КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

ОЦІНКА ПАРАМЕТРІВ СИСТЕМ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Виконав студент 4-го курсу
Ярослав ГАВРИЛЮК

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Ярослав ЛІНДЕР

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
« 29 » травня 2023 р.,
протокол № 11
Завідувач кафедри
Олександр ПРОВОТАР

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 47 сторінок, 9 ілюстрацій, 15 джерел посилань.

АЛГОРИТМ, АНАЛІЗ, ГРАФІК, ДИНАМІКА, ДИФЕРЕНЦІАЛЬНЕ РІВНЯННЯ, ЖЕРТВА, КОЕФІЦІЄНТИ, МЕТОД РУНГЕ-КУТТИ, НЕЙРОННА МЕРЕЖА, ПАРАМЕТРИ, РІВНЯННЯ ЛОТКИ-ВОЛЬТЕРРА, ХИЖАК.

Об’єктом роботи є процес дослідження нейромережевих алгоритмів розв’язування систем диференціальних рівнянь, порівняння їх точності, ефективності, швидкодії із класичними алгоритмами розв’язування систем диференціальних рівнянь, що базуються на чисельних методах, а також розв’язання задачі оцінки параметрів диференціального рівняння Лотки-Вольтерра за допомогою нейронних мереж.

Метою роботи є порівняння класичного та нейромережевого методів розв’язання рівняння Лотки-Вольтерра, розкриття переваг та недоліків цих алгоритмів, а також розв’язання задачі оцінки параметрів диференціального рівняння Лотки-Вольтерра за допомогою нейромережевого алгоритму.

Методи дослідження та розроблення: мова програмування Python, чисельний метод Рунге-Кутти, бібліотека мови Python DeepXDE, вільно поширюване інтегроване середовище розробки PyCharm IDE, бібліотека машинного навчання TensorFlow, бібліотека мови Python Numpy, бібліотека мови Python Matplotlib.

Результати роботи: проведено аналіз ефективності застосування методів Рунге-Кутти та нейромережевого методу для розв’язання рівняння Лотки-Вольтерра, що описує модель хижак-жертва, розв’язано задачу знаходження параметрів рівняння Лотки-Вольтерра за відомими даними про кількість хижаків та жертв у певні моменти часу.

За методами розробки та інструментальними засобами робота виконувалася сумісно з роботою по оптимізації нейромережевих алгоритмів розв’язання диференціальних рівнянь.

Результати роботи можуть використовуватися у сфері екології та дослідження екосистем, зокрема дослідження змін популяцій тварин-хижаків та жертв протягом тривалого часу та прогнозування майбутніх змін в екосистемах.

Даний об’єкт дослідження потребує подальшого розвитку та подальших досліджень, оскільки безліч процесів та задач у сферах фізики, астрономії, хімії, екології, геології та інших наук містять в своїй основі диференціальні рівняння, а невпинний розвиток машинного навчання дозволить знаходити розв’язки таких задач значно легше, швидше й ефективніше ніж це відбувається сьогодні.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта дослідження. Розв'язування диференціальних рівнянь за допомогою глибинного навчання з'явилося як потенційно нова галузь наукового машинного навчання відносно недавно [1]. Вона дає можливість замінити традиційні чисельні методи нейронною мережею. У порівнянні з традиційними методами, методи глибинного навчання мають переваги, наприклад, подолання проблеми розмірeності. Крім того дані методи дозволяють уникнути помилок скорочення та числових квадратурних помилок варіаційних форм [2, 3].

Так було введено термін фізично-інформовані нейронні мережі (physics-informed neural networks). Вони дозволяють розв'язувати обернені задачі (оцінку параметрів) диференціальних рівнянь із мінімальною зміною коду прямих задач [4, 5]. Також фізично-інформовані нейронні мережі були розширені для розв'язування інтегро-диференціальних рівнянь, дробових диференціальних рівнянь та стохастичних диференціальних рівнянь [6].

Різні алгоритми цих нейронних мереж було реалізовано в бібліотеці DeepXDE мови Python. Дана бібліотека дає можливість розв'язувати мультифізичні задачі і підтримує складну геометрію, що дозволяє уникнути складних обчислень геометричних завдань. Крім того там реалізовано можливість задавати початкові та граничні умови у різних формах [7].

Актуальність роботи та підстави для її виконання. Модель «хижак - жертва», що описується рівнянням Лотки-Вольтерра (Lotka-Volterra equation) є однією із ключових моделей при дослідженні екологічної системи. Вона дає можливість відстежувати зміну чисельності популяцій тварин-хижаків та їх жертв з часом і робити висновки про вплив зовнішніх факторів на екосистему або ж приймати важливі рішення про необхідність людського втручання в цю

систему з метою врегулювання чисельності популяцій тварин і запобігання вимиранню видів [8].

Зазвичай на практиці можна досить легко отримати дані про чисельність популяцій тварин у конкретний проміжок часу, просто спостерігаючи за екосистемою. Ці дані можна використати у задачі оцінки параметрів рівняння Лотки-Вольтерра, розв'язання якої дасть змогу отримати готову математичну модель природного процесу взаємодії популяцій хижаків та жертв, дозволить прогнозувати майбутні зміни стану екосистеми та запобігати негативним для системи наслідкам.

Мета й завдання роботи. Метою кваліфікаційної роботи є аналіз нейромережевих алгоритмів розв'язання та оцінки параметрів систем диференціальних рівнянь та їх порівняння із класичними алгоритмами на основі чисельних методів, розв'язання задачі оцінки параметрів рівняння Лотки-Вольтерра за допомогою методу глибинного навчання. Для досягнення цієї мети необхідно виконати наступні завдання:

- дослідити алгоритм розв'язання рівняння Лотки-Вольтерра із використанням нейронних мереж.
- дослідити алгоритм розв'язання рівняння Лотки-Вольтерра за допомогою чисельних методів
- порівняти результати роботи вище описаних алгоритмів. Знайти переваги та недоліки кожного з методів.
- реалізувати нейромережевий алгоритм для оцінки параметрів рівняння Лотки-Вольтерра та проаналізувати його роботу.

Об'єкт, методи й засоби розроблення. Об'єктом дослідження нейромережевих алгоритмів для розв'язання систем диференціальних рівнянь є процес аналізу розв'язання рівняння Лотки-Вольтерра, що характеризує складну екосистему хижак-жертва за допомогою такого алгоритму, а також процес розв'язання задачі оцінки коефіцієнтів для даного рівняння.

Дослідження було проведено із використанням мови Python, зокрема її

бібліотеки DeepXDE – потужного засобу для наукового машинного навчання та роботи із математичними моделями задач на основі диференціальних рівнянь, що дозволяє реалізовувати алгоритми для розв’язання різних видів диференціальних рівнянь із різними видами граничних умов.

Для обробки даних та побудови нейронної мережі також використовувалися бібліотеки Numpy та TensorFlow, що також є потужними і широко використовуваними в даній області задач засобами мови Python.

Аналіз і порівняння результатів роботи здійснювалося методом їх графічного представлення та отримання числових даних і показників. Для графічного представлення було використано можливості мови Python, бібліотеки Matplotlib.

Можливі сфери застосування. Результати дослідження, проведеного під час виконання роботи, можуть бути застосовані у різних сферах наукової та господарської діяльності, зокрема при дослідженні екосистем та вивченні взаємозв’язку між популяціями тварин-хижаків та їх жертв, що базується на спостереженні за динамікою кількісті тварин в межах екосистеми протягом певного періоду часу, вивчення історії екосистеми, її теперішнього стану та прогнозування її майбутнього.

Взаємозв’язок з іншими роботами. За методами дослідження та інструментальними засобами робота виконувалась сумісно з роботами щодо оптимізації нейромережевих алгоритмів для побудови математичних моделей і розв’язання прикладних задач в галузях фізики, хімії, екології, біології.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Диференціальні рівняння

Диференціальні рівняння є одним із різновидів математичних рівнянь, що включають в себе похідні функцій. Вони використовуються для моделювання залежностей між незалежними змінними, невідомими функціями та похідними цих функцій. Причому невідома функція може бути скалярною або векторною [9].

Кожне диференціальне рівняння має порядок – найвищий порядок похідної, що використовується в даному рівнянні.

Найвищий степінь похідної найвищого порядку визначає степінь диференціального рівняння.

Досить часто при заданні диференціального рівняння задаються і його початкові або граничні умови – значення функції та її похідних в певних точках незалежної змінної. Дані умови використовуються для знаходження конкретного розв’язку диференціального рівняння. Задане диференціальне рівняння із початковими умовами утворюють задачу Коші.

За своїм типом диференціальні рівняння поділяються на звичайні та рівняння у частинних похідних. Звичайні диференціальні рівняння мають лише одну незалежну змінну, в той час як у рівняннях у частинних похідних їх є декілька.

Звичайне диференціальне рівняння в загальному вигляді можна задати формулою (1):

$$F\left(t, x, x', x'', \dots, x^{(n)}\right) = 0, \tag{1}$$

де $x = x(t)$ – шукана функція, залежна від часу t, n – порядок диференціального рівняння [9].

Рівняння у частинних похідних також має формулу загального вигляду (2):

$$F(x_1, x_2, \dots, x_m, z, \frac{dz}{dx_1}, \frac{dz}{dx_2}, \dots, \frac{dz}{dx_m}, \frac{d^2z}{dx_1^2}, \frac{d^2z}{dx_1 dx_2}, \frac{d^2z}{dx_2^2}, \dots, \frac{d^nz}{dx_m^n}) = 0, \quad (2)$$

де x_1, $x_{2, \dots,}$ x_m – незалежні змінні, z – функція від цих змінних [9].

Окрім цих двох, можна виокремити ще й інші класи диференціальних рівнянь, зокрема:

– Лінійні диференціальні рівняння. Такі рівняння складаються із функцій та їхніх похідних, що утворюють собою лінійну комбінацію. Такі рівняння можуть бути розв’язані аналітично методами розділення змінних, інтегруючого множника, методом Фур’є.
– Нелінійні диференціальні рівняння. Такі рівняння містять нелінійні функції або їх похідні. Вони є значно складнішими у розв’язанні і часто потребують використання чисельних або апроксимаційних методів, які будуть розглянуті пізніше.
– Системи диференціальних рівнянь. Вони складаються із кількох диференціальних рівнянь, що залежать від кількох змінних і функцій. Вони показують залежність між цими змінними у системі.

Диференціальні рівняння знайшли своє широке застосування для опису і створення математичних моделей динамічних процесів у фізиці, хімії, біології, екології, механіці та багатьох інших науках. Наприклад, модель «хижак - жертва» можна описати через рівняння Лотки-Вольтерра, яке ми розглянемо далі. За допомогою рівняння Пуассона (Poisson equation) можна описати поведінку електростатичного поля, поля температури, тиску, потенціалу швидкості в області гідродинаміки. Рівняння Гельмгольца (Helmholtz equation) дозволяє розрахувати ізохорний тепловий ефект хімічної реакції. Течії рідин та газів можна змоделювати через рівняння Нав’є-Стокса (Navier-Stokes equation). Існують також окремі рівняння для опису таких

фізичних процесів як нагрівання та дифузія, а рівняння Шредінгера (Schrodinger equation) стало відоме, як основне рівняння руху у нерелятивістській квантовій механіці, воно дає можливість описати еволюцію квантової системи за певний проміжок часу.

Отже, враховуючи різноманіття сфер застосування диференціальних рівнянь, можна зробити висновок про їх величезний вклад у сучасний рівень наукового розвитку світу, а тому очевидною є необхідність поглиблено досліджувати дану область математики, виводити нові види диференціальних рівнянь для опису важливих природних і технічних процесів, а також знаходити якомога більш оптимальні способи розв'язання таких рівнянь.

1.2 Рівняння Лотки-Вольтерра

Рівняння Лотки-Вольтерра, також відоме як модель «хижак - жертва», є системою диференціальних рівнянь, що складається із двох нелінійних диференціальних рівнянь першого порядку. Воно часто застосовується при моделюванні взаємодії двох видів тварин всередині екологічної системи, де один із видів є хижаком, а інший – жертвою [8].

Назване дане рівняння на честь своїх авторів: американського математика та статиста Альфреда Джеймса Лотки (Alfred James Lotka) та італійського математика і фізика Віто Вольтерри (Vito Volterra), що запропонували дану модель у 1925 - 1926 роках, оприлюднивши свої відкриття незалежно один від одного.

Дане рівняння має вигляд:

$$\frac{dx}{dt} = (\alpha - \beta y)x$$
$$\frac{dy}{dt} = (-\gamma + \delta x)y, \qquad (3)$$

де:

– x – популяція жертв (наприклад кількість особин зайця на квадратний

кілометр)

– y – популяція хижаків (наприклад кількість особин вовка на квадратний кілометр)
– t – час
– α - максимальна швидкість росту жертви на душу населення
– β - вплив присутності хижаків на швидкість росту жертви
– γ – рівень смертності хижака на душу населення
– δ – вплив присутності жертви на швидкість росту хижака

Перше рівняння цієї системи (3) характеризує зміну популяції жертви залежно від її самостійного розмноження, а також від взаємодії з хижаком. Друге рівняння системи (3) описує зміну популяції хижака залежно від його самостійного вимирання і доступності їжі, тобто взаємодії з жертвою [10]

Рівняння Лотки-Вольтерра дають можливість вивчати динаміку популяцій хижака та жертви залежно від початкових умов (x, y) і параметрів системи $(\alpha, \beta, \gamma, \delta)$. В них можна спостерігати циклічні коливання (рис. 1), стійкі точки рівноваги й інші типи поведінки залежно від параметрів.

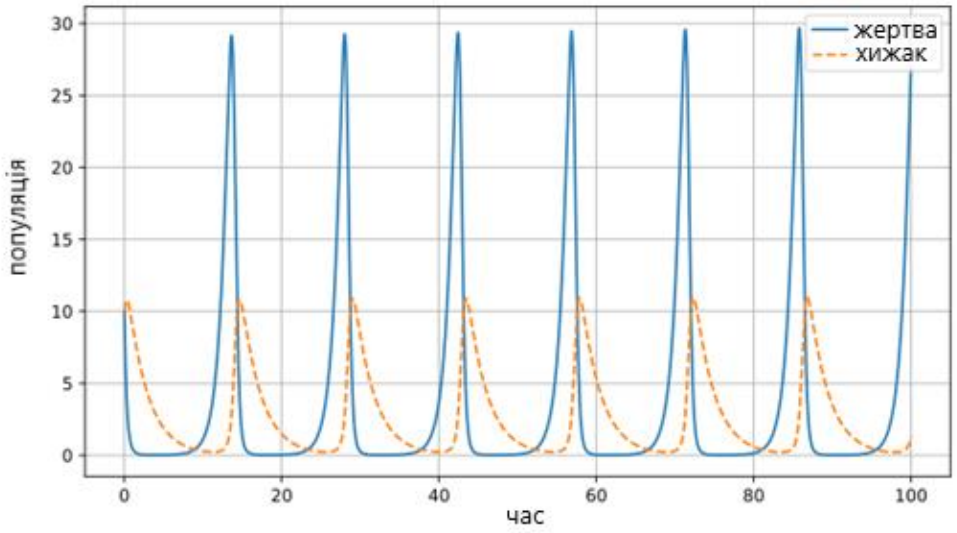


Рисунок 1 – Приклад графіка рівняння Лотки-Вольтерра

Уважно простеживши за циклічними коливаннями графіка функції, які

добре спостерігаються на рис. 1 можна визначити, що стрімкий ріст популяції хижака помічається одразу після стрімкого росту популяції жертви. Це пояснюється тим, що наявність достатньої кількості їжі створює сприятливі умови для існування хижака. В той же час коли популяція хижака стає надто великою і запас їжі починає зменшуватись, то настають несприятливі умови для хижака, вони починають вимирати і їх популяція стрімко падає, що в свою чергу знову створює передумови для настання сприятливих умов для жертв і стрімкого росту їх популяції. Таку природну циклічність легко можна побачити на фазово-просторовому графіку рівняння Лотки-Вольтерра (рис. 2), який теж досить часто можна зустріти в науковій літературі, присвяченій дослідженні екосистем і зокрема моделі «хижак - жертва».

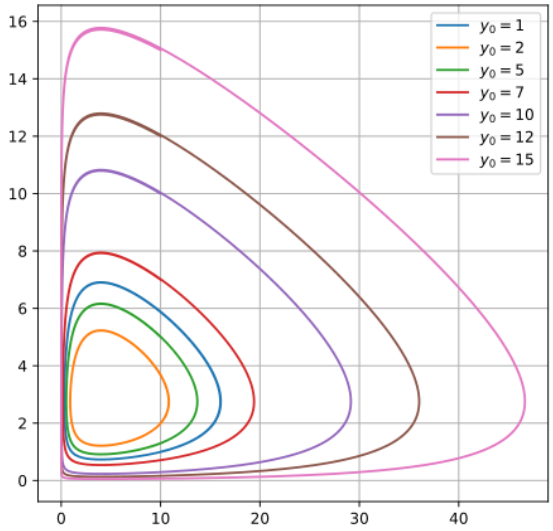


Рисунок 2 – Фазово-просторовий графік рівняння Лотки-Вольтерра за різних початкових умов

Для розв’язування рівнянь Лотки-Вольтерра можна користуватись аналітичними методами, зокрема розділенням змінних або методом Лапласа. А також можливе використання чисельних методів, як, наприклад, метод Рунге-Кутти.

Отже, рівняння Лотки-Вольтерра знайшло своє широке застосування в

галузі екології та біології і дає змогу краще досліджувати складні природні процеси, що відбуваються в екосистемах з плином часу.

1.3 Чисельні методи розв’язання диференціальних рівнянь

Чисельні методи розв’язання диференціальних рівнянь зазвичай використовуються для знаходження наближених чисельних розв’язків диференціальних рівнянь у тих випадках коли неможливо або важко знайти розв’язок цих рівнянь за допомогою аналітичних методів. Найчастіше це стосується нелінійних диференціальних рівнянь. Чисельні методи розв’язання диференціальних рівнянь базуються на апроксимації похідних та інтегруванні рівнянь за допомогою обчислювальних алгоритмів [11].

Розглянемо основні чисельні методи, що використовуються для розв’язання диференціальних рівнянь:

- Метод Ейлера (Euler method). Це досить простий і популярний метод розв’язування звичайних диференціальних рівнянь, що базується на процесі апроксимації похідних, використовуючи диференційні коефіцієнти та ітераційні формули. Він є основою для розвитку більш складних методів.
- Метод Рунге-Кутти (Runge-Kutta method). Цей метод є складнішим і точнішим у порівнянні з методом Ейлера, оскільки має кілька кроків і використовує обчислений набір проміжних значень для отримання кінцевого результату.
- Метод пристрілки (Shooting method). Це метод розв’язання крайових задач шляхом зведення їх до задач початкових умов. Далі таку задачу можна звести до звичайного алгебраїчного рівняння, яке уже відносно легко розв’язати.
- Метод скінченних різниць (Finite difference method). Цей метод

ще називають методом сіток, оскільки він часто використовується для розв’язання задач на нерегулярних сітках або в областях комплексної геометрії. Він використовує апроксимацію похідних через скінченні різниці, що врешті-решт дозволяє звести диференціальне рівняння спочатку до системи нелінійних, а згодом і лінійних алгебраїчних рівнянь.

– Метод скінченних елементів (Finite elements method). Цей метод використовує розбиття досліджуваної області на скінченні елементи, далі відбувається апроксимація функцій. Метод дозволяє звести диференціальні рівняння у частинних похідних до системи звичайних диференціальних рівнянь, які потім можуть бути розв’язані за допомогою простіших методів Ейлера чи Рунге-Кутти.
– Метод скінченних різниць у часі (Finite Difference Time Domain method). Він використовується для розв’язання диференціальних рівнянь у частинних похідних, враховуючи часову залежність, зокрема рівнянь Максвела (Maxwell’s equations), записаних у диференціальній формі. Він апроксимує похідні у часі та просторовій змінній. Це дозволяє розв’язувати рівняння на послідовних часових кроках.
– Метод кінцевих різниць у часі (Finite differences in time method). Даний метод є варіацією методу скінченних різниць у часі, але використовує обчислювальну сітку, що складається з крайових точок в просторі і часі.

Отже, різні чисельні методи дають можливість розв’язувати як звичайні диференціальні рівняння, так і диференціальні рівняння у частинних похідних. Вибір методу може залежати від типу рівняння, граничних умов, обчислювальних обмежень та складності задачі.

При виборі чисельного методу слід обов’язково враховувати його швидкодію та точність. Варто також постійно слідкувати за часовою і

просторовою областю, а також за чисельними похибками під час виконання методу.

Розглянемо деякі з основних чисельних методів розв'язання диференціальних рівнянь більш детально.

1.3.1 Метод Ейлера

Метод Ейлера – найбільш базовий метод для розв'язання звичайних диференціальних рівнянь із заданими початковими умовами. Він базується на використанні диференційного коефіцієнта та ітераційних формул для апроксимації похідної. Для диференціальних рівнянь першого порядку (4) алгоритм методу включає наступні кроки:

$$x'(t) = f\big(t, x(t)\big), \;\; x(t_0) = x_0 \tag{4}$$

а) вибрати часовий крок h для розбиття часового проміжку

б) визначити початкові умови t_0 та x_0

в) використати ітераційну формулу (5) для знаходження наступних значень розв'язку x_{n+1} на наступному часовому кроці:

$$x_{n+1} = x_n + hf(t_n, x_n), \tag{5}$$

де $t_n = t_0 + nh$, x_n – попереднє значення розв'язку на часовому кроці n, а функція $f(t,x)$ – функція правої частини рівняння

г) повторити крок **в** для бажаної кількості часових кроків або досягнення потрібного значення t

Перевага методу Ейлера полягає в простоті його реалізації, але він має і недоліки, а саме низьку точність для великих кроків h та складних задач.

1.3.2 Метод Рунге-Кутти

Метод Рунге-Кутти також є методом для розв'язання звичайних диференціальних рівнянь, але на відміну від методу Ейлера він уже

використовує додаткові проміжні значення для апроксимації розв’язку та похідної.

Названий даний метод на честь своїх відкривачів, німецьких математиків Карла Рунге (Carl David Tolme Runge) та Мартіна Кутти (Martin Wilhelm Kutta).

Основна ідея методу Рунге-Кутти полягає в обчисленні кількох проміжних значень і використання їх для оновлення значення розв’язку на кожному кроці. Кількість проміжних значень визначає порядок методу Рунге-Кутти. Найбільшого поширення набув метод Рунге-Кутти четвертого порядку, який ще називають класичним методом Рунге-Кутти.

Ідея методу Рунге-Кутти четвертого порядку полягає в обчисленні чотирьох проміжних значень для оновлення розв’язку, а ітераційна формула (6) для розв’язання рівняння, заданого формулою (4) має наступний вигляд:

$$
\begin{aligned}
k_1 &= h * f(t_n, x_n), \\
k_2 &= h * f\left(t_n + \frac{h}{2}, x_n + \frac{k_1}{2}\right), \\
k_3 &= h * f\left(t_n + \frac{h}{2}, x_n + \frac{k_2}{2}\right), \\
k_4 &= h * f(t_n + h, x_n + k_3), \\
t_{n+1} &= t_n + h, \\
x_{n+1} &= x_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6},
\end{aligned}
\tag{6}
$$

де $t_n = t_0 + nh$, x_n – попереднє значення розв’язку на часовому кроці n, а функція $f(t,x)$ – функція правої частини рівняння [11].

Перевагами методу Рунге-Кутти є гнучкість використання для широкого спектра диференціальних рівнянь, простота реалізації та вища точність ніж у простіших методів, зокрема методу Ейлера. Метод Рунге-Кутти добре підходить і для розв’язання систем диференціальних рівнянь.

1.3.3 Метод пристрілки

Метод пристрілки застосовується для розв’язання крайових задач, зокрема диференціальних рівнянь другого порядку з крайовими умовами. Він дає змогу перетворити крайову задачу на задачу знаходження коренів нелінійного рівняння. Основна ідея полягає в пошуку початкових значень при яких задовольняються крайові умови [12].

Основними кроками методу пристрілки є:

а) формування крайової задачі, що включає диференціальне рівняння та крайові умови.

б) задання початкового наближення для початкових умов, наприклад значення початкової точки та значення похідної.

в) розв’язування диференціального рівняння за допомогою чисельного методу

г) перевірка, на відповідність отриманих значень крайовим умовам.

д) корекція початкових умов для досягнення крайових умов. Цей крок включає процес знаходження кореня нелінійного рівняння.

е) повторення кроків **в – д** поки не отримано результат, що задовольняє крайові умови з потрібною точністю.

Метод пристрілки є ефективним для розв’язання крайових задач у складних фізичних, економічних моделях, при розв’язанні систем диференціальних рівнянь.

Основними недоліками методу пристрілки є велика кількість ітерацій або неефективність на задачах, які мають багато розв’язків або у яких важко знайти початкові умови. Метод також може бути неточним або погано збігатися при розв’язанні жорстких крайових задач, де похідні швидко змінюються.

1.4 Штучні нейронні мережі

1.4.1 Загальний огляд

Штучні нейронні мережі (Artificial Neural Networks) – це комп'ютерні обчислювальні системи, що намагаються симулювати роботу людського мозку. Основною їх властивістю є здатність до навчання, тобто вони тренуються на заданих наборах даних при цьому постійно покращуючи свою продуктивність, а не програмуються під конкретну задачу. Такі мережі успішно розв'язують задачі розпізнавання зображень, обробки природної мови, прогнозування, надання рекомендацій, розв'язання складних математичних задач та багато інших [13, 14].

Наприклад у задачі визначення наявності собак на картинці, нейронна мережа, не маючи ніяких власних знань про собак, вчиться визначати для себе доречні характеристики із набору картинок, які передаються їй для навчання та набору правильних міток до них. Потім ці отримані в процесі навчання характеристики вона застосовує до розпізнавання вже тестових, нових зображень і може працювати успішно.

Штучна нейронна мережа складається зі штучних нейронів, що наслідують біологічні нейрони у мозку людей чи тварин. Кожен штучний нейрон може приймати певну інформацію, обробляти її і передавати далі по мережі [14].

Зазвичай сигналом, що передається по з'єднанню нейронів є дійсне число, а результатом роботи нейрона є сума його входів, до якої застосована нелінійна функція активації. Штучні нейрони і з'єднання мають ваги, які налаштовуються в процесі навчання і визначають силу сигналу, що передається через нейронні шари від вхідного і до вихідного.

Початком історії розвитку штучних нейронних мереж вважають 1943 рік, коли двоє американських учених Ворен МакКаллок (Warren McCulloch) та Волтер Піттс (Walter Pitts) створили математичну модель штучного

нейрону, що опрацьовував вхідні дані і генерував вихідний сигнал. Після цього дана галузь почала швидко розвиватися, було винайдено перцептрон (perceptron) – найпростішу модель нейронної мережі (рис. 3), що дозволило розв'язувати прості задачі прогнозування, розпізнавання образів.

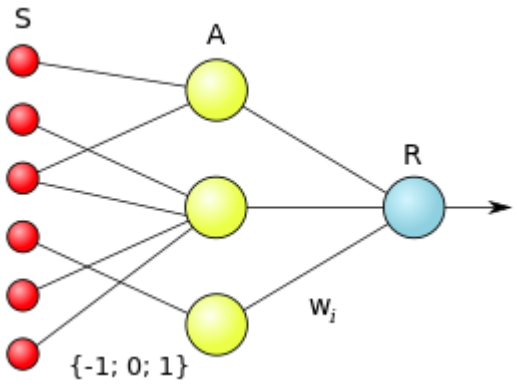


Рисунок 3 – Перцептрон

Пізніше почали з'являтися й інші, більш досконалі типи нейронних мереж, які будуть розглянуті в наступних підрозділах. Це дозволяло розв'язувати все більш і більш складні задачі за допомогою методів глибинного навчання. Стрімкий розвиток даної галузі ми спостерігаємо і сьогодні, машинне навчання – один із найперспективніших напрямків досліджень в сфері інформаційних технологій.

1.4.2 Архітектура нейронних мереж

Штучні нейрони об'єднуються у мережі, утворюючи системи обробки інформації, що адаптують модель до постійних змін зовнішнього середовища. На основі архітектури штучні нейронні мережі поділяють на повнозв'язні (fully connected), де входи нейрона зв'язані з виходами усіх інших нейронів, та слабозв'язні (weakly connected), де нейрон має зв'язки лише зі своїми сусідами.

На даний момент найбільш використовуваними є багатошарові

нейронні мережі, де можна виділити вхідний шар, який приймає вектор вхідних значень, приховані шари, які передають вектори сигналів від попереднього до наступного, та вихідний шар, який повертає кінцеве значення, обраховане нейронною мережею у зовнішнє середовище.

За архітектурою зв'язків нейронні мережі також поділяють на мережі прямого поширення (forward propagation) та зворотнього поширення (back propagation). Мережі прямого поширення не залежать від попереднього стану, а тому їх ще називають статичними. Мережі зворотнього поширення називають рекурентними, оскільки вони мають зворотні зв'язки і змінюють стани мережі [14].

Налаштування архітектури є одним із ключових моментів процесу розв'язання задачі із нейронною мережею, оскільки воно суттєво впливає на швидкість навчання та точність результатів.

1.4.3 Навчання нейронних мереж

Наступним етапом є навчання нейронної мережі, тобто процедура налаштування її ваг для отримання необхідних результатів. Навчатися нейронна мережа може трьома способами:

- З вчителем. При такому способі навчання на вхід нейронної мережі задається тренувальний набір даних із готовими мітками-відповідями. Ваги налаштовуються так, щоб мінімізувати похибку – різницю між виходом мережі та міткою.
- Без вчителя. Такий тип навчання не потребує наявності готових міток-відповідей. Нейронна мережа отримує на вхід лише вектори-зразки і намагається згрупувати їх у класи за подібними ознаками, тобто налаштовує ваги таким чином, щоб для подібних вхідних векторів видавати подібні результати.
- Навчання з підкріпленням. Такий вид навчання передбачає налаштування ваг нейронної мережі шляхом її взаємодії з певним

середовищем. Середовище посилає моделі позитивні оцінки у випадку правильних дій та штрафи у випадку неправильних. Таким чином перед нейронною мережею постає задача максимізації кінцевої оцінки.

Процес навчання нейронної мережі є ітеративним і включає в себе певну кількість епох – кроків, на яких нейронній мережі пред’являються всі зразки із навчальної вибірки та їх мітки. Вкінці кожної епохи за необхідності додають процес валідації – перевірки якості навчання нейронної мережі на спеціально виокремленій контрольній множині.

Під час навчання нейронна мережа оцінює різницю між своїм вихідним значенням та реальною відповіддю і таким чином утворюється функція похибки – цільова функція, яка показує неточність результатів, отриманих моделлю. Завдання нейронної мережі в процесі навчання полягає в мінімізації цієї функції. Врешті-решт вагові коефіцієнти стабілізуються і тоді можна говорити, що модель є натренованою і готовою до тестування та роботи [13].

На якість навчання моделі дуже впливає архітектура нейронної мережі, а саме її відповідність поставленій задачі. Також важливими гіперпараметрами є кількість епох навчання, розмір мінівибірки. Дуже важливою є проблема розміру датасету – кількості прикладів для навчання, адже при зростанні складності задачі, кількість необхідних даних дуже швидко зростає. Цю проблему називають ще «прокляттям розмірності».

Досить часто при тренуванні нейронної мережі виникає таке неприємне явище, як перенавчання (overfitting). При неправильно підібраній структурі нейронної мережі, занадто великій кількості епох навчання або ж малому розмірі навчальної вибірки вагові коефіцієнти починають занадто сильно підганятись під певний тренувальний набір даних. Точність на тренувальному наборі може бути дуже високою, але вже при роботі на нових даних виникне серйозна проблема: результати роботи можуть виявитися досить невтішними. Для подолання даної проблеми рекомендується

заздалегідь розбивати датасет на тренувальні та тестові дані, а також дані для валідації з метою контролю процесу навчання на кожній ітерації. Також важливо розумно підходити до вибору архітектури нейромережі та вибору гіперпараметрів для навчання. За потреби можна застосовувати спеціальні методи регуляризації, спрямовані на запбігання перенавчанню, зокрема метод ранньої зупинки (early stopping).

1.4.4 Огляд основних типів нейронних мереж

На сьогоднішній день існує достатньо багато типів нейронних мереж, кожен з яких призначений для розв’язання певних типів задач. Коротко розглянемо основні типи нейромереж та задачі, які вони розв’язують:

- Перцептрон. Це найпростіший тип нейронної мережі, що може використовуватись, наприклад, для простих задач регресії або бінарної класифікації.
- Згорткові нейронні мережі (Convolutional Neural Networks, CNN). Даний тип нейронної мережі широко використовується в роботі зі зображеннями. Застосування операцій згортки і пулінгу дає можливість добре розпізнавати локальні особливості зображень і робити процес їх розпізнавання значно ефективнішим.
- Рекурентні нейронні мережі (Recurrent Neural Networks, RNN). Даний тип нейронної мережі має зворотні зв’язки, що є ефективним при аналізі мовних виразів чи часових рядів.
- Довга короткочасна пам’ять (Long Short-Term Memory, LSTM). Різновид рекурентних нейронних мереж, ефективний для прогнозування часових подій при розділених важливих подіях.
- Глибинні нейронні мережі (Deep Neural Networks, DNN). Багатошарові нейронні мережі, вони дозволяють розв’язувати складні завдання розпізнавання образів, голосового синтезу, обробки природньої мови.
- Автокодувальник (Autoencoder). Застосовується для зменшення

розмірності даних та відтворення оригінальних даних по зменшеному представленню. Ефективний у задачах розпізнавання облич, визначення семантичного значення слів.

– Рекурентні згорткові нейронні мережі (Recurrent Convolutional Neural Networks, RCNN). Це комбінація рекурентних та згорткових нейронних мереж, використовуються для просторово-часових даних, наприклад, для обробки відео.

1.4.5 Переваги та недоліки нейронних мереж. Методи покращення їх роботи

Врешті-решт коротко розглянемо основні переваги та недоліки нейронних мереж. Основними перевагами нейронних мереж є:

– Здатність до самонавчання. Нейронні мережі можуть самі визначати для себе корисні ознаки в зразках у процесі роботи і поступово покращувати свої результати.
– Можливість роботи з великими об’ємами даних.
– Гнучкість та адаптивність до різних типів даних.
– Здатність помічати складні й неочевидні залежності між даними.
– Можливість використання паралельних обчислень для швидшої роботи та навчання.

До недоліків нейронних мереж можна віднести:

– Потреба у великій кількості даних для навчання, які досить часто буває важко знайти у вільному доступі.
– Для роботи з великими об’ємами даних потрібні високі обчислювальні ресурси та потужне обладнання, які є досить дороговартісними.
– Нейронні мережі можуть бути складними для розуміння, так як буває досить важко відслідковувати внутрішні процеси під час їх навчання.
– Нейронні мережі можуть проявляти надто високу чутливість до даних і реагувати на їх найменші зміни.

Методи покращення роботи нейронних мереж:

– Збільшення кількості та якості даних для навчання. Від правильно складеного датасету дуже залежить успішність розв’язання поставленої задачі.
– Використання алгоритмів оптимізації для налаштування параметрів нейронної мережі.
– Використання більш потужного обладнання для пришвидшення роботи алгоритму.
– Відповідальне ставлення до вибору архітектури нейронної мережі, яка має відповідати поставленій задачі. Правильний підбір гіперпараметрів під час навчання.
– Використання технік регуляризації з метою покращення загальної роботи нейронної мережі та уникнення перенавчання.

1.5 Бібліотека DeepXDE

1.5.1 Загальний опис бібліотеки

Мова програмування Python широко використовується для програмування нейромережевих алгоритмів і має багато бібліотек, призначених для роботи із нейронними мережами та глибинним навчанням. Однією із них є бібліотека DeepXDE, яка і буде розглянута в даному підрозділі.

DeepXDE – бібліотека, що призначена для наукового машинного навчання, навчання в області фізики та розв’язування різноманітних задач, пов’язаних із диференціальними рівняннями [7, 15]

У порівнянні із класичними алгоритмами на основі чисельних методів, код, написаний за допомогою бібліотеки DeepXDE, може бути значно коротшим і повнішим, а також більш близьким до математичного

формулювання задачі.

Процес розв’язання диференціального рівняння за допомогою DeepXDE є дуже автоматизованим і не вимагає від програміста-користувача заглиблення в реалізацію окремих компонентів алгоритму. Роль користувача в цьому процесі зводиться до постановки задачі, тобто задання самого диференціального рівняння, обчислювальної області, початкових або граничних умов, обмежень. Також програміст-користувач задає дані для навчання, архітектуру нейронної мережі та гіперпараметри для навчання. Всі інші процеси розв’язання задачі відбуваються уже всередині готових модулів, що значно полегшує процес розробки програм через DeepXDE для програміста [7]

1.5.2 Особливості і переваги DeepXDE

Бібліотека DeepXDE є потужним засобом в галузі наукового машинного навчання і має ряд переваг та особливостей, а саме:

- в бібліотеці реалізовані алгоритми для фізично-інформованих нейронних мереж (physics-informed neural networks, PINN) та для підвищення їх точності, для глибоких операторських мереж (deep operator networks, DeepONet), багатофункціональних нейронних мереж (multifidelity neural networks, MFNN).
- підтримка п’яти популярних тензорних бібліотек для роботи з нейронними мережами
- можливість розв’язання обернених задач диференціальних рівнянь (задач оцінки параметрів) без необхідності робити великі зміни в коді програми.
- різні типи геометрій обчислювальної області
- п’ять модулів для задання граничних умов
- велика різноманітність типів нейронних мереж
- багато методів вибірки для задання навчальних точок

- чотири простори функцій
- можливість паралельного навчання даних
- великий вибір оптимізаторів нейронної мережі
- можливість зручного зберігання та завантаження натренованої моделі
- наявність зворотних викликів для моніторингу процесу навчання
- підтримка різних типів числових даних
- можливість побудови графіків результатів, функцій втрат, метрик

Бібліотека DeepXDE легко і зрозуміло встановлюється на комп'ютер та добре структурована завдяки слабким зв'язкам між компонентами [15].

1.5.3 Алгоритм розв'язання диференціальних рівнянь за допомогою DeepXDE

Основні кроки алгоритму розв'язання диференціальних рівнянь за допомогою бібліотеки DeepXDE:

- Задання обчислювальної області за допомогою модуля geometry.
- Задання диференціального рівняння, використовуючи граматику TensorFlow або інших тензорних бібліотек, які підтримує DeepXDE.
- Задання початкових або граничних умов рівняння.
- Об'єднання геометрії, диференціального рівняння та початкових умов за допомогою модуля PDE для незалежних від часу проблем, або ж TimePDE для часово-залежних проблем. На цьому кроці також задається навчальний набір через вказання шляху до файлу із даними, або ж просто заданням кількості точок для тренування.
- Побудова нейронної мережі.
- Об'єднання поставленої задачі та нейронної мережі у модель через модуль Model.
- За допомогою методу Model.compile() задаються гіперпараметри для навчання моделі, такі як швидкість навчання (learning rate) та оптимізатор.

– Навчання моделі за допомогою методу Model.train(). На даному етапі є можливість задати кількість епох для навчання, а також параметр callbacks, за допомогою якого можна задавати певну поведінку моделі під час навчання.
– Тестування, виведення результатів моделі за допомогою методу Model.predict().

Розглянемо детальніше деякі із кроків, їх реалізацію та можливості, які надає DeepXDE.

DeepXDE надає сім видів примітивних геометрій для визначення області обчислень: інтервал, трикутник, прямокутник, полігон, диск, куб та сфера. За необхідності побудови складніших геометрій користувач може створювати їх самостійно, використовуючи примітивні геометрії та операції об’єднання, різниці та перетину.

Саме диференціальне рівняння задається математичною формулою, використовуючи змінні та знаки арифметичних операцій. Диференціали першого порядку задаються через функцію deepxde.grad.jacobian(), диференціали другого порядку – deepxde.grad.hessian(). Диференціали вищих порядків можна задавати комбінуючи дані функції.

У DeepXDE реалізовано чотири головних модулі для задання граничних умов, а саме граничних умов Діріхле (DirichletBC), Неймана (NeumannBC), Робіна (RobinBC) та періодичних граничних умов (PeriodicBC). Більш загальні граничні умови можуть задаватися через модуль OperatorBC. Модуль IC використовується для задання початкових умов.

Для об’єднання рівняння, геометрії та граничних умов використовуються модулі PDE або TimePDE, враховуючи залежність або незалежність задачі від часу. В цих модулях надається можливість задати тренувальний набір даних, або кількість точок всередині обчислюваної області та на її кінцях, додаткові точки для тренування, кількість точок для тестування, точки для виключення. Також сюди можна задати аналітичний

розв’язок рівняння, якщо вдалося його знайти.

DeepXDE підтримує архітектуру різних типів нейронних мереж, які підтримуються тензорними бібліотеками, доступними для використання в DeepXDE: TensorFlow 1.x, TensorFlow 2.x, PyTorch, JAX та PaddlePaddle. Користувач може обрати повнозв’язні, паралельні повнозв’язні, багатофункціональні, багатовхідні нейронні мережі, багатомасштабні мережі Фур’є, залишкові нейронні мережі залежно від типу задачі, яку він хоче розв’язати. Також користувачем задається архітектура нейронної мережі, тобто кількість шарів та нейронів у них, функція активації, ініціалізатор, регуляризація, нормалізація вибірки та інші параметри.

Під час навчання нейронної мережі є можливість використовувати зворотні виклики (callbacks), які дозволяють контролювати процес тренування і вносити в нього зміни в реальному часі. Наприклад, за допомогою callbacks можна налаштовувати нейронну мережу на пошук невідомих параметрів у задачі оцінки параметрів диференціального рівняння або ж змінювати швидкість навчання безпосередньо під час процесу [7].

2 ПОСТАНОВКА ЗАДАЧІ

У даному розділі буде розглянуто завдання, яке необхідно виконати для досягнення поставлених результатів роботи, його умови, початкові дані та засоби, за допомогою яких буде здійснюватися процес розв’язання цієї задачі.

2.1 Умови завдання

Перед виконанням роботи було поставлено завдання комплексного аналізу ефективності нейромережевих алгоритмів для розв’язання диференціальних рівнянь, їх порівняння із класичними алгоритмами на основі чисельних методів та розв’язання задачі оцінки параметрів диференціального рівняння за допомогою нейронних мереж.

Для виконання даного завдання і досягнення усіх поставлених цілей взято рівняння Лотки-Вольтерра із заданими початковими умовами як об’єкт для дослідження. Спочатку, для кращого ознайомлення з досліджуваною областю, ми розв’яжемо дане рівняння за допомогою методу Рунге-Кутти, а також використовуючи нейронні мережі. Після цього порівняємо швидкодію, простоту реалізації та результати роботи даних алгоритмів. Врешті-решт, використавши набір точок розв’язку, згенерований методом Рунге-Кутти, оцінимо невідомі параметри рівняння Лотки-Вольтерра, оцінимо результати роботи алгоритму, порівнявши згенеровані нейронною мережею значення параметрів рівняння із реальними, які були задані нами спочатку. Для детальнішого аналізу розглянемо роботу алгоритму оцінки параметрів для різних кількостей вхідних точок та різних проміжків часу.

2.2 Початкові дані

Для розв’язання задачі оцінки параметрів рівняння Лотки-Вольтерра нам необхідно зафіксувати певну кількість моментів часу із досліджуваного проміжку, достатню для розв’язання задачі із очікуваною точністю, а також знати значення чисельності популяцій тварин-хижаків та тварин-жертв у дані моменти часу.

На практиці, при реальних дослідженнях екосистем, дослідник-еколог має визначити часовий проміжок, впродовж якого спостерігатиме за тваринами і робити заміри чисельності їхніх популяцій з певною періодичністю та фіксувати ці дані, кожного разу зазначивши дату проведення обрахунку, фактичну чисельність хижаків та фактичну чисельність жертв. При завершенні досліджуваного періоду, дані, зібрані екологом, задаються нейронній мережі для оцінки параметрів рівняння Лотки-Вольтерра та складання математичної моделі динаміки популяцій хижаків та жертв в екосистемі.

Але для реалізації алгоритму розв’язання задачі програмісту немає необхідності вираховувати чисельність популяцій тварин безпосередньо в середовищі їх існування та впродовж досить тривалого часу. Для цього достатньо згенерувати точки рівняння Лотки-Вольтерра, які будуть передаватися нейронній мережі як дані реальних досліджень. Отримати ці точки можна запустивши метод Рунге-Кутти для уже готового рівняння Лотки-Вольтерра. Більше того, відомі параметри рівняння можна одразу порівняти із тими, які видасть нейронна мережа та зробити висновок про якість роботи реалізованого алгоритму.

2.3 Засоби реалізації

Для виконання даної роботи використано мову програмування Python, версії 3.8 та інтегроване середовище для розробки PyCharm Community Edition, версії 2021.3.1. З огляду на те, що дана мова програмування широко використовується для реалізації нейромережевих алгоритмів, а також різних математичних алгоритмів, має багато вільнодоступних бібліотек, призначених для реалізації таких алгоритмів, то використання мови Python є найбільш доцільним для даної роботи з точки зору простоти і часу реалізації та ефективності алгоритмів.

Для реалізації нейромережевого алгоритму розв’язання рівняння Лотки-Вольтерра та для оцінки його параметрів використано бібліотеку наукового машинного навчання DeepXDE, яка дозволяє значно зменшити кількість необхідного коду та оптимізувати його за допомогою різних вбудованих в неї модулів. В якості бекенду для DeepXDE було використано бібліотеку TensorFlow.

Бібліотека Numpy також знайшла своє застосування в даній роботі як зручна бібліотека для зберігання та обробки даних. Усі вхідні та вихідні числові дані алгоритмів зберігаються і передаються для наступних етапів роботи саме у вигляді масивів бібліотеки Numpy, які мають ряд переваг перед звичайними масивами при реалізації нейромережевих алгоритмів.

Для візуалізації проміжних та кінцевих результатів роботи алгоритмів використано бібліотеку Matplotlib, що дозволяє будувати різноманітні графічні представлення наборів числових даних. Графіки набагато краще сприймаються читачем і дають набагато більше уявлення про роботу і результат алгоритму ніж таблиці з числами, а також дозволяють візуально оцінити вагу похибок, які дає алгоритм.

3 РОЗВ'ЯЗАННЯ ЗАДАЧІ

3.1 Розв'язання рівняння Лотки-Вольтерра методом Рунге-Кутти

Візьмемо рівняння Лотки-Вольтерра із відомими параметрами α = 1.1, β = 0.4, γ = 0.4, δ = 0.1. Воно має наступний вигляд:

$$\frac{dx}{dt} = 1.1x - 0.4xy$$
$$\frac{dy}{dt} = 0.1xy - 0.4y \quad (7)$$

В якості досліджуваного часового проміжку візьмемо проміжок від 0 до 24 [0; 24], де t_0 = 0 – початковий момент часу досліджень, а t_n = 24 – кінцевий момент часу досліджень.

$$T = [0; 24] \quad (8)$$

Крок методу Рунге-Кутти h = 0.25. Такий малий крок дасть можливість отримати згладжений графік розв'язку рівняння. В реальних умовах даний часовий проміжок можна представити, як двадцять чотири місяці, або ж два роки. При цьому крок може бути більшим, наприклад 1. Тобто спостереження за популяціями тварин мають проводитися кожного місяця протягом двох років. В початковий момент часу 0 візьмемо кількість жертв рівну 20, а кількість хижаків рівну 5, це будуть початкові умови для рівняння:

$$x(0) = 20, \qquad y(0) = 5 \quad (9)$$

Задамо змінні *x* та *y* у вигляді масиву x, де x[0] = *x*, а x[1] = y та запишемо праві частини рівняння (7), використовуючи синтаксис мови Python:

[1.1 * x[0] – 0.4 * x[0] * x[1], 0.1 * x[0] * x[1] – 0.4 * x[1]]

Далі ми маємо передати дане рівняння, його початкові умови, початкову та кінцеву точки часу, а також часовий крок у функцію, яка обчислює розв’язок диференціального рівняння методом Рунге-Кутти і повертає два параметри x і t, де t – масив точок часу, який містить 97 елементів – чисел від 0 до 24 з кроком 0.25, а x – також масив із 97 елементів, кожним елементом якого є ще один двоелементний масив – пара числових значень, що відповідають чисельності популяцій жертв та хижаків у відповідний момент часу. Крім того, під час своєї роботи дана функція записує деякі зі значень у текстовий файл, а саме значення в моменти часу, що відповідають цілим числам. Ці записані 25 точок будуть використані як вхідні дані для нейронної мережі під час розв’язання задачі оцінки параметрів рівняння Лотки-Вольтерра.

Врешті-решт, отримавши точки функції-розв’язку, ми можемо візуально її представити, згенерувавши графік за допомогою інструменту Matplotlib. Графік розв’язку рівняння (7) на часовому проміжку (8) та з початковими умовами (9), отриманий методом Рунге-Кутти матиме вигляд як на рис. 4.

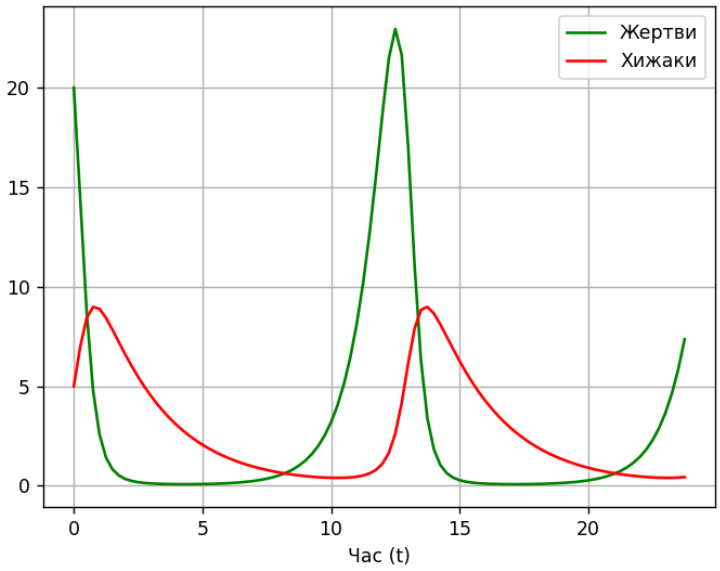


Рисунок 4 – Розв’язок рівняння Лотки-Вольтерра методом Рунге-Кутти

3.2 Розв’язання рівняння Лотки-Вольтерра нейромережевим алгоритмом

Розв’яжемо це ж саме рівняння Лотки-Вольтерра (7), використовуючи бібліотеку DeepXDE та нейронні мережі.

Розпочнімо із часового проміжку. Для цього можемо скористатися вбудованим модулем deepxde.geometry.TimeDomain(), задавши йому в параметри початкову і кінцеву точки проміжку (8).

Після цього задамо саме рівняння, виразивши похідні по часу dx_t та dy_t через засіб deepxde.grad.jacobian(). Задане рівняння матиме наступний вигляд:

dx_t - 1.1 * x + 0.4 * x* y,
dy_t - 0.1 * x * y + 0.4 * y

Наступний крок – задання початкових умов рівняння (9). Задати їх можна, скориставшись модулем deepxde.icbc.IC(), передавши йому досліджуваний часовий проміжок та значення кожної із функцій на його початку. Параметр component визначає, якої із функцій стосується дана початкова умова.

Далі необхідно об’єднати умову рівняння, досліджуваний часовий проміжок та початкові умови, передавши їх у модуль deepxde.data.PDE(), який об’єднує всю необхідну інформацію про задачу і готує її до відправки нейронній мережі. На даному етапі також вкажемо, що збираємося використати 3000 точок всередині часового проміжку та 2 точки на його краях для тренування нейромережі, а також 3000 тестових точок.

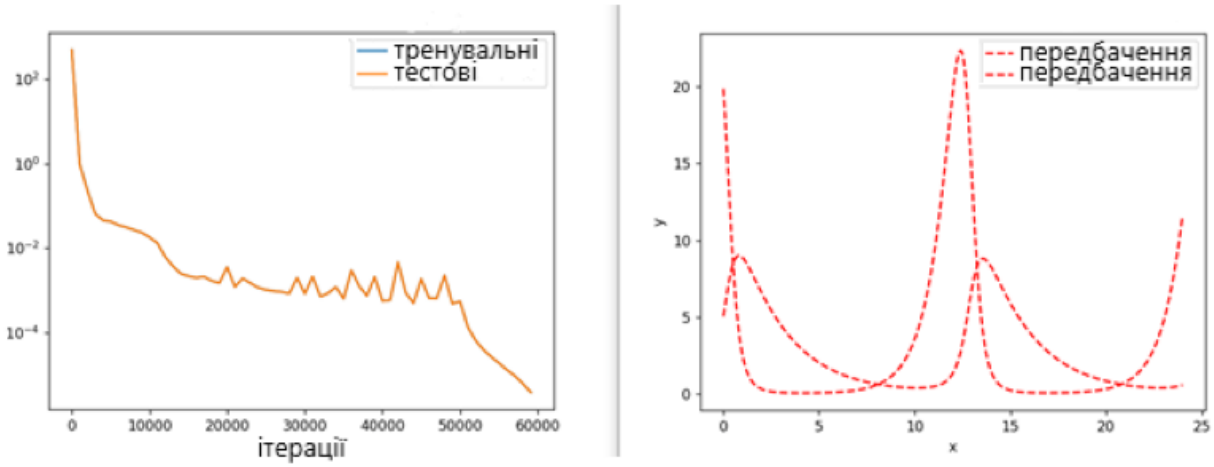
Після цього задамо гіперпараметри для нейронної мережі. Для даної задачі використаємо повнозв’язну нейронну мережу із вхідним шаром розміру 1 нейрон, 6 прихованими шарами по 64 нейрони кожен та вихідним шаром на 2 нейрони. В якості функції активації обираємо tanh, а в якості

ініціалізатора – Glorot normal. Усі ці параметри передаємо в модуль deepxde.nn.FNN(), який будує нейронну мережу.

Наступним кроком об’єднуємо нашу задачу і нейронну мережу в єдину модель за допомогою deepxde.Model(). Далі компілюємо цю мережу методом compile(), передавши adam в якості оптимізатора і обравши швидкість навчання (learning rate) рівним 0.001. Врешті-решт тренуємо нейронну мережу, задавши 50000 ітерацій для тренування. Під час тренування зберігаємо дані про зміну функції втрат та про результати тренування.

Для того, щоб отримати ще менші втрати, після першого тренування функції з оптимізатором adam обираємо інший оптимізатор L-BFGS та дотреновуємо модель із цим оптимізатором, отримуємо повністю навчену модель, готову до тестування.

Вкінці побудуємо графік зміни значень функції втрат під час тренування та отримані результати, які згенерувала нейронна мережа (рис. 5)



а) *б)*

Рисунок 5 – Результати роботи нейромережевого алгоритму: а – графік функції втрат; б – розв’язок рівняння Лотки-Вольтерра

3.3 Оцінка параметрів рівняння Лотки-Вольтерра

Візьмемо набір даних із текстового файлу, які ми записали туди під час розв'язання рівняння Лотки-Вольтерра за допомогою методу Рунге-Кутти. Цей набір даних містить 25 часових точок – цілих чисел із проміжку, заданого формулою (8). Крім цього ми маємо 25 пар значень – кількостей жертв та хижаків у відповідні моменти часу. Цей, згенерований методом Рунге-Кутти, набір даних моделює реальні дані, які ми могли б отримати безпосередньо спостерігаючи за екосистемою протягом двох років.

Спочатку вичитаємо дані з файлу і розділимо їх на часові точки та значення функцій.

Далі за допомогою модуля deepxde.Variable() задамо чотири невідомі змінні C_1, C_2, C_3 та C_4 для невідомих параметрів α, β, γ та δ відповідно, які нам треба оцінити в даній задачі. На початку задамо їх усі рівними 1.

Наступним кроком задаємо часовий проміжок (8), як це було зроблено при розв'язанні прямої задачі (знаходження розв'язку рівняння) в попередньому підпункті.

Після цього записуємо саме рівняння, знову використовуючи deepxde.grad.jacobian() для похідних першого порядку, але на цей раз, виразивши невідомі параметри через змінні C_1, C_2, C_3 та C_4:

$$\frac{dx}{dt} = C_1 x - C_2 xy,$$

$$\frac{dy}{dt} = C_4 xy - C_3 y$$

Далі необхідно задати початкові умови рівняння (9). Це робиться аналогічно до того, як ми це робили в попередній задачі. Але крім цього на даному етапі ми ще маємо передати набір точок для тренування нейронної мережі, який було вичитано з файлу. Це робиться за допомогою граничної умови deepxde.icbc.PointSetBC() – граничної умови Діріхле для множини

точок.

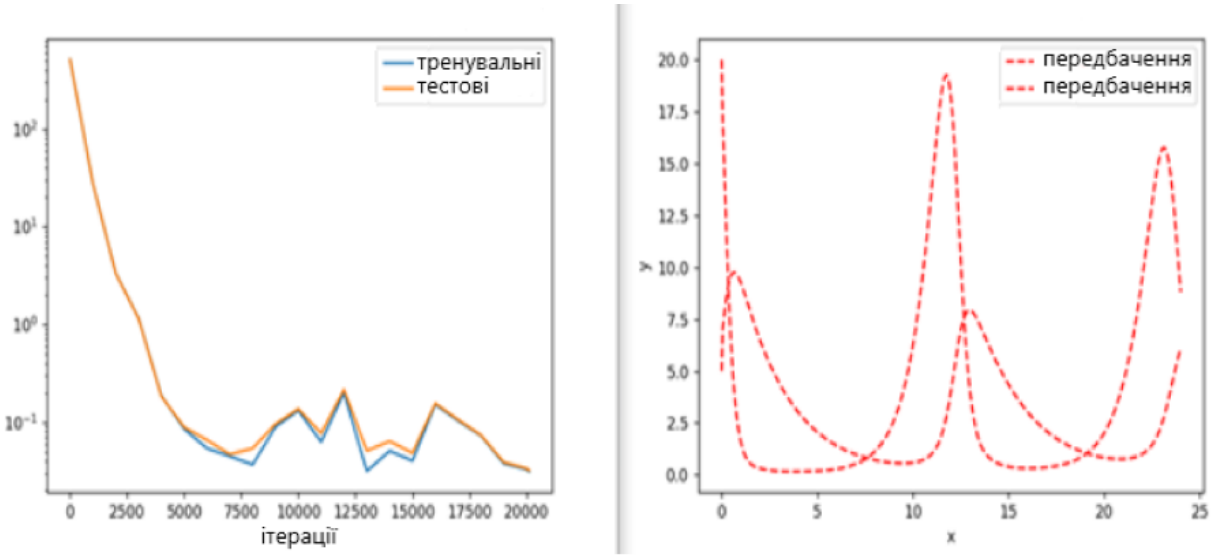
На наступному кроці об’єднуємо нашу задачу, часовий проміжок, початкові умови і набір точок для тренування за допомогою deepxde.data.PDE(). Точки з набору можуть повторюватися, це не є критично на даному етапі, але навпаки, дозволяє отримати кращу точність результату при меншій кількості необхідних реальних спостережень, тому візьмемо 400 точок всередині часового проміжку і 2 точки на його кінцях. Часові точки із файлу передамо в якості параметра anchors в додаток до основних точок, а також візьмемо 400 точок для тестування.

Архітектуру та гіперпараметри для нейронної мережі візмемо із попередньої задачі, зменшивши лише кількість ітерацій для тренування із оптимізатором adam до 20000. Але для успішної оцінки невідомих параметрів варто скористатися такою можливістю бібліотеки DeepXDE як callbacks або зворотні виклики. Об’єднаємо шукані параметри рівняння в масив та передамо їх моделі на етапі компіляції в якості зовнішніх тренованих змінних (external_trainable_variables). В процесі тренування моделі кожні 600 ітерацій будем отримувати проміжні значення шуканих змінних за допомогою модуля deepxde.callbacks.VariableValue() та записувати їх у текстовий файл. Під час виклику методу тренування моделі ми маємо передати їй наші шукані параметри через callbacks.

Після завершення тренування і тестування моделі побудуємо графік функції втрат та розв’язку отриманого рівняння Лотки-Вольтерра (рис. 6).

Також у згенерованому файлі із значеннями параметрів рівняння можна проспостерігати, як змінювалися значення шуканих параметрів впродовж тренування моделі. На ітерації 0 вони усі дорівнювали 1, як було задано на початку. Через кожні 600 ітерацій їх значення змінювалися, постійно наближаючись до фінальних, і врешті-решт вкінці процесу навчання ми отримали значення параметрів рівняння Лотки-Вольтерра, які оцінила нейронна мережа, а саме:

$$\alpha = 1.1, \qquad \beta = 0.396, \qquad \gamma = 0.415, \qquad \delta = 0.1$$



а) *б)*

Рисунок 6 – Робота нейромережі в задачі оцінки параметрів: а – графік функції втрат; б – розв’язок отриманого рівняння Лотки-Вольтерра

4 АНАЛІЗ РЕЗУЛЬТАТІВ

В цьому розділі буде проведено аналіз результатів розв’язання задач із попереднього розділу, аналіз роботи алгоритмів, за допомогою яких ці задачі були розв’язані та зроблено висновок про ефективність цих алгоритмів і доцільність їх використання для даних типів задач.

4.1 Знаходження розв’язку рівняння Лотки-Вольтерра

Для початку порівняємо алгоритм Рунге-Кутти та нейромережевий алгоритм для знаходження розв’язку диференціального рівняння Лотки-Вольтерра за заданими початковими умовами.

Алгоритм Рунге-Кутти уже давно використовується на практиці для розв’язку таких диференціальних рівнянь і відомий хорошою точністю своєї роботи при достатньо малих часових кроках. На відміну від нього, нейромережевий алгоритм з’явився пізніше, ще не набув такої популярності для розв’язування такого класу задач і потребує додаткових досліджень результатів його роботи. Порівняємо точність розв’язків обох алгоритмів, співставивши їх на графіку (рис. 7).

Як можемо бачити із рис. 7, нейромережевий метод при правильно заданих вхідних даних та гіперпараметрах нейронної мережі теж може відзначитися достатньо високою точністю розв’язку, достатньою для його використання в реальних практичних задачах. Але тим не менш, метод Рунге-Кутти є значно простішим і зрозумілішим у реалізації, не потребує додаткової обробки вхідних даних, необхідної для нейромережі, не вимагає підбору правильних гіперпараметрів та працює значно швидше за рахунок відсутності процесу навчання, який зазвичай є найтривалішим етапом у процесі роботи нейромережевих алгоритмів. До речі використання

нейронних мереж у алгоритмі також потребує значно більших потужностей комп’ютерної техніки, на якій він виконується на відміну від простого і швидкого методу Рунге-Кутти.

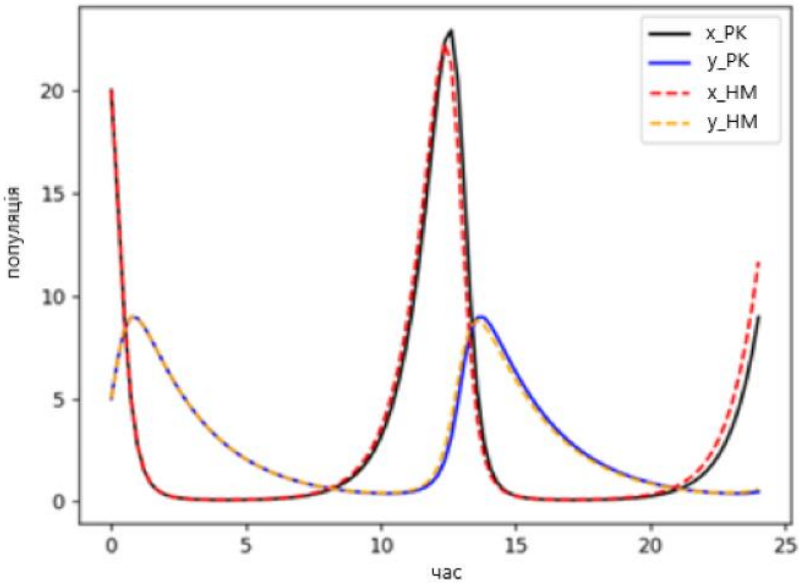


Рисунок 7 – Порівняння розв’язків рівняння Лотки-Вольтерра методом Рунге-Кутти (x_true, y_true) та нейромережевим методом (x_pred, y_pred)

Отже, для розв’язання рівнянь Лотки-Вольтерра краще скористатися класичним методом Рунге-Кутти, проте нейромережевий метод може стати в нагоді для знаходження розв’язків значно складніших рівнянь, зокрема диференціальних рівнянь у частинних похідних, або коли початкові чи граничні умови задані так, що роблять неможливим використання методу Рунге-Кутти або інших чисельних методів.

4.2 Оцінки параметрів рівняння Лотки-Вольтерра

Тепер проаналізуємо роботу алгоритму оцінки параметрів рівняння Лотки-Вольтерра та результати, які він видав.

Для оцінки точності результатів ще раз запустимо алгоритм Рунге-Кутти для розв’язання задачі (7) на часовому проміжку (8) та з початковими

умовами (9), а також для розв’язання аналогічної задачі, але з параметрами, які видала нейронна мережа при оцінці параметрів рівняння за набором точок. Виведемо розв’язки обох рівнянь, співставивши їх на одному графіку (рис. 8).

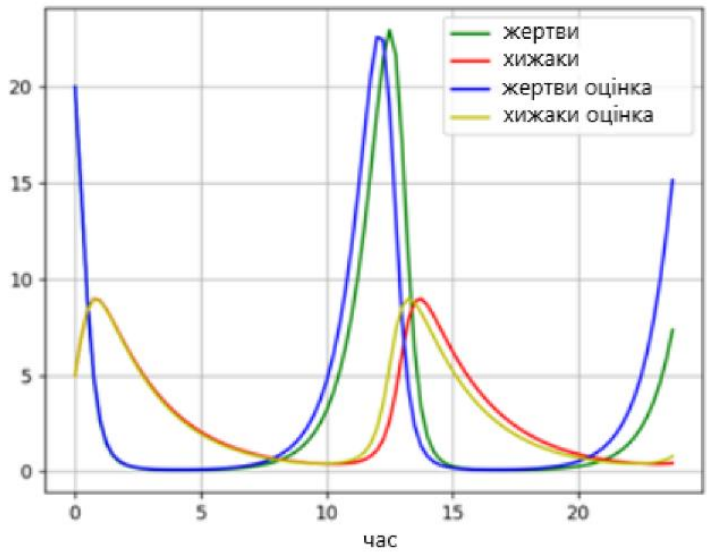


Рисунок 8 – Оцінка точності роботи нейромережевого алгоритму для оцінки параметрів рівняння Лотки-Вольтерра

Проаналізувавши графік на рис. 8 можемо сказати, що алгоритм, оцінивши невідомі параметри, згенерував майже аналогічне рівняння Лотки-Вольтерра, яке відрізняється від оригінального лише невеликою похибкою в часі. Такі малі відмінності є допустимими, але за потреби більш точного розв’язку їх можна зменшити, збільшивши розмір вхідного набору точок. Щоправда це вимагатиме на практиці частіших підрахунків кількості тварин протягом заданого проміжку часу. Наприклад, при збільшенні кількості точок до 50 алгоритм видав наступні оцінки параметрів:

$$\alpha = 1.1, \qquad \beta = 0.399, \qquad \gamma = 0.401, \qquad \delta = 0.0995$$

Вони уже майже не відрізняються від оригінальних і дадуть настільки точний результат, що відмінності на графіку фактично непомітні (рис. 9)

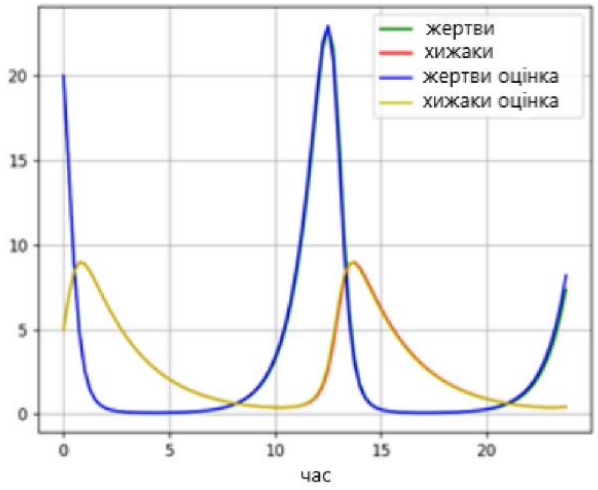


Рисунок 9 - Оцінка точності роботи нейромережевого алгоритму для оцінки параметрів рівняння Лотки-Вольтерра після подвоєння кількості вхідних точок

Окрім достатньо високої точності алгоритм є нескладним у реалізації та містить небагато програмного коду. Як можна було бачити під час реалізації алгоритму, він має багато спільних деталей із нейромережевим алгоритмом розв’язання рівняння Лотки-Вольтерра, а отже може бути швидко отриманий шляхом зміни попереднього. Більше того, нейронна мережа тут навчається досить швидко та якісно і при відносно невеликій кількості ітерацій.

Підсумовуючи вище сказане, можна зробити висновок про доцільність використання нейромережевих алгоритмів для розв’язання обернених задач диференціальних рівнянь (оцінки параметрів рівняння).

ВИСНОВКИ

В ході виконання роботи вдалося досягти поставлених цілей, а саме проаналізувати алгоритми розв’язання систем диференціальних рівнянь, побудовані на основі чисельних методів та на основі нейронних мереж, взявши як об’єкт дослідження рівняння Лотки-Вольтерра, порівняти їх ефективність та точність, вказати явні переваги та недоліки кожного з них і зробити висновок про доцільність їх використання для даного типу задач.

Також вдалося реалізувати алгоритм оцінки параметрів рівняння Лотки-Вольтерра нейромережевим методом, проаналізувати як сам алгоритм, так і результати його роботи та зробити висновок про доцільність його використання у відстежуванні чисельності популяцій тварин в екологічних системах.

Під час виконання роботи використовувалися технології з галузей машинного навчання та нейронних мереж – одних із найбільш сучасних та перспективних в сфері інформаційних технологій. Вони дають можливість розв’язувати складні проблеми, які часто буває важко або неможливо розв’язати за допомогою інших технологій і завдяки цьому знаходять своє застосування у різних сферах людської діяльності.

Результати даної роботи мають доцільність використання при дослідженні різних видів екосистем, де присутні хижі тварини та травоїдні, які слугують їжею для хижаків. Науковцям-екологам достатньо відслідковувати чисельність популяцій даних тварин впродовж певного проміжку часу і зберігати результати. Після цього, використавши отримані в результаті спостережень дані та алгоритм оцінки параметрів рівняння Лотки-Вольтерра, можна скласти математичну модель процесів, які відбуваються з популяціями тварин в екосистемі. Ця модель уже дасть змогу екологам прогнозувати майбутні зміни в чисельності популяцій, виявляти різні побічні

впливи на екосистему та приймати важливі рішення щодо необхідності втручання людини в екосистему з метою охорони певних видів тварин або ж врегулювання їх популяцій.

На сьогоднішній день існує ще багато проблемних процесів в областях фізики, медицини, екології, хімії, біології, які не можуть бути розв’язані за допомогою знань та засобів, які зараз доступні людству. А отже існує велика необхідність в подальших дослідженнях, розробках та реалізаціях нейромережевих алгоритмів для розв’язування складних диференціальних рівнянь, оцінки їх параметрів. Це врешті-решт дасть нам змогу подолати певні складні перешкоди на нашому шляху і здобути ще безліч важливих досягнень в різних галузях науки та людської діяльності, таких необхідних для всього світу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Deep learning / Y. LeCun, Y. Bengio, G. Hinton., 2015. – 444 с.
2. Why and when can deep - but not shallow - networks avoid the curse of dimensionality / [T. Poggio, H. Mhaskar, L. Rosasco та ін.]. – A review, Internat. J. Automation Comput, 2017. – 519 с.
3. A Proof That Artifi-cial Neural Networks Overcome the Curse of Dimensionality in the Numerical Approxi-mation of Black-Scholes Partial Differential Equations [Електронний ресурс] / P. Grohs, F. Hornung, A. Jentzen, P. Von Wurstemberger // preprint. – 2018. – Режим доступу: https://arxiv.org/abs/1809.02362.
4. Learning Parameters and Constitutive Relationships with Physics Informed Deep Neural Networks [Електронний ресурс] / [A. M. Tartakovsky, C. O. Marrero, P. Perdikaris та ін.] // preprint. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1808.03398 .
5. Physics-Informed Neural Networks for Multiphysics Data Assimilation with Application to Subsurface Transport [Електронний ресурс] / Q.He, D. Brajas-Solano, G. Tartakovsky, A. M. Tartakovsky // preprint. – 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1912.02968 .
6. Yang L. Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations [Електронний ресурс] / L. Yang, D. Zhang, G. E. Karniadakis // preprint. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1811.02033 .
7. DeepXDE: A Deep Learning Library for Solving Differential Equations [Електронний ресурс] / L.Lu, X. Meng, Z. Mao, G. Em Karniadakis // SIAM Review. – 2021. – 63:1 – С.208 – 228. - Режим доступу до ресурсу: https://epubs.siam.org/doi/epdf/10.1137/19M1274067 .

8. Базикін А. Д. Математична біофізика взаємодіючих популяцій / А. Д. Базикін. – М. : Наука, 1985.
9. Бугрій О. М. Основи диференціальних рівнянь: теорія, приклади та задачі : Навчальний посібник / О. М. Бугрій, Н. П. Процах, Н. В. Бугрій. – Львів, 2011. – ISBN 978-966-2645-01-9.
10. Бакаер Н. Коротка історія математичної динаміки населення [Електронний ресурс] / Н. Бакаер, П. Є. Шевчук // ISBN 979-10-343-8562-1. – 2021. – 187с. – Режим доступу до ресурсу: http://www.ummisco.ird.fr/perso/bacaer/uk.pdf .
11. Гаврилюк І. П. Методи обчислень: підручник у 2 ч. / І. П. Гаврилюк, В. Л. Макаров. – Київ: Вища школа, 1995. – 431 с. – (ч.2).
12. Маринець В. В. Теорія крайових задач для звичайних диференціальних рівнянь: Навчальний посібник [Електронний ресурс] / В. В. Маринець, В. Л. Рего, К. В. Маринець // Ужгород: Вид-во УжНУ «Говерла» – 2013. – 196 с. – Режим доступу до ресурсу: https://www.uzhnu.edu.ua/uk/infocentre/get/27991 .
13. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. [Електронний ресурс] / С. О. Субботін // Житомир : Вид. О. О. Євенок – 2020. – 184 с. – ISBN 978-966-995-189-2. – Режим доступу до ресурсу: http://eir.zntu.edu.ua/bitstream/123456789/6800/1/Subbotin_Neural.pdf .
14. Gurney K. An introduction to neural networks [Електронний ресурс] / Kevin Gurney // – 1997. – ISBN 1857286731. – Режим доступу до ресурсу: https://www.worldcat.org/title/37875698 .
15. A Survey on Solving and Discovering Differential Equations Using Deep Neural Networks [Електронний ресурс] / [H. Jung, J. Gupta, B. Jayaprakash та ін.]. – 2023. – Режим доступу до ресурсу: https://arxiv.org/abs/2304.13807 .