, while the output layer of the ENN (i.e., the

second layer) is composed of five neurons. RoNN is also composed of two layers. The first layer is composed of five neurons that receive outputs from the VQC, while the second layer is composed of a single neuron that represents the displacement u of the bar.

The sandwiched quantum module is a five-qubit VQC, with a cycling number n of 1. The learning rate is switched to 5×10^{-3} and the number of training iterations is reduced to 2×10^{4} in this case. Other configurations are kept consistent with the previous case. This time, a PINN with two hidden layers is used for comparison. This can be understood as the VQC structure in PIHCQNN is replaced with a fully connected layer, and the hidden layer width still remains at 5.

The total loss and l_2 error of PIHCQNN and PINN during the training process when solving the inverse problem are illustrated in Figure 5-9. Figure 5-9(a) illustrates that PIHCQNN demonstrates better convergence performance in resolving this inverse problem compared to PINN. The reduction in the total loss of PIHCQNN is slightly greater than that of PINN. More specifically, the total loss of PIHCQNN is 2.69×10^{-3} after 2×10^{4} training iterations, whereas the ultimate total loss of PINN is 3.48×10^{-3}. Upon examination of the downward trend of the total loss in Fig. 5-5(a) and Fig. 5-9(a), it is possible that PIHCQNN will attain an even lower total loss if the training process is further extended.

Considering the l_2 error, as depicted in Fig. 5-9(b), the two values are also comparable and of similar magnitude. During the training phase, the $EA(x)$ is optimized alongside other parameters, progressively converging towards the analytical solution, which is 1. After 2×10^{4} training iterations, we successfully reach the ultimate value of $EA(x)$. To minimize the random error, three repetitions are conducted to calculate the average of these three trials as the final result, as presented in Table 5-4. Table 5-4 reveals that the averaged relative error of PIHCQNN

is 0.067%, while that of PINN is 0.080%, which suggests that in this case, PIHCQNN outperforms PINN in solving the inverse problem and yields superior accuracy in resolving undetermined parameters.

It should be noted that PIHCQNN still exhibits notable disadvantages in computational efficiency, as evidenced by the fact that 2×10^4 training iterations require 7×10^3 seconds for a PIHCQNN compared to only 120 seconds for a classical PINN. The reason for this is that classical computers require the storage of all potential states in order to simulate the superposition of quantum states. As the number of qubits increases, the computational expense of the simulation will become exceedingly high. In addition, classical computers also incur much overhead in simulating quantum entanglement.

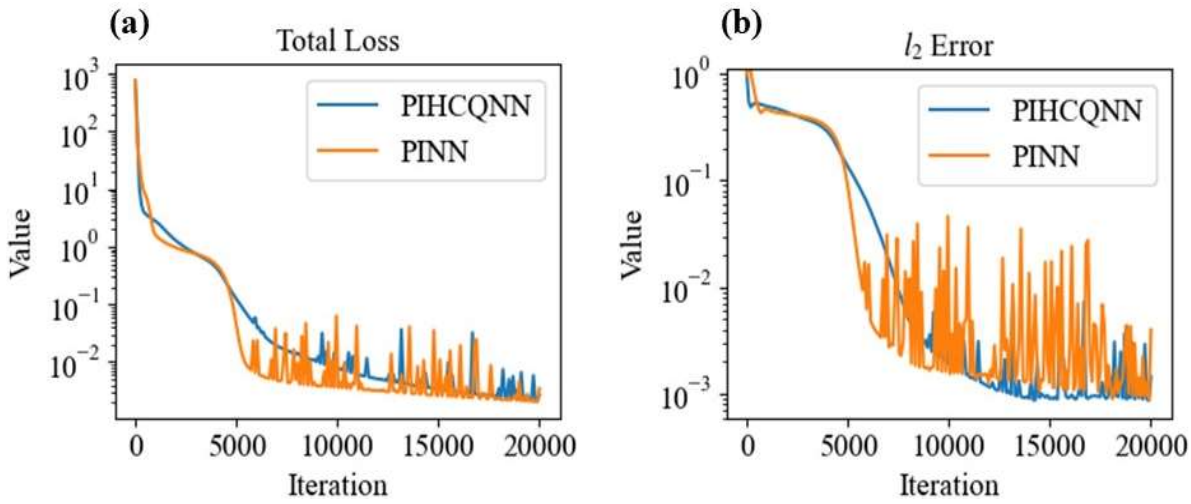


Figure 5-9. **(a)** The total loss of PIHCQNN and PINN, and **(b)** the l_2 error of PIHCQNN and PINN's predictions.

In the above case studies, the results demonstrate that PIHCQNN outperforms PINN in the solution of both forward and inverse ODE and PDE problems in terms of accuracy. The reason for this may be that PIHCQNN, as a hybrid classical-quantum machine learning model, has a more robust nonlinear function fitting capability and generalizability. This may be

attributed to the reuploading strategy in VQC or the fact that VQC can learn features, that neural networks find hard to capture, more efficiently.

Table 5-4. The ultimate value of $EA(x)$ and the averaged relative error.

	PIHCQNN			**PINN**		
	Trial 1	Trial 2	Trial 3	Trial 1	Trial 2	Trial 3
Value of $EA(x)$	0.9999	1.0013	0.9994	0.9987	1.0004	0.9993
Averaged error		0.067%			0.080%	

5.4 Further Exploration

The aforementioned cases have yielded remarkable results from the integration of a quantum layer into a PINN, which has been able to obtain more accurate results in the solution of both forward and inverse ODE and PDE problems than a classical PINN. One possible explanation is that classical PINNs are generally believed to have difficulty accurately capturing the high-frequency components in PDE solutions due to the spectral bias phenomenon of an FCNN (Ye *et al.* 2024). However, it is found that the integration of a quantum circuit in fully connected layers may potentially alleviate such a phenomenon. Here, we endeavor to demonstrate this concept by fitting a simple target function, i.e. $f(x) = \sin(x) + \sin(3x) + \sin(5x)$, using a dressed quantum circuit and classical FCNNs respectively. The spatial coordinate x is within $[-\pi, \pi]$, and 201 evenly spaced labeled points are sampled to train the models, which is the same configuration as in reference (Xu *et al.* 2019). The structures of PIHCQNN and PINN models remain consistent with those in Figures 5-1 and 5-3, except that the ENN takes a three-layer structure with 1, 20, and 10 neurons and the RoNN also takes a

three-layer structure with 10, 20, and 1 neuron. The sandwiched VQC is a 10-qubit quantum circuit with a cycling number n of 2. To facilitate comparison, one, two, and three classical fully connected layers (with a width of 10) are incorporated between the ENN and RoNN. This results in the existence of FCNNs with six, seven, and eight layers, as illustrated in Figure 5-10, which illustrates the convergence behaviors of three different frequency components during the training processes of the above models. Discrete Fourier Transform is employed to calculate the model predictions of the amplitudes of the three low-to-high frequencies in $f(x)$. Results indicate that the 5-layer PINN tends to prioritize learning low-frequency features during the training process, with the neural network progressively mastering the highest-frequency features at approximately 3000 steps. The depth of an FCNN can be increased to expedite the learning of high-frequency features. Nevertheless, the efficacy of such assistance is restricted, and there is no additional benefit beyond a certain depth of the neural network. On the contrary, PIHCQNN enters the high-frequency learning stage earlier (less than 500 iterations as shown in the leftmost subgraph in Figure 5-10), which may be one of the reasons for its success in resolving intricate ODE and PDE problems.

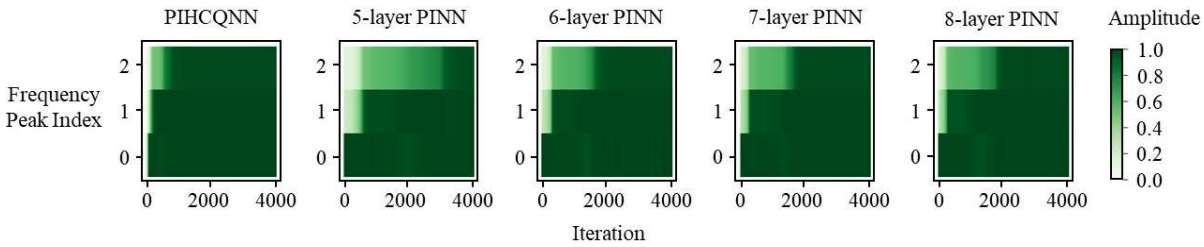


Figure 5-10. Convergence behaviors of three different frequency components.

A counterexample which is encountered during our investigation may support the above viewpoint, which occurs during our attempt to solve the forward problem of the one-dimensional elastostatics equation of Eq. (5-19). Specifically, it is given that the material

parameter $EA(x)$ is equal to 1. Therefore, the objective turns to be the determination of the bar displacements within the entire computational domain, where Eq. (5-22) becomes the analytical solution of the forward ODE problem. The models and their structure configurations remain consistent with those described in Section 3.3. The test is still repeated three times, and Figure 5-11 illustrates the absolute errors between the analytical solution and the results obtained. The results in the figure indicate that PIHCQNN is prone to more significant errors than PINN when solving the forward problem. The mean l_2 error of PIHCQNN is 2.4×10^{-3}, while that of PINN is only 1.6×10^{-3}. In contrast, the relative error of PINN's solution has decreased by one-third. The reason that this counterexample may support the aforementioned perspective is that in this instance, it is only required to solve a simple sin function that contains information with a single frequency. In this respect, PIHCQNN may not yield superior results to PINN.

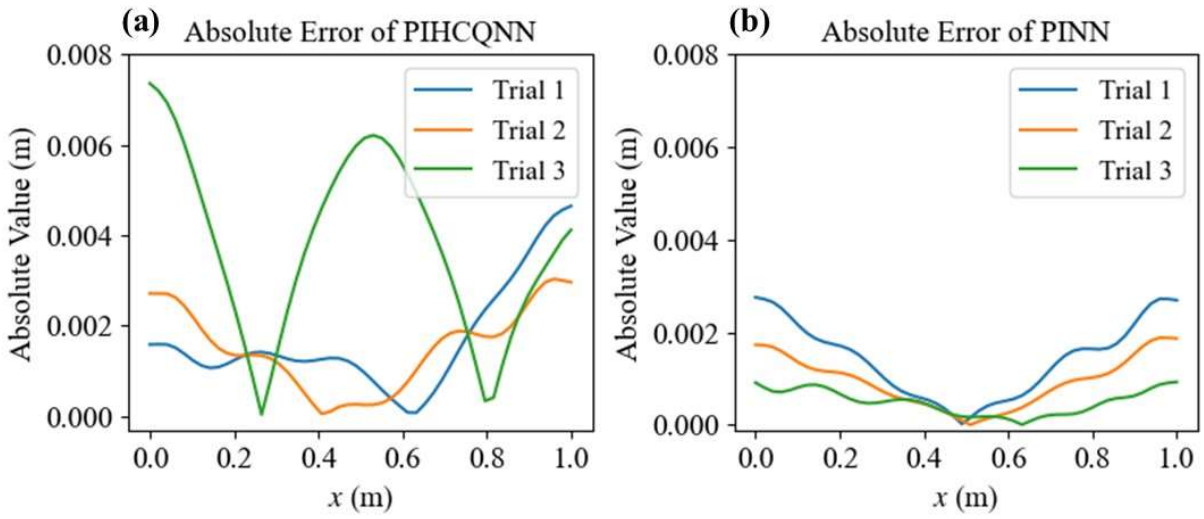


Figure 5-11. The absolute errors between the analytical solution and **(a)** PIHCQNN's results, and **(b)** PINN's results.

Also, it should be emphasized that the precision of the PIHCQNN's solution may be influenced by a variety of factors, such as the configuration of the RoNN and the ENN, the

structure of the ansatz in VQC, and the setting of the cycling number n. The impact of these factors deserves further research in the future.

5.5 Conclusions

This chapter proposes a hybrid classical-quantum model, namely the PIHCQNN, to solve the forward and inverse problems involving ODEs and PDEs. PIHCQNN's function fitting module is a dressed quantum circuit that resembles a sandwich structure with three substructures: an ENN, a VQC, and a RoNN. VQC performs a bridging function among the ENN and the RoNN, and its data reuploading strategy improves the model's nonlinear expression and feature learning capability. On the other hand, RoNN and ENN ingeniously enhance the flexibility of the data structure that VQC can receive and produce. Following the approach of the PINN, the knowledge of physical laws is incorporated into the cost function of the dressed quantum circuit, resulting in a hybrid classical-quantum model that can deal with forward and inverse problems involving differential equations. This chapter illustrates the efficacy of PIHCQNN through a case analysis of three PDE forward problems, i.e., the RANS equations, the heat equation and the Poisson equation, and an ODE inverse problem, i.e., the one-dimensional elastic statistics equation. The accuracy advantage it possesses over classical PINN is also demonstrated in these three cases. Nevertheless, the accuracy of PIHCQNN is inferior to that of a classical PINN when addressing the forward problem of the one-dimensional elastostatics equation, and potential influencing factors are discussed. The main conclusions drawn from the case analysis are as follows:

- In most cases, integrating a quantum layer, i.e., VQC, within an FCNN can greatly enhance the model's loss convergence capability, while the substitution of an

additional classical fully connected layer cannot achieve such an effect. Classical PINN has been outperformed by PIHCQNN in terms of accuracy in solving PDE forward problems and in solving undetermined parameters in inverse problems. The introduction of the VQC structure may be the reason for this, as it enhances the hybrid model's capability for nonlinear expression and feature learning in comparison to a classical FCNN;

- The computational efficiency of PIHCQNN is significantly lower than that of PINN as a result of the use of quantum circuits simulated by classical computers in this research. PIHCQNN requires approximately 60-90 times more computational time than PINN in both cases. The reason for this is that classical computers necessitate the storage of all possible quantum states in order to replicate the superposition.

- The potential of PIHCQNN to accelerate the learning of high-frequency features in ODE and PDE solutions has been discovered, which may potentially relieve the spectral bias observed in classical PINNs to a certain extent.

Nevertheless, it is imperative to recognize the constraints of our current work. Currently, our study relies on classical computers to simulate quantum circuits, which severely restricts the size of the simulated quantum system, i.e., the number of qubits. Thus, the proposed model's potential for extra expressive capabilities is significantly restricted by this limitation.

On the other hand, current quantum computers are NISQ computers with noise (Sweke *et al.* 2020, Torlai and Melko 2020). This implies that the results given by NISQ devices have a certain probability of being incorrect. Executing the algorithm proposed in this chapter while maintaining a certain level of accuracy on NISQ devices is a highly challenging problem that is worth further investigation. Furthermore, a mathematical proof is imperatively required that

PIHCQNN can accelerate the learning of high-frequency features in order to elucidate the findings in Figure 5-10, which is also our next step of work.

CHAPTER 6

WEIGHTED SUM TURBULENCE MODEL

6.1 Foreword

As a variant form of the NS equations which govern the fluid motions, the RANS equations have become a mainstream approach for addressing fluid dynamics problems in engineering practices (Salim *et al.* 2011, Qureshi and Chan 2020, Ricci and Blocken 2020). RANS equations are grounded on the assumption that an instantaneous flow variable can be decomposed into a time-averaged mean quantity and a fluctuation quantity. The time-averaged flow field, which is also the focus of real projects, can be obtained by solving the RANS equation, albeit at the expense of introducing new unknown terms (i.e., Reynolds stress components) into the physical governing equations. But mathematically, the existence of Reynolds stress terms will lead to the non-closure of RANS equations, so it is necessary to connect the Reynolds stress components with the time-averaged fluid variables through mathematical expressions or to introduce new equations to address the turbulent closure problem. Such a process of modeling the Reynolds stress to close RANS equations is also known as turbulence modeling (Durbin and Shih 2005, Duraisamy *et al.* 2019).

Turbulence models can generally be categorized into several groups, which are zero-equation models, one-equation models, two-equation models, and so forth, depending on the number of additional governing equations introduced into modeling the Reynolds stress (Hino 1995, Menter *et al.* 2003, Gao *et al.* 2017). Among them are several models with solid

reputations which are extensively used. For instance, the standard k-ε RANS turbulence model is a fundamental tool in CFD simulations, offering a two-equation framework for simulating the characteristics of turbulent flow (Launder and Spalding 1983). It achieves a balance between the capacity to capture turbulent characteristics and the computational efficiency through the use of transport equations. This model has been extensively validated in various scenarios and is particularly effective for flows with moderate pressure gradients (Chen 1995, Rumsey *et al.* 2006).

Again as an example, a zero-equation RANS turbulence model, introduced by Chen in 1998, represents a significant advancement in the simulation of indoor airflow dynamics (Chen and Xu 1998). This innovative model simplifies the computational process by assuming turbulent viscosity as a function of length scale and local mean velocity. The Chen model has demonstrated its effectiveness in predicting various types of convection within enclosed environments, showing commendable agreement with experimental data (Chen and Srebric 2000).

On the other hand, as was discussed in Chapters 1 and 2, there is no universal turbulence model applicable to all flow conditions. Give the simplest example to prove this viewpoint, which is the problem of determining the value of empirical constants in turbulence models. It is worth noting that there exist some constants in each of the above RANS turbulence models, and the values of these constants are usually determined based on experience or experiments. Despite the existence of recommended values for these constants, there is no unified standard, and strictly speaking, they should be determined on a case-by-case basis (Guillas *et al.* 2014). For example, the coefficient C_μ in the standard k-ε model, with a typical value of 0.09, affects the turbulent energy transfer process. A large value may suppress the development of turbulence, making the flow field overly smooth, while a small value may amplify turbulent

eddies, making the flow field overly turbulent and not in line with actual physical phenomena. In addition, inappropriate values can also lead to numerical instability and divergence, making the simulation unable to converge (Luo *et al.* 2020).

Under such circumstances, a novel weighted sum RANS turbulence model is proposed in this chapter. In addition, a novel PINN structure is proposed here for automatically optimizing the hyperparameters in the proposed turbulence model, which serves as the solver for the RANS equations. PINN, which was proposed by Raissi *et al.* (2019), has been placed great expectation to become a powerful alternative to traditional CFD methods to solve PDEs (Mao *et al.* 2020, Xiao *et al.* 2024, Zeng *et al.* 2024). The rest of the chapter is organized as follows. In Section 6.2, the basic principle of the proposed weighted sum RANS turbulence model as well as the PINN structure for implementation of the proposed turbulence model will be briefly introduced. In Section 6.3, a case study will be utilized to verify the feasibility of the proposed method. In Section 6.4, another flow case study will be used to illustrate its broad applicability. The main conclusions will be drawn in Section 6.5.

6.2 Methodology

6.2.1 RANS equations and turbulence models

As mentioned earlier, the NS equations are averaged in the time domain to form the RANS equations for solving a time-averaged flow field rather than an instantaneous flow field. The computational burden can be significantly lessened since the turbulent fluctuation on each scale is no longer calculated. The RANS equations for fluid flow simulations are shown as follows

$$\frac{\partial}{\partial x}(\rho\bar{u}_i) = 0 \tag{6-1}$$

$$\rho\bar{u}_j\frac{\partial\bar{u}_i}{\partial x_j} = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\frac{\partial\bar{u}_i}{\partial x_j} - \rho\overline{u_i'u_j'}\right) \tag{6-2}$$

where ρ is the fluid density, $\bar{u}_i$ is the velocity component in x_i-direction, $\bar{p}$ is the pressure, μ is the laminar viscosity, and $-\rho\overline{u_i'u_j'}$ is the Reynolds stress. Based on the Boussinesq eddy viscosity assumption, the Reynolds stress can be expressed as follows

$$-\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho\delta_{ij}k \tag{6-3}$$

where k is the turbulent kinetic energy, and $k = \frac{1}{2}\overline{u_i'u_i'}$. Turbulence models are adopted to describe the influence of the Reynolds stress term in the momentum equations. The zero-equation model, also known as the algebraic model, has the simplest form among various turbulence models. The essence of the zero-equation model is to describe turbulent viscosity using the averaged characteristics of the flow.

The weighted sum RANS turbulence model proposed in this chapter is essentially a linear combination of various candidate zero-equation models. Therefore, the concept of a candidate turbulence model is now proposed. A candidate turbulence model is the base model for the weighted sum operation and should be a zero-equation model. Some well-known zero-equation models that have the potential to be candidates include the constant eddy viscosity model, the Prandtl mixing-length model, the Chen model, and so on.

The constant eddy viscosity model adopts the most straightforward approach for turbulence modeling, which sets μ_t to be a constant (Apsley 2024). This method works effectively for some cases of wake simulation. In the constant eddy viscosity model, μ_t takes

the following form

$$\mu_t = C_s \rho V_s l_s \qquad (6-4)$$

where C_s is an empirical constant. In this chapter, its value is set to 0.01. V_s and l_s are turbulence velocity and length scales. In this chapter, V_s is equal to the mean- low velocity and l_s is equal to one-tenth of the flow width.

The Prandtl mixing-length model used by Pioch *et al.* (2023) is adopted in this chapter. In the model, the turbulent viscosity μ_t takes the following form

$$\mu_t = \rho l_m^2 \sqrt{G} \qquad (6-5)$$

where G is the mean strain rate. For example, G can be described as follows in a two-dimensional flow

$$G = 2\left(\frac{\partial \bar{u}}{\partial x}\right)^2 + 2\left(\frac{\partial \bar{v}}{\partial y}\right)^2 + \left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right)^2 \qquad (6-6)$$

In Eq. (6-5), l_m is the mixing length which varies in different flow cases. In this chapter, the mixing length is calculated as follows

$$l_m = \min\left(0.419l, 0.09l_{max}\right)$$

where l is the nearest distance from the wall, and l_{max} is its maximal value.

The Chen model is a zero-equation turbulence model that is widely adopted to simulate indoor airflow behaviors. In the Chen model, the turbulent viscosity μ_t takes the form of a simple algebraic function:

$$\mu_t = C_c \rho V l \qquad (6-7)$$

where V is the mean fluid velocity, and. C_c is a constant which is equal to 0.03874.

6.2.2 Weighted sum RANS turbulence model

A weighted sum zero-equation RANS turbulence model in PINN-based RANS simulation is proposed in this study. The turbulent viscosity μ_t in the proposed weighted sum RANS turbulence model can be described as follows

$$\mu_t = \sum_{i=1}^{n_c} r_i \mu_{ti} \tag{6-8}$$

where μ_{ti} is the turbulent viscosity calculated using the *i*th candidate turbulence model. n_c is the number of the candidate turbulence models. r_i is the weight coefficient, whose value needs to be optimized. $r_i \in [0, 1]$ should also meet the following constraints

$$\sum_{i=1}^{n_c} r_i = 1 \tag{6-9}$$

For example, if there are two candidate models, i.e., the constant eddy viscosity model and Chen model, the weighted sum turbulent viscosity μ_t is expressed as follows

$$\mu_t = r_1 \mu_{t1} + r_2 \mu_{t2} = r_1 C_s \rho V_s l_s + r_2 C_c \rho V l \tag{6-10}$$

Therefore, now the problem turns to be how to find the optimal value for the weight coefficient r_i. The answer lies in the optimization of the weighting coefficients r_i by minimizing the difference between the measured Reynolds stresses and the ones derived from the PINN's predictions. To be precise, PINN directly outputs flow velocities and fluid pressure, while PINN's predictions of the Reynolds stresses are further calculated based on the Boussinesq eddy viscosity assumption, i.e. Eq. (6-3).

6.2.3 PINN structure for implementation of the weighted sum model

The PINN structure proposed in this chapter for implementation of the weighted sum RANS turbulence model in two-dimensional flows is shown in Figure 6-1. First, it demonstrates similarities to a default PINN. Its FCNN mainbody is exactly the same as that in a default PINN. In other words, an FCNN has been used as the module for function expression, which describes the relationship between spatial coordinates (x, y) and fluid characteristics (u, v, and p). AD is also implemented here to calculate the gradients, forming the partial derivative terms in the loss functions (Baydin *et al.* 2018). The section right after AD shows significant differences from a default PINN, as illustrated in Figure 6-1. Next, the proposed PINN structure will be described step by step according to the flow sequence of the algorithm. After AD, Eq. (6-8) will be used to calculate the turbulent viscosity μ_t based on the proposed weighted sum RANS turbulence model. The weight coefficients r_i will be included here as additional trainable parameters in the calculation of the turbulent viscosity (it will be assigned an initial value at the beginning of training). Then, the Reynolds stress loss is further calculated using the turbulent viscosity obtained in the previous step, the neural network outputs, and the Boussinesq eddy viscosity assumption. The Reynolds stress loss L_R can be defined as follows for a two-dimensional flow case

$$L_R = \frac{1}{N_R}\sum_{i=1}^{N_R}\left(\left|r_{Rx}^i\right|^2 + \left|r_{Rxy}^i\right|^2 + \left|r_{Ryy}^i\right|^2\right) \qquad (6-11)$$

where N_R is the number of training points on which the Reynolds stress can be obtained. r_{Rx}^i, r_{Rxy}^i, and r_{Ryy}^i are residuals of different Reynolds stress terms on the ith point, which take the

following form

$$r_{Rxx} = 2\mu_t \frac{\partial \bar{u}}{\partial x} - \frac{2}{3}\rho k + \rho\overline{u'u'} \qquad (6-12)$$

$$r_{Rxy} = \mu_t \left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y}\right) + \rho\overline{u'v'} \qquad (6-13)$$

$$r_{Ryy} = 2\mu_t \frac{\partial \bar{v}}{\partial y} - \frac{2}{3}\rho k + \rho\overline{v'v'} \qquad (6-14)$$

where $\overline{u'u'}$, $\overline{u'v'}$, and $\overline{v'v'}$ are measured Reynolds stresses. It is worth noting that the existence of the terms r_{Rxx}^i, r_{Rxy}^i, and r_{Ry}^i in Eq. (6-11) are not mandatory, and some of them may be absent depending on whether there is the corresponding measurement data of the Reynolds stresses at these measurement points. Based on the Eqs. (6-11), (6-12), (6-13) and (6-14), it can be observed that the Reynolds stress loss L_R is a function of the turbulent viscosity μ_t. Also, because in the proposed weighted sum RANS turbulence model, the turbulent viscosity μ_t is a function of the weight coefficient r_i, consequently, we may come to the conclusion that the Reynolds stress loss L_R is a function of the weight coefficient r_i.

An important assumption in this research is now proposed to simplify the problem. That is, except in the early stages of PINN-based RANS simulation, changes in the value of weight coefficient r_i will not have a direct significant impact on the simulation results of the fluid velocity and pressure within the computational domain. This is easy to understand because, despite the variations caused by using different turbulence models, the overall simulation results are quite analogous from a holistic perspective.

Based on the above assumption, the optimization of the weight coefficients r_i may be achieved by minimizing the Reynolds stress loss since L_R is a function of velocity gradients,

r_i, and a series of constants. When the change in velocity gradients can be ignored, the change in r_i value becomes the dominant factor in minimizing the Reynolds stress loss. In the chapter, the Adam optimizer is adopted to find the optimal values for r_i. In order to distinguish this training process from the later one for optimizing FCNN parameters (note that only r_i is trained here), the optimizer here is referred to as optimizer 1 and the training iteration here is referred to as iteration 1. In order to minimize the computational costs, an early stopping strategy should be adopted here. When optimizer 1 is used to optimize the weight coefficients r_i, the trend of parameter changes will be considered. If the range of parameter changes is less than a threshold within a certain tolerance, the optimization process will be terminated in advance. After the optimal value for r_i is obtained, it is then input into the weighted sum model to calculate the updated turbulent viscosity μ_t. The entire process above is marked as Step 1 in Figure 6-1. In summary, the purpose of Step 1 is to find the optimal value for r_i, so that the predicted Reynolds stress is closest to the measured value.

It should be noted that, although it is mentioned before that the weight coefficients r_i will be included as additional trainable parameters of the neural network, it is not directly equivalent to these trainable parameters. This is because trainable parameters have no upper or lower limits, but r_i is different. In this chapter, the trainable parameters are standardized using the hyperbolic tangent function, and only the standardized results can be considered as the weight coefficients r_i. Next, Step 2 involves training a traditional PINN using the turbulent viscosity μ_t obtained from the weighted sum RANS turbulence model updated in Step 1. A traditional PINN is trained using the physics-based loss function, which will be minimized to find the optimal values of the neural network parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ by using optimizer 2. The training iteration here will be referred to as iteration 2 from this time on in this chapter. After the neural network parameters have been trained and updated, Step 1 will be repeated in the next round.

However, it is worth noting the stability issue during the training process of the FCNN, as the value of turbulent viscosity μ_t is constantly changing during the training process. As previously assumed, except in the early stages of PINN-based RANS simulation, changes in the values of r_i will not significantly influence the simulation result. This implicitly affirms that variations in the value of r_i will not substantially influence the values of the neural network parameters, and hence, will not alter the stability of the neural network's training process.

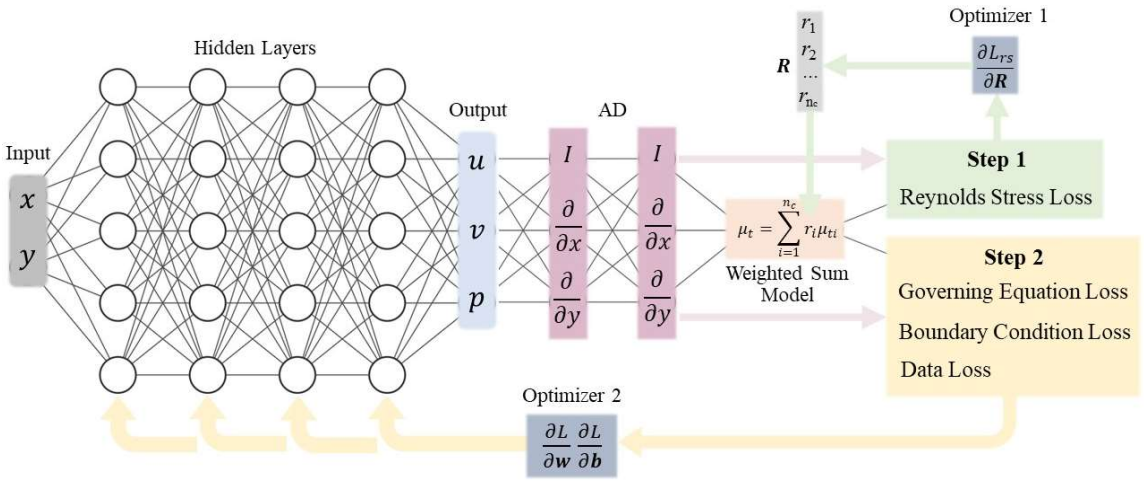


Figure 6-1. The proposed PINN structure for implementation of the weighted sum RANS turbulence model in two-dimensional flows.

In summary, in the proposed model, the measured Reynolds stress will be utilized to optimize the weight coefficients r_i in the weighted sum model, and the optimized turbulent viscosity will be used to solve the RANS equations and simulate the time-averaged flow field using the physics-based loss function. Step 1 and Step 2 are alternated to achieve a dynamic equilibrium relationship in the training process. Algorithm 6-1 also demonstrates how to use the proposed PINN structure for implementation of the weighted sum RANS turbulence model.

Algorithm 6-1 Proposed PINN for implementation of the weighted sum turbulence model

Require: Training dataset, numbers of iterations 1 and 2, optimizers 1 and 2, initial values of weighting coefficients r_i.

Target: Find the optimal values of neural network parameters and weight coefficients r_i.

Step 1: Construct an FCNN with the specified hyperparameters and initial parameters.

Step 2: Specify the collocation points in the computational domain.

for each iteration 2:

1) Optimize the values of the weighting coefficients r_i as follows:

for each iteration 1:

a) Calculate the turbulent viscosity using AD and weighted sum turbulence model.

b) Calculate the Reynolds stress loss based on Eq. (6-11).

c) Update the values of r_i using optimizer 1.

d) Distinguish whether the criterion for the early stopping strategy has been met.

end for

2) Calculate the physics-based loss function based on the optimized r_i.

3) Optimize the values of the neural network parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ using optimizer 2.

end for

6.3 Case Study: Square Cylinder Flow

6.3.1 Computational domain and boundary conditions

The flow case which was described in Section 5.3.1 is used here again for the verification of the proposed weighted sum RANS turbulence model and the proposed PINN structure for its implementation. The computational domain and boundary conditions are illustrated in Figure 6-2 for clarity. The detailed information can be found in the figure, and please note that the coordinates of point A are (0.18, 0.26). The inflow velocity is 0.535 m/s and the Reynolds number reaches 21400 when water serves as the flowing medium.

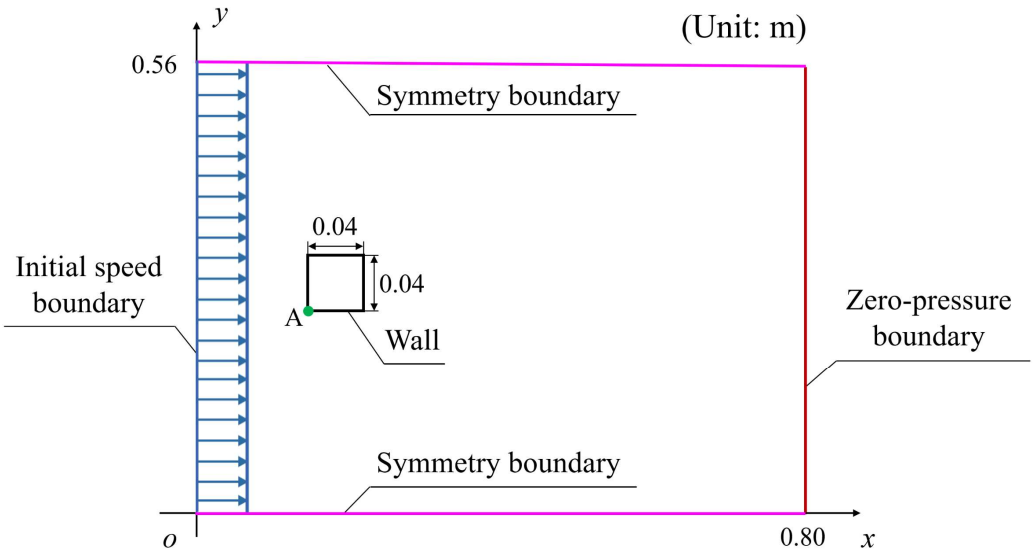


Figure 6-2. Computational domain of the flow passing a 2D square cylinder.

6.3.2 Validation data

Analogous to the preceding chapter, the experimental data obtained in a closed water channel test will be used as the data basis for this case study (Lyn and Rodi 1994). 517 measurement points were distributed throughout the entire computational domain to acquire

the fluid characteristics (the fluid velocities and Reynolds stresses) in the test. The measured data on 36 out of the total measurement points are chosen to train the PINN model. The spatial coordinates of these training points are tabulated in Table 6-1. The principle of distributing them across the entire computational domain as equitably as possible is adhered to during the selection of the training points.

Table 6-1. Spatial coordinates of the training points.

***x* (mm)**	***y* (mm)**						
	280	300	320	340	360	400	440
80			✓		✓	✓	✓
180			✓		✓	✓	✓
235	✓	✓	✓	✓			
280	✓		✓		✓		
330	✓		✓		✓		
380	✓		✓		✓	✓	✓
430	✓		✓		✓		
472	✓		✓		✓	✓	✓
520	✓		✓		✓	✓	✓

Footnote: ✓ indicates the existence of a training point at the specified location.

6.3.3 Training using two zero-equation models

In this section, two default PINNs are adopted when the Chen model and Prandtl mixing-length model are used respectively for the closure of RANS equations. An FCNN with four

hidden layers serves as the main body of the PINN. In each hidden layer, there are twenty neurons. The hyperbolic tangent function is adopted as the activation function here. Adam optimizer with a learning rate of 2×10^{-4} is used to minimize the loss. The number of training iterations is set to 1×10^{5}. The values of the weight coefficients of the governing equation loss, boundary condition loss, and data loss are set to 1, 1, and 100, respectively.

Figure 6-3 depicts the total losses and l_2 errors of u and v of the two default PINNs during the training process. The following conclusions can be drawn from the figure. First, upon the completion of 1×10^{5} iterations, both models achieve the goal of convergence. In fact, after 6×10^{4} iterations, the results tend to stabilize. The total loss of the Chen model reaches 1.144×10^{-1} after the training is completed, compared to 1.961×10^{-1} of the Prandtl mixing-length model. Despite the Chen model exhibiting a reduced loss compared to the Prandtl model, its simulation results were far less accurate in terms of the relative error when compared to experimental measurements than those of the Prandtl model. The l_2 errors of the flow velocity components u and v are 8.58×10^{-2} and 3.96×10^{-1} when the Prandtl mixing-length model is adopted for RANS simulation. In contrast, the errors of the flow velocity components u and v from the Chen model are 9.21×10^{-2} and 5.03×10^{-1} respectively.

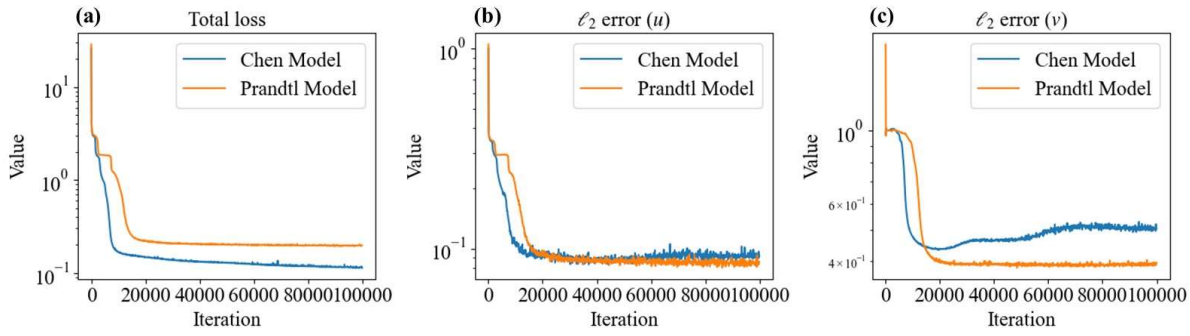


Figure 6-3. **(a)** Total losses, **(b)** l_2 errors of u, and **(c)** l_2 errors of v of two default PINNs.

The above results further affirm that the applicability issue of the RANS turbulence model

persists within the PINN framework, despite the incorporation of measurement data to inform the solution of the RANS equations within the domain. In terms of relative error, the Prandtl mixing-length model is found to be more applicable to this two-dimensional square cylinder flow. In the subsequent section, following the inclusion of the proposed weighted sum RANS turbulence model for comparative research, a more detailed analysis of this matter will be presented.

6.3.4 Training using the weighted sum RANS turbulence model

Two zero-equation RANS turbulence models are included here as the candidate turbulence models, which are the Chen model and the Prandtl mixing-length model. Thus, the turbulent viscosity μ_t in this case can be described as follows

$$\mu_t = r_1 C_c \rho V l + r_2 \rho l_m^2 \sqrt{G} \qquad (6-15)$$

The initial values for the weight coefficients r_1 and r_2 are both set to 0.5. Adam is used as optimizer 1 with a learning rate of 2.5×10^{-4}. The maximum number of iterations for optimizer 1 is set to 50, accompanied by an early stopping judgement. When using the proposed weighted sum RANS turbulence model, the structure of the FCNN, the settings for optimizer 2, and the weight coefficients in the loss function remain consistent with those described in Section 6.3.3.

The convergence curve of the proposed model is depicted in Figure 6-4(a), compared with those from the default PINNs using the Chen and Prandtl models for the closure of the RANS equations. Figure 6-4(b) shows the convergence process of the Reynolds stress loss. In addition, the variations of the r_i values during the training process are depicted in Figure 6-4(c). The figure indicates that the training of the weighted sum model ultimately converges, with both

the loss values and weight coefficient values achieving stability. The values of r_1 and r_2 ultimately stabilize at approximately 0.393 and 0.607, respectively. This indicates that the turbulent viscosity derived from the weighted sum model based on this ratio exhibits tinimal relative error compared to the actual measured Reynolds stress components at the training points. This viewpoint is also demonstrated in Figure 6-4(b). The Reynolds stress loss decreases from an initial value of 8.6×10^{-4} to 6.4×10^{-4} eventually, while the changes in r_i values definitely contribute to this reduction.

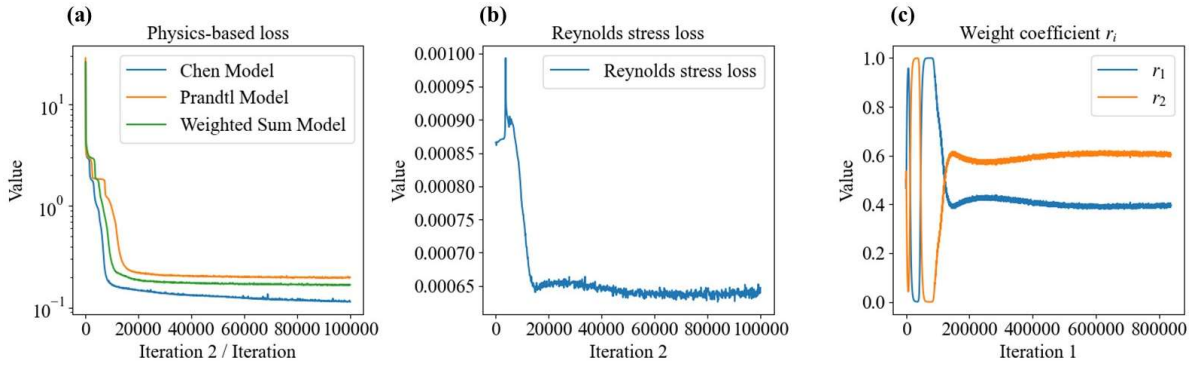


Figure 6-4. The convergence curves of **(a)** the physics-based loss, **(b)** the Reynolds stress loss, and **(c)** the weight coefficients r_i.

Figure 6-5 shows the l_2 errors of u and v from the weighted sum RANS turbulence model, compared to those from the Chen and Prandtl models. Firstly, it can be observed that although an additional optimization is introduced during the training process, there is no visible difference in convergence speed compared to traditional methods.

Secondly, the proposed weighted sum model exhibits a minimal discrepancy between the predicted mainstream velocity u and the experimental observations. The l_2 error of the flow velocity component u from the weighted sum turbulence model reaches 8.15×10^{-2} eventually, which has decreased by roughly 5% and 12% compared to the Chen model and Prandtl model,

respectively. The l_2 error of v reaches 4.07×10^{-1} at the end of the training, which is a slight improvement compared to the Prandtl model but a significant decrease compared to the Chen model. In this flow case, the l_2 errors indicate that the performance of the proposed weighted sum model is more like to that of the Prandtl model. Such a result is in line with expectations as the r_2 value ultimately stabilizes at 0.607. That is, in this flow case, the weighted sum model is dominated by the Prandtl mixing-length model, while the Chen model acts as a supporting role. However, this is not definitive. It is believed that in other flow cases, the dominant model may alter based on the agreements between the candidate models and the actual flow conditions.

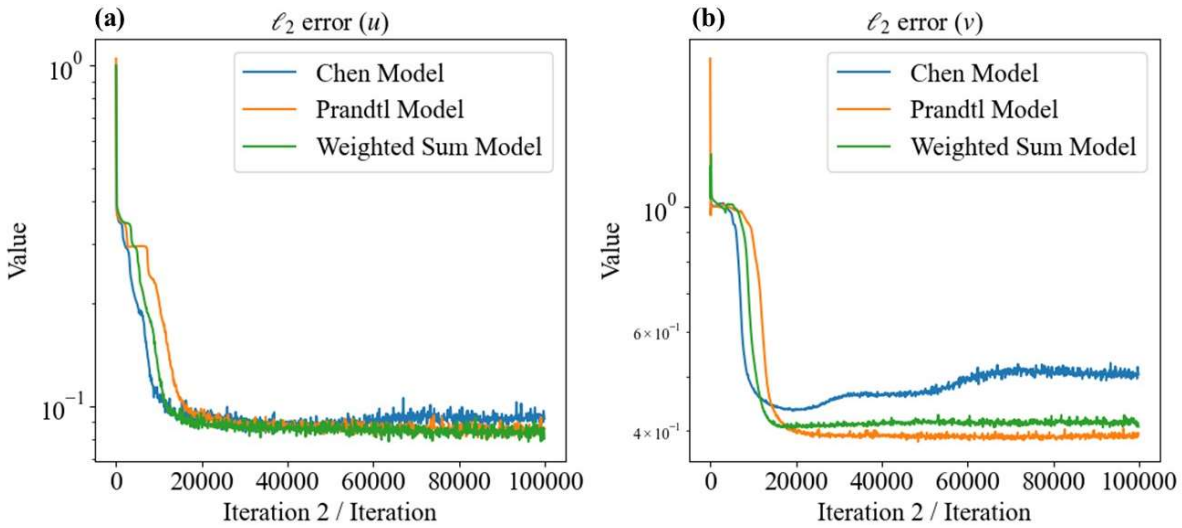


Figure 6-5. **(a)** l_2 errors of u and **(b)** l_2 errors of v of the weighted sum model.

A detailed comparison of the mainstream velocity u on the line x = 0.21 m is shown in Figure 6-6. From the left figure, it can be seen that this line is located at the flank region of the square cylinder, which is the area characterized by a sharp variation in the gradients of the flow velocities. From the subgraph on the right, it can be observed that among the three models, the proposed model is clearly closer to the measured values. Figures 6-7 and 6-8 show the comparison results of u in the wake region. The results show that the proposed model is closest

to the experimental results, whether in the flow separation region or the external mainstream region. More specifically, the proposed model is the only one that simulates a flow velocity less than zero (i.e. reverse flow) when x equals 0.25 and y is less than 0.29, which is consistent with the experimental results. Figure 6-9 depicts the comparison results of u at a downstream position. The results in the figure indicate that there is not much difference between the simulation results of each model at the downstream position away from the blunt body.

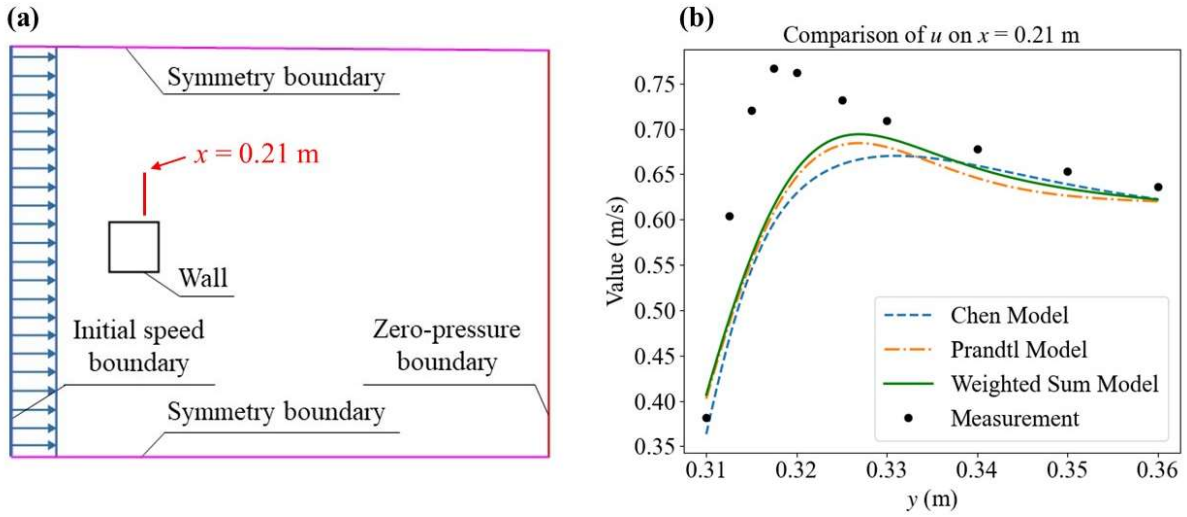


Figure 6-6. **(a)** Position of the line x = 0.21 m, and **(b)** Comparison of u on this line.

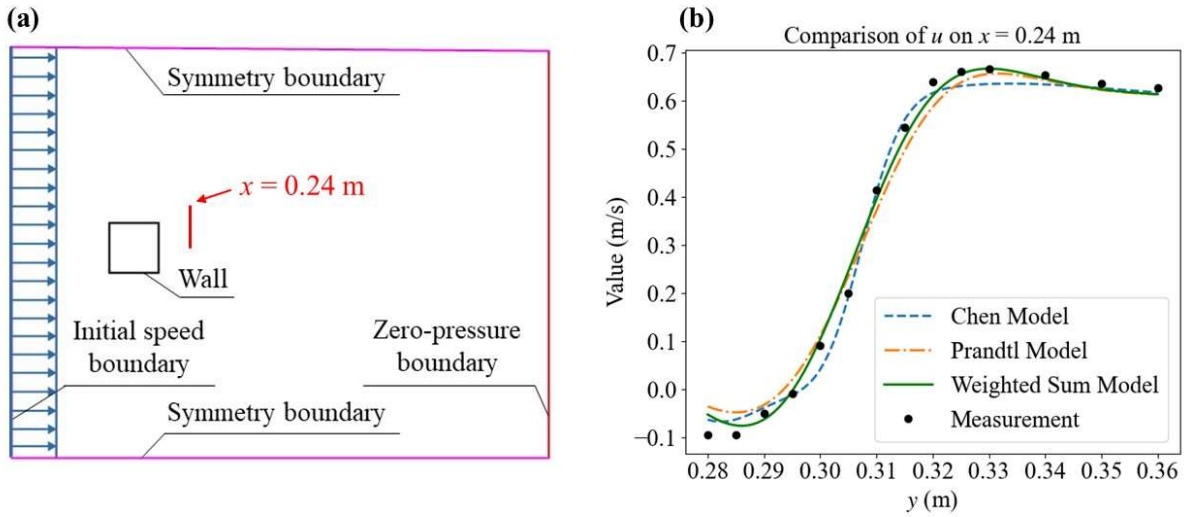


Figure 6-7. **(a)** Position of the line x = 0.24 m, and **(b)** Comparison of u on this line.

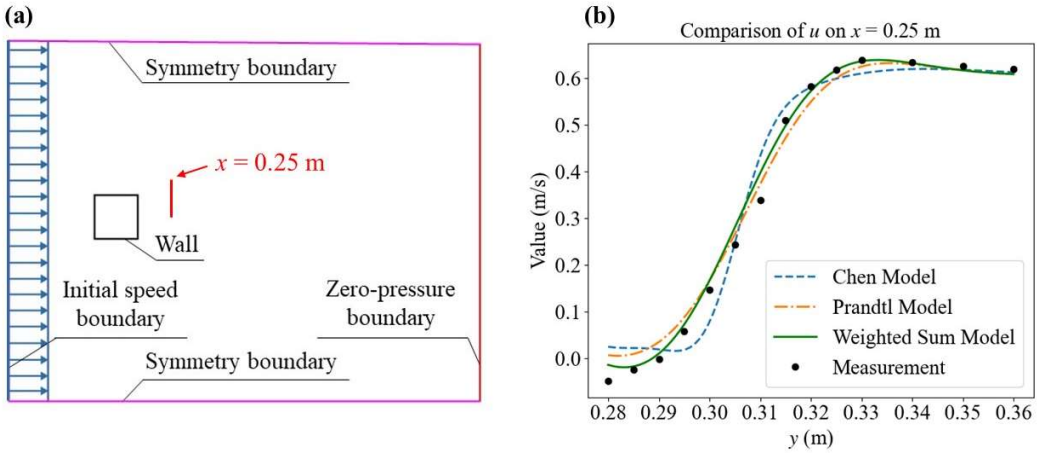


Figure 6-8. **(a)** Position of the line x = 0.25 m, and **(b)** Comparison of u on this line.

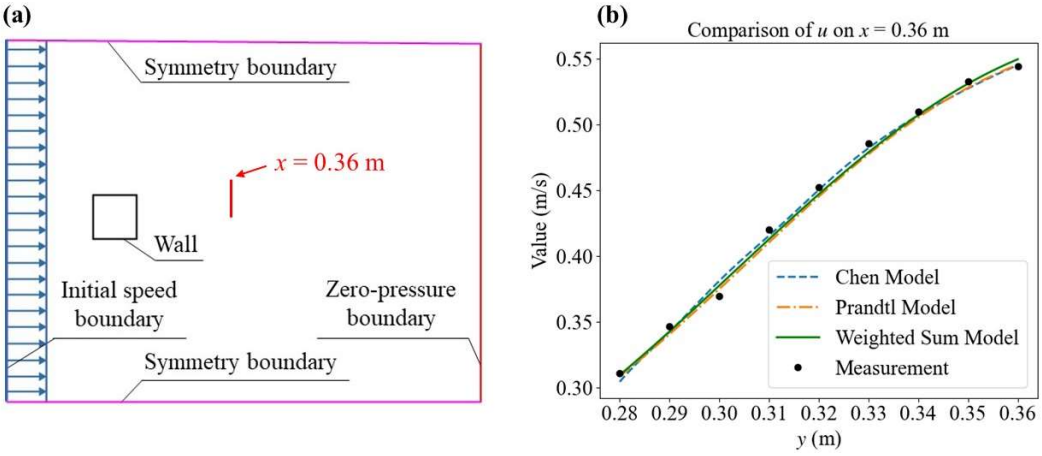


Figure 6-9. **(a)** Position of the line x = 0.36 m, and **(b)** Comparison of u on this line.

From the above figures, it can also be observed that in this flow case, the results of the proposed weighted sum RANS turbulence model align more closely with the Prandtl model, which confirms the previous conclusion and deduction. In summary, during the training process of the proposed model, the optimal values for weight coefficients r_i are obtained by minimizing the Reynolds stress loss, resulting in the expression for the turbulent viscosity μ_t. The results of the proposed model have demonstrated a smaller relative error with the experimental measurement results compared to candidate models. The proposed method for turbulence modeling greatly improves the applicability of the RANS turbulence model, as the

proposed model can automatically adjust the weight coefficient r_i based on the measured Reynolds stress components, and then automatically identify the most suitable one in the candidate models to become the 'leader', or form a completely new one by the means of linear superposition of the candidate models.

6.3.5 Training using the pretrained models

In this section, attempts are made to replace the randomly initialized parameters of the neural network with those from two pretrained models, which are described previously in Section 6.3.3. That is, assuming the existence of two well-trained PINN models based on the conventional zero-equation models, the current attempt is to use the proposed weighted sum model to further modify the simulation results. This significantly reduces the computational expense relative to training a model from the ground up. The settings for model training remain consistent with the previous sections, except that the number of training iterations 2 has been reduced to 8×10^4. In addition, the learning rate of the optimizer 2 has been reduced to 1×10^{-4}. The initial values for the weight coefficients r_i have to be changed based on the specified zero-equation model for pretraining. More specifically, if the Chen model is used for pretraining, r_1 will be assigned a value infinitely close to 1. If not, it will be assigned a value infinitely close to 0 (At the code level, weight is a trainable parameter that has been normalized by a hyperbolic tangent function, so the weight can only approach 0 or 1 infinitely).

Figure 6-10 shows the adjustment process when the proposed weighted sum model is adopted to modify the simulation results obtained from the pretrained models. It should be noted that the results of the initial 1×10^5 iterations are consistent with the corresponding results in Figures 6-4 and 6-5, while the subsequent 8×10^4 iterations are the adjustment process of using the proposed weighted sum model to modify the simulation results of the pretrained

model.

Compared to the physics-based losses before the adjustment, the loss of the Chen model has decreased by 0.6% to 1.137×10^{-1} while the loss of the Prandtl mixing-length model has dropped by 13.3% to 1.701×10^{-1}. The l_2 error of u has reduced by 4.2% to 8.22×10^{-2} when the Prandtl mixing-length model is used as the pretrained model and has reduced by 3.8% to 8.86×10^{-2} when the Chen model is adopted.

During the adjustment process, the l_2 error of v in the Prandtl model slightly increases. In comparison, the l_2 error of v in the Chen model has reduced by 19.7% to 4.04×10^{-1}. For comparison purposes, the l_2 errors of u and v obtained from all the models mentioned in this chapter are now summarized in Table 6-2. The results shown in the table and figure consistently demonstrate that utilizing the proposed RANS turbulence model substantially reduces the relative error between the simulated flow velocities and the measured values. This can be achieved by integrating the weighted sum RANS turbulence model into the initial training stage of a PINN or by rectifying the existing simulation results of conventional models using the proposed model. Such a result presents definitive evidence of the viability and efficacy of the proposed model.

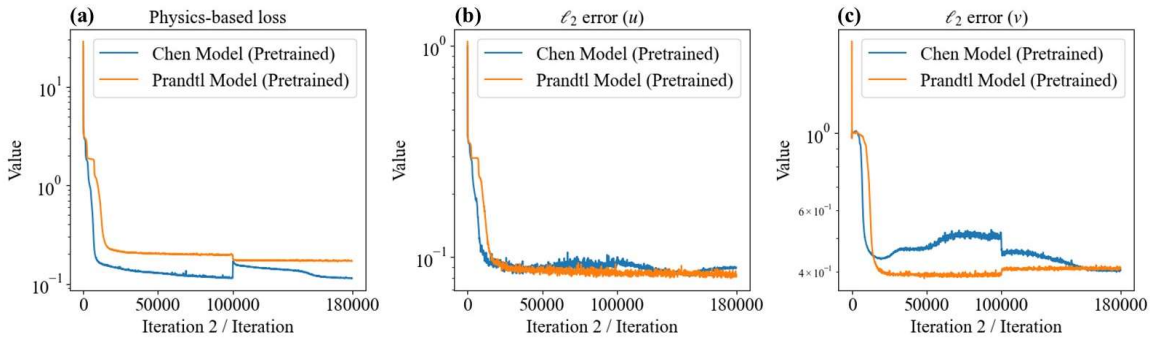


Figure 6-10. The convergence curves of **(a)** the physics-based loss, **(b)** the l_2 errors of u, and **(c)** the l_2 errors of v during the adjustment process.

Table 6-2. The l_2 errors of u and v obtained from different models.

Strategy / Model	Default PINN		Proposed Model		Pretraining	
	u	v	u	v	u	v
Chen	9.2%	50.3%	8.2%	40.7%	8.9%	40.4%
Prandtl	8.6%	39.6%			8.2%	40.8%

Figure 6-10 shows the convergence processes of the Reynolds stress losses and the weight coefficient r_1 during the adjustment process (r_2 is not drawn here because it can be derived from the value of r_1.). The results in Section 6.3.4 are also included here for comparison. The author initially holds the belief that, regardless of the turbulence models employed during pretraining, the final revised results should align with those presented in Section 6.3.4.

Nonetheless, the results depicted in the figure unequivocally demonstrate that the mixing ratio in the final weighted sum model does not align with expectations when the Chen model was employed for pretraining. That is, 72% of the turbulent viscosity in the weighted sum model comes from the Chen model, while the remaining 28% comes from the Prandtl mixing-length model. In contrast, the ratio of these two in other models is about 0.4:0.6. This difference in proportion also leads to the different values of Reynolds stress losses. Such a phenomenon may result from the pretraining model having already produced a preliminary estimation of the flow field, including the velocity gradient, which influenced the calculation of the Reynolds stress terms using the Boussinesq eddy viscosity assumption. Nonetheless, the accuracy of the revised model has been enhanced. In addition, the Reynolds stress losses have decreased relative to the initial values, underscoring the significance of the proposed model.

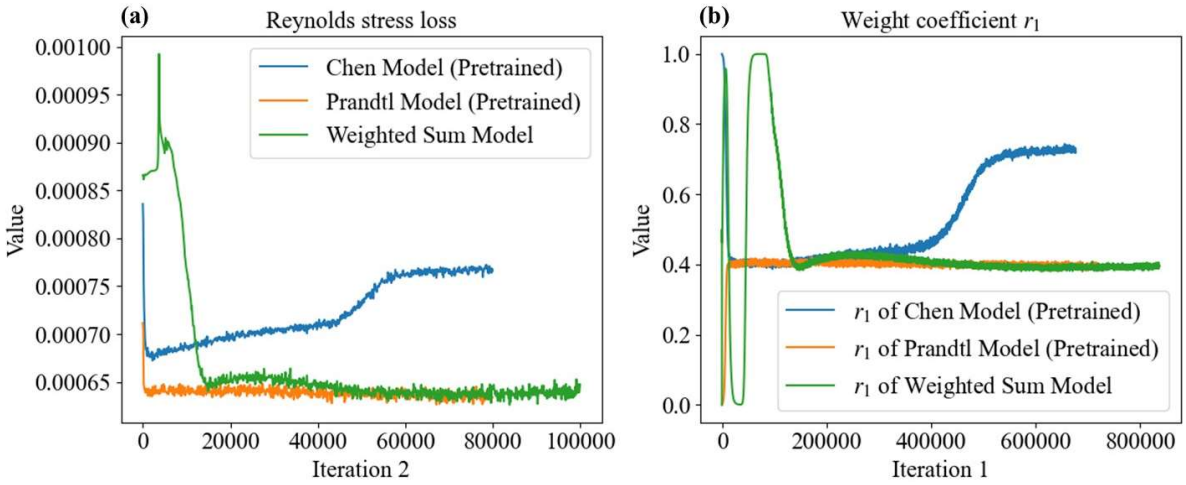


Figure 6-11. The convergence curves of **(a)** the Reynolds stress loss, and **(b)** r_1.

6.3.6 Training with three candidate models

On the basis of the original two turbulence models, a third zero-equation RANS turbulence model has been integrated in the weighted sum model as an additional candidate model, which is the aforementioned constant eddy viscosity model. Now, the turbulent viscosity in the weighted sum model is transformed into the following form

$$\mu_t = r_1 C_c \rho V l + r_2 \rho l_m^2 \sqrt{G} + r_3 C_s \rho V_s l_s \qquad (6-16)$$

All setups for the RANS simulation remain consistent with those in Section 6.3.3, except that the initial weights are set to 0.5, 0.25, and 0.25. Furthermore, in the first 2×10^4 iterations of minimizing the physics-based loss, the optimization of the Reynolds stress loss is omitted. The weight coefficients r_i are frozen at the initial values of 0.5, 0.25, and 0.25 without any changes during these 2×10^4 iterations. This is predicated on the belief that a rough solution is adequate enough in the primary training phase, rendering the optimization of r_i values unnecessary and so conserving computational costs. In addition, from another aspect, it can be seen that the implementation of the weighted sum RANS turbulence model proposed in this

chapter is very flexible and can be enabled and terminated at any stage of training. The l_2 errors of u and v ultimately decrease to 8.9% and 41.7%, which are almost identical to the results listed in Table 6-2.

Figure 6-12 depicts the convergence curves of the Reynolds stress loss, and r_i. It is obvious that the turbulent viscosity in the weighted sum model is still dominated by the Prandtl mixing-length model, accounting for about 54%, with the constant eddy viscosity model playing an auxiliary role, accounting for about 46%, while the Chen model has a very small proportion and can be ignored. Another finding is that when the physics-based loss is optimized for 2×10^4 iterations and the weighted sum model is activated, there is a significant decrease in the Reynolds stress loss. This indicates that the proposed method is effective in minimizing the discrepancies between predicted Reynolds stress terms and measured values.

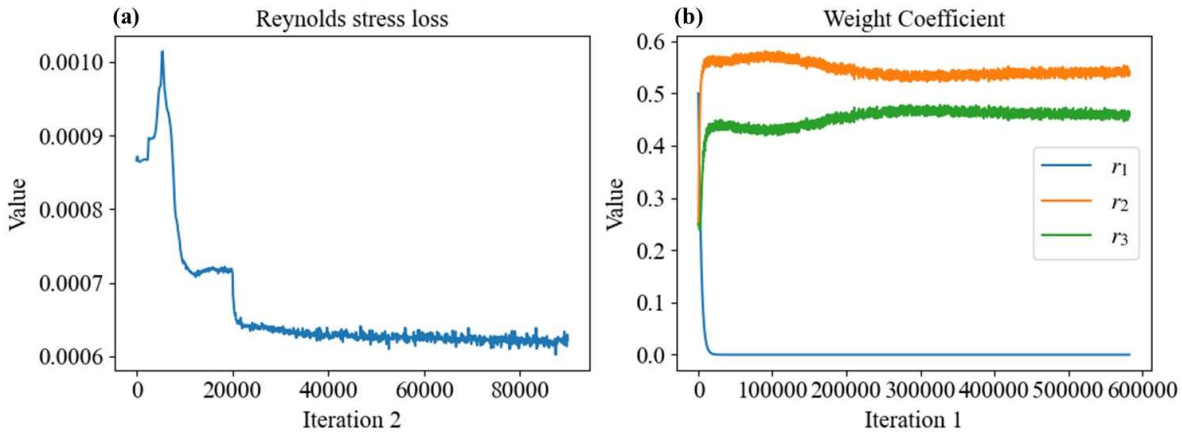


Figure 6-12. The convergence curves of **(a)** the Reynolds stress loss, and **(b)** r_i when there are three candidate models.

Figure 6-13 compares the predicted and measured Reynolds shear stress on the line x = 0.28 m. The results indicate that the Reynolds shear stress predicted by the proposed weighted sum model is a compromise between the Prandtl mixing-length model and the constant eddy

viscosity model, which is consistent with the results in Figure 6-12. Additionally, the Reynolds shear stress that is predicted by the weighted sum model is able to mirror the trend of those that have been observed, and it is relatively close to the measured values, which is a strong indication of the accuracy advantage that this model possesses.

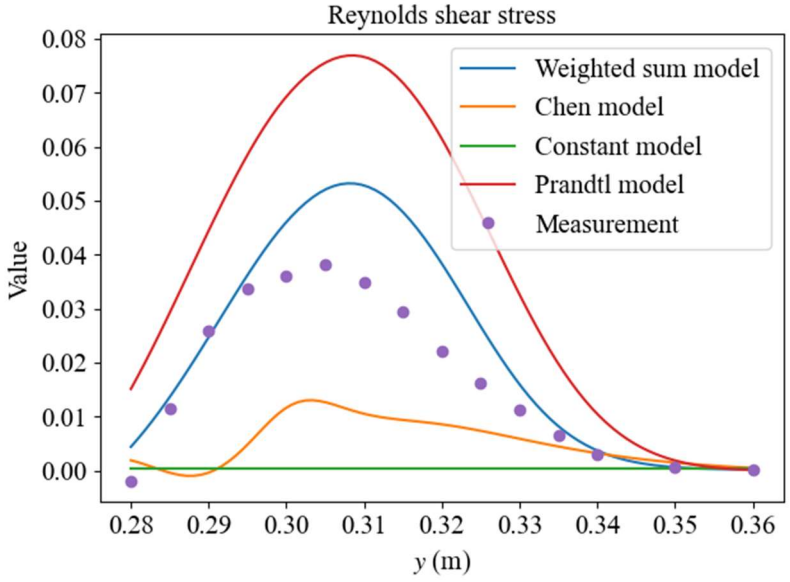


Figure 6-13. Comparison between the predicted and measured Reynolds shear stress.

6.4 Case Study: Flow Past a Single Hill

In this section, the numerical example described in Chapter 4 is utilized, namely the flow past a two-dimensional hill (see Figure 4-3 for details). The neural network has six hidden layers, each layer containing 40 neurons. Tanh is used as the activation function. On each of the four boundaries, there are 50 evenly distributed collocation points to calculate the residuals of the boundary condition loss. Furthermore, the collocation points, which constitute a uniformly distributed 50 by 50 lattice, were organized within the computational domain to compute the residuals of the governing equation loss. Adam is selected as the optimizer with a

stable learning rate of 0.0001. The number of iteration 2 is fixed at 50000. The time-averaged velocities at the measurement points listed in Table 4-1 are used for PINN's training, and the Reynolds shear stresses at these points are also used to calculate the Reynolds stress loss.

In this case study, only the Chen model (candidate model 1) and the constant eddy viscosity model (candidate model 2) are selected as the candidate models. Figure 6-14 reveals that in this flow case, the Chen model is more suitable for describing turbulent characteristics than the constant eddy viscosity model, as after 50000 training iterations, the l_2 error of the mainstream velocity u using the Chen model decreased to 11.7%, while the l_2 error of the mainstream velocity u using the constant eddy viscosity model reached 26.7%. Upon implementing the proposed weighted sum RANS turbulence model, it can be observed that the weight coefficients eventually stabilize at around 0.46 and 0.54 for candidate models 1 and 2, respectively. Furthermore, we observed that the use of the weighted sum turbulence model results in a reduction of the l_2 error of u to 11.0%, the lowest value among the three, thus validating the efficacy of the weighted sum turbulence model.

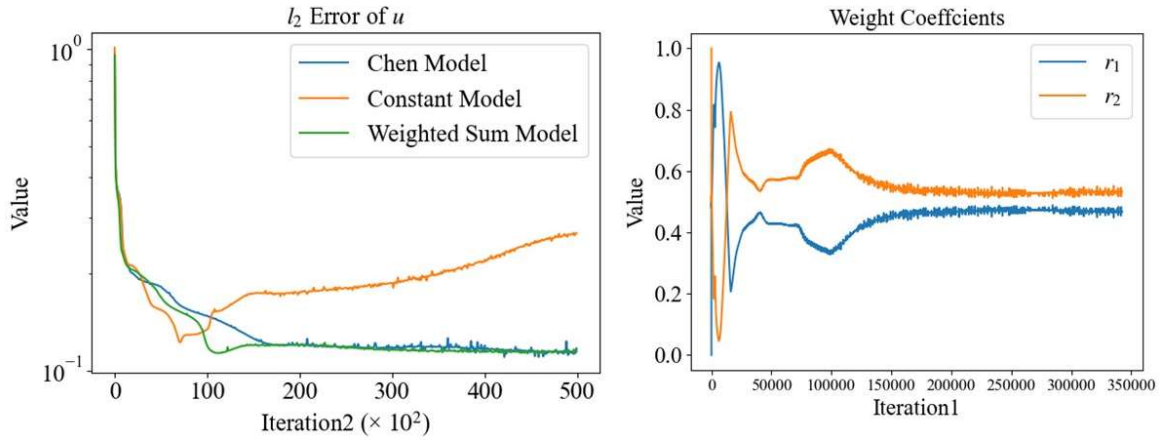


Figure 6-14. The l_2 errors of the mainstream velocity u and the weight coefficients in the weighted sum model.

The above results indicate that the weight coefficients of the weighted sum model vary

depending on the specific flow condition. Specific weight combinations might reduce Reynolds stress loss, therefore attaining optimal simulation accuracy based on the given candidate models, which is the fundamental concept and core value of the proposed weighted sum RANS turbulence model.

6.5 Conclusions

This chapter proposes a weighted sum RANS turbulence model for PINN-based RANS simulations. The existing zero-equation RANS turbulence models are considered to be candidate models. The turbulent viscosity of the proposed model is a linear combination of the turbulent viscosities of the candidate models. The key to the problem then shifts to the optimization of the weights for each candidate. This chapter formulates a new loss term, i.e., the Reynolds stress loss, by using the measured Reynolds stress and the Boussinesq eddy viscosity assumption, therefore attaining the objective of weight optimization via the minimization of the Reynolds stress loss. Meanwhile, a novel PINN structure is proposed for the implementation of the proposed weighted sum model. The feasibility of the proposed model has been verified through a case study of a two-dimensional square cylinder flow and the flow past a two-dimensional hill. Some conclusions can be drawn as follows

- The applicability issue of the RANS turbulence model still exists in the PINN framework, despite the assistance of the measured data. For example, when using two different turbulence models for PINN-based RANS simulation, there is a discrepancy in results. The Prandtl mixing-length model is superior in the two-dimensional square cylinder flow case in the matter of relative errors.

- The proposed model minimizes the Reynolds stress loss by finding the optimal values for weight coefficients r_i. The proposed method calculates the Reynolds stress based on the Boussinesq eddy viscosity assumption and the predicted turbulent viscosity from the weighted sum model. The difference between the measured Reynolds stresses and the predicted values is significantly reduced during the training process.

- The proposed weighted sum RANS turbulence model exhibits optimal performance in both two flow cases. In comparison to existing zero-equation models, the discrepancy between the simulated mainstream velocities and the measurements is minimized when the weighted sum model is adopted.

- The proposed weighted sum RANS turbulence model can also be used to modify the pretrained models using the conventional turbulence models. When using an inappropriate turbulence model for pretraining of a PINN, the modified results show significant improvements compared to those before adjustment, and the relative error is greatly reduced.

However, it must be acknowledged that the method proposed in this chapter also has its drawbacks. Firstly, it exacerbates the problem of high computational cost in PINN's training to a certain extent. Secondly, this method requires training data and cannot be implemented without known Reynolds stress. The impact of the measured Reynolds stress on the results of the proposed model remains uncertain. To address these issues, more investigations are necessitated, becoming one of the key focuses of future work.

CHAPTER 7

CONCLUSIONS & RECOMMENDATIONS

This chapter aims to encapsulate the novel contributions previously discussed in this thesis and provide prospects for future research.

7.1 Conclusions

In the first chapter of this thesis, the background of turbulence simulation is presented for the first time. This chapter gives a detailed description of the physical governing equations. i.e., the RANS equations, for turbulence simulation. Additionally, this chapter introduces the traditional numerical methods that are used for turbulence simulation, namely various CFD methods, and provides a comprehensive analysis of the benefits and drawbacks associated with these approaches in turbulence simulation. Following that, this chapter presents a revolutionary PDE solver designed on the basis of machine learning methods, which is referred to as PINN. The utilizations of PINNs in both forward and inverse physical problems involving ODEs and PDEs are presented. Then, a short introduction to the current applications of PINNs in fluid mechanics and turbulence modeling is presented. The effectiveness of PINN directly used in turbulence simulation appears to be average at the present time, with the exception of a few report cases. That is, the further application and development of PINNs in the field of turbulence simulation is hindered by a number of factors, which is what sparks the discussions on research motivation in Section 1.2. The three main factors currently limiting the further

development of PINN-based RANS simulation are summarized as follows.

Firstly, the loss function of a PINN is made up of a number of distinct and complicated components, and its convergence turns into a non-convex optimization, which often leads to an unsatisfying convergence performance. Secondly, PINN is a PDE solver based on a neural network structure, while the nonlinear expression and feature learning capabilities of the PINN are constrained by its neural network architecture. For example, the frequency principle is a phenomenon that usually occurs while training a neural network, but it may also be discovered during PINN's training. Thirdly, within the realm of PINN-based RANS simulation, the issue of poor applicability of RANS turbulence models persists. Since the Reynolds stress components induced by Reynolds averaging cause the governing equations to be unclosed, it is important to model the Reynolds stress components in mathematical forms to solve the RANS equations. The RANS turbulence model is the aggregate name for various physics-based mathematical models that are used to describe the Reynolds stress components. However, studies have shown that the most suitable RANS turbulence model does not remain constant over a variety of flow cases. This problem is referred to as the applicability issue of the RANS turbulence model. This kind of issue is still present not only in conventional CFD-based RANS simulations but also in PINN simulations, becoming a major factor restricting the development of RANS simulations. The above three factors constitute the main constraints on PINN-based RANS simulation.

In Chapter 2, an overview of the RANS equations, which are the governing equations used in turbulence simulation in this thesis, is first provided. This chapter explores in detail how the NS equations evolve step by step into the RANS equations. The rules regarding Reynolds averaging are elaborated in detail as well. The necessity of RANS turbulence modeling and common turbulence models are also elaborated here. In order to close the RANS equations,

additional equations need to be introduced to describe the unknown variables. According to the number of equations introduced, common RANS turbulence models can be classified into zero-equation models, one-equation models, and two-equation models, all of which have been elaborated one by one in that section.

Following that, a review of the conventional numerical approaches to solving the RANS equations is carried out. For complex PDEs such as the RANS equations, finding their analytical solutions is a daunting task (in fact, even the existence of the analytical solutions of the RANS equations has not been proven), so researchers can only rely on numerical methods to approximate the solution of the RANS equations. The FVM, a kind of CFD method and the most commonly used numerical method to solve the RANS equations, has been reviewed from its principle to application. Another approach that is emphasized is the FVM method. Although it is used extensively in the field of solid mechanics, FVM also has a slot in the field of turbulence modeling. At the end of this chapter, a systematic review of the PINN has been conducted. In the beginning, its history is broken down, and then the evolution of PINN in the field of fluid mechanics is discussed in more detail. In the next step, the application of PINN in RANS simulation has been expounded upon, and in the last step, the limiting factors in PINN-based RANS simulation at the present level are evaluated and backed by research instances.

In this thesis, four ameliorative methods for physics-informed machine learning based RANS simulations are presented. Chapter 3 proposes one of these ameliorative methods, i.e., a novel self-adaptive loss balancing approach which is referred to as dpPINN. This strategy is achieved by rewriting the total loss function in the original PINN. The proposed strategy constructs loss functions based on specific flow variables, rather than using the traditional boundaries and data losses. Following a designated number of training iterations, the weight

coefficient for each loss component will be reassigned. The update is based on the relative error between the predictions of these flow variables and the measured values. This chapter uses a building outdoor flow case to validate the proposed method. A zero-equation RANS turbulence model is then adopted in order to simulate the flow field surrounding a scale model of a building that is located inside a wind tunnel. The utilization of sparse near-wall wind velocity data is for the purpose of supervised learning.

Additionally, the experiment data in a wind tunnel test conducted in Japan serves as the database that verifies the dpPINN's feasibility. In addition to this, the influence that various neural network structures and incorporated turbulence models have on the prediction accuracy of dpPINN is investigated. The dpPINN is demonstrated to be capable of offering an auxiliary means to simulate the flow field around a bluff body based on the findings. Lastly, according to the findings, the proposed dynamic loss balancing strategy can successfully speed the early convergence of a PINN and significantly increase its accuracy in the middle to late stages of training, which is beneficial for alleviating the aforementioned PINN's convergence issue.

The second ameliorative method for physics-informed machine learning based RANS simulations is presented in Chapter 4. For ease of understanding, the term fidelity should be first introduced. To be noted here, initially, fidelity is a term that describes the degree to which a prototype is similar to the original product. Here, the term fidelity is regarded as the degree of similarity and detail preservation between the simulated or measured flow data and the theoretical solutions. The higher the fidelity, the closer it is to the theoretical solutions, while the lower the fidelity, it is believed to only roughly conform to the trend, and basically lose details and accuracy.

As an emerging algorithm, multifidelity modeling algorithm balances the accessibility of

low-fidelity data with the accuracy of high-fidelity data, attempting to rely on low-fidelity data trends and sparse high-fidelity data accuracy to restore theoretical solutions. In this chapter, the results based on the PINN-based RANS simulation are regarded to have a low fidelity owing to the faults that were mentioned before, but the measurement data is deemed to have a high fidelity.

The NIF algorithm proposed by Perdikaris *et al.* (2017) is adopted here for multifidelity modeling. In the NIF algorithm, the multifidelity model is generally constructed based on the GP regression model. In this chapter, the multifidelity model is essentially a GP regression model that takes spatial coordinates and low-fidelity data as inputs, and outputs high-fidelity data. It is also equipped with a well-built Gaussian kernel to capture the nonlinear nonfunctional cross-correlations of space-dependent nonlinearity.

The multifidelity model that was proposed has been validated by two flow cases, and the results attained are astonishing. In spite of the fact that PINN has a low fidelity of accuracy, the findings demonstrate that it is able to accurately capture the trend of theoretical solutions in the computational domain. Furthermore, sparse measurement data also provides a corrective function for low-fidelity simulations obtained from the PINN. The findings obtained from the multifidelity model demonstrate strong concordance with the measured values. Meanwhile, as PINN only aims to provide rough low-fidelity data, the applicability of RANS turbulence models can be partially alleviated.

Chapter 5 presents the third ameliorative method for physics-informed machine learning based RANS simulations. This chapter starts with the function expression module of a PINN, i.e., the FCNN mainbody. In previous experience, the FCNN has been known to suffer some difficulties, such as the difficulty in learning high-frequency features. This issue is mirrored in

the PINN-based RANS simulation as the phenomena of the lack of turbulent details in local regions. It is praiseworthy that the dressed quantum circuit, which is a hybrid classical-quantum algorithm, has proved its capacity to speed up the learning of high-frequency characteristics in advance in the numerical example that has been provided. In light of this, the aim of this chapter is to utilize a dressed quantum circuit as an alternative to the conventional fully connected layers that serve as the backbone of a PINN. Then, the proposed model is referred to as PIHCQNN.

The effectiveness of PIHCQNN has been validated on four forward problems involving PDEs or ODEs, as well as one inverse problem involving ODEs, which includes the forward problem of simulating the flow around a square cylinder using the RANS equations. According to the findings, the results achieved through the utilization of the PIHCQNN model have demonstrated a degree of enhancement in comparison to the conventional PINNs. Such a result also indicates that the embedding of quantum layers in the FCNN structure can enhance the nonlinear expression and feature learning capabilities of a PINN to a certain extent.

An innovative turbulence model for PINN-based RANS simulation is proposed in Chapter 6. A persistent and challenging issue in RANS turbulence modeling has always been the poor applicability of various turbulence models. Within the realm of RANS turbulence simulation, the linear eddy viscosity models that are founded on the Boussinesq eddy viscosity assumption have consistently been the predominant models.

The linear eddy viscosity models can be further subdivided into zero-equation models, one-equation models, two-equation models, and so on, depending on the number of additional equations introduced. To be mentioned, the zero-equation models have certain advantages in PINN-based RANS simulation, mainly due to their simple forms and the absence of additional

unknowns and PDEs for solving the RANS equations. The proposed model is referred to as the weighted sum RANS turbulence model, which may be viewed as a linear combination of the conventional zero-equation RANS turbulence models. Then, the challenge of turbulence modeling is transformed into determining the linear coefficients in the weighted sum model, namely the weight coefficients r_i.

To address this issue, this chapter introduces the concept of Reynolds stress loss, which refers to the difference between the PINN-predicted Reynolds stress components and the measured values. The proposed method determines the values of the weight coefficient r_i by minimizing the Reynolds stress loss. That is, the proposed method aims to find a set of optimal r_i values that minimize the difference between the simulated Reynolds stress components and the measured values.

Meanwhile, this chapter also proposes a novel PINN structure to achieve simultaneous optimization of the traditional PINN loss and the Reynolds stress loss. The proposed turbulence model has been validated on a square cylinder flow, and it has shown superior accuracy performance in comparison to conventional zero-equation models. The proposed turbulence model is concurrently employed to rectify the simulation results derived from the pretraining PINN models when conventional zero-equation RANS turbulence models are used, still achieving satisfactory results. The results demonstrate that, during the simulation process, the weight of the most suitable candidate model in the weighted sum model will inherently be emphasized, which may greatly alleviate the applicability issue of RANS turbulence models.

In summary, the three existing challenges in the PINN-based RANS simulation concluded in this thesis have been addressed through the four proposed ameliorative methods. The thesis has preliminarily achieved its research objective, which is to develop physics-informed

machine learning methods which are accurate, dependable, stable, and affordable to address fluid dynamics issues involving the RANS equations.

7.2 Recommendations

In this section, insights on future work will be provided based on the research work conducted thus far.

7.2.1 Discovering underlying physics in turbulence

As aforementioned, the determination of the turbulent viscosity at a particular position is one of the most important challenges that should be addressed in RANS turbulence modeling. Although Chapter 6 makes an effort to enhance the generalization capability of RANS turbulence modeling through the linear superposition of zero-equation RANS turbulence models, it is still far from achieving the desired result.

The reason for this is that the zero-equation model has not been the focus of research over the past decades, and there are not many options available for the choice of candidate models. This means that these candidate models cannot serve as a complete basis for a universal RANS turbulence model. In addition, with an increasing number of candidate models, there is also the danger of being mired in a situation known as a local optimal solution.

As is known to all, PINN is able to address inverse problems by identifying unknown coefficients in physical governing equations. However, in RANS turbulence simulations, the governing equation that is used to characterize turbulent viscosity is fully unknown, not only the coefficients. This is an issue that cannot be solved by PINN itself.

Chen *et al.* (2021) successfully discovered governing PDEs for nonlinear spatiotemporal systems using PINN and sparse regression from scarce and noisy data. A collection of partial derivatives of each potential component element in the physical governing equation, i.e., the library, was created by the authors, as can be seen in Figure 7-1. Unknown physical governing equations may be identified by utilizing the residual between the linear combination of these partial derivatives and the observed values as the loss function of the neural network.

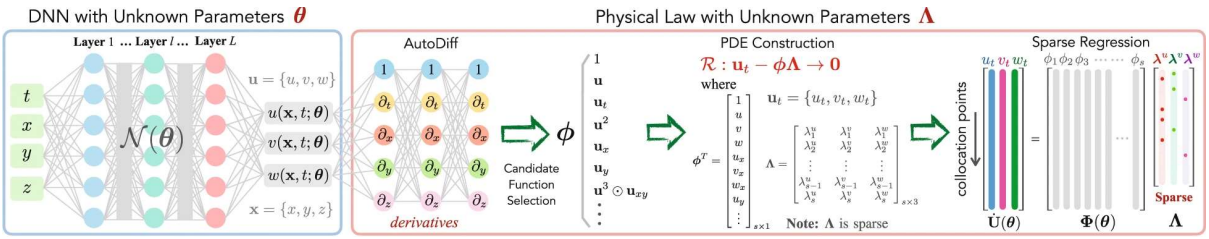


Figure 7-1. The structure of the neural network for governing equation identification proposed by Chen *et al.* (2021).

However, it must be acknowledged that such kind of methods requires a large amount of high-quality data for inference (de Silva *et al.* 2020, Zhou *et al.* 2022). When it comes to turbulence simulations, this circumstance does not frequently occur. This is also why in this thesis, an attempt is made to construct a new RANS turbulence model using a linear combination of the existing turbulence models rather than starting from scratch. The benefit of this approach is that the interpolation and extrapolation accuracy of the new turbulence model will remain reasonable. In summary, the strategy illustrated in Figure 7-1 offers valuable insights for ongoing research and merits additional investigation.

7.2.2 Empirical constants in RANS turbulence models

As mentioned in Section 6.1, the RANS turbulence model, as an artificially designed mathematical model for closure of the RANS equations, inevitably include a few empirical

constants (Margheri *et al*. 2014).

Nevertheless, it is found that the determination of the values of the empirical constants in RANS turbulence models might be a challenging issue (Geng and Escaler 2020). On one hand, the values of these empirical constants are usually determined based on experience or experiments (Xiao and Cinnella 2019). Despite the existence of recommended values for these constants, there is no unified standard, and strictly speaking, they should be determined on a case-by-case basis (Lien *et al.* 1998). The values of these empirical constants, on the other hand, will have a certain impact on the RANS simulation results (Yazdani and Tahani 2024).

Given that the values of these constants have a certain impact on the final simulation results, a supervised learning strategy is suggested to be adopted to optimize the values of these empirical constants. The question now shifts to how to integrate the supervised learning strategy into the training process of PINN to guide the training of these empirical constants.

It is recommended that similar to when PINNs are adopted to solve inverse problems, empirical constants can be considered as parameters to be optimized during the training process of the neural network. The conventional physics-based loss that a PINN adopts is not utilized as the basis of training; rather, their values should be optimized by minimizing the Reynolds stress loss, as described in Chapter 6.

There have been attempts to solve such an issue. For example, as shown in Figure 7-2, Rui *et al.* (2024) proposed a weak-form zero-equation RANS turbulence model that releases the constants in the turbulence model and converts them into neural network outputs. In their proposed model, the distribution of the empirical constant value in space is found by incorporating deviation errors in the loss function and defining upper and lower limits, hence converting the RANS turbulence model into a weak-form model. The findings indicate that the

model can achieve results with lower relative errors. However, in their model, challenges like the inability to precisely simulate the positive pressure distributions on the windward side still persist, and further research is desired to address such issues.

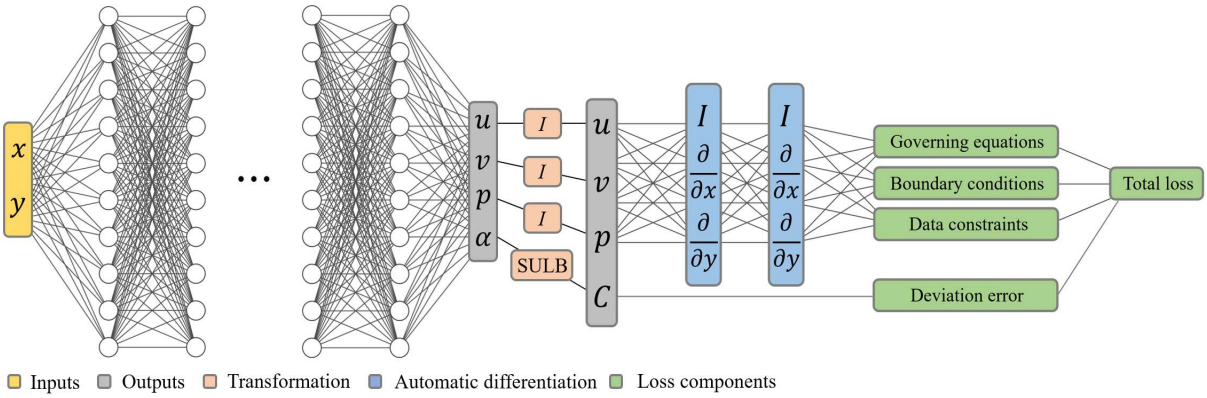


Figure 7-2. The PINN structure for RANS simulations using a weak-form turbulence model adopted by Rui *et al.* (2024).

7.2.3 Quantum transfer learning

Transfer learning is a commonly used strategy in the realm of PINN, and fine-tuning in transfer learning is a common approach in model-based transfer learning (Goswami *et al.* 2020, Liu *et al.* 2023, Prantikos *et al.* 2023). Fine-tuning eliminates the need to train the neural network from the start for new tasks (Tajbakhsh *et al.* 2016, Touvron *et al.* 2023). As a result of its increased computational cost in comparison to conventional numerical methods, PINN is better suited for fine-tuning applications.

Within the realm of transfer learning, VQCs have demonstrated the capability to extract features and post-process the features derived from conventional neural network models (Mari *et al.* 2020). More specifically, VQCs, as well as dressed quantum circuits, may be used as feature extractors to extract knowledge in general tasks and pass the pre-acquired knowledge

to conventional neural networks for post-processing. Additionally, they can be used as post-processing modules to further compress and process features that have been retrieved by conventional models (Azevedo *et al.* 2022, Mogalapalli *et al.* 2022, Kim *et al.* 2023). An illustration of the quantum transfer learning framework that Otgonbaatar *et al.* (2023) adopted may be found in Figure 7-3. As can be seen from the figure, the feature extractor adopted in this quantum transfer learning model is a pre-trained classical neural network model, namely VGG16. This is in line with the conventional transfer learning strategy. The only difference is that the features extracted from the VGG16 model are post-processed using quantum machine learning models in quantum transfer learning. As reported in their work, quantum models generally have higher accuracy than classical models.

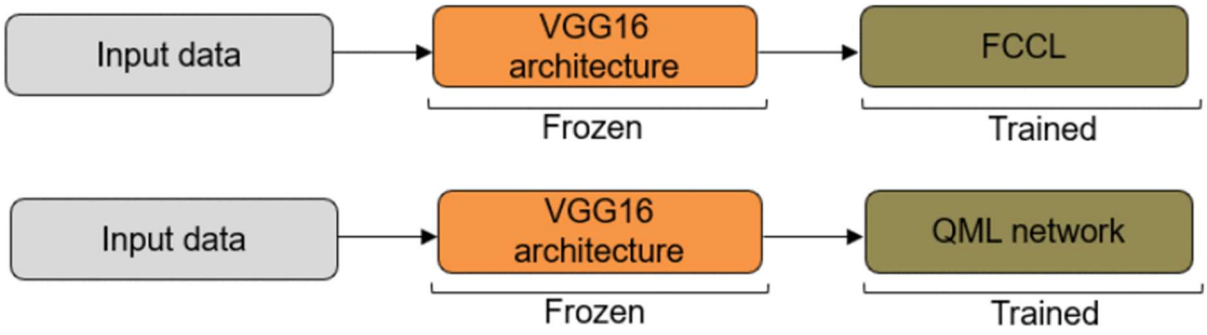


Figure 7-3. The quantum transfer learning strategy adopted by Otgonbaatar *et al.* (2023).

The applicability issue of turbulence models in PIHCQNN-based RANS simulations is expected to be alleviated through the use of quantum transfer learning. When RANS simulation is performed using a variety of turbulence models, the flow velocity and pressure field that are derived from simulations are distinct from one another. However, despite the fact that there are disparities, there is consistency in the overall trend of solutions within the domain. Therefore, the flow velocity and pressure fields that are calculated using a certain turbulence model may be considered to be the source domain, while those calculated using a different turbulence

model can be considered to be the target domain. It is feasible to accomplish the task of the target domain by employing a neural network model that has already been trained in the source domain and a minimal number of labeled training samples in the target domain, which can be accomplished by freezing the parameters in the inner layers and releasing those in the outermost layers for training (Tang *et al.* 2022, Prantikos *et al.* 2023).

REFERENCE

Abbas, H. (2024). "Quantum Machine Learning-Models and Algorithms: Studying Quantum Machine Learning Models and Algorithms for Leveraging Quantum Computing Advantages in Data Analysis, Pattern Recognition, and Optimization." *Australian Journal of Machine Learning Research and Applications* **4**(1): 221-232.

Abrahamson, S. and Lonnes, S. (1995). "Uncertainty in Calculating Vorticity from 2D Velocity Fields Using Circulation and Least-Squares Approaches." *Experiments in Fluids* **20**(1): 10-20.

Adhikary, S., Dangwal, S. and Bhowmik, D. (2020). "Supervised Learning with a Quantum Classifier Using Multi-Level Systems." *Quantum Information Processing* **19**: 1-12.

Albino, A. S., Jardim, L. C., Knupp, D. C., Neto, A. J. S., Pires, O. M. and Nascimento, E. G. S. (2022). "Solving Partial Differential Equations on near-Term Quantum Computers." *arXiv preprint arXiv:.05805*.

Almajid, M. M. and Abu-Al-Saud, M. O. (2022). "Prediction of Porous Media Fluid Flow Using Physics Informed Neural Networks." *Journal of Petroleum Science and Engineering* **208**: 109205.

Almeida, G., Durao, D. and Heitor, M. (1993). "Wake Flows Behind Two-Dimensional Model Hills." *Experimental Thermal and Fluid Science* **7**(1): 87-101.

Apsley, D. (2024). "Turbulence Modelling." https://personalpages.manchester.ac.uk/staff/da

vid.d.apsley/lectures/comphydr/turbmodel.pdf

Arzani, A., Wang, J.-X. and D'Souza, R. M. (2021). "Uncovering Near-Wall Blood Flow from Sparse Data with Physics-Informed Neural Networks." *Physics of Fluids* **33**(7): 071905.

Ashton, N., West, A., Lardeau, S. and Revell, A. (2016). "Assessment of RANS and DES Methods for Realistic Automotive Models." *Computers & Fluids* **128**: 1-15.

Azevedo, V., Silva, C. and Dutra, I. (2022). "Quantum Transfer Learning for Breast Cancer Detection." *Quantum Machine Intelligence* **4**(1): 5.

Bai, X.-D., Wang, Y. and Zhang, W. (2020). "Applying Physics Informed Neural Network for Flow Data Assimilation." *Journal of Hydrodynamics* **32**(6): 1050-1058.

Baker, C. and Brockie, N. (1991). "Wind Tunnel Tests to Obtain Train Aerodynamic Drag Coefficients: Reynolds Number and Ground Simulation Effects." *Journal of Wind Engineering and Industrial Aerodynamics* **38**(1): 23-28.

Baker, C., Hemida, H., Iwnicki, S., Xie, G. and Ongaro, D. (2011). "Integration of Crosswind Forces into Train Dynamic Modelling." *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail Rapid Transit* **225**(2): 154-164.

Baldwin, B. and Barth, T. (1991). A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows. 29th aerospace sciences meeting.

Baldwin, B. and Lomax, H. (1978). Thin-Layer Approximation and Algebraic Model for Separated Turbulentflows. 16th aerospace sciences meeting.

Bassi, F., Ghidoni, A., Perbellini, A., Rebay, S., Crivellini, A., Franchina, N. and Savini, M.

(2014). "A High-Order Discontinuous Galerkin Solver for the Incompressible RANS and k–Ω Turbulence Model Equations." *Computers & Fluids* **98**: 54-68.

Bauer, W., Haag, O. and Hennecke, D. (2000). "Accuracy and Robustness of Nonlinear Eddy Viscosity Models." *International Journal of Heat and Fluid Flow* **21**(3): 312-319.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A. and Siskind, J. M. (2018). "Automatic Differentiation in Machine Learning: A Survey." *Journal of Marchine Learning Research* **18**: 1-43.

Benedetti, M., Realpe-Gómez, J. and Perdomo-Ortiz, A. (2018). "Quantum-Assisted Helmholtz Machines: A Quantum–Classical Deep Learning Framework for Industrial Datasets in near-Term Devices." *Quantum Science Technology* **3**(3): 034007.

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S. and Menke, T. (2022). "Noisy Intermediate-Scale Quantum Algorithms." *Reviews of Modern Physics* **94**(1): 015004.

Bischof, R. and Kraus, M. (2021). "Multi-Objective Loss Balancing for Physics-Informed Deep Learning." *arXiv preprint arXiv:.09813*.

Bishop, C. M. (1994). "Neural Networks and Their Applications." *Review of scientific instruments* **65**(6): 1803-1832.

Biswas, S. K. and Anand, N. (2023). "Three-Dimensional Laminar Flow Using Physics Informed Deep Neural Networks." *Physics of Fluids* **35**(12): 121703.

Boris, J. P., Grinstein, F. F., Oran, E. S. and Kolbe, R. L. (1992). "New Insights into Large Eddy

Simulation." *Fluid Dynamics Research* **10**(4-6): 199.

Brockie, N. J. and Baker, C. (1990). "The Aerodynamic Drag of High Speed Trains." *Journal of Wind Engineering and Industrial Aerodynamics* **34**(3): 273-290.

Cai, S., Mao, Z., Wang, Z., Yin, M. and Karniadakis, G. E. (2022). "Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review." *Acta Mechanica Sinica* **37**: 1727-1738.

Cao, Y., Fang, Z., Wu, Y., Zhou, D.-X. and Gu, Q. (2019). "Towards Understanding the Spectral Bias of Deep Learning." *arXiv preprint arXiv:.01198*.

Casey, M. and Wintergerste, T. (2000). "European Research Community on Flow, Turbulence and Combustion." *ERCOFTAC Best Practice Guidelines: ERCOFTAC Special Interest Group on Quality and Trust in Industrial CFD*.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X. and Cincio, L. (2021). "Variational Quantum Algorithms." *Nature Reviews Physics* **3**(9): 625-644.

Challoner, A. and Gill, L. (2014). "Indoor/Outdoor Air Pollution Relationships in Ten Commercial Buildings: $PM_{2.5}$ and NO_2." *Building and Environment* **80**: 159-173.

Chen, H., Katelhon, E. and Compton, R. G. (2022). "Predicting Voltammetry Using Physics-Informed Neural Networks." *Journal of Physical Chemistry Letters* **13**(2): 536-543.

Chen, Q. (1995). "Comparison of Different k-e Models for Indoor Air Flow Computations." *Numerical Heat Transfer, Part B Fundamentals* **28**(3): 353-369.

Chen, Q. and Srebric, J. (2000). "Application of CFD Tools for Indoor and Outdoor Environment Design." *International Journal on Architectural Science* **1**(1): 14-29.

Chen, Q. and Xu, W. (1998). "A Zero-Equation Turbulence Model for Indoor Airflow Simulation." *Energy and Buildings* **28**(2): 137-144.

Chen, S. Y.-C., Huang, C.-M., Hsing, C.-W. and Kao, Y.-J. (2021). "An End-to-End Trainable Hybrid Classical-Quantum Classifier." *Machine Learning: Science and Technology* **2**(4): 045021.

Chen, Z.-W., Rui, E.-Z., Liu, T.-H., Ni, Y.-Q., Huo, X.-S., Xia, Y.-T., Li, W.-H., Guo, Z.-J. and Zhou, L. (2022). "Unsteady Aerodynamic Characteristics of a High-Speed Train Induced by the Sudden Change of Windbreak Wall Structure: A Case Study of the Xinjiang Railway." *Applied Sciences* **12**(14): 7217.

Chen, Z., Liu, T., Chen, X., Xie, T. and Li, W. (2017). Comparison of the Trackside Pressure with Respect to Different Nose Lengths of High-Speed Trains. 2nd International Conference on Industrial Aerodynamics. Qingdao, China, DEStech Transactions on Engineering and Technology Research.

Chen, Z., Liu, T. and Li, W. (2020). "Numerical Analysis of Different Nose Shapes on the Train Aerodynamic Performance at a Windbreak Transition under Crosswinds." *Journal of Applied Mathematics and Physics* **8**(11): 2519-2525.

Chen, Z., Liu, Y. and Sun, H. (2021). "Physics-Informed Learning of Governing Equations from Scarce Data." *Nature Communications* **12**(1): 6136.

Chitta, V., Dhakal, T. P. and Walters, D. K. (2013). Development and Application of a New

Four-Equation Eddy-Viscosity Model for Flows with Transition, Curvature and Rotation Effects. Fluids Engineering Division Summer Meeting, American Society of Mechanical Engineers.

Chiu, P.-H., Wong, J. C., Ooi, C., Dao, M. H. and Ong, Y.-S. (2022). "CAN-PINN: A Fast Physics-Informed Neural Network Based on Coupled-Automatic–Numerical Differentiation Method." *Computer Methods in Applied Mechanics and Engineering* **395**: 114909.

Choi, S., Jung, I., Kim, H., Na, J. and 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.

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S. and Wossnig, L. (2018). "Quantum Machine Learning: A Classical Perspective." *Proceedings of the Royal Society A: Mathematical, Physical Engineering Sciences* **474**(2209): 20170551.

Coroneo, M., Montante, G., Paglianti, A. and Magelli, F. (2011). "CFD Prediction of Fluid Flow and Mixing in Stirred Tanks: Numerical Issues About the Rans Simulations." *Computers & Chemical Engineering* **35**(10): 1959-1968.

Craft, T., Launder, B. and Suga, K. (1997). "Prediction of Turbulent Transitional Phenomena with a Nonlinear Eddy-Viscosity Model." *International Journal of Heat and Fluid Flow* **18**(1): 15-28.

Crivellini, A., D'Alessandro, V. and Bassi, F. (2013). "High-Order Discontinuous Galerkin Rans Solutions of the Incompressible Flow over a Delta Wing." *Computers & Fluids* **88**:

663-677.

Crivellini, A., D'Alessandro, V. and Bassi, F. (2013). "High-Order Discontinuous Galerkin Solutions of Three-Dimensional Incompressible Rans Equations." *Computers & Fluids* **81**: 122-133.

De La Mata, F. F., Gijón, A., Molina-Solana, M. and Gómez-Romero, J. (2023). "Physics-Informed Neural Networks for Data-Driven Simulation: Advantages, Limitations, and Opportunities." *Physica A: Statistical Mechanics and its Applications* **610**: 128415.

De Silva, B. M., Higdon, D. M., Brunton, S. L. and Kutz, J. N. (2020). "Discovery of Physics from Data: Universal Laws and Discrepancies." *Frontiers in Artificial Intelligence* **3**: 25.

Dehaghani, N. B., Aguiar, A. P. and Wisniewski, R. (2024). "A Hybrid Quantum-Classical Physics-Informed Neural Network Architecture for Solving Quantum Optimal Control Problems." *arXiv preprint arXiv:.15015*.

Deng, E., Yang, W., Deng, L., Zhu, Z., He, X. and Wang, A. (2020). "Time-Resolved Aerodynamic Loads on High-Speed Trains During Running on a Tunnel–Bridge–Tunnel Infrastructure under Crosswind." *Engineering Applications of Computational Fluid Mechanics* **14**(1): 202-221.

Deng, E., Yang, W., Lei, M., Zhu, Z. and Zhang, P. (2019). "Aerodynamic Loads and Traffic Safety of High-Speed Trains When Passing through Two Windproof Facilities under Crosswind: A Comparative Study." *Engineering Structures* **188**: 320-339.

Diedrichs, B. (2010). "Aerodynamic Crosswind Stability of a Regional Train Model." *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail Rapid*

Transit **224**(6): 580-591.

Du, Y., Huang, T., You, S., Hsieh, M.-H. and Tao, D. (2022). "Quantum Circuit Architecture Search for Variational Quantum Algorithms." *NPJ Quantum Information* **8**(1): 62.

Duraisamy, K., Iaccarino, G. and Xiao, H. (2019). "Turbulence Modeling in the Age of Data." *Annual Review of Fluid Mechanics* **51**(1): 357-377.

Durbin, P. and Shih, T. (2005). "An Overview of Turbulence Modeling." *Modelling and Simulation of Turbulent Heat Transfer* **16**(3): 3-34.

Durrani, F., Cook, M. J. and McGuirk, J. (2015). "Evaluation of LES and RANS CFD Modelling of Multiple Steady States in Natural Ventilation." *Building and Environment* **92**: 167-181.

Eivazi, H., Tahani, M., Schlatter, P. and Vinuesa, R. (2022). "Physics-Informed Neural Networks for Solving Reynolds-Averaged Navier-Stokes Equations." *Physics of Fluids* **34**: 065129.

Eivazi, H. and Vinuesa, R. (2022). "Physics-Informed Deep-Learning Applications to Experimental Fluid Mechanics." *arXiv preprint arXiv:2203.15402*.

Ekaterinaris, J. A. (2005). "High-Order Accurate, Low Numerical Diffusion Methods for Aerodynamics." *Progress in Aerospace Sciences* **41**(3-4): 192-300.

Eymard, R., Gallouët, T. and Herbin, R. (2000). "Finite Volume Methods." *Handbook of numerical analysis* **7**: 713-1018.

Fang, Z. and Zhan, J. (2019). "A Physics-Informed Neural Network Framework for PDEs on

3D Surfaces: Time Independent Problems." *IEEE Access* **8**: 26328-26335.

Farhadi, A., Mayrhofer, A., Tritthart, M., Glas, M. and Habersack, H. (2018). "Accuracy and Comparison of Standard k-ϵ with Two Variants of k-Ω Turbulence Models in Fluvial Applications." *Engineering Applications of Computational Fluid Mechanics* **12**(1): 216-235.

Fasel, H. (1976). "Investigation of the Stability of Boundary Layers by a Finite-Difference Model of the Navier-Stokes Equations." *Journal of Fluid Mechanics* **78**(2): 355-383.

Gallagher, M., Morden, J., Baker, C., Soper, D., Quinn, A., Hemida, H., Sterling, M. J. J. o. W. E. and Aerodynamics, I. (2018). "Trains in Crosswinds–Comparison of Full-Scale on-Train Measurements, Physical Model Tests and CFD Calculations." *Journal of Wind Engineering and Industrial Aerodynamics* **175**: 428-444.

Gao, F., Wang, H. and Wang, H. (2017). "Comparison of Different Turbulence Models in Simulating Unsteady Flow." *Procedia Engineering* **205**: 3970-3977.

Geng, L. and Escaler, X. (2020). "Assessment of Rans Turbulence Models and Zwart Cavitation Model Empirical Coefficients for the Simulation of Unsteady Cloud Cavitation." *Engineering Applications of Computational Fluid Mechanics* **14**(1): 151-167.

Goswami, S., Anitescu, C., Chakraborty, S. and Rabczuk, T. (2020). "Transfer Learning Enhanced Physics Informed Neural Network for Phase-Field Modeling of Fracture." *Theoretical and Applied Fracture Mechanics* **106**: 102447.

Griol-Barres, I., Milla, S., Cebrián, A., Mansoori, Y. and Millet, J. (2021). "Variational Quantum Circuits for Machine Learning. An Application for the Detection of Weak

Signals." *Applied Sciences* **11**(14): 6427.

Grunloh, T. P. (2019). "Four Equation k-omega Based Turbulence Model with Algebraic Flux for Supercritical Flows." *Annals of Nuclear Energy* **123**: 210-221.

Guerreschi, G. G. and Matsuura, A. Y. (2019). "Qaoa for Max-Cut Requires Hundreds of Qubits for Quantum Speed-Up." *Scientific reports* **9**(1): 6903.

Guillas, S., Glover, N. and Malki-Epshtein, L. (2014). "Bayesian Calibration of the Constants of the k–e Turbulence Model for a CFD Model of Street Canyon Flow." *Computer Methods in Applied Mechanics Engineering* **279**: 536-553.

Guo, M., Haque, A., Huang, D., Yeung, S. and Li, F. (2018). Dynamic Task Prioritization for Multitask Learning. 15th European Conference on Computer Vision (ECCV 2018)**:** 270-287.

Guo, X.-Y. and Fang, S.-E. (2023). "Structural Parameter Identification Using Physics-Informed Neural Networks." *Measurement* **220**: 113334.

Guo, Y., Cao, X., Liu, B. and Gao, M. (2020). "Solving Partial Differential Equations Using Deep Learning and Physical Constraints." *Applied Sciences* **10**(17): 5917.

Hanrahan, S., Kozul, M. and Sandberg, R. (2023). "Studying Turbulent Flows with Physics-Informed Neural Networks and Sparse Data." *International Journal of Heat and Fluid Flow* **104**: 109232.

Harmening, J. H., Peitzmann, F.-J. and el Moctar, O. (2024). "Effect of Network Architecture on Physics-Informed Deep Learning of the Reynolds-Averaged Turbulent Flow Field

around Cylinders without Training Data." *Frontiers in Physics* **12**: 1385381.

Heldmann, F., Berkhahn, S., Ehrhardt, M. and Klamroth, K. (2023). "PINN Training Using Biobjective Optimization: The Trade-Off between Data Loss and Residual Loss." *Journal of Computational physics* **488**: 112211.

Hertwig, D., Efthimiou, G. C., Bartzis, J. G. and Leitl, B. (2012). "CFD-RANS Model Validation of Turbulent Flow in a Semi-Idealized Urban Canopy." *Journal of Wind Engineering and Industrial Aerodynamics* **111**: 61-72.

Hino, T. (1995). "Viscous Flow Computations around a Ship Using One-Equation Turbulence Models." *Journal of the Society of Naval Architects of Japan* **1995**(178): 9-22.

Hou, X., Zhou, X. and Liu, Y. (2024). "Reconstruction of Ship Propeller Wake Field Based on Self-Adaptive Loss Balanced Physics-Informed Neural Networks." *Ocean Engineering* **309**: 118341.

Hu, B. and McDaniel, D. (2023). "Applying Physics-Informed Neural Networks to Solve Navier–Stokes Equations for Laminar Flow around a Particle." *Mathematical and Computational Applications* **28**(5): 102.

Huang, R., Tan, X. and Xu, Q. (2022). "Learning to Learn Variational Quantum Algorithm." *IEEE Transactions on Neural Networks and Learning Systems* **34**(11): 8430 - 8440.

Huang, Y., Zhang, Z. and Zhang, X. (2022). "A Direct-Forcing Immersed Boundary Method for Incompressible Flows Based on Physics-Informed Neural Network." *Fluids* **7**(2): 56.

Huembeli, P. and Dauphin, A. (2021). "Characterizing the Loss Landscape of Variational

Quantum Circuits." *Quantum Science Technology* **6**(2): 025011.

Iaccarino, G., Ooi, A., Durbin, P. and Behnia, M. (2003). "Reynolds Averaged Simulation of Unsteady Separated Flow." *International Journal of Heat and Fluid Flow* **24**(2): 147-156.

Jędrzejewski, M., Poćwierz, M. and Zielonko-Jung, K. (2017). "The Problem of Airflow around Building Clusters in Different Configurations." *Archive of Mechanical Engineering* **64**(3): 401-418.

Jiang, Z., Yan, C., Yu, J. and Yuan, W. (2015). "Practical Aspects of P-Multigrid Discontinuous Galerkin Solver for Steady and Unsteady Rans Simulations." *International Journal for Numerical Methods in Fluids* **78**(11): 670-690.

Jones, D. A. and Titi, E. S. (1992). "Determining Finite Volume Elements for the 2D Navier-Stokes Equations." *Physica D: Nonlinear Phenomena* **60**(1-4): 165-174.

Katz, A. and Sankaran, V. (2011). "Mesh Quality Effects on the Accuracy of CFD Solutions on Unstructured Meshes." *Journal of Computational physics* **230**(20): 7670-7686.

Kavitha, S. and Kaulgud, N. (2024). "Quantum Machine Learning for Support Vector Machine Classification." *Evolutionary Intelligence* **17**(2): 819-828.

Kendall, A., Gal, Y. and Cipolla, R. (2018). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. the IEEE Conference on Computer Vision and Pattern Recognition**:** 7482-7491.

Khan, T. M. and Robles-Kelly, A. (2020). "Machine Learning: Quantum vs Classical." *IEEE Access* **8**: 219275-219294.

Kim, J., Huh, J. and Park, D. K. (2023). "Classical-to-Quantum Convolutional Neural Network Transfer Learning." *Neurocomputing* **555**: 126643.

Kingma, D. P. (2014). "Adam: A Method for Stochastic Optimization." *arXiv preprint arXiv:.1412.6980*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R. and Mahoney, M. W. (2021). "Characterizing Possible Failure Modes in Physics-Informed Neural Networks." *Advances in Neural Information Processing Systems* **34**: 26548-26560.

Kyriienko, O., Paine, A. E. and Elfving, V. E. (2021). "Solving Nonlinear Differential Equations with Differentiable Quantum Circuits." *Physical Review A* **103**(5): 052416.

Lakshminarayana, B. (1986). "Turbulence Modeling for Complex Shear Flows." *AIAA journal* **24**(12): 1900-1917.

Launder, B. E. and Sharma, B. I. (1974). "Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow near a Spinning Disc." *Letters in Heat and Mass Transfer* **1**(2): 131-137.

Launder, B. E. and Spalding, D. B. (1983). The Numerical Computation of Turbulent Flows. Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, Elsevier**:** 96-116.

Le Gratiet, L. and Garnier, J. (2014). "Recursive Co-Kriging Model for Design of Computer Experiments with Multiple Levels of Fidelity." *International Journal for Uncertainty Quantification* **4**(5): 365-386.

Li, C., Li, X., Su, Y. and Zhu, Y. (2012). "A New Zero-Equation Turbulence Model for Micro-Scale Climate Simulation." *Building and Environment* **47**: 243-255.

Li, C., Li, X., Su, Y. and Zhu, Y. (2013). "Zero-Equation Turbulence Model for Outdoor Airflow Simulation." *Journal of Tsinghua University (Science and Technology)* **53**: 589-594.

Li, H., Zhang, Y. and Chen, H. (2020). "Aerodynamic Prediction of Iced Airfoils Based on Modified Three-Equation Turbulence Model." *AIAA journal* **58**(9): 3863-3876.

Li, K. and Chitre, M. (2023). "Data-Aided Underwater Acoustic Ray Propagation Modeling." *IEEE Journal of Oceanic Engineering* **48**(4): 1127-1149.

Li, M., Liu, W., Zhang, L. and He, X. (2015). "Applications of High Order Hybrid DG/FV Schemes for Two-Dimensional Rans Simulations." *Procedia Engineering* **126**: 628-632.

Li, S. and Feng, X. (2022). "Dynamic Weight Strategy of Physics-Informed Neural Networks for the 2D Navier–Stokes Equations." *Entropy* **24**(9): 1254.

Li, T., Qin, D. and Zhang, J. (2019). "Effect of Rans Turbulence Model on Aerodynamic Behavior of Trains in Crosswind." *Chinese Journal of Mechanical Engineering* **32**: 1-12.

Lien, F. S., Kalitzin, G. and Durbin, P. A. (1998). "RANS Modeling for Compressible and Transitional Flows." In Proceedings of the Summer Program (Vol. 1, p. 1998). Stanford University USA.

Lin, T., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017). Focal Loss for Dense Object Detection. IEEE International Conference on Computer Vision (ICCV 2017)**:** 2980-2988.

Liu, M., Liang, L. and Sun, W. (2020). "A Generic Physics-Informed Neural Network-Based Constitutive Model for Soft Biological Tissues." *Computer Methods in Applied Mechanics and Engineering* **372**: 113402.

Liu, Y., Cai, L., Chen, Y. and Wang, B. (2022). "Physics-Informed Neural Networks Based on Adaptive Weighted Loss Functions for Hamilton-Jacobi Equations." *Mathematical Biosciences and Engineering* **19**(12): 12866–12896.

Liu, Y., Liao, S., Yang, Y. and Zhang, B. (2024). "Data-Driven and Physics-Informed Neural Network for Predicting Tunnelling-Induced Ground Deformation with Sparse Data of Field Measurement." *Tunnelling and Underground Space Technology* **152**: 105951.

Liu, Y., Liu, W., Yan, X., Guo, S. and Zhang, C.-a. (2023). "Adaptive Transfer Learning for Pinn." *Journal of Computational physics* **490**: 112291.

Lloyd, S., De Palma, G., Gokler, C., Kiani, B., Liu, Z.-W., Marvian, M., Tennie, F. and Palmer, T. (2020). "Quantum Algorithm for Nonlinear Differential Equations." *arXiv preprint arXiv:.06571*.

Lorini, M., Bassi, F., Colombo, A., Ghidoni, A. and Noventa, G. (2021). "Discontinuous Galerkin Solution of the RANS and kl−k−log(Ω) Equations for Natural and Bypass Transition." *Computers & Fluids* **214**: 104767.

Lu, J., Biswas, S. and Tryggvason, G. (2006). "A DNS Study of Laminar Bubbly Flows in a Vertical Channel." *International Journal of Multiphase Flow* **32**(6): 643-660.

Lubasch, M., Joo, J., Moinier, P., Kiffner, M. and Jaksch, D. (2020). "Variational Quantum Algorithms for Nonlinear Problems." *Physical Review A* **101**(1): 010301.

Luo, S., Vellakal, M., Koric, S., Kindratenko, V. and Cui, J. (2020). Parameter Identification of RANS Turbulence Model Using Physics-Embedded Neural Network. International Conference on High Performance Computing, Springer**:** 137-149.

Lyn, D. and Rodi, W. (1994). "The Flapping Shear Layer Formed by Flow Separation from the Forward Corner of a Square Cylinder." *Journal of fluid Mechanics* **267**: 353-376.

Ma, Y., Tresp, V., Zhao, L. and Wang, Y. (2019). "Variational Quantum Circuit Model for Knowledge Graph Embedding." *Advanced Quantum Technologies* **2**(7-8): 1800078.

Maejima, S., Tanino, K. and Kawai, S. (2024). "Physics-Informed Machine-Learning Solution to Log-Layer Mismatch in Wall-Modeled Large-Eddy Simulation." *Physical Review Fluids* **9**(8): 084609.

Mao, Z., Jagtap, A. D. and Karniadakis, G. E. (2020). "Physics-Informed Neural Networks for High-Speed Flows." *Computer Methods in Applied Mechanics and Engineering* **360**: 112789.

Margossian, C. C. (2019). "A Review of Automatic Differentiation and Its Efficient Implementation." *Wiley interdisciplinary reviews: data mining knowledge discovery* **9**(4): e1305.

Mari, A., Bromley, T. R., Izaac, J., Schuld, M. and Killoran, N. (2020). "Transfer Learning in Hybrid Classical-Quantum Neural Networks." *Quantum* **4**: 340.

Meng, X. and Karniadakis, G. E. (2020). "A Composite Neural Network That Learns from Multi-Fidelity Data: Application to Function Approximation and Inverse PDE Problems." *Journal of Computational Physics* **401**: 109020.

Meng, Y. and Hibi, K. (1998). "Turbulent Measurments of the Flow Field around a High-Rise Building." *Wind Engineers, JAWE* **1998**(76): 55-64.

Menter, F. (1993). "Zonal Two Equation k-Ω Models for Aerodynamic Flows." *AIAA paper* **93**: 2906.

Mfula, A., Kukadia, V., Griffiths, R. and Hall, D. (2005). "Wind Tunnel Modelling of Urban Building Exposure to Outdoor Pollution." *Atmospheric Environment* **39**(15): 2737-2745.

Mikulevicius, R. and Rozovskii, B. L. (2004). "Stochastic Navier--Stokes Equations for Turbulent Flows." *SIAM Journal on Mathematical Analysis* **35**(5): 1250-1310.

Misu, Y. and Ishihara, T. (2018). "Prediction of Frequency Distribution of Strong Crosswind in a Control Section for Train Operations by Using Onsite Measurement and Numerical Simulation." *Journal of Wind Engineering and Industrial Aerodynamics* **174**: 69-79.

Mogalapalli, H., Abburi, M., Nithya, B. and Bandreddi, S. K. V. (2022). "Classical–Quantum Transfer Learning for Image Classification." *SN Computer Science* **3**(1): 20.

Moll, N., Barkoutsos, P., Bishop, L. S., Chow, J. M., Cross, A., Egger, D. J., Filipp, S., Fuhrer, A., Gambetta, J. M. and Ganzhorn, M. (2018). "Quantum Optimization Using Variational Algorithms on Near-Term Quantum Devices." *Quantum Science and Technology* **3**(3): 030503.

Munoz-Paniagua, J., García, J. and Lehugeur, B. (2017). "Evaluation of RANS, SAS and IDDES Models for the Simulation of the Flow around a High-Speed Train Subjected to Crosswind." *Journal of Wind Engineering and Industrial Aerodynamics* **171**: 50-66.

Nakhl, A. C., Quella, T. and Usman, M. (2024). "Calibrating the Role of Entanglement in Variational Quantum Circuits." *Physical Review A* **109**(3): 032413.

Ngo, S. I. and Lim, Y.-I. (2021). "Solution and Parameter Identification of a Fixed-Bed Reactor Model for Catalytic CO_2 Methanation Using Physics-Informed Neural Networks." *Catalysts* **11**(11): 1304.

Ostaszewski, M., Trenkwalder, L. M., Masarczyk, W., Scerri, E. and Dunjko, V. (2021). "Reinforcement Learning for Optimization of Variational Quantum Circuit Architectures." *Advances in Neural Information Processing Systems* **34**: 18182-18194.

Otgonbaatar, S., Schwarz, G., Datcu, M. and Kranzlmüller, D. (2023). "Quantum Transfer Learning for Real-World, Small, and High-Dimensional Remotely Sensed Datasets." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **16**: 9223-9230.

Paciorri, R., Dieudonne, W., Degrez, G., Charbonnier, J.-M., Deconinck, H., Paciorri, R., Dieudonne, W., Degrez, G., Charbonnier, J.-M. and Deconinck, H. (1997). Validation of the Spalart-Allmaras Turbulence Model for Application in Hypersonic Flows. 28th Fluid Dynamics Conference.

Pang, G., Lu, L. and Karniadakis, G. E. (2019). "Fpinns: Fractional Physics-Informed Neural Networks." *SIAM Journal on Scientific Computing* **41**(4): 2603-2626.

Pantidis, P., Eldababy, H., Tagle, C. M. and Mobasher, M. E. (2023). "Error Convergence and Engineering-Guided Hyperparameter Search of PINNs: Towards Optimized I-FENN Performance." *Computer Methods in Applied Mechanics and Engineering* **414**: 116160.

Patel, Y., Mons, V., Marquet, O. and Rigas, G. (2024). "Turbulence Model Augmented Physics-Informed Neural Networks for Mean-Flow Reconstruction." *Physical Review Fluids* **9**(3): 034605.

Peltier, L. and Hambric, S. (2007). "Estimating Turbulent-Boundary-Layer Wall-Pressure Spectra from CFD RANS Solutions." *Journal of Fluids and Structures* **23**(6): 920-937.

Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N. D. and Karniadakis, G. E. (2017). "Nonlinear Information Fusion Algorithms for Data-Efficient Multi-Fidelity Modelling." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **473**(2198): 20160751.

Pioch, F., Harmening, J. H., Müller, A. M., Peitzmann, F.-J., Schramm, D. and Moctar, O. e. (2023). "Turbulence Modeling for Physics-Informed Neural Networks: Comparison of Different Rans Models for the Backward-Facing Step Flow." *Fluids* **8**(2): 43.

Posner, J., Buchanan, C. and Dunn-Rankin, D. (2003). "Measurement and Prediction of Indoor Air Flow in a Model Room." *Energy and Buildings* **35**(5): 515-526.

Prandtl, L. (1925). "Über Die Ausgebildete Turbulenz." *Z. Angew. Math. Mech* **5**: 136-139.

Prandtl, L. (1945). "Über Ein Neues Formelsystem Für Die Ausgebildete Turbulenz." *Nacr. Akad. Wiss. Gottingen, Math-Phys. Kl*: 6-19.

Prantikos, K., Chatzidakis, S., Tsoukalas, L. H. and Heifetz, A. (2023). "Physics-Informed Neural Network with Transfer Learning (TL-PINN) Based on Domain Similarity Measure for Prediction of Nuclear Reactor Transients." *Scientific Reports* **13**(1): 16840.

Pu, J., Peng, W. and Chen, Y. (2021). "The Data-Driven Localized Wave Solutions of the Derivative Nonlinear Schrödinger Equation by Using Improved PINN Approach." *Wave Motion* **107**: 102823.

Qin, S.-M., Li, M., Xu, T. and Dong, S.-Q. (2023). "A-Wpinn Algorithm for the Data-Driven Vector-Soliton Solutions and Parameter Discovery of General Coupled Nonlinear Equations." *Physica D: Nonlinear Phenomena* **443**: 133562.

Qureshi, M. Z. I. and Chan, A. (2020). "Influence of Eddy Viscosity Parameterisation on the Characteristics of Turbulence and Wind Flow: Assessment of Steady RANS Turbulence Model." *Journal of Building Engineering* **27**: 100934.

Raghunathan, R. S., Kim, H.-D. and Setoguchi, T. (2002). "Aerodynamics of High-Speed Railway Train." *Progress in Aerospace Sciences* **38**(6-7): 469-514.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y. and Courville, A. (2019). On the Spectral Bias of Neural Networks. International conference on machine learning, PMLR.

Raissi, M., Perdikaris, P. and 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.

Rao, C., Sun, H. and Liu, Y. (2020). "Physics-Informed Deep Learning for Incompressible Laminar Flows." *Theoretical and Applied Mechanics Letters* **10**(3): 207-212.

Rasmussen, C. E. (2003). Gaussian Processes in Machine Learning, Springer.

Rathore, P., Lei, W., Frangella, Z., Lu, L. and Udell, M. (2024). "Challenges in Training PINNs: A Loss Landscape Perspective." *arXiv preprint arXiv:.01868*.

Rezvani, M. A. and Mohebbi, M. (2014). "Numerical Calculations of Aerodynamic Performance for Atm Train at Crosswind Conditions." *Wind and Structures* **18**(5): 529-548.

Ricci, A. and Blocken, B. (2020). "On the Reliability of the 3d Steady RANS Approach in Predicting Microscale Wind Conditions in Seaport Areas: The Case of the Ijmuiden Sea Lock." *Journal of Wind Engineering and Industrial Aerodynamics* **207**: 104437.

Riel, B., Minchew, B. and 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.

Robbins, H. and Monro, S. (1951). "A Stochastic Approximation Method." *The Annals of Mathematical Statistics*: 400-407.

Rui, E.-Z., Chen, Z.-W., Ni, Y.-Q., Yuan, L. and Zeng, G.-Z. (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.

Rui, E.-Z., Zeng, G.-Z., Ni, Y.-Q., Chen, Z.-W. and Hao, S. (2024). "Time-Averaged Flow Field Reconstruction Based on a Multifidelity Model Using Physics-Informed Neural Network (PINN) and Nonlinear Information Fusion." *International Journal of Numerical Methods*

for Heat and Fluid Flow **34**(1): 131-149.

Rumsey, C. L., Pettersson Reif, B. A. and Gatski, T. B. (2006). "Arbitrary Steady-State Solutions with the k-epsilon Model." *AIAA journal* **44**(7): 1586-1592.

Salim, S. M., Cheah, S. C. and Chan, A. (2011). "Numerical Simulation of Dispersion in Urban Street Canyons with Avenue-Like Tree Plantings: Comparison between RANS and LES." *Building and Environment* **46**(9): 1735-1746.

Sallam, O. and Fürth, M. (2023). "On the Use of Fourier Features-Physics Informed Neural Networks (ff-PINN) for Forward and Inverse Fluid Mechanics Problems." *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* **237**(4): 846-866.

Scarselli, F. and Tsoi, A. C. (1998). "Universal Approximation Using Feedforward Neural Networks: A Survey of Some Existing Methods, and Some New Results." *Neural Networks* **11**(1): 15-37.

Schuld, M. and Killoran, N. (2019). "Quantum Machine Learning in Feature Hilbert Spaces." *Physical review letters* **122**(4): 040504.

Schuld, M., Sinayskiy, I. and Petruccione, F. (2015). "An Introduction to Quantum Machine Learning." *Contemporary Physics* **56**(2): 172-185.

Schuld, M., Sweke, R. and Meyer, J. J. (2021). "Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models." *Physical Review A* **103**(3): 032430.

Sedykh, A., Podapaka, M., Sagingalieva, A., Pinto, K., Pflitsch, M. and Melnikov, A. (2024). "Hybrid Quantum Physics-Informed Neural Networks for Simulating Computational Fluid Dynamics in Complex Shapes." *Machine Learning: Science and Technology* **5**(2): 025045.

Sener, O. and Koltun, V. (2018). Multi-Task Learning as Multi-Objective Optimization. Advances in Neural Information Processing Systems. **31**.

Sharma, R. and Shankar, V. (2022). "Accelerated Training of Physics-Informed Neural Networks (PINNs) Using Meshless Discretizations." *Advances in Neural Information Processing Systems* **35**: 1034-1046.

Shih, T.-H., Liou, W. W., Shabbir, A., Yang, Z. and Zhu, J. (1995). "A New k-ϵ Eddy Viscosity Model for High Reynolds Number Turbulent Flows." *Computers & Fluids* **24**(3): 227-238.

Siegl, P., Wassing, S., Mieth, D. M., Langer, S. and Bekemeyer, P. (2023). "Solving Transport Equations on Quantum Computers-Potential and Limitations of Physics-Informed Quantum Circuits." *CEAS Aeronautical Journal,* https://doi.org/10.1007/s13272-024-00774-2.

Smith, L. M. and Woodruff, S. L. (1998). "Renormalization-Group Analysis of Turbulence." *Annual Review of Fluid Mechanics* **30**(1): 275-310.

Spalart, P. and Allmaras, S. (1992). A One-Equation Turbulence Model for Aerodynamic Flows. 30th aerospace sciences meeting and exhibit.

Stamou, A. and Katsiris, I. (2006). "Verification of a CFD Model for Indoor Airflow and Heat

Transfer." *Building and Environment* **41**(9): 1171-1181.

Sun, Y., Sun, Q. and Qin, K. (2021). "Physics-Based Deep Learning for Flow Problems." *Energies* **14**(22): 7760.

Sweke, R., Kesselring, M. S., van Nieuwenburg, E. P. and Eisert, J. (2020). "Reinforcement Learning Decoders for Fault-Tolerant Quantum Computation." *Machine Learning: Science and Technology* **2**(2): 025005.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B. and Liang, J. (2016). "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?" *IEEE Transactions on Medical Imaging* **35**(5): 1299-1312.

Tang, H., Liao, Y., Yang, H. and Xie, L. (2022). "A Transfer Learning-Physics Informed Neural Network (TL-PINN) for Vortex-Induced Vibration." *Ocean Engineering* **266**: 113101.

Tian, Y., Woodward, M., Stepanov, M., Fryer, C., Hyett, C., Livescu, D. and Chertkov, M. (2023). "Lagrangian Large Eddy Simulations Via Physics-Informed Machine Learning." *Proceedings of the National Academy of Sciences* **120**(34): e2213638120.

Tiberga, M., Hennink, A., Kloosterman, J. L. and Lathouwers, D. (2020). "A High-Order Discontinuous Galerkin Solver for the Incompressible Rans Equations Coupled to the k−ϵ Turbulence Model." *Computers & Fluids* **212**: 104710.

Tobiska, L. and Verfürth, R. (1996). "Analysis of a Streamline Diffusion Finite Element Method for the Stokes and Navier–Stokes Equations." *SIAM Journal on Numerical Analysis* **33**(1): 107-127.

Tominaga, Y. and Stathopoulos, T. (2011). "CFD Modeling of Pollution Dispersion in a Street Canyon: Comparison between LES and RANS." *Journal of Wind Engineering and Industrial Aerodynamics* **99**(4): 340-348.

Torlai, G. and Melko, R. G. (2020). "Machine-Learning Quantum States in the NISQ Era." *Annual Review of Condensed Matter Physics* **11**: 325-344.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P. and Bhosale, S. (2023). "Llama 2: Open Foundation and Fine-Tuned Chat Models." *arXiv preprint arXiv:.09288*.

Van Hooff, T., Blocken, B. and Tominaga, Y. (2017). "On the Accuracy of CFD Simulations of Cross-Ventilation Flows for a Generic Isolated Building: Comparison of RANS, LES and Experiments." *Building and Environment* **114**: 148-165.

Vita, G., Salvadori, S., Misul, D. A. and Hemida, H. (2020). "Effects of Inflow Condition on RANS and LES Predictions of the Flow around a High-Rise Building." *Fluids* **5**(4): 233.

Von Saldern, J. G., Reumschüssel, J. M., Kaiser, T. L., Sieber, M. and Oberleithner, K. (2022). "Mean Flow Data Assimilation Based on Physics-Informed Neural Networks." *Physics of Fluids* **34**(11).

Walters, D. K. and Cokljat, D. (2008). "A Three-Equation Eddy-Viscosity Model for Reynolds-Averaged Navier–Stokes Simulations of Transitional Flow." *Journal of Fluids Engineering* **130**(12): 121401.

Wang, S., Teng, Y. and Perdikaris, P. (2021). "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks." *SIAM Journal on Scientific*

Computing **43**(5): A3055-A3081.

Wang, Y., Yao, Y., Guo, J. and Gao, Z. (2024). "A Practical PINN Framework for Multi-Scale Problems with Multi-Magnitude Loss Terms." *Journal of Computational Physics* **510**: 113112.

Wilcox, D. C. (1988). "Reassessment of the Scale-Determining Equation for Advanced Turbulence Models." *AIAA Journal* **26**(11): 1299-1310.

Wissink, J., Rodi, W. J. F., turbulence and combustion (2003). "DNS of a Laminar Separation Bubble in the Presence of Oscillating External Flow." *Flow, Turbulence and Combustion* **71**: 311-331.

Wong, B. (2020). "Modeling Turbulence with the Navier-Stokes Equations." *J Nonlinear Sci Appl* **13**: 97-99.

Xiang, Z., Peng, W., Liu, X. and Yao, W. (2022). "Self-Adaptive Loss Balanced Physics-Informed Neural Networks." *Neurocomputing* **496**: 11-34.

Xiao, H. and Cinnella, P. (2019). "Quantification of Model Uncertainty in RANS Simulations: A Review." *Progress in Aerospace Sciences*, **108**: 1-31.

Xiao, Z., Ju, Y., Li, Z., Zhang, J. and Zhang, C. (2024). "On the Hard Boundary Constraint Method for Fluid Flow Prediction Based on the Physics-Informed Neural Network." *Applied Sciences* **14**(2): 859.

Xie, J., Chai, Z., Xu, L., Ren, X., Liu, S. and Chen, X. (2022). "3D Temperature Field Prediction in Direct Energy Deposition of Metals Using Physics Informed Neural

Network." *The International Journal of Advanced Manufacturing Technology* **119**(5): 3449-3468.

Xu, Z.-Q. J., Zhang, Y. and Luo, T. (2024). "Overview Frequency Principle/Spectral Bias in Deep Learning." *Communications on Applied Mathematics and Computation*: 1-38.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y. and Ma, Z. (2020). "Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks." *Communications in Computational Physics* **28**(5): 1746-1767.

Yakhot, V., Orszag, S. A., Thangam, S., Gatski, T. and Speziale, C. (1992). "Development of Turbulence Models for Shear Flows by a Double Expansion Technique." *Physics of Fluids A: Fluid Dynamics* **4**(7): 1510-1520.

Yang, J., Yang, Y., Sun, D., Jin, C. and Xiao, X. (2021). "Influence of Urban Morphological Characteristics on Thermal Environment." *Sustainable Cities and Society* **72**: 103045.

Yang, X., Du, Y., Li, L., Zhou, Z. and Zhang, X. (2023). "Physics-Informed Neural Network for Model Prediction and Dynamics Parameter Identification of Collaborative Robot Joints." *IEEE Robotics and Automation Letters* **8**(12): 8462-8469.

Yang, X. and Wang, Z. (2022). "Solving Benjamin–Ono Equation Via Gradient Balanced Pinns Approach." *The European Physical Journal Plus* **137**(7): 864.

Yang, X., Zafar, S., Wang, J.-X. and Xiao, H. (2019). "Predictive Large-Eddy-Simulation Wall Modeling Via Physics-Informed Neural Networks." *Physical Review Fluids* **4**(3): 034602.

Yazdani, S. and Tahani, M. (2024). "Data-Driven Discovery of Turbulent Flow Equations

Using Physics-Informed Neural Networks." *Physics of Fluids* **36**(3): 035107.

Ye, X., Ni, Y.-Q., Ao, W. K. and Yuan, L. (2024). "Modeling of the Hysteretic Behavior of Nonlinear Particle Damping by Fourier Neural Network with Transfer Learning." *Mechanical Systems and Signal Processing* **208**: 111006.

Ye, X., Ni, Y.-Q., Sajjadi, M., Wang, Y. W. and Lin, C. S. (2022). "Physics-Guided, Data-Refined Modeling of Granular Material-Filled Particle Dampers by Deep Transfer Learning." *Mechanical Systems and Signal Processing*, **180**: 109437.

Yuan, L., Ni, Y.-Q., Deng, X.-Y. and 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.

Yucesan, Y. A. and Viana, F. A. (2020). "A Physics-Informed Neural Network for Wind Turbine Main Bearing Fatigue." *International Journal of Prognostics and Health Management* **11**(1): 2594.

Yusuf, S. N. A., Asako, Y., Sidik, N. A. C., Mohamed, S. B. and Japar, W. M. A. A. (2020). "A Short Review on RANS Turbulence Models." *CFD Letters* **12**(11): 83-96.

Zeng, G.-Z., Chen, Z.-W., Ni, Y.-Q. and Rui, E.-Z. (2024). "Investigating Embedded Data Distribution Strategy on Reconstruction Accuracy of Flow Field around the Crosswind-Affected Train Based on Physics-Informed Neural Networks." *International Journal of Numerical Methods for Heat and Fluid Flow*: ahead-of-print.

Zhang, T., Yan, R., Zhang, S., Yang, D. and Chen, A. (2024). "Application of Fourier Feature Physics-Information Neural Network in Model of Pipeline Conveying Fluid." *Thin-*

Walled Structures **198**: 111693.

Zhang, Z., Zhang, W., Zhai, Z. J. and Chen, Q. Y. (2007). "Evaluation of Various Turbulence Models in Predicting Airflow and Turbulence in Enclosed Environments by CFD: Part 2—Comparison with Experimental Data from Literature." *Hvac&R Research* **13**(6): 871-886.

Zheng, X., Montazeri, H. and Blocken, B. (2020). "CFD Simulations of Wind Flow and Mean Surface Pressure for Buildings with Balconies: Comparison of RANS and LES." *Building and Environment* **173**: 106747.

Zhou, T., Yang, Q., Yan, B., Deng, X. and Yuan, Y. (2022). "Detached Eddy Simulation of Turbulent Flow Fields over Steep Hilly Terrain." *Journal of Wind Engineering and Industrial Aerodynamics* **221**: 104906.