THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學
Pao Yue-kong Library
包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

Pao Yue-kong Library, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

http://www.lib.polyu.edu.hk

SECOND-ORDER ANALYSIS OF PILE-SUPPORTED FRAMED STRUCTURES BY MACHINE LEARNING-ENHANCED FINITE-ELEMENT METHOD

WEIHANG OUYANG

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University

Department of Civil and Environmental Engineering

Second-order Analysis of Pile-supported Framed Structures by Machine Learning-Enhanced Finite-Element Method

Weihang Ouyang

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Oct 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)

Weihang Ouyang (Name of student)

To my Family

ABSTRACT

The second-order analysis method, a modern simulation-based design approach, is widely recommended by global design guidelines. This approach requires accurate and efficient numerical methods to evaluate design results for large-scale structural systems. Among various numerical methods, the line finite element method (LFEM) is preferred in current design practices. The advanced LFEMs offer notable advantages: a single element can model an entire structural member and effectively capture critical nonlinear behaviors, streamlining both modeling efforts and computational demands. Despite its proven success in superstructure design, the application of the second-order analysis method to pile-supported framed structures remains underexplored.

Current second-order analysis methods often simplify building structures by modeling them as fixed on the ground, neglecting pile foundations due to the complexity of nonlinear Soil-Pile Interaction (SPI). Various soil resistance models, such as *p-y* curves, have been developed and implemented within the LFEM to model SPI. However, these methods are mainly suited for single piles and are rarely applied to pile-supported structures with multiple piles due to high computational costs. Accurately modeling SPI along piles typically requires a dense mesh, often exceeding 100 elements per pile. For buildings with multiple piles, LFEM requires large degrees of freedom (DOFs), resulting in a tedious analysis process. Therefore, developing an efficient numerical method that accurately captures SPI in pile-supported structures using a coarse mesh, ideally one element per pile, is essential for the broader practical adoption of second-order analysis methods.

To enable a more effective SPI analysis, the refined Three-dimensional Pile Element (3DPE) formulation is presented for pile-supported framed structures. The proposed refined 3DPE formulation can efficiently capture the nonlinear SPI that is over-simplified in the conventional analysis method. Compared with the existing LFEM, the proposed refined 3DPE formulation can more accurately capture the structural behaviors of piles under complex loading conditions, including the shear

effect and axial-torsional coupled response. Furthermore, three soil stiffness matrices are derived to improve the numerical stability of spatial analysis using the existing pile element formulation. Extensive examples are provided to validate the effectiveness of the refined 3DPE formulation, indicating its capability in accurately modeling three-dimensional SPI with coarse mesh. Nevertheless, this method still cannot efficiently model pile-supported framed structures with one element per pile.

In addition to conventional mesh-based numerical methods, machine learning (ML) techniques have significantly transformed structural analysis recently. ML models, trained to predict structural system responses, provide near-instant solutions with negligible computational cost. However, applying ML techniques to large-scale systems remains challenging due to the high costs of training and data collection. Moreover, large-scale structural systems are often project-specific, indicating pre-trained ML models are less reusable for different projects. This lack of reusability significantly limits the practical application of ML techniques.

In this thesis, a hybrid numerical framework combining the finite element method (FEM) and ML is proposed for solving general partial differential equation (PDE) problems. The proposed method, named the Neural Operator Element Method (NOEM), utilizes ML models known as neural operators (NOs) to model specific subdomains of the system. This streamlined approach reduces the training process time. These trained NOs are then used as neural-operator elements (NOEs) to model the corresponding subdomains, while other areas are modeled with standard finite elements (FEs). This hybrid modeling approach significantly reduces computational costs by replacing dense meshes in NOE-modeled subdomains with single NOEs, without compromising accuracy. Additionally, the ML models can be reused in different PDE systems without additional training, enhancing their reusability in design practice.

As ML models, especially NOs in operator learning, play a vital role in NOEM, this thesis develops a novel sampling method to enhance ML model training processes.

The proposed method, called the Residual-based Adversarial-gradient Moving Sample (RAMS) method, demonstrates stable performance across various tasks, including physics-informed neural networks (PINN), physics-informed operator learning, and data-driven operator learning. Notably, RAMS is the first adaptive sampling approach that can be efficiently implemented for operator learning, representing a significant contribution to the field.

To implement NOEM for pile-supported framed structures, the governing equations describing the structural responses of single piles are derived. These equations account for the geometric nonlinearity of single piles undergoing large deformations and the nonlinear SPI. In addition, the governing equations derived are then implemented within the PINN, a data-free ML training paradigm, to train models for large deflection analysis of single piles.

Finally, the governing equations are integrated into NOEM to provide a practical numerical method for second-order analysis of pile-supported framed structures. To improve training efficiency, the proposed RAMS method is also used to train NOs for single piles within the entire structural system. This numerical method can accurately model the structural behavior of piles in pile-supported structures using one NOE per pile, showcasing its potential for advancing second-order analysis methods.

A key feature of this research is the development of a hybrid numerical framework that unifies two distinct methodologies, FEM and ML, to solve PDE problems and then to apply for large-scale pile-supported framed structures. It is believed that this study will not only facilitate the application of the second-order analysis method to pile-supported structures but also demonstrate the potential for efficiently handling large-scale, complex numerical simulations in other challenging engineering analysis problems.

Keywords: Second-order analysis method; Numerical method; (Line) Finite element method; Machine learning; Operator learning; Soil-Pile Interaction.

PUBLICATION AND AWARD

Journal Paper:

[1] **Ouyang, W.**, Shin, Y., Liu, S.-W. & Lu, L. Neural-operator element method: Efficient and scalable finite element method enabled by reusable neural operators. *arXiv preprint arXiv:2506.18427* (2025; Accepted in *Nature Computational Science*).

[2] **Ouyang, Weihang**, Liu, S.-W. & Chan, S.-L. Second-order Analysis Method for Pile-supported Structures using Neural-Operator Element Method. (Under Review in *Engineering Structures*)

[3] **Ouyang, W.**, Zhu, M., Liu, S.-W. & Lu, L. RAMS: Residual-based adversarial-gradient moving sample method for scientific machine learning in solving partial differential equations. *arXiv preprint arXiv:2509.01234* (2025; Accepted in *Advanced Intelligent Discovery*).

[4] **Ouyang, Weihang**, Liu, K. & Liu, S.-W. Soil Nail Design Optimization through Adaptive Slope Stability Analysis using Improved Limit Equilibrium Method. (Under Revision in *International Journal for Numerical and Analytical Methods in Geomechanics*)

[5] **Ouyang, Weihang**, Ouyang, W., Liu, S.-W., Liu, Y.-P., & Chan, S.-L. Neural Network-based Spring Element Method using Deep Operator Network for Efficient Uncertainty Evaluation of Pile-supported Structures. (Under Revision in *Acta Geotechnica*)

[6] Ouyang, W., **Ouyang, Weihang**, Pan, D., Liu, S.-W., & Chan, S.-L. Physics-Informed Deep Operator Networks for Rapid Nonlinear Analysis of Glass Panes under Variable Wind-Induced Loads. *Engineering Structures* (2025; Accepted).

[7] Chen, L., Zhang, H.-Y., **Ouyang, Weihang** & Liu, S.-W. Implementation of physics informed neural networks for geometrically nonlinear analysis of non-prismatic members. *Structures* 71, 108149 (2025).

[8] Ouyang, W., **Ouyang, Weihang**, Tian, X., Liu, S.-W., Wang, J. & Chan, S.-L. Nonlinear analysis of glass panes under complex lateral pressures through physics informed neural networks. *Engineering Structures* 334, 120262 (2025).

[9] **Ouyang, Weihang**, Chen, L., Liang, A.-R. & Liu, S.-W. Neural networks-based line element method for large deflection frame analysis. *Computers and Structures* 300, 107425 (2024).

[10] **Ouyang, Weihang**, Li, G., Chen, L. & Liu, S.-W. Physics-informed neural networks for large deflection analysis of slender piles incorporating non-differentiable soil-structure interaction. *International Journal for Numerical and Analytical Methods in Geomechanics* 48, 1278–1308 (2024).

[11] **Ouyang, Weihang**, Liu, S., Liu, K. & Yin, J. Efficient numerical implementation of limit equilibrium method for stability analysis of unsaturated soil slopes using Gaussian integral. *Acta Geotechnica*, 1–13 (2024).

[12] **Ouyang, Weihang**, Chen, L. & Liu, S.-W. Neural networks-based spring element for second-order analysis of pile-supported structures with nonlinear soil-structure interaction. *Engineering Structures* 310, 118093 (2024).

[13] Li, G., Ouyang, W., **Ouyang, Weihang** & Liu, S.-W. Static analysis of two-side supported 2-ply laminated glass panes through physics-informed neural networks. *Engineering Structures* 309, 118038 (2024).

[14] **Ouyang, Weihang**, Bai, R., Liu, S.-W. & Chan, S.-L. Refined three-dimensional pile element formulation for second-order analysis of pile-supported structures accounting for complex loading conditions. *Engineering Structures* 301, 117281 (2024).

[15] Li, G., Gu, Z.-Z., Zhang, H.-Y., **Ouyang, Weihang** & Liu, S.-W. Line finite element method for geometrically nonlinear analysis of functionally graded members accounting for twisting effects. *Composite Structures* 343, 118268 (2024).

[16] Li, G., Gu, Z.-Z., **Ouyang, Weihang**, Liu, S.-W. & Abdelrahman, A. Finite-element-based cross-section analysis method for functionally graded sandwich

members with arbitrary shapes and gradients. *Journal of Sandwich Structures & Materials*, 10996362241275557 (2024).

[17] **Ouyang, Weihang**, Li, G., Chen, L. & Liu, S.-W. Machine learning-based soil–structure interaction analysis of laterally loaded piles through physics-informed neural networks. *Acta Geotechnica*, 1–26 (2024).

[18] **Ouyang, Weihang**, Liang, A.-R. & Liu, S.-W. Efficient numerical implementation for nonlinear analysis of pile-supported structures during soil liquefaction through adaptive pile element formulations. *Soil Dynamics and Earthquake Engineering* 174, 108207 (2023).

[19] Yang, Y., **Ouyang, Weihang**, Liu, K. & Liu, S.-W. Efficient numerical algorithms for assessing the mechanical performance of corroded offshore steel sheet piles. *Ocean Engineering* 266, 112776 (2022).

[20] **Ouyang, Weihang**, Liu, S.-W. & Yang, Y. An improved Morgenstern-Price method using gaussian quadrature. *Computers and Geotechnics* 148, 104754 (2022).

Conference Paper:

[1] Liu, K., **Ouyang, Weihang**, Liu, S.-W. & Yin, J. Implementation of Improved Limit Equilibrium Method for Soil Slope Stability Analysis Considering Unsaturated Features in 8th Asia-Pacific Conference on Unsaturated Soils (2024).

[2] **Ouyang, Weihang**, Liu, S.-W. & Chan, S.-L. Second-Order Analysis Method for Pile-Supported Structures with Nonlinear Soil-Structure Interaction in Eleventh International Conference on Advances in Steel Structures (2023).

[3] **Ouyang, Weihang**, Chen, L. & Liu, S.-W. Implementation of Emerging Machine-learning Techniques for Second-order Analysis of Beam-columns through Physics-informed Neural Networks in The 3rd ZHITU Symposium on Advances in Civil Engineering (2023).

[4] Chen, L., **Ouyang, Weihang**, Liu, S.-W. & Ziemian, R. D. Numerical Implementation of GMNIA for Steel Frame with Nonsymmetric Sections in Cold-Formed Steel Research Consortium (CFSRC) Colloquium (2022).

[5] **Ouyang, Weihang**, Chen, L. & Liu, S.-W. Second-Order Direct Analysis for Steel H-Piles Accounting for Post-Driving Residual Stresses in The Tenth International Conference on Advances in Steel Structures (2021).

Awards

[1] International Collaborative Research Fellowship by *Hong Kong Polytechnic University* 2024

[2] Best Student Paper Award – Modeling Insights – by The Cold-formed Steel Research Consortium in CFSRC Colloquium 2022

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my chief supervisor, Dr. Si-Wei Liu, for his invaluable guidance, constant encouragement, and insightful advice throughout my Ph.D. study. His rigorous attitude toward research and continuous support have greatly inspired me and shaped my academic growth.

My sincere appreciation also goes to my co-supervisors, Prof. Siu-Lai Chan from the South China University of Technology and Prof. Tak-Ming Chan from The University of Hong Kong, for their constructive suggestions, thoughtful discussions, and generous support. Their expertise and kind guidance have been instrumental to the success of my research. I would also like to give my special thanks to Prof. Yao-Peng Liu for his thoughtfulness and continuous support. His valuable insights and warm encouragement have been a great source of motivation for me.

I am also very grateful to Dr. Lu Lu from Yale University for his guidance and support during my study at Yale. His enlightening discussions and advice provided me with valuable academic experiences and new perspectives.

I wish to express my heartfelt thanks to all NIDA team members, including Dr. Liang Chen, Dr. Wenfeng Chen, Mr. Haoyi Zhang, Mr. Guanhua Li, Mr. Yueyang Ding, Mr. Wenlong Gao, Mr. Anrui Liang, Miss. Wenjing Ouyang, Mr. Haien Xue, Mr. Zizang Gu for their collaboration, helpful discussions, and friendship. Their companionship has made my research journey both productive and enjoyable.

Finally, I would like to express my warmest gratitude to my family, especially my mother, for their unconditional love, patience, and encouragement throughout my study and life.

CONTENTS

LIST OF FIGURES

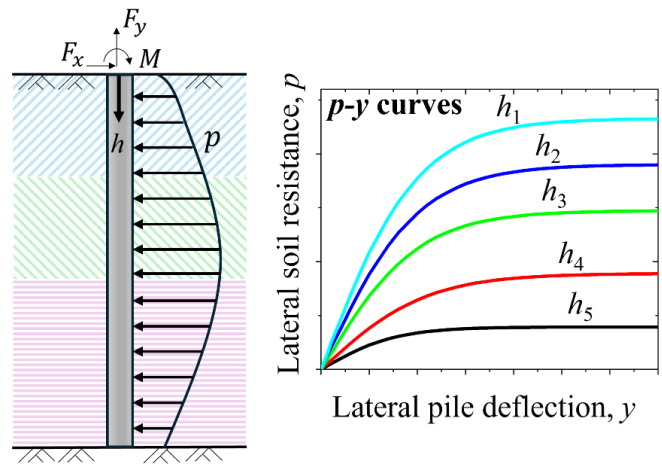
LIST OF TABLES

Chapter 1. INTRODUCTION

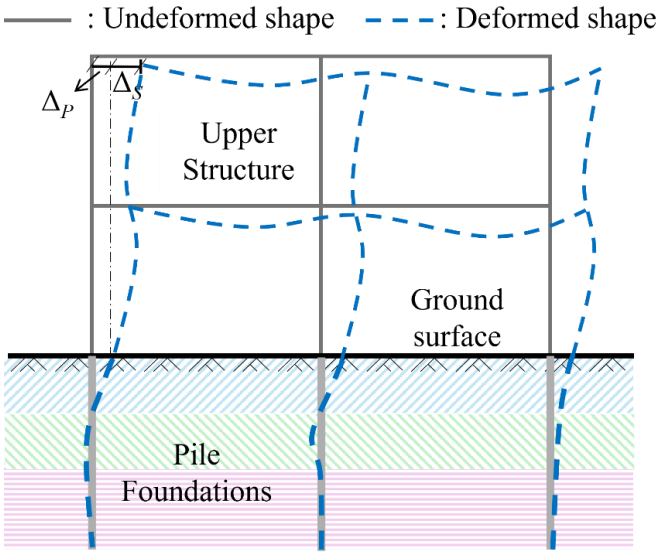
1.1 Background

The second-order analysis method is a simulation-based design approach that has gained increasing prominence in structural engineering practice. In contrast to traditional design methods, which typically rely on empirical formulations or pre-established design tables, the second-order analysis method evaluates structural performance using advanced and accurate simulation techniques. As it enables a more accurate and comprehensive structural analysis, this modern simulation-based design method can result in more effective and safer design results. Owing to these advantages, the second-order analysis method has been recommended and adopted by different modern structural design codes, representing a paradigm shift from conventional design practices to advanced, performance-based solutions.

As a simulation-based design approach, the second-order analysis method places stringent requirements on the underlying numerical simulation techniques. To ensure the reliability of design results, these numerical methods must be capable of comprehensively and accurately capturing the structural response under various loading conditions. Additionally, given the design process often involves evaluating numerous load cases and iterative design-schedule updates, the efficiency of the numerical approaches must also be ensured to maintain the overall effectiveness of the design process. Consequently, accurate and efficient numerical simulation methods are critical to the practical application of the second-order analysis method.



(A)



(B)

Figure 1-1 Illustration of Soil-Pile-Building Interaction of pile-supported framed structures: (A) Soil-Pile Interaction. (B) Pile-Building Interaction.

Over the past decades, a wide range of numerical analysis techniques have been developed and integrated into second-order analysis workflows for structural design. Among various numerical techniques, the line finite element method (LFEM) is currently one of the most frequently used in design practice as it poses a great balance between accuracy and efficiency. With the benefit of different advanced line finite element formulation, the modern LFEM can capture the structural response of the

structural system with one element per member without compromising accuracy. While this design philosophy has been widely adopted for upper structures, its application to pile-supported framed structures remains limited, primarily due to the lack of suitable numerical techniques tailored for these complex systems.

Piles have emerged as a fundamental solution in the foundation design of modern framed structures, offering several distinct advantages over shallow foundations. These pile foundations can transfer structural loads to deeper, more stable soil strata or bedrock, thereby ensuring the safety and stability of buildings constructed on sites with weak or compressible surface soils. This load-transfer capability is particularly advantageous in urban environments where high-rise buildings impose substantial loads that cannot be supported by near-surface soils alone. Furthermore, piles provide superior resistance to differential settlement and are effective in mitigating the adverse effects of expansive or liquefiable soils. In addition to enhancing load distribution, pile foundations can also be adapted for use in challenging geotechnical conditions, such as waterfront developments or areas prone to seismic activity, thus broadening the application of such foundations. Collectively, these attributes make pile foundations indispensable in the advancement of modern civil engineering and urban development. However, the complexity of the Soil-Pile-Building Interaction (SPBI) poses significant challenges to the numerical simulation approach, impeding the application of the second-order analysis method in pile-supported framed structures.

The SPBI can generally be divided into two components: (1) Soil–Pile Interaction (SPI) and (2) Pile–Building Interaction (PBI). Owing to the inherent discreteness of soil materials, the SPI is often characterized by significant nonlinearity and spatial variability. To accurately describe the complex soil resistance along piles, various soil resistance models have been developed, including the *p-y* curve model for lateral soil resistance (Figure 1-1). These models can then be integrated into the line finite element method (LFEM), wherein the pile is represented by beam–column elements and the

continuous soil medium is modeled using discrete soil springs. The springs, aligned with the element nodes, are constructed based on the corresponding soil resistance models to simulate the SPI. Although the LFEM employing these soil resistance models provides an accurate approach for modeling SPI, it is not well-suited to efficiently capture the full scope of SPBI.

As mentioned above, the LFEM represents the continuous surrounding soil through discretized soil spring elements. To accurately capture SPI effects, it is often necessary to divide a single pile into numerous small elements, sometimes exceeding 100 elements per pile, especially when accounting for soil nonlinearity and spatial soil variability. When simulating pile-supported structures with multiple piles, the computational cost associated with LFEM can become prohibitively high for practical applications due to the fine mesh requirement for accuracy. Consequently, the LFEM in the current design practices often models pile-supported framed structures in two separate parts: (1) the pile foundations and (2) the framed structure with fixed base conditions. While this approach significantly reduces computational effort, it can also lead to substantial underestimation of structural deformations, as it neglects pile head displacements and underestimates the corresponding P-Δ effect in the PBI (Figure 1-1). Therefore, to advance the application of the second-order analysis method in pile-supported framed structures, it is essential to develop an efficient numerical alternative capable of comprehensively capturing the SPBI.

To address this limitation, this thesis introduces a refined three-dimensional pile element (3DPE) that embeds depth-dependent nonlinear soil resistance directly within the element formulation, thereby modeling continuous SPI without resorting to discrete soil springs. This refinement enables accurate modeling of nonlinear SPI with a coarse mesh, substantially reducing computational cost. It also mitigates the numerical instabilities that occur in existing line-element formulations when dealing with complex SPI. Nevertheless, in complex geological settings (e.g., multilayered soils), high

accuracy may still require mesh refinement. To realize one-element-per-pile modeling under such conditions, this thesis further investigates machine-learning (ML) approaches.

ML techniques have gained popularity among various numerical alternatives, largely due to recent significant advancements in artificial intelligence technology. In the realm of SPI, ML models, such as the neural networks (NN), can learn to predict structural responses during the training process, enabling them to forecast the behavior of systems under specific inputs using nearly zero computational cost after training. This capability makes them ideally suited for computationally demanding tasks like uncertainty evaluation. While the rapid prediction capabilities of these ML models offer considerable potential, the implementation processes for pile-supported framed structures considering SPBI can sometimes be challenging, thereby impeding their practical applications.

In conventional ML methods, the model training process is commonly data-driven, where the ML models learn to approximate the relationship between inputs and outputs using pre-collected datasets. To ensure model accuracy, the pre-collected training dataset must be both high-quality and large in quantity. While this data-driven training paradigm has proven effective for predicting the SPI of single piles, it is less suitable for the SPBI analysis of large-scale pile-supported systems, since both the cost for each datapoint and the required dataset quantity grows with the scale of the modelled systems. To address these challenges, a new ML training paradigm called physics-informed neural networks (PINN) has been proposed recently. PINN eliminates the need for extensive datasets by incorporating the residuals of governing equations directly into the loss function, thus guiding model training on physical laws rather than pre-collected data. Although PINN offers a promising data-free training approach, their application to pile-supported structures containing multiple piles can still pose challenges, which arise from the need to include numerous terms in the loss function to comprehensively

describe the systems being modeled. Moreover, large-scale structural systems, like pile-supported structures, are commonly project-specific. Additionally, large-scale structural models (like entire buildings with piles) tend to be project-specific. An ML model trained on one such structure cannot be directly reused on another; typically it would need retraining. This lack of reusability greatly limits the practical usefulness of current ML approaches for SPBI.

In summary, both the LFEM and ML techniques present distinct advantages and challenges for the structural analysis of pile-supported framed structures. LFEM, renowned for its robustness for large-scale structural analysis, is the preferred method in current structural design practices. However, it necessitates a fine element mesh for accurate modelling of the complex SPI, rendering the analysis process tedious. Conversely, ML offers a promising mesh-free approach for structural analysis with excellent efficiency. Yet, the extensive dataset requirements for training on large-scale structural systems and the limited reusability of trained models hinder their widespread adoption in engineering practices. To promote the second-order analysis method to pile-supported framed structures, there is an urgent need to develop a new computational framework to synergistically combine the unique advantages of LFEM and ML while avoiding their disadvantages for efficient simulation of large-scale SPBI analysis.

This thesis proposes a hybrid numerical framework that synergistically integrates an emerging ML technique, operator learning, with the conventional Finite Element Method (FEM) for efficiently solving general partial differential equation (PDE) problems. In this proposed approach, the trained neural operator is not directly applied to predict PDE solutions; rather, it is employed to formulate finite elements defined on specific subdomains that exhibit complex geometries or material properties. These elements, referred to as Neural Operator Elements (NOEs), enable accurate recovery of solutions within these subdomains using very coarse meshes without compromising accuracy. Additionally, the Neural Operator Element Method (NOEM) allows the

trained neural operator to be reused across different problems without retraining, as long as the new problem has a similar subdomain. This greatly improves model reusability compared to conventional project-specific ML models.

The accuracy of NOEM heavily depends on the performance of the neural operator. In the emerging field of AI for scientific computing, encompassing both PINN and operator learning, the sampling method during the training phase significantly influences the performance of the trained ML model. Consequently, substantial research has been dedicated to developing appropriate sampling strategies in PINN. Nevertheless, existing sampling methods have proven ineffective or inadequate for operator learning due to the inherently high-dimensional characteristics of input parameterization. To address this challenge, a novel sampling technique named Residual-based Adversarial-gradient Moving Sample (RAMS) is proposed specifically for ML models solving PDE problems. Extensive numerical investigations demonstrate that RAMS is compatible with a broad spectrum of existing sampling methods. Furthermore, RAMS exhibits consistently stable performance across diverse tasks, including PINN for function learning, PI operator learning, and data-driven operator learning, making it the first sampling method capable of effectively addressing operator learning tasks.

To implement the NOEM and RAMS for pile-supported framed structures, the governing equations must accurately reflect their structural responses. Although certain studies have investigated pile foundations by considering complex SPI, these typically neglect the geometric nonlinearity of piles. Such an omission can lead to significant errors, particularly for slender piles experiencing large deflections, consequently causing considerable underestimation of the deformation of pile-supported buildings when further accounting for PBI and P-Δ effects in superstructures. Hence, the governing equations that accurately incorporate both complex SPI and geometric nonlinearities are derived in this thesis. These governing equations are subsequently

integrated into the PINN framework, providing an efficient numerical alternative for analyzing single piles.

These governing equations are incorporated into the NOEM for simulating large-scale pile-supported structures, where RAMS is also utilized to enhance neural operator training. Within the NOEM framework, a single NOE formulated using the trained neural operator can efficiently represent an individual pile. This significantly simplifies the modeling process and substantially improves computational efficiency compared to conventional LFEM when analyzing SPBI. Consequently, the proposed hybrid numerical approach is expected to facilitate the broader adoption of second-order analysis methods for pile-supported framed structures by providing a more efficient and effective numerical solution.

1.2 Objectives

The thesis develops a hybrid numerical framework that integrates ML techniques with FEM to enable efficient second-order analysis of pile-supported framed structures. First, an efficient line finite element formulation, the refined 3DPE, is derived to accurately capture complex SPI along piles. Although this element alone cannot model pile-supported building structures with one-element-per-pile, it potentially offers a fast and accurate data generator for subsequent ML models trained on single-pile behavior. Next, the NOEM is developed for general PDE problems and then implemented to capture complex SPBI of large-scale pile-supported framed structures with high accuracy and efficiency. To enhance NOEM's effectiveness, a novel sampling method, RAMS, is introduced to improve the training of neural operators. Furthermore, governing equations that account for pile geometric nonlinearity and complex SPI are derived and implemented within this hybrid framework, providing an effective numerical alternative to conventional methods.

The research roadmap of this thesis is visualized in Figure 1-2, where the research objectives can be summarized as follows:

- **Develop a refined line-element formulation for efficient SPI modeling.** The refined 3D pile element (3DPE) accurately captures nonlinear SPI on coarse meshes, providing a robust analysis tool for pile-supported frames and serving as an efficient data generator for ML models of single piles.
- **Propose a hybrid numerical framework that integrates the emerging operator learning technique with conventional FEM:** In the proposed NOEM, the neural operators are trained and subsequently employed to formulate finite elements for subdomains with complex geometry or varying properties, allowing accurate simulations with significantly coarser mesh discretization.
- **Develop a novel sampling method for effective ML model training:** Recognizing the inadequacy of existing sampling methods in addressing high-dimensional input parameterization, RAMS is designed to be compatible with various existing sampling approaches and demonstrates stable performance across multiple tasks.
- **Derive accurate governing equations for single piles:** The governing equations accounting for the complex SPI and the geometry nonlinearity of slender piles under large deflections are derived, thus ensuring accurate assessment of pile-supported structural responses.
- **Implement the NOEM framework for pile-supported structures**: The governing equations derived for single piles are integrated within the NOEM and the RAMS to analysis the SPBI of large-scale pile-supported structures.
- **Extend the second-order analysis methods for pile-supported framed structures**: By providing an efficient, accurate, and computationally feasible numerical alternative, this thesis aims to overcome existing computational limitations, thereby promoting advanced simulation-based structural design practices to pile-supported framed structures.

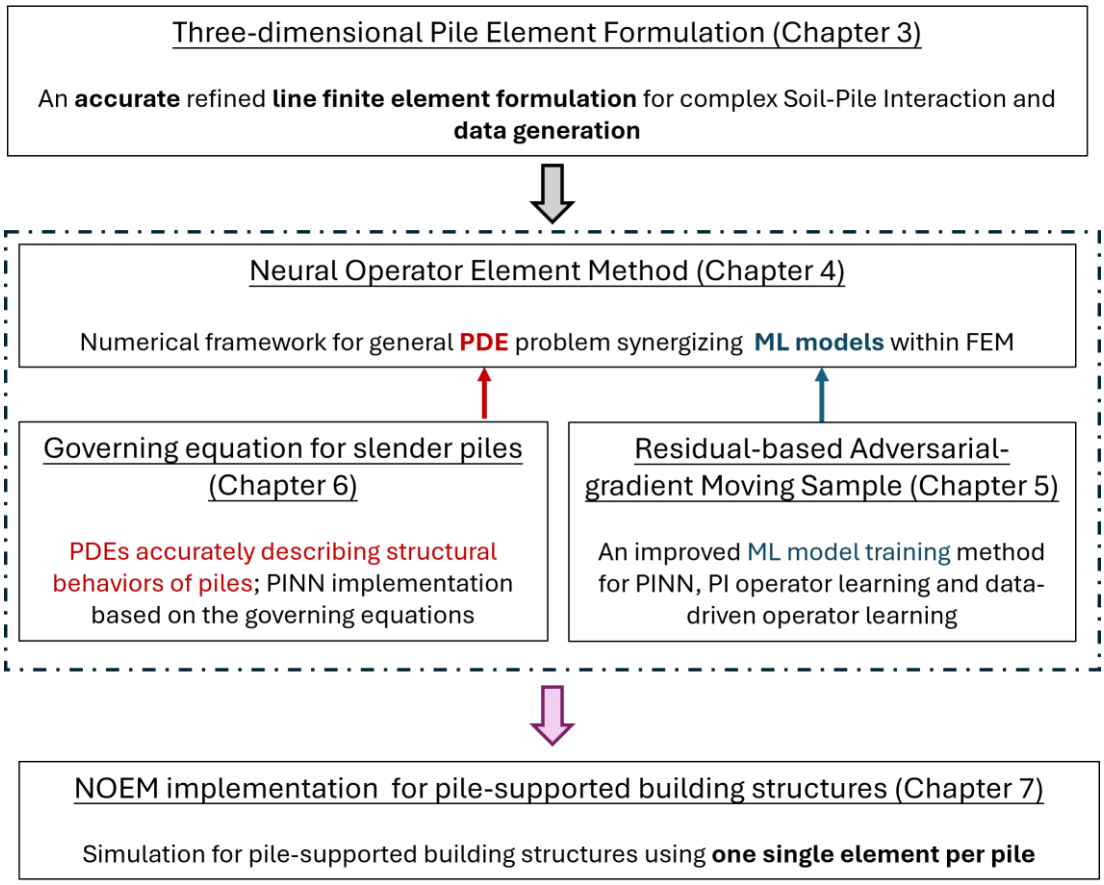


Figure 1-2 Research roadmap of the thesis.

1.3 Outline of the Thesis

This thesis consists of eight chapters. The overall structure is presented as follows.

Chapter 1 introduces the thesis background, emphasizing the urgent need for an efficient simulation method to extend second-order analysis to pile-supported framed structures by accurately capturing the SPBI. The research objectives and the thesis outline are also provided in this chapter.

Chapter 2 reviews previous studies on numerical methods for general PDE problems and structural analysis. It begins with an overview of the general FEM and the LFEM specifically developed for structural applications. Subsequently, advanced ML techniques for numerical analysis are discussed. The chapter concludes with a review of recent developments in hybrid numerical methods that integrate ML with classical approaches.

Chapter 3 presents an efficient line-finite-element formulation, the refined 3DPE, which embeds nonlinear SPI directly into the element formulation. By eliminating discrete spring elements, the 3DPE enables accurate SPI modeling on coarse meshes and improves computational efficiency.

Chapter 4 proposes a hybrid numerical simulation framework termed NOEM, which synergizes machine learning techniques with FEM for general PDE problems. In this framework, trained neural operators are embedded within the variational formulation to model subdomains with complex geometry and material properties using a single element, thereby significantly enhancing computational efficiency.

Chapter 5 introduces a novel sampling method, RAMS, designed to improve the training efficiency of machine learning models solving scientific computing problems. In RAMS, samples, whether collocation points in PINN or function samples in operator learning, are treated as trainable parameters optimized via gradient-based methods. By maximizing PDE residuals on these samples, RAMS identifies undertrained regions and adaptively reallocates samples, leading to more effective training with the same sample size.

Chapter 6 derives the governing equations for slender piles undergoing large deflections, incorporating both complex SPI and geometric nonlinearity to accurately describe the structural behavior of single piles. These equations are then implemented within the PINN framework, providing an efficient numerical alternative for SPI analysis for single piles.

Chapter 7 integrates the governing equations derived for single piles into the NOEM framework to simulate large-scale pile-supported structures. RAMS is also employed to facilitate effective training of the neural operators modeling individual piles. The resulting NOEM enables accurate simulation of SPBI using only a single element per pile, thereby advancing second-order analysis of pile-supported structures

through an efficient numerical solution.

Chapter 8 summarizes the key findings and conclusions of this thesis, highlighting the significance of the research and providing recommendations for future work.

Chapter 2. LITERATURE REVIEW

2.1 Introduction

This chapter provides a comprehensive review of various numerical methods and their applications in structural analysis. The finite element method (FEM), including both its weak and variational forms, is revisited. Building upon this foundation, the line finite element method (LFEM), the most employed numerical approach in structural engineering, is introduced with the second-order analysis method. Accordingly, the implementation of the current LFEM for the second-order analysis of pile-supported framed structures is discussed. Then, recent advancements in machine learning (ML), specifically within the realm of AI for scientific computing (AI4SCI), are examined, where the physics-informed neural networks (PINN) and operator learning techniques are highlighted and discussed in detail.

2.2 Finite Element Method

Since its introduction in the early 1960s, the FEM has become a cornerstone of scientific computing [1-3]. It is widely preferred in engineering simulation because of its exceptional flexibility in handling complex geometries and nonlinear material properties, enabling high-fidelity modelling of real-world structures that often possess irregular shapes. Starting from weak-form formulations [2], extensive research over the past decades has produced numerous variants, including the least-squares FEM [4-6], the extended FEM (XFEM) [7, 8], the discontinuous Galerkin methods [9, 10], and so on. Among these variants, the variational FEM based on the principle of minimum potential energy remains the most commonly used in structural engineering practice [11, 12].

In variational FEM, solving the governing equations is recast as an optimization of the energy functional (also termed as the total potential energy). For a solid mechanics problem, the computational domain is discretized into finite elements, and the global optimum of the energy functional over the trial (shape) function space gives

the solution

$$\boldsymbol{u} \coloneqq \arg\min_{\boldsymbol{w} \in U_h} J[\boldsymbol{w}], \tag{2.1}$$

where $\boldsymbol{u}$ may be either a scalar-valued or a vector-valued function; U_h is the trial (shape) function space commonly spanned by piecewise-polynomial shape functions determined by the meshes; and J is the (total potential) energy functional.

In structural engineering applications, components or whole systems are often discretized into shell or solid elements for precise modeling [13-15]. Although this strategy yields accurate solutions for complex structural behaviors, e.g., in local-buckling analysis of steel members [16] or in Soil-Pile Interaction (SPI) studies [17, 18], it can become impractical for large-scale buildings as the associated computational cost is prohibitive. Consequently, current second-order analyses typically employ the line finite element method (LFEM) [11, 12].

2.2.1 Line finite element method

LFEM represents members by one-dimensional line elements under the planar-section assumption [15] rather than by shell or solid meshes. Beams and columns that would otherwise require hundreds of shell/solid elements can therefore be modeled with only several line elements, greatly reducing both modelling effort and computation time. The earliest line element is the linear truss element, which uses a first-order polynomial and is suitable only for axial-force members as it neglects lateral deflection and rotation. To extend LFEM, extensive research has been devoted to improving both accuracy and efficiency.

The development of LFEM has evolved significantly over the years, with substantial contributions made to improve accuracy and efficiency in structural analysis. The Hermite cubic element was an important milestone in extending the analysis capabilities of LFEM. The Hermite cubic element, initially developed by Connor Jr et

al. [19] and later refined by Bathe and Bolourchi [20], employs third-order polynomials to represent lateral deflection and rotation along the member axis. Such shape functions for representing displacement allow for a relatively accurate representation of beam deformations under different loading conditions. However, studies have shown that this element can produce large errors for compression-dominated members when a coarse mesh is used [21, 22]. This limitation spurred the development of alternative formulations aimed at improving performance under compression and nonlinear conditions.

In response to these challenges, the stability-function was introduced by Oran [23] within the element formulation to enhance the element modeling capability of compression-dominated structures. By embedding the $P-\delta$ effect through the stability function, this element can accurately capture geometric nonlinearity for structures experiencing large deformations. Furthermore, the stability-function element has demonstrated high accuracy in nonlinear stability analysis, as demonstrated by various studies [24-27]. This improvement has made it particularly useful in the analysis of buckling problems, where traditional elements fail to deliver satisfactory results under high compressive loads.

Another significant development in element formulation for nonlinear analysis is the force-based (flexibility-based) element formulation [28, 29]. These force-based element formulations differ from traditional displacement-based formulations by representing internal force distributions rather than member deformations. This formulation is advantageous because it simplifies the representation of the structure's internal force state, making it highly efficient in modeling the behavior of complex structures under various loadings. The force-based elements are particularly suited for scenarios where the distribution of internal forces is more straightforward to describe than the deformations themselves. Zhang and Tien [30] further refined this approach by incorporating advanced numerical techniques to improve the accuracy and

computational efficiency of force-based formulations. However, despite their efficiency in modeling internal forces, force-based elements present challenges in recovering deformations. This requires additional numerical integration, which complicates their practical implementation in large-scale problems. Thus, while force-based elements offer significant improvements in modeling efficiency, they come with a trade-off in terms of the additional computational complexity needed to recover displacement fields.

The above element formulations are generally built on the local coordinates, which might lead to significant errors when considering the structural system undergoing large deformation. Thus, one essential technique in the LFEM to capture such geometric nonlinearity is the kinematic description. Several commonly used kinematic descriptions are reviewed, including the Total-Lagrangian (TL) and Updated-Lagrangian (UL) formulations, and the co-rotational method.

The TL formulation assumes that all material and kinematic variables are referenced to the initial, undeformed configuration. Thie method is effective in situations where the structure undergoes significant changes, especially in the context of nonlinear static and dynamic analyses. However, it has limitations when dealing with very large rotations due to its reliance on the initial configuration, which can complicate the analysis under large rotation [31].

The UL formulation updates the reference configuration at each iteration, referencing the current deformed configuration. This approach is more suitable for cases where the structure undergoes moderate deformations. It simplifies the process by using updated geometries at each step, offering better computational efficiency when the deformations are not excessively large. The UL formulation is widely employed in static and dynamic analyses where the structure's geometry is frequently updated, as seen in applications like post-buckling analysis [32, 33].

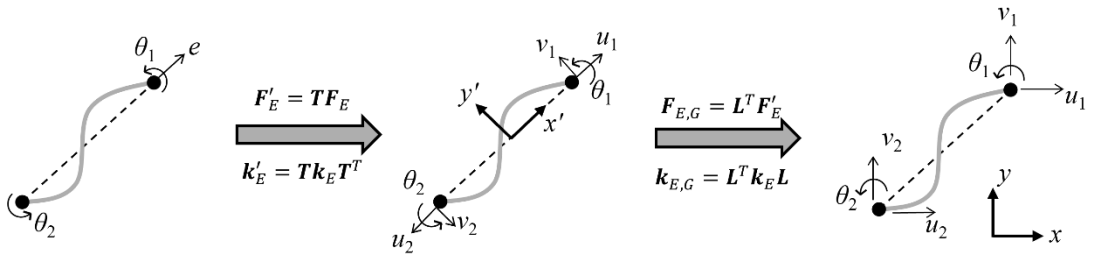


Figure 2-1 Illustration of the co-rotational method.

The co-rotational method introduces a more refined approach by separating rigid body rotations from the deformations. This method uses a set of orthogonal axes attached to the element's ends, which rotate with the element's deformations. The rotations relative to the chord are then calculated based on these axes (Figure 2-1), eliminating the limitations associated with vectorial rotations in large deformations. This formulation is particularly effective for analyzing structures undergoing large rotations, as it accounts for rotational changes directly within the element's local coordinate system [34, 35]. The co-rotational approach improves upon traditional formulations by allowing for large rotations without the loss of accuracy, making it ideal for analyzing structures with complex geometries subjected to significant rotational displacements.

Even though the kinematic description provides an effective tool for capturing the geometric nonlinearity of structural members undergoing large deflection. Considering such nonlinearity also results in solving Eq. (2.1) become complicated. Extensive study has been undertaken for different numerical methods to search for the solution. These methods include the incremental method, Newton-Raphson method, the displacement control method, the arc-length method, and the minimum residual displacement method, each offering distinct advantages depending on the type of problem.

The incremental method is one of the earliest numerical methods developed for nonlinear analysis. It transforms a nonlinear problem into a series of smaller linear

problems by applying small load increments at each step. This method is convenient for implementation in practice. However, it suffers from the drawback of not providing a clear error estimate, especially in highly nonlinear scenarios where larger load increments may cause significant discrepancies in the results. This method also requires the increment to be sufficiently small, thereby sometimes resulting in substantial computational resources for large-scale structural systems undergoing large deformation.

The Newton-Raphson method is a widely used iterative method that improves accuracy for large deformation method compared to the incremental methods. It works by solving a system of nonlinear equations using linear approximations in each step, updating the tangent stiffness matrix with each iteration. This method is robust and converges faster than the pure incremental method and allows a larger increment at each step, making it a preferred choice in practical applications. Due to these advantages, different versions of modified Newton-Raphson method [36, 37] are continuously developed to improve its computational efficiency. However, it encounters difficulties in situations involving limit points or post-buckling analysis due to the potential instability caused by the ill-conditioned tangent stiffness matrix.

The displacement control method, introduced by Batoz and Dhatt [38], is designed to address the limitations of the Newton-Raphson method, particularly in cases involving negative stiffness matrices or snap-back problems. This method works by controlling the displacement rather than the load. In each iteration, a specific degree of freedom is chosen to control displacement, and the corresponding load increments are adjusted to maintain equilibrium. This approach is particularly useful for problems involving buckling and large deformations where the Newton-Raphson method struggles to predict accurate results. Despite its advantages, the method can diverge in snap-back problems if the appropriate degree of freedom is not carefully chosen. Accordingly, different “path-following” algorithms are further developed.

The arc-length method, introduced by Riks [39], is a powerful technique used to solve nonlinear problems, especially those involving snap-through or post-buckling behavior. It modifies the load increment based on the path of the solution in the load-displacement space, ensuring that the solution follows the equilibrium path even when the structure undergoes significant deformations. This method allows for the analysis of multiple bifurcation points and the determination of equilibrium paths that the Newton-Raphson method cannot capture. Several variations of arc-length methods, such as spherical, cylindrical, elliptical, and linearized forms, have been developed for the nonlinear structural analysis [40-42]. Among these, the cylindrical arc-length method has become particularly popular due to its superior performance in comparison to other arc-length variants [43, 44]. Nevertheless, challenges remain in efficiently tracing equilibrium paths when using various arc-length formulations. Consequently, substantial research has focused on enhancing numerical stability and resolving issues related to limit points encountered with arc-length techniques [43, 45]. Alongside arc-length methods, alternative strategies, such as the minimum residual displacement [46], have also been proposed to address these computational difficulties.

The minimum residual displacement method, introduced by Chan [46], is designed to minimize the residual displacements during each iteration, offering a more efficient approach to convergence. This method focuses on minimizing the error in displacement residuals, which helps in handling issues like snap-through problems, where other methods might fail. By introducing a load correction factor, the minimum residual displacement method ensures that the solution follows the shortest path to convergence. It has been found to be highly effective in nonlinear analysis, particularly for structures with complex loading conditions and multiple bifurcation points [44, 47].

2.2.2 Implementation for second-order analysis method

The LFEM offers a powerful framework for second-order analysis, a codified simulation-based design philosophy endorsed by Eurocode 3 [48], AISC-LRFD [49],

and Code of Practice for the Structural Use of Steel in Hong Kong [50]. Yet, most conventional LFEM formulations are not directly applicable to practical structural design. For instance, current design codes require the consideration of member initial imperfections, deformations of structural members that exist before external loads are applied, but the conventional "straight" line element cannot represent these curvatures within its shape functions [51]. Consequently, each structural member must be subdivided into multiple elements to model initial curvature, substantially complicating modelling and increasing computational cost. To make LFEM feasible in practical second-order analysis, more advanced line-element formulations that capture the full structural response with a single element per member are essential.

For accurately modelling real structural members using one element per member in the LFEM, Chan and Zhou [52] proposed the pointwise equilibrating polynomial (PEP) element, which enforces compatibility at the end nodes and equilibrium at mid-span, allowing a fifth-order displacement polynomial and yielding superior accuracy and efficiency. Then, this PEP element formulation is then enriched to capture key practical effects. It was soon extended to incorporate initial imperfections directly into the shape functions [53], allowing initially bent members to be modelled efficiently with one element per member.

Materials in structural engineering often exhibit significant nonlinear behavior, especially under large deflections. As such, the inelastic analysis method is crucial for the practical implementation of the second-order analysis and has been extensively studied over the past decades. To consider the material nonlinearity, one commonly-used analysis method in practice is the plastic-hinge method (PHM) [54-57], which models plastic zones as discrete rotational springs that activate when bending moments exceed the yield limit. To capture material nonlinearity within a single element, Chen and Chan [58] introduced an internal spring as the plastic hinge at mid-span by combining two Hermite elements. Zhou and Chan [59] later generalized the PEP

formulation to place a plastic hinge at any location along the member axis. More recently, Liu et al. [60, 61] proposed the arbitrarily-located plastic-hinge (ALH) element, which partitions the shape functions with two dependent polynomials at the hinge, thereby increasing analysis accuracy by providing a more sophisticated shape function.

Moreover, research has been undertaken to account for other important structural features, such as warping effects in thin-walled members [62-64] and Wagner effects in members with asymmetric cross-sections [32, 63, 65]. These advanced line element formulations commonly demand more detailed cross-section information, such as the yield surface and the Wagner coefficients, compared to the conventional LFEM. Thus, extensive study has been carried out to develop different cross-section analysis techniques [66-69]. One notable recent advancement in the cross-section analysis is MSASect2 [67], a software provides comprehensive sectional properties for arbitrary non-symmetric cross-section.

All these developments synergistically enhance LFEM to precisely simulate building structures with one element per member, thereby facilitating the extensive application of the second-order analysis method for the design of upper structures [11, 28, 58, 70-76].

2.3 Machine Learning for Structural Analysis

2.3.1 Physics-informed machine learning

Extensive research has been conducted to develop more effective numerical alternatives using ML methods for scientific computing and engineering simulations, thereby promptly revolutionizing the structural analysis. These ML techniques offer efficient and accurate predictions after sufficient training. Conventional ML approaches typically follow a data-driven training paradigm, where model performance is evaluated through a loss function based on pre-collected datasets [77-79]. To achieve optimal performance, datasets must be of high quality and sufficient quantity. However,

acquiring such datasets through field experiments or high-fidelity simulations can be prohibitively expensive, limiting the practical application of data-driven ML methods.

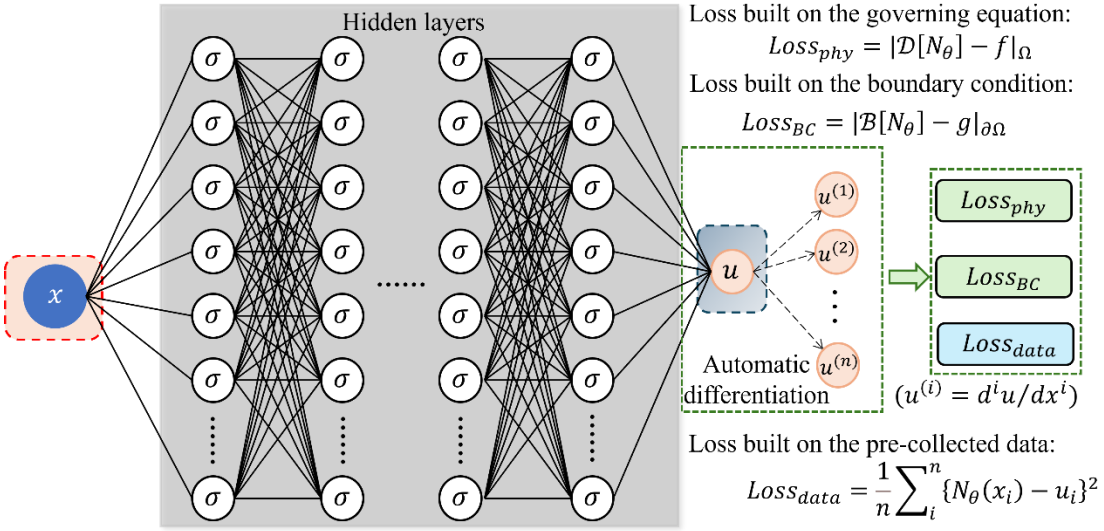


Figure 2-2 Physics-informed neural networks using MLP.

In 2019, Raissi et al. [80] introduced a data-free ML paradigm for scientific computing (Figure 2-2), known as PINN, designed to solve PDE problems. PINN integrates physical laws, typically PDEs, directly into the training loss function. Consider a PDE given by:

$$\mathcal{D}[u] = f, \quad \text{for } x \in \Omega,$$

$$\mathcal{B}[u] = g, \quad \text{for } x \in \partial\Omega$$

The corresponding loss function in PINN can be expressed as:

$$\mathcal{L}(\theta) = |\mathcal{D}[N_\theta] - f|_\Omega + |\mathcal{B}[N_\theta] - g|_{\partial\Omega} + \frac{1}{n}\sum_i^n \{N_\theta(x_i) - u_i\}^2, \tag{2.2}$$

where the first two terms and the last term represents the physics-informed loss and the

data-driven loss, respectively; $|\cdot|_{\Omega}$ and $|\cdot|_{\partial\Omega}$ denote the chosen norms on the domain Ω and boundary $\partial\Omega$, respectively; and $\{x_i \quad u_i\}_{i=1}^{n}$ represents the pre-collected dataset.

Unlike traditional data-driven methods, PINN leverages governing equations to guide learning, enabling accurate predictions even with limited or noisy data [81-83]. This physics-based approach ensures consistency with known physical laws and enhances generalization to unseen scenarios, alleviating concerns related to the "black-box" nature of conventional ML [84]. Since its inception, extensive research has advanced PINN, focusing primarily on improving model architecture, loss formulations, and sampling strategies to facilitate broader practical applications.

Model architecture

In the first article introducing PINN, Raissi et al. [80] utilized the multi-layer perceptron (MLP) to approximate solutions of PDEs. The differentiable nature of MLPs enables the straightforward formulation of physics-informed loss functions, and their universal approximation capability guaranteed by the universal approximation theorem [85] further ensures predictive performance. Consequently, MLPs have become the foundation in subsequent PINN studies [86]. To enhance the performance of PINN, Wang et al. [87] proposed modifications to the conventional MLP architecture by explicitly incorporating multiplicative interactions between input dimensions and employing residual connections within hidden layers. These modifications have been demonstrated numerically to surpass the conventional MLP in performance. Additionally, Jagtap et al. [88, 89] enhanced MLP flexibility by introducing trainable parameters within activation functions, facilitating quicker convergence during training. Some studies further decomposed the weight matrix into directional expressions and magnitude parameters [90, 91], highlighting through numerical tests and theoretical analyses that such decomposition significantly enhances model performance.

In addition to the traditional MLP architecture, various classical neural networks models have been employed in PINN, leveraging their intrinsic strengths. Convolutional Neural Networks (CNN) [92, 93], renowned for capturing translation invariance, have been introduced into PINN studies to incorporate essential symmetries for specific tasks. For instance, Gao et al. [94] proposed PhyGeoNet, integrating CNN into PINN to effectively model irregular geometries within computational domains. Furthermore, sequential models such as Recurrent Neural Networks (RNN) [95] and Long Short-Term Memory (LSTM) networks [96, 97] have also been adopted in PINN for their proficiency in managing sequential data. For example, Zhang et al. [96] employed LSTM within PINN for path-dependent inelastic analysis of building structures. Beyond these networks-based models above, PINN also incorporates other conventional ML models, such as Gaussian processes [98-100] and kernel methods [101, 102].

Recently, Kolmogorov-Arnold Networks (KAN) have emerged as a notable model in AI4SCI. Unlike traditional neural networks, KAN represent functions by parameterizing activation functions based on the Kolmogorov-Arnold representation theorem [103] extended to deep networks, rather than relying on linear maps as trainable parameters. Liu et al. [104] demonstrated that KAN can effectively represent complex functions using fewer parameters compared to MLPs, without sacrificing approximation capability. Despite identified training instabilities, KANs continue to attract considerable research interest due to their promising potential in PINN applications [105-107].

In PINN, boundary conditions are commonly incorporated through penalty terms in the loss function, a method often referred to as a soft constraint. However, predictions may occasionally violate boundary conditions under this approach. To address this, explicit boundary condition enforcement (hard constraints) has been investigated, embedding boundary conditions directly within function approximations. Hard

constraints have been numerically shown to alleviate training difficulties and enhance model accuracy [87, 108]. Additionally, theoretical analyses, such as [109], have indicated that hard constraints yield lower error estimates compared to soft constraints. Research on enforcing various boundary conditions, such Dirichlet boundary condition [110, 111] and periodic boundary condition [108, 112], continues actively.

Moreover, extensive studies have proposed architectural modifications through feature expansion to address inherent limitations of traditional PINN, such as difficulty learning high-frequency functions due to spectral bias [113]. To counteract this, researchers have enhanced baseline model formulations by augmenting input representations—originally limited to spatial or spatio-temporal coordinates—with additional features. Numerous expansion techniques have been explored, with specific choices contingent upon problem characteristics. For example, Wang et al. [114] demonstrated that integrating random Fourier features mitigates spectral bias. Other expansions incorporating polynomial and exponential terms have also been investigated, showcasing their potential in enhancing PINN performance for complex applications [115-117].

Loss formulation

Loss function incorporating the physical laws stands as an important part in the PINN. Starting from the original PINN applied from some normal PDE problems, continuous efforts have been actively undertaken for more challenging PDE problems, such as fractional differential equations (FDE), stochastic differentia equation (SDE), and integral differential equation (IDE), via modifying loss formulation. For instance, fractional differential terms cannot be trivially computed by the automatic differentiation (AD) [118]. Thus, Pang et al. [119] develop fractional PINN (fPINN) via combining AD with numerical discretization for the fractional operators. To consider the stochasticity, Yang et al. [82] reformulate the physics-informed loss into the posterior form that calculated via the Hamiltonian Monte Carlo and the variational

inference. For the IDE problem, Yuan et al. [120] developed an A-PINN, where additional terms approximating the integral terms, to use additional auxiliary differential term to replace the numerical quadrature, which has been shown numerically in significantly improving training efficiency. Such algorithm that introducing additional auxiliary terms also shows great improvement in some high-order PDE problems, where the high-order term can be approximated by several low-order differential term for better training efficiency and stability [111].

Beyond solving broader types of PDE problems, recent studies have emphasized that refining the loss function through strategic weighting of individual loss terms significantly enhances the efficiency and performance of PINN. The weighting strategies can be classified as fixed, dynamic, or sequential. The original PINN formulation employed fixed weights, predetermined before training. However, subsequent advancements introduced dynamic weighting strategies which facilitate adaptation during the training process. For instance, Wang et al. [87] developed a learning rate annealing algorithm that dynamically modifies global weights based on the back-propagation gradients and has been employed in some practical applications [121]. Similarly, self-adaptive loss-balancing techniques, such as the method proposed by [122], adjust global weights iteratively throughout training, enhancing the model performance. Moreover, sequential training approaches incorporate iteration-based weight adjustments. For example, Wang et al. [112] introduced a causal parameter for time-dependent problems, enforcing sequential learning by progressively adjusting weights to capture the inherent causality across time steps. Wang et al. [123] include leveraging the Neural Tangent Kernel (NTK) framework within PINN. Specifically, the adaptive global weighting strategy extends the NTK methodology to PINN, substantially improving both trainability and predictive accuracy.

Sampling strategy

Estimating the norms of PDE residuals in physics-informed loss functions

typically involves numerical quadrature at selected collocation points. The choice of the number of collocation points is crucial; while using more points can improve model accuracy, it significantly increases computational requirements for both data sampling and model training. Conversely, using too few points may adversely affect model performance.

To address this challenge, recent research has focused on developing efficient sampling and resampling methods that minimize the number of required points without sacrificing accuracy. One prominent method is the Residual-based Adaptive Refinement (RAR) approach [124], which strategically identifies and refines collocation points in regions exhibiting large PDE residuals, signaling insufficient training coverage.

Building upon RAR, several enhanced resampling strategies have been introduced, including RAR-D [125] and R3, [126] both aiming to concentrate on sampling near areas with notably high residuals. Beyond these adaptive resampling techniques, importance-sampling (IS)-based methods have also been developed [127-131]. These methods sample collocation points according to a probability distribution aligned with the magnitude of physics residuals to directly minimize the generalization error. Despite differing implementation details, IS-based and RAR-type methods share a common approach by preferentially sampling regions where physics residuals are largest, thus locating area inadequately trained.

2.3.2 Operator learning

Operator learning addresses the challenge of learning mappings between infinite-dimensional function spaces, such as those encountered in PDEs. Operator learning offers an approximation alternative that can generalize across different parameterizations of functions, providing efficient solutions to a broad class of PDEs.

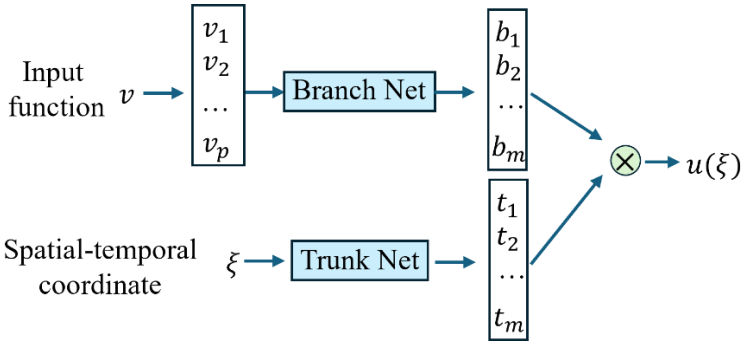


Figure 2-3 Schematics of DeepONet architecture.

The Deep Operator Network (DeepONet), which is the first NO architeture, was introduced by Lu et al. [132] based on the universal operator approximation theorem [133] as a framework to learn nonlinear operators. It consists of two neural networks (Figure 2-3): a branch net that encodes the input function and a trunk net that decodes the output at specific locations. This architecture allows DeepONet to approximate a wide range of operators. Then, Wang et al. [134] extended the PINN to DeepONet for the data-free operator learning. Subsequent extensions of DeepONet have enhanced its capabilities. The multiple-input DeepONet allows for the incorporation of multiple input functions, expanding its applicability to more complex scenarios. The Diffeomorphic Mapping Operator Network (DIMON) [135] introduces diffeomorphic transformations to better capture the underlying geometry of the operator mappings, improving accuracy in applications involving complex domain geometries. Separable DeepONet [136] enhances the scalability of physics-informed DeepONet for high-dimensional PDEs by factorizing networks across input dimensions and using forward-mode differentiation, achieving comparable or better accuracy with significantly reduced computational cost.

The Fourier Neural Operator (FNO), proposed by Li et al. [137], also advanced operator learning by directly parameterizing the integral kernel in Fourier space. This approach enables efficient learning of mappings between function spaces, significantly

outperforming traditional neural networks in solving PDEs. Extensions of FNO have further improved its performance. The Factorized Fourier Neural Operator (F-FNO) [138] introduces separable spectral layers and improved residual connections, achieving superior accuracy and scalability in simulating PDEs. Additionally, the Gabor-filtered FNO [139] incorporates Gabor filters to enhance the model's ability to capture localized features in the input functions, leading to better performance in applications requiring high-resolution solutions. The Laplace Neural Operator (LNO) [140] replaces the Fourier transform in the FNO by the Laplace transform for improved interpretability and generalization, outperforming the original FNO in capturing transient responses and accurately solving various ODEs and PDEs.

Comparative studies between DeepONet and FNO have highlighted their respective strengths and limitations [141]. DeepONet excels in handling complex geometries and noisy data, maintaining robustness even in challenging scenarios. In contrast, FNO demonstrates great efficiency and accuracy, particularly in capturing low-frequency components of the operator mappings [142]. However, its performance may degrade in the presence of high-frequency noise or intricate geometries. To synergetic these two model architectures, hybrid models, such as Fourier-DeepONet [143, 144], have been proposed to leverage their unique strengths.

Operator learning has been increasingly integrated with other machine learning paradigms to address complex tasks. In active learning, operator learning models can guide the selection of informative data points, improving the efficiency of the learning process [145]. Additionally, the incorporation of operator learning into foundation models enables the development of general-purpose models capable of solving a wide range of tasks across different domains [146, 147].

Chapter 3. REFINED THREE-DIMENSIONAL PILE ELEMENT FORMULATION FOR PILE-SUPPORTED STRUCTURES

3.1 Introduction

The second-order analysis method is a codified simulation-based design method [51] widely recommended by specifications around the world, including AISC [148] and Eurocode 3 [149]. To implement the second-order analysis method in design practice, extensive study has been conducted for efficient and robust structural simulation methods, such as Line Finite Element Method (LFEM) [65, 150, 151] and Stability Function Method [54, 152, 153]. Despite the extensive use of this design method for upper structures, its application for pile-supported structures is limited due to the absence of appropriate analysis methods for piles. Consequently, in the second-order analysis method, the column bases in upper structures are commonly assumed to be directly fixed or pinned at the ground surface without pile foundations, which might result in unrealistic analysis results of structural deformation patterns and internal force distributions [154]. To extend the second-order analysis method to pile-supported structures, an efficient and reliable structural analysis method for piles is necessary.

One standard analysis method for piles is the Discrete Spring Element Method (DSEM), which simplifies the pile and the surrounding soil as the line finite element and a series of concentrated spring elements aligned at the nodes of the line elements. To facilitate the DSEM, different spring element constitutive models, including the *p-y* and *t-z* curves, are developed for describing the Soil-Pile Interactions (SPI) along piles embedded in different ground mediums [155-160] and recommended by specifications, such as API [155]. Based on these curves, the DSEM for different design scenarios are developed. For instance, Zhang et al. [161] developed the DSEM model to simulate piles with geometric imperfections by utilizing beam-column elements. Wang and Orense [162] presented an approach to model raked piles in liquefiable ground based on the DSEM. Limkatanyu et al. [163] introduced a fiber beam-column element to

capture the plastic hinges of piles accounting for nonlinear SPI. Nonetheless, the analytical methods mainly focused on the assessment of single piles and are seldom employed in the second-order analysis of pile-supported structures.

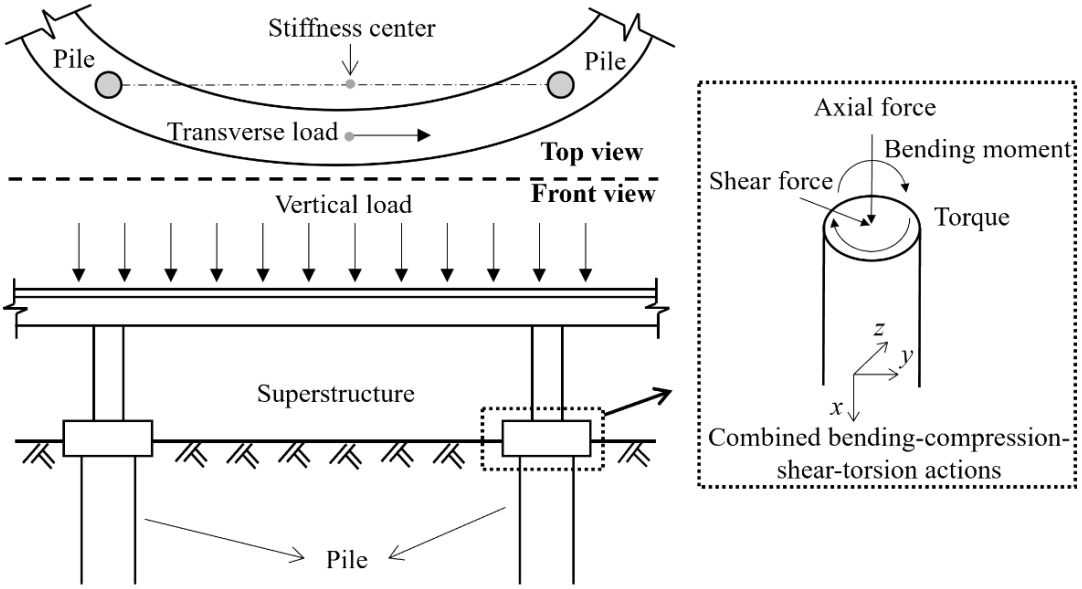


Figure 3-1 Loading condition on piles in pile-supported structures.

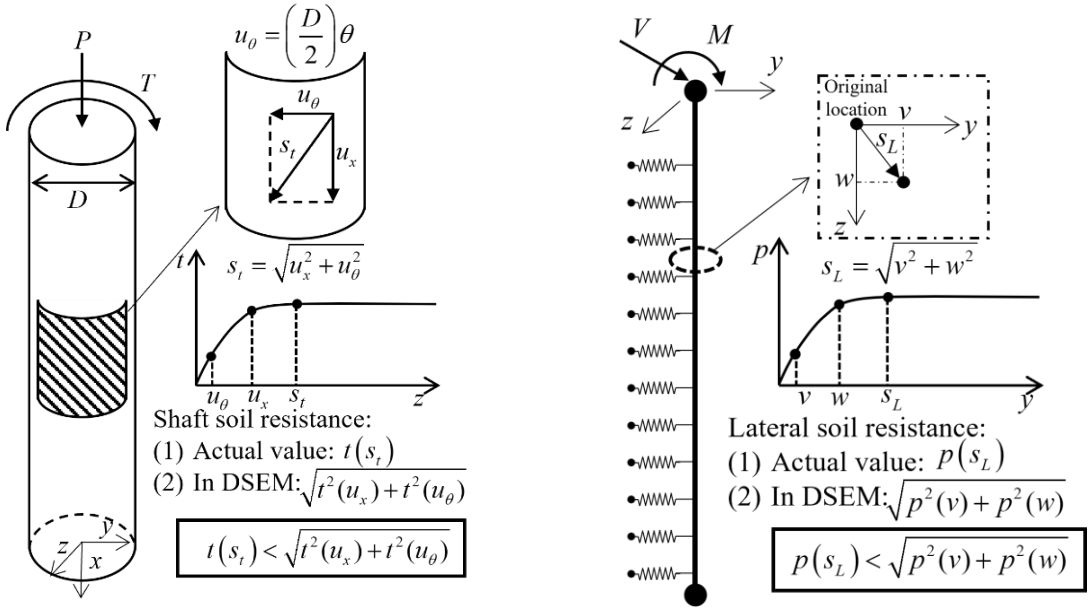
One of the reasons impeding the DSEM implementation in the second-order analysis of pile-supported structures is the direct combination of DSEM with the model of superstructure in the analysis of pile-supported structures is time consuming. For accurately modelling highly nonlinear SPI along piles in the DSEM, the distance between two adjacent discrete spring elements should be sufficiently small, sometimes less than 0.25 m [164, 165]. Incorporating a large numbers of elements in the modelling could lead to a significant increase in computational time and memory, especially for pile-supported structures containing multiple piles.

Moreover, the DSEM cannot accurately capture the structural behaviors of piles in the spatial analysis of pile-supported structures. Extensive previous studies have

pointed out that the load acted at the piles in pile-supported structures might be complex [166-172], such as piles in curved bridge-foundation systems that are subjected to combined bending-compression-shear-torsion actions [173], as shown in Figure 3-1. However, the SPI along piles under such complex loading conditions cannot be accurately captured by the DSEM since it generally simplifies the surrounding soil as the uncoupled spring elements in specified directions [174-177]. For example, the axial and torsional resistance are both provided by the shaft soil resistance. When analyzing a pile under axial and torsional forces in soil with nonlinear stiffness, the soil resistance is not the square root of the sum of the squares of the soil resistance obtained by the projected displacements in axial and torsional directions separately, as demonstrated in Figure 3-2A. Ignoring such axial-torsional coupled response in the DSEM might lead to significant overestimation of soil resistance and underestimation of pile displacement, sometimes even more than 20 % [176-178]. The problem is the same for calculating the lateral soil resistance shown in Figure 3-2B. Thus, an efficient element formulation for accurately capturing the structural behaviors of piles under complex loading conditions in the spatial analysis is necessary for extending the second-order analysis method for pile-supported structures.

To improve the efficiency of the analysis of piles accounting for nonlinear SPI, Liu et al. [179] recently proposed a new line element formulation named the Pile Element (PE). Instead of using the discrete spring element, the varied nonlinear SPI are directly considered in the PE formulation through Gaussian quadrature. This method has been applied to different design scenarios of piles [180-182]. Subsequently, Li et al. [183] preliminarily extended this research to consider the lateral soil resistance in the spatial analysis. However, when extending the PE to the spatial analysis, inappropriate linearization for the SPI was introduced, leading to the inaccurate element stiffness matrix, which further results in severe numerical instability in the spatial analysis of pile-supported structures. Moreover, the above element formulations do not fully consider all the crucial factors of piles under complex loading conditions.

Specifically, the existing PE formulation for the spatial analysis [183] is based on the Euler-Bernoulli theory, which ignores the shear deformation of piles [184]. Besides, the nonlinear SPI of piles in pile-supported structures, such as the pile subjected to combined bending-compression-shear-torsion actions, cannot be appropriately captured, resulting in underestimated deflections in the analysis results. Therefore, all the previous PE studies are only applicable to the analysis of single piles [179-183], hindering their practical applications for the second-order analysis of pile-supported structures.



(A) (B)

Figure 3-2 Soil-Pile Interactions of piles under bending-compression-shear-torsion actions: (A) Shaft soil resistance. (B) Lateral soil resistance.

In this chapter, a refined Three-Dimensional Pile Element (3DPE) formulation is developed for the second-order analysis of pile-supported structures. The employed element shape functions are formulated based on the Timoshenko theory [185] to consider the shear effect of piles. The nonlinear SPI in the spatial analysis is directly

incorporated into the element formulation through Gaussian quadrature. Compared with all the existing PEs, the proposed refined 3DPE can accurately capture the SPI along the piles under complex loading conditions. Moreover, three soil stiffness matrices, $\boldsymbol{k}_{p,p}$, $\boldsymbol{k}_{t,k}$, and $\boldsymbol{k}_{t,t}$, are derived in this chapter to overcome numerical instability when implementing the existing PEs to the spatial analysis for pile-supported structures. Therefore, the SPI along piles under complex loading conditions can be accurately and robustly captured. Extensive comparisons between different analysis approaches are provided to confirm the efficiency and accuracy of the refined 3DPE. The results demonstrate that the present study takes only about 30 % computational cost to generate reliable results compared with the DSEM when conducting the spatial analysis for pile-supported structures while avoiding the numerical instability resulted from the previous PEs. Furthermore, the DSEM may result in dangerous underestimation of the pile deflection for both single piles and piles in pile-supported structures, whose error sometimes grows larger than 15 %. Thus, the refined 3DPE is believed to be a useful contribution to extending the second-order analysis to pile-supported structures in design practice.

3.2 Refined Three-Dimensional Pile Element Formulation

3.2.1 Assumptions

The following assumptions are adopted: (1) the Timoshenko beam theory is used; (2) the pile cross-section is circular; (3) strains are small, while nodal displacements and rotations can be moderately large; (4) the pile material remains elastic; (5) the external forces are conservative; and (6) the warping deformation of the pile is insignificant and can be ignored.

3.2.2 Shape functions

The element deformation can be described by submitting the nodal displacement vector into the element shape functions. As reported by Bazoune et al. [186], the shape functions developed based on the Timoshenko beam theory can be expressed as:

$$u(x) = \boldsymbol{\alpha}_x(x)^T \boldsymbol{u}_x$$

$$v(x) = \boldsymbol{\alpha}_y(x)^T \boldsymbol{u}_y$$

$$w(x) = \boldsymbol{\alpha}_z(x)^T \boldsymbol{u}_z$$

$$\theta_x(x) = \boldsymbol{\alpha}_\theta(x)^T \boldsymbol{u}_\theta$$

in which,

$$\boldsymbol{u}_x = [u_1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad u_2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\boldsymbol{u}_y = [0 \quad v_1 \quad 0 \quad 0 \quad 0 \quad \theta_{z1} \quad 0 \quad v_2 \quad 0 \quad 0 \quad 0 \quad \theta_{z2}]^T$$

$$\boldsymbol{u}_z = [0 \quad 0 \quad w_1 \quad 0 \quad \theta_{y1} \quad 0 \quad 0 \quad w_2 \quad 0 \quad 0 \quad 0 \quad \theta_{y2}]^T$$

$$\boldsymbol{u}_\theta = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \theta_{x1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \theta_{x2}]^T$$

$$\boldsymbol{\alpha}_x(x) = [\alpha_{x,1}(x) \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \alpha_{x,2}(x) \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\boldsymbol{\alpha}_y(x) = [0 \quad \alpha_{y,1}(x) \quad 0 \quad 0 \quad 0 \quad \alpha_{y,2}(x) \quad 0 \quad \alpha_{y,3}(x) \quad 0 \quad 0 \quad 0 \quad \alpha_{y,4}(x)]^T$$

$$\boldsymbol{\alpha}_z(x) = [0 \quad 0 \quad \alpha_{z,1}(x) \quad 0 \quad \alpha_{z,2}(x) \quad 0 \quad 0 \quad \alpha_{z,3}(x) \quad 0 \quad 0 \quad 0 \quad \alpha_{z,4}(x)]^T$$

$$\boldsymbol{\alpha}_\theta(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \alpha_{\theta,1}(x) \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \alpha_{\theta,2}(x)]^T$$

where, u, v, and w are the deflections along the element on x-, y-, and z-axis (as shown in Figure 3-3), respectively; θ is the element rotation; the subscripts $_x$, $_y$, and $_z$ denote x-, y-, and z-axis, respectively; the subscripts $_1$ and $_2$ denote two element ends, respectively (as demonstrated in Figure 3-3); and α_x, α_y, α_z, and α_θ are the shape function coefficients provided in Appendix.

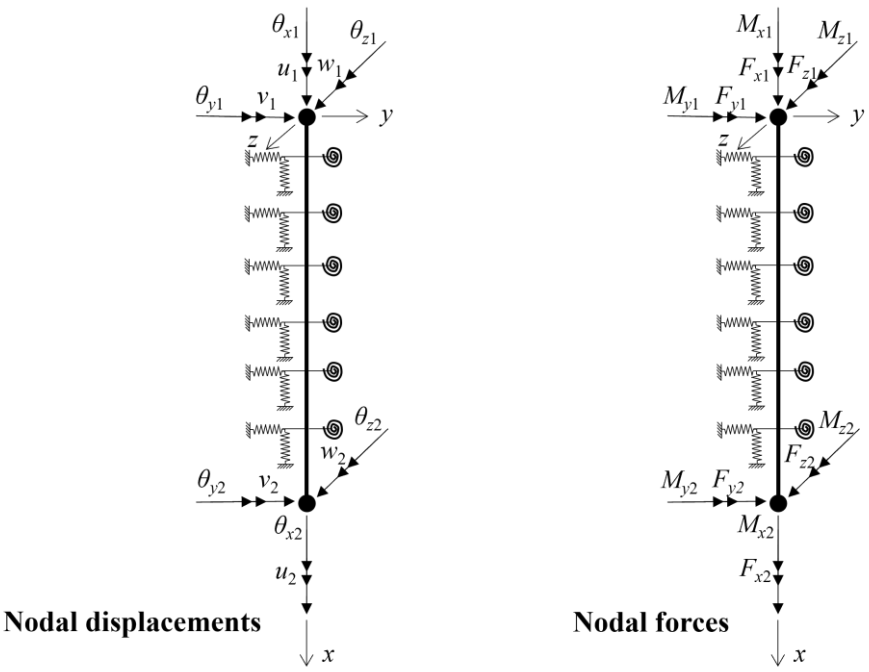


Figure 3-3 Element nodal displacements and element nodal forces.

3.2.3 Total potential energy

The element stiffness matrix and the secant relations are derived from the total potential energy, Π, which can be divided into three parts as:

$$\Pi = U_S + U_E - W$$

in which, U_S denotes the consumed energy by the soil resistance; U_E is the element strain energy; and W represents the work done by the external load. The last two terms, U_E and W, can be found in many finite element studies, such as Przemieniecki [187] and Abdelrahman et al. [188]. For simplicity, this chapter will omit the corresponding energy formula.

For the first part in Π, the consumed energy by the soil resistance can be further divided into two parts:

$$U_S = U_p + U_t$$

where, U_p denotes the energy consumed by the lateral soil resistance; and U_t is the energy consumed by the shaft soil resistance.

U_p can be calculated according to:

$$U_p = \int_0^L \int_0^{s_L(x)} p(x, \quad y(x)) dy dx$$

where,

$$s_L(x) = \sqrt{v^2(x) + w^2(x)}$$

in which s_L denotes the lateral element deflection on *y*-*z* (horizontal) plane; p is the lateral soil resistance on the unit pile length at the given depth, x, with the corresponding lateral deflection $y(x)$, which can be obtained from the selected *p*-*y* curves.

The current study assumes that the tangential displacement of piles at the pile-soil interface is resulted only from vertical movement (axial displacement) and rotation of the pile in the horizontal plane, implying axial symmetry with the tangential displacement of the piles. Based on this assumption, the energy consumed by the shaft soil resistance, U_t, can be written as:

$$U_t = \int_{A_{PS}} \int_0^{s_t(x)} t(x, \quad z(x)) dz dA_{PS} = 2\pi R \int_0^L \int_0^{s_t(x)} t(x, \quad z(x)) dz dx$$

where A_{PS} denotes the area of the pile-soil interface; s_t is the element tangential displacement on the pile-soil interface; R represents the radius of the pile cross-section; t denotes the shaft soil resistance on the unit area of the pile-soil interface at the given depth, x, with the corresponding tangential displacement $z(x)$, which can be obtained from the selected *t*-*z* curves. In addition, while this study utilizes the *t*-*z* curve model to characterize the soil resistance along the pile shafts, alternative models for shaft soil resistance may also be adopted as required.

The pile cross-section is assumed to be circular. Additionally, the shaft soil resistance only caused by axial settlement and torsional deformation. Thus, the element

tangential displacement on the pile-soil interface can be calculated according to:

$$s_t(x) = \sqrt{u^2(x) + [R\theta_x(x)]^2}$$

3.2.4 Element stiffness matrix

The element stiffness matrix is necessary when predicting the element displacement and deformation, which can be derived from the second-order variation of the total potential energy:

$$\boldsymbol{k} = \frac{\partial^2 \Pi}{\partial \boldsymbol{u}^2}$$

where,

$$\begin{aligned}\boldsymbol{u} &= \boldsymbol{u}_x + \boldsymbol{u}_y + \boldsymbol{u}_z + \boldsymbol{u}_\theta \\ &= \begin{bmatrix} u_1 & v_1 & w_1 & \theta_{x1} & \theta_{y1} & \theta_{z1} & u_2 & v_2 & w_2 & \theta_{x2} & \theta_{y2} & \theta_{z2} \end{bmatrix}^T\end{aligned}$$

The element stiffness matrix can be divided into three parts:

$$\boldsymbol{k} = \boldsymbol{k}_L + \boldsymbol{k}_G + \boldsymbol{k}_S$$

in which $\boldsymbol{k}_L$ and $\boldsymbol{k}_G$ are the linear stiffness matrix and the geometric stiffness matrix of the Timoshenko beam element, respectively; and, $\boldsymbol{k}_S$ denotes the soil stiffness matrix, which can be further divided into two parts:

$$\boldsymbol{k}_S = \boldsymbol{k}_{S,p} + \boldsymbol{k}_{S,t} = \frac{\partial^2 U_p}{\partial \boldsymbol{u}^2} + \frac{\partial^2 U_t}{\partial \boldsymbol{u}^2}$$

To obtain the soil stiffness matrix calculated from the *p*-*y* curves, the expression of the energy consumed by the lateral soil resistance, U_p, can be rewritten as:

$$U_p = \int_0^L \overline{U_p}(x)dx$$

where $\overline{U_p}$ is the density of the energy consumed by the lateral soil resistance along the pile:

$$\overline{U_p}(x) = \int_0^{s_L(x)} p(x, \quad y(x))dy$$

Consequently, the $\boldsymbol{k}_{S,p}$ can be expressed as:

$$\boldsymbol{k}_{S,p} = \int_0^L \frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}^2} dx \tag{3.1}$$

Since the expression of U_p only involves $\boldsymbol{u}_y$ and $\boldsymbol{u}_z$, Eq. (3.1) can be simplified as:

$$\boldsymbol{k}_{S,p} = \int_0^L \frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_y{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_z{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} dx + \int_0^L \frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y} dx \tag{3.2}$$

To calculate the second-order partial derivatives as shown above, the first-order partial derivatives of $\int_0^{s_L(x)} p(x, \quad y(x))dy$ should be obtained:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{u}_y}\left[\int_0^{s_L(x)} p(x, \quad y(x))dy\right] &= \left\{\frac{\partial}{\partial s_L(x)}\left[\int_0^{s_L(x)} p(x, \quad y(x))dy\right]\right\}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y} \\ &= p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{u}_z}\left[\int_0^{s_L(x)} p(x, \quad y(x))dy\right] &= \left\{\frac{\partial}{\partial s_L(x)}\left[\int_0^{s_L(x)} p(x, \quad y(x))dy\right]\right\}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z} \\ &= p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z} \end{aligned} \tag{3.4}$$

where the derivatives of $s_L(x)$ is:

$$\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y} = \frac{\partial s_L(x)}{\partial v(x)}\frac{\partial v(x)}{\partial \boldsymbol{u}_y} = \frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_y(x) \tag{3.5}$$

$$\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z} = \frac{\partial s_L(x)}{\partial w(x)}\frac{\partial w(x)}{\partial \boldsymbol{u}_z} = \frac{w(x)}{s_L(x)}\boldsymbol{\alpha}_z(x) \tag{3.6}$$

By submitting the above two equations, Eqs. (3.5)-(3.6), into Eqs. (3.3)-(3.4), the second-order derivatives of $\overline{U_p}(x)$ can be obtained as:

$$\frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_y{}^2} = \frac{\partial^2}{\partial \boldsymbol{u}_y{}^2}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right]$$

$$= \frac{\partial}{\partial \boldsymbol{u}_y}\left\{\frac{\partial}{\partial \boldsymbol{u}_y}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right]\right\}$$

$$= \frac{\partial}{\partial \boldsymbol{u}_y}\left\{p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right\}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial \boldsymbol{u}_y}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y{}^2}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial s_L(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y{}^2}$$

$$= k_p(x, \quad s_L(x))\left[\frac{v(x)}{s_L(x)}\right]^2 \boldsymbol{\alpha}_y(x)\boldsymbol{\alpha}_y(x)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y{}^2}$$

$$\frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_z{}^2} = \frac{\partial^2}{\partial \boldsymbol{u}_z{}^2}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right] = \frac{\partial}{\partial \boldsymbol{u}_z}\left\{\frac{\partial}{\partial \boldsymbol{u}_z}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right]\right\}$$

$$= \frac{\partial}{\partial \boldsymbol{u}_z}\left\{p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right\}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial \boldsymbol{u}_z}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z{}^2}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial s_L(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z{}^2}$$

$$= k_p(x, \quad s_L(x))\left[\frac{w(x)}{s_L(x)}\right]^2 \boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_z(x)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z{}^2}$$

$$\frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} = \frac{\partial^2}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right] = \frac{\partial}{\partial \boldsymbol{u}_y}\left\{\frac{\partial}{\partial \boldsymbol{u}_z}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right]\right\}$$

$$= \frac{\partial}{\partial \boldsymbol{u}_y}\left\{p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right\}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial \boldsymbol{u}_y}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial s_L(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z}$$

$$= k_p(x, \quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)}\boldsymbol{\alpha}_y(x)\boldsymbol{\alpha}_z(x)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z}$$

$$\frac{\partial^2 \overline{U_p}(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y} = \frac{\partial^2}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right] = \frac{\partial}{\partial \boldsymbol{u}_z}\left\{\frac{\partial}{\partial \boldsymbol{u}_y}\left[\int_0^{s_L(x)} p(x, \quad y(x))\, dy\right]\right\}$$

$$= \frac{\partial}{\partial \boldsymbol{u}_z}\left\{p(x, \quad s_L(x))\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right\}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial \boldsymbol{u}_z}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}$$

$$= \frac{\partial p(x, \quad s_L(x))}{\partial s_L(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\left(\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}$$

$$= k_p(x, \quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T + p(x, \quad s_L(x))\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}$$

where k_p (x, $s_L(x)$) represents the tangential value of the *p*-*y* curve at the given depth, x, when the lateral deflection equals to s_L.

By taking the derivative of Eqs. (3.5)-(3.6), the second-order partial derivatives of s_L can be expressed as:

$$\frac{\partial s_L^2(x)}{\partial \boldsymbol{u}_y^2} = \frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_y(x)\right] = \frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{v(x)}{s_L(x)}\right]\boldsymbol{\alpha}_y(x)^T$$
$$= \left\{\frac{\partial v(x)}{\partial \boldsymbol{u}_y}\frac{1}{s_L(x)} + v(x)\frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{1}{s_L(x)}\right]\right\}\boldsymbol{\alpha}_y(x)^T$$
$$= \left\{\frac{1}{s_L(x)}\boldsymbol{\alpha}_y(x)^T - \frac{v(x)}{s_L^2(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\right\}\boldsymbol{\alpha}_y(x)^T$$
$$= \left\{\frac{1}{s_L(x)}\boldsymbol{\alpha}_y(x) - \frac{v(x)}{s_L^2(x)}\left[\frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_y(x)\right]\right\}\boldsymbol{\alpha}_y(x)^T$$
$$= \left[\frac{1}{s_L(x)} - \frac{v^2(x)}{s_L^3(x)}\right]\boldsymbol{\alpha}_y(x)\boldsymbol{\alpha}_y(x)^T$$

$$\frac{\partial s_L^2(x)}{\partial \boldsymbol{u}_z^2} = \frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{w(x)}{s_L(x)}\boldsymbol{\alpha}_z(x)\right] = \frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{w(x)}{s_L(x)}\right]\boldsymbol{\alpha}_z(x)^T$$
$$= \left\{\frac{\partial w(x)}{\partial \boldsymbol{u}_z}\frac{1}{s_L(x)} + w(x)\frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{1}{s_L(x)}\right]\right\}\boldsymbol{\alpha}_z(x)^T$$
$$= \left\{\frac{1}{s_L(x)}\boldsymbol{\alpha}_z(x) - \frac{w(x)}{s_L^2(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\right\}\boldsymbol{\alpha}_z(x)^T$$
$$= \left\{\frac{1}{s_L(x)}\boldsymbol{\alpha}_z(x) - \frac{w(x)}{s_L^2(x)}\left[\frac{w(x)}{s_L(x)}\boldsymbol{\alpha}_z(x)\right]\right\}\boldsymbol{\alpha}_z(x)^T$$
$$= \left[\frac{1}{s_L(x)} - \frac{w^2(x)}{s_L^3(x)}\right]\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_z(x)^T$$

$$\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} = \frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{w(x)}{s_L(x)}\boldsymbol{\alpha}_z(x)\right] = \frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{w(x)}{s_L(x)}\right]\boldsymbol{\alpha}_y(x)^T$$
$$= \left\{\frac{\partial w(x)}{\partial \boldsymbol{u}_y}\frac{1}{s_L(x)} + w(x)\frac{\partial}{\partial \boldsymbol{u}_y}\left[\frac{1}{s_L(x)}\right]\right\}\boldsymbol{\alpha}_y(x)^T$$
$$= -\frac{w(x)}{s_L^2(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_y}\boldsymbol{\alpha}_y(x)^T = -\frac{w(x)}{s_L^2(x)}\left[\frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_y(x)\right]\boldsymbol{\alpha}_y(x)^T$$
$$= -\frac{w(x)v(x)}{s_L^3(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T$$

$$\frac{\partial^2 s_L(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y} = \frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_y(x)\right] = \frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{v(x)}{s_L(x)}\right]\boldsymbol{\alpha}_y(x)^T$$
$$= \left\{\frac{\partial v(x)}{\partial \boldsymbol{u}_z}\frac{1}{s_L(x)} + v(x)\frac{\partial}{\partial \boldsymbol{u}_z}\left[\frac{1}{s_L(x)}\right]\right\}\boldsymbol{\alpha}_y(x)^T$$
$$= -\frac{v(x)}{s_L^2(x)}\frac{\partial s_L(x)}{\partial \boldsymbol{u}_z}\boldsymbol{\alpha}_y(x)^T = -\frac{w(x)}{s_L^2(x)}\left[\frac{v(x)}{s_L(x)}\boldsymbol{\alpha}_z(x)\right]\boldsymbol{\alpha}_y(x)^T$$
$$= -\frac{w(x)v(x)}{s_L^3(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T$$

Consequently, the second-order partial derivatives of $\overline{U_p}(x)$ can be obtained as:

$$\frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} = k_p(x,\quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)}\boldsymbol{\alpha}_y(x)\boldsymbol{\alpha}_z(x)^T - p(x,\quad s_L(x))\frac{w(x)v(x)}{s_L^3(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T = \left[k_p(x,\quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)} - p(x,\quad s_L(x))\frac{w(x)v(x)}{s_L^3(x)}\right]\boldsymbol{\alpha}_y(x)\boldsymbol{\alpha}_z(x)^T$$

$$\frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y} = k_p(x,\quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T - p(x,\quad s_L(x))\frac{w(x)v(x)}{s_L^3(x)}\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T = \left[k_p(x,\quad s_L(x))\frac{v(x)w(x)}{s_L^2(x)} - p(x,\quad s_L(x))\frac{w(x)v(x)}{s_L^3(x)}\right]\boldsymbol{\alpha}_z(x)\boldsymbol{\alpha}_y(x)^T$$

But even with the expression of the second-order derivatives of $\overline{U_p}(x)$, it is still challenging to write down the lateral soil stiffness matrix, $\boldsymbol{k}_{S,p}$, in an analytical form given the complexity of the employed *p-y* curves. To address this issue, Gaussian quadrature [189] is employed to simplify the expression of Eq. (3.2):

$$\begin{aligned}\boldsymbol{k}_{S,p} &= \int_0^L \left[\frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_y{}^2} + \frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_z{}^2} + \frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} + \frac{\partial^2\overline{U_p}(x)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}\right]dx \\ &\approx L\sum_{i=1}^{n} H_i\left[\frac{\partial^2\overline{U_p}(x_i)}{\partial \boldsymbol{u}_y{}^2} + \frac{\partial^2\overline{U_p}(x_i)}{\partial \boldsymbol{u}_z{}^2} + \frac{\partial^2\overline{U_p}(x_i)}{\partial \boldsymbol{u}_y \partial \boldsymbol{u}_z} + \frac{\partial^2\overline{U_p}(x_i)}{\partial \boldsymbol{u}_z \partial \boldsymbol{u}_y}\right] \\ &= \boldsymbol{k}_{p,k} + \boldsymbol{k}_{p,p}\end{aligned}$$

$$x_i = \gamma_i L$$

in which, the soil stiffness matrix, $\boldsymbol{k}_{S,p}$ can be divided into two parts resulted from the later soil stiffness and resistance, respectively:

$$\boldsymbol{k}_{p,k} = L\sum_{i=1}^{n} H_i \left\{ k_p(x_i, \quad s_L(x_i)) \left[\frac{v(x_i)}{s_L(x_i)}\right]^2 \right\} \boldsymbol{\alpha}_y(x_i)\boldsymbol{\alpha}_y(x_i)^T$$
$$+ L\sum_{i=1}^{n} H_i \left\{ k_p(x_i, \quad s_L(x_i)) \left[\frac{w(x_i)}{s_L(x_i)}\right]^2 \right\} \boldsymbol{\alpha}_z(x_i)\boldsymbol{\alpha}_z(x_i)^T$$
$$+ L\sum_{i=1}^{n} H_i \left[k_p(x_i, \quad s_L(x_i)) \frac{v(x_i)w(x_i)}{s_L^2(x_i)} \right] \boldsymbol{\alpha}_y(x_i)\boldsymbol{\alpha}_z(x_i)^T$$
$$+ L\sum_{i=1}^{n} H_i \left[k_p(x_i, \quad s_L(x_i)) \frac{v(x_i)w(x_i)}{s_L^2(x_i)} \right] \boldsymbol{\alpha}_z(x_i)\boldsymbol{\alpha}_y(x_i)^T$$

$$\boldsymbol{k}_{p,p} = L\sum_{i=1}^{n} H_i \left\{ p(x_i, \quad s_L(x_i)) \left[\frac{1}{s_L(x_i)} - \frac{v^2(x_i)}{s_L^3(x_i)}\right] \right\} \boldsymbol{\alpha}_y(x_i)\boldsymbol{\alpha}_y(x_i)^T$$
$$+ L\sum_{i=1}^{n} H_i \left\{ p(x_i, \quad s_L(x_i)) \left[\frac{1}{s_L(x_i)} - \frac{w^2(x_i)}{s_L^3(x_i)}\right] \right\} \boldsymbol{\alpha}_z(x_i)\boldsymbol{\alpha}_z(x_i)^T$$
$$- L\sum_{i=1}^{n} H_i \left[p(x_i, \quad s_L(x_i)) \frac{w(x_i)v(x_i)}{s_L^3(x_i)} \right] \boldsymbol{\alpha}_y(x_i)\boldsymbol{\alpha}_z(x_i)^T$$
$$- L\sum_{i=1}^{n} H_i \left[p(x_i, \quad s_L(x_i)) \frac{w(x_i)v(x_i)}{s_L^3(x_i)} \right] \boldsymbol{\alpha}_z(x_i)\boldsymbol{\alpha}_y(x_i)^T$$

where the subscript $_i$ is the indices of the Gauss point number; n denotes the number of the employed Gauss points, which equals seven in this chapter; H_i and γ_i are the weight coefficient and the dimensionless Gauss point location.

The second part in the soil stiffness matrix, $\boldsymbol{k}_{S,t}$, can be written as:

$$\boldsymbol{k}_{S,t} = 2\pi R \left(\int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} dx \right.$$
$$\left. + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x} dx \right)$$

where,

$$\overline{U_t}(x) = \int_0^{s_t(x)} t(x, \quad z(x))dz$$

Before obtaining the second-order partial derivative of $\overline{U_t}(x)$, the corresponding first-order partial derivatives should be derived as:

$$\frac{\partial}{\partial \boldsymbol{u}_x}\left[\int_0^{s_t(x)} t(x, \quad z(x))dz\right] = \left\{\frac{\partial}{\partial s_t(x)}\left[\int_0^{s_t(x)} t(x, \quad z(x))dz\right]\right\}\frac{\partial s_t(x)}{\partial \boldsymbol{u}_x}$$
$$= t(x, \quad s_t(x))\frac{\partial s_t(x)}{\partial \boldsymbol{u}_x}$$

$$\frac{\partial}{\partial \boldsymbol{u}_\theta}\left[\int_0^{s_t(x)} t(x, \quad z(x))dz\right] = \left\{\frac{\partial}{\partial s_t(x)}\left[\int_0^{s_t(x)} t(x, \quad z(x))dz\right]\right\}\frac{\partial s_t(x)}{\partial \boldsymbol{u}_\theta}$$
$$= t(x, \quad s_t(x))\frac{\partial s_t(x)}{\partial \boldsymbol{u}_\theta}$$

The derivatives of the tangential displacement $z(x)$ with respect to $\boldsymbol{u}_x$ and $\boldsymbol{u}_\theta$ are given as follows:

$$\frac{\partial s_t(x)}{\partial \boldsymbol{u}_x} = \frac{\partial s_t(x)}{\partial u(x)}\frac{\partial u(x)}{\partial \boldsymbol{u}_x} = \frac{u(x)}{s_t(x)}\boldsymbol{\alpha}_x(x)$$

$$\frac{\partial s_t(x)}{\partial \boldsymbol{u}_\theta} = \frac{\partial s_t(x)}{\partial \theta(x)}\frac{\partial \theta(x)}{\partial \boldsymbol{u}_\theta} = R^2\frac{\theta(x)}{s_t(x)}\boldsymbol{\alpha}_\theta(x)$$

By submitting the above first-order derivatives of the tangential displacement, the second-order derivatives of $\overline{U_t}(x)$ can be obtained as:

$$\frac{\partial^2\overline{U_t}(x)}{\partial {\boldsymbol{u}_x}^2} = k_t(x, \quad s_t(x))\left[\frac{u(x)}{s_t(x)}\right]^2\boldsymbol{\alpha}_x(x)\boldsymbol{\alpha}_x(x)^T + t(x, \quad s_t(x))\frac{\partial^2 s_t(x)}{\partial {\boldsymbol{u}_x}^2}$$

$$\frac{\partial^2\overline{U_t}(x)}{\partial {\boldsymbol{u}_\theta}^2} = k_t(x, \quad s_t(x))\left[R^2\frac{\theta(x)}{s_t(x)}\right]^2\boldsymbol{\alpha}_\theta(x)\boldsymbol{\alpha}_\theta(x)^T + t(x, \quad s_t(x))\frac{\partial^2 s_t(x)}{\partial {\boldsymbol{u}_\theta}^2}$$

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} = k_t(x, \quad s_t(x)) R^2 \frac{u(x)\theta(x)}{s_L^2(x)} \boldsymbol{\alpha}_x(x) \boldsymbol{\alpha}_\theta(x)^T + t(x, \quad s_t(x)) \frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta}$$

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x} = k_t(x, \quad s_t(x)) R^2 \frac{u(x)\theta(x)}{s_L^2(x)} \boldsymbol{\alpha}_\theta(x) \boldsymbol{\alpha}_x(x)^T + t(x, \quad s_t(x)) \frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x}$$

where k_t (x, $s_t(x)$) denotes the tangential value of the *t-z* curve at the embedded depth, x, when the element tangential displacement on the pile-soil interface equals to s_t.

The second-order derivatives of $s_t(x)$ with respect to $\boldsymbol{u}_x$ and $\boldsymbol{u}_\theta$ can be formulated as:

$$\frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_x{}^2} = \left[\frac{1}{s_t(x)} - \frac{u^2(x)}{s_t^3(x)}\right] \boldsymbol{\alpha}_x(x) \boldsymbol{\alpha}_x(x)^T$$

$$\frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_\theta{}^2} = R^2 \left[\frac{1}{s_t(x)} - R^2 \frac{\theta^2(x)}{s_t^3(x)}\right] \boldsymbol{\alpha}_\theta(x) \boldsymbol{\alpha}_\theta(x)^T$$

$$\frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} = -R^2 \frac{\theta(x)u(x)}{s_t^3(x)} \boldsymbol{\alpha}_x(x) \boldsymbol{\alpha}_\theta(x)^T$$

$$\frac{\partial^2 s_t(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x} = -R \frac{\theta(x)u(x)}{s_t^3(x)} \boldsymbol{\alpha}_\theta(x) \boldsymbol{\alpha}_x(x)^T$$

Consequently, the second-order derivatives of $\overline{U_t}(x)$ can be rewritten as:

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x{}^2} = \left\{k_t(x, \quad s_t(x)) \left[\frac{u(x)}{s_t(x)}\right]^2 + t(x, \quad s_t(x)) \left[\frac{1}{s_t(x)} - \frac{u^2(x)}{s_t^3(x)}\right]\right\} \boldsymbol{\alpha}_x(x) \boldsymbol{\alpha}_x(x)^T$$

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta{}^2} = \left\{k_t(x, \quad s_t(x)) \left[R^2 \frac{\theta(x)}{s_t(x)}\right]^2 + t(x, \quad s_t(x)) R^2 \left[\frac{1}{s_t(x)} - R^2 \frac{\theta^2(x)}{s_t^3(x)}\right]\right\} \boldsymbol{\alpha}_\theta(x) \boldsymbol{\alpha}_\theta(x)^T$$

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} = \left[k_t(x, \quad s_t(x)) R^2 \frac{u(x)\theta(x)}{s_L^2(x)} - t(x, \quad s_t(x)) R^2 \frac{\theta(x)u(x)}{s_t^3(x)}\right] \boldsymbol{\alpha}_x(x) \boldsymbol{\alpha}_\theta(x)^T$$

$$\frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x} = \left[k_t(x, \quad s_t(x)) R^2 \frac{u(x)\theta(x)}{s_L^2(x)} - t(x, \quad s_t(x)) R \frac{\theta(x) u(x)}{s_t^3(x)} \right] \boldsymbol{\alpha}_\theta(x) \boldsymbol{\alpha}_x(x)^T$$

Similarly, by introducing Gaussian quadrature, the soil stiffness matrix determined by the shaft soil resistance can be calculated by:

$$\begin{aligned} \boldsymbol{k}_{S,t} &= 2\pi R \left(\int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta{}^2} dx + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} dx + \int_0^L \frac{\partial^2 \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta \partial \boldsymbol{u}_x} dx \right) \\ &\approx 2\pi RL \sum_{i=1}^{n} H_i \left[\frac{\partial^2 \overline{U_t}(x_i)}{\partial \boldsymbol{u}_x{}^2} + \frac{\partial^2 \overline{U_t}(x_i)}{\partial \boldsymbol{u}_\theta{}^2} + \frac{\partial^2 \overline{U_t}(x_i)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} + \frac{\partial^2 \overline{U_t}(x_i)}{\partial \boldsymbol{u}_x \partial \boldsymbol{u}_\theta} \right] \\ &= \boldsymbol{k}_{t,k} + \boldsymbol{k}_{t,t} \end{aligned}$$

in which, the soil stiffness matrix, $\boldsymbol{k}_{S,t}$ can be divided into two parts resulted from the shaft soil stiffness and resistance, respectively:

$$\begin{aligned} \boldsymbol{k}_{t,k} = {} & 2\pi RL \sum_{i=1}^{n} H_i \, k_t(x_i, \quad s_t(x_i)) \left[\frac{u(x_i)}{s_t(x_i)} \right]^2 \boldsymbol{\alpha}_x(x_i) \boldsymbol{\alpha}_x(x_i)^T \\ & + 2\pi RL \sum_{i=1}^{n} H_i \, k_t(x_i, \quad s_t(x_i)) \left[R^2 \frac{\theta(x_i)}{s_t(x_i)} \right]^2 \boldsymbol{\alpha}_\theta(x_i) \boldsymbol{\alpha}_\theta(x_i)^T \\ & + 2\pi RL \sum_{i=1}^{n} H_i \, k_t(x_i, \quad s_t(x_i)) R^2 \frac{u(x_i)\theta(x_i)}{s_L^2(x_i)} \boldsymbol{\alpha}_x(x_i) \boldsymbol{\alpha}_\theta(x_i)^T \\ & + 2\pi RL \sum_{i=1}^{n} H_i \, k_t(x_i, \quad s_t(x_i)) \boldsymbol{\alpha}_\theta(x_i) \boldsymbol{\alpha}_x(x_i)^T \end{aligned}$$

$$\boldsymbol{k}_{t,t} = 2\pi RL\sum_{i=1}^{n} H_i\, t(x_i,\quad s_t(x_i))\left[\frac{1}{s_t(x_i)} - \frac{u^2(x_i)}{s_t^3(x_i)}\right]\boldsymbol{\alpha}_x(x_i)\boldsymbol{\alpha}_x(x_i)^T$$
$$+ 2\pi RL\sum_{i=1}^{n} H_i\, t(x_i,\quad s_t(x_i))R^2\left[\frac{1}{s_t(x_i)}\right.$$
$$\left. - R^2\frac{\theta^2(x_i)}{s_t^3(x_i)}\right]\boldsymbol{\alpha}_\theta(x_i)\boldsymbol{\alpha}_\theta(x_i)^T$$
$$+ 2\pi RL\sum_{i=1}^{n} H_i\, t(x_i,\quad s_t(x_i))R^2\frac{\theta(x_i)u(x_i)}{s_t^3(x_i)}\boldsymbol{\alpha}_x(x_i)\boldsymbol{\alpha}_\theta(x_i)^T$$
$$+ 2\pi RL\sum_{i=1}^{n} H_i\, t(x_i,\quad s_t(x_i))R^2\frac{\theta(x_i)u(x_i)}{s_t^3(x_i)}\boldsymbol{\alpha}_\theta(x_i)\boldsymbol{\alpha}_x(x_i)^T$$

The linear stiffness matrix and the geometric matrix, $\boldsymbol{k}_L$ and $\boldsymbol{k}_G$, are commonly used in the Timoshenko beam element, which can be found in some other works [187, 190-192].

3.2.5 Secant relations

The element equilibrium condition is derived from the minimum energy theory, which can be described as:

$$\frac{\partial \Pi}{\partial \boldsymbol{u}} = 0$$

Hence, the element resisting forces under the given displacement, also named the secant relations, can be generated to minimize the numerical error accumulated in the nonlinear finite element analysis. The element resisting force vector can be divided into two parts:

$$\boldsymbol{F} = \boldsymbol{F}_S + \boldsymbol{F}_E$$

where,

$$\boldsymbol{F} = [F_{x1} \quad F_{y1} \quad F_{z1} \quad M_{x1} \quad M_{y1} \quad M_{z1} \quad F_{x2} \quad F_{y2} \quad F_{z2} \quad M_{x2} \quad M_{y2} \quad M_{z2}]^T$$

in which, $\boldsymbol{F}$ represent the element resisting vector, whose components are demonstrated

in Figure 3-3; $\boldsymbol{F}_S$ is the soil resisting vector; $\boldsymbol{F}_E$ is the resisting vector resulted from the element strain; F and M denote the force and bending moment at the element node, respectively.

The soil resisting vector, $\boldsymbol{F}_S$, can be calculated by use of expressions presented in the following::

$$\boldsymbol{F}_S = [F_{x1,S} \quad F_{y1,S} \quad F_{z1,S} \quad M_{x1,S} \quad M_{y1,S} \quad M_{z1,S} \quad F_{x2,S} \quad F_{y2,S} \quad F_{z2,S} \quad M_{x2,S} \quad M_{y2,S} \quad M_{z2,S}]^T$$

$$\boldsymbol{F}_S = \frac{\partial U_S}{\partial \boldsymbol{u}} = \frac{\partial U_p}{\partial \boldsymbol{u}_y} + \frac{\partial U_p}{\partial \boldsymbol{u}_z} + \frac{\partial U_t}{\partial \boldsymbol{u}_x} + \frac{\partial U_t}{\partial \boldsymbol{u}_\theta}$$

$$= \int_0^L \frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_y} dx + \int_0^L \frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_z} dx + 2\pi R \left(\int_0^L \frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_x} dx + \int_0^L \frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta} dx \right)$$

The first-order derivative of $\overline{U_p}(x)$ and $\overline{U_t}(x)$ with respect to different degrees of freedom can be expressed as:

$$\frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_y} = \frac{\partial}{\partial \boldsymbol{u}_y} \left[\int_0^{s_L(x)} p(x, \quad y(x)) dx \right] = p(x, \quad s_L(x)) \frac{\partial s_L(x)}{\partial \boldsymbol{u}_y} = p(x, \quad s_L(x)) \frac{v(x)}{s_L(x)} \boldsymbol{\alpha}_y(x)$$

$$\frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_z} = \frac{\partial}{\partial \boldsymbol{u}_z} \left[\int_0^{s_L(x)} p(x, \quad y(x)) dx \right] = p(x, \quad s_L(x)) \frac{\partial s_L(x)}{\partial \boldsymbol{u}_z} = p(x, \quad s_L(x)) \frac{w(x)}{s_L(x)} \boldsymbol{\alpha}_z(x)$$

$$\frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_x} = \frac{\partial}{\partial \boldsymbol{u}_x} \left[\int_0^{s_t(x)} t(x, \quad z(x)) dz \right] = t(x, \quad s_t(x)) \frac{\partial s_t(x)}{\partial \boldsymbol{u}_x} = t(x, \quad s_t(x)) \frac{u(x)}{s_t(x)} \boldsymbol{\alpha}_x(x)$$

$$\frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta} = \frac{\partial}{\partial \boldsymbol{u}_\theta} \left[\int_0^{s_t(x)} t(x, \quad z(x)) dz \right] = t(x, \quad s_t(x)) \frac{\partial s_t(x)}{\partial \boldsymbol{u}_\theta} = t(x, \quad s_t(x)) R^2 \frac{\theta(x)}{s_t(x)} \boldsymbol{\alpha}_\theta(x)$$

Accordingly, by introducing Gaussian quadrature, the secant relations generated from the soil resistance $\boldsymbol{F}_S$ can be calculated by:

$$\boldsymbol{F}_S = \int_0^L \frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_y} dx + \int_0^L \frac{\partial \overline{U_p}(x)}{\partial \boldsymbol{u}_z} dx + 2\pi R \left(\int_0^L \frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_x} dx + \int_0^L \frac{\partial \overline{U_t}(x)}{\partial \boldsymbol{u}_\theta} dx \right)$$

$$\approx L \sum_{i=1}^{n} H_i \left[\frac{\partial \overline{U_p}(x_i)}{\partial \boldsymbol{u}_y} + \frac{\partial \overline{U_p}(x_i)}{\partial \boldsymbol{u}_y} + 2\pi R \frac{\partial \overline{U_p}(x_i)}{\partial \boldsymbol{u}_y} + 2\pi R \frac{\partial \overline{U_p}(x_i)}{\partial \boldsymbol{u}_y} \right]$$

$$= L \sum_{i=1}^{n} H_i \, p(x_i, \quad s_L(x_i)) \frac{v(x_i)}{s_L(x_i)} \boldsymbol{\alpha}_y(x_i)$$

$$+ \sum_{i=1}^{n} H_i \, p(x_i, \quad s_L(x_i)) \frac{w(x_i)}{s_L(x_i)} \boldsymbol{\alpha}_z(x_i)$$

$$+ 2\pi RL \left[\sum_{i=1}^{n} H_i \, t(x_i, \quad s_t(x_i)) \frac{u(x_i)}{s_t(x_i)} \boldsymbol{\alpha}_x(x_i) \right.$$

$$\left. + \sum_{i=1}^{n} H_i \, t(x_i, \quad s_t(x_i)) R^2 \frac{\theta(x_i)}{s_t(x_i)} \boldsymbol{\alpha}_\theta(x_i) \right]$$

The resisting force caused by the element strain energy, $\boldsymbol{F}_E$, is frequently discussed in the research related to Timoshenko beam element, such as [187, 190-192].

3.3 Numerical Procedure

For describing the large deflection of the pile, the Updated-Lagrangian method [33] is adopted, in which the equilibrium conditions is established according to the last-known element configuration. Then, the proposed refined 3DPE and the Updated-Lagrangian method are integrated into the Newton-Raphson incremental-iterative procedure [1] to minimize the accumulated error in the analysis procedure, as illustrated in Figure 3-4.

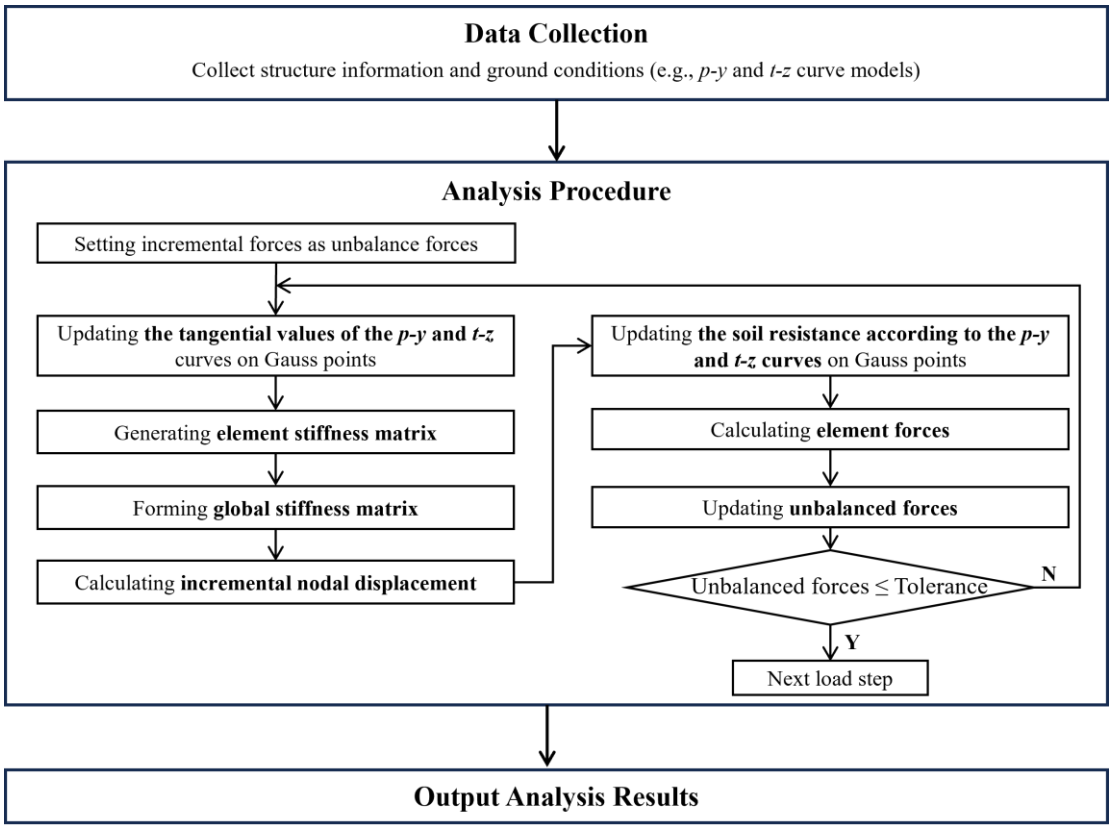


Figure 3-4 Flowchart of numerical procedure.

3.4 Validation Examples

In this section, the proposed refined 3DPE is compared with various existing analysis methods to establish its accuracy and computational efficiency. Furthermore, two pile-supported structures are investigated to illustrate the importance of the present study in facilitating the second-order analysis of such structures.

3.4.1 Example 1 – Comparisons with existing analysis methods

In this example group, four different soil conditions in are studied. The lateral soil resistance stiffness, k_L, is varied linearly with the embedded depth in the first soil condition [193, 194]. And the stiffness of the shaft soil resistance on the pile-soil interface, k_t, is also regarded as linearly changed with the embedded depth, as demonstrated in Figure 3-5A. The SPI curves of other three soil conditions are constructed according to the *p-y* and *t-z* curves recommended in API [155]. The corresponding soil parameters including the effective unit weigh (γ), the undrained

shear strength (c), the dimensionless empirical constant (J) and the strain which occurs at one-half the maximum stress (ε_c), are provided in Figure 3-5B-D. The bottom of the pile is taken as fixed in the bedrock. The proposed refined 3DPE is compared with different numerical analysis methods for validating its accuracy and efficiency for the spatial analysis of single piles in complex load and soil conditions. This comparison aims to illustrate the effectiveness of the proposed method in addressing the limitations of existing analysis techniques.

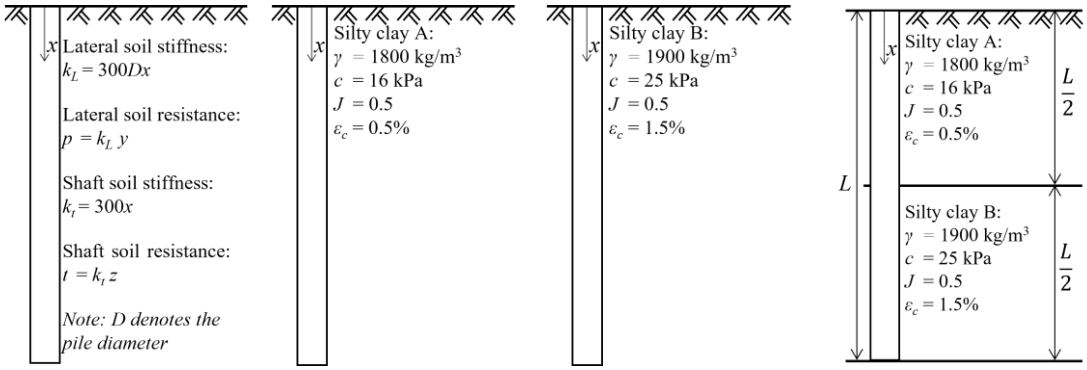


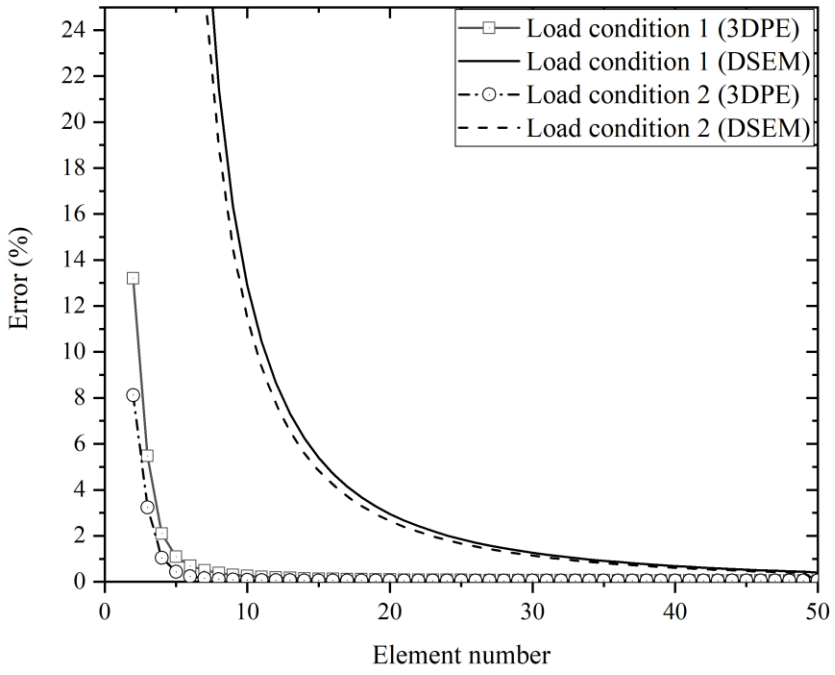
Figure 3-5 Four soil conditions in Example 1 : (A) Soil Condition 1. (B) Soil Condition 2. (C) Soil Condition 3. (D) Soil Condition 4.

Example 1.1 – Numerical accuracy and efficiency of the refined 3DPE

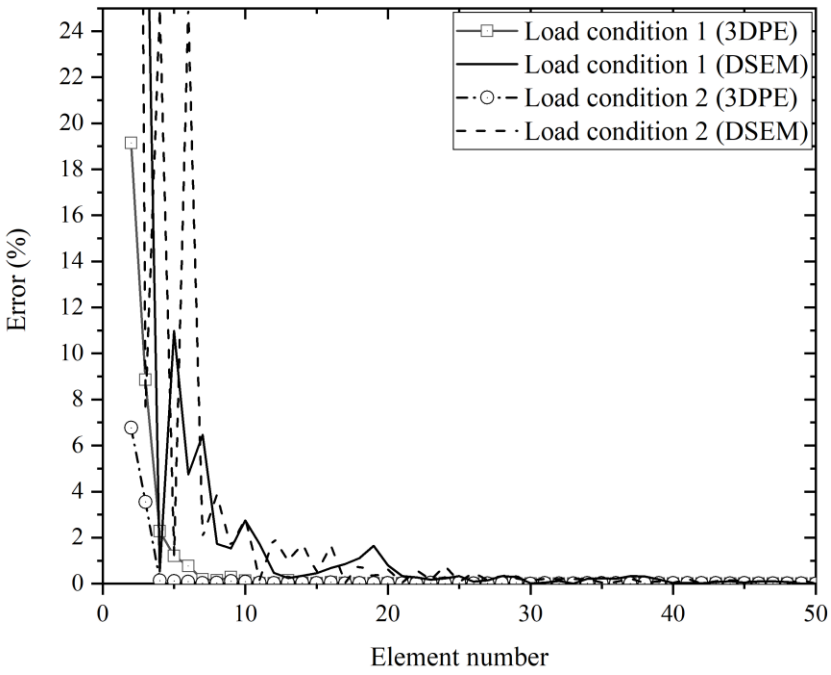
In this example, the numerical efficiency of the present study is validated and compared with DSEM. A RC pile with 40 m length and 1 m diameter is studied, whose cross-section is solid and circular. The Young's modulus and the shear modulus of the pile material are 32.5 GPa and 13.8 GPa, respectively. Two different load combinations at the pile head are included: (1) Loading condition 1: the compressive axial force P = 5000 kN, and the shear force along the y-axis V = 1000 kN; and (2) Loading condition 2: the shear force along the y-axis V = 3000 kN, and the bending moment along the z-axis M = 200 kN·m. The traditional DSEM using 150 elements are selected as the benchmark solution. The lateral deflections of the pile at the top end are obtained from

two different methods: (1) the proposed refined 3DPE; and (2) the DSEM. The element numbers excluding the spring elements are varied from 2 to 50. The relative error between the results from different methods compared with the benchmark solutions are illustrated in Figure 3-6.

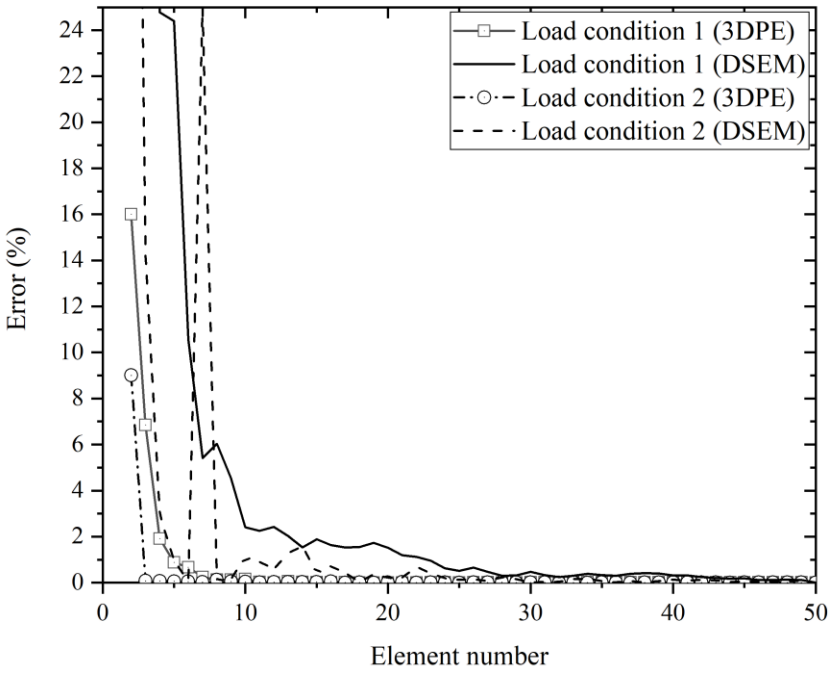
As indicated in the error-element number curves in Figure 3-6, the proposed 3DPE shows excellent performance in both result accuracy and numerical efficiency. It can be clearly observed from Figure 3-6 that the proposed refined 3DPE using only 3-4 elements can robustly achieve almost identical results to the benchmark solution when considering piles under different geological conditions and loading conditions. However, for the traditional DSEM, a large element number is required for reliable analysis results. For some complicated ground conditions, such as Soil conditions 4, it takes more than 40 elements in DSEM to narrow down the analysis error within 0.5 %. In summary, the proposed refined 3DPE can generate accurate analysis results with only 3-4 elements, implying its excellent performance in computational efficiency.



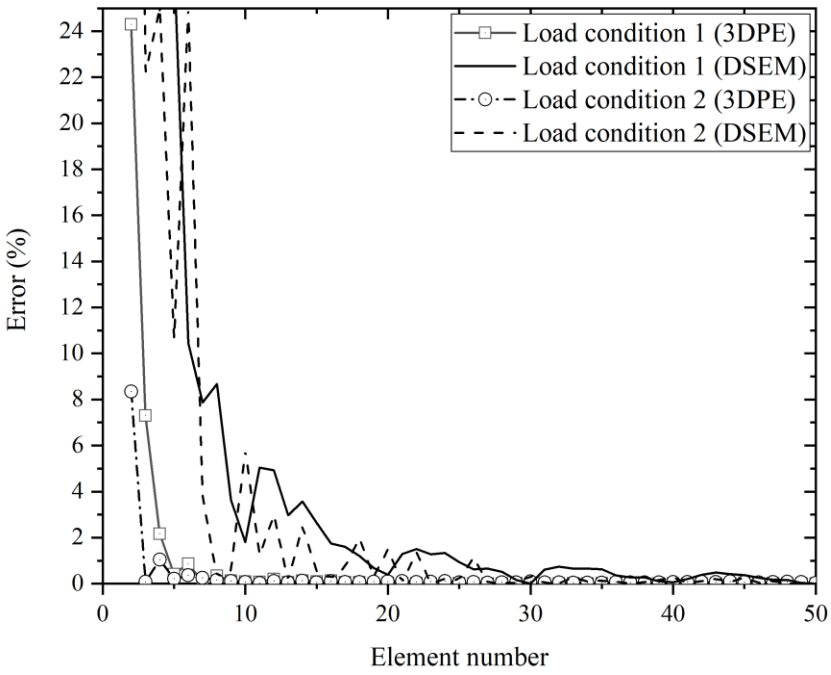
(A)


Load condition 1 (3DPE)
Load condition 1 (DSEM)
Load condition 2 (3DPE)
Load condition 2 (DSEM)
Error (%)
Element number
0
2
4
6
8
10
12
14
16
18
20
22
24
0
10
20
30
40
50

(B)


Load condition 1 (3DPE)
Load condition 1 (DSEM)
Load condition 2 (3DPE)
Load condition 2 (DSEM)
Error (%)
Element number
0
2
4
6
8
10
12
14
16
18
20
22
24
0
10
20
30
40
50

(C)



(D)

Figure 3-6 Element number versus relative error under different soil conditions: (A) Soil Condition 1. (B) Soil Condition 2. (C) Soil Condition 3. (D) Soil Condition 4.

Example 1.2 - Single piles under shear forces with nonlinear Soil-Pile Interaction

In the second example in Example 1, piles under the shear forces with different directions are studied. The pile studied in Example 1.1, with 1 m diameter and 40 m length, is reused in this example again. The magnitude of the shear force applied at the pile head, V, equals 2000 kN. Two different analysis methods, 3DPE using 6 elements and DSEM using 50 elements with orthogonal springs, are adopted to calculate the lateral deflections of pile heads on the y-axis when the direction angles of the applied shear forces vary from 0° to 45° to the y-axis. Theoretically, the deformations of the pile in the load direction should be same. Therefore, the benchmarks of the pile lateral deflections in y-axis direction are obtained following the equation,

$$u_y(\varphi) = u_{\varphi=0^\circ} \cos\varphi$$

where u_y is the projection of the lateral deflections of pile heads on the y-axis; φ is the angle between the shear force and the y-axis; $u_{\varphi=0^\circ}$ represents the magnitude of the

lateral deflection at the pile head when the shear force is parallel to the *y*-axis, which is obtained from DSEM using 150 elements in this example.

The relative error of the results from the 3DPE and DSEM to the benchmark solutions are plotted in Figure 3-7. It can be found that DSEM can precisely predict the structural behaviour of piles when the SPI is linear (Soil condition 1). When the SPI preforms nonlinearly, DSEM cannot generate reliable results. The relative error of the results from the DSEM grows with the angle between the shear force and the *y*-axis, which comes to 20% when φ equals 45°. By contrast, the proposed element can accurately simulate the nonlinear SPI in the spatial pile analysis, which further confirms the advantages of the proposed refined 3DPE.

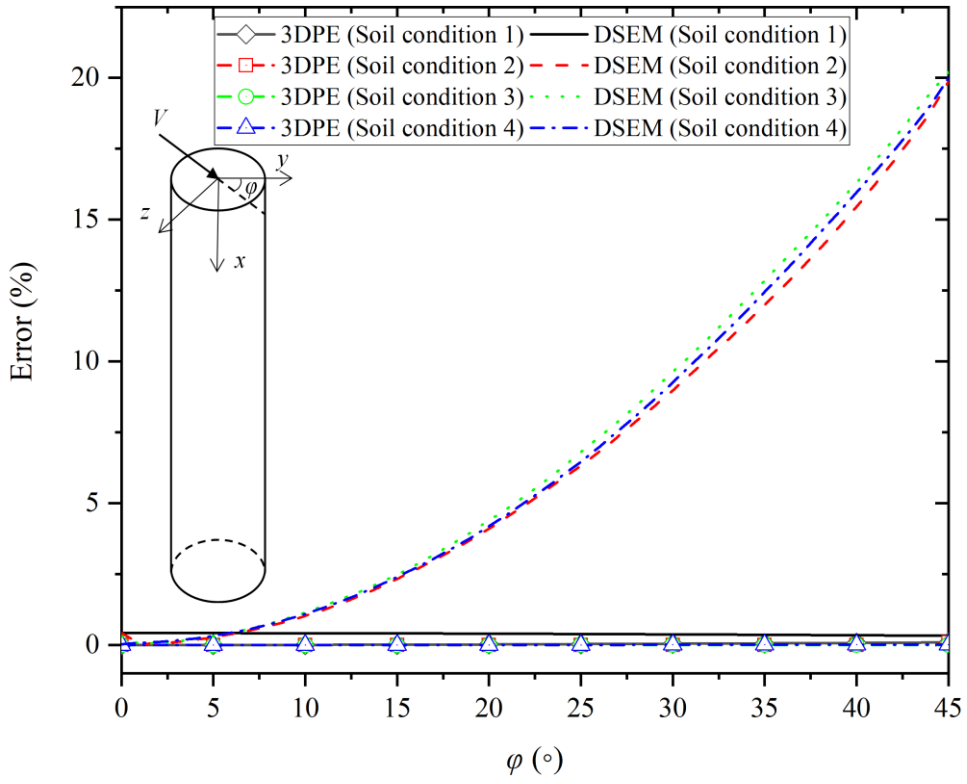


Figure 3-7 Errors of the lateral deformations obtained with different methods.

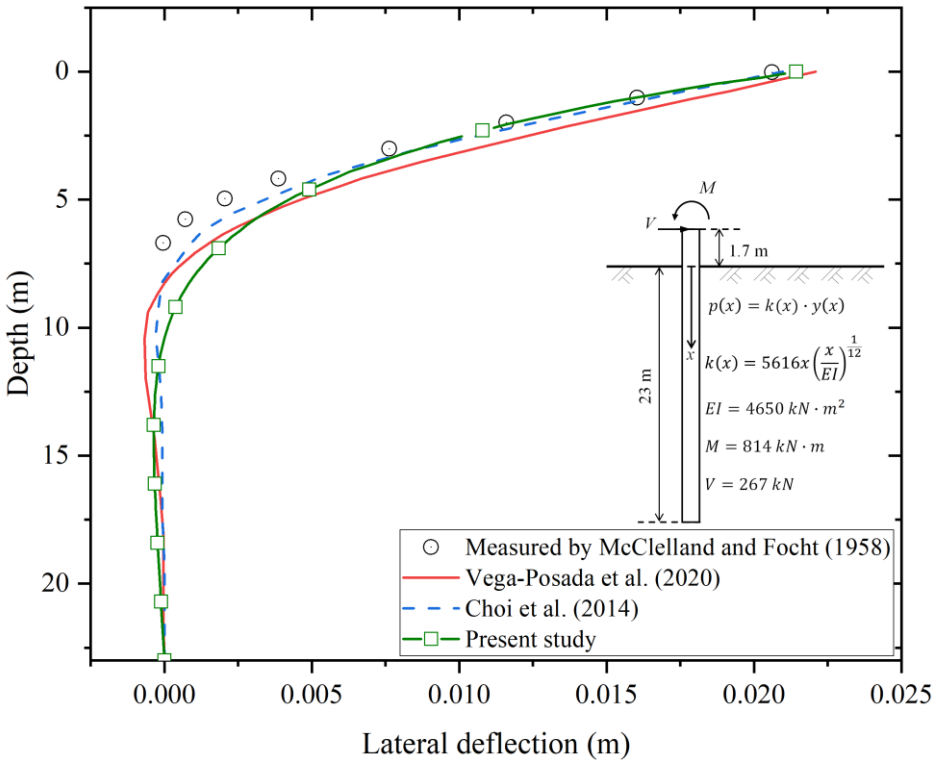


Figure 3-8 Lateral deflection along the circular pile obtained from different sources.

The proposed method is further verified via the comparison with the field test results recorded by McClelland and Focht [195]. A circular pile that partially embedded in the post-glacial deltaic deposits of the Mississippi River is revisited. The pile length is equal to 24.7 m. The flexural rigidity of the pile is EI = 4650 kN·m^2. The loading conditions assigned on the pile top can be found in . The *p-y* curve model proposed by Randolph [52] is introduced to describe the lateral soil resistance. The proposed refined 3DPE is employed to obtain the pile response and its results are compared with the measured data from field tests [195], as well as solutions from Vega-Posada et al. [196] and Choi et al. [197]. The comparison results, as shown in Figure 3-8, demonstrates a great agreement between the outcomes of the proposed method, the field measurements, and the referenced solutions, thereby reinforcing the accuracy of the proposed method.

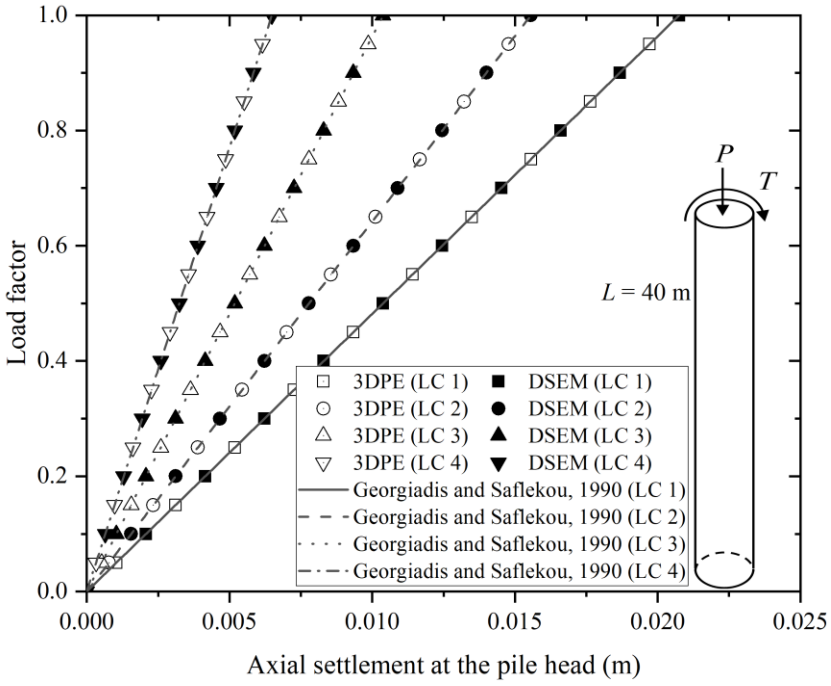
Example 1.3 - Single piles under axial and torsional load with nonlinear Soil-Pile Interaction

The combined axial-torsional response is investigated in this example. The pile with 40 m length and 1 m diameter is studied here. The pile cross-section is solid and circular. The Young's modulus and the shear modulus are 32.5 GPa and 13.8 GPa, respectively. Four different Loading Conditions (LC) of the compressive axial load, P, and torque, T, are applied at the pile head and summarized in Table 3-1. Four different soil conditions plotted in Figure 3-5 are also simulated in this example. The refined 3DPE, which considers the combined axial-torsional pile response, and DSEM, which treats the axial and torsional pile response independently, are adopted to obtain the axial settlement at the pile head. The method provided by Georgiadis and Saflekou [177] is employed as the benchmark solution. The comparison results are illustrated through the load factor versus the displacement curves in Figure 3-9.

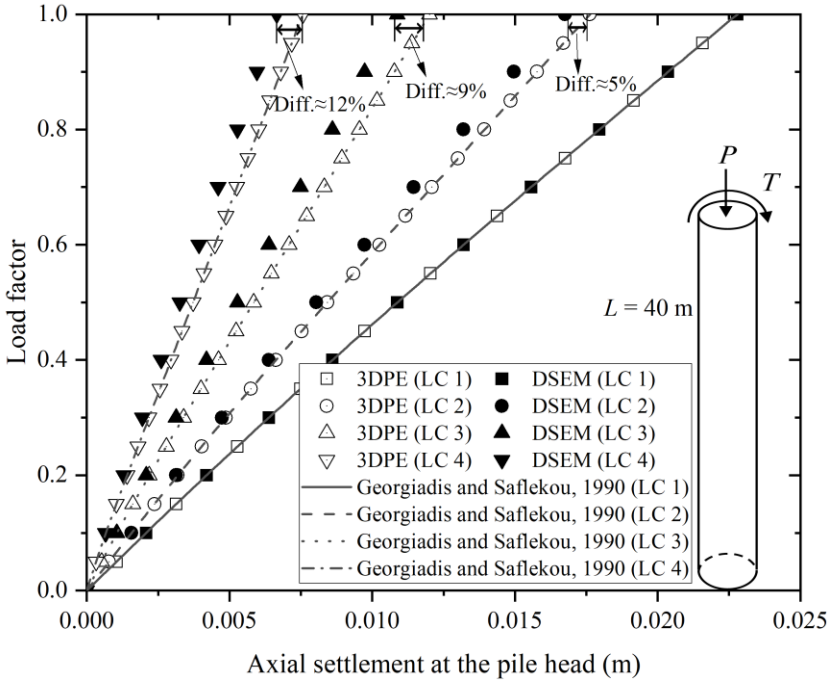
Table 3-1 Loading conditions in Example 1.3

Loading condition	LC 1	LC 2	LC 3	LC 4
P (kN)	16000	12000	8000	4000
T (kN·m)	0	4000	8000	12000

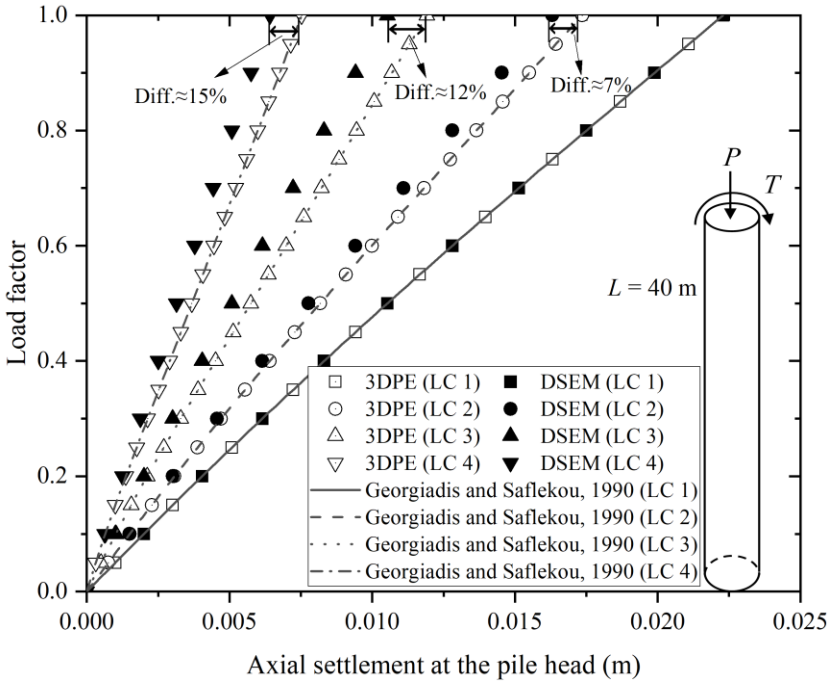
Figure 3-9 demonstrates that the results from the proposed method are identical to the benchmark solutions. However, the errors of DSEM are significant when the soil properties are nonlinear. The maximum difference of the DSEM method achieves 15%. This example confirms the importance of the present study in capturing the piles' response under combined axial-torsional loads.


Load factor
Axial settlement at the pile head (m)
P
T
L = 40 m
3DPE (LC 1)
DSEM (LC 1)
3DPE (LC 2)
DSEM (LC 2)
3DPE (LC 3)
DSEM (LC 3)
3DPE (LC 4)
DSEM (LC 4)
Georgiadis and Saflekou, 1990 (LC 1)
Georgiadis and Saflekou, 1990 (LC 2)
Georgiadis and Saflekou, 1990 (LC 3)
Georgiadis and Saflekou, 1990 (LC 4)

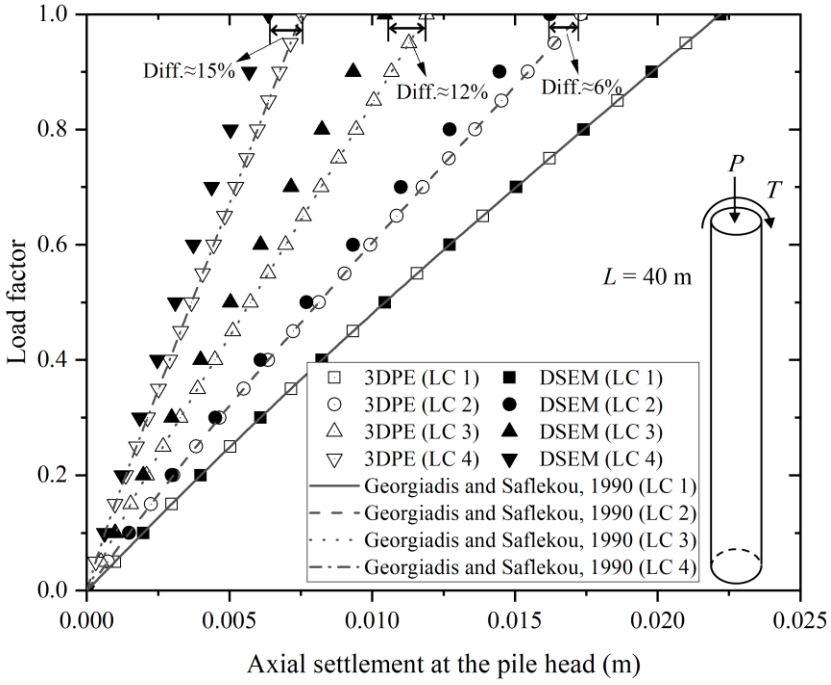
(A)


Diff.≈12%
Diff.≈9%
Diff.≈5%
Load factor
Axial settlement at the pile head (m)
P
T
L = 40 m
3DPE (LC 1)
DSEM (LC 1)
3DPE (LC 2)
DSEM (LC 2)
3DPE (LC 3)
DSEM (LC 3)
3DPE (LC 4)
DSEM (LC 4)
Georgiadis and Saflekou, 1990 (LC 1)
Georgiadis and Saflekou, 1990 (LC 2)
Georgiadis and Saflekou, 1990 (LC 3)
Georgiadis and Saflekou, 1990 (LC 4)

(B)



(C)



(D)

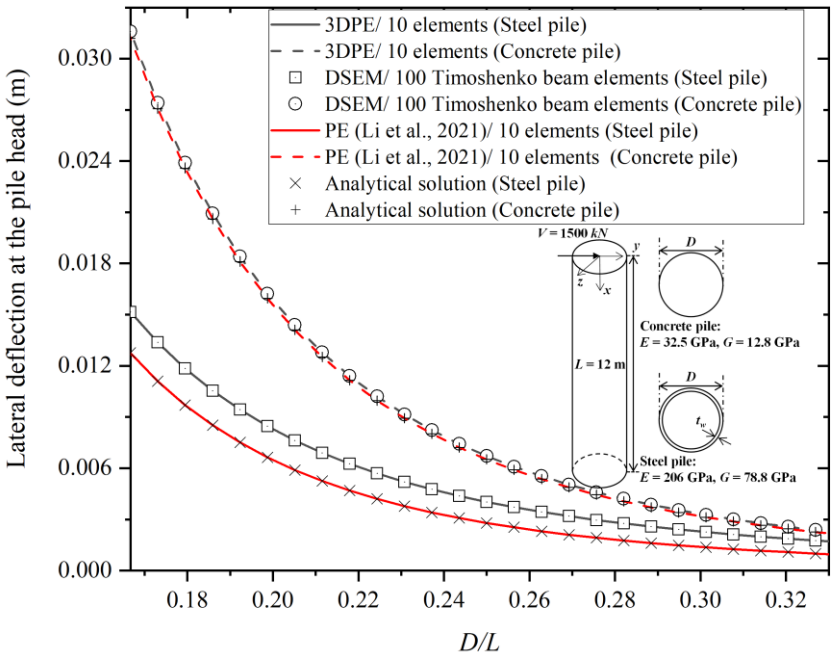
Figure 3-9 Load factor versus axial settlement at the pile head under different soil conditions: (A) Soil Condition 1. (B) Soil Condition 2. (C) Soil Condition 3. (D) Soil Condition 4.

Example 1.4 - Single piles with large diameters under shear forces

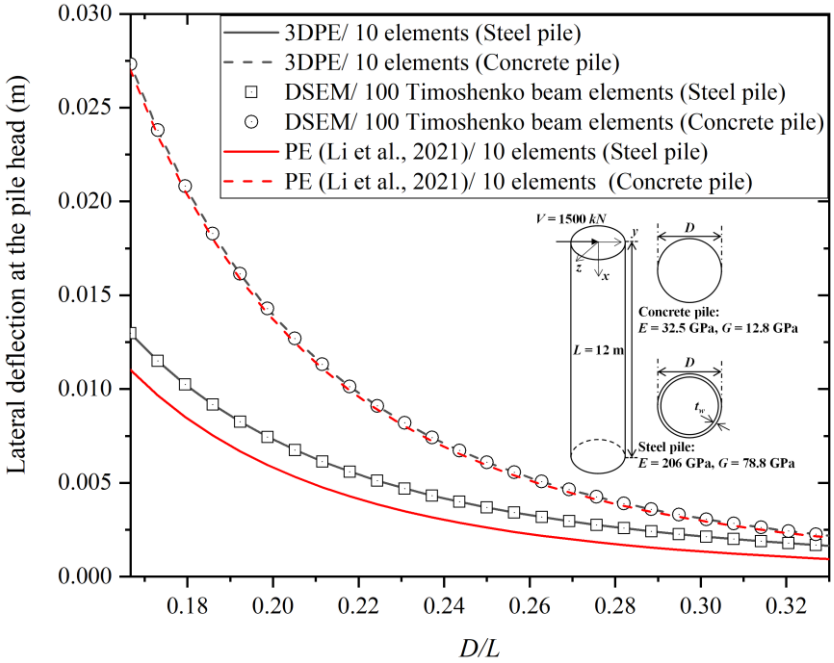
This example investigates the accuracy of the proposed refined 3DPE in analyzing piles with different cross-sections while considering nonlinear SPI and shear effect. Two cross-section types of piles are used, (1) steel piles with hollow and circular cross-section, and (2) concrete piles with solid and circular cross-section. The pile lengths, L, in this example are all equal to 12 m, whereas the diameter of the piles, D, varies from 2 m to 4 m. The wall thickness of the hollow cross-section is selected according to [155] as $t_w = 0.00635 + \frac{D}{100}$. The Young's modulus and the shear modulus of different pile materials are given in Figure 3-10. The shear correction factor are calculated according to Cowper [198]. A lateral force, V = 1500 kN, is applied at the pile head. Even though the API *p-y* curve model might sometimes be inaccurate for large-diameter piles [160], the present study still employs this model for piles of different diameters to maintain consistency. The DSEM using 100 Timoshenko beam elements is selected as the benchmark solution. Two analysis approach are implemented for the analysis of pile behaviors under different soil conditions, including: (1) the proposed refined 3DPE using 10 elements; and (2) the existing PE for the spatial analysis developed by Li et al. [183]. In addition, for soil condition 1, where soil stiffness is proportional to the embedded depth [194], an analytical solution from the Euler-Bernoulli beam theory by Reese and Matlock [193] is also employed for comparison.

The analysis results from different methods are demonstrated through the curves of the ratios between D and L versus the lateral deflection at the pile head. From the observation from Figure 3-10, it can be found that the existing PE for the spatial analysis [183] might lead to significant underestimation of pile deflection, especially for steel piles with hollow cross-sections, since the shear effect is ignored. The results obtained using the proposed method are found to be nearly identical to the benchmark solutions, demonstrating the effectiveness of the proposed refined 3DPE in accurately predicting lateral deflection of piles with varying slenderness ratios under various

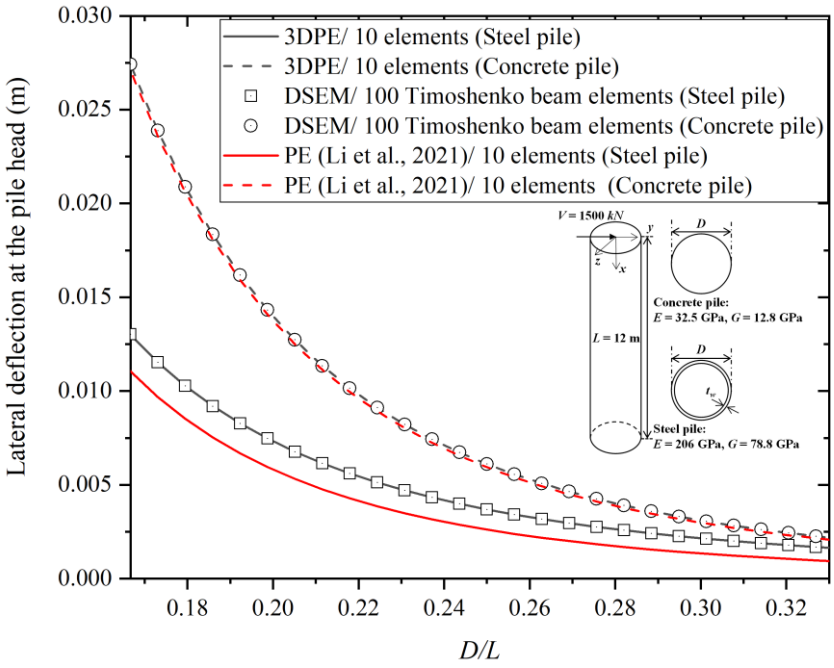
geological conditions.



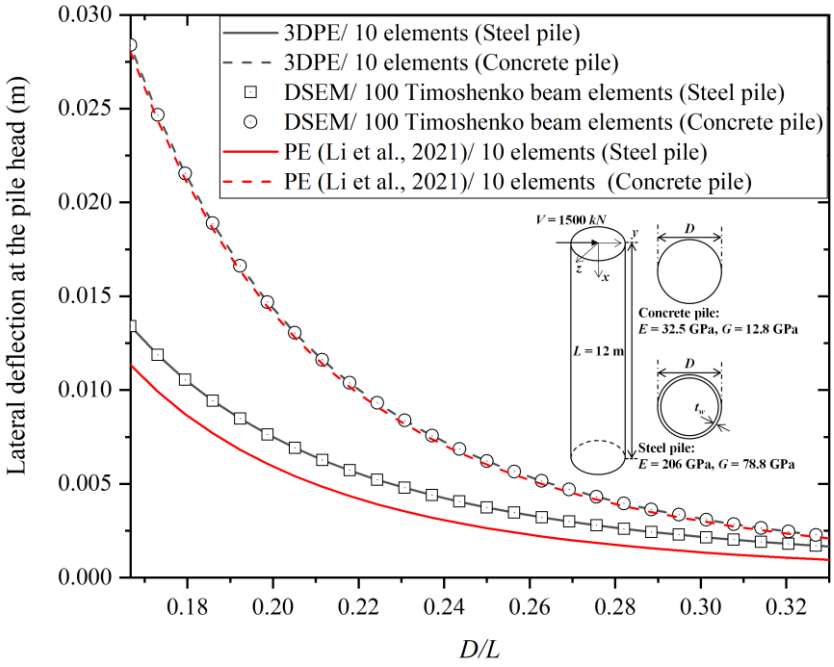
(A)



(B)



(C)



(D)

Figure 3-10 Lateral deflection at the pile head under different soil conditions: (A) Soil Condition 1. (B) Soil Condition 2. (C) Soil Condition 3. (D) Soil Condition 4.

3.4.2 Example 2 – Second-order analysis of a two-story pile-supported frame

After validating the accuracy of the present study in Example 1, the proposed

refined 3DPE is employed for the second-order analysis of pile-supported structures. Also, the corresponding pile-supported structures is also investigated to illustrate how the numerical instability existed in the previous pile element can be eliminated in the proposed refined 3DPE. The upper part of the pile-supported structure is a two-story frame (Figure 3-11) with rectangular cross-section beams and columns measuring 500 x 500 mm, all having an elastic modulus and shear modulus of 32.5 GPa and 13.8 GPa, respectively. The column bottoms of the two-story frame (Noda A to D) are connected to 5 m long concrete piles, with solid circular cross-sections of 0.5 m in radius. The first and second soil conditions in Example 1 (as shown in Figure 3-5A-B) are further employed as the surrounding soil conditions for the piles in the two-story pile-supported structures. The piles have the same elastic and shear modulus as the beams and columns in the upper structure. The loading conditions applied to the upper structure are depicted in Figure 3-11. For the upper structure, four beam elements are employed to model each member. Three modelling methods for the piles in the pile-supported structures are utilized, including: (1) the DSEM uses 50 beam elements for each pile; (2) the proposed refined 3DPE uses 4 elements for each pile; and (3) the existing PE for the spatial analysis proposed by Li et al. [183] uses 4 elements for each pile. Moreover, the common analysis approach in design practice, where the frame is directly fixed on the ground [154], is also employed for comparison. The results from these analysis methods are presented through the load factor versus the lateral deflection about the x- and y-axis, u_x and u_y, of the monitoring point.

Figure 3-12 demonstrates that assuming the upper structure is directly fixed at the column bottom might result in a considerable underestimation of lateral deflection in both directions. The underestimation occurs since the interaction between the upper structures and piles is not taken into account, emphasizing the importance of accurately modeling this interaction in the second-order analysis of pile-supported structures. The analysis results from the DSEM and the proposed refined 3DPE for Soil Condition 1 are nearly identical since the SPI is linear, which is consistent with the results from

Example 1. However, the corresponding computational times differ significantly. All analysis procedures are conducted on a desktop computer outfitted with 32 GB RAM and an Intel Core i7 CPU. The DSEM method requires 60.1 seconds for the analysis procedure, whereas the proposed refined 3DPE method takes only 20.2 seconds, highlighting the efficiency of the proposed approach.

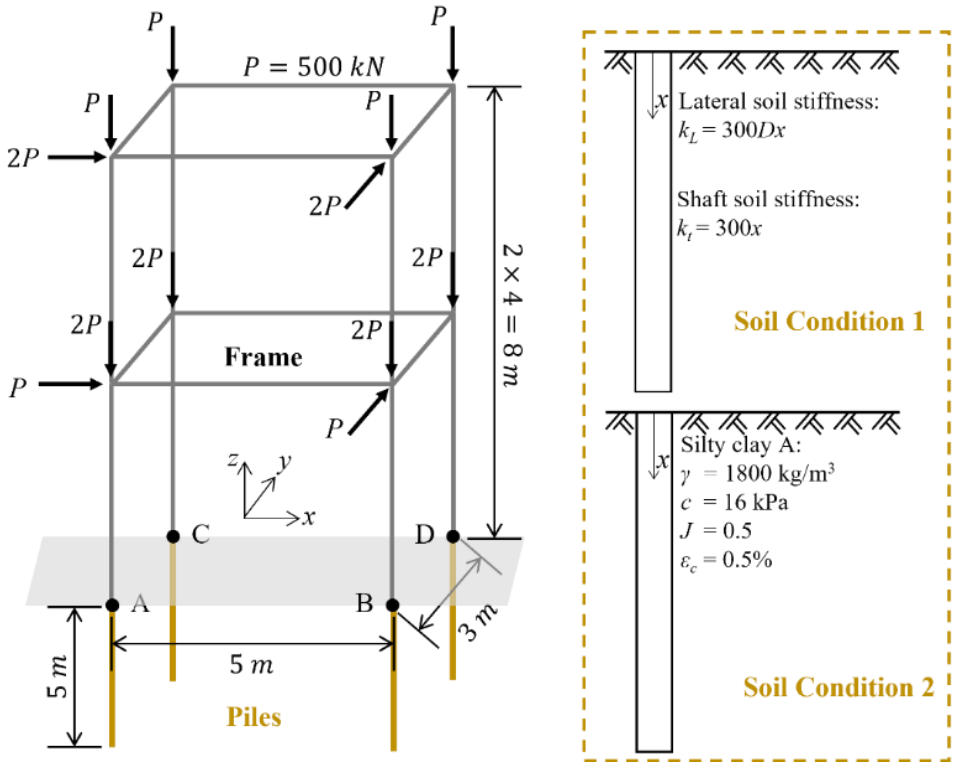
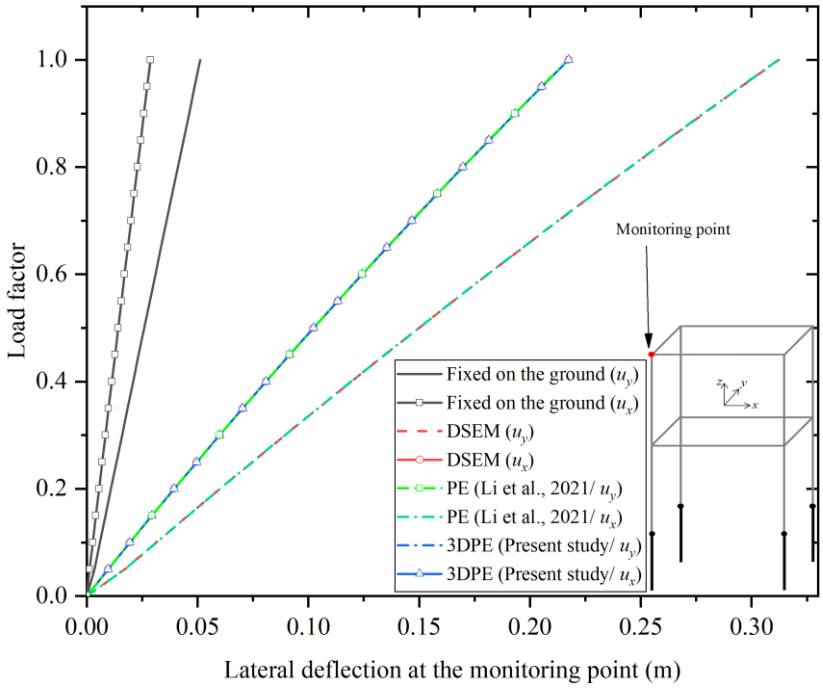
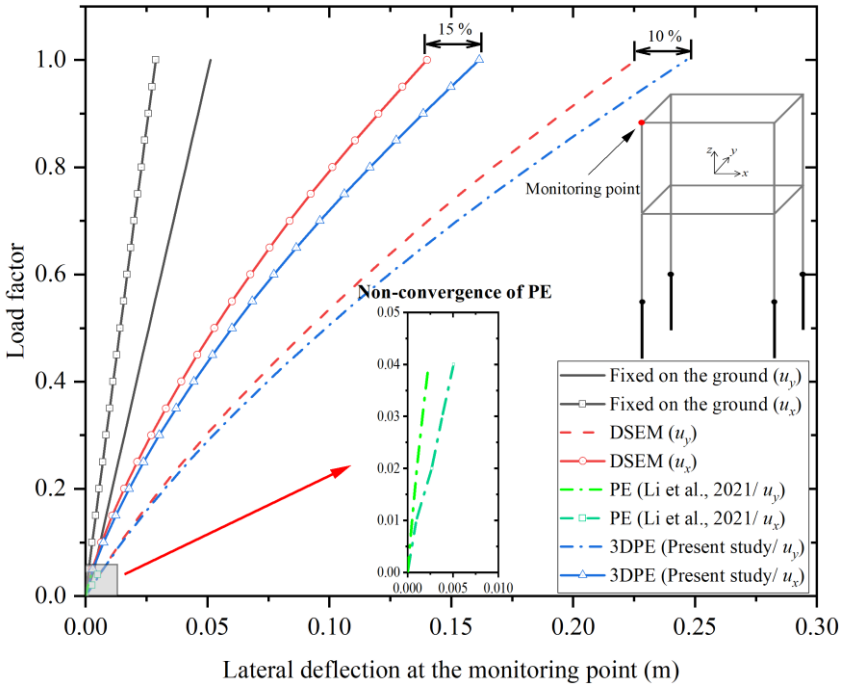


Figure 3-11 Schematic of the pile-supported two-story frame.



(A)



(B)

Figure 3-12 Load factor versus lateral deflection at the monitoring point for the pile-supported structure with different soil conditions: (A) Soil Condition 1. (B) Soil Condition 2.

When considering Soil Condition 2, the DSEM substantially underestimates the deflection of the monitoring point, sometimes exceeding 15%. In addition, the existing PE developed by Li et al. [183] fails to converge (as shown in Figure 3-12B), even when assigned load factor = 0.05 is small. Conversely, the proposed refined 3DPE successfully overcomes numerical instability by formulating the proper soil stiffness matrix, which comprises $\boldsymbol{k}_{p,p}$, $\boldsymbol{k}_{t,k}$, and $\boldsymbol{k}_{t,t}$ neglected in the previous study. Consequently, the importance of the current research for the second-order analysis of pile-supported structures is validated.

3.4.3 Example 3 – Second-order analysis of a twenty-story pile-supported frame

This example further investigates the response of large-scale pile-supported structures considering the nonlinear SPI in spatial analysis. A twenty-story composite frame, shown in Figure 3-13, is selected as the upper structure, which is extensively used in the studies such as [199-201]. In this study, the column bottoms, Nodes A to J, are connected to concrete piles. The lengths of the piles are 30 m. The pile cross-sections, with 1 m in diameter, are solid and circular. The Young' modulus and the shear modulus of the studied pile are 32.5 GPa and 13.8 GPa, respectively. The soil conditions of the studied pile are assumed to be the Soil Condition 4 in Example 1. The corresponding *p*-*y* and *t*-*z* curves can be constructed according to API [155]. The group effect of pile foundations is ignored in this example. The loading condition including a gravity load and a wind load acting in the *y*-direction is adopted [200]. The gravity load is equal to 4.8 kN/m^2, and the wind load equals 0.96 kN/m^2. For simplicity, the frame material is assumed to maintain elasticity in the analysis. For the upper structure, four beam elements are employed to model each member. Different methods are adopted to model the pile-supported structures, including: (1) the frame is directly fixed on the ground; (2) the DSEM uses 50 elements for each pile; and (3) the proposed refined 3DPE uses 4 elements for each pile. The results from the different approaches and past research are present through the load factor versus the deflection of the monitoring

points and Nodes A-D in Figure 3-13.

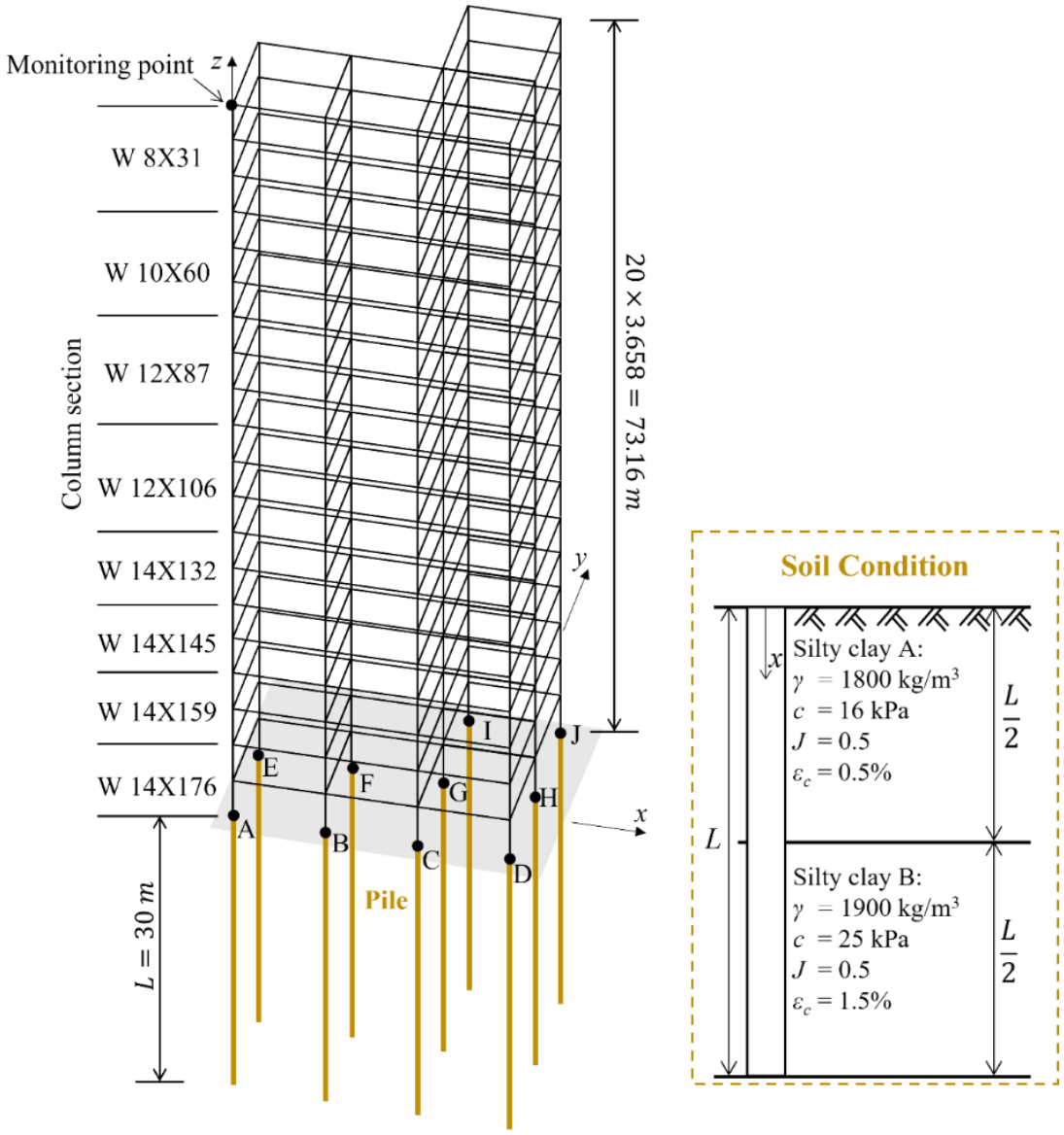
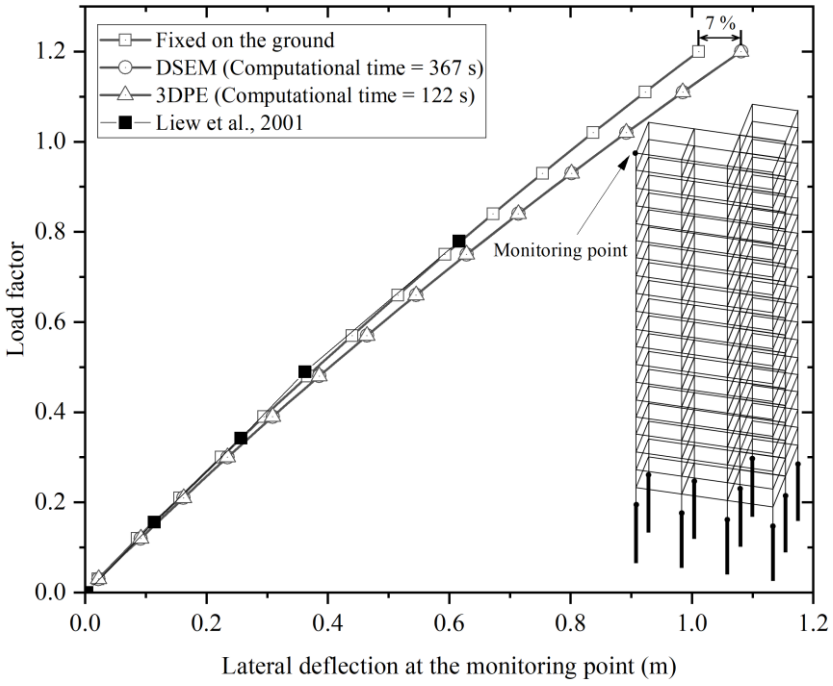
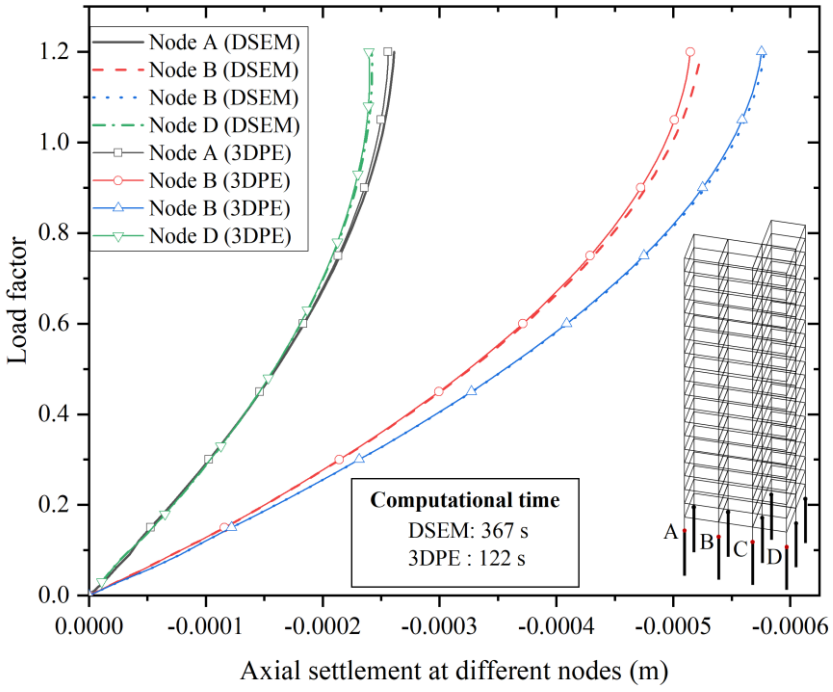


Figure 3-13 Schematic of the pile-supported twenty-story composite frame.



(A)



(B)

Figure 3-14 Load factor versus deflection at different nodes: (A) Lateral deflection at the monitoring point. (B) Axial settlement at the pile heads.

The results in Figure 3-14A indicate that disregarding the interaction between the

upper structure and piles by assuming the upper structure is directly fixed on the ground might lead to the underestimation of the structural deformation. In addition, Figure 3-14B shows that the DSEM will also lead to the underestimation of the settlement as it ignores the axial-torsional coupled response. Furthermore, computational efficiency of the DSEM and the proposed method differ significantly. Specifically, DSEM requires approximately 367 s to complete the calculation on a desktop computer with 32 GB RAM and one Intel Core i7 CPU, while the proposed refined 3DPE method only takes 122 s to complete the computational procedure on the same computer, which represents only 33% of the computational time required by DSEM in the spatial analysis, further validating the computational efficiency of the proposed refined 3DPE.

Appendix A - Shape Function Coefficients

The coefficients in the element shape function are provided here:

$$\alpha_{x,1}(x) = 1 - \frac{x}{L}$$

$$\alpha_{x,2}(x) = \frac{x}{L}$$

$$\alpha_{y,1}(x) = 1 - b_z\kappa_z\frac{x}{L} - 3\kappa_z\left(\frac{x}{L}\right)^2 + 2\kappa_z\left(\frac{x}{L}\right)^3$$

$$\alpha_{y,2}(x) = \left[(2+b_z)\frac{\kappa_z}{2}\frac{x}{L} - (4+b_z)\frac{\kappa_z}{2}\left(\frac{x}{L}\right)^2 + \kappa_z\left(\frac{x}{L}\right)^3\right]L$$

$$\alpha_{y,3}(x) = b_z\kappa_z\frac{x}{L} + 3\kappa_z\left(\frac{x}{L}\right)^2 - 2\kappa_z\left(\frac{x}{L}\right)^3$$

$$\alpha_{y,4}(x) = -\left[b_z\frac{\kappa_z}{2}\frac{x}{L} + (2-b_z)\frac{\kappa_z}{2}\left(\frac{x}{L}\right)^2 - \kappa_z\left(\frac{x}{L}\right)^3\right]L$$

$$\alpha_{z,1}(x) = 1 - b_y\kappa_y\frac{x}{L} - 3\kappa_y\left(\frac{x}{L}\right)^2 + 2\kappa_y\left(\frac{x}{L}\right)^3$$

$$\alpha_{z,2}(x) = -\left[(2+b_y)\frac{\kappa_y}{2}\frac{x}{L} - (4+b_y)\frac{\kappa_y}{2}\left(\frac{x}{L}\right)^2 + \kappa_y\left(\frac{x}{L}\right)^3\right]L$$

$$\alpha_{z,3}(x) = b_y\kappa_y\frac{x}{L} + 3\kappa_y\left(\frac{x}{L}\right)^2 - 2\kappa_y\left(\frac{x}{L}\right)^3$$

$$\alpha_{z,4}(x) = \left[b_y\frac{\kappa_y}{2}\frac{x}{L} + (2-b_y)\frac{\kappa_y}{2}\left(\frac{x}{L}\right)^2 - \kappa_y\left(\frac{x}{L}\right)^3\right]L$$

$$\alpha_{\theta,1}(x) = 1 - \frac{x}{L}$$

$$\alpha_{\theta,2}(x) = \frac{x}{L}$$

where:

$$b_i = \frac{12EI_i}{Gk_iAL^2} \quad (i = y \quad or \quad z)$$

$$\kappa_i = \frac{1}{1+b_i} \quad (i = y \quad or \quad z)$$

in which, E and G are the Young's modulus and the shear modulus of the pile martial, respectively; A and I denote the cross-section area and the inertia moment of the pile, respectively; and κ is the shear correction factor of the pile cross-section.

Chapter 4. NEURAL-OPERATOR ELEMENT METHOD FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS

4.1 Introduction

Efficient and scalable numerical simulation methods are crucial for both scientific research and engineering practice. Extensive efforts since the last century have led to the development of various simulation techniques, which can broadly be categorized into two groups: (1) mesh-based methods [1, 202, 203], such as the finite element method (FEM), and (2) mesh-free methods [204-208], such as those based on machine learning (ML). Despite these advancements, achieving accuracy, efficiency, and scalability in large-scale simulations of complex multiscale systems remains significantly challenging.

In many science and engineering disciplines, FEM is a popular numerical method for solving partial differential equations (PDEs), due to its versatility and robustness [209-212]. However, when it comes to systems exhibiting, for example, multiscale features or defined on complex geometries (e.g., non-polygon shapes), FEM requires a large number of elements (mesh) (Figure 4-1A) to produce accurate solutions [1, 213]. Such a mesh-dependent nature makes FEM computationally expensive and limits its applicability for large-scale problems. While considerable efforts have been made to address this challenge, most involve a trade-off between computational efficiency and accuracy [214, 215], or are only applicable for a special class of problems [216-221].

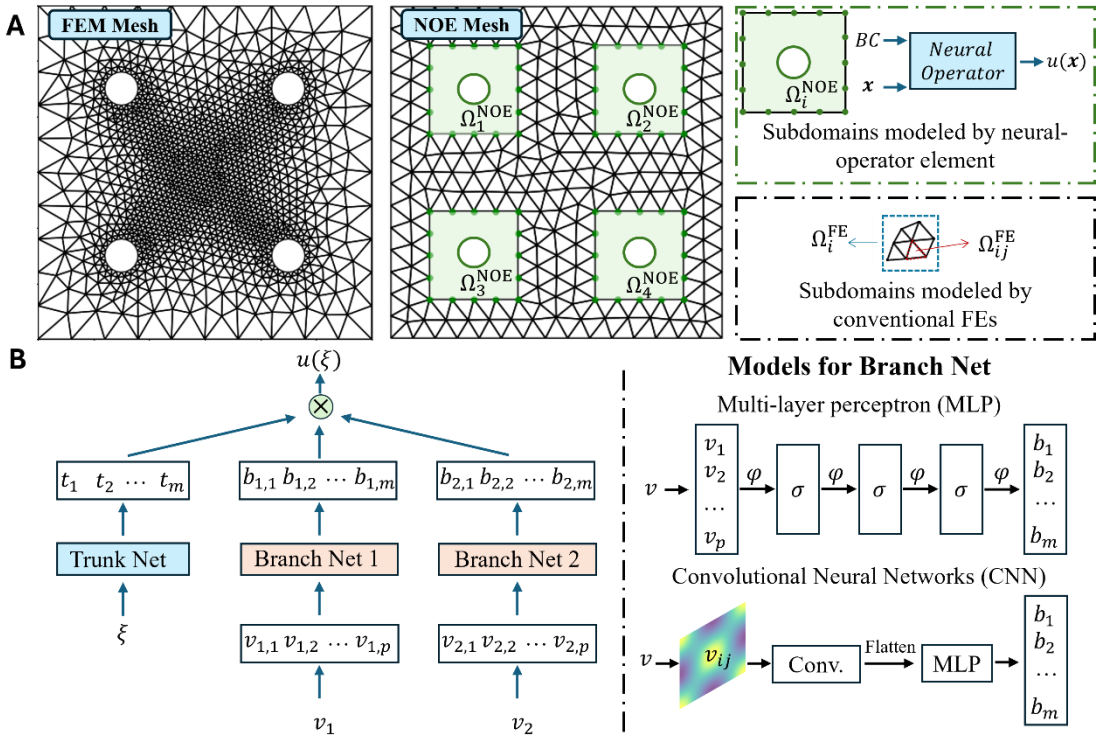


Figure 4-1 Illustration of NOEM: (A) Comparison between FEM and NOEM. (B) Illustration of the neural operator models via the multiple-input deep operator network (MIONet). MIONet is simplified to the deep operator network (DeepONet) when only one branch network is employed. Different models can be used for the branch network, such as multi-layer perceptron (MLP) and convolutional neural networks (CNN).

On the other hand, operator learning has emerged as a new ML framework that aims to learn mappings between infinite-dimensional function spaces [133]. It differentiates itself from traditional ML tasks, such as function approximation which only learns mappings between finite-dimensional spaces. Due to its close connections to PDE solution operators, it has garnered huge attention from the scientific computing community, and many neural network models for operator learning, namely neural operators (NOs), have been proposed. Among them, deep operator networks (DeepONets) [132], as the first NO model, are inspired by the universal operator approximation theorem [133]. DeepONets have demonstrated superior performance in

many challenging tasks [132, 222-226]. In addition, the inherent structure from the universal approximation theorem offers significant flexibility in designing sub-architectures. Despite empirical successes, operator learning faces multiple challenges for large-scale simulations [121, 227], such as high-fidelity data generation, demanding training costs, and reusability. To overcome the limitations of the pure data-driven nature, physics-informed ML (PIML) has the potential to circumvent the need for a large amount of data by incorporating PDEs into the loss [206, 208, 228]. However, physics-informed training remains difficult as the loss involved with PDEs often incurs new challenges [229-231], especially for large-scale complex systems.

In the present work, we propose a hybrid method, namely, the neural-operator element method (NOEM; Figure 4-1A). The motivation is to combine the strengths of FEM and operator learning to mitigate their limitations. The underlying idea of NOEM is to represent the PDE solution using both finite elements and neural-operator elements (NOEs). NOEs are constructed by pretrained NOs and assigned to parts of computational domains where a large number of finite elements would be required if FEM was used. Each NOE is assigned to a specific non-overlapping subdomain, and NOEM solves the energy norm minimization problem, which is similar to FEM. The adoption of NOEs substantially reduces the complexity of the problem, enabling NOEM to be applied effectively to large-scale problems. Extensive numerical examples are performed to validate the proposed NOEM and illustrate the key features of the NOEM (Table 4-1).

Hybridizing FEM and ML is not new and has been explored in the literature. For example, Refs. [232-235] use piecewise polynomials and weak-form residuals in designing ML architectures and physics-informed loss functions, which are found to be effective to some extent. However, the reusability of these models remains elusive, especially for large-scale systems. Yin et al. [236] has integrated NOs into the domain decomposition (DD) approach of FEM by modeling different regions separately with

NOs and finite elements. Chung et al. [237] also embedded learned reduced-order models (ROM) within the discontinuous Galerkin domain decomposition (DG-DD) framework, enabling efficient simulation by applying ROMs to unit components and interfaces. Despite these advances, modeling large-scale systems with multiple subdomains using DD is still challenging, as it commonly requires tedious iteration processes and may fail to converge if certain parameters are not carefully chosen [236, 237].

What sets NOEM asides from existing hybrid methods are (1) the approach of learning NO elements that cover a large subdomain and (2) the use of standard FEs for subdomains that FEM works well. Particularly, this feature is preferred in industrial applications. The contemporary industry practice frequently employs a "standardized product module" philosophy, constructing large-scale systems by interconnecting standardized modules or components, e.g., modular integrated construction [238]. In this context, NOEM offers pre-trained NO libraries, i.e., NOEs, tailored for simulating standardized modular components across various projects. Additionally, the inherent differentiability of NOEs provides differentiable modeling approach for large-scale systems. While not the focus of this chapter, such differentiable feature holds significant potential for optimal design tasks.

Table 4-1 Summary of key features and results

Features		Description & Key results
Compared to FEM	Efficiency	*Reduce computational time* • Example 3.2: Boosted computational speed by around 14 times over FEM
	Scalability	*Support effective large-scale domain modeling* • Example 3: Analysis across varying domain sizes, scaling up by 100 times
Compared to NO	Training Efficiency	*Require smaller training datasets and faster training processes than direct modeling of entire systems using NOs*
	Model Reusability	*Directly reuse pretrained NOs across varied problems* • Example 1: Single NO for 160 problems of varying coefficient functions • Example 3: Single NO for 200 problems with varying geometries
Others	ML Compatibility	*Achieve compatibility with different NO models* • Example 1-4: Various utilized NOs, such as DeepONet, MIONet, and CNN
	PDE Applicability	*Enable application to various PDE problems* • Example 2: Multiscale problems • Example 4.4: Nonlinear problems

4.2 Preliminaries

In this section, we first briefly revisit several NOs, including DeepONet and its variants, and then review FEM, covering both the standard weak form and the variational form. Finally, we discuss the enforcement of hard constraints on NOs under varying boundary conditions.

4.2.1 Neural operators

Let $\mathcal{G}: V \to H$ be an operator of interest where V and H are the spaces of the input and output functions, respectively. Each function in V and H is real-valued defined on $\Omega' \subset \mathbb{R}^{d'}$ and $\Omega \subset R^d$, respectively. The goal of operator learning is to construct a NO, $\mathcal{G}_{\mathrm{N}}$, from a set of training data consisting of pairs $\left(v_j, u_j = \mathcal{G}[v_j]\right)$ in discretized forms. For the sake of simplicity, we slightly abuse notation and refer to a discretized form of v as v. We revisit three DeepONet variants: the vanilla DeepONet [132], the convolutional DeepONet [141, 239], and the multi-input DeepONet (MIONet) [240]. These NOs are introduced and employed in the present study given their superior generalization capability [222-224].

Vanilla DeepONet

DeepONet [132] constructs $\mathcal{G}_{\mathrm{N}}$ through two distinct sub-networks: a branch and a trunk network (Figure 4-1B). The branch network takes a discretized function v as input and outputs a vector $\boldsymbol{b}(v) = [b_0, b_1(v), \ldots, b_m(v)]^{\mathsf{T}}$, and the trunk network takes a point $x \in \Omega$ as input and outputs a vector $\boldsymbol{t}(x) = [1, t_1(x), \ldots, t_m(x)]^{\mathsf{T}}$.

DeepONet is then represented by the Euclidean inner product of these outputs:

$$\mathcal{G}_{\mathrm{N}}[v](x) = \langle \boldsymbol{b}(v), \boldsymbol{t}(x) \rangle = \sum_{i=1}^{m} b_i(v) t_i(x) + b_0$$

MLPs are often used as the default for both subnetworks. If this is the case, we refer to DeepONet as vanilla DeepONet.

Convolutional DeepONet

Convolutional DeepONet [141, 239] is a variant of vanilla DeepONet, which uses a convolutional neural network (CNN) as a branch network (Figure 4-1B). This variant is found to be particularly effective when the discretized input function has a substantially large size and has an image-like shape.

Multi-input DeepONet

MIONet [240] is designed to learn an operator that takes several input functions. Similar to DeepONet, the architecture of MIONet is based on the universal operator approximation theorem, which extends from single to multiple Banach spaces. The operator of interest is $\mathcal{G}: V_1 \times \cdots \times V_n \to H$, where $V_1, \ldots, V_n$ are n input function spaces and each $v_i \in V_i$ is a real-valued function defined on $\Omega_i' \subset R^{d_i'}$. To handle multiple inputs, MIONet employs multiple branch networks, each corresponding to a distinct input function space, and processes them independently while keeping a single trunk network.

$$\mathcal{G}_{\mathrm{N}}[v_1, v_2, \ldots, v_n](x) = \sum_{i=1}^{m} \left(\prod_{j=1}^{n} b_{i,j}(v_j) \right) t_i(x) + b_0$$

where $b_{i,j}$represents the i^{th} output feature from the j^{th} branch network.

4.2.2 Hard-constraint neural operators with varying boundary conditions

Let us consider a general PDE defined in Ω with a Dirichlet boundary condition:

$$u = g \quad \text{on } \partial\Omega. \tag{4.1}$$

For the simplicity of discussion, let the PDE solution operator from Dirichlet boundary conditions be $\mathcal{G}: g \mapsto u$. It is often desirable to construct NOs that exactly satisfy the imposed boundary conditions. We refer to such NOs as hard-constraint NOs, which are proven to be effective in some problems [241, 242].

Let α be an auxiliary function satisfying $\alpha(x) = 0$ if $x \in \partial\Omega$ and $\alpha(x) > 0$ if $x \notin \partial\Omega$, and let β_g be an extension of g, i.e., $\beta_g(x) = g(x)$ if $x \in \partial\Omega$. Then hard-constraint NOs can be constructed by

$$\mathcal{G}_{\mathrm{N}}^{hc}[g](x) = \alpha(x)\mathcal{G}_{\mathrm{N}}[g](x) + \beta_g(x) \quad \text{for } x \in \Omega \tag{4.2}$$

where $\mathcal{G}_{\mathrm{N}}[g](x)$ represents a NO model with learnable parameters. However, the construction of α and β_g could be challenging. We therefore consider an approximation approach where β_g is approximated by a mesh-based polynomial p_g constructed from a set of discretized boundary data$\{x_i \in \partial\Omega, g(x_i)\}_{i=1}^{m}$. Appendix A presents the details on constructing α and β_g.

4.2.3 Finite element method

This section briefly reviews FEM. Suppose that the PDE has a variational form:

$$\text{Find } u \in H, \text{ such that} \quad a(u, v) = L(v) \quad \forall\, v \in H_0 \tag{4.3}$$

where $a(\cdot,\cdot)$ and $L(\cdot)$ are the bilinear form and the linear functional, respectively, and H_0 is a test function space. The Lax–Milgram lemma guarantees the existence and uniqueness of the solution to Eq. (4.3) under certain assumptions on a and L . Furthermore, we assume that Eq. (4.3) is equivalent to the energy minimization problem:

$$u = \arg\min_{w \in H} J[w] \coloneqq \frac{1}{2}\, a(w, w) - L(w) \tag{4.4}$$

Let Ω be the computational domain of interest. We consider $\{\Omega_i^{\mathrm{FE}}\}_i$ be a partition of Ω. Furthermore, we can let $\{\Omega_{ij}^{\mathrm{FE}}\}_j$ be a partition of Ω_i^{FE} , where $\Omega_{ij}^{\mathrm{FE}}$ is commonly selected as the polygon with regular geometry. In general, FEM represents an approximation to the solution on Ω_i^{FE} as a linear combination of shape functions $\{p_{ij}^{\mathrm{FE}}\}_j$:

$$\phi_i^{\mathrm{FE}}(x;\boldsymbol{\alpha_i}) = \sum_j \alpha_{ij} p_{ij}^{\mathrm{FE}}(x) \tag{4.5}$$

where $\boldsymbol{\alpha_i} = \{\alpha_{ij}\}_j$ is the collection of α_{ij} ; and each function p_{ij}^{FE} is supported on $\Omega_{ij}^{\mathrm{FE}}$, i.e., vanishing outside of $\Omega_{ij}^{\mathrm{FE}}$. As illustrated in in Figure 4-1A, p_{ij}^{FE} could be the piecewise polynomial defined on the triangular area $\Omega_{ij}^{\mathrm{FE}}$. In our numerical experiments, p_{ij}^{FE} is selected as the piecewise linear function defined on either one-dimensional line segment or two-dimensional triangular area.

Then, the FEM solution defined on the entire computational domain Ω can be represented via

$$q(x;\{\boldsymbol{\alpha_i}\}_i) = \sum_i \phi_i^{\mathrm{FE}}(x;\boldsymbol{\alpha_i}) \tag{4.6}$$

where $\{\boldsymbol{\alpha_i}\}_i$ is then obtained by solving Eq. (4.4):

$$\{\boldsymbol{\alpha_i}\}_i^* = \arg\min_{\{\boldsymbol{\alpha_i}\}_i} J\left[q(\cdot;\{\boldsymbol{\alpha_i}\}_i)\right]$$

4.3 Method

We consider the following PDE defined on a domain Ω:

$$\mathcal{D}[u](x) = 0, \quad \text{for } x \in \Omega \tag{4.7}$$

where $\mathcal{D}$ is the underlying differential operator of the PDE. In NOEM, we partition Ω into two non-overlapping subdomains Ω_{FE} and Ω_{NOE}, i.e.,

$$\Omega = \Omega^{\mathrm{FE}} \cup \Omega^{\mathrm{NOE}} \tag{4.8}$$

where $\Omega^{\mathrm{FE}} = \{\Omega_i^{\mathrm{FE}}\}_{i\in I_{\mathrm{FE}}}$ is a subdomain represented by classical FE meshes, and $\Omega^{\mathrm{NOE}} = \{\Omega_j^{\mathrm{NOE}}\}_{j\in I_{\mathrm{NOE}}}$ is represented by NOE. For instance, in Figure 4-1A, the FE subdomain Ω^{FE} is represented by the black grid.

The subdomain Ω^{NOE} modeled by the NOEs (visualized by the green areas) is further partitioned into four subdomains as $\Omega^{\text{NOE}} = \{\Omega_j^{\text{NOE}}\}_{j\in\{1,2,3,4\}}$, where each Ω_j^{NOE} is modeled by one single NOE. In the following content, we first introduce how to construct NOEs by NOs. Next, we present the NOEM framework for solving PDEs. Finally, a discussion of the NOEM is provided.

4.3.1 NOE represented by a neural operator

Assume that the Dirichlet boundary condition is sufficient to guarantee the existence and uniqueness of the solution to Eq. (4.7). For each subdomain modeled by the NOE, Ω_j^{NOE}, we assume that a sufficiently trained NO exists to approximate the Dirichlet-to-solution operator $\mathcal{G}_\dagger: g \to u$ where u is the solution to

$$\mathcal{D}[v](x) = 0, \quad \text{for } x \in \Omega_j^{\text{NOE}} \tag{4.9}$$

with the Dirichlet boundary condition of g on $\partial\Omega_j$. Here, $\mathcal{D}$ is the same differential operator as in Eq. (4.7) defined in the subdomain Ω_j^{NOE}. Thus, if the Dirichlet condition is restricted on $\partial\Omega_j^{\text{NOE}}$, we have $\mathcal{G}_\dagger\left[u|_{\partial\Omega_j^{\text{NOE}}}\right] = u|_{\Omega_j^{\text{NOE}}}$where $u|_{\partial\Omega_j^{\text{NOE}}}$ and $u|_{\Omega_j^{\text{NOE}}}$ represent the solution u restricted on $\partial\Omega_j^{\text{NOE}}$ and Ω_j^{NOE}, respectively.

The Dirichlet boundary condition for the NO input is parametrized by the sensor values, $\boldsymbol{\beta}_j$, whose sensor locations are aligned with the nodes determined by the adjacent elements, as illustrated by the green dots in Figure 4-1. Notably, Eq. (4.9) is a significantly smaller system compared with the full PDE in Eq.(4.7) and is assumed to be efficiently solved by FEM (or any other numerical methods) for varying boundary conditions, indicating the training cost of NOs for such smaller system will be significantly lower than directly modeling entire computational domains.

For simplicity of discussion, we suppress the trainable parameters of NOs and

assume NOs are pre-trained. Further details about the training process are briefly presented in Appendix B.

4.3.2 Neural-operator element method

The underlying principle of NOEM is the replacement of a large number of finite elements by means of a single NO, i.e., NOE. Thus, NOEM represents the solution as

$$q(x;\boldsymbol{c}) = \sum_{i\in I_{\mathrm{FE}}} \phi_i^{\mathrm{FE}}(x;\boldsymbol{\alpha_i}) + \sum_{j\in I_{\mathrm{NOE}}} \phi_j^{\mathrm{NOE}}\left(x;\boldsymbol{\beta_j}\right) \tag{4.10}$$

where $\boldsymbol{c} = \{\boldsymbol{\alpha_i}\} \cup \{\boldsymbol{\beta_j}\}$; $\phi_i^{\mathrm{FE}}(x;\boldsymbol{\alpha_i})$ is the standard FE representation in Eq. (4.5) whose support is Ω_i^{FE}; and $\phi_j^{\mathrm{NOE}}\left(x;\boldsymbol{\beta_j}\right)$ of the support Ω_j^{NOE} is the NOE formulated via the NO as:

$$\phi_j^{\mathrm{NOE}}\left(\cdot;\boldsymbol{\beta_j}\right) = \mathcal{G}_{\not\theta}\left[u|_{\partial\Omega_j^{\mathrm{NOE}}}\right](\cdot)$$

where $\boldsymbol{\beta_j}$ is the discrete representation of $u|_{\partial\Omega_j^{\mathrm{NOE}}}$ as defined in Section 3.1. A simple example for constructing $q(x;\boldsymbol{c})$ can be found in Section 4.1 for illustration.

Noted that for each $j \in I_{\mathrm{NOE}}$, we assume that if FEs were used for Ω_j^{NOE}, a large number of elements would be needed. Importantly, the number of degrees of freedom, $\boldsymbol{\alpha_j'}$, increases proportionally with the required number of finite elements, thereby increasing computational complexity. However, this is not the case for NOEs, as they take the parametrized boundary conditions $\boldsymbol{\beta_j}$ as inputs whose cardinality is significantly smaller than that of $\boldsymbol{\alpha_j'}$. The NOEM solution $q(\cdot;\boldsymbol{c}^*)$ is then obtained by solving Eq. (4.4) in the mixed element space, i.e.,

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} J\left[q(\cdot;\boldsymbol{c})\right] \tag{4.11}$$

whose complexity is significantly reduced compared to the full FEM. Yet, the

optimization problem in Eq. (4.11) is no longer convex. While it is not guaranteed to find a global minimizer, extensive numerical experiments indicate that it does not incur severe issues in this regard.

NOEM solves the energy minimization in Eq. (4.11) using the solution representation of Eq. (4.10), which requires numerical integration for implementation. Typically, the terms involving only FEs can be exactly computed due to the exactness of quadrature. Thus, we only apply numerical quadrature rules (e.g., the Monte-Carlo method) to approximate the NOE involved terms. Afterwards, we employ Newton's method (as presented in Appendix B) to solve Eq. (4.11).

4.3.3 Discussion

Generally, the inputs of NOs, $\mathcal{G}_j$, are required to incorporate the boundary conditions, but they can also include additional functions, such as the source term and the coefficient function. In this case, the NOs, $\mathcal{G}_j: \left(u|_{\partial\Omega_j^{\mathrm{NOE}}}, f\right) \to u|_{\Omega_j^{\mathrm{NOE}}}$, approximate the operators that map the boundary conditions and the additional function of interest, f, to the corresponding solution. And the additional NO input, f, will be determined by the PDE of interest in Eq. (4.7) and fixed during the NOEM solving process for each Ω_j^{NOE}.

We remark that the present work focuses on the case where NOEs take the Dirichlet boundary conditions as inputs. Additionally, for some PDEs where the Dirichlet boundary condition is insufficient, such as the biharmonic equation, the boundary conditions as the input of NOs constructing the NOE should be modified accordingly.

In this chapter, given the translation invariance of the PDEs, the NOEM in our numerical experiments requires only one single pretrained NO for each PDE, which can be adapted to each specific Ω_j^{NOE} through translation. However, users can also include multiple NOs for different Ω_j^{NOE} when these NOE-modeled subdomains care

characterized by significantly different properties or geometry.

In order for the NOEM solution in Eq. (4.10) to be consistent with the theory of PDEs and FEM, Eq. (4.10) is required to satisfy a regularity condition, such as C^0 or C^1 continuities. Therefore, in principle, the hard-constraint NOs in Eq. (4.2) should be used for NOEs. However, constructing them poses another computational challenge and costs in general. If this is the case, standard NOs can be used, which yields satisfactory performance in our numerical experiments. This resembles the discontinuous Galerkin approach, which will be covered in future work.

4.4 Validation Examples

In this section, ten examples in four groups are conducted to validate the performance of the proposed NOEM. For pedagogical purposes, the first example presents a simple ODE problem to illustrate the implementation process of the NOEM. The second example demonstrates the potential of the NOEM in addressing multi-scale problems and illustrates its scalability via the strong linear correlation between NO performance and NOEM accuracy. The third example is designed to illustrate the reusability of the NO models within the NOEM. Additionally, Example 3 also demonstrates the computational efficiency and scalability of the proposed method on two-dimensional problems. Finally, Example 4 investigates different types of Darcy flow problem, showcasing the ability of the NOEM to handle both linear and nonlinear problems. The illustrated features of the proposed NOEM and example results are summarized in Table 4-1. And certain details for training NOs are included in Appendix C.

4.4.1 Pedagogical example

To validate the performance of the proposed NOEM, a 1D elliptic ODE is analyzed. The governing equation and boundary conditions are detailed as follows:

$$\frac{d}{dx}\left(k(x)\frac{du}{dx}\right) = 0, \quad x \in \Omega,$$

$$u(x) = 0, \quad \text{at } x = 0, \tag{4.12}$$

$$\frac{du}{dx} = 0.1, \quad \text{at } x = 3.$$

$$k(x) = \begin{cases} 1 & \text{for } x \in [0,1) \cup (2,3] \\ k'(x) & \text{for } x \in [1,2] \end{cases}$$

where $\Omega = [0,3]$; and $k'(x)$ is the coefficient function selected as different nonlinear functions in this example.

For the interval $[0,1)$ and $(2,3]$, we can use only one conventional 1D linear FE for each interval to accurately recover the solutions. However, for the middle segment, $x \in [1,2]$, the conventional FEM requires a dense mesh to accurately recover the solution. Thus, for the first and last segments, only one conventional 1D element is used for each segment. As shown in Figure 4-2A, two different modeling methods are used for $x \in [1,2]$ as follows: (1) the conventional FEM using 100 FEs; and (2) the proposed NOEM using one NOE formulated through the pretrained NO model.

This example is designed to provide a simple example to illustrate the implementation process of the proposed NOEM for pedagogical purposes. Here, $k'(x)$ is defined as a quadratic function, $k'(x) = 1 + 3(x-1)(x-2)$. To model the middle segment, $x \in [1,2]$, a DeepONet is trained to approximate the mapping from the Dirichlet boundary conditions of the following PDEs to the corresponding solutions as:

$$\frac{d}{dx}\left(k'(x)\frac{du_{\mathcal{G}}}{dx}\right) = 0, \quad \text{for } x \in [1,2],$$

$$u_{\mathcal{G}}(x) = u_{B,1}, \quad \text{at } x = 1,$$

$$u_{\mathcal{G}}(x) = u_{B,2}, \quad \text{at } x = 2.$$

$$\mathcal{G}: \boldsymbol{u}_{\Omega_{\text{NOE}}} \mapsto u_{\mathcal{G}}$$

where $\boldsymbol{u}_{\Omega_{\text{NOE}}} = \{u_{B,1}, u_{B,2}\}$ denotes the boundary condition.

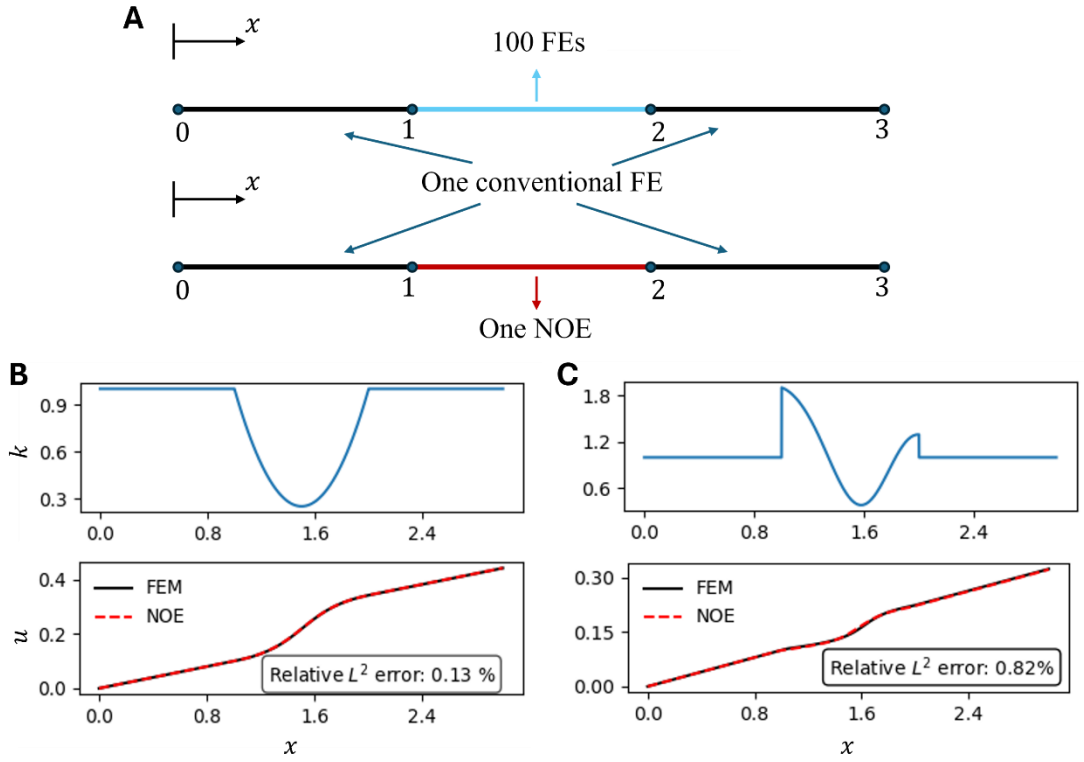


Figure 4-2 Modeling methods and analysis results in Example 1: (A) Two modeling methods used in Example 1 are plotted in the left side. The first and last segments (the black lines) are both modeled by one conventional 1D linear FE. The conventional FEM and the proposed NOEM use 100 elements and one NOE to model the middle segment, respectively. The comparison results in Example 1.1 and Example 1.2 are plotted in (B) and (C), respectively. The coefficient functions, k, and the solutions from different methods are visualized in the upper and lower subplots, respectively.

Accordingly, the NOEM only requires four DOFs to model the domain as: $\boldsymbol{c} = \{u_i\}_{i=0}^{3}$, where u_i corresponding to the values at the mesh points, $\boldsymbol{x} = \{x_i = i\}_{i=0}^{3}$ (illustrated as blue dots in Figure 4-2A). $\{u_0, u_1\}$ and $\{u_2, u_3\}$ represent the element DOFs for the conventional FEs modeling the first and last segments,

respectively. $\{u_1, u_2\}$ corresponds to the DOFs for the NOE, which also serve as the input, $\boldsymbol{u}_{\Omega_{\text{NOE}}}$, to the branch net when formulating the NOE using the pretrained DeepONet. More specifically, according to Eq. (4.10), the NOEM represents the solution as:

$$q(x;\boldsymbol{c}) = \phi_1^{\text{FE}}(x;\boldsymbol{\alpha_1}) + \phi_1^{\text{NOE}}(x;\boldsymbol{\beta_1}) + \phi_2^{\text{FE}}(x;\boldsymbol{\alpha_2})$$

where $\boldsymbol{\alpha_1} = \{u_0, u_1\}$ and $\boldsymbol{\alpha_2} = \{u_2, u_3\}$ are the coefficients parametrized the FE representation; and $\boldsymbol{\beta_1} = \{u_1, u_2\}$ parametrizes the NOE representation. And ϕ_1^{FE}, ϕ_2^{FE}, and ϕ_1^{NOE} can be given as

$$\phi_1^{\text{FE}}(x;\boldsymbol{\alpha_1}) = \begin{cases} (u_1 - u_0)x + u_0 & \text{for } x \in \Omega_1^{\text{FE}} \\ 0 & \text{for } x \in \Omega \setminus \Omega_1^{\text{FE}} \end{cases}$$

$$\phi_2^{\text{FE}}(x;\boldsymbol{\alpha_2}) = \begin{cases} (u_3 - u_2)(x-2) + u_2 & \text{for } x \in \Omega_2^{\text{FE}} \\ 0 & \text{for } x \in \Omega \setminus \Omega_2^{\text{FE}} \end{cases}$$

$$\phi_2^{\text{FE}}(x;\boldsymbol{\beta_1}) = \begin{cases} \mathcal{G}[\boldsymbol{\beta_1}](x) & \text{for } x \in \Omega_1^{\text{NOE}} \\ 0 & \text{for } x \in \Omega \setminus \Omega_1^{\text{NOE}} \end{cases}$$

where Ω_1^{FE}, Ω_2^{FE}, and Ω_1^{NOE} are $[0,1]$, $[2,3]$, and $[1,2]$, respectively.

The comparison results are presented in Figure 4-2B. It can be found that the NOEM can achieve great accuracy with only around 0.1% relative error even when using only one NOE to model the middle segment with the nonlinear coefficient function.

Then, we further test the proposed method on Eq. (4.12) with $k'(x)$ generated from a Gaussian process (GP) expressed as:

$$k'(x) = \exp\big(v(x)\big)$$

$$v(x) \sim \mathcal{GP}\big(0, k_l(x, x')\big)$$

$$k_l(x, x') = -\frac{1}{2}\left(\frac{x - x'}{l}\right)^2$$

where $\mathcal{GP}$ denotes a GP with the kernel function k_l; and l is the correlation length equal to 0.3.

To formulate the NOE, a MIONet, $\mathcal{G}: k', \boldsymbol{u}_{\Omega_{\text{NOE}}} \mapsto u_{\mathcal{G}}$, is trained to approximate the solution of the following ODE as:

$$\begin{gathered} \frac{d}{dx}\left(k'(x)\frac{du_{\mathcal{G}}}{dx}\right) = 0, \quad \text{for } x \in [1,2] \\ u_{\mathcal{G}}(1) = u_{B,1}, \quad u_{\mathcal{G}}(2) = u_{B,2} \\ k'(x) = \exp\big(v(x)\big), \quad v(x) \sim \mathcal{GP}\big(0, k_l(x,x')\big) \end{gathered} \tag{4.13}$$

The FEM and the proposed NOEM are employed to compute the solution, whose modeling approaches maintain the same as plotted in Figure 4-2A. The generated k' is plotted in Figure 4-2C with the comparison results of 0.82% relative error, which further showcase the performance of the proposed method.

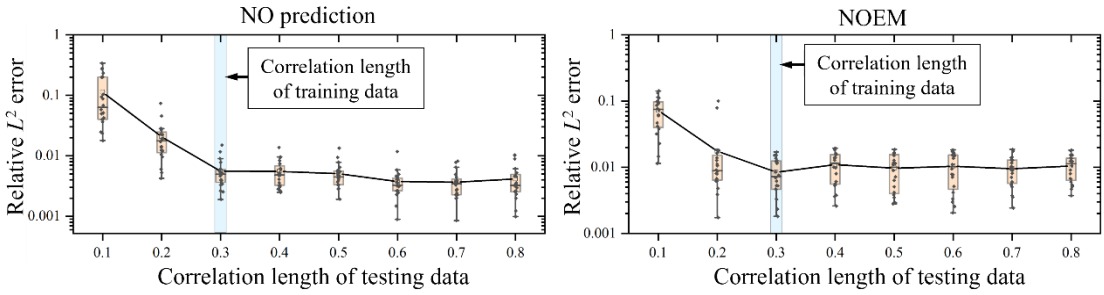


Figure 4-3 Error analysis for the NOEM: The left subplot shows the relative L^2 error from the direct prediction using the trained MIONet. The other subplot illustrates the relative L^2 error from the NOEM.

Further tests are conducted to explore the relation between the accuracy of the NOEM result and the associated NO models. A comparison between the NO model

performance and the NOEM accuracy is performed. The trained MIONet is directly employed to predict the solution of Eq. (4.13) with k' generated from the GP of varying correlation length in the range of [0.1, 0.8]. Concurrently, the NOEM is employed to obtain the solution of Eq. (4.12), where another group of k' is also generated from the GP with different correlation lengths. For each correlation length, 20 samples are collected independently to evaluate the performance of the NO and the NOEM, respectively. The results from the direct prediction and the NOEM are demonstrated via the relative L^2 error compared to the FEM solutions, as shown in Figure 4-3. Both the NO prediction and the NOEM demonstrated a similar trend: the relative L^2 errors decrease as the correlation length increases up to 0.3, and stabilize for $l > 0.3$. Such correlation between the direct prediction error and the NOEM error demonstrated in Figure 4-3 indicates that the NOEM accuracy can be guaranteed by only ensuring the NO performance on small elementary domains.

4.4.2 1D multi-scale problem

In this example, the performance of the proposed NOEM is evaluated through the analysis of two 1D multi-scale problems.

Example 2.1 - Multi-scale coefficient function

A 1D problem with multi-scale coefficient functions from [243] is revisited here. The problem is detailed as follows:

$$
\begin{aligned}
&-\frac{d}{dx}\left(a\left(\frac{x}{\epsilon}\right)\frac{du}{dx}\right) = f(x), \quad x \in [0,1],\\
&a\left(\frac{x}{\epsilon}\right) = 0.5 \times \sin\left(2\pi\frac{x}{\epsilon}\right) + 0.8,\\
&u(x) = u_1, \quad \text{at } x = 0,\\
&u(x) = u_2, \quad \text{at } x = 1.
\end{aligned}
\tag{4.14}
$$

where $u_1 = u_2 = 0$, $f(x) = 0.5$, and $\epsilon = \frac{1}{16}$.

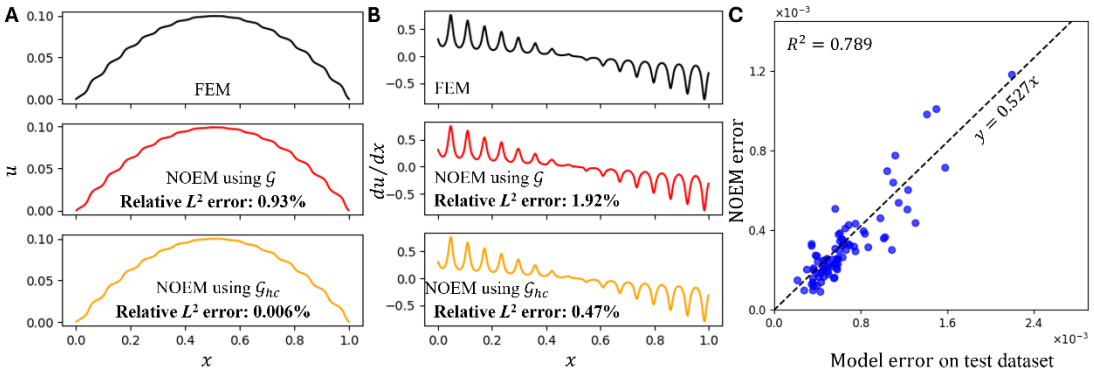


Figure 4-4 Comparison results of Example 2.1: (A) and (B) demonstrate the comparison results with respect to solution values and derivative values, respectively. The ground truth computed from the FEM is plotted in the top subplot. The results from the NOEM using the vanilla DeepONet, $\mathcal{G}$, and the hard-constraint DeepONet, $\mathcal{G}_{hc}$, are plotted in the subplots at the middle and bottom, respectively. Additionally, 80 different $\mathcal{G}_{hc}$ are trained with different settings. (C) visualizes the performance of each NO on the test dataset and the analysis error of the NOEM when using the corresponding model. Model performance is represented by the mean relative L^2 errors on a test dataset of 100 samples from $\left[0, \frac{1}{8}\right]$, shown on the x-axis. Similarly, the NOEM accuracy for different NOs is depicted through the mean relative L^2 errors on Eq. (4.14), using 100 different boundary conditions, with results shown on the y-axis.

The proposed NOEM is employed to compute the solution of Eq. (4.14). To implement the NOEM, a DeepONet, $\mathcal{G}$, and a hard-constraint DeepONet, $\mathcal{G}_{hc}$, are trained to approximate the solution of the following PDEs:

$$\frac{d}{dx}\left(a\left(\frac{x}{\epsilon}\right)\frac{du_{\mathcal{G}}}{dx}\right) = f(x), \quad x \in \left[0, \frac{1}{8}\right], a\left(\frac{x}{\epsilon}\right) = 0.5 \times \sin\left(2\pi\frac{x}{\epsilon}\right) + 0.8, \qquad (4.15)$$

$$u(x) = u_{B,1}, \quad \text{at } x = 0 u(x) = u_2, \quad \text{at } x = 1.$$

$$u(x) = u_{B,2}, \quad \text{at } x = \frac{1}{8},$$

$$\mathcal{G} \text{ or } \mathcal{G}_{\hbar c} : \boldsymbol{u}_{\Omega_{\text{NOE}}} \mapsto u_{\mathcal{G}}$$

where $\boldsymbol{u}_{\Omega_{\text{NOE}}} = \{u_{B,1}, u_{B,2}\}$ is the boundary conditions, which is also served as the branch net input of the DeepONet, $\mathcal{G}$ and $\mathcal{G}_{\hbar c}$.

In the proposed NOEM, the domain of $[0,1]$ is divided into 8 NOEs of length in $\frac{1}{8}$. The ground truth from the FEM and the solutions computed from the proposed NOEM using two different NO models are compared in Figure 4-4A-B. The NOEM using different NO models both yield accurate results for this multi-scale problem with respect to both the solution value and the derivative value, further indicating the robustness of the proposed method. Next, we illustrate the relationship between the NO performance on the test dataset and the accuracy of the NOEM solution.

We trained 80 DeepONets with hard constraints, randomly varying the width and depth of the subnets in the DeepONets, as well as the training sample size. Each NO's performance is evaluated using a standard error metric—the mean relative L^2 error—calculated on a test dataset comprising 100 samples from Eq. (4.15) with $u_{B,1}, u_{B,2} \sim U(-0.1, 0.1)$.

Additionally, the corresponding NOEM accuracy for each NO is assessed by the mean relative L^2 error of the solution of Eq. (4.15) with 100 boundary conditions randomly sampled from $u_1, u_2 \sim U(-0.1, 0.1)$. The relationship between the performance of each DeepONet and the corresponding NOEM accuracy is illustrated in Figure 4-4C. A strong linear correlation with $R^2 > 0.75$ between NO performance and NOEM accuracy is observed, providing a convenient means to estimate NOEM accuracy using a commonly adopted NO performance metric. Moreover, these results indicate that ensuring NO performance on a small elementary domain is sufficient to

guarantee the accuracy of the NOEM solution across the entire larger domain, implying the scalability of the proposed method.

Example 2.2 - Multi-scale source term

Another multi-scale problem is also investigated here, whose setup can be detailed as:

$$\begin{aligned} -\frac{d^2u}{dx^2}+\frac{du}{dx} &= f(x), \quad \text{for } x \in [0,10] \\ u(x) &= 0, \quad \text{at } x = 0, \\ u(x) &= 0, \quad \text{at } x = 10. \end{aligned} \tag{4.16}$$

where $f(x) = (x\ mod\ 1)(1 - x\ mod\ 1)$.

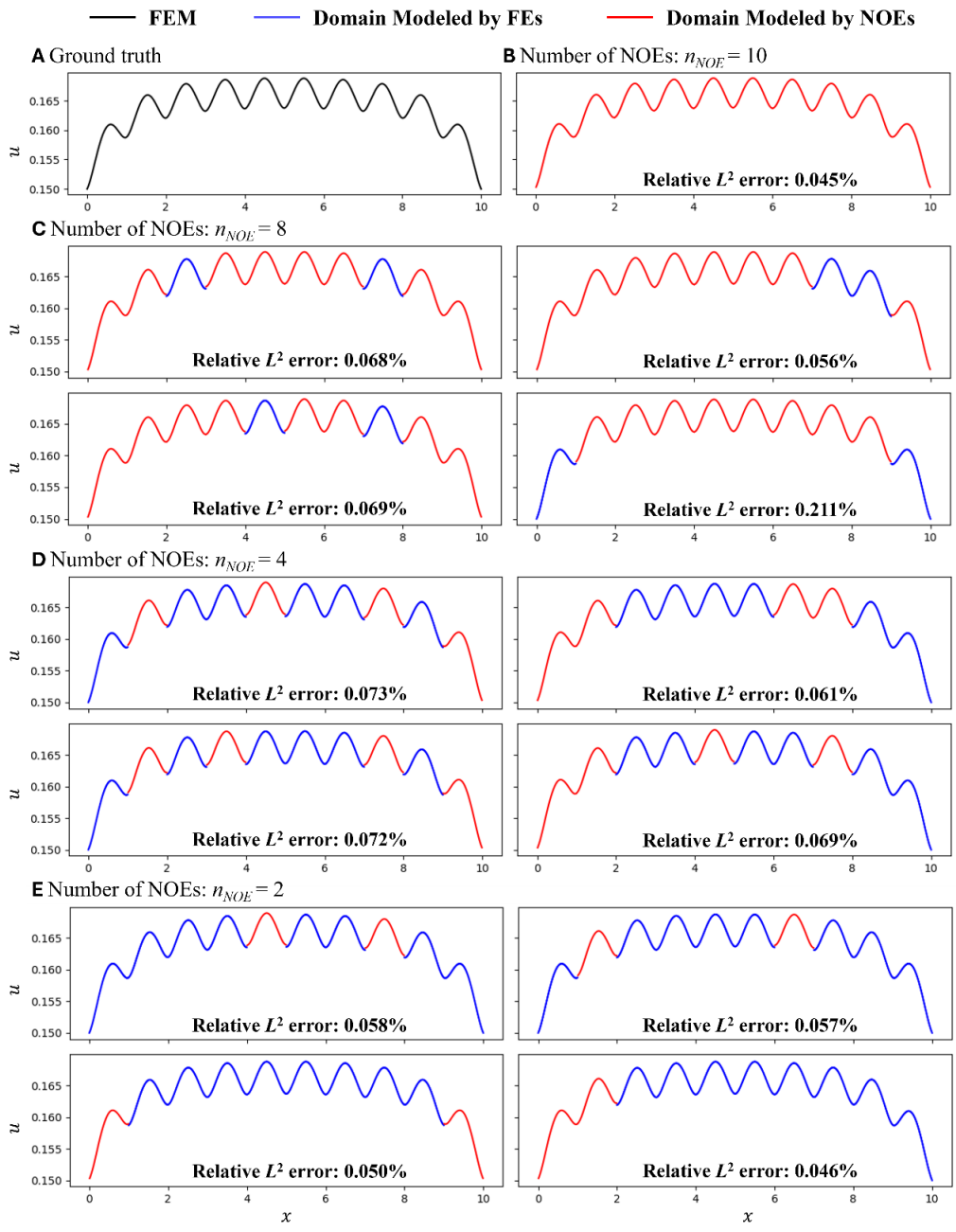


Figure 4-5 Comparison results of Example 2.2: (A) The ground truth from the FEM. (B) The results from the NOEM using 10 NOEs ($n_{NOE} = 10$). (C) The results from the NOEM using 8 NOEs ($n_{NOE} = 8$). (D) The results from the NOEM using 4 NOEs ($n_{NOE} = 4$). (E) The results from the NOEM using 2 NOEs ($n_{NOE} = 2$). The employed mesh configurations for different n_{NOE} are randomly chosen and illustrated via the color. For instance, the subplot at the left bottom corner of (E) represents that $[0,1]$ and $[9,10]$ are modeled by the NOEs and the rest domains are modeled by the conventional FEs.

A DeepONet is utilized to approximate the mapping between the boundary condition and the solution corresponding to the following PDEs:

$$-\frac{d^2u_{\mathcal{G}}}{dx^2}+\frac{du_{\mathcal{G}}}{dx}=f(x), \quad \text{for } x \in [0,1],$$

$$u_{\mathcal{G}}(x)=u_{B,1}, \quad \text{at } x=0,$$

$$u_{\mathcal{G}}(x)=u_{B,2}, \quad \text{at } x=10.$$

$$\mathcal{G}: \boldsymbol{u}_{\Omega_{\text{NOE}}} \mapsto u_{\mathcal{G}}$$

where $\boldsymbol{u}_{\Omega_{\text{NOE}}}=\{u_{B,1},u_{B,2}\}$ represents the boundary condition and serves as the branch net input of the pre-trained DeepONet.

For the implementation of the NOEM, the domain $[0,10]$ is divided into 10 segments as: $\{[i,i+1]\}_{i=0}^{9}$. Within each segment, two modeling strategies are deployed: (1) 100 conventional FEs per segment, and (2) one NOE per segment. The quantity of deployed NOEs, denoted as n_{NOE}, varies from 2 to 10, with different configurations of NOE meshes tested for each n_{NOE}. The comparison results in Figure 4-5 suggest that the accuracy of the NOEM is relatively robust to variations in the location of the domains modeled by the NOEs, illustrating the insensitivity to mesh configuration within the modeled domain.

4.4.3 Heat transfer on complex geometries with varying scales

Several two-dimensional heat transfer problems defined on different geometries with varying scales are investigated here to validate the computational efficiency and model reusability of the proposed NOEM. The governing equation of the heat transfer problem is formulated as follows:

$$-\nabla \cdot \big(K\nabla u(x,y)\big)=0 \quad \text{for } (x,y) \in \Omega \tag{4.17}$$

where K denotes the thermal diffusivity, which is chosen as 1.0; and Ω represents a rectangular area that includes single or multiple circular openings.

Example 3.1 - Rectangular area with single hole

We investigate the heat transfer problem in a rectangular domain with a central circular opening. As illustrated in Figure 4-6A, the domain for Eq. (4.17) is defined as:

$$\Omega = \Omega_1 \setminus \Omega_C$$

where $\Omega_1 = [0,1.6]^2$ is a rectangular area; and $\Omega_C = \{(x,y) \mid (x-0.8)^2 + (y-0.8)^2 \leq 0.15^2\}$ denotes the central circular area with a radius of 0.15.

Two different boundary conditions are investigated here. In the first boundary condition, the Dirichlet boundary conditions are randomly chosen from $v \sim \mathcal{GP}$ with a correlation length equal to 0.3, which are applied to the left and right boundaries of the rectangular area, $\Gamma = \{x \in \{0,1.6\} \text{ and } y \in [0,1.6]\}$. For the remaining boundaries of Ω, zero-Neumann boundary conditions are employed. Thus, this first boundary condition, named Boundary Condition 1, can be written as:

$$\begin{aligned} u(x,y) &= v(x,y) \quad \text{for } (x,y) \in \Gamma, \\ \partial_n u(x,y) &= 0 \quad \text{for } (x,y) \in \partial\Omega \setminus \Gamma \end{aligned} \tag{4.18}$$

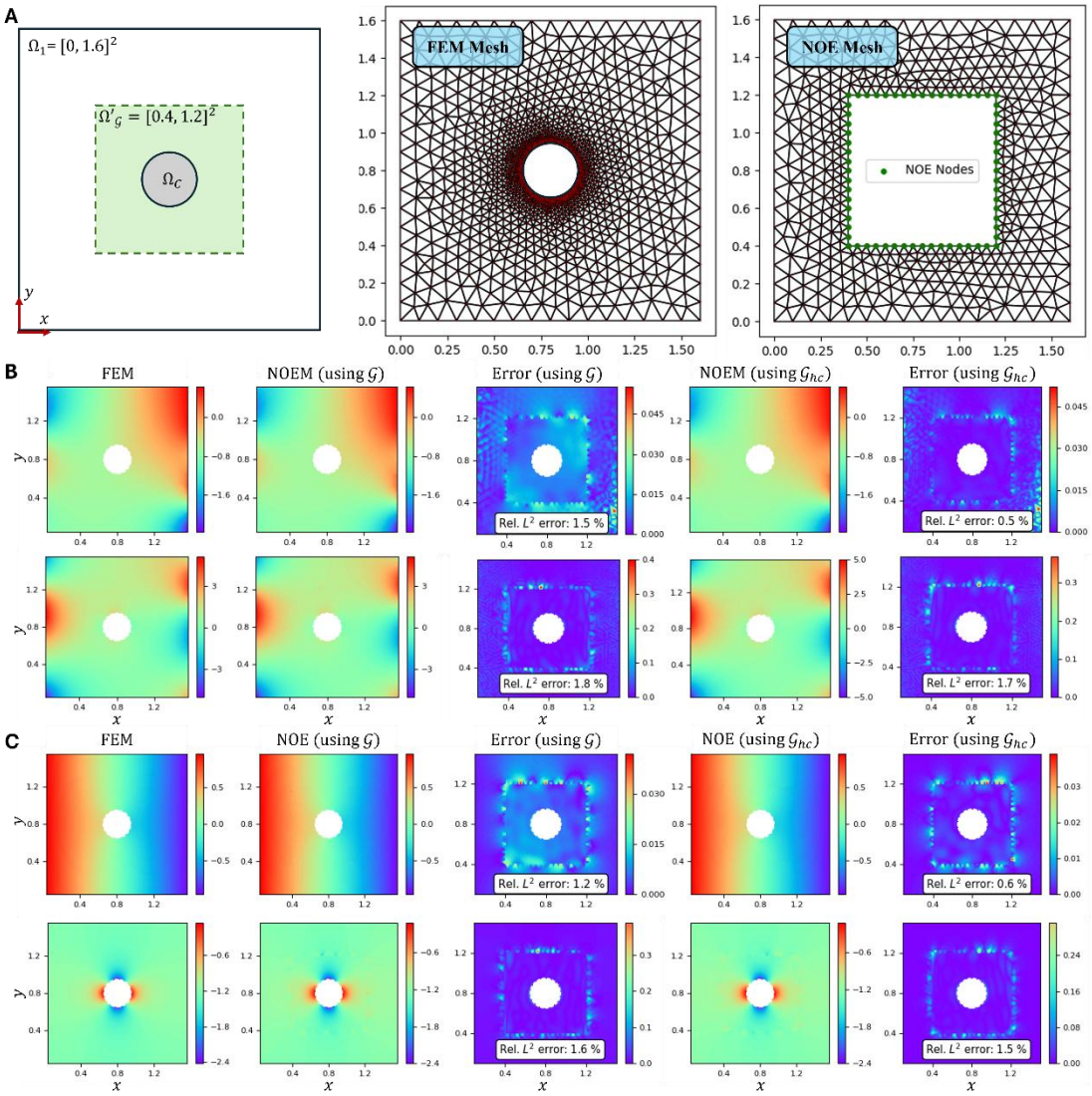


Figure 4-6 Heat transfer analysis on the rectangular area with single hole: (**A**) The domain of the problem is illustrated in the left subplot, where the gray area and the green areas denote the central circular opening, Ω_C, the domain modeled by the NOE, $\Omega_{\mathcal{G}}$, respectively. The meshes used in the FEM and the NOEM are visualized in the middle and right subplots, respectively. (B) and (C) illustrate the comparison results for Boundary Condition 1 and Boundary Condition 2, respectively. In addition, the upper and lower rows in (B) and (C) represent the comparison results about the solution values, u, and the derivative values, $\frac{\partial u}{\partial x}$, respectively.

Except from Boundary Condition 1 described in Eq. (4.18), another set of

boundary conditions, referred to as Boundary Condition 2, is also investigated and can be given as:

$$\begin{aligned} u(x,y) &= 1, \quad \text{for } x = 0, \\ u(x,y) &= -1, \quad \text{for } x = 1.6, \\ \partial_n u(x,y) &= 0, \quad \text{for } (x,y) \in \partial\Omega \setminus \Gamma. \end{aligned} \tag{4.19}$$

The meshes generated for the FEM and the NOEM are depicted in Figure 4-6A. Two NOs, including a vanilla DeepONet ($\mathcal{G}$) and a hard-constraint DeepONet ($\mathcal{G}_{\hbar c}$), are trained and employed to map the Dirichlet boundary conditions to the solutions over $\Omega_{\mathcal{G}}$ as:

$$\mathcal{G} \text{ or } \mathcal{G}_{\hbar c}: u|_{\partial\Omega_{\mathcal{G}}} \mapsto u_{\mathcal{G}}$$

$$-\nabla \cdot \big(K \nabla u(x,y)\big) = 0 \quad \text{for } (x,y) \in \Omega_{\mathcal{G}},$$

$$u_{\mathcal{G}}(x,y) = u|_{\partial\Omega_{\mathcal{G}}}(x,y) \quad \text{for } (x,y) \in \partial\Omega'_{\mathcal{G}},$$

$$\partial_n u_{\mathcal{G}}(x,y) = 0 \quad \text{for } (x,y) \in \partial\Omega_{\mathcal{G}} \setminus \partial\Omega'_{\mathcal{G}},$$

where $\Omega_{\mathcal{G}} = \Omega'_{\mathcal{G}} \setminus \Omega_C$ denotes the domain modeled by the NOE (see the green area in Figure 4-6A; and $\Omega'_{\mathcal{G}} = [0.4,1.2]^2$ is a rectangular area.

$$\mathcal{G} \text{ or } \mathcal{G}_{\hbar c}: u|_{\partial\Omega_{\mathcal{G}}} \mapsto u_{\mathcal{G}}$$

$$-\nabla \cdot \big(K \nabla u(x,y)\big) = 0 \quad \text{for } (x,y) \in \Omega_{\mathcal{G}},$$

$$u_{\mathcal{G}}(x,y) = u|_{\partial\Omega_{\mathcal{G}}}(x,y) \quad \text{for } (x,y) \in \partial\Omega'_{\mathcal{G}},$$

$$\partial_n u_{\mathcal{G}}(x,y) = 0 \quad \text{for } (x,y) \in \partial\Omega_{\mathcal{G}} \setminus \partial\Omega'_{\mathcal{G}},$$

where $\Omega_{\mathcal{G}} = \Omega'_{\mathcal{G}} \setminus \Omega_C$ denotes the domain modeled by the NOE (see the green area in Figure 4-6A; and $\Omega'_{\mathcal{G}} = [0.4,1.2]^2$ is a rectangular area.

Comparisons between the FEM and the proposed NOEM with respect to both the solution values, u, and their derivatives, $\frac{\partial u}{\partial x}$, are visualized in Figure 4-6B-C. The proposed method demonstrates great accuracy in approximating both solution values and derivatives, thereby affirming NOEM's capability in simulating objects with complex geometries using a coarse mesh.

Example 3.2 - Rectangular area with multiple holes

In this example, we further investigate the proposed method for the heat transfer analysis of a rectangular plate featuring multiple holes (as visualized in Figure 4-6A) to demonstrate the efficiency and scalability of the NOEM. The employed boundary conditions are similar with the one in Eq (4.19), where the Dirichlet boundary condition is randomly generated and applied to the left and right boundaries, while the zero-Neumann boundary condition is employed on the remaining parts of the boundary. To elaborate the rectangular domain with multiple holes, Ω in Eq. (4.17) can be defined as follows:

$$\Omega = \Omega_n \setminus \Omega_{C,n}$$

where n represents the row and column numbers of holes; $\Omega_n = [1.5n, 1.5n]^2$ a square region that scales with n; $\Omega_{C,n}$ denotes the combined area of the circular holes, which is given by:

$$\Omega_{C,n} = \{(x,y)|(x-(i-0.5)\times 1.5)^2 + (y-(j-0.5)\times 1.5)^2 \leq 0.15^2 \forall i,j: 1 \leq i,j \leq n\}$$

First, the rectangular area containing 3 x 3 holes ($n = 3$) is investigated. The mesh size for the FEM is similar to the one used in Example 3.1. The mesh employed in the NOEM is visualized in Figure 4-7A. The DeepONet without the hard-constraint trained

in Example 3.1 is directly reused here to formulate the NOEs without additional tuning or training. Comparative results between the FEM and the NOEM are presented in Figure 4-7B. The NOEs, directly formulated with the previously trained vanilla DeepONet, $\mathcal{G}$, demonstrate accurate results, underscoring the model reusability within the proposed NOEM.

Subsequent tests were conducted to assess the robustness and efficiency of the NOEM. The domain scalers, n, representing the hole numbers at each row and column, were varied from 2 to 10. For each configuration, 20 boundary conditions were randomly generated and analyzed. The relative L^2 errors of the NOEM are summarized in Figure 4-7C, illustrating stable performance across different n values, which indicates its robustness and scalability for problems of varying scales. Additionally, the average computational costs of the FEM and the NOEM are presented in Figure 4-7D, where around 14 times improvement in the computational speed is observed, indicating the efficiency and scalability of the NOEM.

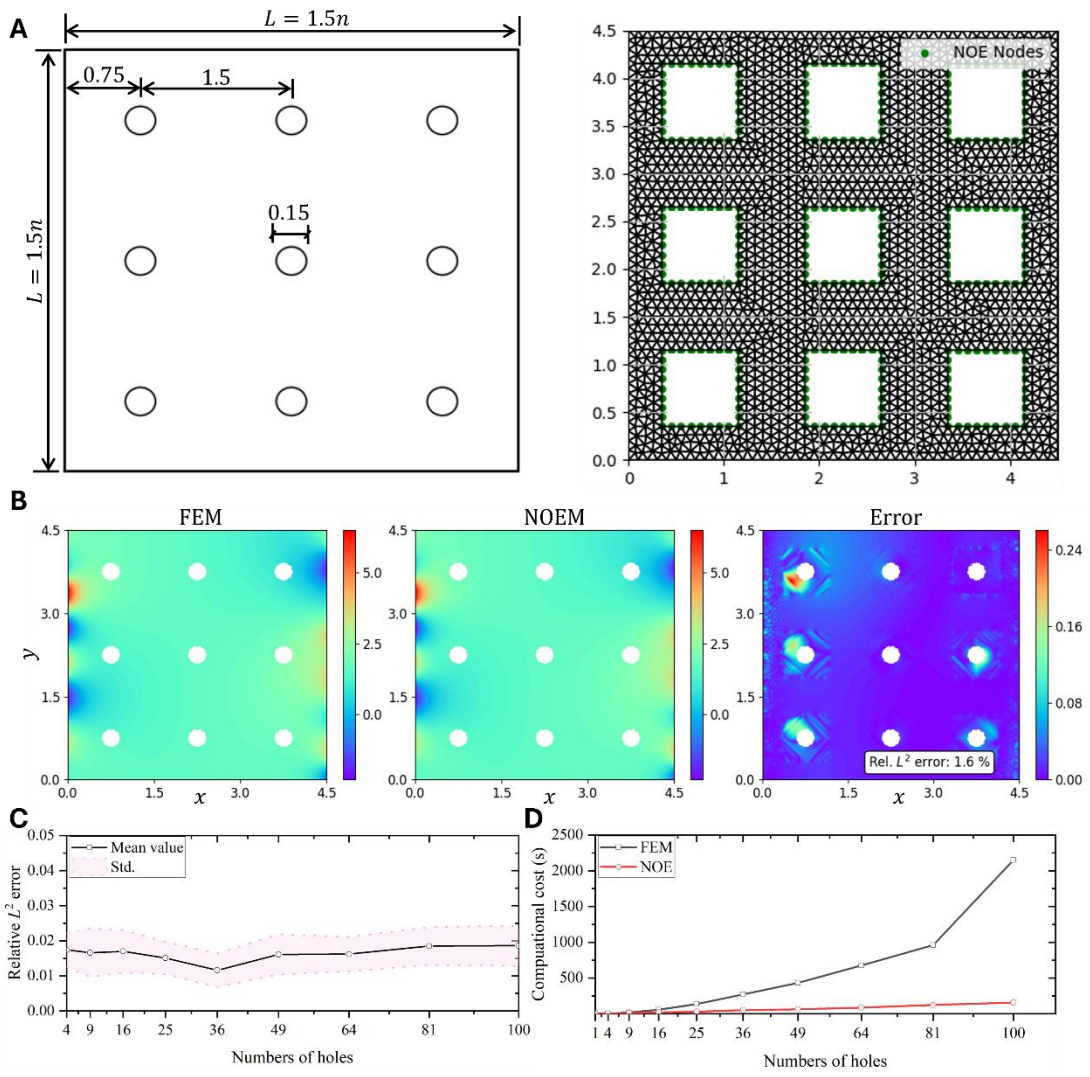


Figure 4-7 Heat transfer analysis on the rectangular area with multiple holes: (A) The configuration of the rectangular area containing $3 \times 3 = 9$ holes is plotted in the left side. The mesh for the NOEM, where the boxes outlined by the green dots are the domains modeled by the NOEs, is visualized in the right subplot. (B) The comparison results for the heat transfer on the rectangular area with multiple holes are visualized. The relative L^2 error of the NOEM equals 1.6 %. (C) The curve of the relative L^2 errors versus the hole numbers is plotted. (D) The average computational times for the rectangular area with different holes using different methods are visualized.

4.4.4 Darcy flow with complex permeability fields

In this example, we revisit the Darcy flow problem on a rectangular domain. The problem is formulated as follows:

$$-\nabla \cdot \big(K(x,y)\nabla u(x,y)\big) = 0 \quad \text{for } (x,y) \in \Omega_1$$

$$u(x,y) = v(x,y) \quad \text{for } x = 0 \text{ and } x = 2$$

$$K(x,y)\partial_n u(x,y) = 0 \quad \text{for } y = 0 \text{ and } y = 2$$

where $\Omega_1 = [0,2]^2$ is a rectangular area; u denotes the pressure; v is a function representing the Dirichlet boundary conditions; and $K(x,y)$ is the permeability field defined as:

$$K(x,y) = \begin{cases} 1 & \text{for } (x,y) \in \Omega_1 \setminus \Omega_2 \\ K'(x,y) & \text{for } x \in \Omega_2 \end{cases}$$

in which, $\Omega_2 = [0.5,1.5]^2$.

Accordingly, four types of permeability fields are investigated in Examples 4.1 to 4.4 as follows:

Continuous permeability field:

$$K'(x,y) = \exp\big(w(x,y)\big), \quad w(x,y) \sim \mathcal{GP}\left(0, k_l\big((x,y),(x',y')\big)\right) \tag{4.20}$$

where $\mathcal{GP}\left(0, k_l\big((x,y),(x',y')\big)\right)$ is a GP with a correlation length equal, l, to 0.3.

Piecewise constant permeability field:

$$K'(x,y) = \begin{cases} 3 & \text{for } w \le 0 \\ 12 & \text{for } w > 0 \end{cases} \tag{4.21}$$

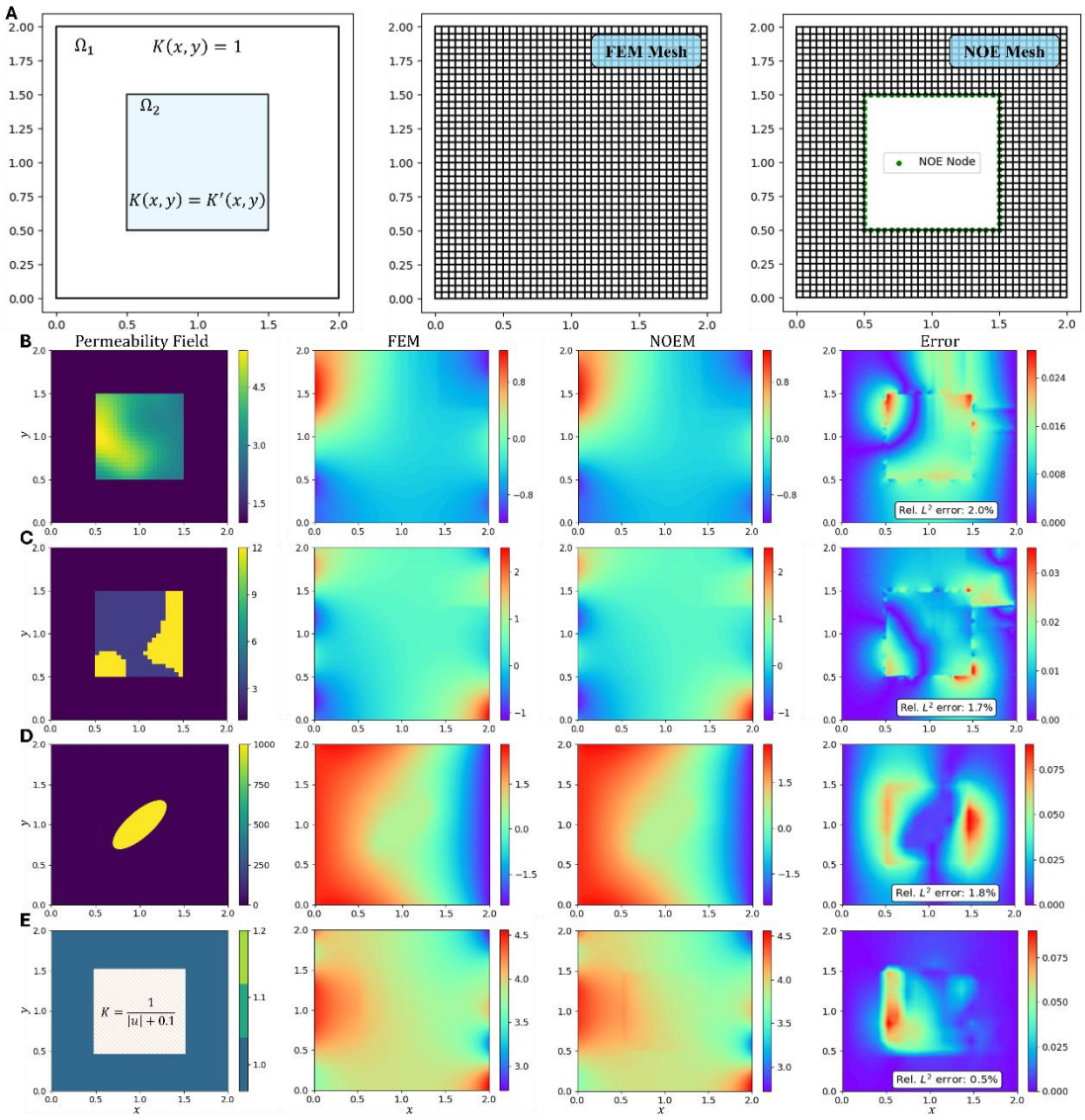


Figure 4-8 Darcy flow with different complex permeability fields: (A) The domain of Darcy flow problem is plotted at the left side. The meshes for the FEM and the NOEM are visualized in the middle and right subplots, respectively. (B) Comparison results considering the continuous permeability field. (C) Comparison results considering the piecewise constant permeability field. (D) Comparison results considering the permeability field parameterized by elliptic shape. The shape parameters for the elliptic shape are: $a = 0.4, b = 0.15$,and $\theta = \frac{\pi}{4}$. (E) Comparison results considering the nonlinear permeability field

Permeability field parameterized by elliptic shape

$$K'(x,y) = \begin{cases} 1 & \text{for } (x,y) \notin \Omega_E \\ 1000 & \text{for } (x,y) \in \Omega_E \end{cases}$$

$$\Omega_E = \left\{ (x,y) \;\middle|\; \left[\frac{(x-1)\cos\theta + (y-1)\sin\theta}{a}\right]^2 + \left[\frac{-(x-1)\sin\theta + (y-1)\cos\theta}{b}\right]^2 \leq 1 \right\} \tag{4.22}$$

where Ω_E is an elliptic area characterized by a, b, and θ.

Nonlinear permeability field:

$$K'(x,y) = \frac{1}{|u(x,y)|} + 0.1 \tag{4.23}$$

The FEM is utilized as the benchmark solution to solve the Darcy flow problem on a 40×40 mesh. In the proposed NOEM, the entire region Ω_2 is modeled using a single NOE. The mesh configurations for both the FEM and NOEMs are depicted in Figure 4-8.

Continuous permeability field

In this section, we investigate the continuous permeability field as defined in Eq. (4.20). The Dirichlet boundary condition here is randomly chosen from a GRP with the correlation length of 0.3, $v \sim \mathcal{GP}$. To facilitate the NOEM, a MIONet is employed to approximate the solution of the following PDE:

$$-\nabla \cdot \left(K'(x,y)\nabla u_{\mathcal{G}}(x,y)\right) = 0 \quad \text{for } (x,y) \in \Omega_{\mathcal{G}}$$

$$u_{\mathcal{G}}(x,y) = u|_{\partial\Omega_{\mathcal{G}}}(x,y) \quad \text{for } (x,y) \in \partial\Omega_{\mathcal{G}}$$

where $\Omega_{\mathcal{G}} = [0,1.0]^2$ is a rectangular area; and $K'(x,y)$ is defined by Eq. (4.20).

The MIONet is trained to learn the mapping from the combined inputs of boundary

conditions and the permeability field to the pressure field as follows:

$$\mathcal{G}: \left(u|_{\partial\Omega_{\mathcal{G}}}, K'\right) \mapsto u_{\mathcal{G}} \tag{4.24}$$

The generated permeability field and the comparison results between the FEM and the NOEM are plotted in Figure 4-8B. The relative L^2 error of the NOEM is approximately 2%.

Piecewise constant permeability field

The Darcy flow problem with the piecewise constant permeability field is investigated here. The boundary condition, v, here is also generated from the GRP of the correlation length equal 0.3. The employed MIONet, $\mathcal{G}: \left(u|_{\partial\Omega_{\mathcal{G}}}, K'\right) \mapsto u_{\mathcal{G}}$, is also constructed according to Eq. (4.24) while the permeability field, K', follows the piecewise constant definition given by Eq. (4.21). The corresponding permeability field and the comparison results between the FEM and the NOEM are presented in Figure 4-8C, where the relative L^2 error of the proposed method is smaller than 2%.

Permeability field parameterized by elliptic shape

The permeability field in this example can be parametrized via the shape parameters of the elliptic areas in Eq. (4.22). The Dirichlet boundary conditions at the left end and at the right end are selected as the constants, $v(0, y) = 2.5$ and $v(2, y) = -2.5$, respectively. Accordingly, a MIONet is trained to approximate the operator $\mathcal{G}$ mapping the boundary condition, $u|_{\partial\Omega_{\mathcal{G}}}$, and the elliptic shape parameter, $\boldsymbol{\theta} = \{a, b, \theta\}$, to the pressure field as:

$$\mathcal{G}: \left(u|_{\partial\Omega_{\mathcal{G}}}, \boldsymbol{\theta}\right) \mapsto u_{\mathcal{G}}$$

The comparison results between the FEM and the NOEM when taking $a = 0.4, b = 0.15, \theta = \frac{\pi}{4}$ is provided in presented in Figure 4-8D, where the similar dents

around the elliptic area are observed in the solutions from different methods.

Nonlinear permeability field

We further investigated the Darcy flow problem with the nonlinear permeability field in this example. The boundary condition, $u|_{\partial\Omega_{\mathcal{G}}}$, is generated from a GRP, whose the mean value and the correlation length equal to 5 and 0.3, respectively. Here, a DeepONet is trained to approximate the mapping between the boundary condition to the corresponding solution of the PDE defined as follows:

$$\mathcal{G}: u|_{\partial\Omega_{\mathcal{G}}} \mapsto u_{\mathcal{G}}$$

$$-\nabla \cdot \Big(K'(x,y)\nabla u_{\mathcal{G}}(x,y)\Big) = 0 \quad \text{for } (x,y) \in \Omega_{\mathcal{G}}$$

$$u_{\mathcal{G}}(x,y) = u|_{\partial\Omega_{\mathcal{G}}}(x,y) \quad \text{for } (x,y) \in \partial\Omega_{\mathcal{G}}$$

where $\Omega_{\mathcal{G}} = [0,1.0]^2$ is a rectangular area; and K' is the nonlinear permeability field defined by Eq. (4.23).

The FEM results and the NOEM results are both plotted in Figure 4-8E. The relative L^2 error of the NOEM is around 0.5%, indicating the robustness of the applicability of proposed method in solving nonlinear PDE problems.

Appendix A - Mesh-based Auxiliary Functions for Hard-Constraint Neural Operators

The hard-constraint method serves as an effective tool for implementing both NOs and PINN. However, applying hard-constraint can be challenging for some PDEs defined on complex two-dimensional or three-dimensional domains, primarily because it is difficult to analytically formulate auxiliary functions. To address this, various efforts have been made to construct these auxiliary functions. For example, Sheng and Yang [244] utilized spline interpolation to construct the function α, thereby enabling a

penalty-free PINN for second-order boundary-value problems on intricate geometries. Moreover, Sukumar and Srivastava [245] proposed a method to construct α using R-functions based on distance fields, which further ensures the differentiability of the auxiliary function.

Although these approaches to formulate the auxiliary functions in Eq. (4.2) for complex geometries have shown promising results in past studies, such as in [110], they are predominantly developed for PINN rather than operator learning. Consequently, these studies focus primarily on constructing the function α, and taking $\beta_g = g$ as the boundary condition, under the assumption that g can be easily represented. However, when considering an NO with varying boundary conditions, such as $\mathcal{G}: g \mapsto u$, where g and u represent the Dirichlet boundary condition and the corresponding PDE solution, respectively, the formulation of β_g might not be straightforward. In practice and our NOEM, the boundary conditions g are often parameterized by the sensor values, $g \coloneqq \{g(\xi_i)\}_{i=1}^m$, whose sensor locations are $\Gamma \coloneqq \{\xi_i \in \partial\Omega\}_{i=1}^m$. In these cases, formulating the auxiliary function β_g can be challenging, as g are not analytically given but only expressed through discretized sensor values.

In the present work, we propose a novel method to construct β_g for the hard-constraint NO using piecewise polynomials as $\beta_g = p_g$. Let's consider the PDE defined on Ω with the boundary condition given in Eq. (4.1). To enforce the NO compliant with Eq. (4.2), the auxiliary function in Eq. (4.1) should satisfy

$$\beta_g(\xi_i) = g(\xi_i) \quad \text{for } \xi_i \in \Gamma \tag{4.25}$$

We assume that the domain Ω can be discretized into a mesh $\mathcal{H}$. On this mesh, we can define a piecewise polynomial, $\mathcal{P}_{\mathcal{H}}(\cdot; N_{\partial\Omega}, N_\Omega)$, which is constructed based on the nodal values at the mesh nodes. Specifically, $N_{\partial\Omega}$ contains the values of the nodes located on the boundary $\gamma^{\partial\Omega} \coloneqq \{\xi_i^{\partial\Omega} \in \partial\Omega\}_{i=i}^m$, and N_Ω includes the values at all other nodes situated strictly within the domain, denoted by $\gamma^\Omega \coloneqq \{\xi_i^\Omega \in \Omega\}_{i=i}^M$. To satisfy Eq.

(4.25), we first align the nodes in $\mathcal{H}$ on the boundary with the sensor locations parametrizing the boundary condition as$\gamma^{\partial\Omega} = \Gamma$. Then, we can construct p_g as

$$p_g(\cdot) = \mathcal{P}_{\mathcal{H}}\big(\cdot; N_{\partial\Omega} = g, N_\Omega = \mathcal{N}(g)\big)$$

where $\mathcal{N}: \mathbb{R}^m \to \mathbb{R}^M$ could be an mapping from g to the nodal values on γ^Ω.

For simplicity, one could set all nodal values within the domain, $\{\xi_i^\Omega\}_{i=1}^M$, to zero as $\mathcal{N}(g) = \{0\}_{i=i}^M$. However, in this chapter, we adopt a different approach to construct $\mathcal{N}$. First, we train a NO without the hard-constraint to map from the boundary condition to the solution as $\mathcal{G}'_{\mathrm{N}}: g \to u$. Subsequently, we define $\mathcal{N}$ as follows:

$$\mathcal{N}(g) = \{\mathcal{G}'_{\mathrm{N}}[g]\big(\xi_i^\Omega\big) \mid \xi_i^\Omega \in \gamma^\Omega\}_{i=1}^M.$$

In NOEM, to ensure continuity of the NOEM solution, we employ the same types of piecewise polynomials for β_g and the FEs in the NOEM. Furthermore, we align the sensor locations of the hard-constraint NO with the mesh node locations in the NOEM on the interface between the domains modeled by the conventional FEs and the NOEs. This alignment guarantees the continuity of the NOEM solution across the interface.

Appendix B - NOEM Implementation

This section provides implementation details for both the NO training and the NOEM solving processes.

The employed NO, $\mathcal{G}_j$ that constructs the NOE defined on Ω_j^{NOE}, is trained to approximate the operators mapping from the boundary conditions to the corresponding solution of the PDE in Eq. (4.9):

$$\mathcal{G}_j: u|_{\partial\Omega_j^{\mathrm{NOE}}} \mapsto u|_{\Omega_j^{\mathrm{NOE}}}, \quad \forall j \in I_{\mathrm{NOE}}.$$

Let $\Gamma_j \coloneqq \{x_k^{(j)}\}_{k=1}^m \subset \partial\Omega_j^{\mathrm{NOE}}$ be a set of m points and consider

$$\boldsymbol{u}_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)} \coloneqq \begin{bmatrix} u_{\Omega_j^{\mathrm{NOE}}}^{(i)}\left(x_1^{(j)}\right) \\ \vdots \\ u_{\Omega_j^{\mathrm{NOE}}}^{(i)}\left(x_m^{(j)}\right) \end{bmatrix} \in \mathbb{R}^m$$

which is a discretization of $u|_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)}$. Similarly, $u|_{\Omega_j^{\mathrm{NOE}}}^{(i)}$ is the solution of the PDE in Eq. (4.9), obtained when considering the boundary condition $u|_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)}$. $u|_{\Omega_j^{\mathrm{NOE}}}^{(i)}$ is defined based on a set $M_j \subset \overline{\Omega_j^{\mathrm{NOE}}}$ of discretization points to represent $u|_{\Omega_j^{\mathrm{NOE}}}^{(i)}$, which could be obtained from the conventional numerical solvers. A popular loss function is the mean squared error defined by

$$\mathcal{L}(\theta) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \left\| \mathcal{G}_{\not\theta}\left[\boldsymbol{u}_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)}\right](M_j) - u|_{\Omega_j^{\mathrm{NOE}}}^{(i)} \right\|^2$$

where θ denotes the collection of the trainable parameters of $\mathcal{G}_{\not\theta}$; and $\mathcal{G}_{\not\theta}\left[\boldsymbol{u}_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)}\right](M_j)$ is the collection of $\left\{\mathcal{G}_{\not\theta}\left[\boldsymbol{u}_{\partial\Omega_j^{\mathrm{NOE}}}^{(i)}\right](x)\right\}_{x \in M_j}$.

Then, the NOEM solving process using the pre-trained NO, $\mathcal{G}_{\not\theta}$, is illustrated. We assume that the functional J can be written as

$$J[u] = \int_{\Omega} F[u](x)dx \tag{4.26}$$

where F is an appropriate operator associated with the PDE in Eq. (4.7).

Using the representation in Eq. (4.10), the energy functional can be written as:

$$J[q(\cdot;c)] = \sum_{i\in I_{\mathrm{FE}}} \int_{\Omega_i^{\mathrm{FE}}} F\left[\phi_i^{\mathrm{FE}}(\cdot;\alpha_i)\right](x)dx + \sum_{j\in I_{\mathrm{NOE}}} \int_{\Omega_j^{\mathrm{NOE}}} F\left[\phi_j^{\mathrm{NOE}}(\cdot;\beta_j)\right](x)dx$$

For numerical implementation, we apply numerical quadrature rules to approximate the integral terms involving NOEs, which gives

$$\tilde{J}[q(\cdot;c)] = \sum_{i\in I_{\mathrm{FE}}} \int_{\Omega_i^{\mathrm{FE}}} F\left[\phi_i^{\mathrm{FE}}(\cdot;\alpha_i)\right](x)dx + \sum_{j\in I_{\mathrm{NOE}}} \left[\sum_s w_s^{(j)} F\left[\phi_j^{\mathrm{NOE}}(\cdot;\beta_j)\right]\left(z_s^{(j)}\right)\right],$$

where $\{z_s^{(j)}, w_s^{(j)}\}$ is a quadrature rule for the domain Ω_j^{NOE}.

Newton's method is then employed in the present work to solve Eq. (4.11). In Newton's method, the solving process relies on the Hessian matrix and gradient vector of $\tilde{J}[q(\cdot;c)]$, which could be assembled by the Hessian matrix and gradient vector of the energy norm of each element as

$$\tilde{J}\left[\phi_i^{\mathrm{FE}}(\cdot;\alpha_i)\right] = \int_{\Omega_i^{\mathrm{FE}}} F\left[\phi_i^{\mathrm{FE}}(\cdot;\alpha_i)\right](x)dx, \quad \tilde{J}\left[\phi_j^{\mathrm{NOE}}(\cdot;\beta_j)\right] = \left[\sum_s w_s^{(j)} F\left[\phi_j^{\mathrm{NOE}}(\cdot;\beta_j)\right]\left(z_s^{(j)}\right)\right],$$

$$g_i^{\mathrm{FE}}(\alpha_i) = \frac{\partial \tilde{J}\left[\phi_i^{\mathrm{FE}}(\cdot;\alpha_i)\right]}{\partial \alpha_i}, \quad k_i^{\mathrm{FE}}(\alpha_i) = \frac{\partial g_i^{\mathrm{F}}(\alpha_i)}{\partial \alpha_i},$$

$$g_j^{\mathrm{NOE}}(\beta_j) = \frac{\partial \tilde{J}\left[\phi_j^{\mathrm{NOE}}(\cdot;\beta_j)\right]}{\partial \beta_j}, \quad k_j^{\mathrm{NOE}}(\beta_j) = \frac{\partial g_i^{\mathrm{NOE}}(\beta_j)}{\partial \beta_j},\mathrm{w}$$

where $\boldsymbol{g}$ and $\boldsymbol{k}$ represent the gradient vector and Hessian matrix of different elements, ϕ_i^{FE} and ϕ_j^{NOE}, respectively.

Since ϕ_i^{FE} are commonly expressed explicitly, the gradient vector $\boldsymbol{g}_i^{\mathrm{FE}}(\alpha_i)$ and the Hessian matrix $\boldsymbol{k}_i^{\mathrm{FE}}(\alpha_i)$ can be derived in closed form. Conversely, given the complexity of the NOs, $\mathcal{G}_{\mathcal{J}}$, automatic differentiation (AD) is utilized to compute $\boldsymbol{g}_j^{\mathrm{NOE}}(\beta_j)$ and $\boldsymbol{k}_j^{\mathrm{NOE}}(\beta_j)$.

Subsequently, the overall gradient vector $G(c)$ and Hessian matrix $K(c)$ of $\tilde{J}[q(\cdot;c)]$ are assembled from these individual components. The assembled $G(c)$ and $K(c)$ can be employed in Newton's method to solve Eq. (4.11).

Although the NO inputs in our study are expressed using sensor values at the NOE mesh grid points (illustrated by the green dots in Figure 4-1A), alternative discretization

methods could also be utilized. We continue to represent the boundary conditions parametrized by the sensor values at the mesh nodes as β_j and denote the alternative representation of interest as β'_j.

When employing alternative representation, the gradient vector $\boldsymbol{g}_j^{\mathrm{NOE}}(\boldsymbol{\beta}_j)$ and Hessian matrix $\boldsymbol{k}_j^{\mathrm{NOE}}(\boldsymbol{\beta}_j)$ can be computed via leveraging the Jacobian matrix of β'_j with respect to β_j as

$$\boldsymbol{g}_j^{\mathrm{NOE}}(\boldsymbol{\beta}_j) = \frac{\partial \tilde{J}[\phi_j^{\mathrm{NOE}}(\cdot;\beta'_j)]}{\partial \beta'_j} J_j,$$

$$\boldsymbol{k}_j^{\mathrm{NOE}}(\boldsymbol{\beta}_j) = J_j^T \left[\frac{\partial}{\partial \beta'_j}\left(\frac{\partial \tilde{J}[\phi_j^{\mathrm{NOE}}(\cdot;\beta'_j)]}{\partial u_v^{(j)}}\right)\right] J_j + \sum_{i=1}^{m} \left[\frac{\partial \tilde{J}[\phi_j^{\mathrm{NOE}}(\cdot;\beta'_j)]}{\partial \beta'_j}\right]_i T_i^{(j)},$$

$$T_i^{(j)} = \frac{\partial}{\partial \beta_j}\left(\frac{\partial [\beta'_j]_i}{\partial \beta_j}\right),$$

where J_j is the Jacobian matrix of β'_j with respect to the NOE mesh node value vector β_j; $[\cdot]_i$ represents the i^{th} components of the vector; and $T_i^{(j)}$ is the Hessian matrix of $[\beta'_j]_i$ with respect to β_j.

Appendix C - Training Details of Neural Operators

Here, we present some training details of the NO utilized in the numerical example. The subnetworks employed can be classified into two types: (1) MLP, which comprise three to four hidden layers, each containing 50 to 200 neurons; and (2) CNN models, consisting of two convolutional layers with a kernel size of 5 and a stride of 2, followed by two fully-connected layers.

For the training process, we employed the Adam optimizer with a learning rate of 0.001. The model performance is evaluated via the mean square error in the loss

function. During model training, we set the maximum training epoch as 1,000,000 and employed an early-stop criterion that terminates training if there is no improvement in model performance within a span of 10,000 epochs. The training sample numbers are set as 1,000 and 5,000 for the NOs employed for 1D and 2D problems, respectively.

Chapter 5. RAMS: AN IMPROVED SAMPLING METHOD FOR ML MODELS SOLVING PARTIAL DIFFERENTIAL EQUATIONS

5.1 Introduction

Artificial intelligence (AI) has rapidly become an ubiquitous force, revolutionizing various domains within scientific research and engineering applications [246-248]. Among these applications, predicting the solution of partial differential equations (PDEs) using AI—referred to as AI4PDE—is attracting increasing attention from the research community [249, 250]. This burgeoning field leverages machine learning (ML) algorithms and models to address complex scientific challenges governed by the underlying PDEs. In AI4PDE, researchers aim to develop models, denoted as $N_{\boldsymbol{\theta}}$, that adhere to constraints imposed by the underlying differential operators $\mathcal{F}$, as $\mathcal{F}[N_{\boldsymbol{\theta}}](\xi) = 0$.

The implementation of AI4PDE typically unfolds in three phases as illustrated in Figure 5-1: (1) building model architecture, (2) collecting samples, and (3) training model. To adhere to PDE constraints, the training paradigms of machine learning (ML) models in AI4PDE are broadly classified into two categories: data-driven and physics-informed (PI) methods. In the data-driven approach [251-253], samples and their corresponding solutions, $\boldsymbol{x}$ and $u(\boldsymbol{x})$, are generally collected from experimental tests or numerical solvers. The loss function is then formulated based on the discrepancy between the model predictions and the collected data samples. On the other hand, the PI training method [80, 206, 208] formulates the loss function using the norm of the physics residual, $\mathcal{F}[N_{\boldsymbol{\theta}}](\xi)$, leveraging numerical differentiation, e.g., automatic or finite differentiation, to calculate the derivative terms. Following the extensive application of these training paradigms, various ML models have been introduced to advance AI4PDE. The existing conventional ML models, such as the multi-layer perceptron (MLP) and the long-short term memory (LSTM) [96, 254], has been broadly

employed to predict the PDE solution as these ML models generally provide efficient function approximation [255]. However, these models may underperform in some complex scenarios, such as operator learning, where the models are required to approximate mappings between function spaces [256, 257]. To address this, Lu et al. [132] introduced a new ML model, the deep operator network (DeepONet), which extends the universal operator approximation [133] to deep neural networks, thus enhancing generalization. After that, extensive operator learning models have been proposed, which can be generally divided into two types: (1) the DeepONet-type models, e.g., MIONet [240], which are developed based on the separated basis functions and coefficients structures; and (2) the kernel integral models that are initiated from in Fourier Neural Operator (FNO) [137] and then extended to other operator learning models, like Wavelet Neural Operator (WNO) [258] and Laplace Neural Operator (LNO) [259]. Despite these advances in model training and architecture of operator learning, the sampling process—an important component of AI4PDE implementation—remains relatively underexplored and is mostly investigated only in the physics-informed neural networks (PINN) where the models are only trained to approximate PDE solution.

Both data-driven and PI training paradigms are fundamentally reliant on samples, for either collecting data samples or computing physics residuals at sample points. These training paradigms inevitably face a trade-off between the number of samples and model performance. Utilizing more samples increases computational demands for data collection and model training processes, while insufficient samples might deteriorate model performance [128]. In response, certain study has been conducted on developing efficient sampling and resampling methods that use fewer samples without compromising performance. A notable contribution is the residual-based adaptive refinement (RAR) method for adaptive sampling [206]. This technique strategically selects collocation points in areas exhibiting large physics residuals, which are indicative of regions that are insufficiently trained. Following RAR, similar adaptive

sampling strategies, such as RAR-D [125, 260] and R3 [126], have been introduced to target collocation points around areas with significant physics residuals. Beyond these RAR-type methods, other adaptive sampling strategies for PINN have been developed using importance sampling (IS) [127] to minimize the generalization error, where collocation points are sampled according to the probability distribution aligned with the physics residual [128-130, 261]. Despite the intent to reduce the variance in estimating physics residual norms, these IS-based methods share a similar feature with RAR-type methods as they are all designated to densely sample in regions with significant residuals [126].

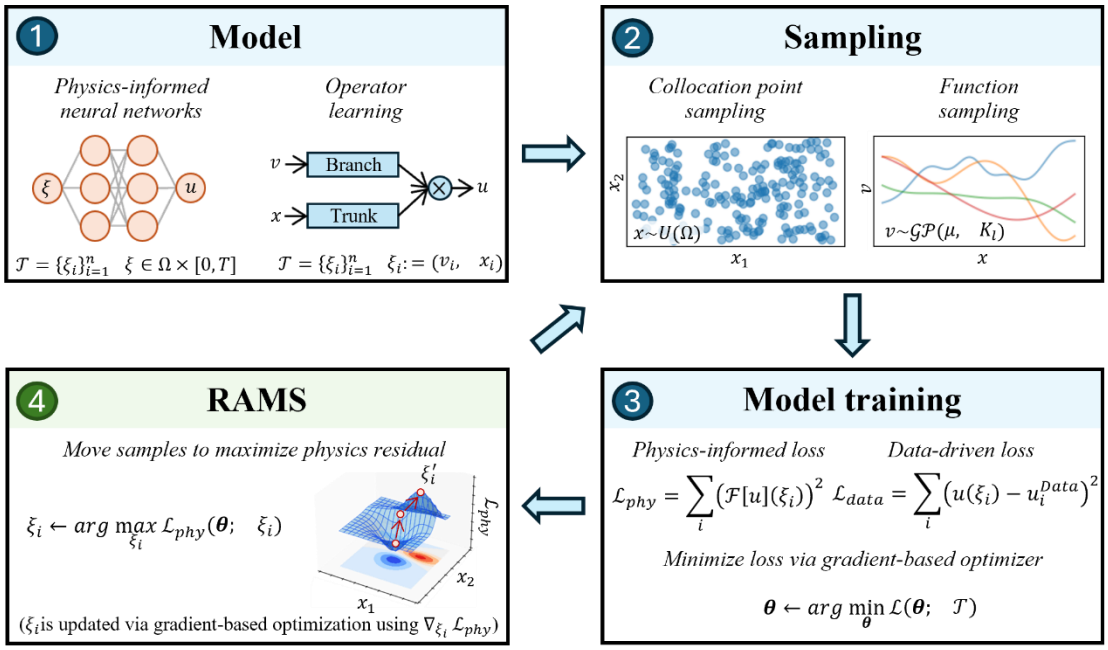


Figure 5-1 Implementation process of ML model solving PDE problems.

Even though these methods with dense sampling around areas with large physics residuals have been proven effective in many PINN for function approximation, they are still less capable for high-dimensional scenarios, such as high-dimensional PDE problems. In these cases, effectively sampling around areas with large physics residuals

becomes computationally daunting due to the sparsity of the high-dimensional space. Furthermore, given the operator learning commonly requires parametrizing the input functions as high-dimensional vectors, none a feasible resampling strategy for operator learning has been proposed according to our best knowledge. For instance, Wang et al. [261] proposed a novel sampling method, DAS^2, using a flow model to estimate the physics residual distribution as the underlying sampling distribution for high-dimensional problems. However, applying DAS^2 to operator learning problems for dynamic systems still requires a prohibitively large number of function samples to yield reliable results. This underscores the urgency for further development of more efficient sampling methods tailored for AI4PDE, particularly in high-dimensional contexts.

As established, an effective sampling method should provide dense sampling in regions exhibiting large residuals. However, current approaches predominantly rely on generating a large volume of samples and strategically retaining them to areas with substantial residuals. Thus, they are less effective for some high-dimensional problem as locating areas with high physics residuals based on the estimations on the placed samples could be challenging. To maximize the physics residual in high-dimensional space, we introduce a powerful tool, the gradient-based optimization method (e.g., ADAM [262]), to move samples according to the adversarial gradient, $\nabla_{\boldsymbol{x}}\{\mathcal{F}[N_{\boldsymbol{\theta}}](\xi)\}^2$, for maximizing the physics residuals (as illustrated in Figure 5-1. Given $\nabla_{\boldsymbol{x}}\{\mathcal{F}[N_{\boldsymbol{\theta}}](\xi)\}^2$ can be easily computed via the automatic differentiation [118], our proposed Residual-based Adversarial-gradient Moving Sample (RAMS) method can be conveniently implemented with those existing sampling method by maximizing the generated samples using the gradient-based optimization. Instead of only implementing for PINN, our proposed RAMS method is also applicable for operator learning problem where the input functions are commonly parametrized via hundreds or thousands of variables. Furthermore, the RAMS can also be implemented for data-driven problem where it can lead to significant model performance improvement with the same amount of training samples. The following observations have been made through extensive case

studies employing RAMS:

- Our RAMS method exhibits great compatibility with existing sampling approaches for PINN, leading to an order of magnitude improvement in accuracy without increasing the number of collocation points.
- The RAMS method substantially reduces the computational cost when applied to solving high-dimensional problem.
- It performs stable improvement in accuracy for both PI and data-driven operator learning problems with the same amount of the training samples.

The chapter is organized as follows. In Section 2, we briefly revisit the PINN, operator learning, and several classical sampling and resampling techniques, based on which the proposed RAMS method is presented. In Section 3, comprehensive tests are conducted across a wide range of problems, including PINN, PI operator learning, and data-driven operator learning, to demonstrate the effectiveness of the proposed approach.

5.2 Methods

In this section, we first briefly introduce PINN to predict PDE solutions by minimizing the loss function of physics residual. Then, the operator learning using DeepONet is presented for both PI and data-driven training. Several sampling methods in PINN are then discussed, based on which our residual-based adversarial-gradient moving sample (RAMS) method is developed.

5.2.1 Physics-informed neural networks (PINN)

Consider the following PDE problem with the solution u

$$\begin{aligned} \mathcal{F}[u](x,t) &= 0, \quad \text{for } (x,t) \in \Omega \times [0,T], \\ \mathcal{B}[u](x,t) &= 0, \quad \text{for } (x,t) \in \partial\Omega \times [0,T], \\ u(x,0) &= u_0(x), \quad \text{for } x \in \Omega, \end{aligned} \tag{5.1}$$

where $\mathcal{F}$ and $\mathcal{B}$ are the differentiation and boundary operators defined according to the investigated PDEs; and u_0 denotes initial condition.

In PINN, a neural network $N_\theta(\xi)$ is trained to predict the PDE solution, where $\xi \coloneqq (x,t) \in \Omega \times [0,T]$ is the spatial-temperate coordinate of the PDE solution. In this study, the widely-used multi-layer perceptron (MLP) is employed, which is parametrized as

$$N_{\boldsymbol{\theta}}(\xi) = g^n \circ \sigma \circ g^{n-1} \circ \sigma \circ \cdots \sigma \circ g^1(\xi),$$

$$g^i(\xi) = W_i^T \xi + b_i, \quad \theta = \{W_i, b_i\}_{i=1}^n,$$

where $\xi \coloneqq (x,t) \in \Omega \times [0,T]$ is the input of the MLP, which commonly represents the spatial-temperate coordinate of the PDE solution; σ is the nonlinear activation function, e.g., the hyperbolic tangent function in this work; n denotes the depth of the MLP model; and $\theta \in R^D$ is the collection of D trainable parameters of $N_{\boldsymbol{\theta}}$ that contains all the weight matrix and bias vectors, W_i and b_i, in the linear map, g^i.

The PINN loss is defined based on the collocation points $\mathcal{T} = \mathcal{T}_{phy} \cup \mathcal{T}_{BC} \cup \mathcal{T}_{IC}$ as

$$\mathcal{L}(\theta;\mathcal{T}) = \mathcal{L}_{phy}(\theta;\mathcal{T}) + \mathcal{L}_{BC}(\theta;\mathcal{T}) + \mathcal{L}_{IC}(\theta;\mathcal{T}),$$

$$\begin{aligned}
\mathcal{L}_{phy}(\theta;\mathcal{T}) &= \frac{1}{N_{phy}} \sum_{i=1}^{N_{phy}} \left(\mathcal{F}[N_\theta]\left(\xi_i^{phy}\right)\right)^2, \mathcal{T}_{phy} = \{\xi_i^{phy} \in \Omega \times [0,T]\}_{i=1}^{N_{phy}}, \\
\mathcal{L}_{BC}(\theta;\mathcal{T}) &= \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left(\mathcal{B}[N_\theta]\left(\xi_i^{BC}\right)\right)^2, \mathcal{T}_{BC} = \{\xi_i^{BC} \in \partial\Omega \times [0,T]\}_{i=1}^{N_{BC}}, \\
\mathcal{L}_{IC}(\theta;\mathcal{T}) &= \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left(N_\theta\left(\xi_i^{IC}\right) - u_0\left(\xi_i^{IC}\right)\right)^2, \mathcal{T}_{IC} = \{\xi_i^{IC} \in \Omega \times [0]\}_{i=1}^{N_{IC}},
\end{aligned} \tag{5.2}$$

where N is the number of the sampled collocation points. The optimal MLP

parameters θ^* are then computed by

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta; \mathcal{T}),$$

which is commonly solved via a stochastic gradient descent method, , e.g., Adam [262].

5.2.2 Operator learning

The PINN described above solves the PDE problem using the MLP model, which has been proven by the universal approximation theorem [85] that can provide a good approximation for arbitrary functions when the appropriate model hyper-parameters are selected. The original PINN is tailored for learning solutions of single given PDE problems, limiting its boarder application. Hence, extensive study has been conducted to include PDE parameters within the model input ξ, enabling the MLP can predict the parametric PDE solutions after training [263-265]. However, when considering the more complicated operator learning problem, this PIML paradigm relies on this conventional ML models is sometimes less capable as it is limited by the MLP expressive capability.

The MLP and other conventional ML models are generally designed to approximate the function, a mapping between two finite dimensional spaces. However, in the operator learning problem, the model should approximate the mapping between two infinite dimensional Banach spaces, i.e., an operator $\mathcal{G}$ mapping a function to another function as

$$\mathcal{G}: v \mapsto u,$$

where v could be a function parametrizing the PDE problem, such as the initial condition u_0, and u could be the PDE solution.

Unlike conventional ML models, DeepONet was developed to address operator learning problems. DeepONet approximates the operator using two networks, a branch

net and a trunk net. The branch net takes a finite-dimensional representation of the input function $\tilde{v}$ as the input (e.g., the function values at certain locations), and the trunk net takes a spatial-temporal coordinate of the output function $(x,t) \in \Omega \times [0,T]$ as the input. Then, the value of the output function at the corresponding coordinate is computed via the element-wise product of the output vectors of the two subnets as

$$\mathcal{G}_\theta[v](x,t) = \sum_{i=1}^{n} N_i^b(\tilde{v}) N_i^t(x,t) + N_0^b,$$

where $\mathcal{G}_\theta$ is a DeepONet model parameterized by θ; $\{N_i^b\}_{i=1}^n$ and $\{N_i^t\}_{i=1}^n$ are the output features from the branch net and the trunk net, respectively; and $N_0^b \in R$ is a trainable bias.

DeepONet can be trained using data-driven methods, whose loss function is formulated based on pre-collected data samples. The data-driven loss function is

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (\mathcal{G}_\theta[\tilde{v}_i](x_i,t_i) - u_i)^2,$$

where u_i is the solution data of the input samples including both the function sample v_i and the spatial-temporal collocation point (x_i, t_i).

Similarly, the PI loss in Eq. (5.2) can also be applied to DeepONet by sampling both input functions v and collocation points. For instance, training a DeepONet to approximate the mapping from the initial condition to the PDE solution, $\mathcal{G}: u_0 \mapsto u$, the PI loss for the operator learning is

$$\mathcal{L}_{phy}(\theta;\mathcal{T}) = \frac{1}{N_{phy}} \sum_{i=1}^{N_{phy}} \left(\mathcal{F}\left[\mathcal{G}_\theta\left[\tilde{v}_i^{phy}\right]\right](x_i^{phy}, t_i^{phy}) \right)^2, \mathcal{T}_{phy} = \{\xi_i^{phy}\}_{i=1}^{N_{phy}},$$

where $\xi_i^{phy} = \left(\tilde{v}_i^{phy}, x_i^{phy}, t_i^{phy}\right)$ represents a sample including both the input function and coordinates. The BC loss $\mathcal{L}_{BC}$ and the IC loss $\mathcal{L}_{IC}$ are computed

similarly to those PINN losses in Eq. (5.2).

5.2.3 Sampling and resampling method

Despite extensive research in SciML, the development of sampling methods for $\mathcal{T}$ mainly focuses on PINN and remains relatively limited for operator learning. A commonly employed technique is random sampling, where collocation points are uniformly randomly sampled over the domain. Beyond uniform sampling, quasi-random low-discrepancy sequences like Latin hypercube sampling (LHS) and Halton sequence sampling are also utilized to reduce sample variance, thereby leading to improved accuracy [125].

Algorithm 5-1 Adaptive sampling for PINN

Initialize the model N_{θ};
Generate the initial samples $\mathcal{T}$ using a predefined sampling method;
repeat
  Train the model for a certain number of iterations;
  Update the samples $\mathcal{T}$ using the selected sampling method;
until the stop criterion is achieved
Train the model for a certain number of iterations.

Apart from non-adaptive random and quasi-random techniques, adaptive sampling methods have been developed for PINN. The first adaptive sampling method is the residual-based adaptive refinement (RAR) method [206], which initially samples some collocation points and then, at each resampling stage, adds more points with the largest PDE residuals into the existing collocation points. Then, a RAR with distribution (RAR-D) sampling method [125, 260] is further developed, where the resampled collocation points follow the distribution of the PDE residuals. To better distinguish these two methods, the greedy RAR method in Ref. [206] is referred to as RAR with greed (RAR-G) in the rest of this chapter. Daw et al. [126] also proposed a Retain-

Resample-Release (R3) sampling method, where the resampled collocation points are selected in a pattern similar to the evolutionary algorithm to aggregate them around the area with large PDE residuals. For a comprehensive comparison of different sampling methods for PINN, we refer the reader to Ref. [125].

In summary, these adaptive sampling methods typically update the initially sampled collocation points after certain training epochs (Algorithm 5-1). Most of these approaches are developed based on the IS, where new collocation points are sampled according to a specially designed distribution, e.g., the PDE residual.

These IS-based adaptive sampling methods usually encourage the resampled points congregated around areas with large PDE residuals \cite{daw2022r3, zhang2025annealed}.

5.2.4 Residual-based Adversarial-gradient Moving Sample method

Method overview

Although existing sampling methods demonstrate promising performance in various PINN applications, the study of high-dimensional PDEs and operator learning problems is still limited, primarily because locating areas with large PDE residuals in high-dimensional spaces is computationally demanding. To address this issue, we develop a residual-based adversarial-gradient moving sample (RAMS) method by leveraging gradient-based optimization techniques. RAMS effectively congregates samples around regions with large PDE residuals by moving samples according to the adversarial gradient direction.

RAMS can be seamlessly integrated with existing sampling techniques by maximizing the PDE residual for the generated samples. In RAMS (Algorithm 5-1), we treat the samples (collocation points in PINN or function samples in DeepONet) as trainable parameters. These trainable samples are optimized to maximize $\mathcal{L}_{phy}$ via

gradient ascent based on the gradient $\nabla_{\xi}\left(\mathcal{F}[N_{\theta}](\xi)\right)^2$. After optimization, when necessary, the trainable samples are projected back to the original domain through an appropriate projection technique $\mathcal{P}$.

For PINN, if the trained samples are outside the domain, a simple projection method is to map the optimized collocation points to the nearest point on the domain boundary. For operator learning, when input functions are generated from a Gaussian random field (GRF) and parameterized using sensor values, we use the kernel smoothing method [266] to smooth the trained samples after each training iteration. This method utilizes the same covariance kernel function k as defined in the GRF to preserve similar function continuity and smoothness. We illustrate the necessity of the kernel smoothing method as the projector in RAMS for operator learning in Section 3.2.

To illustrate the employed kernel smoothing method, consider a function defined as:

$$f: \Omega \ni x \mapsto y \in R,$$

where Ω represents the domain of the function. The function values are parametrized at sensor locations by:

$$\boldsymbol{f} = [f(x_1), f(x_2), \dots, f(x_n)]^T,$$

where $\boldsymbol{f} \in \mathbb{R}^n$ denotes the vector of sensor values parametrizing f, and $\boldsymbol{x} = \{x_i \in \Omega\}_{i=1}^n$ denotes the set of sensor locations.

The projection, $\mathcal{P}$, using the kernel smoothing method is given by:

$$\mathcal{P}(\boldsymbol{f}) = \boldsymbol{K}\boldsymbol{f} \oslash \boldsymbol{k}',$$

where $\boldsymbol{K} = \left[K_{ij}\right] \in R^{n \times n}$ represents the kernel matrix with $K_{ij} = k\left(x_i, x_j\right)$ computed via the kernel function; $\oslash$ denotes element-wise division; and $\boldsymbol{k}' \in R^n$ is

a vector whose components are defined as:

$$\boldsymbol{k}' = \left[\sum_{j=1}^{n} K_{1j}, \sum_{j=1}^{n} K_{2j}, \ldots, \sum_{j=1}^{n} K_{nj}\right]^T.$$

In addition to PINN and physics-informed operator learning, we also apply RAMS to data-driven operator learning by guiding the collection of new data, which works as an active learning strategy. Specifically, we first collect a small training dataset and then progressively add new data points to the dataset. During each sampling stage, newly generated input function samples are optimized to maximize the PDE residual through RAMS. These new input samples are then processed by a numerical solver or experiment to generate the corresponding ground-truth outputs to form new data points, which are added to the training dataset.

Algorithm 5-2 Residual-based Adversarial-gradient Moving Sample method

for $\xi \in \hat{\mathcal{T}}$ **do**
  Compute the gradient $\nabla_{\xi}\mathcal{L}_{phy}$ using automatic differentiation;
  Update ξ with gradient ascent to maximize $\mathcal{L}_{phy}$ for n_{RAMS} iterations;
  If needed, project ξ back to the sampling space by $\mathcal{P}(\xi)$.
end for

RAMS for PINN

In this section, we choose several sampling methods to demonstrate how to integrate RAMS for PINN. First, we apply RAMS to enhance three non-adaptive sampling methods (Algorithm 5-3), including uniformly random sampling (Random), LHS, and Halton sequence. For these non-adaptive sampling methods, we first generate a set of samples and split it into two subsets before network training: a fixed set that remains unchanged and a trainable set. At each resampling stage, we randomly pick a subset from the trainable set and update it by RAMS, while keeping the fixed set unchanged.

Algorithm 5-3 Non-adaptive sampling method with RAMS for PI machine learning

Initialize the network N_θ;
Train the network for a certain number of iterations using $\mathcal{T}$;
Partition $\mathcal{T}$ into a fixed set $\mathcal{T}_1$ and a trainable set $\mathcal{T}_2$;
for $i \leftarrow 1$ **to** t_r **do**
  Train the network for n_{train} iterations using $\mathcal{T}$;
  Randomly choose a subset $\hat{\mathcal{T}} \subset \mathcal{T}_2$, and denote the remainder as $\mathcal{T}_2'$;
  Update $\hat{\mathcal{T}}$ using **RAMS**;
  Update the trainable set: $\mathcal{T}_2 \leftarrow \hat{\mathcal{T}} \cup \mathcal{T}_2'$;
  Update all samples: $\mathcal{T} \leftarrow \mathcal{T}_1 \cup \mathcal{T}_2$;
end for
Train the network for a certain number of iterations using $\mathcal{T}$.

Algorithm 5-4 RAR (RAR-G or RAR-D) with RAMS for PI machine learning

Initialize the network N_θ;
Sample the initial samples $\mathcal{T}$ of size n_{ini} using the selected sampling method;
for $i \leftarrow 1$ **to** t_r **do**
  Train the network for n_{train} iterations using $\mathcal{T}$;
  Randomly generate M samples to form $\mathcal{T}_0$;
  Select m samples from $\mathcal{T}_0$ to form $\hat{\mathcal{T}}$ using one of:
  RAR-G: Choose the m samples with the largest PDE residuals in $\mathcal{T}_0$;
  RAR-D: Sample m points from $\mathcal{T}_0$ according to the probability density function (PDF) proportional to the PDE residual as $p(\xi) \propto \left(\mathcal{F}[N_\theta](\xi)\right)^2$;
  Update $\hat{\mathcal{T}}$ using **RAMS**;
  Update all samples: $\mathcal{T} \leftarrow \mathcal{T} \cup \hat{\mathcal{T}}$;
end for
Train the network for a certain number of iterations using $\mathcal{T}$.

Algorithm 5-5 R3 with RAMS for PI machine learning

Initialize the network N_θ;

2: Sample the initial samples $\mathcal{T}$ using the selected sampling method;
3: **for** $i \leftarrow 1$ **to** t_r **do**
4: Train the network for n_{train} iterations using $\mathcal{T}$;
5: Compute the PDE residuals for the samples in $\mathcal{T}$;
6: Compute the threshold τ as the average PDE residual;
7: Select the samples $\hat{\mathcal{T}}$ with the PDE residuals larger than τ;
8: Update $\hat{\mathcal{T}}$ using **RAMS**;
9: Randomly generate a new set of samples $\mathcal{T}_s$ of size $|\mathcal{T}| - |\hat{\mathcal{T}}|$;
10: Update all samples: $\mathcal{T} \leftarrow \mathcal{T}_s \cup \hat{\mathcal{T}}$;
11: **end for**
12: Train the network for a certain number of iterations using $\mathcal{T}$.

We then describe the integration of RAMS with three adaptive sampling methods: RAR-G, RAR-D, and R3. In RAR (including both RAR-G and RAR-D) with RAMS, new samples added at each resampling stage are treated as trainable and updated by RAMS (Algorithm 5-4). In contrast, R3 does not add points near high-residual regions; instead, it retains the points with the largest residuals at each stage. In R3 with RAMS (Algorithm 5-5), these retained points are trainable and updated by RAMS.

RAMS for operator learning

Similar to PINN, we also apply RAMS to physics-informed operator learning with random sampling and RAR-G. The workflow follows Algorithm 5-3 and Algorithm 5-5. The only difference is that in operator learning, we use GRFs to randomly sample input functions parametrized by sensor values. Moreover, the kernel smoothing method is utilized as the projector in the RAMS method.

Algorithm 5-6 RAR-G with RAMS for data-driven operator learning.

1: Initialize the network N_θ;
2: Collect n_{ini} input functions with the corresponding outputs to form the training set $\mathcal{T}$;

3: **for** $i \leftarrow 1$ **to** t_r **do**
4: Train the network for n_{train} iterations using $\mathcal{T}$;
5: Randomly generate M input functions to form $\mathcal{V}$;
6: Select the m input functions from $\mathcal{V}$ with the largest PDE residuals to form $\hat{\mathcal{V}}$;
7: Update $\hat{\mathcal{T}}$ using **RAMS**;
8: Compute the outputs for $\hat{\mathcal{V}}$ to build a new training dataset $\hat{\mathcal{T}}$;
9: Update the training dataset: $\mathcal{T} \leftarrow \mathcal{T} \cup \hat{\mathcal{T}}$;
10: **end for**
11: Train the network for a certain number of iterations using $\mathcal{T}$.

In addition, we develop RAR-G with RAMS for data-driven operator learning (Algorithm 5-6). In this case, we only generate a small training dataset prior to network training, and the network is first trained on this initial dataset. To improve the network accuracy, we randomly generate a large pool of candidate input functions and select the subset with the largest PDE residuals, which are then refined via RAMS. We compute their corresponding outputs and add the resulting input-output pairs to the training dataset for further training.

5.3 Validation Examples

In this section, we present ten examples to demonstrate the effectiveness of the proposed RAMS method in PINN (Section 3.1), PI operator learning (Section 3.2), and data-driven operator learning (Section 3.3). These examples show that RAMS significantly improves model performance by effectively moving data points to critical areas, all while using the same amount of training data. The network architectures used in all examples are summarized in Appendix A. The sampling and training setups for all examples are summarized in Appendix B, unless otherwise stated.

5.3.1 Physics-informed neural networks

We first investigate the performance of RAMS for PINN and assess its compatibility with various established sampling methods. Specifically, we implement

RAMS alongside three non-adaptive sampling methods (uniformly random sampling, LHS, and Halton sequence) and three adaptive sampling methods (RAR-G, RAR-D, and R3. We refer to the original versions of these methods as "w/o RAMS" and the versions augmented by RAMS as "with RAMS". For each method, we evaluate the performance via the mean relative L^2 error from ten independent experiments on three PDE problems, including 1D Burgers' equation, 1D wave equation, and 2D Poisson equation. Then, a high-dimensional PDE (up to 10 dimensions) is tested to illustrate the performance of the proposed RAMS in handling high-dimensional problems.

Burgers' equation

A Burgers' equation with zero Dirichlet boundary conditions is considered here as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad \text{for } x \in [-1,1], t \in [0,1],$$

$$u(x,0) = -\sin(\pi x), \quad u(-1,t) = u(1,t) = 0$$

where u represent the flow velocity and $\nu = \frac{0.01}{\pi}$ is the viscosity of the fluid.

We visualize the ground-truth solution, the solution from RAR-G with RAMS, and the corresponding pointwise error in Figure 5-10A. Model performances from different sampling methods are reported via the relative L^2 error over 10 independent runs (Figure 5-3A). RAMS significantly improves the non-adaptive sampling methods, reducing mean errors by more than an order of magnitude. For example, random sampling drops from 0.181 to 0.010. RAMS also clearly enhances adaptive sampling methods. For example, for R3, the error decreases by approximately 80%.

Wave equation

In this example, the wave equation is considered, which can be written as

$$\frac{\partial^2 u}{\partial t^2} - 4\frac{\partial^2 u}{\partial x^2} = 0, \quad \text{for } x \in [0,1], t \in [0,1],$$

$$u(0,t) = u(1,t) = 0, \quad \text{for } t \in [0,1],$$

$$\frac{\partial u}{\partial t}(x,0) = 0, \quad \text{for } x \in [0,1],$$

$$u(x,0) = \sin(\pi x) + \frac{1}{2}\sin(4\pi x), \quad \text{for } x \in [0,1],$$

whose solution can be given in a closed-form [125] as

$$u(x,t) = \sin(\pi x)\cos(2\pi t) + \frac{1}{2}\sin(4\pi x)\cos(8\pi t)$$

The performances of different sampling methods are illustrated via the relative L^2 error over 10 independent runs in Figure 5-3B. For all sampling methods, RAMS consistently improves their accuracy, and on average, the improvement exceeds 50%. In Figure 5-10B, we show the ground-truth solution, the prediction of MLP from RAR-G with RAMS, and the corresponding pointwise error.

Poisson equation

We consider a benchmark problem from adaptive finite element tests: a Poisson equation defined as

$$-\Delta u(x_1,x_2) = s(x_1,x_2), \quad \text{for } (x_1,x_2) \in \Omega,$$

$$u(x_1,x_2) = g(x_1,x_2), \quad \text{for } (x_1,x_2) \in \partial\Omega,$$

whose domain is $\Omega = [-1,1]^2$. The reference solution of u is

$$u(x_1,x_2) = \exp(-1000[(x_1-0.5)^2 + (x_2-0.5)^2])$$

which shows a high peak around $[0.5,0.5]$ and rapidly decreases to zero when getting away from the peak points.

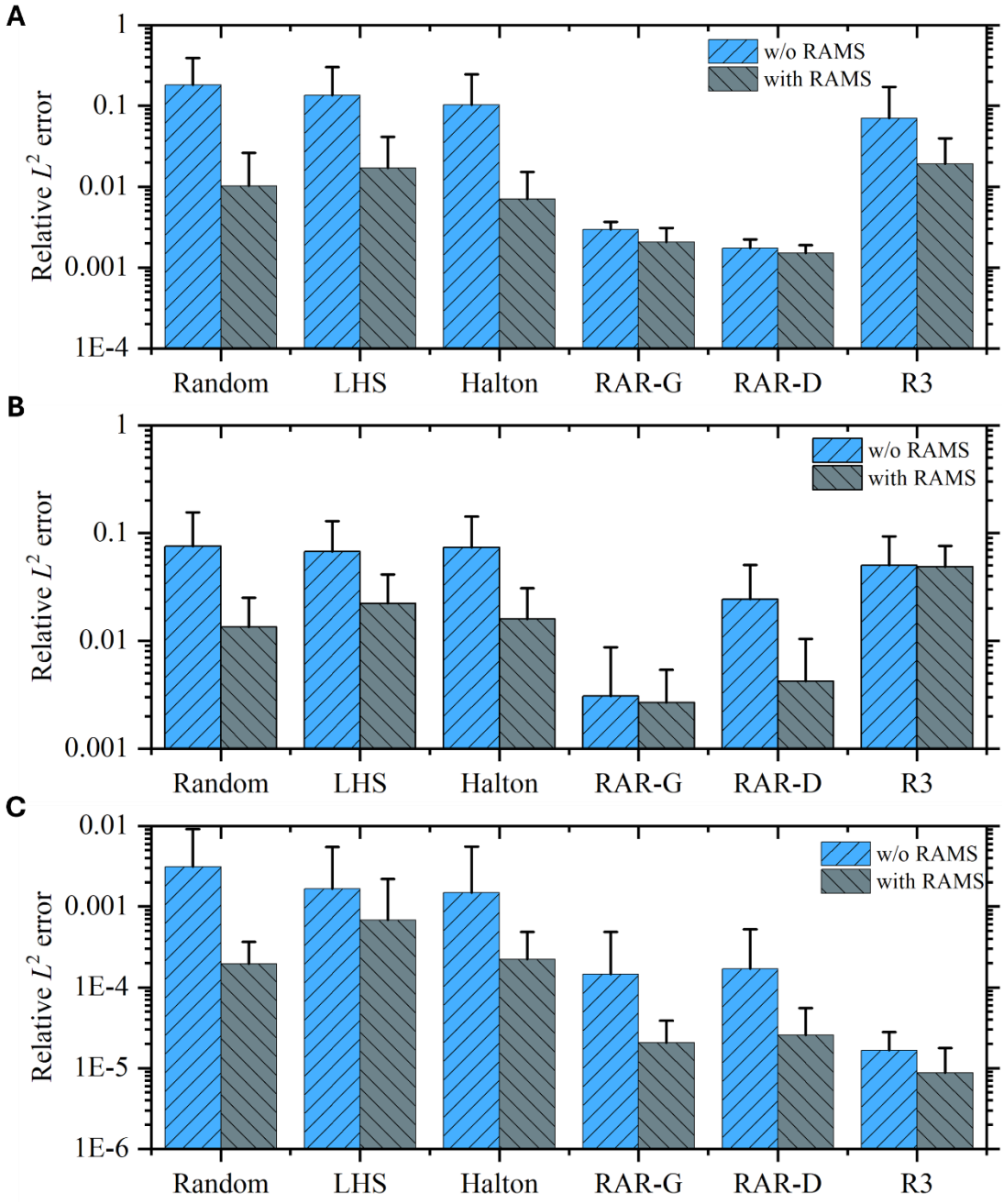


Figure 5-2 Comparison results of with and without the trainable resampling method: The results for Burgers' equation, wave equation, and Poisson equation are illustrated in (A), (B), and (C), respectively. Vertical error bars denote one standard deviation.

The relative L^2 errors over 10 independent runs for different sampling methods are shown in Figure 5-3C. RAMS reduces error by approximately 40-95% across all non-adaptive and adaptive sampling methods, indicating its robust compatibility with

different sampling methods. The ground-truth solution and MLP solution from RAR-G with RAMS are plotted with the pointwise error in Figure 5-10C.

High-dimensional problem

A high-dimensional PDE [267] is revisited here to illustrate the effectiveness of the proposed trainable resampling method. The considered PDE can be written as:

$$-\Delta u(\boldsymbol{x}) = s(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \Omega,$$

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \partial\Omega,$$

where $\Omega = [-1,1]^d$ in a domain in d-dimensional space. The corresponding exact solution is

$$u(\boldsymbol{x}) = \exp(-10\|\boldsymbol{x}\|_2^2).$$

We apply random sampling with and without RAMS using a total of $|\mathcal{T}| = 20{,}000$ collocation points. For the random sampling with RAMS, $|\mathcal{T}_2| = 10{,}000$ points are trainable, and at each resampling stage, $|\hat{\mathcal{T}}| = 2{,}000$ trainable points are updated by Adam for $n_{RAMS} = 100$ epochs. The number of resampling stages and the Adam epochs for model training in each stage are set to $t_r = 160$ and $n_{train} = 500$, respectively. After the final resampling stage, the model is further trained for 20,000 Adam epochs, followed by 300 L-BFGS iterations.

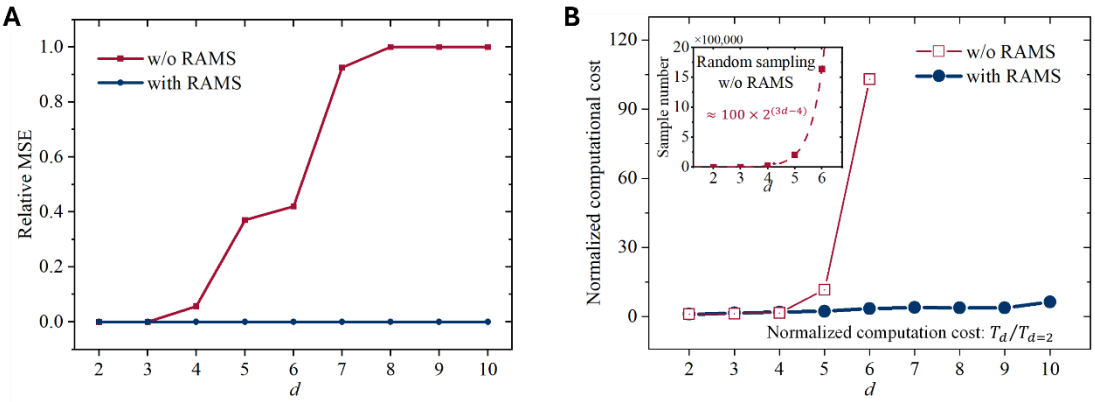


Figure 5-3 PINN results for high-dimensional PDEs: (A) shows the results from two random sampling methods with identical training settings. (B) illustrates the minimal computational cost associated with different sampling approaches. The results in (B) are normalized by the corresponding minimal computational cost for $d = 2$, denoted as $T_{d=2}$, for each sampling method. The inset panel reports the minimal number of collocation points needed by random sampling without RAMS, which grows exponentially with d.

To assess the method performance, we use the relative mean squared error (RMSE) E computed over two disjoint test sets:

$$E = \frac{\sum_{\xi_i \in \xi_1} \left[u_{pre}(\xi_i) - u(\xi_i)\right]^2 + \sum_{\xi_i \in \xi_2} \left[u_{pre}(\xi_i) - u(\xi_i)\right]^2}{\sum_{\xi_i \in \xi_1} u(\xi_i)^2 + \sum_{\xi_i \in \xi_2} u(\xi_i)^2}$$

where ξ_1 is a set of 1000 points sampled in hyperspherical coordinates centered at the origin with radial coordinate $r \sim U(0,1)$, and ξ_2 is a set of 1000 points sampled uniformly from Ω. We show the RMSE for different dimensions d in Figure 5-4A. Without RAMS, random sampling fails to recover the solution as d increases; for $d \geq 7$, the RMSE saturates near 1 ($E \approx 1$), indicating failure on these high-dimensional PDEs. In contrast, RAMS enables the random sampling to remain a small error below 10^{-2} using the same amount of collocation points even when $d = 10$.

We next estimate the minimal computational cost for training needed to achieve $E \leq 10^{-3}$ for two sampling strategies. The training and sampling setups vary with the dimension d as follows.

Collocation points for random sampling without RAMS:

For each d, we train the PINN for 100,000 Adam epochs, followed by 300 L-BFGS iterations. The number of collocation points begins at $|\mathcal{T}| = 100$; if the target accuracy is not met, $|\mathcal{T}|$ is doubled and PINN is retrained. This procedure is repeated until $E \leq 10^{-3}$.

Training iterations for random sampling with RAMS:

We use $|\mathcal{T}| = 20{,}000$ collocation points, with $|\mathcal{T}_2| = 10{,}000$ designated as trainable for all d. At each resampling stage (every $n_{train} = 500$ epochs), a subset of size $|\hat{\mathcal{T}}| = 2{,}000$ is updated. After the final resampling stage, the model is further trained for 10,000 Adam epochs, followed by 300 L-BFGS iterations. For each d, we tune the number of resampling stages t_r and the sample-training iterations n_{RAMS} to achieve $E \leq 10^{-3}$. Specifically, we initialize $t_r = 10$ and $n_{RAMS} = 0$, and if $E \geq 10^{-3}$, we increase t_r by 10 and n_{RAMS} by 20, and repeat until the criterion is met.

For each sampling method, computational costs are normalized by the corresponding cost at $d = 2$ (denoted by $T_{d=2}$). All costs were measured as the run time on the same machine with a single NVIDIA A40 GPU. The normalized minimum costs for different sampling methods across varying d are shown in Figure 5-4B. We also show that the minimum number of collocation points required for random sampling grows exponentially with d. This growth, thereby, drives a rapid increase in computational cost. As for $d = 6$, the cost already exceeds $100\,T_{d=2}$, we stop testing the random sampling for $d \geq 7$ due to prohibitive runtime. By contrast, the cost of random sampling with RAMS scales approximately linearly in d: the computational

costs at $d = 6$ and $d = 10$ are about $3.5\,T_{d=2}$ and $6\,T_{d=2}$, respectively, highlighting the efficiency of the proposed RAMS method.

5.3.2 Examples for physics-informed operator learning

We evaluate RAMS on PI operator learning using two sampling methods: random sampling and RAR-G. During each resampling stage, we update only function samples, and the collocation points for the trunk net input remain fixed. To demonstrate the effectiveness of RAMS, we first test three PDEs: the diffusion-reaction equation, the advection equation, and the Poisson equation with piecewise-constant conductivity. We then test a one-dimensional dynamic system on which conventional sampling methods perform poorly [131].

Diffusion-reaction equation

We consider a diffusion-reaction equation with zero initial and boundary conditions:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial t^2} + ku^2 + v(x), \quad \text{for } (x,t) \in [0,1] \times [0,1].$$

where $k = 10^{-2}$. We use a PI-DeepONet to learn the operator mapping from v to u:

The PI DeepONet composed of two MLPs with 5 hidden layers and 150 in width is utilized here to learn the operator mapping from $v(x)$ to the solution $u(x)$ as

$$\mathcal{G}: v \mapsto u.$$

where the input functions v are sampled from a zero-mean GRF with a Gaussian kernel as $v \sim \mathcal{GP}\left(0, \exp\left(\frac{|x-y|^2}{2l_{train}^2}\right)\right)$, where the correlation length of training data l_{train} is sampled randomly from $U(0.1,0.8)$.

We use the random sampling and RAR-G with and without RAMS for this example. For each method, we evaluate the trained network on eight test datasets with correlation

lengths $l_{test} \in \{0.1, 0.2, \ldots, 0.8\}$. Performances of different sampling methods are reported via the relative L^2 errors on all eight test datasets (Figure 5-4). RAMS yields clear accuracy improvement across different test datasets, especially for those of small correlation lengths. For example, for random sampling, when $l = 0.1$, the mean error decreases from approximately 0.25 (without RAMS) to 0.08 (with RAMS), i.e., a 70\% error reduction. A representative example of the input function, the ground-truth solution, the PI-DeepONet prediction with RAR-G and RAMS, and the piecewise error are shown in Figure 5-11A. Additionally, we show the necessity of the kernel smoothing method as the projector in RAMS in Appendix D.

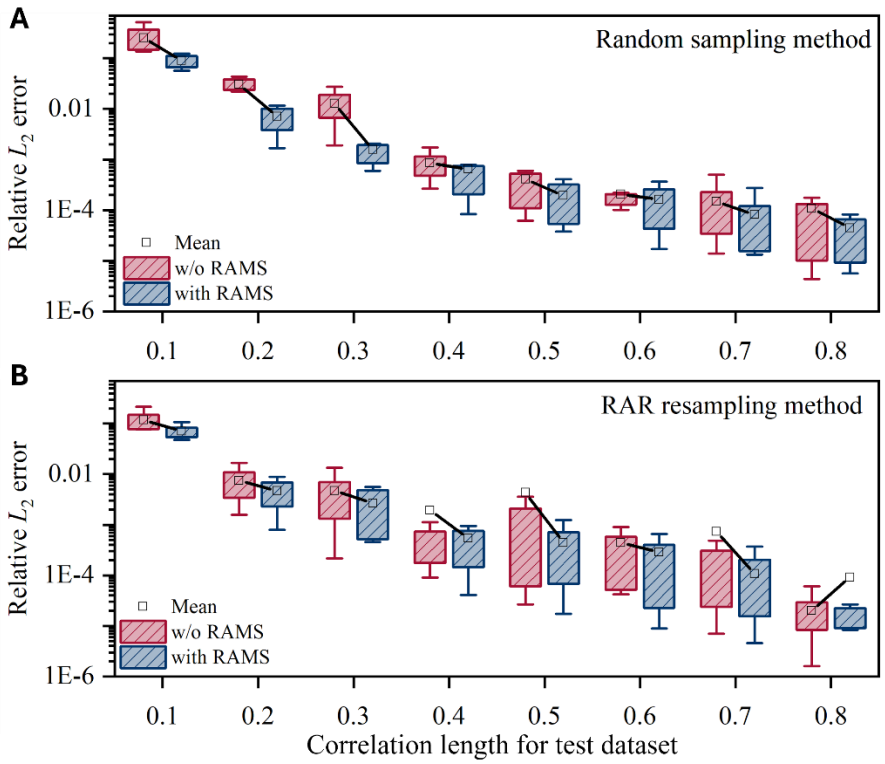


Figure 5-4 PI operator learning of the diffusion-reaction equation: The correlation length l_{train} of training data is randomly sampled from $U(0.1, 0.8)$. Relative L^2 errors for different test data correlation lengths l_{test} are computed from eight independent runs. (A) Random sampling with and without RAMS. (B) RAR-G with and without RAMS.

Advection equation

This example aims to investigate the extrapolation capability and computational cost of the proposed RAMS method. We utilize the PI operator learning on the following advection equation tested in [134]:

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = 0, \quad \text{for } (x,t) \in (0,1) \times (0,1),$$

$$s(x,0) = \sin(\pi x), \quad \text{for } x \in (0,1),$$

$$s(0,t) = \sin\left(\frac{\pi}{2}t\right), \quad \text{for } t \in (0,1).$$

A DeepONet $\mathcal{G}$ is trained to approximate the operator mapping from u to s as

$$\mathcal{G}: u \mapsto s.$$

For network training, the input function is sampled from a zero-mean GRF with a Gaussian kernel of a fixed correlation length $l_{train} = 0.2$.

We apply random sampling with and without RAMS and find that RAMS improves the network generalizability. Specifically, we evaluate the trained PI-DeepONet on three test datasets with input functions sampled from GRFs with correlation lengths $l_{test} \in \{0.2, 0.1, 0.05\}$. The number of RAMS sample-training iterations, n_{RAMS}, is varied from 0 to 400. RAMS substantially improves interpolation accuracy: when $l_{test} = l_{train} = 0.2$, the error is reduced by approximately 30% (Figure 5-5A). RAMS also enhances extrapolation, yielding roughly 40% lower error on test datasets with $l_{test} = 0.1$ (Figure 5-5B) and 0.05 (Figure 5-5C), demonstrating improvements in both interpolation and extrapolation. In addition, an example of the input function, the ground-truth solution, PI-DeepONet prediction, and its pointwise error are shown in Figure 5-5B.

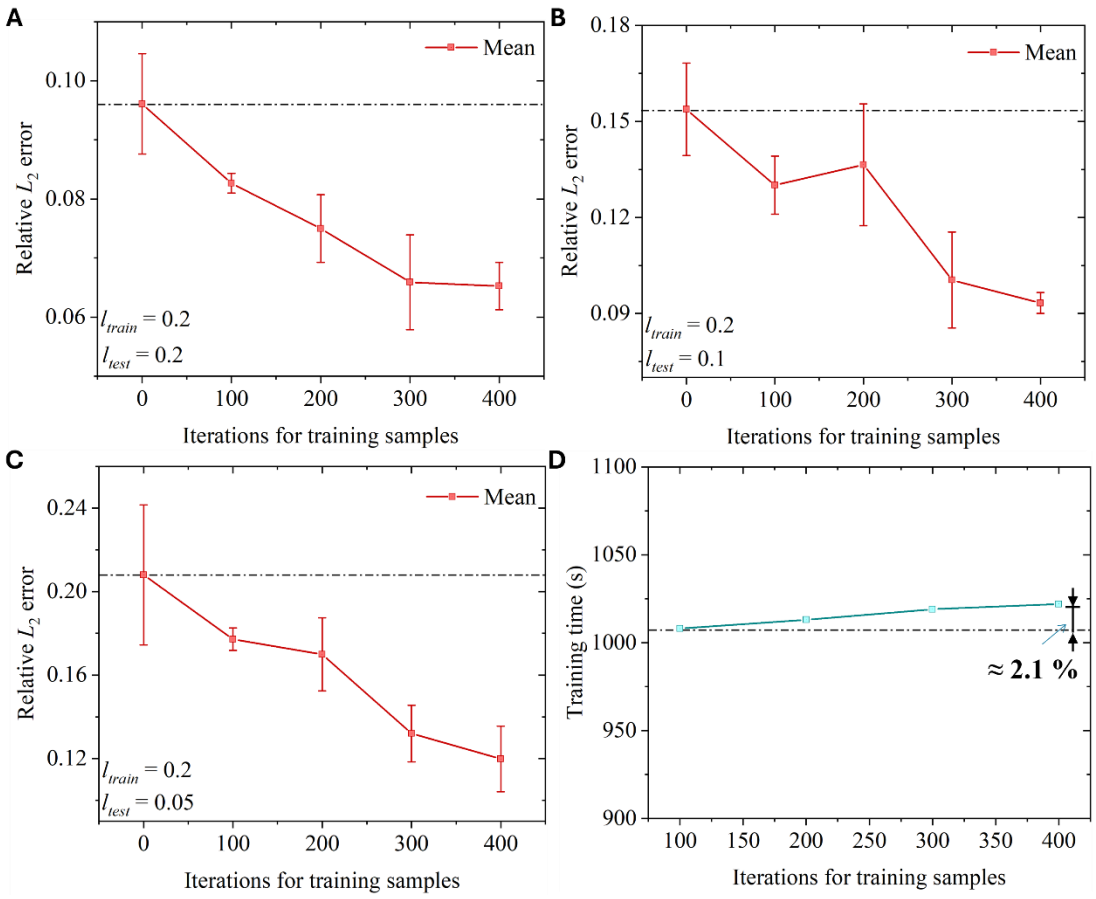


Figure 5-5 PI operator learning of the advection equation: (A) Relative L^2 error on the test dataset with the correlation length $l_{test} = 0.2$. (B) $l_{test} = 0.1$. (C) $l_{test} = 0.05$. The red lines are the mean values, and vertical bars indicate one standard deviation from eight independent runs. The black dashed lines denote the baseline results of random sampling without RAMS. (D) The training cost versus the sample training iterations in RAMS.

We further observe that the effectiveness of RAMS generally increases with the number of sample training iterations n_{RAMS}: larger n_{RAMS} can help in locating regions of higher PDE residuals, leading to a more informative resampling. RAMS introduces additional computational cost, which increases with the sample training iteration n_{RAMS}. However, this additional cost is very small compared with network training (Figure 5-5D). Even at $n_{RAMS} = 400$, the computational overhead of RAMS is about

2.1%. This is because the number of sample parameters is negligible relative to the network parameters.

Poisson equation

We consider a Poisson problem with piecewise-constant conductivity k and zero Dirichlet boundary conditions:

$$\nabla \cdot (k\nabla u) = f, \quad \text{for } x \in \Omega,$$

$$k = \begin{cases} 0.5, & \text{for } x \in \Omega', \\ 1.0, & \text{for } x \in \Omega \setminus \Omega'. \end{cases}$$

$$u = 0, \quad \text{for } x \in \partial\Omega,$$

where $\Omega = [-1,1]^2$ and $\Omega' = [-0.3,0.3]^2$ are two rectangular areas, and $f = \frac{v}{|v|}$ is a normalized function sampled via a GRF, $v \sim \mathcal{GP}(0, k_l)$, of the correlation length $l = 0.3$. A DeepONet is trained to learn the operator mapping from f to u:

$$\mathcal{G}: f \mapsto u.$$

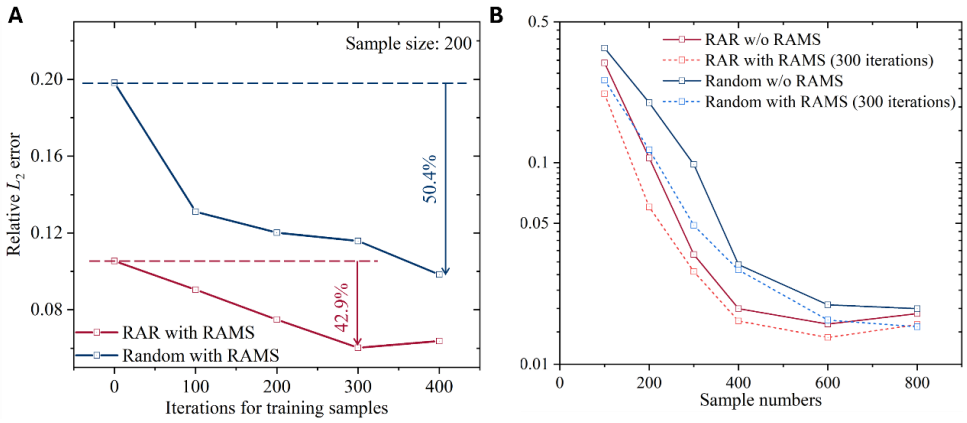


Figure 5-6 PI operator learning of the Poisson equation: (A) Relative L^2 error for different sampling methods with different sample-training iterations in RAMS. (B) Relative L^2 error for different sampling methods with different training dataset sizes.

We evaluate random sampling and RAR-G, each with and without RAMS. For random sampling, we use n_{sam} samples, while for RAR-G, we gradually increase the sample size to n_{sam}. Figure 5-6A shows the performances of different sampling methods versus the sample-training iterations in RAMS, n_{RAMS}, when fixing $n_{sam} = 200$. RAMS reduces error by more than 40\% for both random sampling and RAR-G, demonstrating its robustness across sampling strategies. However, performance does not always improve with additional iterations: for RAR-G with RAMS, the best result occurs at $n_{RAMS} = 300$ rather than $n_{RAMS} = 400$, underscoring the importance of tuning n_{RAMS}.

Next, we vary n_{sam} from 100 to 800 while fixing the sample-training iterations in RAMS at $n_{RAMS} = 300$. RAMS leads to substantial improvements across all sample sizes (Figure 5-6B), over 30% at $n_{sam} = 100$ and about 20% at $n_{sam} = 800$, indicating stable performances of the RAMS method irrespective of training dataset size. In addition, we visualize a representative ground-truth solution, the corresponding PI-DeepONet prediction from RAR-G with RAMS, and the associated piecewise error in Figure 5-11C.

One-dimensional dynamic system

In this example, we revisit an operator learning problem, where both the random sampling and the vanilla RAR resampling methods failed to efficiently generate accurate solutions, as reported in [261]. Consider a dynamic system described as follows:

$$\frac{du}{dx} = \exp(-D|\xi - 0.5|^2)\, f(x, \xi), \quad \text{for } x \in [0,1]$$

where $D = 6$ is a constant parameter, and $\xi \in [-1,1]^d$ is a coefficient vector for the source term with dimension $d = 8$. The boundary condition is given by:

$$u(x) = 0, \quad \text{for } x = 0.$$

We assume that the function f can be parameterized by a space spanned by orthogonal Chebyshev polynomials of the first kind. Denoting these polynomials by f_i, where i denotes the degree, the polynomial space of degree d is defined as:

$$\mathcal{V} = \left\{ f(x, \xi) = \sum_{i=0}^{d-1} \xi_i f_i(x) : |\xi_i| \leq 1 \right\}.$$

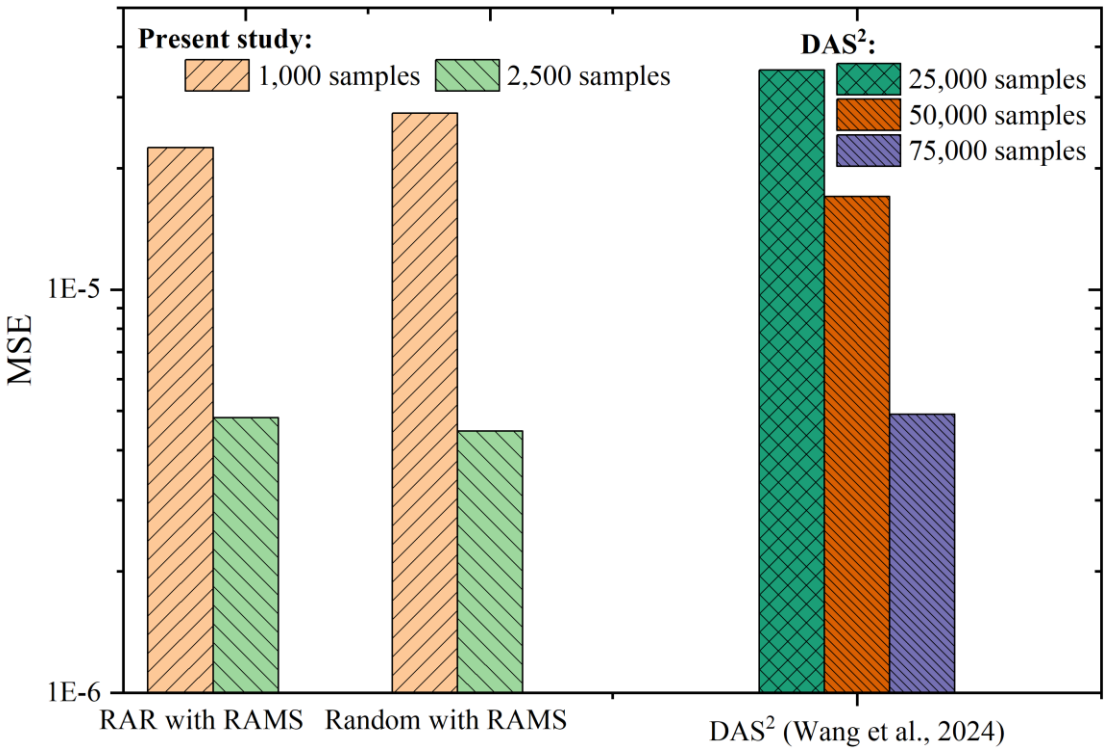


Figure 5-7 Results of PI operator learning on a dynamic system: RAR-G with RAMS and random sampling with RAMS achieve comparable accuracy to DAS^2 [261] while using significantly fewer samples. 1,000 and 2,500 samples in our methods match the performance of DAS^2 trained with 25,000 and 75,000 samples, respectively.

We test random sampling and RAR-G with RAMS. To align with the experimental

setup of [261], we use the same model architecture, training procedure, and sampling setup, differing only in the inclusion of RAMS.

To evaluate the performance of the trained DeepONet, we use a test dataset consisting of 10,000 ξ samples, uniformly sampled from a d-dimensional ball centered at 0.5 with a radius of 0.5. We compare RAR-G with RAMS and random sampling with RAMS with DAS^2 [261] in Figure 5-7. While DAS^2 can achieve accurate solutions, it requires a prohibitively large number of samples. In contrast, our RAMS method achieves similar accuracy with only about 3% of the samples required by DAS^2. Specifically, 1,000 and 2,500 samples in our method match the performance of DAS^2 trained with 25,000 and 75,000 samples, respectively. This significant reduction in sample size underscores the efficiency and effectiveness of RAMS in operator learning.

5.3.3 Examples for data-driven operator learning

The effectiveness of the proposed RAMS method in PI machine learning, including both PINN and PI operator learning, has been demonstrated in previous examples. Here, we further illustrate the performance of our RAMS method in data-driven operator learning applied to wave equation with discontinuous velocity and 2D Burgers' equation.

Wave equation with discontinuous velocity

In this example, we consider the wave equation characterized by discontinuous velocity, represented as follows:

$$\frac{\partial^2 u}{\partial t^2} - c^2(x)\frac{\partial^2 u}{\partial x^2} = 0, \quad \text{for } x \in [0,1], t \in [0,4],$$

$$u(0,t) = u(1,t) = 0, \quad \text{for } t \in [0,4],$$

$$\frac{\partial u}{\partial t}(x,0) = 0, \quad \text{for } x \in [0,1],$$

$$u(x,0) = v(x), \quad \text{for } x \in [0,1],$$

where the velocity c is discontinuous at $x = 0.7$ as

$$c(x) = \begin{cases} 1.0, & \text{for } 0 \le x < 0.7, \\ 0.5, & \text{for } 0.7 \le x \le 1.0. \end{cases}$$

We employ a DeepONet to learn the operator mapping from the initial condition to the solution:

$$\mathcal{G}: v \mapsto u$$

where v is generated from v' via $v(x) = x(1-x)v'(x)$; and v' is from a GRF with an Gaussian kernel of correlation length $l = 0.3$.

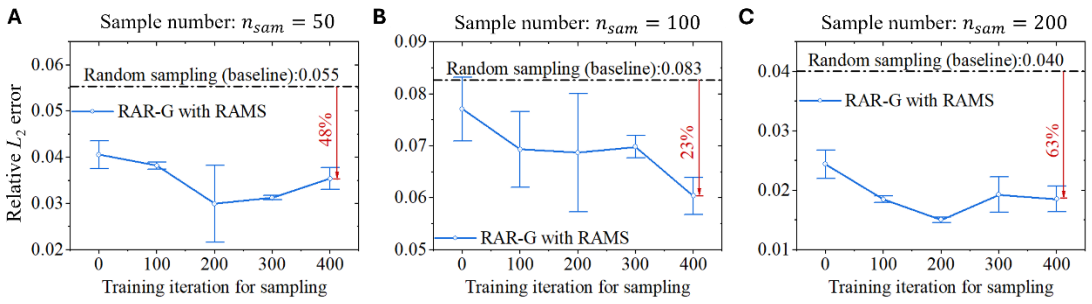


Figure 5-8 Data-driven operator learning of the wave equation with discontinuous velocity: (A) Relative L^2 error for DeepONets trained with 50 samples. (B) Relative L^2 error for DeepONets trained with 100 samples. (C) Relative L^2 error for DeepONets trained with 200 samples. The blue lines are the mean errors, and vertical bars indicate one standard deviation from three independent runs of RAR-G with RAMS. The black dashed lines correspond to the mean errors from the random sampling method.

The vanilla random sampling and RAR-G with RAMS are utilized to generate the training datasets. The total training sample sizes, n_{sam}, are varied from 50 to 200. In RAR-G with RAMS, the iterations for sample-training, n_{RAMS}, in each resampling

stage are varied from 0 to 400. Detailed setups for different sampling methods can be found in Appendix B.

The mean relative L^2 errors for different settings are shown inFigure 5-8. The RAMS method achieves significant accuracy improvements, up to 63%, using the same number of training samples as the baseline method (random sampling). This substantial improvement confirms the effectiveness of RAMS in data-driven operator learning. We also observe that the error reduction from RAMS does not necessarily increase with the number of sampling–training iterations n_{RAMS}. For example, when $n_{sam} = 100$ (Figure 5-8B), the best performance occurs at $n_{RAMS} = 200$ rather than $n_{RAMS} = 400$, underscoring the need to tune n_{RAMS}. Finally, the effectiveness of RAMS increases with the total training-sample size: the error reduction rises from 23% at $n_{sam} = 50$ (Figure 5-8A) to 63% at $n_{sam} = 200$ (Figure 5-8C), suggesting that more samples allow the physics residual to more accurately reflect the model's training adequacy. In addition, a representative example of an input function, ground-truth solution, DeepONet prediction from RAR-G with RAMS, and pointwise error are shown in Figure 5-12.

Two-dimensional Burgers' equation

In this example, the two-dimensional Burgers' equation [268] is considered

$$\frac{\partial u}{\partial t} + u\left(\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2}\right) = \nu \Delta u, \quad \text{for } (x_1, x_2) \in \Omega, t \in [0,1],$$

$$u(x_1, x_2, 0) = v(x_1, x_2), \quad \text{for } (x_1, x_2) \in \Omega,$$

$$u(x_1, x_2, t) = 0, \quad \text{for } (x_1, x_2) \in \partial\Omega, t \in [0,1],$$

where ν is the viscosity coefficient; and $\Omega = [-1,1]^2$ is the analysis domain. A DeepONet is trained to approximate the operator mapping from the initial condition to the solution as

$$\mathcal{G}: v \mapsto u$$

where v is generated from v' via $v(x_1, x_2) = x_1(1 - x_1)x_2(1 - x_2)v'(x_1, x_2)$; and v' is sampled from the Gaussian process with the Gaussian kernel of the correlation length equal to l.

We consider two cases, differentiated by their viscosity coefficients, correlation lengths, and training sample sizes. These variations represent different complexities of the operator learning problem.

- Case 1 (small viscosity and sample size): The viscosity coefficient and correlation length are set to $\nu = 0.02$ and $l = 0.3$, respectively. The sample size n_{sam} ranges from 600 to 1,200.
- Case 2 (large viscosity and sample size): The viscosity coefficient and correlation length are set to $\nu = 0.1$ and $l = 0.5$, respectively. The sample size n_{sam} ranges from 1,500 to 3,000.

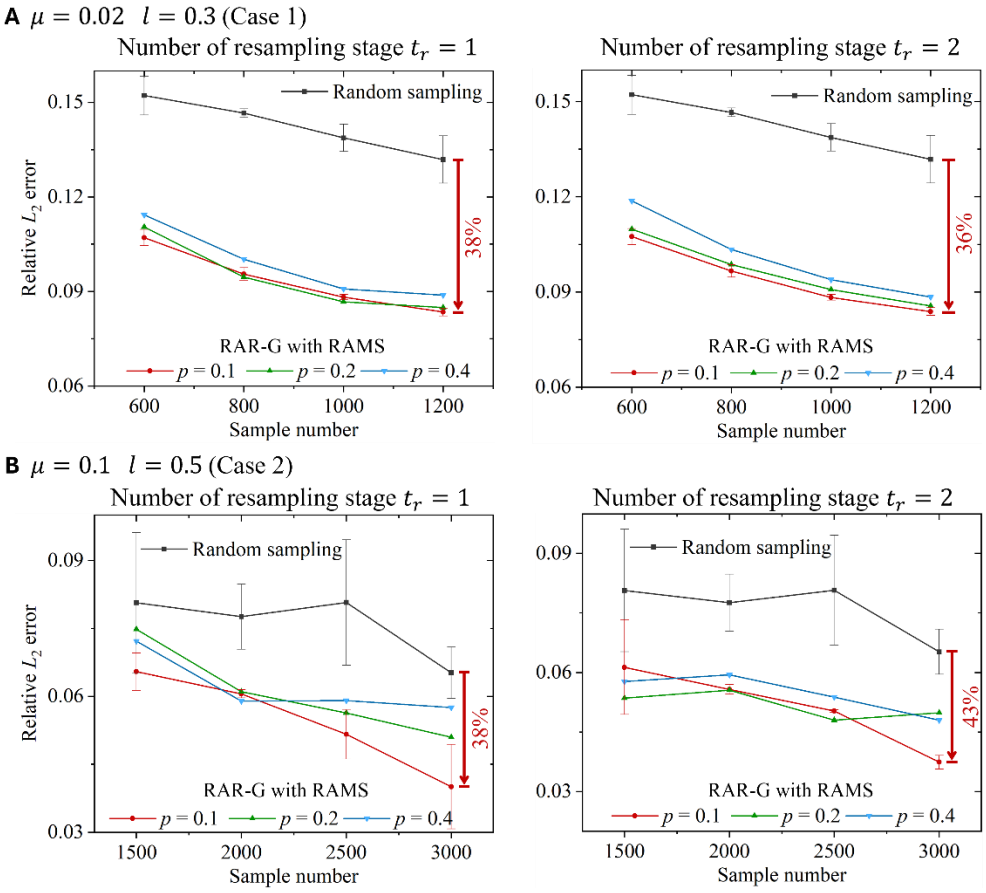


Figure 5-9 Data-driven operator learning of the two-dimensional Burgers' equation. (A) Relative L^2 errors of Case 1 for different trainable sample sizes p and numbers of resampling stages t_r. (B) Relative L^2 errors of Case 2. For clarity, the one standard deviation from three independent runs is shown only for the random sampling and RAR-G with RAMS of $p = 0.1$.

We compare the vanilla random sampling method and the RAR-G with RAMS across different settings. For RAR-G with RAMS, $n_{ini} = (1-p)n_{sam}$ samples with $p \in \{0.1, 0.2, 0.4\}$ are initially generated. In each resampling stage of RAMS, $m = \frac{p}{t_r} n_{sam}$ additional samples chosen from $M = 1000$ candidates are added to the training dataset, where $t_r \in \{1, 2\}$ represents the number of resampling stages. In RAMS, the samples are updated using the Adam optimizer for $n_{RAMS} = 100$ iterations at each resampling stage.

The performance of different sampling methods are evaluated on the mean relative L_2 error (Figure 5-9). Across all settings, the largest error reduction occurs at $p = 0.1$, suggesting that allocating more samples early helps the model capture the underlying physics and makes the physics residuals more indicative of training adequacy. The robustness of RAMS is further demonstrated by its consistent performance, reducing errors by approximately 40% in both a complex case (Case 1; Figure 5-9A) and a simpler one (Case 2; Figure 5-9B). Additionally, Figure 5-13 shows a representative example of a ground-truth solution, the prediction of a DeepONet from RAR-G with RAMS, and the associated pointwise error.

Appendix A - Network Architectures

We provide the network architectures used in all the examples. The hyperbolic tangent function is used as the activation function for all the models employed. Table 5-1 summarizes the network architectures for the PINN problems in Section 3.1. Table 5-2 lists the DeepONet architectures for the operator learning problems in Sections 3.2-3.3. In Section 3.2.2, the DeepONet branch and trunk networks adopt the modified MLP architecture proposed in [134] with better performance.

Table 5-1 Network architectures for PINN problems.

Problem	Network
Section 3.1.1 1D Burgers' equation	MLP (3 × 100)
Section 3.1.2 1D wave equation	MLP (3 × 100)
Section 3.1.3 2D Poisson equation	MLP (3 × 100)
Section 3.1.4 High-dimensional PDE	MLP (3 × 100)

(Note: MLP ($a \times b$) denotes a MLP with a hidden layers and b neurons per hidden layer.)

Table 5-2 DeepONet architectures for operator learning problems.

Problem	Branch net	Trunk net

Section 3.2.1 Diffusion-reaction equation	MLP (3 × 150)	MLP (3 × 150)
Section 3.2.2 Advection equation	Modified MLP (3 × 150)	Modified MLP (3 × 150)
Section 3.2.3 Poisson equation	CNN	MLP (3 × 350)
Section 3.2.4 Dynamic system	MLP (3 × 50)	MLP (3 × 50)
Section 3.3.1 Wave equation	MLP (5 × 150)	MLP (5 × 150)
Section 3.3.2 2D Burgers' equation	CNN	MLP (4 × 350)

Appendix B - Sampling and Training Hyperparameters

This section summarizes the training and sampling configurations used in Section 3. For network training, we employ Adam with a learning rate of 10^{-3} and L-BFGS. For sample training in RAMS, we use Adam with a learning rate of 10^{-2}. Other setups for network training and sampling are summarized below.

- **PINN in Sections 3.1.1-3.1.3:** The PINN for the Burgers' equation, the wave equation, and the Poisson equation use the same setups for network training and sampling. For all sampling methods, the number of resampling stages and the Adam epochs per stage are set to $t_r = 75$ and $n_{train} = 200$, respectively. After the final resampling stage, the model is further trained for 5,000 Adam epochs, followed by 1,000 L-BFGS iterations. The other hyperparameters for sampling are listed in Table 5-4.
- **PI operator learning for the diffusion-reaction equation in Section 3.2.1:** For all sampling methods, the number of resampling stages and the Adam epochs per stage are set to $t_r = 50$ and $n_{train} = 1{,}000$, respectively. After the final resampling stage, the model is further trained for 10,000 Adam epochs, followed by 2,000 L-BFGS iterations. The collocation points for each function are in a 100×100 uniform grid. The other hyperparameters for sampling are listed in Table 5-4.
- **PI operator learning for the advection equation in Section 3.2:** For all sampling methods, the number of resampling stages and the Adam epochs per stage are set to $t_r = 20$ and $n_{train} = 5{,}000$, respectively. After the final resampling stage,

the model is further trained for 50,000 Adam epochs, followed by 2,000 L-BFGS iterations. All function samples use the same set of 10,000 collocation points, uniformly sampled over the domain. The other hyperparameters for sampling are listed in Table 5-5.

- **PI operator learning for the Poisson equation in Section 3.2.3:** For all sampling methods, the number of resampling stages and the Adam epochs per stage are set to $t_r = 25$ and $n_{train} = 5{,}000$, respectively. After the final resampling stage, the model is further trained for 25,000 Adam epochs, followed by 2,000 L-BFGS iterations. All function samples use the same set of 2,500 collocation points, uniformly sampled over the domain. The other hyperparameters for sampling are listed in Table 5-6.
- **PI operator learning for the dynamic system in Section 2.4:** For all sampling methods, the number of resampling stages and the Adam epochs per stage are set to $t_r = 30$ and $n_{train} = 500$, respectively. All function samples use the same set of 100 collocation points, uniformly sampled over the domain. The other hyperparameters for sampling are listed in Table 5-7.
- **Data-driven operator learning for the wave equation in Section 3.3.1:** In RAR-G with RAMS, the number of resampling stages is fixed at $t_r = 2$. In RAMS, the number of sample-training iterations n_{RAMS} is varied from 0 to 400. The other hyperparameters for sampling are listed in Table 5-8.

Table 5-3 Sampling setups in Sections 3.1.1–3.1.3

Sampling method	**Hyperparameters**
Random / LHS / Halton	$\lvert\mathcal{T}\rvert = 5{,}000$, $\lvert\mathcal{T}_1\rvert = 4{,}500$, $\lvert\mathcal{T}_2\rvert = 500$, $\lvert\hat{\mathcal{T}}\rvert = 50$, $n_{RAMS} = 10$
RAR-G / RAR-D	$n_{ini} = 2{,}500$, $M = 1{,}000$, $m = 40$, $n_{RAMS} = 5$
R3	$\lvert\mathcal{T}\rvert = 2{,}500$, $n_{RAMS} = 5$

Table 5-4 Sampling setups in Section 3.2.1

Sampling method	Hyperparameters
Random	$\lvert\mathcal{T}\rvert = 600$, $\lvert\mathcal{T}_1\rvert = 300$, $\lvert\mathcal{T}_2\rvert = 300$, $\lvert\hat{\mathcal{T}}\rvert = 60$, $n_{RAMS} = 50$
RAR-G	$n_{ini} = 800$, $M = 100$, $m = 8$, $n_{RAMS} = 50$

Table 5-5 Sampling setups in Section 3.2.2

Sampling method	Hyperparameters
Random	$\lvert\mathcal{T}\rvert = 200$, $\lvert\mathcal{T}_1\rvert = 180$, $\lvert\mathcal{T}_2\rvert = 20$, $\lvert\hat{\mathcal{T}}\rvert = 2$, $n_{RAMS} \in \{0,100,200,300,400\}$

Table 5-6 Sampling setups in Section 3.2.3

Sampling method	Hyperparameters
Random	$\lvert\mathcal{T}\rvert = n_{sam}$, $\lvert\mathcal{T}_1\rvert = 0.5n_{sam}$, $\lvert\mathcal{T}_2\rvert = 0.5n_{sam}$, $\lvert\hat{\mathcal{T}}\rvert = 0.05n_{sam}$
RAR-G	$n_{ini} = 0.7n_{sam}$, $M = 0.3n_{sam}$, $m = 0.02n_{sam}$

Table 5-7 Sampling setups in Section 3.2.4

Sampling method	Hyperparameters
Random	$\lvert\mathcal{T}\rvert = n_{sam}$, $\lvert\mathcal{T}_1\rvert = \lvert\mathcal{T}_2\rvert = 0.5n_{sam}$, $\lvert\hat{\mathcal{T}}\rvert = 0.05n_{sam}$, $n_{RAMS} = 400$
RAR-G	$n_{ini} = 0.4n_{sam}$, $M = 1000$, $m = 0.02n_{sam}$, $n_{RAMS} = 400$

(Note: Two sample sizes, $n_{RAMS} \in \{1000,2500\}$, are considered.)

Table 5-8 Sampling setups in Section 3.3.1

Sampling method	Hyperparameters
Random without RAMS	$\lvert\mathcal{T}\rvert = n_{sam}$
RAR-G with RAMS	$n_{ini} = 0.6n_{sam}$, $M = 1000$, $m = 0.2n_{sam}$

(Note: Two sample sizes, $n_{RAMS} \in \{50,100,200\}$, are considered.)

Appendix C - Visualization of Results

Some visualization results from our experiments are provided here. Figure 5-10 displays the analysis results from the RAR resampling method with RAMS applied to PINN when solving Burger's equation, the wave equation, and the Poisson equation, respectively. Additionally, Figure 5-11 showcases some representative results from the PI operator learning model, which was trained using our proposed RAMS methods. Then, Figure 5-12 and Figure 5-13 present visualizations from the data-driven DeepONet, trained with the RAR with RAMS, on predicting solutions for the wave equation with discontinuous velocity and the two-dimensional Burger's equation.

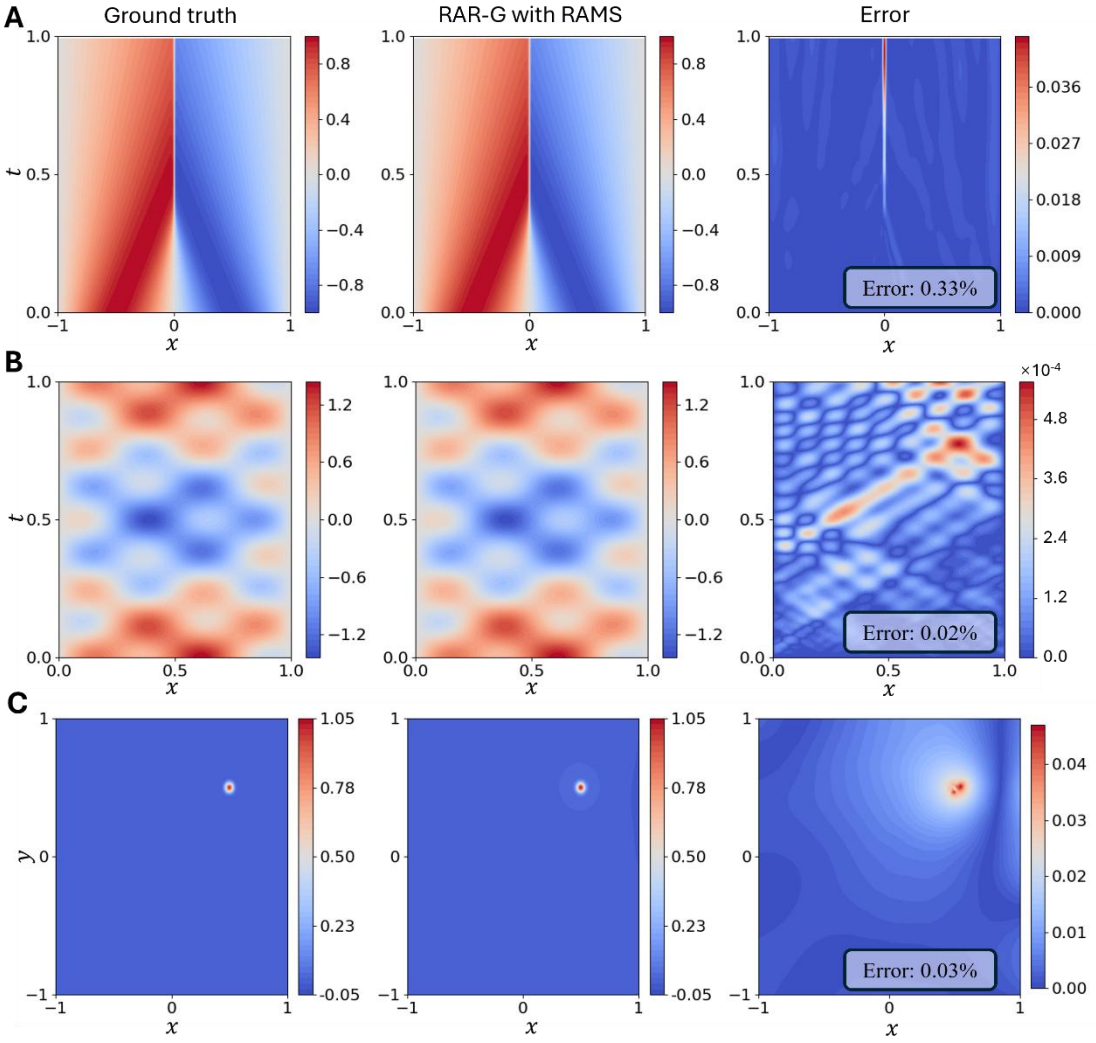


Figure 5-10 Visualization of PINN results: (A), (B), and (C) illustrate the results for solving Burger's equation, wave equation, and Poisson equation, respectively.

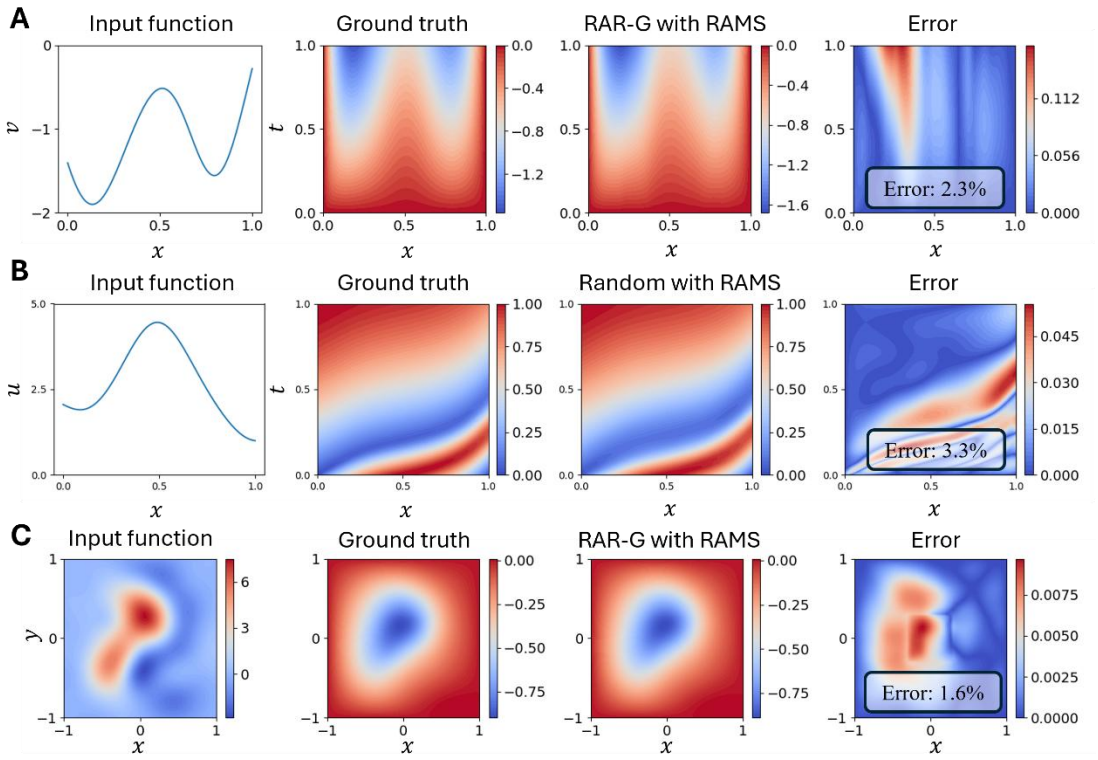


Figure 5-11 Representative visualization of PI operator learning: (A), (B), and (C) illustrate the results for solving Diffusion-reaction equation, adjective equation, and Poisson equation, respectively

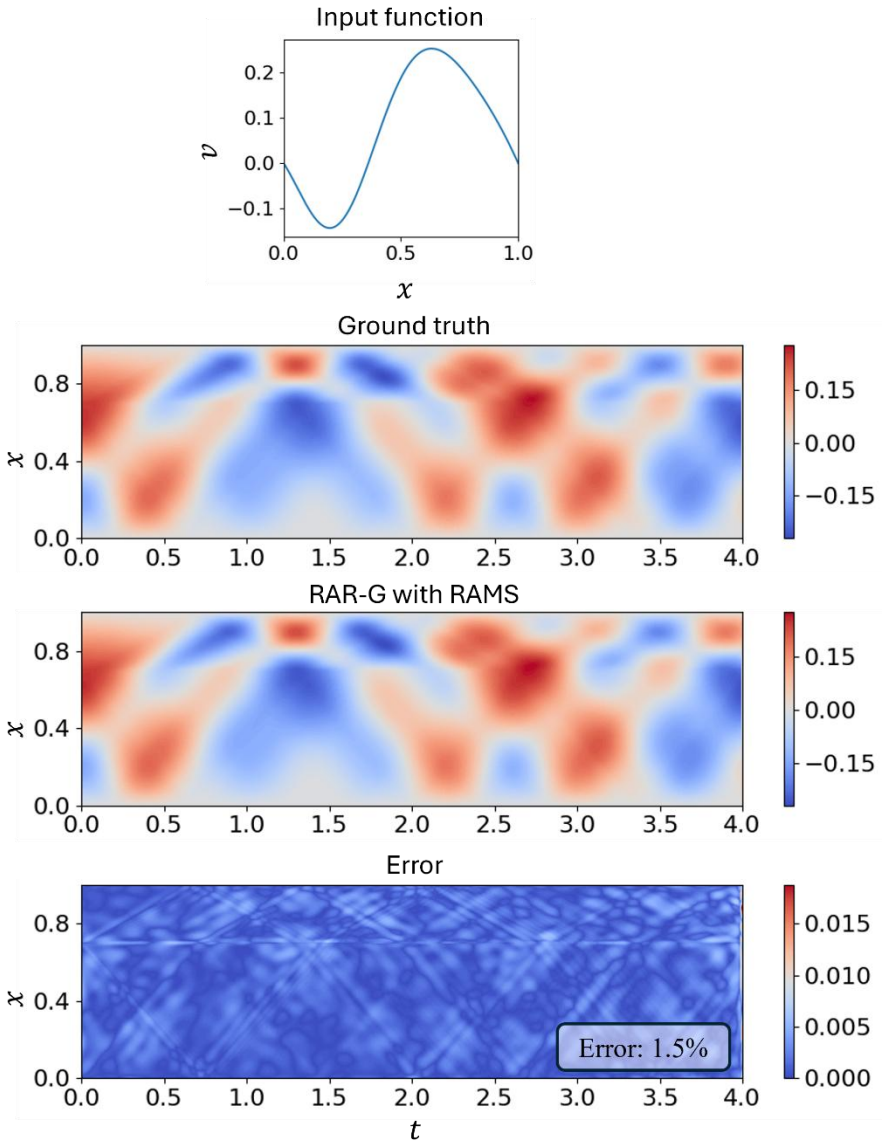


Figure 5-12 Representative visualization of wave equation with discontinuous velocity in data-driven operator learning.

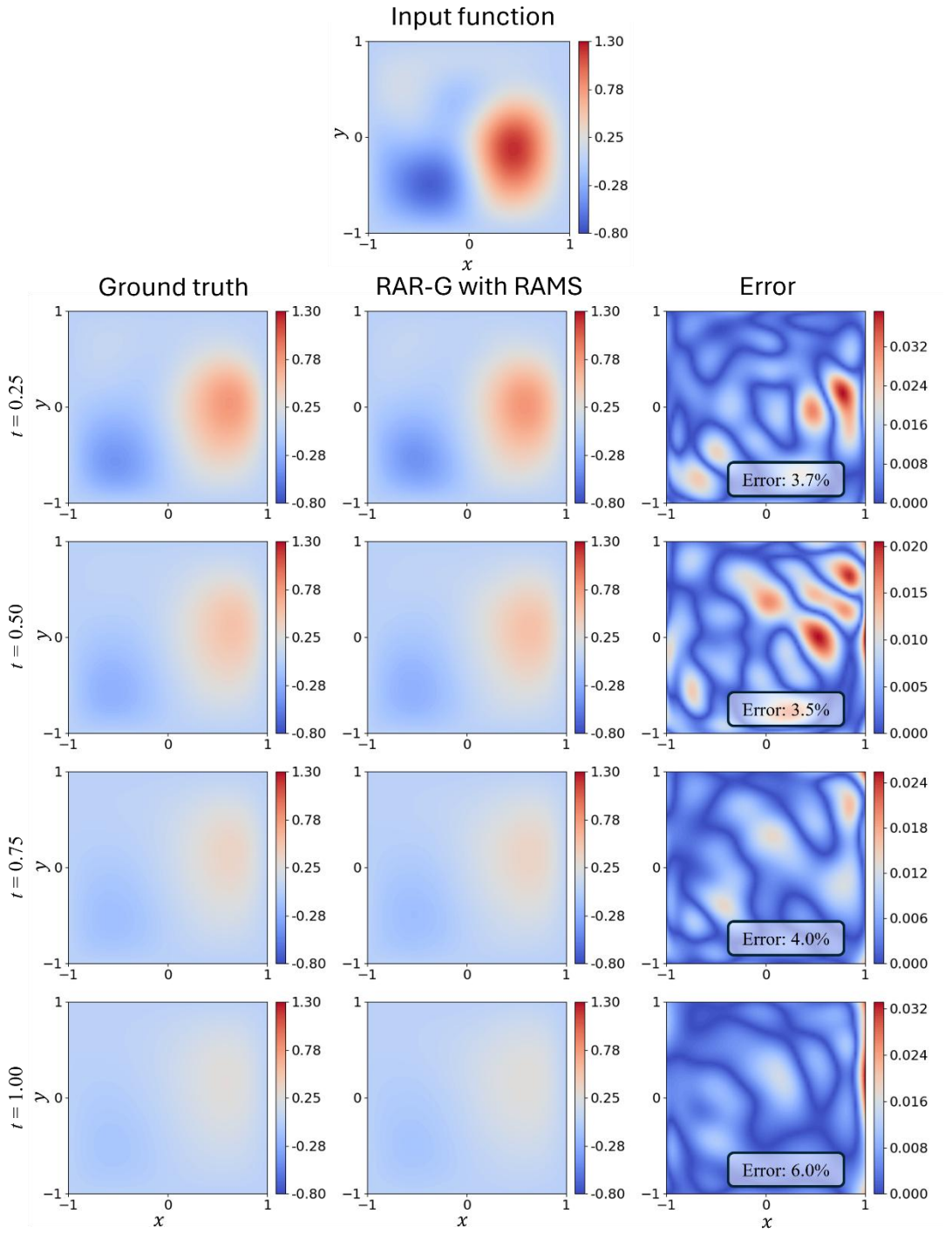


Figure 5-13 Representative visualization of two-dimensional Burger's equation in data-driven operator learning. The solutions at different time steps are plotted in different rows.

Appendix D - Projector in RAMS

We want to demonstrate the necessity of the projector in RAMS for PI operator

learning of the diffusion-reaction equation (Section 3.1). We use the same experimental settings as in Section 3.1, except for the removal of the function projector component in RAMS. Figure 5-14 compares the trained function samples during RAMS with and without the kernel smoothing method as the projector. Samples adjusted without the projector exhibit significant irregular fluctuations, in stark contrast to the stable profiles obtained when using the projector, indicating the importance of the kernel smoothing method in RAMS.

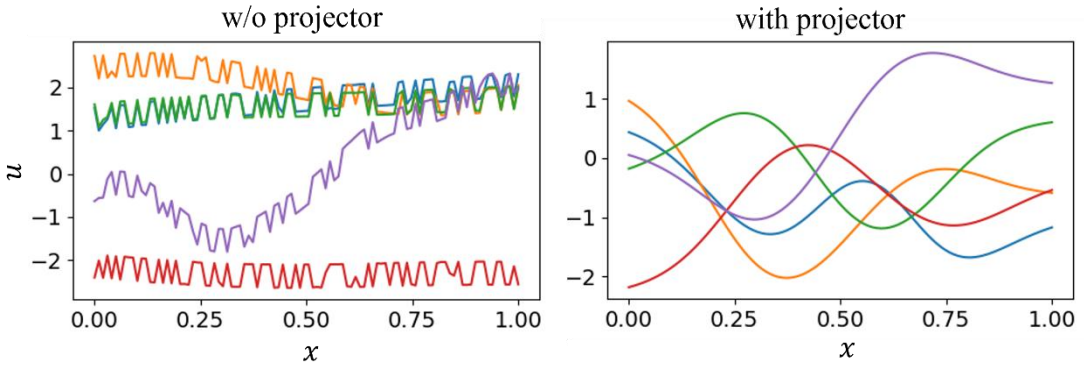


Figure 5-14 Function samples optimized by RAMS with and without projector for the advection equation in PI operator learning: For each case, five function samples are visualized.

Chapter 6. GOVERNING EQUATIONS FOR SINGLE PILES AND PINN IMPLEMENTATION

6.1 Introduction

Large deflection analysis for slender piles considering nonlinear Soil-Pile Interaction (SPI) is an essential yet challenging aspect of geotechnical engineering. Currently, this large deflection analysis commonly relies on the Winkler model [269, 270] (Figure 6-1A), characterizing the pile as a beam surrounding by lateral soil springs defined by *p-y* curves [160, 271]. A standard solution technique for this model is the Line Finite Element Method (LFEM) [161, 179, 272, 273] (Figure 6-1A), which represents the pile with line elements and the surrounded soil with nonlinear springs at element nodes. Despite its robustness and accuracy, the computational procedure of the LFEM is tedious, given the element mesh must be dense to accurately capture the complex SPI variated along the pile [179]. Consequently, when costly repetitive simulations are required, such as for reliability assessment [274, 275] and design optimization [276-278], the computational cost of the LFEM may become prohibitively high [121, 279, 280].

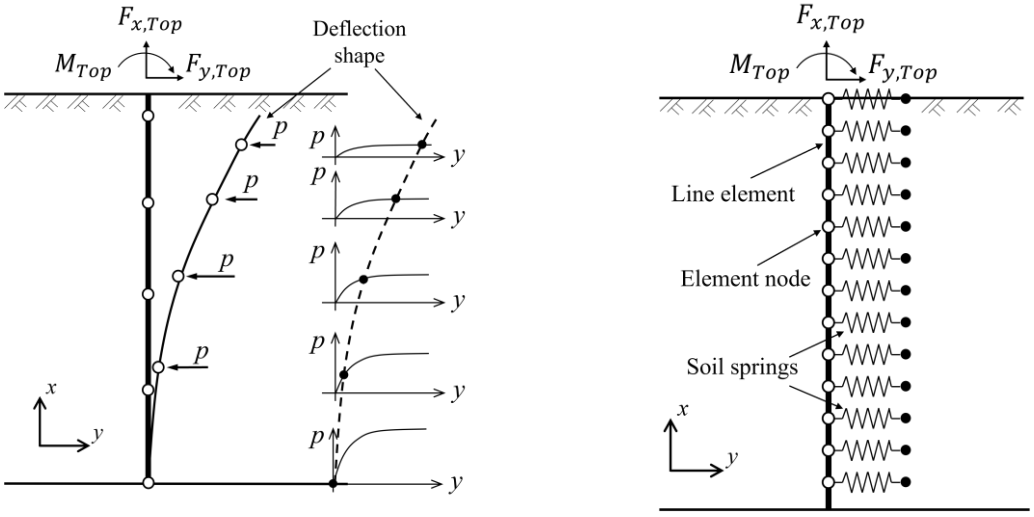
To enhance the computational efficiency of the analysis, machine learning (ML) models, such as the neural networks (NN), have been initially utilized to predict the analysis results without the tedious computational process in recent research [281-285]. Nevertheless, conventional ML model training typically relies on pre-collected data, which can be challenging to obtain through expensive physical tests or time-consuming numerical analyses [279, 286, 287], rendering their application sometimes ineffective. Additionally, the black-box nature of ML models raises concerns about their reliability, as they are not trained based on clear physical or mechanical laws. To address these shortcomings, extensive efforts have been made for a more effective and interpretable ML model implementation process.

Recently, Raissi et al. [288] proposed an innovative ML paradigm, named the physics-informed neural networks (PINN), to encode the NN model of the underlying physical laws governing the studied problem. Unlike traditional NN training, the PINN constructs the loss function using the model output and its derivatives in accordance with the residual terms of the governing equations, eliminating the need for rich pre-collected data. The integration of physical constraints and sample-free features has made PINN a popular subject of research [289], such as inverse problems [80, 290, 291], stochastic analysis [292], and meta-modelling [96, 121]. For instance, Zhang et al. [293] applied PINN to reconstruct the governing equations of Terzaghi consolidation, confirming its robustness in addressing inverse problems. Similarly, Depina et al. [291] leveraged PINN for inverse analysis of unsaturated groundwater flow, effectively identifying parameters of the constitutive model. More recently, Chen et al. [32] utilized the PINN to effectively predict the water flow in unsaturated soils governed by the intricate Richardson-Richards equation. All these studies highlight the potential of PINN in solving pile-related emerging concerns such as damage detection [294, 295], reliability assessment [274, 275], and design optimization [276-278]. Even though preliminary investigation of PINN for pile assessment has been conducted [296], it does not comprehensively capture all the crucial factor for pile response, such as geometric nonlinearity, impeding its application. To fully realize the potential of PINN, the first step is to develop an accurate and reliable PINN framework for the large deflection analysis of piles considering nonlinear SPI.

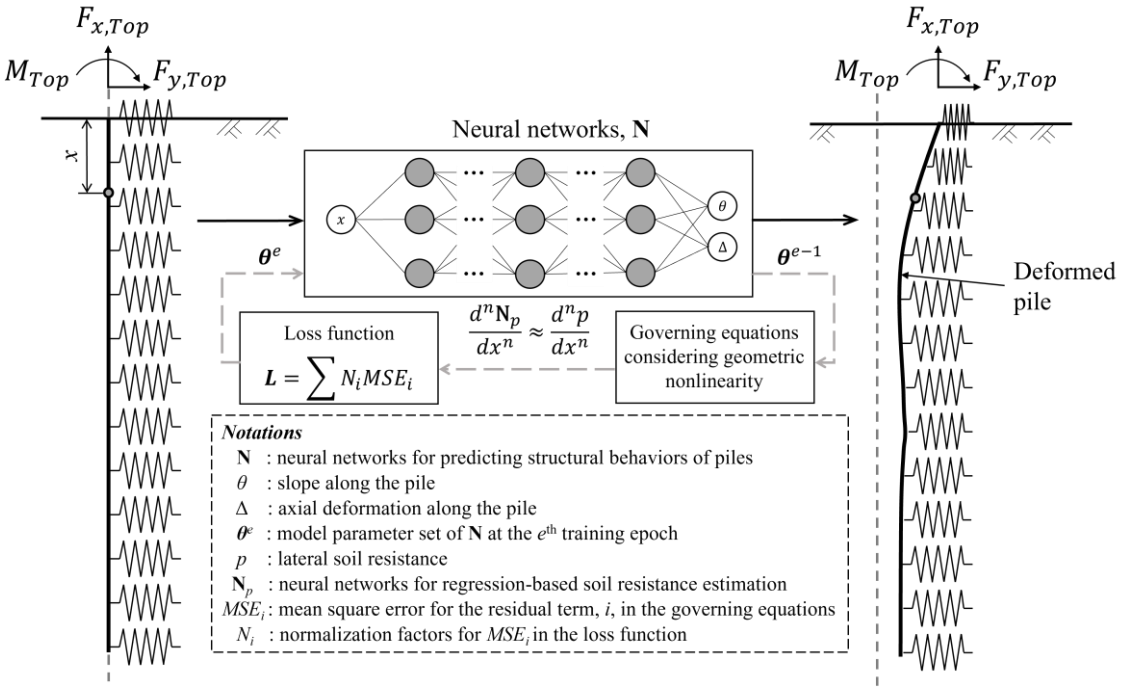
The extension of the PINN to the large deflection analysis of slender piles presents several challenges that must be identified. Conventional governing equations for piles [180, 297] are formulated based on small-deflection assumptions, overlooking geometric nonlinearity [46], which can substantially underestimate pile deflection, especially when evaluating pile stability [179] or considering soil liquefication [161]. Another issue is that severe convergence problems might be encountered when considering some complex *p-y* relations described by non-differentiable functions,

which are constructed by joining the recorded data points but not expressed in an analytical form [155] or have derivatives trends to infinity at certain points [158]. When encountering such functions, the automatic differentiation technique [298] may struggle to accurately and robustly compute the loss function derivatives for updating the model parameters, leading to the failure of the model training.

Apart from the above two issues, the current loss function also limits the practical PINN application. Typically, the loss function, defined as the Mean Squared Error (MSE) at sampling points, ends training when its value falls below a certain tolerance. Despite its widespread use, this unnormalized loss function can present challenges in predicting solutions of varying magnitudes. When the magnitude of the exact solution is small, the relative error in the NN prediction may still be substantial, even if the tolerance is satisfied. Conversely, the loss function tolerance may be unattainable when the solution magnitude is small. Additionally, varying magnitudes of unnormalized loss terms may also cause severe training instability. Thus, appropriate loss function normalization is crucial for effective PINN implementation.



(A) (B)



(C)

Figure 6-1 Large deflection analysis of slender piles considering nonlinear SPI: (A) Slender piles undergoing large deflection; (B) Line Finite element method; (C) Present study

This chapter proposes a large deflection analysis method of slender piles via the PINN (Figure 6-1C) by addressing the aforementioned problems. The partial differential equations (PDEs) for the structural behaviors of slender piles accounting for the geometric nonlinearity are derived, based on which the corresponding PINN framework is built. Within the PINN framework, a series of factors are introduced to normalize the loss function that calibrates the error of the NN prediction. Additionally, a regression via the external NN model is built to estimate lateral soil resistance along the pile during training to tackle the convergence issue when encountering non-differentiable *p-y* relations. The proposed method is validated through extensive examples and case studies to demonstrate its potential for practical application and highlight the importance of the developed numerical strategies.

6.2 Governing Equations

This chapter proposed a large deflection analysis method for slender piles accounting for the nonlinear SPI via the PINN, where the accuracy of the underlying physical information in neural networks is crucial for achieving reliable solutions. The detailed derivation of the governing equations describing the physical information is given in the following section.

6.2.1 Assumptions

The following assumptions are employed, including:

- The material of piles is homogenous and isotropic while following Hooke's material laws.
- The pile deformation follows the Euler-Bernoulli beam theory, and the shear effect of large-diameter piles is ignored.
- The lateral soil resistance along the pile can be described by the *p*-*y* relation.

6.2.2 *p*-*y* relation

The lateral soil resistance distribution along the pile is complex. In geotechnical practice, one standard method for describing the lateral soil resistance at the specific embedded depth is the *p*-*y* relation, in which p and y denote the lateral soil resistance and the lateral pile deflection, respectively. Once ensuring the geological condition, such as the ground stratigraphy and the soil properties, the corresponding *p*-*y* relations at different embedded depths can be constructed according to the past literature [160, 271] or the local specification [155].

6.2.3 Mechanical model

In the mechanical model, the expression about the deflection and internal force along the pile are formulated. The deflection of the pile about the x- and y-axis, u and y, are expressed using the axial compression, Δ, and the slope, θ, along the pile. Based on the derived pile deflection formulas, the expressions of the internal force distribution are formulated.

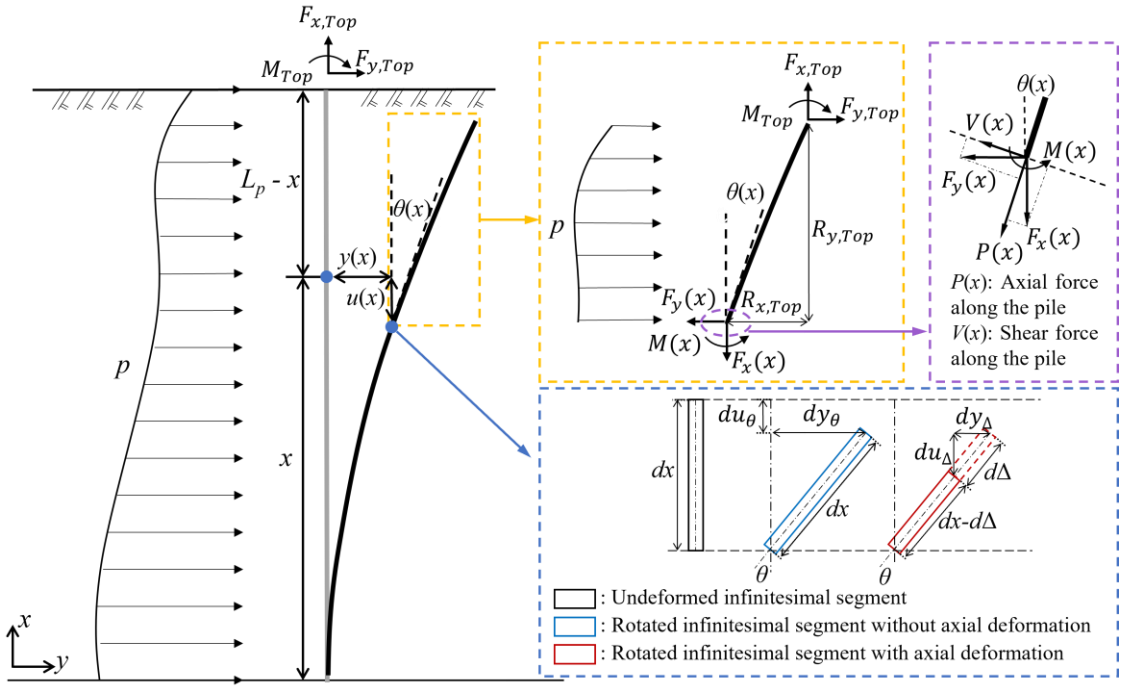


Figure 6-2 Mechanical model of slender pile undergoing large deflection

As shown in Figure 6-2, the pile deflection along the pile on the x- and y-axial can be divided into two parts, including the part induced by the rotation and the other resulted from the axial deformation of the rotated pile segment. Thus, the pile deflection on two axials, u and y, can be written as:

$$u(x) = \int_0^x du_\theta + \int_0^x du_\Delta$$

$$y(x) = \int_0^x dy_\theta + \int_0^x dy_\Delta$$

where,

$$du_\theta = -(1 - cos\,\theta)dx$$

$$du_\Delta = cos\,\theta\, d\Delta$$

$$dy_\theta = sin\,\theta\, dx$$

$$dy_{\Delta} = \sin\theta \, d\Delta$$

When the pile deflection along the pile is obtained, the force equilibrium conditions of the pile segment above the location x (as illustrated in Figure 6-2) can be established:

$$M(x) = -F_{x,Top}\left[y|_{x=L_p} - y(x)\right] + F_{y,Top}\left[u|_{x=L_p} + L - u(x) - x\right] + M_{Top} + M_S(x)$$

$$F_x(x) = F_{x,Top}$$

$$F_y(x) = F_{y,Top} + F_S(x)$$

where M denotes the bending moment along the pile; F_x and F_y are the internal force about the x- and y-axial along the pile, respectively; $F_{x,Top}$, $F_{y,Top}$ and M_{Top} are the applied forces and bending moment at the top end of the pile; M_S and F_S are the forced caused by the lateral soil resistance; and L_p is the pile length. When the p-y relations are given, M_S and F_S can be written as:

$$M_S(x) = \int_x^{L_p} [t + u(t) - x - u|_{t=x}]p(t)dt$$

$$F_S(x) = \int_x^{L_p} p(t)dt$$

Then, by applying the parallelogram law, the axial force, P, and the shear force, V, along the pile can be formulated:

$$P(x) = \cos\theta \, F_x(x) - \sin\theta \, F_y(x)$$

$$V(x) = \sin\theta \, F_x(x) + \cos\theta \, F_y(x)$$

6.2.4 Governing equations

According to the Euler- Bernoulli assumption, which regarding the cross-section of the pile remains plane after deformation, the slope along the pile can be expressed as:

$$\frac{M(x)}{EI} = \frac{d\theta(x)}{dx}$$

where E represents the elastic modulus of the pile material; and I denotes the moment of inertia of the pile cross-section.

When the Hooke's law is validated, the axial compression along the pile is governed by the axial force as:

$$\frac{P(x)}{EA} = \frac{d\Delta(x)}{dx}$$

where A is the cross-section area of the pile.

6.2.5 Boundary condition

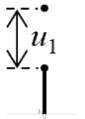
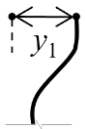
For a slender pile, the slope and deflection at the pile bottom is negligible that could be ignored [179]. Thus, the pile is assumed to be fixed at the bottom end in the present study. Consequently, the corresponding mathematical expression of the boundary conditions at the bottom end of the pile can be written as:

$$\Delta|_{x=0} = 0$$

$$\theta|_{x=0} = 0$$

For the top end of the pile, there are totally three cases of boundaries, whose corresponding expressions are summarized in Table 6-1.

Table 6-1 Mathematical expressions for boundary conditions

Boundary*	Fixed	Free
	$u\|_{x=L_p} = 0$	$EA\left(\cos\theta\frac{d\Delta}{dx}\right)\Big\|_{x=L_p} - EI\left(\sin\theta\frac{d^2\theta}{dx^2}\right)\Big\|_{x=L_p} = F_{x,Top}$
	$y\|_{x=L_p} = 0$	$EA\left(\sin\theta\frac{d\Delta}{dx}\right)\Big\|_{x=L_p} + EI\left(\cos\theta\frac{d^2\theta}{dx^2}\right)\Big\|_{x=L_p} = F_{y,Top}$

	$\theta\|_{x=L_p} = 0$	$EI\dfrac{d\theta}{dx}\Big\|_{x=L_p} = M_{Top}$

(*Note: The subscript $_1$ denotes the top end of the pile)

6.2.6 Existing solution

It may be challenging to derive the exact solutions for the above PDEs given large numbers of integral terms are contained and the complexity of the *p-y* relations. To simplify the governing equation, two assumptions are typically made: linearization of the SPI and neglect of the geometric nonlinearity of the pile. Accordingly, the equation governing lateral pile deflection can be re-formulated as:

$$p(x) = -ky(x)$$

$$\frac{d^4y}{dx^4} = p(x)$$

where *k* is the soil modulus denoting the soil stiffness.

Consequently, the corresponding analytical solutions of pile structural behavior can be obtained [297]. However, the relationship between soil resistance and pile deflection used in design practice, such as the clay *p-y* relations recommended by API [155], can exhibit significant nonlinearity. Besides, previous studies have highlighted that lateral pile deflection can be considerable when assessing the stability of slender piles [179] or evaluating soil liquefaction [161]. Therefore, it is imperative to appropriately account for the geometric nonlinearity of the pile. Otherwise, the lateral pile deflection may be significantly underpredicted [21].

The Winkler model solved by the LFEM is a reliable approach for the structural analysis of piles undergoing large deflections. This approach accurately captures geometric nonlinearity by kinetic descriptions, such as the Updated-Lagrangian and Co-rotational descriptions [299]. Additionally, the complex SPI is simulated using

concentrated spring elements constructed based on selected *p-y* relations. However, the complex lateral soil resistance distribution may require a fine mesh, leading to a relatively time-consuming analysis process. Thus, this chapter investigates an alternative solution, the PINN, for solving the governing equation and conducting large deflection analysis of piles while accounting for nonlinear SPI.

6.3 Physics-informed Neural Networks Implementation

In this research, the governing equations are solved using an innovative solution, the PINN. The NN model employed to approximate the exact solutions of the governing equations is presented. Then, the loss function calibrating the NN model accuracy is constructed according to the residuals of the governing equations and normalized by a series of normalization factors.

6.3.1 Neural networks model

In the PINN, a fully-connected NN model, **N**, is employed to approximate the exact solution of the governing equations, $\boldsymbol{y} := (\theta, \quad \Delta)$, as:

$$N(\boldsymbol{x}^0; \boldsymbol{\theta}) := (\theta_{\mathrm{N}}, \ \Delta_{\mathrm{N}}) \approx \boldsymbol{y}(\boldsymbol{x}^0)$$

where $(\theta_N, \quad \Delta_N)$ denotes the NN model output to approximate the exact solution, $\boldsymbol{y}$; and $\boldsymbol{x}^0$ is the input feature that denotes the normalized embedded depth:

$$\boldsymbol{x}^0 := (\overline{x}) \quad where \quad \overline{x} = \frac{x}{L_p}$$

where x denotes the coordinate about the x-axis; and $N(\boldsymbol{x}^0; \boldsymbol{\theta})$ is a fully-connected NN model with the input feature, $\boldsymbol{x}$, and the model parameter set, $\boldsymbol{\theta}$.

The employed NN model of depth N is composited of an input layer, N hidden layers, and an output layer, as shown in Figure 6-3 Schematic of a fully-connected neural networks model. In the i^{th} hidden layer, n_i neuron are contained. Each hidden layer receives a vector, $\boldsymbol{x}^{i-1} \in \mathbb{R}^{n_{i-1}}$, from the previous layer after a linear map is conducted:

$$f_i(\boldsymbol{x}^{i-1}) := \boldsymbol{W}^i \boldsymbol{x}^{i-1} + \boldsymbol{b}^i$$

where f_i denotes the linear map for the i^{th} layer; $\boldsymbol{W}^i \in \mathbb{R}^{n_i \times n_{i-1}}$ and $\boldsymbol{b}^i \in \mathbb{R}^{n_i}$ are the corresponding weight matrix and bias vector, which are all included in the model parameter set, $\boldsymbol{\theta} = \{\boldsymbol{W}^i, \ \boldsymbol{b}^i\}_{i=1}^{N+1}$. The activation function, $\sigma_i(\cdot)$, is applied to each component of $f_i(\boldsymbol{x}^{i-1})$ before is sent to the next layer. Thus, the analytical formulation of the constructed NN model can be expressed in the following compositional function:

$$N(\boldsymbol{x}^0; \boldsymbol{\theta}) = (f^{N+1} \circ \sigma^N \circ f^N \circ \ldots \circ \sigma^1 \circ f^1)(\boldsymbol{x}^0) \tag{5.3}$$

in which the operator ◦ is the composition operator.

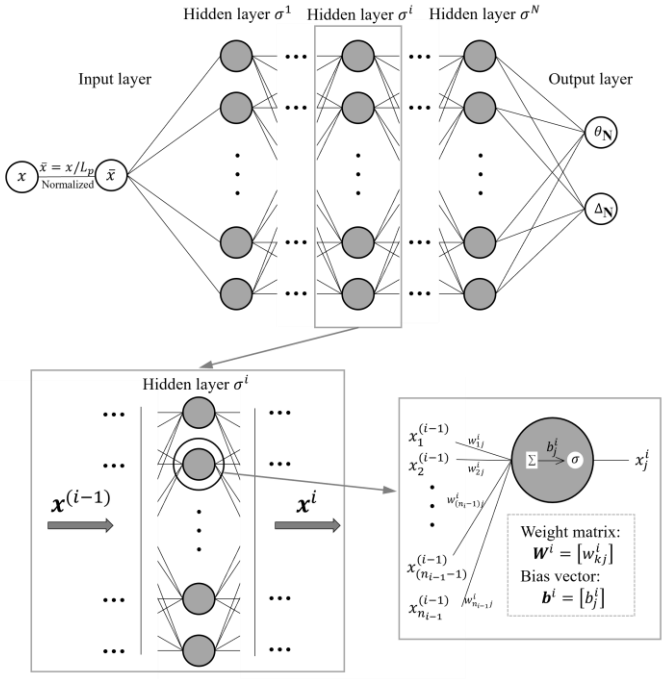


Figure 6-3 Schematic of a fully-connected neural networks model

Various activation functions are employed in past PINN study, including *Sigmoid*, *ReLU*, *Tanhshrink*, and *Tanh.* Extensive tests for different selections of the activation function are conducted, whose results are reported in Appendix A. According to their

performance, two activation functions are employed in the present study for the first/last hidden layers and the hidden layers in the middle, which can be expressed as:

$$\sigma^i(x) = \begin{cases} Tanhshrink(x) & when \quad i = 1 \quad or \quad N \\ Tanh(x) & when \quad i = 2,3,\cdots,N-1 \end{cases}$$

in which:

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$Tanhshrink(x) = Tanh(x) - x$$

6.3.2 Loss function

The key idea of the PINN is evaluating the model accuracy using the model output and their derivatives in accordance with the residuals of the governing equations and boundary conditions. Commonly, the loss function employed for calibrating the mode error is directly formulated based on the MSE criterion. The loss function in the present study can be decoupled into three parts:

$$\boldsymbol{L} = \boldsymbol{L}_G + \boldsymbol{L}_B + \boldsymbol{L}_T$$

where $\boldsymbol{L}$ is the loss function; and the subscripts $_G$, $_B$, and $_T$ the loss terms denoting the governing equation, the bottom end boundary condition, and the top end boundary condition.

The first part of the loss function, $\boldsymbol{L}_G$, is formulated in accordance with the residual terms of the governing equation:

$$\boldsymbol{L}_G = MSE_\theta + MSE_\Delta$$

where,

$$MSE_\theta = \frac{1}{n}\sum_i^n \left(\frac{1}{L_p}\frac{d\theta_N}{d\overline{x}}\bigg|_{\overline{x}=\overline{x}_i} - \frac{M}{EI}\bigg|_{\overline{x}=\overline{x}_i} \right)^2$$

$$MSE_\Delta = \frac{1}{n}\sum_i^n \left(\frac{1}{L_p}\frac{d\Delta_N}{d\overline{x}}\bigg|_{\overline{x}=\overline{x}_i} - \frac{P}{EI}\bigg|_{\overline{x}=\overline{x}_i} \right)^2$$

in which n is the sampling point number, which is taken as 50 according to the results of the sensitivity study in Appendix B; $\overline{x_i}$ denotes the sampling point that is randomly placed within [0, 1] in sequence as $0 = \overline{x}_1 < \overline{x}_2 < \cdots < \overline{x}_{n-1} < \overline{x}_n = 1$; and the bending moment, $M|_{\overline{x}=\overline{x}_i}$, and axial force, $P|_{\overline{x}=\overline{x}_i}$, at the sampling points that can be calculated according to:

$$M|_{\overline{x}=\overline{x}_i} = -F_{x,Top}\left(y|_{\overline{x}=\overline{x}_1} - y|_{\overline{x}=\overline{x}_i}\right) + F_{y,Top}\left[u|_{\overline{x}=\overline{x}_1} + L - u|_{\overline{x}=\overline{x}_i} - \overline{x}L\right] + M_{y,Top} + M_S|_{\overline{x}=\overline{x}_i}$$

$$P|_{\overline{x}=\overline{x}_i} = F_{x,Top}\cos\theta_N|_{\overline{x}=\overline{x}_i} + \left(F_{y,Top} + F_S|_{\overline{x}=\overline{x}_i}\right)\sin\theta_N|_{\overline{x}=\overline{x}_i}$$

where $u|_{\overline{x}=\overline{x}_i}$ and $y|_{\overline{x}=\overline{x}_i}$ are the pile deflection about the x- and y-axis at the sampling point, respectively; and, $M_S|_{\overline{x}=\overline{x}_i}$ and $F_S|_{\overline{x}=\overline{x}_i}$ are the internal forces on the pile caused by the lateral soil resistance at the sampling point, respectively.

According to the constructed mechanical model, these pile deflection and internal forces at the sampling point can be expressed as:

$$u|_{\overline{x}=\overline{x}_i} = -L_p\int_0^{\overline{x}_i}(1-\cos\theta_N)dt + \int_0^{\overline{x}_i}\cos\theta_N\left(\frac{d\Delta_N}{dt}\right)dt$$

$$y|_{\overline{x}=\overline{x}_i} = -L_p\int_0^{\overline{x}_i}\sin\theta_N\,dt + \int_0^{\overline{x}_i}\sin\theta_N\left(\frac{d\Delta_N}{dt}\right)dt$$

$$M_S|_{\overline{x}=\overline{x}_i} = -L_p\int_{\overline{x}_i}^1\left[Lt + u(t) - L\overline{x}_i - u|_{\overline{x}=\overline{x}_i}\right]p(t)dt$$

$$F_S|_{\overline{x}=\overline{x}_i} = L_p \int_{\overline{x}_i}^{1} p(t)dt$$

However, even when the expression of the internal force and deflection along the pile at the sampling point is formulated, the analytical solutions of the above equations are still hard to achieve since they are composed of some complex integral terms. Therefore, this chapter employed the trapezoid rules to estimate the values of the above equations:

$$u|_{\overline{x}=\overline{x}_i} \approx \sum_{j=1}^{i-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left[-L_p\left(1-cos\,\theta_N|_{\overline{x}=\overline{x}_j}\right)-L_p\left(1-cos\,\theta_N|_{\overline{x}=\overline{x}_{j+1}}\right)\right] + \sum_{j=1}^{i-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left[\left(cos\,\theta_N\frac{d\Delta_N}{dt}\right)\Big|_{\overline{x}=\overline{x}_j}+\left(cos\,\theta_N\frac{d\Delta_N}{dt}\right)\Big|_{\overline{x}=\overline{x}_{j+1}}\right]$$

$$y|_{\overline{x}=\overline{x}_i} \approx \sum_{j=1}^{i-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left(-L_p sin\,\theta_N|_{\overline{x}=\overline{x}_j}-L_p sin\,\theta_N|_{\overline{x}=\overline{x}_{j+1}}\right) + \sum_{j=1}^{i-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left[\left(sin\,\theta_N\frac{d\Delta_N}{dt}\right)\Big|_{\overline{x}=\overline{x}_j}+\left(sin\,\theta_N\frac{d\Delta_N}{dt}\right)\Big|_{\overline{x}=\overline{x}_{j+1}}\right]$$

$$M|_{\overline{x}=\overline{x}_i} \approx \sum_{j=i}^{n-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left[\left(L_p\overline{x}_j+u|_{\overline{x}=\overline{x}_j}-L_p\overline{x}_i-u|_{\overline{x}=\overline{x}_i}\right)p|_{\overline{x}=\overline{x}_j} + \left(L_p\overline{x}_{j+1}+u|_{\overline{x}=\overline{x}_{j+1}}-L_p\overline{x}_i-u|_{\overline{x}=\overline{x}_i}\right)p|_{\overline{x}=\overline{x}_{j+1}}\right]$$

$$F_S|_{\overline{x}=\overline{x}_i} \approx \sum_{j=i}^{n-1}\left(\frac{\overline{x}_{j+1}-\overline{x}_j}{2}\right)\left(L_p p|_{\overline{x}=\overline{x}_j}+L_p p|_{\overline{x}=\overline{x}_{j+1}}\right)$$

For the second part in the loss function, the loss term denoting the boundary condition at the pile's bottom end can be calculated by:

$$\boldsymbol{L}_B = MSE_{B,\theta} + MSE_{B,\Delta}$$

where,

$$MSE_{B,\theta} = \left(\theta_{\mathrm{N}}|_{\overline{x}=\overline{x}_1} - 0\right)^2$$

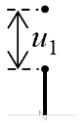
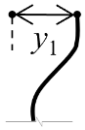
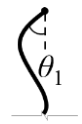
$$MSE_{B,\Delta} = \left(\Delta_{\mathrm{N}}|_{\overline{x}=\overline{x}_1} - 0\right)^2$$

The loss term, $\boldsymbol{L}_T$, is determined according to the boundary conditions at the pile's top end, which can be divided into three parts as:

$$\boldsymbol{L}_T = MSE_{T,u} + MSE_{T,y} + MSE_{T,\theta}$$

According to different restraint conditions, the components of $\boldsymbol{L}_T$ are summarized in Table 6-2.

Table 6-2 Loss terms for boundary conditions

Boundary*	Status	Components of $\boldsymbol{L}_T$
	Free	$MSE_{T,u} = \left[\frac{EA}{L_p}\left(cos\,\theta\,\frac{d\Delta_N}{dx}\right)\Big\|_{\overline{x}=\overline{x}_n} - \frac{EI}{L_p^2}\left(sin\,\theta\,\frac{d^2\theta_N}{dx^2}\right)\Big\|_{\overline{x}=\overline{x}_n} - F_{x,Top}\right]^2$
	Fixed	$MSE_{T,u} = \left(u|_{\overline{x}=\overline{x}_n} - 0\right)^2$
	Free	$MSE_{T,y} = \left[\frac{EA}{L_p}\left(sin\,\theta\,\frac{d\Delta_N}{dx}\right)\Big\|_{\overline{x}=\overline{x}_n} + \frac{EI}{L_p^2}\left(cos\,\theta\,\frac{d^2\theta_N}{dx^2}\right)\Big\|_{\overline{x}=\overline{x}_n} - F_{y,Top}\right]^2$
	Fixed	$MSE_{T,y} = \left(y|_{\overline{x}=\overline{x}_n} - 0\right)^2$
	Free	$MSE_{T,\theta} = \left(\frac{EI}{L}\frac{d\theta_N}{dx}\Big\|_{\overline{x}=\overline{x}_n} - M_{Top}\right)^2$
	Fixed	$MSE_{T,\theta} = \left(\theta_N|_{\overline{x}=\overline{x}_n} - 0\right)^2$

(*Note: The subscript 1 denotes the top end of the pile)

6.3.3 Normalization factors for loss function

The loss function is constructed by directly measuring the residuals of the governing equation in the form of MSE, a commonly used approach in PINN studies. However, the lack of normalization in this straightforward construction method may lead to issues in its application in certain engineering applications, including:

(a) The unnormalized loss function poses challenges to finding an appropriate error tolerance for assessing the accuracy of trained NN models, given the potential wide variance in solution magnitudes. Specifically, a small solution magnitude can yield significant relative errors of model predictions, even when the loss function meets tolerance levels. Conversely, large solution magnitudes might make standard tolerance levels unattainably strict.

(b) The magnitudes of loss terms might be significantly different, resulting in the failure of minimizing the loss function during the model training.

The two problems discussed above are further demonstrated through a group of examples in Case Study 1. To ensure accurate results, appropriate normalization is necessary when constructing the loss function. This section presents a series of factors inspired by the FEM to address these issues.

In the FEM for the beam-column analysis [300], the convergency criteria is not set as the magnitude of the residual terms but as the ratio values as:

$$\begin{cases} \dfrac{\|\boldsymbol{R}\|_{L2}}{\|\boldsymbol{F}\|_{L2}} \leq TOL \\ \dfrac{\|\Delta\boldsymbol{U}\|_{L2}}{\|\boldsymbol{U}\|_{L2}} \leq TOL \end{cases}$$

where $\|\cdot\|_{L2}$ is the L2 norm of the vector; $\boldsymbol{R}$ and $\Delta\boldsymbol{U}$ are the residual terms for the nodal force and displacement vectors, respective; $\boldsymbol{F}$ and $\boldsymbol{U}$ denote the applied force and current nodal displacement vectors, respective; and *TOL* is the error tolerance.

In the present study, the loss function in the PINN, inspired by the convergency criteria in the FEM, is not directly formulated based on the MSE, but should rather be set as the ratio of residual terms to the corresponding current value. Thus, a series of factors are introduced in the loss function for the normalization:

$$\boldsymbol{L} = \boldsymbol{L}_G + \boldsymbol{L}_B + \boldsymbol{L}_T$$

$$\boldsymbol{L}_G = N_\theta MSE_\theta + N_\Delta MSE_\Delta$$

$$\boldsymbol{L}_B = N_{B,\theta} MSE_{B,\theta} + N_{B,\Delta} MSE_{B,\Delta}$$

$$\boldsymbol{L}_T = N_{T,u} MSE_{T,u} + N_{T,u} MSE_{T,y} + N_{T,u} MSE_{T,\theta}$$

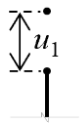
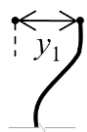
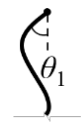
where N_i denotes the normalization factors for different loss terms.

For the first parts in the loss function, the corresponding normalization factor, N_θ and N_Δ, is set as:

$$N_\theta = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{M}{EI}\Big|_{\overline{x}=\overline{x}_i}\right)^2$$

$$N_\Delta = \frac{1}{n}\sum_{i}^{n}\left(\frac{P}{EI}\Big|_{\overline{x}=\overline{x}_i}\right)^2$$

Table 6-3 Normalization factors for boundary conditions

Boundary*	***Status***	***Normalization factor***
	Free	$N_{T,u} = \left({F_{x,Top}}^2 + {F_{y,Top}}^2 + {M_{Top}}^2\right)$
	Fixed	$N_{T,u} = \frac{1}{n}\sum_{i=1}^{n}\left(u|_{\overline{x}=\overline{x}_i}\right)^2$
	Free	$N_{T,y} = \left({F_{x,Top}}^2 + {F_{y,Top}}^2 + {M_{Top}}^2\right)$
	Fixed	$N_{T,y} = \frac{1}{n}\sum_{i=1}^{n}\left(y|_{\overline{x}=\overline{x}_i}\right)^2$
	Free	$N_{T,\theta} = \left({F_{x,Top}}^2 + {F_{y,Top}}^2 + {M_{Top}}^2\right)$
	Fixed	$N_{T,\theta} = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_N|_{\overline{x}=\overline{x}_i}\right)^2$

(*Note: The subscript $_1$ denotes the top end of the pile)

For the second part in the loss function, the corresponding normalization factor for the loss term denoting the boundary conditions at the pile's bottom end can be expressed as:

$$N_{B,\theta} = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_N|_{\overline{x}=\overline{x}_i}\right)^2$$

$$N_{B,\Delta} = \frac{1}{n}\sum_{i=1}^{n}\left(\Delta_N|_{\overline{x}=\overline{x}_i}\right)^2$$

And for different restraints on the pile top end, the corresponding normalization factors are summarized in

Table 6-3.

6.4 Model Training

In the previous section, an NN model is constructed for predicting the solution to governing equations, with the corresponding loss function being formulated to calibrate the prediction error. This section provides a detailed illustration of the standard PINN model training process, where the loss is minimized for searching for the optimal model parameter set. A regression method via the NN model is introduced to estimate the soil resistance during training to overcome convergence failure of the loss function when encountering some non-differentiable *p-y* relations. Additionally, an algorithm that automatically assigns weights to loss terms is utilized to increase training efficiency.

6.4.1 Standard training process

The model training is the procedure for searching for the optimal model parameter set, $\boldsymbol{\theta}$*, of the minimum loss value. This can be mathematically represented as:

$$\boldsymbol{\theta} *= \underset{\boldsymbol{\theta}\in R^D}{argmin}\{\boldsymbol{L}(\boldsymbol{\theta})\} \tag{5.4}$$

where D denotes the number of parameters contained in all the weight matrix and bias vectors.

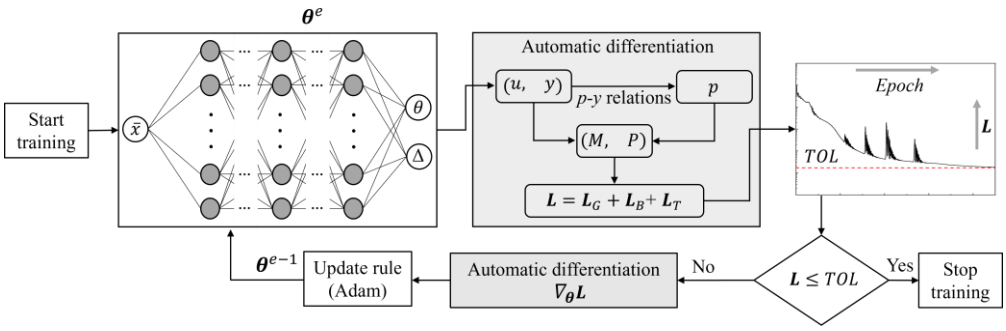


Figure 6-4 Standard PINN model training process

To achieve the global minimizer of Eq. (5.4), various type of optimization algorithm has been employed in the ML technique, such the stochastic gradient decedent. Among different type of optimization strategies, the adaptive moment estimation [301], also known as Adam, is applied in the present study given its satisfactory robustness and efficiency. The update rule of Adam optimizer is provided in Appendix C. The standard PINN model training process based on the Adam, as shown in Figure 6-4, can be described as the following steps:

(1) Place sampling points along the pile, $\boldsymbol{x}^0 = \overline{x}$.

(2) Generate the model output from the sampling point, $\boldsymbol{N}(\boldsymbol{x}^0; \boldsymbol{\theta})$.

(3) Calculate the pile displacement at the sampling point, $u(\boldsymbol{x}^0)$ and $y(\boldsymbol{x}^0)$,.

(4) Obtain the lateral soil resistance at the sampling point, $p(\boldsymbol{x}^0)$, according to the pile displacement and the selected p-y relations.

(5) Calculate the bending moment and axial force at the sampling point, $M(\boldsymbol{x}^0)$

and $P(\boldsymbol{x}^0)$.

(6) Calculate the total loss function, $\boldsymbol{L}(\boldsymbol{\theta})$.

(7) Update the model parameter set $\boldsymbol{\theta}$ using the loss function derivative with aspect of the model parameter set, $\nabla_{\boldsymbol{\theta}}\boldsymbol{L}$, according to the update rule of Adam. Then return to Step 1 until the following error tolerance, TOL = 1 x 10^{-5}, is achieved:

$$\boldsymbol{L}_G + \boldsymbol{L}_B + \boldsymbol{L}_T \leq TOL$$

6.4.2 Regression-based soil resistance estimation with neural networks

As shown in Figure 6-4, the automatic differentiation technique plays a crucial role in the model training, which is implemented to construct the loss function and obtain its derivative. However, the automatic differentiation technique is only capable of calculating the derivative of the function with the analytical form. For some scenarios in practice, such as the API clay [155], the lateral soil resistance is not obtained from an analytical equation but by linear interpolation according to the recorded data. However, such numerical values will be ignored by the automatic differentiation technique when creating the computational graph. Thus, the automatic differentiation technique cannot appropriately obtain the derivative about the lateral soil resistance, sometimes leading to severe convergence failure of the loss function. Even though the indicator function, $I(\cdot)$, could be employed to transform the above *p*-*y* relations into analytical forms [296] as $p(y) = \sum[(\frac{p_i - p_{i-1}}{y_i - y_{i-1}}) \cdot (y_{i-1} - y) + p_{i-1}] \cdot I(y_{i-1} < y \leq y_i)$, the corresponding expression could sometimes be complicated, resulting in great implementation difficulties. Additionally, severe numerical instability will also occur when some other non-differentiable *p*-*y* relations are encountered, whose derivatives trend to infinity at certain locations. The above issues will be illustrated through a group of examples in Case Study 3.

To address this problem, a regression for the complex *p*-*y* relations is introduced to estimate the lateral soil resistance in the PINN model training process. Among

various types of regression methods, such as polynomial regression and support vector regression, the NN model is selected here for its powerful capability in function approximation and the availability of its derivatives of arbitrary order. The implemented fully-connected NN model, as a general regression approach to the selected *p*-*y* relations, can be written as:

$$N_p(\boldsymbol{x}^0;\boldsymbol{\theta}_p) := p_{\mathrm{N}} \approx p(\boldsymbol{x}^0)$$

where N_p denote the implemented NN model for regression; $\boldsymbol{x}^0$ is the input feature contained two elements, $\boldsymbol{x}^0 := (\overline{x},\ \ y)$; $\overline{x}$ and y represents the normalized x-coordinate and the corresponding lateral deflection; $\boldsymbol{\theta}_p$ denotes the model parameter set; $p_{\boldsymbol{N}}$ is the output feature of $\boldsymbol{N}_p$; p is the lateral soil resistance at the normalized x-coordinate, $\overline{x}$, with the corresponding lateral deflection, y, obtained from the selected *p*-*y* relations.

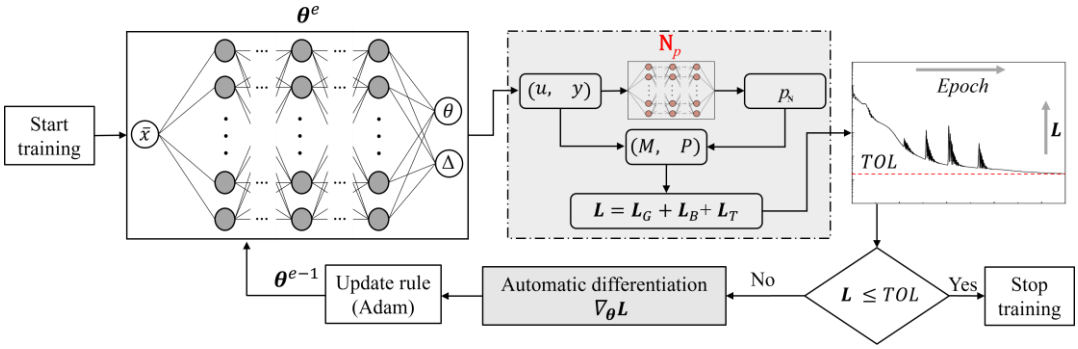


Figure 6-5 Modified PINN model training process implemented with regression-based soil resistance estimation

The loss function for the above NN model can be expressed as:

$$\boldsymbol{L}_p(\boldsymbol{\theta}_p) = \frac{1}{n_p}\sum_{i=1}^{n_p}\left(p_N|_{\boldsymbol{x}^0=\boldsymbol{x}_i^0} - p|_{\boldsymbol{x}^0=\boldsymbol{x}_i^0}\right)^2$$

where n_p is the number of sampling points, $\boldsymbol{x}_i^0$, which is taken as 1 x 10^5 in the present study and can be defined by users according to needs.

Consequently, the lateral soil resistance obtained during the PINN model training process are estimated by the constructed NN model, $\boldsymbol{N}_p$, as shown in Figure 6-5. Due to the continuous and differentiable nature of the NN model, the derivative of the lateral soil resistance during the model training can be approximated by:

$$\frac{\partial^n p_N}{\partial x^n} \approx \frac{\partial^n p}{\partial x^n}$$

6.4.3 Adaptive loss weight algorithm

As discussed by Raissi et al. [80], the training process for the multi-objective PINN model is sometimes challenging. The inappropriate weights assigned for different loss terms will cause gradient pathologies that lower the accuracy of the model training [87]. To address this problem, applying the PINN in practice sometimes suffers from a tedious trial-and-error procedure to manually find an appropriate weight vector for the total loss function.

For this reason, the present study adopted an adaptive loss weight algorithm, proposed by Wang et al. [87], for automatically updating the loss weight vector in the model training. In this adaptive loss weight algorithm, the total loss function is rewritten as:

$$\boldsymbol{L} = \boldsymbol{L}_G + \omega_B \boldsymbol{L}_B + \omega_T \boldsymbol{L}_T$$

where ω is the loss weight for the loss term. All the loss weights are set to be one at the beginning of the model training. The update step for ω in the training procedure can be expressed as:

$$\omega_i^e = \beta\omega_i^{e-1} + (1-\beta)\tilde{\omega}_i^{e-1} \quad where \quad i = B, T$$

where,

$$\tilde{\omega}_i^e = \frac{1}{\omega_i^{e-1}} \frac{max|\nabla_{\boldsymbol{\theta}} \boldsymbol{L}_G|}{mean|\nabla_{\boldsymbol{\theta}} \boldsymbol{L}_i|} \quad where \quad i = B, T$$

in which, the superscript ^{e} denotes the e^{th} epoch in the model training; β is a hyper-parameter and taken as 0.9 in the present study; and $\nabla_{\boldsymbol{\theta}} \boldsymbol{L}$ represents the gradient vector of the loss term with the aspect of the model parameter set $\boldsymbol{\theta}$; *max* and *mean* are the operation to obtain the max value and the mean value of the vector, respectively; and $|\cdot|$ denotes the operation for transforming all the terms contained in the vector into their absolute values.

In the present study, the loss weights are updated using the above-mentioned scheme every fifth epoch instead of every training time. It should be noted that users can customize the frequency of the adaptive loss weight algorithm according to their specific needs.

6.5 Validation Example

In this section, a series of piles under various geological conditions are examined using different approaches, including the proposed method, existing solutions, and past studies, to exam the accuracy of the present study for slender piles undergoing larger deflection.

6.5.1 Example 1 - Single pile embedded in soil of constant modulus

To demonstrate the accuracy of the proposed method, a single steel H-pile of L_p = 25 m in length embedded in homogenous soil with constant stiffness is studied, as shown in Figure 6-6. The cross-section of the pile is W14X26, whose moment of inertia and cross-section area are equal to $I = 2.8179 \times 10^{-4}$ m^4 and $A = 2.275 \times 10^{-2}$ m^2, respectively. The steel elastic modulus equals $E = 2.0 \times 10^8$ kPa. The SPI is assumed to be linear and can be described with the soil modulus, k = 200 kN/m^2. As illustrated

in Figure 6-7, the pile's bottom end is assumed to be fixed and three boundary conditions are assigned at the pile top, including fixed, pinned, and fixed with sway. The load applied at the pile top is composited of the axial load, P = 800 kN, shear force, V = 150 kN, and bending moment, M = 300 kN·m. Two different load factors (LFs), LF = 1.0 and LF = 0.2, are assigned for the applied load to proportionally control the load magnitude. The LFEM using 30 elements is selected as the benchmark solutions to investigate the steel H-pile due to their satisfied performance for piles embedded in the homogenous soil. Different analysis methods are employed to predict the structural behaviors of piles, which are: (1) the close-form solutions of the simplified governing equations derived by Matlock and Reese [297]; and (2) the proposed large deflection analysis method via the PINN. The lateral pile deflections along the pile calculated from different methods are plotted in Figure 6-7.

According to Figure 6-7, it can be observed that the simplified closed-form solution fails to accurately predict the structural behaviors of the slender pile, particularly when the deflection along the pile is large. When the load factor grows to LF = 1.0, the close-form solution considerably underestimates the pile deflection due to its ignorance of the geometric nonlinearity. As illustrated in Figure 6-7B, this underestimation can lead to relative errors that sometimes escalate up to 20%. Conversely, the results obtained from the proposed method exhibit a high degree of agreement with the benchmark solutions, thus validating the accuracy and reliability of the derived governing equations for the pile undergoing large deflection, as well as the corresponding PINN solutions.

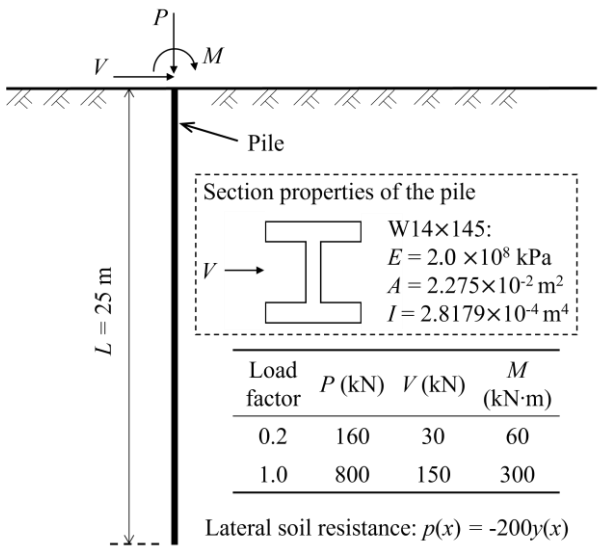
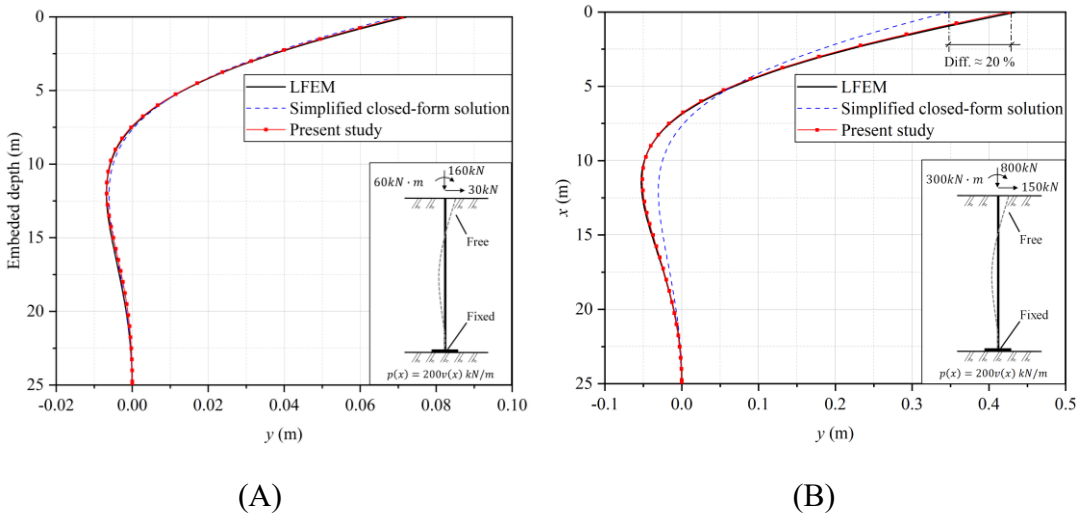
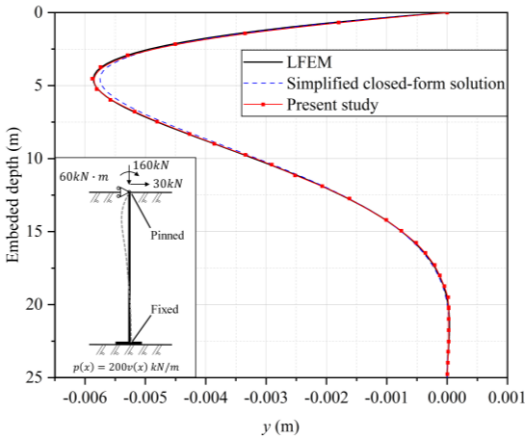


Figure 6-6 Single pile embedded in soil with constant modulus

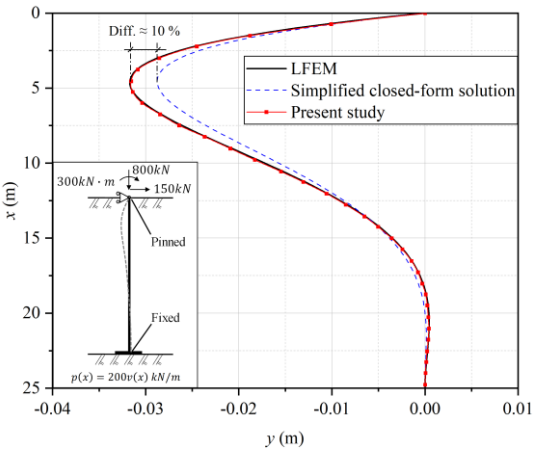


(A)

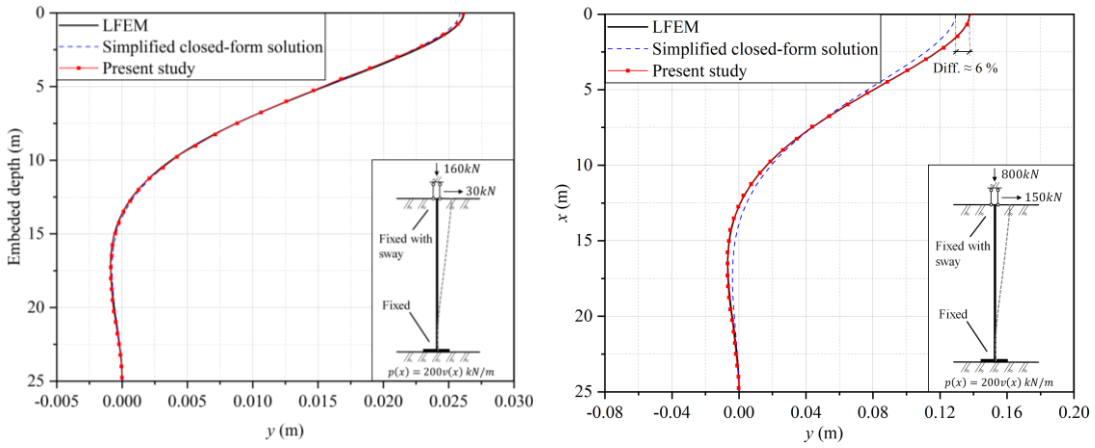
(B)



(C)



(D)



(E) (F)

Figure 6-7 Deflection curves of piles with different boundary conditions: (A) Free (LF = 0.2); (B) Free (LF = 1.0); (C) Pinned (LF = 0.2); (D) Pinned (LF = 1.0); (E) Fixed with sway (LF = 0.2); and (F) Fixed with sway (LF = 1.0)

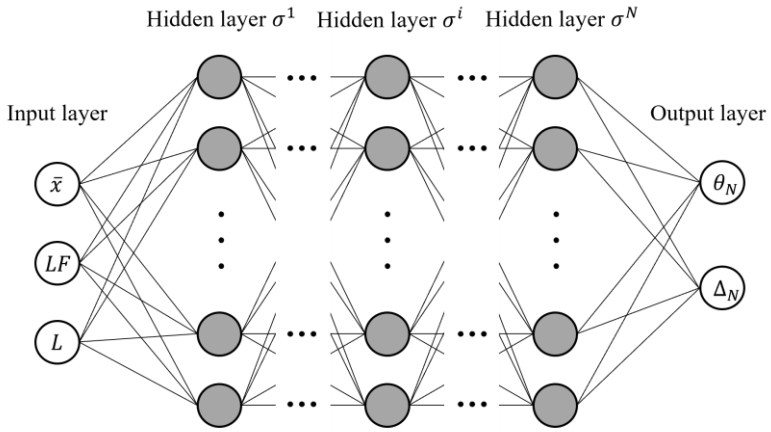
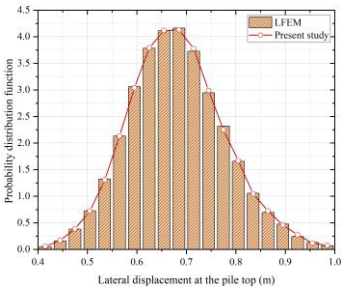


Figure 6-8 Schematics of surrogate model incorporating load factor (LF) and pile length in input

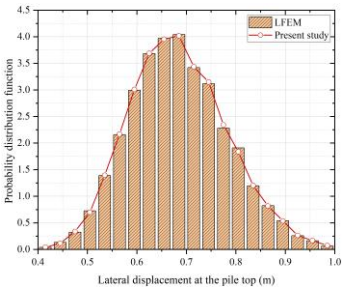
To demonstrate the computational efficiency of the proposed method, this example is further extended to an evaluation of piles with different lengths considering load uncertainty, which is commonly encountered during reliability-based design optimization [302]. The pile lengths are varied from 20 m to 30 m, while other pile and

soil parameters maintain consistency with the previous part of this example. According to Ellingwood [303, 304], the LF can be taken as following the normal distribution. Thus, a normal distribution with a mean of 1.0 and a standard deviation of 10% is employed to describe the LF for load uncertainty. The Monte-Carlo Simulation [305] is utilized to evaluate load uncertainty, where 10000 LFs are randomly generated according to the prescribed probability distribution and assigned to piles of different lengths. The LFEM using 30 elements is selected as the benchmark solution when solving the deterministic models generated during the Monte-Carlo Simulation. The proposed PINN method is extended to construct the surrogate model and employed in the Monte-Carlo Simulation for comparison. The constructed PINN surrogate model incorporates the LF and pile length, L, into the model input (as shown in Figure 6-8), and the loss function and model training process remain consistent with Sections 3 & 4.

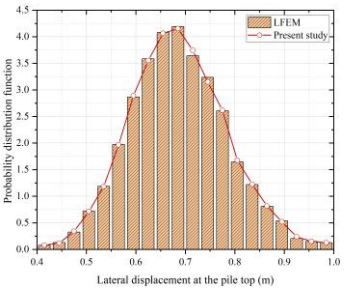
The uncertainty evaluation results are presented via the probability distribution function of the lateral deflection at the pile top end. Figure 6-9 shows that the results from the proposed method and LFEM are almost identical to each other, reinforcing the accuracy of the proposed PINN method. However, the computational costs of different methods are significantly different, even though both evaluation processes are conducted on the same computer device with i9-12900K and 32 GB RAM without the GPU device. The LFEM requires approximately 29.3 hours to solve $11 \times 10000 = 110000$ deterministic models throughout the Monte-Carlo Simulation. In contrast, the proposed PINN surrogate model takes only 2.1 hours for model training, plus an additional 2 minutes to generate results from the trained model, indicating the superiority of the proposed PINN method in computational efficiency when encountering some computationally intensive tasks.



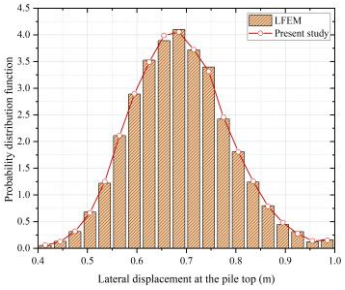
(A) L = 20 m



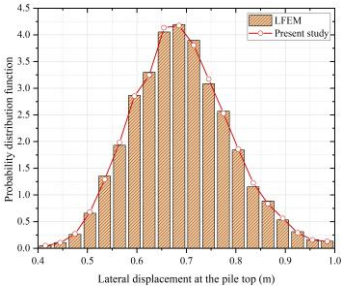
(B) L = 21 m



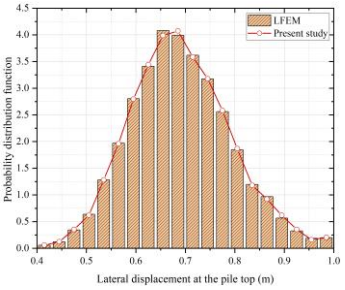
(C) L = 22 m



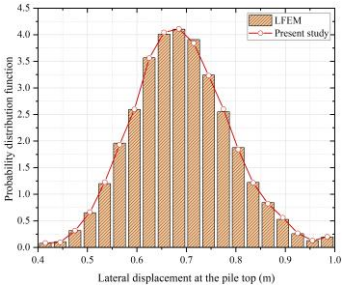
(D) L = 23 m



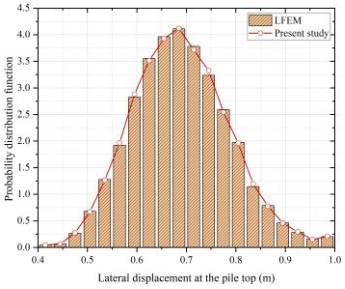
(E) L = 24 m



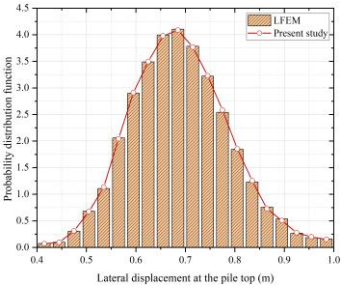
(F) L = 25 m



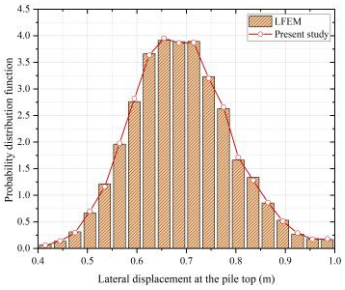
(G) L = 26 m



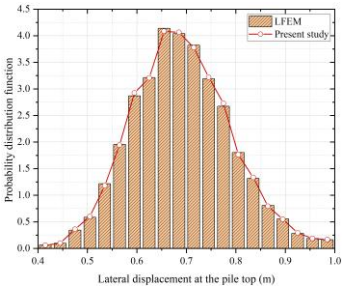
(H) L = 27 m



(I) L = 28 m



(J) L = 29 m



(K) L = 30 m

Figure 6-9 Probability distribution function of lateral pile top deflection: (A) L = 20 m; (B) L = 21 m; (C) L = 22 m; (D) L = 23 m; (E) L = 24 m; (F) L = 25 m; (g) L = 26 m; (H) L = 27 m; (I) L = 28 m; (J) L = 29 m; (K) L = 30 m;

6.5.2 Example 2 - Single pile embedded in soft clay

The SPI relations in the design practice sometimes possess significant nonlinearity.

In this example, a concrete pile of L = 30 m in length embedded in the soft clay is studied. The parameters of the soft clay include the cohesion efficient c = 16 kPa, fictional angle φ = 2 °, and unit weight γ = 1900 kg/m^3. According to API [155], the *p-y* relations for describing the corresponding nonlinear SPI can be generated from Table 6-4, where p_u and y_c are the coefficients that can be calculated by:

$$p_u = 3s_u + \gamma X + J\frac{s_u X}{D}$$

$$y_c = 2.5\varepsilon_c D$$

$$s_u = c + \gamma X \tan\varphi$$

where p_u is the ultimate soil resistance; y_c denotes pile displacement resulting from the surrounding soil reaching 50% of its ultimate resistance; s_u is the shear strength of the clay; X represents the embedded depth, which is equal to ($L - x$) in the present study; D is the pile diameter that equals 1 m here; J denotes dimensionless empirical constant that equals 0.5 in the present study; and ε_c is the strain coefficient and taken as 0.005 in the present study.

Table 6-4. API *p-y* relations for piles embedded in clay [155]

p/p_u	y/y_c
0.00	0.0
0.50	1.0
0.72	3.0
1.00	8.0
1.00	∞

The generated *p*-*y* relations from Table 6-4 are plotted in Figure 6-10. The moment of inertia and area of the pile cross-section are $I = 4.909 \times 10^{-1}$ m^4 and $A = 7.854 \times 10^{-1}$ m^2, respectively. The pile material elastic modulus is equal to $E = 34.5$ GPa. The pile's top end is free and subjected to an axial load $P = 4500$ kN, a shear force $V = 1000$ kN, and a bending moment $M = 2500$ kN·m. The LFEM using 50 elements is selected as the benchmark solution and compared with the proposed method to analyze the lateral deflection of the pile with different LFs. Additionally, the regression for the *p*-*y* relations is employed to estimate the lateral soil resistance during the PINN training process.

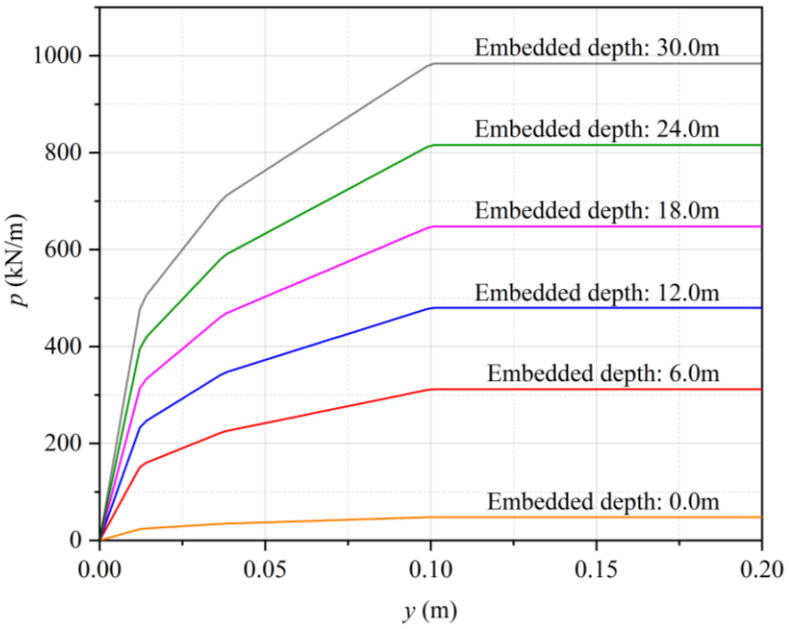


Figure 6-10 *p*-*y* relations at different embedded depths

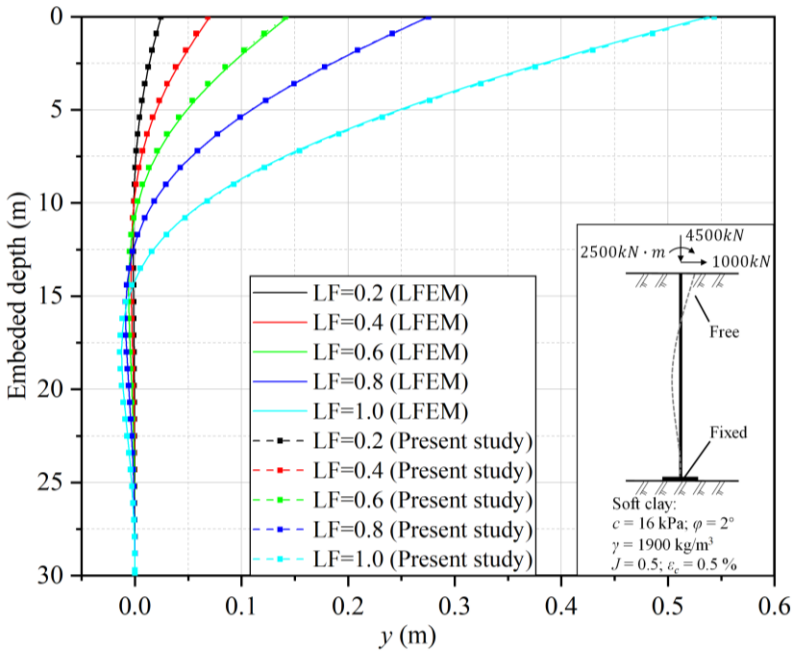


Figure 6-11 Deflection curves of piles with different load factors

The comparison results between the benchmark solutions and the proposed methods are illustrated though the lateral deflection along the pile, as shown in Figure 6-11. It can be found that the proposed large deflection analysis method can generate almost identical solutions with the LFEM under different LFs, proving the accuracy and reliability of the proposed large deflection analysis of slender piles accounting for the nonlinear SPI.

6.5.3 Example 3 - Single pile embedded in multi-layer soil

A case study reported by Liu et al. [179] is revisited in this example. The studied slender pile of 50 m in length is embedded in triple-layer of medias consisting of rock, sand, and marine clay. The corresponding *p*-*y* relations of different ground medias are documented by Jeong et al. [306], Ismael [307], and Kawamata et al. [308], respectively, which are all plotted in Figure 6-12. The cross-section of the studied pile is an I-section size of UPB356 × 368 × 174 with moment of inertia and cross-section area equal to $I = 4.909 \times 10^{-1}$ m^4 and $A = 2.21 \times 10^{-1}$ m^2, respectively. The material of the pile has an elastic modulus of E = 205 GPa. The pile is subjected to an axial load P = 15000

kN·m, and a horizontal load V = 500 kN. The lengths of the embedded pile segments in rock, sand, and marine clay are 7 m, 18 m, and 25 m, respectively, as depicted in Figure 6-13. The proposed method is employed to obtain the curve of the lateral deflection at the pile top versus the LF (as shown in Figure 6-13), which is compared with the benchmark solution generated from the LFEM using 100 elements and the reported results from Liu et al. [179]. The regression for the *p-y* relations is employed to estimate the lateral soil resistance during the PINN model training.

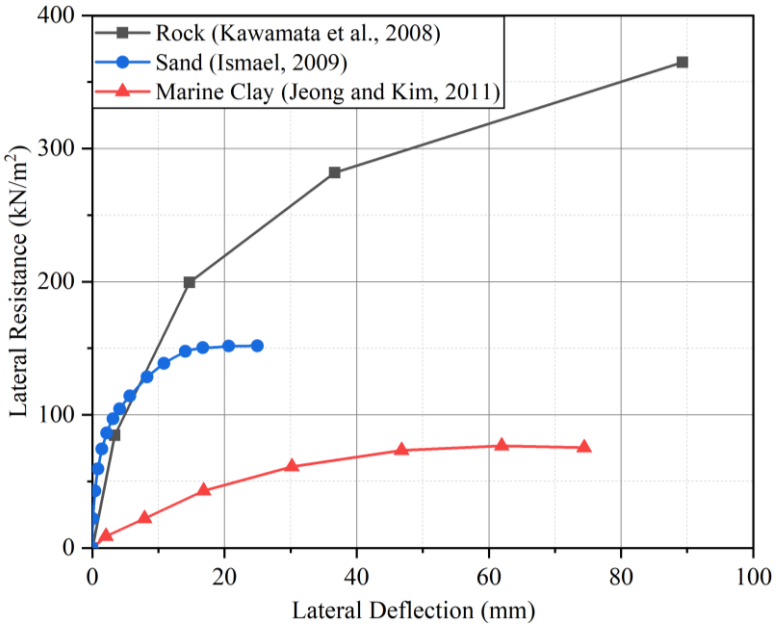


Figure 6-12 The *p-y* relations of different soil layers

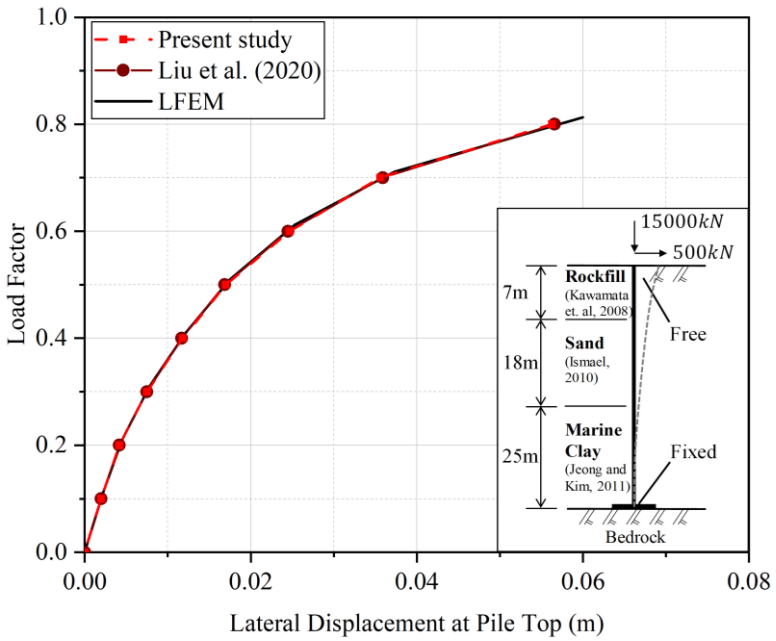


Figure 6-13 Lateral deflection at the top end of pile

Figure 6-13 demonstrates that the results from the proposed method are almost identical with the reported results from Liu et al. [179] and the benchmark solution generated from the LFEM. Consequently, the proposed method has been demonstrated to accurately predict the structural behavior of slender piles under complex geological conditions.

6.5.4 Example 4 – Single pile embedded in multi-layer sand

In this example, the proposed method is further validated via the comparison with the field test results reported by Pando et al. [309]. The dimensions of the investigated pile and the corresponding ground conditions are illustrated in Figure 6-14. The width of the rectangular pile cross-section is 610 mm. The Young's modulus of the pile material equals E = 30.82 GPa. The effective unit weights of the sands from top to bottom are γ = 19.8, 10, 10, and 11 kN/m^3, respectively. The internal friction angles of the sands from top to bottom are φ = 31°, 27°, 33°, and 35°, respectively. The *p*-*y* curves (as shown in Figure 6-14) are generated according to Li et al. [183]. Different lateral loads of V = 51.2 kN & 97.3 kN and bending moment, $M = 0.69m \times V$, are applied at

the pile top end during the field test.

The lateral pile deflections obtained from the proposed PINN method are compared with the measured field test results from Pando et al. [309] and the numerical analysis results reported by Li et al. [183]. Figure 6-15 shows that the results from the PINN method match well with the results from both field test and past literature, further indicating the accuracy of the proposed method.

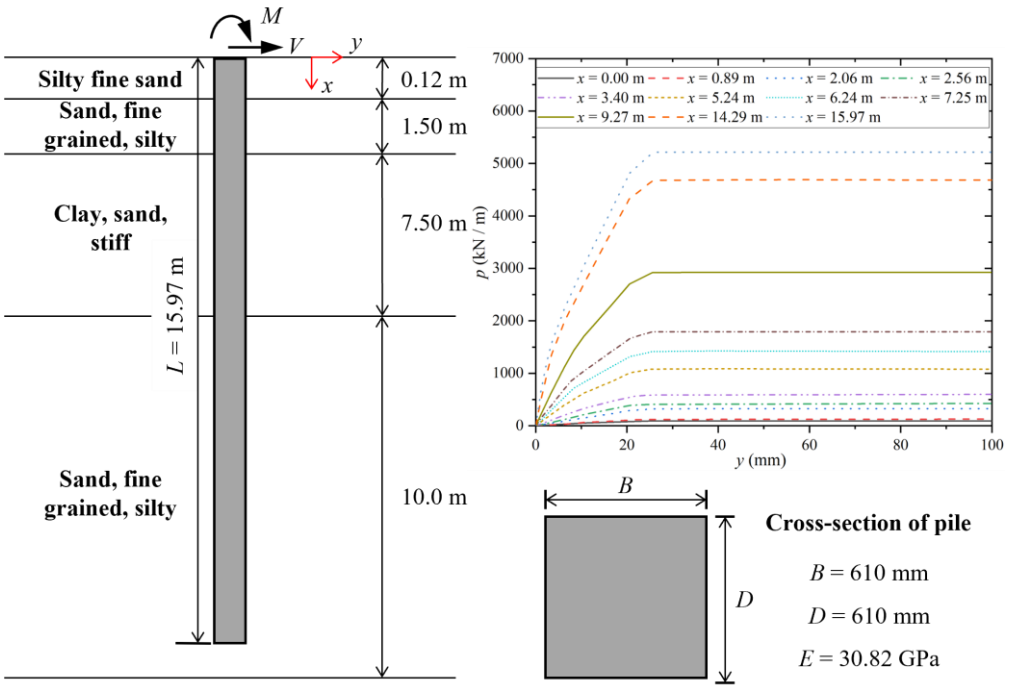


Figure 6-14 Schematic of laterally-loaded pile in multi-layer sand

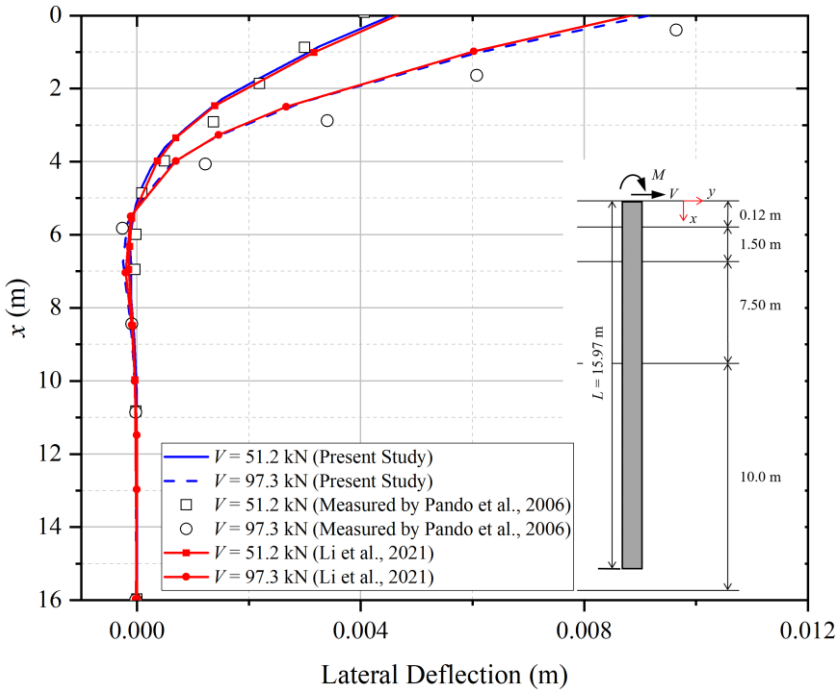


Figure 6-15 Lateral pile deflection under different lateral loads

6.6 Case Study

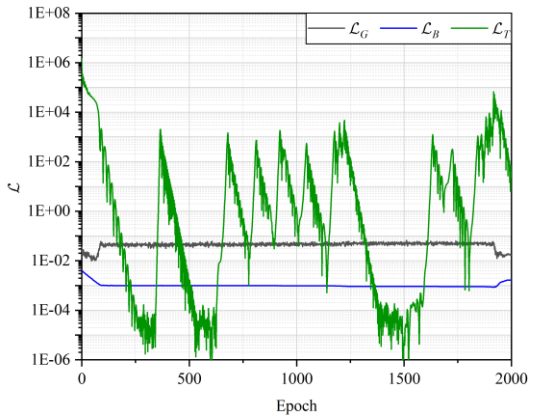
The accuracy of the derived governing equation and the corresponding PINN solution have been proven in the previous validation examples. In the following section, three groups of case studies are conducted to further illustrate the effectiveness of the numerical strategies employed in the present study, including the loss function normalization, the adaptive loss weight algorithm, and the regression of *p*-*y* relations.

6.6.1 Case Study 1 - Effect of loss normalization

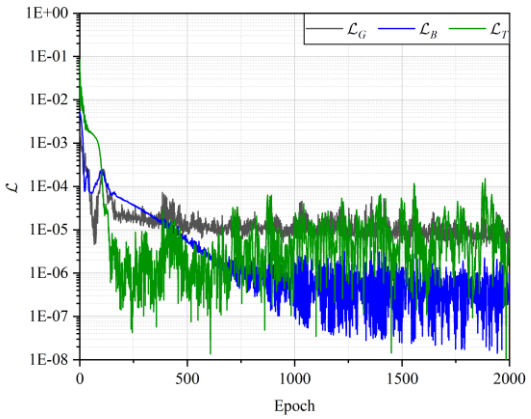
In this case study, a concrete pile with the circular cross-section is analyzed to demonstrate the necessity of appropriate loss normalization. The pile has a length of L_p = 30 m and a diameter of D = 1 m. The elastic modulus of the concrete material is specified as E = 32.5 GPa. The linear *p*-*y* relation is assumed for the pile, and the soil modulus is set to k = 200 kN/m^2. The load applied at the top of the pile includes an axial compression of 5000 kN, a shear force of 1500 kN, and a bending moment of 1000 kN·m. Two load factors, LF = 0.5 and 1.0, are considered in the analysis. To demonstrate the necessity of loss normalization, the proposed PINN is implemented

using two different loss functions: (1) loss function without normalization, where all normalization factors, N_i, are set as one; and (2) loss function with normalization, where all normalization factors, N_i, are calculated according to Section 3.3. The effect of loss normalization is demonstrated through the evolution of loss terms during model training.

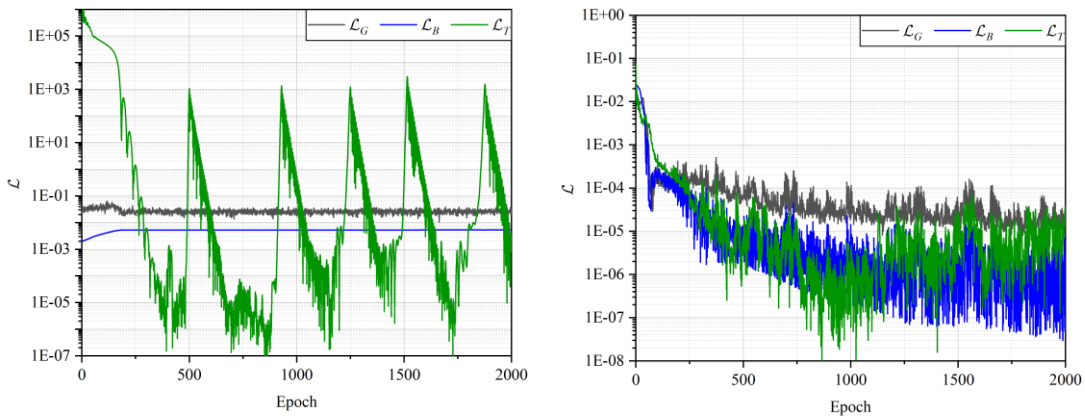
The loss term evolutions during training with and without loss normalization are depicted in Figure 6-16. It is evident that when loss normalization is ignored, the magnitudes of the different loss terms are highly disparate, leading to a significant difference between the magnitude of $\boldsymbol{L}_T$ and other two loss terms, $\boldsymbol{L}_G$ and $\boldsymbol{L}_B$. As a result, the optimizer, Adam, focuses only on minimizing $\boldsymbol{L}_T$, while disregarding the minimization of $\boldsymbol{L}_G$ and $\boldsymbol{L}_B$, leading to the magnitudes of these two loss terms remaining almost invariant during the model training. Additionally, the magnitudes of $\boldsymbol{L}_T$ exhibit significant fluctuations during the model training process. Consequently, the loss without normalization fails to be minimized. On the other hand, the implementation of loss normalization successfully narrows the magnitudes of the different loss terms within a close range, thereby enabling successful optimization of the loss function.



(A)

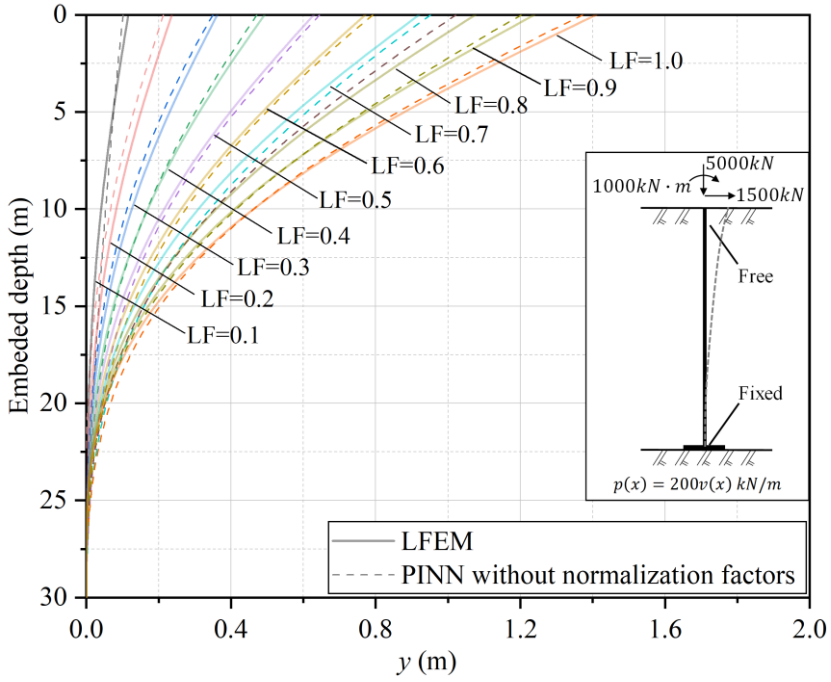


(B)

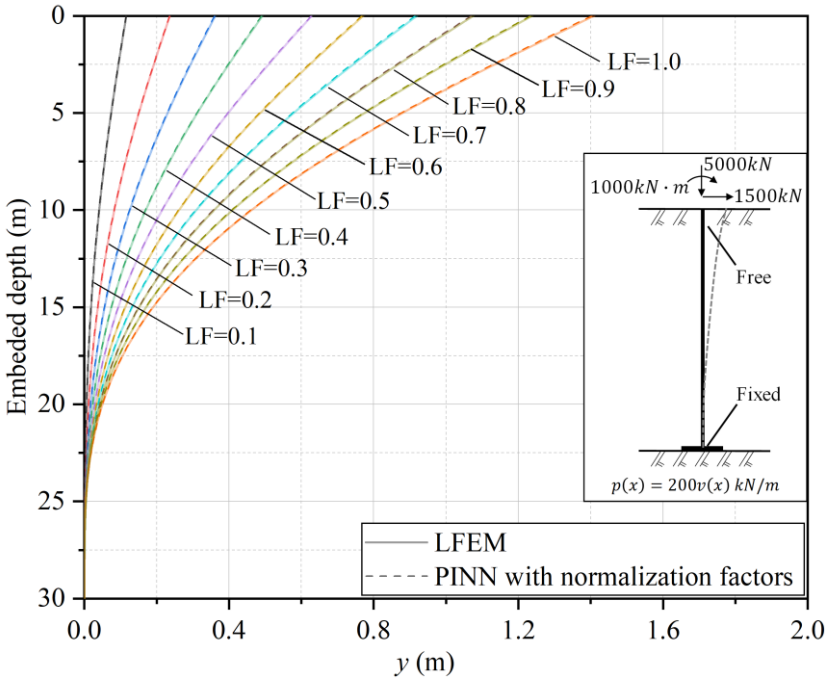


(C) (D)

Figure 6-16 Evolution of loss terms with/ without loss normalization: (A) LF = 0.5/ without loss normalization; (B) LF = 0.5/ with loss normalization; (C) LF = 1.0/ without loss normalization; and (D) LF = 1.0/ with loss normalization.



(A)



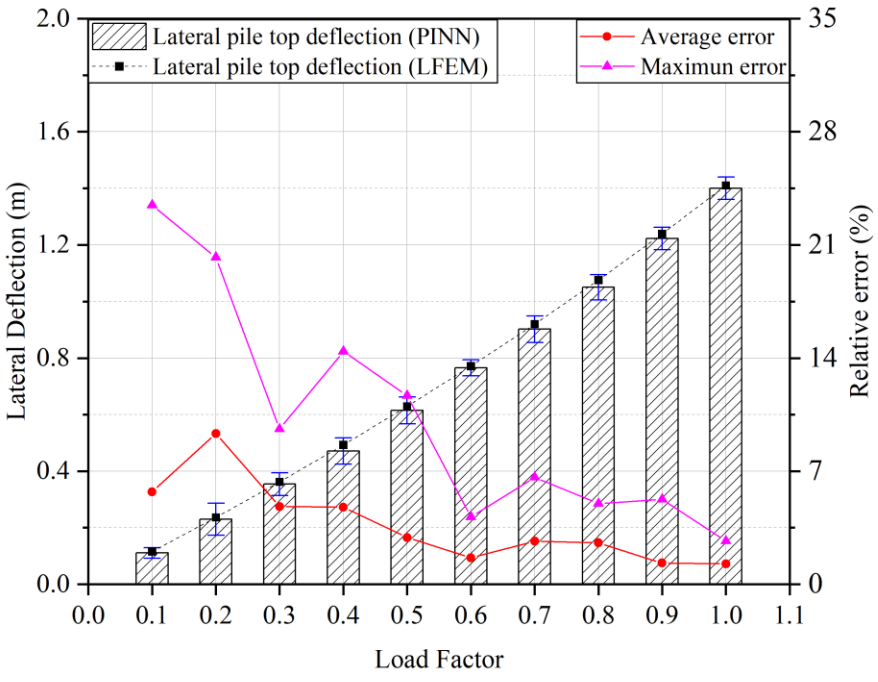
(B)

Figure 6-17 Lateral deflection along the pile under different load conditions: (A) Results of PINN with inappropriate loss normalization; and (B) Results of PINN with appropriate loss normalization.

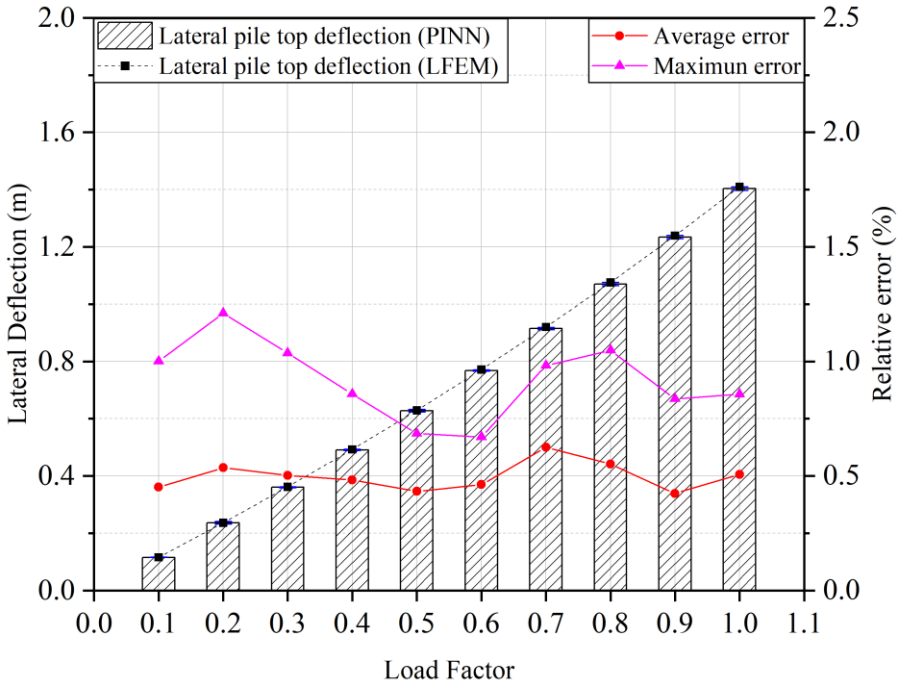
In the second part of this case study, the necessity of appropriate loss function normalization is further demonstrated. The lateral deflection of the studied pile, subjected to ten LFs ranging from 0.1 to 1.0, is investigated. The benchmark solution for the lateral deflection is obtained through the LFEM method. To overcome the failure in minimizing the loss function, the normalization factors for $\boldsymbol{L}_T$, N_T, is set according to Section 3.3 to maintain the magnitudes of all loss terms within a close range. Two PINN models with different loss normalizations are used to predict the structural behavior of the pile: (1) the PINN with inappropriate loss normalization, where N_G and N_B are set as one, and (2) the PINN with appropriate loss normalization, where N_G and N_B are set according to Section 3.3. The lateral deflections obtained from two PINN models are plotted with the benchmark solutions in Figure 6-17. It is clearly found that the PINN with appropriate loss normalization is capable of more accurately predicting the

structural behavior of the pile compared with the one with inappropriate loss normalization.

Due to the stochastic nature of the Adam optimizer, it is challenging to draw general conclusions based on once numerical experiment. To further investigate the effect of loss normalization, multiple training processes of the PINN models are conducted for the aforementioned problem in the rest part of this case study. Specifically, for each LF, the previous PINN models with two loss normalization are trained individually ten times and then used to predict the lateral deflection at the top end of the pile. The prediction results obtained from different PINN models are compared against the benchmark solutions obtained from the LFEM and presented in Figure 6-18, where the histogram and error bars are employed to illustrate the average values and standard deviations, respectively. Additionally, the relative error of the PINN prediction, calculated by the ratio of the absolute PINN prediction error to that of the benchmark solution, is also presented in Figure 6-18.



(A)



(B)

Figure 6-18 Pile top lateral deflection under different load factors: (A) Results from PINN with inappropriate loss normalization; and (B) Results from PINN with appropriate loss normalization.

Observed from Figure 6-18, it can be found that, for the PINN model with inappropriate loss normalization, even though the average values of the prediction are close to the benchmark solutions, the corresponding standard deviation is larger than the prediction results from the PINN model with appropriate loss normalization. Besides, the standard deviation of the PINN prediction with inappropriate loss normalization is less sensitive to the LF. When the LF is small, indicating that the corresponding exact solution for the lateral pile top deflection is also small, such standard deviations can result in a considerable relative error of the PINN model with inappropriate loss normalization, sometimes exceeding 20%. On the other sides, the PINN model with appropriate loss normalization can accurately and robustly predict the structural behaviors of the pile subjected to different LFs. The relative error of the prediction from the PINN model with appropriate loss normalization maintains within

a reasonable range even when the magnitudes of the exact solutions vary widely. From the above observation, it can be concluded that appropriate loss normalization of the PINN is necessary, especially when the magnitude of the exact solution may vary widely.

6.6.2 Case study 2 - Effect of adaptive loss weight algorithm

In the present study, the adaptive loss weight algorithm proposed by Wang et al. [87] is employed to enhance the numerical efficiency of the multi-objective PINN model training process. The effect of this algorithm is investigated by studying a series of piles with varying lengths from 20 m to 45 m. The cross sections of the studied piles are with a moment of inertia of I = 4.909 x 10^{-2} m^4 and a section area of A = 7.854 x 10^{-1} m^2. The applied load at the pile top is maintained constant for all piles with an axial compression of 3500 kN, a shear force of 500 kN, and a bending moment of 1000 kN·m. To assess the effect of the adaptive loss weight algorithm, two types of weights for loss terms are employed during the model training: (1) constant loss weight, which denotes that all the loss term weights are set to one, and (2) adaptive loss weight, which denotes all the loss term weights are automatically assigned and updated by the algorithm during the model training. The results are demonstrated by tracking the evolution of the summation of all the unweighted loss terms, ($\boldsymbol{L}_G + \boldsymbol{L}_B + \boldsymbol{L}_T$), during the model training.

Table 6-5 Average epochs during the model training

	Average training epochs	
***L* (*m*)**	**Constant loss weight**	**Adaptive loss weight**
20	96	127
25	208	171
30	425	285
35	2535	554
40	1854	1191
45	3163	1743

Σ	8281	4071
	$(4071 \div 8281) \times 100\% = 49.1\%$	

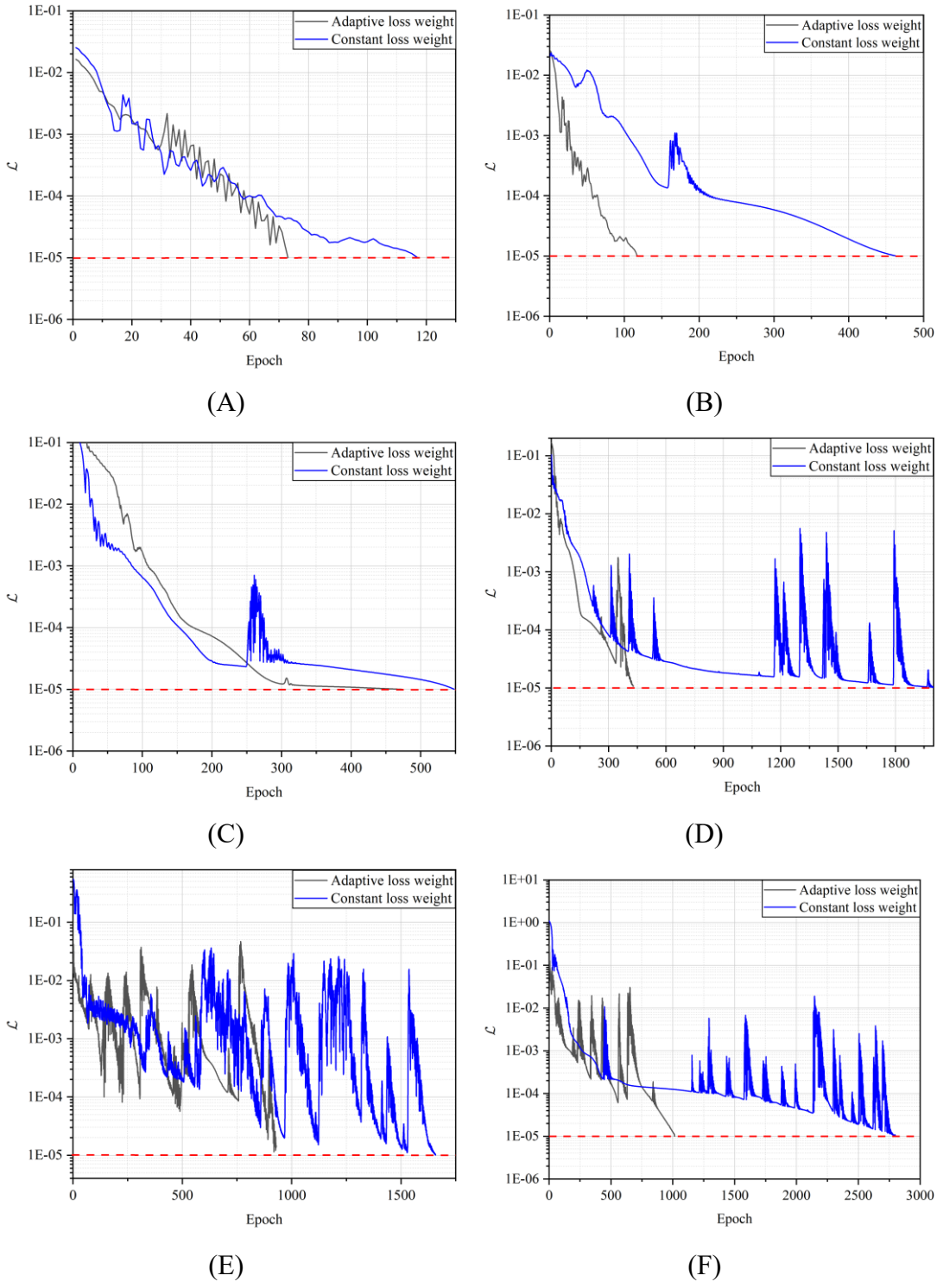


Figure 6-19 Evolution of summation of unweighted loss terms: (A) L = 20 m; (B) L = 25 m; (C) L = 30 m; (D) L = 35 m; (E) L = 40 m; and (F) L = 45 m.

Figure 6-19 demonstrates that the adaptive loss weight algorithm significantly reduces the required training epochs to minimize the loss function within the tolerance when compared to using the constant loss weight during model training. To further evaluate the impact of the adaptive loss weight algorithm, multiple model training processes are conducted. The PINN models with two different loss weights are individually trained ten times to obtain the numbers of the required training epochs for minimizing the loss function within the tolerance. The average numbers of training epochs for different models are summarized in Table 6-5. It can be found that the average number of required training epochs for the PINN model with adaptive loss weight is reduced to 50% of the one with constant loss weight. This reduction indicates a substantial improvement in computational efficiency associated with the adaptive loss weight algorithm during model training.

6.6.3 Case study 3 - Effect of regression-based soil resistance estimation

The accuracy of predicting the structural behaviors of piles when considering some complex SPI using the proposed method is validated in Examples 2 - 3, where the regression-based soil resistance estimation is implemented to reflect the *p-y* relations during training, as presented in Section 4.2. Here, some examples are provided to illustrate the training failure induced by directly obtaining the soil resistance through the *p-y* relations and the effectiveness of incorporating regression of *p-y* relations via the NN model during the training process of PINN.

Except for API clay *p-y* relations, another *p-y* relations developed by Matlock [158] is utilized in this case study to describe the lateral soil resistance of the pile embedded in the clay. It should be noted that these two *p-y* relations are basically equivalent since the API *p-y* relations are the interpolation of the Matlock clay *p-y* relations, which can be expressed as:

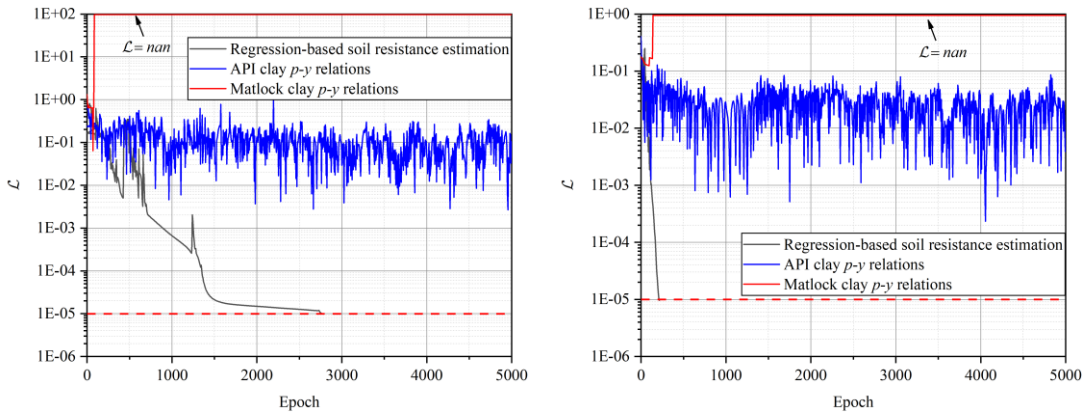
$$p = \begin{cases} \frac{1}{2} p_u \left(\frac{y}{y_c}\right)^{0.5} & when \quad y < y_c \\ p_u & when \quad y \geq y_c \end{cases}$$

where p is the lateral pile deflection; y represents the lateral pile deflection; and p_u and y_c are the ultimate resistance and pile displacement resulting from the soil around the piles reaching 50% of its ultimate resistance, respectively.

A circular concrete pile of L = 30 m in length and D = 1 m in diameter is studied. The pile is assumed to be embedded in two types of clay, which are: (1) silty clay of cohesion coefficient, c = 15.7 kPa, fictional angle, φ = 19 °, and unit weight, γ = 18.1 kg/m^3; and (2) soft clay of cohesion coefficient, c = 16.0 kPa, fictional angle, φ = 2°, and unit weight, γ = 19.0 kg/m^3. The pile is subjected to an axial compression of 3000 kN, a shear force of 1000 kN, and a bending moment of 2000 kN·m at its top. To predict the structural behavior of the pile, a PINN model is utilized. During the training process, various methods are employed to obtain the soil resistance, including (1) obtaining the soil resistance through the linear interpolation of API clay *p-y* relations, (2) obtaining the soil resistance directly from Matlock clay *p-y* relations, and (3) estimate the soil resistance through the regression via the NN model. The loss function evolution during different training processes is plotted in Figure 6-20.

From the observations in Figure 6-20, it is evident that the loss functions of the PINN model training processes, where the lateral soil resistance is obtained through the linear interpolation of API clay *p-y* relations, fail to minimize within the tolerance even when the training epoch grows to 5000. This non-convergence is induced by the automatic differentiation technique being unable to generate the loss function derivative accurately since the computational graph of the lateral soil resistance obtained by linear interpolation from API clay *p-y* relations is not created. Additionally, severe numerical instability is observed in the training process where the lateral soil resistance is directly obtained through Matlock clay *p-y* relations, as indicated by the loss function values growing to "nan", which denotes infinity in the ML study. Such numerical instability

arises from the n^{th} order derivative (where $n > 1$) of Matlock clay *p-y* relations, which tends towards infinity when y is close to zero. Conversely, the implementation of regression-based soil resistance estimation can prevent the aforementioned issue and result in a successful training process of the PINN.



(A) (B)

Figure 6-20 Evolution of loss function of the PINN for piles in two geological conditions: (A) Silty clay; and (B) Soft clay.

Table 6-6. Numbers of status for model training processes

Ground media	**Silty clay**			**Soft clay**		
Training status*	***S***	***I***	***N***	***S***	***I***	***N***
Regression-based soil resistance estimation	10	0	0	10	0	0
API clay *p-y* relations	0	0	10	0	0	10
Matlock clay *p-y* relations	0	10	0	0	10	0

(*: ***S***, ***I***, and ***N*** denote the success, instability, and non-convergence of the training, respectively.)

To validate the necessity of regression-based soil resistance estimation, ten individual training processes are conducted for all the PINN models described above.

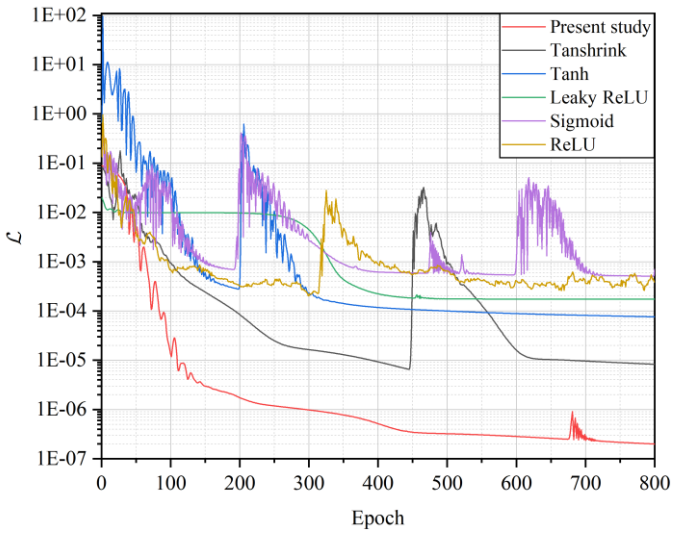
The non-convergence of the model training process is defined as the loss function not meeting the tolerance level when the training epoch reaches 5000, while the instability of the training process is defined as the loss function becoming "nan" during the training. Conversely, if the loss function meets the tolerance level within 5000 epochs, the training process is regarded as a success. The numbers of statuses in different repetitive PINN model training processes are documented in Table 6-6. The results indicate that all model training processes that directly obtain the soil resistance from *p-y* relations experience non-convergence or instability, while the implementation of regression-based soil resistance estimation successfully avoids such issues. Thus, it is believed that implementing regression via the NN model is an effective solution to avoid training failures when dealing with non-differentiable *p-y* relations to estimate the soil resistance during the PINN model training process. More generally, introducing regression via the NN model is believed to be an effective solution for reflecting non-differentiable functions in other PINN studies.

Appendix A– Effect of Activation Functions

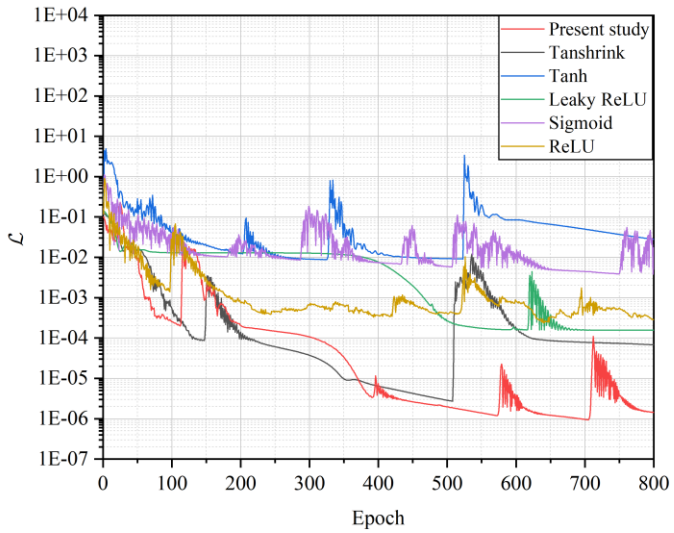
The efficiency and effectiveness of NN model training is significantly influenced by the selection of activation functions. This study aims to discuss the effect of activation functions by investigating a circular concrete pile of D = 1 m in diameter. The elastic modulus of the investigated pile is E = 34.5 GPa. The pile length, L, is varied from 20 m to 40 m. The *p-y* relations of the pile is regarded as linear with the constant soil modulus k = 200 kN/m^2. The applied axial compression, shear force, and bending moment at the pile top equal 2500 kN, 600 kN, and 1000 kN·m, respectively. Several PINN models with different activation functions are trained to predict the structural behaviors of the pile. The activation functions employed in the models include: (1) *Tanshrink*, *Tanh*, *Leaky ReLU*, *Sigmoid*, and *ReLU* as the activation functions in all neural layers; and (2) *Tanshrink* as the activation functions in both input and output layers, and *Tanh* as the activation functions in the hidden layers, as presented Section 3.2. The comparison results between models with different activation functions are

demonstrated through the loss function evolutions during the model training.

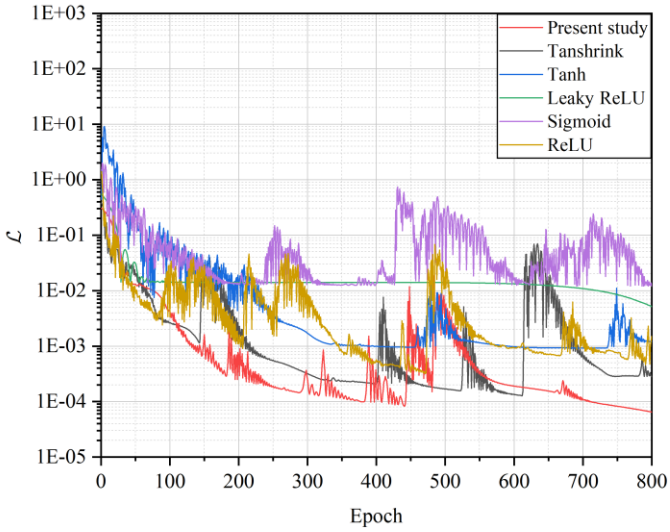
The results of the comparison are depicted in Figure 6-21. The results clearly show that using *Tanshrink* as the activation functions in both input and output layers, and *Tanh* as the activation functions in the hidden layers, outperforms other selections of activation functions in terms of training efficiency. Therefore, in this study, all the PINN models are constructed using *Tanshrink* and *Tanh* as the activation functions.



(A)



(B)



(C)

Figure 6-21 Evolution of loss functions for different PINN models: (A) L = 20 m; (B) L = 30 m; and (C) L = 40 m.

Appendix B – Sensitivity Study of Sampling Point Numbers

The sensitivity study of the sampling point number is conducted. The pile in Appendix A is revisited here. The pile length is taken as 30 m. The LFEM using 30 elements is selected as the benchmark solution to obtain the lateral pile deflection at the top end. For comparison, different sampling point numbers, including n = 3 / 5 / 10 / 15 / 20 / 25 / 30 / 40 / 50, are employed during the PINN model training process. For each sampling point numbers, ten individual model training processes are conducted. The analysis results are presented through the relative error of the PINN model prediction compared to the benchmark solution. As demonstrated in Table 6-7, when the sampling point number grows to 50, the relative error of the PINN prediction is smaller than 0.5 %, which is accurate enough. Thus, in this chapter, the sampling point number is recommended as 50.

Table 6-7 Relative error of the PINN prediction with different sampling point numbers

	Relative error (%)	
n	**Average value**	**Maximum value**

3	44.67	67.54
5	30.09	77.29
10	9.03	18.44
15	3.71	7.55
20	2.05	2.53
25	1.22	2.36
30	0.62	0.83
40	0.30	0.63
50	0.25	0.44

Appendix C – Adam Optimizer

The Adam update rule be expressed as:

$$\boldsymbol{\theta}^{e+1} = \boldsymbol{\theta}^{e} - \frac{\alpha}{\sqrt{\hat{v}^{e}} + \varepsilon}\hat{m}^{e}$$

$$\hat{m}^{e} = \frac{m^{e}}{1-\beta_1}, \hat{v}^{e} = \frac{v^{e}}{1-\beta_2}$$

$$m^{e} = \beta_1 m^{e-1} + (1-\beta_1)\nabla_{\boldsymbol{\theta}}\boldsymbol{L}$$

$$v^{e} = \beta_2 v^{e-1} + (1-\beta_2)(\nabla_{\boldsymbol{\theta}}\boldsymbol{L})^2$$

where the superscripts ^{e} denotes the current epoch index in the model training; m and v denote the estimates of the first moment and the second moment of the gradients, respectively, both of which are initialized as zero at the training start; $\nabla_{\boldsymbol{\theta}}\boldsymbol{L}$ represents the gradients of the loss function, $\boldsymbol{L}$, with respect to model parameter set, $\boldsymbol{\theta}$; $(\nabla_{\boldsymbol{\theta}}\boldsymbol{L})^2$ is the element-wise squaring of $\nabla_{\boldsymbol{\theta}}\boldsymbol{L}$; β_1, β_2, ε, and α are the hyperparameters, which are taken as 0.9, 0.999, 10^{-8}, and 0.001, respective.

Chapter 7. SECOND-ORDER ANALYSIS METHOD FOR PILE-SUPPORTED FRAMED STRUCTURES USING NOEM

7.1 Introduction

The traditional line finite element method (LFEM) provides accurate single-pile analysis of nonlinear Soil–Pile Interaction (SPI) using discrete spring elements, but it is inefficient for pile-supported structures with many piles. Although the refined three-dimensional pile element (3DPE; Chapter 3) offers a viable numerical alternative, it still requires a relatively dense mesh to achieve comparable accuracy. To address these limitations, this chapter implements the neural operator element method (NOEM) for second-order analysis, using a single element per pile while maintaining accuracy.

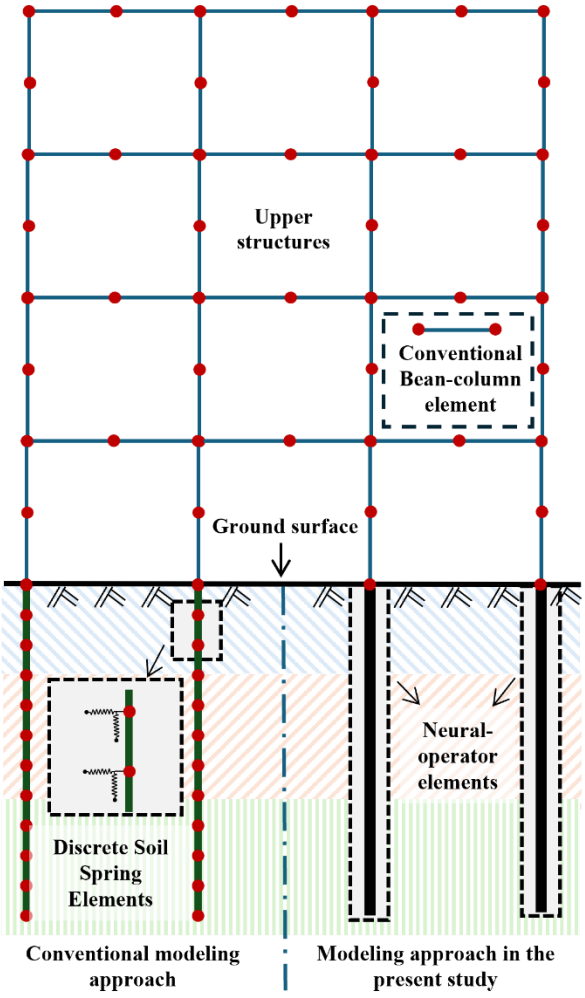


Figure 7-1 Comparison between two modeling approaches for Soil-Pile Interactions

In the chapter, the neural operator element method (NOEM; Chapter 4) is implemented for the nonlinear SPI analysis of large-scale pile-supported framed structures using the corresponding governing equations (Chapter 6). A classical NO model, named the deep operator network (DeepONet), is introduced, based on which the general framework for the NOEM is reviewed. Then, the NOEM is implemented for the simulation of large-scale pile-supported structures, where the conventional Euler-Bernoulli element and the NOE formulated with the DeepONet trained for single piles are employed to model upper structures and the pile foundations. Moreover, an improving ML training strategy, Residual-based Adversarial-gradient Moving Sample (RAMS; Chapter 5), is also introduced to enhance the NO performance when trained by the same amounts of dataset. To account for the complex SPI within the NOEM framework, a local linearization technique is proposed to simplify the original *p-y* curve model into a linear *p-y* curve model, which is proven to be accurate as it preserves the zero- and first-order derivative values of the nonlinear model. Extensive numerical examples illustrate the accuracy of the NOEM implementation for pile-supported structures. The results demonstrate that NOEM, using a single NOE, can accurately model individual piles in complex geological conditions without compromising accuracy while also effectively captures the structural response along the piles (Figure 7-1). Thus, it is believed that the proposed method could offer a promising tool for the further development of second-order analysis method for pile-supported framed structures in design practices.

7.2 Neural Operator Element Method for Pile-supported Framed structures

In this section, the employed ML model architeture, DeepONet, for modeling structural response of single piles is introduced. Accordingly, an improved ML training strategy, named Residual-based Adversarial-gradient Moving Sample (RAMS) method,

is introduced. Next, the NOEM for the general partial differential equation problem is brief review. Finally, a detailed implementation process of the NOEM for pile-supported framed structures considering the nonlinear SPI is presented.

7.2.1 Deep Operator Network for modeling single pile

In this section, DeepONet, a famous NO model, is revisited and implemented as the surrogate model approximates the relations between the pile top displacement and the deflection along piles.

The conventional ML models are commonly designed to approximate some given function mapping between two scaler or vector variables. Among various ML models, the multi-layer perceptron (MLP) is the most-commonly used given its strong approximation capability guaranteed by the universal approximation theorem. The MLP model can be written as:

$$N(x) \coloneqq f_n \circ \sigma_{n-1} \circ f_{n-1} \dots \sigma_2 \circ f_1 \circ \sigma_0 \circ f_0(x)$$

where the model input, x, could be either a scaler or vector; σ_i is the nonlinear activation function in i^{th} layer, which is selected as the hyperbolic tangential function, $\sigma_i = \frac{e_x - e_{-e}}{e_x + e_{-e}}$, in the present study; and f_i is linear maps at the i$^{\text{th}}$ layer, respectively, which can be further given as:

$$f_i(y_{i-1}) \coloneqq W_i y_{i-1} + b_i$$

where y_i is the output of the i^{th} layer; and W_i and b_i are the weight matrix and the bias vectors at the i^{th} layer, which are all trainable parameters. 1

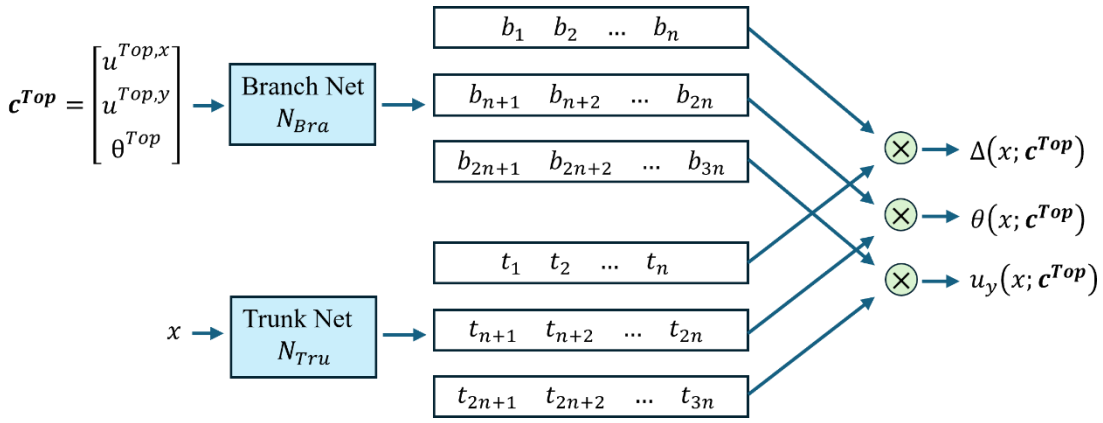


Figure 7-2. Architecture of DeepONet modeling single piles.

Even though the conventional ML models, such as the MLP, have already shown satisfactory performance as the surrogate models in the function approximation. However, when considering some more complicated tasks requiring approximating operator mapping between function spaces, those conventional ML models designated for function approximation sometimes perform poor generalization. Thus, Lu et al. [132] proposed a new ML architecture, DeepONet, via extending the universal operator approximation theorem [133] to the deep neural networks. The DeepONet can effectively approximate the operator mapping to the function space with two independent MLPs, termed as the branch net and the trunk net, respectively. In this chapter, we introduce this novel ML architecture (Figure 7-2) to approximate the mapping between the pile top deformation to the structural response along pile (Figure 7-3) according to the governing equations in Chapter 6, which can be written as:

$$\mathcal{G}: \mathbb{R}^3 \ni \boldsymbol{c^{Top}} \mapsto v \in \mathcal{V},$$

where $\boldsymbol{c^{Top}} = [u^{Top,x} \quad u^{Top,y} \quad \theta^{Top}]$ is the boundary condition represented by the pile top displacement including the vertical pile top displacement, $u^{Top,x}$, the lateral pile top displacement, $u^{Top,y}$, and the pile top rotation, θ^{Top}; v is a vector-value function which can be given as:

$$v(x;\boldsymbol{c^{Top}}) = \left[\Delta(x;\boldsymbol{c^{Top}}) \quad \theta(x;\boldsymbol{c^{Top}}) \quad u_y(x;\boldsymbol{c^{Top}})\right]$$

where $\Delta(x;\boldsymbol{c^{Top}})$, $\theta(x;\boldsymbol{c^{Top}})$, and $u_y(x;\boldsymbol{c^{Top}})$ denotes the axial deformation, the rotation, and the lateral pile displacement at the location x when considering the boundary condition, $\boldsymbol{c^{Top}}$, respectively.

The employed DeepONet (Figure 7-2) is composed of two MLPs, termed as the branch net and the trunk net, respectively. The two subnets take the boundary condition and the axial coordinate of piles as the input:

$$N_{Bra}(\boldsymbol{c^{Top}}) = [b_1(\boldsymbol{c^{Top}}) \quad b_2(\boldsymbol{c^{Top}}) \quad \dots \quad b_{3n-1}(\boldsymbol{c^{Top}}) \quad b_{3n}(\boldsymbol{c^{Top}})]^T$$

$$N_{Tru}(x) = [t_1(x) \quad t_2(x) \quad \dots \quad t_{3n-1}(x) \quad t_{3n}(x)]^T$$

where N_{Bra} and N_{Tru} are the branch net and the trunk net, respectively; and $3n$ is the size of the outputs from N_{Bra} and N_{Tru}.

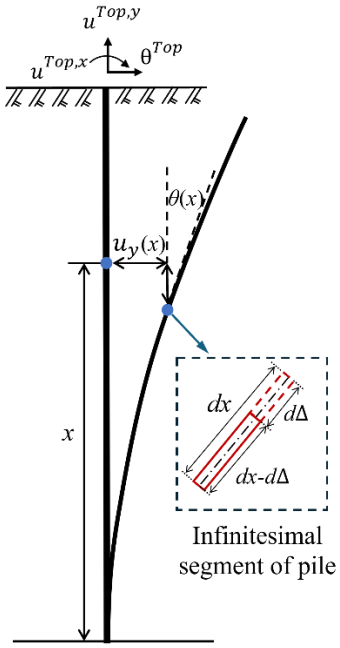


Figure 7-3. Illustration of structural response of single piles.

Then, the pile response along the pile at the location x when the boundary conditions $\boldsymbol{c^{Top}}$ is given can be approximated as:

$$\mathcal{G}(\boldsymbol{c^{Top}})(x) = \left[\sum_{i=1}^{n} b_i(\boldsymbol{c^{Top}})t_i(x) \quad \sum_{i=n+1}^{2n} b_i(\boldsymbol{c^{Top}})t_i(x) \quad \sum_{i=2n+1}^{3n} b_i(\boldsymbol{c^{Top}})t_i(x)\right]$$

where these three terms are employed to approximate the axial deformation, $\Delta(x; \boldsymbol{c^{Top}})$, the rotation, $\theta(x; \boldsymbol{c^{Top}})$, and the lateral pile displacement, $u_y(x; \boldsymbol{c^{Top}})$, respectively, as:

$$\Delta(x; \boldsymbol{c^{Top}}) \approx \sum_{i=1}^{n} b_i(\boldsymbol{c^{Top}})t_i(x)$$

$$\theta(x; \boldsymbol{c^{Top}}) \approx \sum_{i=n+1}^{2n} b_i(\boldsymbol{c^{Top}})t_i(x)$$

$$u_y(x; \boldsymbol{c^{Top}}) \approx \sum_{i=2n+1}^{3n} b_i(\boldsymbol{c^{Top}})t_i(x)$$

In the present study, the data-driven training method is employed. Thus, the loss function of the DeepONet can be given as:

$$\mathcal{L}(\theta_{\mathcal{G}}) = \frac{1}{n_{Sam}} \sum_{i=1}^{n_{Sam}} \left\| \mathcal{G}(\boldsymbol{c_i^{Top}})(x_i) - p_i \right\|^2$$

where $\theta_{\mathcal{G}}$ is the collection of all the trainable parameters in the model, $\mathcal{G}$; $\{\boldsymbol{c_i^{Top}} \quad x_i \quad p_i\}_{i=1}^{n_{Sam}}$ are the pre-collected dataset, where p_i represents a 1 x 3 vectors for the pile response as:

$$p_i = [\Delta_i \quad \theta_i \quad u_{y,i}]$$

Accordingly, the trainable model parameters, $\theta_{\mathcal{G}}$, can be computed by:

$$\theta_{\mathcal{G}} = arg \min_{\theta_{\mathcal{G}}} \{\mathcal{L}(\theta_{\mathcal{G}})\}$$

which is solved via the Adam optimizer [262] built in PyTorch [118].

7.2.2 Residual-based adversarial-gradient moving sample method

The performance of ML models is generally highly dependent on the size of the

training dataset. However, training models with large datasets often entails expensive data collection and significant training cost. As a result, extensive research has been conducted in recent years to develop various training and sampling strategies that enhance model performance while using the same sample size. This study implement the RAMS technique to improve the performance of DeepONet for modeling single piles, utilizing the same sample size. In RAMS, the samples are treated as trainable parameters, maximizing the physics-informed loss given by the underlying governing equations. This approach allows for the identification and sampling of areas with high physics-informed loss, indicating regions that require further learning. For simplicity, the implementation process of RAMS for DeepONet modeling single piles is omitted here. Detailed descriptions of the RAMS implementation and the physics-informed loss for single piles are provided in Chapters 4 and 5, respectively.

7.2.3 Neural Operator Element Method

Recently, a hybrid numerical framework has been developed to synergistically combine emerging machine-learning techniques with the conventional finite-element method (FEM), enabling efficient and accurate simulation of large-scale partial differential-equation (PDE) problems. This framework, termed the Neural Operator Element Method (NOEM; as detailed in Chapter 3), embeds a pretrained neural operator into the standard variational formulation. A brief overview is given below.

Consider the following PDE problem:

$$\mathcal{D}[u] = 0, \qquad for\ x \in \Omega$$

$$\mathcal{B}[u] = \bar{u}, \qquad for\ x \in \partial\Omega$$

where $\mathcal{D}$ and $\mathcal{B}$ are the differentiation operator and the boundary condition operator, respectively; Ω is the computational domain of the PDE; and $\bar{u}$ is the prescribed boundary condition on $\partial\Omega$.

Generally, the variation method can be written as:

$$\boldsymbol{c} = arg \min_{\boldsymbol{c} \in \mathbb{R}^D} \{J[u(\boldsymbol{c})]\} \tag{7.1}$$

where $J[\cdot]$ is the energy functional defined according to the PDE; $u(\boldsymbol{c})$ is the trial (shape) function; and $\boldsymbol{c} \in \mathbb{R}^D$, known as the nodal value vector, collects the nodal degrees of freedom.

In the conventional FEM, the shape function commonly represented by the piecewise polynomials. However, NOEM instead partitions the domain into two parts: the domain modelled by the conventional FEs, $\{\Omega_i^{FE}\}_{i=1}^{n_{FE}}$, and the domain modelled by the NOE, $\{\Omega_i^{NOE}\}_{i=1}^{n_{NOE}}$. Accordingly, the shape function can be represented as:

$$u(\boldsymbol{c}) = \sum_{i=1}^{n_{FE}} \phi_i^{FE}(x; \boldsymbol{c}_i^{FE}) + \sum_{i=1}^{n_{NOE}} \phi_i^{NOE}(x; \boldsymbol{c}_i^{NOE})$$

where $\sum_{i=1}^{n_{FE}} \phi_i^{FE}(x; \boldsymbol{c}_i^{FE})$ is the standard piecewise polynomial defined on Ω_i^{FE}; and the NOE element, ϕ_i^{NOE}, is constructed by the pre-trained neural operator as:

$$\phi_i^{NOE}(x; \boldsymbol{c}_i^{NOE}) = \begin{cases} \mathcal{G}(\boldsymbol{c}_i^{NOE})(x) & for\ x \in \Omega_i^{NOE} \\ 0 & for\ x \notin \Omega_i^{NOE} \end{cases}$$

where the neural operator, $\mathcal{G}$, maps Dirichlet data on the element boundary to the corresponding interior solution on Ω_i^{NOE} as:

$$\mathcal{G}: \overline{u_i^{NOE}} \mapsto u_i^{NOE}$$

$$\mathcal{D}[u_i^{NOE}] = 0, \qquad for\ x \in \Omega_i^{NOE}$$

$$\mathcal{B}[u_i^{NOE}] = \overline{u_i^{NOE}}, \qquad for\ x \in \partial\Omega_i^{NOE}$$

where $\overline{u_i^{NOE}}$ is the boundary conditions defined on $\partial\Omega_i^{NOE}$ and parametrized by $\boldsymbol{c}_i^{NOE}$.

7.2.4 Implementation of NOEM for pile-supported framed structures

The previous subsection presents the mathematical framework of the NOEM. In

this subsection, the detailed implementation process of NOEM for the pile-supported framed structures is illustrated. To solve Eq. (7.1), the Newton's method is introduced. Let's denote the global stiffness matrix and the global resisting force vector as:

$$\boldsymbol{K}(\boldsymbol{c}) = \frac{\partial^2 J[u(\boldsymbol{c})]}{\partial c_i \partial c_j} = \sum\nolimits_{i=1}^{n_{FE}} \boldsymbol{k}_i^{FE}(\boldsymbol{c}_i^{FE}) + \sum\nolimits_{i=1}^{n_{NOE}} \boldsymbol{k}_i^{NOE}(\boldsymbol{c}_i^{NOE})$$

$$\boldsymbol{F}(\boldsymbol{c}) = \frac{\partial J[u(\boldsymbol{c})]}{\partial c_i} = \sum\nolimits_{i=1}^{n_{FE}} \boldsymbol{f}_i^{FE}(\boldsymbol{c}_i^{FE}) + \sum\nolimits_{i=1}^{n_{NOE}} \boldsymbol{f}_i^{NOE}(\boldsymbol{c}_i^{NOE})$$

where $\boldsymbol{K}(\boldsymbol{c}) \in \mathbb{R}^D \times \mathbb{R}^D$ is the Hessian matrix of the energy functional, which is also known as the global stiffness matrix; and $\boldsymbol{F}(\boldsymbol{c}) \in \mathbb{R}^D$ represents the gradient vector of the energy functional, which is also named as the global resisting force vector.

A pile-supported framed structure can be generally partitioned into two parts: the superstructure and the pile foundations. For the former part, the conventional Euler-Bernoulli element with Hermit shape functions can effectively model the structural behavior. Thus, in the present study, the Euler-Bernoulli element is employed to model the upper structures. Accordingly, $\boldsymbol{k}_i^{FE}$ and $\boldsymbol{f}_i^{FE}$ are the element stiffness matrix and the element resisting force vector of the Euler-Bernoulli element, respectively. To account for the geometric nonlinearity from the large deformation, the co-rotational kinematic description is also introduced. For simplicity, the present study omits the detailed formulas of the Euler-Bernoulli element and the co-rotational description, which could be found in [11].

For the pile foundations, the conventional LFEM commonly requires a dense mesh to accurately capture its structural response due to the complex ground condition and the nonlinear SPI. Thus, the NOE composed of $\boldsymbol{k}_i^{NOE}$ and $\boldsymbol{f}_i^{NOE}$ is employed to model pile foundations in pile-supported structures.

The employed neural operator maps the displacement boundary condition at the pile top to the structural response along the pile as:

$$\boldsymbol{c}_i^{NOE} = \{u_i^{Top,x} \quad u_i^{Top,y} \quad \theta_i^{Top}\}$$

$$\mathcal{G}(\boldsymbol{c}_i^{NOE})(x) = \{\Delta(x, \boldsymbol{c}_i^{NOE}) \quad \theta(x, \boldsymbol{c}_i^{NOE}) \quad u_y(x, \boldsymbol{c}_i^{NOE})\}$$

where $u_i^{Top,x}$, $u_i^{Top,y}$, and θ_i^{Top} are the vertical displacement, the lateral displacement, and the rotation at the i^{th} pile top, respectively; and Δ, θ, and u_y are the axial deformation, the rotation, and the lateral pile displacement along the single pile modelled by the i^{th} NOE.

According to the governing equations in Chapter 6, the energy functional of the NOE modeling the i^{th} pile can be written as:

$$J_i^{NOE}(\boldsymbol{c}_i^{NOE}) = \frac{1}{2}\int_0^L EA\delta^2(x; \boldsymbol{c}_i^{NOE})dx + \frac{1}{2}\int_0^L EI\theta^2(x; \boldsymbol{c}_i^{NOE})dx + \int_0^L \int_0^{u_y(x,\boldsymbol{c}_i^{NOE})} p(x; u_y)du_y\, dx$$

in which L is the pile length.

Monte-Carlo Integral is utilized to approximate the energy functional as:

$$J_i^{NOE}(\boldsymbol{c}_i^{NOE}) \approx \frac{L}{2N}\sum_j^N [EA\delta^2(x_j, \boldsymbol{c}_i^{NOE}) + EI\theta^2(x_j, \boldsymbol{c}_i^{NOE})] + \frac{L}{N}\sum_j^N \left[\int_0^{u_y(x_j,\boldsymbol{c}_i^{NOE})} p(x_j, u_y)du_y\right]$$

where N is the sample number for Monte-Carlo Integral; and x_j is the sample location in Monte-Carlo Integral.

However, even with Monte-Carlo Integral is employed, the last term accounting for the SPI is still hard to calculate given the double integral existed. Considering that $J_i^{NOE}(\boldsymbol{c}_i^{NOE})$ is only utilized to compute the Hessian matrix and the gradient vector, we only need to guarantee its first- and second-order derivatives are accurate, which can

be written as:

$$\frac{\partial}{\partial \boldsymbol{c}_i^{NOE}} \int_0^{u_y(x_j, \boldsymbol{c}_i^{NOE})} p(x_j, u_y) du_y = p\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right) \frac{\partial}{\partial \boldsymbol{c}_i^{NOE}} u_y(x_j; \boldsymbol{c}_i^{NOE})$$

$$\mathcal{H}_i\left(\int_0^{u_y(x_j, \boldsymbol{c}_i^{NOE})} p(x_j, u_y) du_y\right) = p\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right) \mathcal{H}_i\left(u_y(x_j; \boldsymbol{c}_i^{NOE})\right)$$

$$+ \frac{\partial}{\partial u_y} p\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right) \frac{\partial}{\partial \boldsymbol{c}_i^{NOE}} u_y(x_j, \boldsymbol{c}_i^{NOE}) \left\{\frac{\partial}{\partial \boldsymbol{c}_i^{NOE}} u_y(x_j; \boldsymbol{c}_i^{NOE})\right\}^T$$

where $\frac{\partial \cdot}{\partial \boldsymbol{c}_i^{NOE}}$ and $\mathcal{H}_i(\cdot)$ are the first-order gradient and the Hessian matrix with respect to $\boldsymbol{c}_i^{NOE}$, respectively.

As illustrated above, we proposed a local linearization technique for the *p-y* model employed. In this technique, the *p-y* curve can locally linearize the *p-y* curve model at x_j, as $\bar{p}(x_j, u_y)$, while ensuring it satisfies the following conditions:

$$\frac{\partial}{\partial u_y} \bar{p}\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right) = \frac{\partial}{\partial u_y} p\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right)$$

$$\bar{p}\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right) = p\left(x_j, u_y(x_j; \boldsymbol{c}_i^{NOE})\right)$$

Thus, we replace the original nonlinear *p-y* curve as a linear model:

$$\bar{p}(x_j; u_y) = k_j(u_y - u_{j,y}) + p(x_j, u_{j,y})$$

where $u_{j,y} = u_y(x_j, \boldsymbol{c}_i^{NOE})$; and $k_j = \frac{\partial}{\partial u_y} p(x_j, u_{j,y})$.

Accordingly, the energy functional for the NOE can be rewritten as:

$$J_i^{NOE}(\boldsymbol{c}_i^{NOE}) \approx \frac{L}{2N} \sum_j^N \left[EA\Delta^2(x_j; \boldsymbol{c}_i^{NOE}) + EI\theta^2(x_j; \boldsymbol{c}_i^{NOE})\right] + \frac{L}{N} \sum_j^N \left[\int_0^{u_y'} \bar{p}(x_j, u_y) du_y\right]$$

$$= \frac{L}{2N}\sum_{j}^{N}\left\{EA\Delta^2(x_j;\boldsymbol{c}_i^{NOE}) + EI\theta^2(x_j;\boldsymbol{c}_i^{NOE}) + k_j u_y^2(x_j;\boldsymbol{c}_i^{NOE}) + [p(x_j;u_{j,y}) - k_j u_{j,y}]u_y(x_j;\boldsymbol{c}_i^{NOE})\right\}$$

(7.2)

Then, the iteration process for solving Eq. (7.1) based on the Newton-Raphson method can be written as:

Set unbalanced force vector, $\Delta\boldsymbol{F}(\boldsymbol{c})$ according to incremental external load.

Compute the stiffness matrix of the Euler-Bernoulli elements, $\boldsymbol{k}_i^{FE}(\boldsymbol{c}_i^{FE})$.

Compute the energy functional of the NOEs, $J_i^{NOE}(\boldsymbol{c}_i^{NOE})$, according to Eq. (7.2).

Compute the stiffness matrix of the NOEs using the automation differentiation:

$$\boldsymbol{k}_i^{NOE}(\boldsymbol{c}_i^{NOE}) = \mathcal{H}\left(J_i^{NOE}(\boldsymbol{c}_i^{NOE})\right)$$

Assemble the element stiffness matrix to the global stiffness matrix:

$$\boldsymbol{K}(\boldsymbol{c}) = \frac{\partial^2 J[u(\boldsymbol{c})]}{\partial c_i \partial c_j} = \sum\nolimits_{i=1}^{n_{FE}} \boldsymbol{k}_i^{FE}(\boldsymbol{c}_i^{FE}) + \sum\nolimits_{i=1}^{n_{NOE}} \boldsymbol{k}_i^{NOE}(\boldsymbol{c}_i^{NOE})$$

Compute the incremental nodal value vector:

$$\Delta\boldsymbol{c} = \boldsymbol{K}^{-1}(\boldsymbol{c})\Delta\boldsymbol{F}(\boldsymbol{c})$$

Update the nodal value vector as $\boldsymbol{c} = \boldsymbol{c} + \Delta\boldsymbol{c}$

Compute the element resisting force vector of the Euler-Bernoulli elements, $\boldsymbol{f}_i^{FE}(\boldsymbol{c}_i^{FE})$.

Compute the energy functional of the NOEs, $J_i^{NOE}(\boldsymbol{c}_i^{NOE})$, according to Eq. (7.2).

Compute the element resisting force vector of the NOEs using the automation differentiation:

$$\boldsymbol{f}_i^{NOE}(\boldsymbol{c}_i^{NOE}) = \frac{\partial}{\partial \boldsymbol{c}_i^{NOE}} J_i^{NOE}(\boldsymbol{c}_i^{NOE})$$

Assemble the element resisting force vector to the global resisting force vector:

$$\boldsymbol{F}(\boldsymbol{c}) = \frac{\partial J[u(\boldsymbol{c})]}{\partial c_i} = \sum\nolimits_{i=1}^{n_{FE}} \boldsymbol{f}_i^{FE}(\boldsymbol{c}_i^{FE}) + \sum\nolimits_{i=1}^{n_{NOE}} \boldsymbol{f}_i^{NOE}(\boldsymbol{c}_i^{NOE})$$

Compute the unbalanced force using the global resisting force vector and the external load vector:

$$\Delta \boldsymbol{F}(\boldsymbol{c}) = \boldsymbol{F}_{External} - \boldsymbol{F}(\boldsymbol{c})$$

If the convergency criterion is satisfied, step into the next load step. Otherwise, return to Step 2.

The convergency criterion employed in the present study can be written as:

$$\frac{\|\Delta \boldsymbol{F}(\boldsymbol{c})\|}{\boldsymbol{F}_{External}} \leq TOL \text{ and } \frac{\|\Delta \boldsymbol{c}\|}{\boldsymbol{c}} \leq TOL$$

where $TOL = 0.001$ is the error tolerance.

7.3 Validation Examples

This section presents three case studies that evaluate the performance of the NOEM for pile-supported structures. With only one element assigned to model each pile, the NOEM can accurately capture the nonlinear SPI of the complex geological condition. Moreover, only one single NO is trained and reused across all examples, underscoring the excellent reusability of the models in the proposed method.

7.3.1 Single pile in multi-layer soil medium

In this study, the steel pile previously reported in [179]is re-examined to validate the effectiveness of the NOEM for the nonlinear analysis of single piles. The elastic modulus of the pile is equal to $E = 205\ GPa$. The cross-section area and the moment of inertia of the pile are $A = 2.21 \times 10^{-1} m^2$ and $I = 5.10 \times 10^{-4} m^4$, respectively. Two loading conditions (LCs) are considered: (1) a combination of axial compression, $P = 5000\ kN$, and shear force, $V = 500\ kN$; and (2) a combination of axial compression $P = 5000\ kN$, shear force $V = 200\ kN$, and bending moment $M = 200\ kN \cdot m$. As depicted in Figure 7-4, a multi-layered soil profile consisting of three distinct soil media is adopted. The *p-y* curves for rockfill, sand, and marine clay are constructed according to Jeong et al. [306], Ismael [307], and Kawamata et al. [308], respectively. To model the pile under these LCs, two numerical methods are employed: (1) the conventional LFEM utilizing 50, 25, and 12 elements per pile; and (2) the NOEM with a single element per pile. For constructing the NOE, a DeepONet

composed of two MLPs, each containing three hidden layers with 75 neurons per layer, is trained on 1000 samples. The LFEM with 100 elements serves as the benchmark solution here.

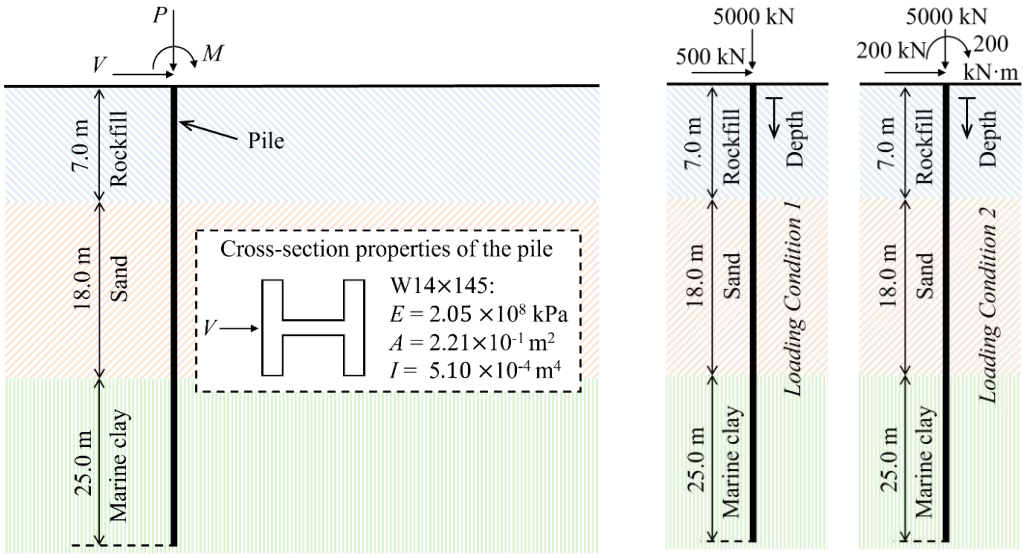


Figure 7-4 Schematics of single pile embedded in multi-layer soil medium: (A) Single pile in multi-layer soil; (B) Loading conditions.

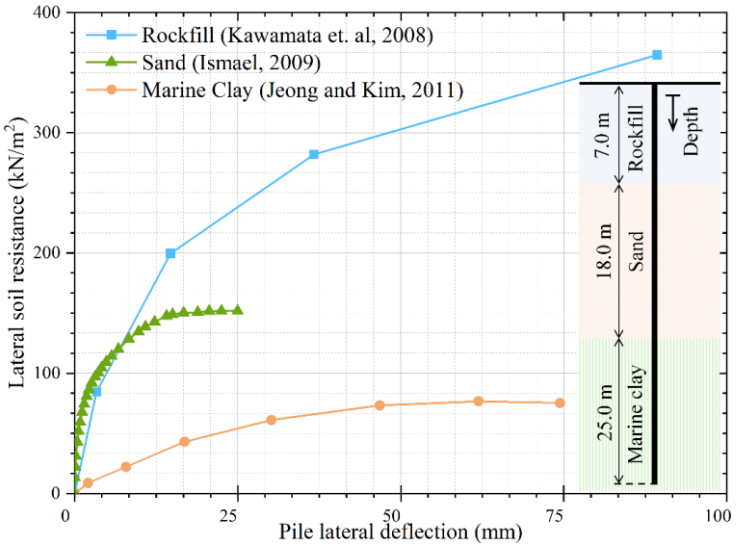
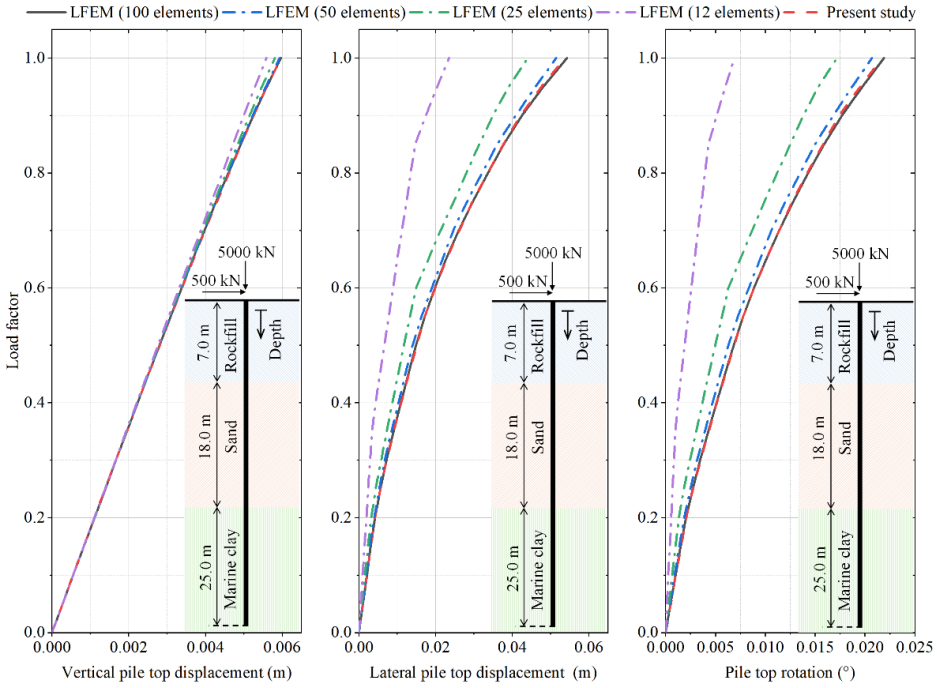
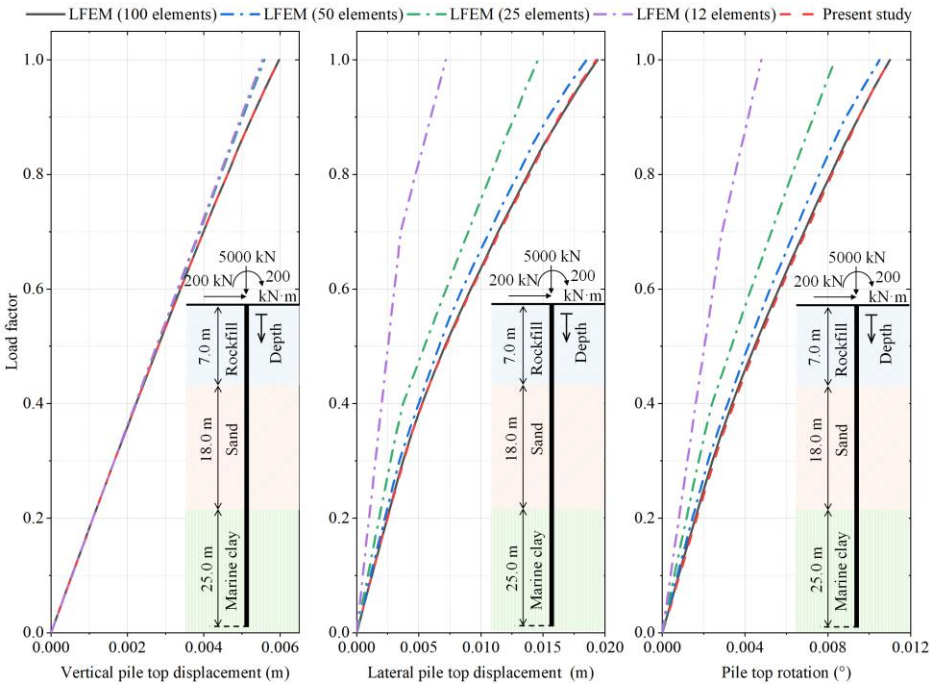


Figure 7-5 *p-y* curve for multi-layer soil medium

The comparison between different analysis methods is presented by examining the pile-top displacement. As illustrated in Figure 7-6, the LFEM with coarse meshes significantly underestimates the pile-top displacement, including the vertical and lateral deflections, as well as the pile-top rotation. Specifically, under LC 2, the LFEM with 12 elements results in approximately a 63% underestimation of lateral deflection at the pile top. Even when the mesh number is increased to 50 elements, the LFEM still produces more than a 5% underestimation in the pile-top rotation. In contrast, NOEM with only a single element accurately simulates the structural response of the single pile under complex soil conditions, achieving relative errors of less than 1%. These results highlight the accuracy and efficiency of the proposed NOEM approach. Furthermore, the displacement profiles along the pile for different LCs with different LFs are examined. As depicted in Figure 7-7, the NOEM accurately capture the pile's structural response across varying magnitudes of applied loads and deformation, consistent with the benchmark results from the LFEM using 100 elements. Thus, the robustness of the proposed method is further validated.

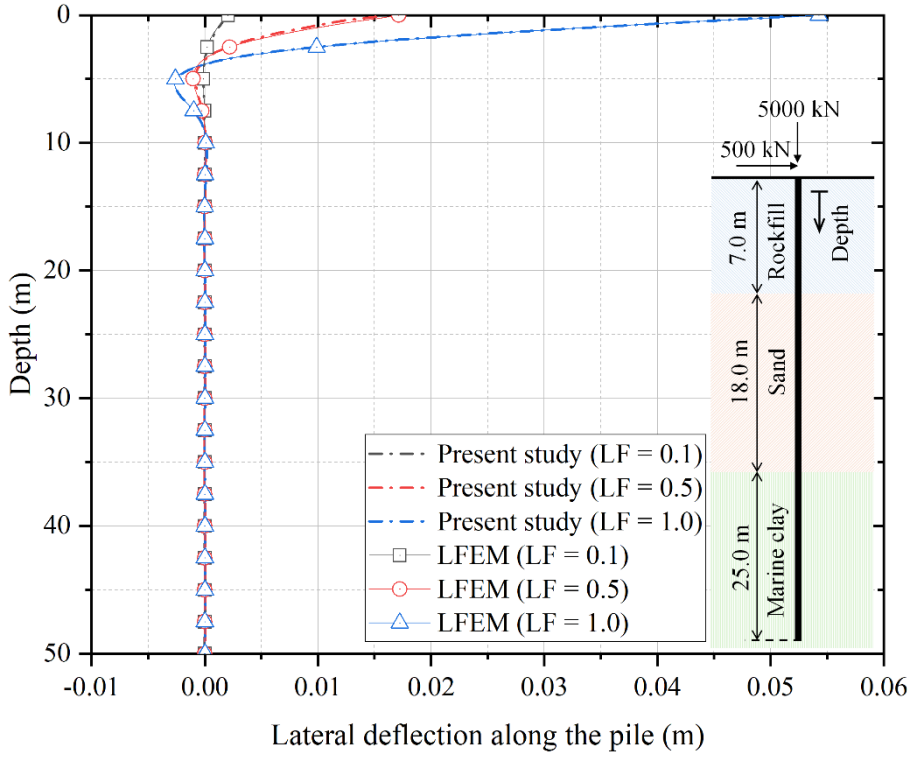


(A)

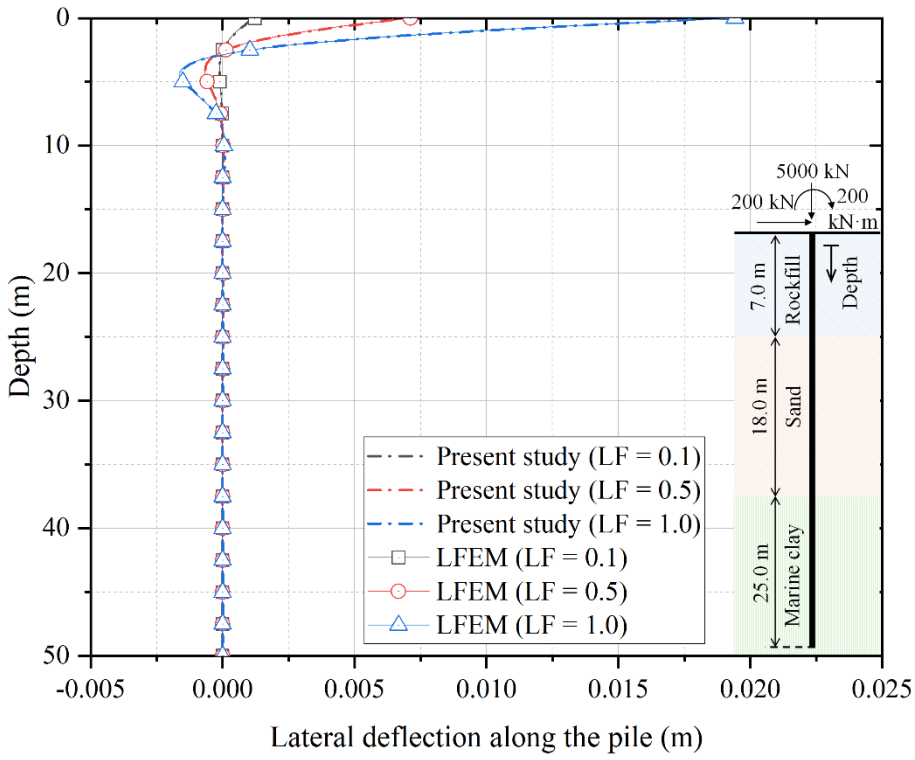


(B)

Figure 7-6 Pile top displacement in different loading conditions:(A) Loading condition 1; (B) Loading condition 2.



(A)



(B) Loading condition 2

Figure 7-7 Pile top displacement in different loading conditions: (A) Loading condition 1; (B) Loading condition 2.

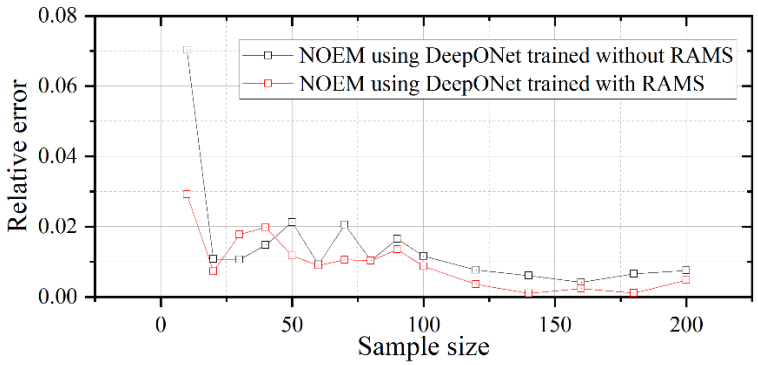


Figure 7-8 Comparison results between NOEM using DeepONet trained with/ without RAMS.

To further validate the effectiveness of the employed RAMS, a comparative study is conducted. Multiple DeepONets are trained with varying training sample sizes,

ranging from 10 to 200 samples, both with and without RAMS. These trained DeepONets are subsequently incorporated into the NOEM for analyzing the single pile under LC 2. The accuracy of the NOEM employing different DeepONet models is evaluated based on the relative error of the lateral displacement at the pile top. As illustrated in Figure 7-8, the implementation of RAMS significantly enhances the accuracy of the NOEM by improving the performance of the trained DeepONets. This demonstrates the effectiveness and necessity of the proposed RAMS approach.

7.3.2 Three-story pile-supported framed structure

In this example, the NOEM is further utilized to analyze the structural response of pile-supported framed structures. The superstructure investigated is a three-story frame, whose loading conditions, geometry, and cross-sectional details are depicted in Figure 7-9. The elastic modulus of the upper frame is set as $E = 32.5\,GPa$. To accurately account for SPI, the base of each frame column is connected to a single pile identical to the one analyzed in Example 1. For the superstructure, each structural member is modeled using four Euler–Bernoulli beam-column elements. For pile foundation modeling, two methods are employed: (1) the conventional LFEM approach using 100 elements per pile, and (2) the NOEM utilizing only one element per pile. The pile-group effect is ignored in this paper.

The comparative performance of the two analysis methods is evaluated through nodal displacements at selected monitoring points. As shown in Figure 7-10 - Figure 7-12, the NOEM provides analysis results nearly identical to those obtained by the LFEM, demonstrating the accuracy of the proposed approach. However, computational efficiency differs significantly between the two methods. The proposed NOEM approach requires approximately 8.24 seconds for computation, whereas the traditional LFEM, employing 100 elements per pile, takes around 17.62 seconds—more than twice the computational cost of the NOEM. Therefore, the superior computational efficiency of the proposed method is also validated.

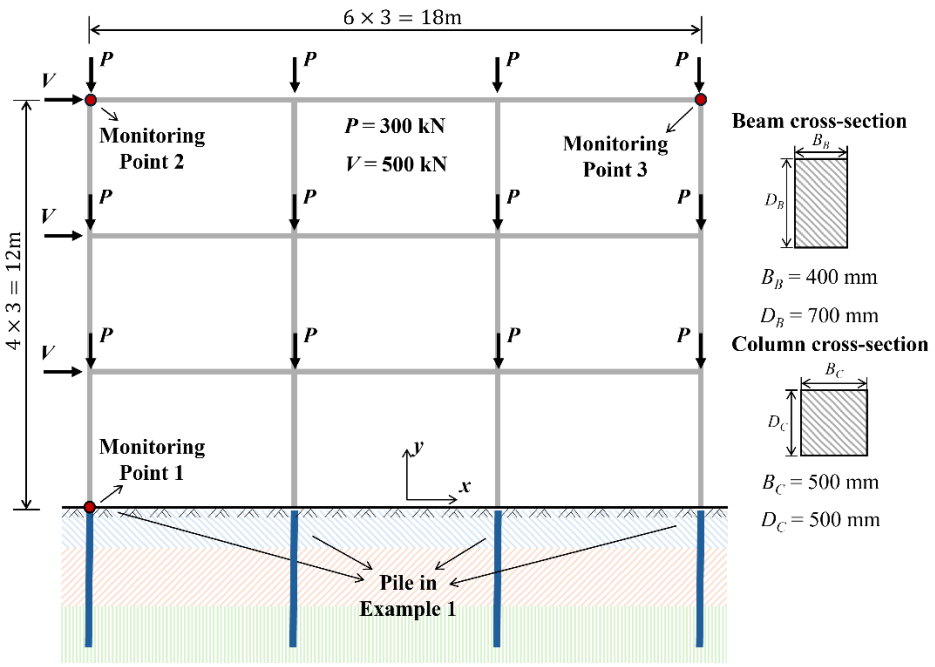


Figure 7-9 Schematics of three-story pile-supported framed structure.

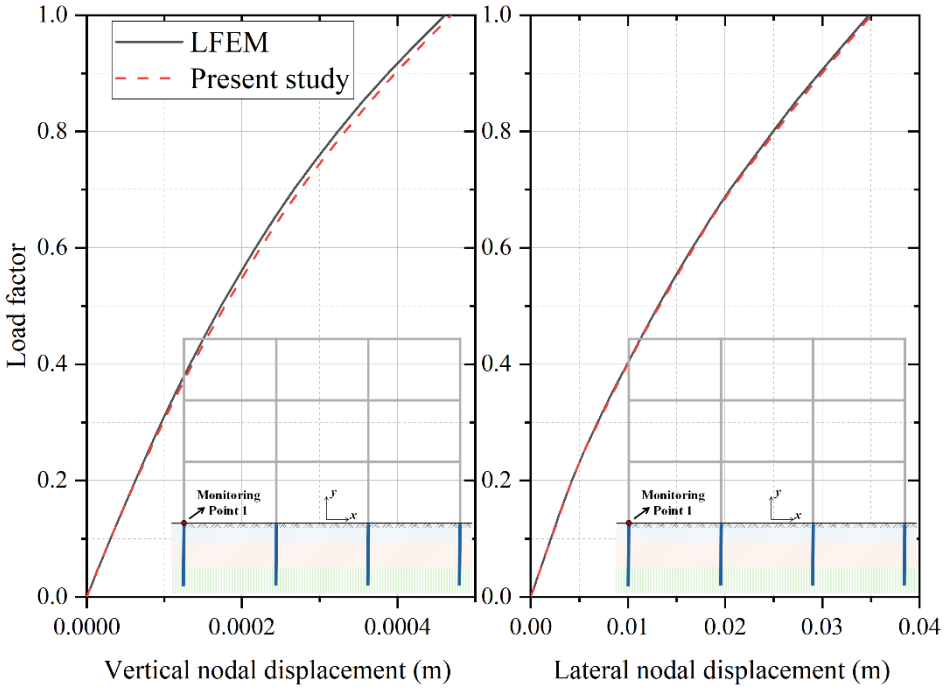


Figure 7-10 Nodal displacement of Monitoring Point 1 on the three-story structure.

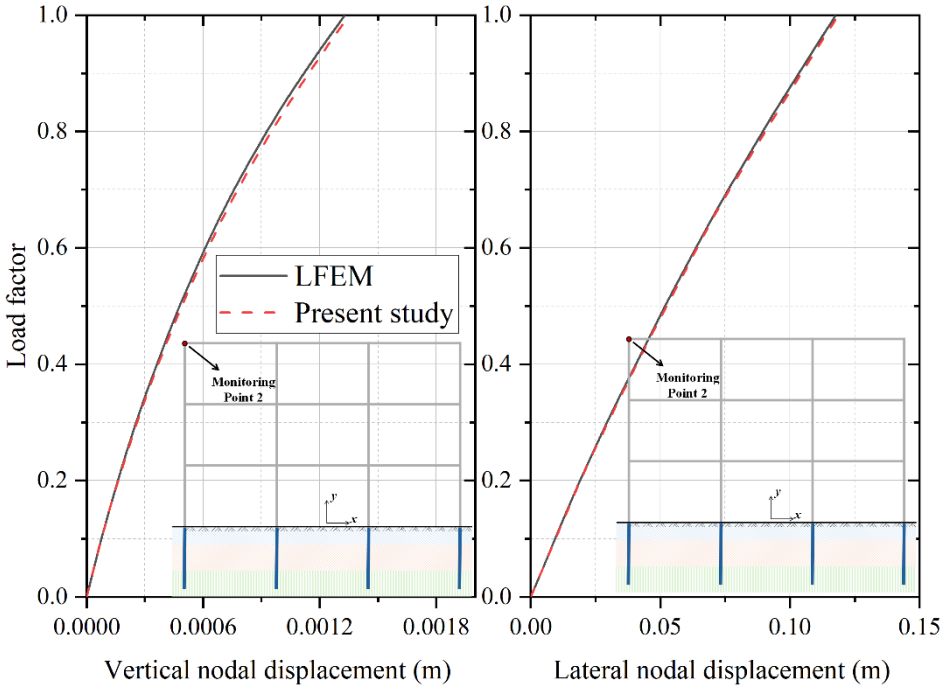


Figure 7-11 Nodal displacement of Monitoring Point 2 on the three-story structure.

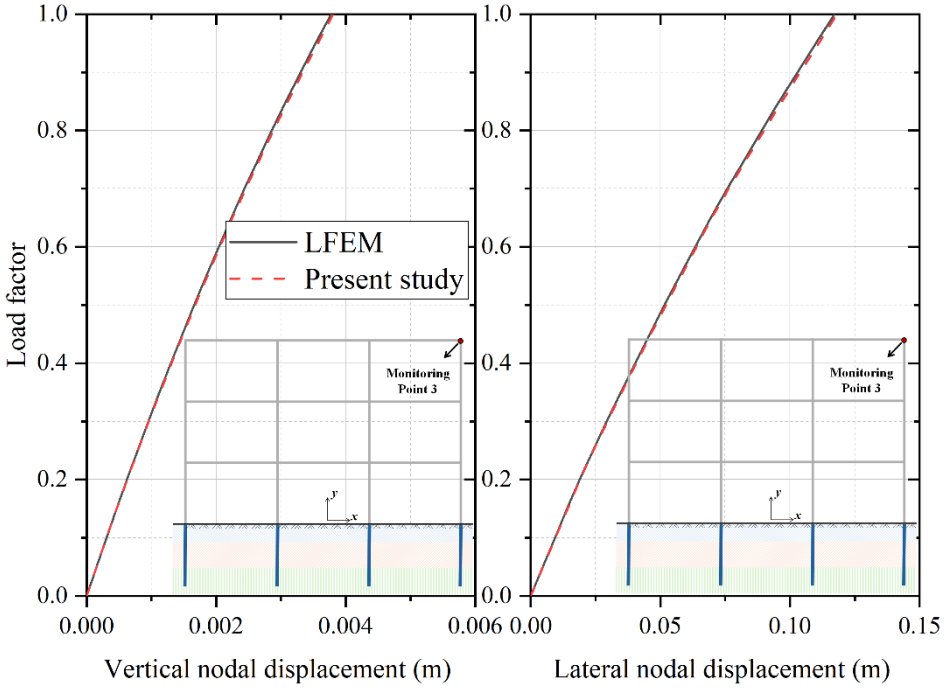


Figure 7-12 Nodal displacement of Monitoring Point 3 on the three-story structure.

7.3.3 Twenty-story pile-supported framed structure

To further illustrate the robustness of the proposed approach, a large-scale pile-

supported building subjected to seismic loading is investigated. The superstructure is a twenty-story building previously examined in [310, 311], with structural member properties given as a cross-sectional area $A = 3.02 \times 10^{-2} m^2$, a moment of inertia I $I = 8.70 \times 10^{-4} m^4$, and an elastic modulus $E = 207\ GPa$. Similar to Example 2, the base of each column is connected to a single pile identical to that analyzed in Example 1. The lateral seismic loading conditions reported in [311] are adopted for analysis. Each member in the superstructure is modeled using four Euler–Bernoulli beam-column elements. Two strategies are employed to model the pile foundations: (1) the LFEM using 100 elements per pile, and (2) the NOEM using a single element per pile.

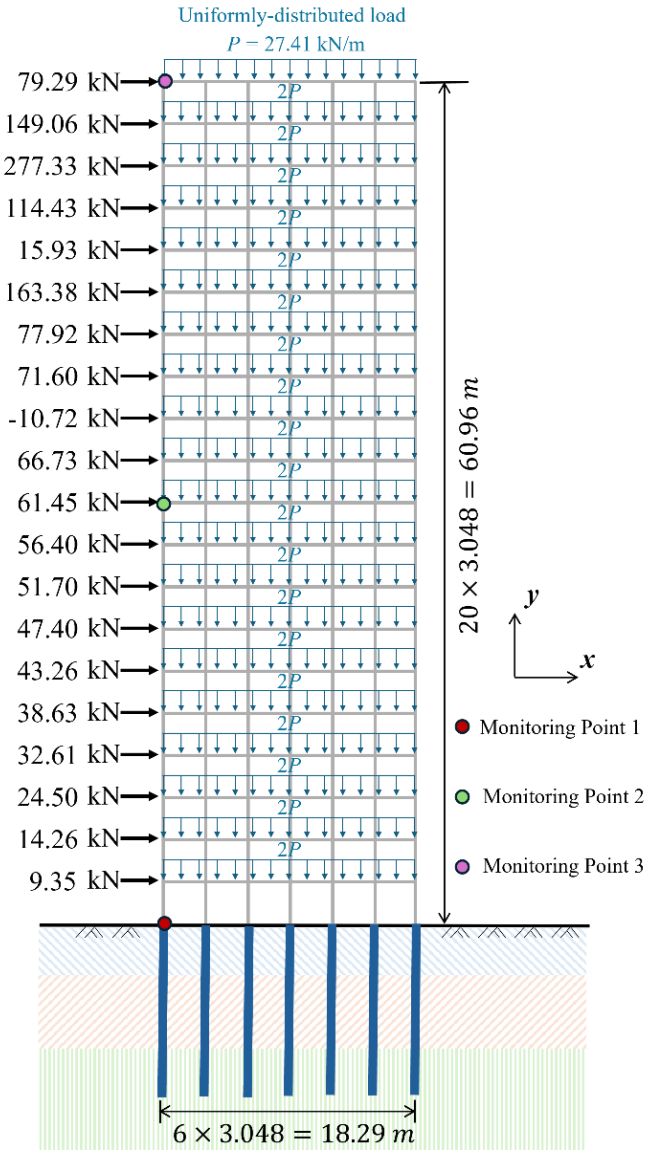


Figure 7-13 Schematics of twenty-story pile-supported framed structure.

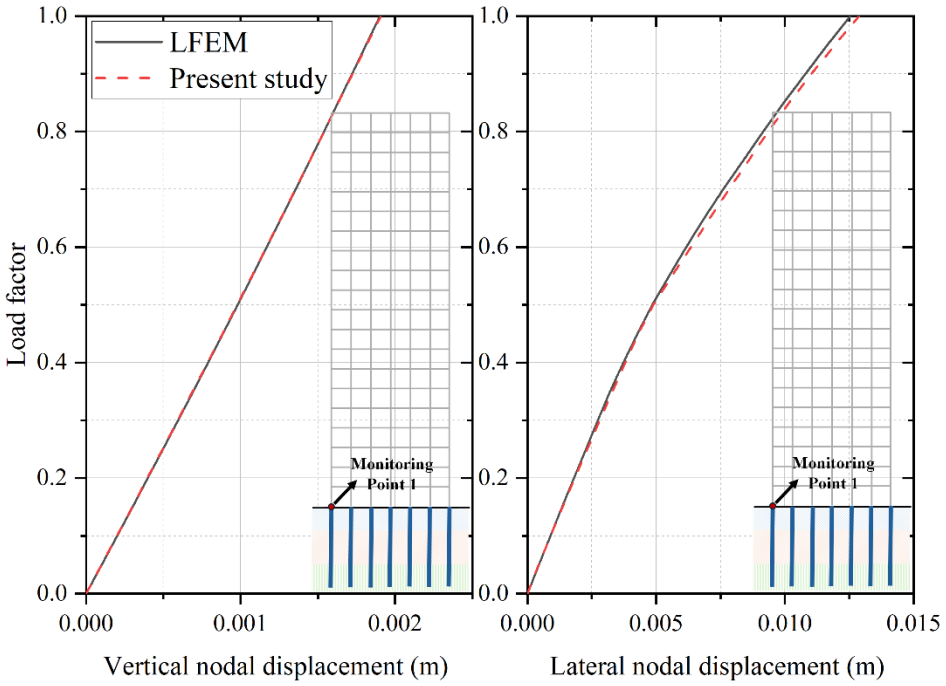


Figure 7-14 Nodal displacement of Monitoring Point 1 on the twenty-story structure.

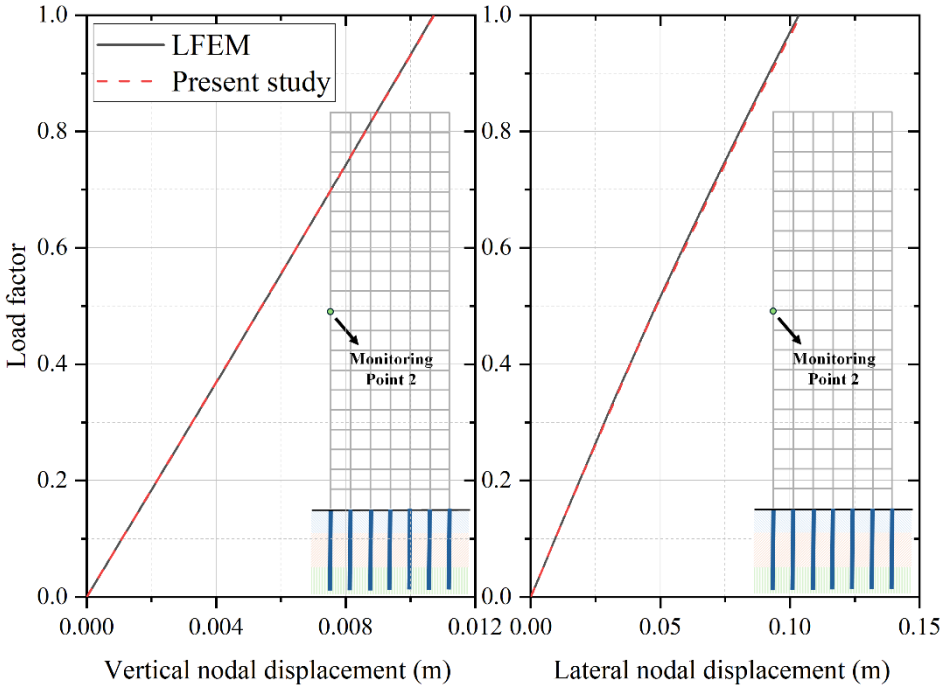


Figure 7-15 Nodal displacement of Monitoring Point 2 on the twenty-story structure.

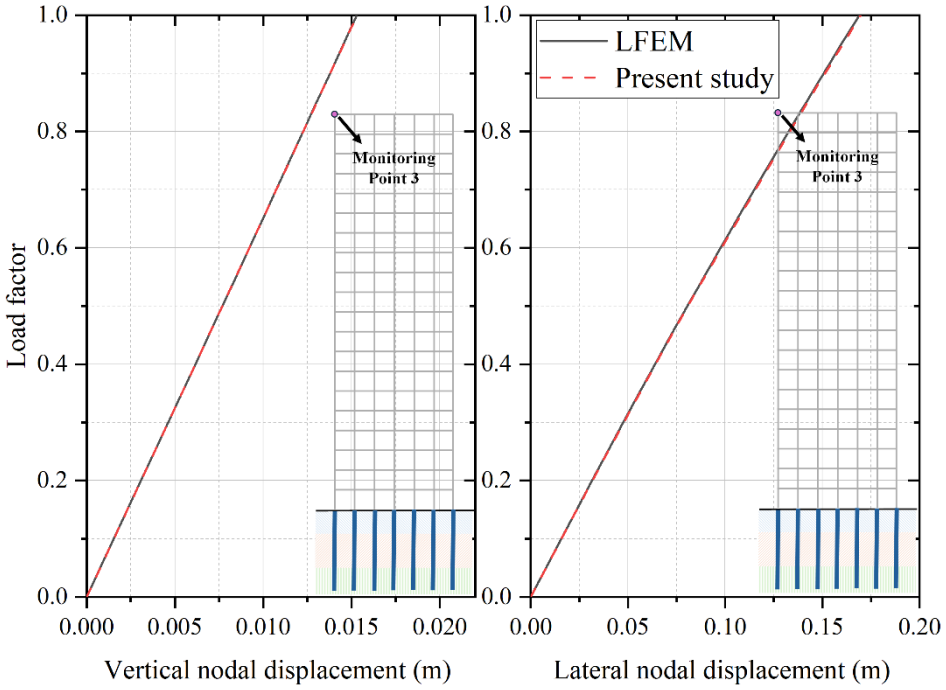


Figure 7-16 Nodal displacement of Monitoring Point 3 on the twenty-story structure

Figure 7-14 - Figure 7-16 present comparative analyses of nodal displacement curves versus the LFs at selected monitoring points. Results demonstrate that the proposed NOEM, utilizing only one element per pile, yields accurate predictions with a maximum relative error below 2%, confirming its accuracy and robustness. Additionally, the computational time of the NOEM is approximately 27.06 seconds, which is nearly half the computational time required by the LFEM (45.43 seconds), highlighting the efficiency of the proposed approach. Furthermore, it is noteworthy that the neural operator used in Examples 2 and 3 is directly adopted from Example 1 with minimal additional training or fine-tuning, showcasing the strong model reusability and considerable application potential of the NOEM.

Chapter 8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

In this thesis, a numerical framework is developed for the second-order analysis of pile-supported framed structures. First, a refined three-dimensional pile element (3DPE) is proposed to accurately capture the complex Soil-Pile Interaction (SPI) in spatial analysis while overcoming the potential numerical instability in the existing methods. Then, a hybrid numerical framework is proposed for general partial differential equation (PDE) problems by integrating the finite element method (FEM) with machine learning (ML) techniques. To enhance the ML training process, a novel sample-augmentation method specifically designed for ML models solving PDE problems is proposed. Subsequently, governing equations for single piles, accounting for nonlinear SPI and geometric nonlinearity due to large deformations, are derived and implemented within the framework of physics-informed neural networks (PINN). Finally, these advancements are synergistically integrated into a practical numerical method that enables accurate modeling of pile-supported framed structures using a simplified approach of one element per pile.

The main findings and contributions of this thesis are summarized as follows:

1) A refined 3DPE formulation is proposed to efficiently capture the nonlinear SPI that is over-simplified in the conventional analysis method. Compared with the existing pile element formulation, the proposed refined 3DPE formulation can more accurately capture the structural behaviors of piles under complex loading conditions, including the shear effect and axial-torsional coupled response. Furthermore, three soil stiffness matrices are proposed to improve the numerical stability of the spatial analysis using the existing element formulation.

2) A hybrid numerical framework that integrates two distinct numerical

methodologies, FEM and ML, is developed for solving general PDE problems. In the proposed neural-operator element method (NOEM), neural operators (NOs) are trained to model specific subdomains rather than entire PDE systems, resulting in a more efficient training process compared to conventional ML methods. The pre-trained NOs are subsequently used as neural-operator elements (NOEs) to represent subdomains that typically require dense meshing in traditional FEM, while standard finite elements represent the remaining regions. This hybrid approach substantially reduces computational costs and greatly enhances model reusability, as pre-trained NOs can be readily applied across various PDE systems.

3) To improve performance of ML models trained with limited samples, a novel sample-augmentation technique, termed Residual-Based Adversarial-Gradient Moving Sample (RAMS) Method, is proposed. RAMS is specifically tailored for ML-based PDE solutions and demonstrates superior performance compared to existing sampling methods. Its remarkable versatility and effectiveness are validated across diverse ML approaches, including PINN, physics-informed operator learning, and data-driven operator learning. Notably, RAMS is the first sampling method capable of effectively supporting operator-learning problems.

4) Governing equations for single piles are derived, incorporating both geometric nonlinearity due to large pile deformation and nonlinear Soil-Pile Interaction. These equations are then implemented within the PINN framework, enabling precise simulation and analysis of single pile behavior.

5) A practical numerical method for the second-order analysis of pile-supported framed structures considering complex SPI, is established based on the NOEM framework. By integrating the derived governing equations and the proposed RAMS sampling approach, the NOEM efficiently models complex structural behaviors of individual piles, capturing geometric nonlinearity and nonlinear

SPI accurately with only a single element per pile. This efficient method significantly facilitates the practical application of the second-order analysis in the design and evaluation of pile-supported framed structures.

8.2 Recommendations for Future Work

This thesis develops a hybrid numerical method combining FEM and ML and implements it for the second-order analysis method of pile-supported framed structures. Some studies, however, still need to be conducted in the future:

i. Extend the NOEM to time-dependent problems: Structural dynamic analysis is crucial in engineering applications. Extending the current NOEM framework from static analysis to dynamic, time-dependent problems is essential for broader applicability.

ii. Apply the NOEM for inelastic analysis: Real-world pile materials sometimes exhibit nonlinear characteristics. Future studies should incorporate material plasticity into the NOEM to capture these nonlinear behaviors accurately.

iii. Consider spatial Soil-Pile Interaction: SPI in three-dimensional spaces involves more complex nonlinear interactions compared to two-dimensional models. Future work should focus on addressing these spatial effects comprehensively in the NOEM.

iv. Develop a NO library for various geological conditions: the reusability of NOs in the NOEM framework suggests they can be directly applied across different systems without additional training. Establishing a comprehensive library of NOs for piles under various geological conditions would significantly enhance the practical application and versatility of the NOEM method for second-order analyses.

v. The proposed NOEM method will be employed to more realistic pile-supported

structures for further validation.

REFERENCES

[1] Bathe K-J. Finite element procedures: Klaus-Jurgen Bathe; 2006.

[2] Johnson C. Numerical solution of partial differential equations by the finite element method: Courier Corporation; 2009.

[3] Reddy JN. An introduction to the finite element method. New York. 1993;27.

[4] Bochev PB, Gunzburger MD. Least-squares finite element methods: Springer Science & Business Media; 2009.

[5] Bochev PB, Gunzburger MD. Analysis of least squares finite element methods for the Stokes equations. Mathematics of Computation. 1994;63:479-506.

[6] Keith B, Petrides S, Fuentes F, Demkowicz L. Discrete least-squares finite element methods. Computer Methods in Applied Mechanics and Engineering. 2017;327:226-55.

[7] Abdelaziz Y, Hamouine A. A survey of the extended finite element. Computers & structures. 2008;86:1141-51.

[8] Fries TP, Belytschko T. The extended/generalized finite element method: an overview of the method and its applications. International journal for numerical methods in engineering. 2010;84:253-304.

[9] Cockburn B. Discontinuous galerkin methods. ZAMM - Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics. 2003;83:731-54.

[10] Cockburn B, Karniadakis GE, Shu C-W. Discontinuous Galerkin methods: theory,

computation and applications: Springer Science & Business Media; 2012.

[11] Chan S-L, Chui P-T. Non-linear static and cyclic analysis of steel frames with semi-rigid connections: Elsevier; 2000.

[12] McGuire W, Gallagher RH, Ziemian RD. Matrix structural analysis2000.

[13] Bucalem M, Bathe K-J. Finite element analysis of shell structures. Archives of Computational Methods in Engineering. 1997;4:3-61.

[14] Chapelle D, Bathe K-J. The finite element analysis of shells-fundamentals: Springer Science & Business Media; 2010.

[15] Zienkiewicz OC, Taylor RL. The finite element method for solid and structural mechanics: Elsevier; 2005.

[16] Song Y-C, Wang R-P, Li J. Local and post-local buckling behavior of welded steel shapes in partially encased composite columns. Thin-Walled Structures. 2016;108:93-108.

[17] Chau KT, Shen C, Guo X. Nonlinear seismic soil–pile–structure interactions: shaking table tests and FEM analyses. Soil Dynamics and Earthquake Engineering. 2009;29:300-10.

[18] Said Id, De Gennaro V, Frank R. Axisymmetric finite element analysis of pile loading tests. Computers and Geotechnics. 2009;36:6-19.

[19] Connor Jr JJ, Logcher RD, Chan S-C. Nonlinear analysis of elastic framed structures. Journal of the Structural Division. 1968;94:1525-47.

[20] Bathe KJ, Bolourchi S. Large displacement analysis of three-dimensional beam structures. International journal for numerical methods in engineering.

1979;14:961-86.

[21] So AKW, Chan SL. Buckling and geometrically nonlinear analysis of frames using one element/member. Journal of Constructional Steel Research. 1991;20:271-89.

[22] White DW, Hajjar JF. Application of second-order elastic analysis in LRFD: research to practice. Engineering Journal. 1991;28:133-48.

[23] Oran C. Tangent stiffness in space frames. Journal of the Structural Division. 1973;99:987-1001.

[24] Lui E, Chen W. Steel frame analysis with flexible joints. Journal of Constructional Steel Research. 1987;8:161-202.

[25] Liew JR, Chen H, Shanmugam N. Stability functions for second-order inelastic analysis of space frames. Light-Weight Steel and Aluminium Structures, Proc, 4th Int Conf on Steel and Aluminium Structures1999. p. 19-26.

[26] Chan S-L, Gu J-X. Exact tangent stiffness for imperfect beam-column members. Journal of Structural Engineering. 2000;126:1094-102.

[27] Feng D-C, Wu J-Y. Improved displacement-based Timoshenko beam element with enhanced strains. Journal of Structural Engineering. 2020;146:04019221.

[28] Izzuddin B, Smith DL. Large-displacement analysis of elastoplastic thin-walled frames. I: Formulation and implementation. Journal of structural Engineering. 1996;122:905-14.

[29] Neuenhofer A, Filippou FC. Evaluation of nonlinear frame finite-element models. Journal of structural engineering. 1997;123:958-66.

[30] Zhang Y, Tien I. Methodology for regularization of force-based elements to model

reinforced concrete columns with short lap splices. Journal of Engineering Mechanics. 2020;146:04020073.

[31] Argyris J, Dunne P, Scharpf D. On large displacement-small strain analysis of structures with rotational degrees of freedom. Computer Methods in Applied Mechanics and Engineering. 1978;14:401-51.

[32] Chan SL, Kitipornchai S. Geometric nonlinear analysis of asymmetric thin-walled beam-columns. Engineering Structures. 1987;9:243-54.

[33] Chan SL. Large deflection kinematic formulations for three-dimensional framed structures. Computer Methods in Applied Mechanics and Engineering. 1992;95:17-36.

[34] Oran C, Kassimali A. Large deformations of framed structures under static and dynamic loads. Computers & structures. 1976;6:539-47.

[35] Meek J, Tan HS. Geometrically nonlinear analysis of space frames by an incremental iterative technique. Computer methods in applied mechanics and engineering. 1984;47:261-82.

[36] Crisfield M. Accelerating and damping the modified Newton-Raphson method. Computers & structures. 1984;18:395-407.

[37] Crisfield M. A faster modified Newton-Raphson iteration. Computer methods in applied mechanics and engineering. 1979;20:267-78.

[38] Batoz JL, Dhatt G. Incremental displacement algorithms for nonlinear problems. International Journal for Numerical Methods in Engineering. 1979;14:1262-7.

[39] Riks E. An incremental approach to the solution of snapping and buckling

problems. International journal of solids and structures. 1979;15:529-51.

[40] Crisfield MA. A fast incremental/iterative solution procedure that handles “snap-through”. Computational methods in nonlinear structural and solid mechanics: Elsevier; 1981. p. 55-62.

[41] Crisfield M. Variable step-lengths for non-linear structural analysis. 1982.

[42] Crisfield M. An arc - length method including line searches and accelerations. International journal for numerical methods in engineering. 1983;19:1269-89.

[43] Ritto-Corrêa M, Camotim D. On the arc-length and other quadratic control methods: Established, less known and new implementation procedures. Computers & Structures. 2008;86:1353-68.

[44] Rezaiee-Pajand M, Ghalishooyan M, Salehi-Ahmadabad M. Comprehensive evaluation of structural geometrical nonlinear solution techniques Part II: Comparing efficiencies of the methods. Structural Engineering and Mechanics. 2013;48:879-914.

[45] Schweizerhof K, Wriggers P. Consistent linearization for path following methods in nonlinear FE analysis. Computer Methods in Applied Mechanics and Engineering. 1986;59:261-79.

[46] Chan SL. Geometric and material non - linear analysis of beam - columns and frames using the minimum residual displacement method. International journal for numerical methods in engineering. 1988;26:2657-69.

[47] Ahmad-Abad MS, Maghami A, Ghalishooyan M, Shooshtari A. A family of minimum residual displacement methods as nonlinear solution schemes for equilibrium path-following in structural mechanics. Computers & Structures.

2024;300:107407.

[48] Standardization ECf. Eurocode 3: Design of steel structures. 2005.

[49] Construction AIoS. Resistance factor design specification for structural steel buildings. American Institute of Steel Construction: Chicago, IL, USA2011.

[50] Department HKSGB. Code of practice for structural uses of steel. 2011.

[51] Chan S-L, Liu Y-P, Liu S-W. A new codified design theory of second-order direct analysis for steel and composite structures–From research to practice. Structures: Elsevier; 2017. p. 105-11.

[52] Chan SL, Zhou ZH. Pointwise equilibrating polynomial element for nonlinear analysis of frames. Journal of structural engineering. 1994;120:1703-17.

[53] Chan SL, Zhou ZH. Second-order elastic analysis of frames using single imperfect element per member. Journal of Structural Engineering. 1995;121:939-45.

[54] Liew JR, Chen H, Shanmugam N, Chen W. Improved nonlinear plastic hinge analysis of space frame structures. Engineering structures. 2000;22:1324-38.

[55] Liew JR, White D, Chen W-F. Second-order refined plastic-hinge analysis for frame design. Part I. Journal of Structural Engineering. 1993;119:3196-216.

[56] White DW. Plastic-hinge methods for advanced analysis of steel frames. Journal of Constructional Steel Research. 1993;24:121-52.

[57] Ziemian RD, McGuire W. Modified tangent modulus approach, a contribution to plastic hinge analysis. Journal of Structural Engineering. 2002;128:1301-7.

[58] Chen W, Chan SL. Second-order inelastic analysis of steel frames using element

with midspan and end springs. Journal of Structural Engineering. 1995;121:530-41.

[59] Zhou Z-H, Chan S-L. Elastoplastic and large deflection analysis of steel frames by one element per member. I: One hinge along member. Journal of Structural Engineering. 2004;130:538-44.

[60] Liu S-W, Liu Y-P, Chan S-L. Direct analysis by an arbitrarily-located-plastic-hinge element—Part 1: Planar analysis. Journal of Constructional Steel Research. 2014;103:303-15.

[61] Liu S-W, Liu Y-P, Chan S-L. Direct analysis by an arbitrarily-located-plastic-hinge element—Part 2: Spatial analysis. Journal of Constructional Steel Research. 2014;103:316-26.

[62] Battini J-M, Pacoste C. Co-rotational beam elements with warping effects in instability problems. Computer Methods in Applied Mechanics and Engineering. 2002;191:1755-89.

[63] Chen L, Liu S-W, Bai R, Chan S-L. Co-rotational formulations for geometrically nonlinear analysis of beam-columns including warping and Wagner effects. International Journal of Structural Stability and Dynamics. 2023;23:2350052.

[64] Sapountzakis E, Dikaros I. Advanced 3D beam element of arbitrary composite cross section including generalized warping effects. International Journal for Numerical Methods in Engineering. 2015;102:44-78.

[65] Liu S-W, Ziemian RD, Chen L, Chan S-L. Bifurcation and large-deflection analyses of thin-walled beam-columns with non-symmetric open-sections. Thin-walled structures. 2018;132:287-301.

[66] Chen L, Zhang H-Y, Liu S-W, Ziemian RD. Efficient line-element method for the second-order analysis of steel members with nonsymmetric thick-walled cross sections. Journal of Structural Engineering. 2024;150:04023226.

[67] Liu S-W, Ziemian R. MSASect2 - Matrix Structural Analysis Software for Arbitrary Cross-sections. 2023.

[68] Liu S-W, Gao W-L, Ziemian RD. Improved line-element formulations for the stability analysis of arbitrarily-shaped open-section beam-columns. Thin-Walled Structures. 2019;144:106290.

[69] Ziemian RD. Guide to stability design criteria for metal structures: John Wiley & Sons; 2010.

[70] Liew J, Chen W. Implications of using refined plastic hinge analysis for load and resistance factor design. Thin-walled structures. 1994;20:17-47.

[71] Liew JR, Chen W, Chen H. Advanced inelastic analysis of frame structures. Journal of Constructional Steel Research. 2000;55:245-65.

[72] White DW, Hajjar JF. Design of steel frames without consideration of effective length. Engineering Structures. 1997;19:797-810.

[73] Yang Y-B, Kuo S-R. Theory & analysis of nonlinear framed structures. (No Title). 1994.

[74] Albermani F, Kitipornchai S. Numerical simulation of structural behaviour of transmission towers. Thin-walled structures. 2003;41:167-77.

[75] Chan SL, Cho S. Second-order P-Δ-δ analysis and design of angle trusses allowing for imperfections and semi-rigid connections. International Journal of Advanced

Steel Construction. 2005;1:157-72.

[76] Liu S-W, Bai R, Chan S-L, Liu Y-P. Second-order direct analysis of domelike structures consisting of tapered members with I-sections. Journal of Structural Engineering. 2016;142:04016009.

[77] Zhang ZJ, Duraisamy K. Machine learning methods for data-driven turbulence modeling. 22nd AIAA computational fluid dynamics conference2015. p. 2460.

[78] Tseranidis S, Brown NC, Mueller CT. Data-driven approximation algorithms for rapid performance evaluation and optimization of civil structures. Automation in Construction. 2016;72:279-93.

[79] Asher MJ, Croke BF, Jakeman AJ, Peeters LJ. A review of surrogate models and their application to groundwater modeling. Water Resources Research. 2015;51:5957-73.

[80] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics. 2019;378:686-707.

[81] Sun L, Wang J-X. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. Theoretical and Applied Mechanics Letters. 2020;10:161-9.

[82] Yang L, Meng X, Karniadakis GE. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. Journal of Computational Physics. 2021;425:109913.

[83] Liu X, Yao W, Peng W, Zhou W. Bayesian physics-informed extreme learning

machine for forward and inverse PDE problems with noisy data. Neurocomputing. 2023;549:126425.

[84] Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence. 2019;1:206-15.

[85] Pinkus A. Approximation theory of the MLP model in neural networks. Acta numerica. 1999;8:143-95.

[86] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nature Reviews Physics. 2021;3:422-40.

[87] Wang S, Teng Y, Perdikaris P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing. 2021;43:A3055-A81.

[88] Jagtap AD, Kawaguchi K, Karniadakis GE. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics. 2020;404:109136.

[89] Jagtap AD, Mitsotakis D, Karniadakis GE. Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre–Green–Naghdi equations. Ocean Engineering. 2022;248:110775.

[90] Salimans T, Kingma DP. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. Advances in neural information processing systems. 2016;29.

[91] Wang S, Wang H, Seidman JH, Perdikaris P. Random weight factorization improves the training of continuous neural representations. arXiv preprint

arXiv:221001274. 2022.

[92] O'shea K, Nash R. An introduction to convolutional neural networks. arXiv preprint arXiv:151108458. 2015.

[93] Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems. 2021;33:6999-7019.

[94] Gao H, Sun L, Wang J-X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. Journal of Computational Physics. 2021;428:110079.

[95] Ren P, Rao C, Liu Y, Wang J-X, Sun H. PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. Computer Methods in Applied Mechanics and Engineering. 2022;389:114399.

[96] Zhang R, Liu Y, Sun H. Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. Computer Methods in Applied Mechanics and Engineering. 2020;369:113226.

[97] Chen Y, Rao M, Feng K, Zuo MJ. Physics-Informed LSTM hyperparameters selection for gearbox fault detection. Mechanical Systems and Signal Processing. 2022;171:108907.

[98] Yang X, Tartakovsky G, Tartakovsky A. Physics-informed kriging: A physics-informed Gaussian process regression method for data-model convergence. arXiv preprint arXiv:180903461. 2018.

[99] Pang G, Karniadakis GE. Physics-informed learning machines for partial differential equations: Gaussian processes versus neural networks. Emerging

frontiers in nonlinear science. 2020:323-43.

[100] Nevin JW, Vaquero-Caballero F, Ives DJ, Savory SJ. Physics-informed Gaussian process regression for optical fiber communication systems. Journal of Lightwave Technology. 2021;39:6833-44.

[101] Wu J-L, Michelén-Ströfer C, Xiao H. Physics-informed covariance kernel for model-form uncertainty quantification with application to turbulent flows. Computers & Fluids. 2019;193:104292.

[102] Wang Z, Xing W, Kirby R, Zhe S. Physics informed deep kernel learning. International Conference on Artificial Intelligence and Statistics: PMLR; 2022. p. 1206-18.

[103] Braun J, Griebel M. On a constructive proof of Kolmogorov's superposition theorem. Constructive approximation. 2009;30:653-75.

[104] Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljačić M et al. Kan: Kolmogorov-arnold networks. arXiv preprint arXiv:240419756. 2024.

[105] Yu R, Yu W, Wang X. Kan or mlp: A fairer comparison. arXiv preprint arXiv:240716674. 2024.

[106] Shukla K, Toscano JD, Wang Z, Zou Z, Karniadakis GE. A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks. Computer Methods in Applied Mechanics and Engineering. 2024;431:117290.

[107] Toscano JD, Oommen V, Varghese AJ, Zou Z, Ahmadi Daryakenari N, Wu C et al. From pinns to pikans: Recent advances in physics-informed machine learning. Machine Learning for Computational Science and Engineering. 2025;1:1-43.

[108] Dong S, Ni N. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. Journal of Computational Physics. 2021;435:110242.

[109] Zeinhofer M, Masri R, Mardal KA. A unified framework for the error analysis of physics-informed neural networks. IMA Journal of Numerical Analysis. 2024:drae081.

[110] Lu L, Pestourie R, Yao W, Wang Z, Verdugo F, Johnson SG. Physics-informed neural networks with hard constraints for inverse design. SIAM Journal on Scientific Computing. 2021;43:B1105-B32.

[111] Ouyang W, Ouyang W, Tian X, Liu S-W, Wang J, Chan S-L. Nonlinear analysis of glass panes under complex lateral pressures through physics informed neural networks. Engineering Structures. 2025;334:120262.

[112] Wang S, Sankaran S, Perdikaris P. Respecting causality for training physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering. 2024;421:116813.

[113] Cao Y, Fang Z, Wu Y, Zhou D-X, Gu Q. Towards understanding the spectral bias of deep learning. arXiv preprint arXiv:191201198. 2019.

[114] Wang S, Wang H, Perdikaris P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering. 2021;384:113938.

[115] Wang B, Zhang W, Cai W. Multi-scale deep neural network (MscaleDNN) methods for oscillatory stokes flows in complex domains. arXiv preprint arXiv:200912729. 2020.

[116] Liu Z, Cai W, Xu Z-QJ. Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. arXiv preprint arXiv:200711207. 2020.

[117] Liu L, Cai W. Linearized learning with multiscale deep neural networks for stationary Navier-Stokes equations with oscillatory solutions. East Asian Journal on Applied Mathematics. 2022;13.

[118] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z et al. Automatic differentiation in pytorch. 2017.

[119] Pang G, Lu L, Karniadakis GE. fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing. 2019;41:A2603-A26.

[120] Yuan L, Ni Y-Q, Deng X-Y, Hao S. A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. Journal of Computational Physics. 2022;462:111260.

[121] Niaki SA, Haghighat E, Campbell T, Poursartip A, Vaziri R. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. Computer Methods in Applied Mechanics and Engineering. 2021;384:113959.

[122] McClenny LD, Braga-Neto UM. Self-adaptive physics-informed neural networks. Journal of Computational Physics. 2023;474:111722.

[123] Wang S, Yu X, Perdikaris P. When and why PINNs fail to train: A neural tangent kernel perspective. Journal of Computational Physics. 2022;449:110768.

[124] Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: A deep learning library for solving differential equations. SIAM review. 2021;63:208-28.

[125] Wu C, Zhu M, Tan Q, Kartha Y, Lu L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering. 2023;403:115671.

[126] Daw A, Bu J, Wang S, Perdikaris P, Karpatne A. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. arXiv preprint arXiv:220702338. 2022.

[127] Tokdar ST, Kass RE. Importance sampling: a review. Wiley Interdisciplinary Reviews: Computational Statistics. 2010;2:54-60.

[128] Nabian MA, Gladstone RJ, Meidani H. Efficient training of physics-informed neural networks via importance sampling. Computer - Aided Civil and Infrastructure Engineering. 2021;36:962-77.

[129] Zhang Z, Li J, Liu B. Annealed adaptive importance sampling method in PINNs for solving high dimensional partial differential equations. Journal of Computational Physics. 2025;521:113561.

[130] Tang K, Wan X, Yang C. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. Journal of Computational Physics. 2023;476:111868.

[131] Wang X, Tang K, Zhai J, Wan X, Yang C. Deep adaptive sampling for surrogate modeling without labeled data. Journal of Scientific Computing. 2024;101:1-33.

[132] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature machine intelligence. 2021;3:218-29.

[133] Chen T, Chen H. Universal approximation to nonlinear operators by neural

networks with arbitrary activation functions and its application to dynamical systems. IEEE transactions on neural networks. 1995;6:911-7.

[134] Wang S, Wang H, Perdikaris P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science advances. 2021;7:eabi8605.

[135] Yin M, Charon N, Brody R, Lu L, Trayanova N, Maggioni M. Dimon: Learning solution operators of partial differential equations on a diffeomorphic family of domains. arXiv preprint arXiv:240207250. 2024.

[136] Mandl L, Goswami S, Lambers L, Ricken T. Separable deeponet: Breaking the curse of dimensionality in physics-informed machine learning. arXiv preprint arXiv:240715887. 2024.

[137] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A et al. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:201008895. 2020.

[138] Tran A, Mathews A, Xie L, Ong CS. Factorized fourier neural operators. arXiv preprint arXiv:211113802. 2021.

[139] Qi K, Sun J. Gabor-filtered fourier neural operator for solving partial differential equations. Computers & Fluids. 2024;274:106239.

[140] Cao Q, Goswami S, Karniadakis GE. Laplace neural operator for solving differential equations. Nature Machine Intelligence. 2024;6:631-40.

[141] Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Computer Methods in Applied Mechanics and Engineering.

2022;393:114778.

[142] Qin S, Lyu F, Peng W, Geng D, Wang J, Gao N et al. Toward a better understanding of fourier neural operators: Analysis and improvement from a spectral perspective. arXiv e-prints. 2024:arXiv: 2404.07200.

[143] Lee JE, Zhu M, Xi Z, Wang K, Yuan YO, Lu L. Efficient and generalizable nested Fourier-DeepONet for three-dimensional geological carbon sequestration. Engineering Applications of Computational Fluid Mechanics. 2024;18:2435457.

[144] Zhu M, Feng S, Lin Y, Lu L. Fourier-DeepONet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness. Computer Methods in Applied Mechanics and Engineering. 2023;416:116300.

[145] Pickering E, Guth S, Karniadakis GE, Sapsis TP. Discovering and forecasting extreme events via active learning in neural operators. Nature Computational Science. 2022;2:823-33.

[146] Liu Y, Sun J, He X, Pinney G, Zhang Z, Schaeffer H. Prose-fd: A multimodal pde foundation model for learning multiple operators for forecasting fluid dynamics. arXiv preprint arXiv:240909811. 2024.

[147] Sun J, Liu Y, Zhang Z, Schaeffer H. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. Physical Review E. 2025;111:035304.

[148] AISC. Specification for structural steel buildings. 2016.

[149] Standardization ECf. Eurocode 3: Design of steel structures. 2006.

[150] Chen L, Abdelrahman A, Liu S-W, Ziemian RD, Chan S-L. Gaussian Beam–Column Element Formulation for Large-Deflection Analysis of Steel Members with Open Sections Subjected to Torsion. Journal of Structural Engineering. 2021;147:04021206.

[151] Tang Y-Q, Liu Y-P, Chan S-L, Du E-F. An innovative co-rotational pointwise equilibrating polynomial element based on Timoshenko beam theory for second-order analysis. Thin-Walled Structures. 2019;141:15-27.

[152] Chen W-F. Structural stability: from theory to practice. Engineering Structures. 2000;22:116-22.

[153] Chen W-F, Goto Y, Liew JR. Stability design of semi-rigid frames: John Wiley & Sons; 1995.

[154] Agency FEM. A Practical Guide to Soil-Structure Interaction. 2020.

[155] API. Recommended practice for planning, designing, and constructing fixed offshore platforms. RP2A-WSD: American Petroleum Institute; 2011.

[156] Choo YW, Kim D. Experimental development of the p-y relationship for large-diameter offshore monopiles in sands: Centrifuge tests. Journal of Geotechnical and Geoenvironmental Engineering. 2016;142:04015058.

[157] Kraft Jr LM, Ray RP, Kagawa T. Theoretical t-z curves. Journal of the Geotechnical Engineering Division. 1981;107:1543-61.

[158] Matlock H. Correlation for design of laterally loaded piles in soft clay. Offshore technology conference: OnePetro; 1970.

[159] Bateman AH, Crispin JJ, Vardanega PJ, Mylonakis GE. Theoretical t-z Curves

for Axially Loaded Piles. Journal of Geotechnical and Geoenvironmental Engineering. 2022;148:04022052.

[160] Liu X, Cai G, Liu L, Liu S, Duan W, Puppala AJ. Improved py curve models for large diameter and super-long cast-in-place piles using piezocone penetration test data. Computers and Geotechnics. 2021;130:103911.

[161] Zhang X, Tang L, Ling X, Chan A. Critical buckling load of pile in liquefied soil. Soil Dynamics and Earthquake Engineering. 2020;135:106197.

[162] Wang S, Orense RP. Modelling of raked pile foundations in liquefiable ground. Soil Dynamics and Earthquake Engineering. 2014;64:11-23.

[163] Limkatanyu S, Kuntiyawichai K, Spacone E. Response of reinforced concrete piles including soil–pile interaction effects. Engineering Structures. 2009;31:1976-86.

[164] Dash SR, Bhattacharya S, Blakeborough A. Bending–buckling interaction as a failure mechanism of piles in liquefiable soils. Soil Dynamics and Earthquake Engineering. 2010;30:32-9.

[165] Lombardi D, Bhattacharya S. Evaluation of seismic performance of pile - supported models in liquefiable soils. Earthquake engineering & structural dynamics. 2016;45:1019-38.

[166] Mehra S, Trivedi A. Experimental studies on model single pile and pile groups subjected to torque. Proceedings of China-Europe Conference on Geotechnical Engineering: Volume 2: Springer; 2018. p. 997-1000.

[167] Zhang L, Tsang CY. Three-dimensional analysis of torsionally loaded large-diameter bored pile groups. Tall Buildings: From Engineering to Sustainability:

World Scientific; 2005. p. 311-7.

[168] Nghiem HM, Chang N-Y. Efficient solution for a single pile under torsion. Soils and Foundations. 2019;59:13-26.

[169] Kong L, Zhang L. Experimental study of interaction and coupling effects in pile groups subjected to torsion. Canadian Geotechnical Journal. 2008;45:1006-17.

[170] Cai Y, Gould P, Desai C. Nonlinear analysis of 3D seismic interaction of soil–pile–structure systems and application. Engineering Structures. 2000;22:191-9.

[171] Kong L, Zhang L. Centrifuge modeling of torsionally loaded pile groups. Journal of geotechnical and geoenvironmental engineering. 2007;133:1374-84.

[172] Rajeswari J, Sarkar R. Seismic behavior of batter pile groups embedded in liquefiable soil. Earthquake Engineering and Engineering Vibration. 2021;20:583-604.

[173] Zhi Z, Xiaojun L, Riqing L, Chenning S. Shaking table tests and numerical simulations of a small radius curved bridge considering SSI effect. Soil Dynamics and Earthquake Engineering. 2019;118:1-18.

[174] Borthakur B, Bhatacharjee A. Behavior of Single Pile Embedded in Multi-layered Soil Subjected to Dynamic Excitation. Dynamics of Soil and Modelling of Geotechnical Problems: Springer; 2022. p. 331-9.

[175] Hasan AQ, Qasim RM. Study The Behavior of Pile Group Under Torsional and Horizontal Load. Journal of Physics: Conference Series: IOP Publishing; 2021. p. 012032.

[176] Georgiadis M. Interaction between torsional and axial pile responses.

International journal for numerical and analytical methods in geomechanics. 1987;11:645-50.

[177] Georgiadis M, Saflekou S. Piles under axial and torsional loads. Computers and Geotechnics. 1990;9:291-305.

[178] Basack S, Sen S. Numerical solution of single pile subjected to simultaneous torsional and axial loads. International Journal of Geomechanics. 2014;14:06014006.

[179] Liu S-W, Wan J-H, Zhou C-Y, Liu Z, Yang X. Efficient beam–column finite-element method for stability design of slender single pile in soft ground mediums. International Journal of Geomechanics. 2020;20:04019148.

[180] Ouyang W, Liu S-W, Wan J, Yang Y. Euler–Bernoulli pile element for nonlinear buckling analysis of single piles in slope. International Journal of Geomechanics. 2021;21:04021170.

[181] Ouyang W-H, Yang Y, Wan J-H, Liu S-W. Second-order analysis of steel sheet piles by pile element considering nonlinear soil-structure interactions. Adv Steel Constr. 2020;16:354-62.

[182] Li X-Y, Zhao H-P, Liu S-W, Wan J-H, Bai R. Innovative Timoshenko soil-pile integrated element for lager diameter laterally-loaded piles considering soil-pile interactions. Computers and Geotechnics. 2022;152:105020.

[183] Li X-Y, Wan J-H, Zhao H-P, Liu S-W. Three-dimensional analysis of nonlinear pile–soil interaction responses using 3D pile element model. International Journal of Geomechanics. 2021;21:04021129.

[184] Gupta B, Basu D. Applicability of Timoshenko, Euler–Bernoulli and rigid beam

theories in analysis of laterally loaded monopiles and piles. Géotechnique. 2018;68:772-85.

[185] Yin J-H. Comparative modeling study of reinforced beam on elastic foundation. Journal of Geotechnical and Geoenvironmental Engineering. 2000;126:265-71.

[186] Bazoune A, Khulief Y, Stephen N. Shape functions of three-dimensional Timoshenko beam element. Journal of Sound and Vibration. 2003;259:473-80.

[187] Przemieniecki JS. Theory of matrix structural analysis: Courier Corporation; 1985.

[188] Abdelrahman A, Chen L, Liu S-W, Ziemian RD. Timoshenko line-element for stability analysis of tapered I-section steel members considering warping effects. Thin-Walled Structures. 2022;175:109198.

[189] Hildebrand FB. Introduction to numerical analysis: Courier Corporation; 1987.

[190] Rodrigues MAC, Burgos RB, Martha LF. A unified approach to the Timoshenko 3d beam-column element tangent stiffness matrix considering higher-order terms in the strain tensor and large rotations. International Journal of Solids and Structures. 2021;222:111003.

[191] Bai R, Hajjar JF, Liu S-W, Chan S-L. A mixed-field Timoshenko beam-column element for direct analysis of tapered I-sections members. Journal of Constructional Steel Research. 2020;172:106157.

[192] Timoshenko SP, Gere JM. Theory of elastic stability: Courier Corporation; 2009.

[193] Reese LC, Matlock H. Non-dimensional solution for laterally loaded piles with soil modulus assumed proportional to depth. Proc 8th Texas Conf SMFE: The

Univ. of Texas; 1965.

[194] Sinha R, Sarkar R, Rajeswari J. Flexural response of pile foundation in liquefiable soil using finite-difference formulation following pseudostatic approach. Indian Geotechnical Journal. 2020;50:880-906.

[195] McClelland B, Focht J. Soil modulus for laterally loaded piles. Journal of the Soil Mechanics and Foundations division. 1956;82:1081-1--22.

[196] Vega-Posada CA, Gallant AP, Areiza-Hurtado M. Simple approach for analysis of beam-column elements on homogeneous and non-homogeneous elastic soil. Engineering Structures. 2020;221:111110.

[197] Choi YS, Basu D, Salgado R, Prezzi M. Response of laterally loaded rectangular and circular piles in soils with properties varying with depth. Journal of Geotechnical and Geoenvironmental Engineering. 2014;140:04013049.

[198] Cowper G. The shear coefficient in Timoshenko's beam theory. 1966.

[199] Chiorean C. A computer method for nonlinear inelastic analysis of 3D composite steel–concrete frame structures. Engineering Structures. 2013;57:125-52.

[200] Liew JR, Chen H, Shanmugam N. Inelastic analysis of steel frames with composite beams. Journal of Structural Engineering. 2001;127:194-202.

[201] Ngo-Huu C, Kim S-E, Oh J-R. Nonlinear analysis of space steel frames using fiber plastic hinge concept. Engineering Structures. 2007;29:649-57.

[202] Eymard R, Gallouët T, Herbin R. Finite volume methods. Handbook of numerical analysis. 2000;7:713–1018-713–.

[203] LeVeque RJ. Finite difference methods for ordinary and partial differential

equations: steady-state and time-dependent problems: SIAM; 2007.

[204] Monaghan JJ. Smoothed particle hydrodynamics and its diverse applications. Annual Review of Fluid Mechanics. 2012;44:323–46-–46.

[205] Yreux E, Chen J-S. A quasi-linear reproducing kernel particle method. International Journal for Numerical Methods in Engineering. 2017;109:1045–64-–64.

[206] Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: A deep learning library for solving differential equations. SIAM review. 2021;63:208–28-–28.

[207] Wang H, Fu T, Du Y, Gao W, Huang K, Liu Z et al. Scientific discovery in the age of artificial intelligence. Nature. 2023;620:47–60-47–60.

[208] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nature Reviews Physics. 2021;3:422–40-–40.

[209] Belardi VG, Fanelli P, Vivio F. Analysis of multi-bolt composite joints with a user-defined finite element for the evaluation of load distribution and secondary bending. Composites Part B: Engineering. 2021;227:109378-.

[210] Liu F, Yao W, Zhao L, Wu H, Zhang X, Zhang J. An improved 2D finite element model for bolt load distribution analysis of composite multi-bolt single-lap joints. Composite Structures. 2020;253:112770-.

[211] Naser MZ, Hawileh RA, Abdalla J. Modeling strategies of finite element simulation of reinforced concrete beams strengthened with FRP: A review. Journal of Composites Science. 2021;5:19-.

[212] Li X. Finite element analysis of slope stability using a nonlinear failure criterion.

Computers and Geotechnics. 2007;34:127–36-–36.

[213] Huang J, Bathe K-J. On the convergence of overlapping elements and overlapping meshes. Computers & Structures. 2021;244:106429-.

[214] Lucia DJ, Beran PS, Silva WA. Reduced-order modeling: new approaches for computational physics. Progress in aerospace sciences. 2004;40:51–117-51–.

[215] Li Q, Sigmund O, Jensen JS, Aage N. Reduced-order methods for dynamic problems in topology optimization: A comparative study. Computer Methods in Applied Mechanics and Engineering. 2021;387:114149-.

[216] Hafeez MB, Krawczuk M. A review: Applications of the spectral finite element method. Archives of Computational Methods in Engineering. 2023;30:3453–65-–65.

[217] Ouyang W, Bai R, Liu S-W, Chan S-L. Refined three-dimensional pile element formulation for second-order analysis of pile-supported structures accounting for complex loading conditions. Engineering Structures. 2024;301:117281-.

[218] Scott MH, Fenves GL. Plastic hinge integration methods for force-based beam–column elements. Journal of Structural Engineering. 2006;132:244–52-–52.

[219] Ko Y, Lee P-S, Bathe K-J. A new MITC4+ shell element. Computers & Structures. 2017;182:404–18-–18.

[220] Zou Z, Hughes TJR, Scott MA, Sauer RA, Savitha EJ. Galerkin formulations of isogeometric shell analysis: Alleviating locking with Greville quadratures and higher-order elements. Computer Methods in Applied Mechanics and Engineering. 2021;380:113757-.

[221] Lee U. Spectral element method in structural dynamics: John Wiley & Sons; 2009.

[222] Goswami S, Yin M, Yu Y, Karniadakis GE. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Computer Methods in Applied Mechanics and Engineering. 2022;391:114587-.

[223] Azizzadenesheli K, Kovachki N, Li Z, Liu-Schiaffini M, Kossaifi J, Anandkumar A. Neural operators for accelerating scientific simulations and design. Nature Reviews Physics. 2024:1–9-1–9.

[224] He J, Koric S, Kushwaha S, Park J, Abueidda D, Jasiuk I. Novel DeepONet architecture to predict stresses in elastoplastic structures with variable complex geometries and loads. Computer Methods in Applied Mechanics and Engineering. 2023;415:116277-.

[225] Kontolati K, Goswami S, Em Karniadakis G, Shields MD. Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems. Nature Communications. 2024;15:5101-.

[226] Lee S, Shin Y. On the training and generalization of deep operator networks. SIAM Journal on Scientific Computing. 2024;46:C273–C96-C–C96.

[227] Yan CA, Vescovini R, Dozio L. A framework based on physics-informed neural networks and extreme learning for the analysis of composite structures. Computers & Structures. 2022;265:106761-.

[228] Jagtap AD, Kharazmi E, Karniadakis GE. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. Computer Methods in Applied Mechanics and Engineering. 2020;365:113028-.

[229] Jagtap AD, Karniadakis GE. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics. 2020;28.

[230] Kharazmi E, Zhang Z, Karniadakis GE. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. Computer Methods in Applied Mechanics and Engineering. 2021;374:113547-.

[231] Ainsworth M, Shin Y. Active Neuron Least Squares: A training method for multivariate rectified neural networks. SIAM Journal on Scientific Computing. 2022;44:A2253–A75-A–A75.

[232] Saha S, Gan Z, Cheng L, Gao J, Kafka OL, Xie X et al. Hierarchical deep learning neural network (HiDeNN): an artificial intelligence (AI) framework for computational science and engineering. Computer Methods in Applied Mechanics and Engineering. 2021;373:113452-.

[233] Wang J, Mo YL, Izzuddin B, Kim C-W. Exact Dirichlet boundary Physics-informed Neural Network EPINN for solid mechanics. Computer Methods in Applied Mechanics and Engineering. 2023;414:116184-.

[234] Rezaei S, Asl RN, Taghikhani K, Moeineddin A, Kaliske M, Apel M. Finite operator learning: Bridging neural operators and numerical methods for efficient parametric solution and optimization of pdes. arXiv preprint arXiv:240704157. 2024.

[235] Zang Y, Bao G, Ye X, Zhou H. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics. 2020;411:109409-.

[236] Yin M, Zhang E, Yu Y, Karniadakis GE. Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. Computer methods in applied mechanics and engineering. 2022;402:115027-.

[237] Chung SW, Choi Y, Roy P, Moore T, Roy T, Lin TY et al. Train small, model big: Scalable physics simulators via reduced order modeling and domain decomposition. Computer Methods in Applied Mechanics and Engineering. 2024;427:117041-.

[238] Wuni IY, Shen GQ. Critical success factors for modular integrated construction projects: A review. Building research & information. 2020;48:763–84-–84.

[239] Mei Y, Zhang Y, Zhu X, Gou R, Gao J. Fully Convolutional Network enhanced DeepONet-based surrogate of predicting the travel-time fields. IEEE Transactions on Geoscience and Remote Sensing. 2024.

[240] Jin P, Meng S, Lu L. MIONet: Learning multiple-input operators via tensor product. SIAM Journal on Scientific Computing. 2022;44:A3490-A514.

[241] Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Computer Methods in Applied Mechanics and Engineering. 2022;393:114778-.

[242] Brecht R, Popovych DR, Bihlo A, Popovych RO. Improving physics-informed DeepONets with hard constraints. arXiv preprint arXiv:230907899. 2023.

[243] Zhang Z, Moya C, Leung WT, Lin G, Schaeffer H. Bayesian deep operator learning for homogenized to fine-scale maps for multiscale PDE. Multiscale Modeling & Simulation. 2024;22:956–72-–72.

[244] Sheng H, Yang C. PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. Journal of Computational Physics. 2021;428:110085-.

[245] Sukumar N, Srivastava A. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. Computer Methods in Applied Mechanics and Engineering. 2022;389:114333.

[246] Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O et al. Highly accurate protein structure prediction with AlphaFold. nature. 2021;596:583–9-–9.

[247] Price I, Sanchez-Gonzalez A, Alet F, Andersson TR, El-Kadi A, Masters D et al. Probabilistic weather forecasting with machine learning. Nature. 2025;637:84–90-84–90.

[248] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. Advances in neural information processing systems. 2020;33:6840–51-–51.

[249] Brunton SL, Kutz JN. Promising directions of machine learning for partial differential equations. Nature Computational Science. 2024;4:483–94-–94.

[250] Wang Y, Bai J, Lin Z, Wang Q, Anitescu C, Sun J et al. Artificial intelligence for partial differential equations in computational mechanics: A review. arXiv preprint arXiv:241019843. 2024.

[251] Rudy S, Alla A, Brunton SL, Kutz JN. Data-driven identification of parametric partial differential equations. SIAM Journal on Applied Dynamical Systems. 2019;18:643–60-–60.

[252] Wu K, Xiu D. Data-driven deep learning of partial differential equations in modal space. Journal of Computational Physics. 2020;408:109307-.

[253] Dal Santo N, Deparis S, Pegolotti L. Data driven approximation of parametrized PDEs by reduced basis and neural networks. Journal of Computational Physics. 2020;416:109550-.

[254] Zhang A, Lipton ZC, Li M, Smola AJ. Dive into deep learning. arXiv preprint arXiv:210611342. 2021.

[255] Pinkus A. Approximation theory of the MLP model in neural networks. Acta numerica. 1999;8:143–95-–95.

[256] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature machine intelligence. 2021;3:218–29-–29.

[257] Xiong W, Huang X, Zhang Z, Deng R, Sun P, Tian Y. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. Journal of Computational Physics. 2024:113194-.

[258] Tripura T, Chakraborty S. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. Computer Methods in Applied Mechanics and Engineering. 2023;404:115783-.

[259] Cao Q, Goswami S, Karniadakis GE. Laplace neural operator for solving differential equations. Nature Machine Intelligence. 2024;6:631–40-–40.

[260] Hanna JM, Aguado JV, Comas-Cardona S, Askri R, Borzacchiello D. Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering. 2022;396:115100-.

[261] Wang X, Tang K, Zhai J, Wan X, Yang C. Deep adaptive sampling for surrogate

modeling without labeled data. arXiv preprint arXiv:240211283. 2024.

[262] Kingma DP. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.

[263] Gao H, Sun L, Wang J-X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. Journal of Computational Physics. 2021;428:110079-.

[264] Ouyang W, Li G, Chen L, Liu S-W. Physics-informed neural networks for large deflection analysis of slender piles incorporating non-differentiable soil-structure interaction. International Journal for Numerical and Analytical Methods in Geomechanics. 2024;48:1278–308-–308.

[265] Ouyang W, Li G, Chen L, Liu S-W. Machine learning-based soil–structure interaction analysis of laterally loaded piles through physics-informed neural networks. Acta Geotechnica. 2024:1–26-1–.

[266] Wand MP, Jones MC. Kernel smoothing: CRC press; 1994.

[267] Dash S, Rouholamin M, Lombardi D, Bhattacharya S. A practical method for construction of py curves for liquefiable soils. Soil Dynamics and Earthquake Engineering. 2017;97:478-81.

[268] Kundu S, Pani AK. Global stabilization of two dimensional viscous Burgers' equation by nonlinear neumann boundary feedback control and its finite element analysis. Journal of Scientific Computing. 2020;84:45-.

[269] Bouzid DA, Bhattacharya S, Dash S. Winkler Springs (py curves) for pile design from stress-strain of soils: FE assessment of scaling coefficients using the Mobilized Strength Design concept. Geomechanics & engineering. 2013;5:379-

99.

[270] Tak Kim B, Kim N-K, Jin Lee W, Su Kim Y. Experimental load–transfer curves of laterally loaded piles in Nak-Dong River sand. J Geotech Geoenviron Eng. 2004;130:416-25.

[271] Reese LC, Cox WR, Koop FD. Field testing and analysis of laterally loaded piles om stiff clay. Offshore technology conference: OnePetro; 1975.

[272] Nadeem M, Chakraborty T, Matsagar V. Nonlinear buckling analysis of slender piles with geometric imperfections. Journal of Geotechnical and Geoenvironmental Engineering. 2015;141:06014014.

[273] Li X, Wan J, Liu S, Zhang L. Numerical formulation and implementation of Euler-Bernoulli pile elements considering soil-structure-interaction responses. Int J Numer Anal Methods Geomech. 2020;44:1903-25.

[274] Tang C, Phoon K-K. Evaluation of model uncertainties in reliability-based design of steel H-piles in axial compression. Canadian Geotechnical Journal. 2018;55:1513-32.

[275] Zhang L, Tang WH, Ng CW. Reliability of axially loaded driven pile groups. J Geotech Geoenviron Eng. 2001;127:1051-60.

[276] Gong W, Tang H, Juang CH, Wang L. Optimization design of stabilizing piles in slopes considering spatial variability. Acta Geotech. 2020;15:3243-59.

[277] Chan CM, Zhang L, Ng JT. Optimization of pile groups using hybrid genetic algorithms. J Geotech Geoenviron Eng. 2009;135:497-505.

[278] Tang H, Gong W, Wang L, Juang CH, Martin JR, Li C. Multiobjective

optimization-based design of stabilizing piles in earth slopes. Int J Numer Anal Methods Geomech. 2019;43:1516-36.

[279] Yan C, Vescovini R, Dozio L. A framework based on physics-informed neural networks and extreme learning for the analysis of composite structures. Computers & Structures. 2022;265:106761.

[280] Koric S, Viswantah A, Abueidda DW, Sobh NA, Khan K. Deep learning operator network for plastic deformation with variable loads and material properties. Engineering with Computers. 2023:1-13.

[281] Bisagni C, Lanzi L. Post-buckling optimisation of composite stiffened panels using neural networks. Compos Struct. 2002;58:237-47.

[282] Zhang W, Goh AT. Multivariate adaptive regression splines and neural network models for prediction of pile drivability. Geosci Front. 2016;7:45-52.

[283] Chojaczyk AA, Teixeira AP, Neves LC, Cardoso JB, Soares CG. Review and application of Artificial Neural Networks models in reliability analysis of steel structures. Struct Saf. 2015;52:78-89.

[284] Mata J. Interpretation of concrete dam behaviour with artificial neural network and multiple linear regression models. Eng Struct. 2011;33:903-10.

[285] García-Segura T, Yepes V, Frangopol DM. Multi-objective design of post-tensioned concrete road bridges using artificial neural networks. Struct Multidiscip Optim. 2017;56:139-50.

[286] Nguyen LTK, Keip M-A. A data-driven approach to nonlinear elasticity. Computers & Structures. 2018;194:97-115.

[287] Harandizadeh H, Jahed Armaghani D, Khari M. A new development of ANFIS–GMDH optimized by PSO to predict pile bearing capacity based on experimental datasets. Engineering with Computers. 2021;37:685-700.

[288] Raissi M, Perdikaris P, Karniadakis GE. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:171110561. 2017.

[289] Zhang P, Yin ZY, Jin YF, Sheil B. Physics-constrained hierarchical data-driven modelling framework for complex path-dependent behaviour of soils. Int J Numer Anal Methods Geomech. 2022;46:1831-50.

[290] Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Computer Methods in Applied Mechanics and Engineering. 2021;379:113741.

[291] Depina I, Jain S, Mar Valsson S, Gotovac H. Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards. 2022;16:21-36.

[292] Yang L, Zhang D, Karniadakis GE. Physics-informed generative adversarial networks for stochastic differential equations. SIAM Journal on Scientific Computing. 2020;42:A292-A317.

[293] Zhang P, Yin Z-Y, Sheil B. A physics-informed data-driven approach for consolidation analysis. Géotechnique. 2023:1-12.

[294] Feng Q, Kong Q, Song G. Damage detection of concrete piles subject to typical damage types based on stress wave measurement using embedded smart aggregates transducers. Measurement. 2016;88:345-52.

[295] Nguyen V, Dackermann U, Li J, Alamdari MM, Mustapha S, Runcie P et al. Damage identification of a concrete arch beam based on frequency response functions and artificial neural networks. Electronic Journal of Structural Engineering. 2015;14:75-84.

[296] Ouyang W, Li G, Chen L, Liu S-W. Machine Learning-based Soil-Structure Interaction Analysis of Laterally-loaded Piles through Physics-Informed Neural Networks. Acta Geotech. 2023.

[297] Matlock H, Reese LC. Generalized solutions for laterally loaded piles. Journal of the Soil Mechanics and foundations Division. 1960;86:63-92.

[298] Corliss G, Faure C, Griewank A, Hascoet L, Naumann U. Automatic differentiation of algorithms: from simulation to optimization: Springer Science & Business Media; 2002.

[299] Teh LH, Clarke MJ. Co-rotational and Lagrangian formulations for elastic three-dimensional beam finite elements. J Constr Steel Res. 1998;48:123-44.

[300] Liu S-W, Pekoz T, Gao W-L, Ziemian RD, Crews J. Frame analysis and design of industrial rack structures with perforated cold-formed steel columns. Thin-Walled Structures. 2021;163:107755.

[301] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.

[302] Yang H, Zhu Y, Lu Q, Zhang J. Dynamic reliability based design optimization of the tripod sub-structure of offshore wind turbines. Renewable Energy. 2015;78:16-25.

[303] Ellingwood B, Galambos TV, MacGregor JG, Cornell CA. Development of a

probability based load criterion for American National Standard A58, NBS Special Publication 577. Washington, DC: National Bureau of Standards. 1980.

[304] Ellingwood BR, Tekie PB. Wind load statistics for probability-based structural design. J Struct Eng. 1999;125:453-63.

[305] Zio E. Monte carlo simulation: The method: Springer; 2013.

[306] Jeong S, Kim Y, Kim J. Influence on lateral rigidity of offshore piles using proposed p–y curves. Ocean engineering. 2011;38:397-408.

[307] Ismael NF. Behavior of step tapered bored piles in sand under static lateral loading. Journal of geotechnical and geoenvironmental engineering. 2010;136:669-76.

[308] Kawamata Y, Ashford SA, Nimityongskul N. Full-scale lateral pile load test in rock fill. Geotechnical earthquake engineering and soil dynamics IV2008. p. 1-9.

[309] Pando MA, Ealy CD, Filz GM, Lesko J, Hoppe E. A laboratory and field study of composite piles for bridge substructures. United States. Federal Highway Administration. Office of Infrastructure …; 2006.

[310] Sayegh AF, Tso FK. Analysis of linear structures on nonlinear pile foundations. Computers & structures. 1988;29:633-43.

[311] Ouyang W, Liang A-R, Liu S-W. Efficient numerical implementation for nonlinear analysis of pile-supported structures during soil liquefaction through adaptive pile element formulations. Soil Dynamics and Earthquake Engineering. 2023;174:108207.