МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова праця на правах рукопису

ЯРОШ АНАСТАСІЯ ОЛЕКСАНДРІВНА

УДК 004.8.032.26:517.954

ДИСЕРТАЦІЯ

НЕЙРОМЕРЕЖЕВІ МЕТОДИ РОЗВ’ЯЗАННЯ КРАЙОВИХ ЗАДАЧ

Спеціальність: 122 Комп’ютерні науки
Галузь знань: 12 Інформаційні технології

Подається на здобуття наукового ступеня **доктора філософії**

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ А. О. Ярош

Науковий керівник: **Кудін Олексій Володимирович**, кандидат фізико-математичних наук, доцент

Запоріжжя – 2024

АНОТАЦІЯ

Ярош А. О. Нейромережеві методи розв’язання крайових задач. – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 “Комп’ютерні науки”. – Запорізький національний університет, Запоріжжя, 2024.

Дисертаційна робота присвячена розробці відкритої об’єктно-орієнтованої архітектури бібліотеки нейромережевих методів розв’язання крайових задач.

Значущість розвитку наближених методів розв’язання диференціальних рівнянь визначається їх широким застосуванням у ключових галузях науки та техніки. Оскільки багато фізичних явищ можна математично описати диференціальними рівняннями, але знаходження їх аналітичних розв’язків часто є складним завданням, чисельні методи наближеного розв’язання стають критично важливими. Ці методи необхідні для комп’ютерного моделювання та симуляції поведінки складних технічних систем. Класичні методи розв’язання крайових задач (метод колокації, метод Гальоркіна, метод Рітца) потребують вибору базисних функцій для побудови наближеного розв’язку. Невірний вибір може призвести до некоректних результатів. Крім того, збільшення кількості базисних функцій для поліпшення точності може призвести до зростання обчислювальної складності, особливо для великих систем диференціальних рівнянь.

Використання нейронних мереж з фізичною інформацією для розв’язання крайових задач має кілька переваг порівняно з класичними методами. По-перше, нейронні мережі дозволяють здійснювати апроксимацію складних фізичних процесів без потреби у виборі певних базисних функцій. По-друге, нейронні мережі здатні автоматично виявляти нелінійні залежності в даних, що робить їх ефективними для моделювання складних фізичних явищ. Крім того, нейронні мережі можуть адаптуватися до нових даних та умов задачі без необхідності перегляду аналітичних апроксимацій, що робить їх більш гнучкими і придатними для застосування в різних областях науки та інженерії. Нейромережеві методи

також ефективно використовуються для розв’язання обернених задач. Вони дозволяють визначати параметри системи або властивості середовища на основі вимірювань або спостережень. Невідомі константи оберненої задачі, що підлягають визначенню вводяться в число параметрів нейронної мережі та оптимізуються під час навчання.

Наразі, існують програмні бібліотеки, що реалізують нейронні мережі з фізичною інформацією, зокрема, DeepXDE, NeuralPDE, Nvidia Modulus, SciANN та PINNs-Torch. Однак, формат запису задачі є, зазвичай, досить складним, і потребує знань з програмування та вивчення документації до бібліотек. Також, наявні бібліотеки надають користувачеві фіксований набір функції, зазвичай, без можливості розширення та додавання власних методів та програмного коду.

В дисертаційній роботі розроблено архітектуру об’єктно-орієнтованої бібліотеки, що реалізує метод нейронних мереж з фізичною інформацією для розв’язання крайових задач. Розроблено предметно-орієнтовану мову PLang (Problem Language), яка використовується для формального опису крайових задач. Використання спеціалізованої мови дозволяє визначати задачу та крайові умови у зрозумілий для науковців та інженерів формі. Це суттєво спрощує використання запропонованої в дисертації бібліотеки. Структура класів бібліотеки передбачає можливість масштабування користувачами бібліотеки.

Для тестування розроблених в роботі методів реалізації нейронних мереж з фізичною інформацією розв’язуються лінійні та нелінійні задачі пружності, прямі та обернені задачі рівнянь Бюргерса, диференціальні рівняння. Продемонстровано збіжність на числових прикладах з різними крайовими умовами та параметрами.

Налаштування гіперпараметрів нейромереж з фізичною інформацією є актуальною задачею з кількох причин. Гіперпараметри, такі як кількість шарів, кількість нейронів у кожному шарі, швидкість навчання, тип активаційних функцій та параметри регуляризації, значно впливають на здатність моделі навчатися та узагальнювати дані. Для таких нейромереж правильне налаштування гіперпараметрів може значно покращити точність та стабільність моделювання

фізичних процесів, забезпечуючи краще задоволення граничних умов і диференціальних рівнянь.

В дисертаційній роботі реалізовано еволюційні методи оптимізації гіперпараметрів нейромереж за допомогою рою часток та генетичні алгоритми. Це дозволить автоматично визначати найкращу мережу для розв’язання задачі.

Ще одним засобом визначення оптимальних гіперпараметрів є методи планування експериментів. Вони полягають у розробці ефективної стратегії для проведення експериментів, яка дозволяє зменшити кількість необхідних проб та ресурсів, забезпечуючи при цьому точність та надійність отриманих результатів.

Застосування методів планування експериментів у контексті оптимізації гіперпараметрів нейронних мереж дозволяє зменшити кількість необхідних експериментів. Це економить обчислювальні ресурси та час, зменшуючи витрати на проведення великої кількості тренувань моделей. Дослідження допомагають виявити ключові гіперпараметри, що мають найбільший вплив на продуктивність моделі, дозволяючи зосередитися на їх оптимізації.

В дисертаційній роботі реалізовано метод планування експериментів Generalized Subset Designs (GSD), який дозволяє ефективно зменшувати кількість необхідних експериментів з багатьма гіперпараметрами нейромереж з фізичною інформацією.

В роботі досліджено вплив на стійкість та збіжність нейромереж параметрів крайових задач, зокрема, коефіцієнтів у диференціальних рівняннях Бюргерса. Показано, що при збільшенні коефіцієнтів, точність моделей нейромереж з фізичною інформацією знижується. Для подолання цієї проблеми запропоновано використання більш глибоких нейромереж, які здатні моделювати складніші нелінійні паттерни в даних. Процес пошуку оптимальною структури мережі в даному випадку не є інтуїтивним і потребує використання методів автоматичного пошуку оптимальних гіперпараметрів, наприклад, еволюційних алгоритмів або методів планування експериментів.

Отже, у дисертаційній роботі було вирішено актуальну задачу створення відкритої об’єктно-орієнтованої архітектури бібліотеки нейромережевих методів

розв’язання крайових задач. Реалізовано методи нейромереж з фізичною інформацією. Розроблено предметно-орієнтовану мову PLang (Problem Language), яка використовується для формального опису крайових задач. Для покращення збіжності нейромережевих методів реалізовано еволюційні підходи та методи планування експериментів для оптимізації гіперпараметрів. А також досліджено вплив параметрів крайових задач на збіжність нейронних мереж з фізичною інформацією.

Запропоновані в роботі методи протестовано на модельних задачах пружності та математичної фізики.

Реалізована бібліотека може використовуватись дослідниками та інженерами для розв’язання широкого класу крайових задач. А також може бути використана як частина систем комп’ютерної алгебри.

Ключові слова: еволюційна оптимізація, задачі пружності, крайові задачі, методи покращення збіжності, нейромережі з фізичною інформацією, обернені задачі, рівняння Бюргерса

ABSTRACT

Yarosh A. O. Neural Network Methods for Solving Boundary Value Problems. – Qualifying scientific work on the rights of the manuscript. The dissertation on competition of a scientific degree of the doctor of philosophy on a specialty 122 “Computer Science”. – Zaporizhia National University, Zaporizhia, 2024.

The dissertation is dedicated to developing an open object-oriented architecture for a library of neural network methods for solving boundary value problems.

The significance of developing approximate methods for solving differential equations is underscored by their extensive application in key fields of science and engineering. Many physical phenomena can be mathematically described by differential equations, but finding their analytical solutions is often challenging. Thus, numerical

methods for approximate solutions become critically important for computer modeling and simulating the behavior of complex technical systems. Classical methods for solving boundary value problems (such as the collocation method, Galerkin method, and Ritz method) require the selection of basis functions to construct an approximate solution. An incorrect choice can lead to inaccurate results. Additionally, increasing the number of basis functions to improve accuracy can lead to computational complexity, especially for large systems of differential equations.

Using physics-informed neural networks (PINNs) for solving boundary value problems has several advantages over classical methods. First, neural networks allow the approximation of complex physical processes without the need for selecting specific basis functions. Second, neural networks can automatically detect nonlinear dependencies in the data, making them effective for modeling complex physical phenomena. Furthermore, neural networks can adapt to new data and problem conditions without needing to revise analytical approximations, making them more flexible and suitable for various fields of science and engineering. Neural network methods are also effectively used for solving inverse problems, allowing the determination of system parameters or medium properties based on measurements or observations. Unknown constants of the inverse problem, which need to be determined, are introduced as parameters of the neural network and optimized during training.

Currently, there are software libraries implementing physics-informed neural networks, such as DeepXDE, NeuralPDE, Nvidia Modulus, SciANN, and PINNs-Torch. However, the task formulation format is usually quite complex, requiring programming knowledge and familiarity with library documentation. Additionally, existing libraries provide users with a fixed set of functions, typically without the possibility of extending and adding custom methods and code.

The dissertation work developed an object-oriented library architecture that implements physics-informed neural network methods for solving boundary value problems. A domain-specific language, PLang (Problem Language), was developed for the formal description of boundary value problems. The use of a specialized language allows defining the problem and boundary conditions in a form understandable to

scientists and engineers, significantly simplifying the use of the proposed library. The class structure of the library is designed for scalability by users.

To test the developed methods for implementing physics-informed neural networks, linear and nonlinear elasticity problems, direct and inverse problems of Burgers' equations, and differential equations are solved. Convergence is demonstrated on numerical examples with different boundary conditions and parameters.

Tuning the hyperparameters of physics-informed neural networks is a relevant task for several reasons. Hyperparameters, such as the number of layers, the number of neurons in each layer, learning rate, activation functions, and regularization parameters, significantly affect the model's ability to learn and generalize data. Proper tuning of hyperparameters can greatly improve the accuracy and stability of modeling physical processes, ensuring better satisfaction of boundary conditions and differential equations. The dissertation implements evolutionary methods for optimizing neural network hyperparameters using particle swarm optimization and genetic algorithms, enabling the automatic determination of the best network for solving the problem.

Another method for determining optimal hyperparameters is experimental design methods. These involve developing an efficient strategy for conducting experiments, allowing for a reduction in the number of necessary trials and resources while ensuring the accuracy and reliability of the results. Applying experimental design methods in the context of optimizing neural network hyperparameters helps reduce the number of required experiments, saving computational resources and time. The research identifies key hyperparameters that have the greatest impact on model performance, allowing for focused optimization.

The dissertation implements the Generalized Subset Designs (GSD) method, effectively reducing the number of necessary experiments for many hyperparameters of physics-informed neural networks.

The study investigates the impact of boundary condition parameters on the stability and convergence of neural networks, particularly focusing on the coefficients in Burgers' differential equations. It is shown that as these coefficients increase, the accuracy of physics-informed neural network models decreases. To address this issue,

the use of deeper neural networks is proposed, as they are capable of modeling more complex nonlinear patterns in the data. The process of finding the optimal network structure in this case is not intuitive and requires the use of automated hyperparameter optimization methods, such as evolutionary algorithms or design of experiments techniques.

Thus, the dissertation addresses the relevant task of creating an open object-oriented architecture for a library of neural network methods for solving boundary value problems. The work implements methods of physics-informed neural networks and develops a domain-specific language, PLang (Problem Language), for the formal description of boundary value problems. Evolutionary approaches and experimental design methods for optimizing hyperparameters are implemented to improve the convergence of neural network methods. The impact of boundary condition parameters on the convergence of physics-informed neural networks has also been studied.

The proposed methods are tested on model problems of elasticity and mathematical physics. The implemented library can be used by researchers and engineers for solving a wide range of boundary value problems and can also be integrated as part of computer algebra systems.

Keywords: evolutionary optimization, elasticity problems, boundary value problems, convergence improvement methods, physics-informed neural networks, inverse problems, Burgers' equations.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ, В ЯКИХ ОПУБЛІКОВАНІ ОСНОВНІ НАУКОВІ РЕЗУЛЬТАТИ ДИСЕРТАЦІЇ:

1) Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання прямих і обернених крайових задач. Computer Science and Applied Mathematics. 2024. (1), 92-100. https://doi.org/10.26661/2786-6254-2024-1-11

2) Ярош А.О., Кудін О.В. Нейроеволюційний метод колокації розв’язання диференціальних рівнянь. Таврійський науковий вісник. Серія: Технічні науки. 2024. (6), 72-81. https://doi.org/10.32782/tnv-tech.2023.6.9

3) Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання задач пружності. Вісник Херсонського національного технічного університету. 2024. № 1(88), 295-305. https://doi.org/10.35546/kntu2078-4481.2024.1.41

4) Ярош А.О.,Кудін О.В. Застосування нейронних мереж у розв’язанні диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Дванадцятої Всеукраїнської, дев’ятнадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2021, 79-80.

5) Ярош А.О., Кудін О.В. Нейроеволюційний метод розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Тринадцятої Всеукраїнської, двадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2022, 61-63.

6) Сачковська А.І, Ярош А.О., Кудін О.В. Нейроеволюційний варіант методу Гальоркіна розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Чотирнадцятої Всеукраїнської, двадцять першої регіональної наукової конференції молодих дослідників. Запоріжжя, 2023, 90-92.

7) Ярош А.О., Кудін О.В. Нейромережіві обчислювальні методи у задачах згину функціональноградієнтних балок. Міжнародна наукова конференція «актуальні проблеми механіки — 2023» до 145-річчя від дня народження С.П.Тимошенка, Київ, Дніпро, Львів, Харків – 2023, 227-228.

8) Ярош А.О., Кудін О.В. Об’єктно-орієнтована бібліотека нейромережевих методів розв’язання крайових задач. Актуальні проблеми математики та інформатики : збірка тез доповідей П’ятнадцятої Всеукраїнської, двадцять другої регіональної наукової конференції молодих дослідників. Запоріжжя, 2024, 137-139.

9) Ярош А.О., Кудін О.В. Проєктування бібліотеки нейромережевих методів розв’язання крайових задач. Розвиток наук в умовах нової реальності: проблеми та перспективи: збірник наукових праць з матеріалами II Міжнародної наукової конференції, м. Київ, 3 травня, 2024 р. 143-146. https://doi.org/10.62731/mcnd-03.05.2024.004

10) Ярош А.О., Кудін О.В. Планування експериментів та метод рою часток оптимізації нейромережевих методів розв’язання крайових задач. Інновації та науковий потенціал світу: збірник наукових праць з матеріалами IV Міжнародної наукової конференції, м. Запоріжжя, 24 травня, 2024 р. 160-163. https://doi.org/10.62731/mcnd-24.05.2024.002

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ООП – об’єктно-орієнтоване програмування

PINN – Physics-Informed Neural Networks

PSO – Particle Swarm Optimization

GA – Genetic Algorithm

GSD – Generalized Subset Design

UML – Unified Modeling Language

PDE – Partial differential equation

EBNF – Extended Backus–Naur Form

ВСТУП

Актуальність теми. Розробка ефективного математичного забезпечення систем аналізу інженерних задач є актуальним напрямом досліджень. Це зумовлено тим, що ефективне математичне забезпечення дозволяє зменшити витрати часу і фінансових ресурсів на розробку та тестування нових інженерних рішень. Наприклад, використання комп’ютерних симуляцій замість фізичних прототипів може суттєво знизити витрати на дослідження і розробки.

Разом із класичними підходами, наприклад, методом скінченних елементів, у сучасних системах інженерного аналізу використовуються також нейронні мережі при прогнозуванні та оптимізації, обробці сигналів тощо [1]. Інженерні задачі часто зводяться до аналізу математичних моделей, які описуються диференціальними рівняннями як лінійними, так і нелінійними [2].

На даний час розвинуті потужні наближені методи розв’язання крайових задач для диференціальних рівнянь та систем з частинними похідними, такі, наприклад, як метод Рітца, Гальоркіна, колокації або скінченних елементів [3]. Альтернативою їм є напрям “scientific machine learning” (SciML) або “Physics-informed machine learning ” (PIML) [4, 5], особливістю якого є застосування методів машинного навчання у моделюванні вимогливих до ресурсів наукових задач. Основою для таких методів є теореми про збіжність апроксимації нейронними мережами [6]. Ідея цих підходів полягає у заміні невідомої функції та її похідних нейронною мережею та застосування додаткової інформації з диференціальних рівнянь та граничних умов при навчанні мережі. Були розроблені глибинні варіанти методів Рітца, Гальоркіна, колокації тощо [7-17]. Результатом роботи цих алгоритмів є нейронна мережа, що відповідає диференціальному рівнянню та крайовим умовам.

Застосування нейронної архітектури додає чисельним методам такі переваги [5]: штучні нейронні мережі дозволяють апроксимувати нелінійні залежності довільної складності, налаштування параметрів мережі відбувається під час

навчання; такі методи є загальними і можуть бути застосованими до звичайних диференціальних рівнянь та рівнянь у частинних похідних; ефективно працюють на задачах високої розмірності; глибинні методи можуть бути ефективно реалізовані на паралельних архітектурах.

До недоліків можна віднести: необхідність налаштування гіперпараметрів нейромереж, що може бути обчислювально складною задачею; недостатня точність глибинних методів у порівнянні з класичними; нейронні мережі стандартно розглядаються як методи «чорної скрині».

Актуальною задачею є розвиток обчислювальних методів розв’язання диференціальних рівнянь та їх систем у напрямі розширення застосування нейромереж для розв’язання нелінійних фізичних та інженерних задач, застосуванні методів покращення збіжності нейромереж.

Зв’язок роботи з науковими програмами, планами, темами. Одержані в дисертаційній роботі результати повністю відповідають основним напрямам наукових досліджень, що виконуються у Запорізькому національному університеті. Зокрема, робота виконувалася у відповідності до плану науково-технічних робіт Запорізького національного університету при виконанні науково-дослідної теми: “Математичне та програмне забезпечення наукових досліджень” (номер державної реєстрації 0121U114694), яка виконувалась у межах робочого часу викладачів.

Мета і задачі дослідження. Метою дисертаційного дослідження є розробка відкритої об’єктно-орієнтованої архітектури бібліотеки нейромережевих методів розв’язання крайових задач.

Досягнення поставленої мети передбачає вирішення наступних задач:

1) проведення аналітичного огляду наявних підходів та бібліотек нейромережевих обчислювальних методів, їх програмної архітектури і способів застосування;

2) створення відкритої об’єктно-орієнтованої архітектури бібліотеки нейромережевих обчислювальних методів, яка б дозволяла масштабування та додавання нових підходів;

3) розробка відповідних алгоритмів реалізації нейромережевих методів розв’язання крайових задач;

4) створення відповідного програмного забезпечення;

5) виконання тестових розрахунків для верифікації ефективності запропонованої архітектури та алгоритмів.

Об’єктом дослідження є процес розв’язання крайових задач за допомогою нейромережевих обчислювальних методів.

Предметом дослідження є об’єктно-орієнтована архітектура відкритого програмного забезпечення, що реалізує нейромережеві обчислювальні методи розв’язання крайових задач.

Методи дослідження базуються на застосуванні апарату обчислювальної математики, прикладного та системного програмування, об’єктно-орієнтованого аналізу.

Наукова новизна одержаних результатів полягає у розв’язанні актуальної задачі створення ефективної об’єктно-орієнтованої архітектури бібліотеки нейромережевих обчислювальних методів.

При виконанні дисертаційної роботи отримано такі наукові результати:

1) *вперше* запропоновано об’єктно-орієнтовану архітектуру бібліотеки нейромережевих методів розв’язання крайових задач, яка, на відміну від існуючих бібліотек, реалізує мову опису крайових задач та методи покращення збіжності нейромереж;

2) *отримали подальшого розвитку* методи нейронних мереж з фізичною інформацією, які було адаптовано для розв’язання задач пружності;

3) *вперше* реалізовано предметно-орієнтовану мову для формального опису крайових задач та їх розв’язання засобами нейронних мереж;

4) *вперше* отримано чисельні розрахунки певних класів крайових задач методами нейромереж з фізичною інформацією.

Практичне значення одержаних результатів. Запропонована в дисертації архітектура бібліотеки нейромережевих методів розв’язання крайових задач сприятиме розширенню застосування цих методів в інженерній практиці.

Розроблена бібліотека та програмне забезпечення дає можливість автоматизувати процес розв’язання крайових задач та може використовуватись як складова систем автоматизованого проєктування або систем комп’ютерної алгебри.

Апробація результатів дослідження. Результати дисертаційного дослідження були оприлюднені на таких наукових конференціях:

1) Дванадцята Всеукраїнська, дев’ятнадцята регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 2021 р.);

2) Тринадцята Всеукраїнська, двадцята регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 2022 р.);

3) Чотирнадцята Всеукраїнська, двадцять перша регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 2023 р.);

4) Міжнародна наукова конференція «актуальні проблеми механіки — 2023» до 145-річчя від дня народження С.П. Тимошенка (Київ, 2023 р.);

5) П’ятнадцята Всеукраїнська, двадцять друга регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 2024 р.);

6) II Міжнародної наукової конференції «Розвиток наук в умовах нової реальності: проблеми та перспективи» (Київ, 2024 р.);

7) Міжнародна наукова конференція «Інновації та науковий потенціал світу» (Запоріжжя, 2024 р.).

Публікації. Основні положення роботи було опубліковано у 10 наукових працях, із них: 3 статті опубліковано у фахових виданнях, що належать до категорії Б; 7 тез доповідей на науково-практичних конференціях.

Структура та обсяг роботи. Дисертація складається зі вступу, чотирьох розділів, списку використаних джерел та додатків. Загальний обсяг дисертації становить 119 сторінки. Робота містить 46 рисунків та 2 таблиці. Список

використаних джерел налічує 109 найменувань (13 сторінок). Додатки – 6 сторінок.

1 АНАЛІЗ СТАНУ ПРОБЛЕМИ

1.1 Огляд публікацій з теми дослідження

Публікації, присвячені застосуванню PINN мереж до розв’язання крайових задач, варіюються від теоретичних досліджень до практичних застосувань. Вони охоплюють широкий спектр тем, включаючи розробку та оптимізацію архітектур PINNs, порівняльний аналіз із традиційними чисельними методами, впровадження в галузі, такі як механіка рідин і газів, електродинаміка, та обчислювальна механіка. Практичні задачі, які часто розглядаються, включають моделювання руху рідин, теплових процесів та динаміки біологічних систем. Особливу увагу приділяється валідації отриманих рішень та аналізу їх точності і стабільності. Розглянемо далі основні публікації цього напряму.

Статтю [7] присвячено розробці загального метода розв’язання звичайних диференціальних рівнянь та рівнянь у частинних похідних, який застосовує нейронні мережі для апроксимації невідомої функції. Використовується мережа прямого поширення сигналу, параметри якої налаштовуються при мінімізації відповідної функції втрат. В свою чергу, функція втрат складається з двох частин. Перший член відповідає початковим або граничним умовам задачі. Другий член задає нейронну мережу, яка повинна задовольняти диференціальному рівнянню. Особливість цього методу полягає в тому, що розв’язок представляється у замкнутій диференційованій формі, яку можна використовувати у подальших обчисленнях. В той час як традиційні методи пропонують дискретний розв’язок (метод Рунге-Кутта, послідовних наближень тощо). Демонструється збіжність запропонованого метода з точними розв’язками модельних задач.

В роботі [8] розробляється підхід до навчання нейронних мереж на основі даних, що описують деякий фізичний процес. Автори пропонують

використовувати апріорні знання про відповідні фізичні закони та гіпотетичні залежності як регуляризатори функції втрат нейромережі. В залежності від характеристик наявних даних, розроблено два типи моделей: з неперервною та дискретною часовою шкалою. Перший тип може використовуватись для апроксимації просторово-часових функцій. Моделі другого типу передбачають ітераційний процес з кроком за часом. В роботі розглянуто параметричні та нелінійні диференціальні рівняння в частинних похідних.

Стаття [9] присвячена розробці нейромережевого варіанта метода Гальоркіна розв’язання багатовимірних параболічних диференціальних рівнянь. Цей варіант метода в цілому відповідає класичному підходу та має такі основні етапи: невідома функція замінюється нейронною мережею, із застосуванням метода автоматичного диференціювання обчислюються необхідні похідні; формується цільова функція, яка є комбінацією квадратичних відхилень значень рівняння та граничних умов; генерується випадкова множина пробних точок з області визначення шуканої функції та граничних умов; обчислюється значення нев’язок цільової функції у випадкових точках; застосовується крок градієнтного спуску до значень параметрів нейронної мережі, причому параметр швидкості навчання зменшується зі зростанням кількості ітерацій алгоритму. Отже, нейромережевий варіант метода Гальоркіна замінює базисні функції на нейронну мережу. Під час навчання мережі стохастичним градієнтним спуском налаштовуються параметри нейромережі з урахуванням диференціального рівняння та крайових умов.

Нейромережевий варіант метода Рітца пропонується в роботі [10]. Основна ідея цього підходу також схожа на попередні з врахуванням того, що цей метод застосовується для варіаційних задач. Функції апроксимації замінюються на нейромережу з параметрами, які налаштовуються під час навчання методом градієнтного спуску.

Робота [11] присвячена адаптації нейромереж до метода колокації на прикладі розв’язання задачі згину тонких квадратних та круглих пластин. Результати обчислювальних експериментів демонструють узгодженість

прогнозованої деформації пластини з точним розв’язком. Зазначається, що зі збільшенням кількості шарів нейронної мережі прямого поширення сигналу та кількості нейронів в них, прогнозоване значення наближається до точного. При цьому, використовувались випадкові точки колокації, середня квадратична похибка для оцінки функції втрат та варіант метода градієнтного спуска з адаптивною швидкістю навчання.

Роботу [12] присвячено використанню функціоналу першого порядку методу найменших квадратів у якості функції втрат нейромережі прямого поширення сигналу. Метод використовується для розв’язання еліптичних диференціальних рівнянь.

В статті [13] представлено архітектуру нейромережі динамічного глибокого навчання на основі методу скінченних елементів для розв’язання лінійних параметричних диференціальних рівнянь з частинними похідними. Під час уточнення сітки зв’язки між нейронами в архітектурі мережі імітують графік зв’язності скінченних елементів. Розглянуто декілька функцій втрат. Метод реалізовано для просторової області 1D.

Статтю [14] присвячено поєднанню класичного метода скінченних елементів з нейронними мережами для розв’язання прямих та обернених задач.

В [15] розглядається адаптивний алгоритм hp-FEM, який генерує оптимальне уточнення сітки та забезпечує експоненціальну збіжність чисельної похибки відносно розміру сітки. Таким чином, це дозволяє вирішувати складні інженерні задачі з максимально можливою чисельною точністю. У цій роботі замінюється обчислювально дороге ядро алгоритму уточнення глибокою нейронною мережею. Мережа вчиться оптимально уточнювати елементи та змінювати порядок поліномів. Отже, детермінований алгоритм замінюється нейронною мережею.

Нейронні мережі радіально базисних функцій (РБФ мережі) з фізичною інформацією розробляються в роботі [16]. На відміну від глибоких нейронних мереж, радіальна базисна мережа містить тільки один прихований шар і відповідно радіальні базисні функції активації. Продемонстровано, що даний тип

мереж з використанням методів градієнтного спуску є збіжним. Чисельні приклади показали, що РБФ мережі є більш ефективним у розв’язанні нелінійних диференціальних рівнянь в частинних похідних, ніж глибинні нейромережі з фізичною інформацією.

В [17] розглянуто підхід ансамблювання PINN мереж для більшої стійкості розв’язку.

Статтю [18] присвячено впровадженню вдосконаленої нейронної мережі з фізичною інформацією для оцінки параметрів пристроїв електромережі. Розроблено методологію перетворення даних, яка значно прискорює процес навчання, досягаючи збільшення швидкості до 82,87% порівняно з оригінальним PINN.

В [19] запропоновано нову структуру глибокого навчання під назвою покращених дробових нейронних мереж з фізичною інформацією.

У статті [20] пропонується підхід для покращення навчання моделі нейронної мережі з фізикою для параболічних задач із різко збуреною початковою умовою. Як приклад параболічної задачі розглядається рівняння адвекції-дисперсії з початковою умовою точкового гаусового джерела.

У [21] запропоновано вдосконалений метод прогнозування на основі PINN з метою розв’язання диференціальних рівнянь в частинних похідних зі складними граничними умовами, такими як граничні умови Неймана, в яких характерна інформація просторового розподілу збільшується за допомогою невеликої кількості виміряних даних, а рівняння втрат динамічно коригується за ваговими коефіцієнтами втрат. Виміряні дані перетворюються на квадратичний регулярний член і додаються до функції втрат як дані ознак, щоб керувати процесом оновлення для ваги та коефіцієнта зсуву кожного нейрона в нейронній мережі.

Зворотно сумісна нейронна мережа з фізичними відомостями (bc-PINN) є часовою послідовною схемою для розв’язання PDE протягом послідовних сегментів часу, задовольняючи всі раніше отримані рішення. У [22] пропонується вдосконалення оригінального алгоритму bc-PINN у двох аспектах на основі характеристик розповсюдження помилок. Один полягає в тому, щоб змінити член

функції втрат для забезпечення зворотної сумісності, вибравши найбільш раннє вивчене рішення для кожного піддомену як псевдоеталонне рішення. Інший полягає в прийнятті конкатенації рішень, отриманих від окремих підмереж, як остаточної форми прогнозованого рішення. Покращена зворотна сумісність PINN (Ibc-PINN) застосована для дослідження хвиль вищого порядку, керованих даними, для нелінійного рівняння Шредінгера.

Моделювання потоку на основі нейронних мереж з фізичною інформацією PINN стає розвиненою технікою штучного інтелекту для вирішення проблем динаміки рідин. Однак звичайні PINN стикаються з властивими обмеженнями при моделюванні нестисливих рідин, такими як труднощі у виборі точок відбору проб, балансуванні елементів втрат та оптимізації гіперпараметрів. Ці обмеження часто призводять до поганої збіжності PINN. Щоб подолати ці проблеми, у [23] пропонується покращений і загальний PINN для аналізу рідинної динаміки. Цей підхід включає три ключові вдосконалення: адаптивний відбір спостережень з набору даних на основі залишків, який автоматично відбирає спостереження в областях із більшими залишками; адаптивні ваги; використання алгоритму оптимізації диференціальної еволюції. Результати моделювання демонструють хорошу узгодженість як з аналітичними рішеннями, так і з результатами порівняльного обчислення засобами обчислювальної гідродинаміки.

У [24] запропоновано покращений метод глибокого навчання для моделювання солітонного розв’язку рівняння Хакслі.

Бібліотека розв’язання диференціальних рівнянь DeepXDE, яка є Python реалізацією підходу на основі нейронних мереж з фізичною інформацією розглядається в статті [25]. Пропонується метод адаптивного уточнення на основі залишків. Результати моделювання порівнюються з методом скінченних елементів. Розглядаються задачі апроксимації заданої функції нейронною мережею, розв’язання звичайних диференціальних рівнянь та рівнянь у частинних похідних, а також обернена проблема диференціальних рівнянь в частинних похідних. Бібліотека широко застосовується у наукових дослідженнях, зокрема в [26] фізично-інформовані нейронні мережі застосовуються в задачах оптимізації.

В [27] мережі з додатковою фізичною інформацією застосовуються для розробки системи неперервного моніторингу стану механічної системи на основі даних. Розв’язується задача прогину балки Ейлера-Бернуллі. Розглядається розподілене поперечне та точкове навантаження. Результати порівнюються з аналітичним та скінченно-елементним розв’язками, продемонстрована задовільна збіжність нейромережевого методу.

Роботу [28] присвячено адаптації нейронних мереж на основі фізики для визначення вертикальних переміщень і кутів закручування балок з функціонально-градієнтних матеріалів. Особливість використання PINN в цій роботі полягає у використанні енергетичного підходу. Метод тестується на матеріалі з різними за напрямами властивостями. Наведено кілька чисельних прикладів, які показують гарну узгодженість із розв’язками закритої форми.

В статті [29] представлено застосування PINN для аналізу вигину та вільної вібрації тривимірних пористих балок з функціонально градієнтного матеріалу. Припускається, що властивості матеріалу балки безперервно змінюються в трьох вимірах відповідно до довільної функції. Основні рівняння руху отримані з використанням принципу Гамільтона та розв’язані за допомогою обчислювального підходу PINN. Прогин балки апроксимується за допомогою глибокої нейронної мережі прямого поширення сигналу, вхідною інформацією якої є просторова координата. Параметри мережі навчаються шляхом мінімізації функції втрат, що складається з керівного диференціального рівняння та граничних умов. Власна частота балки розглядається як невідомий параметр у керівному рівнянні; таким чином, його потрібно отримати, розв’язуючи обернену задачу. Ця процедура дозволяє знаходити вищі власні частоти мод, що неможливо за попередніми методами PINN.

Метод найменших квадратів у контексті методології PINN розглядається в роботі [30]. Ключова ідея підходу полягає в тому, щоб перетворити диференціальне рівняння вищого порядку в систему рівнянь нижчого порядку за допомогою додаткових змінних, тоді функція втрат, що складається з інтегралів відповідних квадратів залишків над областю визначення невідомої функції,

мінімізується. Запропонований підхід демонструє переваги порівняно з оригінальним методом PINN з точки зору точності розв’язування, обчислювальної вартості для кількох задач згинання балки, що мають неперервні та розривні розв’язки.

В статті [31] пропонується модель PINN-Stress, яка дозволяє зменшити витрати на обчислення, зберігаючи точність для задач визначення розподілу напруги на основі моделювання кінцевих елементів та з використанням розв’язуючих рівнянь у частинних похідних Використовуючи автоматичне диференціювання, диференціальне рівняння вбудовується у функцію втрат глибокої нейронної мережі. Модель PINN-Stress може прогнозувати послідовність розподілу напруг майже в реальному часі та може узагальнювати краще, ніж модель без PINN.

Неоднорідна балка, що спирається на пружну основу та піддається довільному зовнішньому навантаженню досліджується в [32]. Підхід з використанням нейромереж має на меті передбачити не лише саму шукану функцію, але й її похідні вищих порядків. У контексті цієї роботи основною змінною є прогин балки, тоді як її похідні вищого порядку пов’язані з напругою зсуву та моментом балки.

В [33] розглядається задача нелінійного згину тривимірних функціонально-градієнтних балок, що спирається на основу Вінклера-Пастернака, використовується платформа глибокого навчання TensorFlow спільно з бібліотекою DeepXDE для проєктування мережі.

Статтю [34] присвячено застосуванню PINN у поєднанні з функціями напружень Ейрі та рядами Фур’є для пошуку оптимальних розв’язків кількох еталонних бігармонічних задач пружності. В роботі виявлено, що розширення простору ознак за допомогою функцій напружень Ейрі може значно підвищити точність розв’язання за допомогою PINN для бігармонічних диференціальних рівнянь.

В [35] представлено нову методологію для моделювання динаміки балок на пружних основах. Зокрема, моделі балок Ейлера-Бернуллі та Тимошенка на основі

Вінклера моделюються з використанням підходу PINN з урахуванням причинності. Звичайні PINN стикаються з проблемами при обробці великих просторово-часових областей, навіть для проблем із аналітичними розв’язками закритої форми. Для подолання цього обмеження використовується функція втрат PINN, яка враховує причинно-наслідковий зв’язок та ефективно описує базову фізику.

Фреймворк створення нейронної мережі з фізичною інформацією пропонується в [36] для аналізу поведінки нелінійного вигину тривимірної пористої тонкої балки з функціонально градієнтного матеріалу, яка спирається на основу Вінклера-Пастернака.

У [37] пропонується новий метод PINN на основі сітки, який називається M-PINN, який базується на ідеях методу скінченних елементів. Розбиваючи область рішення на кілька субдоменів і включаючи обмеження розподілу даних скінченних елементів, підхід M-PINN ефективно зменшує труднощі оптимізації звичайних PINN. Крім того, іноді важко безпосередньо отримати точні граничні умови в деяких практичних застосуваннях. Цей метод може бути використаний для вирішення задач PINN з невідомими граничними умовами, таким чином маючи більш широку застосовність.

Робота [38] містить опис метода глибокої колокації. Продуктивність цього методу залежить від архітектури нейронної мережі та відповідних гіперпараметрів. Представлений підхід не містить сіток і уникає будь-якої просторової дискретності, яка зазвичай необхідна для методу скінченних елементів. Продемонстровано стійкість розв’язку, що був отриманий за даним методом без необхідності генерувати дані за допомогою інших чисельних методів, таких як метод скінченних елементів.

У статті [39] використовуються глибокі згорткові мережі як альтернатива мережам прямого поширення сигналу при розв’язанні задач на власні значення.

В [40] пропонується поетапна нейронна мережа з фізичною інформацією (sPINN) для розв’язання задач деформації гіпопружних матеріалів. Весь процес sPINN можна розділити на ряд часових кроків. На кожному часовому етапі

визначальне рівняння швидкості, виражене алгоритмом Хьюза-Вінгета, і рівняння, що регулює імпульс, включені до функції втрат як фізичні обмеження. Поля переміщення та напруги можна визначити, завершивши процес навчання кожного кроку часу.

В роботі [41] пропонується метод напіваналітичної PINN мережі для розв’язання сингулярно збурених крайових задач. Чисельні експерименти охоплюють широкий спектр лінійних і нелінійних диференціальних рівнянь із сингулярно збуреними змінами.

У [42] розроблено багатоканальні PINN для навчання представлення, щоб в повній мірі використовувати розріджені точки даних.

Еластографія зсувної хвилі (Shear wave elastography, SWE) дозволяє вимірювати пружні властивості м’яких матеріалів неінвазивним способом і знаходить широке застосування в різних дисциплінах. Сучасні методи SWE покладаються на вимірювання швидкості локальної зсувної хвилі для визначення параметрів матеріалу та страждають від дифракції хвилі при застосуванні до м’яких матеріалів із сильною неоднорідністю. У [43] запропоновано метод SWE на основі нейронної мережі (SWENet). Просторова зміна пружних властивостей неоднорідних матеріалів була введена в керівні рівняння, які закодовані в SWENet як функції втрат. Миттєві знімки хвильових рухів використовувалися для навчання нейронних мереж, і під час цього курсу одночасно виводилися пружні властивості в межах проблемної області, освітленої зсувними хвилями.

В [44] запропоновано алгоритм BCMO-ANN для оптимізації вібрації та вигину функціонально-градієнтних пористих мікропластин. Теорія базується на єдиній структурі теорії деформації зсуву вищого порядку та модифікованої теорії парних напруг. Комбінація штучної нейронної мережі і оптимізації балансуючого композитного руху (Balancing Composite Motion Optimization, BCMO) розроблена для розв’язання задач оптимізації та прогнозування стохастичної вібрації та прогину функціонально-градієнтних пористих мікропластин із невизначеністю властивостей матеріалу.

В [45] запропоновано нейромережевий варіант енергетичного методу розв’язання задач вигину та вільних коливань нерегулярних пластин Кірхгофа.

В роботі [46] розглядається керований даними розв’язок прямих та обернених задачі для рівняння Хіроти зі змінними коефіцієнтами. Використовується варіант мереж PINN з локальними функціями активації. Результати, отримані в цій роботі, підтверджують, що прямі та обернені проблеми, включаючи виявлення керованої даними функції рівняння зі змінними коефіцієнтами, можуть бути розв’язані на основі глибокого навчання.

Статтю [47] присвячено розв’язанню прямих та обернених проблем хаотичних систем, пов’язаних із системами Лоренца та Ресслера, за допомогою двох алгоритмів машинного навчання, а саме нейронних операторів Фур’є.

В роботах [48-49] PINN мережі використовуються для розв’язання задачі дифузії [48], параметричних диференціальних рівнянь [49].

В роботах [50, 51] PINN мережі використовуються для розв’язання обернених задач рівняння Бюргерса.

В [52] представлено нейромережу з невеликою похибкою узагальнення, мережу глибокого оператора (DeepONet), яка складається з глибинної нейромережі для кодування простору дискретних вхідних функцій (мережі розгалужень) та іншої нейромережі для кодування області вихідних функцій (магістральна мережа). Продемонстровано, що DeepONet може вивчати різні явні оператори, такі як інтеграли та дробові лапласіани, а також неявні оператори, які представляють детерміновані та стохастичні диференціальні рівняння.

В [53] представлено огляд DeepONet, нейронного оператора Фур’є та графового нейронного оператора, а також відповідних розширень із розширенням функцій, проаналізовано їх корисність у різноманітних додатках у обчислювальній механіці, включаючи пористі середовища, механіку рідини та механіку твердого тіла.

У роботі [54] розширено класичне формулювання DeepONets, введено моделі послідовного навчання та блоки GRU та LSTM, щоб забезпечити точні

прогнози контурних графіків розв’язків за параметричними входами. Запропонована модель отримала назву послідовні DeepONets (S-DeepONets).

У [55] розглянуто Phase-Field DeepONet, фреймворк для створення нейронних мереж оператора з фізичною інформацією, який використовується для прогнозування динамічних реакцій систем, керованих градієнтними потоками функціоналів вільної енергії. Приклади, використані для підтвердження здійсненності та точності методу, включають рівняння Аллена-Кана та Кана-Хілліарда, як окремі випадки моделей реактивного фазового поля для нерівноважної термодинаміки хімічних сумішей.

Фізично-інформовані нейронні мережі показали себе ефективними інструментами для вирішення як прямих, так і обернених задач рівнянь у частинних похідних. PINN вбудовують PDE у помилку нейронної мережі за допомогою автоматичного диференціювання. Згодом, ця функція помилки PDE оцінюється на наборі розсіяних просторово-часових точок (так звані залишки або нев’язки). Розташування та розподіл цих залишкових точок дуже важливі для продуктивності PINN. Проте, в існуючих дослідженнях PINN в основному використовувалося лише кілька простих методів вибірки залишкових точок. У [56] представлено комплексне дослідження двох категорій вибірки для PINN: неадаптивної рівномірної вибірки та адаптивної нерівномірної вибірки. Розглянуто шість однорідних методів вибірки, включаючи рівномірну сітку, рівномірну випадкову вибірку, латинський гіперкуб, послідовність Халтона, послідовність Хаммерслі та послідовність Соболя. Розглянуто стратегію повторної вибірки для однорідних залишкових точок. Щоб підвищити ефективність вибірки та точність PINN, запропоновано два нові методи адаптивної вибірки точок: адаптивний розподіл на основі залишків (RAD) і адаптивне уточнення на основі залишків із розподілом (RAR-D), які динамічно покращують розподіл залишкових точок на основі значень помилок прогнозування PDE під час навчання.

У [57] наведено, що недолік першого покоління PINN полягає в тому, що вони зазвичай мають обмежену точність навіть з великою кількістю точок

навчання. В роботі запропоновано метод градієнтно-посилених фізично-інформованих нейронних мереж (gPINN) для підвищення точності PINN. gPINN використовують інформацію про градієнт залишку PDE та вбудовують градієнт у функцію втрат.

Роботу [58] присвячено інверсному дизайну, що виникає в різних областях техніки, таких як акустика, механіка, тепловий/електронний транспорт, електромагнетизм і оптика. Оптимізація топології є важливою формою інверсного проєктування, де оптимізують спроєктовану геометрію для досягнення цільових властивостей, параметризованих матеріалами в кожній точці проєктної області. Ця оптимізація є складною, оскільки вона має дуже високу розмірність і зазвичай обмежена рівняннями в частинних похідних (PDE) і додатковими нерівностями. У роботі [58] запропоновано новий метод глибокого навчання – фізичні нейронні мережі з жорсткими обмеженнями (hPINN) – для оптимізації топології. hPINN використовує останні розробки PINN для розв’язування PDE. Тому для навчання не потрібен великий набір даних додатково створений чисельними розв’язувачами PDE. Однак, усі обмеження в PINN є м’якими обмеженнями, і тому додатково накладаються жорсткі обмеження за допомогою методу штрафу та розширеного методу Лагранжа. Продемонстровано ефективність hPINN для задачі голографії в оптиці та проблеми потоку рідини Стокса.

Стаття [59] розглядає недоліки PINN, зокрема, брак кількісної оцінки невизначеності прогнозованого розв’язку, яка може виникати через притаманну даним випадковість (параметрична невизначеність). Іншим джерелом невизначеності є випадковість у архітектурі нейромереж (невизначеність наближення). Для врахування цих двох типів пропонується моделювати, наприклад, коефіцієнти диференціальних рівнянь як стохастичні процеси.

У [60] зазначено, що PINN мережі демонструють надзвичайну перспективу в інтеграції фізичних моделей із даними спостережень із пропусками та шумами, але вони все ще мають проблеми у випадках, коли цільові функції, які потрібно апроксимувати, демонструють високочастотні або багатомасштабні характеристики. В роботі досліджено це обмеження засобами теорії нейронного

дотичного ядра (Neural tangent kernel, NTK), розглянуто вплив цієї теорії на функції навчання PINN вздовж домінуючих власних напрямків їх обмежувальної NTK. Створено нові архітектури, які використовують просторово-часові та багатомасштабні випадкові функції Фур’є та обґрунтовано, як такі шари вбудовування координат можуть призвести до надійних і точних моделей PINN.

Роботу [61] присвячено розробці глибокої операторної мережі Фур’є (Fourier-DeepONet) для розв’язання задачі інверсії форми хвилі з узагальненням сейсмічних джерел, включаючи частоти та розташування джерел.

Перспективи використання нейронного дотичного ядра при вивченні механізму навчання PINN мереж методом градієнтного спуску досліджено у [62]. Нейронне дотичне ядро дає змогу фіксувати поведінку повнозв’язаних нейронних мереж під час навчання за допомогою градієнтного спуску.

У статті [63] пропонується використання дискретної структури PINN мережі, заснованої на графовій згортковій мережі (Graph Convolutional Networks, GCN) і варіаційній структурі часткових диференціальних рівнянь, щоб розв’язувати прямі та обернені задачі єдиним способом. Використання кусково-поліноміального базису може зменшити розмірність простору пошуку та полегшити навчання та збіжність мережі.

Архітектура згорткових нейромереж з фізичними обмеженнями, спрямована на вивчення розв’язків параметричних диференціальних рівнянь в нерегулярних областях без будь-яких позначених даних, пропонується у [64]. Для того, щоб використовувати класичні перетворення згорткових мереж, введено еліптичне відображення координат, щоб уможливити перетворення координат між нерегулярною фізичною областю та звичайною еталонною областю.

Роботу [65] присвячено розгляду рівняння Больцмана з моделлю зіткнення Бхатнагара-Гросса-Крука (рівняння Больцмана-БГК), яке широко використовується для опису багатомасштабних потоків від гідродинамічної мережі до вільного молекулярного потоку. Використано нейронні мережі з фізичною інформацією для розв’язання прямих і обернених проблем за допомогою формулювання Больцмана-BGK (PINN-BGK), що дозволяє PINN

моделювати потоки як у континуальному, так і в розрідженому режимах. Зокрема, PINN-BGK складається з трьох підмереж, де перша використовується для апроксимації рівноважної функції розподілу, друга – для апроксимації нерівноважної функції розподілу, а третя – для кодування рівняння Больцмана-BGK.

Використання нейронних мереж з фізичною інформацією апроксимації рівнянь Ейлера, які моделюють високошвидкісні аеродинамічні потоки досліджено у [66]. Зокрема, розв’язано прямі та обернені задачі в одновимірній та двовимірній областях. Для прямої задачі використовується рівняння Ейлера та початкові/граничні умови для формулювання функції втрат, розв’язано одновимірні рівняння Ейлера з гладкими розв’язками та з розв’язками, які мають контактний розрив. Продемонстровано, що можна отримати стійкий розв’язок лише за допомогою розподілених точок, випадковим чином згрупованих навколо розривів.

У [67] пропонується консервативна нейронна мережа на основі фізики (cPINN) у дискретних областях для нелінійних законів збереження. Тут термін «дискретна область» означає дискретні підобласті, отримані після поділу обчислювальної області, де застосовано PINN, а властивість збереження cPINN досягається шляхом забезпечення безперервності потоку в сильній формі вздовж інтерфейсів піддоменів. У випадку гіперболічних законів збереження конвективний потік вносить внесок на межі розділу, тоді як у випадку в’язких законів збереження внески вносять як конвективний, так і дифузійний потоки.

У розглянутих роботах, основними методами розв’язання задач з різних прикладних областей є методи Physics-Informed Neural Networks (PINN) та Deep Operator Networks (DeepONet). Вони використовують нейронні мережі для розв’язання диференціальних рівнянь, але мають різні базові ідеї. PINN інтегрує фізичні закони безпосередньо у функцію втрат нейронної мережі, використовуючи інформацію про фізичні обмеження та граничні умови для навчання моделі. Цей підхід добре підходить для задач, де необхідно розв’язати конкретне диференціальне рівняння на конкретній області з визначеними граничними

умовами, і може розв’язувати як прямі, так і обернені задачі. DeepONet, натомість, розроблено для навчання операторів, що відображають функції на функції. Замість того, щоб розв’язувати диференціальне рівняння безпосередньо, DeepONet навчається апроксимувати оператор, який розв’язує ці рівняння для будь-якої вхідної функції. Цей підхід особливо корисний, коли потрібно швидко обчислювати розв’язки для різних початкових або граничних умов, або для різних параметрів системи. Основна відмінність між цими методами полягає у підході до розв’язання диференціальних рівнянь: PINN включає диференціальне рівняння у функцію втрат для навчання конкретної задачі, тоді як DeepONet навчає загальний оператор для розв’язання сімейства задач. Таким чином, PINN є потужним інструментом для задач з відомими фізичними законами, тоді як DeepONet забезпечує гнучкість і швидкість для задач з широким спектром вхідних умов.

1.2 Огляд бібліотек нейромережевих методів розв’язання крайових задач

Бібліотека DeepXDE [25] є відкритою, програмний код знаходиться за посиланням https://github.com/lululxvi/deepxde. Саме ця бібліотека часто використовується дослідниками для розв’язання математичних моделей різних типів із використанням PINN мереж. Офіційна документація містить посилання на дослідницькі публікації, які використовують DeepXDE https://deepxde.readthedocs.io/en/latest/user/research.html. Бібліотека містить набір засобів для розв’язання прямих та обернених задач диференціальних та інтегро-диференціальних рівнянь, дробових диференціальних рівнянь [68]. Також реалізовано глибинні нейромережеві оператори (DeepONet) [52] та нейромережеві оператори Фур’є [61]. Використовуються методи покращення збіжності, зокрема, адаптивний вибір пробних точок даних (точок колокації) [56], градієнтний метод [57], ознаки Фур’є [60]. Бібліотека підтримує можливість моделювати складні

форми областей визначення функцій засобами конструктивної твердотільної геометрії.

З точки зору користувача, у порівнянні з традиційними чисельними методами код, написаний за допомогою DeepXDE, є набагато коротшим і більш комплексним, дуже нагадуючи математичне формулювання. Розв’язування диференціальних рівнянь у DeepXDE зводиться до специфікації задачі за допомогою вбудованих модулів, де необхідно визначити обчислювальну область (геометрія та час), диференціальне рівняння, граничні/початкові умови, обмеження, дані навчання, архітектуру нейронної мережі та гіперпараметри навчання. Слід зазначити, що використання бібліотеки потребує знання мов програмування та, що найголовніше, дотримання певного внутрішнього формату запису задачі.

Так, наприклад, нехай задано диференціальне рівняння

$$\frac{d^2y(t)}{dt^2} - 10\frac{dy(t)}{dt} + 9y(t) = 5t$$

з початковими умовами

$$y(0) = -1, \frac{dy}{dt}(0) = 2$$

та для області $t \in [0, 0.25]$.

Тоді, для його розв’язання засобами бібліотеки DeepXDE необхідно виконати такий програмний код (відповідно до офіційної документації https://deepxde.readthedocs.io/en/latest/demos/pinn_forward/ode.2nd.html):

Відповідно до рис. 1.1 необхідно визначити Python функції самого диференціального рівняння, граничних та початкових умов (функції ode(), func(), boundary_l(), bc_func1(), bc_func2()) та поєднати всі дані за допомогою dde.data.TimePDE(). Після чого необхідно визначити параметри нейромережі засобами функції dde.nn.FNN() та створити розрахункову задачу за допомогою dde.Model().

```
import deepxde as dde
import numpy as np
def ode(t, y):
    dy_dt = dde.grad.jacobian(y, t)
    d2y_dt2 = dde.grad.hessian(y, t)
    return d2y_dt2 - 10 * dy_dt + 9 * y - 5 * t
def func(t):
    return 50 / 81 + t * 5 / 9 - 2 * np.exp(t) + (31 / 81) * np.exp(9 * t)
geom = dde.geometry.TimeDomain(0, 0.25)
def boundary_l(t, on_initial):
    return on_initial and dde.utils.isclose(t[0], 0)
def bc_func1(inputs, outputs, X):
    return outputs + 1
def bc_func2(inputs, outputs, X):
    return dde.grad.jacobian(outputs, inputs, i=0, j=None) - 2
ic1 = dde.icbc.IC(geom, lambda x: -1, lambda _, on_initial: on_initial)
ic2 = dde.icbc.OperatorBC(geom, bc_func2, boundary_l)
data = dde.data.TimePDE(geom, ode, [ic1, ic2], 16, 2, solution=func, num_test=500)
layer_size = [1] + [50] * 3 + [1]
activation = "tanh"
initializer = "Glorot uniform"
net = dde.nn.FNN(layer_size, activation, initializer)
model = dde.Model(data, net)
model.compile(
    "adam", lr=0.001, metrics=["l2 relative error"], loss_weights=[0.01, 1, 1]
)
losshistory, train_state = model.train(iterations=10000)
dde.saveplot(losshistory, train_state, issave=True, isplot=True)
```

Рисунок 1.1 – Приклад застосування бібліотеки DeepXDE

Отже, як можна побачити з наведеного прикладу, коректне застосування бібліотеки DeepXDE потребує ґрунтовних знань з програмування та розуміння загальної архітектури бібліотеки.

Бібліотека NeuralPDE [69] (https://github.com/SciML/NeuralPDE.jl) реалізована на мові програмування Julia. Відмінність від інших бібліотек PINN, наприклад, DeepXDE, полягає в тому, що вони, зазвичай, вимагають від користувача написання низькорівневого формулювання рівняння в частинних похідних та крайових умов. NeuralPDE абстрагує реалізацію від структури за допомогою символьного інтерфейсу (визначеного через Symbolics.jl). Результатом є те, що запис задачі дуже нагадує математичну постановку. Наприклад, розв’язок рівняння

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\sin(\pi x)\sin(\pi y)$$

з початковими та граничними умовами:

$$\begin{aligned} u(0, y) &= 0, \\ u(1, y) &= -\sin(\pi)\sin(\pi y), \\ u(x, 0) &= 0, \\ u(x, 1) &= -\sin(\pi x)\sin(\pi) \end{aligned}$$

наведено на рисунку 1.2 [69].

```
using NeuralPDE, Lux, ModelingToolkit, Optimization, OptimizationOptimisers
import ModelingToolkit: Interval, infimum, supremum
@parameters x y
@variables u(..)
Dxx = Differential(x)^2
Dyy = Differential(y)^2
# 2D PDE
eq = Dxx(u(x, y)) + Dyy(u(x, y)) ~ -sin(pi * x) * sin(pi * y)
# Boundary conditions
bcs = [u(0, y) ~ 0.0, u(1, y) ~ 0,
   u(x, 0) ~ 0.0, u(x, 1) ~ 0]
# Space and time domains
domains = [x ∈ Interval(0.0, 1.0),
   y ∈ Interval(0.0, 1.0)]
# Discretization
dx = 0.1
# Neural network
dim = 2 # number of dimensions
chain = Lux.Chain(Dense(dim, 16, Lux.σ), Dense(16, 16, Lux.σ), Dense(16, 1))
discretization = PhysicsInformedNN(chain, QuadratureTraining())
@named pde_system = PDESystem(eq, bcs, domains, [x, y], [u(x, y)])
prob = discretize(pde_system, discretization)
callback = function (p, l)
   println("Current loss is: $l")
   return false
end
res = Optimization.solve(prob, ADAM(0.1); callback = callback, maxiters = 4000)
prob = remake(prob, u0 = res.minimizer)
res = Optimization.solve(prob, ADAM(0.01); callback = callback, maxiters = 2000)
phi = discretization.phi
```

Рисунок 1.2 – Приклад застосування бібліотеки NeuralPDE

Як можна побачити, бібліотеки NeuralPDE та DeepXDE використовують власні функції для операцій диференціювання, що також потребує детального попереднього ознайомлення з документацією бібліотек та може бути перешкодою для наукових співробітників та інженерів.

Бібліотека SimNet (Nvidia Modulus) [70] є потужним засобом моделювання мультифізичних процесів. Підтримує роботу з конструктивною твердотільною геометрією, масивами точок, STL файлами для визначення геометрії області визначення задач. Є можливість безпосередньої роботи з графічними процесорами засобами бібліотеки (рис. 1.3).

Бібліотека SciANN [71] (https://github.com/sciann/sciann) використовує широко використовувані пакети глибокого навчання TensorFlow і Keras для побудови глибоких нейронних мереж і оптимізаційних моделей, таким чином успадковуючи багато функціональних можливостей Keras, наприклад, пакетну оптимізацію та повторне використання моделі для переносу навчання (Transfer Learning). SciANN розроблено для абстрактної побудови нейронної мережі для наукових обчислень і розв’язання та виявлення диференціальних рівнянь із частинними похідними за допомогою архітектури нейронних мереж з фізичними відомостями, що забезпечує гнучкість налаштування складних функціональних форм.

Розглянемо процес розв’язання рівняння Бюргерса засобами SciANN. Рівняння має виглядає

$$u_{,t} + uu_{,x} - (0.01/\pi)u_{,xx} = 0, \quad t \in [0, 1], \quad x \in [-1, 1].$$

Крайові умови:

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

```
def run(cfg: ModulusConfig) -> None:
    we = WaveEquation1D(c=1.0)
    wave_net = instantiate_arch(
        input_keys=[Key("x"), Key("t")],
        output_keys=[Key("u")],
        cfg=cfg.arch.fully_connected,
    )
    nodes = we.make_nodes() + [wave_net.make_node(name="wave_network")]
    x_symbol, t_symbol = Symbol("x"), Symbol("t")
    L = float(np.pi)
    geo = Line1D(0,L)
    time_range = {t_symbol: (0,2*L)}
    domain = Domain()
    IC = PointwiseInteriorConstraint(
        nodes = nodes,
        geometry = geo,
        outvar = {"u": sin(x_symbol), "u__t" : sin(x_symbol)},
        batch_size = cfg.batch_size.IC,
        lambda_weighting = {"u":1.0, "u__t":1.0},
        parameterization = {t_symbol: 0.0},
    )
    domain.add_constraint(IC, "IC")
    BC = PointwiseBoundaryConstraint(
        nodes = nodes,
        geometry = geo,
        outvar = {"u": 0},
        batch_size = cfg.batch_size.BC,
        parameterization = time_range,
    )
    domain.add_constraint(BC, "BC")
    interior = PointwiseInteriorConstraint(
        nodes = nodes,
        geometry = geo,
        outvar = {"wave_equation": 0},
        batch_size = cfg.batch_size.interior,
        parameterization = time_range,
    )
    domain.add_constraint(interior, "interior")
    deltaT = 0.01
    deltaX = 0.01
    x = np.arange(0,L, deltaX)
    t = np.arange(0,2*L, deltaT)
    X,T = np.meshgrid(x,t)
    X = np.expand_dims(X.flatten(), axis = -1)
    T = np.expand_dims(T.flatten(), axis = -1)
    u = np.sin(X) * (np.cos(T) + np.sin(T))
    invar_numpy = {"x": X, "t":T}
    outvar_numpy = {"u": u}
    validator = PointwiseValidator (
        nodes = nodes, invar = invar_numpy, true_outvar = outvar_numpy, batch_size = 128
    )
    domain.add_validator(validator)
    slv = Solver(cfg, domain)
    slv.solve()
if __name__ =="__main__":
    run()
```

Рисунок 1.3 – Приклад застосування бібліотеки Nvidia Modulus

На рисунку 1.4 наведено програмний код розв’язку засобами бібліотеки SciANN.

```
import numpy as np
import matplotlib.pyplot as plt
import sciann as sn

x = sn.Variable('x')
t = sn.Variable('t')
u = sn.Functional('u', [t,x], 8*[20], 'tanh')

L1 = diff(u, t) + u*diff(u,x) - (0.01/pi)*diff(u, x, order=2)
TOL = 0.001
C1 = (1-sign(t - TOL)) * (u + sin(pi*x))
C2 = (1-sign(x - (-1+TOL))) * (u)
C3 = (1+sign(x - ( 1-TOL))) * (u)

m = sn.SciModel([x, t], [L1, C1, C2, C3])

x_data, t_data = np.meshgrid(
    np.linspace(-1, 1, 100),
    np.linspace(0, 1, 100)
)

h = m.train([x_data, t_data], 4*['zero'], learning_rate=0.002, epochs=5000, verbose=0)

x_test, t_test = np.meshgrid(
    np.linspace(-1, 1, 200),
    np.linspace(0, 1, 200)
)
u_pred = u.eval(m, [x_test, t_test])

fig = plt.figure(figsize=(3, 4))
plt.pcolor(x_test, t_test, u_pred, cmap='seismic')
plt.xlabel('x')
plt.ylabel('t')
plt.colorbar()
```

Рисунок 1.4 – Приклад застосування бібліотеки SciANN

На відміну від попередніх розглянутих бібліотек, процес опису задачі засобами SciANN є символьним та не потребує написання програмного коду. Це дозволяє користувачеві використовувати синтаксис, схожий на поширені системи комп’ютерної алгебри. Однак, перевагою Nvidia Modulus є орієнтація на задачі, що вимагають високої продуктивності та використовують GPU, а також підтримка

широкого кола фізичних задач. В той же час, DeepXDE підтримує різні фреймворки та підходить для широкого спектра диференціальних рівнянь.

Бібліотека PINNs-Torch [72] (https://github.com/rezaakb/pinns-torch) представляє нейронні мережі з фізичними даними, реалізовані за допомогою PyTorch. Відмінною функцією є використання графів CUDA та компіляторів JIT (TorchScript) для компіляції моделей, що призводить до збільшення продуктивності у порівнянні з використанням TensorFlow. Як приклад також розглянемо розв’язання рівняння Бюргерса:

$$u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx} = 0, x \in [-1, 1], t \in [0, 1]$$

з початковими та граничними умовами:

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

Програмний код розв’язку наведено на рисунку 1.5.

Для дослідників, таких як інженери та математики, важливо мати простий інтерфейс роботи з бібліотекою розв’язання крайових задач засобами PINN з кількох причин. По-перше, простий і інтуїтивно зрозумілий інтерфейс зменшує поріг входження, дозволяючи фахівцям з різним рівнем програмування швидко почати використовувати бібліотеку для своїх задач. По-друге, це дозволяє більше часу приділяти безпосередньому розв’язанню наукових і технічних проблем, а не вивченню складної документації та налаштувань. По-третє, простий інтерфейс сприяє більшій продуктивності та ефективності, що особливо важливо в академічних і промислових дослідженнях, де швидкість розробки і точність рішень є критичними.

Розглянуті бібліотеки DeepXDE та SciANN надають простий і інтуїтивно зрозумілий інтерфейс для користувачів. NeuralPDE і PINNs-Torch використовують безпосередньо екосистему мови програмування Julia та PyTorch відповідно. Nvidia

Modulus оптимізована для GPU, що робить цю бібліотеку придатною для високопродуктивних обчислень і візуалізації.

```
from typing import Any, Dict, List, Optional, Tuple
import hydra
import numpy as np
import rootutils
import torch
from omegaconf import DictConfig
import pinnstorch
def read_data_fn(root_path):
    """Read and preprocess data from the specified root path.
    :param root_path: The root directory containing the data.
    :return: Processed data will be used in Mesh class.
    """
    data = pinnstorch.utils.load_data(root_path, "burgers_shock.mat")
    exact_u = np.real(data["usol"])
    return {"u": exact_u}
def pde_fn(outputs: Dict[str, torch.Tensor],
           x: torch.Tensor,
           t: torch.Tensor):
    """Define the partial differential equations (PDEs)."""
    u_x, u_t = pinnstorch.utils.gradient(outputs["u"], [x, t])
    u_xx = pinnstorch.utils.gradient(u_x, x)[0]
    outputs["f"] = u_t + outputs["u"] * u_x - (0.01 / np.pi) * u_xx
    return outputs
@hydra.main(version_base="1.3", config_path="configs", config_name="config.yaml")
def main(cfg: DictConfig) -> Optional[float]:
    """Main entry point for training.
    :param cfg: DictConfig configuration composed by Hydra.
    :return: Optional[float] with optimized metric value.
    """
    # apply extra utilities
    # (e.g. ask for tags if none are provided in cfg, print cfg tree, etc.)
    pinnstorch.utils.extras(cfg)
    # train the model
    metric_dict, _ = pinnstorch.train(
        cfg, read_data_fn=read_data_fn, pde_fn=pde_fn, output_fn=None
    )
    # safely retrieve metric value for hydra-based hyperparameter optimization
    metric_value = pinnstorch.utils.get_metric_value(
        metric_dict=metric_dict, metric_names=cfg.get("optimized_metric")
    )
    # return optimized metric
    return metric_value
if __name__ == "__main__":
    main()
```

Рисунок 1.5 – Приклад застосування бібліотеки PINNs-Torch

З порівняння форми запису тестових задач бібліотек DeepXDE, NeuralPDE, Nvidia Modulus, SciANN та PINNs-Torch можна зробити висновок, що формат запису задачі у символьному вигляді засобами NeuralPDE більш наближений до

математичної постановки, однак, все ще потребує знання внутрішнього формату запису.

Отже, розробка бібліотеки для реалізації обчислювальних методів нейронних мереж з фізичною інформацією із зручним та інтуїтивно зрозумілим командним інтерфейсом є актуальною задачею.

Стандартизовані інструменти та бібліотеки сприятимуть швидкому створенню і тестуванню моделей. Це значно прискорює наукові дослідження та промислове впровадження, дозволяючи розробникам фокусуватися на основних аспектах своїх задач, а не на вирішенні технічних деталей.

Крім того, використання спільних бібліотек підвищує відтворюваність досліджень. Це важливий аспект наукової роботи, оскільки дозволяє іншим дослідникам відтворювати та перевіряти результати, що підвищує довіру до отриманих даних і сприяє науковому прогресу.

Також варто зазначити, що спільнота розробників, яка працює з однією бібліотекою, може ділитися досвідом, розширювати функціональність та швидко усувати помилки. Це покращує загальну якість та стабільність інструменту, роблячи його більш надійним і ефективним.

Бібліотека, яка легко інтегрується з існуючими фреймворками машинного навчання, такими як TensorFlow або PyTorch, дозволить ефективніше використовувати існуючі ресурси та інструменти. Це забезпечує ширшу адаптацію та інтеграцію PINN методів у різноманітні проекти та дослідження.

Висновки до розділу 1

Нейронні мережі з фізичною інформацією за останні роки отримали значний розвиток при розв’язанні наукових та технічних проблем. Основні аспекти розвитку методів PINN у літературі включають теоретичне обґрунтування, розробку різних архітектур і алгоритмів оптимізації для

підвищення ефективності та стабільності, а також успішне застосування у різних галузях, таких як механіка рідин, теплопередача, матеріалознавство та біомеханіка. Ці результати свідчать про високу точність і ефективність методів PINN.

Перспективи розвитку методів PINN включають підвищення масштабованості для обробки великих систем диференціальних рівнянь та складних геометрій, інтеграцію з традиційними числовими методами, такими як метод скінченних елементів, для досягнення кращих результатів, а також впровадження методів адаптивного навчання, що дозволяють ефективно використовувати обчислювальні ресурси та зменшувати час навчання.

Бібліотеки для реалізації методів PINN, такі як DeepXDE, SciANN, NeuralPDE, PINNs-Torch та Nvidia Modulus, мають значний потенціал для подальшого розвитку. Вони вже зараз забезпечують високу продуктивність і гнучкість, що робить їх корисними інструментами для вирішення складних наукових і інженерних задач. Майбутні напрямки розвитку включають підвищення масштабованості, інтеграцію з новими методами та інструментами, а також покращення документації та підтримки користувачів. Ці вдосконалення дозволять розширити можливості застосування методів PINN у різних галузях науки і техніки.

Основні науково-практичні результати першого розділу опубліковано в роботах [84-93].

2 ПРОЄКТУВАННЯ БІБЛІОТЕКИ НЕЙРОМЕРЕЖЕВИХ МЕТОДІВ

З огляду бібліотек, що реалізують нейромережеві обчислювальні методи можна зробити висновок, що важливими вимогами є наявність інтуїтивно зрозумілого способу опису задач та можливість роботи з поширеними фреймворками створення нейромереж. В даному розділі приводиться опис розробленої в дисертації предметно-орієнтованої мови PLang (Problem Language), призначеної для формального опису крайових задач.

2.1 Формалізація опису крайової задачі

Використання обчислювальної техніки для автоматизації розв’язання крайових задач потребує наявності формалізованих засобів опису диференціальних рівнянь (систем) та крайових умов. Зазвичай такі описи робляться безпосередньо на мові програмування, за допомогою якої реалізується відповідний алгоритм розв’язання. Проте створення програмного засобу, який дозволяє користувачу не програмісту самостійно описувати задачу, потребує розробки спеціалізованого інструменту для опису задач. Тут може бути два основних напрями: 1) спеціалізований графічний інтерфейс користувача (Graphical User Interface, GUI) [76] або 2) використання спеціалізованих проблемно-орієнтованих мов (DSL – Domain-Specific Language) [77]. Очевидно, що використання GUI є більш зручним для більшості користувачів, проте застосування DSL-мов є більш універсальним і гнучким, оскільки дозволяє описувати задачі довільної складності [77].

2.2 Основні символи мови PLang

Формальний опис DSL-мов потрібен для однозначного й несуперечливого задання синтаксису та семантики мови. Під синтаксисом проблемно-орієнтованої мови розуміється набір правил, за допомогою яких описується структура мови. Відповідно, під семантикою розуміються правила інтерпретації мовних конструкцій. Отже, сукупність синтаксичних правил DSL-мови утворює її формальну граматику.

Опис синтаксису штучних мов частіше за все здійснюється з використанням розширеної нотації Бекуса-Наура (EBNF – Extended Backus–Naur Form), за допомогою якої можна послідовно виразити одні синтаксичні конструкції через інші [78]. Формальний опис основних символів PLang з використанням EBNF можна реалізувати наступним чином.

буква ::= "A"| "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | "_";

цифра ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";

знак ::= "+" | "-";

розділювач ::= "+" | "-" | "*" | "/" | "^" | "(" | ")" | "," | "=" | зарезервоване-слово;

зарезервоване-слово ::= "abs" | "acos" | "asin" | "atan" | "atan2" | "begin" | "constant" | "cos" | "cosh" | "diff" | "equation" | "function" | "problem" | "end" | "exp" | "sin" | "sinh" | "tan" | "tanh";

ідентифікатор ::= буква { буква | цифра };

число-без-знаку ::= ціле-без-знаку | дійсне-без-знаку;

число-зі-знаком ::= [знак] число-без-знаку;

ціле-без-знаку ::= послідовність-цифр;

послідовність-цифр ::= цифра {цифра};

дійсне-без-знаку ::= ціле-без-знаку"."дробова-частина ["E" | "e" порядок] | ціле-без-знаку ["E" | "e" порядок];

дробова-частина ::= послідовність-цифр;

порядок ::= ціле-зі-знаком;

ціле-зі-знаком ::= [знак] ціле-без-знаку;

коментар ::= "#" [ASCII-послідовність];

ASCII-послідовність ::= порожньо | ASCII-символ | ASCII-послідовність ASCII-символ;

порожньо ::= ;

Тут слід зазначити, що під терміном «ідентифікатор» розуміється слово, яке позначає або зарезервоване слово мови PLang, або ім’я змінної (в якості останнього зарезервоване слово використовуватися не може). Також у вище наведеному описі поняття «ASCII-символ» із використанням EBNF формально не описується, так як в цій якості це будь-який з 127 символів таблиці кодування ASCII [79].

2.3 Типи даних PLang

У мові PLang всі змінні належать до дійсного числового типу. Їх довжина визначається апаратною платформою, на якій виконується реалізація цієї мови. У відповідності до EBNF тип даних визначається наступним чином.

тип-даних ::= числовий-тип-даних;

числовий-тип-даних ::= число-без-знаку | число-зі-знаком;

Змінні у мові PLang описуються так.

Декларація-змінної ::= "constant" оператор-присвоювання{, оператор-присвоювання};

оператор-присвоювання ::= ідентифікатор "=" вираз;

Таким чином, змінні у мові PLang декларуються за допомогою оператора “constant”, в якому описується список ідентифікаторів з обов’язковою початковою їх ініціалізацією. В якості ініціалізатора може використовуватися як значення – константа, так і арифметичний вираз (формально буде описано нижче).

Так, наприклад, вираз наступного виду constant E_4 = 2,71828*0.25 задає дійсну змінну E_4 і присвоює їй початкове значення, яке дорівнює результату обчислення відповідного виразу.

2.4 Арифметичні вирази у мові PLang

За допомогою EBNF вирази в PLang можна формально описати наступним чином.

вираз ::= арифметичний-вираз;

арифметичний-вираз ::= константа | змінна | функція | арифметичний-вираз арифметична-операція арифметичний-вираз;

константа ::= число-зі-знаком;

змінна ::= ідентифікатор;

функція ::= ідентифікатор "(" [список-параметрів] ")";

список-параметрів ::= параметр {"," параметр};

параметр ::= арифметичний-вираз | ідентифікатор;

арифметична-операція = "+" | "-" | "*" | "/" | "^";

Вирази призначені для опису формул, які є диференціальними рівняннями та крайовими умовами. У PLang є набір вбудованих функцій, перелік яких наведено в табл. 2.1. Цей перелік може бути розширений, наприклад, спеціальними функціями, які можуть виникати в деяких практичних застосуваннях. Зокрема, гамма-функція використовується у рівнянні Струве.

Таблиця 2.1 – Вбудовані функції мови PLang

№	Функція	Опис
1	abs(x)	Абсолютне значення
2	acos(x)	Арккосинус
3	asin(x)	Арксинус
4	atan(x)	Арктангенс
5	atan2(x,y)	Арктангенс виразу y/x (у радіанах)
6	cos(x)	Косинус
7	cosh(x)	Косинус гіперболічний
8	diff(u, x)	Похідна функції u по x
9	exp(x)	Експонента
10	sin(x)	Синус
11	sinh(x)	Синус гіперболічний
12	tan(x)	Тангенс
13	tanh(x)	Тангенс гіперболічний

Тут слід лише зазначити, що всі функції крім diff() в якості аргументу приймають дійсні числа. Функція diff() першим параметром обов’язково повинна приймати ідентифікатор, який є назвою шуканої функції (тобто оголошений у секції “function”), а другим – ідентифікатор, який обов’язково є аргументом шуканої функції.

2.5 Структура опису крайової задачі мовою PLang

Структуру опису крайової задачі за допомогою проблемно-орієнтованої мови PLang можна засобами EBNF описати таким чином.

Опис-задачі ::= блок { блок };

блок ::= "problem" ідентифікатор EOL початок-блока декларація-функції [декларація-константи] рівняння крайова-умова кінець-блока;

початок-блока ::= "begin" EOL;

кінець-блока ::= "end" EOL;

декларація-функції ::= "function" ідентифікатор-функції "(" список-аргументів ")" EOL;

ідентифікатор-функції ::= ідентифікатор;

список-аргументів ::= ідентифікатор {, ідентифікатор}EOL;

декларація-константи ::= "constant" ідентифікатор "=" арифметичний-вираз {, "constant" ідентифікатор "=" арифметичний-вираз} EOL;

рівняння ::= "equation" "=" арифметичний-вираз EOL;

крайова-умова ::= ідентифікатор-функції "(" ідентифікатор | константа {, ідентифікатор | константа}")" EOL;

Згідно з наведеним вище описом крайова задача формалізується у вигляді секції “problem” (їх можу бути декілька у одному файлі). Кожна така секція складатися з наступних частин:

1) заголовка “problem”, в якому задається назва задачі;

2) програмної дужки “begin”, яка маркує початок блоку опису задачі;

3) оператора декларації шуканої функції “function”, який визначає ім’я функції та список її параметрів;

4) необов’язкового оператора “constant”, який декларує допоміжні константи;

5) оператора “equation”, який визначає рівняння, що власно й утворює крайову задачу;

6) операторів, що визначають крайові умови;

7) програмної дужки “end”, яка маркує кінець блоку опису задачі.

Така структура опису крайової задачі мовою PLang робить її зручною для автоматичної трансляції і підтримує обробку із застосуванням паралельних розрахунків, оскільки кожна секція “problem” може обчислюватися незалежно і паралельно одна від одної, що дасть можливість істотно підвищить загальну швидкість обчислень. Розглянемо далі приклади опису крайових задач із застосуванням мови Plang.

Розглянемо приклад опису рівняння Бюргерса [84] із застосуванням мови PLang. Рівняння описується наступним чином:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{2.1}$$

де $u(x,t)$ – невідома функція (густина газу чи рідини); ν – кінематична в'язкість середовища.

Граничні умови: $u(0,t) = u(2,t) = 0$.

Початкові умови: $u(x,0) = 2\nu\pi\dfrac{\sin(\pi x) + 4\sin(2\pi x)}{4 + \cos(\pi x) + 2\cos(2\pi x)}$.

На мові PLang наведену крайову задачу (2.1) можна описати, наприклад, таким способом.

```
# Рівняння Бюргерса
problem Burgers
begin
   function u(x, t)
   constant v = 18.6, PI = 3.14159

   equation = diff(u, t) + u * diff(u, x) - v * diff(diff(u, x), x)
   u(x, 0) = 2 * v * PI * (sin(PI * x) + 4 * sin(2 * PI * x)) / (4 + cos(PI * x) +
                  2 * cos(2 * PI * x))
   u(0, t) = 0
   u(2, t) = 0
end
```

Рисунок 2.1 – Приклад рівняння Бюргерса

Прогин балки. Задачу про прогин балки у випадку геометричної нелінійності можна описати наступним рівнянням:

$$\frac{d^2 w}{dx^2} = \frac{M(x)}{EI}\left(1 + \left(\frac{dw}{dx}\right)^2\right)^{3/2}, \tag{2.2}$$

де $w(x)$ – шукана функція прогину; M_x – згинальний момент.

Граничні умови: $w(0) = 0, \frac{dw}{dx}(0) = 0.$

На мові PLang задачу (2.2) можна описати так.

```
# Прогин балки
problem Deflection_of_the_beam
begin
   function w(x)
   constant Mx = 1000, E = 1.0E+10, I = 500

   equation = diff(diff(w, x), x) - Mx / (E * I) * (1 + diff(w, x) ^ 2) ^ 1.5
   w(x, 0) = 0
   diff(w, x) = 0
end
```

Рисунок 2.2 – Приклад рівняння згину балки

2.6 Загальний алгоритм трансляції формального опису задачі мовою PLang у клас мови Python

Розглянемо далі загальний алгоритм трансляції формального опису задачі мовою PLang у клас мови Python. Трансляцією називається процес перекладу початкового коду програми, написаної однією мовою програмування, у код на іншій мові програмування. Відповідну програму, що автоматично здійснює такий переклад, називають транслятором [81].

Розрізняють два основних типи трансляторів: компілятори та інтерпретатори. Компілятор (за один чи декілька проходів) перетворює всю вихідну програму в програму на іншій мові, яка є еквівалентною початковій. Інтерпретатор (на відміну від компілятора) послідовно читає початковий код, здійснює його трансляцію в машинну мову (байт-код) та негайно виконує.

Зазвичай код, згенерований компілятором, є більш якісним, оскільки компіляція може виконуватися з оптимізацією. Проте інтерпретатор, на відміну від компілятора, часто є багатоплатформовим, що робить його використання більш доцільним у певних випадках [81].

Для трансляції скриптів на мові PLang у класи на мові Python було реалізовано інтерпретатор. Загальна схема його роботи наведена на рис. 2.3. Вхідною інформацією для транслятора (парсера) є скрипт на мові PLang, в якому описано одну або декілька крайових задач. На першому етапі парсер виконує лексичний і синтаксичний аналіз початкового коду на предмет його коректності. На другому етапі виконується побудова абстрактного синтаксичного дерева (AST – Abstract Syntax Tree) [82] для математичних виразів, що описують крайові задачі.

Транслятор мови PLang було реалізовано з використанням мови програмування Python. UML-модель [83] класів, що реалізують відповідний функціонал, наведена на рис. 2.4. Тут слід прокоментувати наступне: головним класом парсера є Parser, який описує всі необхідні сутності транслятора. У ньому описано низку методів, призначених для розбору скриптів мови PLang та генерації відповідного коду на мові Python. Головною властивістю цього класу є об’єкт problem_list, який містить список екземплярів класу Problem, що інкапсулює поняття крайової задачі. Його властивостями є:

– name – назва задачі;

– function – назва шуканої функції;

– arguments – список аргументів функції;

– constant – список допоміжних констант;

– result – абстрактне синтаксичне дерево (AST).

У цього класу передбачено низку методів, що встановлюють (додають) відповідні значення, а також метод generate(), який безпосередньо запускає генерацію коду (так званий бекенд транслятора).

Рисунок 2.3 – Загальна схема трансляції скрипту на мові PLang

Абстрактне синтаксичне дерево в даному випадку є структурою даних, яка використовується для представлення синтаксичної структури вихідного коду опису крайової задачі в абстрактній формі. Основною складовою дерева є вузли (Nodes), які представляють конструкції мови опису крайових задач, такі як операції, функції, змінні, вирази тощо.

Абстрактне синтаксичне дерево описується класом Tree, який містить одну властивість – об’єкт абстрактного класу Node і один метод – value(), який повертає значення виразу (в даному випадку – програмний код).

Нащадками Node є наступні класи: RealNode – містить код, що описує дійсне число; UnaryNode – інкапсулює унарну операцію; BinaryNode – описує бінарну операцію; FunctionNode – описує функцію.

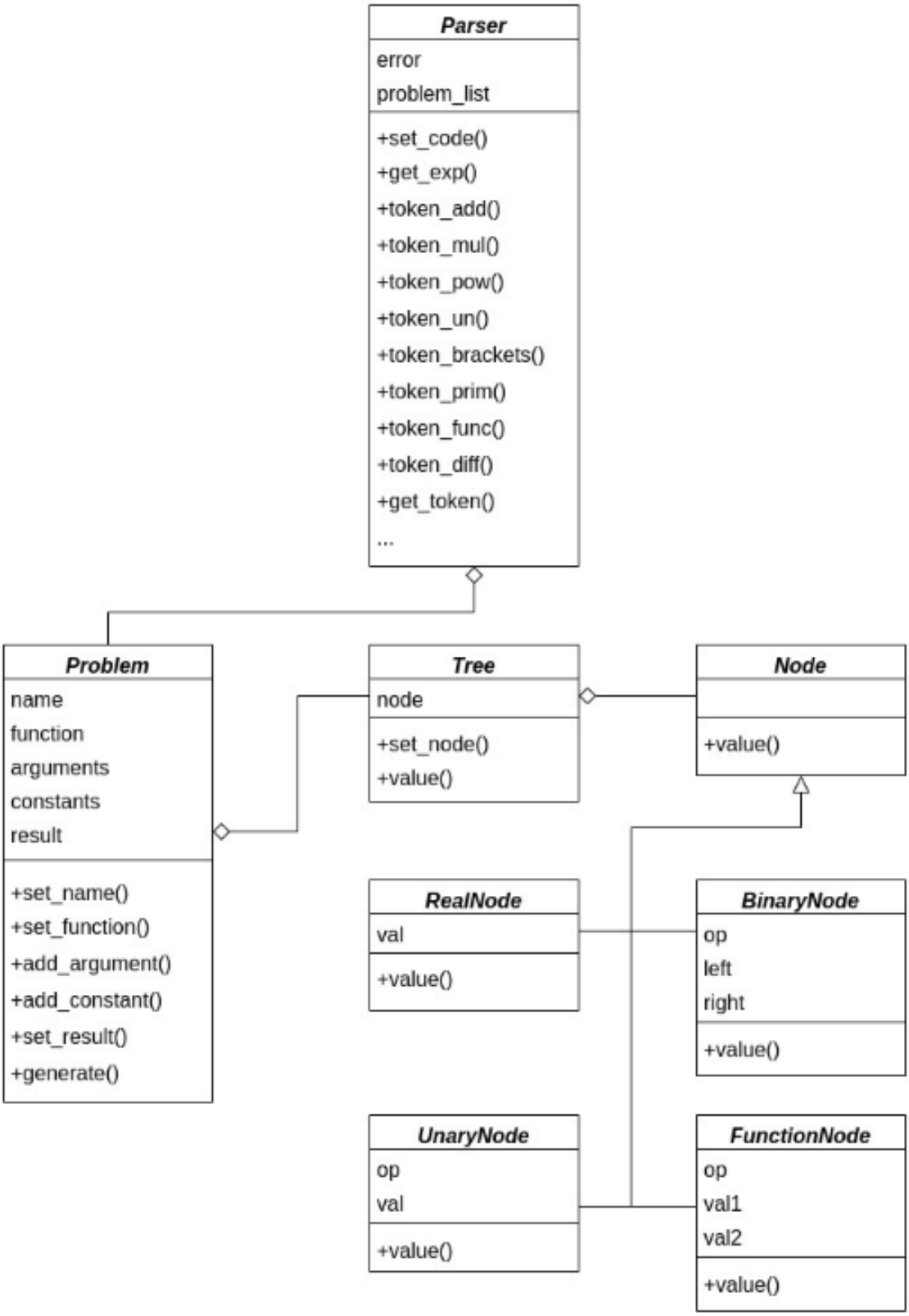


Рисунок 2.4 – Ієрархія класів транслятора мови PLang

Таким чином, кожен з класів-нащадків від Node реалізує певний тип вузла дерева виразу.

Одним з найбільш важливих етапів розбору початкового коду є його лексичний та синтаксичний аналіз. Вихідний код функції, яка виконує ці операції в парсері, наведена в Додатку В.

Прикладом результату роботи парсеру для задачі Бюргерса є наступний клас на мові Python.

```
class Burgers:

    def u(self, x, t):
        pass

    def diff(self, x):
        pass

    def u_bc(self, x):
        if x == 0:
            res = 0
        else:
            res = np.sin(np.pi*x) + 4*np.sin(2*np.pi*x)/(4+np.cos(np.pi*x)+
                  2*np.cos(2*np.pi*x))
    return res

    def equation(self, x, t):
        return self.diff(t)+self.u(x, t)*self.diff(x)-0.1*self.diff(self.diff(x), x)
```

Рисунок 2.5 – Результат роботи парсеру для задачі Бюргерса

У подальшому методи u() та diff() будуть визначені через відповідні класи нейронних мереж. А методи u_bc() та equation() будуть використовуватись при визначенні функції втрат.

Отже, розроблені класи парсеру мови опису крайових задач є суттєвою складовою бібліотеки нейромережевих методів. Простота опису та подібність синтаксису до поширених систем комп’ютерної алгебри дозволить зменшити час на засвоєння документації бібліотеки. Далі визначимо основні класи, які відповідають безпосередньо за метод нейронних мереж з фізичною інформацією та додаткові класи, що використовуються для автоматизації побудови нейромереж у відповідності до оптимальних гіперпараметрів.

2.7 Проєктування бібліотеки нейромережевих обчислювальних методів

Розглянемо основні класи бібліотеки розв’язання крайових задач нейромережевими методами. На рисунку 2.6 наведено концептуальну UML діаграму класів [92].

Зокрема, клас ANN містить поля та методи для визначення архітектури нейромережі: кількості шарів (n_layers), кількості нейронів в шарах (список n_units), список активацій в шарах (activations) та об’єкт PyTorch, який містить власне нейромережу. Метод forward() – програмує прямий хід обчислень мережі [92].

Клас Net містить PINN мережу, яка є наближеним методом розв’язання крайової задачі. Поле model містить екземпляр класу ANN. Набори даних X_train та y_train описують поведінку в області визначення та області значень невідомої функції. Ці набори кодують початкові та граничні умови. Поля optimizer та criterion містять відповідно оптимізатор та метрику похибки бібліотеки PyTorch. Метод loss_func() відповідає за формування функції втрат мережі, яка містить похибку на крайових умовах та похибку диференціального рівняння. В цьому методі відбувається диференціювання мережі. Метод train() виконує навчання PINN мережі [92].

Класи, що реалізують еволюційні підходи до оптимізації гіперпараметрів GA_ANN та PSO_ANN. Клас GA_ANN використовується для налаштування гіперпараметрів нейромережі генетичним алгоритмом. Поля класу визначають головні параметри алгоритму: розмір популяції генетичного алгоритму (population_size), ймовірність мутації при генерації нових особин (mutation_rate), ймовірність перехресного схрещування між батьками для створення нащадків (crossover_rate), кількість поколінь або ітерацій для виконання генетичного алгоритму (generations), об’єкт нейромережі, який буде налаштовуватись за допомогою генетичного алгоритму [92].

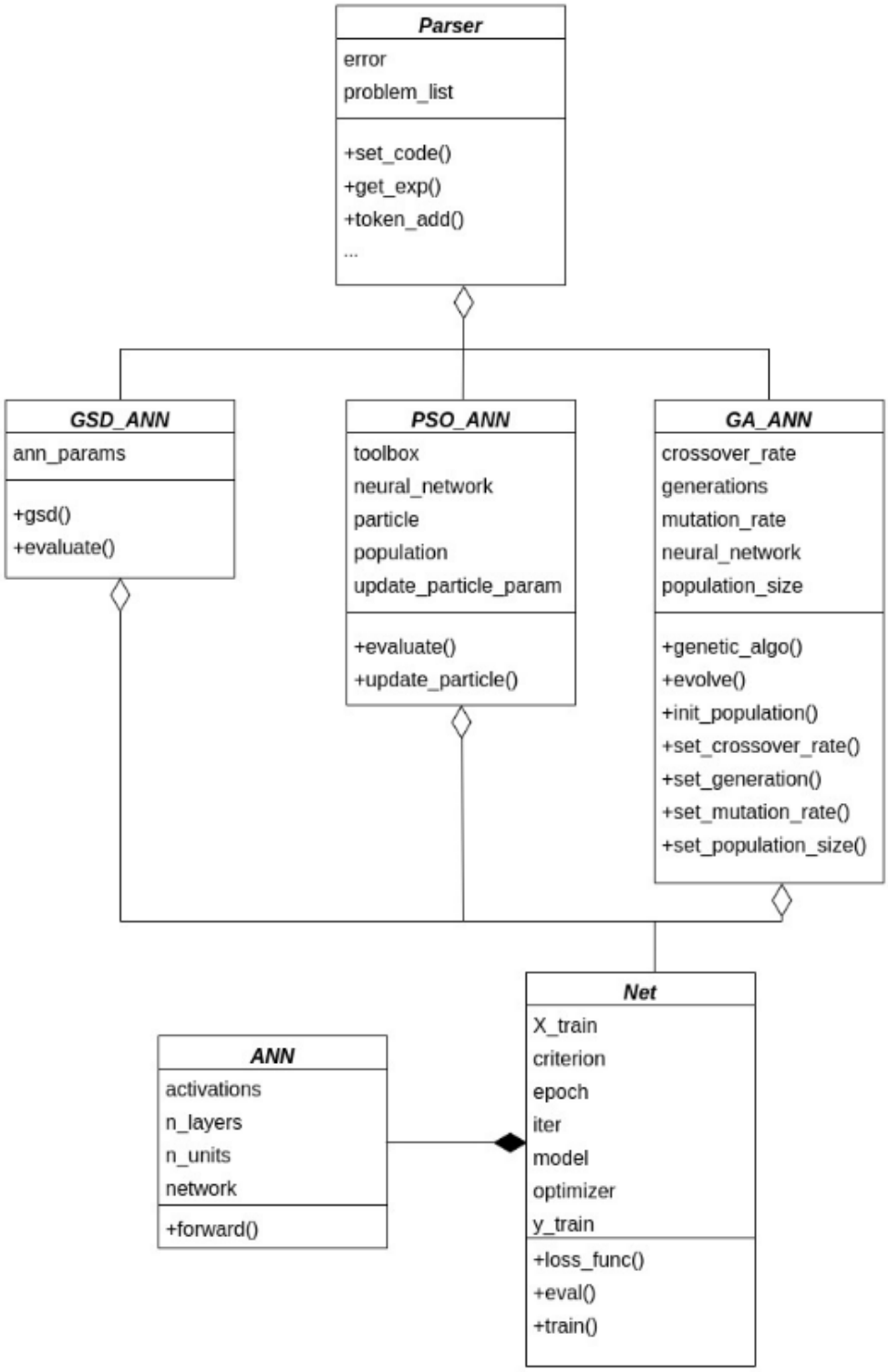


Рисунок 2.6 – Діаграма класів бібліотеки, що проєктується

Відповідні методи класу GA_ANN відповідають за реалізацію основних операторів: метод для встановлення розміру популяції (set_population_size), метод

для встановлення ймовірності мутації (set_mutation_rate), метод для встановлення ймовірності перехресного схрещування (setCrossoverRate) тощо [92].

Клас GSD_ANN реалізує методи планування експериментів оптимізації гіперпараметрів нейромереж з фізичною інформацією.

Висновки до розділу 2

У другому розділі спроєктовано та реалізовано предметно-орієнтовану мову PLang (Problem Language), призначену для формального опису крайових задач. Дана мова дозволяє робити опис задачі у інтуїтивній та зрозумілій формі, що схожа на багато систем комп’ютерної алгебри. Застосування даного підходу дозволить суттєво спростити використання бібліотеки нейромережевих обчислювальних методів та розширити їх практичне застосування дослідниками та інженерами.

Запропонована проблемно-орієнтована може використовуватись як стандартизований інструмент опису задач, що сприятиме швидкому створенню і тестуванню моделей. Це значно прискорить наукові дослідження та промислове впровадження нейромережевих обчислювальних методів.

Практичне значення мови PLang полягає, також, у відтворюваності досліджень, оскільки сприятиме зменшенню помилок при написанні програмного коду для розв’язання крайових задач у порівнянні з іншими бібліотеками.

Основні наукові і практичні результати даного розділу опубліковано в роботах [91, 92].

3 ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ ТА ПРИКЛАДИ ЗАСТОСУВАННЯ НЕЙРОМЕРЕЖЕВИХ МЕТОДІВ

3.1 Основні поняття нейронних мереж з фізичною інформацією

Нейромережу можна подати як складну нелінійну функцію, яка виконує завдання відображення вхідних даних у вихідні. Вона складається з нейронів, які організовані в шари, і кожен нейрон взаємодіє з іншими за допомогою зважених зв’язків та функцій активацій [94].

Припустимо, маємо одношарову нейронну мережу з вектором входу x, виходом y та матрицею ваг W, а також вектором зсуву b. Математично цю мережу можна описати наступним чином:

$$y = f(Wx + b),$$

де $f(\cdot)$ – активаційна функція, яка додає нелінійність до моделі.

Наведене рівняння відображає трансформацію вхідних даних x у вихід y. Активаційна функція важлива для того, щоб дати нейронній мережі можливість навчатися складнішим взаємозв’язкам та нелінійним шаблонам у даних [94].

Для багатошарових нейронних мереж цей опис може бути розширений, додаючи додаткові шари та ваги. В такому випадку вираз може виглядати як послідовність функцій, де виходи одного шару є входами наступного. Такий підхід дозволяє нейронним мережам моделювати більш складні взаємозв’язки в даних.

Багатошарову нейронну мережу можна узагальнити як композицію функцій, де кожен шар є лінійною трансформацією вхідних даних, а активаційні функції надають нелінійність моделі. Для задач багатовимірних входів та виходів загальна математична форма може виглядати наступним чином.

Нехай маємо L шарів у багатошаровій нейронній мережі. Кожен шар має свою матрицю ваг W^l та зсув b^l, а також активаційну функцію f^l. Вхідний вектор позначений x, а вихідний вектор мережі – y. Опис функції мережі для одного прикладу може бути подано наступним чином [94]:

$$z^l = W^l \cdot a^{l-1} + b^l,$$
$$a^l = f^l(z^l),$$

де l – індекс шару (від 1 до L); z^l – лінійна комбінація входів та параметрів шару; a^l – вихідний вектор шару після застосування активаційної функції.

Для багатошарової мережі вихід буде остаточним результатом останнього шару: $y = a^L$. Отже, загальна функція, яка описує багатошарову нейронну мережу прямого поширення сигналу, може бути визначена як композиція функцій, представлену наступним чином [94]:

$$y = f^L\left(W^L \cdot f^{L-1} \cdot \left(W^{L-1} \cdot \left(... \cdot f^1\left(W^1 \cdot x + b^1\right) + ... + b^{L-1}\right) + b^L\right)\right),$$

де L – кількість шарів у нейронній мережі; x – вхідний вектор; W^l та b^l – ваги та зсуви шару l; $f^l(\cdot)$ – функція активації шару l.

Ця функція представляє собою послідовність лінійних та нелінійних операцій для обробки вхідного вектора та отримання вихідного результату. Кожен шар мережі вносить власний вклад у зміну структури та формування складної залежності між вхідними та вихідними даними.

Ваги W^l та b^l мережі є параметрами, які оптимізуються під час навчання для досягнення певного вихідного результату.

У багатошаровій мережі прямого поширення сигналу кожен нейрон у шарі пов’язаний з кожним нейроном в попередньому та наступному шарах. В процесі тренування мережі використовується функція втрат для визначення різниці між прогнозованими та фактичними значеннями. Мета оптимізації – мінімізувати функцію втрат для досягнення точного прогнозу.

Оптимізатор у контексті нейронних мереж – це ключовий елемент тренування, що відповідає за адаптацію параметрів мережі при мінімізації функції втрат. Процес оптимізації розпочинається з обчислення градієнтів функції втрат, які вказують напрямок найшвидшого зменшення втрат. Зазвичай, використовуються такі методи оптимізації, як градієнтний спуск, стохастичний градієнтний спуск та різні варіації, що визначаються задачею та властивостями даних.

Наведені компоненти дозволяють нейромережі моделювати складні та нелінійні залежності в даних. Складність функції може зростати з кількістю шарів, нейронів та загальною складністю архітектури. Тренування нейромережі полягає в підборі оптимальних ваг та зсувів для вирішення конкретної задачі.

PINN представляють собою підхід в області машинного навчання та фізичного моделювання, який поєднує потужність нейронних мереж з диференціальними рівняннями для розв’язання завдань, пов’язаних із фізичними процесами. У контексті PINN, нейронна мережа навчається на основі даних, де вхідні та вихідні пари відображають точки в домені (x, u), де x – це точка в просторі, а u – відповідне значення фізичної величини. Однак відмінність PINN полягає в тому, що до функції витрат додаються фізичні рівняння, такі як диференціальні рівняння та граничні умови. Це дозволяє моделі не лише апроксимувати дані, а й враховувати фізичні обмеження та відомі закони. Під час оптимізації, PINN шукає розв’язок, який відповідає фізичним законам та навчальним даним, забезпечуючи таким чином фізичну правдоподібність та ефективність у вирішенні задач фізики та інженерії [11].

Розглянемо загальне формулювання для диференціального рівняння n-го порядку з граничними умовами. Нехай $u(x)$ – шукана функція, яка задовольняє рівняння та граничні умови.

У загальному випадку диференціальне рівняння, що описує фізичний процес:

$$F(u, \nabla u, \nabla^2 u, ..., \nabla^n u, x, t) = 0,$$

де $u(x)$ – функція, яка залежить від n змінних $x = (x_1, x_2, ..., x_n)$; ∇ – градієнт; x – просторові координати; t – час. Функція F виражає диференціальне рівняння з урахуванням похідних до n-го порядку.

Граничні умови виглядають так:

$$G(u(a), u(b)) = 0,$$

де a та b – вектори, представляють нижню та верхню границі домену.

Функція втрат нейромережі, яка використовується при оптимізації ваг мережі [11]:

$$L = L_{data} + L_{physics},$$

де L_{data} відображає середньоквадратичну помилку на основі навчальних даних (в точках колокації); $L_{physics}$ відображає фізичну помилку, яка включає у себе вирази, що випливають з диференціального рівняння та граничних умов.

Отже, під час оптимізації нейронної мережі визначається функція $u(x)$, яка задовольняє як навчальні дані, так і фізичні закони, що описують систему.

3.2 Розв’язання задач пружності

Розглянемо декілька модельних задач лінійного та нелінійного згину круглих пластин та балок. Побудуємо наближений розв’язок засобами PINN мереж та порівняємо отримані результати з точним розв’язком.

Приклад 3.2.1. Згин круглої пластини з защемленим контуром. Диференціальне рівняння прогину $w(r)$ [95]:

$$\frac{d^4w}{dr^4} + \frac{2}{r}\frac{d^3w}{dr^3} - \frac{1}{r^2}\frac{d^2w}{dr^2} + \frac{1}{r^3}\frac{dw}{dr} = \frac{q}{D},$$

де q – значення розподіленого поперечного навантаження; D – циліндрична жорсткість пластини; r – просторова координата в полярній системі координат.

Граничні умови мають вигляд [95]:

$$\frac{dw}{dr}(a) = 0, \frac{dw}{dr}(0) = 0, w(a) = 0,$$

де a – радіус круглої пластини.

Розглянемо поперечний згин круглої одношарової пластини з такими параметрами: товщина пластини $h = 18 \cdot 10^{-3}$ м, радіус $a = 0.4$ м, модуль зсуву та коефіцієнт Пуассона матеріалу – $G = 2.77 \cdot 10^4$ МПа та $\nu = 0.3$ відповідно, розподілене навантаження $q = 0.5$ МПа. Результати порівняння отриманого нейромережевого розв’язку з точним [95, 96] наведено на рисунку 3.1 [86].

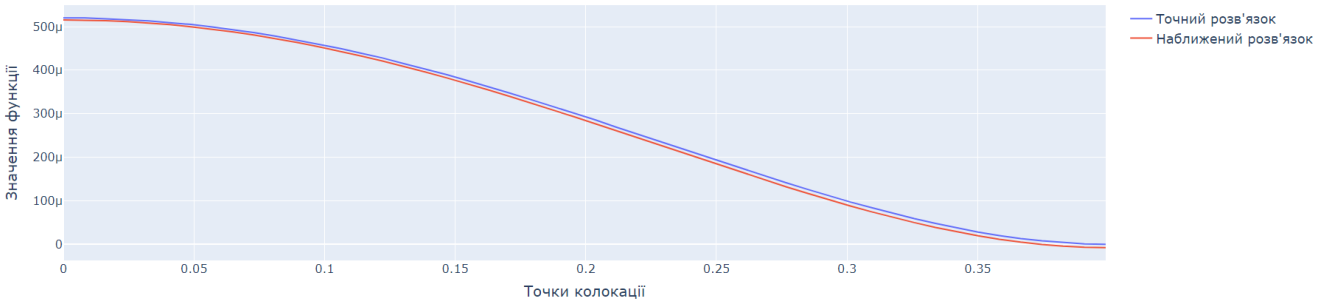


Рисунок 3.1 – Защемлення круглої пластини: точний та наближений нейромережевий розв’язки

Точну та наближену функцію прогину на всій області пластини зображено на рисунку 3.2

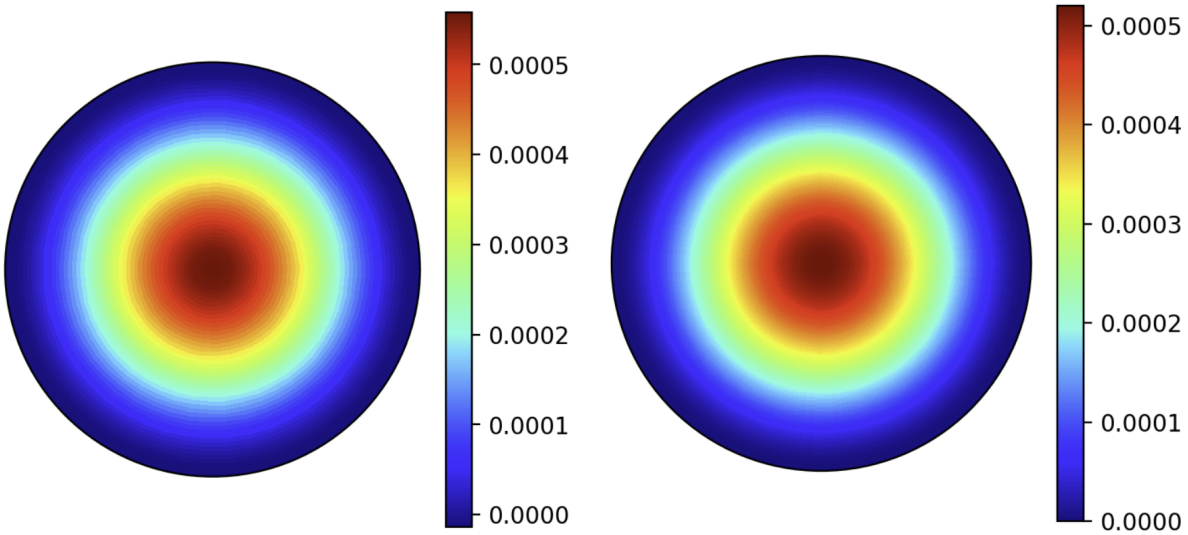


Рисунок 3.2 – Точна та наближена функція прогину на всій області защемленої пластини

Приклад 3.2.2. Згин круглої пластини з шарнірно опертим контуром. Диференціальне рівняння співпадає з рівнянням прикладу 3.2.1, а граничні умови:

$$w(a) = 0,$$
$$-D\left(\frac{d^2w}{dr^2} + \frac{\nu}{r}\frac{dw}{dr}\right) = 0,$$

де ν – коефіцієнт Пуассона.

Параметри пластини співпадають з прикладом 3.2.1. На рисунку 3.3 наведено результати обчислювальних експериментів [86].

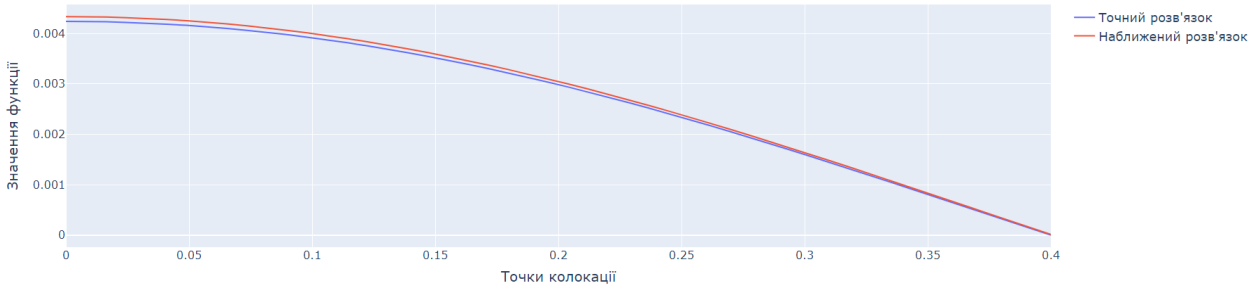


Рисунок 3.3 – Вільне опирання круглої пластини: точний та наближений нейромережевий розв’язки

Точну та наближену функцію прогину на всій області пластини зображено на рисунку 3.4. Відносна помилка отриманих наближених розв’язків в задачах згину круглої пластини становить до 2%.

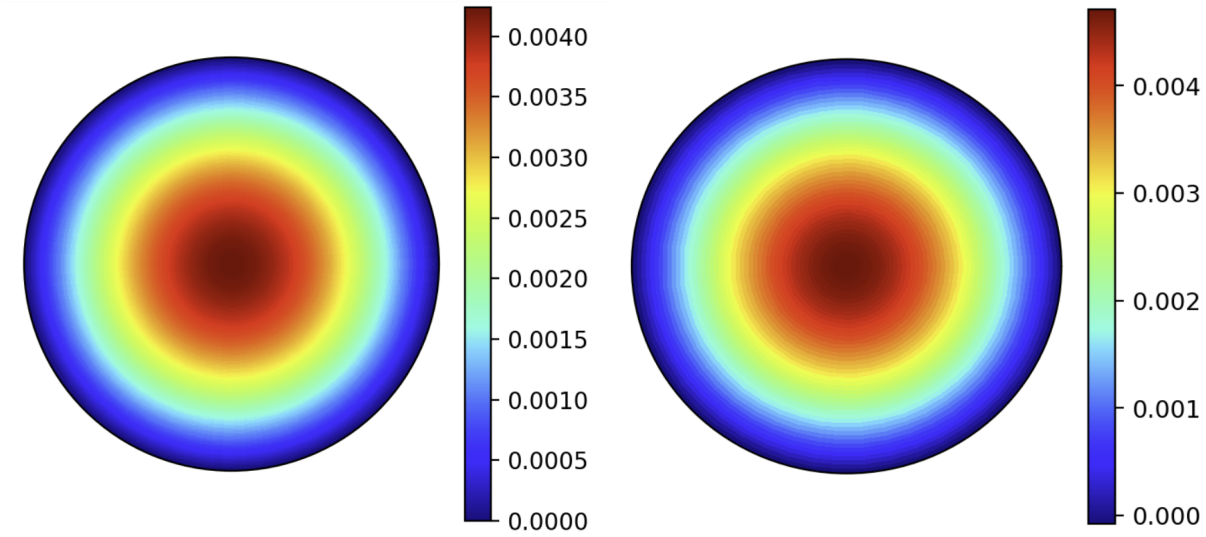


Рисунок 3.4 – Точна та наближена функція прогину на всій області вільно опертої пластини

Слід зазначити, деякі особливості PINN мереж для розв’язання задач згину пластин у порівнянні, наприклад, з методом Гальоркіна.

Зокрема, перевагою нейромережевих методів є їх універсальність та гнучкість. Вони можуть застосовуватися до широкого спектра задач без значної модифікації, інтегруючи різні типи граничних умов і фізичних законів. Традиційні методи, такі як метод Бубнова-Гальоркіна, потребують вибору апроксимаційних функцій і можуть вимагати ручного налаштування для кожної нової задачі.

Приклад 3.2.3. Згин балки з закріпленими кінцями. Диференціальне рівняння має вигляд [97]:

$$\frac{d^4w}{dx^4} = \frac{q}{EI},$$

де q – значення розподіленого поперечного навантаження; E – модель Юнга; I – момент інерції балки.

Граничні умови:

$$w(L) = 0, w(0) = 0, \frac{dw}{dx}(0) = 0, \frac{dw}{dx}(L) = 0,$$

де L – довжина балки.

Результати обчислень наведено в таблиці 3.1. Розглянуто задачу згину балки з круговим перерізом з такими параметрами: довжина $L = 2$ м, розподілене поперечне навантаження $q = 1$ МПа, модуль зсуву $G = 2.77 \cdot 10^4$ МПа, коефіцієнт Пуассона $\nu = 0.3$, діаметр $d = 0.5$ м [86].

Таблиця 3.1 – Згин балки. Порівняння значень точного та наближеного розв'язку

Точка колокації	Точний розв'язок	Нейромережевий розв'язок
0	0	0
0.2	$1.4396 \cdot 10^{-4}$	$1.1106 \cdot 10^{-4}$
0.4	$4.5498 \cdot 10^{-4}$	$4.2581 \cdot 10^{-4}$
0.6	$7.8378 \cdot 10^{-4}$	$7.6258 \cdot 10^{-4}$
0.8	$10.237 \cdot 10^{-4}$	$10.133 \cdot 10^{-4}$
1	$11.108 \cdot 10^{-4}$	$11.127 \cdot 10^{-4}$

Як можна побачити, похибка максимального прогину становить менше 1%.

Приклад 3.2.4. Згин балки з одним закріпленим кінцем і одним вільним кінцем [96].

Граничні умови мають вигляд:

$$w(0) = 0, \frac{dw}{dx}(0) = 0.$$

Згідно з результатами обчислень, наведеними вище відносна помилка для цієї задачі не перевищує 1%.

Приклад 3.2.5. Згин балки з одним закріпленим кінцем в геометрично нелінійній постановці [98]. Нелінійне диференціальне рівняння в даному випадку набуває вигляду

$$\frac{d^2w}{dx^2} = \frac{M(x)}{EI}\left(1 + \left(\frac{dw}{dx}\right)^2\right)^{3/2}.$$

Розглядається задача згину балки з коцентрованим навантаженням на вільному кінці з такими параметрами: довжина $L = 1$ м, концентроване навантаження $q = 30$ Н, модуль Юнга $E = 70$ ГПа, діаметр кругового перерізу $d = 10$ мм. Результати порівняння лінійного та нелінійного випадків наведено на рисунку 3.5 [86].

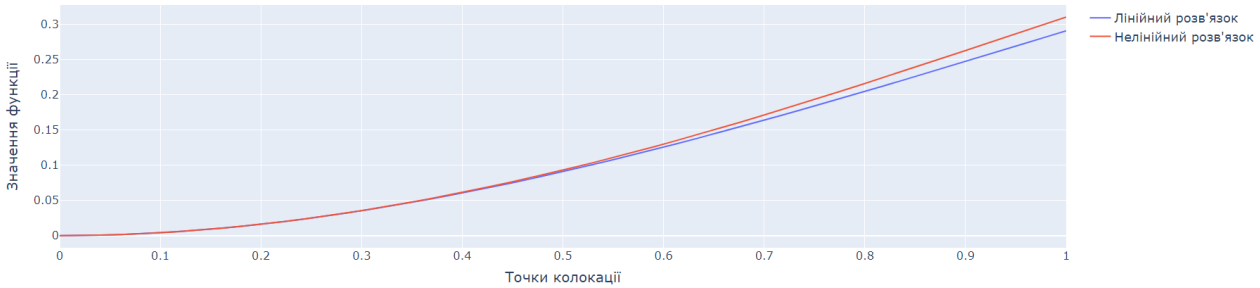


Рисунок 3.5 – Геометрично нелінійний згин балки

Максимальний прогин в нелінійному випадку на 6.6 % більше лінійного випадку, що відповідає результатам [98].

Приклад 3.2.6. Згин балки на фізично нелінійнійній основі. Розглядається задача згину балки на нелінійно-пружній основі Вінклера під дією концентрованого навантаження в центрі. Нелінійне диференціальне рівняння наведено в роботі [99]:

$$\frac{d^4w}{dz^4} + \frac{4w}{1 + \mu w} = \frac{4}{k}p(z),$$

де z – безрозмірна просторова координата; μ – коефіцієнт нелінійності основи Вінклера; k – коефіцієнт навантаження.

Функція навантаження визначається через дельта функцію Дірака:

$$p(z) = \overline{P}\delta(z - \alpha),$$

де $\overline{P}$ – безрозмірне значення концентрованого навантаження; α – точка прикладення концентрованого навантаження.

Граничні умови:

$$w(\beta) = 0, w(0) = 0, \frac{dw}{dx}(0) = 0, \frac{dw}{dx}(\beta) = 0,$$

де β – безрозмірна довжина балки.

Розглянемо задачу згину з такими параметрами: довжина балки $\beta = 5$, коефіцієнт нелінійності $\mu = 6$, фактор навантаження $\overline{P}/k = 0.5$ [86]. Результати обчислень наведено на рисунку 3.6, що відповідає роботі [99].

Отже, розв’язання лінійних та нелінійних задач пружності балок та пластин засобами PINN мереж є досить ефективним. Одна нейромережа може застосовуватись при прогнозуванні переміщень для різних об’єктів з різними граничними умовами.

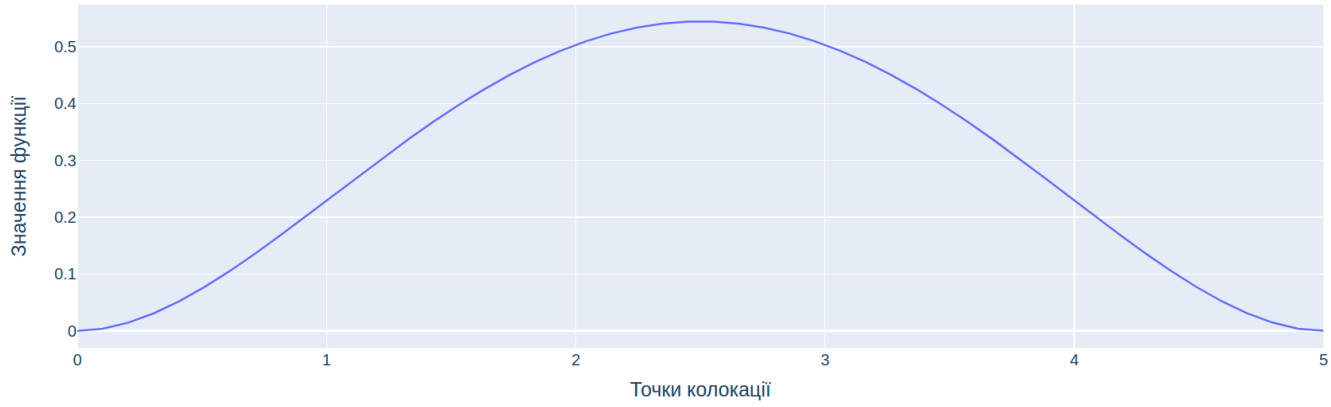


Рисунок 3.6 – Згин балки на нелінійній основі

Така універсальність призводить до легкості програмної реалізації відповідних методів.

3.3 Розв’язання прямих та обернених задач

Розв’язання обернених задач для диференціальних рівнянь має велике практичне значення з кількох причин. У різних галузях науки та інженерії ці задачі дозволяють отримувати важливу інформацію про системи, які описуються диференціальними рівняннями.

Обернені задачі дозволяють відновлювати невідомі параметри систем, які важко або неможливо виміряти безпосередньо. Наприклад, у задачах механіки можна визначати параметри конструкцій, які задовольняють певним виміряним або спроєктованим значенням напружено-деформованого стану.

В той же час, для калібрування та валідації моделей, які описують фізичні або інженерні процеси, необхідно їх налаштовувати на основі реальних даних. Обернені задачі дозволяють проводити ці калібрування, що підвищує достовірність і точність моделей.

Для розв’язання обернених задач із використанням класичних підходів можуть використовуватися як аналітичні, так і чисельні методи. Аналітичні методи, зазвичай, вимагають складних математичних перетворень і не завжди дають змогу отримати розв’язок у явному вигляді. Чисельні методи, з іншого боку, можуть вимагати значних ресурсів для досягнення збіжності.

Нейронні мережі з фізичною інформацією також використовуються для розв’язання обернених задач [84]. В цьому випадку, в параметри нейронної мережі вводяться невідомі коефіцієнти диференціального рівняння, які підлягають визначенню в оберненій постановці. Отже, під час пошуку розв’язку, який задовільняє крайовій задачі налаштовуються не тільки ваги мережі, а також коефіцієнти, які необхідно визначити за умовами оберненої задачі. Вхідними

даними для розв’язання оберненої задачі нейромережевими методами є розв’язок прямої задачі, який отримано будь-яким наближеним або точним методом. Зокрема, можливим підходом може бути застосування PINN мереж для розв’язання спочатку прямої задачі, а потім використання отриманого розв’язку у якості даних у оберненій задачі [84].

Методика PINNs може бути легко адаптована для різних типів диференціальних рівнянь та обернених задач, що робить цей метод дуже універсальним і застосовним до широкого спектра проблем у різних галузях науки та техніки.

Отже, метод PINNs є потужним інструментом для розв’язання обернених задач, дозволяючи ефективно визначати коефіцієнти диференціальних рівнянь з урахуванням вхідних даних. Завдяки поєднанню нейронних мереж та фізичних законів, цей метод забезпечує точні та стабільні результати, навіть для складних і нелінійних задач.

Розглянемо нейромережі з фізичною інформацією для розв’язання прямих та обернених задач рівняння Бюргерса [100, 103]:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2},$$

де: $u(t,x)$ – функція, що описує швидкість потоку (наприклад, швидкість руху рідини або газу) у залежності від часу t і координати x; ν – коефіцієнт кінематичної в’язкості середовища, який визначає внутрішнє тертя в рідині чи газі.

Рівняння Бюргерса є одним з фундаментальних нелінійних рівнянь у математичній фізиці, яке описує рух неідеальної рідини або газу з урахуванням ефектів дифузії і конвекції [103]. Поширеним є використання саме цього рівняння для тестування наближених обчислювальних методів, оскільки є відомими точні розв’язки для різних початкових та граничних умов [103].

Далі наводяться приклади розв’язання прямих та обернених задач рівняння Бюргерса методом PINN мереж для п’яти крайових умов. Виконується порівняння з відповідними точними розв’язками, відомими з літератури.

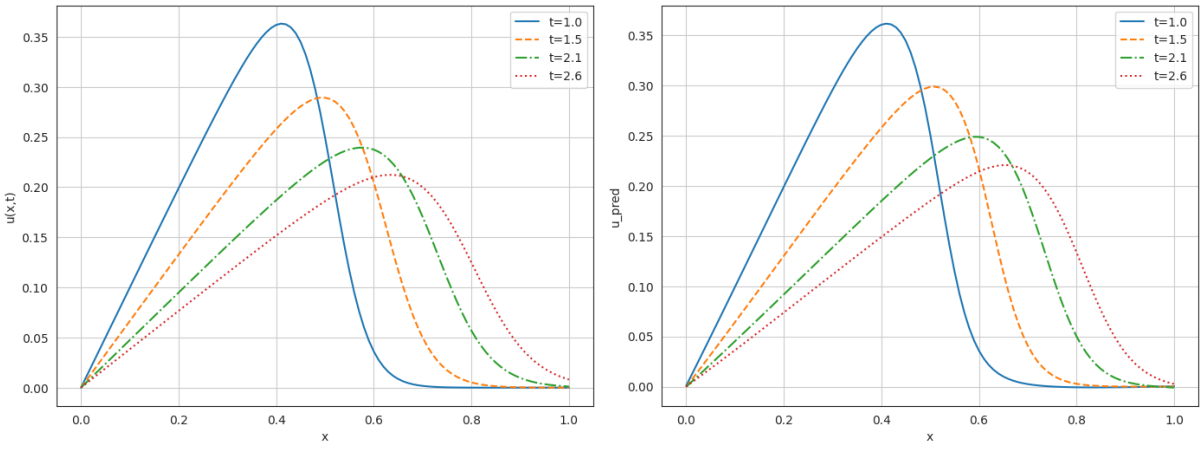
Приклад 3.3.1. Область визначення $x \in [0, 1]$, $\nu = 0.01$. Граничні умови $u(0, t) = u(1, t) = 0$. Початкова умова:

$$u(x, 1) = \frac{x}{1 + \sqrt{\frac{t}{t_0}} e^{\frac{x^2}{4\nu}}}, t \geq 1, t_0 = e^{\frac{1}{8\nu}}.$$

Точний розв’язок має вигляд [100]:

$$u(x, t) = \frac{1}{t} \frac{x}{1 + \sqrt{\frac{t}{t_0}} e^{\frac{x^2}{4\nu t}}}, t \geq 1.$$

На рисунку 3.7 зображено точні та наближені розв’язки [84]



а) б)

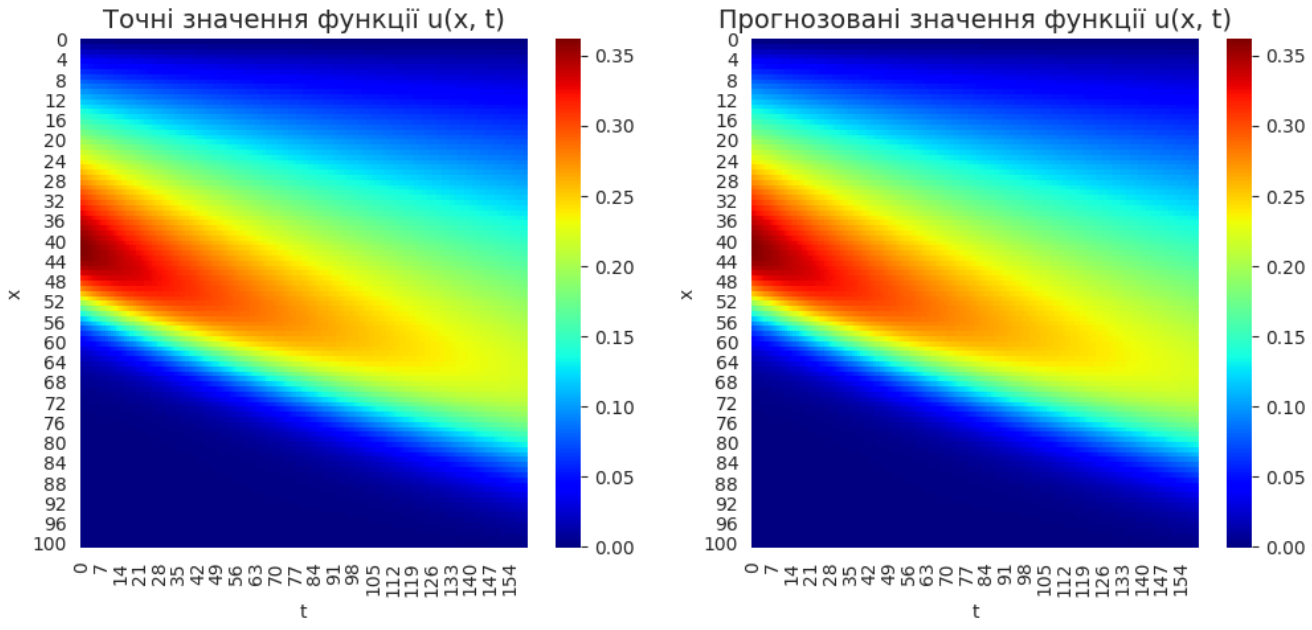
Рисунок 3.7 – Точний (а) та наближений (б) розв’язки прикладу 3.3.1

Як метрику схожості точних та прогнозованих значень функції $u(t, x)$ використаємо коефіцієнт детермінації R^2 [104]. Значення $R^2 = 1$ означає, що всі спостереження точного розв’язку співпадають з прогнозованими значеннями. Для даного прикладу $R^2 = 0.996$.

Обернену задачу визначення коефіцієнта кінематичної в’язкості ν як параметра нейромережі ν_{pinn}, розв’язано з відносною похибкою $\frac{|\nu - \nu_{pinn}|}{\nu} \cdot 100\% = 0.15\%$.

Розподіл точного та наближеного розв’язку зображено на рисунку 3.8. Можна побачити, що в даному випадку точність побудованої PINN нейромережі є високою. Використовувалась рівномірна сітка точок колокації для навчання мережі із кроком 0.01 за координатою та часом.

Слід зазначити, що дана точність результатів досягається при відносно невисокій кількості ітерацій навчання нейронної мережі. У порівнянні з класичними чисельними методами на основі нев’язок, наприклад, методом колокації або Гальоркіна, при використанні нейромережевого підходу не потрібно визначати пробні функції.



а) б)

Рисунок 3.8 – Точний (а) та наближений (б) розв’язки прикладу 3.3.1

Розподіл значень квадратичної помилки прогнозу за областю визначення функції є важливим параметром для вибору оптимального набору точок колокації.

Зокрема, на практиці можуть використовуватись адаптивні методу вибору точок, на яких навчається нейромережа [56].

Значення помилки на всій області визначення невідомої функції наведено на рисунку 3.9.

Розподіл помилки на області визначення є важливою характеристикою нейромережевих обчислювальних методів. Оскільки ці значення безпосередньо можуть впливати на метод вибору точок для навчання нейромережевих моделей. Один із способів покращення збіжності є адаптивний вибір точок колокації, які використовуються при навчанні [56]. Наприклад, адаптивні алгоритми передбачають біль щільну вибірку точок навчання в областях з великими значеннями помилки.

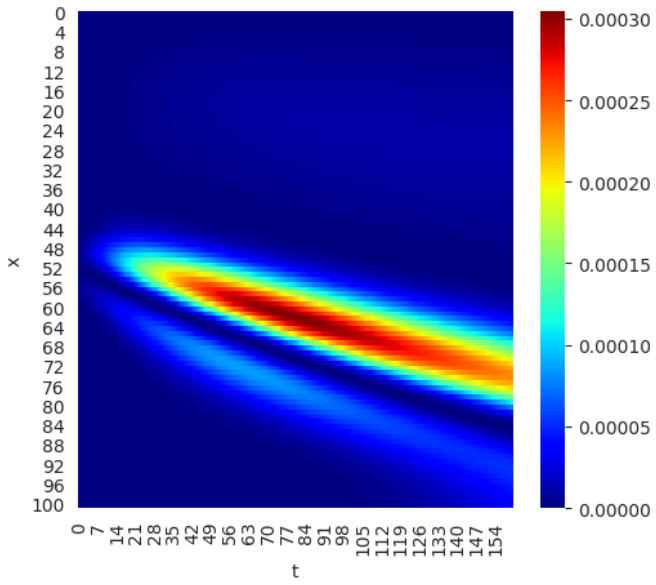
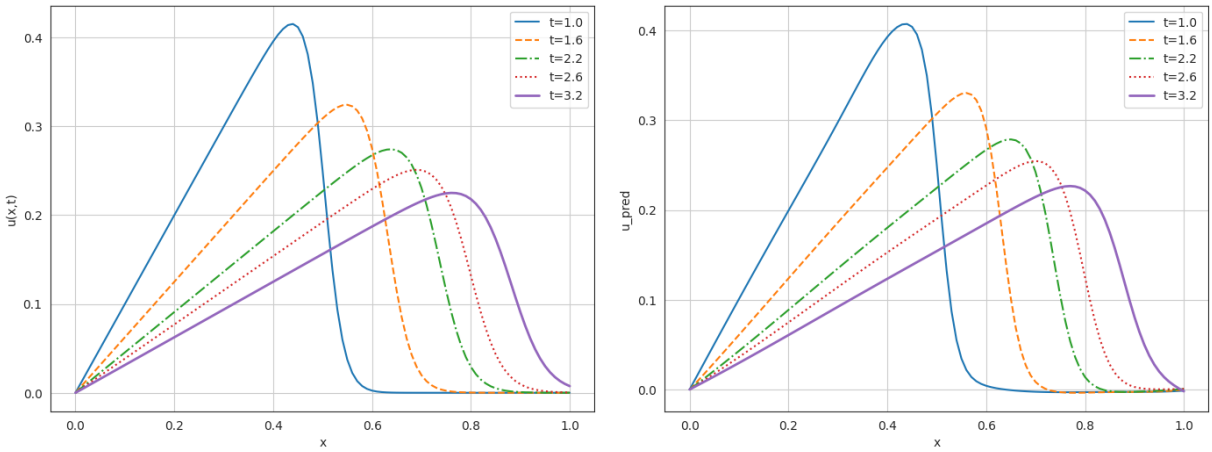


Рисунок 3.9 – Розподіл помилки прогнозування прикладу 3.3.1

Приклад 3.3.2. Область визначення, початкові та граничні умови такі ж як у прикладі 3.3.1, відрізняється тільки коефіцієнт $\nu = 0.005$.

На рисунку 3.10 зображено точні та наближені розв’язки для різних значень часу t [84].

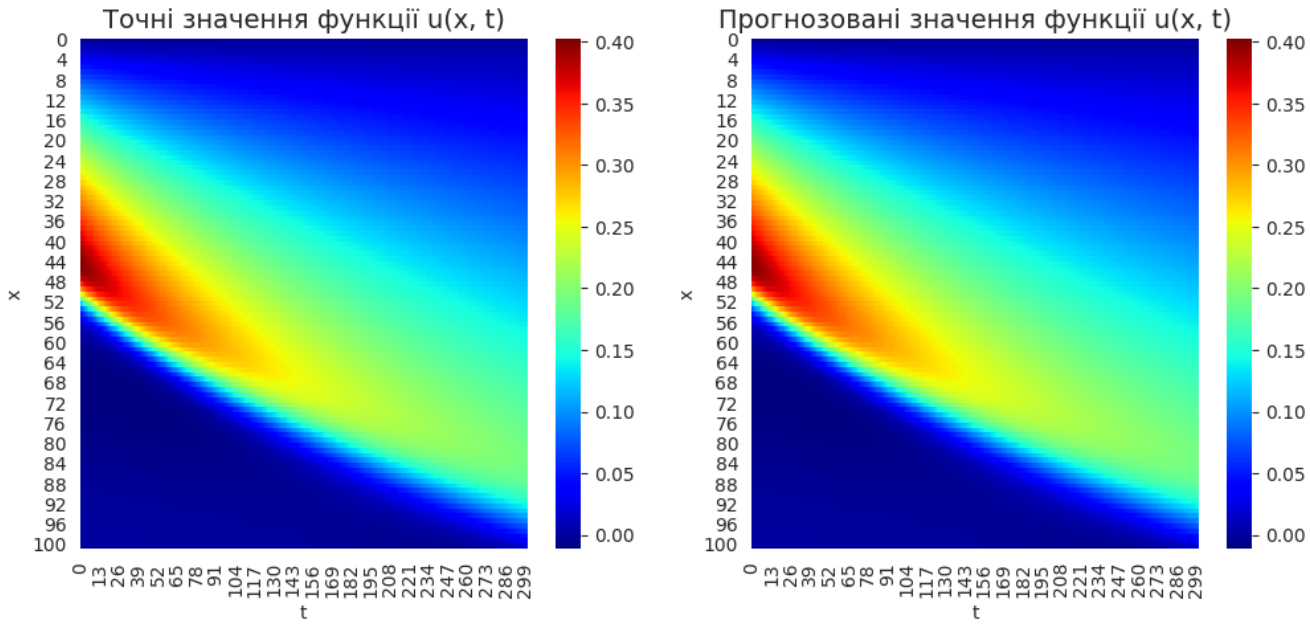


а) б)

Рисунок 3.10 – Точний (а) та наближений (б) розв’язки прикладу 3.3.2

Коефіцієнт детермінації точного та наближеного розв’язку становить $R^2 = 0.994$. Відносна похибка визначення коефіцієнта кінематичної в’язкості ν становить 0.11%.

Розподіл точного та наближеного розв’язку зображено на рисунку 3.11.



а) б)

Рисунок 3.11 – Точний (а) та наближений (б) розв’язки прикладу 3.3.2

Слід відзначити, що висока точність побудованих нейромережевих розв’язків забезпечується відносно простою архітектурою PINN мереж. Зокрема, було використано тришарову модель з оптимізаторами Adam та LBFGS, функцією втрат – середня квадратична помилка. Використовувалось дві тисячі епох навчання. Нейромережа такої структури може ефективно навчатись навіть у системах тільки з центральним процесором. Використання графічних процесорів суттєво зменшує час тренування мережі. Тестування розробленої моделі відбувалось на випадковому розподілі точок колокації, або на рівномірній сітці, але з іншим кроком, ніж при тренуванні.

Значення помилки на всій області визначення невідомої функції наведено на рисунку 3.12.

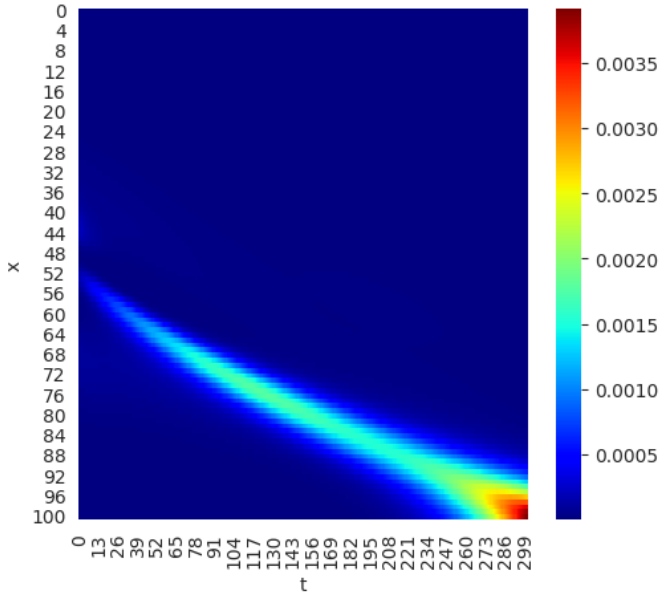
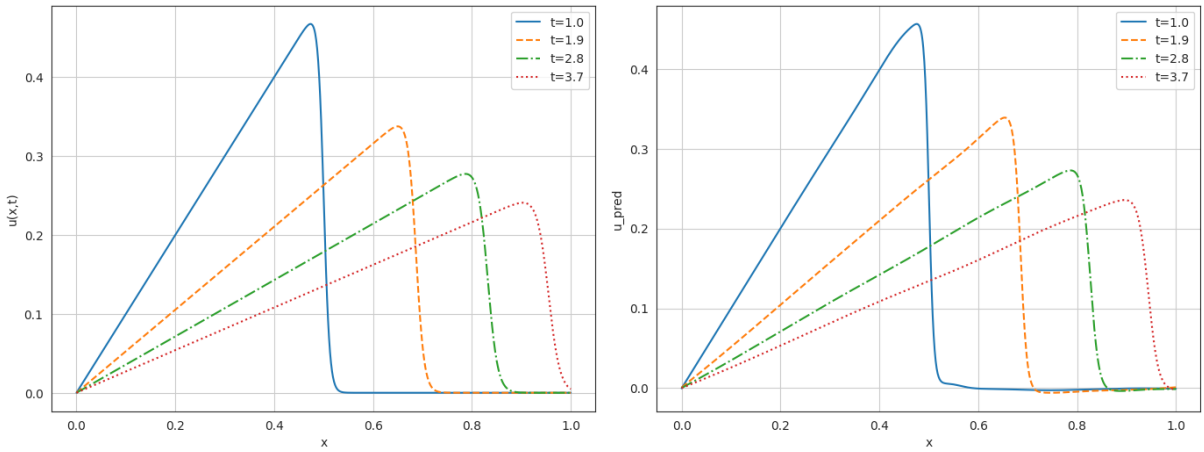


Рисунок 3.12 – Розподіл помилки прогнозування прикладу 3.3.2

Приклад 3.3.3. Коефіцієнт в’язкості $\nu = 0.0015$. Область визначення, початкові та граничні умови такі ж як у прикладі 3.3.1.

На рисунку 3.13 зображено точні та наближені розв’язки [84].

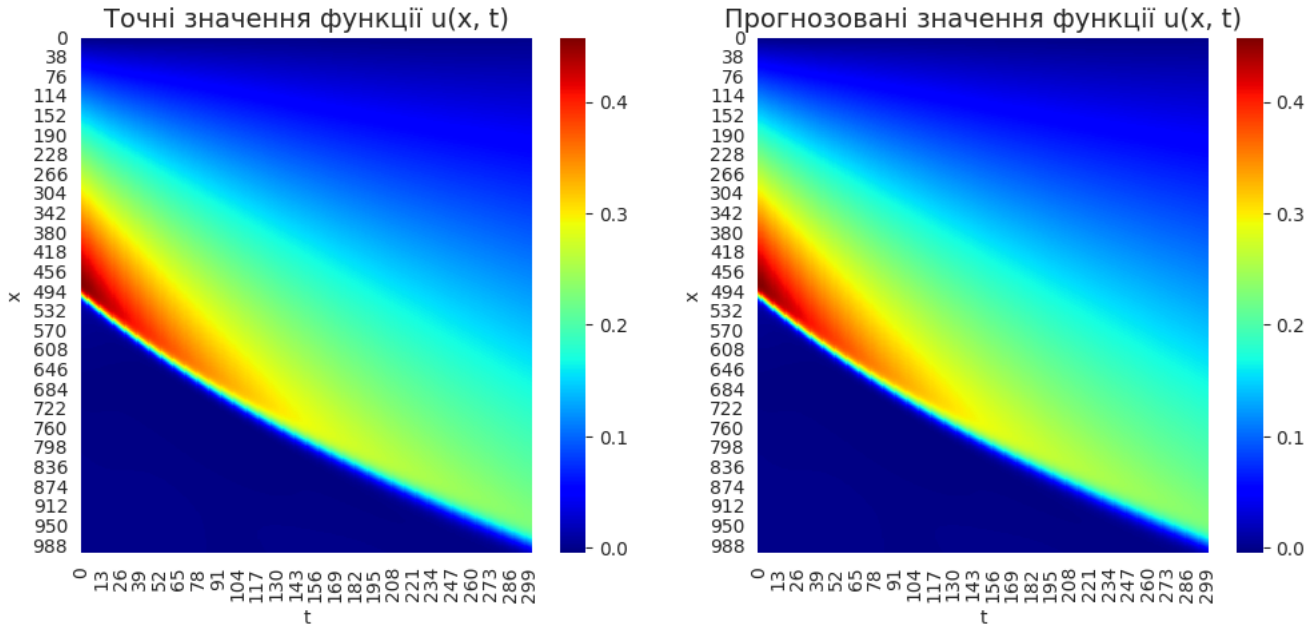


а) б)

Рисунок 3.13 – Точний (а) та наближений (б) розв’язки прикладу 3.3.3

Коефіцієнт детермінації точного та наближеного розв’язку становить $R^2 = 0.991$. Відносна похибка визначення коефіцієнта кінематичної в’язкості ν становить 2.18%.

Розподіл точного та наближеного розв’язку зображено на рисунку 3.14.



а) б)

Рисунок 3.14 – Точний (а) та наближений (б) розв’язки прикладу 3.3.3

Наведені розв’язки прикладів 3.3.1–3.3.3 для рівняння Бюргерса відрізняються виключно коефіцієнтом ν в діапазоні від $\nu = 0.0015$ до $\nu = 0.01$. Як буде продемонстровано в підрозділі 3.4, значення коефіцієнтів диференціальних рівнянь Бюргерса можуть впливати на точність PINN мереж. З отриманих результатів можна зробити висновок, що нейромережевий розв’язок залишається точним при малих значеннях коефіцієнту кінематичної в’язкості, що в цілому відповідає результатами роботи [108] для інших типів рівнянь. Згідно з [108] коефіцієнти диференціальних рівнянь значно впливають на збіжність моделей PINN. Якщо вони занадто великі або складні, модель може не зуміти навчитися адекватно відображати фізичні явища, що призводить до великих помилок.

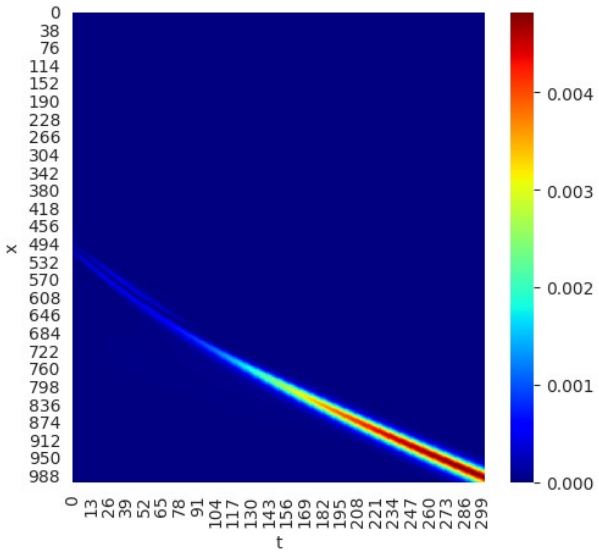
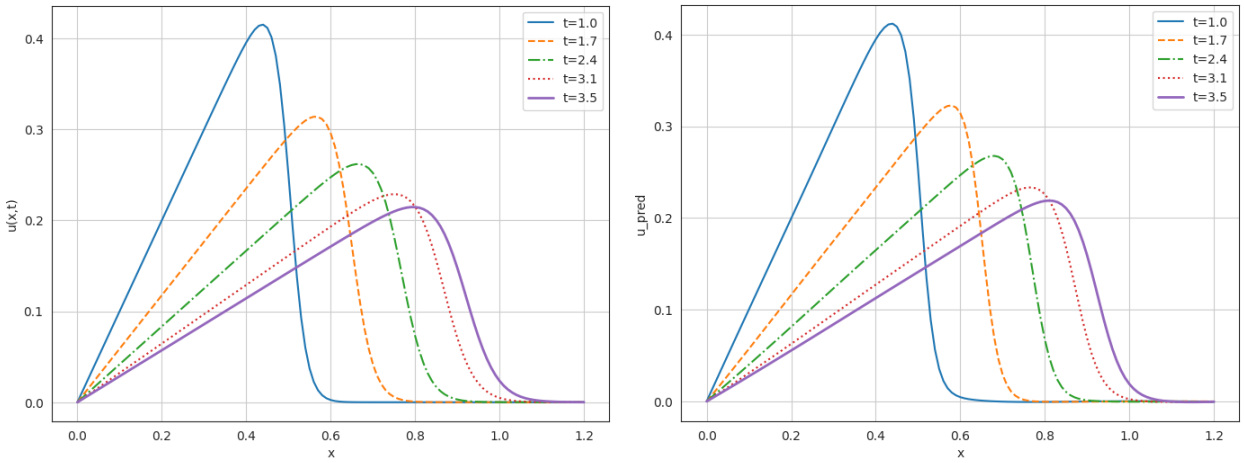


Рисунок 3.15 – Розподіл помилки прогнозування прикладу 3.3.3

Помилку у точках області визначення невідомої функції наведено на рисунку 3.15.

Приклад 3.3.4. Область визначення $x \in [0, 1.2]$, $\nu = 0.005$. Граничні умови $u(0, t) = u(1.2, t) = 0$. Початкова умова [84]:

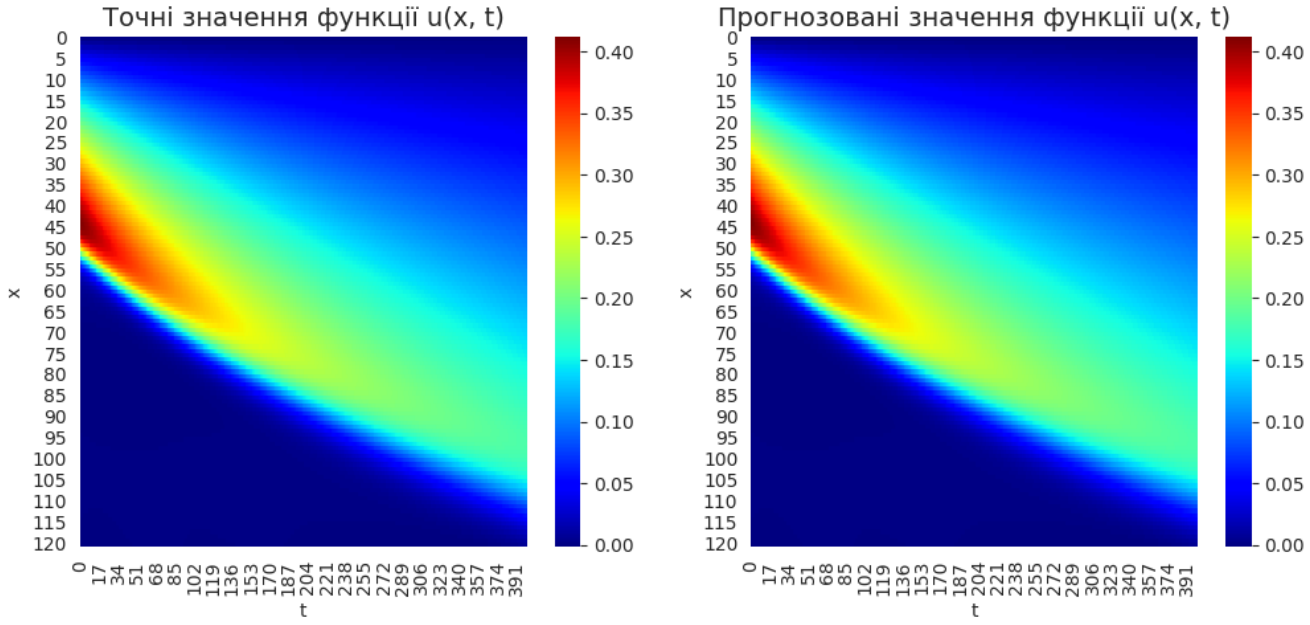
$$u(x, 1) = \frac{x}{1 + e^{\frac{x^2 - 0.25}{4\nu}}}, t \geq 1.$$



а) б)

Рисунок 3.16 – Точний (а) та наближений (б) розв’язки прикладу 3.3.4

На рисунку 3.16 зображено точні та наближені розв’язки рівняння Бюргерса. Розподіли розв’язів зображено на рисунку 3.17.



а) б)

Рисунок 3.17 – Точний (а) та наближений (б) розв’язки прикладу 3.3.4

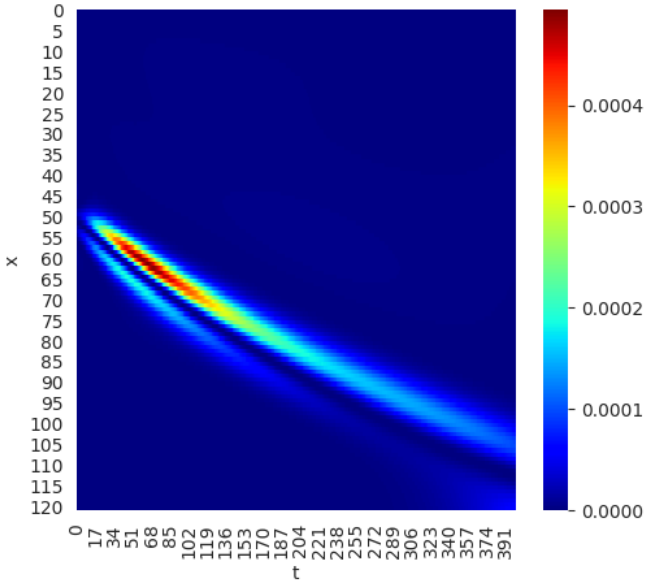


Рисунок 3.18 – Розподіл помилки прогнозування прикладу 3.3.4

Значення квадратичної помилки зображено на рисунку 3.18.

Коефіцієнт детермінації точного та наближеного розв'язку становить $R^2 = 0.997$. Відносна похибка визначення коефіцієнта кінематичної в'язкості ν становить 0.47%.

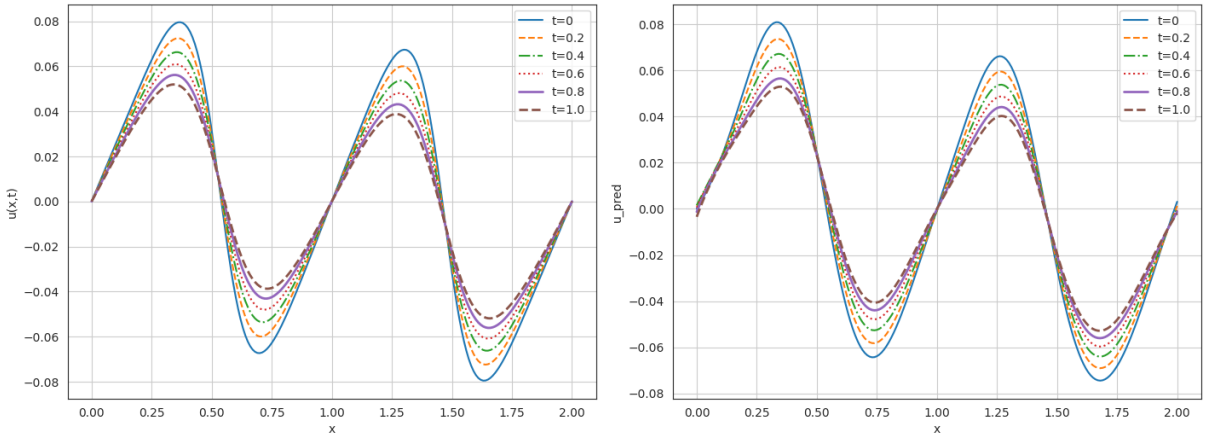
Приклад 3.3.5. Область визначення $x \in [0, 2]$, $\nu = 0.01$. Граничні умови $u(0, t) = u(2, t) = 0$. Початкова умова:

$$u(x, 0) = 2\nu\pi \frac{\sin(\pi x) + 4\sin(2\pi x)}{4 + \cos(\pi x) + 2\cos(2\pi x)}.$$

Точний розв'язок [100, 103]:

$$u(x, t) = 2\nu\pi \frac{\sin(\pi x)e^{-\pi^2\nu t} + 4\sin(2\pi x)e^{-4\pi^2\nu t}}{4 + \cos(\pi x)e^{-\pi^2\nu t} + 2\cos(2\pi x)e^{-4\pi^2\nu t}}.$$

На рисунку 3.19 зображено точні та наближені розв'язки [84].

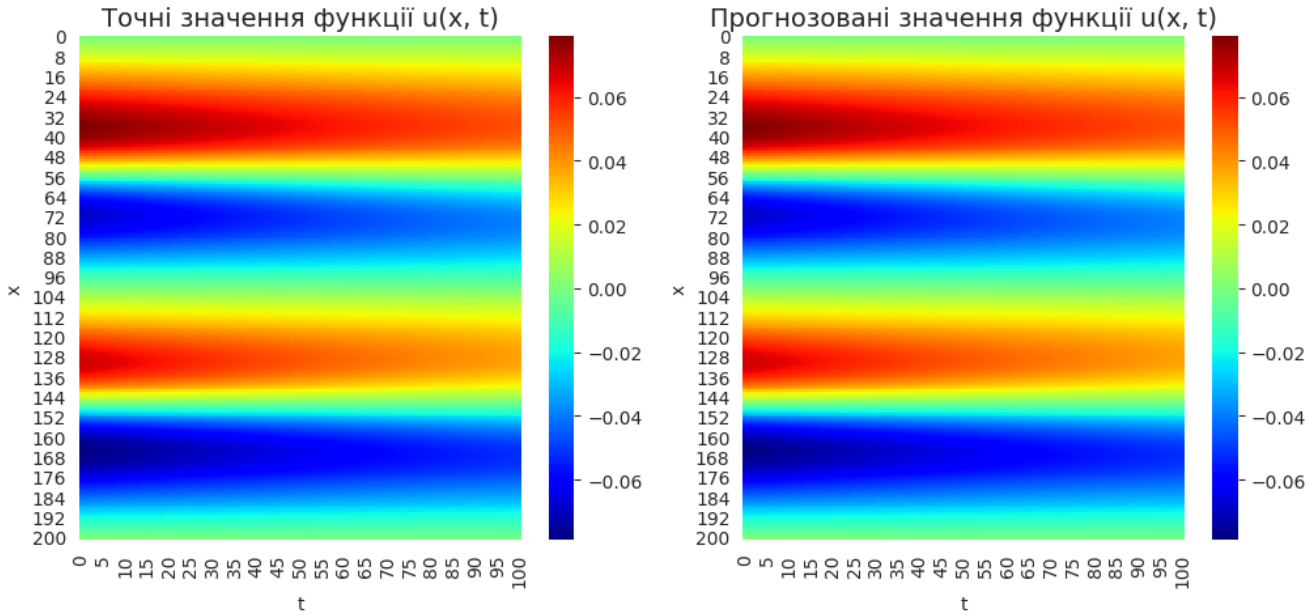


а) б)

Рисунок 3.19 – Точний (а) та наближений (б) розв’язки прикладу 3.3.5

Коефіцієнт детермінації точного та наближеного розв’язку становить $R^2 = 0.993$. Відносна похибка визначення коефіцієнта кінематичної в’язкості ν становить 4.58%.

Розподіл точного та наближеного розв’язку зображено на рисунку 3.20.



а) б)

Рисунок 3.20 – Точний (а) та наближений (б) розв’язки прикладу 3.3.5

Квадратичну помилку прогнозування розв’язку для прикладу 3.3.5 зображено на рисунку 3.21.

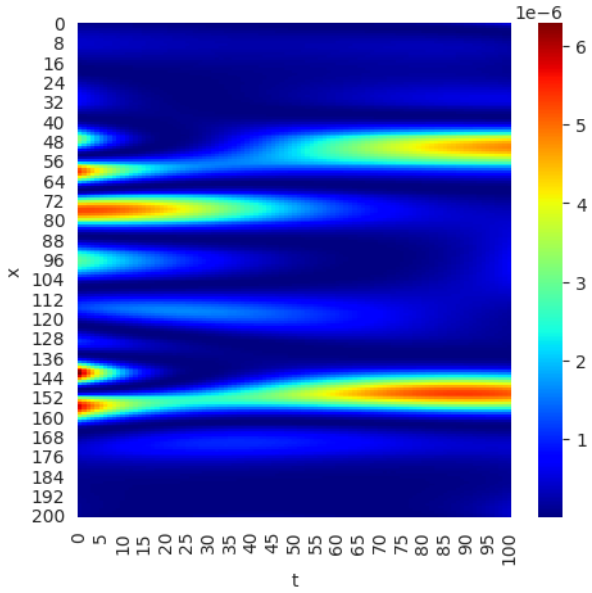


Рисунок 3.21 – Розподіл помилки прогнозування прикладу 3.3.5

В наведених обчислювальних експериментах використовувалась PINN мережа з трьох шарів (рис. 3.22) та 32-х нейронів в кожному шарі. Функція активації – тангенс гіперболічний.

```
NN(
  (net): Sequential(
    (0): Linear(in_features=2, out_features=32, bias=True)
    (1): Tanh()
    (2): Linear(in_features=32, out_features=32, bias=True)
    (3): Tanh()
    (4): Linear(in_features=32, out_features=1, bias=True)
  )
)
```

Рисунок 3.22 – Структура PINN мережі

Використовувались такі гіперпараметри. Оптимізатори Adam (швидкість навчання 0.01) та LBFGS (швидкість навчання 1.0), функцією втрат – середня квадратична помилка. Використовувалось дві тисячі епох навчання. Для навчання

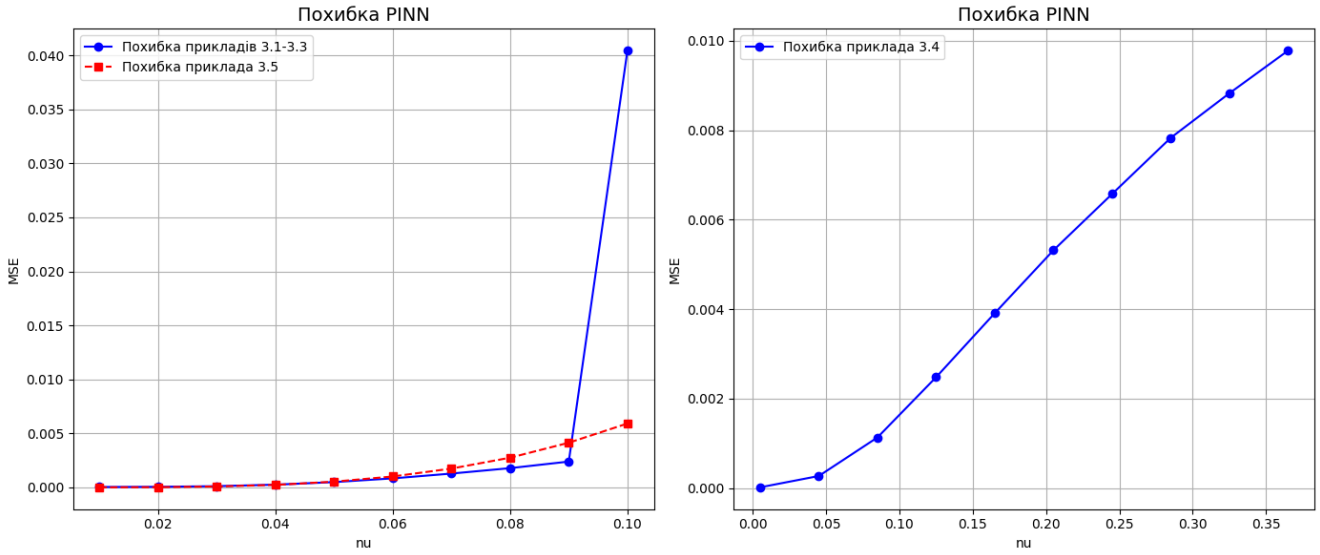
всіх прикладів використовувалась рівномірна сітка точок з кроком 0.01 за координатою та часом.

Репозиторій з програмною реалізацією наведених обчислювальних експериментів засобами бібліотеки PyTroch знаходиться за посиланням https://github.com/avk256/AutoPINN.

3.4 Вплив коефіцієнтів рівнянь на точність розв’язку

З наведених у попередніх розділах прикладів та аналізу публікацій із застосування методу PINN мереж до розв’язання крайових задач, можна зробити висновок про високу точність отриманих чисельних розв’язків. При цьому, слід зазначити, що задовільні результати були отримані при розв’язанні як лінійних, так і нелінійних задач. Однак, згідно з [108] при моделюванні деяких задач математичної фізики із великими за модулем коефіцієнтами, PINN мережі прогнозують значення, які суттєво відхиляються від тестових розв’язків. В [108] пропонується метод навчання за допомогою регуляризації навчального плану, який передбачає поступове ускладнення завдань під час тренування нейронної мережі. Спочатку модель тренується на простіших прикладах (з меншими значеннями коефіцієнтів), а потім поступово переходить до більш складних завдань. Це дозволяє моделі краще узагальнювати точки даних та ефективніше навчатися. У випадку PINN, такий підхід допомагає уникнути збіжності до локальних мінімумів та покращує загальну продуктивність моделі, особливо при розв'язуванні складних фізичних рівнянь.

В цьому підрозділі дослідимо вплив коефіцієнта кінематичної в’язкості середовища ν рівняння Бюргерса прикладів підрозділу 3.3.

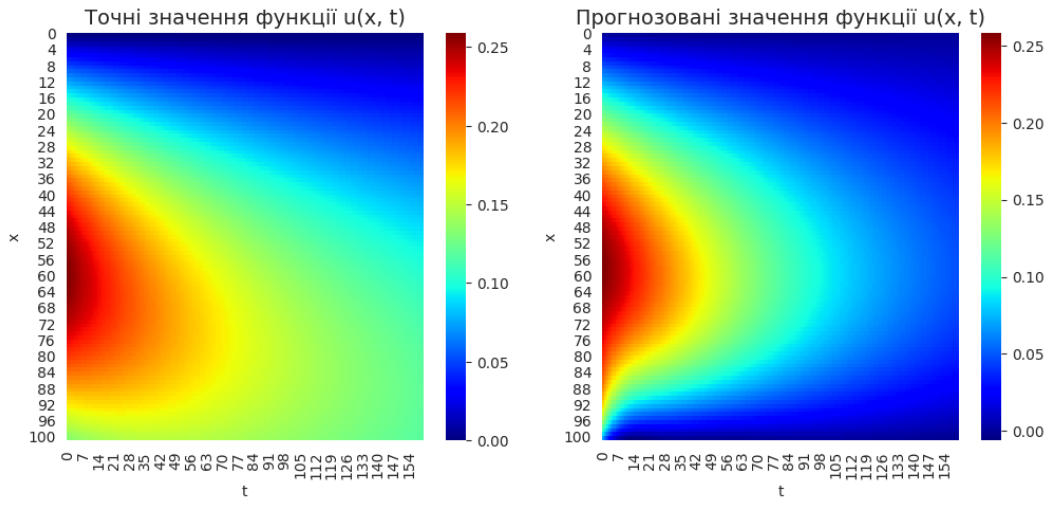


а) б)

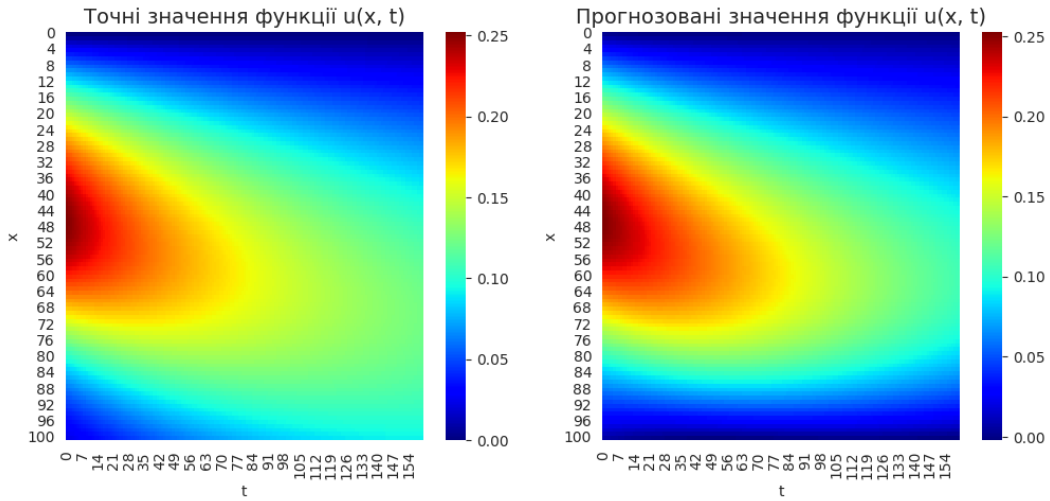
Рисунок 3.23 – Точність PINN із зростанням ν

Для задач 3.3.1–3.3.3 та 3.3.5 значення коефіцієнта змінюється в діапазоні від 0.01 до 0.1 з кроком 0.01. Для задачі 3.3.4 діапазон ν – від 0.005 до 0.365 з кроком 0.04. Відхилення від точного розв’язку обчислюється як середня квадратична похибка прогнозування. Зміну точності моделі із зростання коефіцієнта ν зображено на рисунку 3.23.

Розподіли точної та прогнозованої функції за областю визначення в залежності від ν для задач 3.3.1–3.3.3 зображено на рисунку 3.24.



а) б)

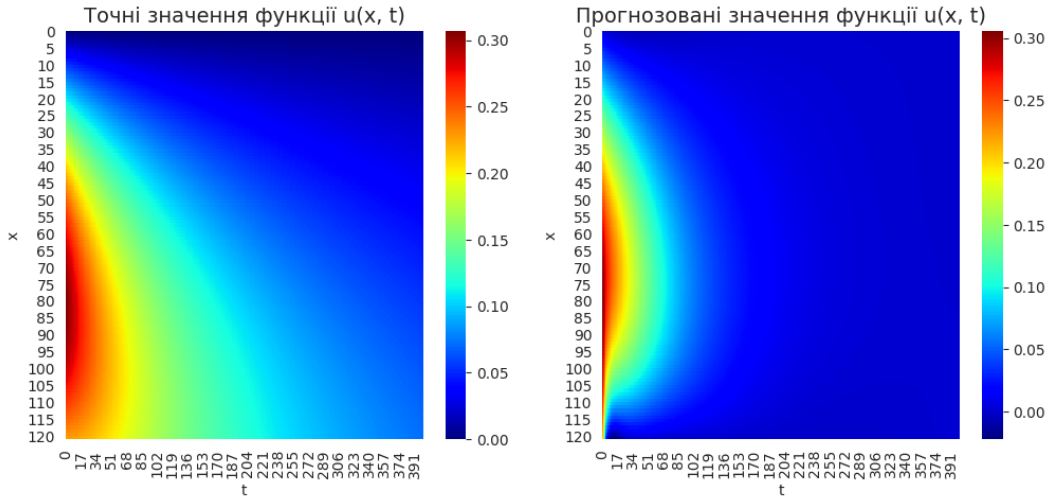


в) г)

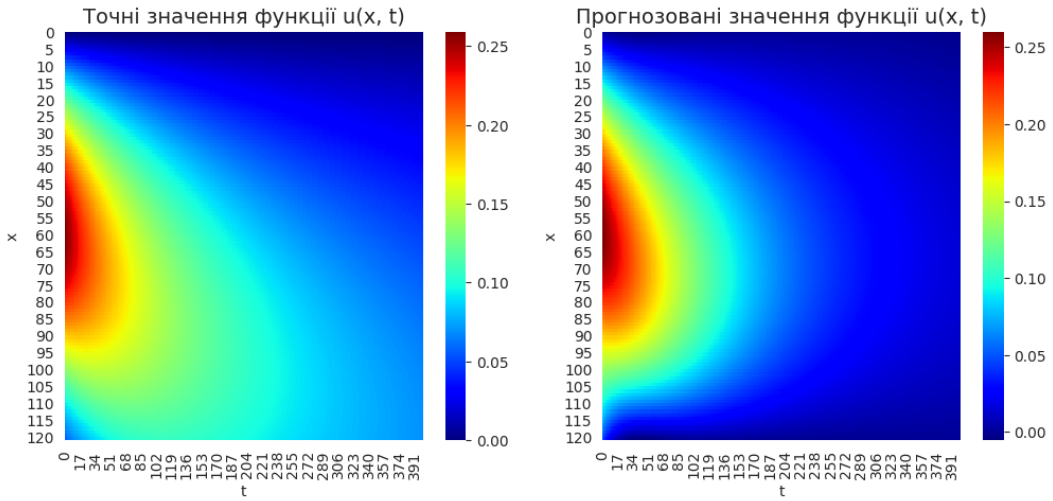
Рисунок 3.24 – Точні та наближені розв’язки прикладів 3.3.1–3.3.3 в залежності від ν

Рисунки 3.24 а), б) зображують відповідно точний та наближений розв’язки при $\nu = 0.1$; в), г) – точний та наближений розв’язки при $\nu = 0.05$. Можна побачити, що із зростанням коефіцієнта кінематичної в’язкості середовища прознозована функція відхиляється від точного розв’язку не тільки за модулями значень, але й змінюється її характер (рис. 3.24 б).

Аналогічно, на рисунках 3.25: а), б) зображено відповідно точний та наближений розв’язки при $\nu = 0.1$; в), г) – точний та наближений розв’язки при $\nu = 0.205$ для прикладу 3.3.4.



а) б)

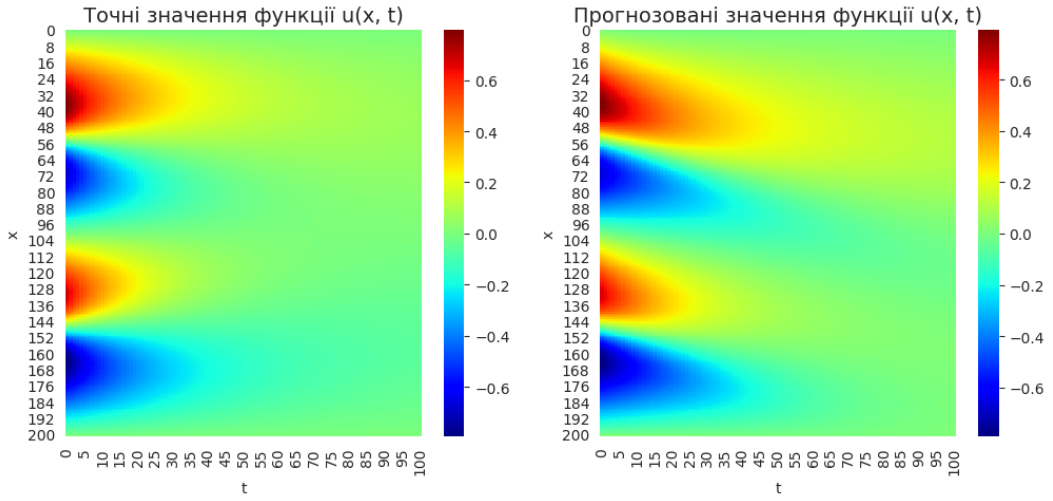


в) г)

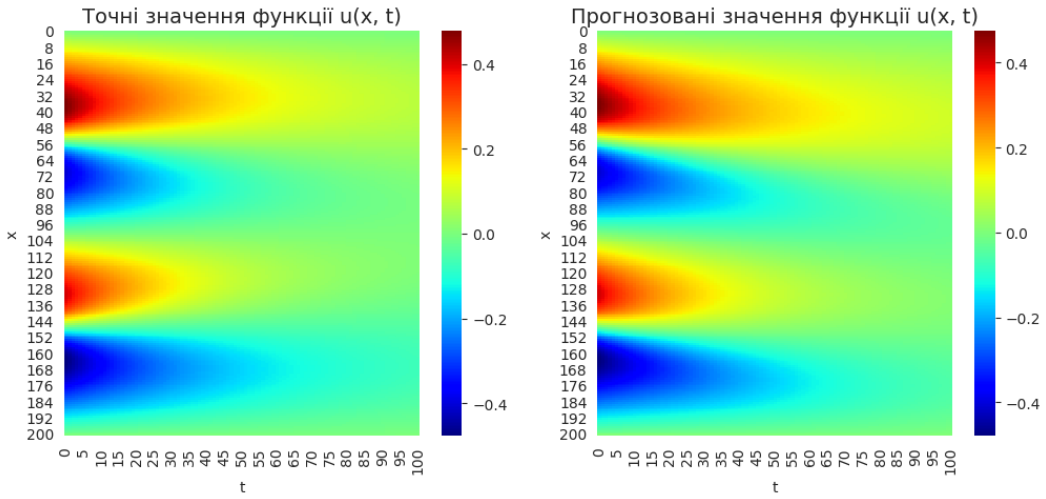
Рисунок 3.25 – Точні та наближені розв’язки приклада 3.3.4 в залежності від ν

Вплив коефіцієнта в’язкості у випадку задачі 3.3.5 розглянуто на рисунку 3.26. Тут а), б) зображують відповідно точний та наближений розв’язки при $\nu = 0.1$; в), г) – точний та наближений розв’язки при $\nu = 0.06$ для прикладу 3.3.5.

Хоча з рисунку 3.24 можна побачити, що похибка плавно зростає при зростанні ν, суттєвим для аналізу діапазонів застосувань розроблених методів є максимальні значення помилок у розподілі за областю, що досліджується.



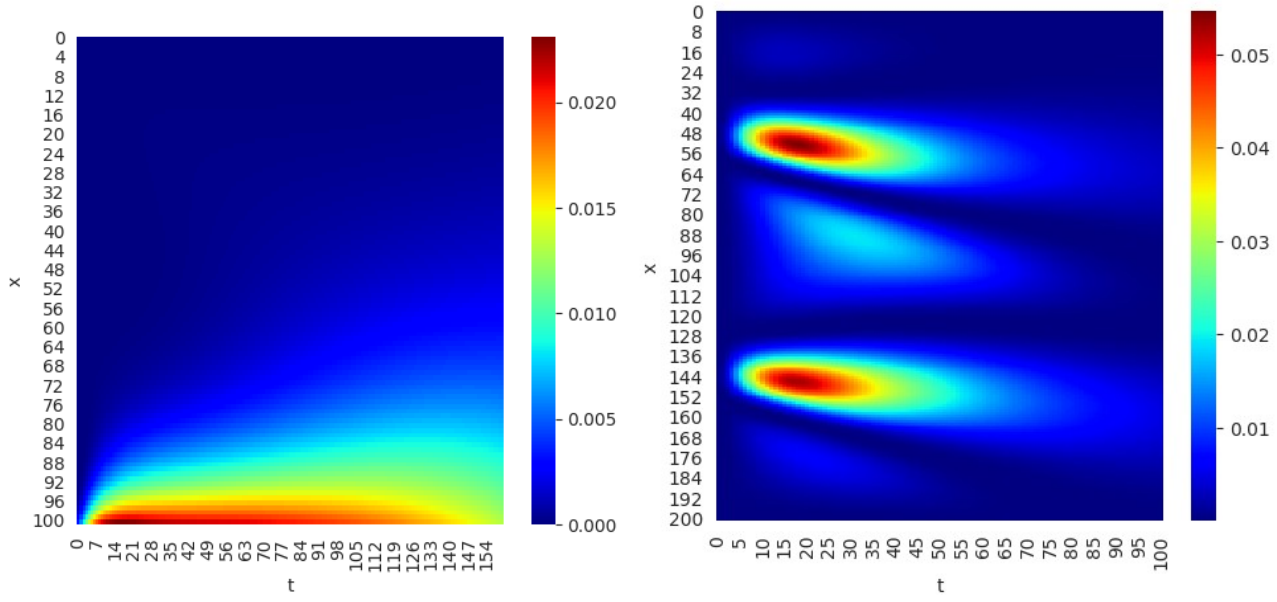
а) б)



в) г)

Рисунок 3.26 – Точні та наближені розв’язки прикладів 3.3.5 в залежності від ν

Наприклад, на рисунку 3.27 наведено розподіл помилок для задач 3.3.1–3.3.3 (а) та 3.3.5 (б) при значенні $\nu = 0.1$.



а) б)

Рисунок 3.27 – Розподіл помилок в області визначення
для задач 3.3.1–3.3.3 та 3.3.5

Отже, наведені результати є важливими для визначення діапазону застосувань нейронних мереж з фізичною інформацією до розв’язання рівняння

Бюргерса з наведеними граничними умовами. Це є актуальною задачею, оскільки дозволяє забезпечити точність і стабільність розв’язків, а також ефективно використовувати обчислювальні ресурси.

Крім того, слід зазначити, що в цьому підрозділі аналізувались числова стабільність розв’язку при різних значеннях коефіцієнта ν без аналізу фізичного змісту наведених значень.

Одним з можливих методів подолання цієї проблеми, окрім розглянутих у [108], є побудова більш глибоких нейромереж, які здатні моделювати складніші нелінійні паттерни в даних. Процес пошуку оптимальною структури мережі в даному випадку не є інтуїтивним і потребує використання методів автоматичного пошуку оптимальних гіперпараметрів, наприклад, еволюційних алгоритмів або методів планування експериментів. Розгляду цих підходів присвячено розділ 4.

Висновки до розділу 3

Отже, в третьому розділі розв’язано ряд тестових модельних задач засобами нейромереж з фізичною інформацією. Всі отримані результати порівнюються з наявними в літературі, показано, що розроблені моделі мають високу точність. Зокрема, розв’язано лінійні та нелінійні задачі пружності балок та пластин. Розв’язано рівняння Бюргерса для різних граничних та початкових умов, а також для різного діапазону коефіцієнтів. Показано вплив значень коефіцієнтів рівняння Бюргерса на збіжність апроксимуючої нейронної мережі.

Основні наукові і практичні результати даного розділу опубліковано в роботах [84, 86, 87, 90].

4 МЕТОДИ АКТИВІЗАЦІЇ ПОШУКУ ОПТИМАЛЬНИХ ГІПЕРПАРАМЕТРІВ МЕРЕЖІ

4.1 Оптимізація гіперпараметрів засобами генетичних алгоритмів

Зазвичай, в розглянутих у розділах 1-3 публікаціях із застосування нейромереж з фізичною інформацією, більше уваги приділяється деталям реалізації алгоритмів та числовим результатам, однак, не досить докладно розглядається питання оптимізації гіперпараметрів PINN мереж, наприклад, кількості шарів, тип функції активації і т.д.

Для автоматизації пошуку оптимальних нейронних мереж, широко використовуються еволюційні алгоритми, описані у роботі [105].

Розглянемо далі використання генетичного алгоритму для оптимізації гіперпараметрів нейромережі.

Приклад 4.1. Розв’яжемо рівняння $\frac{dy}{dx} = y(x)x + y(x)^2, y(1) = 1$.

Точний розв’язок в спеціальних функціях має такий вигляд (https://cutt.ly/tZXpJHM):

$$y = \frac{2e^{x^2/2}}{-2\sqrt{2\pi}erfi(x/\sqrt{2}) + 2\sqrt{2\pi}erfi(1/\sqrt{2}) + 2\sqrt{e}},$$

де $erfi()$ – функція помилок Гауса.

Для оптимізації гіперпараметрів нейронної мережі в роботі використовується класичний генетичний алгоритм Дж. Г. Голланда [105].

Формально методи генетичного пошуку можуть бути описані у вигляді такої функції [85, 88]:

$$Gen = Gen(P_0, N, L, f, \Omega, \Psi, \theta, T),$$

де $P_0 = (H_1^0, H_2^0, ..., H_N^0)$ – початкова популяція – множина рішень задачі, поданих у вигляді хромосом; $H_j^0 = (h_{1j}^0, h_{2j}^0, ..., h_{Lj}^0)$ – *j*-та хромосома популяції P_0, набір значень гіперпараметрів, поданих у вигляді генів; h_{ij}^0 – i-ий ген j-ої хромосоми популяції P_0 – значення i-го оптимізованого параметру задачі, що входить в j-те рішення; N – кількість хромосом в популяції; L – довжина хромосом, кількість генів; f – цільова функція; Ω – оператор відбору; Ψ – оператор схрещування; θ – оператор мутації; T – критерії зупинення [85, 88].

В термінах цього підходу, хромосома є закодованим варіантом нейронної мережі певної структури, тобто гени хромосоми відповідають певним гіперпараметрам мережі, як вказано далі:

1. Тип ініціалізатора ядра.
2. Тип ініціалізатора зсуву.
3. Кількість шарів.
4. Кількість нейронів (однакова для кожного шару).
5. Тип оптимізатора.
6. Розмір пакету навчання.
7. Тип функції активації.
8. Швидкість навчання.

Всі гени кодуються цілими числами. Наприклад, ген швидкості навчання кодується словником з ключами від 0 до 4 та значеннями 0.01, 0.001, 0.0001, 0.00001 відповідно. Мінімальне значення для кількості шарів 2, максимальне – 10. Кількість нейронів в кожному шарі однакова та може змінюватись від 1 до 100. Тип кросовера – одноточковий. Тип мутації – випадковий. Ймовірність мутації – 70%, 60% генів піддаються мутації.

Збіжність значень генів подаються на рисунку 4.1.

З рисунку 4.1 можна побачити збіжність значень генів, тобто структура нейронної мережі є відносно стійкою від генерації до генерації.

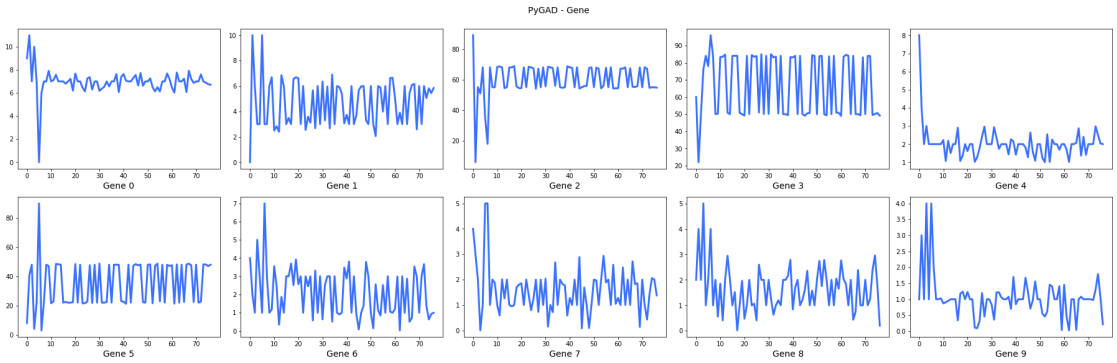


Рисунок 4.1 – Значення генів за генераціями

Порівняння наближеного та точного розв’язку наведено на рисунку 4.2.

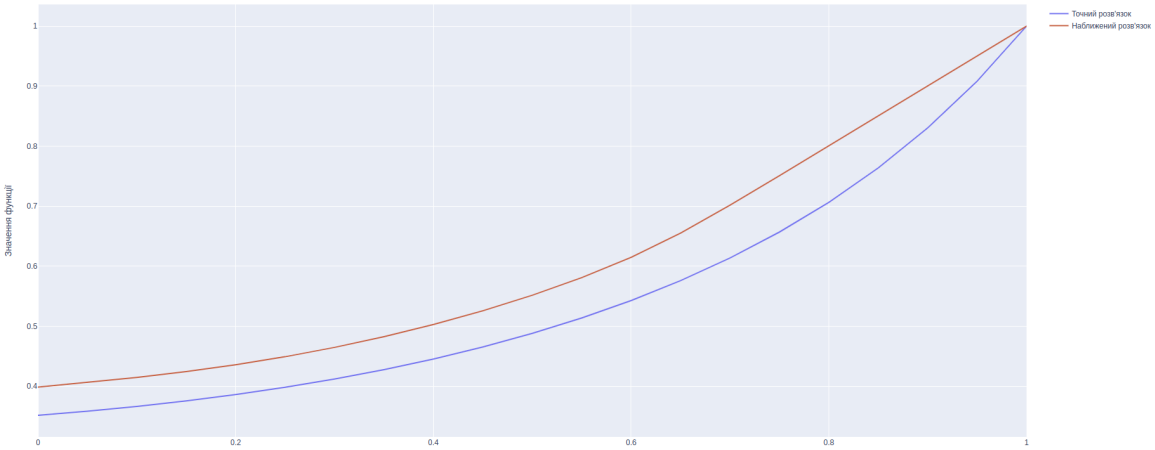


Рисунок 4.2 – Точний та наближений розв’язок

Отже, в роботі запропоновано розширення алгоритму [4] з використанням генетичного алгоритму. Результати свідчать про збіжність алгоритму, відносна точність наближення в даному обчислювальному експерименті 10%.

Слід зазначити, що вибір способу кодування потенційних архітектур нейромереж – хромосом, методу кросовера та мутації безпосередньо впливають на результат роботи генетичних алгоритмів. Структура нейромережі, зазвичай, кодується цілими числами, які можуть бути ключами словників із типовими значеннями параметрів, наприклад, швидкості навчання. Оператор мутації в цьому випадку доцільно обирати типу перемішування.

4.2 Оптимізація гіперпараметрів засобами методу рою часток

Метод рою часток (Particle Swarm Optimization, PSO) є ще одним алгоритмом оптимізації, який використовує популяційний підхід, натхнений соціальною поведінкою рою птахів або зграї риб [105]. Основна ідея PSO полягає в тому, що кожна частка в рої представляє потенційне розв'язання задачі і рухається в просторі розв'язків під впливом своєї власної найкращої позиції та найкращої позиції серед сусідів.

Основні кроки алгоритму PSO пошуку оптимальних гіперпараметрів PINN мереж такі [105].

Ініціалізація. Спочатку визначається кількість часток у рої. Кожна частка ініціалізується випадковою позицією та швидкістю в просторі розв'язків. Для кожної частки обчислюється значення цільової функції, що визначає якість розв'язання. Цільова функція, відповідно, повинна обчислювати метрику якості PINN мережі з визначеними координатами частки. Координати частки є значеннями гіперпараметрів.

Оновлення швидкості та позиції. Кожна частка рухається в просторі розв'язків, оновлюючи свою швидкість та позицію. Швидкість частки оновлюється за формулою

$$\mathrm{v}_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i^{best} - x_i(t)) + c_2 \cdot r_2 \cdot (g^{best} - x_i(t)),$$

де $v_i(t)$ – швидкість частки i в момент часу t, w – коефіцієнт інерції, c_1 і c_2 – коефіцієнти прискорення, r_1 і r_2 – випадкові числа в інтервалі [0, 1], p_i^{best} – найкраща позиція частки i, g^{best} – найкраща позиція серед усіх часток.

Позиція частки оновлюється за формулою:

$$x_i(t+1) = x_i(t) + v_i(t+1).$$

Оцінка та оновлення найкращих значень. Для кожної частки обчислюється нове значення цільової функції в новій позиції. Якщо нове значення цільової функції краще, ніж найкраще знайдене часткою раніше p_i^{best}, то оновлюється p_i^{best}.

Якщо нове значення цільової функції краще, ніж найкраще знайдене всіма частками раніше g^{best}, то оновлюється g^{best}.

Завершення. Процес оновлення швидкостей, позицій та оцінок повторюється до досягнення критерію зупинки, який може бути максимальним числом ітерацій або досягненням певного рівня якості розв’язання.

Метод рою часток є простим у реалізації та ефективним для багатьох задач оптимізації, що робить його популярним у багатьох прикладних областях, таких як машинне навчання, оптимізація виробничих процесів та інші.

Метод рою часток [105] і генетичний алгоритм є поширеними еволюційними методами оптимізації, які застосовуються для визначення оптимальних гіперпараметрів нейромереж. Обидва методи мають свої переваги та недоліки, і їх ефективність може залежати від конкретної задачі. Метод рою часток імітує соціальну поведінку рою птахів або зграї риб, де кожна частка представляє потенційне рішення і рухається в просторі розв’язків під впливом власного досвіду та досвіду інших часток. Оновлення позицій і швидкостей часток відбувається на основі найкращих знайдених рішень як індивідуальною часткою, так і всім роєм [105].

PSO здійснює паралельний пошук, використовуючи багато часток одночасно, що дозволяє ефективніше використовувати обчислювальні ресурси. Це часто призводить до швидшої збіжності до оптимального рішення завдяки колективній взаємодії часток. GA також може виконувати паралельний пошук завдяки одночасній обробці популяції індивідів, проте може потребувати більше часу для збіжності через складність і необхідність виконання багатьох операцій відбору, схрещування та мутації [105].

Метод рою часток має порівняно простіші налаштування, такі як кількість часток та коефіцієнти інерції та прискорення. Він легше налаштовується і реалізується, оскільки не вимагає складних генетичних операторів. Натомість

генетичний алгоритм вимагає налаштування кількості поколінь, розміру популяції, ймовірностей схрещування та мутації, що робить його більш гнучким у моделюванні складних процесів відбору та схрещування [105].

Стійкість розглянутих підходів до локальних мінімумів є важливим аспектом при розв’язанні задачі пошуку оптимальних гіперпараметрів нейромереж з фізичною інформацією. PSO добре працює у просторі з гладкими функціями, але може застрягати в локальних мінімумах при недостатній кількості часток або неправильних параметрах. GA завдяки операціям мутації має кращу стійкість до локальних мінімумів і може досліджувати більш широкий простір рішень.

Розглянемо процес визначення оптимальних гіперпараметрів для PINN мережі, що розв’язує крайову задачу рівняння Бюргерса з початковими та граничними умовами з прикладу 3.3.1, підрозділу 3.3.

```
#@title Create Evaluation Function and Register
def evaluate(individual):
    from sklearn.metrics import r2_score
    hp_in = hp.next(individual)
    net_model = train_function(hp_in)
    #---------- test data -------------------------------
    h = 0.01
    k = 0.01
    x = torch.arange(0, net_model.x_max+h,h)
    t = torch.arange(1, net_model.t_max+k,k)
    X = torch.stack(torch.meshgrid(x,t)).reshape(2,-1).T
    X = X.to(net_model.X.device)
    X = X.double()
    #------------ test pred --------------------
    model =  net_model.model
    with torch.no_grad():
      y_pred = model(X)
      y_pred = y_pred.reshape(len(x),len(t)).cpu().numpy()
    #-------------------------------------------
    fitness = r2_score(U_val, y_pred)
    return fitness,

#@title Add Functions to Toolbox
toolbox = base.Toolbox()
toolbox.register("particle",
                 generate, size=hp.size(), pmin=-.25, pmax=.25, smin=-.25, smax=.25)
toolbox.register("population",
                 tools.initRepeat, list, toolbox.particle)
toolbox.register("update",
                 updateParticle, phi1=2, phi2=2)
toolbox.register("evaluate", evaluate)
```

Рисунок 4.3 – Реєстрація параметрів алгоритму PSO засобами бібліотеки Deap

На рисунку 4.3 зображено програмний код визначення алгоритму PSO засобами бібліотеки Deap [109].

На рисунку 4.4 зображено динаміку зміни значень фітнес-функції алгоритму, яка визначається як метрика R^2 точних та прогнозованих даних.

Рисунок 4.4 – Обчислення точності моделі та реєстрація параметрів алгоритму PSO засобами бібліотеки Deap

На рисунку 4.5 наведено структуру згенерованої автоматично нейромережі з оптимальними гіперпараметрами кількості шарів та нейронів.

```
NN(
  (net): Sequential(
    (0): Linear(in_features=2, out_features=32, bias=True)
    (1): Tanh()
    (2): Linear(in_features=32, out_features=32, bias=True)
    (3): Tanh()
    (4): Linear(in_features=32, out_features=1, bias=True)
  )
)
```

Рисунок 4.5 – Структура оптимізованої моделі методом PSO

Гіперпараметри швидкість навчання, кількість епох дорівнюють 0.01, 1000 відповідно. Можна побачити, що в цілому, згенерована структура відповідає тій, яка використовувалась у підрозділі 3.3.

4.3 Оптимізація гіперпараметрів методом планування експериментів

Еволюційні методи, завдяки своїй здатності до глобальної оптимізації та гнучкості, є ефективними у великих і складних просторах параметрів. Вони можуть знайти оптимальні рішення навіть у випадках, коли простір гіперпараметрів є високодисперсним та складним для традиційних методів. Однак вони часто потребують значних обчислювальних ресурсів і можуть бути повільними через велику кількість необхідних ітерацій.

Методи планування експериментів забезпечують системний підхід до дослідження простору гіперпараметрів, дозволяючи зменшити кількість необхідних обчислювальних експериментів і ефективніше використовувати ресурси комп’ютерних систем. Вони особливо корисні в ситуаціях, коли потрібно швидко і точно дослідити обмежений простір параметрів. Проте, ці методи можуть вимагати детального попереднього аналізу і можуть бути менш ефективними у випадках з дуже великим або сильно змінним простором параметрів [93, 106].

Мета методів планування експериментів полягає у розробці ефективної стратегії для проведення експериментів, яка дозволяє зменшити кількість необхідних проб та ресурсів, забезпечуючи при цьому точність та надійність отриманих результатів [93, 106].

Основні методи оптимального планування експериментів включають різноманітні підходи, що дозволяють ефективно досліджувати простір факторів та їх комбінації з метою отримання максимально точної та надійної інформації при мінімальній кількості експериментів.

Повний факторний дизайн (Full Factorial Design) включає всі можливі комбінації рівнів факторів. Він забезпечує повну інформацію про основні ефекти та взаємодії між факторами. Дробовий факторний дизайн (Fractional Factorial Design) використовує лише частину комбінацій факторів, зменшуючи кількість

експериментів, зберігаючи при цьому інформацію про основні ефекти та деякі взаємодії [106].

Генералізований дизайн підмножин (Generalized Subset Design, GSD) є підходом до планування експериментів, який дозволяє ефективно досліджувати простір можливих комбінацій факторів шляхом розбиття їх на підмножини [107]. Цей метод зменшує кількість необхідних експериментів, одночасно забезпечуючи високу точність та надійність результатів. Розглянемо далі покроковий опис методу GSD для оптимізації гіперпараметрів методу розв’язання крайових задач.

Визначення факторів та рівнів. На початковому етапі визначаються всі фактори, які будуть досліджуватися, та їхні можливі рівні. Фактори можуть бути як кількісними (наприклад, типи функцій активацій, оптимізаторів, метрик похибок тощо), так і якісними (наприклад, кількість шарів, нейронів) [107].

Розбиття факторів на підмножини. Фактори розбиваються на підмножини таким чином, щоб у кожній підмножині було невелике число факторів. Це розбиття здійснюється на основі попереднього знання про систему або за допомогою статистичних методів для забезпечення збалансованого розподілу [107].

Планування експериментів у кожній підмножині. Для кожної підмножини факторів створюється план експериментів. Це можуть бути повний факторний дизайн або дробовий факторний дизайн, в залежності від кількості факторів у підмножині та ресурсів, доступних для проведення експериментів [107].

Проведення експериментів. Експерименти проводяться згідно зі створеним планом. Важливо забезпечити точність і послідовність у проведенні всіх експериментів, щоб зібрані дані були надійними [107].

Аналіз даних підмножин. Дані, зібрані з кожної підмножини експериментів, аналізуються окремо. Мета цього аналізу – виявити основні ефекти та взаємодії факторів у кожній підмножині. Статистичні методи, такі як регресійний аналіз, можуть бути використані для цього [107].

Вибір критичних факторів. На основі аналізу даних визначаються критичні фактори, які мають найбільший вплив на результати. Ці фактори будуть включені в подальший детальний аналіз [107].

Оптимізація. На основі побудованої моделі проводиться оптимізація параметрів. Це може включати додаткові експерименти для уточнення оптимальних умов або використання числових методів для знаходження оптимальних значень факторів [107].

Валідація. Останній крок полягає у валідації отриманих результатів. Для цього проводяться додаткові експерименти при оптимальних умовах, щоб підтвердити точність і надійність моделі [107].

Застосування методів планування експериментів у контексті оптимізації гіперпараметрів нейронних мереж дозволяє зменшити кількість необхідних експериментів. Це економить обчислювальні ресурси та час, зменшуючи витрати на проведення великої кількості тренувань моделей. Дослідження допомагають виявити ключові гіперпараметри, що мають найбільший вплив на продуктивність моделі, дозволяючи зосередитися на їх оптимізації. Оптимізація критичних гіперпараметрів може значно покращити точність та узагальнюючу здатність нейронної мережі. Методи планування експериментів забезпечують систематичний підхід до пошуку оптимальних значень гіперпараметрів, зменшуючи випадковість та хаотичність у процесі оптимізації.

Виберемо для обчислювального експерименту деякі фактори, що є гіперпараметрами PINN моделі. Наприклад, кількість точок, кількість прихованих шарів, кількість нейронів, швидкість навчання, тип функції активації та кількість епох (рис. 4.6). Зазначені фактори є важливими при визначенні оптимальної структури нейромережі. Спосіб кодування доцільно обирати, як і у випадку генетичного алгоритму, у вигляді словника. Із ключами – відповідними кодами факторів, а значеннями – типовими величинами, які найчастіше зустрічаються при проєктуванні нейромереж.

```
import pyDOE2
import numpy as np

n_points = {0:0.01, 1:0.1, 2:0.05}
n_hidden = {0:1, 1:2, 2:3, 3:4}
n_units = {0:8, 1:16, 2:32, 3:64}
lr = {0:0.1, 1:0.01, 2:0.001, 3:0.0001}
iter = {0:1000, 1:2000, 2:4000, 3:6000}
activat = {0:'tanh', 1:'sigmoid', 2:'relu'}


# Визначення факторів та рівнів
levels = [len(n_points), len(n_hidden), len(n_units), len(lr), len(iter), len(activat)]
# Зменшити кількість експериментів приблизно до значення reduction
reduction = 13

doe = pyDOE2.gsd(levels, reduction)
```

Рисунок 4.6 – Кодування факторів планування експерименту

Оцінку PINN моделей, що відповідають факторам експериментів згідно методу GSD за допомогою метрики R^2 наведено на рисунку 4.7.

```
[0.9965438095696801, -0.582046370294194, -0.9408813893130561, 0.40194354750376715]
[0.9961426187954642, 0.7710323151555748, 0.7979329778499462, 0.9957649017188026]
[-0.5142798957513082, -0.9408813893130556, -1.5855595013388646, 0.9960601025237763]
[0.9926123479995568, 0.9972374185330717, 0.10874181786931819, 0.996072090757033]
[0.7205917495736355, -0.9408813893130555, -1.5755726589788794, 0.9959321668424423]
[0.787095782893185, 0.8287021414687158, -1.5755726589788794, -1.5060888857140695]
[0.9924111512012243, -0.9408813894245835, 0.996455330477286]
```

Рисунок 4.7 – Оцінка точності моделей, що відповідають експериментам

На рисунку 4.8 наведено структуру згенерованої автоматично нейромережі з оптимальними гіперпараметрами кількості шарів та нейронів.

```
NN(
  (net): Sequential(
    (0): Linear(in_features=2, out_features=8, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=8, out_features=8, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=8, out_features=1, bias=True)
  )
)
```

Рисунок 4.8 – Структура оптимізованої моделі методом GSD

Гіперпараметри швидкість навчання, кількість епох та тип функції активації дорівнюють 0.1, 1000 та $sigmoid$ відповідно. Можна побачити, що в цілому, згенерована структура відповідає тій, яка використовувалась у підрозділі 3.3.

Висновки до розділу 4

У четвертому розділі розглянуто методи оптимізації гіперпараметрів нейромереж з фізичною інформацією. Автоматично розв’язано рівняння Бюргерса та виконано пошук структури нейромережі з фізичною інформацією, яка забезпечує найбільшу точність розв’язання. Використано такі методи як генетичний алгоритм, метод рою часток та метод планування експериментів. Всі використані підходи до оптимізації приводять до схожих архітектур нейромереж та значень гіперпараметрів.

Основні наукові і практичні результати четвертого розділу опубліковано в роботах [85, 88, 89, 93].

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальну науково-технічну задачу підвищення ефективності розробки бібліотек та програмних засобів розв’язання крайових задач засобами нейронних мереж з фізичною інформацією.

Отримано наступні наукові результати:

– виконано аналітичний огляд наявних підходів та бібліотек нейромережевих обчислювальних методів, їх програмної архітектури і способів застосування;

– створено відкриту об’єктно-орієнтовану архітектури бібліотеки нейромережевих обчислювальних методів, яка дозволяє масштабування та додавання нових підходів розв’язання крайових задач;

– реалізовано предметно-орієнтовану мову для формального опису крайових задач та їх розв’язання засобами нейронних мереж;

– розроблено відповідні алгоритмів реалізації нейромережевих методів розв’язання крайових задач;

– створено програмне забезпечення із використанням розробленої бібліотеки для розв’язання модельних крайових задач;

– виконано серію обчислювальних експериментів, які підтвердили ефективність запропонованої архітектури та алгоритмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vladov S., Shmelov Yu., Kotliarov K., Hrybanova S., Husarova O., Derevyanko I., Chyzhova L. Onboard parameter identification method of the TV3-117 aircraft engine of the neural network technologies. Вісник КрНУ імені Михайла Остроградського. Випуск 5/2019 (118), 2019. P. 90-96.

2. Edwards C.H., Penney D.E., Calvis D.T. Differential Equations and Boundary Value Problems: Computing and Modeling. Boston: Pearson, 2014. 797p.

3. Pinder G.F. Numerical Methods for Solving Partial Differential Equations. John Wiley & Sons, Inc., 2018.

4. Karniadakis G.E., Kevrekidis I.G., L.Lu, P. Perdikaris, Wang S., Yang L., Physics-informed machine learning, Nat Rev Phys, vol. 3, no. 6, pp. 422–440, 2021, doi: 10.1038/s42254-021-00314-5.

5. Willard J., Jia X., Xu S., Steinbach M., Kumar V. Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. ACM Comput. Surv., 2022, https://doi.org/10.1145/3514228

6. Cybenko G.V. Approximation by Superpositions of a Sigmoidal function, Mathematics of Control, Signals and Systems, , 1989, vol. 2 no. 4 pp. 303-314

7. Lagaris I.E., Likas A., Fotiadis D.I. Artifial Neural Networks for Solving Ordinary and Partial Differential Equations. arXiv:physics/9705023v1, 1997, https://doi.org/10.1109/72.712178

8. Raissi M., Perdikaris P., Karniadakis G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 2019, 686–707.

9. Sirignano J., Spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations. arXiv:1708.07469v5, 2018.

10. Weinan E, Bing Yu. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Commun. Math. Stat.,2018, 6:1–12. https://doi.org/10.1007/s40304-018-0127-z

11. Hongwei Guo, Timon Rabczuk, and Xiaoying Zhuang. A Deep Collocation Method for the Bending Analysis of Kirchhoff Plate. arXiv:2102.02617v1, 2021.

12. Cai Z., Chen J., Liu M., Liu X., Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420, 2020, 109707, https://doi.org/10.1016/j.jcp.2020.109707.

13. Uriarte C., Pardo D., Omella A.J. A Finite Element based Deep Learning solver for parametric PDEs. Comput. Methods Appl. Mech. Engrg. 391, 2022, 114562, https://doi.org/10.1016/j.cma.2021.114562

14. Meethal, R.E., Kodakkal, A., Khalil, M. et al. Finite element method-enhanced neural network for forward and inverse problems. Adv. Model. and Simul. in Eng. Sci. 10, 6 (2023). https://doi.org/10.1186/s40323-023-00243-1

15. Paszyński, M., Grzeszczuk, R., Pardo, D., Demkowicz, L. (2021). Deep Learning Driven Self-adaptive Hp Finite Element Method. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds) Computational Science – ICCS 2021. ICCS 2021. Lecture Notes in Computer Science, vol 12742. Springer, Cham. https://doi.org/10.1007/978-3-030-77961-0_11

16. Bai J., Liu G.-R., Gupta A., Alzubaidi L., Feng X.-Q., Gu Y. Physics-informed radial basis network (PIRBN): A local approximation neural network for solving nonlinear PDEs, Computer Methods in Applied Mechanics and Engineering, 415, 2023. ISSN 0045-7825, https://doi.org/10.1016/j.cma.2023.116290.

17. Haitsiukevich K., Ilin A. Improved Training of Physics-Informed Neural Networks with Model Ensembles. arXiv:2204.05108v3, 2023.

18. Likun SChen, Xuzhu Dong, Yifan Wang, et al. Improved PINN-Based Parameters Estimation for Distributed Energy Resources Analysis in Microgrid. TechRxiv. April 08, 2024. DOI: https://doi.org/10.36227/techrxiv.171259952.20643985/v1

19. Hongpeng Ren, Xiangyun Meng, Rongrong Liu, Jian Hou, Yongguang Yu. A class of improved fractional physics informed neural networks, Neurocomputing, Volume 562, 2023, 126890, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2023.126890.

20. Yifei Zong, QiZhi He, Alexandre M. Tartakovsky, Improved training of physics-informed neural networks for parabolic differential equations with sharply perturbed initial conditions, Computer Methods in Applied Mechanics and Engineering, Volume 414, 2023, 116125, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2023.116125.

21. Yunpeng Gao, Li Qian, Tianzhi Yao, Zuguo Mo, Jianhai Zhang, Ru Zhang, Enlong Liu, Yonghong Li, "An Improved Physics-Informed Neural Network Algorithm for Predicting the Phreatic Line of Seepage", Advances in Civil Engineering, vol. 2023, Article ID 5499645, 11 pages, 2023. https://doi.org/10.1155/2023/5499645

22. Lin S, Chen Y. The improved backward compatible physics-informed neural networks for reducing error accumulation and applications in data-driven higher-order rogue waves. Chaos. 2024;34(3):033139. https://doi.org/10.1063/5.0191283

23. Wen Zhou, Shuichiro Miwa, Koji Okamoto; Advancing fluid dynamics simulations: A comprehensive approach to optimizing physics-informed neural networks. Physics of Fluids 1 January 2024; 36 (1): 013615. https://doi.org/10.1063/5.0180770

24. Bai, Y., Chaolu, T. & Bilige, S. Solving Huxley equation using an improved PINN method. Nonlinear Dyn 105, 3439–3450 (2021). https://doi.org/10.1007/s11071-021-06819-z

25. Lu Lu, Xuhui Meng, Zhiping Mao, George Em Karniadakis. DEEPXDE: A Deep Learning Library For Solving Differential Equations. SIAM Review 2021 63:1, 208-228. https://doi.org/10.1137/19M1274067

26. Seo J. Solving real-world optimization tasks using physics-informed neural computing. Scientific Reports, 14(1), 202, 2024.

27. Radbakhsh S.H., Zandi K., Nikbakht M. Physics-informed neural network for analyzing elastic beam behavior. Structural Health Monitoring, 2023.

28. Duy T.N. Trinh, Khang A. Luong, Jaehong Lee, An analysis of functionally graded thin-walled beams using physics-informed neural networks, Engineering Structures, Volume 301, 2024, 117290, ISSN 0141-0296, https://doi.org/10.1016/j.engstruct.2023.117290.

29. Fallah A., Aghdam M.M. Physics-informed neural network for bending and free vibration analysis of three-dimensional functionally graded porous beam resting on elastic foundation. Engineering with Computers 40, 437–454 (2024). https://doi.org/10.1007/s00366-023-01799-7

30. Khang A. Luong, Thang Le-Duc, Jaehong Lee, Deep reduced-order least-square method—A parallel neural network structure for solving beam problems, Thin-Walled Structures, Volume 191, 2023, 111044, ISSN 0263-8231, https://doi.org/10.1016/j.tws.2023.111044

31. Bolandi, H., Sreekumar, G., Li, X. et al. Physics informed neural network for dynamic stress prediction. Appl Intell 53, 26313–26328 (2023). https://doi.org/10.1007/s10489-023-04923-8

32. Faroughi, S., Darvishi, A. & Rezaei, S. On the order of derivation in the training of physics-informed neural networks: case studies for non-uniform beam structures. Acta Mech 234, 5673–5695 (2023). https://doi.org/10.1007/s00707-023-03676-2

33. Maziyar Bazmara, Mohammad Silani, Mohammad Mianroodi, Mohsen Sheibanian. Physics-informed neural networks for nonlinear bending of 3D functionally graded beam. Structures 49 (2023) 152–162, https://doi.org/10.1016/j.istruc.2023.01.115

34. Vahab M., Haghighat E., Khaleghi M., Khalili N. A Physics Informed Neural Network Approach to Solution and Identification of Biharmonic Equations of Elasticity. arXiv:2108.07243, 2021

35. Kapoor T., Wang H., Núñez A., Dollevoet R. Transfer Learning For Improved Generalizability In Causal Physics-Informed Neural Networks For Beam Simulations. http://arxiv.org/abs/2311.00578, 2023.

36. M. Bazmara, M. Mianroodi, and M. Silani, Application of physics-informed neural networks for nonlinear buckling analysis of beams, Acta Mech. Sin. 39, 422438 (2023), https://doi.org/10.1007/s10409-023-22438-x

37. Wang L, Liu G, Wang G, Zhang K. M-PINN: A mesh-based physics-informed neural network for linear elastic problems in solid mechanics. Int J Numer Methods Eng. 2024; 125(9):e7444. doi: https://doi.org/10.1002/nme.7444

38. Abueidda DW, Lu Q, Koric S. Meshless physics-informed deep learning method for three-dimensional solid mechanics. Int J Numer Methods Eng. 2021; 122(23): 7182–7201. https://doi.org/10.1002/nme.6828

39. Finol D, Lu Y, Mahadevan V, Srivastava A. Deep convolutional neural networks for eigenvalue problems in mechanics. Int J Numer Methods Eng. 2019; 118: 258–275. https://doi.org/10.1002/nme.6012

40. Luo Z, Wang L, Lu M. A stepwise physics-informed neural network for solving large deformation problems of hypoelastic materials. Int J Numer Methods Eng. 2023; 124(20): 4453–4472. doi: https://doi.org/10.1002/nme.7323

41. Gie, G. M., Hong, Y., & Jung, C. Y. (2024). Semi-analytic PINN methods for singularly perturbed boundary value problems. Applicable Analysis, 1–18. https://doi.org/10.1080/00036811.2024.2302405

42. Lee, M., Kim, H., Joe, H. et al. Multi-channel PINN: investigating scalable and transferable neural networks for drug discovery. J Cheminform 11, 46 (2019). https://doi.org/10.1186/s13321-019-0368-1

43. Yin Z, Li GY, Zhang Z, Zheng Y, Cao Y. SWENet: A Physics-Informed Deep Neural Network (PINN) for Shear Wave Elastography. IEEE Trans Med Imaging. 2024;43(4):1434-1448. doi: https://doi.org/10.1109/TMI.2023.3338178

44. Van-Thien Tran, Trung-Kien Nguyen, H. Nguyen-Xuan, Magd Abdel Wahab. Vibration and buckling optimization of functionally graded porous microplates using BCMO-ANN algorithm, Thin-Walled Structures, Volume 182, Part B, 2023, 110267, ISSN 0263-8231, https://doi.org/10.1016/j.tws.2022.110267.

45. Zhongmin Huang, Linxin Peng, An improved plate deep energy method for the bending, buckling and free vibration problems of irregular Kirchhoff plates,

Engineering Structures, Volume 301, 2024, 117235, ISSN 0141-0296, https://doi.org/10.1016/j.engstruct.2023.117235.

46. Huijuan Z., Juncai P., Yong C. Data-driven forward-inverse problems for the variable coefficients Hirota equation using deep learning method. https://arxiv.org/abs/2210.09656, 2022

47. Ming Z., Zhenya Y., Data-driven forward and inverse problems for chaotic and hyperchaotic dynamic systems based on two machine learning architectures, Physica D: Nonlinear Phenomena, Volume 446, 2023, 133656, https://doi.org/10.1016/j.physd.2023.133656.

48. Yinlin Y., Hongtao F., Yajing L., Xinyi L., Hongbing Z. Deep neural network methods for solving forward and inverse problems of time fractional diffusion equations with conformable derivative, Neurocomputing, Volume 509, 2022, Pages 177-192, https://doi.org/10.1016/j.neucom.2022.08.030.

49. Vadeboncoeur A., Akyildiz Ö. D., Kazlauskaite I., Girolami M., Cirak F. Fully probabilistic deep models for forward and inverse problems in parametric PDEs, Journal of Computational Physics, Volume 491, 2023, https://doi.org/10.1016/j.jcp.2023.112369.

50. 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, 16(1), 2022, 21–36. https://doi.org/10.1080/17499518.2021.1971251

51. Garay J., Dunstan J., Uribe S., Costabal F.S. Physics-informed neural networks for blood flow inverse problems, 2023, https://arxiv.org/abs/2308.00927

52. Lu, L., Jin, P., Pang, G. et al. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell 3, 218–229 (2021). https://doi.org/10.1038/s42256-021-00302-5

53. Somdatta Goswami, Aniruddha Bora, Yue Yu, George Em Karniadakis. Physics-Informed Deep Neural Operator Networks. 2022. https://doi.org/10.48550/arXiv.2207.05748

54. Junyan He, Shashank Kushwaha, Jaewan Park, Seid Koric, Diab Abueidda, Iwona Jasiuk. Sequential Deep Operator Networks (S-DeepONet) for predicting full-field solutions under time-dependent loads, Engineering Applications of Artificial Intelligence,Volume 127, Part A,2024,107258,ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2023.107258.

55. Wei Li, Martin Z. Bazant, Juner Zhu, Phase-Field DeepONet: Physics-informed deep operator neural network for fast simulations of pattern formation governed by gradient flows of free-energy functionals, Computer Methods in Applied Mechanics and Engineering, Volume 416, 2023, 116299, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2023.116299.

56. Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering, Volume 403, Part A, 2023, 115671, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2022.115671.

57. Jeremy Yu, Lu Lu, Xuhui Meng, George Em Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Computer Methods in Applied Mechanics and Engineering, Volume 393, 2022, 114823, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2022.114823.

58. L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson. Physics-informed neural networks with hard constraints for inverse design. SIAM Journal on Scientific ComputingVol. 43, Iss. 6 (2021). https://doi.org/10.1137/21M1397908

59. Dongkun Zhang, Lu Lu, Ling Guo, George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, Journal of Computational Physics, Volume 397, 2019, 108850, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2019.07.048.

60. Sifan Wang, Hanwen Wang, Paris Perdikaris. 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,

Volume 384, 2021, 113938, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2021.113938.

61. Min Zhu, Shihang Feng, Youzuo Lin, Lu Lu, Fourier-DeepONet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness, Computer Methods in Applied Mechanics and Engineering, Volume 416, 2023, 116300, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2023.116300.

62. Sifan Wang, Xinling Yu, Paris Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, Journal of Computational Physics, Volume 449, 2022, 110768, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2021.110768.

63. Han Gao, Matthew J. Zahr, Jian-Xun Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Computer Methods in Applied Mechanics and Engineering, Volume 390, 2022, 114502, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2021.114502.

64. Han Gao, Luning Sun, Jian-Xun Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, Journal of Computational Physics, Volume 428, 2021, 110079, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2020.110079.

65. Qin Lou, Xuhui Meng, George Em Karniadakis, Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation, Journal of Computational Physics, Volume 447, 2021, 110676, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2021.110676.

66. Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis, Physics-informed neural networks for high-speed flows, Computer Methods in Applied Mechanics and Engineering, Volume 360, 2020, 112789, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2019.112789.

67. Ameya D. Jagtap, Ehsan Kharazmi, George Em Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied

Mechanics and Engineering, Volume 365, 2020, 113028, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2020.113028.

68. Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional Physics-Informed Neural Networks. SIAM Journal on Scientific Computing 2019 41:4, A2603-A2626

69. Zubov K. , McCarthy Z. , Ma Y. , Calisto F., Pagliarino V., Azeglio S., Bottero L., Luján E., Sulzer V., Bharambe A., Vinchhi N., Balakrishnan K., Upadhyay D., Rackauckas C. NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with Error Approximations, 2021. https://doi.org/10.48550/arXiv.2107.09443.

70. Hennigh O., Narasimhan S., Nabian M.A., Subramaniam A., Tangsali K., Rietmann M., Aguila Ferrandis J., Byeon W., Fang Z., Choudhry S. NVIDIA SimNet™: an AI-accelerated multi-physics simulation framework, 2020. https://doi.org/10.48550/arXiv.2012.07938

71. Ehsan Haghighat, Ruben Juanes. SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Computer Methods in Applied Mechanics and Engineering, Volume 373, 2021, 113552, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2020.113552.

72. Reza Akbarian Bafghi and Maziar Raissi. PINNs-Torch: Enhancing Speed and Usability of Physics-Informed Neural Networks with PyTorch, 2023. URL: https://openreview.net/forum?id=nl1ZzdHpab

73. Zhang Minte, Guo Tong, Zhang Guodong, Liu Zhongxiang and Xu Weijie 2024Physics-informed deep learning for structural vibration identification and its application on a benchmark structurePhil. Trans. R. Soc. A.38220220400. http://doi.org/10.1098/rsta.2022.0400

74. Chen, Z., Lai, S. K., & Yang, Z. (2024). AT-PINN: Advanced time-marching physics-informed neural network for structural vibration analysis. Thin-Walled Structures, 196, Article 111423. https://doi.org/10.1016/j.tws.2023.111423

75. Lawal ZK, Yassin H, Lai DTC, Che Idris A. Physics-Informed Neural Network (PINN) Evolution and Beyond: A Systematic Literature Review and

Bibliometric Analysis. Big Data and Cognitive Computing. 2022; 6(4):140. https://doi.org/10.3390/bdcc6040140

76. Walters G. Python GUI Programming with PAGE: Create professional-looking GUIs for Python applications efficiently and effectively. London: BPB Publications, 2023. 471 p.

77. Wąsowski A., Berger T. Domain-Specific Languages: Effective Modeling, Automation, and Reuse. Cham: Springer, 2023. 486 p.

78. Extended BNF – A generic base standard. URL: https://www.cl.cam.ac.uk/~mgk25/iso-14977-paper.pdf (дата звернення: 18.04.2024).

79. American Standard Code for Information Interchange. URL: https://en.wikipedia.org/wiki/ASCII (дата звернення: 18.04.2024).

80. Misra S., Raghurama Rao S.V., Bobba M. K. Relaxation system based sub-grid scale modelling for large eddy simulation of Burgers' equation. URL: https://www.tandfonline.com/doi/abs/10.1080/10618562.2010.523518 (дата звернення: 21.04.2024).

81. Aho A. V., Lam M. S., Sethi R., Ullman J. D. Compilers: principles, techniques, & tools. London: Pearson/Addison Wesley, 2007. 1038 p.

82. Jones J. Abstract Syntax Tree Implementation Idioms. URL: https://www.hillside.net/plop/plop2003/Papers/Jones-ImplementingASTs.pdf (дата звернення: 11.05.2024).

83. WelcomeTo UML Web Site! URL: https://www.uml.org/ (дата звернення: 12.05.2024).

84. Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання прямих і обернених крайових задач. Computer Science and Applied Mathematics. 2024. (1), 92-100. https://doi.org/10.26661/2786-6254-2024-1-11

85. Ярош А.О., Кудін О.В. Нейроеволюційний метод колокації розв’язання диференціальних рівнянь. Таврійський науковий вісник. Серія: Технічні науки. 2024. (6), 72-81. https://doi.org/10.32782/tnv-tech.2023.6.9

86. Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання задач пружності. Вісник Херсонського національного технічного університету. 2024. № 1(88), 295-305. https://doi.org/10.32782/tnv-tech.2023.6.9

87. Ярош А.О., Кудін О.В. Застосування нейронних мереж у розв’язанні диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Дванадцятої Всеукраїнської, дев’ятнадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2021, 79-80.

88. Ярош А.О., Кудін О.В. Нейроеволюційний метод розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Тринадцятої Всеукраїнської, двадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2022, 61-63.

89. Сачковська А.І, Ярош А.О., Кудін О.В. Нейроеволюційний варіант методу гальоркіна розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Чотирнадцятої Всеукраїнської, двадцять першої регіональної наукової конференції молодих дослідників. Запоріжжя, 2023, 90-92.

90. Ярош А.О., Кудін О.В. Нейромережіві обчислювальні методи у задачах згину функціональноградієнтних балок. Міжнародна наукова конференція «актуальні проблеми механіки — 2023» до 145-річчя від дня народження С.П.Тимошенка, Київ, Дніпро, Львів, Харків – 2023, 227-228.

91. Ярош А.О., Кудін О.В. Об’єктно-орієнтована бібліотека нейромережевих методів розв’язання крайових задач. Актуальні проблеми математики та інформатики : збірка тез доповідей П’ятнадцятої Всеукраїнської, двадцять другої регіональної наукової конференції молодих дослідників. Запоріжжя, 2024, 90-92.

92. Ярош А.О., Кудін О.В. Проєктування бібліотеки нейромережевих методів розв’язання крайових задач. Розвиток наук в умовах нової реальності: проблеми та перспективи: збірник наукових праць з матеріалами II Міжнародної наукової конференції, м. Київ, 3 травня, 2024 р. 143-146. https://doi.org/10.62731/mcnd-03.05.2024.004

93. Ярош А.О., Кудін О.В. Планування експериментів та метод рою часток оптимізації нейромережевих методів розв’язання крайових задач. Інновації та науковий потенціал світу: збірник наукових праць з матеріалами IV Міжнародної наукової конференції, м. Запоріжжя, 24 травня, 2024 р. 160-163. https://doi.org/10.62731/mcnd-24.05.2024.002

94. Haykin S. Neural Networks and Learning Machines (3rd Edition), Prentice Hall, 2009

95. Reddy J.N. Theory and Analysis of Elastic Plates and Shells (3nd Edition), CRC Press, 2007.

96. Кудін О.В., Сторожук Є.А., Чопоров С.В. Наближені аналітичні та чисельні методи аналізу міцності тришарових тонкостінних конструкцій: монографія. Херсон : Видавничий дім “Гельветика”, 2019. 160 с.

97. Vinson J.R. The Behavior of Thin Walled Structures: Beams, Plates, and Shells. Kluwer Academic Publishers, 1989.

98. Wang Z.-Q., Jiang J., Tnag B.-T., Zheng W. High Precision Numerical Analysis of Nonlinear Beam Bending Problems Under Large Deflection. Applied Mechanics and Materials Vols. 638-640. P. 1705-1709, 2014. https://doi.org/10.4028/www.scientific.net/amm.638-640.1705

99. Mohammadpour A., Rokni E., Fooladi M., Kimiaeifar A. Approximate Analytical Solution For Bernoulli-Euler Beams Under Different Boundary Conditions With Non-Linear Winkler Type Foundation. Journal of Theoretical and Applied Mechanics, 50, 2, pp. 339-355, 2012

100. Başhan A. Nonlinear dynamics of the Burgers’ equation and numerical experiments. Math Sci 16, 183–205, 2022. https://doi.org/10.1007/s40096-021-00410-8

101. Apraiz J., Doubova A., Fernández-Cara E., Yamamoto M. Some inverse problems for the Burgers equation and related systems, Communications in Nonlinear Science and Numerical Simulation, Volume 107, 2022, https://doi.org/10.1016/j.cnsns.2021.106113.

102. Oanh N.T.N. On the inverse problem for one-dimensional Burgers' equation from the interior observation. J Elliptic Parabol Equ 9, 1329–1339 (2023). https://doi.org/10.1007/s41808-023-00244-6

103. Benton E.R., Platzman G.W. A table of solutions of the one-dimensional Burgers equation. Quarterly of Applied Mathematics, 30(2):195–212, 1972.

104. Devore J.L. Probability and Statistics for Engineering and the Sciences (8th ed.). Boston, MA: Cengage Learning. 2011, pp. 508–510. ISBN 978-0-538-73352-6.

105. Simon D. Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence. Wiley, 2013. 776 p

106. Antony J. Design of Experiments for Engineers and Scientists. 3rd Edition. Elsevier, 2023. 296 p.

107. Surowiec I., Vikström L., Hector G., Johansson E., Vikström C., Trygg J.Generalized Subset Designs in Analytical Chemistry. Anal. Chem. 2017, 89, 12, 6491–6497. https://doi.org/10.1021/acs.analchem.7b00506.

108. Krishnapriyan S., Gholami A., Zhe S., Kirby R., Mahoney M.W. Characterizing possible failure modes in physics-informed neural networks. Advances in Neural Information Processing Systems, 34, 2021. https://doi.org/10.48550/arXiv.2109.01050

109. François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau and Christian Gagné, "DEAP: A Python Framework for Evolutionary Algorithms", in !EvoSoft Workshop, Companion proc. of the Genetic and Evolutionary Computation Conference (GECCO 2012), July 07-11 2012.

ДОДАТКИ

Додаток А

Список опублікованих праць за темою дисертації

1) Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання прямих і обернених крайових задач. Computer Science and Applied Mathematics. 2024. (1), 92-100. https://doi.org/10.26661/2786-6254-2024-1-11

2) Ярош А.О., Кудін О.В. Нейроеволюційний метод колокації розв’язання диференціальних рівнянь. Таврійський науковий вісник. Серія: Технічні науки. 2024. (6), 72-81. https://doi.org/10.32782/tnv-tech.2023.6.9

3) Ярош А.О., Кудін О.В. Нейромережеві методи розв’язання задач пружності. Вісник Херсонського національного технічного університету. 2024. № 1(88), 295-305. https://doi.org/10.35546/kntu2078-4481.2024.1.41

4) Ярош А.О., Кудін О.В. Застосування нейронних мереж у розв’язанні диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Дванадцятої Всеукраїнської, дев’ятнадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2021, 79-80.

5) Ярош А.О., Кудін О.В. Нейроеволюційний метод розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Тринадцятої Всеукраїнської, двадцятої регіональної наукової конференції молодих дослідників. Запоріжжя, 2022, 61-63.

6) Сачковська А.І, Ярош А.О., Кудін О.В. Нейроеволюційний варіант методу гальоркіна розв'язання нелінійних диференціальних рівнянь. Актуальні проблеми математики та інформатики : збірка тез доповідей Чотирнадцятої Всеукраїнської,

двадцять першої регіональної наукової конференції молодих дослідників. Запоріжжя, 2023, 90-92.

7) Ярош А.О., Кудін О.В. Нейромережіві обчислювальні методи у задачах згину функціональноградієнтних балок. Міжнародна наукова конференція «актуальні проблеми механіки — 2023» до 145-річчя від дня народження С.П.Тимошенка, Київ, Дніпро, Львів, Харків – 2023, 227-228.

8) Ярош А.О., Кудін О.В. Об’єктно-орієнтована бібліотека нейромережевих методів розв’язання крайових задач. Актуальні проблеми математики та інформатики : збірка тез доповідей П’ятнадцятої Всеукраїнської, двадцять другої регіональної наукової конференції молодих дослідників. Запоріжжя, 2024, 137-139.

9) Ярош А.О., Кудін О.В. Проєктування бібліотеки нейромережевих методів розв’язання крайових задач. Розвиток наук в умовах нової реальності: проблеми та перспективи: збірник наукових праць з матеріалами II Міжнародної наукової конференції, м. Київ, 3 травня, 2024 р. 143-146. https://doi.org/10.62731/mcnd-03.05.2024.004

10) Ярош А.О., Кудін О.В. Планування експериментів та метод рою часток оптимізації нейромережевих методів розв’язання крайових задач. Інновації та науковий потенціал світу: збірник наукових праць з матеріалами IV Міжнародної наукової конференції, м. Запоріжжя, 24 травня, 2024 р. 160-163. https://doi.org/10.62731/mcnd-24.05.2024.002

Додаток Б

Акт впровадження у навчальний процес Запорізького національного університету

ЗАТВЕРДЖУЮ
Перший проректор Запорізького
національного університету
_____ О.Г. Бондар
«__» _____ 2024р.

ДОВІДКА

про впровадження результатів дисертаційної роботи Ярош Анастасії Олександрівни «Нейромережеві методи розв'язання крайових задач», виконаної у Запорізькому національному університеті

Комісія у складі:
Голова комісії: завідувач кафедри комп'ютерних наук, доктор технічних наук, доцент Шило Г.М.
Члени комісії: завідувач кафедри фундаментальної та прикладної математики, доктор технічних наук, професор Гребенюк С.М., завідувач кафедри загальної математики, кандидат фізико-математичних наук, доцент Зіновєєв І.В.

Засвідчує, що результати дисертаційної роботи Ярош Анастасії Олександрівни «Нейромережеві методи розв'язання крайових задач», а саме матеріали розділу «Методи активізації пошуку оптимальних гіперпараметрів мережі» використано в освітній діяльності Запорізького національного університету при викладанні освітнього компоненту «Методи штучного інтелекту» (при викладанні матеріалів змістового модулю 6. «Еволюційна оптимізація нейромереж») для освітньо-наукової програми підготовки здобувачів третього (освітньо-наукового) рівня вищої освіти (ступеня доктора філософії) Комп'ютерні науки з галузі знань 12 Інформаційні технології за спеціальністю 122 Комп'ютерні науки

Голова комісії: _____ Шило Галіна Миколаївна
Члени комісії: _____ Гребенюк Сергій Миколайович
_____ Зіновєєв Ігор Валерійович

Додаток В

Початковий код методу get_token()

```
    # Обробка чергової лексеми
    def get_token(self):
        self.token_type = self.token = ''
        # Обробка порожнього рядка
        if len(self.code) == 0:
            self.token = 'end'
            self.token_type = 'delimiter'
            return
        # Ігнорування пробілів
        i = 0
            while i < len(self.code) and (self.code[i] == ' ' or
self.code[i] == '\t'):
            i += 1
        self.code = self.code[i:]
        if len(self.code) == 0:
            self.token = 'end'
            self.token_type = 'delimiter'
            return
        # Обробка кінця рядка
        if len(self.code) and self.code[0] == '\n':
            self.code = self.code[1:]
            self.token = 'eol'
            self.token_type = 'delimiter'
            return
        # Пропуск коментарів
        if self.code[0] == '#':
            i = 0
            while i < len(self.code) and self.code[i] != '\n':
                i += 1
            i += 1
```

```
            self.code = self.code[i:]
            self.token = 'eol'
            self.token_type = 'delimiter'
            return
        # Обробка розділювачів
        if '+-*/()^=,'.find(self.code[0]) != -1:
            self.token = self.code[0]
            self.code = self.code[1:len(self.code)]
            self.token_type = 'delimiter'
            return
        # Обробка чисел
        if self.code[0].isdigit():
            i = 0
            while i < len(self.code) and self.code[i].isdigit():
                i += 1
             self.token = self.code[0:i] if (i < len(self.code)) else
self.code[0:]
            self.code = self.code[i:]
            if len(self.code) > 0 and self.code[0] == '.':
                self.token += '.'
                self.code = self.code[1:]
                i = 0
                while i < len(self.code) and self.code[i].isdigit():
                    i += 1
                 self.token += self.code[0:i] if (i < len(self.code))
else self.code[0:]
                self.code = self.code[i:]
                 if len(self.code) > 0 and (self.code[0] == 'E' or
self.code[0] == 'e'):
                self.code = self.code[1:]
                self.token += 'E'
                if self.code[0] != '+' and self.code[0] != '-':
                    self.say_error('syntax_err')
                self.token += self.code[0]
                self.code = self.code[1:]
```

```
                i = 0
                while i < len(self.code) and self.code[i].isdigit():
                    i += 1
                self.token += self.code[0:i]
                self.code = self.code[i:]
            self.token_type = 'digit'
            return
        # Обробка фуннкцій та змінних
        if self.code[0].isalpha():
            self.token = self.code[0]
            self.code = self.code[1:]
            i = 0
             while i < len(self.code) and (self.code[i].isalpha() or
self.code[i] == '_' or self.code[i].isdigit()):
                i += 1
               self.token += self.code[0:i] if (i < len(self.code))
else self.code[0:]
            self.code = self.code[i:]
            if self.token in functions:
                self.token_type = 'function'
            elif self.token == 'function':
                self.token_type = 'request'
            elif self.token in statements:
                self.token_type = 'statement'
            elif self.token == 'problem':
                self.token_type = 'problem'
            elif self.token == 'constant':
                self.token_type = 'constant'
            else:
                self.token_type = 'variable'
            return
```