DEVELOPMENT OF PHYSICS-INFORMED MACHINE LEARNING METHODS FOR STRUCTURAL ANALYSIS AND PARAMETER IDENTIFICATION

WEIJIA ZHANG

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Civil and Environmental Engineering

Development of Physics-Informed Machine Learning Methods for Structural Analysis and Parameter Identification

Weijia Zhang

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

May 2025

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)

Weijia Zhang (Name of student)

To my family,

for their love and support

ABSTRACT

This research aims to advance the field of physics-informed machine learning (PIML) in four critical aspects and to apply PIML methods to solve forward and inverse problems in structural engineering. The research begins with a detailed introduction to the basic knowledge of PIML including two powerful frameworks, namely physics-informed neural network (PINN) and physics-informed graph neural network (PIGNN), by reviewing extensive relevant literature. The core idea of PIML is to integrate the physical laws described by governing equations into neural network (NN) architectures by incorporating these equations, along with boundary and initial conditions and other essential constraints, as penalty terms in the loss function. Then, the applications of PINN and PIGNN in the field of structural engineering are comprehensively reviewed, and the current limitations of PIML are also discussed and summarized. To address the existing research gaps, several solutions are proposed in this research to enhance the performance of PIML methods.

Automatic differentiation (AD) is a key function of PINN, as it computes the derivatives of the NN output based on the chain rule to form the physics-informed loss function. Vanilla PINN often struggles with high-order governing equations because the AD function should be applied multiple times to compute the high-order derivatives, which inevitably accumulates computational errors. Therefore, the first improvement lies in the introduction of auxiliary outputs of PINN for reducing the highest order of the governing equation. By defining auxiliary outputs representing the lower-order derivatives of the original NN output, the governing

equation can be reformulated in a downscaled form. The effectiveness of the proposed approach is validated through numerical examples involving high-order differential equations, demonstrating significant improvements in both training efficiency and prediction accuracy.

In addition to high-order governing equations, the large number of boundary conditions and their treatment in vanilla PINN also pose computational challenges. Boundary conditions are embedded by defining a penalty term in the total loss function and are satisfied by enforcing this penalty term to approach zero, which is called the "soft" manner. However, this soft enforcement cannot guarantee zero residuals of the boundary after training. Thus, a series of modulating functions are derived as the second improvement for "hard" enforcement of all boundary conditions, thereby converting the original output of NN to automatically satisfy the boundary conditions without the need for a penalty term. The performance of this method is validated through both forward problems, i.e. structural response prediction, and inverse problems, i.e. identifying the unknown rotational stiffness of semi-rigid joints. Numerical case studies and experimental validation are carried out, showing that PINN with hard-embedded boundary conditions outperforms vanilla PINN in both forward and inverse cases.

Although modulating functions are effective for tackling boundary conditions, this method exhibits limitations when applied to problems involving complex and irregular domains. Deriving analytical forms of modulating functions becomes challenging or even impossible in such cases. To address this issue, a unified framework termed two-phase physics-informed neural network (TP-PINN) is proposed as the third improvement. TP-PINN framework employs pretrained NNs, which can be constructed in arbitrary shapes, to replace modulating functions. TP-PINN is not only suitable for irregular domains but also decouples

the enforcement of boundary conditions from the multi-objective loss function by separating the training process into two phases, thereby mitigating the negative impact of multi-objective optimization on computational efficiency. The effectiveness of this novel framework is demonstrated through a forward problem involving the computation of the deformation of an Euler-Bernoulli beam and a Kirchhoff-Love plate, which are governed by high-order ordinary and partial differential equations.

In the analysis of structures with graph-like connectivity, such as truss structures and cable-strut systems, the geometry and topology of structures are important structural information that can be learned within the PIML framework. These inherent graph-like properties inspire the integration of graph neural network (GNN) into PIML frameworks. In the last part of this research, PIGNN, where GNN is leveraged to replace the commonly used fully-connected feedforward neural network (FCFNN), is proposed as the fourth aspect to improve vanilla PINN. The prestress design task of tensegrity structures, a type of prestressed cable-strut structure, is performed by both vanilla PINN and PIGNN. Two-dimensional and three-dimensional tensegrity structures with regular and irregular geometries are investigated. Results show that PIGNN outperforms PINN in terms of efficiency and accuracy by effectively capturing the geometry and topology information of structures.

LIST OF PUBLICATIONS

Journal Papers:

Zhang, W., Ni, Y. Q., Wang, S. M., Yuan, L., & Hao, S. (2025). A novel approach for identifying rotational stiffness of semirigid joints by physics-informed neural networks. *Journal of Computing in Civil Engineering*, 39(4), 04025039. (SCI, Q1)

Zhang, W., Wang, S. M., Ni, Y. Q., Yuan, X., Feng, Y., Yuan, L., & Hao, S. (2025). Physics-enhanced multi-fidelity neural ordinary differential equation for forecasting long-term creep behavior of steel cables. *Thin-Walled Structures*, 208, 112846. (SCI, Q1)

Hao, S., Li, H. W., Ni, Y. Q., **Zhang, W.**, & Yuan, L. (2025). State estimation in structural dynamics through RNN transfer learning. *Mechanical Systems and Signal Processing*, 233, 112767. (SCI, Q1)

Conference Papers:

Zhang, W., Ni, Y. Q., Zhang, J. Y., Wang, S. M., & Dong, Y. (2024). Graph neural network-based force-finding for tensegrity structures. *Proceedings of the International Association for Shell and Spatial Structures Annual Symposium 2024*, *IASS 2024*, 26-30 August 2024, Zurich, Switzerland.

Zhang, W., Ni, Y. Q., Yuan, L., Hao, S., & Wang, S. M. (2023). Structural parameter identification with a physics-informed neural networks-based framework. *Proceedings of the 14th International Workshop on Structural Health Monitoring*, *IWSHM 2023*, 12-14 September 2023, Stanford, California, USA.

Zhang, W., Yuan, X., Yuan, L., Ni, Y. Q., & Wang, S. M. (2023). Neural ODE-based data-driven approach for prediction of the creep behaviour of steel cables. *Proceedings of the International Association for Shell and Spatial Structures Annual Symposium 2023*, *IASS 2023*, 10-14 July 2023, Melbourne, Australia.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my chief supervisor, Prof. Yi-Qing Ni, for offering me the precious opportunity to pursue my PhD and for his expert guidance throughout my PhD studies, as well as his genuine care beyond academia. Prof. Ni has an insightful vision of research. From the moment I joined this group, Prof. Ni encouraged me to think without bounds and delve into the philosophy of matters. Discussions with Prof. Ni were always inspiring, and his depth of knowledge and innovative ideas pushed me to aim higher. Prof. Ni is supportive, and he encourages us to present our research outcomes on the world-class stage. I am particularly grateful that Prof. Ni offered me the invaluable opportunity to be enrolled in the research student attachment program to study abroad, which broadened my horizons. Prof. Ni is strict and hard-working, and he is always passionate about new technologies. Without his visionary ideas and support, I would not have been able to complete my research work. In addition to cutting-edge knowledge and technologies, the prudent attitude and self-motivated spirit of Prof. Ni are the most important qualities I have gained during my PhD study, and they will continue to inspire me in all my future endeavors. It has been a great honor to study with Prof. Ni.

I would like to express my appreciation to my co-supervisor, Dr. Fangxin Zou, and my team leader, Dr. Su-Mei Wang, for their guidance and kind help in every stage of my study, from developing the research ideas to organizing the research papers.

I would like to appreciate the funding support by the Hong Kong PhD Fellowship Scheme

of the RGC and the financial support from the National Rail Transit and Electrification and Automation Engineering Technology Center (Hong Kong Branch). I sincerely thank Mr. Tai-Tung Wai, Mr. Wing-Hong Kwan, Ms. Josephine Lui, Mr. Han-Zhang Lu, Ms. Karina Leung, Ms. Yiying Wang, and Ms. Qian Guo for their help and support.

I would also extend my sincere thanks to the funding support from the research student attachment program of The Hong Kong Polytechnic University and the valuable opportunity to study at University of California, Berkeley as a visiting student researcher. I am sincerely appreciated to my host supervisor at UC Berkely, Prof. Ziqi Wang, for offering me this chance and for his guidance. I have learned a lot from him. Also, my appreciation goes to Mr. Xuan Hu, Mr. Xiaolei Chu, and Mr. Guanren Zhou, for their company and help during my stay at Berkeley. I really miss the days at Berkeley.

My sincere gratitude also goes to Prof. Jingyao Zhang at Kyoto University for his kind help and strong support. Prof. Zhang has provided me with valuable suggestions on the method presented in Chapter 6. Also, sincere thanks to my master's supervisor Prof. Xingfei Yuan at Zhejiang University, for her warm encouragement during my study. Wish you all the best in your work and life.

I would like to express my thanks to Dr. Kuangyu Fei, Mr. Xinjie Shao, Mr. Cong Cai, Mr. Jianfeng Yang, Mr. Yangxiao Ye, Ms. Shu Li, Ms. Zhenni Wang, Ms. Qiqing Ge, Ms. Sisi Gu, Mr. Fangjie Nan, Dr. Tianyi Huang, Dr. Hao Guo, Dr. Yanling Deng, Mr. Lei Han, Dr. Chengyi Chu, Mr. Bin Xu, Dr. Ruhao Wang, Mr. Yongcan Dong, Mr. Yiqian Chen, Dr. Shuo Ma, Dr. Samy Akram, Dr. Yue Feng, Dr. Manyu Deng, Dr. Xingyu Wang, Dr. Panpan Jin, Mr. Bowen Zheng, Mr. Zehua Yan, Mr. Zihan Wan, Ms. Ping He, Mr. Jun Zhou, Dr. Xin Li, Dr.

Mingkun Dai, Mr. Hao Wang, Dr. Qi Zhu, Mr. You-Liang Zheng, Dr. Lei Yuan, Dr. Xin Ye, Dr. Yuan-Hao Wei, Mr. Yu-Ling Wang, Dr. Wai Kei Ao, Dr. Yan-Ke Tan, Dr. En-Ze Rui, Mr. Jia-Hao Lu, Dr. Wen-Qiang Liu, Mr. Sheng-Yuan Liu, Mr. Zhen Lin, Mr. Xiang-Xiong Li, Dr. Hong-Wei Li, Dr. Gao-Feng Jiang, Dr. Shuo Hao, Dr. Da-Zhi Dang, Dr. Zheng-Wei Chen, Dr. Siqi Ding, Dr. Si-Yi Chen, Dr. Guanghao Zhai, Mr. Yuan-Jiang Zeng, Mr. Guang-Zhi Zeng, Ms. Xin-Yue Xu, Ms. Pei-Lin Li, Mr. Gang Zeng, Ms. Xuan Zhao, Mr. Yi Xu, and other friends and colleagues for their friendship and support. Wish everyone a bright future.

Special thanks to Dr. En-Ze Rui, Dr. Shuo Hao, Dr. Lei Yuan, Dr. Xin Ye, Mr. Xiang-Xiong Li, and Ms. Shu Li for their friendship and company.

Lastly and most importantly, I would like to dedicate my most sincere gratitude to my parents. Their love and support are the most important thing to me. I sincerely wish you good health and happiness every day.

TABLE OF CONTENTS

CERTIFICATE OF ORIGINALITY I

ABSTRACT III

LIST OF PUBLICATIONS VI

ACKNOWLEDGEMENTS VII

TABLE OF CONTENTS X

LIST OF FIGURES XIV

LIST OF TABLES XIX

LIST OF ABBREVIATIONS XXI

CHAPTER 1 Introduction 1

1.1 Research Background 1

1.2 Research Motivations and Objectives 4

1.3 Thesis Outline 8

CHAPTER 2 Literature Review 11

2.1 Physics-Informed Machine Learning 11

2.1.1 Definition of PIML 12

2.1.2 Development of PIML 12

2.1.3 Applications of PIML in Engineering and Beyond 23

2.2 Physics-Informed Neural Network 30

2.2.1 Introduction to PINN 31

2.2.2 Scope of Application 43

2.2.3 Related Variants 46

2.3 Physics-Informed Graph Neural Network 48

2.3.1 Overview of PIGNN 48

2.3.2 Applications of PIGNN in Engineering 52

2.4 Research Gaps 53

CHAPTER 3 Physics-Informed Neural Network with Auxiliary Outputs for Reducing the Order of Governing Equations 57

3.1 Introduction 57

3.2 Vanilla PINN Framework 58

3.2.1 FCFNN and AD 58

3.2.2 Loss Function and Training of PINN 60

3.2.3 Training of Vanilla PINN for High-Order Differential Equations 64

3.3 Methods 73

3.3.1 Training with Auxiliary Outputs for Reducing the Order 74

3.3.2 Hyperparameter Study 76

3.4 Numerical Case Studies 79

3.4.1 Korteweg-de Vries Equation 80

3.4.2 Euler-Bernoulli Equation for Beam Vibration 85

3.4.3 A Sixth-Order Partial Differential Equation 88

3.5 Summary 91

CHAPTER 4 Physics-Informed Neural Network with Modulating Functions for Hard-

Embedding of Boundary Conditions.....92

4.1 Introduction.....92

4.2 Methods.....93

4.2.1 Establishment of Modulating Functions.....93

4.2.2 Training with Auxiliary Outputs.....98

4.3 Numerical Case Studies.....100

4.3.1 Variable Cross-Section Cantilever Beam.....105

4.3.2 Steel Frame.....115

4.4 Experimental Validation.....129

4.4.1 Experimental Setup.....129

4.4.2 Experimental Results.....133

4.4.3 Comparison of Results.....134

4.5 Summary.....136

CHAPTER 5 Two-Phase Physics-Informed Neural Network for Efficient Training and Accurate Solution.....138

5.1 Introduction.....138

5.2 Methods.....139

5.2.1 Establishment of Two-Phase PINN.....139

5.2.2 Loss Functions and Training in Two Phases.....144

5.2.3 Neural Network and Training Parameters.....148

5.3 Numerical Case Studies.....149

5.3.1 Results of Euler-Bernoulli Beam Example.....153

5.3.2 Results of Kirchhoff-Love Plate Example 157

5.4 Discussion 167

5.4.1 Collocation Points Sampling Methods 167

5.4.2 Types of Pretrained Neural Networks 173

5.4.3 Limitations 176

5.5 Summary 177

CHAPTER 6 Physics-Informed Graph Neural Network for Prestress Design of Tensegrity Structures 179

6.1 Introduction 179

6.2 Methods 182

6.2.1 Prestress Design of Tensegrity Structures Based on Force Density 182

6.2.2 Vanilla PINN for Prestress Design 186

6.2.3 Establishment of Physics-Informed Graph Neural Network (PIGNN) 188

6.3 Numerical Case Studies 192

6.3.1 Two-Dimensional Cases 192

6.3.2 Three-Dimensional Cases 196

6.4 Summary 203

CHAPTER 7 Conclusions and Recommendations 204

7.1 Conclusions 204

7.2 Recommendations 208

REFERENCES 211

LIST OF FIGURES

Figure 2-1 Overall structure of the literature review 11

Figure 2-2 An arbitrary two-dimensional domain 42

Figure 3-1 A FCFNN with three hidden layers 60

Figure 3-2 Forward and backward propagations 60

Figure 3-3 A diagram of vanilla PINN framework for forward problems 61

Figure 3-4 A diagram of vanilla PINN framework for inverse problems 63

Figure 3-5 The vanilla PINN framework with a single-layer FCFNN 66

Figure 3-6 Gradients with different values of w_{11} 68

Figure 3-7 Gradients with different values of w_{11} for the second-order differential equation 70

Figure 3-8 Gradients with different values of w_{11} for the neural network regression 70

Figure 3-9 A framework of training with auxiliary outputs for forward problems 74

Figure 3-10 The solution of the vanilla PINN for the KdV equation 83

Figure 3-11 The solution of the proposed framework for the KdV equation 84

Figure 3-12 The absolute error of the vanilla PINN for the KdV equation 84

Figure 3-13 The absolute error of the proposed framework for the KdV equation 84

Figure 3-14 The solution of the vanilla PINN for the Euler-Bernoulli equation 86

Figure 3-15 The solution of the proposed framework for the Euler-Bernoulli equation 87

Figure 3-16 The absolute error of the vanilla PINN for the Euler-Bernoulli equation 87

Figure 3-17 The absolute error of the proposed framework for the Euler-Bernoulli equation 88

Figure 3-18 The solution of the proposed framework for the sixth-order PDE 90

Figure 3-19 The absolute error of the proposed framework for the sixth-order PDE 90

Figure 4-1 A diagram of the improved PINN framework for inverse problems 99

Figure 4-2 Schematic of a variable cross-section cantilever beam with a semi-rigid joint 100

Figure 4-3 Single-bay steel frame with two semi-rigid joints 102

Figure 4-4 Local coordinates for structural members 102

Figure 4-5 The exact value and the vanilla PINN solution of the variable cross-section cantilever beam 108

Figure 4-6 Framework of a two-output FCFNN with hard constraints for inverse problems 113

Figure 4-7 The exact value and the vanilla PINN solution of the steel frame 118

Figure 4-8 The collocation points sampled on the beam by LHS, ST1, and ST2 127

Figure 4-9 The steel frame with semi-rigid joints 130

Figure 4-10 The semi-rigid joints connected by adjustable bolts 130

Figure 4-11 Digital micrometers for deformation measurement 131

Figure 4-12 Placement of the laser pointers 132

Figure 4-13 Measurement of rotational angles using laser pointers 133

Figure 4-14 Illustration for the measurement points and distances for calculating rotational angles 134

Figure 5-1 The vanilla PINN framework and the TP-PINN framework 142
Figure 5-2 Illustration of an Euler-Bernoulli beam 151
Figure 5-3 Illustration of a Kirchhoff-Love plate 152
Figure 5-4 Collocation points of the Euler-Bernoulli beam case 155
Figure 5-5 The PreNN for the fixed-end beam 155
Figure 5-6 The displacements of the fixed-end beam 156
Figure 5-7 The relative errors of vanilla PINN and TP-PINN solutions for the fixed-end beam 156
Figure 5-8 The PreNN for the simply-supported beam 156
Figure 5-9 The displacements of the simply-supported beam 157
Figure 5-10 The relative errors of vanilla PINN and TP-PINN solutions for the simply-supported beam 157
Figure 5-11 The FEM solution for the clamped plate 160
Figure 5-12 The collocation points of the PreNN 160
Figure 5-13 The PreNN for the clamped plate 161
Figure 5-14 The collocation points of the PriNN 161
Figure 5-15 The PINN solution for the clamped plate 162
Figure 5-16 The absolute error of PINN for the clamped plate 162
Figure 5-17 The TP-PINN solution for the clamped plate 163
Figure 5-18 The absolute error of TP-PINN for the clamped plate 163
Figure 5-19 The FEM solution for the simply-supported plate 164
Figure 5-20 The PreNN for the simply-supported plate 164

Figure 5-21 The PINN solution for the simply-supported plate 165
Figure 5-22 The absolute error of PINN for the simply-supported plate 165
Figure 5-23 The TP-PINN solution for the simply-supported plate 166
Figure 5-24 The absolute error of TP-PINN for the simply-supported plate 166
Figure 5-25 Collocation points using sampling method S1 169
Figure 5-26 The TP-PINN solution for sampling method S1 169
Figure 5-27 The absolute error of TP-PINN for sampling method S1 170
Figure 5-28 Collocation points using sampling method S2 170
Figure 5-29 The TP-PINN solution for sampling method S2 171
Figure 5-30 The absolute error of TP-PINN for sampling method S2 171
Figure 5-31 Collocation points using sampling method S3 172
Figure 5-32 The TP-PINN solution for sampling method S3 172
Figure 5-33 The absolute error of TP-PINN for sampling method S3 173
Figure 5-34 The PreNN form 1 174
Figure 5-35 The PreNN form 2 175
Figure 5-36 The displacements of the simply-supported beam based on PreNN form 2 175
Figure 5-37 The relative error between numerical solution and TP-PINN solution based on PreNN form 2 175
Figure 6-1 An illustrative two-dimensional tensegrity structure with four nodes and six elements 183
Figure 6-2 The vanilla PINN framework for prestress design of tensegrity structures 187

Figure 6-3 A GNN framework 189

Figure 6-4 The pooling operation 189

Figure 6-5 The PIGNN framework for prestress design of tensegrity structures 191

Figure 6-6 A five-unit Snelson's X-shape tensegrity arch with regular geometry 193

Figure 6-7 A five-unit Snelson's X-shape tensegrity arch with irregular geometry 194

Figure 6-8 The perspective view of the 3D unit 198

Figure 6-9 The top view of the 3D unit 198

Figure 6-10 The perspective view of the 3D tensegrity structure with regular geometry 199

Figure 6-11 The top view of the 3D tensegrity structure with regular geometry 199

Figure 6-12 The feasible prestress distribution of the 3D tensegrity structure with regular geometry 200

Figure 6-13 The perspective view of the 3D tensegrity structure with irregular geometry 201

Figure 6-14 The top view of the 3D tensegrity structure with irregular geometry 202

Figure 6-15 The feasible prestress distribution of the 3D tensegrity structure with irregular geometry 202

LIST OF TABLES

Table 3-1 Relative L_2 errors of different auxiliary output schemes 77

Table 3-2 Recommended auxiliary output scheme for PDEs with different orders 79

Table 4-1 Comparison of the vanilla PINN solution and exact solution for the forward problem of the variable cross-section cantilever beam 107

Table 4-2 Comparison of the vanilla PINN solution and exact solution for the inverse problem of the variable cross-section cantilever beam 110

Table 4-3 Comparison of the relative L_2 errors with different numbers of collocation points 111

Table 4-4 Comparison of the vanilla PINN with hard constraints solution and exact solution for the inverse problem of the variable cross-section cantilever beam 112

Table 4-5 Comparison of the improved PINN solution and exact values of structural response for the inverse problem of the non-prismatic cantilever beam 115

Table 4-6 Comparison of the vanilla PINN solution and exact solution for the forward problem of the steel frame 118

Table 4-7 Comparison of the vanilla PINN solution and exact solution for the inverse problem of the steel frame 121

Table 4-8 Comparison of the vanilla PINN with hard constraints solution and exact solution for the inverse problem of the steel frame 123

Table 4-9 Comparison of the improved PINN solution and exact values for the inverse

problem of the frame structure.....126

Table 4-10 Relative errors of α_1 and α_2 under different noise levels.....128

Table 4-11 Displacements on measurement points under different loadings (mm).....134

Table 4-12 Comparison of the rotational angles identified by the vanilla PINN and measurement data.....135

Table 4-13 Comparison of the rotational angles identified by the improved PINN and measurement data.....136

Table 5-1 Relative L_2 error and training time of vanilla PINN and TP-PINN for beam cases.....155

Table 5-2 Relative L_2 error and training time of vanilla PINN and TP-PINN for plate cases.....159

Table 5-3 Relative L_2 error and training time of TP-PINN with different sampling methods for the simply supported plate case.....168

Table 6-1 The feasible force densities of the regular tensegrity arch predicted by vanilla PINN and PIGNN.....193

Table 6-2 Nodal coordinates of the 2D tensegrity arch with irregular geometry.....195

Table 6-3 The feasible force densities of the irregular tensegrity arch predicted by vanilla PINN and PIGNN.....196

LIST OF ABBREVIATIONS

AD	Automatic differentiation
AI	Artificial intelligence
ANN	Artificial neural network
ANODE	Augmented neural ordinary differential equation
A-PINN	Auxiliary physics-informed neural network
B-PINN	Bayesian physics-informed neural network
BVP	Boundary value problem
CFD	Computational fluid dynamics
CNN	Convolutional neural network
DCNODE	Data-controlled neural ordinary differential equation
DL	Deep learning
DNN	Deep neural network
DOF	Degree of freedom
ELM	Extreme learning machine
EVD	Eigenvalue decomposition
FCFNN	Fully-connected feedforward neural network
FDM	Finite difference method
FE	Finite element
FEA	Finite element analysis

FEM	Finite element method
FNO	Fourier neural operator
FODE	Fractional-order differential equation
fPINN	Fractional physics-informed neural network
GCN	Graph convolutional network
GNN	Graph neural network
GNODE	Graph neural ordinary differential equation
GPU	Graphics processing unit
HGNN	Hamiltonian graph neural network
HMC	Hamiltonian Monte Carlo
HPC	High-performance computing
hp-VPINN	hp-variational physics-informed neural network
IDE	Integro-differential equation
IS	Importance sampling
IVP	Initial value problem
KdV	Korteweg-de Vries
L-BFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LES	Large eddy simulation
LGNN	Lagrangian graph neural network
LHS	Latin hypercube sampling
LSTM	Long short-term memory
ML	Machine learning

MLP	Multilayer perceptron
MM	Metamaterial
MSE	Mean square error
NN	Neural network
NO	Neural operator
NODE	Neural ordinary differential equation
NS	Navier-Stokes
ODE	Ordinary differential equation
PDE	Partial differential equation
PhyCRNet	Physics-informed convolutional-recurrent network
PIGNN	Physics-informed graph neural network
PIML	Physics-informed machine learning
PINN	Physics-informed neural network
PINO	Physics-informed neural operator
PINODE	Physics-informed neural ordinary differential equation
PPNN	PDE-preserved neural network
PreNN	Pretrained neural network
PriNN	Primary neural network
RANS	Reynolds-averaged Navier-Stokes
ReLU	Rectified linear unit
ResNet	Residual network
RNN	Recurrent neural network

SciML	Scientific machine learning
SGD	Stochastic gradient descent
SHM	Structural health monitoring
SVD	Singular value decomposition
Tanh	Hyperbolic tangent
TP-PINN	Two-phase physics-informed neural network
TL	Transfer learning
VPINN	Variational physics-informed neural network

CHAPTER 1

Introduction

1.1 Research Background

The rapid advancements in machine learning (ML) and deep learning (DL) have shown great potential in solving complex problems across various domains, including engineering, healthcare, finance, and more. In the field of structural engineering, these techniques have been particularly useful for understanding and predicting the behavior of structural systems. In the prediction of structural behavior and the identification of structural parameters, numerical methods such as the finite element method (FEM) are often employed. Traditional approaches face challenges such as requirements of high-quality data, high computational complexity and resource demands, and limited ability to handle nonlinearity, especially in large-scale structures and complex models. To address these challenges, researchers have turned to ML approaches, which can learn intricate relationships and make predictions from measurement data. However, structural response data is often difficult to measure or too expensive to obtain. Meanwhile, traditional data-driven models often struggle with incorporating physics-based knowledge, leading to limitations in accuracy, interpretability, and computational efficiency. This has given rise to the development of scientific machine learning (SciML) techniques, such as physics-

informed machine learning (PIML), which aims to integrate fundamental laws of physics into ML models, thereby improving their performance and interpretability, and overcoming the dilemma posed by limited measurement data.

PIML (Karniadakis et al. 2021) is a rapidly growing interdisciplinary field that combines the strengths of physics-based modeling and data-driven ML to address complex problems in science and engineering. Physics-based models, while grounded in well-established scientific principles, often struggle with computational complexity, scalability, and adaptability, particularly in the presence of uncertainties, nonlinearities, or high-dimensional parameter spaces. On the other hand, purely data-driven ML models can be prone to overfitting, lack of interpretability, and poor generalization, especially when dealing with limited, noisy, or heterogeneous data. PIML was introduced as a solution to bridge the gap between these two approaches by leveraging the power of ML while incorporating underlying physical laws and previously derived principles.

PIML has evolved through the integration of ML techniques such as artificial neural networks (ANNs) (Alhajeri et al. 2022) and Bayesian inference (Yang et al. 2021) with physical knowledge. The primary goal is to develop accurate, robust, and interpretable ML models that can learn underlying patterns and relationships from limited data while respecting the physical laws governing the system. This is achieved by incorporating physical information into the neural network (NN) architecture, training process, or loss function of ML models.

Over the years, various PIML techniques have been developed, such as physics-informed neural network (PINN) (Raissi et al. 2019a), physics-informed neural ordinary differential

equation (PINODE) (Lai et al. 2021), physics-informed neural operator (PINO) (Li et al. 2021b), and physics-informed graph neural network (PIGNN) (Shukla et al. 2022). These PIML techniques have found applications in numerous scientific and engineering fields. In structural engineering, PIML has been utilized for structural parameter identification (Rojas et al. 2021), damage detection (Xu et al. 2023), and topology optimization (Jeong et al. 2023). In fluid dynamics, PIML is employed to model and predict fluid flow, turbulence, and transport phenomena in complex geometries and boundary conditions (Cai et al. 2021a). In materials science, it contributes to the discovery, design, and characterization of new materials with tailored properties (Zhang and Gu 2021; Chen et al. 2020a). In geophysics and earth sciences, PIML can be applied to model and predict the behavior of geological, hydrological, or atmospheric systems (Ahmmed et al. 2022). In biological and medical sciences, it is used to model and understand intricate processes and interactions, such as protein folding, cellular signaling, and tumor growth (Alber et al. 2019).

Despite its promise, there is still room for improvement in several areas of PIML. For example, improving the computational efficiency and robustness of PIML algorithms when solving high-dimensional problems or complex systems with numerous boundary conditions is a pressing need. Also, it is crucial to develop more general and flexible techniques for integrating physical principles and information with ML, as this will broaden its applicability across various problems. Therefore, this thesis focuses on improving PIML from the following four directions: First, reducing the order of the governing equation by decomposing higher-order differential equations into a series of second- or third-order differential equations to

facilitate computation; Second, embedding boundary conditions in a hard manner to enhance the prediction accuracy especially near the boundary; Third, decoupling the training process for satisfying boundary conditions to alleviate the difficulty in multi-objective optimization; Fourth, introducing graph neural network (GNN) to PIML as PIGNN to improve the flexibility and training efficiency by leveraging more physical information. Also, the proposed methods will be applied to solve different kinds of structural problems.

1.2 Research Motivations and Objectives

PIML has emerged as a transformative approach that combines the strengths of data-driven ML models with physical knowledge, offering significant potential for solving complex problems in engineering. Despite its promise, several critical challenges persist in applying PIML methods. These challenges stem from the inherent complexity and computational burden of large-scale, high-dimensional problems and from the difficulty of ensuring interpretability, robustness, and generalizability of the PIML models. This thesis seeks to address these challenges by developing novel PIML methods that enhance their accuracy, efficiency, interpretability, and scalability, ultimately contributing to their applications in structural engineering. The improvements in PIML presented in this thesis are motivated by the following four aspects:

First, structural engineering problems often involve complex, high-order differential equations that describe the behavior of structures under various loads and conditions, such as the fourth-order Euler-Bernoulli beam equation and Kirchhoff-Love plate equation. Traditional

PIML methods may struggle with these high-order equations due to the errors that accumulated from repeated automatic differentiation (AD) functions and gradient vanishing, gradient exploding problems. High-order differential terms can lead to inaccurate computation, unstable solution, slow convergence, and even training failure in NNs.

Second, some problems described by differential equations involve many boundary conditions, and the effective incorporation of these boundary conditions is a primary challenge in PIML. In the vanilla PINN framework, these constraints are often embedded as soft penalty terms in the loss function, meaning that their enforcement depends on the training process. Boundary conditions cannot be strictly satisfied because the penalty term can only be reduced toward zero. This treatment can lead to inaccurate predictions near the boundaries and critical regions, especially when faced with complex, high-dimensional problems.

Third, in typical scenarios, multiple penalty terms in the loss function represent residuals of the governing equation, boundary conditions, initial conditions, and observed data, resulting in a multi-objective optimization problem during training. Balancing these penalty terms remains crucial, as poor weighting can lead to slow convergence, excessive computational cost, and low accuracy.

Lastly, traditional PIML models, such as PINN, often treat the physical system as a continuous domain without explicitly considering the topological relationships between elements of the system. For example, structural systems, such as trusses, frames, and cable-strut structures, are inherently composed of interconnected structural elements. Their elements and connectivity can be naturally represented as graphs. However, vital structural information,

including structural geometries and topologies, is often ignored when constructing the vanilla PINN framework.

In summary, this thesis is motivated by the need to address current challenges in PIML frameworks and the applications of PIML in structural engineering, with a particular focus on parameter identification, response prediction, and design.

This thesis aims to advance the state-of-the-art in PIML in terms of accuracy, efficiency, interpretability, and scalability, and provide more intelligent, reliable, and effective tools for tackling complex problems in structural engineering. To this end, the specific objectives are summarized as follows:

1. Conduct a comprehensive review of state-of-the-art PIML methods. This objective surveys the latest developments in PIML frameworks, emphasizing PINN and PIGNN, and their applications in solving engineering problems. The review covers detailed introductions to the development, implementation, and computational challenges of these methods, followed by a critical assessment of their strengths, weaknesses, and areas for improvement.
2. Develop a novel PINN framework with auxiliary outputs for reducing the order of governing equations. This objective aims to address the computational difficulties associated with high-order governing equations in vanilla PINN. This method introduces auxiliary outputs representing lower-order derivatives of the original output, is integrated into the vanilla PINN to simplify the governing equations. With this method, high-order governing equations are decomposed into a series of lower-order

differential equations, thus reducing the computational burden and improving the stability of the training process.

3. Enhance prediction accuracy by handling boundary conditions in a hard manner within the proposed PINN framework. Modulating functions will be derived for various boundary conditions to transform the original outputs of the NN. By transforming the outputs to strictly satisfy boundary conditions, associated penalty terms are no longer needed. This approach ensures automatic, exact satisfaction of boundary conditions and yields more accurate, reliable predictions, especially near the boundaries.
4. Extend the hard manner treatment of boundary conditions to problems with complex domains and optimize the multi-objective training process. Under this objective, a novel framework termed two-phase physics-informed neural network (TP-PINN) is proposed to decouple the entire training process into two phases. Boundary conditions are satisfied by pretrained NNs in the first phase, while the auxiliary output-based method is adopted in the second phase to reduce computational burden, improving both efficiency and accuracy. In addition, this two-phase design mitigates the challenges of multi-objective optimization.
5. Integrate GNN into the PIML framework to leverage structural topology. This objective aims to develop a novel PIGNN framework by introducing GNN to replace the commonly used fully-connected feedforward neural network (FCFNN) in PINN. By leveraging inherent graph-like properties of certain structures, additional structural information such as structural topology can be naturally incorporated in PIGNN, thus

enhancing the model's ability to capture interactions and dependencies within complex structures.

6. Implement and validate the proposed PIML frameworks through numerical and experimental case studies involving diverse structural systems. The performance of these frameworks will be evaluated in terms of accuracy, computational efficiency, and robustness to noisy data. These results will demonstrate their applicability and effectiveness for a broad range of structural engineering problems.

1.3 Thesis Outline

This thesis is structured into seven chapters, covering various aspects of developing and advancing PIML frameworks for structural engineering applications. The vanilla PINN is selected as the baseline model, as it represents the most fundamental and widely adopted framework in PIML. Using this benchmark allows for a clear and fair comparison with the proposed frameworks, highlighting their effectiveness in addressing the limitations of the vanilla PINN. The outline of the thesis is as follows:

Chapter 1 provides an overview of the research background, motivation, and objectives, emphasizing the significance of integrating ML techniques with physical knowledge to tackle complex problems. The key challenges in PIML are summarized as the motivation for this work. Finally, the objectives are clearly defined: to develop novel PIML methods that enhance model performance, scalability, and applicability in structural engineering applications.

Chapter 2 presents a comprehensive literature review of the state-of-the-art in PIML,

focusing on PINN, PIGNN, and related techniques. This chapter examines the developments and practical applications of these methods, highlighting their strengths and limitations when applied to engineering problems. Based on the literature review, research gaps in PIML methods are identified, motivating the enhancements proposed in subsequent chapters.

Chapter 3 begins with the basics of the vanilla PINN. It then analyzes the underlying reasons for the training failure of vanilla PINN on high-order differential equations from the perspective of training dynamics. Then, this chapter introduces a novel method of auxiliary outputs for reducing the highest order of governing equations. Lastly, numerical case studies of three high-order differential equations are performed to validate the effectiveness of the proposed method.

Chapter 4 further enhances the proposed framework by introducing the concept of embedding boundary conditions in a hard manner using modulating functions. This chapter first details the development of modulating functions for various boundary conditions. Then, the auxiliary output-based method is also integrated to facilitate computation. After that, numerical case studies of structural response prediction and unknown parameter identification are carried out to demonstrate the performance of the proposed method. Finally, experimental validation is conducted to illustrate the practical application of the proposed method.

Chapter 5 proposes the TP-PINN framework, which decouples the training process into two distinct phases. This chapter begins with detailed introduction to the fundamental principles, implementation approach, and training procedures of the TP-PINN method. Then, two numerical case studies on structural analysis problems are presented to demonstrate the

performance of TP-PINN compared to vanilla PINN. Finally, a comprehensive discussion of TP-PINN, including its effectiveness and limitations, is provided.

Chapter 6 introduces PIGNN as an innovative approach for incorporating structural topology into PIML models. The fundamentals of prestress design of tensegrity structures are presented first, followed by the development of PIGNN. A comparison between vanilla PINN and PIGNN is presented through several case studies, demonstrating PIGNN's superior performance in this task.

Chapter 7 presents the conclusions and recommendations for this thesis. It summarizes the key findings and contributions and outlines directions for future research.

CHAPTER 2

Literature Review

2.1 Physics-Informed Machine Learning

This chapter presents a comprehensive literature review of PIML. The structure of this chapter is illustrated in Figure 2-1, and the contents are organized as follows. First, the fundamentals of PIML (Block 1 in Figure 2-1), including its definition, development, and applications, are reviewed. Second, as a well-established PIML framework, PINN (Block 2) is reviewed in detail, including its implementation, application scope, and related variants. Third, the promising PIGNN framework (Block 3) is briefly introduced. Finally, the research gaps (Block 4) are summarized.

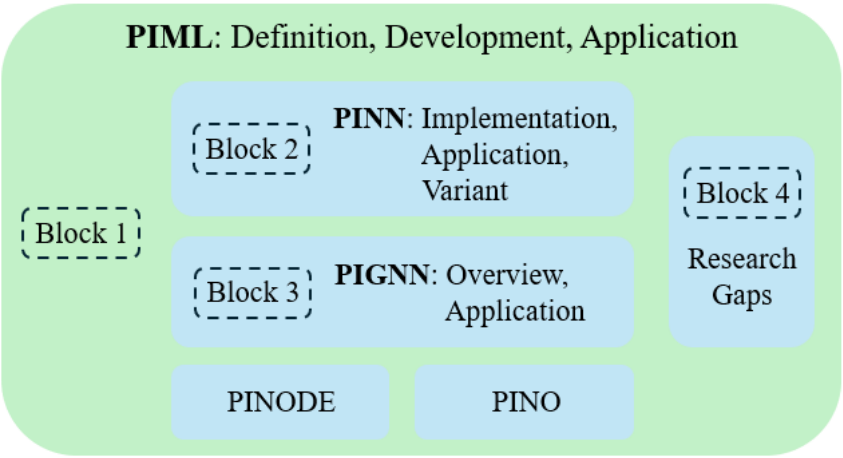


Figure 2-1 Overall structure of the literature review

2.1.1 Definition of PIML

ML can be broadly defined as a field of study that gives computers the ability to learn without being explicitly programmed and to make predictions based on data (Samuel 1959). Building on this, Jordan and Mitchell (2015) describe “learning” as the transformation of information. ML, therefore, refers to automated learning processes that convert training data and prior knowledge into expertise used for performing tasks or making predictions.

PIML (Karniadakis et al. 2021) is an emerging discipline within data science and artificial intelligence (AI), distinct in its explicit integration of physical knowledge into the learning process. While traditional ML relies solely on data, often leading to a “black box” model that lacks interpretability, PIML incorporates physical laws and domain-specific knowledge directly into the model. This approach enhances the robustness, interpretability, and reliability of the results, especially in areas such as fluid mechanics, structural engineering, and materials science, where predictions should align with the constraints of physical reality.

2.1.2 Development of PIML

PIML is developed to overcome the limitations of traditional data-driven ML methods, which may perform well when extensive training data is available but often struggle when data is scarce or when the problem requires adherence to known physical constraints (Karniadakis et al. 2021; Cuomo et al. 2022). By embedding physical principles directly into ML models, PIML addresses these issues, thereby improving accuracy and providing solutions consistent with physical laws.

Emergence and Evolution of PIML

Traditional ML excels at pattern recognition and making predictions from large datasets, but these models often lack the ability to incorporate the physical laws governing the systems (Willard et al. 2022). This can lead to accurate predictions within a dataset but inconsistent results when applied to new scenarios. For instance, in fluid dynamics, purely data-driven models may capture turbulence patterns accurately during training but fail to generalize under different flow conditions, as they do not consider the underlying Navier-Stokes (NS) equations that govern fluid behavior. Therefore, PIML is proposed to overcome these shortcomings by embedding physical knowledge directly into the learning process, ensuring that models respect the governing laws, even in the absence of extensive training data.

The emergence of PIML was facilitated by several critical factors. For example, advances in computer hardware such as graphics processing unit (GPU) and software frameworks like TensorFlow (Abadi et al. 2016) and PyTorch (Paszke et al. 2017; Paszke et al. 2019) also made significant contributions, while the development of AD (Rumelhart et al. 1986; Baydin et al. 2018) allowed for the efficient computation of gradients.

From the historical perspective, the increasing complexity of scientific problems drove the need for methods that could combine physical laws with data-driven models. Traditional numerical solvers, while effective, faced significant computational bottlenecks when dealing with high-dimensional, nonlinear, or multi-scale problems. Large-scale simulations in climate modeling, materials science, or computational fluid dynamics (CFD) often became computationally prohibitive using only classical numerical techniques. Therefore, researchers are looking for ML-based solutions to handle complex problems (Ling et al. 2016).

The evolution of PIML is characterized by the development of various methods aimed at embedding physics into ML models more effectively. Early approaches, such as the deep Ritz method (E and Yu 2018) and the deep Galerkin method (Sirignano and Spiliopoulos 2018), were designed to solve variational and differential problems by approximating the solutions of governing equations using NNs. These pioneering methods paved the way for more sophisticated frameworks, such as PINN. In addition, PINODE (Lai et al. 2021) further extended PIML's capabilities to dynamic systems with continuous time evolution.

Foundation of PIML

PIML is built upon several core foundations. The development of NNs, including architectures like multilayer perceptron (MLP) (Gardner and Dorling 1998; Popescu et al. 1998), convolutional neural networks (CNNs) (O'Shea and Nash 2015), and recurrent neural networks (RNNs) (Salehinejad et al. 2018), has provided the building blocks for PIML. These DL architectures enable the modeling of complex, high-dimensional relationships, which is crucial for accurately capturing the dynamics of physical systems.

The evolution of programming languages and ML libraries has also been instrumental in the development of PIML. Python, with its ease of use and extensive community support, has become the primary language for implementing DL models. Libraries such as TensorFlow and PyTorch have further simplified the process of developing and training ML models. The availability of these tools has lowered the barrier to entry for researchers, enabling the rapid prototyping and testing of PIML models.

Another key foundation of PIML is the development of optimization algorithms that allow

for efficient training of large-scale ML models. Methods such as stochastic gradient descent (SGD) (Amari 1993), Adam (Kingma and Ba 2017), and limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) (Byrd et al. 1995) have proven essential for training PIML models, enabling them to converge efficiently even when dealing with high-dimensional systems. Cuomo et al. (2022) provide a detailed exploration of how these optimization techniques have improved the efficiency and scalability of ML, making it possible to apply these models to scientific problems that were previously unsolvable.

The development of PIML has also been significantly influenced by advancements in computational hardware. The advancements of GPUs and other hardware accelerators have drastically reduced the training time of NNs, making it feasible to apply PIML models to large-scale scientific problems. High-performance computing (HPC) resources have enabled the parallelized training of complex models (Owens et al. 2008; Nickolls et al. 2010), allowing researchers to tackle problems that were previously computationally intractable.

Embed physics into ML

Embedding physical principles into ML models is crucial for ensuring that the resulting predictions are physically consistent and reliable. In PIML, there are three primary paths, observational biases, inductive biases, and learning biases, that can be utilized independently or in combination to guide the learning process and enhance model generalization.

Observational biases involve incorporating data that already reflects the underlying physical principles. This can be achieved by using datasets generated through experiments or simulations that embody physical laws, as well as through data augmentation techniques that

incorporate domain-specific knowledge (Yang and Perdikaris 2019a; Li et al. 2021a; Kashefi et al. 2021).

Inductive biases are introduced by modifying the ML model's architecture to explicitly enforce physical constraints. This could involve tailoring the model structure so that the predictions implicitly satisfy given physical laws, which are typically represented as mathematical constraints. For example, incorporating symmetries directly into the NN architecture ensures that the model inherently respects these properties (Bronstein et al. 2017; Cohen et al. 2019; Mattheakis et al. 2020). While inductive biases are a principled way to embed physical knowledge, they are often limited by the complexity of physical laws and may require specialized implementations (Pfau et al. 2020).

Learning biases involve introducing physical constraints during the training process, often by modifying the loss function to include penalty terms representing physical laws. These constraints can be implemented as soft penalties, which guide the model to favor solutions that approximately satisfy the underlying physical equations (Lagaris et al. 1998; Raissi et al. 2019a). This approach offers a flexible framework for incorporating a broad range of physical laws, which are expressed in integral, differential, or fractional form (Raissi et al. 2019a; Lu et al. 2021b; Pang et al. 2019). Although learning biases may not guarantee strict adherence to physical laws, they provide an adaptable way to ensure that the model's predictions align with physical expectations while maintaining the flexibility to handle diverse scenarios.

By embedding physics into the ML process through observational, inductive, and learning biases, PIML models are well equipped to generalize and provide physically meaningful predictions, even in cases where traditional ML methods may fail due to the lack of data.

Physics-Informed Neural Network

PINN (Raissi et al. 2019a) is a class of PIML framework that integrates governing equations, such as partial differential equations (PDEs), directly into the ML framework. PINN is trained by minimizing a loss function that combines the residuals of governing PDEs, initial and boundary conditions, and observed data. This allows PINN to approximate the solutions to governing PDEs. Unlike traditional supervised ML methods that require labeled training data, PINN operates in an unsupervised manner by converting the solution of governing equations into a problem of loss function optimization.

The concept of embedding physical knowledge in ML models is not entirely new. Early work by researchers such as Dissanayake and Phan-Thien (1994) employed simple NNs to approximate PDE solutions, laying the groundwork for more sophisticated methods like PINN. Building on these foundational ideas, Raissi et al. (2017a; 2017b) introduced Gaussian process regression to represent operator functionals and infer solutions with uncertainty estimates, which was extended to the development of PINN.

PINN offers several advantages over traditional numerical methods like FEM and finite difference method (FDM). It provides a flexible framework for solving both forward and inverse problems using the same NN architecture. Moreover, solutions obtained through PINN are differentiable, enabling easy computation of analytical gradients, which is often beneficial for further analysis and optimization. PINN is advantageous in domains with complex geometries or in high-dimensional settings where traditional numerical solvers face significant challenges. Despite its promise, there remain areas for further research, such as improving optimization techniques, exploring new activation functions, and addressing theoretical

questions to enhance the robustness and efficiency of PINN.

The detailed implementation and applications of PINN will be further elaborated in Section 2.2.

Physics-Informed Neural Ordinary Differential Equation

Neural ordinary differential equation (NODE) (Chen et al. 2018) represents a class of data-driven ML models that leverages ordinary differential equations (ODEs) to describe the evolution of hidden states within a system. NODE has shown significant promise in modeling complex dynamic systems, especially in continuous-time and continuous-depth modeling tasks. NODE originated from the residual networks (ResNets) (He et al. 2016), which can be thought of as approximating differential equations in a discrete manner. In ResNets, the desired mapping between states is expressed in terms of a residual function, allowing the evolution of hidden states to be learned iteratively. As the number of layers in the network becomes very large and the step size approaches zero, the relationship between hidden states can be interpreted as a continuous-time model governed by an ODE.

The fundamental architecture of NODE involves representing the evolution of hidden states as a continuous dynamical system parameterized by a NN, which can be described by:

$$\frac{dh(t)}{dt} = f(h_t, t, \theta) \qquad (2-1)$$

where $h(t)$ represents the hidden state at time *t*, and $f(\cdot)$ is a NN parameterized by weights θ. The input and output of a NODE are related through an initial value problem (IVP), where the initial state $h(0)$ represents the input, and the later state $h(T)$ is the output. To solve the IVP, an ODE solver such as Runge-Kutta methods is used to integrate the dynamics of the

system over time.

With the development of NODE, several variants have been proposed to address specific limitations and extend their applicability. Augmented NODE (ANODE) (Dupont et al. 2019) is a prominent variant designed to overcome representational limitations of standard NODE. Another notable variant is the data-controlled NODE (DCNODE) (Massaroli et al. 2020), which improves NODE's performance by embedding input data directly into the vector field.

To further improve the versatility and applicability of NODE, PINODE was proposed (Lai et al. 2021; Lai et al. 2022). PINODE incorporates prior physical knowledge into the NODE framework, making it more applicable to real-world problems. Unlike standard data-driven NODE, which may struggle with high-dimensional or complex systems, PINODE leverages physical laws and domain knowledge to constrain the model and improve generalization. For instance, PINODE can be used in structural health monitoring (SHM) tasks, such as model updating and damage detection.

PINODE has demonstrated success across various applications. For example, in the context of stochastic dynamical systems, the stochastic PINODE (O'Leary et al. 2022) has been developed to handle stochastic differential equations by predicting moment trajectories and aligning them with empirical data. Another application is the integration of attention mechanisms into NODE, as demonstrated by the attentive dual co-evolving NODE (Jhin et al. 2021). The incorporation of attention mechanisms can enhance the ability of the model to focus on relevant aspects of the input data, improving the model's performance in scenarios with complex dependencies.

In conclusion, PINODE represents an important framework of PIML, bridging the gap

between purely data-driven models and those grounded in physical principles. By incorporating domain-specific knowledge, PINODEs provide improved generalization, robustness, and interpretability, making them well-suited for a wide range of scientific and engineering problems.

Physics-Informed Neural Operator

PINO represents a hybrid paradigm that combines data-driven operator learning with physics-constrained optimization to learn solution operators for families of parameterized PDEs (Li et al. 2021b). Based on the universal approximation properties of the Fourier neural operator (FNO) framework, PINO augments supervised learning on coarse-resolution datasets with high-resolution PDE residuals enforced via AD, thereby overcoming the optimization challenges and data scarcity issues inherent in the traditional PINN and purely data-driven neural operators (NOs).

In the PINO methodology, NO parameterizes the mapping from input functions, such as initial or boundary conditions, to solution fields, while physics-based losses computed at finer discretization ensure strict adherence to governing equations. This multiresolution training regime enables PINO not only to match ground-truth operators where data are available but also to perform zero-shot super-resolution, accurately predicting solutions on meshes orders of magnitude finer than those seen during training (Li et al. 2021b). Importantly, PINO succeeds even in purely physics-constrained settings, where no labeled data exist, thanks to the stability and expressivity of its operator learning backbone.

PINO attains superior accuracy and convergence rates. For canonical problems, including

Kolmogorov flows and nonlinear Burgers equations, PINO exhibits markedly low residual errors and fast training dynamics, while retaining conservation properties and facilitating efficient inference (Maust et al. 2022). Variants of PINO have been developed for coupled forward-backward PDE systems in mean field games and for closure-model-free learning of chaotic dynamics, achieving significant speedups and robust long-term predictions (Wang et al. 2024; Chen et al. 2023).

Recent advances explore self-training strategies that iteratively refine PINO predictions with pseudo-labels to bridge remaining performance gaps when training exclusively with physics loss, further enhancing data efficiency and model robustness (Majumdar et al. 2023). By embedding physical laws into a discretization-free NO framework, PINO offers a powerful and generalizable approach for solving forward and inverse problems across scientific and engineering domains, positioning itself as a cornerstone of modern PIML.

Physics-Informed Graph Neural Network

PIGNN integrates the topological modeling capabilities of GNN with the physical constraints of PINN to solve PDE-governed problems on arbitrary, unstructured domains (Shukla et al. 2022; Peng et al. 2022). By representing spatial discretization as graph nodes and connectivity as edges, PIGNN naturally accommodates irregular meshes and complex geometries that challenge traditional grid-based approaches. The fusion of graph inductive biases with physics-based losses enhances both generalization to unseen configurations and interpretability of the learned surrogates.

The training objective of PIGNN combines data-driven supervision with physics-

informed residuals computed at graph nodes via AD or graph exterior calculus, ensuring strict enforcement of PDE constraints and boundary conditions without requiring dense labeled data (Zhang et al. 2025a; Chenaud et al. 2023). This approach decouples spatial connectivity from the underlying physics, yielding robust surrogate models suitable for both forward simulations and inverse parameter identification.

Benchmark studies demonstrate that PIGNN trained on small-scale or simply meshed examples generalizes zero-shot to larger, more complex systems (Ramos-Osuna et al. 2025; Thangamuthu et al. 2022). Such scalability addresses the in-sample limitations of earlier PIML methods and opens the door to efficient large-scale modeling.

PIGNN has been successfully applied across engineering and scientific domains, including urban wind-field forecasting on unstructured meshes (Liu and Pyrcz 2023), nonlinear fluid dynamics and elasticity modeling in complex geometries (Peng et al. 2023), and inverse material-property estimation using sparse sensor networks (Di Lorenzo et al. 2024). Their ability to embed prior physical knowledge within graph representations reduces data requirements and enhances solution fidelity.

By embedding physical constraints into graph-based architectures, PIGNN bridges data-driven learning and classical numerical solvers, paving the way for efficient, scalable simulations and inverse designs on real-world, unstructured domains (Shi et al. 2025; Goswami et al. 2023). Their growing adoption promises transformative advances in multidisciplinary modeling, from fluid flows and structural mechanics to astrophysical and biomedical systems.

As a promising framework in the PIML family, PIGNN will be further introduced in Section 2.3.

2.1.3 Applications of PIML in Engineering and Beyond

PIML has found widespread application across various fields of engineering and beyond. This section explores several key applications of PIML, emphasizing its use in solving differential equations in engineering, as well as its specific applications in fluid mechanics, solid mechanics, and materials science.

Differential Equations

PIML is built to solve engineering problems that can be described by differential equations, such as ODEs, PDEs, fractional-order differential equations (FODEs), as well as integro-differential equations (IDEs). Before diving into specific applications, different types of differential equations governing various engineering problems that can be dealt with PIML will be reviewed and summarized.

Ordinary Differential Equations

ODEs are fundamental tools for modeling complex nonlinear systems, particularly those are difficult to represent using purely physics-based models (Lai et al. 2021). A typical ODE system is expressed as:

$$\frac{du(x,t)}{dt} = f(u(x,t),t) \qquad (2-2)$$

where initial or boundary conditions can be specified, leading to either an IVP or a boundary value problem (BVP). PINODE approaches have been employed in various engineering applications to solve ODEs effectively. For example, Lai et al. (2021) applied PINODE to the structural identification of a spring-mass system modeled as a four-degree-of-freedom dynamical system with cubic nonlinearity, using both simulated and noisy experimental data

to learn the governing dynamics of structures equipped with negative stiffness devices.

Zhang et al. (2020b) employed a deep long short-term memory (LSTM) network within the PINN framework to solve nonlinear structural systems subjected to seismic excitation. Examples include steel moment-resisting frames and the single-degree-of-freedom Bouc-Wen model, which represents a nonlinear system with rate-dependent hysteresis. Their work aimed at addressing the challenges of modeling nonlinear equations of motion:

$$\ddot{u} + g = -\Gamma a_g \tag{2-3}$$

where $g(t) = M^{-1}h(t)$ represents the mass-normalized restoring force, with M being the mass matrix and h the total nonlinear restoring force.

The use of directed graph models for implementing ODEs as deep neural networks (DNNs) has also been explored (Viana et al. 2021), leveraging techniques such as Euler RNNs for numerical integration. Nascimento et al. (2020) provided a tutorial on how to implement ODE integration using Python and RNNs, offering practical guidance for researchers.

PIML has also been applied to one-dimensional mechanics problems involving ODEs, such as the deformation of beams under various loading conditions. These problems are often described by the Euler-Bernoulli equation:

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 u}{\partial x^2}\right) = q \tag{2-4}$$

PIML approaches, such as PINN, provide a means to solve these ODEs. However, these high-order ODEs require multiple gradient computations within the PIML framework, which can hinder computational efficiency and accuracy.

Partial Differential Equations

PDEs are essential for modeling a wide range of physical phenomena in engineering, including heat transfer, fluid flow, and elasticity. Traditional numerical approaches like FEM and FDM have been widely used to solve PDEs. However, these methods can be computationally intensive, especially for high-dimensional problems or those involving complex geometries. PINN presents an efficient alternative by incorporating the underlying physics directly into the ML framework.

Kharazmi et al. (2019; 2021b) addressed general steady-state PDEs represented by:

$$F_s(u(x); q) = f(x) \quad x \in \Omega \tag{2-5}$$

$$B\big(u(x)\big) = 0 \quad x \in \partial\Omega \tag{2-6}$$

where F_s typically includes differential or integro-differential operators, and $f(x)$ is a forcing term. By incorporating physical constraints into PINN framework, they demonstrated that steady-state problems could be solved effectively, even for complex domains.

Dwivedi and Srinivasan (2020) employed a hybrid approach combining PINN with extreme learning machine (ELM), to solve linear advection-diffusion problems in one and two dimensions. While effective for linear differential operators, this approach faced limitations for more complex cases. Ramabathiran and Ramachandran (2021) extended the use of PINN to linear elliptic PDEs, such as solving the Poisson equation in both regular and irregular domains, showing that PINN is robust in dealing with non-smooth solutions.

Unsteady PDEs, which describe the temporal evolution of physical systems, are also effectively addressed by PINN. Niaki et al. (2021) studied heat transfer in composite materials using a system of PDEs, including the heat equation:

$$\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2} + b\frac{d\alpha}{dt} \qquad (2-7)$$

where parameters a and b are involved. They introduced a PINN consisting of two subnetworks and employed a sequential training algorithm to adaptively adjust the weights in the loss function, leading to improved prediction accuracy.

He and Tartakovsky (2021) used PINN to solve advection-dispersion equations, such as:

$$u_t + \nabla \cdot (-\kappa \nabla u + vu) = s \qquad (2-8)$$

where κ is the dispersion coefficient. They found that PINN produced results superior to those obtained through traditional discretization-based methods. Moreover, Dwivedi and Srinivasan (2020) and He and Tartakovsky (2021) solved the same two-dimensional advection-dispersion problem, but PINN performed better due to a more effective weighting of boundary and initial conditions in the loss function.

Overall, the versatility of PINN in handling both steady and unsteady PDEs makes it suitable for a broad range of engineering applications. Its ability to integrate complex boundary conditions, handle irregular geometries, and embed physical laws during training offers significant advantages over traditional numerical methods, particularly when high accuracy is needed in challenging scenarios.

Fractional Order Differential Equations

FODEs are useful for modeling complex phenomena that exhibit non-local behavior, such as anomalous diffusion and viscoelasticity, which are often challenging to capture using classical integer-order models. However, in practical spatiotemporal domains, experimental measurements are often scarce and affected by noise, making it difficult to accurately determine

the parameters of these models.

PIML has been extended to fractional PDEs by leveraging modified numerical methods to address the challenges posed by fractional operators, which cannot be easily handled using standard AD. For example, Mehta et al. (2019) employed L1 scheme to approximate fractional derivatives and solve problems such as turbulent flow with one-dimensional mean flow.

Pang et al. (2019) introduced the concept of fractional PINN (fPINN), which are designed to identify the parameters of fractional PDEs where the form of the equation is known, but coefficients and operators are unknown. Their approach was further extended by Kharazmi et al. (2021a), who addressed time-dependent fractional orders by employing separate NNs to represent each fractional order, along with a larger NN for representing the system states.

The flexibility and adaptability of PINN to fractional PDEs demonstrate their potential for accurately modeling complex systems with non-local dynamics, even when data is limited or noisy. The hybrid methods employed, such as combining AD with numerical schemes, have paved the way for the effective use of PIML in solving FODEs.

Fluid Mechanics

Fluid mechanics is one of the most promising areas for applying PIML, given the complex and often turbulent nature of fluid flows that require advanced modeling techniques. PIML approaches, particularly PINN, have demonstrated success in modeling a wide range of fluid mechanics problems, including compressible flows, incompressible flows, and biomedical flows (Mao et al. 2020; Jin et al. 2021; Kissas et al. 2020; Arzani et al. 2021).

PINN leverages governing PDEs to make accurate predictions in fluid systems, even in

data-sparse conditions, which allows for more robust and physically consistent models compared to purely data-driven approaches. For example, Mao et al. (2020) utilized PINN for modeling compressible flows, while Jin et al. (2021) applied it to incompressible flow problems, demonstrating the ability of PIML to handle both types of fluid dynamics effectively.

The use of PINN has gained wide attention, especially for predicting spatiotemporal dynamics in fluid flows and for inverse modeling, such as parameter estimation in NS equations (Raissi et al. 2019b; Raissi et al. 2020; Cai et al. 2021; Eivazi et al. 2022). Besides PINN, other PIML methods have also been applied for spatiotemporal modeling in fluid mechanics, including PDE-preserved neural network (PPNN) (Liu et al. 2022) and physics-informed convolutional-recurrent network (PhyCRNet) (Ren et al. 2022). These methods provide robust frameworks for modeling both forward and inverse problems, leveraging physical constraints to improve generalization.

Hybrid approaches are also extensively applied, particularly in turbulence closure modeling. Given the wide range of spatiotemporal scales present in turbulent flows, resolving all scales in practical configurations remains computationally infeasible. As a result, hybrid PIML approaches are commonly employed in Reynolds-Averaged Navier-Stokes (RANS) or large eddy simulations (LES) to model the effects of unresolved small-scale structures on larger scales (Duraisamy et al. 2019). ML-based turbulence closure models, such as those used for evaluating Reynolds stresses in RANS (Ling and Templeton 2015; Parish and Duraisamy 2016; Singh et al. 2017) or subgrid-scale statistics in LES (Lapeyre et al. 2019; Beck et al. 2019; Bode et al. 2021), have shown promising results in improving the accuracy and efficiency of turbulence simulations.

Solid Mechanics

In the field of solid mechanics, PINN has been employed to solve a wide range of engineering problems, including structural mechanics, elastostatics, elastoplasticity, and fracture mechanics. Haghighat et al. (2021a) applied PINN to solve time-dependent PDEs related to structural mechanics and vibration, while Haghighat et al. (2021b) incorporated momentum balance and constitutive relations into PINN for tackling nonlinear problems in solid mechanics.

PINN has demonstrated versatility in addressing both forward and inverse problems in solid mechanics. Forward problems involve solving IVPs and BVPs, whereas inverse problems focus on identifying material properties or detecting defects. For instance, Henkes et al. (2022) used PINN to model micromechanics for linear elastic materials, demonstrating the method's effectiveness in capturing small-scale material behavior. Similarly, Haghighat et al. (2021b) developed surrogate models using PINN for elastostatics and elastoplastic solids, offering significant computational advantages compared to traditional numerical methods.

Bastek and Kochmann (2023) used PINN to model the small strain response of shell structures, highlighting the applicability of PINN to structural elements with complex geometries. Zhang et al. (2020a; 2022a) demonstrated that PINN is capable of effectively identifying inhomogeneous material and geometric distributions under plane-strain conditions, making them useful for material characterization and defect detection in practical engineering scenarios. Although many of these applications rely on synthetic data as proof of concept, the PINN frameworks are also well-suited for experimental mechanics, where continuum mechanics theories apply seamlessly.

Beyond PINN, NO has also been employed for solving complex problems in solid mechanics, particularly those involving multiscale mechanics and nonlinear behavior. NO has been used for fracture mechanics (Goswami et al. 2022), multiscale mechanics (Yin et al. 2022b), and biomechanics (Yin et al. 2022a; You et al. 2022), offering a flexible approach for learning complex system responses across multiple scales and material behaviors.

Materials Science

In materials science, PINN has been applied to various problems, including the design and optimization of novel metamaterials (MMs), modelling of soft biological tissues, and solving inverse problems in photonic materials. Chen et al. (2020) utilized PINN to tackle inverse scattering problems in photonic MMs, facilitating the design of new functional nanomaterials.

Zhang and Gu (2021) developed a specialized PINN to emulate digital materials by minimizing the energy criteria within the loss function, thereby providing a new approach to designing and optimizing functional materials and structures. This approach enables the efficient exploration of a large design space, leading to optimized material properties that satisfy specific design requirements.

PINN has also been employed to construct constitutive models for soft biological tissues, which are known for their complex, nonlinear behavior. Liu et al. (2020) used PINN to model the mechanical response of soft tissues, capturing their intricate material characteristics and providing a data-driven approach that aligns with the underlying physical laws.

2.2 Physics-Informed Neural Network

PINN is an important framework within PIML and serves as the primary method utilized

in this thesis. This section provides an extensive literature review on the concept and implementation of PINN, its scope of applications, and related variants.

2.2.1 Introduction to PINN

PINN represents a cutting-edge approach within PIML designed to address problems involving differential equations (Cuomo et al. 2022). One of the fundamental aspects of PINN is its unsupervised training strategy, which does not require labeled data derived from previous simulations or experiments. Instead, PINN transforms the process of solving differential equations into a loss function optimization problem, thereby leveraging the mathematical model of the system as a direct part of the learning process. This paradigm is built upon earlier research by Owhadi (2015) on structured prior knowledge and extended by Raissi et al. (2019a) as a significant advancement in SciML.

The following contents will delve into the fundamental building blocks of PINN, highlighting the components and mechanisms that enable PINN to effectively integrate physical laws into the DL process.

Building Blocks

The building blocks of PINN comprise three primary components: the NN architecture, the physics-informed framework, and the training mechanism. These components work together to enable PINN to solve complex physical problems.

The first component is the NN architecture, which serves as the backbone of the PINN. It is typically composed of multiple hidden layers with nonlinear activation functions, allowing it to approximate complex relationships and model the solution of the governing equations.

This NN takes input variables, such as spatial and temporal coordinates, and produces outputs of predicted field values.

The second component is the physics-informed framework, which involves embedding physical laws directly into the NN through the loss function. The governing physical equations, such as ODEs and PDEs, are incorporated by computing their residuals at selected collocation points sampled within the problem domain. Also, initial and boundary conditions are considered in the same way. By adding these residuals to the loss function, PINN ensures that the NN predictions adhere to the underlying physical principles.

The third component is the training mechanism, which optimizes the NN parameters, i.e. weights and biases, to minimize a combined loss function. This loss function includes both physics-based losses derived from the governing equations, initial conditions, and boundary conditions, and data-driven losses from measurement data, where available. Optimization algorithms, such as SGD, Adam, and L-BFGS are employed to iteratively update the NN parameters to minimize this loss function.

In the following contents, these building blocks will be reviewed in detail.

Neural Network Architecture

NNs are a core component of PINN, providing the necessary expressiveness to approximate complex relationships. According to the universal approximation theorem, a NN, specifically a MLP with one hidden layer and a sufficient number of neurons, can approximate any continuous function arbitrarily closely (Cybenko 1989; Hornik 1989; Yarotsky 2017; Berman et al. 2019). This capability is vital in PINN, which leverage NNs to approximate the

solution of differential equations. However, identifying optimal parameters, including hyperparameters of the NN architecture, can be challenging in practice, especially when tackling specific PDEs (bin Waheed et al. 2021). Common NN architectures used in PINN include MLP, CNN, and RNN (LeCun et al. 2015). This section explores these architectures and their relevance to PINN.

The MLP (Gardner and Dorling 1998; Popescu et al. 1998), also known as FCFNN, is the most widely used architecture in PINN due to its universal approximation capability and simplicity. MLP consists of multiple layers of neurons, where each neuron is connected to every neuron in the subsequent layer. The input layer receives the input data, while hidden layers perform nonlinear transformations using activation functions such as rectified linear unit (ReLU) and hyperbolic tangent (Tanh) functions, allowing the NN to capture complex, nonlinear relationships (Hornik 1989).

CNN (LeCun et al. 2015; O'Shea and Nash 2015) is another type of NN architecture utilized in certain PINN extensions. CNN, initially popularized for image processing tasks, employes convolutional layers to extract spatial hierarchies from input data. The emergence of CNN began with pioneering work by LeCun et al. (2015) in the development of LeNet, which demonstrated the effectiveness of convolutional operations for feature extraction. In the context of PINN, CNN is useful when spatial patterns in the solution need to be captured efficiently, such as when applying finite-difference filters to construct differential equation residuals in a physics-based loss function. The ability of CNN to model local spatial dependencies makes them advantageous in applications where the underlying physical domain exhibits spatial invariance, such as in fluid mechanics and materials science.

RNN (Salehinejad et al. 2018), known for its ability to model sequential data, has also been adapted for use in PINN, particularly when temporal dependencies are involved. RNN contains cyclic connections, allowing it to maintain a memory of past inputs, which makes it suitable for modeling systems governed by time-dependent differential equations. However, standard RNN often struggles with long-term dependencies due to issues like vanishing gradients, which can impair its learning capabilities. LSTM network, a variant of RNN introduced by Hochreiter and Schmidhuber (1997), addresses this limitation by introducing memory cells that regulate the flow of information, making them more effective for capturing long-range dependencies. This characteristic makes RNN and LSTM suitable for PINN applications involving dynamic systems and problems requiring temporal forecasting.

While MLP or FCFNN is the most commonly used NN architecture in PINN framework, the use of CNN, RNN, and LSTM highlights the versatility of NNs in addressing different types of physical problems. The selection of a suitable NN architecture in PINN is crucial, as it directly influences the model's ability to approximate the solution of differential equations efficiently and accurately.

Injection of Physics

The key innovation of PINN is the incorporation of physical laws into the NN training process. To solve a differential equation using PINN, the network should approximate the solution while satisfying the underlying physics, which is realized by embedding the governing equations, such as ODEs and PDEs, directly into PINN's loss function. The NN's output is differentiated with respect to the input variables to calculate the residuals of the governing

equations, boundary conditions, and initial conditions.

To achieve this, derivatives of the NN's output are computed using AD (Paszke et al. 2017), which is the most commonly used function for PINN. AD offers several advantages over other differentiation methods, such as symbolic and numerical differentiation. Symbolic differentiation can be computationally intensive for complex functions, and numerical differentiation often suffers from floating-point precision errors. In contrast, AD computes exact derivatives without approximation errors, making it well-suited for incorporating physical constraints into PINN.

AD works by systematically applying the chain rule to compute derivatives, generating a computational graph of all operations involved in the PINN framework. This allows for efficient calculation of gradients in both forward and reverse modes, depending on the problem requirements. The forward mode is typically used when the number of inputs is smaller than the number of outputs, while reverse mode is advantageous when the number of outputs is smaller (bin Waheed et al. 2021). In PINN, AD is used to evaluate the differential equation residuals, which are then included in the PINN's loss function, ensuring that the network learns a solution that adheres to the governing physical laws (Yang and Perdikaris 2019b).

Raissi and Karniadakis (2018) originally formulated the differential equation residual by differentiating the NN output with respect to the input variables, using AD to enforce the physical constraints. This formulation allows PINN to minimize the discrepancy between the NN approximation and the true solution by embedding the differential equation residuals into the loss function. The residuals for both the governing equations and the boundary or initial conditions are computed in this manner, ensuring that the learned solution satisfies the entire

problem domain.

However, AD is not always suitable for every type of differential operator. For instance, Pang et al. (2019) proposed the use of discrete approximations for fractional differential operators, which are then incorporated into the PINN loss function to address situations where AD may not be applicable. Additionally, Fang (2021) used a local fitting method instead of AD to approximate differential operators, allowing for the verification of convergence in their specific implementation of PINN.

The flexibility of AD in calculating derivatives makes it the preferred method for injecting physical constraints into PINN. This approach reduces the reliance on labeled training data and enhances the model's generalizability, making PINN a powerful tool for solving complicated physical problems.

Training Process

The training of PINN involves optimizing the NN parameters, i.e. weights and biases, to approximate the solution of a given physical problem by minimizing a loss function consisting of different penalty terms. The penalty terms of this loss function include the residual loss related to the governing equation (L_f), boundary conditions (L_b), initial conditions (L_i), and the data loss from measurement data (L_m), if it is an inverse problem. The overall objective is to find the NN parameters θ, which contain weights and biases, that minimize this combined loss function.

The penalty term L_f represents the discrepancy between the predicted solution and the physical laws described by the governing equations (He et al. 2020). This residual is calculated

at a set of sampled collocation points, which can be distributed uniformly or non-uniformly throughout the problem domain. By minimizing L_f, the network is forced to satisfy the governing equation at these collocation points.

The penalty terms for boundary and initial conditions, denoted as L_b and L_i, respectively, ensure that the NN solution adheres to the given conditions at the domain boundaries or initial time. These conditions are enforced by penalizing deviations from the prescribed values at specific boundary or initial time points.

In some cases, PINN is trained in a supervised manner, where L_m is introduced to account for measurement data points (e.g., observed data). Measurement data is introduced for two possible reasons. First, the problem is an inverse type, in which unknown parameters are to be identified with the involvement of measurement data. Second, measurement data can be incorporated to facilitate computation of some complex problems, in which the loss function is difficult to converge. This term L_m minimizes the difference between the NN's output and the observed data, allowing for the incorporation of additional information into the model. The weight associated with this loss term can be adjusted to account for the quality or reliability of the available data.

The optimization of the loss function is typically performed using gradient-based methods, such as SGD, the Adam optimizer, and the L-BFGS algorithm. The Adam optimizer, which combines adaptive learning rate and momentum techniques, is often used to accelerate convergence, particularly in high-dimensional problems (Zhu et al. 2021). L-BFGS, a quasi-Newton method, is also popular in PINN training due to its faster convergence for smaller datasets or fewer residual points (He et al. 2020). A common training approach that combines

the advantages of different optimizers is to first use Adam for several iterations to achieve a good initialization, followed by L-BFGS to refine the solution.

The initialization of NN parameters can significantly impact the training process. Poor initialization may lead to slow convergence or suboptimal solutions, while good initialization can facilitate faster convergence. To address this potential issue, Pang et al. (2019) proposed a two-step initialization technique that first generates a low-fidelity PINN model of the problem to obtain preliminary parameters, which is then used as the starting point for the second step training process.

Overall, the training process of PINN is designed to optimize the loss function with a balance of different penalty terms. By minimizing the loss function through appropriate optimization techniques, PINN is able to accurately model complex physical systems, even in scenarios where data is scarce or incomplete (Zhang et al. 2020b).

Treatment of Boundary Conditions in PINN

Boundary conditions are constraints necessary for the solution of BVPs. A BVP is a differential equation (or system of differential equations) to be solved in a domain on whose boundary a set of conditions is known. It is opposed to the IVP, in which only the conditions on one extreme of the interval are known.

BVPs are extremely important as they model a vast number of phenomena and applications, from solid mechanics to fluid mechanics, heat transfer, and acoustic diffusion. They arise naturally in every problem based on a differential equation to be solved in space, while IVPs usually refer to problems to be solved in time.

In a problem described by differential equations, boundary conditions can be commonly expressed by:

$$B(u, x) = 0, \quad \text{on } \partial\Omega \tag{2-9}$$

where $B(u, x)$ could be different types of boundary conditions, e.g. Dirichlet boundary condition (also known as Type I), Neumann boundary condition (Type II), Robin boundary condition (Type III), Cauchy boundary condition, mixed boundary condition, and so on.

For time-dependent problems, time *t* is considered as a special component of *x*, and Ω contains the temporal domain. The initial condition can be simply treated as a special type of Dirichlet boundary condition on the spatio-temporal domain.

In the following contents, the soft and hard embedding methods for boundary conditions will be reviewed and introduced.

Soft Constraints

In PINN's configuration, the boundary conditions are enforced as soft constraints through a penalty term L_b in the total loss function as follows:

$$L_b(\theta; T_b) = \frac{1}{T_b} \sum_{x \epsilon T_b} \|B(\hat{u}, x)\|_2^2 \tag{2-10}$$

in which, $T_b \in \partial\Omega$ refers to the set of residual points, $\hat{u}$ is the output of NNs. In addition, there will be a penalty coefficient λ_b multiplied by L_b, then integrated into the total loss function.

This approach can be used for complex domains and any type of boundary conditions. However, the soft-embedding of boundary conditions has several major drawbacks:

(1) There is no quantitative guarantee on how accurate the boundary conditions are being

imposed and thus the solution could be unsatisfactory;

(2) The optimization performance depends on the relative importance of each penalty term, but how to assign weight for each term can be difficult.

Alternatively, the boundary conditions can be imposed in a "hard" manner, where a particular solution that solely satisfies the boundary conditions is added. Hence, the constraints on boundary conditions are automatically fulfilled. By doing so, the penalty term for boundary conditions in the loss function will no longer be counted, thus, the computational efficiency and prediction accuracy can be improved.

Hard Constraints

In order to enforce boundary conditions in a hard manner, various methods for hard-embedding of boundary conditions have been studied (Luo and Yang 2020; Du and Zaki 2021; Zhu et al. 2021; Chen et al. 2021; Zhang et al. 2022b; Huang et al. 2023; Lu et al. 2021c). In this subsection, some typical methods will be reviewed.

Most studies concern Dirichlet boundary conditions. The common way to automatically satisfy Dirichlet boundary conditions is to reconstruct the NN output in the following form:

$$h(x;\theta) = h^*(x) + d(x)u(x;\theta) \qquad (2-11)$$

where $u(x;\theta)$ is the original output of the NN; $h^*(x)$ is a particular function which exactly satisfies the Dirichlet boundary condition; and $d(x)$ is a function which vanishes on the Dirichlet boundary. Due to the introduction of $d(x)$, it is easy to see that the newly constructed output $h(x;\theta)$ automatically satisfies the Dirichlet boundary condition.

Although the key idea is the same, there are different ways to construct $h^*(x)$ and $d(x)$.

In Zhang (2022b), $d(x)$ is chosen as the distance function from the point x to the Dirichlet boundary:

$$d(x) = \frac{(x - x_{min})(x_{max} - x)}{(x_{max} - x_{min})^2} \quad (2-12)$$

In Luo and Yang (2020), a special NN satisfying three different types of boundary conditions are introduced. For Dirichlet boundary conditions $u(a) = a_0$ and $u(b) = b_0$, $d(x)$ is set as:

$$d(x) = (x - a)^{p_a}(x - b)^{p_b} \quad (2-13)$$

where $0 < p_a, p_b < 1$, and $h^*(x)$ can be chosen as:

$$h^*(x) = \frac{(b_0 - a_0)(x - a)}{b - a} + a_0 \quad (2-14)$$

For mixed boundary conditions $u'(a) = a_0$ and $u(b) = b_0$, $d(x)$ is constructed as:

$$d(x) = (x - a)^{p_a} \quad (2-15)$$

with $1 < p_a \leq 2$, and $h^*(x)$ is chosen as:

$$h^*(x) = -(b - a)^{p_a}u(x;\theta) + a_0 x + b_0 - a_0 b \quad (2-16)$$

For Neumann boundary conditions $u'(a) = a_0$ and $u'(b) = b_0$, $h(x;\theta)$ can be constructed as:

$$h(x;\theta, c_1, c_2) = \exp\left(\frac{p_a x}{a - b}\right)(x - a)^{p_a}((x - b)^{p_b}u(x;\theta) + c_2)$$
$$+c_1 + \frac{(b_0 - a_0)}{2(b - a)}(x - a)^2 + a_0 x \quad (2-17)$$

where $1 < p_a, p_b \leq 2$, c_1 and c_2 are two parameters to be trained together with NN's parameter θ.

In Du and Zaki (2021), the form of the transformed NN output is slightly different. A two-dimensional domain with an arbitrary point x inside is shown in Figure 2-2. Horizontal and

vertical rays emanating from x intersect the boundary $\partial\Omega$ at x_e, x_w, x_n, and x_s, with corresponding distances a_e, a_w, a_n, and a_s, which are all functions of x. Then, $h(x;\theta)$ can be constructed as:

$$h(x;\theta) = h^*(x) + u(x;\theta) + c_e u(x_e;\theta) + c_w u(x_w;\theta) + c_n u(x_n;\theta) + c_s u(x_s;\theta) \tag{2-18}$$

where $u(x;\theta)$ is the NN output that has nonzero boundary values. The coefficients c_e, c_w, c_n, and c_s are as follows:

$$\begin{aligned} c_e &= -\frac{a_w a_n a_s}{a_w a_n a_s + a_e} \\ c_w &= -\frac{a_e a_n a_s}{a_e a_n a_s + a_w} \\ c_n &= -\frac{a_w a_e a_s}{a_w a_e a_s + a_n} \\ c_s &= -\frac{a_w a_e a_n}{a_w a_e a_n + a_s} \end{aligned} \tag{2-19}$$

in which, the choice of construction can be motivated by considering, for example, $c_e(a_e, a_w, a_n, a_s)$, which satisfies:

$$\begin{aligned} c_e(0, a_w, a_n, a_s) &= -1 \\ c_e(a_e, 0, a_n, a_s) = c_e(a_e, a_w, 0, a_s) &= c_e(a_e, a_w, a_n, 0) = 0 \end{aligned} \tag{2-20}$$

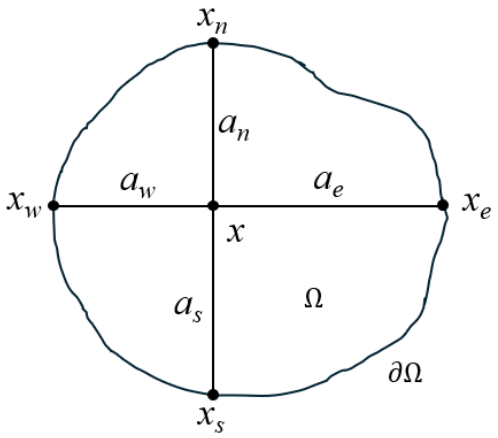


Figure 2-2 An arbitrary two-dimensional domain

In Zhu et al. (2021), a Heaviside function is used to embed Dirichlet boundary condition in a hard manner:

$$H_\epsilon(x) = \begin{cases} 1 - \cos\left[\frac{d(x)\pi}{\epsilon}\right], & if\ d(x) < \epsilon \\ 1 & if\ d(x) \geq \epsilon \end{cases} \quad (2-21)$$

where $d(x)$ is the distance to the Dirichlet boundary; ϵ defines an artificial thickness of the boundary. With $H_\epsilon(x)$, the $h(x;\theta)$ can be defined as:

$$h(x;\theta) = h^*(x)[1 - H_\epsilon(x)] + u(x;\theta)H_\epsilon(x) \quad (2-22)$$

In addition, the penalty method and the augmented Lagrangian method are also used to enforce hard constraints in relevant study (Lu et al. 2021c).

2.2.2 Scope of Application

Forward Problems

Forward problems in PINN involve determining the solution of a physical system based on governing equations, initial conditions, and boundary conditions with known parameters. Essentially, the objective of a forward problem is to predict the behavior of the system under given conditions. In PINN, forward problems are solved by embedding the known governing equations directly into the NN's loss function, allowing the network to learn the system's behavior while adhering to the underlying physics.

An example of a forward problem is predicting the temperature distribution in a heat conduction scenario. Given the thermal properties of a material, boundary conditions, and an initial temperature distribution, the forward problem aims to determine how the temperature evolves over time and space. By using PINN, the heat equation, which governs heat conduction,

is embedded into the loss function, and the NN learns to predict the temperature profile without the need for extensive labeled data.

Another common example is fluid flow simulation. In fluid dynamics, the NS equations describe the motion of fluid substances. A forward problem in this context would involve predicting the velocity and pressure fields of a fluid given initial and boundary conditions. By incorporating the NS equations into the training process, PINN can model complex fluid behaviors, such as laminar and turbulent flows, while ensuring that the solution is consistent with the physical laws. Also, PINN finds applications of forward problems in structural mechanics. PINN and its improved version are used to analysis the deformation of beams and steel frames under loading (Zhang et al. 2025b).

The forward problem formulation in PINN is particularly useful in scenarios where traditional numerical methods, such as FEM or FDM, may be computationally expensive or challenging to implement. PINN provides an efficient alternative by leveraging NNs to approximate the solution, which can be applied to different scenarios by transfer learning (TL).

Inverse Problems

Inverse problems, on the other hand, involve identifying unknown parameters or determining unknown inputs of a physical system based on governing equations and measurement data. In PINN, the goal of an inverse problem is to infer certain properties or parameters of the system that are not directly measurable. The parameters to be determined are defined as trainable parameters and will be updated with NN's parameters. The NN is trained to minimize a loss function that incorporates penalty terms of the governing equation,

measurement data, and boundary and initial conditions, if applicable.

A common example of an inverse problem is parameter identification in a material under stress. Suppose the experimental data on the displacement of a material subject to loading is obtained, but the material's elastic properties, such as Young's modulus, are unknown. By embedding the governing equations of elasticity into the PINN framework and using the observed displacement data, the inverse problem can be formulated to estimate the unknown material properties. Furthermore, parameter identification is also performed by PINN in structural systems. For example, the unknown rotational stiffness of semi-rigid joints in beam structures and frame structures is identified by PINN with limited measurement data from statistic tests (Zhang et al. 2025b).

Another example is in the field of groundwater flow. In hydrogeology, the properties of an aquifer, such as hydraulic conductivity, are often not directly measurable. By using PINN, one can infer these properties by embedding the governing equations for groundwater flow into PINN's loss function and using observed data, such as water levels at specific locations. The network learns both the solution to the flow equations and the unknown aquifer properties, effectively solving the inverse problem.

Inverse problems in medical imaging can also be handled by PINN, such as electrical impedance tomography, where the objective is to determine the internal conductivity distribution of a body from surface voltage measurements. By leveraging the governing equations of electrical conductivity and incorporating the observed data, PINN can be used to reconstruct the internal properties of the body, providing a non-invasive method for medical diagnostics.

Overall, PINN is powerful in dealing with inverse problems where the underlying physical properties are not fully known but sufficient measurement data is available. Another merit of PINN for inverse problems is that it requires the same architecture as forward problems, and only trainable parameters representing unknowns should be added into the framework and updated during the training process. By incorporating both the observed data and the governing physical laws, PINN can infer unknown parameters, providing a feasible approach to solving challenging inverse problems.

2.2.3 Related Variants

A-PINN

The auxiliary physics-informed neural network (A-PINN) (Yuan et al. 2022) framework enhances traditional PINN by enabling direct solution of nonlinear IDEs without numerical integration. By introducing auxiliary outputs of NN to represent integral terms, A-PINN leverages AD to replace quadrature and eliminate discretization and truncation errors. Additional output constraints enforce the exact physical relationships between solution variables and integrals that are represented by auxiliary outputs, ensuring consistency. As a mesh-free, grid-independent method, A-PINN can predict solutions at any domain point without interpolation. Benchmark tests on Volterra and Fredholm IDEs demonstrate superior accuracy and computational efficiency compared to quadrature-based PINN approaches. Furthermore, treating unknown parameters as trainable variables allows A-PINN to address inverse IDE problems effectively, even in the presence of noisy data, offering a robust and versatile approach.

B-PINN

Bayesian physics-informed neural network (B-PINN) (Yang et al. 2021) integrates Bayesian NNs with physics-informed loss functions to solve both forward and inverse PDEs under noisy observations. By treating NN's weights as random variables, B-PINN encodes physical laws as priors and employes Hamiltonian Monte Carlo (HMC) or variational inference to estimate posterior distributions. This framework yields calibrated predictions with quantified aleatoric uncertainty and mitigates overfitting when data noise is large. Comparative studies demonstrate that HMC-based inference outperforms mean-field variational methods and dropout-based uncertainty, while truncated Karhunen-Loève expansions and deep normalizing flows offer alternative posterior representations. B-PINN thus provides an uncertainty-aware approach for data-driven PDE modeling.

hp-VPINN

Variational physics-informed neural network (VPINN) is extended via an hp-refinement strategy that combines global neural trial functions with local polynomial test spaces (Kharazmi et al. 2021b). In hp-VPINN, a feedforward network approximates the solution over the full domain, while the variational residuals are projected onto piecewise high-order polynomials defined on subdomains, enabling domain decomposition. By adjusting element sizes (h-refinement) and polynomial degrees (p-refinement), hp-VPINN localizes optimization and control approximation properties. This dual refinement leverages AD for differential operators and quadrature-based evaluation of integrals in each subdomain. Numerical experiments confirm that hp-VPINN achieves enhanced accuracy and reduced training cost compared to other PINN-based approaches.

2.3 Physics-Informed Graph Neural Network

By introducing GNN into PIML family, PIGNN is proposed as a promising candidate for dealing with graph-structured data. Although PIGNN is not as commonly used as PINN, it holds great potential in engineering applications. In this thesis, a PIGNN framework is proposed for cable-strut structural systems. Besides, similar ideas have also been studied for various engineering fields. As another PIML framework proposed in this thesis, a brief review on PIGNN is presented in this section.

2.3.1 Overview of PIGNN

PIGNN represents an emerging fusion of two powerful paradigms, PINN and GNN, to address problems characterized by underlying physical laws on irregular domains or relational data structures (Shukla et al. 2022). Unlike traditional PINN that embeds differential equations directly into FCFNN, PIGNN leverages the inductive biases of GNN to process mesh-based discretization or graph-structured representations of physical systems, such as finite element (FE) meshes in computational fluid dynamics or molecular interaction networks. This hybridization enables the modeling of spatially complex problems with fewer training samples, capitalizing on graph connectivity to propagate physics constraints across non-Euclidean domains (Peng et al. 2022). Early applications of PIGNN span from rapid emulation of urban wind fields on unstructured meshes to circuit compact modeling, demonstrating improved generalization and interpretability compared to purely data-driven GNN or standard PINN. Benchmark studies have systematically compared various PIGNN architectures, such as Hamiltonian GNN (HGNN), Lagrangian GNN (LGNN), and graph NODE (GNODE),

highlighting the crucial role of explicit physics constraints in preserving conserved quantities and achieving zero-shot scalability to larger system sizes. The physics-informed approach also extends naturally to inverse problems, where model parameters or source terms are inferred from sparse and noisy measurements by incorporating additional regularization via physics-based loss terms. As PIML advances towards multi-physics, multi-fidelity data, PIGNN stands out for its ability to harmonize domain knowledge, graph-based relational inductive biases, and DL flexibility into a coherent framework for both forward and inverse modeling tasks.

Foundational Principles

The core innovation of PIGNN lies in the integration of physical laws, typically expressed as differential equations or conservation constraints, into the message-passing mechanisms of GNN (Shukla et al. 2022). Graphs are constructed to represent discretized spatial domains (e.g., FE meshes) or relational structures (e.g., molecular graphs), with nodes encoding field values and edges capturing adjacency or interaction kernels. The physics-informed branch augments the standard GNN's loss functions, which typically minimize prediction error on labeled data, with additional terms that penalize governing equation residuals evaluated via AD on the graph. In some forward problem cases, even the labeled data is not required. This approach ensures that learned representations satisfy governing equations and fit observed data, if applicable, at graph nodes or along edges, thereby enforcing global consistency with physical principles (Peng et al. 2022). In addition to residual-based penalties, PIGNN frequently incorporates variational formulations, projecting weak-form differential equation residuals onto test-function spaces defined over each graph element, to improve numerical stability and reduce

sensitivity to discretization errors. Conservation laws (e.g., mass or energy) are enforced explicitly either through architectural constraints, such as Hamiltonian or symplectic message-passing schemes, or via augmented Lagrangian methods that impose divergence-free or curl-free conditions on learned fields. Moreover, the graph structure enables localized information aggregation, which respects the non-Euclidean geometry of domains and fosters zero-shot generalization to larger or more refined meshes (Thangamuthu et al. 2022).

Core Architectures

The architectural landscape of PIGNN is diverse, reflecting the rich interaction between graph representation learning and physics embedding. A typical PIGNN comprises several layers of graph convolution or message-passing, where at each layer node features are updated by aggregating neighbor information along edges weighted by learned functions of edge attributes (e.g., distance or material properties) (Shukla et al. 2022). Physics-informed variants augment these updates with differential operators discretized on the graph. For instance, graph Laplacian-based convolutions approximate diffusion terms, while custom edge-function modules capture advection or reaction kinetics. More specialized architectures include HGNN and LGNN, which parametrize the system's Hamiltonian or Lagrangian to ensure symplectic integration and energy conservation over long-term predictions. GNODE further generalizes the concept by defining continuous-time dynamics on graph nodes parameterized by neural message functions, offering flexible step sizes and adaptive integration to handle stiff PDEs (Thangamuthu et al. 2022). Some PIGNN frameworks embed multi-scale hierarchies, using graph coarsening to capture global physics at coarse resolutions and refining with local graph

passes to resolve fine-scale features, which is particularly effective in fluid dynamics and materials science applications. Domain-specific modules, such as physics-informed graph attention layers, prioritize information flow along edges aligned with dominant physical gradients, improving focus on critical flow features or stress concentrations (Di Lorenzo et al. 2024). These core architectures collectively provide a versatile toolkit for modeling diverse physics-driven phenomena on graph-structured data.

Training Methodologies

Training PIGNN involves optimizing network weights to minimize a loss function that balances different penalty terms for underlying physics and available data. The typical procedure employs SGD, Adam, or L-BFGS, with backpropagation through both physics-informed and data-driven penalty terms, leveraging AD for efficiency. Data points are sampled from scattered sensors or simulation outputs for the supervised component, while collocation points, random or structured, are drawn across the graph for physics residual evaluation, ensuring coverage of the domain (Shukla et al. 2022). Adaptive sampling strategies, such as active learning loops, have been introduced to refine collocation point distributions based on physics loss magnitudes, thereby allocating computational effort to regions with high residual error. Hybrid training schemes combine pretraining on supervised tasks followed by physics-informed fine-tuning to accelerate convergence and improve stability (Peng et al. 2022). Regularization techniques, dropout adapted to graph edges and weight decay, mitigate overfitting in data-scarce regimes, while multi-fidelity training incorporates low- and high-resolution data streams via hierarchical loss weighting. For inverse problems, joint

optimization of GNN parameters and unknown physical parameters is performed, often aided by Bayesian formulations to quantify uncertainty in estimated parameters (Di Lorenzo et al. 2024). Finally, scalable training on large graphs leverages mini-batch subgraph sampling and distributed computation frameworks, enabling PIGNN to tackle high-dimensional engineering problems with millions of nodes and edges.

2.3.2 Applications of PIGNN in Engineering

Computational Fluid Dynamics

PIGNN has emerged as an effective tool for surrogate modeling in fluid dynamics, enabling efficient approximation of complex flow fields on unstructured meshes. By embedding conservation laws and boundary conditions directly into the learning process, PIGNN can predict velocity and pressure distributions with high fidelity, even in regimes characterized by turbulence or multiphase interactions (Shao et al. 2023). These models leverage graph representations of the mesh to propagate local physical constraints across the domain, thereby reducing the need for extensive labeled data and accelerating inference by orders of magnitude compared to traditional solvers (Suk et al. 2024). Recent studies also demonstrate that PIGNN maintains robustness under mesh refinement and can generalize across different geometries, offering a pathway toward real-time flow control and optimization (Kim et al. 2025).

Structural Health Monitoring

In SHM, PIGNN facilitates the analysis of sensor networks by modeling the relationships among spatially distributed measurements on structures such as bridges and aircraft. Graph-

based representations of sensor layouts allow PIGNN to infer damage indicators, such as crack initiation or material degradation, by enforcing mechanical equilibrium and modal behavior during training process (Bloemheuvel et al. 2021). By combining physics-informed regularization with data-driven feature extraction, these networks improve detection sensitivity and interpretability compared to purely statistical methods (Karimi and Gündel 2024). Applications have shown that PIGNN can be adapted to varying operational conditions and site-specific characteristics, enhancing prognostic accuracy for maintenance scheduling and safety assessment (Jian et al. 2024).

Power Systems

Power-system engineering has adopted PIGNN for real-time grid management and state estimation by mapping network topology to graph representations. In distribution network reconfiguration, PIGNN integrates switch-status physics constraints into graph message-passing, yielding fast, near-optimal topology decisions under changing load scenarios (Authier et al. 2024). Physics-informed graph state estimators incorporate nodal power-balance equations into loss functions, improving robustness against measurement noise compared to classical approaches (Ngo et al. 2024). Line-graph extensions further enable efficient power-flow calculations by enforcing Kirchhoff's laws on edge-centric networks, demonstrating PIGNNs' versatility in energy-system applications (Zhang et al. 2024).

2.4 Research Gaps

The research gaps identified in the application of PIML, especially PINN, in structural engineering can be summarized as follows:

(1) Difficulties in solving high-order differential equations in structural engineering: Many structural engineering problems involve high-order governing equations, such as the fourth-order Euler-Bernoulli beam equation and plate deflection equations. Solving these high-order equations using PINN presents challenges due to the need for repeated AD, which not only reduces computational efficiency but also leads to error accumulation, thereby decreasing accuracy. High-order derivatives require multiple rounds of differentiation during training, which often results in gradient vanishing or explosion issues, making training unstable and less efficient. One way to address this issue is by reducing the order of the governing equations. This can be achieved by introducing auxiliary outputs in the NN within the PINN framework to represent first- or second-order derivatives of the original outputs. By defining auxiliary variables that denote these lower-order derivatives, the high-order governing equations can be expressed in terms of a series of lower-order differential equations, which not only reduces the computational burden but also mitigates the accumulation of differentiation errors.

(2) Soft enforcement of boundary conditions: In the vanilla PINN framework, boundary conditions are satisfied by incorporating the corresponding penalty term into the total loss function, which can only be minimized rather than reduced to zero, leading to approximate satisfaction of boundary conditions. Since multiple penalty terms are involved, the optimization process becomes a multi-objective problem, where each loss component competes during the training. This often results in a trade-off between minimizing different penalty terms, ultimately affecting the accuracy of PINN in

satisfying boundary conditions. Hard embedding of boundary conditions can overcome these limitations. By transforming the NN's inputs through modulating functions, boundary conditions can be strictly satisfied. This method enhances the accuracy of PINN at boundaries and reduces the total number of penalty terms. Additionally, since boundary conditions are automatically satisfied, the corresponding penalty terms can be omitted, thus the loss function is simplified and the multi-objective optimization can be alleviated. As a result, the computational efficiency is improved, and the training becomes more stable.

(3) Generalization issues with hard enforcement of boundary conditions: Although the hard-embedding method enabled by modulating functions can effectively reduce the number of penalty terms, it comes with limitations. For different boundary conditions, new modulating functions should be derived, limiting the generalizability of the approach. This manual derivation is often difficult, particularly for problems with varying or complex boundary conditions. For irregular domains, deriving analytical modulating functions may be impossible. Therefore, there is a need for a unified framework that can flexibly handle various types of boundary conditions across different domains without requiring repetitive derivation of modulating functions. Such a framework would provide a generalizable mechanism for embedding boundary conditions into the NN architecture, making PINN applicable to a broader range of problems, including those with irregular geometries or complex boundary definitions.

(4) Failure to incorporate structural topologies in PINN: Many structural systems, such as cable-strut structures, trusses, and frames, have inherent topological relationships

between nodes and elements that are crucial to their behavior. The traditional PINN framework, which uses the FCFNN as the NN architecture, cannot effectively capture these topological relationships, as FCFNN is designed for mapping input-output relationships without considering spatial or topological dependencies. By introducing GNN into the PIML family, replacing the FCFNN in vanilla PINN with the GNN, the structural topologies can be explicitly considered in the model. GNN is well-suited for problems involving graph-like data, where the relationships between nodes (e.g., joints in a truss) and edges (e.g., bars or cables) play a significant role. Incorporating GNN allows PINN to inherently account for the connectivity and topological features of the structure, making them more effective for structural analysis. This approach is analogous to FEM, where the connectivity of nodes and elements is explicitly modeled to analyze the behavior of structures. By using GNN, the influence of structural topology can be captured, leading to improved efficiency and accuracy in analyzing complex structural systems, particularly those that involve intricate connectivity and load distribution patterns.

CHAPTER 3

Physics-Informed Neural Network with Auxiliary Outputs for Reducing the Order of Governing Equations

3.1 Introduction

PINN is proposed to approximate solutions or identify unknown coefficients in differential equations that describe complex physical systems. By integrating physical knowledge as prior information to NNs, it has been proven that PINN is a promising path to solve differential equations. Compared with existing numerical methods such as FEM, PINN is a mesh-free method that avoids complex mesh generation and eliminates discretization and truncation errors. It also offers good flexibility in handling complicated boundary and initial conditions and can naturally address both forward and inverse problems within the same framework. Nevertheless, it has been found that computational efficiency and accuracy significantly decrease for differential equations with high orders (Wight and Zhao 2020; Haghighat et al. 2021a). It has also been observed that PINN may fail for high order differential equations since the loss function fails to converge. Therefore, it is crucial to investigate the reasons for the failure of PINN's training and improve the robustness of PINN.

In this chapter, the basic knowledge of the vanilla PINN framework is introduced first.

Then, the training dynamics of PINN, especially for high order differential equations, are analyzed to provide insights into the failure associated with high order derivatives. To overcome this issue and improve PINN's computational efficiency and accuracy, a novel method involving auxiliary outputs is proposed.

The main content is organized as follows: The detailed introduction to the vanilla PINN framework and the analysis of training dynamics of PINN are presented in Section 3.2, followed by the proposed method in Section 3.3. The performance of the proposed method is validated through three numerical case studies in Section 3.4.

3.2 Vanilla PINN Framework

This section introduces the vanilla PINN framework in detail as the foundation of this study. Several critical concepts are detailed. For example, the vanilla PINN is built on a basic component called FCFNN, which is a commonly used type of DNN. AD is a vital function utilized in vanilla PINN to calculate gradients. The loss function is established based on physical laws governing the system and is optimized during the training process. The procedures for addressing both forward and inverse problems under vanilla PINN framework are also introduced.

3.2.1 FCFNN and AD

The FCFNN is the core component of a PINN framework. An example of a FCFNN with three hidden layers is shown in Figure 3-1. Typically, FCFNN takes several inputs and gives a certain number of outputs according to the network architecture. Except for the input and output layers, there are several hidden layers between them. Within a hidden layer, there are several neurons representing weights, biases, and nonlinear activation functions. The inputs will first

be calculated by the weights and biases in each hidden layer and then be transformed through activation functions such as Tanh, ReLU, and Sigmoid (Sun et al. 2020). The calculation of inputs is called the forward propagation of FCFNN, as shown in Figure 3-2, which can be expressed by the following formulae:

$$u_{pred}^{i}(x) = \sigma\left(w^{i} \cdot u_{pred}^{i-1}(x) + b^{i}\right), \qquad 1 \le i \le n \tag{3-1}$$

$$u_{pred}(x) = w^{n+1} \cdot u_{pred}^{n}(x) + b^{n+1} \tag{3-2}$$

where x denotes the inputs of FCFNN; $u_{pred}^{i}(x)$ and $u_{pred}(x)$ denote the outputs of the i-th hidden layer and the output layer, respectively; w^{i} and b^{i} denote the weights and biases of hidden layers, respectively; σ is the activation function. In the following content, θ is used to represent the combination of weights w^{i} and biases b^{i} for brevity.

AD (Baydin et al. 2018) is a useful self-embedded function in commonly used DL packages such as PyTorch (Paszke et al. 2017; Paszke et al. 2019) and TensorFlow (Abadi et al. 2016). From the perspective of forward propagation, the FCFNN can be expressed mathematically as an amalgamation of a series of matrix operations and activation functions on the inputs. If the activation function is differentiable, the derivatives of two related variables in the FCFNN can be calculated explicitly and easily through the AD function. Finally, by applying the chain rule to the network architecture, the derivatives of the outputs with respect to the inputs can be obtained. Although AD is an exact derivative method that avoids discretization and truncation errors at the mathematical level, the expressions of gradients are computed in floating-point, which means there still are round-off, cancellation, overflow, or underflow effects. These effects will bring errors, and these errors will be accumulated during training. The derivatives of outputs are calculated in the backpropagation of FCFNN, which is demonstrated in Figure 3-2. A cyclic combination of the forward and backward propagations constitutes the iteration process of training FCFNN.

The parameters θ of hidden layers, i.e., weights and biases, will be trained and updated by gradient descent algorithms during the training. The training of FCFNN is an optimization process that is usually achieved by optimizers such as Adam (Kingma and Ba 2017) and L-BFGS (Byrd et al. 1995) according to the residuals between the outputs and the targets. The residuals are defined by the loss function, which will be introduced in the next subsection.

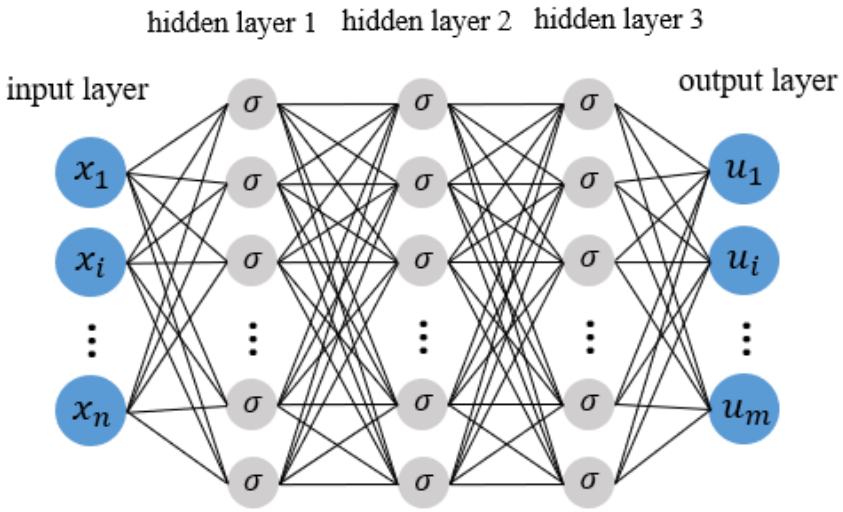


Figure 3-1 A FCFNN with three hidden layers

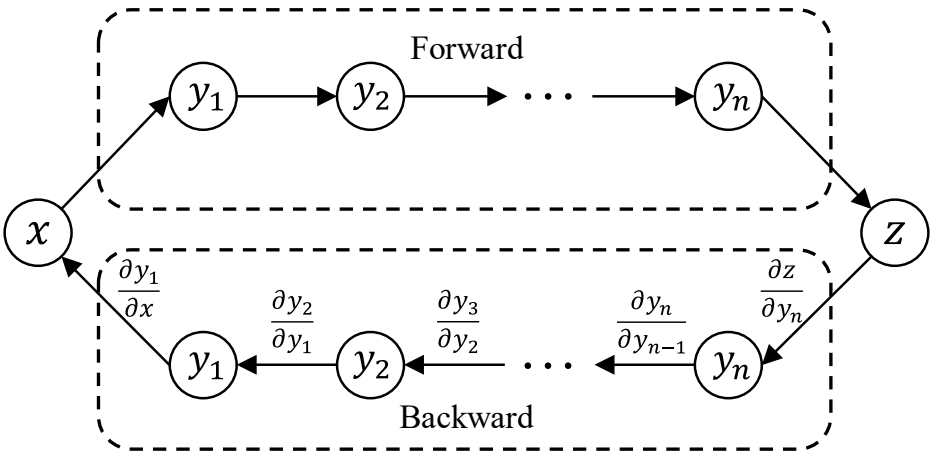


Figure 3-2 Forward and backward propagations

3.2.2 Loss Function and Training of PINN

PINN encodes the underlying physics of a system described by ODEs or PDEs into

FCFNN by defining a physics-informed loss function that contains penalty terms for the governing equation and each condition. By minimizing the loss function, PINN is able to generate an accurate surrogate model to make predictions. To demonstrate how to establish the physics-informed loss function, the following nonlinear governing equation is considered without losing generality:

$$f(x,t) + N[u(x,t);\lambda] = 0, \qquad x \in \Omega, t \in \mathrm{T} \tag{3-3}$$

with boundary conditions and initial conditions:

$$B[u(0,t),0,t] = 0 \ \ on \ \ \Omega \tag{3-4}$$

$$I[u(x,0),x,0] = 0 \ \ on \ \ \mathrm{T} \tag{3-5}$$

where $f(x,t)$ is a function of spatial variable x in the domain Ω and temporal variable t in the time domain T; $u(x,t)$ is the solution; $N(\cdot)$ is a nonlinear differential operator parameterized by λ; $B(\cdot)$ and $I(\cdot)$ are operators for boundary and initial conditions.

In forward problems, which are defined as determining the effects or outputs of a system given known causes or inputs and mathematical model that governing the system's behavior, the parameter λ is known. Figure 3-3 depicts the vanilla PINN framework for solving forward problems.

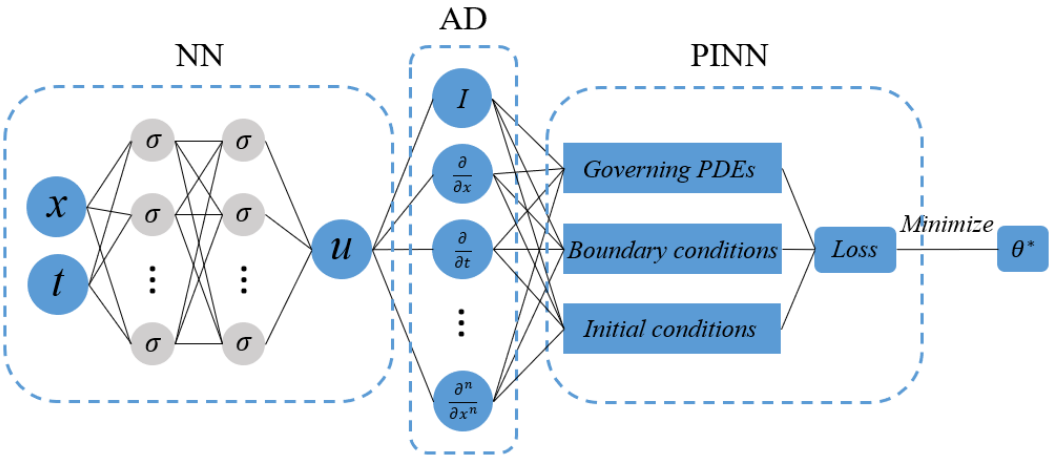


Figure 3-3 A diagram of vanilla PINN framework for forward problems

The total physics-informed loss function L_{total} for the forward problem usually contains several penalty terms for the governing equation, boundary conditions, and initial conditions. Here, the mean square error (*MSE*) is used as the metric to quantify residuals between predicted values by PINN and exact values. The residuals are computed on collocation points sampled within the domain and on the boundary. There are different sampling methods for collocation points, but random uniform sampling is typically used to generate dense sampling of collocation points over the entire domain. Besides, since there is no labeled data involved in vanilla PINN when solving forward problems, the *MSE*s can be computed by substituting the predicted values into the governing equation and each condition that should be satisfied. Consequently, L_{total} can be formulated in the following expression:

$$L_{total} = w_f \cdot MSE_f + w_b \cdot MSE_b + w_i \cdot MSE_i \tag{3-6}$$

where

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(x_i^f, t_i^f\right) + N\left[u_{pred}\left(x_i^f, t_i^f; \theta\right)\right]\right|^2 \tag{3-7}$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|B\left(u_{pred}\left(0, t_i^b; \theta\right), 0, t_i^b\right)\right|^2 \tag{3-8}$$

$$MSE_i = \frac{1}{N_i}\sum_{i=1}^{N_i}\left|I\left(u_{pred}\left(x_i^{ini}, 0; \theta\right), x_i^{ini}, 0\right)\right|^2 \tag{3-9}$$

in which MSE_f, MSE_b, and MSE_i represent the *MSE*s of residuals of the governing equation, boundary conditions, and initial conditions, respectively; w_f, w_b, and w_i are the weights of the three penalty terms; N_f denotes the number of collocation points $\left(x_i^f, t_i^f\right)$ randomly picked in the governing equation's domain; N_b denotes the number of collocation points $(0, t_i^b)$ of the boundary conditions; N_i denotes the number of collocation points $(x_i^{ini}, 0)$ of

the initial conditions; $u_{pred}(x_i^f, t_i^f; \theta)$ is the output, i.e., the predicted value, of the FCFNN; θ is the trainable parameters of the FCFNN.

The vanilla PINN is also competent for solving inverse problems, which are defined as determining the causes such as model parameters or inputs of a system from observed effects. Under vanilla PINN framework, the inverse problems aim to identify λ with few measurement datasets. In this case, the unknown parameter λ is treated as a trainable parameter and is trained together with the FCFNN parameters θ. Figure 3-4 illustrates the vanilla PINN framework for addressing inverse problems, in which the boundary and initial conditions are not considered in the training process because it is oftentimes difficult to get exact boundary and initial conditions in inverse problems. Therefore, only the governing equations and some measured data are available.

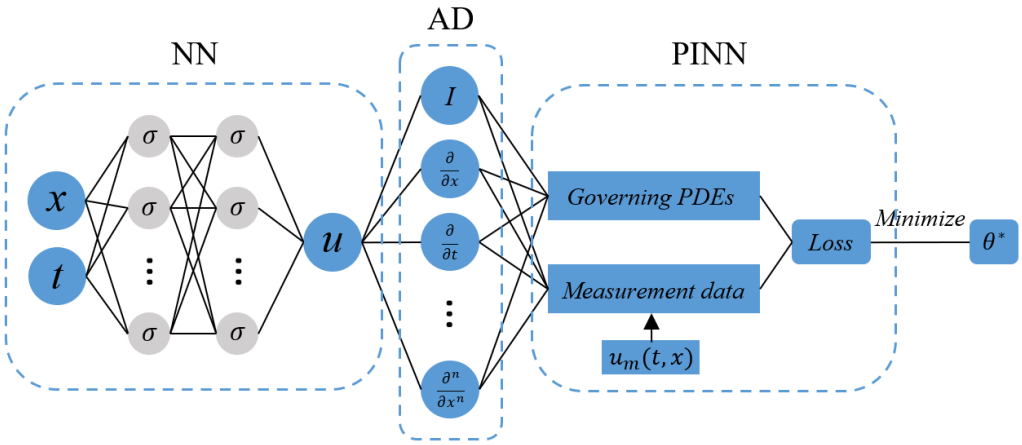


Figure 3-4 A diagram of vanilla PINN framework for inverse problems

According to the framework shown in Figure 3-4, the total physics-informed loss function L_{total} for the inverse problem only contains two penalty terms for the governing equation and measurement data. Here, *MSE*s are adopted again to calculate the residuals. Thus, the L_{total} can be expressed as follows:

$$L_{total} = w_f \cdot MSE_f + w_m \cdot MSE_m \qquad (3-10)$$

where

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(x_i^f, t_i^f\right) + N\left[u_{pred}\left(x_i^f, t_i^f; \lambda; \theta\right); \lambda\right]\right|^2 \qquad (3-11)$$

$$MSE_m = \frac{1}{N_m}\sum_{i=1}^{N_m}\left|u_{pred}(x_i^m, t_i^m; \lambda; \theta) - u_m(x_i^m, t_i^m)\right|^2 \qquad (3-12)$$

in which MSE_f and MSE_m are the *MSE*s of the governing equation and the measurement points' residuals, respectively; w_f and w_m are the weights of the two penalty terms; N_m is the number of measurement points (x_i^m, t_i^m); $u_{pred}(x_i^m, t_i^m; \lambda; \theta)$ is the output, i.e., the predicted value, of the FCFNN; $u_m(x_i^m, t_i^m)$ denotes the measurement values; λ is the unknown parameter in the governing equation, which is treated as a trainable parameter and can be obtained after PINN training.

3.2.3 Training of Vanilla PINN for High-Order Differential Equations

The vanilla PINN framework for solving differential equations is straightforward and easy to implement as described in Section 3.2.2. However, some studies pointed out that there are potential difficulties observed when solving high-order differential equations with vanilla PINN (Wight and Zhao 2020; Yu et al. 2022; Mattey and Ghosh 2022). For example, when dealing with high order derivatives, AD will be repeated multiple times to compute Jacobian or Hessian matrices, which amplify floating-point errors. If the AD chain is long, the errors will be accumulated as well. In this subsection, the computational difficulty of the vanilla PINN is investigated numerically. After that, a novel method incorporating auxiliary outputs of FCFNN will be introduced to address the issue in Section 3.3.

To explore the potential obstacles in the training of vanilla PINN when solving high-order differential equations, a simple sixth-order ordinary differential equation is considered as a numerical example:

$$\frac{d^6u(x)}{dx^6} = 0, \quad x \in [0,1] \tag{3-13}$$

With several specific boundary conditions, the exact solution of Equation (3-13) can be controlled as $u(x) = x^5$. The vanilla PINN framework is used to solve this problem. The FCFNN contains 4 hidden layers with 40 neurons in each layer. The commonly used activation functions such as $tanh(\cdot)$, $sigmoid(\cdot)$, and $sin(\cdot)$ are tested respectively. Several popular optimizers in vanilla PINN framework such as SGD, Adam, and L-BFGS are selected to optimize the loss function respectively. The total physics-informed loss function for this problem is as follows:

$$L_{total} = MSE_f + MSE_i \tag{3-14}$$

where

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^6 u_{pred}(x_i^f;\theta)}{\partial x^6}\right|^2 \tag{3-15}$$

All combinations of above-mentioned activation functions and optimizers are tested, but all of them fail to obtain solutions of satisfied accuracy to this equation. They may fail due to two challenges: the difficulty of convergence caused by the gradient exploding and the notable computational error resulting from the gradient vanishing (Sun 2020).

The potential reasons for vanilla PINN to fall into gradient exploding or vanishing when solving such a simple sixth-order differential equation are worthy exploring, and they may

provide insight into feasible solutions to these problems. Therefore, the training dynamic of the vanilla PINN is analyzed from the perspective of backpropagation. In order to simplify the symbol derivation process, a simple single-layer FCFNN is adopted to solve the sixth-order ODE for the demonstration purpose. The hidden layer of the FCFNN has only two neurons, and the activation function used is $tanh(\cdot)$. The specific symbols are shown in Figure 3-5.

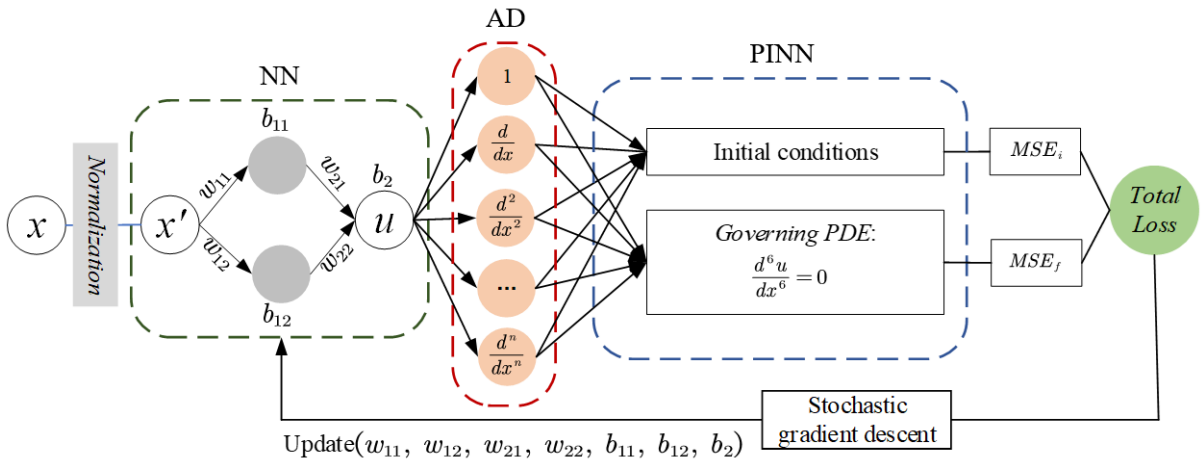


Figure 3-5 The vanilla PINN framework with a single-layer FCFNN

The forward propagation of the FCFNN can be expressed as:

$$u(x;\theta) = \boldsymbol{w_2} \cdot \sigma(\boldsymbol{w_1} \cdot x + \boldsymbol{b_1}) + b_2 \qquad (3-16)$$

where $\theta = (\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{b_1}, b_2)$, $\boldsymbol{w_1} = (w_{11}, w_{12})$, and $\boldsymbol{b_1} = (b_{11}, b_{12})$ are the weights and biases between the input layer and the hidden layer, respectively; $\boldsymbol{w_2} = (w_{21}, w_{22})$ and b_2 are the weights and bias between the hidden layer and the output layer, respectively. The *n*-th order derivative of $u(x)$ with respect to x is calculated as:

$$\frac{\partial^n u(x;\theta)}{\partial x^n} = \left(\boldsymbol{w_2} \circ \boldsymbol{w_1}^{T^n}\right) \cdot \frac{\partial^n \sigma(\boldsymbol{w_1} \cdot x + \boldsymbol{b_1})}{\partial x^n} \qquad (3-17)$$

where $\circ$ is the Hadamard product.

According to Equation (3-17), it is known that as the differential order of $u(x;\theta)$ increases, there are two changes in the derivative of $u(x;\theta)$: one is that the power of $\boldsymbol{w}_1^T$ is increasing, i.e., $\frac{\partial^n u(x;\theta)}{\partial x^n} = O(\boldsymbol{w}_1^n)$; the other is that the differential order of the activation function $\sigma(\cdot)$ is also increasing. The first step is to analyze the differential penalty term [Equation (3-15)] in the loss function. It is obvious that MSE_f is the 12-th order of $\boldsymbol{w}_1$, i.e., $MSE_f = O(\boldsymbol{w}_1^{12})$. As the activation function $\sigma(\cdot)$ is a function of w_{11}, if w_{11} is updated by the gradient of the MSE_f, then the $grad$ is $\frac{\partial MSE_f}{\partial w_{11}} = O(w_{11}^{12})$. Because the $grad$ of w_{11} is the high-order power of w_{11} itself, a terrible feedback mechanism is formed. If $|w_{11}| > 1$, the $grad$ is most likely to be a large number. According to the gradient descent method, the $|w_{11}|$ will further increase, and its $grad$ will be further enlarged. Finally, the vanilla PINN will fail to converge, which leads to the gradient exploding. If $|w_{11}| < 1$, the $grad$ is most likely to be a small number, thus the $|w_{11}|$ will not change significantly, and its $grad$ will keep a small number. Finally, w_{11} is still close to its initial value and the algorithm fails to converge, which leads to the gradient vanishing.

A numerical test is performed on the $\frac{\partial MSE_f}{\partial w_{11}}$ to verify the above statement. The $\frac{\partial MSE_f}{\partial w_{11}}$ involves multiple NN parameters and x. Therefore, the value of w_{11} is randomly set within $[-2, 2]$ to observe its effect on the gradient. Other NN parameters are initialized within $[-1, 1]$, and x is randomly taken from $[0, 1]$. The test is repeated 2000 times, and the test results are shown in Figure 3-6. The vertical axis in Figure 3-6 is the logarithm of the absolute value of the gradient, the blue shadow is the normal update area, the red shadow is the gradient exploding area, and the green shadow is the gradient vanishing area.

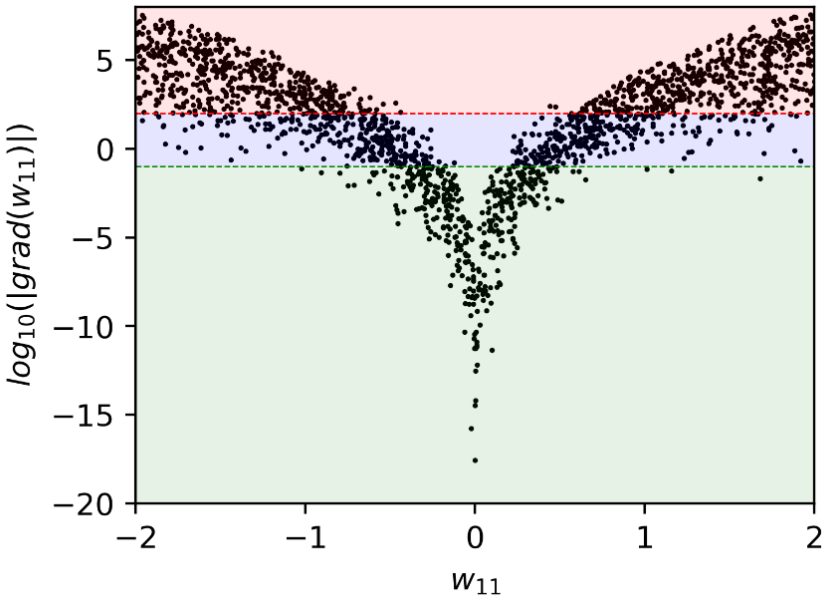


Figure 3-6 Gradients with different values of w_{11}

It can be observed from Figure 3-6 that as the $|w_{11}|$ increases, its $|grad|$ increases significantly. When the $|w_{11}| \geq 1.5$, the $|grad|$ is most likely to be greater than 100. When the $|w_{11}| \leq 0.25$, its $|grad|$ is most likely to be less than 0.1. In the case that the w_{11} is updated using the gradient descent method with a learning rate of 0.01, the initial $w_{11} = 0.5$, and $grad = 200$, after updating $w_{11} = -1.5$, the re-calculated $|grad|$ can be even larger. When the initial $w_{11} = 0.5$ and $grad = 0.1$, after updating $w_{11} = 0.499$, the w_{11} cannot change significantly. For a learning rate of 0.01, the $|grad| \in [0.1, 100]$ can be considered as the range, within which w_{11} can be updated in a normal way, which is shaded in blue in Figure 3-6. The $|grad| > 100$ is prone to gradient exploding, while $|grad| < 0.1$ is prone to gradient vanishing, as shown in red and green shadows in Figure 3-6, respectively.

Although problems like gradient exploding and gradient vanishing occur in solving high-order differential equations, vanilla PINN has been successfully applied to solve many second-order differential equation problems, such as Poisson's equation (Shin et al. 2020; Yang et al.

2021), wave equation (Wang et al. 2022; Moseley et al. 2020), and Burgers equation (Shukla et al. 2021). The difference between the gradient dynamics in the vanilla PINN framework for high-order and low-order (e.g. second-order) differential equation problems should be further studied. The above numerical test is conducted again by changing the sixth-order equation to a second-order equation to illustrate the training dynamic.

According to the above analysis, it is known that $\frac{\partial^2 u(x,\theta)}{\partial x^2} = O(\boldsymbol{w}_1^2)$, and $grad = \frac{\partial MSE_f}{\partial w_{11}} = O(w_{11}^4)$. The test results of the second-order differential equation are illustrated in Figure 3-7. It can be found that the gradients are mainly distributed in $[10^{-3}, 10]$. If the learning rate is set to be 0.1, most gradients fall within the normal update range, thus w_{11} can be updated efficiently, and the training can be avoided falling into gradient exploding or gradient vanishing.

Next, the situation for NN regression, which is a task without AD, is also investigated. The same numerical test is performed again, and the test results are plotted in Figure 3-8. It can be observed from the results that the gradients are mainly distributed in $[10^{-3}, 10]$, and the distribution of the gradient remains unchanged under different w_{11}. If the learning rate is set as 0.1, w_{11} can be updated efficiently. From the above analysis, it can be concluded that the higher the order of derivatives is, the more obvious change in the distributions of the gradients of the NN parameters occur. For a fixed learning rate, the gradient descent method is more likely to make the NN model fall into gradient exploding or gradient vanishing.

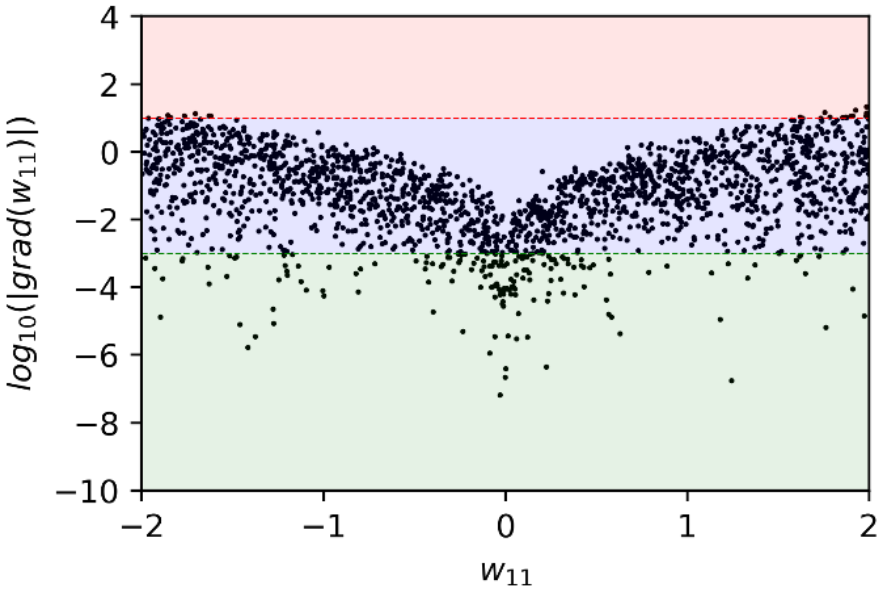


Figure 3-7 Gradients with different values of w_{11} for the second-order differential equation

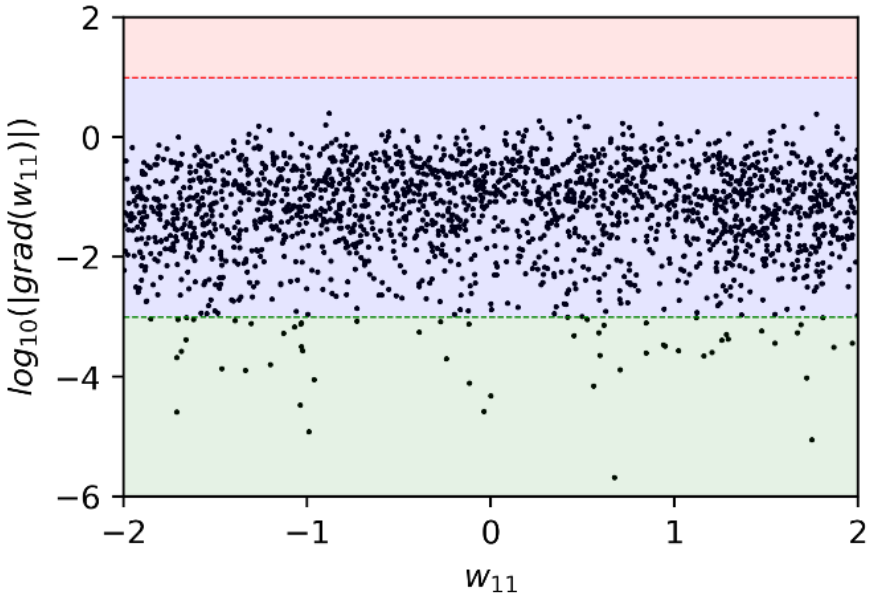


Figure 3-8 Gradients with different values of w_{11} for the neural network regression

The analysis above is based on a very simple NN architecture with only 1 hidden layer and 2 neurons in each hidden layer. The situation may change if the width or depth of the NN is increased, different activation functions are adopted, or the optimizer is varied. Next, different NN settings are investigated.

First, the network architecture with increased width of 6 neurons is tested. According to the above analysis result, $\frac{\partial^n u(x,\theta)}{\partial x^n} = O(\boldsymbol{w}_1^n)$, it is known that the gradient $grad$ of w_{11} equals to $\frac{\partial MSE_f}{\partial w_{11}} = O(w_{11}^{12})$ for a wider network architecture. Therefore, the order of $grad$ is neither increased nor decreased and the vanilla PINN is still prone to gradient vanishing or gradient exploding. Although increasing the width of the hidden layer is not applicable to accurately solve high-order equations, it is helpful to improve the model's ability to express complex functions.

Then, the effect of increasing the depth of the NN, i.e., the number of hidden layers, is investigated. The symbolic calculation results of multi-layer NNs with multiple neurons are very complicated, which bring difficulties to theoretical analysis. In order to simplify the symbolic derivation, the NN with 4 hidden layers and 1 neuron in each layer is used to perform the test. Similarly, the $grad$ is calculated to be $\frac{\partial MSE_f}{\partial w_1}$ and results show that $grad = O(w_1^{12}) * O(w_2^{12}) * O(w_3^{12})$, indicating a worse situation. As long as one of $|w_1|$, $|w_2|$ and $|w_3|$ is greater than 1, the model is easy to face gradient exploding. Thus, it can be concluded that increasing the depth of the model is not an ideal solution to address the gradient exploding and gradient vanishing problems.

Another alternative choice is to change the activation function. Based on Equation (3-17), it is known that the $grad$ is also related to the high-order derivative of the activation function. However, it is found that these activation functions have very limited influence on the order of gradient under the situation of the gradient exploding or gradient vanishing.

The analysis above is based on the gradient descent method, which is the simplest among optimization methods. The influence of different choices of optimizers on the updating of NN

parameters is studied. Several optimizers commonly used in vanilla PINN are explored.

First, SGD without momentum is employed. In the case of full batches, SGD is the same as the gradient descent method. Since the mini-batch is equally distributed with full batches, the number of samples in the training batch has little impact on the distribution of the gradient of w_{11} presented in Figure 3-6. If the SGD with momentum is used, the momentum v is updated to be $\alpha v - \epsilon * grad$, in which the α is the momentum parameter, and ϵ is the learning rate. The parameter w_{11} is updated to be $w_{11} + v$. It can be noted that if $|grad|$ is a large number, the momentum term αv can be ignored. If $|grad|$ is a small number and v is initialized as zero, v remains a small number, and w_{11} will not change significantly as well. Thus, the SGD optimizer is not suitable for tackling gradient exploding and gradient vanishing problems.

Adam (Kingma and Ba 2017) is another momentum-based optimizer. With Adam optimizer, w_{11} is updated to be $w_{11} - \epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$, in which, the first-order moment $\hat{s}$ is the first order of gradient, i.e. $\hat{s} = O(grad)$, and the second-order moment $\hat{r}$ is the second order of gradient, i.e. $\hat{r} = O(grad^2)$. Due to the stabilization of $\hat{r}$ to $\hat{s}$, $\frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$ is the lower-order of w_{11}. Consequently, Adam can alleviate the gradient exploding and gradient vanishing in the vanilla PINN framework.

In addition to the above optimizers, second-order gradient optimizers such as the quasi-Newton method-BFGS (Buckley 1978; Liu and Nocedal 1989) can also be utilized. To avoid computing the complicated Hessian matrix, only the gradient of w_{11} is considered and the second-order gradient $\frac{\partial^2 MSE_f}{\partial {w_{11}}^2}$ is used to replace the Hessian matrix. According to the previous analysis, the first-order gradient $grad = O(w_{11}^{12})$ and the calculation result show that the

second-order gradient $\frac{\partial^2 MSE_f}{\partial {w_{11}}^2}$ is $O(w_{11}^{12})$ as well. Applying the updating rule of BFGS to w_{11}, it becomes $w_{11} - \epsilon \cdot (\frac{\partial^2 MSE_f}{\partial {w_{11}}^2})^{-1} \cdot grad$, in which, ϵ is the adaptive learning rate from the line search. It is obvious that the second-order gradient can reduce the order of $(\frac{\partial^2 MSE_f}{\partial {w_{11}}^2})^{-1} \cdot grad$. Thus, the BFGS method can effectively alleviate the gradient exploding and gradient vanishing caused by high-order of w_{11}. In addition, since the linear search is used in BFGS to iteratively find the optimal learning rate, the adaptive learning rate is capable of preventing the gradient exploding when the learning rate is too large and avoiding the gradient vanishing when the learning rate is too small.

To summarize, altering the architecture of the NN or choosing different activation functions shows limited ability to address the computational difficulties posed by high-order differential equations. Although the performance of vanilla PINN for solving high-order differential equations can be effectively improved by changing the optimizer, the problems of gradient exploding and gradient vanishing cannot be fundamentally eliminated. However, Figures 3-7 and 3-8 show a feasible solution. If the highest order of derivatives in the differential equation can be reduced, the order of AD in the vanilla PINN framework can be reduced subsequently, leading to a fundamental settlement of the challenging issue. In the following subsection, a novel solution by training with auxiliary outputs for reducing the highest order of derivatives will be proposed.

3.3 Methods

To address the challenge of computing high-order derivatives in vanilla PINN, a novel method that trains the FCFNN with auxiliary outputs to reduce the highest order of derivatives

in the governing equation is introduced in this section.

3.3.1 Training with Auxiliary Outputs for Reducing the Order

Usually, the FCFNN in the vanilla PINN only generates the primary output, i.e. target output, leading to high-order AD computation when high-order derivatives are involved. The rationale of the proposed method is to define additional outputs termed as auxiliary outputs to represent multiple order derivatives of the primary output and add an extra penalty term in the loss function in compliance with the connections between the auxiliary outputs and the primary output. The extra penalty term is used to constrain the auxiliary outputs to satisfy the governing equation. The overall schematic of the proposed method for solving forward problems is illustrated in Figure 3-9, in which u is the primary output, while w is the auxiliary output. To adjust the framework to deal with inverse problems, only a few modifications are needed, e.g. adding measurement data and incorporating a penalty term for measurement data in the total loss function.

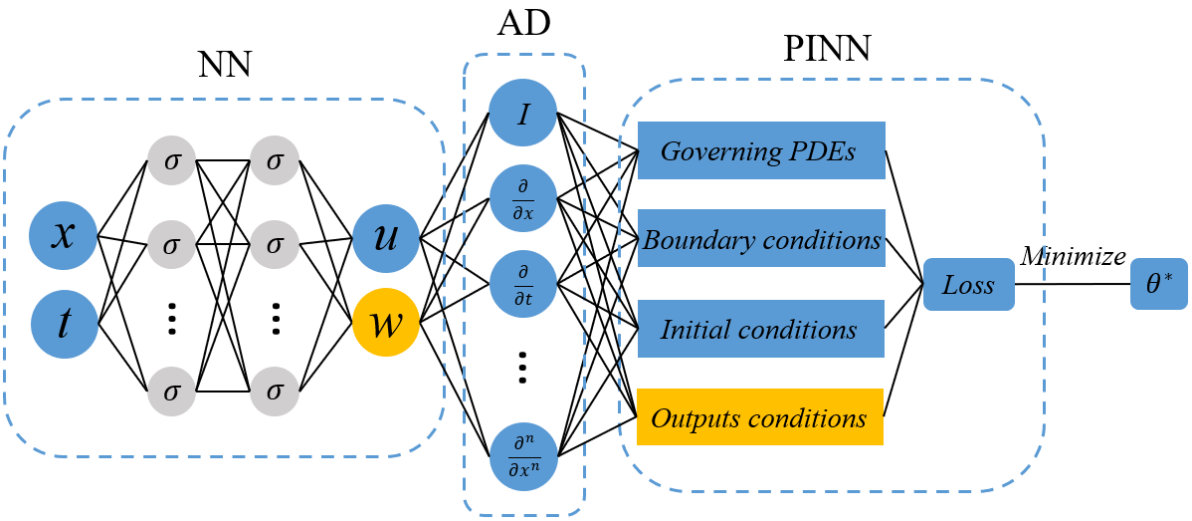


Figure 3-9 A framework of training with auxiliary outputs for forward problems

The core component of the proposed framework is still an FCFNN, which is employed to not only approximate the solution $u(x,t)$ of the governing equation but also generate auxiliary outputs such as $w(x,t)$ as shown in Figure 3-9. By using auxiliary outputs to approximate the derivatives of $u(x,t)$, the high-order derivative of $u(x,t)$ can be decomposed into multiple low-order derivatives, so as to avoid high-order AD computation. The relationship between $u(x,t)$ and $w(x,t)$ is constrained by an extra penalty term in the physics-informed loss function. For example, the following fourth-order PDE is considered:

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial^4 u(x,t)}{\partial x^4} = f(x,t) \tag{3-18}$$

An auxiliary output $w(x,t)$ can be defined to approximate the second-order derivative of $u(x,t)$. The relationship between $w(x,t)$ and $u(x,t)$ can be expressed as:

$$w(x,t) = \frac{\partial^2 u(x,t)}{\partial x^2} \tag{3-19}$$

Therefore, the extra penalty term can be computed as:

$$MSE_a = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2 u_{pred}(x_i^f, t_i^f;\theta)}{\partial x^2} - w_{pred}(x_i^f, t_i^f;\theta)\right|^2 \tag{3-20}$$

With the help of $w(x,t)$, the fourth-order PDE can be transformed as:

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial^2 w(x,t)}{\partial x^2} = f(x,t) \tag{3-21}$$

in which, the highest order is reduced from fourth-order to second-order, avoiding computation of the high-order derivative. Consequently, the total loss function can be expressed as:

$$L_{total} = w_f \cdot MSE_f + w_b \cdot MSE_b + w_i \cdot MSE_i + w_a \cdot MSE_a \tag{3-22}$$

where MSE_b, MSE_i, and MSE_a are given by Equations (3-8), (3-9) and (3-20). MSE_f can be

obtained from Equation (3-23) as:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial u_{pred}(x_i^f, t_i^f; \theta)}{\partial t} + \frac{\partial^2 w_{pred}(x_i^f, t_i^f; \theta)}{\partial x^2} - f(x_i^f, t_i^f)\right|^2 \quad (3-23)$$

According to the analysis in Section 3.2.3, the second-order gradient optimizer can effectively avoid the PINN from falling into gradient vanishing and gradient exploding. Therefore, the L-BFGS can be utilized to minimize the total loss function to make the predicted values of the proposed method gradually approach the exact solution. Theoretically, since the optimization of the FCFNN is a non-convex problem, there is no guarantee that the loss function will finally converge to the global optimal. Nonetheless, according to the results of numerical examples, the accuracy of the proposed method can reach a satisfactory level.

3.3.2 Hyperparameter Study

The proposed framework decomposes the high-order governing differential equation into a set of low-order differential equations by employing auxiliary outputs of FCFNN. It can be noticed that there are different auxiliary output schemes for the decomposition of the high-order differential equation. For example, a fourth-order PDE can be divided into two second-order PDEs or one first-order PDE and one third-order PDE. Therefore, the main hyperparameter of the proposed framework is the auxiliary output scheme for the high-order governing equation. To investigate the performance of different auxiliary output schemes, a hyperparameter study of several cases with various schemes is carried out in this subsection. Since there are countless differential equations of various orders, it is impossible to explore the best setting schemes for all cases. A typical description of PDEs with various orders is adopted as test cases, which can be expressed as follows:

$$\frac{\partial^n u(x,t)}{\partial x^n} = \frac{\partial u(x,t)}{\partial t} \qquad x \in [0,1], t \in [0,1] \qquad (3-24)$$

where n is an integer representing the highest order of the PDE. Equation (3-24) can express a series of PDEs with different orders because n varies. These PDEs consist of only two partial differential operators that are necessary components in all PDEs.

PDEs from second-order to sixth-order with well-defined initial and boundary conditions are solved using both vanilla PINN and the proposed framework under different auxiliary output schemes. The relative L_2 errors of different cases calculated by Equation (3-36) are presented in Table 3-1.

Table 3-1 Relative L_2 errors of different auxiliary output schemes

Orders n	Vanilla PINN	Different auxiliary output schemes for the proposed framework						
2	0.382%	1+1						
		0.208%						
3	0.112%	2+1	1+2	1+1+1				
		0.043%	0.064%	0.061%				
4	0.060%	3+1	2+2	1+3	2+1+1	1+2+1	1+1+2	1+1+1+1
		0.032%	0.010%	0.048%	0.022%	0.135%	0.103%	0.868%
5	0.198%	4+1	3+2	2+3	1+4	3+1+1	2+2+1	1+1+1+1+1
		0.061%	0.017%	0.025%	0.040%	0.102%	0.107%	0.066%
6	Fail	5+1	4+2	3+3	2+4	1+5	2+2+2	1+1+1+1+1+1
		0.108%	0.028%	0.017%	0.073%	0.078%	0.025%	0.115%

In Table 3-1, n is the highest order of the PDEs, and the vanilla PINN column contains the relative L_2 errors of the solutions obtained by the vanilla PINN. In the different auxiliary output schemes column, the first row of each order shows the different auxiliary output schemes. For example, the 4+2 scheme of the sixth-order PDE means that the FCFNN in the proposed framework has two outputs: one is the primary output $u(x, t)$, the other is the auxiliary output $v(x, t)$. The auxiliary output $v(x, t)$ is the fourth-order derivative of the primary output $u(x, t)$, while the sixth-order derivative in the PDE is transformed into the second-order derivative of the auxiliary output $v(x, t)$. Consequently, the original sixth-order PDE is decomposed into the combination of a fourth-order and a second-order equations. The relationship between $u(x, t)$ and $v(x, t)$ is constrained by a penalty term according to $v(x, t) = \frac{\partial^4 u(x,t)}{\partial x^4}$.

All auxiliary output schemes of PDEs with different orders are tested, while only some of the results of them are selected and presented in Table 3-1. It can be found that vanilla PINN successfully solves PDEs from second-order to fifth-order but fails in the sixth-order PDE problem. In most test cases, the strategy of using auxiliary outputs can effectively improve the computational accuracy of the vanilla PINN. The higher the order of the PDE problems, the better performance for improvement of the solution's accuracy can be achieved by the proposed framework. It also can be observed that the optimal auxiliary output scheme always has only one auxiliary output. The reason is that more auxiliary outputs require more penalty terms to constrain their relationships. How to maintain the gradient balance between multiple penalty terms in the loss function during the training of FCFNN is also a difficult problem to address (Wang et al. 2021a). The principle of the optimal scheme is to minimize the order of the PDE

with the fewest auxiliary outputs. According to the relative L_2 error results calculated for different auxiliary output schemes, the recommended schemes for second-order to sixth-order PDEs are summarized in Table 3-2.

Table 3-2 Recommended auxiliary output scheme for PDEs with different orders

The highest order of derivatives	Recommended auxiliary output scheme
2	Primary output $u(x,t)$ and an auxiliary output $v(x,t) = u_x(x,t)$
3	Primary output $u(x,t)$ and an auxiliary output $v(x,t) = u_{xx}(x,t)$
4	Primary output $u(x,t)$ and an auxiliary output $v(x,t) = u_{xx}(x,t)$
5	Primary output $u(x,t)$ and an auxiliary output $v(x,t) = u_{xxx}(x,t)$
6	Primary output $u(x,t)$ and an auxiliary output $v(x,t) = u_{xxx}(x,t)$

3.4 Numerical Case Studies

In this section, three numerical case studies, including a Korteweg-de Vries (KdV) equation, a Euler-Bernoulli equation, and a sixth-order PDE, are conducted using the strategy of auxiliary outputs to validate the performance of the proposed framework for solving high-order differential equations.

3.4.1 Korteweg-de Vries Equation

In the first numerical case study, a third-order KdV equation is solved by the proposed framework. The KdV equation describes the physical phenomenon of waves on shallow water surfaces. It plays an important role in various fields of applied science and engineering, such as hydrodynamics, water waves, and quantum theory. In the research field of PINN, the KdV equation is treated as a benchmark problem involving high-order derivatives, related studies can be found in (Raissi et al. 2019a; Jagtap et al. 2020; Wang et al. 2022).

The governing equation, nonlinear KdV equation parameterized by λ_1 and λ_2, is expressed as:

$$\frac{\partial u}{\partial t} + \lambda_1 u \frac{\partial u}{\partial x} + \lambda_2 \frac{\partial^3 u}{\partial x^3} = 0 \qquad x \in [-1,1],\ t \in [0,1] \tag{3-25}$$

The initial condition is considered as $u(x,0) = \cos(\pi x)$, the boundary conditions are considered periodic, λ_1 is taken as 1, and λ_2 is set as 0.0025. According to the suggested auxiliary output scheme in Table 3-2, one auxiliary output $v(x,t)$ is defined to represent the second-order derivative of $u(x,t)$. Then, the governing equation can be reformulated as:

$$\frac{\partial u}{\partial t} + \lambda_1 u \frac{\partial u}{\partial x} + \lambda_2 \frac{\partial v}{\partial x} = 0 \tag{3-26}$$

$$v = \frac{\partial^2 u}{\partial x^2} \tag{3-27}$$

In the proposed framework, a two-output FCFNN is built to approximate the primary output $u(x,t)$ and the auxiliary output $v(x,t)$, simultaneously.

For the loss function, *MSE* is used to quantify the residuals of the governing equation and

other conditions on collocation points. The penalty term for the governing equation can be formulated as:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial u_{pred}(x_i^f, t_i^f;\theta)}{\partial t} + \lambda_1 u_{pred}(x_i^f, t_i^f;\theta)\frac{\partial u_{pred}(x_i^f, t_i^f;\theta)}{\partial x} + \lambda_2 \frac{\partial v_{pred}(x_i^f, t_i^f;\theta)}{\partial x}\right|^2 \quad (3-28)$$

To make the prediction satisfy the initial condition, a penalty term for calculating the residual of the initial condition is created as:

$$MSE_i = \frac{1}{N_i}\sum_{i=1}^{N_i}\left|u_{pred}(x_i^{ini}, 0;\theta) - cos(\pi \cdot x_i^{ini})\right|^2 \quad (3-29)$$

The periodic boundary conditions of the KdV equation are also integrated into the total loss function by a penalty term as:

$$MSE_b = MSE_{b1} + MSE_{b2} + MSE_{b3} \quad (3-30)$$

where

$$MSE_{b1} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|u_{pred}(-1, t_i^b;\theta) - u_{pred}(1, t_i^b;\theta)\right|^2 \quad (3-31)$$

$$MSE_{b2} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\frac{\partial u_{pred}(-1, t_i^b;\theta)}{\partial x} - \frac{\partial u_{pred}(1, t_i^b;\theta)}{\partial x}\right|^2 \quad (3-32)$$

$$MSE_{b3} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\frac{\partial^2 u_{pred}(-1, t_i^b;\theta)}{\partial x^2} - \frac{\partial^2 u_{pred}(1, t_i^b;\theta)}{\partial x^2}\right|^2 \quad (3-33)$$

The relationship between the primary output and the auxiliary output is constrained by an additional penalty term according to Equation (3-27), which is calculated as:

$$MSE_a = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2 u_{pred}(x_i^f, t_i^f; \theta)}{\partial x^2} - v_{pred}(x_i^f, t_i^f; \theta)\right|^2 \quad (3-34)$$

Then, the total loss function can be formed by the weighted summation of the above four *MSE*s as:

$$L_{total} = w_f \cdot MSE_f + w_b \cdot MSE_b + w_i \cdot MSE_i + w_a \cdot MSE_a \quad (3-35)$$

where w_f, w_b, w_i, and w_a are the weights of each penalty term. In this case, they are set as 1. The four penalty terms can be optimized simultaneously during the training.

In this case, the collocation points set includes $N_i = 200$ points randomly sampled in the time domain for the initial condition, $N_b = 200$ points randomly sampled on the boundary for boundary conditions, and $N_f = 2000$ points randomly sampled inside the entire domain for the governing equation and the output relationship.

The FCFNN in the proposed framework consists of 4 hidden layers and 40 neurons in each layer. The activation function used is Tanh. The L-BFGS is employed to minimize the loss function. The exact benchmark solution is referenced from (Raissi et al. 2019a). The relative L_2 error is chosen as the metric to evaluate the computational accuracy of the solution obtained by the proposed framework. The relative L_2 error can be calculated by:

$$Relative\ L_2\ error = \frac{\sqrt{\sum_{i=1}^{M}\left|u_{pred}^i - u_{exact}^i\right|^2}}{\sqrt{\sum_{i=1}^{M}\left|u_{exact}^i\right|^2}} \quad (3-36)$$

To fairly compare the accuracy of the vanilla PINN the proposed framework, a vanilla PINN model with the same FCFNN architecture is also built to solve the problem.

The predicted solutions by vanilla PINN and the proposed framework after 2000 epochs

are illustrated in Figures 3-10 and 3-11, respectively. Also, the absolute error distributions of the vanilla PINN and the proposed framework compared to the benchmark solution are shown in Figures 3-12 and 3-13, respectively.

The relative L_2 error of the vanilla PINN is 2.899%, indicating that the vanilla PINN can successfully solve this third-order PDE without falling into gradient exploding or gradient vanishing with the help of the L-BFGS optimizer. The performance of the proposed framework is even better, with a relative L_2 error of 0.993%. Moreover, by evaluating the error distribution, it can be noticed that the proposed framework can achieve higher accuracy of the solution than the vanilla PINN over almost the entire domain, which further shows its strong ability.

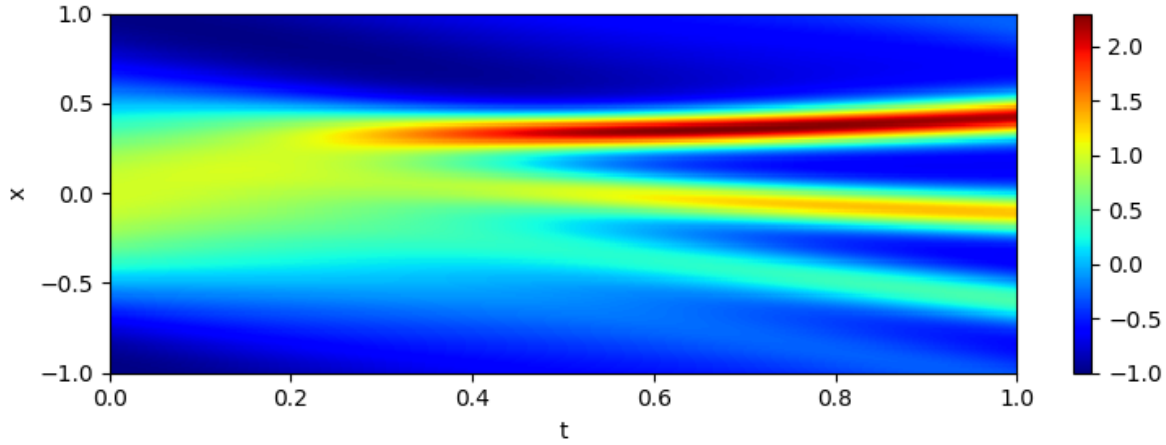


Figure 3-10 The solution of the vanilla PINN for the KdV equation

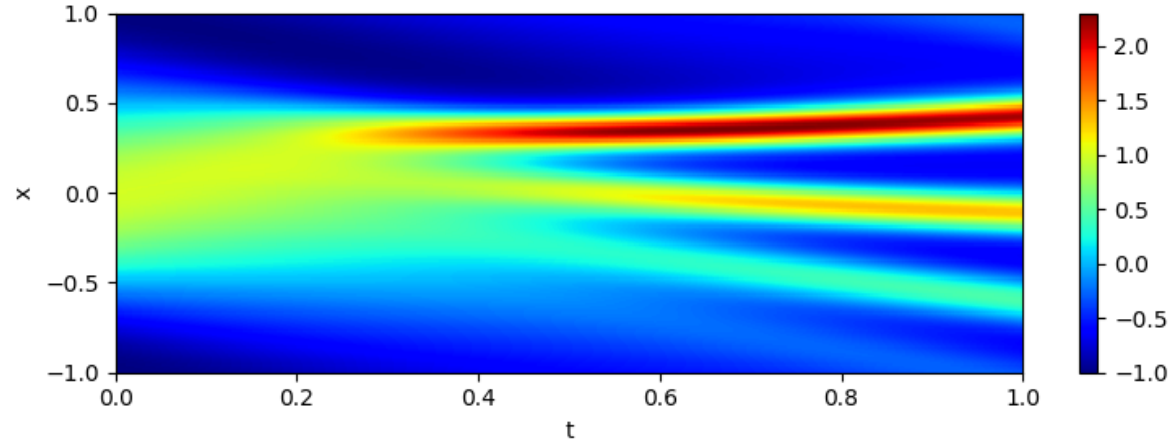


Figure 3-11 The solution of the proposed framework for the KdV equation

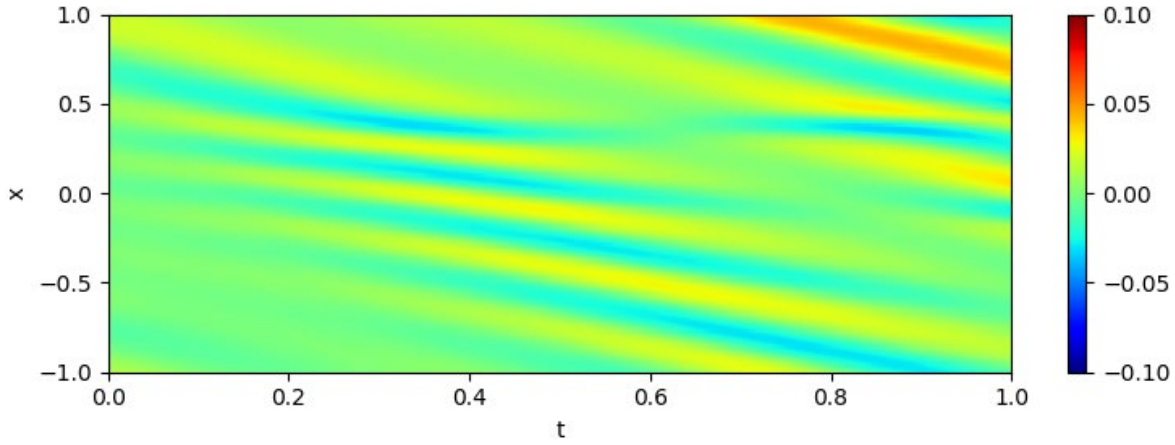


Figure 3-12 The absolute error of the vanilla PINN for the KdV equation

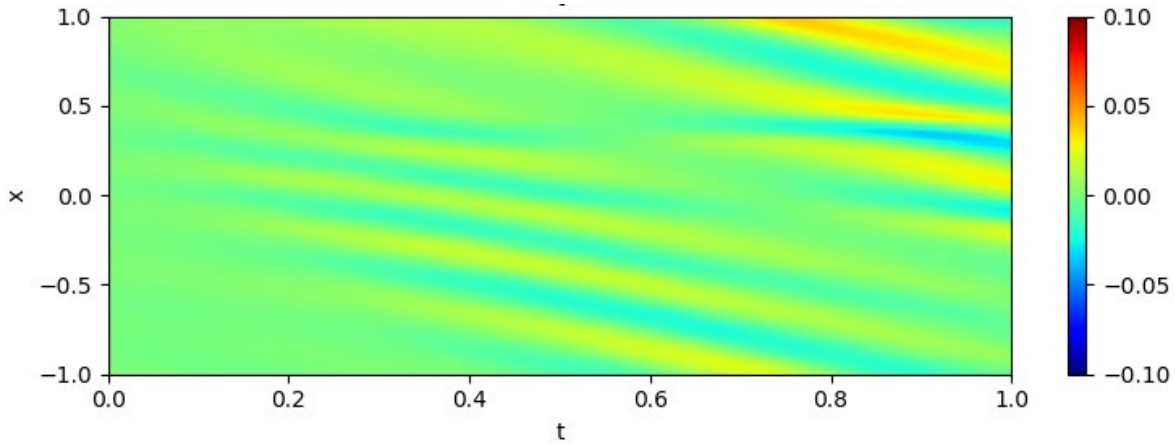


Figure 3-13 The absolute error of the proposed framework for the KdV equation

3.4.2 Euler-Bernoulli Equation for Beam Vibration

In the second numerical case study, a fourth-order PDE named Euler-Bernoulli equation for beam vibration is employed. The Euler-Bernoulli equation describes the vibration and deformation of beams in structural mechanics and is widely used in design and research in the field of structural engineering. The Euler-Bernoulli equation is expressed as:

$$EI(x)\frac{\partial^4 u}{\partial x^4} + \rho A(x)\frac{\partial^2 u}{\partial t^2} = f(x,t) \qquad x \in [0,l] \qquad (3-37)$$

where E denotes the Young's modulus, $I(x)$ represents the inertia moment determined by the shape and size of the beam's cross-section, ρ is the density, $A(x)$ is the cross-section area of the beam, and $f(x,t)$ depicts the external force on the beam. If the cross-section of the beam remains unchanged along the length, $I(x)$ and $A(x)$ are constant. In this case study, a simplified case where $EI(x) = 1$, $\rho A(x) = 1$, and $f(x,t) = 0$ is considered for simplicity.

For the boundary condition, only the left side end ($x = 0$) of the beam is fixed, while elsewhere is unconstrained. An initial load is applied to deform the beam and then cancel the load to make the beam vibrate freely. According to the suggested auxiliary output scheme in Table 3-2, one auxiliary output $v(x,t)$ is defined. Accordingly, the governing equation can be reformulated in the following form:

$$EI(x)\frac{\partial^2 v}{\partial x^2} + \rho A(x)\frac{\partial^2 u}{\partial t^2} = f(x,t) \qquad (3-38)$$

$$v = \frac{\partial^2 u}{\partial x^2} \qquad (3-39)$$

The loss function can be constructed by four penalty terms, MSE_f, MSE_b, MSE_i, and MSE_a for the governing equation, the boundary condition, the initial condition, and the relationship between primary and auxiliary outputs, respectively. They can be formulated

similarly to those in Section 3.4.1.

For the collocation points, 100 points are randomly sampled in the time domain, 200 points are randomly sampled on the boundary, and 1000 points are randomly sampled inside the entire domain.

The FCFNN architectures, activation functions, and optimizers of both the proposed framework and the vanilla PINN are the same as those for solving the KdV equation. The relative L_2 error is also used for comparison of models' performance.

The solutions by the vanilla PINN and the proposed framework after 3000 epochs are illustrated in Figures 3-14 and 3-15. The absolute errors of the vanilla PINN and the proposed framework compared to the exact solution obtained by the FEM are shown in Figures 3-16 and 3-17. The relative L_2 error of the vanilla PINN is 2.899%, while that of the proposed framework is 1.977%, showing improved computational accuracy.

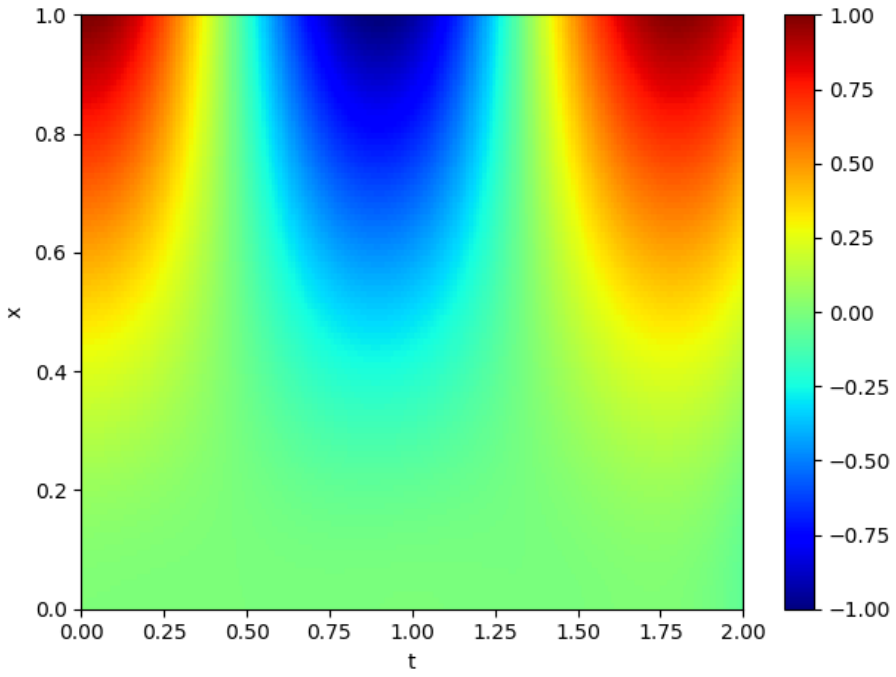


Figure 3-14 The solution of the vanilla PINN for the Euler-Bernoulli equation

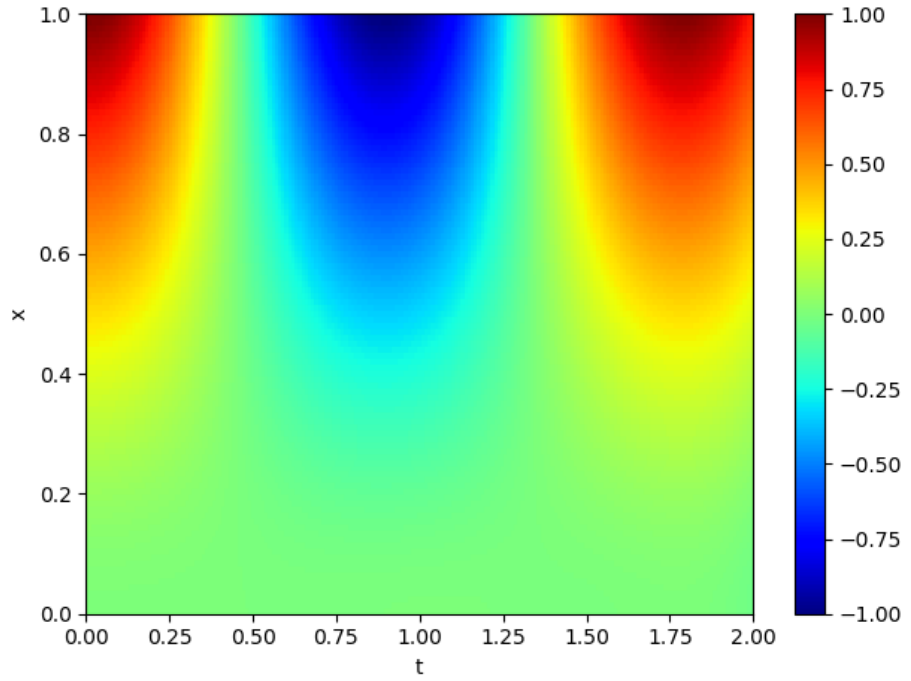


Figure 3-15 The solution of the proposed framework for the Euler-Bernoulli equation

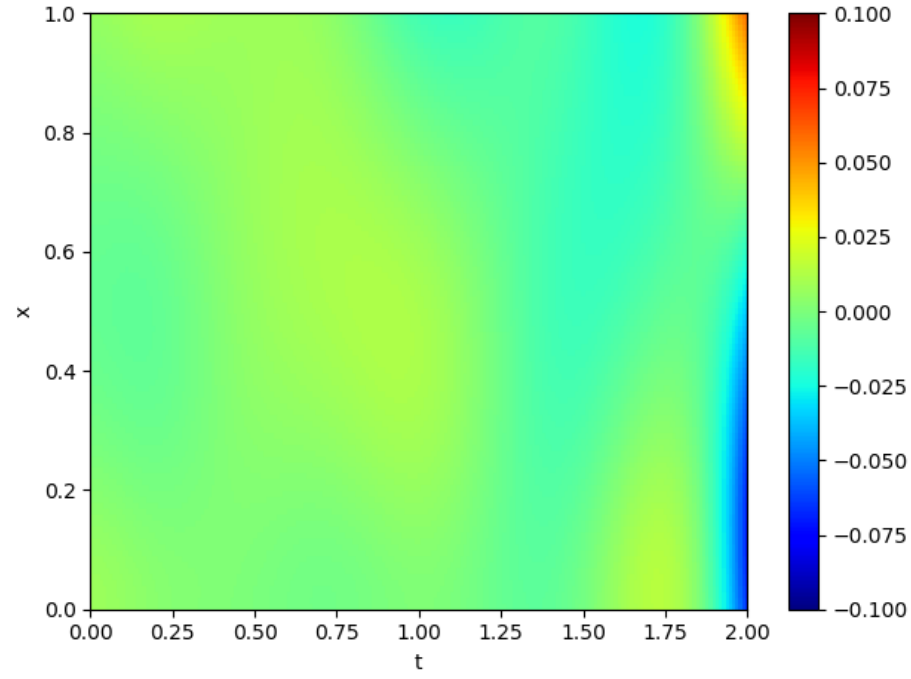


Figure 3-16 The absolute error of the vanilla PINN for the Euler-Bernoulli equation

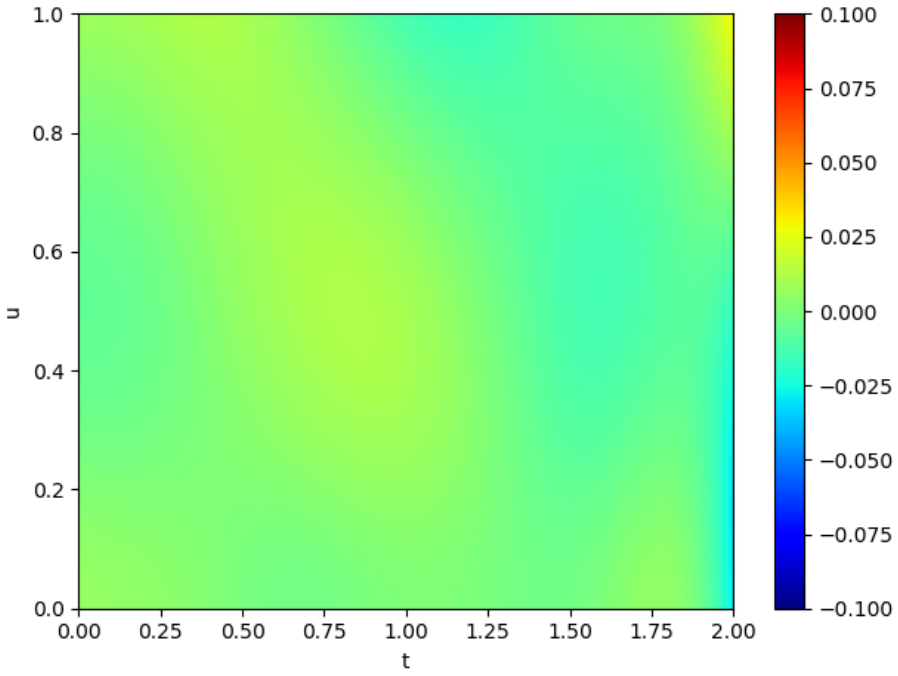


Figure 3-17 The absolute error of the proposed framework for the Euler-Bernoulli equation

3.4.3 A Sixth-Order Partial Differential Equation

In the third numerical case study, a sixth-order PDE that is more difficult to solve than the previous two examples is considered. It can be expressed as:

$$\frac{\partial u}{\partial t} + u\frac{\partial^6 u}{\partial x^6} + g(x,t) = 0 \quad x \in [0,1], t \in [0,1] \tag{3-40}$$

in which, the initial condition is set as $u(x,0) = x \cdot e^{-x}$, the boundary conditions are defined to be the Dirichlet boundary conditions, and $g(x,t) = -x(x \cdot e^{t-x} - 6e^{t-x} + 1) \cdot \mathrm{e}^{t-x}$. The exact solution of the PDE is $u(x,t) = x \cdot e^{t-x}$. According to the suggested auxiliary output scheme in Table 3-2, one auxiliary output $v(x,t)$ is defined to represent the third-order derivative of $u(x,t)$. Then, the original PDE can be transformed as:

$$\frac{\partial u}{\partial t} + u\frac{\partial^3 v}{\partial x^3} + g(x,t) = 0 \tag{3-41}$$

$$v = \frac{\partial^3 u}{\partial x^3} \tag{3-42}$$

The loss function can be formed in the same way as the previous case study. The collocation point set includes three parts: 100 points randomly sampled in the time domain, 200 points randomly sampled on the boundary, and 2000 points randomly sampled inside the entire domain. The depth and the width of the FCFNN in the vanilla PINN and the proposed method are 4 and 40, which are the same as the previous cases. Also, the activation functions and optimizers remain the same, i.e. Tanh and L-BFGS. The relative L_2 error is used again to evaluate models' performance.

The vanilla PINN fails to solve this sixth-order PDE since its loss function does not converge during training, while the proposed method successfully addresses this problem. Therefore, only the results after 2000 epochs of the proposed method are shown in Figures 3-18 and 3-19, illustrating the predicted solution and error distribution, respectively.

It can be noted that the proposed framework can obtain the solution to this high-order problem with very high accuracy, proved by a relative L_2 error of only 0.0787%. The strong ability of the proposed framework, which trains the FCFNN with auxiliary outputs, is demonstrated by solving a high-order PDE with high accuracy.

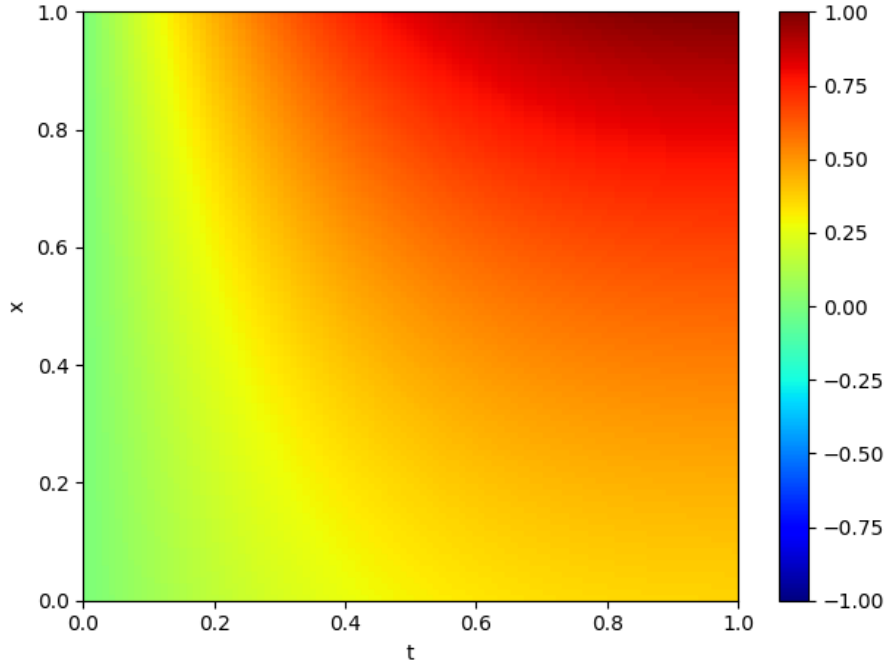


Figure 3-18 The solution of the proposed framework for the sixth-order PDE

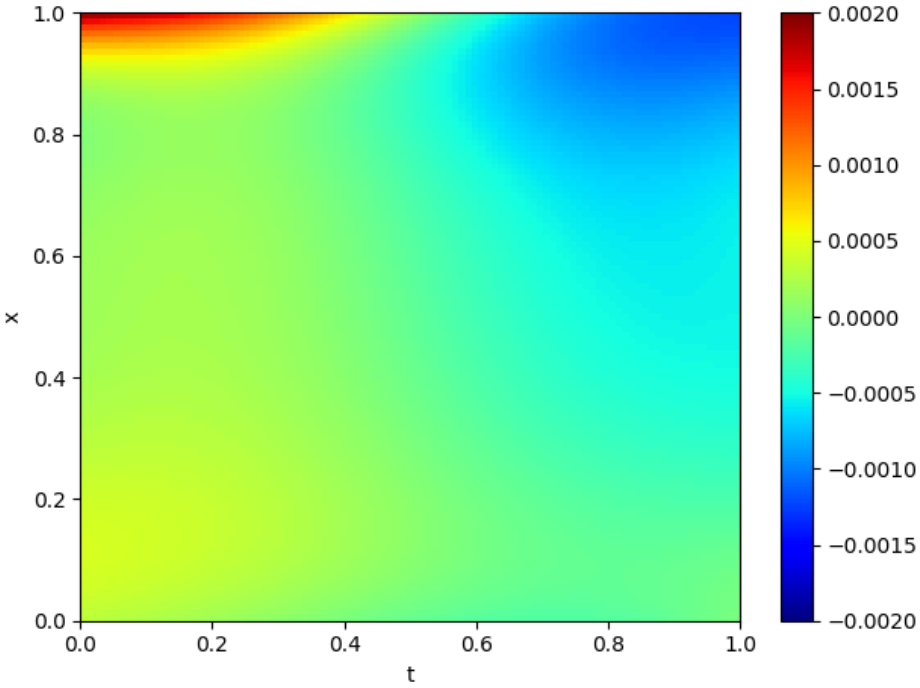


Figure 3-19 The absolute error of the proposed framework for the sixth-order PDE

3.5 Summary

In this chapter, the basic knowledge of PINN is introduced as the foundation of this thesis. Besides, the difficulty encountered when solving high order differential equations are theoretically investigated through the training dynamics of PINN. To address this difficulty, an auxiliary output-based method is proposed to avoid computing high order derivatives in governing equations, thus facilitating computation. The effectiveness of the proposed method in improving PINN's computational efficiency and accuracy is validated through several numerical case studies.

CHAPTER 4

Physics-Informed Neural Network with Modulating Functions for Hard-Embedding of Boundary Conditions

4.1 Introduction

In addition to high order derivatives in governing equations, numerous boundary conditions also hinder computational efficiency and prediction accuracy, especially near the boundary, of PINN. Traditionally, each boundary condition is incorporated through individual penalty term in the total loss function, which can slow down convergence and reduce computational efficiency. The soft constraints approach often fails to ensure boundary residuals reach zero by the end of training. To address this issue, modulating functions for hard-embedding of boundary conditions are introduced in this chapter, enabling automatic satisfaction of boundary conditions without additional penalty terms. Moreover, the auxiliary outputs introduced in Chapter 3 are also adopted to further improve PINN's performance.

For the application case, the identification of rotational stiffness of semi-rigid joints in structural engineering exemplifies the practical benefits of this approach. Semi-rigid joints, which possess rotational stiffness between that of fully rigid and ideally pinned joints, play a significant role in structural performance. Traditional modeling often assumes joints are either

fully rigid or pinned, leading to inaccuracies in FE modeling if simplified assumptions are used. PINN with the proposed method effectively addresses these limitations, allowing for accurate identification of rotational stiffness with minimal data and high noise tolerance, which is crucial for reliable structural analysis and design.

The main content of this chapter is organized as follows. In Section 4.2, the modulating functions are derived and applied to the auxiliary outputs-based framework to hard-embed boundary conditions and reduce the order of governing equations. In Section 4.3, the proposed method is validated through two numerical case studies, in which the unknown rotational stiffness of semi-rigid joints is identified. Then, the experimental validation on a single-bay steel frame is conducted in Section 4.4.

4.2 Methods

The fundamentals of vanilla PINN have been introduced in Section 3.2. In this section, an innovative method for hard-embedding boundary conditions using modulating functions is proposed to improve the vanilla PINN. Moreover, the auxiliary output-enabled training method proposed in Chapter 3 will also be employed to facilitate the hard-manner treatment of boundary conditions and reduce the order of governing equations.

4.2.1 Establishment of Modulating Functions

In the vanilla PINN framework, all boundary conditions will be satisfied by enforcing a penalty term for boundary conditions to approach zero. Nevertheless, there will be errors because the penalty term can only be close to zero but not equal to zero. Dealing with boundary conditions in the above way is called soft constraints (Lu et al. 2021b; Lu et al. 2021c). Even

in a simple structure, for example a cantilever beam (in Section 4.3.1), four boundary conditions with different orders are involved due to the high order of the governing equation. The number of boundary conditions and continuity conditions is significantly increased for multiple-member structural systems, a steel frame for instance (in Section 4.3.2), which results in high complexity of the loss functions and large errors of the boundary condition losses. To handle this problem and improve computational efficiency and accuracy, modulating functions that enables hard embedding of all boundary conditions is derived.

Consider the Euler-Bernoulli beam equation (Gere and Timoshenko 1997) that are widely used to describe the deformation of beam structures under loading:

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 u(x)}{\partial x^2}\right) = q, \qquad x \in [0, l] \tag{4-1}$$

Without losing generality, the following normalized equation (it can be easily elicited through a transformation of Equation (4-1) by scaling the x domain from $[0, l]$ to $[0, 1]$ and defining a dimensionless deflection variable) is considered:

$$\frac{\partial^4 u}{\partial x^4} + 1 = 0, \qquad x \in [0, 1] \tag{4-2}$$

with the following boundary conditions

$$u(0) = a, \quad u'(0) = b, \quad u''(1) = c, \quad u'''(1) = d \tag{4-3}$$

where the first two conditions are on the left side of the boundary, while the other two are on the right side. Because the boundary conditions are in different orders and on different sides of the domain, the hard embedding of them is complicated and not addressed.

The following five cases are discussed:

(1) In the case with only the boundary condition $u(0) = a$, define

$$v(x) = a + xu(x) \tag{4-4}$$

which satisfies $v(x) = v(0) = a$ at $x = 0$. In the PINN formulation, the FCFNN represents

the solution, i.e. $u_{pred}(x;\theta)$, to the governing equation without considering boundary conditions. Then the FCFNN output is modified to satisfy the boundary condition using the following modulating function:

$$v_{pred}(x;\theta) = a + xu_{pred}(x;\theta) \qquad (4-5)$$

and train the PINN by minimizing the loss function as follows:

$$L(\theta) = MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^4 v_{pred}(x_i;\theta)}{\partial x_i^4} + 1\right|^2 \qquad (4-6)$$

The new output $v_{pred}(x_i;\theta)$ automatically satisfies the boundary condition. As such, there is no need to add the penalty term for boundary conditions in PINN, thus reducing the number of penalty terms of the loss function and enhancing the computation efficiency. If it is a parameter identification problem (inverse problem), consider a as an unknown while adding one more penalty term for measurement data to minimize $L(a;\theta)$.

(2) In the case with only the boundary condition $u'(0) = b$, define

$$v(x) = u(0) + bx + x^2u(x) \qquad (4-7)$$

which satisfies $v'(x) = v'(0) = b$ at $x = 0$. In the PINN configuration, the FCFNN output $u_{pred}(x;\theta)$ is modified to satisfy the boundary condition using the following modulating function:

$$v_{pred}(x;\theta) = u_{pred}(0;\theta) + bx + x^2u_{pred}(x;\theta) \qquad (4-8)$$

and train the PINN by minimizing the loss function Equation (4-6). If it is a parameter identification problem, consider b as an unknown while adding one more penalty term for measurement data to minimize $L(b;\theta)$.

(3) In the case with two boundary conditions $u(0) = a$ and $u'(0) = b$, define

$$v(x) = a + bx + x^2u(x) \qquad (4-9)$$

which satisfies $v(x) = v(0) = a$ and $v'(x) = v'(0) = b$ at $x = 0$. It is evident that this case is a combination of cases (1) and (2). In the PINN configuration, the FCFNN output $u_{pred}(x;\theta)$ is modified to satisfy the two boundary conditions automatically by using the following modulating function:

$$v_{pred}(x;\theta) = a + bx + x^2 u_{pred}(x;\theta) \qquad (4-10)$$

and use the new output $v_{pred}(x;\theta)$ to train the PINN by minimizing the loss function Equation (4-6). If it is a parameter identification problem, consider both a and b as two unknowns while adding one more penalty term for measurement data to minimize $L(a,b;\theta)$.

(4) In the case with three boundary conditions $u(0) = a$, $u'(0) = b$, and $u''(1) = c$, define

$$v(x) = a + bx + \frac{c}{2}x^2 + \left(\frac{1}{6}x^3 - \frac{1}{2}x^2\right)[u(0) + u'(0) + u''(1)] + x^2(x-1)^3 u(x) \qquad (4-11)$$

which satisfies $v(x) = v(0) = a$ and $v'(x) = v'(0) = b$ at $x = 0$, and $v''(x) = v''(1) = c$ at $x = 1$. Note that the boundary condition of the second-order derivative of $u(x)$ is on the right side of the boundary, i.e., $x = 1$. Therefore, this case is different from cases (1) to (3). Furthermore, the values of $v(1)$, $v'(1)$, and $v''(0)$ are not fixed, which means no extra conditions are added to the problem. In the PINN configuration, the FCFNN output $u_{pred}(x;\theta)$ is modified to satisfy the three boundary conditions automatically by using the following modulating function:

$$v_{pred}(x;\theta) = a + bx + \frac{c}{2}x^2 + \left(\frac{1}{6}x^3 - \frac{1}{2}x^2\right)\left[u_{pred}(0;\theta) + u'_{pred}(0;\theta) + u''_{pred}(1;\theta)\right] + x^2(x-1)^3 u_{pred}(x;\theta) \qquad (4-12)$$

and train the PINN by minimizing the loss function Equation (4-6). If it is a parameter

identification problem, consider a, b, and c as three unknowns while adding one more penalty term for measurement data to minimize $L(a, b, c; \theta)$.

(5) In the case with four boundary conditions $u(0) = a$, $u'(0) = b$, $u''(1) = c$, and $u'''(1) = d$, define

$$v(x) = a + bx + \frac{c-d}{2}x^2 + \frac{d}{6}x^3 + \left(\frac{1}{6}x^4 - \frac{1}{6}x^3 + \frac{1}{4}x^2\right) \cdot [u(0) + u'(0) + u''(1) + u'''(1)] + x^3(x-1)^4 u(x) \quad (4-13)$$

which satisfies $v(x) = v(0) = a$ and $v'(x) = v'(0) = b$ at $x = 0$, and $v''(x) = v''(1) = c$ and $v'''(x) = v'''(1) = d$ at $x = 1$. This case is similar to case (4), while adding one more boundary condition at the right side of the boundary. Furthermore, the values of $v(1)$, $v'(1)$, $v''(0)$, and $v'''(0)$ are not settled, therefore, no extra conditions are added to the problem as well. In the PINN configuration, modify the DNN output $u_{pred}(x; \theta)$ to satisfy the four boundary conditions automatically by using the following modulating function:

$$v_{pred}(x; \theta) = a + bx + \frac{c-d}{2}x^2 + \frac{d}{6}x^3 + \left(\frac{1}{24}x^4 - \frac{1}{6}x^3 + \frac{1}{4}x^2\right) \cdot \left[u_{pred}(0; \theta) + u'_{pred}(0; \theta) + u''_{pred}(1; \theta) + u'''_{pred}(1; \theta)\right] + x^3(x-1)^4 u_{pred}(x; \theta) \quad (4-14)$$

and train the PINN by minimizing the loss function Equation (4-6). If it is a parameter identification problem, consider a, b, c, and d as four unknowns while adding one more penalty term for measurement data to minimize $L(a, b, c, d; \theta)$.

By applying the above modulating functions, the transformed outputs of FCFNNs will be exactly the defined boundary values, which means the boundary conditions are automatically and strictly fulfilled. The major advantage of the preceding hard-embedding of constraints approach is the elimination of penalty terms associated with all boundary conditions. This

approach is easily adaptable to various scenarios, such as the cases in Section 4.3, and its applications will be discussed in the same section.

4.2.2 Training with Auxiliary Outputs

It has been proven that training with auxiliary outputs can significantly improve the computational efficiency of PINN in solving high-order differential equations by reducing the order of governing equations. Therefore, this useful approach is also employed in this chapter. For example, consider an inverse problem based on Equation (4-1) in Section 4.2.1, and take the primary output $u_{pred}(x;\theta)$ and one auxiliary output $w_{pred}(x;\theta) = u''_{pred}(x;\theta)$. By doing so, the original governing equation (4-1) is converted as:

$$\frac{\partial^2 w}{\partial x^2} + 1 = 0, \quad w = \frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1] \tag{4-15}$$

and the total loss function is expressed as:

$$L_{total} = w_f \cdot MSE_f + w_b \cdot MSE_b + w_m \cdot MSE_m + w_a \cdot MSE_a \tag{4-16}$$

where MSE_b and MSE_m are given by:

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|B\left(u_{pred}\left(0, t_i^b;\theta\right), 0, t_i^b\right)\right|^2 \tag{4-17}$$

$$MSE_m = \frac{1}{N_m}\sum_{i=1}^{N_m}\left|u_{pred}(x_i^m, t_i^m;\lambda;\theta) - u_m(x_i^m, t_i^m)\right|^2 \tag{4-18}$$

while MSE_f and MSE_a can be obtained from Equation (4-15) as:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2 w_{pred}\left(x_i^f;\theta\right)}{\partial {x_i^f}^2} + 1\right|^2 \tag{4-19}$$

$$MSE_a = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|w_{pred}(x_i^f;\theta) - \frac{\partial^2 u_{pred}(x_i^f;\theta)}{\partial {x_i^f}^2}\right|^2 \qquad (4-20)$$

Minimizing MSE_a will make $w_{pred}(x;\theta)$ as close to the second-order derivative of $u_{pred}(x;\theta)$ as possible.

Apart from reducing the order of the governing equation, this approach is also helpful to the hard embedding of boundary conditions. By using this method, the two boundary conditions on the left side in Equation (4-3) remain unchanged, but the two right boundary conditions will be altered because of the lowered order. With the defined auxiliary output, Equation (4-3) can be rewritten as follows:

$$u(0) = a, \quad u'(0) = b, \quad w(1) = c, \quad w'(1) = d \qquad (4-21)$$

It can be seen from the comparison between Equations (4-15) and (4-21) that the highest order of derivatives is now second-order instead of fourth-order. Thus, the potential gradient exploding and gradient vanishing problems can be avoided.

Combining the hard-embedding method in Section 4.2.1 with this training with auxiliary output method, an improved PINN framework is obtained as illustrated in Figure 4-1 (HC = Hard Constraints). In Section 4.3, both the vanilla PINN and the improved PINN frameworks will be applied to identify unknown rotational stiffness of semi-rigid joints, and their performance will be examined through comparison of experimental results.

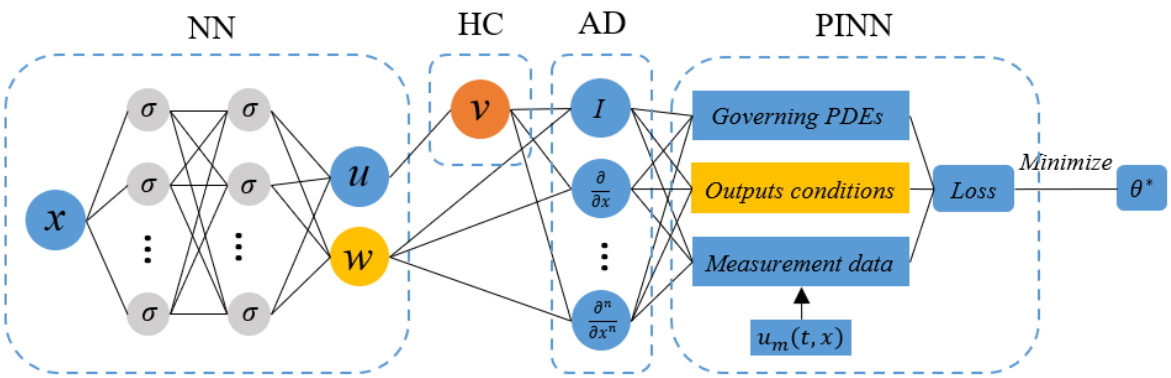


Figure 4-1 A diagram of the improved PINN framework for inverse problems

4.3 Numerical Case Studies

To explore potential applications of PINN and evaluate the performance of the proposed framework in identification of unknown structural parameters, both the vanilla PINN and the improved PINN frameworks are applied to discover the unknown rotational stiffness of semi-rigid joints of different structures.

Variable cross-section cantilever beam with single semi-rigid joint

The first structure is a variable cross-section cantilever beam with one semi-rigid joint on its fixed end, as shown in Figure 4-2. The structure is simple because only one structural member is involved, but its governing equation is a fourth-order differential equation and its boundary conditions include a third-order differential equation, both of which will hinder computation accuracy. Hence, it is a suitable application scenario for the improved PINN method to show its ability in handling boundary conditions and reducing the highest order of both governing equation and boundary conditions. The material and loading properties of the structure are as follows: the left cross-section area $A_1 = 48\,mm \times 60\,mm$, the right cross-section area $A_2 = 48\,mm \times 40\,mm$, Young's modulus $E = 2.1 \times 10^5\,N/mm^2$, length $l = 1\,m$, distributed load $q = -1\,kN/m$.

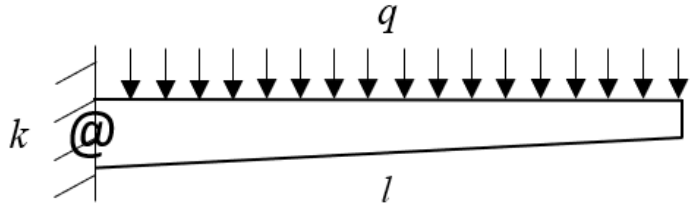


Figure 4-2 Schematic of a variable cross-section cantilever beam with a semi-rigid joint

The governing equation can be expressed by Euler-Bernoulli beam theory (Gere and Timoshenko 1997) as

$$\frac{\partial^2}{\partial x^2}\left(EI(x)\frac{\partial^2 u(x)}{\partial x^2}\right) = q, \qquad x \in [0, l] \tag{4-22}$$

with four boundary conditions

$$u(0) = 0, \quad u'(0) = \alpha, \quad u''(l) = 0, \quad u'''(l) = 0 \tag{4-23}$$

where E denotes Young's modulus of the beam; $I(x)$ denotes moment of inertia of the cross-section at position x of the beam; x is the coordinate in the domain of $[0, l]$; $u(x)$ is the deflection of the beam; q is the distributed load on the beam; α is the unknown parameter involved in one of the boundary conditions, representing the rotation angle at the fixed end due to the semi-rigid joint. Due to the nonlinearity caused by the changing $I(x)$, it is difficult to obtain the analytical solution to the beam response. Hence, the commercial software ANSYS 2021 R1 is used to compute numerical solutions as the training data.

After identifying the unknown parameter α, the rotational stiffness k of the semi-rigid joint can be obtained by

$$k = \frac{M(0)}{\alpha} = \frac{ql^2}{2\alpha} \tag{4-24}$$

where $M(0)$ denotes the moment on the left boundary of the beam.

Thus, the identification of the rotational stiffness amounts to the identification of the rotation angle cast in the boundary condition. The fourth-order derivative in the nonlinear governing equation together with the third-order boundary condition may lead to low computational accuracy by PINN (Lu et al. 2021b). Also, the PINN may result in inaccuracy in the identified parameter since the unknown parameter is included in a boundary condition and all boundary condition loss functions are not necessarily equal to zero when training PINN by the conventional way explicitly considering boundary conditions and their soft embedding.

Steel frame with semi-rigid joints

The second structure is a single-bay steel frame with two semi-rigid joints connecting the beam and two columns, as shown in Figure 4-3. For simplicity, ignore the axial deformation of the members and assume Euler-Bernoulli beams again. The local coordinates of the three members are shown in Figure 4-4. It is symmetric in material, geometry, and loading, but the two semi-rigid joints possess different rotational stiffness values k_1 and k_2. The material and geometrical properties of the two columns are: the cross-section area $A_1 = 30\ mm \times 40\ mm$, Young's modulus $E = 2.1 \times 10^5\ N/mm^2$, moment of inertia $I_1 = 1.6 \times 10^5\ mm^4$, height $h = 4\ m$; the material and geometrical properties of the beam are as follows: the cross-section area $A_2 = 24\ mm \times 40\ mm$, Young's modulus $E = 2.1 \times 10^5\ N/mm^2$, moment of inertia $I_2 = 1.28 \times 10^5\ mm^4$, length $l = 4\ m$, distributed load $q = -1\ kN/m$.

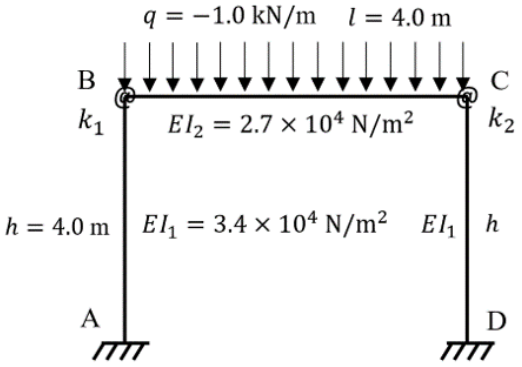


Figure 4-3 Single-bay steel frame with two semi-rigid joints

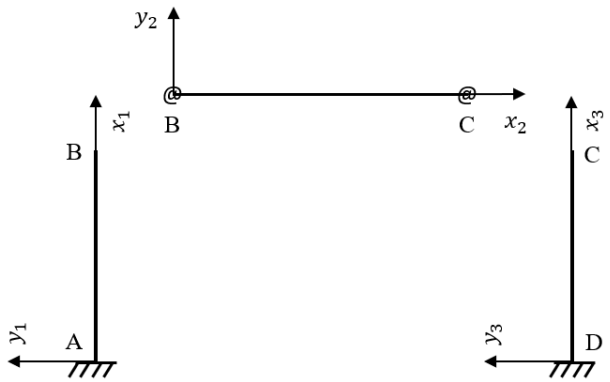


Figure 4-4 Local coordinates for structural members

The governing equations of the three members can be expressed as

$$-\frac{\partial}{\partial x_1}\left(EI_1\frac{\partial^2 u_1(x_1)}{\partial x_1^2}\right)=F_A,\qquad x_1\in[0,h]$$

$$\frac{\partial^2}{\partial x_2^2}\left(EI_2\frac{\partial^2 u_2(x_2)}{\partial x_2^2}\right)=q,\qquad x_2\in[0,l]$$

$$-\frac{\partial}{\partial x_3}\left(EI_1\frac{\partial^2 u_3(x_3)}{\partial x_3^2}\right)=F_D,\qquad x_3\in[0,h] \tag{4-25}$$

with eight boundary conditions:

$$u_1(0)=u_1(h)=0$$

$$u_2(0)=u_2(l)=0$$

$$u_3(0)=u_3(h)=0$$

$$u_1'(0)=0$$

$$u_3'(0)=0 \tag{4-26}$$

In addition, there are four continuity conditions at two conjunction nodes:

$$u_1'(h)=u_2'(0)+\alpha_1$$

$$u_2'(l)=u_3'(h)+\alpha_2$$

$$EI_1u_1''(h)=EI_2u_2''(0)$$

$$EI_2u_2''(l)=-EI_1u_3''(h) \tag{4-27}$$

where E denotes Young's modulus of each member; I_1 and I_2 are moments of inertia of the cross-sections of the beam and columns, respectively; x_1, x_2, and x_3 denote local coordinates; $u_1(x_1)$, $u_2(x_2)$, and $u_3(x_3)$ are deflections of the members AB, BC, and DC; q is the distributed load on the beam; F_A and F_D are shear forces at the two columns; α_1 and α_2 are unknown relative rotation angles of the beam and columns at nodes B and C.

The shear forces of the columns AB and DC are $F_A = -F_D = -\frac{EI_1}{4EI_2} \cdot \frac{ql^3}{h^2+2hl}$. The analytical solution to the deflection responses of the structure is

$$u_1(x_1) = a_1 {x_1}^3 + b_1 {x_1}^2, \qquad x_1 \in [0, h]$$

$$u_2(x_2) = a_2 {x_2}^4 + b_2 {x_2}^3 + c_2 {x_2}^2 + d_2 x_2, \qquad x_2 \in [0, l]$$

$$u_3(x_3) = a_3 {x_3}^3 + b_3 {x_3}^2, \qquad x_3 \in [0, h] \qquad (4-28)$$

where $a_1 = -\frac{F_A}{6EI_1}$, $b_1 = \frac{F_A h}{6EI_1}$; $a_2 = \frac{q}{24EI_2}$, $b_2 = \frac{3a_3h^2+2b_3h-2a_2l^3+d_2+\alpha_2}{l^2}$, $c_2 = \frac{-3a_3h^2-2b_3h+a_2l^3-2d_2-\alpha_2}{l}$, $d_2 = 3a_1h^2 + 2b_1h - \alpha_1$; and $a_3 = -\frac{F_D}{6EI_1}$, $b_3 = \frac{F_D h}{6EI_1}$.

After identifying α_1 and α_2, the rotational stiffness k_1 and k_2 of the semi-rigid joints can be obtained by

$$k_1 = \frac{EI_1 u_1''(h)}{\alpha_1}$$

$$k_2 = -\frac{EI_1 u_3''(h)}{\alpha_2} \qquad (4-29)$$

In this case, the three governing equations need to be solved by PINN simultaneously, and the third- and fourth-order derivatives involved would hinder the computational efficiency. If the soft embedding method is imposed to both boundary and continuity conditions, the identification accuracy would be further undermined.

As aforementioned, PINN can be pursued to address forward as well as inverse problems. The inverse problem concerned here is the identification of the unknown parameters, while the forward problem is the deflection calculation of the structure with given parameters. In the inverse problem, the unknown parameters are identified with the aid of limited measurement data. In the forward problem, with given structural parameters, the response (deflection) of the structure at any location can be directly inferred by PINN without need of any measurement

data, nor need of discretization and approximation such as transferring distributed loads to equivalent nodal forces or interpolating defection by shape functions. As a result, PINN is a mesh-free approach escaping from discretization and truncation errors. A rational way to verify the accuracy of the identified parameters is to compare the calculated responses of the structure using the identified parameter values and true responses. Since the response calculation is also pursued by PINN in this study, the improved PINN method is used to deal with both forward and inverse problems of the above examples in the following case studies.

To evaluate the effectiveness of the improved PINN method, the identification of rotational stiffness of semi-rigid joints in both the cantilever beam and steel frame is pursued by three schemes: (1) vanilla PINN, denoted as S1, (2) vanilla PINN with hard constraints, denoted as S2, (3) improved PINN (vanilla PINN with hard constraints and training with auxiliary outputs), denoted as S3. In all FCFNN configurations, the activation function is taken as Tanh, and L-BFGS with a learning rate of 0.1 is adopted as the optimizer. All codes are written in Python using the ML library PyTorch. The influences of the number of collocation points and measurement noise on the accuracy of identification are also addressed.

4.3.1 Variable Cross-Section Cantilever Beam

The variable cross-section cantilever beam structure subjected to a distributed load is illustrated in Figure 4-2. The unknown rotational stiffness k of the semi-rigid joint can be represented in terms of the rotation angle α, as shown in Equation (4-24). Thus, PINN can be constructed to identify the unknown parameter α involved in the boundary conditions and compute the structural response.

S1: Forward and inverse problems by the vanilla PINN

First, start with the forward problem to show that PINN can solve the governing equation without any simulation or measurement data. Here α is given as -5.0000×10^{-2}.

A one-output FCFNN is constructed to approximate $u(x)$ in Equation (4-22). The input is the coordinate x, while the output is the deflection u of the beam. Three hidden layers with 20 neurons in each layer are considered, since this FCFNN architecture is found to offer sufficient capacity while avoiding overfitting based on preliminary experiments. The total loss function L_{total} is expressed as:

$$L_{total} = MSE_f + MSE_b \tag{4-30}$$

The first component MSE_f is the penalty term for the governing equation, which is the *MSE* of the governing equation's residuals at collocation points. It can be obtained by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2}{\partial {x_i^f}^2}\left[EI\left(x_i^f\right)\frac{\partial^2 u_{pred}\left(x_i^f;\theta\right)}{\partial {x_i^f}^2}\right]-q\right|^2 \tag{4-31}$$

where N_f denotes the number of collocation points; u_{pred} denotes the output of FCFNN, which is the deflection of the beam; θ denotes the parameters of the FCFNN; q is the distributed load; and x_i^f denotes the randomly sampled collocation points within the coordinate's domain.

The second component MSE_b is the penalty term for the boundary conditions, which is the *MSE* of residuals on the boundaries. In the vanilla PINN framework, MSE_b is expressed as:

$$MSE_b = \left|u_{pred}(0;\theta)\right|^2 + \left|u'_{pred}(0;\theta)-\alpha\right|^2 + \left|u''_{pred}(l;\theta)\right|^2 + \left|u'''_{pred}(l;\theta)\right|^2 \tag{4-32}$$

The training data consists of 1000 random sampled collocation points within the area of $x \in [0, l]$ using Latin hypercube sampling (LHS). The reason for using LHS is that it is a

stratified sampling method, which ensures better coverage of the sample space compared to traditional uniform random sampling. The validation data contains 10 selected points as shown in Table 4-1, which should be excluded from the training data to ensure an unbiased assessment of the model's performance. Thus, during the collocation points sampling, points within the interval centered at validation points with a small radius ε ($\varepsilon = 0.005l$ in this case) are removed from the training dataset. The relative L_2 error representing the difference between the predicted value and the exact value is adopted to assess the accuracy of vanilla PINN solution, and it is calculated by

$$Relative\ L_2\ error = \frac{\sqrt{\sum_i \left|u_{exact}^i - u_{pred}^i\right|^2}}{\sqrt{\sum_i \left|u_{exact}^i\right|^2}} \qquad (4-33)$$

where u_{exact} denotes the exact solution at selected points.

The solution by vanilla PINN at 10 selected points is shown in Table 4-1 and Figure 4-5 in comparison with the exact solution. The computation time is 13.64 seconds for this forward problem case, and it is recorded as a reference for further comparison. Comparing the vanilla PINN solution and the true value, their relative L_2 error is 0.0798%, showing that with given α, vanilla PINN can provide accurate prediction of the structural response without using any simulation or measurement data.

Table 4-1 Comparison of the vanilla PINN solution and exact solution for the forward problem of the variable cross-section cantilever beam

x (m)	Exact value (m)	Vanilla PINN solution (m)	Relative error	x (m)	Exact value (m)	Vanilla PINN solution (m)	Relative error
0.1	-5.0135×10^{-3}	-5.0155×10^{-3}	0.0399%	0.6	-3.0390×10^{-2}	-3.0365×10^{-2}	0.0816%
0.2	-1.0052×10^{-2}	-1.0047×10^{-2}	0.0434%	0.7	-3.5504×10^{-2}	-3.5475×10^{-2}	0.0820%

0.3	-1.5112×10^{-2}	-1.5102×10^{-2}	0.0667%	0.8	-4.0624×10^{-2}	-4.0591×10^{-2}	0.0810%
0.4	-2.0190×10^{-2}	-2.0175×10^{-2}	0.0766%	0.9	-4.5746×10^{-2}	-4.5710×10^{-2}	0.0801%
0.5	-2.5284×10^{-2}	-2.5264×10^{-2}	0.0807%	1.0	-5.0870×10^{-2}	-5.0830×10^{-2}	0.0776%

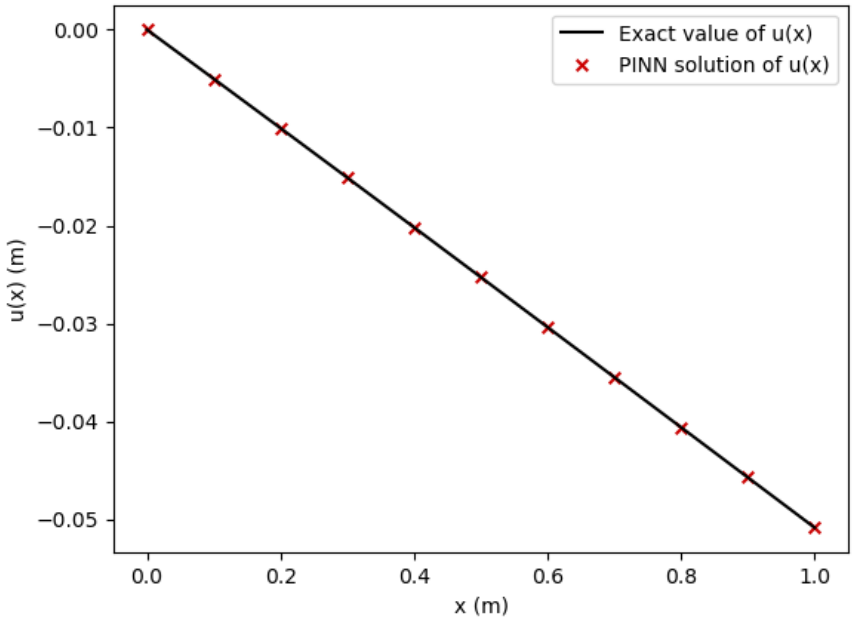


Figure 4-5 The exact value and the vanilla PINN solution of the variable cross-section cantilever beam

Then proceed to the inverse problem to demonstrate vanilla PINN's ability to discover the unknown parameter α with using a small amount of measurement data. A one-output FCFNN with α being a trainable parameter is constructed to approximate $u(x)$. The input is the coordinate x, and the output is the deflection u. Again, consider three hidden layers, and 20 neurons in each layer. In the inverse problem, the total loss function L_{total} contains three components:

$$L_{total} = MSE_f + MSE_b + MSE_m \tag{4-34}$$

The first component MSE_f is the penalty term for the governing equation. It can be calculated including the trainable parameter α by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2}{\partial {x_i^f}^2}\left[EI\left(x_i^f\right)\frac{\partial^2 u_{pred}\left(x_i^f;\alpha;\theta\right)}{\partial {x_i^f}^2}\right]-q\right|^2 \quad (4-35)$$

The second component MSE_b is the penalty term for boundary conditions, which is calculated including the trainable parameter α by:

$$MSE_b = \left|u_{pred}(0;\alpha;\theta)\right|^2 + \left|u'_{pred}(0;\alpha;\theta)-\alpha\right|^2 + \left|u''_{pred}(l;\alpha;\theta)\right|^2 + \left|u'''_{pred}(l;\alpha;\theta)\right|^2 \quad (4-36)$$

The third component MSE_m is the penalty term for residuals between the predicted values and the measured values on measurement points:

$$MSE_m = \frac{1}{N_m}\sum_{i=1}^{N_m}\left|u_{pred}(x_i^m;\alpha;\theta)-u_m(x_i^m)\right|^2 \quad (4-37)$$

where N_m is the number of measurement points; x_i^m denotes the coordinates of measurement points; and u_m denotes the measured values of $u(x)$ at measurement points.

The training data consists of two sets. The first set includes 5 measurement points evenly sampled on the beam, while the second set includes 1000 collocation points randomly sampled on the beam.

The computation time for this inverse problem case is measured to be 12.67 seconds, which is comparable to that of the forward problem case (13.64 seconds), showing that the vanilla PINN is highly adept at solving inverse problems, with only one or more additional parameters to be trained compared to solving forward problems. The parameter α is identified to be -5.0096×10^{-2}, and the difference between the identified value and the true value (-5.0000×10^{-2}) is 0.1920%. The result shows that vanilla PINN can discover unknown parameters even with a small set of measurement data. With the identified α, Table 4-2 shows a comparison of the vanilla PINN solution and the exact values of beam response at 10 selected points. The relative L_2 error is 0.0840%, showing the ability of vanilla PINN for both

parameter and response discovery.

Table 4-2 Comparison of the vanilla PINN solution and exact solution for the inverse problem of the variable cross-section cantilever beam

x (m)	Exact value (m)	Vanilla PINN solution (m)	Relative error	x (m)	Exact value (m)	Vanilla PINN solution (m)	Relative error
0.1	-5.0135×10^{-3}	-4.9874×10^{-3}	0.5208%	0.6	-3.0390×10^{-2}	-3.0376×10^{-2}	0.0443%
0.2	-1.0052×10^{-2}	-1.0023×10^{-2}	0.2877%	0.7	-3.5504×10^{-2}	-3.5501×10^{-2}	0.0085%
0.3	-1.5112×10^{-2}	-1.5083×10^{-2}	0.1913%	0.8	-4.0624×10^{-2}	-4.0634×10^{-2}	0.0253%
0.4	-2.0190×10^{-2}	-2.0164×10^{-2}	0.1312%	0.9	-4.5746×10^{-2}	-4.5773×10^{-2}	0.0577%
0.5	-2.5284×10^{-2}	-2.5263×10^{-2}	0.0829%	1.0	-5.0870×10^{-2}	-5.0915×10^{-2}	0.0898%

To understand how the quantity of collocation points affects the effectiveness of the vanilla PINN, more numerical experiments are conducted. Keeping the architecture of FCFNN unchanged, the quantity of collocation points varies from 100 to 2000. The corresponding results are illustrated in Table 4-3. It is observed that the relative errors of α are lower than 0.3000% in all cases, which suggests that the precision of parameter identification does not change considerably as the number of collocation points alters; however, the relative L_2 errors of u in the cases with 100 and 200 collocation points are an order of magnitude larger than those in the other cases, which indicates that the accuracy of response solution is considerably affected by the number of collocation points. Another observation is that, when the number of collocation points exceeds 1000, both the relative error of α and relative L_2 error of u increase, implying some overfitting of the model. Therefore, setting no less than 500 and no more than 1000 collocation points is suggested in this case.

Table 4-3 Comparison of the relative L_2 errors with different numbers of collocation points

N_f	100	200	500	1000	2000
Relative error of α	0.2934%	0.2235%	0.2026%	0.1920%	0.2105%
Relative L2 error of u	0.6631%	0.2059%	0.0478%	0.0840%	0.0872%

S2: Inverse problem by the vanilla PINN with hard constraints

As indicated in Section 4.2.1, the hard embedding of boundary conditions has advantages such as reducing the number of penalty terms in the loss function to facilitate computation. In this section, the performance of vanilla PINN with hard constraints is examined. According to the modulating functions derived in Section 4.2.1, the output of the FCFNN, $u_{pred}(x;\theta)$, that only satisfies the governing equation, is transformed to satisfy all boundary conditions automatically by setting:

$$v_{pred}(x;\theta) = ax + \left(\frac{1}{24}x^4 - \frac{1}{6}x^3 + \frac{1}{4}x^2\right)\left[u_{pred}(0;\theta) + u'_{pred}(0;\theta) + u''_{pred}(l;\theta) + u'''_{pred}(l;\theta)\right] + x^3(x-1)^4 u_{pred}(x;\theta) \quad (4-38)$$

where α is the unknown parameter to be identified.

A one-output FCFNN including the trainable parameter α is constructed to represent $u(x)$. The input is the coordinate x, and the output is the deflection response u. Again, three hidden layers with 20 neurons in each layer are considered. Since all boundary conditions are satisfied automatically, the total loss function L_{total} consists of only two components:

$$L_{total} = MSE_f + MSE_m \quad (4-39)$$

The first component MSE_f is the penalty term for the governing equation, which is given by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2}{\partial {x_i^f}^2}\left[EI\left(x_i^f\right)\frac{\partial^2 v_{pred}\left(x_i^f;\alpha;\theta\right)}{\partial {x_i^f}^2}\right]-q\right|^2 \qquad (4-40)$$

The second component MSE_m is the penalty term for measurement points, and given by:

$$MSE_m = \frac{1}{N_m}\sum_{i=1}^{N_m}\left|v_{pred}(x_i^m;\alpha;\theta)-u_m(x_i^m)\right|^2 \qquad (4-41)$$

The training data used is the same as that in the previous case. The computation time in this case is 8.59 seconds. Although the computation time appears to have only improved by around 4 seconds compared to the previous two cases, the increases are 37.02% and 32.19%, respectively. This indicates that reducing the number of penalty terms by applying hard boundary conditions can enhance computational efficiency. The parameter α is identified to be -4.9918×10^{-2}, and the relative error of the identified value and the true value (-5.0000×10^{-2}) is 0.1640%. The result shows that the identification accuracy has been improved compared with the vanilla PINN with soft constraints (the relative error is reduced from 0.1920% to 0.1640%). A comparison of the predicted values and the theoretical solution of beam response at 10 selected points is shown in Table 4-4. The relative L_2 error is 0.0783%.

Table 4-4 Comparison of the vanilla PINN with hard constraints solution and exact solution for the inverse problem of the variable cross-section cantilever beam

x (m)	Exact value (m)	PINN with HC solution (m)	Relative error	*x* (m)	Exact value (m)	PINN with HC solution (m)	Relative error
0.1	-5.0135×10^{-3}	-5.0226×10^{-3}	0.1801%	0.6	-3.0390×10^{-2}	-3.0360×10^{-2}	0.0968%
0.2	-1.0052×10^{-2}	-1.0049×10^{-2}	0.0317%	0.7	-3.5504×10^{-2}	-3.5473×10^{-2}	0.0867%
0.3	-1.5112×10^{-2}	-1.5099×10^{-2}	0.0863%	0.8	-4.0624×10^{-2}	-4.0593×10^{-2}	0.0758%
0.4	-2.0190×10^{-2}	-2.0170×10^{-2}	0.1011%	0.9	-4.5746×10^{-2}	-4.5718×10^{-2}	0.0619%
0.5	-2.5284×10^{-2}	-2.5258×10^{-2}	0.1015%	1.0	-5.0870×10^{-2}	-5.0845×10^{-2}	0.0477%

S3: Inverse problem by the improved PINN

The governing equation described by Equation (4-22) is a fourth-order ODE. Thus, the AD manipulation will be executed four times in vanilla PINN, which will inevitably bring about errors and trim down computational efficiency. In the following, the training with auxiliary method is introduced to reduce the order of the governing equation.

First, apply Equation (4-38) to transform the FCFNN output $u_{pred}(x;\theta)$ to $v_{pred}(x;\theta)$ that satisfies all boundary conditions. Then, define an auxiliary output $w_{pred}(x;\theta)$ which is the second-order derivative of the transformed output $v_{pred}(x;\theta)$. The relationship between $w_{pred}(x;\theta)$ and $v_{pred}(x;\theta)$ is expressed by

$$w_{pred}(x;\theta) = \frac{\partial^2 v_{pred}(x;\theta)}{\partial x^2} \qquad (4-42)$$

Thus, the fourth-order derivative of $u_{pred}(x;\theta)$ in the governing equation can be downscaled to the second-order derivative of $w_{pred}(x;\theta)$. With the introduction of the auxiliary output, a two-output FCFNN with hard constraints can be constructed as shown in Figure 4-6.

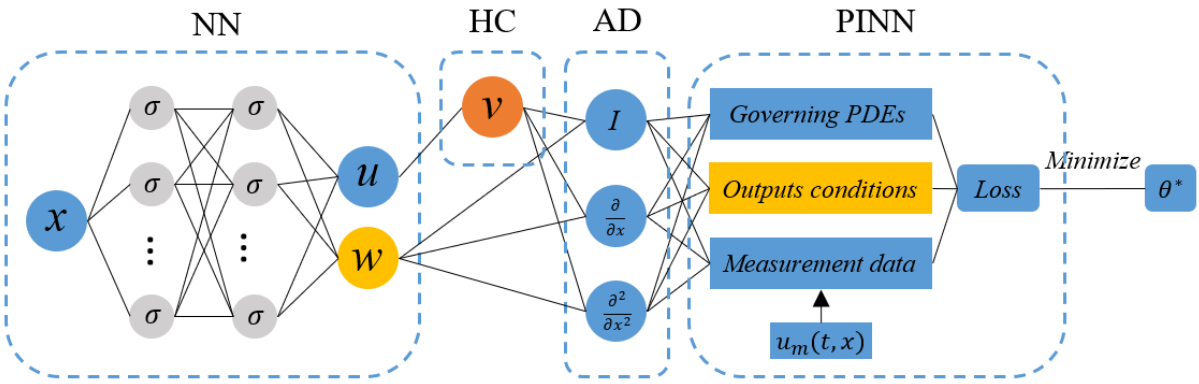


Figure 4-6 Framework of a two-output FCFNN with hard constraints for inverse problems

The input of the two-output FCFNN is the coordinate x, and the outputs are the deflection u and the second-order derivative of u, respectively. Consider again three hidden layers with

20 neurons in each layer. The total loss function L_{total} consists of three components:

$$L_{total} = MSE_f + MSE_m + MSE_a \tag{4-43}$$

The first component MSE_f, which is the penalty term for the governing equation, can be calculated with the auxiliary output $w_{pred}(x;\theta)$ by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial^2}{\partial {x_i^f}^2}\left[EI\left(x_i^f\right)w_{pred}\left(x_i^f;\alpha;\theta\right)\right] - q\right|^2 \tag{4-44}$$

where the auxiliary output w_{pred} is equal to the second derivative of v_{pred}.

The second component MSE_m, which is the penalty term for measurement points, can be calculated with the output $v_{pred}(x;\theta)$ by:

$$MSE_m = \frac{1}{N_m}\sum_{i=1}^{N_m}\left|v_{pred}(x_i^m;\alpha;\theta) - u_m(x_i^m)\right|^2 \tag{4-45}$$

The third component MSE_a in Equation (4-43) is the extra penalty term for the relationship between the auxiliary output and the primary output. It represents the *MSE* of residuals between the auxiliary output $w_{pred}(x;\theta)$ and the transformed output $v_{pred}(x;\theta)$ on collocation points inside the domain in compliance with Equation (4-42). It can be calculated by:

$$MSE_a = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|w_{pred}\left(x_i^f;\alpha;\theta\right) - \frac{\partial^2 v_{pred}\left(x_i^f;\alpha;\theta\right)}{\partial {x_i^f}^2}\right|^2 \tag{4-46}$$

There is no need to consider the penalty term associated with the boundary conditions because they are satisfied automatically by the hard-embedding method. The training data used is the same as that in the previous case. The computation time for this case is 12.38 seconds, which is slightly higher compared to the example using only hard boundary conditions. It is reasonable, as the introduction of the additional penalty term MSE_a slightly reduces

computational efficiency. The parameter α is identified to be -5.0023×10^{-2}, and the relative error between it and the true value (-5.0000×10^{-2}) is 0.0464%. The result shows that the accuracy of identification is further improved by one order of magnitude by the improved PINN (the relative error is reduced from 0.1640% to 0.0464%). It is worth mentioning that the vanilla PINN in general performs well for ODEs or PDEs when the highest order of derivatives is not higher than the second order. A comparison of the predicted values by the improved PINN and the exact values of beam response is shown in Table 4-5. The relative L_2 error between them is 0.0434%, showing that by the improved PINN, the surrogate model can maintain high prediction accuracy.

Table 4-5 Comparison of the improved PINN solution and exact values of structural response for the inverse problem of the non-prismatic cantilever beam

x (m)	Exact value (m)	Improved PINN solution (m)	Relative error	x (m)	Exact value (m)	Improved PINN solution (m)	Relative error
0.1	-5.0135×10^{-3}	-5.0159×10^{-3}	0.0466%	0.6	-3.0390×10^{-2}	-3.0373×10^{-2}	0.0553%
0.2	-1.0052×10^{-2}	-1.0048×10^{-2}	0.0338%	0.7	-3.5504×10^{-2}	-3.5486×10^{-2}	0.0499%
0.3	-1.5112×10^{-2}	-1.5104×10^{-2}	0.0539%	0.8	-4.0624×10^{-2}	-4.0606×10^{-2}	0.0423%
0.4	-2.0190×10^{-2}	-2.0179×10^{-2}	0.0573%	0.9	-4.5746×10^{-2}	-4.5730×10^{-2}	0.0354%
0.5	-2.5284×10^{-2}	-2.5269×10^{-2}	0.0572%	1.0	-5.0870×10^{-2}	-5.0855×10^{-2}	0.0282%

4.3.2 Steel Frame

The steel frame structure is illustrated in Figure 4-3. The unknown rotational stiffness of the two semi-rigid joints can be obtained from α_1 and α_2 by Equation (4-29). Therefore, the purpose of PINN frameworks in this case study is to identify the two different unknown parameters α_1 and α_2 that are directly related to the rotational stiffness.

S1: Forward and inverse problems by the vanilla PINN

Again, start with the forward problem to first demonstrate the ability of vanilla PINN to predict the deflection responses of this frame structure without using any simulation or measurement data. In the forward problem, α_1 and α_2 are assumed as 2.0000×10^{-2} and 4.0000×10^{-2}, respectively.

In the vanilla PINN, three one-output FCFNNs are constructed and connected in parallel to approximate $u_1(x_1)$, $u_2(x_2)$, and $u_3(x_3)$ in Equation (4-25). As a result, there are three inputs which are the local coordinates of the three structural members. Also, there are three outputs which represent the deflection responses of the three structural members. Three hidden layers with 20 neurons in each layer are considered in each of the three FCFNNs. The total loss function L_{total} is expressed as:

$$L_{total} = MSE_f + MSE_b + MSE_c \tag{4-47}$$

The first component MSE_f is the penalty term for governing equations, which is the *MSE* of the three governing equations' residuals and is calculated by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 u_{1pred}(x_i^f;\theta)}{\partial {x_i^f}^3} + F_A\right|^2 + \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_2\frac{\partial^4 u_{2pred}(x_i^f;\theta)}{\partial {x_i^f}^4} - q\right|^2$$
$$+\frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 u_{3pred}(x_i^f;\theta)}{\partial {x_i^f}^3} + F_D\right|^2 \tag{4-48}$$

where N_f is the number of collocation points; u_{1pred}, u_{2pred}, and u_{3pred} denote the outputs of the three FCFNNs, each representing the deflection of a member; θ denotes the parameters of the three FCFNNs; q denotes the distributed load; F_A and F_D are the shear forces of the two columns; and x_i^f denotes the collocation points randomly chosen within the domain of the three coordinates.

The second component MSE_b is the penalty term for boundary conditions, which is the *MSE* of residuals on the boundaries and is calculated by:

$$MSE_b = \left|u_{1pred}(0;\theta)\right|^2 + \left|u_{1pred}(h;\theta)\right|^2 + \left|u_{2pred}(0;\theta)\right|^2 + \left|u_{2pred}(l;\theta)\right|^2$$
$$+\left|u_{3pred}(0;\theta)\right|^2 + \left|u_{3pred}(h;\theta)\right|^2 + \left|u'_{1pred}(0;\theta)\right|^2 + \left|u'_{3pred}(0;\theta)\right|^2 \qquad (4-49)$$

The third component MSE_c is the penalty term for continuity, which is the *MSE* of residuals of the continuity conditions and is given in the following form:

$$MSE_c = \left|u'_{1pred}(h;\theta) - u'_{2pred}(0;\theta) - \alpha_1\right|^2 + \left|u'_{2pred}(l;\theta) - u'_{3pred}(h;\theta) - \alpha_2\right|^2$$
$$+\left|EI_1 u''_{1pred}(h;\theta) - EI_2 u''_{2pred}(0;\theta)\right|^2$$
$$+\left|EI_2 u''_{2pred}(l;\theta) + EI_1 u''_{3pred}(h;\theta)\right|^2 \qquad (4-50)$$

where α_1 and α_2 are the relative rotation angles between each column and the beam.

In the forward problem, measurement data is not needed. Therefore, the training data only contains 3 sets of 1000 collocation points randomly sampled on each structural member using LHS. The relative L_2 error between the predicted values and true values, defined in Equation (4-33), is used to evaluate the precision of the vanilla PINN solution. Since three high-order differential equations are solved simultaneously in this frame case, which is much more complex than the cantilever beam case, the computation time is 131.12 seconds. Table 4-6 and Figure 4-7 show the comparison of the vanilla PINN solution and true values at 7 selected points on each of the members. Considering the symmetry of the structure, only the results of u_1 and u_2 are provided. The relative L_2 errors of u_1 and u_2 are 0.0299% and 0.0135%, respectively, showing that given structural parameters, vanilla PINN can generate a high-fidelity surrogate model for response prediction without need of simulation or measurement data.

Table 4-6 Comparison of the vanilla PINN solution and exact solution for the forward problem of the steel frame

x (m)	Exact value of u_1 (m)	PINN solution of u_1 (m)	Relative error	Exact value of u_2 (m)	PINN solution of u_2 (m)	Relative error
0.5	1.8084×10^{-3}	1.8059×10^{-3}	0.1426%	-2.9059×10^{-2}	-2.9057×10^{-2}	0.0051%
1.0	6.2004×10^{-3}	6.2012×10^{-3}	0.0136%	-5.7502×10^{-2}	-5.7504×10^{-2}	0.0025%
1.5	1.1626×10^{-2}	1.1628×10^{-2}	0.0202%	-7.8582×10^{-2}	-7.8587×10^{-2}	0.0065%
2.0	1.6534×10^{-2}	1.6536×10^{-2}	0.0124%	-8.7870×10^{-2}	-8.7879×10^{-2}	0.0102%
2.5	1.9376×10^{-2}	1.9376×10^{-2}	0.0004%	-8.3269×10^{-2}	-8.3281×10^{-2}	0.0142%
3.0	1.8601×10^{-2}	1.8597×10^{-2}	0.0212%	-6.5002×10^{-2}	-6.5015×10^{-2}	0.0196%
3.5	1.2659×10^{-2}	1.2650×10^{-2}	0.0733%	-3.5621×10^{-2}	-3.5633×10^{-2}	0.0334%

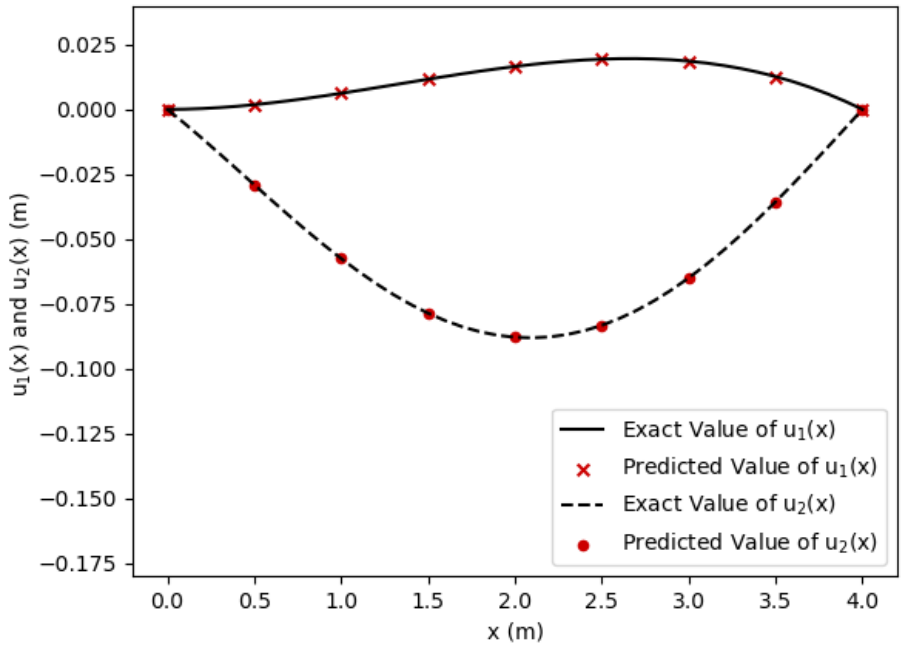


Figure 4-7 The exact value and the vanilla PINN solution of the steel frame

Next, proceed to the inverse problem on identifying the values of the two different parameters α_1 and α_2 by using a small amount of measurement data. Three one-output

FCFNNs with two trainable parameters (α_1 and α_2) are constructed in the vanilla PINN framework to approximate $u_1(x_1)$, $u_2(x_2)$ and $u_3(x_3)$, respectively. There are three inputs (x_1, x_2, x_3) and three outputs (u_1, u_2, u_3). In each FCFNN, three hidden layers with 20 neurons in each layer are considered. The total loss function L_{total}, which consists of four components, is expressed as:

$$L_{total} = MSE_f + MSE_b + MSE_c + MSE_m \tag{4-51}$$

The first component MSE_f is the penalty term for the three governing equations, and it is computed by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 u_{1pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^3} + F_A\right|^2 + \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_2\frac{\partial^4 u_{2pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^4} - q\right|^2$$
$$+\frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 u_{3pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^3} + F_D\right|^2 \tag{4-52}$$

in which α denotes the two unknown parameters α_1 and α_2.

The second component MSE_b is the penalty term for the boundary conditions, and it is computed by:

$$MSE_b = \left|u_{1pred}(0;\alpha;\theta)\right|^2 + \left|u_{1pred}(h;\alpha;\theta)\right|^2 + \left|u_{2pred}(0;\alpha;\theta)\right|^2 + \left|u_{2pred}(l;\alpha;\theta)\right|^2$$
$$+\left|u_{3pred}(0;\alpha;\theta)\right|^2 + \left|u_{3pred}(h;\alpha;\theta)\right|^2 + \left|u'_{1pred}(0;\alpha;\theta)\right|^2 + \left|u'_{3pred}(0;\alpha;\theta)\right|^2 \tag{4-53}$$

The third component MSE_c is the penalty term for the continuity conditions, and it is computed by:

$$MSE_c = \left|u'_{1pred}(h;\alpha;\theta) - u'_{2pred}(0;\alpha;\theta) - \alpha_1\right|^2 +$$

$$\left|u'_{2pred}(l;\alpha;\theta) - u'_{3pred}(h;\alpha;\theta) - \alpha_2\right|^2 +$$

$$\left|EI_1 u''_{1pred}(h;\alpha;\theta) - EI_2 u''_{2pred}(0;\alpha;\theta)\right|^2 +$$

$$\left|EI_2 u''_{2pred}(l;\alpha;\theta) + EI_1 u''_{3pred}(h;\alpha;\theta)\right|^2 \qquad (4-54)$$

The fourth component MSE_m is the penalty term for measurement points, and it is computed by:

$$MSE_m = \frac{1}{N_{m1}}\sum_{i=1}^{N_{m1}}\left|u_{1pred}(x_i^m;\alpha;\theta) - u_{m1}(x_i^m)\right|^2 +$$

$$\frac{1}{N_{m2}}\sum_{i=1}^{N_{m2}}\left|u_{2pred}(x_i^m;\alpha;\theta) - u_{m2}(x_i^m)\right|^2 +$$

$$\frac{1}{N_{m3}}\sum_{i=1}^{N_{m3}}\left|u_{3pred}(x_i^m;\alpha;\theta) - u_{m3}(x_i^m)\right|^2 \qquad (4-55)$$

where N_{m1}, N_{m2}, and N_{m3} are the numbers of measurement points on the three members; x_i^m denotes the coordinates of measurement points; u_{m1}, u_{m2}, and u_{m3} denote the measured deflection responses at measurement points.

The training data comprises two sets. The first set consists of 7 measurement points, including 5 evenly sampled on the beam and 2 respectively at the mid-span of the two columns. The second set contains 3 sets of 1000 collocation points randomly sampled on the three members using LHS.

The computation time is 117.10 seconds, which is 14 seconds faster than the forward problem case. This is because the assistance of measurement data has improved the convergence speed of vanilla PINN. The parameters α_1 and α_2 are identified to be 1.9888×10^{-2} and 4.0116×10^{-2}; and the relative errors compared to their true values (2.0000×10^{-2} and 4.0000×10^{-2}) are 0.5576% and 0.2911%, respectively. It can be seen that the vanilla PINN can satisfactorily identify unknown parameters with a small amount of measurement data.

Table 4-7 shows a comparison of the predicted values by vanilla PINN and the theoretical solution of structural response at 7 selected points on each of the three members. The relative L_2 errors of u_1 and u_2 are 0.0640% and 0.0283%, respectively, showing that vanilla PINN is capable of solving the governing equations while identifying the unknown parameters.

Table 4-7 Comparison of the vanilla PINN solution and exact solution for the inverse problem of the steel frame

x (m)	Exact value of u_1 (m)	Vanilla PINN solution of u_1 (m)	Relative error	Exact value of u_2 (m)	Vanilla PINN solution of u_2 (m)	Relative error
0.5	1.8084×10^{-3}	1.8035×10^{-3}	0.2759%	-2.9059×10^{-2}	-2.9042×10^{-2}	0.0555%
1.0	6.2004×10^{-3}	6.1910×10^{-3}	0.1516%	-5.7502×10^{-2}	-5.7477×10^{-2}	0.0450%
1.5	1.1626×10^{-2}	1.1614×10^{-2}	0.0996%	-7.8582×10^{-2}	-7.8564×10^{-2}	0.0221%
2.0	1.6534×10^{-2}	1.6523×10^{-2}	0.0719%	-8.7870×10^{-2}	-8.7870×10^{-2}	0.0000%
2.5	1.9376×10^{-2}	1.9366×10^{-2}	0.0524%	-8.3269×10^{-2}	-8.3286×10^{-2}	0.0207%
3.0	1.8601×10^{-2}	1.8594×10^{-2}	0.0391%	-6.5002×10^{-2}	-6.5028×10^{-2}	0.0395%
3.5	1.2659×10^{-2}	1.2658×10^{-2}	0.0124%	-3.5621×10^{-2}	-3.5637×10^{-2}	0.0446%

S2: Inverse problem by the vanilla PINN with hard constraints

The case study S2 given in Section 4.3.1 has shown that the hard embedding of boundary conditions is helpful to improve the accuracy of the vanilla PINN. Regarding the inverse problem of the frame structure where there are eight boundary conditions, the penalty term associated with the boundary conditions can be significantly reduced by imposing the hard embedding of boundary conditions.

In the vanilla PINN, $u_{1pred}(x;\alpha;\theta)$, $u_{2pred}(x;\alpha;\theta)$, and $u_{3pred}(x;\alpha;\theta)$ are the outputs of FCFNNs without considering boundary conditions. They can be transformed to

satisfy the boundary conditions automatically by using the following formulae:

$$v_{1pred}(x_1;\alpha;\theta) = ({x_1}^2 - {x_1}^3)u_{1pred}(x_1;\alpha;\theta)$$

$$v_{2pred}(x_2;\alpha;\theta) = ({x_2}^2 - l{x_2})u_{2pred}(x_2;\alpha;\theta)$$

$$v_{3pred}(x_3;\alpha;\theta) = ({x_3}^3 - h{x_3}^2)u_{3pred}(x_3;\alpha;\theta) \qquad (4-56)$$

where α denotes the unknown parameters.

Three one-output FCFNNs with two trainable parameters (α_1 and α_2) are configured in the vanilla PINN framework to approximate $u_1(x_1)$, $u_2(x_2)$ and $u_3(x_3)$, respectively. Again, three hidden layers with 20 neurons in each layer are considered in each FCFNN. The total loss function L_{total}, which contains three components, is expressed as:

$$L_{total} = MSE_f + MSE_c + MSE_m \qquad (4-57)$$

The penalty term for the governing equations, MSE_f, is given by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 v_{1pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^3} + F_A\right|^2 + \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_2\frac{\partial^4 v_{2pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^4} - q\right|^2$$

$$+\frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial^3 v_{3pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^3} + F_D\right|^2 \qquad (4-58)$$

The penalty term for the continuity conditions, MSE_c, is given by:

$$MSE_c = \left|v'_{1pred}(h;\alpha;\theta) - v'_{2pred}(0;\alpha;\theta) - \alpha_1\right|^2 +$$

$$\left|v'_{2pred}(l;\alpha;\theta) - v'_{3pred}(h;\alpha;\theta) - \alpha_2\right|^2 +$$

$$\left|EI_1 v''_{1pred}(h;\alpha;\theta) - EI_2 v''_{2pred}(0;\alpha;\theta) - M_1\right|^2 +$$

$$\left|EI_2 v''_{2pred}(l;\alpha;\theta) + EI_1 v''_{3pred}(h;\alpha;\theta) - M_2\right|^2 \qquad (4-59)$$

The penalty term for measurement points, MSE_m, is given by:

$$MSE_m = \frac{1}{N_{m1}}\sum_{i=1}^{N_{m1}}\left|v_{1pred}(x_i^m;\alpha;\theta) - u_{m1}(x_i^m)\right|^2 +$$

$$\frac{1}{N_{m2}}\sum_{i=1}^{N_{m2}}\left|v_{2pred}(x_i^m;\alpha;\theta) - u_{m2}(x_i^m)\right|^2 +$$

$$\frac{1}{N_{m3}}\sum_{i=1}^{N_{m3}}\left|v_{3pred}(x_i^m;\alpha;\theta) - u_{m3}(x_i^m)\right|^2 \qquad (4-60)$$

As all boundary conditions are satisfied automatically, there is no penalty term for boundary conditions. The training data used is the same as the previous. With the hard embedding of boundary conditions in the vanilla PINN, the computation time decreases to 88.35 seconds compared to conventional PINN (117.10 seconds), resulting in a 24.55% improvement in computational efficiency. The unknown parameters α_1 and α_2 are identified to be 2.0023×10^{-2} and 4.0048×10^{-2}, respectively; the relative errors compared to their true values (2.0000×10^{-2} and 4.0000×10^{-2}) are 0.1148% and 0.1194%. It can be seen that the accuracy of parameter identification has been improved by using the hard embedding method compared with the vanilla PINN. Table 4-8 shows a comparison of the predicted values and the true values at 7 selected points on each of the three members. The relative L_2 errors of u_1 and u_2 are 0.0132% and 0.0104%, respectively.

Table 4-8 Comparison of the vanilla PINN with hard constraints solution and exact solution for the inverse problem of the steel frame

x (m)	Exact value of u_1 (m)	PINN with HC solution of u_1 (m)	Relative error	Exact value of u_2 (m)	PINN with HC solution of u_2 (m)	Relative error
0.5	1.8084×10^{-3}	1.8096×10^{-3}	0.0625%	-2.9059×10^{-2}	-2.9063×10^{-2}	0.0148%
1.0	6.2004×10^{-3}	6.2025×10^{-3}	0.0344%	-5.7502×10^{-2}	-5.7513×10^{-2}	0.0177%

1.5	1.1626×10^{-2}	1.1628×10^{-2}	0.0201%	-7.8582×10^{-2}	-7.8592×10^{-2}	0.0134%
2.0	1.6534×10^{-2}	1.6537×10^{-2}	0.0141%	-8.7870×10^{-2}	-8.7878×10^{-2}	0.0091%
2.5	1.9376×10^{-2}	1.9378×10^{-2}	0.0115%	-8.3269×10^{-2}	-8.3273×10^{-2}	0.0045%
3.0	1.8601×10^{-2}	1.8602×10^{-2}	0.0068%	-6.5002×10^{-2}	-6.5001×10^{-2}	0.0016%
3.5	1.2659×10^{-2}	1.2659×10^{-2}	0.0020%	-3.5621×10^{-2}	-3.5617×10^{-2}	0.0114%

S3: Inverse problem by the improved PINN

To evaluate the improved PINN, apply Equation (4-51) to transform the FCFNN outputs $u_{1pred}(x;\alpha;\theta)$, $u_{2pred}(x;\alpha;\theta)$ and $u_{3pred}(x;\alpha;\theta)$ to $v_{1pred}(x;\alpha;\theta)$, $v_{2pred}(x;\alpha;\theta)$ and $v_{3pred}(x;\alpha;\theta)$ that satisfy all boundary conditions. Then, define three auxiliary outputs $w_{1pred}(x;\alpha;\theta)$, $w_{2pred}(x;\alpha;\theta)$ and $w_{3pred}(x;\alpha;\theta)$, which are the second-order derivatives of $v_{1pred}(x;\alpha;\theta)$, $v_{2pred}(x;\alpha;\theta)$ and $v_{3pred}(x;\alpha;\theta)$, respectively. The relationships between the transformed outputs and their corresponding auxiliary outputs are expressed as:

$$w_{1pred}(x;\alpha;\theta) = \frac{\partial^2 v_{1pred}(x;\alpha;\theta)}{\partial x^2}$$

$$w_{2pred}(x;\alpha;\theta) = \frac{\partial^2 v_{2pred}(x;\alpha;\theta)}{\partial x^2}$$

$$w_{3pred}(x;\alpha;\theta) = \frac{\partial^2 v_{3pred}(x;\alpha;\theta)}{\partial x^2} \qquad (4-61)$$

After introducing the auxiliary outputs, the highest order of the governing equations in Equation (4-25) is lowered to the second-order. In the improved PINN framework, three two-output FCFNNs with two trainable parameters (α_1 and α_2) are constructed to approximate $u_1(x_1)$, $u_2(x_2)$ and $u_3(x_3)$, respectively. There are three inputs (x_1, x_2, x_3) and six outputs (u_1, u_2, u_3, v_1, v_2, v_3) in total. In each FCFNN, three hidden layers with 20 neurons in each layer are considered. The total loss function L_{total} can be expressed by:

$$L_{total} = MSE_f + MSE_c + MSE_m + MSE_a \qquad (4-62)$$

The penalty term for the three governing equations, MSE_f, is determined by:

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial w_{1pred}(x_i^f;\alpha;\theta)}{\partial x_i^f} + F_A\right|^2 +$$

$$\frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_2\frac{\partial^2 w_{2pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^2} - q\right|^2 +$$

$$\frac{1}{N_f}\sum_{i=1}^{N_f}\left|EI_1\frac{\partial w_{3pred}(x_i^f;\alpha;\theta)}{\partial x_i^f} + F_D\right|^2 \qquad (4-63)$$

The penalty term for the continuity conditions, MSE_c, is determined by:

$$MSE_c = \left|v'_{1pred}(h;\alpha;\theta) - v'_{2pred}(0;\alpha;\theta) - \alpha_1\right|^2 +$$

$$\left|v'_{2pred}(l;\alpha;\theta) - v'_{3pred}(h;\alpha;\theta) - \alpha_2\right|^2 +$$

$$\left|EI_1 w_{1pred}(h;\alpha;\theta) - EI_2 w_{2pred}(0;\alpha;\theta) - M_1\right|^2 +$$

$$\left|EI_2 w_{2pred}(l;\alpha;\theta) + EI_1 w_{3pred}(h;\alpha;\theta) - M_2\right|^2 \qquad (4-64)$$

The penalty term for measurement points, MSE_m, is determined by Equation (4-60).

The fourth component MSE_a in Equation (4-62) is the extra penalty term for the auxiliary outputs. It is the *MSE* of residuals between the auxiliary outputs and the transformed outputs in compliance with Equation (4-61), and is determined by:

$$MSE_a = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|w_{1pred}(x_i^f;\alpha;\theta) - \frac{\partial^2 v_{1pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^2}\right|^2 +$$

$$\frac{1}{N_f}\sum_{i=1}^{N_f}\left|w_{2pred}(x_i^f;\alpha;\theta) - \frac{\partial^2 v_{2pred}(x_i^f;\alpha;\theta)}{\partial {x_i^f}^2}\right|^2 +$$

$$\frac{1}{N_f}\sum_{i=1}^{N_f}\left|w_{3pred}\left(x_i^f;\alpha;\theta\right)-\frac{\partial^2 v_{3pred}\left(x_i^f;\alpha;\theta\right)}{\partial {x_i^f}^2}\right|^2 \qquad (4-65)$$

Owing to adopting the hard embedding of boundary conditions, there is no penalty term for boundary conditions. The computation time measured for this case is 100.41 seconds. The training data used is the same as previous case. The unknown parameters α_1 and α_2 are identified to be 2.0016×10^{-2} and 3.9973×10^{-2}, respectively; the relative errors compared to their true values (2.0000×10^{-2} and 4.0000×10^{-2}) are 0.0783% and 0.0684%. It is observed that although the additional loss term slightly reduces computational efficiency, it still shows a 14.25% improvement over conventional PINN, which means the reduced order can enhance computational efficiency. Moreover, the identification accuracy has been further enhanced by using the improved PINN framework. Table 4-9 shows a comparison of the improved PINN solution and the exact values of structural response at selected points. The relative L_2 errors of u_1 and u_2 are 0.0079% and 0.0041%, respectively.

Table 4-9 Comparison of the improved PINN solution and exact values for the inverse problem of the frame structure

x (m)	Exact value of u_1 (m)	Improved PINN solution of u_1 (m)	Relative error	Exact value of u_2 (m)	Improved PINN solution of u_2 (m)	Relative error
0.5	1.8084×10^{-3}	1.8090×10^{-3}	0.0321%	-2.9059×10^{-2}	-2.9060×10^{-2}	0.0067%
1.0	6.2004×10^{-3}	6.2017×10^{-3}	0.0202%	-5.7502×10^{-2}	-5.7507×10^{-2}	0.0073%
1.5	1.1626×10^{-2}	1.1627×10^{-2}	0.0133%	-7.8582×10^{-2}	-7.8585×10^{-2}	0.0045%
2.0	1.6534×10^{-2}	1.6536×10^{-2}	0.0070%	-8.7870×10^{-2}	-8.7872×10^{-2}	0.0016%
2.5	1.9376×10^{-2}	1.9378×10^{-2}	0.0067%	-8.3269×10^{-2}	-8.3268×10^{-2}	0.0014%
3.0	1.8601×10^{-2}	1.8602×10^{-2}	0.0051%	-6.5002×10^{-2}	-6.5000×10^{-2}	0.0043%
3.5	1.2659×10^{-2}	1.2659×10^{-2}	0.0013%	-3.5621×10^{-2}	-3.5619×10^{-2}	0.0067%

Since LHS is adopted as the sampling method to ensure that the collocation points cover the entire domain, it is important to discuss how the model's performance might be affected when other non-uniform sampling methods are used. For more complex physical problems, it may be challenging to achieve full coverage of the domain with collocation points. Here, a different sampling method called importance sampling (IS) is employed and applied two different sampling strategies. In the first strategy (ST1), higher importance is assigned to the central region around $x = 2$, while in the second strategy (ST2), higher importance is given to the areas near the boundaries. The 3 sets of 1000 collocation points are sampled within the domain [0, 4]. The collocation points sampled on the beam by LHS, ST1 and ST2 are visualized in Figure 4-8 for a better illustration.

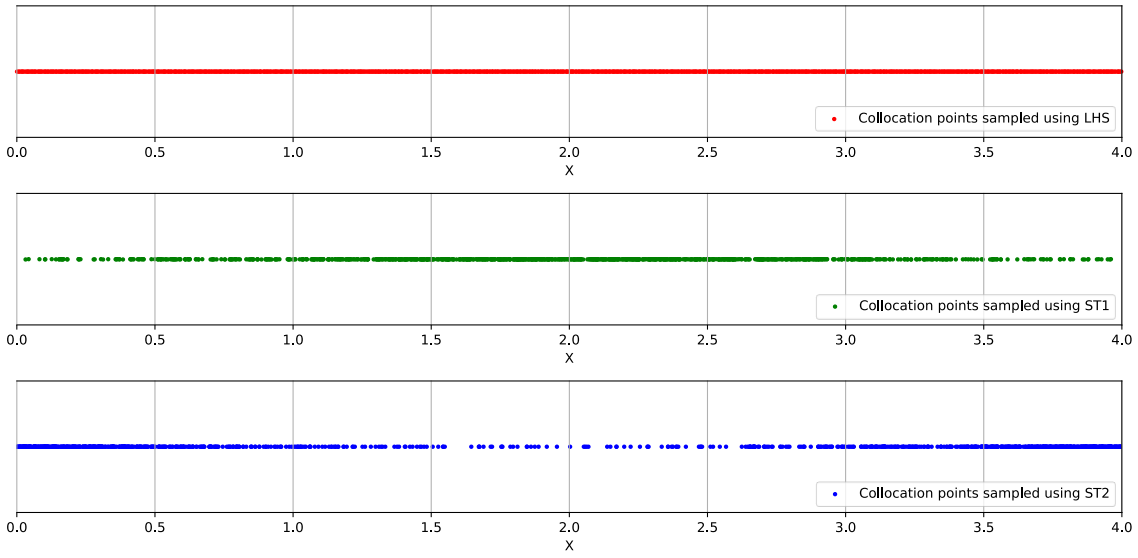


Figure 4-8 The collocation points sampled on the beam by LHS, ST1, and ST2

In ST1, where collocation points are more densely concentrated near the center of the domain, the identification accuracies for the two rotational angles are 3.5560% and 1.3194%, respectively, with relative L_2 errors of 0.4438% and 0.1848% for u_1 and u_2. In contrast, in ST2, where collocation points are concentrated near the boundaries, the identification accuracies are 1.6330% and 0.6787%, with relative L_2 errors of 0.1962% and 0.1161% for

u_1 and u_2, respectively. While both methods show a decrease in accuracy compared to the proposed framework with LHS, likely due to the incomplete coverage of the domain by the non-uniform sampling, ST2 outperforms ST1 slightly. This suggests that concentrating collocation points near the boundaries yields better results than focusing on the central region. Overall, these results indicate that in the proposed method, using non-uniform sampling leads to worse results. Therefore, to achieve satisfactory outcomes, collocation points should be sampled as comprehensively and uniformly as possible across the entire domain.

In order to evaluate the ability of PINN frameworks in dealing with noisy data, random noise of different levels is added to the measurement data. The noisy data is yielded via adding Gaussian noise to noise-free measurement data:

$$u_m^{noisy}(x) = (1 + noise \cdot randn) \cdot u_m(x) \qquad (4-66)$$

where $u_m^{noisy}(x)$ denotes the noisy data; $u_m(x)$ denotes the noise-free measurement data; $noise$ denotes the noise level; $randn$ is a random data sequence generated from the standard normal distribution. Table 4-10 shows the relative errors of the identified unknown parameters α_1 and α_2 under noise levels varying from 0% to 10%. The relative errors of both α_1 and α_2 are getting larger with the increase of noise level. However, even in the case of 10% noise level, the error is less than 5%, indicating that PINN frameworks are tolerant of noise in parameter identification. It is worth mentioning that if the noise level is not larger than 2%, the unknown parameters can be precisely identified with no more than 1% relative error.

Table 4-10 Relative errors of α_1 and α_2 under different noise levels

Noise level	0%	1%	2%	5%	10%
Relative error of α_1	0.0783%	0.2623%	0.8114%	2.4417%	4.5267%
Relative error of α_2	0.0684%	0.2164%	0.3592%	1.6150%	2.5898%

4.4 Experimental Validation

To validate the capability of the proposed approach, an experimental study, in which the improved PINN is applied to identify rotational angles of semi-rigid joints in a steel frame under static loading, is carried out.

4.4.1 Experimental Setup

The experimental setup comprises two key components: one is a steel frame with semi-rigid joints, the other is a measuring system designed to capture displacements of the frame and rotational angles of the semi-rigid joints.

Steel frame with semi-rigid joints

The steel frame, as depicted in Figure 4-9, is designed with a single span, incorporating two columns and a beam. The columns are affixed to square plates, then securely fixed to the ground. Bolts connect the beam to the columns, with two bolts at each joint, as shown in Figure 4-10. The critical aspect of these joints is their semi-rigid nature, which is adjustable through the use of bolts. By regulating the tightness of the bolts, the rotational stiffness of the joints can be controlled, allowing for variable degrees of rotational angles at the beam-column connections when subjected to loading.

The material and geometrical properties of the columns are measured or tested as follows: the cross-section area $A_1 = 30\ mm \times 30\ mm$, Young's modulus $E = 2.0 \times 10^5\ N/mm^2$, moment of inertia $I_1 = 6.75 \times 10^4\ mm^4$, height $h = 400\ mm$; the material and geometrical properties of the beam are measured or tested as follows: the cross-section area $A_2 = 30\ mm \times 5\ mm$, Young's modulus $E = 6.85 \times 10^4\ N/mm^2$, moment of inertia $I = 312.5\ mm^4$, length $l = 400\ mm$. The load is concentrated on the mid-point of the beam and

is ranging from $10\,N$ to $40\,N$.

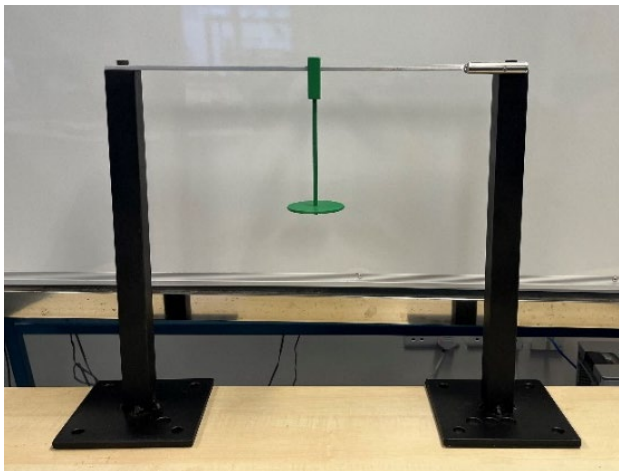


Figure 4-9 The steel frame with semi-rigid joints

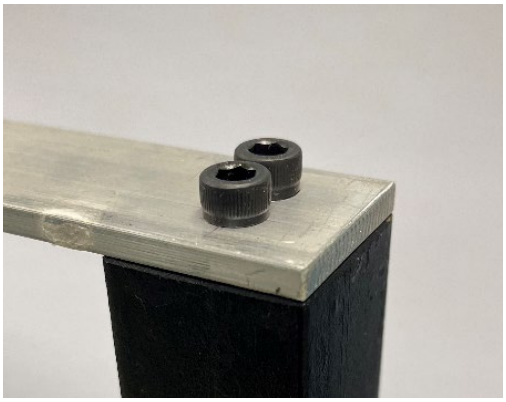


Figure 4-10 The semi-rigid joints connected by adjustable bolts

Measuring system

The measuring system is designed to capture two types of quantities: the deformation of the frame's components (the beam and columns) and the rotational angles of the semi-rigid joints under loadings.

Deformation of the components is measured using digital micrometers, depicted in Figure 4-11. The digital micrometer boasts an accuracy of 0.001 mm and is securely mounted on an existing steel frame. Four measurement points on the beam and two measurement points on the columns are selected to gauge their deformation accurately.

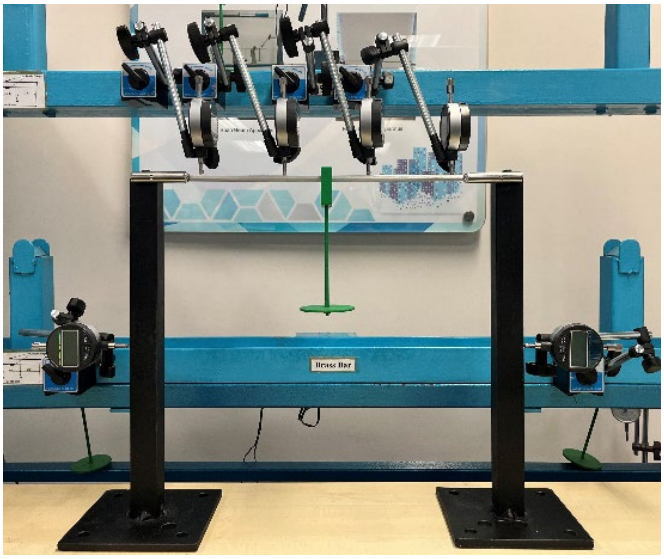
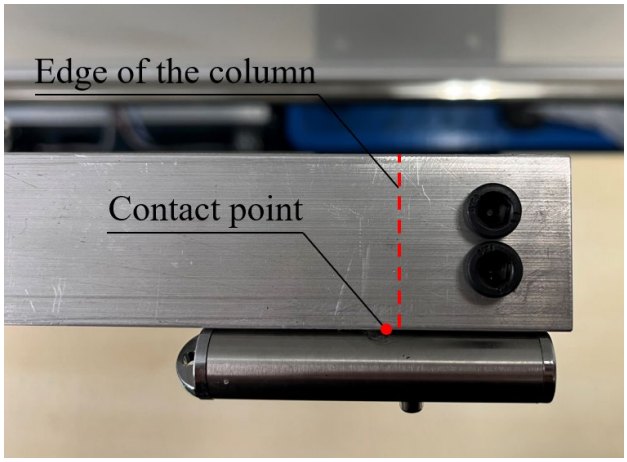


Figure 4-11 Digital micrometers for deformation measurement

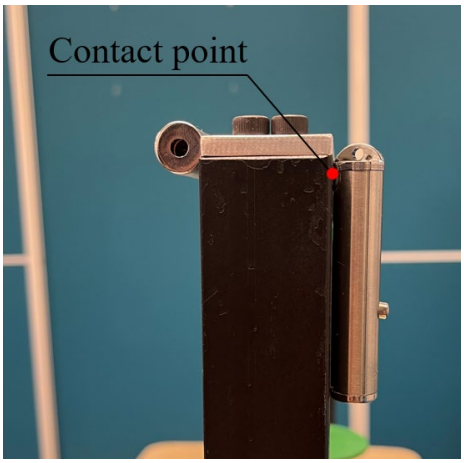
For the measurement of rotational angles, two methods are considered before testing. The first one is using inclinometers. However, the accuracy of the inclinometer cannot meet the requirement. Thus, the second method is adopted, which is using laser pointers.

In this method, small laser pointers are affixed to both sides of the beam and the top ends of two columns. For each laser pointer attached to the beam, it is strategically positioned so that only a single point on one side of the beam, which is at the edge of the column, comes in contact with the laser pointer. Similarly, laser pointers are attached to two columns at two points that are very close to the beam. The details of the placement of the laser pointers are shown in Figure 4-12. Consequently, when rotational angles at both sides of the beam and the ends of two columns occur, they can be precisely reflected by the laser point.

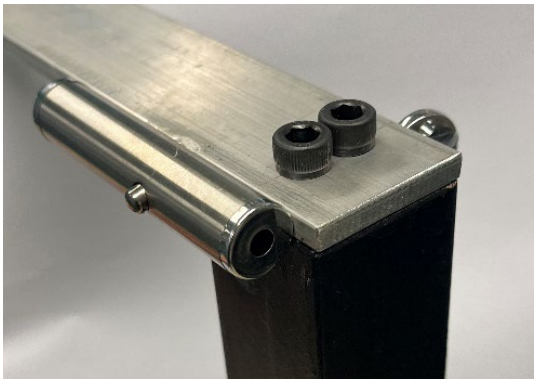
By measuring two values, one is the distance between the laser pointer and the point where the laser intersects with a white paper, the other is the distance traversed by that point, the rotational angles of each component can be calculated accordingly, as demonstrated in Figure 4-13 and Figure 4-14. The difference between the rotational angles of the beam's end and column's end represents the absolute rotational angle of the semi-rigid joint.



(a)



(b)



(c)

Figure 4-12 Placement of the laser pointers

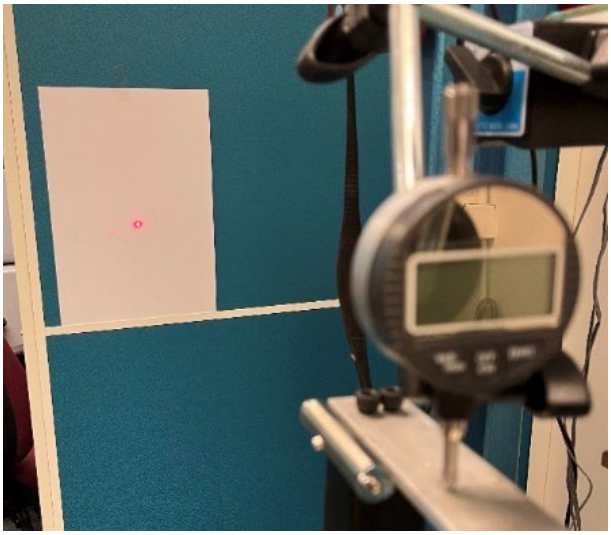


Figure 4-13 Measurement of rotational angles using laser pointers

4.4.2 Experimental Results

The experimental results under different loading scenarios are presented in this subsection. The deformation of the beam and the columns will be used as the input of the proposed framework to identify the rotational stiffness of the semi-rigid joints, while the rotational angles of semi-rigid joints measured by the laser pointers will be used to validate the accuracy of the identified results.

Deformation of the beam and the columns

Only four measurement points on the beam, which are eight-division points P_1, P_2, P_3, and P_4, and two measurement points on two columns, which are mid-points P_5 and P_6, were monitored to capture their displacements under various loading conditions, as shown in Figure 4-14. The recorded displacements under different scenarios are detailed in Table 4-11.

Table 4-11 Displacements on measurement points under different loadings (mm)

Loading	P_1	P_2	P_3	P_4	P_5	P_6
10 N	-0.358	-0.846	-0.797	-0.402	0.016	-0.019
20 N	-0.636	-1.507	-1.454	-0.702	0.028	-0.034
30 N	-0.913	-2.184	-2.152	-1.040	0.041	-0.050
40 N	-1.165	-2.776	-2.711	-1.296	0.052	-0.063

Rotational angles of semi-rigid joints

The distances between the laser pointers and four reference points are depicted in Figure 4-14. The calculated absolute rotational angles α_1 and α_2 under different loadings are presented in Table 4-13. Then, the rotational stiffness can be obtained according to Equation (4-29).

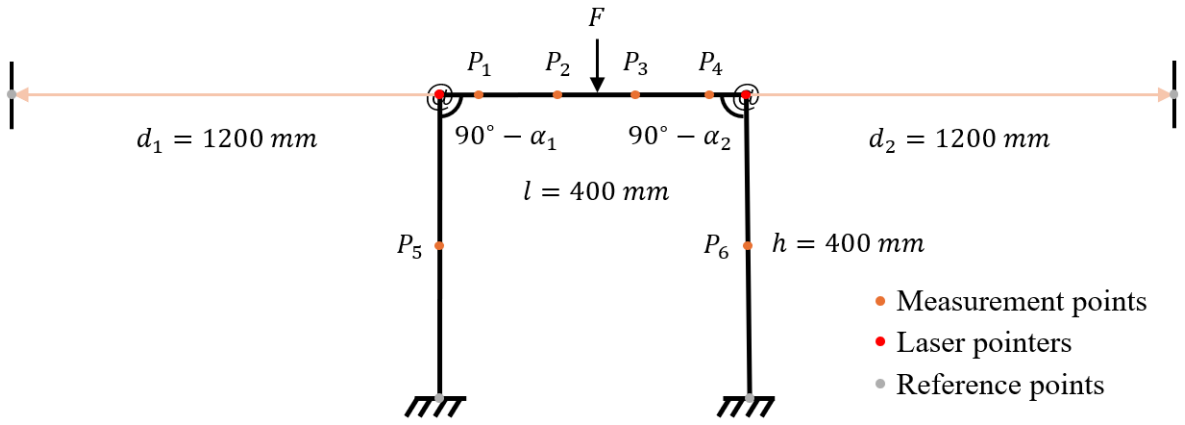


Figure 4-14 Illustration for the measurement points and distances for calculating rotational angles

4.4.3 Comparison of Results

The measurement data of only 6 measurement points in Table 4-12 is used as the training data in both the vanilla PINN and the improved PINN frameworks (i.e. hard constraints and training with auxiliary outputs) to identify the rotational angles. Tables 4-12 and 4-13 provide

the identified rotational angles by the vanilla and improved PINN frameworks, respectively, and those measured by the laser pointer system. It is seen that the relative errors between the identified and measured rotational angles in all four loading scenarios are around 10% when the vanilla PINN is used, while relative errors are all less than 5% when the improved PINN is employed. The relative errors for the vanilla PINN are relatively large because there exist inevitable measurement errors in the limited training data and the governing equation used to train the model is based on several assumptions leading to discrepancies between the theoretical solution and the true structural response. However, the improved PINN reduces the relative errors from around 10% to below 5%, indicating enhanced identification accuracy and high robustness in handling noisy data. It is therefore concluded that the improved PINN framework can achieve high accuracy of identification using limited measurement data in practical cases.

Table 4-12 Comparison of the rotational angles identified by the vanilla PINN and measurement data

Loading	Measured value of α_1	PINN solution of α_1	Relative error	Measured value of α_2	PINN solution of α_2	Relative error
10 N	0.3876	0.4349	12.2033%	0.3706	0.4180	12.7901%
20 N	0.6993	0.7723	10.4390%	0.6645	0.7371	10.9255%
30 N	0.9942	1.0913	9.7666%	0.9626	1.0611	10.2327%
40 N	1.2805	1.4102	10.1289%	1.2096	1.3356	10.4167%

Table 4-13 Comparison of the rotational angles identified by the improved PINN and measurement data

Loading	Measured value of α_1	Improved PINN solution of α_1	Relative error	Measured value of α_2	Improved PINN solution of α_2	Relative error
10 N	0.3876	0.4057	4.6698%	0.3706	0.3881	4.7221%
20 N	0.6993	0.7310	4.5331%	0.6645	0.6956	4.6802%
30 N	0.9942	0.9703	2.4039%	0.9626	0.9404	2.3063%
40 N	1.2805	1.2531	2.1398%	1.2096	1.2412	2.6124%

4.5 Summary

The method of hard-embedding of boundary conditions is proposed to fully eliminate the penalty term associated with all boundary conditions, and the training method enabled by auxiliary outputs is introduced to reduce the order of derivatives involved in the governing equations while facilitating the training of FCFNN. Both improvements help to enhance the performance of the vanilla PINN framework.

Numerical studies of a variable cross-section cantilever beam and a frame structure embracing semi-rigid joints are conducted to illustrate the capability of PINN frameworks in solving forward problems (prediction of structural response) without the use of any simulation and measurement data and solving inverse problems (identification of unknown structural parameters) with using only a small amount of measurement data. In the cantilever beam case, the unknown parameter is identified accurately by the vanilla PINN with only 5 measurement points, giving a relative error of 0.192%. The computation time is 12.7 seconds. In the frame structure case, the relative errors of the two identified parameters are 0.558% and 0.291% when using only 7 measurement points. The computation time is 117 seconds. When adopting the improved PINN framework, both computational efficiency and identification accuracy are

improved in two cases. In the cantilever beam case, the relative error is reduced from 0.192% to 0.0464%, and the computation time decreases slightly from 12.7 seconds to 12.4 seconds; in the frame structure case, the relative errors are reduced from 0.558% and 0.291% to 0.0783% and 0.0684%, and the computation time decreases from 117 seconds to 100 seconds, showing a 14.3% improvement in efficiency. The results indicate that using hard constraints and lowering the order of governing equations provide a viable way to ameliorate the efficiency and accuracy of vanilla PINN in parameter identification. Also, the proposed approach is tolerant of noise in measurement data. The experimental study demonstrates that the relative errors between the identified and measured rotational angles in all test scenarios are less than 5%, testifying the capability of the proposed method in practical applications.

Despite advantages in handling boundary conditions, the proposed method still faces limitations when applied to more complex structural systems. Specifically, for problems involving complex boundaries, achieving uniform and dense sampling is challenging, and deriving the modulating functions is not always feasible. To address these limitations, a unified framework that combines pretrained NNs will be explored in the next chapter.

CHAPTER 5

Two-Phase Physics-Informed Neural Network for Efficient Training and Accurate Solution

5.1 Introduction

The modulating function-based method proposed in Chapter 4 can effectively achieve the hard manner treatment of boundary conditions. However, when involving complex boundaries, it is difficult to obtain modulating functions. To extend the method to more complex scenarios, the TP-PINN framework is introduced in this chapter. TP-PINN divides the training of PINN into two separate phases, where boundary conditions can be totally or partially satisfied and the multi-objective optimization can be eased in the first phase, while the order of governing equations can be reduced in the second phase. Through this novel configuration, both computational efficiency and prediction accuracy can be significantly improved. By comparing the performance of TP-PINN with vanilla PINN, it shows promising potential of TP-PINN in solving structural mechanics problems.

The main content of this chapter is structured as follows. The proposed TP-PINN framework is illustrated in Section 5.2 and validated via numerical case studies in Section 5.3. Then, a detailed discussion is presented in Section 5.4.

5.2 Methods

In this section, a novel framework termed TP-PINN is proposed. The implementation of TP-PINN is detailed, including the establishment of loss functions, training approaches in two phases, and parameters of both the FCFNN (denoted as NN in this section for brevity) and the training process.

5.2.1 Establishment of Two-Phase PINN

As the basic model, the vanilla PINN is illustrated in Figure 5-1(a). To improve the vanilla PINN, modulating functions are proposed in Chapter 4 to transform the NN output, thereby enabling a "hard enforcement" of the boundary conditions:

$$u^t(x) = d(x)u(x) + u^*(x) \qquad (5-1)$$

where $u(x)$ is the original output of the NN, $d(x)$ is a distance function that vanishes on the boundary, $u^*(x)$ is a function that satisfies boundary conditions, $d(x)$ and $u^*(x)$ together are called modulating functions, and $u^t(x)$ is the transformed output. By deriving specific $d(x)$ and $u^*(x)$ and applying them to $u(x)$, the resulting $u^t(x)$ automatically satisfies all boundary conditions, allowing the penalty term for boundary conditions in the loss function to be omitted, thereby effectively enhancing both computational efficiency and accuracy. However, for problems with complex and irregular boundaries (such as the irregularly shaped plate illustrated in Figure 5-3(a)), this method exhibits limitations, which are the difficulties in deriving analytical forms of the modulating functions.

To address this issue, a unified TP-PINN framework, which decouples the treatment of boundary conditions from the multi-objective loss function by separating the training process

into two phases and reduces the order of the governing equation, is proposed. The proposed framework leads to substantial improvements in computational efficiency and accuracy when solving high-order differential equations in structural mechanics and extends its applicability to problems with irregular domains.

The TP-PINN framework is illustrated in Figure 5-1(b). In TP-PINN framework, the training process is divided into two phases. In Phase 1, one or several pretrained NNs, denoted as *PreNN*, are generated that satisfy certain predetermined conditions. These conditions will be derived and explained in detail in this section, while the construction of loss function will be introduced in Section 5.2.2. Different forms can be taken for *PreNN*s, and the forms have significant influences on the computational performance. Three forms are shown in Figure 5-1(b) as examples, and their performance will be examined in Sections 5.3.1 and 5.4.2. In Phase 2, a primary NN, denoted as *PriNN*, is trained to solve the differential equation. During this phase, the *PreNN* serves as the $d(x)$ term in Equation (5-1), together with $u^*(x)$, typically the boundary value, to transform the output $u(x)$ of *PriNN* to $u^t(x)$ so that it automatically satisfies part of or all boundary conditions. Because the *PreNN* is trained by collocation points sampled on and within the boundaries, they are applicable to problems with complex and irregular boundaries. In this way, the use of the *PreNN* realizes a universal solution of "hard-enforcement" of boundary conditions in arbitrary domains without the need for complicated derivations of modulating functions.

Furthermore, extra outputs, such as $u^{(n)}(x)$ and $u^{(m)}(x)$, are defined as the n-th and m-th derivatives of $u(x)$ (with $1 \leq n < m < h$, where h represents the highest order of derivatives in the governing equation). Consequently, the governing equation is transformed into a (h - m)-

th order differential equation in terms of $u^{(m)}(x)$, thereby reducing the burden of high-order differentiation in the original equation. Finally, based on the governing equation, the boundary conditions, and the relationships among $u(x)$, $u^{(n)}(x)$, and $u^{(m)}(x)$, a physics-informed loss function is constructed. In this loss function, the boundary condition terms are simplified or even eliminated due to the involvement of the *PreNN*.

In summary, the advantages of the TP-PINN framework are threefold. First, it reduces the highest order of the governing equation by converting the high-order differential equation into a series of first- and second-order equations, which effectively enhances computational efficiency and reduces computation errors. Second, the boundary condition penalty term is decoupled from the total loss function, originally optimized simultaneously with other penalty terms, by addressing them in a preliminary training phase, thereby mitigating the negative impact of multi-objective optimization on computational efficiency. Third, because the *PreNN*s in the first training phase satisfy certain predetermined criteria, the transformed output of the *PriNN* in the second phase inherently meets part of or all boundary conditions in arbitrary domains. In this way, not only Dirichlet boundary conditions but also Neumann boundary conditions and even higher order of boundary conditions can be handled in a “hard” manner, thus improving accuracy near the boundaries.

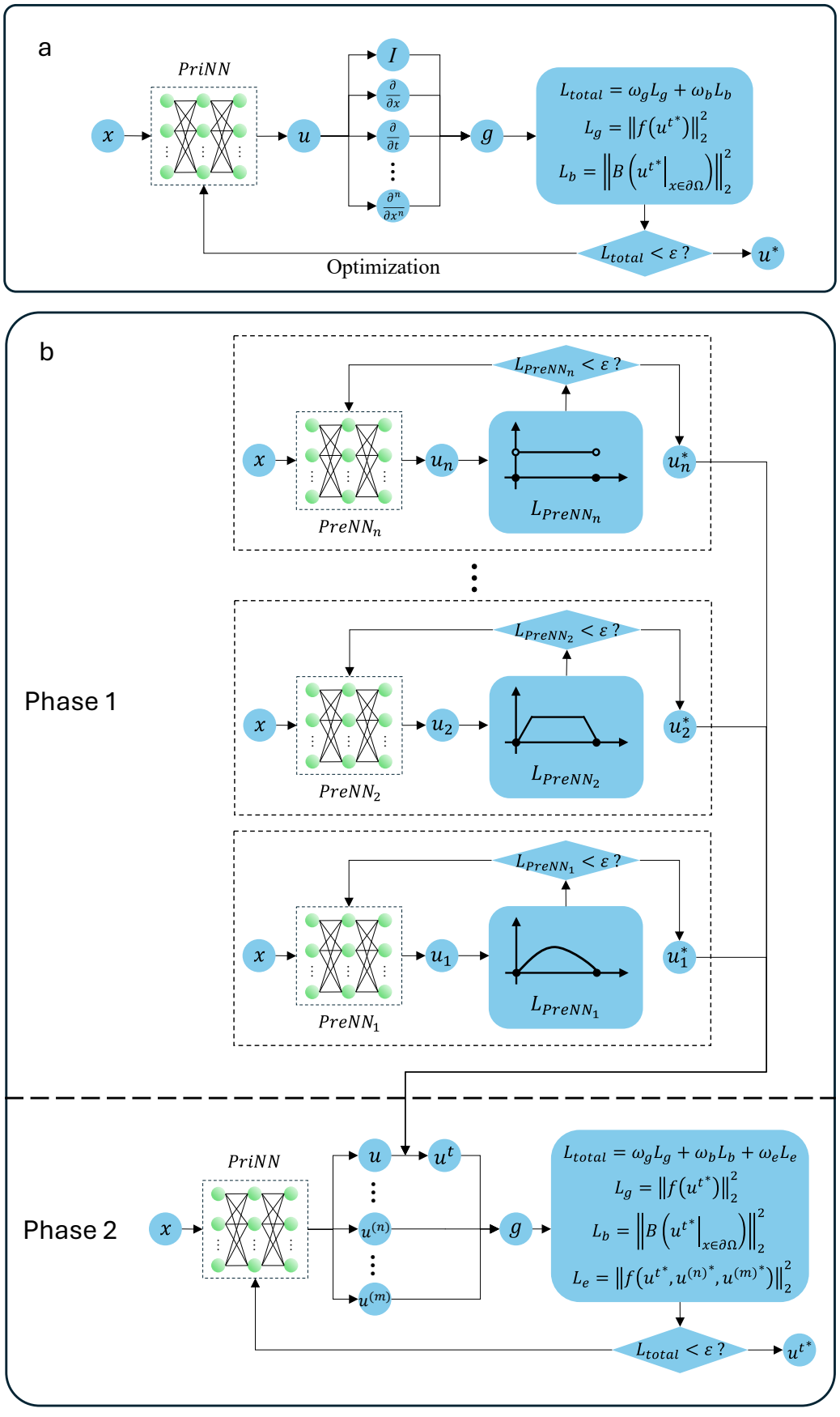


Figure 5-1 The vanilla PINN framework and the TP-PINN framework

Next, a problem with Dirichlet and Neumann boundary conditions is taken as an example to detail the predetermined requirements that the *PreNN* should fulfill. In the case that the boundary of the domain can be expressed analytically as:

$$\partial\Omega: B(x,y) = 0 \tag{5-2}$$

with Dirichlet and Neumann boundary conditions as follows:

$$u(x,y)|_{(x,y)\in\partial\Omega} = a \tag{5-3}$$

$$\left.\frac{\partial u(x,y)}{\partial n}\right|_{(x,y)\in\partial\Omega} = b \tag{5-4}$$

If only the Dirichlet boundary condition [Equation (5-3)] is applied, the output of the *PriNN* can be transformed as:

$$u^t(x,y) = u(x,y)\cdot PreNN_1(x,y) + a \tag{5-5}$$

where $u(x,y)$ is the original output of *PriNN*, $PreNN_1(x,y)$ is the *PreNN*, and $u^t(x,y)$ is the transformed output. The boundary values of $u^t(x,y)$ can be computed as:

$$u^t(x,y)|_{(x,y)\in\partial\Omega} = u(x,y)|_{(x,y)\in\partial\Omega}\cdot PreNN_1(x,y)|_{(x,y)\in\partial\Omega} + a \tag{5-6}$$

To ensure $u^t(x,y)|_{(x,y)\in\partial\Omega}$ satisfies Equation (5-3), the $PreNN_1(x,y)$ should fulfill the following requirement:

$$PreNN_1(x,y)|_{(x,y)\in\partial\Omega} = 0 \tag{5-7}$$

Then, the transformed output $u^t(x,y)$ automatically satisfies the Dirichlet boundary condition.

If both the Dirichlet boundary condition [Equation (5-3)] and the Neumann boundary condition [Equation (5-4)] are applied, the output of the *PriNN* can be transformed as:

$$u^t(x,y) = u(x,y)\cdot PreNN_1(x,y) + b\cdot PreNN_2(x,y) + a \tag{5-8}$$

where $PreNN_1(x,y)$ and $PreNN_2(x,y)$ are two *PreNN*s. The first-order derivative of $u^t(x,y)$ is computed as:

$$(u^t)'(x,y) = \frac{\partial u(x,y)}{\partial n} \cdot PreNN_1(x,y) + u(x,y) \cdot \frac{\partial PreNN_1(x,y)}{\partial n} + b \cdot \frac{\partial PreNN_2(x,y)}{\partial n} \quad (5-9)$$

Therefore, $PreNN_1(x,y)$ should fulfill two requirements, one is Equation (5-7), and the other is:

$$\left.\frac{\partial PreNN_1(x,y)}{\partial n}\right|_{(x,y)\in\partial\Omega} = 0 \quad (5-10)$$

Also, $PreNN_2(x,y)$ should fulfill two requirements:

$$PreNN_2(x,y)|_{(x,y)\in\partial\Omega} = 0 \quad (5-11)$$

$$\left.\frac{\partial PreNN_2(x,y)}{\partial n}\right|_{(x,y)\in\partial\Omega} = 1 \quad (5-12)$$

For cases with higher order boundary conditions, the construction of the transformed output can be performed similarly to Equation (5-5) and Equation (5-8). Also, the *PreNN*s can be obtained by calculating their derivatives and fulfilling corresponding requirements as the procedure shown above.

5.2.2 Loss Functions and Training in Two Phases

The loss function for the *PreNN* in the first training phase can be constructed as follows:

$$L_{PreNN_i} = \sum_j \sum_k \left| u_i^{(j)}(x_k,y_k)\Big|_{(x_k,y_k)\in\partial\Omega} - c_{ij} \right|^2 + \sum_s \left| u_i(x_s,y_s)|_{(x_s,y_s)\in S} - c_i \right|^2 \quad (5-13)$$

where $u_i(x,y)$ is the output of *i*-th *PreNN*, j represents the order of derivatives of *PreNN*, x_k and y_k are collocation points uniformly sampled on the boundary $\partial\Omega$, x_s and y_s are

collocation points in the set S sampled within the domain, c_{ij} is a constant that make *PreNN* fulfill the requirements of Equations (5-7), (5-10), (5-11), and (5-12), c_i is a positive constant.

The first penalty term of Equation (5-13) is to make *PreNN* satisfy the requirements on the boundary derived in Section 3.2 and further make the transformed output in the second training phase automatically satisfy all or part of boundary conditions. The collocation points on the boundary can be sampled densely based on the expression of the boundary [Equation (5-2)]. The aim of the second penalty term of Equation (5-13) is to ensure that *PreNN* is positive and a smooth surface within the domain, which will not bring additional computational difficulty for the second training phase. The collocation points within the domain can be sampled uniformly and sparsely at a certain distance from the boundary and included in set S. The number of collocation points and the value of c_i are determined on a case-by-case basis, e.g. one collocation point on the center of the domain is used and c_i is chosen as 0.1 (see Figure 5-3(b)) in the beam examples, while 20 collocation points are used and c_i is chosen as 0.025 (see Figure 5-4(c)) and 0.08 (see Figure 5-5(a)) in the plate examples. Although there are two target penalty terms to be optimized, only the first one should be strictly satisfied. The other one can be loosely fulfilled because there are no specific requirements on the shape of the *PreNN*. Therefore, the training process of the first phase is very simple and costs little computation time.

The loss function in the second training phase is composed of different penalty terms, which are defined as the MSE of residuals of the governing equation, the boundary conditions, and the relationship between the extra output and the transformed output.

Before constructing the penalty terms for governing equations, extra outputs are

introduced to reduce the order of governing equations. The extra output in the beam case is as follows:

$$\bar{u}_{pred}^{t}(x) = \frac{\partial^2 u_{pred}^{t}(x)}{\partial x^2} \tag{5-14}$$

where $u_{pred}^{t}(x)$ is the transformed output of the primary NN *PriNN* in the second training phase according to Equation (5-8), and $\bar{u}_{pred}^{t}(x)$ represents the second-order derivative of $u_{pred}^{t}(x)$. The extra outputs in the plate case are as follows:

$$\bar{w}_{pred_x}^{t}(x,y) = \frac{\partial^2 w_{pred}^{t}(x,y)}{\partial x^2} \tag{5-15}$$

$$\bar{w}_{pred_y}^{t}(x,y) = \frac{\partial^2 w_{pred}^{t}(x,y)}{\partial y^2} \tag{5-16}$$

where $w_{pred}^{t}(x,y)$ is the transformed output of *PriNN* according to Equation (5-8), $\bar{w}_{pred\_x}^{t}(x,y)$ and $\bar{w}_{pred\_y}^{t}(x,y)$ represent the second-order derivative of $w_{pred}^{t}(x,y)$ in x and y directions, respectively. Then, the governing equations are updated from Equations (5-27) and (5-31) in Section 5.3 to the following:

$$\frac{\partial^2}{\partial x^2}\left(EI(x)\bar{u}(x)\right) = q(x) \tag{5-17}$$

$$\frac{\partial^2 \bar{w}_x(x,y)}{\partial x^2} + 2\frac{\partial^2 \bar{w}_x(x,y)}{\partial y^2} + \frac{\partial^2 \bar{w}_y(x,y)}{\partial y^2} = \frac{q(x,y)}{D} \tag{5-18}$$

where $\bar{u}(x)$ is the second-order derivative of $u(x)$, $\bar{w}_x(x,y)$ and $\bar{w}_y(x,y)$ are the second-order derivatives of $w(x,y)$ in x and y directions.

Based on the above transformation, the penalty terms for the governing equations of the beam (MSE_g^B) and the plate (MSE_g^P) are as follows:

$$MSE_g^B = \sum_{i=1}^{N_g^B} \left| \frac{\partial^2}{\partial x_i^2}\left(EI(x_i)\bar{u}_{pred}^{t}(x_i)\right) - q(x_i) \right|^2 \tag{5-19}$$

$$MSE_g^P = \sum_{i=1}^{N_g^P} \left| \frac{\partial^2 \bar{w}_{pred_x}^t(x_i, y_i)}{\partial x_i^2} + 2\frac{\partial^2 \bar{w}_{pred_x}^t(x_i, y_i)}{\partial y_i^2} + \frac{\partial^2 \bar{w}_{pred_y}^t(x_i, y_i)}{\partial y_i^2} - \frac{q(x_i, y_i)}{D} \right|^2 \quad (5-20)$$

where N_g^B and N_g^P are numbers of collocation points within the domain of the beam and plate cases, respectively.

The penalty terms for boundary conditions can be totally waived in the cases of beam and plate with fixed boundaries due to their continuously ordered boundary conditions. However, if their boundaries are simply supported, which means the existence of discontinuously ordered boundary conditions, only the first few boundary conditions with continuous orders can be automatically satisfied. The rest should be softly enforced by penalty terms of the beam (MSE_b^B) and the plate (MSE_b^P) as follows:

$$MSE_b^B = \left| \left(u_{pred}^t\right)''(0) \right|^2 + \left| \left(u_{pred}^t\right)''(l) \right|^2 \quad (5-21)$$

$$MSE_b^P = \sum_{i=1}^{N_b^P} \left| \frac{\partial^2 w_{pred}^t(x_i, y_i)}{\partial n_i^2}\bigg|_{(x_i, y_i) \in \partial\Omega} \right|^2 \quad (5-22)$$

The penalty terms for the relationship between the extra output and the transformed output of the beam (MSE_e^B) and the plate (MSE_e^P) can be established based on Equation (5-14) and Equations (5-15) and (5-16) as follows:

$$MSE_e^B = \sum_{i=1}^{N_g^B} \left| \bar{u}_{pred}^t(x_i) - \frac{\partial^2 u_{pred}^t(x_i)}{\partial x_i^2} \right|^2 \quad (5-23)$$

$$MSE_e^P = \sum_{i=1}^{N_g^P} \left(\left| \bar{w}_{pred_x}^t(x_i, y_i) - \frac{\partial^2 w_{pred}^t(x_i, y_i)}{\partial x_i^2} \right|^2 + \left| \bar{w}_{pred_y}^t(x_i, y_i) - \frac{\partial^2 w_{pred}^t(x_i, y_i)}{\partial y_i^2} \right|^2 \right) \quad (5-24)$$

Finally, the total loss function of the beam (L_{total}^B) and the plate (L_{total}^P) in the second

training phase can be built as follows:

$$L_{total}^{B} = MSE_{g}^{B} + MSE_{b}^{B} + MSE_{e}^{B} \quad (5-25)$$

$$L_{total}^{P} = MSE_{g}^{P} + MSE_{b}^{P} + MSE_{e}^{P} \quad (5-26)$$

in which, the penalty term MSE_{b}^{i} can be cancelled if all boundary conditions have been automatically satisfied by involving the *PreNN* in the first training phase.

By optimizing the total loss function in the second training phase, TP-PINN framework can finally solve the high-order differential problems with enhanced accuracy and efficiency compared to vanilla PINN. Although the penalty term for boundary conditions is simplified or waived in the second training phase, there exist more than one target penalty terms to be optimized simultaneously and fulfilled strictly. Thus, the computation time of the second phase is much longer than that of the first phase.

5.2.3 Neural Network and Training Parameters

All codes are written in Python based on the ML library PyTorch.

In beam cases, a NN containing three hidden layers with 20 neurons in each is used in the vanilla PINN framework and both the first and the second training phases of TP-PINN framework. Tanh is used as the activation function, and Adam with a learning rate of 10^{-3} is used as the optimizer. The training of the first phase of TP-PINN is terminated when the loss falls below 10^{-6} or when 5000 epochs have been reached. The training of both vanilla PINN and the second phase of TP-PINN is terminated when the loss falls below 10^{-6} or when 10000 epochs have been reached.

In plate cases, a NN containing five hidden layers with 20 neurons in each is used in the

first phase of TP-PINN, while a NN containing five hidden layers with 50 neurons in each is used in vanilla PINN and the second phase of TP-PINN. Tanh is used as the activation function, and Adam with a learning rate of 10^{-4} is used as the optimizer. The training of the first phase of TP-PINN is terminated when the loss drops below 10^{-6} or when 50000 epochs have been reached. The training of both vanilla PINN and the second phase of TP-PINN is terminated when the loss drops below 10^{-6} or when 1000000 epochs have been reached.

5.3 Numerical Case Studies

Building on the theoretical background described in Section 5.1, the vanilla PINN and TP-PINN frameworks are applied to two representative structural mechanics problems.

Deformation of an Euler-Bernoulli beam

In the first example, the deformation of a Euler-Bernoulli beam under variable distributed load (Figure 5-2(a)), which is typically governed by a high-order ODE, is examined. The material properties of the Euler-Bernoulli beam are as follows: the length of the beam $l = 1000\,mm$, the flexural rigidity $EI(x) = (0.5 + 0.1x^2) \times 10^6\ N \cdot mm^2$, and the variable distributed load $q(x) = 0.01 + 0.01sin(\pi x)\ N/mm$.

The deformation of a 1D Euler-Bernoulli beam is governed by the following fourth-order ODE:

$$\frac{\partial^2}{\partial x^2}\left(EI(x)\frac{\partial^2 u(x)}{\partial x^2}\right) = q(x), \qquad x \in [0, l] \tag{5-27}$$

where x is the coordinate, $u(x)$ is the deformation along the x-axis.

If both ends of the beam are fixed, there are two boundary conditions:

$$u(0) = u(l) = 0 \qquad (5-28)$$

$$u'(0) = u'(l) = 0 \qquad (5-29)$$

Equation (5-28) is the Dirichlet boundary condition, while Equation (5-29) is the Neumann boundary condition.

If both ends are simply supported, the Dirichlet boundary condition Equation (5-28) and another second-order boundary condition constraining the moment are applied:

$$u''(0) = u''(l) = 0 \qquad (5-30)$$

Deformation of a Kirchhoff-Love plate

In the second example, the deformation of a Kirchhoff-Love plate with irregular geometry under distributed load (Figure 5-3(a)), which involves a high-order PDE, is investigated. The plate is a square with a quarter circle removed from the lower-left corner of the square. The material properties of the Kirchhoff-Love plate are as follows: the side length of the square $l_1 = 1200\ mm$, the radius of the circle $r = 600\ mm$, the thickness of the plate $t = 10\ mm$, the Young's modulus $E = 2.1 \times 10^5\ MPa$, the Poisson's ratio $\mu = 0.3$, and the distributed load $q(x, y) = 0.5\ N/mm^2$.

The deformation of a 2D Kirchhoff-Love plate is expressed by a fourth-order PDE:

$$\frac{\partial^4 w(x,y)}{\partial x^4} + 2\frac{\partial^4 w(x,y)}{\partial x^2 \partial y^2} + \frac{\partial^4 w(x,y)}{\partial y^4} = \frac{q(x,y)}{D}, \qquad (x,y) \in \Omega \qquad (5-31)$$

where x and y are the coordinates, Ω is the domain of the plate, including the boundary $\partial\Omega$ and the internal area, $w(x, y)$ is the deformation in the domain, and D is a constant that can be computed by:

$$D = \frac{Et^3}{12(1-\mu^2)} \qquad (5-32)$$

If the boundary of the plate is clamped (fixed), the Dirichlet and Neumann boundary conditions are accordingly applied:

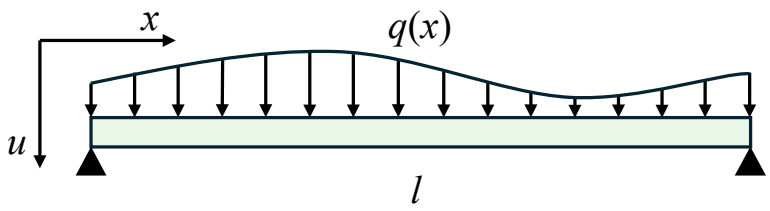
$$w(x,y)|_{(x,y)\in\partial\Omega} = 0 \tag{5-33}$$

$$\left.\frac{\partial w(x,y)}{\partial n}\right|_{(x,y)\in\partial\Omega} = 0 \tag{5-34}$$

where n is the normal vector on the boundary.

If the boundary of the plate is simply supported, the Dirichlet boundary condition Equation (5-33) and the second-order boundary condition are applied:

$$\left.\frac{\partial^2 w(x,y)}{\partial n^2}\right|_{(x,y)\in\partial\Omega} = 0 \tag{5-35}$$

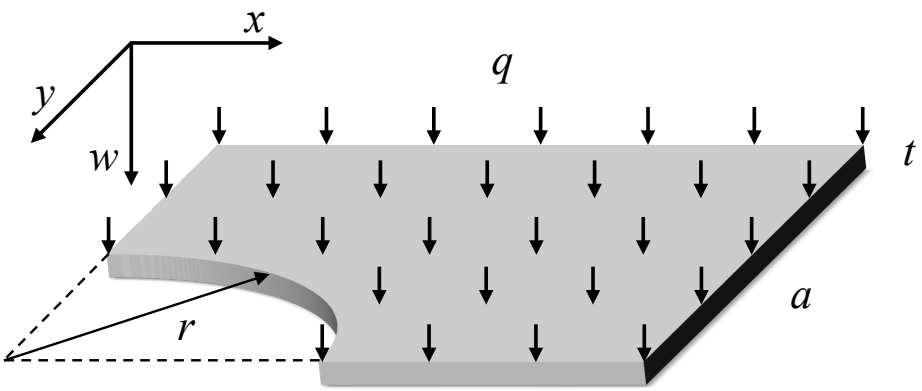


(a) Euler-Bernoulli beam under distributed load

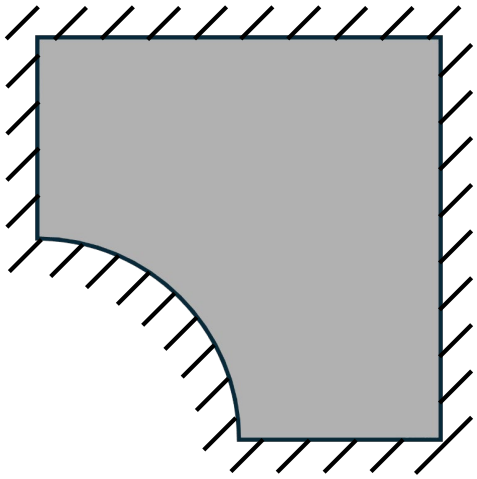
(b) Fixed-end boundary condition

(c) Simply-supported boundary condition

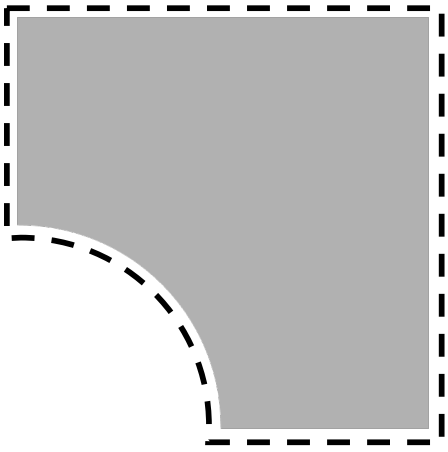
Figure 5-2 Illustration of an Euler-Bernoulli beam



(a) Kirchhoff-Love plate under distributed load



(b) Clamped (fixed) boundary condition



(c) Simply-supported boundary condition

Figure 5-3 Illustration of a Kirchhoff-Love plate

5.3.1 Results of Euler-Bernoulli Beam Example

First, the vanilla PINN and the TP-PINN frameworks are applied to solve the deformation of Euler-Bernoulli beams under two different boundary conditions: fixed and simply-supported at both ends, as shown in Figures 5-2(b) and 5-2(c). FE results are used as reference solutions to compute the relative L_2 error on validation points of both the vanilla PINN framework and the TP-PINN framework, allowing for an assessment of their solution accuracy. The relative L_2 error can be computed by:

$$Relative\ L_2\ error = \frac{\sqrt{\sum_i \left|u_{exact}^i - u_{pred}^i\right|^2}}{\sqrt{\sum_i \left|u_{exact}^i\right|^2}} \qquad (5-36)$$

where u_{exact}^i denotes the reference solution computed by FEM, and u_{pred}^i denotes the predicted solution by PINN or TP-PINN. Additionally, the training time per 100 epochs for both the vanilla PINN and the second phase of TP-PINN is recorded to evaluate their computational efficiency. As stated in Section 5.2.2, the computation time of the first phase of TP-PINN is much shorter than that of the second phase that is negligible, it is not counted in the comparison of computational efficiency.

In vanilla PINN framework, collocation points are commonly sampled uniformly and randomly. However, collocation points are sampled using a modified method in this study. In addition to uniform random sampling, points near the boundaries are sampled more densely, yielding a total of 200 collocation points over the entire domain, as illustrated in Figure 5-4. Detailed discussion of various sampling methods is provided in Section 5.4.1. The *PreNN*s for the fixed-end beam and the simply-supported beam are depicted in Figures 5-5 and 5-8,

respectively, and the conditions they should satisfy, along with the specific training procedures, have been derived and explained comprehensively in Sections 5.2.1 and 5.2.2.

Figures 5-6 and 5-9 display the deformation of the beams computed by the FEM, the vanilla PINN, and TP-PINN under two boundary conditions. Figures 5-7 and 5-10 illustrate absolute errors compared to the reference FE solution for both the vanilla PINN and TP-PINN under two boundary conditions. The relative L_2 errors and training time are summarized in Table 5-1. The relative L_2 errors between the vanilla PINN and the FE solution are calculated based on 100 evenly distributed points on the beam, they are 2.10×10^{-3} and 1.19×10^{-3} for fixed and simply-supported conditions, whereas those between TP-PINN and the FE solution are 2.55×10^{-5} and 1.57×10^{-5}, demonstrating a marked improvement in computational accuracy with TP-PINN. Moreover, by examining the regions near the boundaries in Figures 5-7 and 5-10, it reveals that the hard embedding of boundary conditions significantly enhances accuracy in these areas.

In addition, the average training time per 100 epochs for the TP-PINN is 3.24 seconds and 3.87 seconds in fixed-end and simply-supported cases, compared to 3.54 seconds and 4.18 seconds for the vanilla PINN, corresponding to an efficiency increase of approximately 8%. These results indicate that by partitioning the training into two independent phases to separately address the boundary conditions and other requirements, as well as reducing the order of the governing equation, training efficiency can be slightly improved. Although the training efficiency is improved, the improvement is not significant due to the additional computation required for the *PreNN* during backpropagation, which slightly impacts overall efficiency. Another observation is that, when comparing cases with different boundary conditions, the

simply-supported beam case requires more computation time than the fixed-end beam case. Despite using the same number of collocation points, only the Dirichlet boundary condition of the simply-supported beam has been automatically satisfied, which results in one more penalty term in the loss function compared to the fixed-end case and consequently increases the computation time.

Table 5-1 Relative L_2 error and training time of vanilla PINN and TP-PINN for beam cases

Boundary conditions	Fixed-end beam		Simply-supported beam	
Methods	Vanilla PINN	TP-PINN	Vanilla PINN	TP-PINN
Relative L_2 error	2.10×10^{-3}	2.55×10^{-5}	1.19×10^{-3}	1.57×10^{-5}
Training time per 100 epochs (s)	3.54	3.24	4.18	3.87

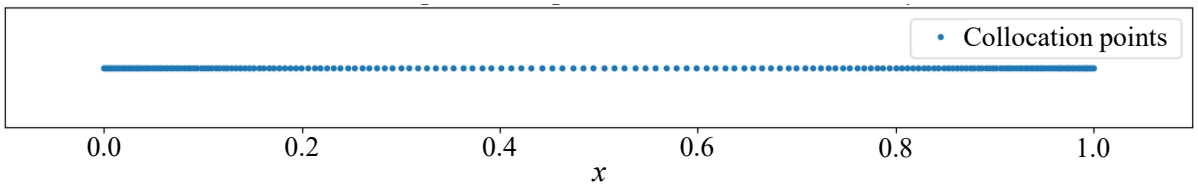


Figure 5-4 Collocation points of the Euler-Bernoulli beam case

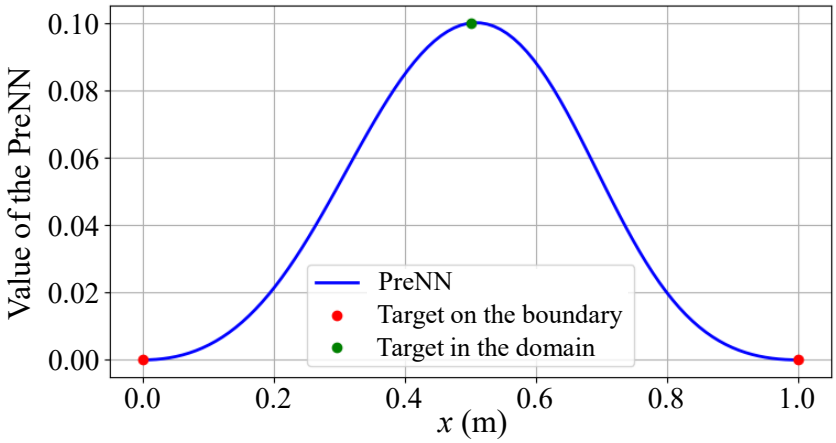


Figure 5-5 The PreNN for the fixed-end beam

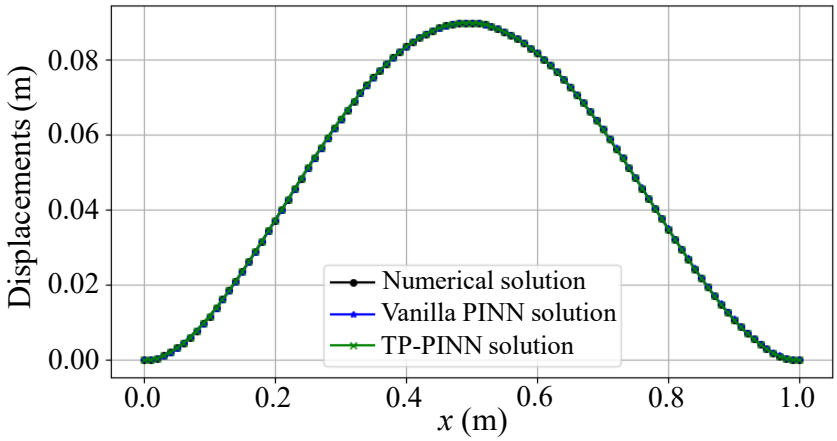


Figure 5-6 The displacements of the fixed-end beam

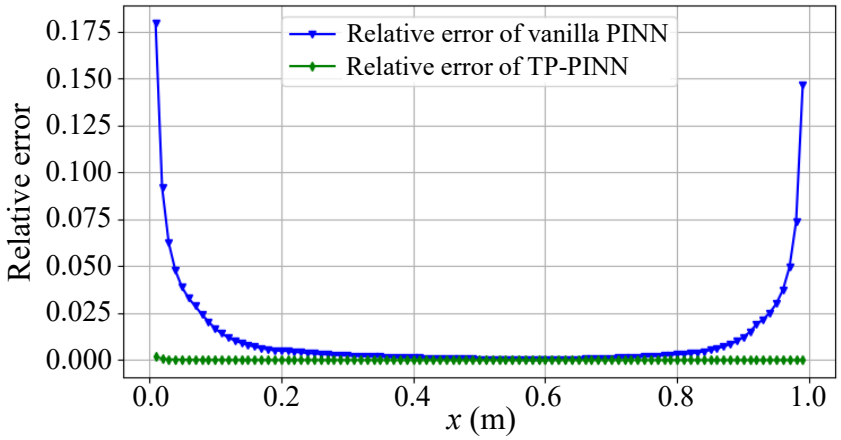


Figure 5-7 The relative errors of vanilla PINN and TP-PINN solutions for the fixed-end beam

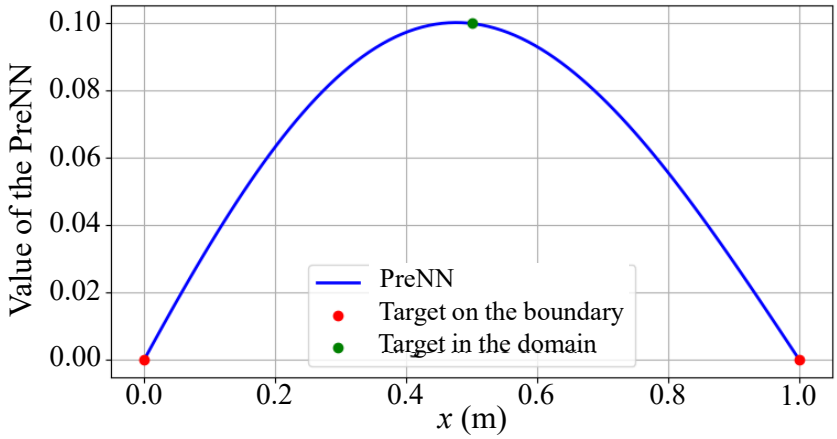


Figure 5-8 The PreNN for the simply-supported beam

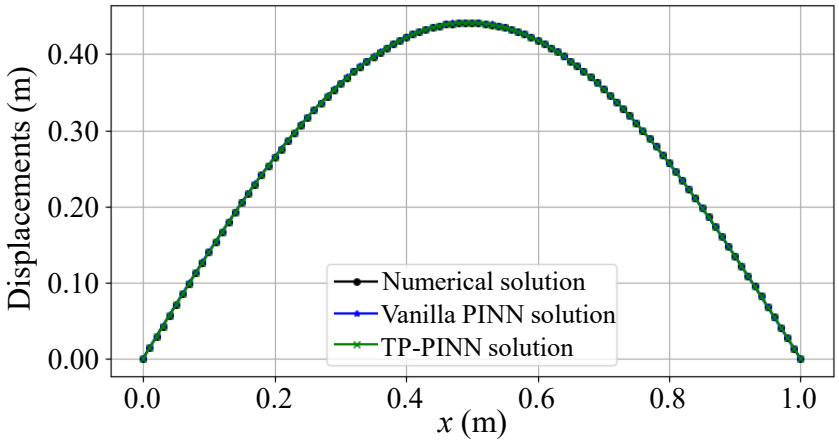


Figure 5-9 The displacements of the simply-supported beam

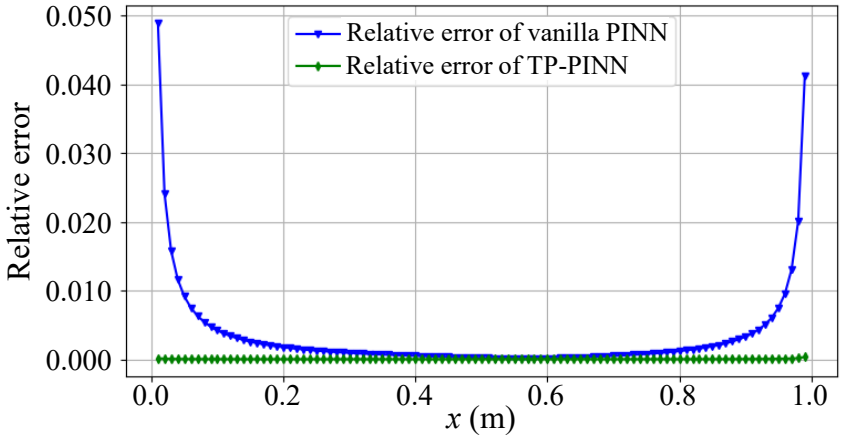


Figure 5-10 The relative errors of vanilla PINN and TP-PINN solutions for the simply-supported beam

5.3.2 Results of Kirchhoff-Love Plate Example

Next, both frameworks are applied to solve the deformation of Kirchhoff-Love plates with irregular boundary, under two different boundary conditions: clamped and simply supported on the boundary, as shown in Figures 5-3(b) and 5-3(c). As in the previous example, FE results serve as the reference solutions for computing the relative L_2 errors of the various frameworks. The training time per 100 epochs is recorded to evaluate computational efficiency.

The collocation points of *PreNN* (as shown in Figure 5-12) and *PriNN* (as shown in Figure 5-14) for both cases are the same. In both Figures 5-12 and 5-14, there are 32 points on Boundaries 1 and 4 each and 64 points on Boundaries 2, 3, and 5 each. For collocation points inside the domain, there are 20 points uniformly sampled at a certain distance from the boundary for the *PreNN* as illustrated in Figure 5-12, while a modified method is employed to achieve a denser sampling near the boundaries, yielding 2048 points for the *PriNN* as shown in Figure 5-14. The *PreNN*s are three-dimensional functions, as shown in Figure 5-13 and Figure 5-20. The conditions they should satisfy as well as the specific training procedures have been detailed in Section 5.2.1.

Figure 5-11 and Figure 5-19 show the FE solutions for both cases. Figure 5-15 and Figure 5-21 show the deformation of the plates with two boundary conditions computed by the vanilla PINN, while absolute errors compared to FE solutions are illustrated in Figure 5-16 and Figure 5-22, respectively. Figure 5-17 and Figure 5-23 show the results computed by the TP-PINN of two cases, while absolute errors compared to FEM are presented in Figure 5-18 and Figure 5-24. The relative L_2 errors and training time are summarized in Table 5-2. The relative L_2 errors between the vanilla PINN and the FEM solution are measured based on 186324 points in the domain, computed as 0.142 and 0.184 for clamped and simply-supported conditions, whereas those for TP-PINN are 0.0108 and 0.0180. Moreover, results reveal that the absolute errors near the boundaries are smaller for TP-PINN. These results indicate that TP-PINN, with its hard-embedding approach, achieves higher computational accuracy.

In addition, the average training time per 100 epochs for the TP-PINN is 7.76 seconds and 8.61 seconds in clamped and simply-supported cases, while 10.02 seconds and 11.56 seconds

for the vanilla PINN, corresponding to an efficiency improvement of TP-PINN of around 24% compared with the vanilla PINN. Hence, TP-PINN can also effectively enhance training efficiency. Compared to the improvement in computational efficiency observed in the beam examples (about 8%), the plate examples more clearly demonstrate the effectiveness of TP-PINN in enhancing computational efficiency. Although the PDE governing plate deformation is more complex than the ODE for beam deformation, reducing the order of the governing equations and employing two-phase training yields even more significant enhancement in efficiency. While additional computation is required for the *PreNN* during backpropagation, the cost of this extra computation is outweighed by the positive impact of TP-PINN on efficiency. Besides, similar to the beam cases, the clamped case with fewer penalty terms computes faster than the simply-supported case.

Table 5-2 Relative L_2 error and training time of vanilla PINN and TP-PINN for plate cases

Boundary conditions	Clamped plate		Simply-supported plate	
Methods	Vanilla PINN	TP-PINN	Vanilla PINN	TP-PINN
Relative L_2 error	0.142	0.0108	0.184	0.0180
Training time per 100 epochs (s)	10.02	7.76	11.56	8.61

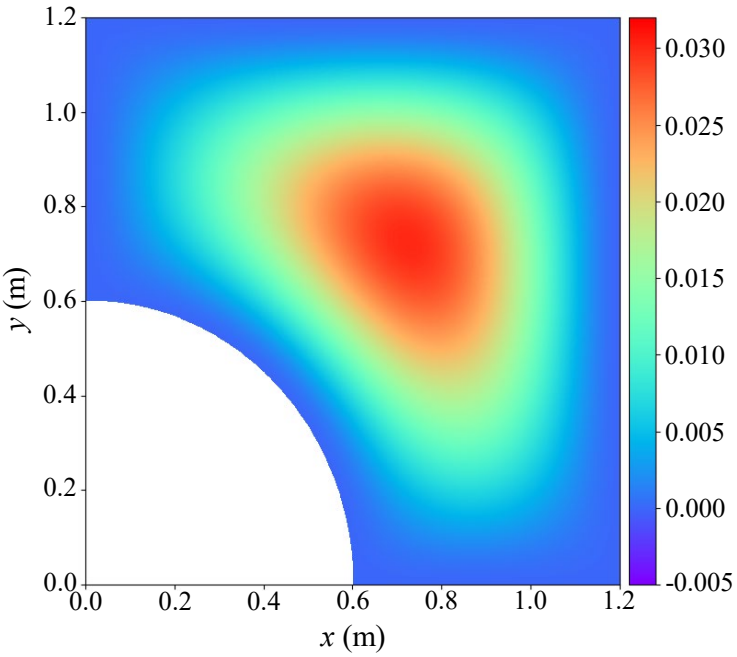


Figure 5-11 The FEM solution for the clamped plate

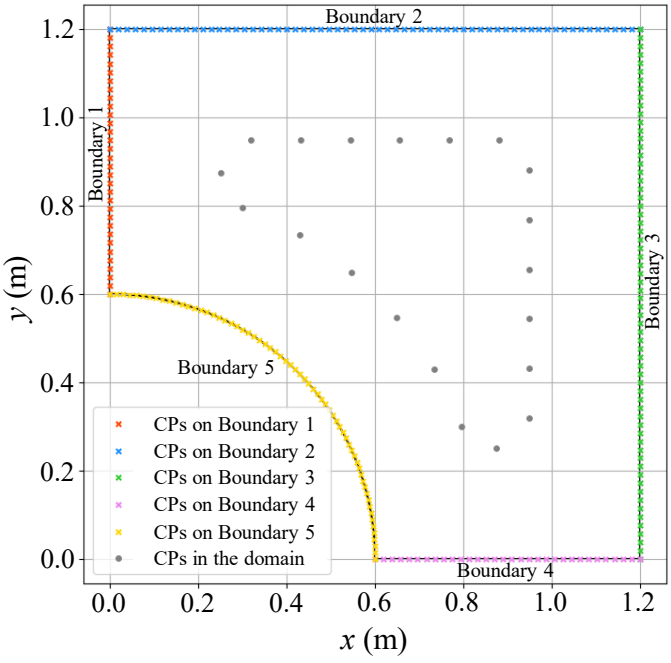


Figure 5-12 The collocation points of the PreNN

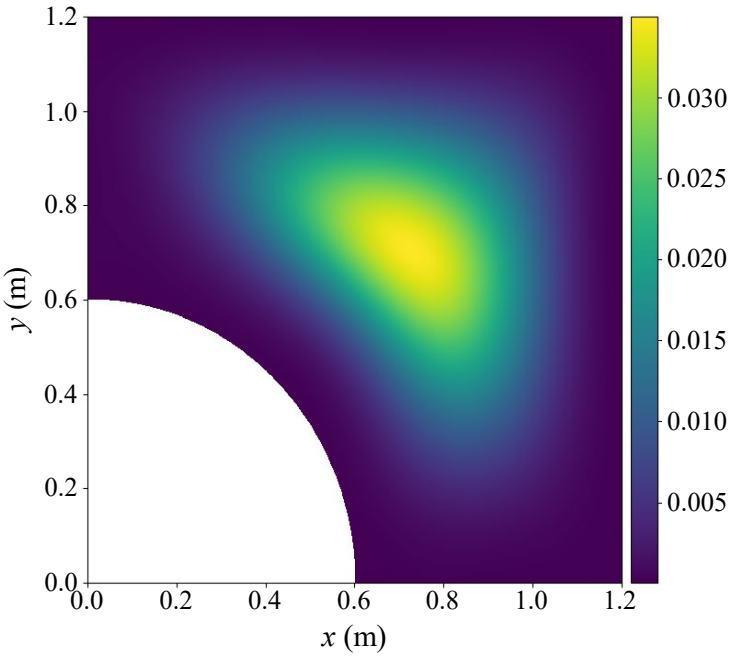


Figure 5-13 The PreNN for the clamped plate

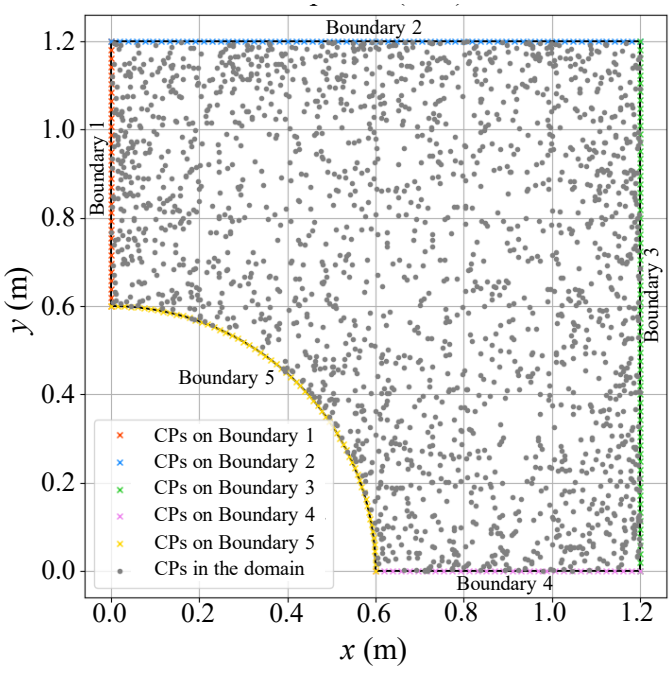


Figure 5-14 The collocation points of the PriNN

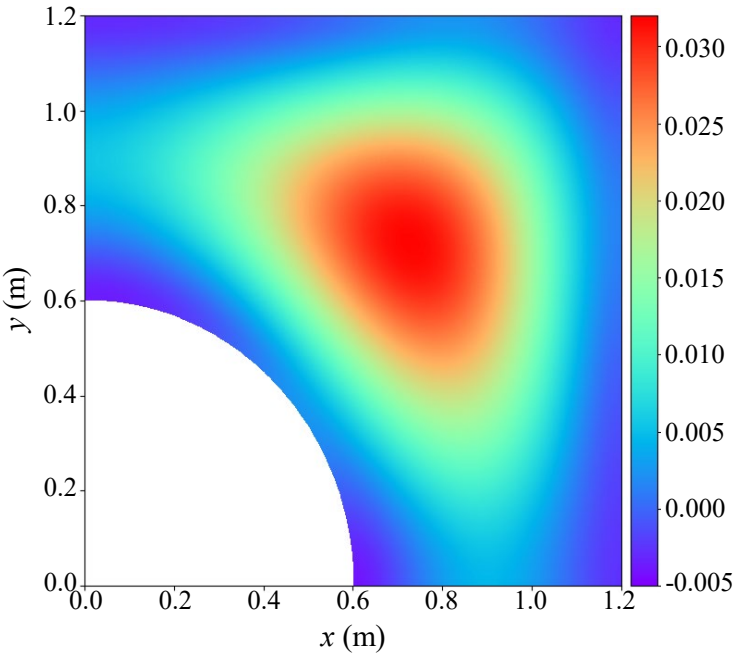


Figure 5-15 The PINN solution for the clamped plate

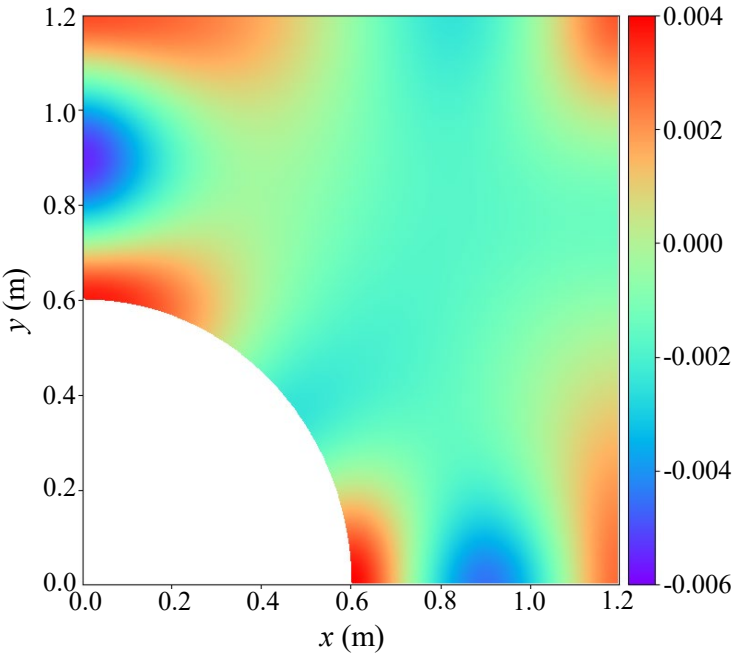


Figure 5-16 The absolute error of PINN for the clamped plate

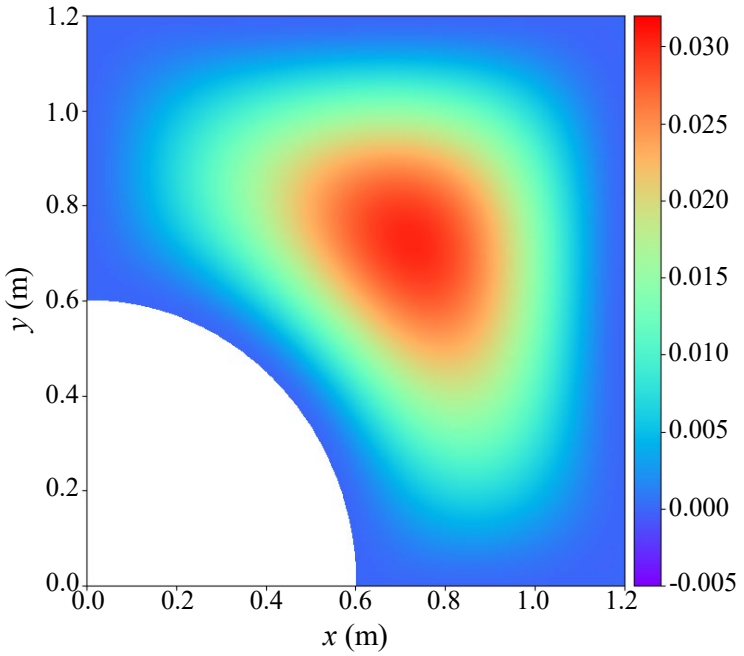


Figure 5-17 The TP-PINN solution for the clamped plate

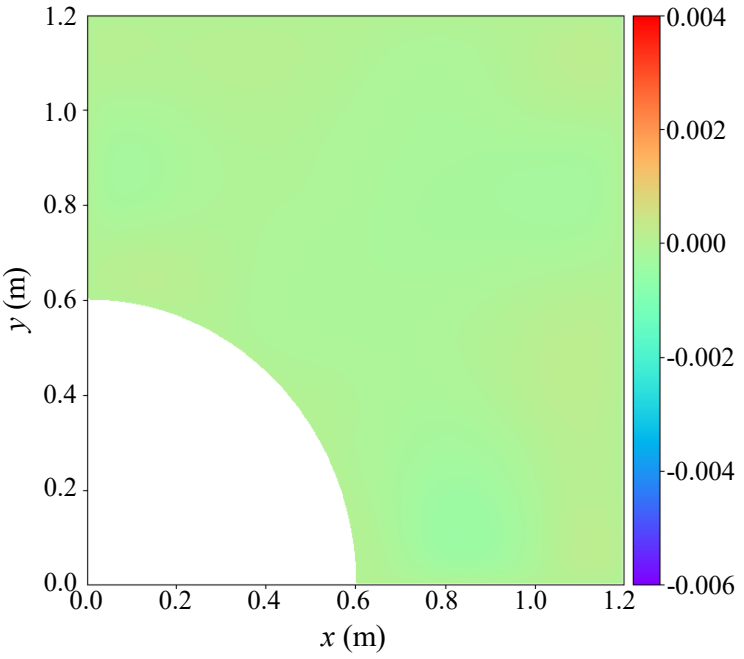


Figure 5-18 The absolute error of TP-PINN for the clamped plate

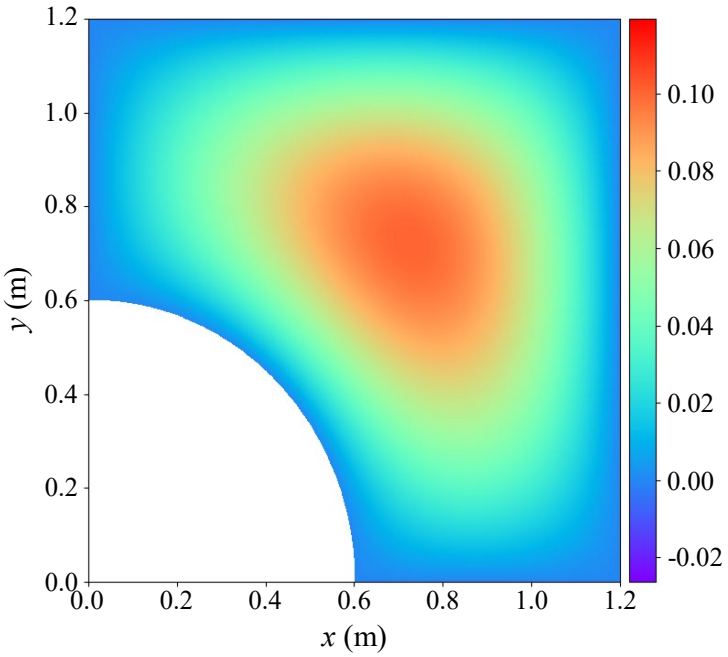


Figure 5-19 The FEM solution for the simply-supported plate

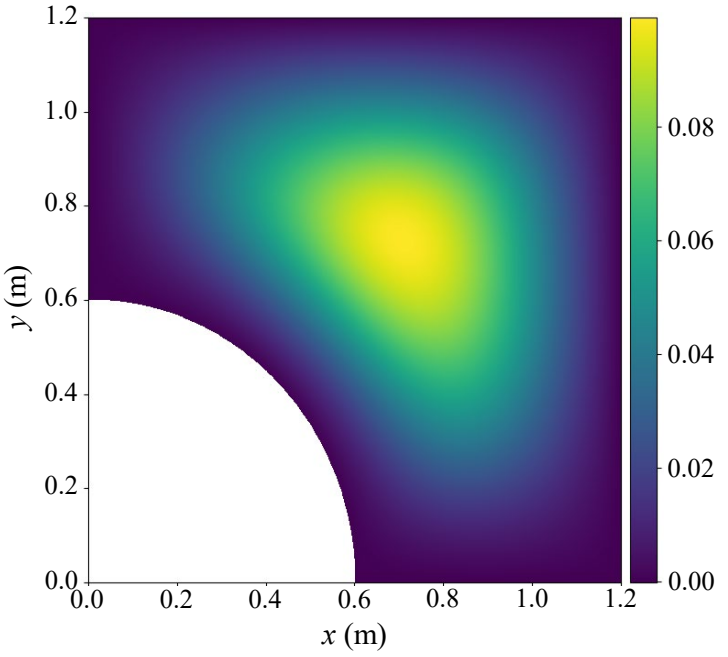


Figure 5-20 The PreNN for the simply-supported plate

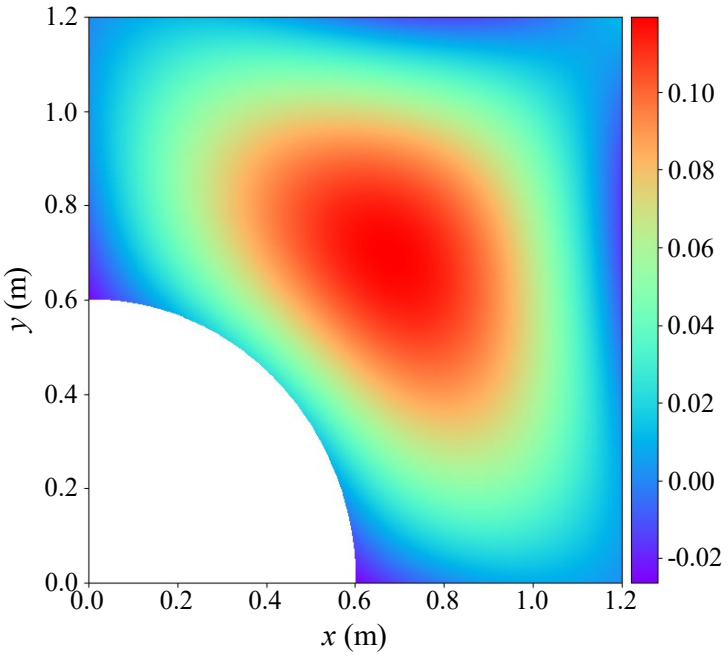


Figure 5-21 The PINN solution for the simply-supported plate

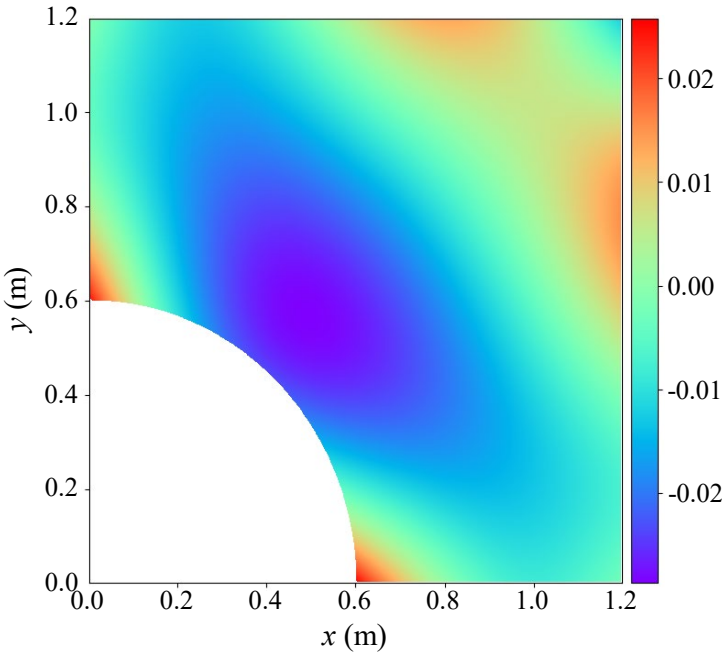


Figure 5-22 The absolute error of PINN for the simply-supported plate

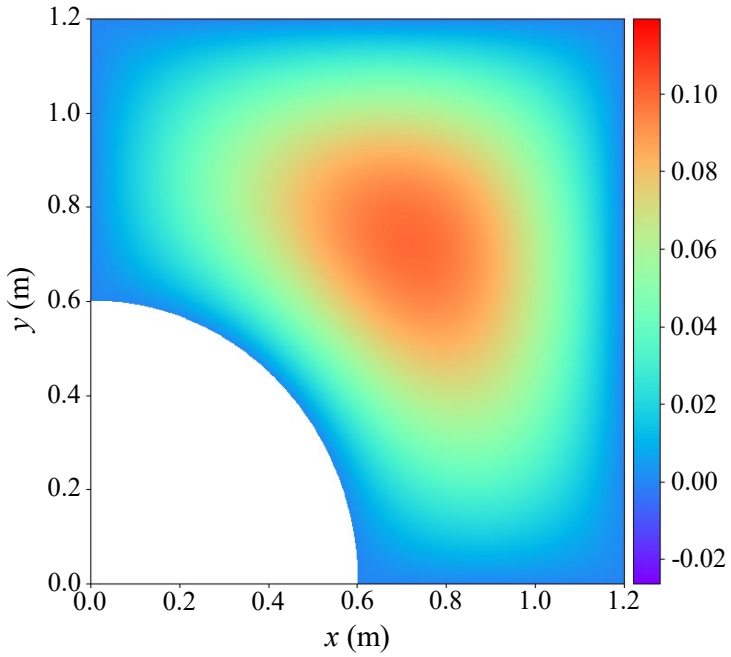


Figure 5-23 The TP-PINN solution for the simply-supported plate

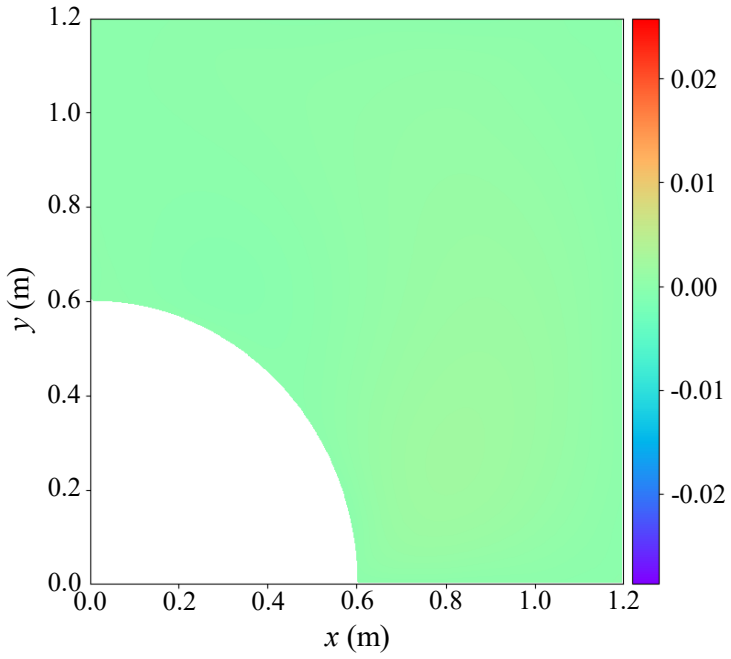


Figure 5-24 The absolute error of TP-PINN for the simply-supported plate

5.4 Discussion

In the numerical examples of this study, various collocation points sampling methods and forms of *PreNN*s are available. To compare their respective performances and to clarify the rationale behind the choices made for the examples, this section provides a detailed discussion of these approaches. In addition, the limitations of the method proposed in this paper are explained and discussed.

5.4.1 Collocation Points Sampling Methods

In the vanilla PINN framework, uniform random sampling for collocation points over the entire domain is commonly performed to ensure that the collocation points densely cover the whole domain. However, such a sampling method does not necessarily guarantee optimal computational results. For example, if the absolute error is larger near the boundaries, it becomes necessary to increase the density of collocation points in those regions. Another potential issue is that, in practical applications, achieving dense sampling may not be feasible, which could adversely affect computational accuracy. This section compares and discusses the performance of various sampling methods in the TP-PINN framework based on the simply-supported plate example.

Figures 5-25, 5-28, and 5-31 illustrate three sampling methods different from the method used in Section 5.3.2, including uniform random sampling over the entire domain (S1), dense sampling away from the boundaries (S2), and sparse sampling (S3), with 2048, 2048, and 200 collocation points in total, respectively. Figures 5-26, 5-29, and 5-32 display deformation results under different sampling methods, and Figures 5-27, 5-30, and 5-33 show absolute

errors compared to the FE solution, respectively. The relative L_2 errors and training time per 100 epochs for each sampling method are presented in Table 5-3. For computational accuracy, dense sampling near the boundaries (Figure 5-14) yields the highest accuracy (0.0180) compared with all cases with S1, S2, and S3, demonstrating that the appropriate choice of collocation point sampling can enhance computational precision. Under conditions of sparse sampling, accuracy is the lowest, indicating that computational accuracy relies on both the number and density of the collocation points.

For computation efficiency, the training time for S1 and S2 is roughly equivalent to that of TP-PINN for simply-supported plate (8.61 seconds), since the number of collocation points and other configurations used in different schemes are the same. It is worth noting that the training time for S3 is the shortest due to the smallest amount of collocations points is used, illustrating the tradeoff between computational accuracy and efficiency.

Table 5-3 Relative L_2 error and training time of TP-PINN with different sampling methods for the simply supported plate case

Sampling methods	S1	S2	S3
Relative L_2 error	0.0495	0.0822	0.165
Training time per 100 epochs (s)	8.55	8.63	6.40

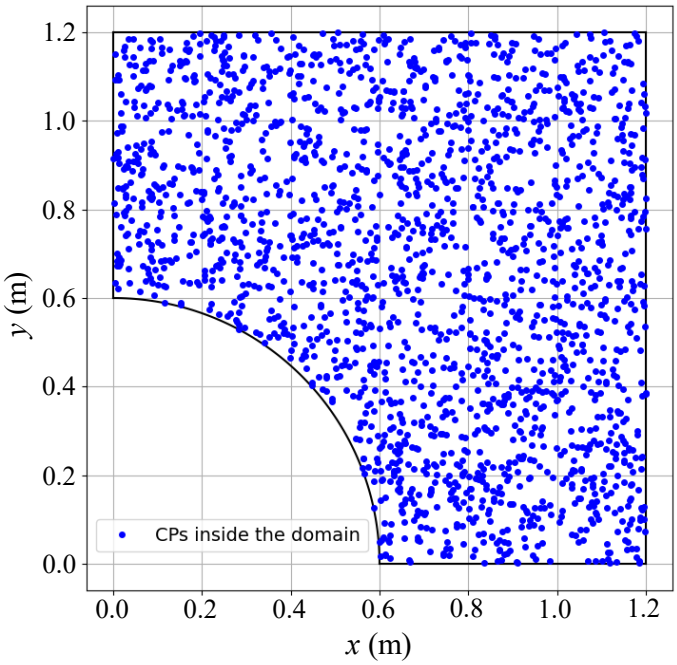


Figure 5-25 Collocation points using sampling method S1

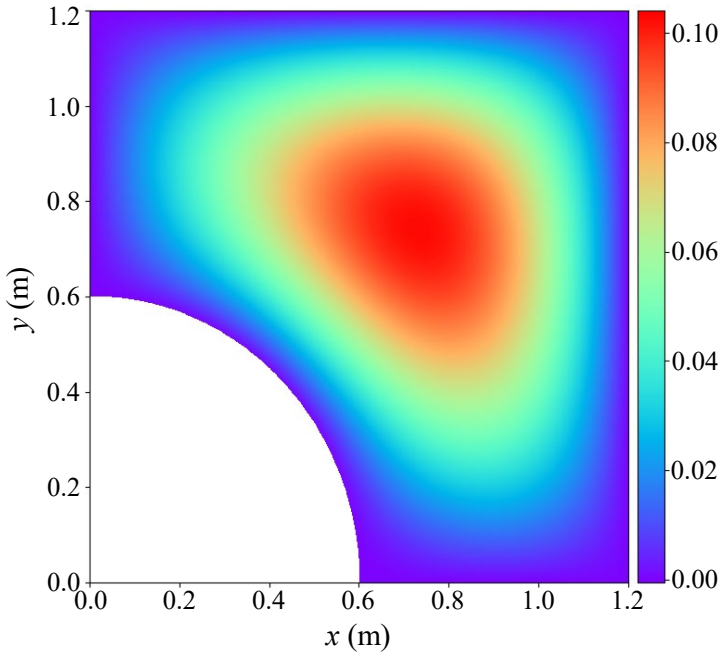


Figure 5-26 The TP-PINN solution for sampling method S1

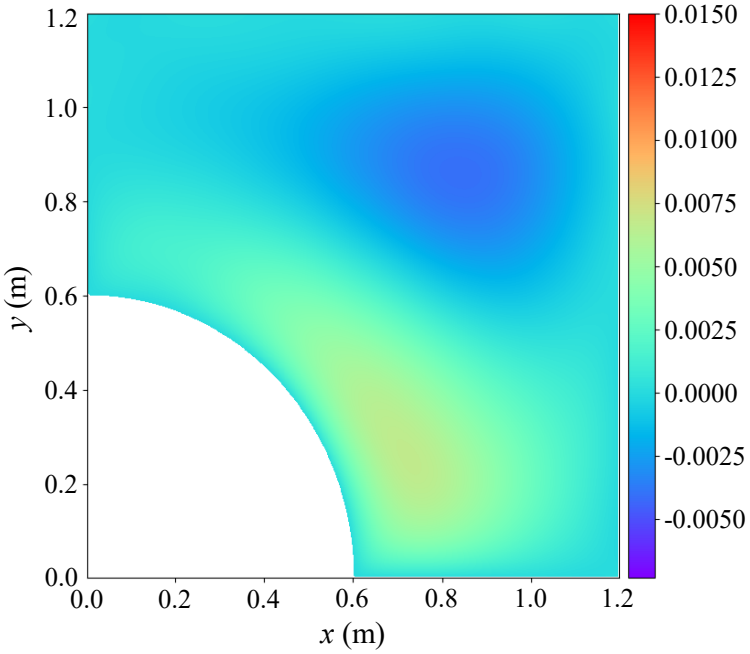


Figure 5-27 The absolute error of TP-PINN for sampling method S1

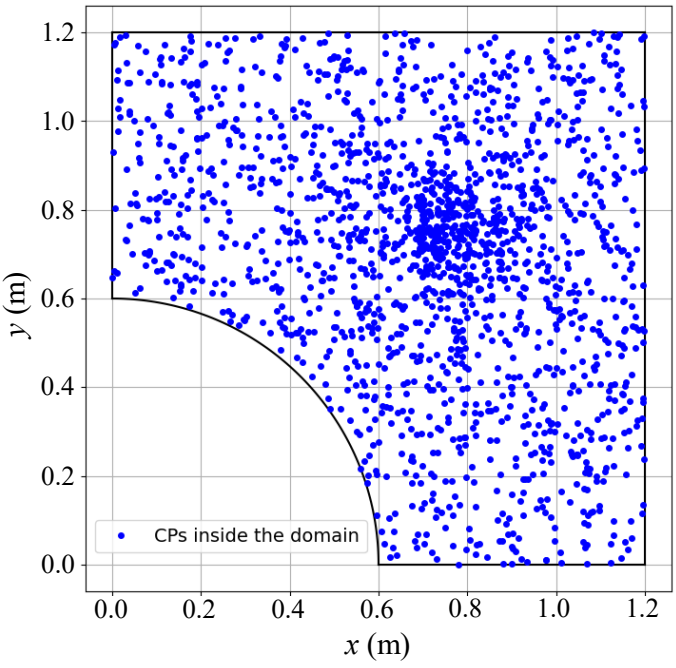


Figure 5-28 Collocation points using sampling method S2

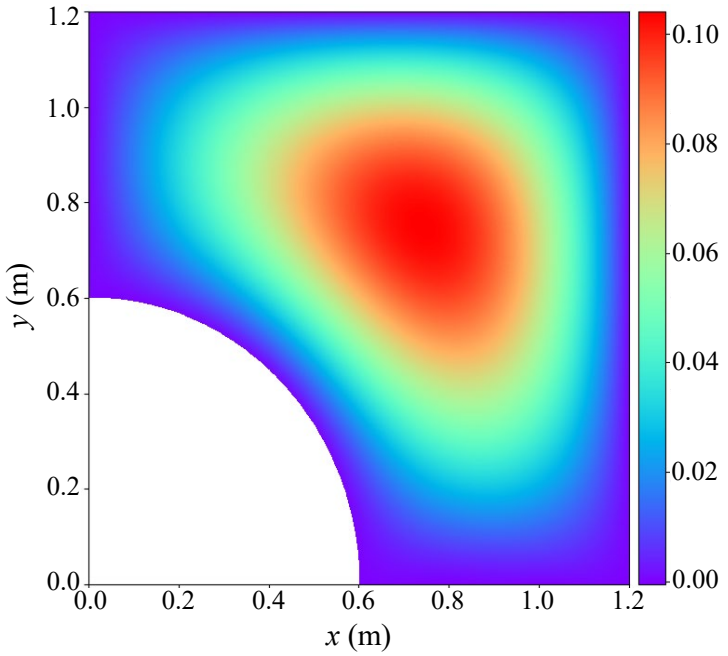


Figure 5-29 The TP-PINN solution for sampling method S2

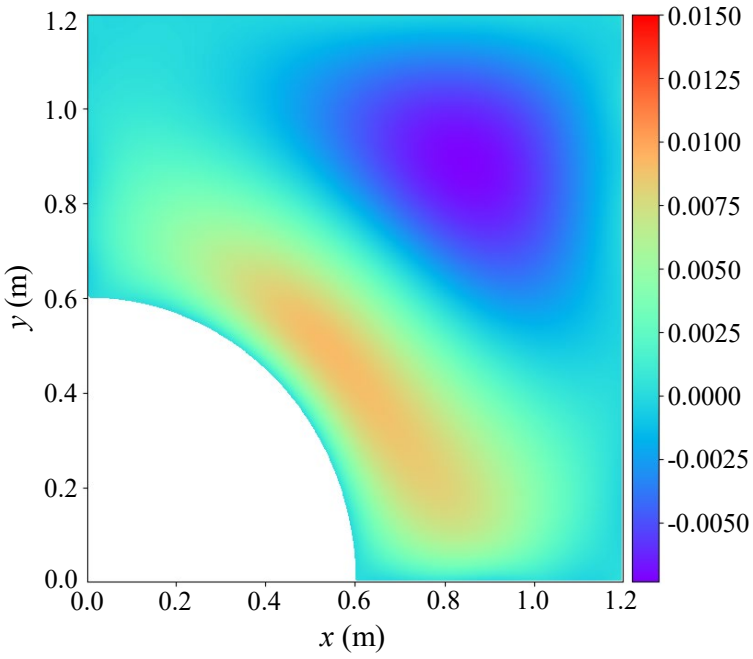


Figure 5-30 The absolute error of TP-PINN for sampling method S2

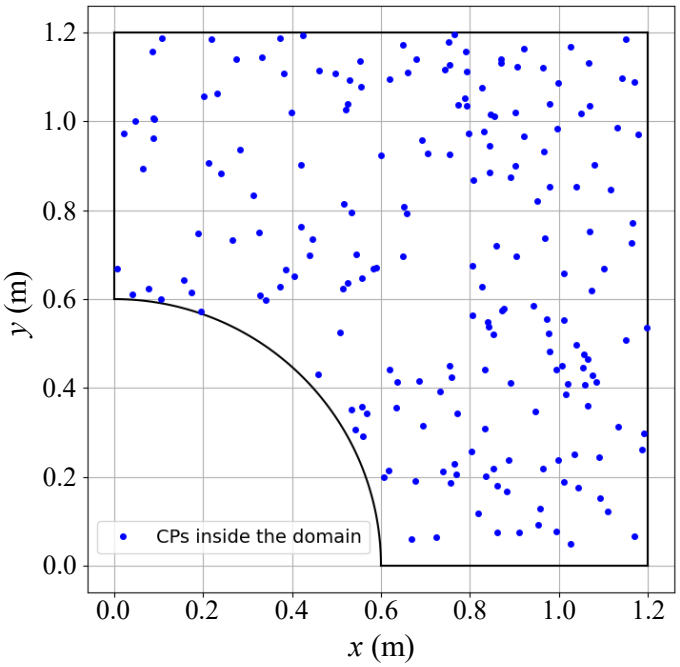


Figure 5-31 Collocation points using sampling method S3

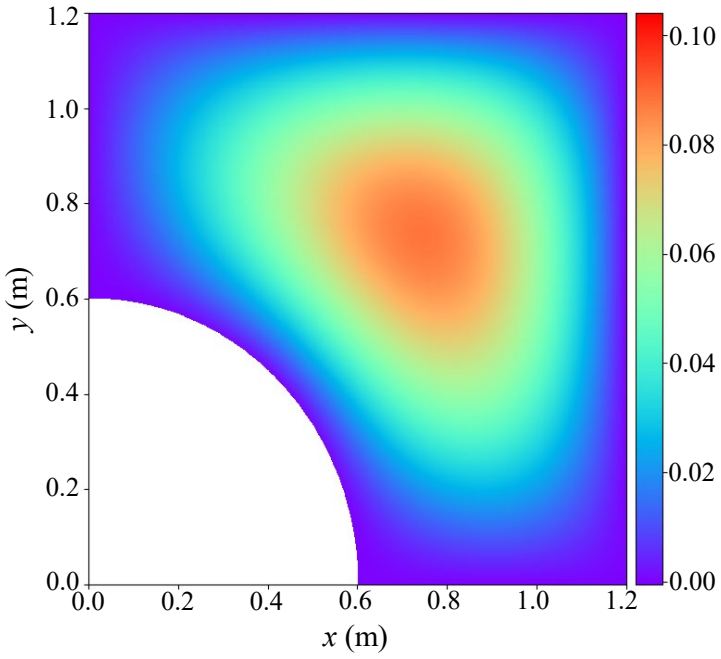


Figure 5-32 The TP-PINN solution for sampling method S3

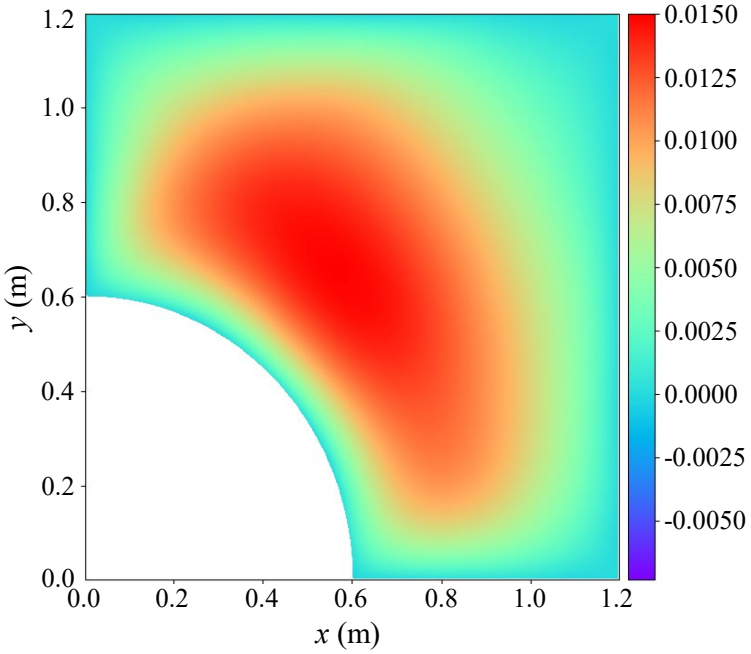


Figure 5-33 The absolute error of TP-PINN for sampling method S3

5.4.2 Types of Pretrained Neural Networks

In the first training phase of TP-PINN, the form of *PreNN* is flexible, even though it should meet certain conditions to ensure that, when applied to the output of the primary NN *PriNN* in the second training phase, the resulting output automatically satisfies the boundary conditions. Depending on the training method used, the *PreNN* can take on different forms. This section discusses different forms of *PreNN* to illustrate their applicability based on the simply-supported beam example.

Figures 5-34 and 5-35 illustrate two forms of *PreNN* that differ from that presented in Section 5.3.1. In Form 1, the function value is constant throughout the entire domain except at the boundaries, where it is zero; in Form 2, the function value is constant over a specific segment of the domain, while the remainder exhibits smooth, continuous variation.

The computation failed using Form 1. Figures 5-36 and 5-37 present the deformation result and error using Form 2. The training time per 100 epoch for Form 2 is 3.49 seconds, which is at the same level as that in Section 5.3.1. However, it takes much longer for this case to converge, indicating the importance of adopting a proper form of *PreNN*. Moreover, it can be observed that when both types of *PreNN* are employed, the results are inaccurate or even fail to compute. The reason is that they generate excessively large or discontinuous gradients on some points, leading to gradient explosion or vanishing problems during computation. Therefore, when choosing the form of *PreNN*, it is important to obey the principle that the entire function maintains relatively moderate gradients.

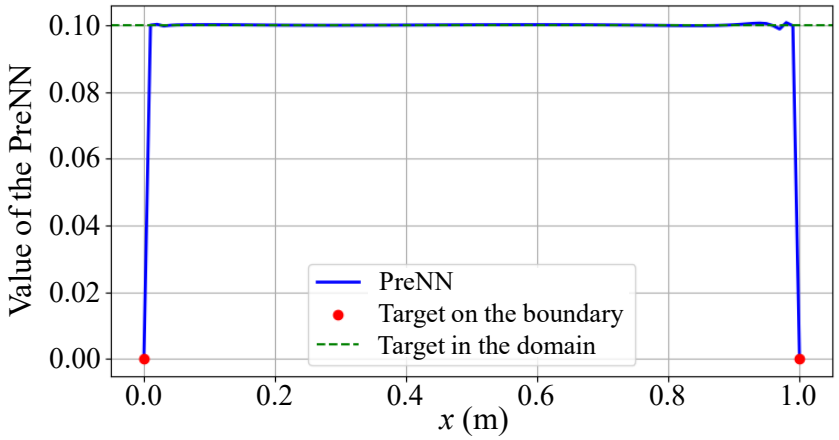


Figure 5-34 The PreNN form 1

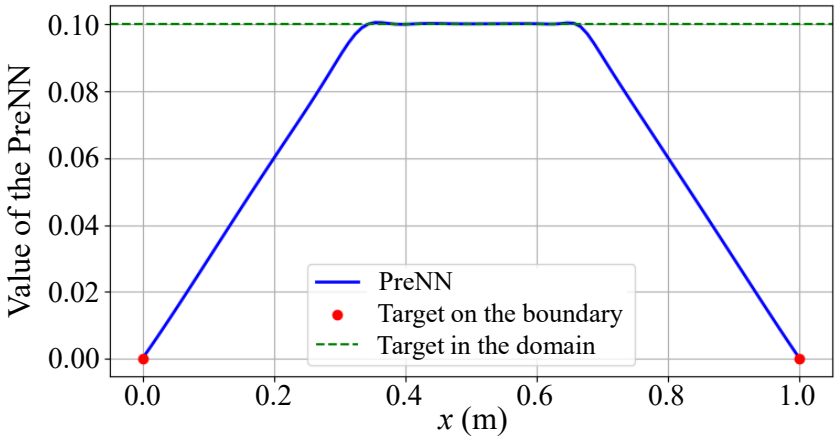


Figure 5-35 The PreNN form 2

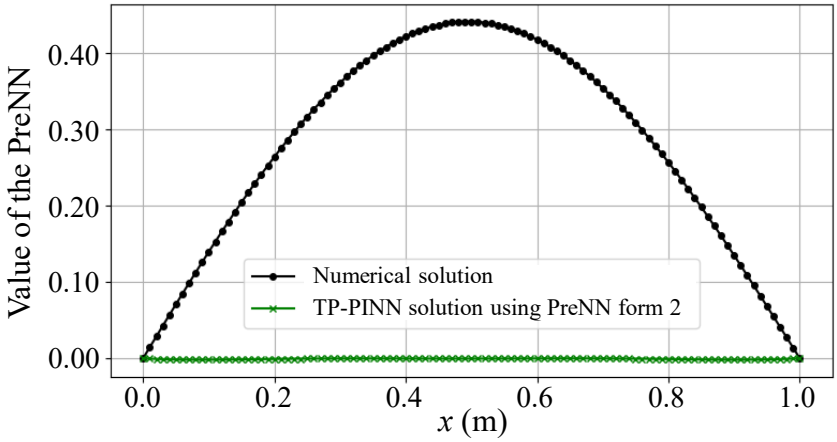


Figure 5-36 The displacements of the simply-supported beam based on PreNN form 2

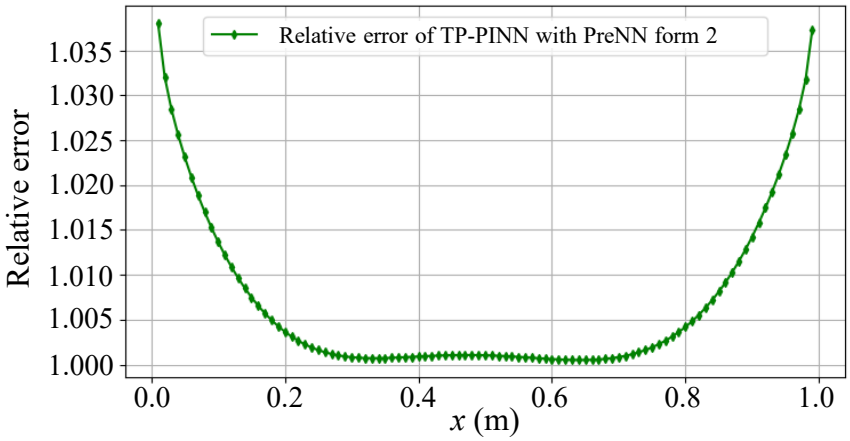


Figure 5-37 The relative error between numerical solution and TP-PINN solution based on PreNN form 2

5.4.3 Limitations

Although the TP-PINN framework proposed in this study, which divides the training process into two distinct phases, can significantly improve the efficiency and accuracy of PINN for solving structural mechanics problems involving high-order differential equations, it is undeniable that this method still has certain limitations.

First, as stated in Sections 5.3.1 and 5.3.2, TP-PINN requires additional training of *PreNN*s, and the introduction of *PreNN*s in the second training phase will bring extra computational cost in backpropagation. When the domain concerned is regular and the modulating functions proposed in Chapter 4 can be derived analytically, adopting modulating functions is preferable to TP-PINN, since it avoids additional training of *PreNN*s.

Second, as demonstrated by the simply-supported Euler-Bernoulli beam and Kirchhoff-Love plate examples, when the boundary conditions are discontinuous in order (for instance, when constraints are imposed only on the function values, first-order derivatives, and third-order derivatives, while no constraint is provided for the second-order derivatives), the *PreNN* in the first phase can only ensure that the output of the *PriNN* in the second-phase automatically satisfies the boundary conditions with continuous orders (i.e., the function values and first-order derivatives). The remaining boundary conditions (i.e., the third-order derivatives) still need to be enforced through the loss function. This limitation arises from the inherent mathematical coupling during differentiation, which restricts the method to automatically satisfy only those boundary conditions of continuous orders, meaning that perfect hard embedding of all boundary conditions cannot be achieved in some cases.

Third, as discussed in the section on collocation point sampling methods, uniformly and densely sampling collocation points across the entire domain may not yield sufficient accuracy in certain regions; therefore, collocation points should be adaptively allocated based on the relative importance of different areas. Moreover, if practical constraints necessitate sparse sampling of collocation points, the solution accuracy will likewise be compromised. Thus, enhancing accuracy under conditions of sparse sampling remains an area for further improvement of the proposed method.

5.5 Summary

A novel framework termed TP-PINN is proposed to achieve efficient computation and accurate solution for solving structural mechanics problems governed by differential equations. Building on the limitations of vanilla PINN framework, TP-PINN divides the training process into two distinct phases. In the first phase, *PreNN*s are employed to decouple the optimization target associated with boundary conditions from the multi-objective loss function. In the second phase, *PreNN*s are applied to transform the output of the *PriNN* to enable a hard-manner treatment of boundary conditions even in irregular domains. Moreover, additional outputs representing lower-order derivatives of the output of *PriNN* are introduced, which effectively reduce the highest order of the governing equations. This two-phase architecture overcomes the challenges of handling high-order derivatives and complicated boundary conditions that often lead to error accumulation and slow convergence in vanilla PINN.

Several numerical examples on Euler-Bernoulli beams and Kirchhoff-Love plates demonstrate the considerable advantages of the TP-PINN framework. For the beam problems,

relative L_2 errors were reduced from the order of 10^{-3} with vanilla PINN to the order of 10^{-5} with TP-PINN, alongside a training time reduction of around 8%. In the plate examples, the framework achieved a significant drop in relative L_2 errors, from over 0.14 to 0.19 down to approximately 0.01 to 0.02, with a significant efficiency improvement of about 24%. These results not only confirm the high accuracy of TP-PINN, particularly near the boundaries, but also its potential to efficiently solve high-order differential equations in structural mechanics.

Nonetheless, some limitations remain. When boundary conditions exhibit discontinuities in order, the *PreNN*s can automatically enforce only part of boundary conditions, leaving the rest to be softly treated via penalty terms. Moreover, the performance of the proposed framework is sensitive to the distribution of collocation points. Future research will focus on incorporating adaptive collocation points sampling strategies, refining *PreNN* configurations to better handle discontinuous boundary conditions, and extending the TP-PINN approach to more complex and multi-physics problems.

CHAPTER 6

Physics-Informed Graph Neural Network for Prestress Design of Tensegrity Structures

6.1 Introduction

In previous chapters, the problems addressed by PINN are all described by differential equations because the useful AD function can effectively solve derivatives in complex functions. However, the application of PINN can be expanded to solve other forms of equations. In this chapter, vanilla PINN and a novel PIGNN frameworks will be applied to the prestress design task for tensegrity structures based on solving equilibrium equations.

Tensegrity structures, a class of self-stressed systems, have gathered significant attention due to their unique properties and potential applications (Skelton and De Oliveira 2009). These structures consist of an assembly of compression elements, i.e. struts, and tension elements, i.e. cables, arranged in a stable pre-stressed configuration. The stability of tensegrity structures stems from the continuous tensile stress in the cables and compressive stress in the struts, resulting in a lightweight yet robust system (Zhang and Ohsaki 2015).

Form-finding is a crucial task for the design of tensegrity structures. Form-finding involves determining the geometric configuration and the internal force distribution within the

elements (struts and cables) that satisfies the equilibrium and stability conditions. In this chapter, the main focus is the determination of the internal force distribution under a given structural topology and geometric configuration, which is called prestress design or force-finding. Numerous force-finding methods have been developed. Tran and Lee (2010) proposed a numerical form-finding method based on the techniques of eigenvalue decomposition (EVD) and singular value decomposition (SVD). Koohestani and Guest (2013) developed a new optimization approach for form-finding using grouping method. Wang (2021b) proposed a general computational form-finding framework through rank minimization of force density matrix. Besides, particle swarm optimization algorithm-based method is proposed by Chen (2020b). However, these approaches rely on structural symmetry and other numerical methods. When dealing with complicated and asymmetric structures, it may be difficult to find a feasible prestress distribution.

In recent years, ML techniques have emerged as powerful tools for solving complex problems in structural engineering. ML algorithms can learn patterns and relationships from data or physical knowledge, enabling them to make accurate predictions or decisions. Efforts on application of ML techniques to force-finding problems in tensegrity structures have been made as it offers the potential to overcome the abovementioned limitations of traditional methods. For example, Lee et al. (2022) applied DNN for form-finding of tensegrity structures, and Zhu et al. (2023) used ANN to aid force-finding of cable domes. These methods are data-driven, which means that a large number of data is required for training the NNs. Their performance cannot be guaranteed because of the insufficiency of training data and the black-box feature of NNs without the guidance of physical knowledge. To make use of physical

knowledge and improve the interpretability of NNs, PINN is considered in this study to tackle this prestress design problem. The performance of vanilla PINN in prestress design will be evaluated in this chapter and the results show that vanilla PINN is able to handle this task. However, although vanilla PINN successfully overcomes the complex preparation of dataset by integrating physical knowledge, it is found that it requires a large number of computational resources and it fails to make full use of known structural information such as the geometry and topology of structures, which remains a huge room for improvement of vanilla PINN's computational efficiency.

To step forward, further modification of vanilla PINN is demanded. GNN is found to be an ideal solution to this problem because GNN can effectively deal with graph-structured data. Inspired by the similarity between the inherent graph-like connectivity of tensegrity structures and the GNN architecture, the paradigm of integrating physical knowledge is introduced to GNN. A novel PIGNN framework is proposed for prestress design of tensegrity structures, which improves the computational efficiency by involving more structural information including the structural geometry and topology compared to vanilla PINN. The graph-like properties of tensegrity structures are naturally embedded through GNN, and it is self-supervised by constructing the loss function based on the equilibrium equation. The proposed framework is able to predict the force density distribution effectively without using grouping and SVD techniques. Moreover, the proposed method can be easily extended to 3D cases, showing a promising potential for dealing with more complicated and asymmetric structures.

The rest of this chapter is arranged as follows: Section 6.2 briefs the method, including the prestress design procedure of tensegrity structures with given topology of elements and

nodal coordinates, the vanilla PINN framework for prestress design, and the detailed introduction to PIGNN. Several numerical cases including regular and irregular forms of two-dimensional and three-dimensional tensegrity structures are presented in Section 6.3, the results of both the vanilla PINN and PIGNN frameworks are presented and compared. Lastly, the conclusion and future research directions are summarized in Section 6.4.

6.2 Methods

In this section, the basic knowledge of tensegrity structures is first introduced as the fundamental of this study, including the concept of force density, equilibrium equation, and the prestress design procedure. Then, physical knowledge is incorporated into PINN framework. After introducing how to achieve prestress design by the vanilla PINN, the proposed PIGNN framework is then detailed for the same task.

6.2.1 Prestress Design of Tensegrity Structures Based on Force Density

First of all, the following assumptions are considered in this chapter for prestress design of tensegrity structures:

(1) Geometry (i.e. nodal coordinates) and topology (i.e. connectivity of cables and struts) of tensegrity structures are given.

(2) Cables and struts are connected by pin joints.

(3) Self-weight of tensegrity structures, external loads, and global and local buckling are not considered.

Since the self-weight and external loads are not considered, tensegrity structures can be self-equilibrated without any support nodes, which are called free-standing rigid structures (Zhang and Ohsaki 2015).

A two-dimensional tensegrity structure with four nodes and six elements, shown with the numbering of elements and coordinates in Figure 6-1, is used as an example to illustrate the process of prestress design based on force density.

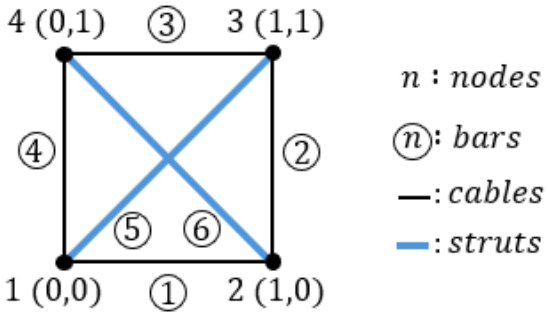


Figure 6-1 An illustrative two-dimensional tensegrity structure with four nodes and six elements

The force density vector $\boldsymbol{q}$ (Pellegrino and Calladine 1986) consists of force density q_k of element *k*, which is defined as the ratio of internal force f_k to length l_k:

$$q_k = \frac{f_k}{l_k} \tag{6-1}$$

The connectivity matrix $\mathbf{C} \in \mathbb{R}^{b \times n}$ is defined to describe the topology of a tensegrity structure with *b* elements and *n* nodes, i.e. connectivity of nodes and elements. In Figure 6-1, the nodes and elements are numbered respectively. The *i*-th and *j*-th entries $c_{k,i}$ and $c_{k,j}$ in *k*-th row in connectivity matrix $\boldsymbol{C}$ are set to -1 and 1 if the element *k* connects nodes *i* and *j* (*i*<*j*), as follows:

$$c_{k,m} = \begin{cases} 1 & \text{for } m = i \\ -1 & \text{for } m = j \\ 0 & \text{otherwise} \end{cases} \tag{6-2}$$

Then, for a two-dimensional tensegrity structure (*d*=2), the equilibrium matrix $\mathbf{A} \in \mathbb{R}^{dn \times b}$ can be denoted by the connectivity matrix $\boldsymbol{C}$ and nodal coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ as (Pellegrino and Calladine 1986):

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}^{\mathrm{T}} diag(\mathbf{C}\mathbf{x}) \\ \mathbf{C}^{\mathrm{T}} diag(\mathbf{C}\mathbf{y}) \end{pmatrix} \tag{6-3}$$

For example, the connectivity matrix $\mathbf{C} \in \mathbb{R}^{b\times n} = \mathbb{R}^{6\times 4}$ and the equilibrium matrix $\mathbf{A} \in \mathbb{R}^{dn\times b} = \mathbb{R}^{8\times 6}$ of the illustrative structure are shown respectively as follows:

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{6-4}$$

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{6-5}$$

For a three-dimensional tensegrity structure (d=3), the connectivity matrix $\boldsymbol{C}$ also belongs to $\mathbb{R}^{b\times n}$, while the equilibrium matrix $\mathbf{A} \in \mathbb{R}^{dn\times b} = \mathbb{R}^{3n\times b}$ can be formulated as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}^{\mathrm{T}} diag(\mathbf{C}\mathbf{x}) \\ \mathbf{C}^{\mathrm{T}} diag(\mathbf{C}\mathbf{y}) \\ \mathbf{C}^{\mathrm{T}} diag(\mathbf{C}\mathbf{z}) \end{pmatrix} \tag{6-6}$$

The equilibrium equation can be derived by satisfying the equilibrium condition for zero unbalanced forces on each node as follows:

$$\mathbf{A}\mathbf{q} = \mathbf{0} \tag{6-7}$$

The aim of prestress design is to find a feasible force density vector by solving the above equilibrium equation. A tensegrity structure may have one or more independent self-stress modes. The number of independent self-stress modes is denoted by s, which is related to the rank deficiency of the equilibrium matrix A and can be computed by:

$$s = b - \mathrm{rank}(\mathbf{A}) \tag{6-8}$$

The entire space of self-stress modes is given by the null space of the equilibrium matrix $\mathbf{A}$. The solution to Equation (6-7) can be a linear combination of these self-stress modes. However, not all solutions are feasible. To verify the feasibility of the obtained prestress, the

unilateral properties of elements and the stability of the entire structure should be checked. Specifically, for cables that can only carry tension, their force densities must satisfy:

$$q_i \geq 0 \qquad (6-9)$$

while struts that are under compression, their force densities must satisfy:

$$q_i \leq 0 \qquad (6-10)$$

These conditions ensure that cables are not subjected to compression and struts are not in tension. Verification involves checking that every component of the obtained force density vector $\mathbf{q}$ meets its corresponding inequality.

In addition to verifying unilateral behavior, the stability of the entire structure must be confirmed by analyzing the minimum rank deficiency $d+1$ and the force density matrix $\mathbf{F}$:

$$\mathbf{F} = \mathbf{C}^{\mathrm{T}} diag(\mathbf{q})\mathbf{C} \qquad (6-11)$$

The minimum rank deficiency $d+1$ should be three in two-dimensional case and four in three-dimensional case. If the force density matrix $\mathbf{F}$ is positive semi-definite, the structure is super stable (Connelly 2002; Zhang and Ohsaki 2007) without considering other properties. In the case that $\mathbf{F}$ is negative semi-definite, the structure can be considered as stable if the tangent stiffness matrix is positive definite by investigating its eigenvalues (Ohsaki and Zhang 2006; Guest 2006). By satisfying the above requirements, the prestress design procedure is finally completed.

Many prestress design methods were developed based on the member grouping technique, SVD, genetic algorithm, and rank minimization. However, it is still hard to find feasible force density vectors when the structure is complicated or asymmetric. To tackle this problem, ML methods are explored as possible solutions because of their promising potential in dealing with complex structural forms. In the following subsections, the detailed implementation of PINN frameworks is introduced.

6.2.2 Vanilla PINN for Prestress Design

First of all, the reasons of applying PINN frameworks for prestress design of tensegrity structures can be summarized as follows: (1) PINN is not only able to solve differential equations but also flexible to solve non-differential equations; (2) PINN can transform prestress design into a NN parameter optimization problem, which can be handled by mature optimization algorithms; (3) Under the PINN framework, asymmetrical geometries can be easily handled and complex matrix operations can be avoided.

The framework of the vanilla PINN for prestress design of tensegrity structures is illustrated in Figure 6-2, which contains the representation of the inputs, the type of NN, the outputs, and the formulation of the loss function.

There are different choices for the inputs. For example, the coordinates of the nodes of each cable or strut can represent the element. To reduce the number of inputs, the middle point of each element can be used. These two choices contain structural information, i.e. the nodal coordinates, which are physical knowledge of the structure. However, non-structural information can also be treated as inputs, e.g. the numbering of each element, as shown in the Input box in Figure 6-2, which means that the only requirement for the choice of inputs is to represent the elements, and the structural information is actually not necessary in this vanilla PINN framework. In this chapter, the numbering of each element is adopted as the input, and therefore, the dimension of the input is one.

Similar to previous chapters, the NN employed in this framework is FCFNN, which is detailed in Chapter 3. The parameters of the FCFNN are the optimization objects.

The output is the force density of each corresponding structural element predicted by the FCFNN. Therefore, the dimension of the output is the same as the dimension of the input. Then, the force density vector can be obtained.

Next, to formulate the loss function, the predicted force density vector is substituted into Equation (6-7) to calculate the unbalanced force of each node. According to the force equilibrium condition, the unbalanced forces are squared and summed as the loss function:

$$L = \frac{1}{nd}\sum_{i=1}^{nd}\varepsilon_i^2(\boldsymbol{q}(\theta)) \qquad (6-12)$$

where nd is the number of elements, $\boldsymbol{q}$ is the force density vector, θ is the parameters of the FCFNN, and ε is the unbalanced forces.

By minimizing the loss function, a design of the prestress distribution can be obtained without using any grouping and SVD techniques and complicated matrix operations.

Finally, the unilateral property and stability condition will be used to check the feasibility of the prestress design.

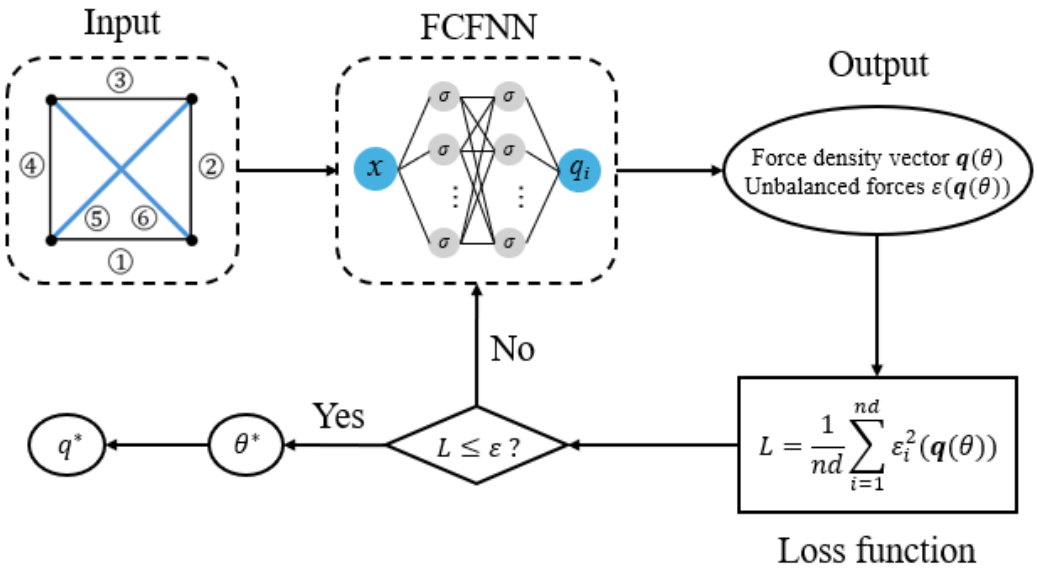


Figure 6-2 The vanilla PINN framework for prestress design of tensegrity structures

6.2.3 Establishment of Physics-Informed Graph Neural Network (PIGNN)

To improve the efficiency of vanilla PINN in prestress design, a novel framework PIGNN is proposed, in which GNN is employed to replace the FCFNN in vanilla PINN. The idea of introducing GNN into PIML family is inspired by the following reasons: (1) The graph-like connectivity of tensegrity structures can be naturally represented by GNN, in other words, GNN is especially suitable for dealing with cable-strut systems such as tensegrity structures; (2) By utilizing GNN, additional structural information including geometries and topologies can be incorporated into the GNN architecture, bringing promising potential for learning structural behaviors.

Before looking into the proposed PIGNN framework, it is better to provide a brief introduction of GNN. GNN is a class of ML models designed to operate on graph-structured data. A GNN is a transformation on attributes of a graph, i.e. nodes, edges, and global information, which preserves the graph topology.

In a typical GNN framework, as shown in Figure 6-3, a “graph-in, graph-out” architecture is adopted. This model takes a graph as input, with information embedded in its nodes, edges, and global-level content, progressively transforms those attributes through several layers of graph convolutional network (GCN) and activation function, ReLU for example, and finally outputs a graph without changing its topology. More details about GCN and activation functions can be found in other references (Zhou et al. 2020).

A GNN can address different types of tasks including node-level, edge-level, and graph-level tasks. They are concerned with predicting the properties of nodes, edges, and the entire

graph, respectively. It is obvious that the task type is edge-level in the concerned prestress design problem because it is trying to find the feasible force density vector of elements. However, in the present case, the only information that is already known is the node-level feature, i.e. the nodal coordinates. Therefore, the edge prediction problem should be transformed into node prediction by pooling information, which means the node-level information will be first collected and aggregated, then given to edges for prediction. The pooling operation is illustrated in Figure 6-4.

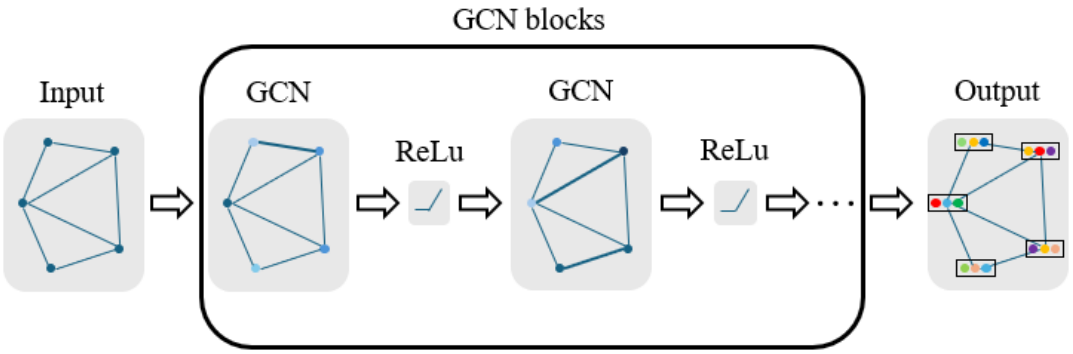


Figure 6-3 A GNN framework

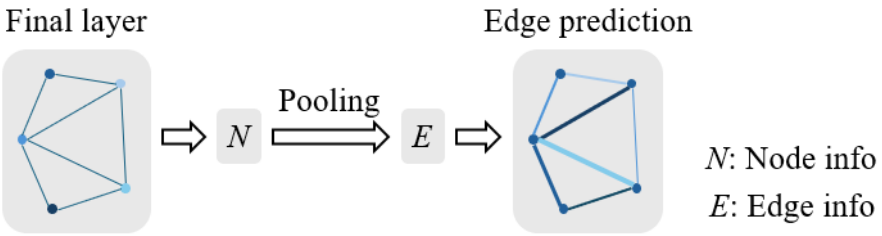


Figure 6-4 The pooling operation

Based on the general GNN framework and the pooling operation, the PIGNN is proposed for prestress design of tensegrity structures. The detailed description of PIGNN is as follows.

The input of PIGNN is a graph representation of the tensegrity structure, where structural information is embedded, including coordinates of nodes and connectivity of elements (i.e. structural topology). The graph is undirected, and the topology is described by an adjacent

matrix, which is slightly different from the connectivity matrix in the form. The unilateral properties of elements are defined, under which cables can only carry tension and struts are only under compression. Then, the input graph is processed through forward propagation with several GNN layers and activation functions. In this chapter, GCN and ReLU are adopted as the hidden layer and the activation function. The final layer is the transformed graph with sufficient information on nodes. GCN's outputs are set to be positive when initially generated. The pooling operation is applied to node information to generate edge information for prediction. For example, each node yields n_e values, where n_e is the number of edges. The sum of the *i*-th value of each node represents the predicted information of the *i*-th edge. Then, these outputs on edges are multiplied by 1 for cables and -1 for struts. In this case, the prediction of each edge means the force density q, which results in the force density vector $\boldsymbol{q}$ that satisfies unilateral properties. Subsequently, the unbalanced force on each node can be computed according to Equation (6-7) and the sum of these unbalanced forces are treated as the loss function to be minimized as follows:

$$L = \frac{1}{nd}\sum_{i=1}^{nd} \varepsilon_i^2\big(\boldsymbol{q}(\theta)\big) = \frac{1}{nd}\|\mathbf{Aq}\|_2^2 \qquad (6-13)$$

where nd is the number of unbalanced forces, ε_i is the entry of the vector $\mathbf{Aq}$, and θ is the trainable parameters of GCN layers. After the calculation of the loss function, it can be minimized by an optimizer, e.g. Adam in this case, through backpropagation during the training process. In the numerical cases in the following section, the GNN is trained for 1000 epochs with a learning rate of 0.01. In the backpropagation process, the gradients of the loss function with respect to inputs will be computed automatically based on the chain rule by AD function

and the trainable parameters θ of GNN will be updated accordingly. The candidate feasible force-density vector $\boldsymbol{q}$ will be obtained with the loss function approaching zero. After the prestress design procedure, the candidate $\boldsymbol{q}$ will be checked based on unilateral properties of elements, equilibrium condition, and stability condition as described in Section 6.2.1.

The overall framework of the proposed PIGNN framework is illustrated in Figure 6-5. It can be noticed that there is no training data involved in the whole process, which means this framework is physics-informed and self-supervised. The merits of this method are manifolds. It is easier to achieve compared to other data-driven ML methods by waiving the requirement of large amount of training data and the complicated data preparation process; it is explainable due to the involvement of structural geometry, topology, and physical laws; it is feasible to realize prestress design through gradient descent algorithms without any grouping or SVD techniques; finally, it is more efficient compared to vanilla PINN framework by integrating additional structural information.

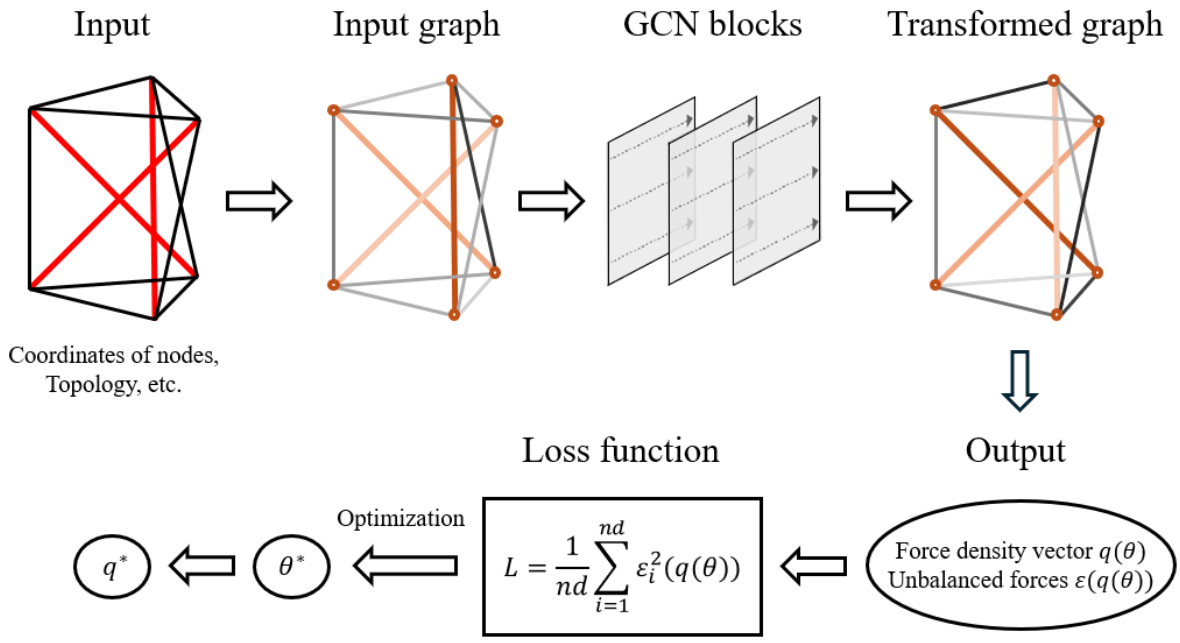


Figure 6-5 The PIGNN framework for prestress design of tensegrity structures

6.3 Numerical Case Studies

In this section, comprehensive numerical case studies, including two-dimensional, three-dimensional, regular geometry, and irregular geometry cases, solved by vanilla PINN and PIGNN frameworks will be presented and compared to demonstrate the effectiveness of the proposed framework. All codes are written based on Python 3.7.13, the DL library PyTorch 1.11.0, and the GNN library PyG 2.0.4.

6.3.1 Two-Dimensional Cases

Five-unit Snelson's X-shape tensegrity arches with regular and irregular geometries are adopted as two-dimensional cases. They have the same topology but different geometries.

2D regular geometry case

The five-unit Snelson's X-shape tensegrity arch with regular geometry is shown in Figure 6-6. It contains 12 nodes and 26 elements (16 cables and 10 struts), and they are numbered respectively. The coordinates of nodes are also illustrated in Figure 6-6. This structure is statically indeterminate and kinematically determinate.

The feasible force density of each element of this structure predicted by vanilla PINN and PIGNN is listed in Table 6-1. The results are checked to satisfy the unilateral properties, equilibrium condition, and stability condition. Therefore, the results of both methods can be considered as feasible force density distributions. As for the computational performance, vanilla PINN takes 300 epochs to reach a loss value of 2×10^{-5}, while PIGNN only requires 90 epochs to reach a lower loss value of 1×10^{-5}. It is obvious that computational efficiency is highly improved by the proposed PIGNN framework compared to vanilla PINN because

processing structured data is the core competency of GNN.

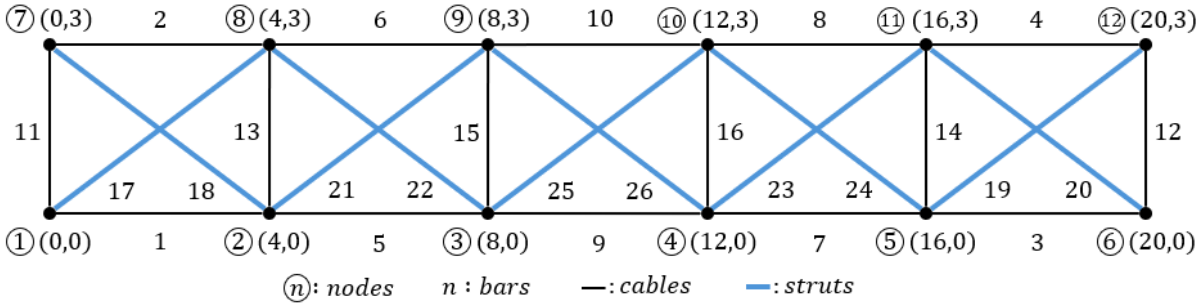


Figure 6-6 A five-unit Snelson's X-shape tensegrity arch with regular geometry

Table 6-1 The feasible force densities of the regular tensegrity arch predicted by vanilla PINN and PIGNN

Force density	PINN	PIGNN	Force density	PINN	PIGNN
q_1	0.4732	0.5191	q_{14}	0.9488	1.0000
q_2	0.4732	0.5191	q_{15}	0.7687	0.6848
q_3	0.3691	0.4228	q_{16}	1.0000	0.7995
q_4	0.3691	0.4228	q_{17}	-0.4732	-0.5191
q_5	0.3483	0.4624	q_{18}	-0.4732	-0.5191
q_6	0.3483	0.4624	q_{19}	-0.3691	-0.4228
q_7	0.5796	0.5772	q_{20}	-0.3691	-0.4228
q_8	0.5796	0.5772	q_{21}	-0.3483	-0.4624
q_9	0.4204	0.2223	q_{22}	-0.3483	-0.4624
q_{10}	0.4204	0.2223	q_{23}	-0.5796	-0.5772
q_{11}	0.4732	0.5191	q_{24}	-0.5796	-0.5772
q_{12}	0.3691	0.4228	q_{25}	-0.4204	-0.2223
q_{13}	0.8215	0.9815	q_{26}	-0.4204	-0.2223

2D irregular geometry case

The five-unit Snelson's X-shape tensegrity arch with irregular geometry, which is a variant of the previous case, is shown in Figure 6-7. It has the same components and topology as the regular case, containing 12 nodes and 26 elements (16 cables and 10 struts), and they are numbered respectively. The coordinates of nodes are listed in Table 6-2. This structure is also statically indeterminate and kinematically determinate.

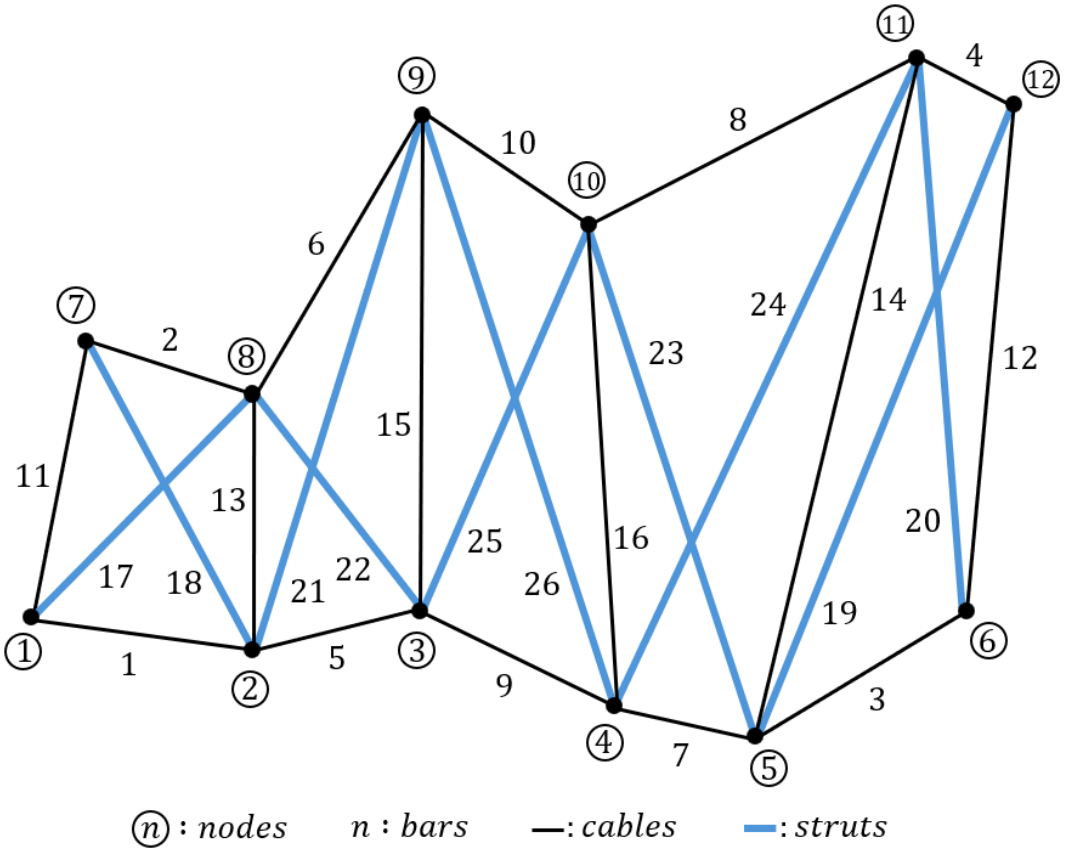


Figure 6-7 A five-unit Snelson's X-shape tensegrity arch with irregular geometry

Table 6-2 Nodal coordinates of the 2D tensegrity arch with irregular geometry

Node No.	(x, y)	Node No.	(x, y)
1	(0.0, 0.0)	7	(0.3, 1.7)
2	(1.4, -0.2)	8	(1.4, 1.4)
3	(2.5, 0.0)	9	(2.5, 3.2)
4	(3.7, -0.5)	10	(3.6, 2.5)
5	(4.6, -0.7)	11	(5.6, 3.6)
6	(6.0, 0.0)	12	(6.3, 3.3)

The feasible force density of each element of this structure predicted by vanilla PINN and PIGNN is shown in Table 6-3. The results of both methods are feasible force density distributions because they fulfill the unilateral property requirement, equilibrium condition, and stability condition. Compared to the previous regular geometry case, vanilla PINN takes 800 epochs to reach a loss value of 8×10^{-5}, which means a more complicated geometry will influence the performance of vanilla PINN. However, it only takes 80 epochs for the proposed PIGNN framework to reach a lower loss value of 1×10^{-5}, which means that the performance of PIGNN framework is not affected by changing the structural geometry. It shows the powerful ability of the proposed framework in dealing with more complex problems.

Table 6-3 The feasible force densities of the irregular tensegrity arch predicted by vanilla PINN and PIGNN

Force density	PINN	PIGNN	Force density	PINN	PIGNN
q_1	0.1437	0.3691	q_{14}	0.5502	0.5350
q_2	0.2277	0.5848	q_{15}	0.4238	0.4417
q_3	0.2141	0.2446	q_{16}	0.7575	0.7666
q_4	0.6324	0.7224	q_{17}	-0.1789	-0.4595
q_5	0.4004	0.2442	q_{18}	-0.1829	-0.4697
q_6	0.4004	0.2442	q_{19}	-0.3442	-0.3931
q_7	0.7414	0.5852	q_{20}	-0.3935	-0.4494
q_8	0.1765	0.1393	q_{21}	-0.4004	-0.2442
q_9	0.2217	0.3169	q_{22}	-0.4004	-0.2442
q_{10}	0.2659	0.3800	q_{23}	-0.3252	-0.2567
q_{11}	0.1643	0.4218	q_{24}	-0.4024	-0.3176
q_{12}	0.4747	0.5422	q_{25}	-0.2419	-0.3457
q_{13}	1.0000	1.0000	q_{26}	-0.2437	-0.3483

6.3.2 Three-Dimensional Cases

The performance of both vanilla PINN and PIGNN frameworks is demonstrated in two-dimensional cases. The proposed PIGNN outperforms vanilla PINN in terms of computational efficiency because it is not notably influenced by irregular geometry. Moreover, the proposed framework can be easily extended to solve more complex prestress design problems of three-

dimensional tensegrity structures, which involve more structural elements (i.e. more design variables) and more complex structural topology (i.e. more complicated matrix operations).

In this subsection, prestress design of three-dimensional tensegrity structures with regular and irregular geometries is solved by the proposed PIGNN framework to demonstrate its performance in 3D cases.

3D regular geometry case

The 3D tensegrity structure with regular geometry is composed of repeated 3D units. The 3D unit contains 8 nodes, 12 cables, and 4 struts. The perspective and top views of the 3D unit are shown in Figures 6-8 and 6-9, respectively. The length of the cable of the bottom square is 100 cm, and the height of the unit is 50 cm. The concerned regular tensegrity structure consists of 9 units. Therefore, it contains 40 nodes, 96 cables, and 36 struts. The perspective and top views of this structure are shown in Figures 6-10 and 6-11, respectively.

Although geometry and topology of the 3D tensegrity structure are much more complicated compared to 2D cases, e.g. the connectivity matrix $\mathbf{C} \in \mathbb{R}^{132\times40}$ is a very large sparse matrix, and 132 unknown force densities should be determined simultaneously, PIGNN can easily handle this complex case. The prestress distribution obtained by PIGNN is shown in Figure 6-12, in which blue lines stand for struts under compression and red lines represent cables in tension. By checking the unilateral property requirement, equilibrium condition, and stability condition, the result is a feasible solution. As for the computational efficiency, PIGNN takes 180 epochs to reach a loss value of 1×10^{-5}, more epochs are required than those of the 2D case. However, it still shows the ability of PIGNN framework in prestress design that involves more than 100 design variables and adaptability when applying it to complex 3D cases.

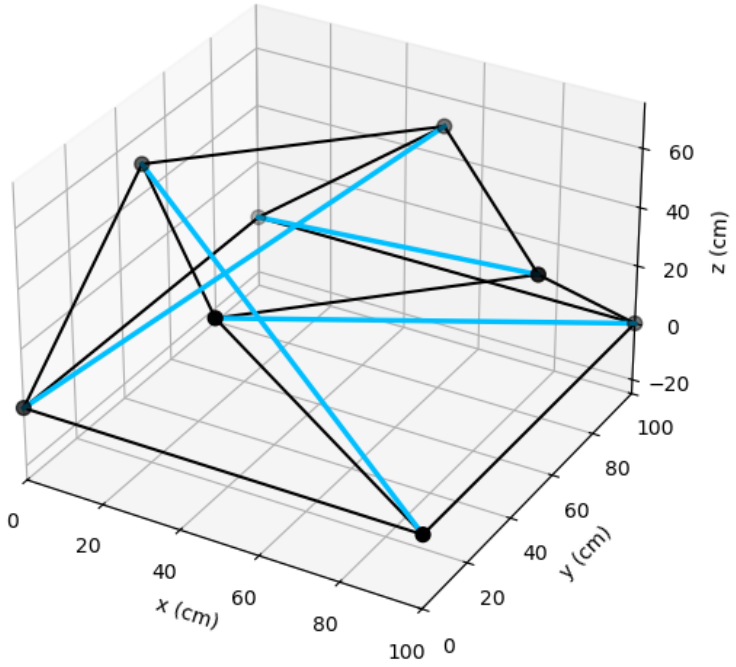


Figure 6-8 The perspective view of the 3D unit

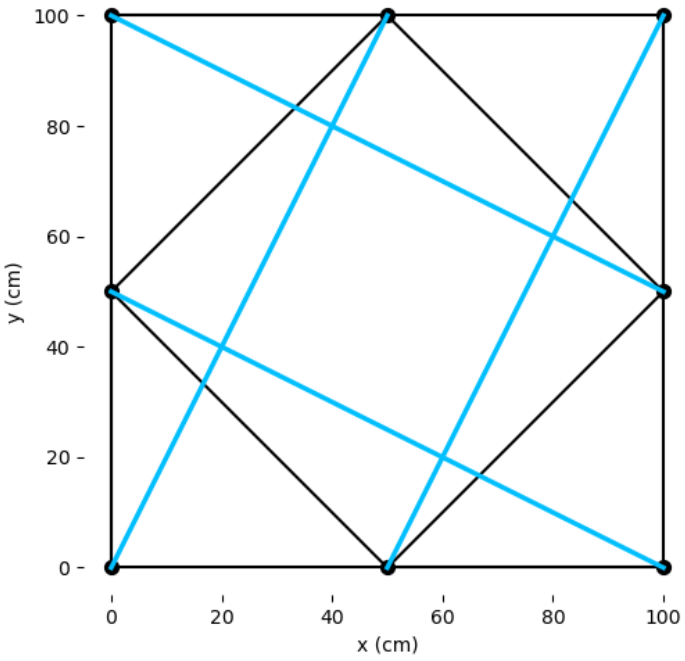


Figure 6-9 The top view of the 3D unit

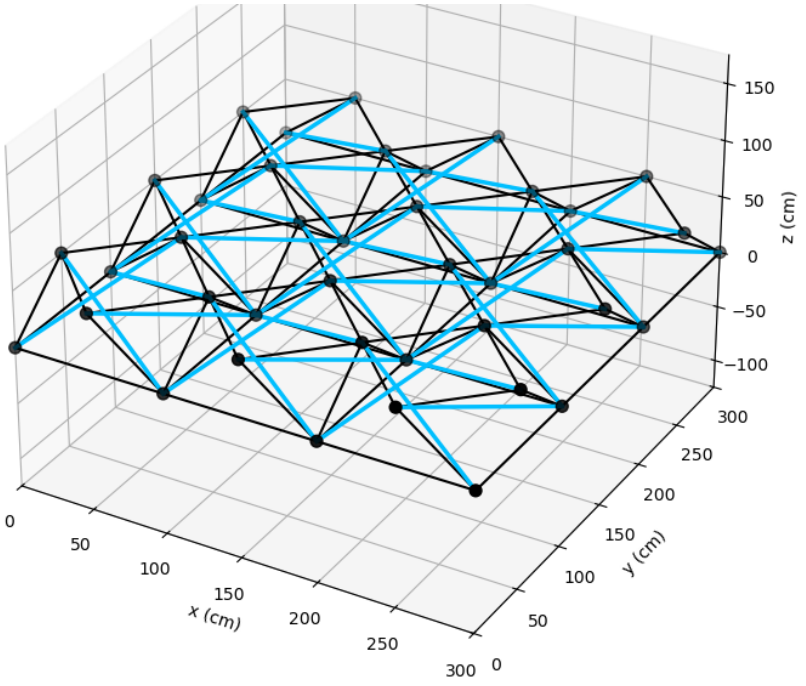


Figure 6-10 The perspective view of the 3D tensegrity structure with regular geometry

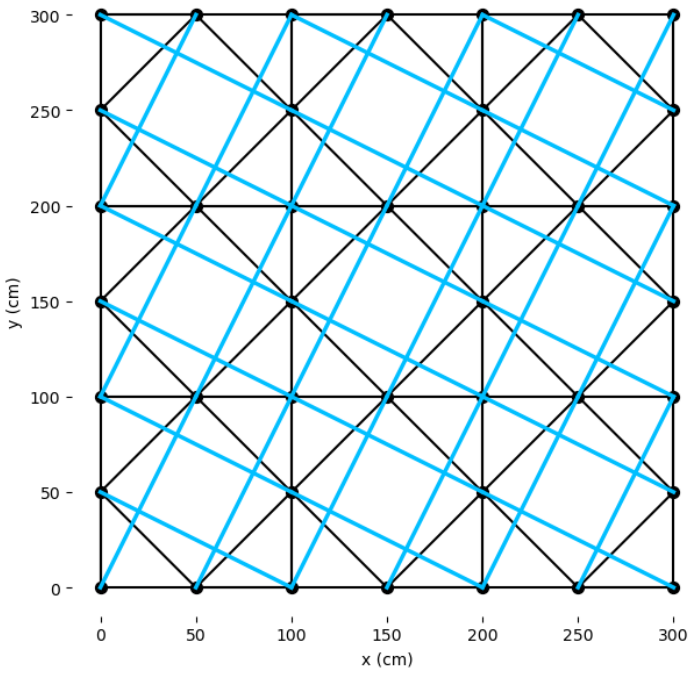


Figure 6-11 The top view of the 3D tensegrity structure with regular geometry

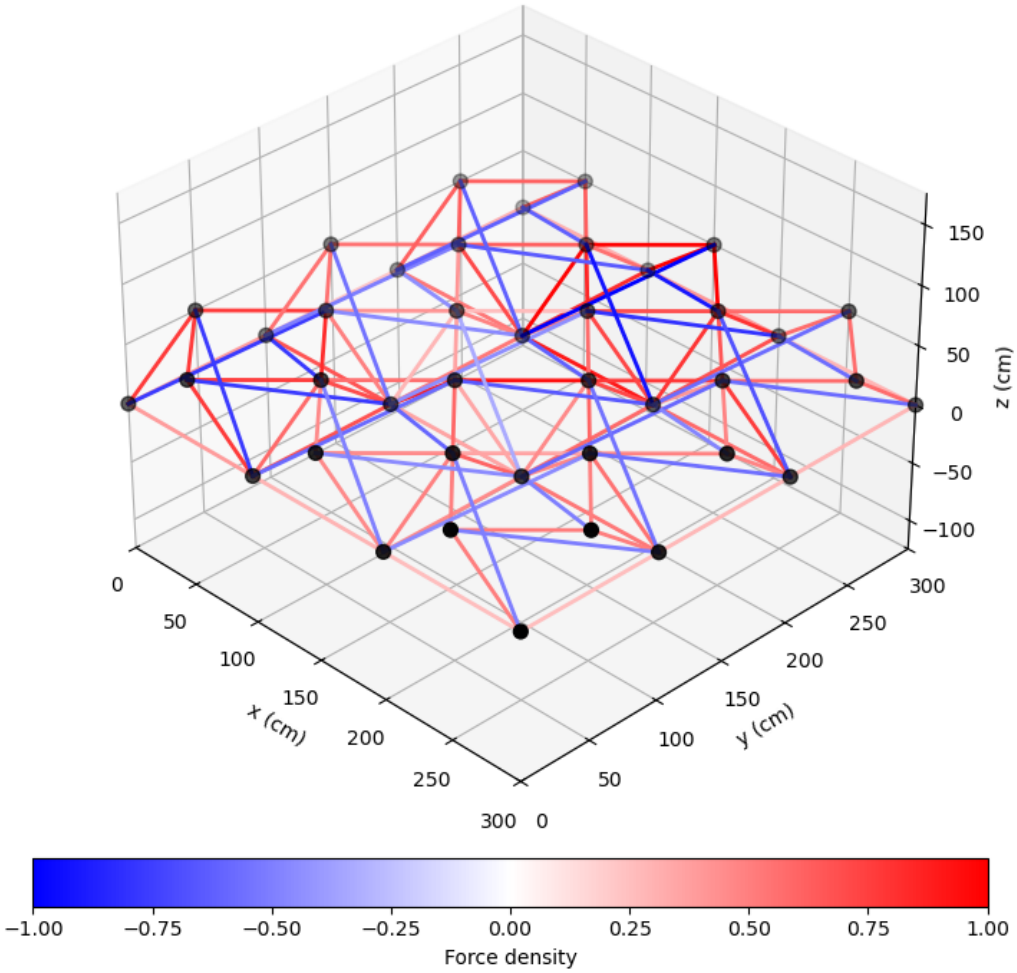


Figure 6-12 The feasible prestress distribution of the 3D tensegrity structure with regular geometry

3D irregular geometry case

The 3D tensegrity structure with irregular geometry is created based on the regular one (Figures 6-10 and 6-11) by modifying the coordinates of four nodes, which are labelled in Figure 6-14. The coordinates of the four changed nodes are (-30, 50, 60), (250, -40, 50), (-30, 250, 40), and (250, 360, 50), respectively. The unit is cm. The coordinates of other nodes remain unchanged, and the structure contains 40 nodes, 96 cables, and 36 struts. The perspective and top views of this irregular structure are shown in Figures 6-13 and 6-14, respectively.

PIGNN is used to design the prestress distribution, and the results are shown in Figure 6-

15, in which blue lines represent struts under compression and red lines depict cables in tension. This prestress distribution is feasible since it fulfills the unilateral property requirement, equilibrium condition, and stability condition. It costs 200 epochs for PIGNN to reach a loss value of 1×10^{-5}, which means that the computational efficiency is comparable to that of the regular geometry case. It leads to the same conclusion as in the 2D case that the irregular geometry does not affect PIGNN's performance.

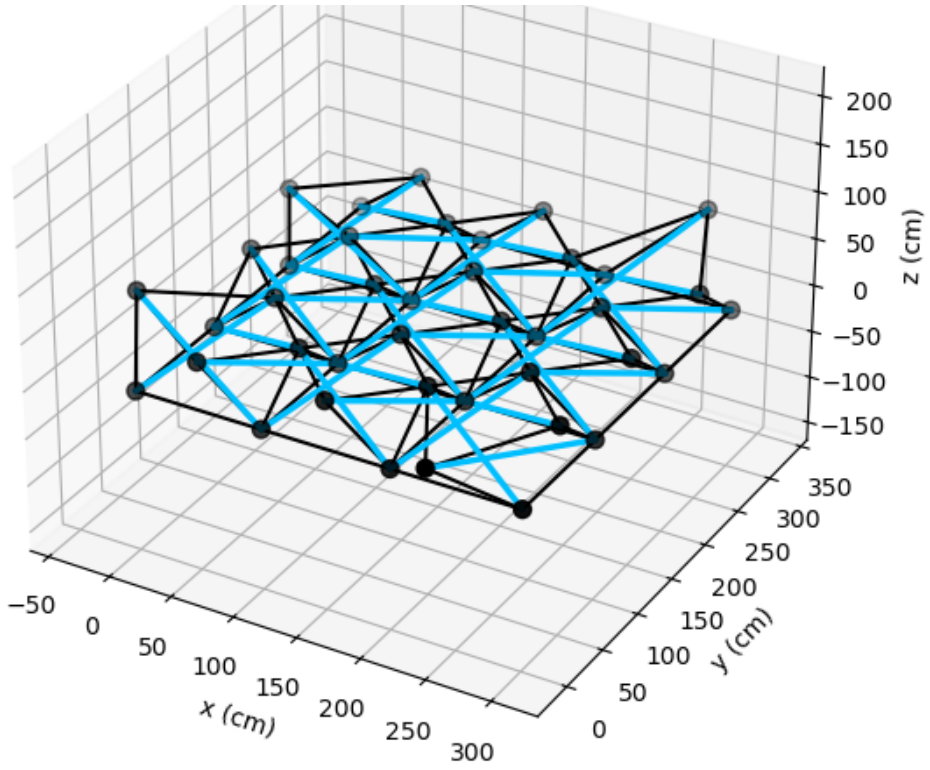


Figure 6-13 The perspective view of the 3D tensegrity structure with irregular geometry

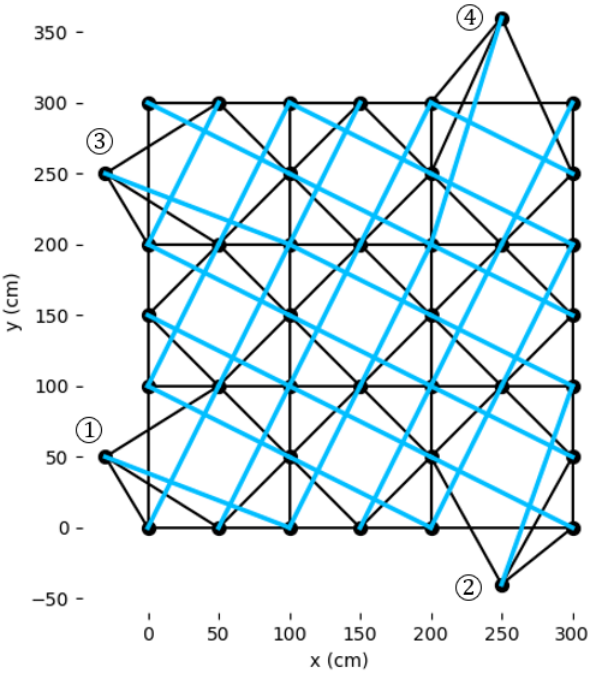


Figure 6-14 The top view of the 3D tensegrity structure with irregular geometry

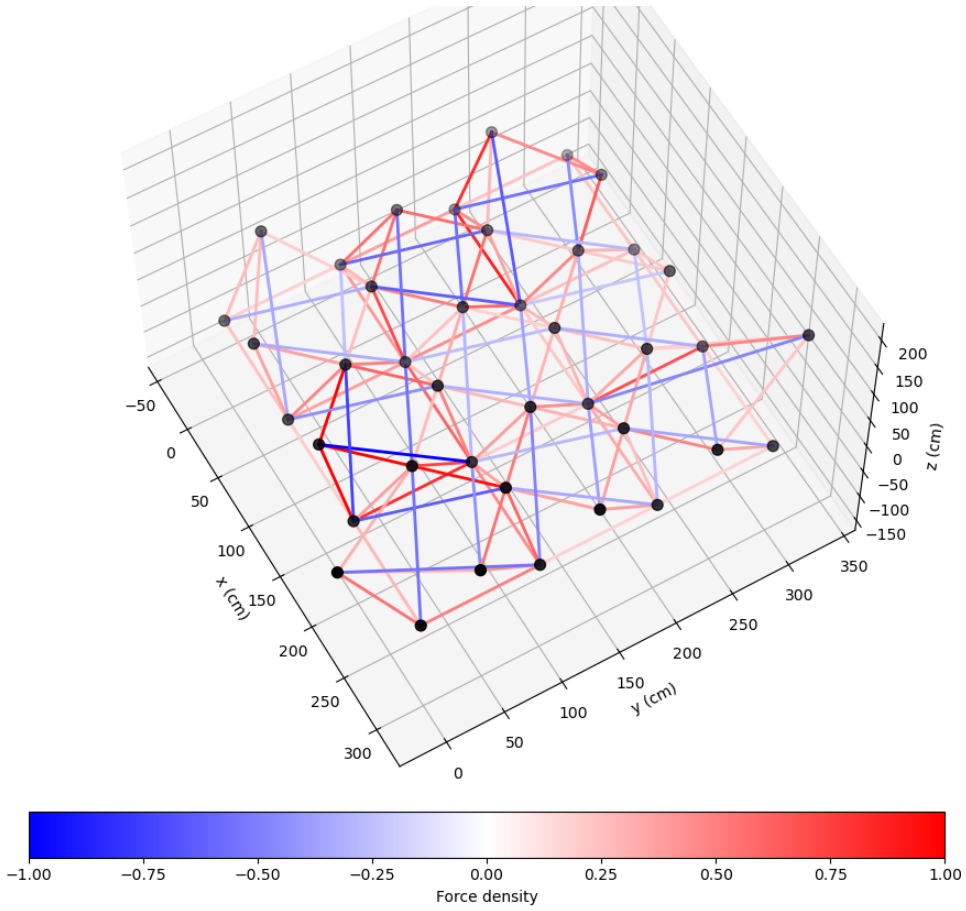


Figure 6-15 The feasible prestress distribution of the 3D tensegrity structure with irregular geometry

6.4 Summary

In this chapter, a novel PIGNN framework is proposed to search for feasible prestress distribution designs of tensegrity structures. This framework is inspired by the inherent graph-like topology of tensegrity structures, and its advantages are manifold. First, it is self-supervised compared to other data-driven methods because the loss function is established based on the equilibrium condition, which means no labelled training data is required; Second, more physical information of tensegrity structures is embedded compared to vanilla PINN because the topology of the structure is naturally incorporated in GNN; Third, it avoids complicated matrix operations involved in numerical methods such as EVD and SVD. Fourth, complex 2D or 3D geometries can be easily tackled without using grouping techniques.

The performance of the proposed PIGNN framework is presented via several 2D and 3D numerical cases with regular and irregular geometries. It can achieve smaller loss values while less training epochs are required compared with vanilla PINN method. It shows great potential in solving the force-finding problem with complicated structural topologies and geometries, since the geometric configuration has little influence on the efficiency of PIGNN framework.

This novel PIGNN framework is proposed and applied to the prestress design of tensegrity structures for the first time, exhibiting feasibility and strong ability in this field. The potential of PIGNN in addressing more complex problems of tensegrity structures such as topology optimization needs further exploration in the future.

CHAPTER 7

Conclusions and Recommendations

7.1 Conclusions

This thesis aims to advance PIML for applications in structural engineering by targeting four critical challenges: (1) the computational burden and error accumulation arising from high-order differential equations; (2) the difficulty of strictly satisfying boundary conditions within the vanilla PINN framework; (3) the computational difficulty and inaccuracy caused by the multi-objective optimization process; (4) the limited capacity of the vanilla PINN to exploit the intrinsic geometric and topological properties of certain structural systems. To address these challenges, four methodological innovations are proposed in the thesis and validated through comprehensive numerical studies and experiments:

1. Auxiliary outputs for reducing the order of differential equations. Vanilla PINN relies on AD to compute derivatives of the network's outputs. When the differential equation involves high-order derivatives, high-order AD may lead to severe gradient exploding or gradient vanishing issues, slow convergence, and reduced accuracy. In Chapter 3, the cause of training failure for such high-order differential equations is analyzed theoretically. To address this issue, auxiliary outputs are introduced to approximate

lower-order derivatives of primary outputs. By reformulating the high-order differential equation (e.g. a fourth-order differential equation) into a coupled system of lower-order equations (e.g. two second-order differential equations), the original differential equation is reduced to lower order. Numerical tests on three cases, including the third-order KdV equation, the fourth-order Euler-Bernoulli equation, and a sixth-order equation, demonstrate that the proposed method can effectively achieve higher solution accuracy compared to the vanilla PINN.

2. Hard enforcement of boundary conditions via modulating functions. The vanilla PINN traditionally embeds boundary conditions as "soft" penalty terms in the loss function, resulting in non-zero boundary residuals and sometimes poor convergence near critical regions. In Chapter 4, analytical modulating functions are derived to transform the original outputs of the NN to automatically satisfy Dirichlet, Neumann, and mixed boundary conditions in simple geometries. This "hard" manner treatment ensures zero boundary residuals by construction, and therefore, eliminates the need for the penalty term for boundary conditions. The proposed method is validated through the identification of unknown rotational stiffness of semi-rigid joints, which is an inverse problem. Numerical case studies, including identifying the rotational stiffness of a variable cross-section cantilever beam and a steel frame, are carried out to comprehensively analyze the performance of the vanilla PINN and the proposed method. In addition, experimental validation on a single-bay steel frame is conducted to demonstrate the proposed method in practical applications. The results show that the proposed modulating function-based method can improve prediction accuracy as

well as computational efficiency.

3. TP-PINN framework for decoupling the penalty term for boundary conditions to a separate training stage. To extend the “hard” enforcement of boundary conditions to irregular domains, Chapter 5 develops a unified framework employing pretrained NNs in place of closed-form modulating functions. By structuring the overall framework into two phases, one phase encodes boundary conditions, and the other handles the governing equation, any domain shape can be accommodated. Moreover, since the training of boundary conditions and the training of the governing equation are decoupled, the adverse effects of the multi-objective optimization can be mitigated. The proposed TP-PINN framework significantly enhances computational efficiency and prediction accuracy. The performance of TP-PINN is demonstrated through two classic structural problems: deformation of an Euler-Bernoulli beam and a Kirchhoff-Love plate under loading.
4. Leverage additional structural information with PIGNN. Structural systems such as trusses, frames, and tensegrity structures exhibit intrinsic graph-like connectivity, which FCFNN in the vanilla PINN cannot exploit. In Chapter 6, a novel framework termed PIGNN that integrates equilibrium-based physical penalty and structural topology with GNN, is proposed. In the PIGNN framework, structural topology can be learned naturally from the connectivity of nodes and edges of graphs. Both vanilla PINN and PIGNN are applied to solve prestress design problems of tensegrity structures using a force density-based method. Results show that by incorporating structural topology, PIGNN outperforms vanilla PINN in prestress design and offers a

scalable way to PIML solutions for large, complex structures.

These methodological advances collectively broaden the applicability of PIML in structural engineering, ranging from forward problems, e.g. structural analysis, to inverse problems, e.g. parameter identification.

Despite these advancements, several limitations warrant further consideration:

1. The auxiliary output approach increases network complexity and memory demands, which may strain hardware by introducing multiple auxiliary outputs when scaling to very high dimensional or very fine discretization.
2. The additional penalty term introduced by auxiliary outputs may cause difficulties in the balance of multiple loss terms during training.
3. While analytical modulating functions excel in simple geometries by completely eliminating the penalty term for boundary conditions, they cannot be derived for complex domains. TP-PINN addresses this gap but may fail in certain cases.
4. For cases with boundary conditions involving high-order derivatives, pretrained NNs in the first phase of TP-PINN bring additional computational cost, which may hinder computational efficiency.
5. PIGNN targets structures with graph-like connectivity such as cable-strut systems. Its applicability in continuum-like domains or multi-scale structures needs further exploration.
6. Finally, it is essential to translate these theoretical PIML innovations into robust, deployable tools for real world structural engineering applications. The following difficulties might be encountered. First, measurement data is often noisy, incomplete,

and irregularly sampled, which may lead to unstable and complicated training dynamics of PIML models. Second, real structures involve complex boundary conditions, heterogeneous materials, and uncertain loading conditions, which may cause difficulties in establishing accurate loss functions and uncertainties in results. Third, computational costs may become significant for large-scale structures due to the high dimensionality of the degrees of freedom (DOFs) and the huge volume of measurement data. Fourth, validation may become more challenging because the ground truth is typically unavailable, requiring indirect assessment against monitoring data or reduced-order models. These aspects indicate both difficulties and potentials of applying the proposed methods in real world engineering practice.

7.2 Recommendations

Building on the foundational methods proposed in this thesis, the following avenues are recommended for future investigation to further enhance PIML's capabilities and practical impact in structural engineering:

1. Laboratory and field validation. While numerical studies have confirmed the improvements in efficiency and accuracy achieved by auxiliary outputs, modulating functions, TP-PINN, and PIGNN, their performance under practical conditions should be further tested. Laboratory experiments such as vibration tests on beams and plates, and field tests on existing structures will expose noise in measurements and uncertainty in identified parameters. Experimental validation will identify gaps between simulation and practice, guiding improvements in the robustness of the

proposed frameworks.

2. Hybrid coupling with FE solvers. To leverage established engineering analysis tools, future work should integrate PIML models into finite element analysis (FEA) packages. One promising approach is to use PIML models (e.g. PINN with auxiliary outputs or TP-PINN) to approximate local constitutive or structural behavior, embedding them as custom element models within an FE solver. This integrated framework combines the global discretization accuracy of FEA with high adaptability of PIML, enabling efficient real-time simulations for design iterations, digital twins, and SHM.
3. Uncertainty quantification (UQ) via Bayesian-based PIML. Experimental data and model parameters inevitably carry uncertainties. Incorporating UQ into PIML is crucial for risk-informed decision-making. Future research should extend the proposed methods to Bayesian frameworks. By treating trainable parameters of NNs as random variables, uncertainties in inputs and measurements can be propagated through the PIML framework, yielding posterior distributions for predicted fields and parameters. This probabilistic PIML would provide confidence intervals for stress, displacement, and stiffness estimates, which are vital for engineering applications.
4. Extending to multi-physics problems. Many real-world challenges such as fluid-structure interaction and thermal-stress coupling require solving multi-physics PDE systems. Future work should generalize the proposed frameworks to handle coupled PDE systems and expand applications of PIML into aerospace, geotechnical, and biomedical engineering.

5. Automated architecture search and TL. Designing the optimal NN architecture currently requires manual tuning of NN hyperparameters, including network depth, width, and activation functions. Automated ML techniques can be employed to discover optimal network architecture for a specific condition. Furthermore, TL across multiple problem scenarios could yield pretrained models that accelerate convergence on new tasks using minimal additional data.

By pursuing these directions, the methods developed in this thesis can evolve into robust, versatile, and widely adopted tools that redefine the modelling, designing, and analyzing of structural engineering systems.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 265-283.

Ahmmed, B., Vesselinov, V. V., & Mudunuru, M. K. (2022). SmartTensors: Unsupervised and physics-informed machine learning framework for the geoscience applications. In *SEG International Exposition and Annual Meeting*, D011S196R001. SEG.

Alber, M., Buganza Tepole, A., Cannon, W. R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W. W., Perdikaris, P., Petzold, L., & Kuhl, E. (2019). Integrating machine learning and multiscale modeling-perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ Digital Medicine*, 2(1), 115.

Alhajeri, M. S., Abdullah, F., Wu, Z., & Christofides, P. D. (2022). Physics-informed machine learning modeling for predictive control using noisy data. *Chemical Engineering Research and Design*, 186, 34-49.

Amari, S. I. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5), 185-196.

Andersen, K., Cook, G. E., Karsai, G., & Ramaswamy, K. (1990). Artificial neural networks applied to ARC welding process modeling and control. *IEEE Transactions on Industry Applications*, 26(5), 824-830.

Arzani, A., Wang, J. X., & D'Souza, R. M. (2021). Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, 33(7).

Authier, J., Haider, R., Annaswamy, A., & Dörfler, F. (2024). Physics-informed graph neural network for dynamic reconfiguration of power systems. *Electric Power Systems Research*, 235, 110817.

Bastek, J. H., & Kochmann, D. M. (2023). Physics-informed neural networks for shell structures. *European Journal of Mechanics-A/Solids*, 97, 104849.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(153), 1-43.

Beck, A., Flad, D., & Munz, C. D. (2019). Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, 398, 108910.

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, 10(4), 122.

Bloemheuvel, S., van den Hoogen, J., & Atzmueller, M. (2021). A computational framework for modeling complex sensor network data using graph signal processing and graph neural networks in structural health monitoring. *Applied Network Science*, 6(1), 97.

Bode, M., Gauding, M., Lian, Z., Denker, D., Davidovic, M., Kleinheinz, K., Jitsev, J., & Pitsch, H. (2021). Using physics-informed enhanced super-resolution generative adversarial

networks for subfilter modeling in turbulent reactive flows. *Proceedings of the Combustion Institute*, 38(2), 2617-2625.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4), 18-42.

Buckley, A. G. (1978). A combined conjugate-gradient quasi-Newton minimization algorithm. *Mathematical Programming*, 15(1), 200-210.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5), 1190-1208.

Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021a). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12), 1727-1738.

Cai, S., Wang, Z., Fuest, F., Jeon, Y. J., Gray, C., & Karniadakis, G. E. (2021b). Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, A102.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6572-6583.

Chen, X., Fu, Y., Liu, S., & Di, X. (2023). Physics-informed neural operator for coupled forward-backward partial differential equations. In *1st Workshop on the Synergy of Scientific and Machine Learning Modeling*. ICML2023.

Chen, Y., Huang, D., Zhang, D., Zeng, J., Wang, N., Zhang, H., & Yan, J. (2021). Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method. *Journal of Computational Physics*, 445, 110624.

Chen, Y., Lu, L., Karniadakis, G. E., & Dal Negro, L. (2020a). Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8), 11618-11633.

Chen, Y., Yan, J., Sareh, P., & Feng, J. (2020b). Feasible prestress modes for cable-strut structures with multiple self-stress states using particle swarm optimization. *Journal of Computing in Civil Engineering*, 34(3), 04020003.

Chenaud, M., Alves, J., & Magoulès, F. (2023). Physics-informed graph convolutional networks: Towards a generalized framework for complex geometries. *arXiv preprint* arXiv:2310.14948.

Cohen, T., Weiler, M., Kicanaoglu, B., & Welling, M. (2019). Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning*, 1321-1330. PMLR.

Connelly, R. (2002). Tensegrity structures: why are they stable?. In *Rigidity Theory and Applications*. Boston, MA: Springer US.

Cuomo, S., Cola, V. S. D., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3), 88.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4), 303-314.

Di Lorenzo, D., Champaney, V., Ghnatios, C., Cueto, E., & Chinesta, F. (2024). Physics-informed and graph neural networks for enhanced inverse analysis. *Engineering Computations*.

Dissanayake, M. G., & Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3), 195-201.

Du, Y., & Zaki, T. A. (2021). Evolutional deep neural network. *Physical Review E*, 104(4), 045303.

Dupont, E., Doucet, A., & Teh, Y. W. (2019). Augmented Neural ODEs. *Advances in Neural Information Processing Systems*, 32.

Duraisamy, K., Iaccarino, G., & Xiao, H. (2019). Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1), 357-377.

Dwivedi, V., & Srinivasan, B. (2020). Physics informed extreme learning machine (PIELM)-A rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391, 96-118.

E, W., & Yu, B. (2018). The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1), 1-12.

Eivazi, H., Tahani, M., Schlatter, P., & Vinuesa, R. (2022). Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. *Physics of Fluids*, 34(7).

Fang, Z. (2021). A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10), 5514-5526.

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)-A review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14-15), 2627-2636.

Gere, J. M., Timoshenko, S.P. (1997). Mechanics of materials. 4th ed. Boston: PWS.

Goswami, S., Bora, A., Yu, Y., & Karniadakis, G. E. (2023). Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*, 219-254. Cham: Springer International Publishing.

Goswami, S., Yin, M., Yu, Y., & Karniadakis, G. E. (2022). A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391, 114587.

Guest, S. (2006). The stiffness of prestressed frameworks: A unifying approach. *International Journal of Solids and Structures*, 43(3-4), 842-854.

Haghighat, E., Can Bekar, A., Madenci, E., & Juanes, R. (2021a). Deep learning for solution and inversion of structural mechanics and vibrations. In *Modeling and Computation in Vibration Problems, Volume 2: Soft Computing and Uncertainty*, 1-1. Bristol, UK: IOP Publishing.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2021b). A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379, 113741.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778.

He, Q., & Tartakovsky, A. M. (2021). Physics-informed neural network method for forward and backward advection-dispersion equations. *Water Resources Research*, 57(7), e2020WR029479.

He, Q., Barajas-Solano, D., Tartakovsky, G., & Tartakovsky, A. M. (2020). Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141, 103610.

Henkes, A., Wessels, H., & Mahnken, R. (2022). Physics informed neural networks for continuum micromechanics. *Computer Methods in Applied Mechanics and Engineering*, 393, 114790.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359-366.

Huang, Y. H., Xu, Z., Qian, C., & Liu, L. (2023). Solving free-surface problems for non-shallow water using boundary and initial conditions-free physics-informed neural network (bif-PINN). *Journal of Computational Physics*, 479, 112003.

Jagtap, A. D., Kharazmi, E., & Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365, 113028.

Jeong, H., Bai, J., Batuwatta-Gamage, C. P., Rathnayaka, C., Zhou, Y., & Gu, Y. (2023). A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization. *Engineering Structures*, 278, 115484.

Jhin, S. Y., Jo, M., Kong, T., Jeon, J., & Park, N. (2021). ACE-NODE: Attentive co-evolving neural ordinary differential equations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 736-745.

Jian, X., Xia, Y., Duthé, G., Bacsa, K., Liu, W., & Chatzi, E. (2024). Using graph neural networks and frequency domain data for automated operational modal analysis of populations of structures. *arXiv preprint* arXiv:2407.06492.

Jin, H., Zhang, E., & Espinosa, H. D. (2023). Recent advances and applications of machine learning in experimental solid mechanics: A Review. *Applied Mechanics Reviews*, 75(6), 061001.

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426, 109951.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260.

Karimi, S., & Gündel, M. (2024). Advancing structural health monitoring in civil engineering with grey-box modelling: A review. In *11th European Workshop on Structural Health Monitoring (EWSHM 2024)*.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3, 422-440.

Kashefi, A., Rempe, D., & Guibas, L. J. (2021). A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2), 027104.

Kharazmi, E., Cai, M., Zheng, X., Zhang, Z., Lin, G., & Karniadakis, G. E. (2021a). Identifiability and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks. *Nature Computational Science*, 1(11), 744-753.

Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2019). Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint* arXiv: 1912.00873.

Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2021b). hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374, 113547.

Kim, M., Chau, N. K., Park, S., Nguyen, P. C., Baek, S. S., & Choi, S. (2025). Physics-aware machine learning for computational fluid dynamics surrogate model to estimate ventilation performance. *Physics of Fluids*, 37(2).

Kingma, D. P. & Ba, Y. (2017). Adam: A method for stochastic optimization. *arXiv preprint* arXiv: 1412.6980.

Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., & Perdikaris, P. (2020). Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358, 112623.

Koohestani, K., & Guest, S. D. (2013). A new approach to the analytical and numerical form-finding of tensegrity structures. *International Journal of Solids and Structures*, 50(19), 2995-3007.

Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 987-1000.

Lai, Z., Liu, W., Jian, X., Bacsa, K., Sun, L., & Chatzi, E. (2022). Neural modal ordinary differential equations: Integrating physics-based modeling with neural ordinary differential equations for modeling high-dimensional monitored structures. *Data-Centric Engineering*, 3, e34.

Lai, Z., Mylonas, C., Nagarajaiah, S., & Chatzi, E. (2021). Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508, 116196.

Lapeyre, C. J., Misdariis, A., Cazard, N., Veynante, D., & Poinsot, T. (2019). Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates. *Combustion and Flame*, 203, 255-264.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

Lee, S., Lieu, Q. X., Vo, T. P., & Lee, J. (2022). Deep neural networks for form-finding of tensegrity structures. *Mathematics*, 10(11), 1822.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021a). Fourier neural operator for parametric partial differential equations. *arXiv preprint* arXiv: 2010.08895.

Li, Z., Liu, F., Yang, W., Peng, S. & Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999-7019.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., & Anandkumar, A. (2021b). Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3), 1-27.

Ling, J., & Templeton, J. (2015). Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. *Physics of Fluids*, 27(8), 085103.

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807, 155-166.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1), 503-528.

Liu, M., Liang, L., & 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, W., & Pyrcz, M. J. (2023). Physics-informed graph neural network for spatial-temporal production forecasting. *Geoenergy Science and Engineering*, 223, 211486.

Liu, X. Y., Sun, H., Zhu, M., Lu, L., & Wang, J. X. (2022). Predicting parametric spatiotemporal dynamics by multi-resolution PDE structure-preserved deep learning. *Bulletin of the American Physical Society*, 67.

Lu, L., Jin, P., & Karniadakis, G. E. (2019). DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint* arXiv:1910.03193.

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021a). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3), 218-229.

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021b). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1), 208-228.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., & Johnson, S. G. (2021c). Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6), B1105-B1132.

Luo, T., & Yang, H. (2024). Two-layer neural networks for partial differential equations: Optimization and generalization theory. *Handbook of Numerical Analysis*, 25, 515-554.

Majumdar, R., Varhade, A., Karande, S., & Vig, L. (2023). Can physics informed neural operators self improve?. *arXiv preprint* arXiv:2311.13885.

Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360, 112789.

Massaroli, S., Poli, M., Park, J., Yamashita, A., & Asama, H. (2020). Dissecting Neural ODEs. *Advances in Neural Information Processing Systems*, 33, 3952-3963.

Mattey, R., & Ghosh, S. (2022). A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390, 114474.

Mattheakis, M., Protopapas, P., Sondak, D., Giovanni, M. D., & Kaxiras, E. (2020). Physical symmetries embedded in neural networks. *arXiv preprint* arXiv: 1904.08991.

Maust, H., Li, Z., Wang, Y., Leibovici, D., Bruno, O., Hou, T., & Anandkumar, A. (2022). Fourier continuation for exact derivative computation in physics-informed neural operators. *arXiv preprint* arXiv:2211.15960.

Mehta, P. P., Pang, G., Song, F., & Karniadakis, G. E. (2019). Discovering a universal variable-order fractional model for turbulent Couette flow using a physics-informed neural network. *Fractional Calculus and Applied Analysis*, 22(6), 1675-1688.

Moseley, B., Markham, A., & Nissen-Meyer, T. (2020). Solving the wave equation with physics-informed deep learning. *arXiv preprint* arXiv:2006.11894.

Nascimento, R. G., Fricke, K., & Viana, F. A. (2020). A tutorial on solving ordinary differential equations using Python and hybrid physics-informed neural network. *Engineering Applications of Artificial Intelligence*, 96, 103996.

Ngo, Q. H., Nguyen, B. L., Vu, T. V., Zhang, J., & Ngo, T. (2024). Physics-informed graphical neural network for power system state estimation. *Applied Energy*, 358, 122602.

Niaki, S. A., Haghighat, E., Campbell, T., Poursartip, A., & Vaziri, R. (2021). Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Computer Methods in Applied Mechanics and Engineering*, 384, 113959.

Nickolls, J., Dally, W. J. (2010). The GPU computing era. *IEEE Micro*, 30(2), 56-69.

Ohsaki, M., & Zhang, J. (2006). Stability conditions of prestressed pin-jointed structures. *International Journal of Non-Linear Mechanics*, 41(10), 1109-1117.

O'Leary, J., Paulson, J. A., & Mesbah, A. (2022). Stochastic physics-informed neural ordinary differential equations. *Journal of Computational Physics*, 468, 111466.

O'Shea, K. & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint* arXiv: 1511.08458.

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. *Proceedings of the IEEE*, 96(5), 879-899.

Owhadi, H. (2015). Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, 13(3), 812-828.

Pang, G., Lu, L., & Karniadakis, G. E. (2019). fPINNs: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4), A2603-A2626.

Parish, E. J., & Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305, 758-774.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in Pytorch. In *31st Conference on Neural Information Processing Systems*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8026-8037.

Pellegrino, S., & Calladine, C. R. (1986). Matrix analysis of statically and kinematically indeterminate frameworks. *International Journal of Solids and Structures*, 22(4), 409-428.

Peng, C., Xia, F., Saikrishna, V., & Liu, H. (2022). Physics-informed graph learning. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, 732-739. IEEE.

Peng, J. Z., Aubry, N., Li, Y. B., Mei, M., Chen, Z. H., & Wu, W. T. (2023). Physics-informed graph convolutional neural network for modeling geometry-adaptive steady-state natural convection. *International Journal of Heat and Mass Transfer*, 216, 124593.

Pfau, D., Spencer, J. S., Matthews, A. G., & Foulkes, W. M. C. (2020). Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*, 2(3), 033429.

Popescu, M. C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7), 579-588.

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357, 125-141.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017a). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348, 683-693.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017b). Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint* arXiv:1711.10561.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019a). 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.

Raissi, M., Wang, Z., Triantafyllou, M. S., & Karniadakis, G. E. (2019b). Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861, 119-137.

Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481), 1026-1030.

Ramabathiran, A. A., & Ramachandran, P. (2021). SPINN: Sparse, physics-based, and partially interpretable neural networks for PDEs. *Journal of Computational Physics*, 445, 110600.

Ramos-Osuna, V., Díaz-Álvarez, A., & Lara-Cabrera, R. (2025). Efficient n-body simulations using physics informed graph neural networks. *arXiv preprint* arXiv:2504.01169.

Ren, P., Rao, C., Liu, Y., Wang, J. X., & Sun, H. (2022). PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering*, 389, 114399.

Rojas, C. J., Bitterncourt, M. L., & Boldrini, J. L. (2021). Parameter identification for a damage model using a physics informed neural network. *arXiv preprint* arXiv:2107.08781.

Rumelhart, D. E., Hinton, G. E., & Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). Recent advances in recurrent neural networks. *arXiv preprint* arXiv: 1801.01078.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3), 210-229.

Shao, X., Liu, Z., Zhang, S., Zhao, Z., & Hu, C. (2023). PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh. *Building and Environment*, 232, 110056.

Sharma, P., Chung, W. T., Akoush, B., & Ihme, M. (2023). A review of physics-informed machine learning in fluid mechanics. *Energies*, 16(5), 2343.

Shi, D., Han, A., Lin, L., Guo, Y., Wang, Z., & Gao, J. (2025). Design your own universe: A physics-informed agnostic method for enhancing graph neural networks. *International Journal of Machine Learning and Cybernetics*, 16(2), 1129-1144.

Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv preprint* arXiv:2004.01806.

Shukla, K., Jagtap, A. D., & Karniadakis, G. E. (2021). Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447, 110683.

Shukla, K., Xu, M., Trask, N., & Karniadakis, G. E. (2022). Scalable algorithms for physics-informed neural and graph networks. *Data-Centric Engineering*, 3, e24.

Singh, A. P., Medida, S., & Duraisamy, K. (2017). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, 55(7), 2215-2227.

Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 1339-1364.

Skelton, R. E., & De Oliveira, M. C. (2009). Tensegrity systems. New York: Springer.

Suk, J., Alblas, D., Hutten, B. A., Wiegman, A., Brune, C., van Ooij, P., & Wolterink, J. M. (2024). Physics-informed graph neural networks for flow field estimation in carotid arteries. *arXiv preprint* arXiv:2408.07110.

Sun, L., Gao, H., Pan, S., & Wang, J. X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361, 112732.

Sun, R. Y. (2020). Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2), 249-294.

Thangamuthu, A., Kumar, G., Bishnoi, S., Bhattoo, R., Krishnan, N. M., & Ranu, S. (2022). Unravelling the performance of physics-informed graph neural networks for dynamical systems. *Advances in Neural Information Processing Systems*, 35, 3691-3702.

Tran, H. C., & Lee, J. (2010). Advanced form-finding of tensegrity structures. *Computers & structures*, 88(3-4), 237-246.

Viana, F. A., Nascimento, R. G., Dourado, A., & Yucesan, Y. A. (2021). Estimating model inadequacy in ordinary differential equations with physics-informed neural networks. *Computers & Structures*, 245, 106458.

bin Waheed, U., Haghighat, E., Alkhalifah, T., Song, C., & Hao, Q. (2021). PINNeik: Eikonal solution using physics-informed neural networks. *Computers & Geosciences*, 155, 104833.

Wang, C., Berner, J., Li, Z., Zhou, D., Wang, J., Bae, J., & Anandkumar, A. (2024). Beyond closure models: Learning chaotic-systems via physics-informed neural operators. *arXiv preprint* arXiv:2408.05177.

Wang, S., Teng, Y., & Perdikaris, P. (2021a). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5), A3055-A3081.

Wang, S., Yu, X., & Perdikaris, P. (2022). When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449, 110768.

Wang, Y., Xu, X., & Luo, Y. (2021b). Form-finding of tensegrity structures via rank minimization of force density matrix. *Engineering Structures*, 227, 111419.

Wight, C. L., & Zhao, J. (2020). Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint* arXiv:2007.04542.

Willard, J., Jia, X., Xu, S., Steinbach, M., & Kumar, V. (2022). Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55(4), 1-37.

Xu, Y., Kohtz, S., Boakye, J., Gardoni, P., & Wang, P. (2023). Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges. *Reliability Engineering & System Safety*, 230, 108900.

Yang, L., Meng, X., & Karniadakis, G. E. (2021). B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425, 109913.

Yang, Y., & Perdikaris, P. (2019a). Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394, 136-152.

Yang, Y., & Perdikaris, P. (2019b). Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems. *Computational Mechanics*, 64, 417-434.

Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94, 103-114.

Yin, M., Ban, E., Rego, B. V., Zhang, E., Cavinato, C., Humphrey, J. D., & Em Karniadakis, G. (2022a). Simulating progressive intramural damage leading to aortic dissection using

DeepONet: An operator-regression neural network. *Journal of the Royal Society Interface*, 19(187), 20210670.

Yin, M., Zhang, E., Yu, Y., & Karniadakis, G. E. (2022b). Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 402, 115027.

You, H., Zhang, Q., Ross, C. J., Lee, C. H., Hsu, M. C., & Yu, Y. (2022). A physics-guided neural operator learning approach to model biological tissues from digital image correlation measurements. *Journal of Biomechanical Engineering*, 144(12), 121012.

Yu, J., Lu, L., Meng, X., & Karniadakis, G. E. (2022). Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393, 114823.

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7), 1235-1270.

Yuan, L., Ni, Y. Q., Deng, X. Y., & 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.

Zhang, E., Dao, M., Karniadakis, G. E., & Suresh, S. (2022a). Analyses of internal structures and defects in materials using physics-informed neural networks. *Science Advances*, 8(7), eabk0644.

Zhang, E., Yin, M., & Karniadakis, G. E. (2020a). Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging. *arXiv preprint* arXiv:2009.04525.

Zhang, H. F., Lu, X. L., Ding, X., & Zhang, X. M. (2024). Physics-informed line graph neural network for power flow calculation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(11).

Zhang, H., Jiang, L., Chu, X., Wen, Y., Li, L., Liu, J., Xiao, Y., & Wang, L. (2025a). Combining physics-informed graph neural network and finite difference for solving forward and inverse spatiotemporal PDEs. *Computer Physics Communications*, 308, 109462.

Zhang, J. Y., & Ohsaki, M. (2007). Stability conditions for tensegrity structures. *International Journal of Solids and Structures*, 44(11-12), 3875-3886.

Zhang, J. Y., & Ohsaki, M. (2015). Tensegrity structures. New York: Springer.

Zhang, R., Liu, Y., & Sun, H. (2020b). Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 369, 113226.

Zhang, W., Ni, Y. Q., Wang, S. M., Yuan, L., & Hao, S. (2025b) A novel approach for identifying rotational stiffness of semirigid joints by physics-informed neural networks. *Journal of Computing in Civil Engineering*, 39(4), 04025039.

Zhang, X., Zhu, Y., Wang, J., Ju, L., Qian, Y., Ye, M., & Yang, J. (2022b). GW-PINN: A deep learning algorithm for solving groundwater flow equations. *Advances in Water Resources*, 165, 104243.

Zhang, Z., & Gu, G. X. (2021). Physics-informed deep learning for digital materials. *Theoretical and Applied Mechanics Letters*, 11(1), 100220.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57-81.

Zhu, M., Peng, Y., Ma, W., Guo, J., & Lu, J. (2023). Artificial neural network-aided force finding of cable dome structures with diverse integral self-stress states-framework and case study. *Engineering Structures*, 285, 116004.

Zhu, Q., Liu, Z., & Yan, J. (2021). Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, 67, 619-635.