Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи машинного навчання у чисельному дослідженні
нелінійних крайових задач для звичайних диференціальних рівнянь
(тема)

Виконав:
здобувач _2_ року навчання, групи _ПМм-23-1 _
Леховіцький Д.О.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Яловега І.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
Кафедра прикладної математики
Рівень вищої освіти другий (магістерський)
Спеціальність 113 Прикладна математика
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
"____" _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Леховіцькому Дмитрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи машинного навчання у чисельному дослідженні нелінійних крайових задач для звичайних диференціальних рівнянь

затверджена наказом по університету від 22 листопада 2024 р. № 1223 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 6 січня 2025 р.

3. Вихідні дані до роботи нелінійні моделі математичної фізики

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв'язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	25 листопада – 1 грудня 2024 р.	виконано
2	Вибір та обґрунтування методу	2 – 8 грудня 2024 р.	виконано
3	Розробка алгоритму і програми	9 – 22 грудня 2023 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	23 – 29 грудня 2024 р.	виконано
5	Робота над текстом пояснювальної записки	30 грудня 2024 р. – 9 січня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2025 р.	виконано

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Яловега І.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 93 с., 10 рис., 1 дод., 13 джерел.

АДАПТИВНА ВИБІРКА, ГЛИБОКЕ НАВЧАННЯ, ДИФЕРЕНЦІАЛЬНІ РІВНЯННЯ, КРАЙОВІ ЗАДАЧІ, МЕТОД КОЛОКАЦІЇ, НАУКОВІ ОБЧИСЛЕННЯ, НЕЙРОННІ МЕРЕЖІ, ЧИСЕЛЬНІ МЕТОДИ.

Об’єкт дослідження – нелінійні крайові задачі для диференціальних рівнянь.

Мета роботи – дослідження ефективності застосування нейронних мереж прямого поширення для чисельного дослідження нелінійних крайових задач для диференціальних рівнянь.

Методи дослідження – аналітичний та експериментальний.

Кваліфікаційна робота присвячена використанню нейронних мереж для чисельного дослідження та розв’язку нелінійних крайових задач для диференціальних рівнянь.

Нейронні мережі використовуються для апроксимації невідомої функції в методі колокацій. Параметри нейронної мережі підбираються шляхом мінімізації функції втрат, що агрегує похибку основного рівняння та граничних умов в обраних точках колокації.

Запропоновано стратегію адаптивного вибору точок колокації для покращення точності та ефективності розв’язку. Програмне рішення поставляється у вигляді Python-пакету neural-pde-solver, що включає в себе всі необхідні функції та класи. Для його реалізації використовується фреймворк машинного навчання PyTorch.

Проведено низку обчислювальних експериментів для крайових задач для одно- та двовимірних нелінійних рівнянь Пуассона, а також для початково-крайових задач для нелінійного рівняння теплопровідності, рівняння Бюргерса та рівнянь Нав’є-Стокса.

ABSTRACT

Introductory note: 93 pages, 10 figures, 1 appendix, 13 sources.

ADAPTIVE SAMPLING, DEEP LEARNING, DIFFERENTIAL EQUATIONS, BOUNDARY VALUE PROBLEMS, COLLOCATION METHOD, SCIENTIFIC COMPUTING, NEURAL NETWORKS, NUMERICAL METHODS.

Object of research – nonlinear boundary value problems for differential equations.

Purpose of the work – study of the efficiency of using feedforward neural networks for numerical investigation of nonlinear boundary value problems for differential equations.

Methods of research – analytical and experimental.

The qualification work is devoted to the use of neural networks for numerical investigation and solution of nonlinear boundary value problems for differential equations.

Neural networks are used for approximation of the unknown function in the collocation method. Parameters of the neural network are fitted by minimizing the loss function that aggregates the error of the main equation and boundary conditions at chosen collocation points.

An adaptive sampling strategy is proposed to improve the accuracy and efficiency of the solution. The software implementation is provided as a Python package neural-pde-solver, which includes all necessary functions and classes. For its implementation, the PyTorch machine learning framework is used.

A series of computational experiments was conducted for boundary value problems for one- and two-dimensional nonlinear Poisson's equations, as well as for initial-boundary value problems for nonlinear heat conduction equation, Burgers' equation and Navier-Stokes equations.

ЗМІСТ

ВСТУП

Актуальність теми. Крайові задачі займають центральне місце серед проблем сучасної прикладної математики. Вони виникають при математичному моделюванні широкого спектру фізичних процесів, таких як теплопровідність, течії рідин, деформація твердих тіл, електромагнітні явища тощо, а в останні десятиліття результати з фізики стали широко застосовуватися і в інших областях науки, наприклад, в економіці чи фінансах.

Особливий інтерес становлять нелінійні крайові задачі, оскільки більшість реальних процесів є принципово нелінійними, проте їх розв'язання класичними методами (метод скінченних різниць, метод скінченних елементів) пов'язане з суттєвими обчислювальними труднощами та часто вимагає спрощень, що можуть призвести до серйозних втрат точності.

Нейронні мережі є перспективним альтернативним підходом до розв'язання нелінійних крайових задач завдяки їх природній здатності до апроксимації нелінійних функцій, можливості автоматичного обчислення похідних довільного порядку та ефективному масштабуванню на паралельних обчислювальних системах.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є дослідження ефективності застосування нейронних мереж прямого поширення для чисельного дослідження нелінійних крайових задач для диференціальних рівнянь.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі «нелінійні крайові задачі для диференціальних рівнянь» та класичних методів її розв'язання;

– дослідити актуальні результати застосування нейронних мереж для чисельного розв'язку простіших лінійних крайових задач для диференціальних рівнянь;

– розробити алгоритм для чисельного розв'язку нелінійних крайових задач для диференціальних рівнянь за допомогою нейронних мереж та сформу-

лювати критерії оцінки його ефективності;

– запропонувати евристичні стратегії покращення ефективності розробленого алгоритму;

– виконати програмну реалізацію розробленого алгоритму за допомогою бібліотеки машинного навчання PyTorch;

– провести низку обчислювальних експериментів для тестових задач.

Об'єктом дослідження є нелінійні крайові задачі для диференціальних рівнянь.

Предметом дослідження є чисельне дослідження нелінійних крайових задач для диференціальних рівнянь за допомогою нейронних мереж.

Методи дослідження. У кваліфікаційній роботі використовуються аналітичний та експериментальний методи дослідження.

Публікації. Проміжні результати, отримані у роботі, було представлено на 28-му Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті» (м. Харків, 16-18 квітня 2024 р. [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Нелінійні крайові задачі для диференціальних рівнянь

Крайові задачі формують важливий клас задач прикладної математики і використовуються для моделювання різноманітних фізичних, хімічних чи економічних явищ.

В найбільш загальному випадку, такі задачі описують поведінку системи з кількох величин в деякій обмеженій області, при цьому основне операторне рівняння описує взаємозв'язок між цими величинами в точках всередині області, а рівняння граничних умов описують їх поведінку на її межі. Якщо стан системи залежить від часу, то часову змінну зазвичай розглядають окремо від просторових і обмежують лише з одного боку. Граничні умови, що описують стан системи на початку відліку при цьому називають початковими.

В прямій крайовій задачі модель системи та всі її параметри відомі, і необхідно знайти функцію, що ставить точкам простору у відповідність значення досліджуваних величин.

Має місце також і обернена задача, коли за певним набором спостережень (наприклад, на межі області) необхідно оцінити невідомі параметри моделі. Якщо оператор основного рівняння є диференціальним, говорять про крайову задачу для диференціального рівняння.

Граничні умови в цьому випадку пов'язують значення невідомої функції та/або її молодших похідних (меншого порядку ніж порядок головного рівняння) на межі області.

Поширеними типами граничних умов є умова Діріхле, в якій фіксується значення шуканої функції на всій межі області чи її частині, та умова Неймана, в якій фіксується значення її першої похідної.

Початкові умови зазвичай також фіксують значення шуканої функції чи її похідної на початку відліку.

Нарешті, якщо диференціальний оператор основного рівняння є нелінійним відносно невідомої функції та її похідних, то говорять про нелінійну крайову задачу для диференціального рівняння.

На задачах такого типу ми й зосередимося в даній роботі через їх велику практичну цінність, адже нелінійні моделі зазвичай точніше представляють реальні явища, аніж їх спрощені лінійні аналоги.

Наведемо кілька класичних прикладів нелінійних диференційних рівнянь. Невідому скалярну функцію позначатимемо як u, а векторну - як $\mathbf{u}$.

Для компактності запису, використовуватимемо наступні диференціальні оператори (за наявності часової залежності - лише по просторовим змінним):

∇u – оператор Гамільтона (вектор з перших часткових похідних по всім змінним);

$\nabla^2 u$ – оператор Лапласа (сума других часткових похідних по всім змінним);

$\nabla \cdot \mathbf{u}$ – дивергенція (сума перших часткових похідних значень функції по відповідним змінним).

Почнемо зі стаціонарного звичайного диференціального рівняння:

$$u'' + e^{-u} = 0. \tag{1.1}$$

Таке рівняння виникає в задачах математичної фізики (наприклад, воно описує стаціонарний розподіл електричного потенціалу в газах, коли густина заряду залежить від електричного потенціалу за експоненційним законом), математичної хімії (в моделях реакційно-дифузійного типу, коли швидкість реакції залежить від концентрації за експоненційним законом) тощо.

Його узагальненням для багатовимірного простору є нелінійне рівняння Пуассона, яке використовується, наприклад, для моделювання стаціонарного розподілу температури в середовищі, теплопровідність в якому залежить від самої температури:

$$\nabla\cdot(k(u)\nabla u)+f(u)=0. \tag{1.2}$$

Нестаціонарне рівняння теплопровідності в загальному випадку має схожий вигляд, проте додатково враховує зміну температури в часі $\frac{\partial u}{\partial t}$:

$$\frac{\partial u}{\partial t}=\nabla\cdot(k(u)\nabla u)+f(u). \tag{1.3}$$

В обох випадках, нелінійність може виникати як через залежність коефіцієнту теплопровідності k від невідомої функції, так і через нелінійну залежність від неї ж зовнішніх сил f.

Іншим класичним прикладом нелінійного диференціального рівняння є рівняння Бюргерса, що описує потік в'язкої рідини в одновимірному випадку:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=\nu\frac{\partial^2 u}{\partial x^2}. \tag{1.4}$$

Нелінійний член $u\frac{\partial u}{\partial x}$ тут відповідає за конвекцію, тобто за рух речовини чи потоку через середовище, тоді як лінійний член $\nu\frac{\partial^2 u}{\partial x^2}$ описує дифузію або в'язкість, тобто процеси, при яких енергія або інші властивості рівномірно розподіляються в просторі.

Узагальненням рівняння Бюргерса для багатовимірного випадку є рівняння Нав'є-Стокса:

$$\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\frac{1}{\rho}\nabla p+\nu\nabla^2\mathbf{u}+\mathbf{f}(\mathbf{u}), \tag{1.5}$$

$$\nabla\cdot\mathbf{u}=0.$$

Невідомими функціями тут є вектор швидкості потоку $\mathbf{u}$ та тиск рідини p. На жаль, пошук точного аналітичного розв'язку розглянутих рівнянь в загальному випадку не є можливим, тож далі ми зосередимося на чисельних методах їх дослідження.

1.2 Чисельні методи дослідження нелінійних крайових задач для диференціальних рівнянь

Як і в більшості розділів математики, розв'язання нелінійної задачі починається із застосування методу, який працює для аналогічної лінійної задачі, з подальшою ліквідацією наслідків.

І справді, деякі із класичних методів можна узагальнити і для нелінійних крайових задач.

Серед них, наприклад, метод скінченних різниць (МСК) [2], який полягає в наступному:

– проводиться дискретизація області за допомогою рівномірної сітки;

– похідні у рівнянні замінюються на їх скінченно-різницеві аналоги в кожній точці сітки;

– отримана система рівнянь відносно значень шуканої функції в точках сітки розв'язується обраним чисельним методом;

– за необхідності, будується інтерполяційний поліном для відновлення значень розв'язку поза точками сітки.

Перевагою даного методу є можливість його застосування для нестаціонарних крайових задач без додаткових адаптацій.

Однак, попри свою простоту, метод складно застосовувати у випадку, якщо геометрія області не є тривіальною, а також він швидко призводить до занадто великих систем рівнянь зі збільшенням розмірності.

Окрім цього, на відміну від лінійних крайових задач, що призводять до систем лінійних алгебраїчних рівнянь, для яких існують ефективні методи

розв’язання, нелінійні задачі призводять до, відповідно, систем нелінійних рівнянь, розв’язок яких є значно складнішою та менш стабільною процедурою.

Другий популярний метод, що може бути застосований для розв’язання нелінійних крайових задач, – це метод скінчених елементів (МСЕ) [3]:

– проводиться дискретизація області шляхом її розбиття на скінченну кількість простих підобластей (елементів);

– шукану функцію апроксимують за допомогою базисних функцій, визначених на кожному елементі;

– початкове рівняння формулюється в слабкій (варіаційній) формі для врахування апроксимації;

– для кожного елемента формується система рівнянь, яка зводиться до глобальної системи рівнянь для вузлових значень шуканої функції;

– отримана система рівнянь розв’язується обраним чисельним методом;

– за необхідності, розв’язок у межах елементів уточнюється на основі вузлових значень та базисних функцій.

Перевагою даного методу є можливість його застосування для областей довільної форми.

На жаль, нелінійність оператора унеможливлює аналітичне інтегрування в пунктах 3-4, що призводить до втрати точності вже на етапі побудови системи.

Отримана система, як і в МСР, є нелінійною і вимагає застосування відповідних методів. Тим не менш, існує багато ефективних реалізацій МСЕ, які активно застосовують навіть для нелінійних задач.

Ще один метод, якому і буде присвячена увага в даній роботі, – це метод колокацій (МК), який передбачає наступну послідовність дій:

– обирається параметричне сімейство функцій в якості наближеного рішення;

– обираються множини точок колокації всередині області та на її межі;

– параметри наближеного рішення підбираються так, щоб воно точно задовольняло диференціальним рівнянням в точках колокації, або, що є більш

практичним, мінімізувало агреговану нев'язку рівнянь в цих точках.

Даний метод, взагалі кажучи, є досить абстрактним і історично не мав широкого поширення, проте з розвитком нейронних мереж, що є універсальними параметричними функціями, та технологій для ефективного обчислення їх похідних довільного порядку та оптимізації їх параметрів, його використання поступово набуває сенсу.

Як і МСЕ, МК можна застосовувати для областей довільної форми, а нелінійність оператора не впливає на саму процедуру, на відміну від МСР та МСЕ. Окрім класичних методів, існують і такі, що були розроблені спеціально під конкретні типи нелінійних операторів. Серед них, наприклад, метод двобічних наближень [4, 5].

1.3 Змістовна та формальна постановка задачі

Нехай задано обмежену область $\Omega \subset \mathbb{R}^n$ з границею $\partial\Omega$, і $\mathrm{T} = [0;+\infty)$. Далі, нехай $\mathbb{U}$ – деякий простір достатньо диференційованих функцій вигляду $u(\mathbf{x},t): \Omega \times \mathrm{T} \to \mathbb{R}$. Нарешті, нехай $\mathcal{G}_\theta$, $\mathcal{B}_\theta$ та $\mathcal{I}_\theta$ - нелінійні диференціальні оператори (де перший має вищий порядок ніж решта), що залежать від параметрів $\theta \in \Theta$.

Тоді початково-крайова задача складається з основного рівняння

$$\mathcal{G}_\theta u = \mathbf{0},\ \mathbf{x} \in \Omega,\ t > 0, \tag{1.6}$$

граничних умов

$$\mathcal{B}_\theta u = \mathbf{0},\ \mathbf{x} \in \partial\Omega,\ t > 0, \tag{1.7}$$

та початкових умов

$$\mathcal{I}_\theta u = \mathbf{0},\ t = 0. \tag{1.8}$$

В прямій задачі за відомих значень параметрів моделі θ необхідно знайти невідому функцію $u \in \mathbb{U}$, що задовольняє основне рівняння, граничні та початкові умови. В оберненій задачі за заданим набором значень функції та її похідних на межі або всередині області необхідно оцінити невідомі параметри моделі θ.

1.4 Постановка задач дослідження

Метою кваліфікаційної роботи є дослідження ефективності застосування нейронних мереж прямого поширення для чисельного дослідження нелінійних крайових задач для диференціальних рівнянь.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі «нелінійні крайові задачі для диференціальних рівнянь» та класичних методів її розв’язання;

– дослідити актуальні результати застосування нейронних мереж для чисельного розв’язку простіших лінійних крайових задач для диференціальних рівнянь;

– розробити алгоритм для чисельного розв’язку нелінійних крайових задач для диференціальних рівнянь за допомогою нейронних мереж та сформулювати критерії оцінки його ефективності;

– запропонувати евристичні стратегії покращення ефективності розробленого алгоритму;

– виконати програмну реалізацію розробленого алгоритму за допомогою бібліотеки машинного навчання PyTorch;

– провести низку обчислювальних експериментів для тестових задач.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Основні відомості про нейронні мережі

Нейронні мережі є одним із ключових інструментів сучасного машинного навчання та наукових обчислень.

Їх здатність апроксимувати довільні функції історично викликала значний інтерес, адже вони могли стати ефективним інструментом для розв'язання задач, де класичні методи виявлялися неефективними.

Стрімкий розвиток програмних бібліотек і апаратного забезпечення для автоматичного диференціювання та паралельних тензорних обчислень у останні роки спричинив вибухове зростання їхньої популярності, сприяв появі найрізноманітніших архітектур та застосуванню технології в усіх можливих сферах людської діяльності.

Осідлати цю хвилю популярності спробуємо і ми, застосувавши їх до розв'язання одного з найважливіших класів задач прикладної математики.

Проте спершу розглянемо основи цих технологій, щоб забезпечити розуміння базових концепцій.

Найпростішим та водночас найважливішим для розуміння класом нейронних мереж є нейронні мережі прямого поширення (див. рис. 2.1). Вони представлять собою параметричні функції $\mathbf{y} = \mathrm{NN}(\mathbf{x};\omega)$, що схематично можуть бути представлені у вигляді направленого графа: Кожен вузол цього графу називається нейроном, а кожен стовпець нейронів – шаром.

Перший та останній шар називаються вхідним та вихідним відповідно, проміжні ж шари називаються прихованими. Словосполучення "пряме поширення" в назві пов'язане з механізмом розрахунку виходів функції:

– для кожного нейрону вхідного шару значення нейрону $a_j^{(0)}$ прирівнюється відповідній компоненті входу функції $\mathbf{x}$;

– значення нейронів шару k обчислюються на основі значень нейронів

шару $k-1$:

$$a_j^{(k)} = f_k\underbrace{\left(\sum_{i=1}^{n_k} w_{ij}^{(k)} a_i^{(k-1)} + b_j^{(k)}\right)}_{=:z_j^{(k)}}, \qquad (2.1)$$

де $w_{ij}^{(k)}$ – ваговий коефіцієнт між нейроном i шару $k-1$ та нейроном j шару k;

$b_j^{(k)}$ – коефіцієнт зсуву для нейрону j шару k;

n_k – кількість нейронів в шарі k;

f_k – функція активації шару k.

– отримані значення нейронів вихідного шару $a_j^{(K)}$ стають виходами **y** для заданих входів **x**.

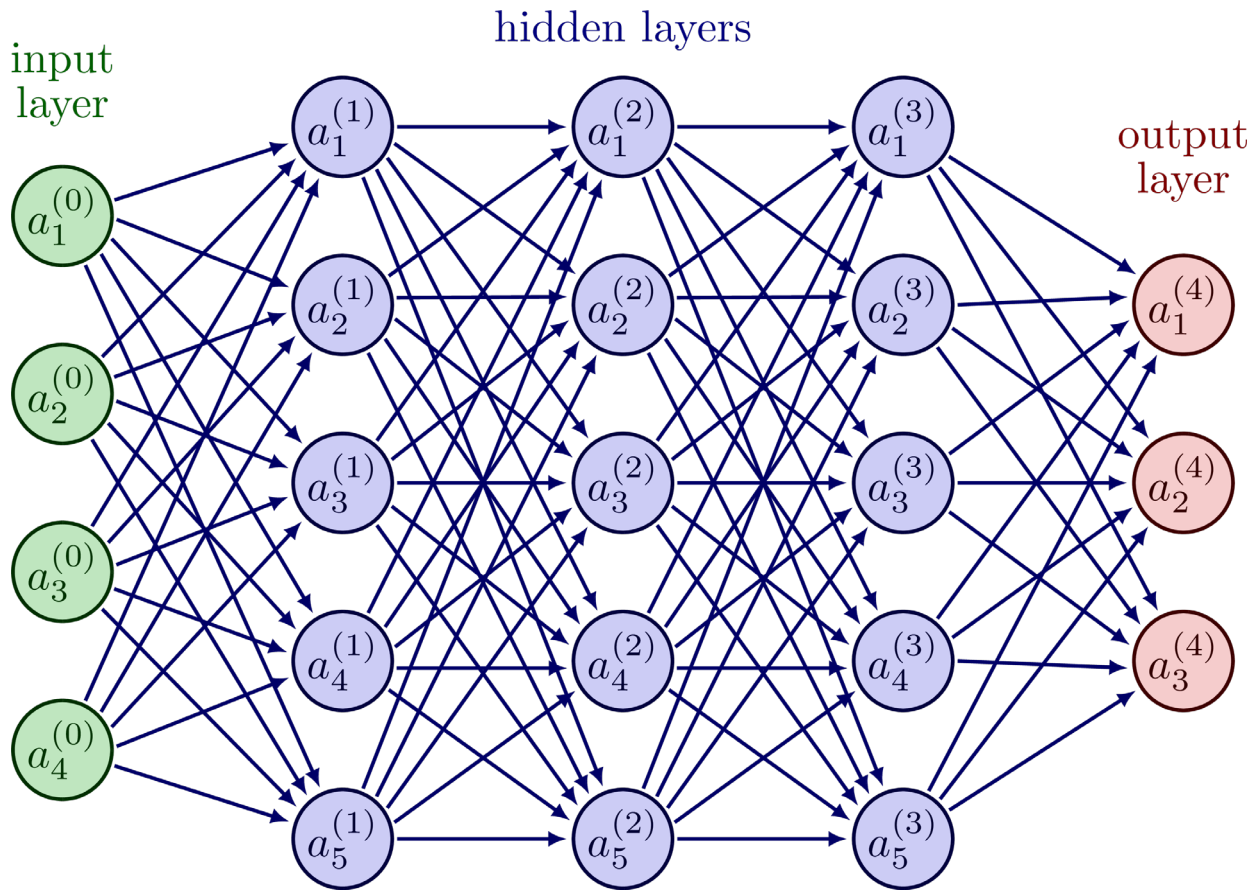

Рисунок 2.1 – Графічне представлення нейромережі прямого поширення

Вектор ω параметрів нейронної мережі складається з вагових коефіцієнтів $w_{ij}^{(k)}$ та коефіцієнтів зсуву $b_j^{(k)}$.

Процесом навчання нейронної мережі називається мінімізація деякої цільової функції відносно цих параметрів.

В задачах регресії, наприклад, цільовою функцією є агрегована похибка апроксимації на тренувальному наборі даних $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$:

$$L(\omega) = \frac{1}{2N} \sum_{i=1}^{N} \left\| \mathbf{y}_i - \mathrm{NN}(\mathbf{x}_i; \omega) \right\|^2. \tag{2.2}$$

Оптимальні параметри шукаються ітеративно за допомогою градієнтного спуску:

$$\omega^{(t+1)} = \omega^{(t)} - \eta \nabla L(\omega^{(t)}), \tag{2.3}$$

де $\omega^{(t)}$ – значення параметрів на ітерації t;

η – параметр, що визначає швидкість навчання.

На практиці для покращення збіжності та зменшення обчислювального навантаження використовують складніші алгоритми (наприклад, стохастичний градієнтний спуск, що на кожній ітерації враховує лише частину тренувального набору даних) та різноманітні евристики.

Важливою властивістю нейронних мереж є існування ефективного алгоритму для підрахунку градієнту $\nabla L(\omega)$ цільової функції відносно параметрів, що називається алгоритмом зворотнього поширення похибки, побудованого на основі правила ланцюжка для диференціювання вкладених функцій:

– для кожного нейрону вихідного шару обчислюється похибка:

$$\delta_j^{(K)} = \frac{\partial L}{\partial a_j^{(K)}} \cdot f_K'(z_j^{(K)}); \tag{2.4}$$

– похибки для нейронів шару k обчислюються на основі похибок для нейронів шару $k+1$:

$$\delta_j^{(k)} = \left(\sum_{i=1}^{n_{k+1}} w_{ji}^{(k+1)} \delta_i^{(k+1)} \right) f_k'(z_j^{(k)}); \tag{2.5}$$

– градієнти цільової функції відносно коефіцієнтів шару k рахуються за наступними формулами (і, за необхідності, агрегуються по всіх точках тренувального набору даних):

$$\frac{\partial L}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} a_i^{(k-1)}, \quad \frac{\partial L}{\partial b_j^{(k)}} = \delta_j^{(k)}. \tag{2.6}$$

Варто відмітити, що в формулах похибок при зворотньому проході мережею використовуються значення $a_j^{(k)}$ та $z_j^{(k)}$ вже пораховані при прямому проході, тож програмні реалізації зазвичай кешують їх для пришвидшення виконання.

Вибір правильної архітектури нейронної мережі (кількості шарів K, кількості нейронів в шарах n_k, функцій активації f_k) є важливою складовою побудови моделі, оскільки від нього залежать її апроксимаційні властивості та чисельна стабільність процесу навчання.

Так, кількість прихованих шарів має бути достатньо великою, аби мережею можна було апроксимувати будь-яку функцію [6].

При цьому, кількість параметрів моделі має бути співрозмірною з обсягом інформації, що міститься в тренувальному наборі даних, аби процес навчання збігся.

Функції активації також мають бути підібрані так, щоб забезпечити достатню нелінійність мережі, при цьому уникаючи чисельних проблем на кшталт вибуху чи затухання градієнтів [7].

Поширені функції активації:

– сігмоїдна функція $\sigma(z) = \dfrac{1}{1+e^{-z}}$;

– гіперболічний тангенс $\tanh(z) = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$;

– зрізана лінійна функція $\mathrm{ReLU}(z) = \max\{z, 0\}$.

Існує безліч інших класів нейронних мереж, що здатні розв'язувати найрізноманітніші задачі.

Серед них, наприклад, рекурентні нейронні мережі, що можуть вловлювати часову залежність в даних і тому є важливим інструментом для моделювання часових рядів, текстів, сигналів тощо.

Існує також і безліч незгаданих проблем, з якими інженерам доводиться зіштовхуватися на практиці.

Всі вони виходять за рамки даної роботи.

2.2 Застосування нейронних мереж для чисельного дослідження нелінійних крайових задач

Розглянувши основні відомості про нейронні мережі, спробуємо тепер використати їх універсальність для чисельного дослідження нелінійних крайових задач для диференціальних рівнянь.

Як вже було анонсовано в підрозділі 1.2, ми будемо використовувати нейронні мережі в якості апроксимаційних функцій в методі колокацій.

Постановка задачі (1.6) – (1.8) є найбільш загальною, проте не зовсім зручною для практичної реалізації.

Почнемо з кількох спрощень, що дозволять краще проілюструвати застосування підходу:

– розглядатимемо лише прямі стаціонарні нелінійні крайові задачі;

– розглядатимемо лише рівняння 2-го порядку відносно скалярних функцій;

– розглядатимемо лише одно- та двовимірні області (для задач більшої розмірності підхід принципово не відрізняється).

Для одновимірного випадку, крайову задачу тепер можна записати наступним чином:

$$G(x,u,u',u'')=0,\ \ x\in(a;b),\qquad (2.7)$$
$$B_a(u,u')=0,\ \ B_b(u,u')=0.$$

Для двовимірного ж випадку, задача має наступний вигляд:

$$G\left(x,y,u,\frac{\partial u}{\partial x},\frac{\partial u}{\partial y},\frac{\partial^2 u}{\partial x^2},\frac{\partial^2 u}{\partial x\partial y},\frac{\partial^2 u}{\partial y^2}\right)=0,\quad (x,y)\in\Omega,$$
$$\forall i: B_i\left(x,y,u,\frac{\partial u}{\partial x},\frac{\partial u}{\partial y}\right)=0,\quad (x,y)\in\partial\Omega_i,\qquad (2.8)$$
$$\bigsqcup_i \partial\Omega_i=\partial\Omega.$$

Невідому функцію u будемо апроксимувати нейронною мережею прямого поширення $\mathrm{NN}(\cdot;\omega)$ з одним виходом та кількістю входів рівною розмірності області.

Параметри цієї мережі навчимо шляхом мінімізації цільової функції $L(\omega)$, що агрегує нев'язку основного та граничних рівнянь в обраних точках колокації. Цільова функція для одновимірного випадку:

$$L(\omega)=\frac{1}{N}\sum_{j=1}^{N} G_j(\omega)^2+B_a(\omega)^2+B_b(\omega)^2,\qquad (2.9)$$

де N – кількість точок колокації;

$G_j(\omega)$ – нев'язка основного рівняння в точці колокації $x_j\in(a;b)$ при підстановці в нього значень мережі $\mathrm{NN}(\cdot;\omega)$ та її похідних, обчислених в ній;

$B_a(\omega)$ – нев'язка граничного рівняння на лівому кінці відрізку при підстановці в нього значень мережі $\mathrm{NN}(\cdot;\omega)$ та її похідних, обчислених в точці a (аналогічно для $B_b(\omega)$).

Цільова функція для двовимірного випадку:

$$L(\omega) = \frac{1}{N}\sum_{j=1}^{N} G_j(\omega)^2 + \sum_i \frac{1}{N_i}\sum_{j=1}^{N_i} B_{ij}(\omega)^2, \tag{2.10}$$

де N – кількість точок колокації всередині області Ω;

N_i – кількість точок колокації на i-й ділянці межі області;

$G_j(\omega)$ – нев'язка основного рівняння в точці колокації $x_j \in \Omega$ при підстановці в нього значень мережі $\mathrm{NN}(\cdot;\omega)$ та її похідних, обчислених в ній;

$B_{ij}(\omega)$ – нев'язка граничного рівняння на i-й ділянці межі області в точці колокації $x_j \in \partial\Omega_i$ при підстановці в нього значень мережі $\mathrm{NN}(\cdot;\omega)$ та її похідних, обчислених в ній.

Точки колокації можна обирати довільним чином, наприклад взявши вузли рівномірної сітки, чи згенерувавши випадково.

Серед евристичних стратегій покращення ефективності алгоритму можна запропонувати адаптивний підхід, що з більшою ймовірністю генерує нові точки колокації біля тих точок всередині області / на ділянках границі, нев'язка відповідних рівнянь в яких на попередній ітерації була вищою.

Схожі ідеї в контексті лінійних крайових задач розглядаються в [8]. Описаний підхід можна застосовувати і для нестаціонарних задач, сприймаючи часову змінну так само як і просторові.

І все ж, доцільніше використовувати не мережу прямого поширення, а архітектуру, що здатна вловити часову залежність, наприклад, Long Short-Term Memory (LSTM) як запропоновано в [9].

Висновки за розділом 2

У даному розділі розглянуто теоретичні основи нейронних мереж прямого поширення та запропоновано підхід до їх застосування для розв’язання нелінійних крайових задач.

Описано архітектуру нейронних мереж, механізми їх навчання та алгоритм зворотнього поширення похибки. Запропоновано використання методу колокацій, де нейронна мережа виступає як апроксимаційна функція, а її параметри оптимізуються шляхом мінімізації агрегованої нев’язки основного рівняння та граничних умов в обраних точках.

Розглянуто як рівномірну, так і адаптивну стратегії вибору точок колокації для покращення ефективності методу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Фреймворк машинного навчання PyTorch

Python є однією з найпопулярніших мов програмування для задач машинного навчання та наукових обчислень [10]. Серед її переваг можна відзначити:

– простий та інтуїтивний синтаксис, що дозволяє швидко прототипувати алгоритми;

– велика кількість бібліотек для наукових обчислень (NumPy, SciPy, Pandas);

– потужні фреймворки для машинного навчання (PyTorch, TensorFlow);

– активна спільнота розробників та дослідників;

– відкритий код та безкоштовність.

PyTorch — це фреймворк машинного навчання з відкритим кодом, розроблений компанією Meta (раніше Facebook) у 2016 році [11]. Він був створений як альтернатива TensorFlow від Google, що на той час домінував на ринку.

Основною перевагою PyTorch стала його простота та гнучкість: на відміну від TensorFlow, який вимагав попереднього визначення обчислювального графу, PyTorch дозволив виконувати обчислення динамічно, що значно спростило процес розробки та налагодження моделей.

Серед ключових можливостей PyTorch можна виділити:

– автоматичне диференціювання тензорних операцій;

– підтримка GPU та розподілених обчислень;

– широкий набір готових шарів та архітектур нейронних мереж;

– інтеграція з іншими інструментами екосистеми Python;

– можливість експорту моделей для промислового використання.

На сьогодні PyTorch є де-факто стандартом в області досліджень глибокого навчання, про що свідчить його використання в більшості наукових публікацій та проєктів з відкритим кодом.

Серед альтернатив можна відзначити:

– TensorFlow – потужний фреймворк від Google, що історично був більш орієнтований на промислове застосування;

– Keras – високорівневий інтерфейс для TensorFlow, що спрощує створення простих моделей;

– MXNet – фреймворк від Amazon, що набув популярності в області хмарних обчислень.

Через свою зручність та гнучкість PyTorch було обрано як основний фреймворк для реалізації запропонованого рішення.

3.2 Опис архітектури пакету neural-pde-solver

Програмне рішення поставляється у вигляді Python-пакету neural-pde-solver, що включає в себе всі необхідні для роботи функції та класи.

За основу його реалізації було взято пакети torchdiffeq [12] та nangs [13], що також використовують нейронні мережі для розв’язання диференціальних рівнянь.

Код пакету та всіх експериментів наведено в додатку А.

Центральним компонентом пакету є клас `BVPSolver`, що реалізує метод колокацій для розв’язання крайових задач за допомогою нейронних мереж.

Конструктор класу приймає наступні аргументи:

– model – нейронна мережа, що буде використовуватися для апроксимації розв’язку;

– pde – основне диференціальне рівняння (або список рівнянь);

– bcs – список граничних умов;

– optimizer_cls – клас оптимізатора для навчання мережі (за замовчуванням Adam);

– optimizer_kwargs – параметри оптимізатора (за замовчуванням швидкість навчання 10^{-3});

– device – пристрій для обчислень (за замовчуванням GPU, якщо доступний).

Основним методом класу є `train`, що виконує навчання мережі.

Його параметри:

– n_epochs – максимальна кількість епох навчання;

– warmup_epochs – кількість епох до початку перевірки умови дострокової зупинки;

– batch_sizes – розміри вибірок для кожного рівняння;

– patience – кількість епох без покращення для дострокової зупинки;

– min_delta – мінімальна зміна функції втрат, що вважається покращенням.

На кожній ітерації навчання алгоритм вкинує такі кроки.

Крок 1. Генерує точки колокації для кожного рівняння (основного та граничних умов) за допомогою відповідних семплерів.

Крок 2. Обчислює нев’язку рівнянь в цих точках, використовуючи автоматичне диференціювання для обчислення похідних.

Крок 3. Оновлює статистику нев’язок в семплерах.

Крок 4. Агрегує нев’язки в єдину функцію втрат як середнє квадратичне відхилення.

Крок 5. Оновлює параметри мережі за допомогою градієнтного спуску.

Алгоритм навчається протягом n_epochs епох, проте додатково реалізовує механізм дострокової зупинки, що працює наступним чином.

Крок 1. Протягом перших warmup_epochs епох умови дострокової зупинки не застосовуються.

Крок 2. Після цього починається відслідковування найкращого значення функції втрат.

Крок 3. Якщо протягом patience епох не було покращення більше ніж на min_delta, навчання зупиняється.

Пакет має модульну структуру і складається з наступних модулів:

- pde/, що містить реалізації різних диференціальних рівнянь:
  - NonlinearPoissonEquation – нелінійне рівняння Пуассона з довільними коефіцієнтами дифузії та джерела;
  - NonlinearHeatEquation – нелінійне рівняння теплопровідності з довільними коефіцієнтами дифузії та джерела;
  - BurgersEquation – рівняння Бюргерса з заданим коефіцієнтом в'язкості;
  - NavierStokesEquation – рівняння Нав'є-Стокса з заданими коефіцієнтами в'язкості та густини;
  - інші рівняння можуть бути легко додані за необхідності.
- bc/, що містить реалізації різних типів граничних умов:
  - DirichletCondition – умова Діріхле (фіксоване значення функції);
  - NeumannCondition – умова Неймана (фіксоване значення похідної);
  - інші типи умов можуть бути легко додані за необхідності.
- model/, що містить реалізації різних архітектур нейронних мереж:
  - MLP – багатошаровий перцептрон з можливістю налаштування кількості та розмірів шарів;
  - інші архітектури можуть бути додані за потреби.
- sampler/, що містить реалізації різних стратегій вибору точок колокації:
  - UniformSampler – рівномірна вибірка точок в заданій прямокутній області;
  - AdaptiveSampler – адаптивна вибірка з урахуванням нев'язки рівнянь.

Така модульна архітектура забезпечує гнучкість та розширюваність рішення, дозволяючи легко додавати нові типи рівнянь, граничних умов та методів вибірки точок колокації.

Висновки за розділом 3

У даному розділі описано програмну реалізацію запропонованого підходу.

Обґрунтовано вибір мови програмування Python та фреймворку машинного навчання PyTorch.

Розроблено модульну архітектуру пакету `neural-pde-solver`, що включає реалізації різних диференціальних рівнянь, граничних умов, архітектур нейронних мереж та стратегій вибору точок колокації.

Описано основні компоненти пакету, їх взаємодію та механізми розширення функціональності.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Одновимірне рівняння Пуассона

Розглянемо одновимірну крайову задачу на відрізку [0; 1], що складається з рівняння

$$u'' + e^{-u} = 0, \quad x \in (0;1) \tag{4.1}$$

та граничних умов

$$u(0) = 0\,, \;\; u'(1) = -1\,. \tag{4.2}$$

Для розв'язання задачі використаємо нейронну мережу з одним вхідним та одним вихідним нейронами, 3 прихованими шарами по 32 нейрони та активаціями $\tanh$. Обрані параметри навчання:

- оптимізатор Adam з швидкістю навчання $\eta = 10^{-3}$;
- n_epochs: 1000;
- warmup_epochs: 100;
- patience: 5;
- min_delta: 10^{-4}.

В цьому експерименті використаємо лише рівномірну вибірку з 1000 точок всередині відрізку та по 1 точці на кожному з його кінців. Алгоритм зупинився після 207 епох із значенням втрат 0.001321 через відсутність покращення протягом 5 епох, що свідчить про досягнення локального мінімуму. Швидкість навчання склала приблизно 330 епох в секунду. Графік зміни втрат та розв'язок наведено на рисунку 4.1.

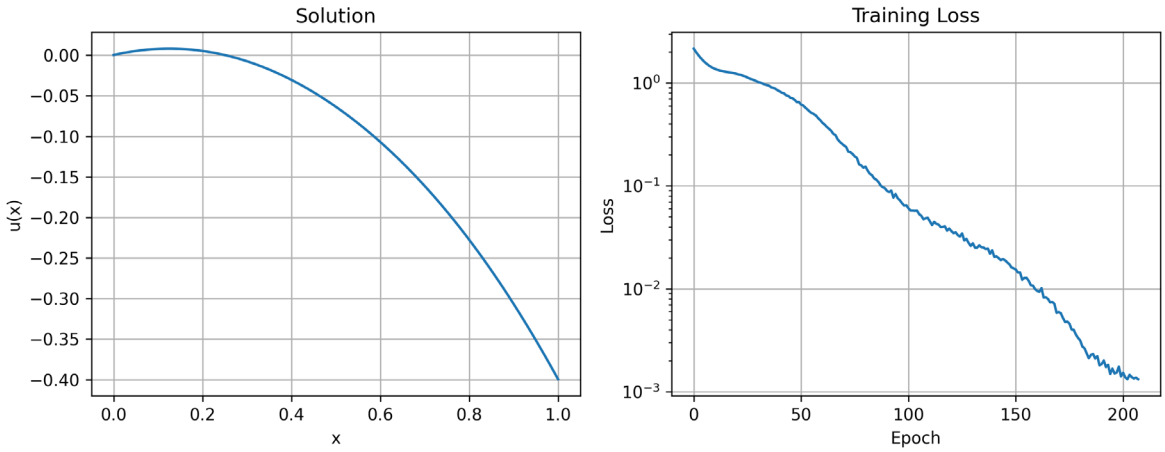

Рисунок 4.1 – Розв’язок одновимірного рівняння Пуассона

Отриманий розв’язок демонструє:

– зменшення втрат з часом, що свідчить про навчання нейронної мережі;

– виконання граничної умови Діріхле $u(0) = 0$;

– виконання умови Неймана $u'(1) = -1$;

– плавну зміну функції, що відповідає фізичній природі задачі.

4.2 Двовимірне рівняння Пуассона

Розглянемо двовимірну крайову задачу на прямокутнику $[0;2]\times[0;1]$, що складається з нелінійного рівняння Пуассона:

$$\nabla\cdot((1+u^2)\nabla u) = \sin(\pi u), \quad (x,y)\in(0;2)\times(0;1) \tag{4.3}$$

та граничних умов:

$$u(x,0) = 0 \ \text{(нижня границя)},$$

$$u(x,1) = x \ \text{(верхня границя)},$$

$$\frac{\partial u}{\partial x}(0,y) = 0 \ \text{(ліва границя)},$$

$$\frac{\partial u}{\partial x}(2, y) = 0 \text{ (права границя).} \tag{4.4}$$

Для розв'язання задачі використаємо нейронну мережу з двома вхідними та одним вихідним нейронами, 4 прихованими шарами по 64 нейрони та активаціями $\tanh$. Обрані параметри навчання:

– оптимізатор Adam з швидкістю навчання $\eta = 10^{-3}$;

– n_epochs: 2000;

– warmu_epochs: 200;

– patience: 100;

– min_delta: 10^{-4}.

Спершу використаємо рівномірну вибірку з 2000 точок всередині області та по 100 точок на кожній з її границь. Алгоритм зупинився після 675 епох із значенням втрат 0.014368 через відсутність покращення протягом 100 епох, що свідчить про досягнення локального мінімуму. Швидкість навчання склала приблизно 56 ітерацій в секунду. Графік зміни втрат та розв'язок наведено на рисунку 4.2.

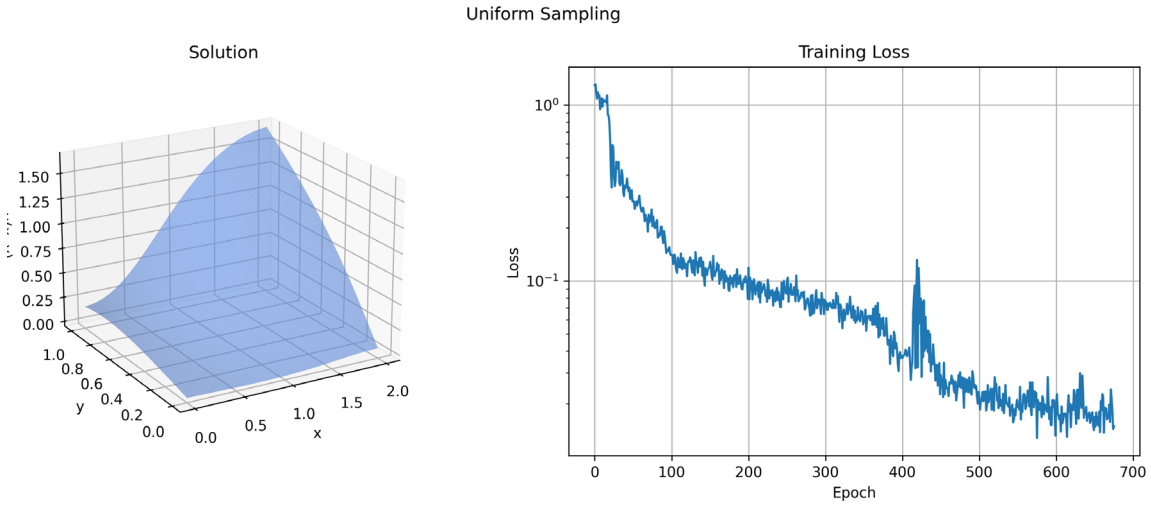

Рисунок 4.2 – Розв'язок двовимірного рівняння Пуассона (рівномірна)

Отриманий розв'язок демонструє:

– зменшення втрат з часом, що свідчить про навчання мережі;

– виконання граничних умов Діріхле на нижній ($u(x,0)=0$) та верхній ($u(x,1)=x$) межах;

– виконання умов Неймана на лівій та правій границях ($\partial u/\partial x=0$);

– плавну зміну функції, що відповідає фізичній природі задачі.

Після цього використаємо адаптивний підхід до вибірки точок. Параметри генератора:

– розбиття області на 32×32 комірок;

– коефіцієнт згладжування $\alpha=0.95$;

– температура $T=1$.

Алгоритм працював дещо довше (779 епох із шивдкістю навчання 25 ітерацій на секунду) і досягнув аналогічних витрат на рівні 0.017972.

Графік зміни втрат та розв’язок наведено на рисунку 4.3.

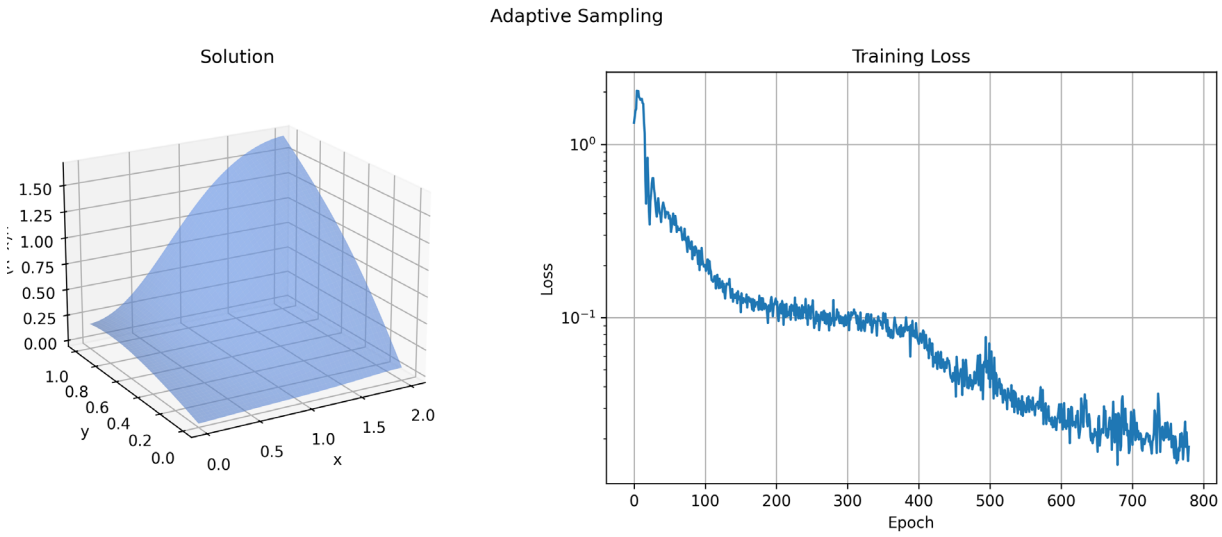

Рисунок 4.3 – Розв’язок двовимірного рівняння Пуассона (адаптивна)

Порівнявши рисунки 4.2 та 4.3, можна стверджувати, що розв’язки з рівномірною та адаптивною вибірками є досить близькими, проте процес навчання при адаптивній вибірці є більш стабільним.

4.3 Рівняння теплопровідності

Розглянемо одновимірну нестаціонарну крайову задачу на відрізку $[-1; 1]$ часовому проміжку $[0; 2]$, що складається з нелінійного рівняння теплопровідності:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left((1+u^2)\frac{\partial u}{\partial x} \right) + \sin(\pi u), \quad (x,t) \in (-1;1)\times(0;2], \tag{4.5}$$

та початково-крайових умов:

$$\begin{gathered} u(x,0) = \sin(\pi x) \text{ (початкова умова)}, \\ u(t,-1) = 0 \text{ (ліва границя)}, \\ u(t,1) = 0 \text{ (права границя)}. \end{gathered} \tag{4.6}$$

Попри те, що задача є нестаціонарною, спробуємо застосувати описаний метод, сприймаючи час як ще одну простову змінну.

Для розв’язання задачі використаємо нейронну мережу з двома вхідними та одним вихідним нейронами, 4 прихованими шарами по 64 нейрони та активаціями $\tanh$.

Обрані параметри навчання:

- оптимізатор Adam з швидкістю навчання $\eta = 10^{-3}$;
- n_epochs: 2000;
- warmup_epochs: 200;
- patience: 100;
- min_delta: 10^{-4}.

Спершу використаємо рівномірну вибірку з 2000 точок всередині області та по 100 точок на кожній з її границь.

Алгоритм зупинився після 861 епохи із значенням втрат 0,077266 через

відсутність покращення протягом 100 епох, що свідчить про досягнення локального мінімуму. Швидкість навчання склала приблизно 56 ітерацій в секунду. Графік зміни втрат та розв’язок наведено на рисунку 4.4.

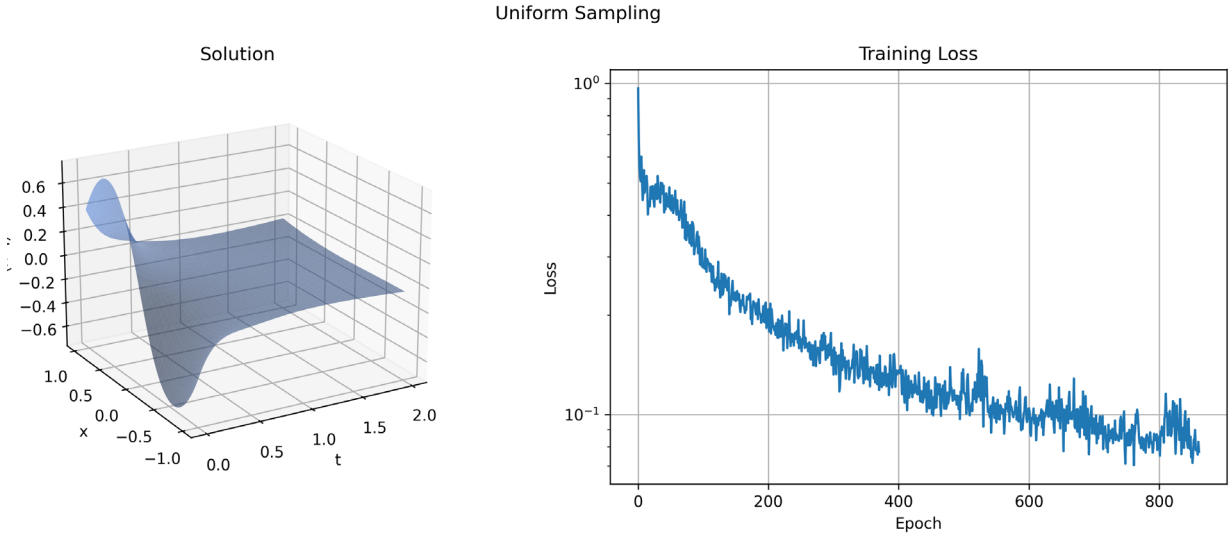

Рисунок 4.4 – Розв’язок рівняння теплопровідності (рівномірна)

Отриманий розв’язок демонструє:

– зменшення втрат з часом, що свідчить про навчання мережі;

– виконання початкової умови $u(x,0)=\sin(\pi x)$;

– виконання граничних умов Діріхле $u(t,-1)=u(t,1)=0$;

– поступовий вихід системи на стаціонарний стан;

– плавну зміну функції, що відповідає фізичній природі задачі.

Після цього використаємо адаптивний підхід до вибірки точок. Параметри генератора:

– розбиття області на 64×64 комірок;

– коефіцієнт згладжування $\alpha=0.95$;

– температура $T=1$.

Алгоритм працював до кінця відведених епох і досягнув втрат на рівні 0,042078. Швидкість навчання склала приблизно 25 ітерацій на секунду.

Графік зміни втрат та розв’язок наведено на рисунку 4.5.

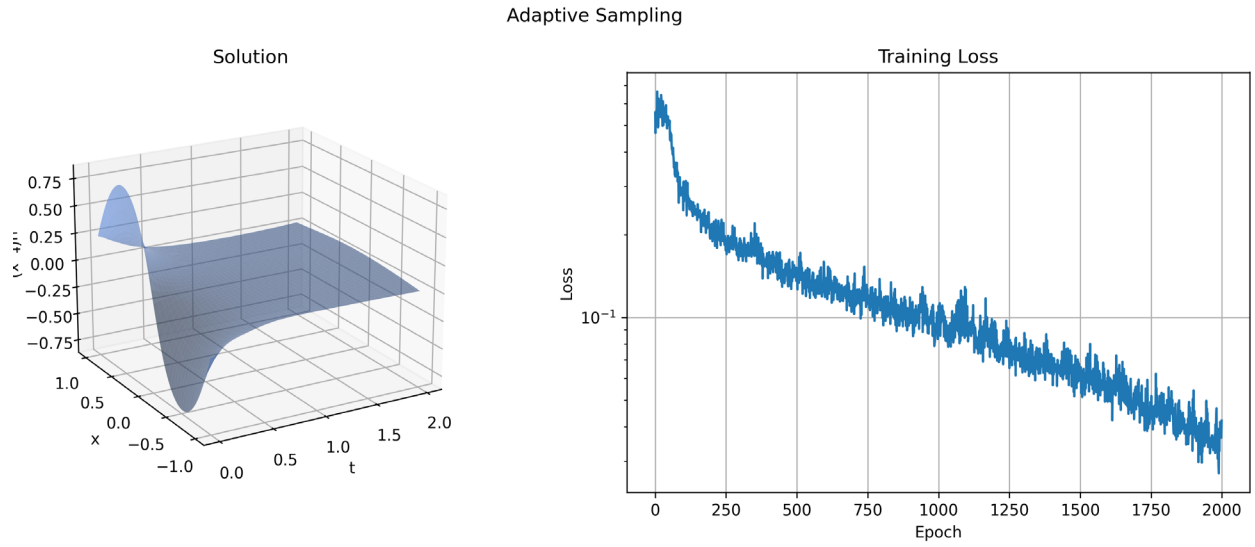

Рисунок 4.5 – Розв’язок рівняння теплопровідності (адаптивна)

Порівнявши рисунки 4.4 та 4.5, можна стверджувати, що розв’язки з рівномірною та адаптивною вибірками є досить близькими, але використання адаптивної вибірки дозволило алгоритму значно краще навчити мережу, оминувши локальні мінімуми.

4.4 Рівняння Бюргерса

Розглянемо одновимірне нестаціонарне рівняння Бюргерса на відрізку $[0; 1]$ та часовому проміжку $[0; 2]$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad (x,t) \in (0;1)\times(0;2] \tag{4.7}$$

з початково-крайовими умовами:

$$u(x,0) = \sin(\pi x) \text{ (початкова умова)}, \tag{4.8}$$

$$u(t,0) = 0 \text{ (ліва границя)},$$

$$u(t,1) = 0 \text{ (права границя)}$$

та коефіцієнтом в'язкості $\nu = 0.1$. Продовжимо сприймати час як просторову змінну.

Для розв'язання задачі використаємо нейронну мережу з двома вхідними та одним вихідним нейронами, 4 прихованими шарами по 64 нейрони та активаціями tanh. Обрані параметри навчання:

– оптимізатор Adam з швидкістю навчання $\eta = 10^{-3}$;

– n_epochs: 2000;

– warmup_epochs: 200;

– patience: 100;

– min_delta: 10^{-4}.

Спершу використаємо рівномірну вибірку з 2000 точок всередині області та по 100 точок на кожній з її границь.

Алгоритм зупинився після 1304 епох із значенням втрат 0.003159 через відсутність покращення протягом 100 епох, що свідчить про досягнення локального мінімуму. Швидкість навчання склала приблизно 87 ітерацій в секунду. Графік зміни втрат та розв'язок наведено на рисунку 4.6.

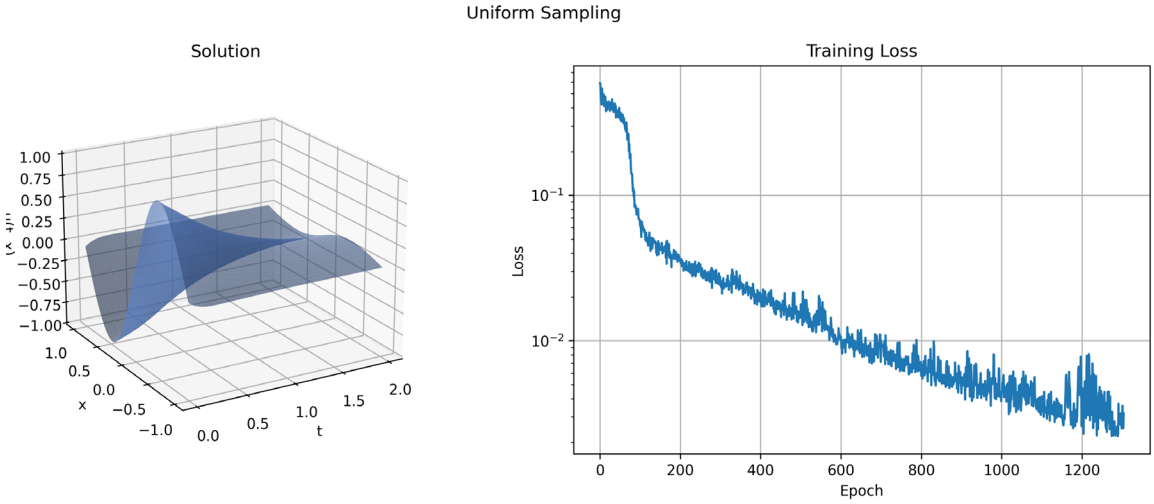

Рисунок 4.6 – Розв'язок рівняння Бюргерса (рівномірна)

Отриманий розв'язок демонструє:

– зменшення втрат з часом, що свідчить про навчання мережі;

– виконання початкової умови $u(x,0)=\sin(\pi x)$;

– виконання граничних умов Діріхле $u(t,0)=u(t,1)=0$;

– плавну зміну функції, що відповідає фізичній природі задачі.

Після цього використаємо адаптивний підхід до вибірки точок. Параметри генератора:

– розбиття області на 32×32 комірок;

– коефіцієнт згладжування $\alpha=0.99$;

– температура $T=0.1$.

Алгоритм зупинився після 1725 епох через відсутність покращення втрат протягом 100 епох і досягнув втрат на рівні 0.003589. Швидкість навчання склала приблизно 29 ітерацій на секунду.

Графік зміни втрат та розв’язок наведено на рисунку 4.7.

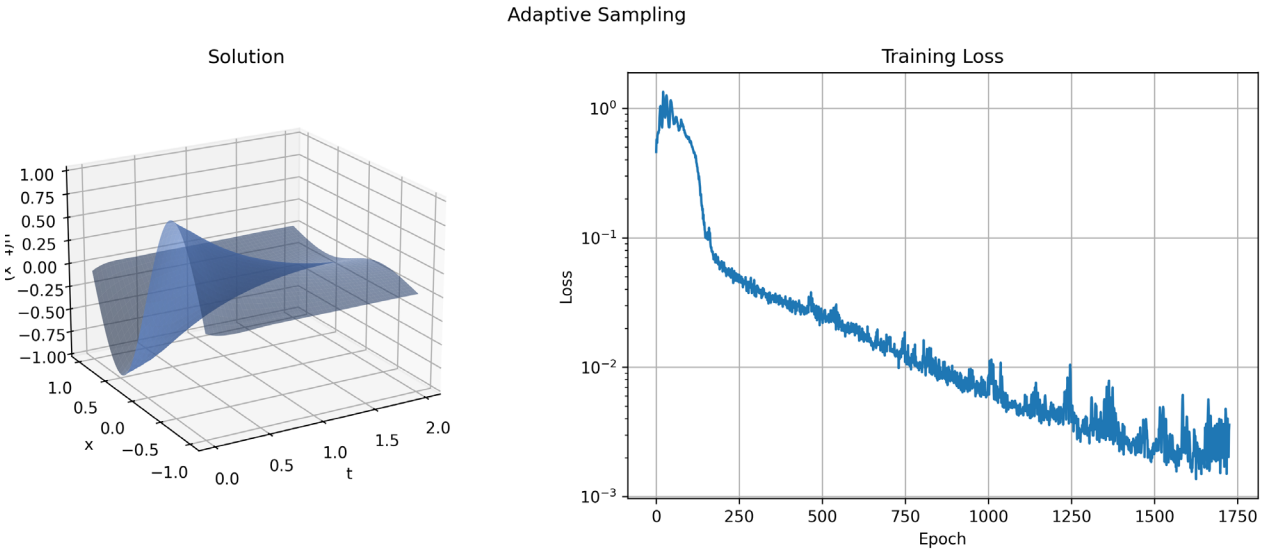

Рисунок 4.7 – Розв’язок рівняння Бюргерса (адаптивна)

Порівнявши рисунки 4.6 та 4.7, можна стверджувати, що розв’язки з рівномірною та адаптивною вибірками є досить близькими, але використання адаптивної вибірки в даному випадку не виправдало себе, оскільки збіжність до того ж результату зайняла суттєво більше часу.

4.5 Рівняння Нав’є-Стокса

Розглянемо двовимірні нестаціонарні рівняння Нав’є-Стокса для нестисливої рідини на прямокутнику $[0;2]\times[0;1]$ та часовому проміжку [0; 2]:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} \text{ (рівняння руху)}, \tag{4.9}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ (рівняння нерозривності)}$$

з початково-крайовими умовами:

$$\mathbf{u}(0,x,y) = (0,0) \text{ (початкова умова)}, \tag{4.10}$$

$$\mathbf{u}(t,x,0) = \mathbf{u}(t,x,1) = (0,0) \text{ (умова прилипання)},$$

$$\mathbf{u}(t,0,y) = (1,0) \text{ (вхідний потік)},$$

$$\frac{\partial \mathbf{u}}{\partial x}(t,2,y) = (0,0) \text{ (вихідний потік)}$$

та параметрами:

- кінематична в’язкість $\nu = 0.1$;
- густина $\rho = 1.0$.

Як і в попередніх експериментах, сприйматимемо час як просторову змінну. Для розв’язання задачі використаємо нейронну мережу з трьома вхідними (час та дві просторові координати) та трьома вихідними (дві компоненти швидкості та тиск) нейронами, 4 прихованими шарами по 64 нейрони та активаціями $\tanh$. Обрані параметри навчання:

- оптимізатор Adam з швидкістю навчання $\eta = 10^{-3}$;
- n_epochs: 5000;
- warmup_epochs: 500;
- patience: 500;

– min_delta: 10^{-3}.

Спершу використаємо рівномірну вибірку з 2000 точок всередині області та по 100 точок на кожній з її границь. Алгоритм зупинився після 1983 епох із значенням втрат 0.080153 через відсутність покращення протягом 500 епох, що свідчить про досягнення локального мінімуму. Швидкість навчання склала приблизно 19 ітерацій в секунду. Графіки поля швидкості та тиску в різні моменти часу, а також зміни втрат наведено на рисунку 4.8.

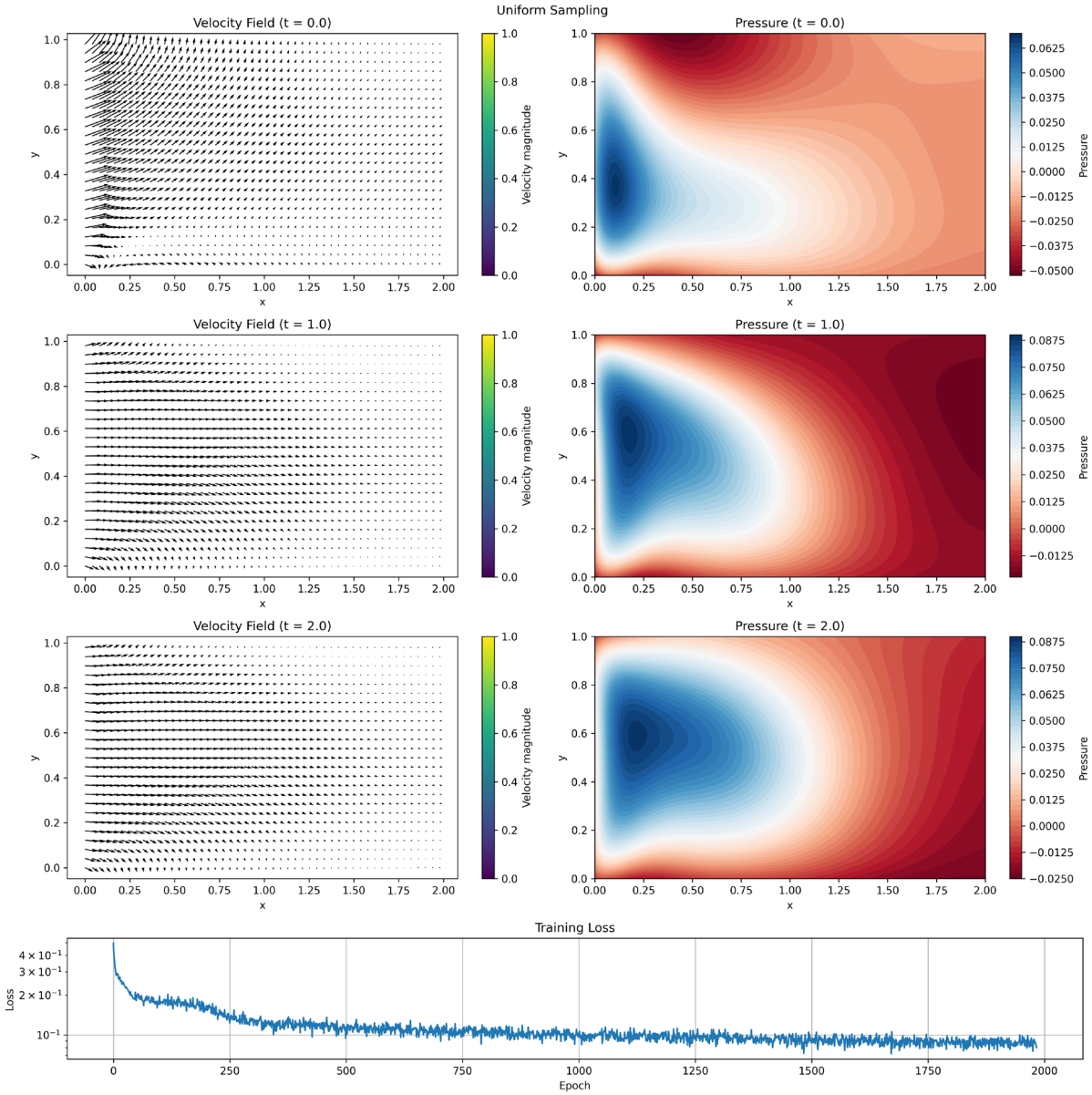

Рисунок 4.8 – Розв’язок рівнянь Нав’є-Стокса (рівномірна)

Отриманий розв’язок демонструє:

– зменшення втрат з часом, що свідчить про навчання мережі;

– виконання початкової умови $\mathbf{u}(0, x, y) = (0, 0)$;

– виконання умови прилипання на верхній та нижній границях;

– виконання умови вхідного потоку на лівій границі;

– виконання умови вихідного потоку на правій границі;

– плавну зміну полів швидкості та тиску, що відповідає природі задачі.

Після цього використаємо адаптивний підхід до вибірки точок. Параметри генератора:

– розбиття області на 64×64 комірок;

– коефіцієнт згладжування $\alpha = 0.95$;

– температура $T = 0.5$.

Алгоритм зупинився після 2466 епох через відсутність покращення втрат протягом 500 епох і досягнув втрат на рівні 0.078069.

Швидкість навчання склала приблизно 10 ітерацій на секунду.

Графіки поля швидкості та тиску в різні моменти часу, а також зміни втрат наведено на рисунку 4.9.

Порівнявши рисунки 4.8 та 4.9, ми вперше бачимо помітну розбіжність між розв’язками з рівномірною та адаптивною вибірками, особливо на графіках тиску.

В поєднанні з приблизно однаковими значеннями втрат, це може свідчити про те, що обидва рази алгоритм не збігся до прийнятного рішення.

Серед варіантів розв’язання цієї проблеми можна відзначити:

– збільшення кількості точок в вибірках;

– оптимізація гіперпараметрів алгоритму (архітектури мережі, параметрів оптимізатора, параметрів генератора точок тощо);

– використання рекурентних нейронних мереж, які краще відповідають динамічним задачам.

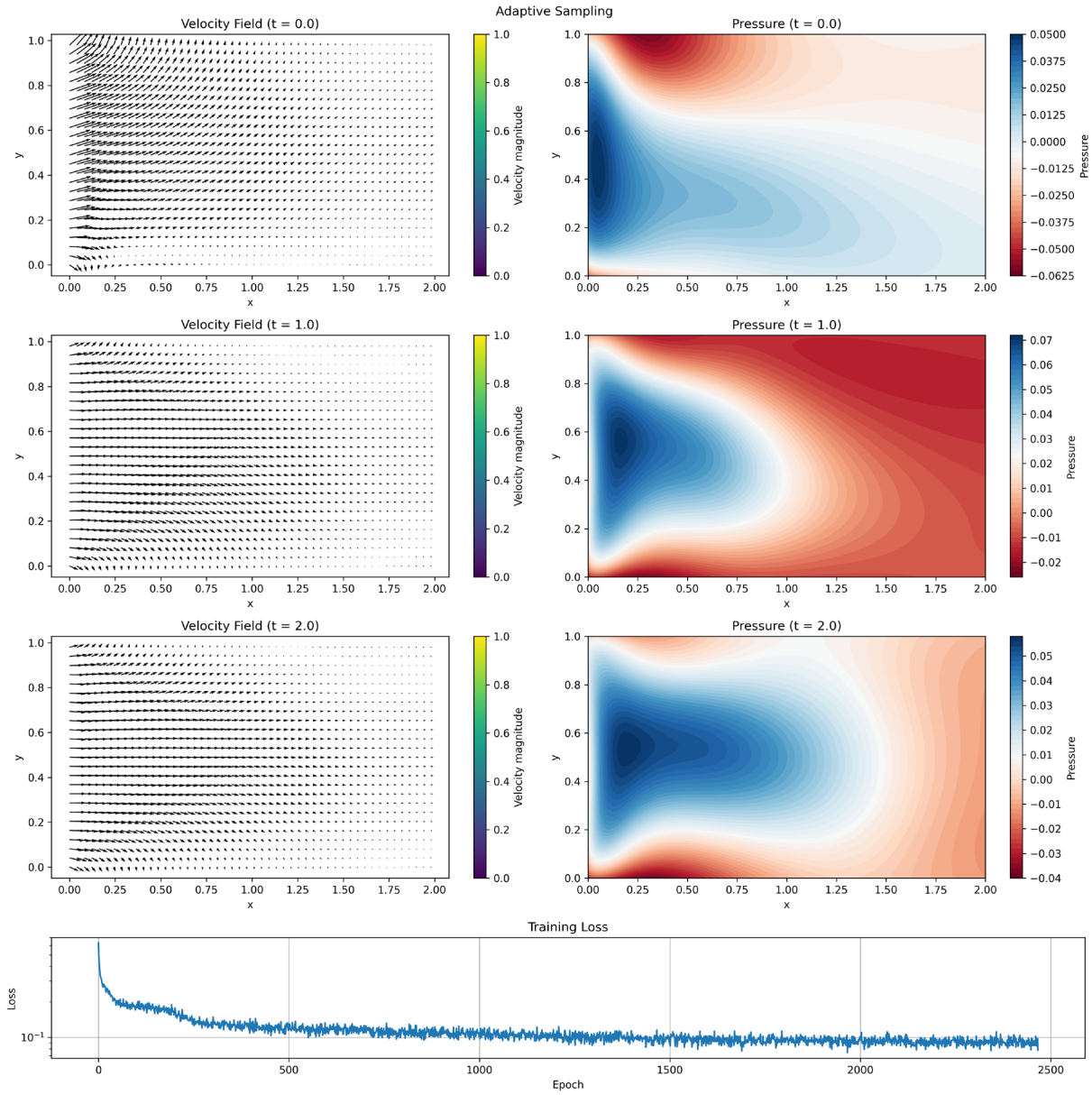

Рисунок 4.9 – Розв’язок рівнянь Нав’є-Стокса (адаптивна вибірка)

Варто зазначити, що в цьому та в усіх попередніх експериментах для відтворюваності результатів було використано початкове значення генератора випадкових чисел рівне 42.

Висновки за розділом 4

У даному розділі представлено результати обчислювальних експериментів із застосування розробленого пакету до розв’язання різних типів нелінійних

крайових задач.

Розглянуто одно- та двовимірні рівняння Пуассона, нелінійне рівняння теплопровідності, рівняння Бюргерса та рівняння Нав’є-Стокса.

Для кожної задачі проведено порівняння ефективності рівномірної та адаптивної стратегій вибору точок колокації.

Результати експериментів підтверджують працездатність запропонованого підходу та демонструють потенційні переваги адаптивної вибірки, хоча й виявляють певні обмеження методу при розв’язанні складніших задач.

ВИСНОВКИ

У кваліфікаційній роботі досліджено ефективність застосування нейронних мереж прямого поширення для чисельного дослідження та розв’язання нелінійних крайових задач для диференціальних рівнянь.

Спершу наведена загальна постановка нелінійної крайової задачі для диференціального рівняння. Далі проведено огляд та аналіз класичних методів її розв’язання, зокрема методу скінченних різниць та методу скінченних елементів, виявлено їх переваги, недоліки та складнощі, що виникають при спробі їх застосування саме до нелінійних задач.

Запропоновано підхід до розв’язання нелінійних крайових задач, що базується на використанні нейронних мереж для апроксимації невідомої функції в методі колокацій.

Параметри мережі підбираються шляхом мінімізації функції втрат, що агрегує похибку основного рівняння та граничних умов в обраних точках колокації. Для покращення точності та ефективності розв’язку запропоновано стратегію адаптивного вибору точок колокації та ранньої зупинки навчання в разі досягнення оптимізатором локального мінімуму.

Створено програмну реалізацію запропонованого підходу у вигляді Python-пакету neural-pde-solver з використанням фреймворку машинного навчання PyTorch.

Пакет включає в себе всі необхідні функції та класи для розв’язання широкого класу нелінійних крайових задач для диференціальних рівнянь а також містить реалізацію кількох класичних нелінійних диференціальних рівнянь та граничних умов.

За допомогою створеного пакету проведено низку обчислювальних експериментів, що демонструють ефективність запропонованого підходу. Розглянуто крайові задачі для одно- та двовимірних нелінійних рівнянь Пуассона, а також початково-крайові задачі для нелінійного рівняння теплопровідності, рівняння Бюргерса та рівнянь Нав’є-Стокса.

Результати експериментів підтверджують, що використання адаптивної стратегії вибору точок колокації в деяких випадках дозволяє покращити точність розв’язку порівняно з рівномірною вибіркою при тій самій кількості точок.

Отримані результати свідчать про перспективність застосування нейронних мереж для чисельного розв’язання нелінійних крайових задач для диференціальних рівнянь, однак висвітлюють і практичні складнощі, що виникають при спробі вирішення задач з багатьма змінними або високим ступенем нелінійності.

Ці складнощі в свою чергу формують потенційні напрямки для майбутніх досліджень в даній області:

– використання різних вибірок для оптимізації параметрів та рішення про ранню зупинку навчання;

– використання більш просунутих стратегій оптимізації (в тому числі гіперпараметрів мережі, алгоритму навчання та стратегії генерації точок колокації);

– застосування більш ефективних архітектур нейронних мереж, як-от LSTM, що дозволяють краще вловлювати часову залежність в системі, використовуючи менше параметрів, чи GAN, що дозволять генерувати точки колокації для яких похибка буде найбільшою;

– залучення експериментальних спостережень, що дозволить покращити ефективність навчання та розглядати обернені задачі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Леховіцький Д. О. Застосування нейронних мереж для розв’язання крайових задач для звичайних диференціальних рівнянь. *28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»* : збірник матеріалів форуму (м. Харків, 16-18 квітня 2024 р.). Т. 7. Харків : ХНУРЕ, 2024. С. 224–225.

2. LeVeque R. J. Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. Philadelphia : Society for Industrial and Applied Mathematics, 2007. 341 p. ISBN: 978-0-89871-629-0. DOI: 10.1137/1.9780898717839.

3. Zienkiewicz O. C., Taylor R. L., Zhu J. Z. The Finite Element Method: Its Basis and Fundamentals. 7th ed. Oxford : Butterworth-Heinemann, 2013. 756 p. ISBN: 978-1-85617-633-0. DOI: 10.1016/C2009-0-24909-9.

4. Сидоров М. В. Метод двобічних наближень і метод прямих розв’язання задач для одновимірного напівлінійного рівняння теплопровідності. *Дослідження в математиці і механіці*. 2018. Т. 23, № 2 (32). С. 70–85. DOI: 10.18524/2519-206x.2018.2(32).149705.

5. Двосторонні наближені методи / Б. А. Шувар та ін. Івано-Франківськ : ВДВ ЦІТ, 2007. 515 с.

6. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989. Vol. 2, No. 5. P. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

7. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 1998. Vol. 6, No. 2. P. 107–116. DOI:10.1142/S0218488598000094.

8. Visser C., Heinlein A., Giovanardi B. PACMANN: Point Adaptive Collocation Method for Artificial Neural Networks. arXiv preprint arXiv:2411.19632. 2024. 22 p.

9. Neural-PDE: A RNN based neural network for solving time dependent PDEs / Hu Y. et al. *Journal of Computational Physics*. 2022. Vol. 449. Article 110788. DOI: 10.48550/arXiv.2009.03892.

10. Van Rossum G., Drake F. L. The Python Language Reference Manual. Network Theory Ltd., 2011. 150 p.

11. PyTorch: An Imperative Style, High-Performance Deep Learning Library / Paszke A. et al. Advances in Neural Information Processing Systems 32 / ed. H. Wallach et al. Curran Associates, Inc., 2019. P. 8024–8035. DOI: 10.48550/arXiv.1912.01703.

12. Chen R. T. Q. torchdiffeq: Differentiable ODE Solvers with Full GPU Support and O(1)-Memory Backpropagation. GitHub repository. 2018. URL: https://github.com/rtqichen/torchdiffeq (дата звернення: 01.01.2025).

13. Sensio J. nangs: A Python Library for Solving Partial Differential Equations with Neural Networks. GitHub repository. 2020. URL: https://github.com/juansensio/nangs (дата звернення: 01.01.2025).