НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ
ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

До захисту допущено
Завідувач кафедри
___Оксана ТИМОЩУК
«___»_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління» спеціальності 124 «Системний аналіз»
на тему «Глибокі нейронні мережі у прогнозуванні популяційної динаміки»

Виконав:
студент IV курсу, групи КА-12
Кобєлєв Володимир Євгенійович _____

Керівник:
Професор кафедри ММСА, д. т. н., проф.
Дмитрієва Ольга Анатоліївна _____

Консультант з економічного розділу:
Доцент, к.е.н., Рощина Надія Василівна _____

Консультант з нормоконтролю:
к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:
к.т.н., доцент кафедри штучного інтелекту ННК ІПСА
Аркадій Маркусович Шахновський _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
____Оксана ТИМОЩУК
«__»_____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Кобєлєву Володимиру Євгенійовичу

1. Тема роботи «Глибокі нейронні мережі у прогнозуванні популяційної динаміки», керівник роботи Дмитрієва Ольга Анатоліївна, доктор технічних наук, професор кафедри ММСА, затверджені наказом по університету від 26.05.2025 р. № 1759-с
2. Термін подання студентом роботи ___.06.2025 р.
3. Вихідні дані до роботи: системи диференціальних рівнянь, що описують динаміку популяційних процесів; симуляційні траєкторії, отримані чисельними методами інтегрування; структуровані навчальні та тестові вибірки, архітектура глибокої нейронної мережі диференціального типу; кількісні метрики для оцінювання точності прогнозування.
4. Зміст роботи: аналіз методів прогнозування популяційної динаміки та оцінювання якості моделей, проєктування та начання глибокої нейронної мережі для прогнозування популяційної динаміки, оцінювання точності та узагальнювальної здатності диференціальної нейронної моделі, практична реалізація та візуалізації поведінки динамічних систем, функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): схеми архітектури глибокої нейронної мережі диференціального типу, графіки траєкторій динамічних систем на тестових вибірках, порівняльні таблиці метрик точності моделювання, візуалізація результатів симуляцій і поведінки моделей у часі, узагальнені таблиці з

результатами оцінювання, презентація до захисту кваліфікаційної роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, доц., к.е.н.		

7. Дата видачі роботи: 17.03.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Порівняльний аналіз сучасних підходів до прогнозування популяційної динаміки	21.04.2025 – 27.04.2025	Виконано
2	Обґрунтування обрання метрик для оцінювання якості моделей	28.04.2025 – 01.05.2025	Виконано
3	Проєктування архітектури глибокої нейронної мережі диференціального типу	02.05.2025 – 03.05.2025	Виконано
4	Формування навчальної вибірки на основі математичних моделей популяційної динаміки	03.05.2025 – 08.05.2025	Виконано
5	Побудова наближених вирішувачів на однокрокових стадійних і вкладених методах	09.05.2025 – 18.05.2025	Виконано
6	Реалізація програмного модуля, візуалізація результатів моделювання	19.05.2025 – 24.05.2025	Виконано
7	Експериментальне дослідження точності прогнозування на тестових траєкторіях	25.05.2025 – 05.06.2025	Виконано
8	Оформлення пояснювальної записки, підготовка презентаційних матеріалів	06.06.2025 – 15.06.2025	Виконано
9	Захист дипломної роботи	16.06.2025 – 21.06.2025	Виконано

Студент Володимир КОБЄЛЄВ

Керівник роботи Ольга ДМИТРІЄВА

РЕФЕРАТ

Дипломна робота: 84 с., 13 рис., 10 табл., 2 додатки, 17 джерел.

ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, ПОПУЛЯЦІЙНА ДИНАМІКА, СИСТЕМИ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ, НЕЙРОННІ ОБЧИСЛЕННЯ, ПРОГНОЗУВАННЯ, СИМУЛЯЦІЙНІ ТРАЄКТОРІЇ, ВИБІРКОВІ СУКУПНОСТІ, МЕТРИКИ

Об’єктом дослідження кваліфікаційної роботи є динамічні процеси популяційної динаміки, що описуються звичайними диференціальними рівняннями та їх системами із сконцентрованими параметрами.

Предметом дослідження виступають наближені вирішувачі, побудовані за допомогою глибоких нейронних мереж і орієнтовані на відновлення поведінки динамічних об’єктів.

Мета роботи полягає у розробці інтеграторів з використанням глибоких нейронних мереж диференціального типу, орієнтованих на відновлення поведінкових траєкторій динамічних об’єктів.

У роботі реалізовано побудову нейронної мережі диференціального типу, що апроксимує похідну стану системи у часі. Навчальні та тестові дані сформовано на основі симульованих траєкторій, отриманих класичними однокроковими методами інтегрування. Проведено навчання, оцінено точність прогнозу з використанням стандартних метрик.

Розроблений програмний продукт реалізовано мовою Python із використанням бібліотек PyTorch, NumPy, Matplotlib, Pandas та SciPy.

Методологія дослідження базується на математичних засадах теорії звичайних диференціальних рівнянь, теорії алгоритмів, використовує підходи обчислювальної математики, математичної статистики, машинного навчання.

Практичне значення роботи полягає у побудові універсального інтегратора для реалізації прогностичних моделей у випадках обмеженості обсягів експериментальних даних.

ABSTRACT

This diploma thesis contains: 84 p., 13 fig., 10 tabl., 2 appendices, 17 references.

DEEP NEURAL NETWORKS, POPULATION DYNAMICS, SYSTEMS OF DIFFERENTIAL EQUATIONS, NEURAL COMPUTATION, FORECASTING, SIMULATION TRAJECTORIES, SAMPLE POPULATIONS, METRICS

Object of research of this qualification thesis is dynamic processes of population dynamics described by ordinary differential equations and their systems with lumped parameters.

Subject of research is approximate solvers based on deep neural networks, aimed at reconstructing the behavior of dynamic systems.

The aim of the thesis is to develop integrators using differential-type deep neural networks, focused on reconstructing behavioral trajectories of dynamic systems.

This thesis implements a differential neural network designed to approximate the time derivative of a system’s state. The training and validation datasets were generated from simulated trajectories using classical one-step integration methods. The model was trained and its predictive accuracy assessed using standard evaluation metrics.

The developed software product was implemented in Python using the PyTorch, NumPy, Matplotlib, Pandas, and SciPy libraries.

The research methodology builds on the mathematical foundations of ordinary differential equations and algorithm theory, and incorporates tools from computational mathematics, statistical analysis, and machine learning.

The practical significance of the thesis lies in the development of a universal integrator for implementing forecasting models in cases of limited availability of experimental data.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ

ODE – Ordinary Differential Equation – звичайне диференціальне рівняння

Neural ODE – Neural Ordinary Differential Equation – диференціальна нейронна мережа

FNN – Feedforward Neural Network – прямий багатошаровий перцептрон

RNN – Recurrent Neural Network – рекурентна нейронна мережа

LSTM – Long Short-Term Memory – довготривала короткочасна пам’ять

GRU – Gated Recurrent Unit – гейтовий рекурентний блок

RK4 – Runge–Kutta of 4th order – метод Рунге–Кутти четвертого порядку

RKF45 – Runge–Kutta–Fehlberg – адаптивний метод Рунге–Кутти–Фельберга

MAE – Mean Absolute Error – середня абсолютна похибка

MSE – Mean Squared Error – середньоквадратична похибка

RMSE – Root Mean Squared Error – корінь середньоквадратичної похибки

MAPE – Mean Absolute Percentage Error – середня абсолютна відносна похибка

R^2 – Coefficient of Determination – коефіцієнт детермінації

GSW – Grass–Sheep–Wolves – модель тропічної екосистеми

V – вірусне навантаження (модель ВІЛ)

T – концентрація Т-клітин (модель ВІЛ)

G, S, W – біомаса трави, чисельність вівць, чисельність вовків (модель GSW)

model_id – ознака математичної моделі у вибірці (0 – ВІЛ, 1 – GSW)

sim_id – ідентифікатор симуляції

odeint – чисельний інтегратор ODE з бібліотеки TorchDiffEq

ВСТУП

Зростаюча складність моделей, що описують динаміку популяцій, а також необхідність у прогнозуванні їхнього довгострокового розвитку зумовлюють потребу у застосуванні сучасних обчислювальних підходів. Традиційні математичні методи опису та симуляції поведінки, такі як системи диференціальних рівнянь, хоч і залишаються важливим інструментом, мають обмежену ефективність при моделюванні складних, багатофакторних процесів із вираженою стохастичною та нелінійною складовими.

З огляду на це, значний інтерес викликають методи штучного інтелекту, зокрема глибокі нейронні мережі. Ці моделі здатні обробляти великі обсяги даних, виявляючи у них приховані залежності без потреби у попередньому визначенні аналітичної форми функціональних зв’язків. Завдяки цій властивості глибокі моделі демонструють високу результативність у завданнях прогнозування та аналізу поведінки складних динамічних систем із сконцентрованими параметрами [1].

Актуальність тематики дослідження обумовлена практичною потребою у підвищенні точності та стабільності прогнозних моделей, які застосовуються для підтримки прийняття рішень у таких галузях, як екологічне планування, управління біологічними ресурсами, епідеміологічний моніторинг і демографічне прогнозування.

Під популяційною динамікою розуміють зміни кількісного або якісного складу сукупностей взаємодіючих агентів (організмів, об’єктів, індивідів) у часі, що зумовлені як внутрішніми механізмами (розмноження, конкуренція), так і зовнішніми впливами (екологічні умови, інтервенції). Такі процеси є предметом дослідження в біології, екології, епідеміології, соціології та інших сферах. Надійне прогнозування траєкторії змін у таких системах є критично важливим для своєчасного виявлення потенційних ризиків і формування обґрунтованих стратегічних рішень.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ПРОГНОЗУВАННЯ ПОПУЛЯЦІЙНОЇ ДИНАМІКИ ТА ОЦІНЮВАННЯ ЯКОСТІ МОДЕЛЕЙ

1.1 Аналіз сучасних методів прогнозування поведінки популяційних динамічних систем

Моделювання популяційної динаміки ґрунтується на використанні математичних підходів, що дозволяють описати зміну чисельності або структури популяцій під дією внутрішніх механізмів взаємодії та змінних зовнішніх чинників середовища. У переважній більшості випадків задачі такого типу не мають аналітичного розв’язку через складність математичних моделей та багатофакторність динаміки.

Для прогнозування поведінки подібних систем застосовуються підходи, які можна поділити на кілька основних категорій:

– методи чисельного розв’язання диференціальних рівнянь та їх систем;

– статистичні моделі часових рядів;

– алгоритми класичного машинного навчання;

– методи глибокого нейронного навчання.

Методи чисельного розв’язання диференціальних рівнянь, як правило, базуються на їхній дискретизації [2]. У загальному вигляді задача Коші для системи рівнянь першого порядку записується так:

$$\frac{dy(t)}{dt} = f(t, y(t)), \qquad y(t_0) = y_0, \tag{1.1}$$

де $y(t)$ — вектор станів системи в момент часу t;

$f(t, y(t))$ — задана функція правих частин;

t_0 — початковий момент часу;

y_0 — початкові умови.

Для розв’язання таких задач використовуються чисельні методи, які можна класифікувати за різними критеріями.

Явні чисельні методи дозволяють визначити наступний стан системи, використовуючи лише інформацію з попереднього кроку, без необхідності розв’язувати допоміжні рівняння на кожному етапі інтегрування. Одним з найпростіших прикладів є метод Ейлера [3], що має вигляд:

$$y_{n+1} = y_n + hf(t_n, y_n), \tag{1.2}$$

де h — крок інтегрування;

$t_n = t_0 + nh.$

Перевагою методу є простота реалізації, проте він є умовно стійким і не може бути застосовним при розв’язанні завдань довготривалого моделювання та/або погано обумовлених і жорстких.

Неявні методи враховують значення функції у наступній точці, що вимагає розв’язання системи нелінійних рівнянь на кожному кроці. Прикладом є метод зворотного (неявного) Ейлера:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{1.3}$$

Хоча обчислювальна складність таких методів вища, вони, як правило, забезпечують абсолютну або A-α стійкість, що є принциповою умовою при роботі з жорсткими диференціальними рівняннями.

За способом використання інформації попередніх обчислювальних кроків чисельні методи поділяються на однокрокові та багатокрокові.

Однокрокові методи використовують інформацію лише з однієї попередньої точки для обчислення нового значення. Найбільш відомим представником є метод Рунге–Кутти [4] 4-го порядку, який формально записується як:

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2 * k_1}\right),$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2 * k_2}\right),$$
$$k_4 = f(t_n + h, y_n + h * k_3),$$
$$y_{n+1} = y_n + \frac{h}{6} * (k_1 + 2 * k_2 + 2 * k_3 + k_4) \quad (1.4)$$

Метод забезпечує достатньо високу точність при розумному обраній величині h і є одним з найпопулярніших однокрокових підходів. Використання методів з високим порядком точності є доцільним лише за умови контролю чисельної стійкості, оскільки навіть незначна похибка може призвести до експоненційного накопичення помилок у жорстких системах [5].

Окрему категорію складають адаптивні однокрокові методи, наприклад метод Фельберга (Runge–Kutta–Fehlberg) [6]. Він оцінює похибку інтегрування шляхом порівняння результатів 4-го і 5-го порядків та динамічно змінює величину кроку h. Це дозволяє зменшувати обчислювальні витрати без втрати точності.

Багатокрокові методи враховують значення функції у декількох попередніх точках. Відомими представниками є методи Адамса-Бешфорта (явні) та Адамса-Мултона (неявні). Метод Адамса-Бешфорта k-го порядку має загальний вигляд:

$$y_{n+1} = y_n + h * \sum_{j=0}^{k-1} \beta_j f_{n-j}, \quad (1.5)$$

де β_j — коефіцієнти методу;

$f_{n-j} = f(t_{n-j}, y_{n-j}), h$ — крок інтегрування.

Методи Адамса-Мултона — це неявні багатокрокові методи. Вони включають нове значення функції $f_{n+1} = f(t_{n+1}, y_{n+1})$, що вимагає розв'язання нелінійного рівняння на кожному кроці. Це забезпечує покращену стабільність, особливо для жорстких задач. Загальний вигляд методу Адамса-Мултона k-го порядку:

$$y_{n+1} = y_n + h * \sum_{j=0}^{k} \alpha_j f_{n+1-j}, \quad (1.6)$$

де α_j — вагові коефіцієнти методу.

Схема Бутчера застосовується для компактного подання коефіцієнтів методів Рунге–Кутти (див. табл. 1.1)

Таблиця 1.1. Загальний вигляд схеми Бутчера для стадійних методів

c_1	a_{11}	a_{12}	…	a_{1s-1}	a_{1s}
c_2	a_{21}	a_{22}	…	a_{2s-1}	a_{2s}
…	…	…	…	…	…
c_s	a_{s1}	a_{s2}	…	a_{ss-1}	a_{ss}
	b_1	b_2	…	b_{s-1}	b_s

Коефіцієнти a_{ij} визначають, як використовуються попередні етапи, b_i — вагові коефіцієнти для обчислення підсумкового значення, а c_i — проміжні точки в часовому інтервалі.

Окрему роль у моделюванні складних динамічних систем відіграють обернені задачі [7]. У таких задачах необхідно не лише розв’язати систему рівнянь, але й відновити невідомі параметри моделі за відомими експериментальними спостереженнями.

Це особливо актуально у біологічних та медичних системах, де параметри (наприклад, темпи інфікування, реакції імунної системи або рівень взаємодії популяцій) не можуть бути визначені безпосередньо, а лише оцінюються на основі непрямих вимірювань.

Такі задачі формуються як задачі мінімізації функціоналу невідповідності між результатами моделювання та даними експерименту:

$$J(q) = \sum_{k=1}^{N} \|X(t_k; q) - \phi(t_k)\|^2, \quad (1.7)$$

де q — вектор параметрів моделі;

$X(t_k; q)$ — чисельний розв’язок системи у момент часу t_k;

$\phi(t_k)$ — виміряні значення.

Для їх розв’язання застосовуються різноманітні оптимізаційні підходи — градієнтні методи, методи найменших квадратів, генетичні алгоритми, симульований відпал тощо. Вони дозволяють адаптувати чисельну модель до

реальних даних, забезпечуючи точніше відтворення реальної поведінки системи.

Статистичні методи застосовуються для аналізу та прогнозування значень у часових рядах, ґрунтуючись не на фізичній чи біологічній природі процесу, а на виявленні внутрішніх залежностей між попередніми та поточними станами. У контексті популяційної динаміки ці підходи дозволяють будувати прогнози на основі історичних спостережень, не потребуючи складної математичної структури системи.

Одним із базових представників є авторегресійна модель порядку p (AR(p)). Вона описує поточне значення змінної як лінійну комбінацію її попередніх значень:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ \dots + \phi_p y_{t-p} + \varepsilon_t, \qquad (1.8)$$

де y_t — значення змінної у момент часу t;

$\phi_1, \dots, \phi_p$ — параметри моделі;

ε_t — білий шум (незалежні випадкові змінні з нульовим середнім).

Іншим важливим підходом є модель ковзного середнього порядку q (MA(q)), яка враховує вплив поточних та попередніх збурень:

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ \dots \ + \theta_q \varepsilon_{t-q}, \qquad (1.9)$$

де θ_j — коефіцієнти моделі MA.

Поєднання AR і MA дає модель ARMA (p, q):

$$y_t = \phi_1 y_{t-1} + \ \dots \ + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \qquad (1.10)$$

Проте багато природних процесів, зокрема популяційні, мають нестабільну дисперсію або виражений тренд, що робить їх нестаціонарними. У таких випадках застосовують модель ARIMA (p, d, q), яка враховує попереднє диференціювання ряду:

$$\Delta^d y_t = \phi_1 \Delta^d y_{t-1} + \cdots + \phi_p \Delta^d y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}, \quad (1.11)$$

де $\Delta^d y_t$ — результат d-кратного диференціювання часового ряду.

ARIMA моделі є ефективним інструментом прогнозування динаміки складних систем, зокрема популяційних. Вони дозволяють будувати моделі без необхідності явного опису фізичної природи процесу, ґрунтуючись на статистичних взаємозв'язках між вхідними та вихідними змінними. Такі алгоритми ефективно працюють як з лінійними, так і з виражено нелінійними залежностями, що робить їх універсальними для багатьох прикладних задач [8].

Алгоритми класичного машинного навчання стали ефективним інструментом для прогнозування динаміки складних систем, включно з популяційними процесами. Їхньою перевагою є здатність моделювати як лінійні, так і сильно нелінійні залежності без явної формалізації фізичної сутності процесу. Ці методи опираються на алгоритмічне виявлення закономірностей у структурованих або неструктурованих даних.

До найбільш поширених підходів відносять дерева рішень, метод опорних векторів, а також ансамблеві методи, як-от Random Forest та Gradient Boosting Machines [9].

Алгоритм дерева рішень послідовно розбиває простір ознак на підпростори, обираючи ті поділи, які мінімізують функцію втрат. У випадку регресії цільова функція має вигляд:

$$\min_{j,t}\left[\sum_{i\in R_1(j,t)}\left(y_i-\bar{y}_{R_1}\right)^2+\sum_{i\in R_2(j,t)}\left(y_i-\bar{y}_{R_2}\right)^2\right], \qquad (1.12)$$

де R_1 та R_2 — області простору, сформовані на основі умови $x_j \le t, x_j > t$;

$\bar{y}_{R_k}$ — середнє значення цільової змінної в регіоні R_k.

Метод опорних векторів будує гіперплощину, яка розділяє дані з максимальним зазором. У випадку задачі регресії (SVR) мета полягає у знаходженні функції $f(x)$, яка має похибку не більшу за ε:

$$\min_{\omega,b}\frac{1}{2}\|\omega\|^2 \text{ за умови } |y_i-\langle\omega,x_i\rangle-b|\le\varepsilon \qquad (1.13)$$

Для ситуацій, де така гіперплощина не існує, вводяться допоміжні змінні ξ_i, ξ_i^*, які контролюють перевищення допуску.

Ансамблеві методи поєднують кілька моделей (базових алгоритмів) для підвищення стабільності та точності прогнозу.

Random Forest – це сукупність дерев рішень, які будуються на випадкових підмножинах ознак і даних. Остаточний прогноз — це середнє значення прогнозів окремих дерев:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x), \tag{1.14}$$

де T — кількість дерев у лісі;

$h_t(x)$ — прогноз t-го дерева.

Gradient Boosting Machines формує нову модель як покращення попередньої за допомогою врахування помилок:

$$F_k(x) = F_{k-1}(x) + \gamma_k h_k(x), \tag{1.15}$$

де $h_k(x)$ — нове дерево-регресор, яке апроксимує залишкову помилку;

γ_k — коефіцієнт навчання.

Таким чином, методи машинного навчання дозволяють моделювати складні залежності між вхідними параметрами та виходом системи без потреби у явній аналітичній моделі. Їх ефективність у прогнозуванні популяційної динаміки зростає зі збільшенням обсягу даних, якщо налаштування моделі виконано з урахуванням особливостей задачі та структури вхідної інформації. Водночас використання надто складних моделей потребує впровадження додаткових механізмів, які запобігають надмірному пристосуванню до навчальних даних, зокрема через багаторазове тестування та обмеження гнучкості моделі.

Методи глибокого навчання відкривають нові можливості у задачах моделювання популяційних динамічних процесів. Їх ключова перевага — здатність виявляти складні залежності між вхідними та вихідними характеристиками без потреби у формулюванні явної аналітичної моделі. Такі

моделі добре працюють у задачах, де присутні великий обсяг даних і довготривалі часові зв’язки.

Багатошаровий перцептрон (Feedforward Neural Network, FNN) складається з послідовно з’єднаних шарів нейронів. Кожен шар виконує афінне перетворення з наступною нелінійною активацією:

$$h^{(l)} = \sigma\left(W^{(l)}h^{(l-1)} + b^{(l)}\right), \quad (1.16)$$

де $h^{(l)}$ – вихід l-го шару;

$W^{(l)}$, $b^{(l)}$ — ваги і зсуви;

σ — функція активації.

Багатошарові перцептрони добре справляються з обробкою стаціонарних вхідних даних, але не враховують часову залежність.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) дозволяють моделювати послідовності, зберігаючи інформацію про попередні стани у внутрішній пам’яті. Типова формула оновлення стану RNN має вигляд:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b), \quad (1.17)$$

де x_t — вхід у момент часу t;

h_t — прихований стан;

σ — нелінійна функція активації.

Однак базові RNN страждають від проблеми зникнення градієнтів при довгих послідовностях. Для подолання цих проблем використовуються моделі з довготривалою короткочасною пам’яттю (Long Short-Term Memory, LSTM) та гейтові рекурентні блоки (Gated Recurrent Unit, GRU).

LSTM вводить механізм «воріт», що контролюють потоки інформації:

$$
\begin{aligned}
f_t &= \sigma\big(W_f[h_{t-1}, x_t] + b_f\big), \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\
\tilde{c}_t &= tanh(W_c[h_{t-1}, x_t] + b_c), \\
c_t &= f_t \oplus c_{t-1} + i_t \oplus \tilde{c}_t, \\
h_t &= o_t \oplus tanh(c_t),
\end{aligned}
\tag{1.18}
$$

де $\oplus$ — поелементне множення.

GRU спрощують цю структуру, поєднуючи деякі ворота, зменшуючи обчислювальну складність при збереженні ефективності.

Архітектура трансформера не використовує рекурентність, а замість цього спирається на механізм уваги (self-attention), який дозволяє мережі одночасно обробляти всі елементи послідовності [10] :

$$
Attention(Q, K, V) = softmax\left(\frac{QK}{\sqrt{d_k}}\right)V, \tag{1.19}
$$

де Q, K, V — матриці запитів, ключів та значень відповідно;

d_k — розмірність простору ключів.

Трансформери є основою сучасних моделей прогнозування часових рядів [11], оскільки забезпечують кращу паралелізацію та ефективне врахування довготривалих залежностей.

Методи глибокого навчання є найперспективнішим інструментом для прогнозування популяційної динаміки, особливо у випадках складних, багатофакторних систем із наявністю часових залежностей. Їх ефективність зростає зі збільшенням обсягу навчальних даних та за умови використання потужних обчислювальних ресурсів [12]. Вибір між багатошаровими перцептронами, рекурентними нейронними мережами, довготривалою короткочасною пам’яттю, гейтовими рекурентними блоками або трансформерами залежить від характеристик задачі, бажаної точності прогнозу та доступності обчислювальних потужностей.

1.2 Метрики для оцінювання точності прогнозування

При побудові моделей для опису популяційної динаміки важливо не лише отримати результати, але й переконатися, наскільки ці результати відповідають реальній поведінці системи. Саме тому оцінювання точності прогнозування є необхідним етапом аналізу, який дозволяє порівнювати моделі між собою та виявляти найбільш придатні для практичного застосування. Метрики, які використовуються для цієї мети, допомагають виміряти різницю між прогнозованими значеннями та фактичними даними. Існує багато таких показників, і кожен із них має свої переваги залежно від типу задачі, структури даних та бажаної точності. Одні метрики краще підходять для даних зі значними коливаннями, інші — для стабільних процесів або невеликих вибірок.

У випадку з популяційними моделями часто важливо враховувати як абсолютну похибку, так і відносну, а також чутливість до поодиноких аномальних значень. Через це використання однієї метрики зазвичай недостатньо — доцільно комбінувати кілька підходів, щоб отримати більш повну картину.

У цьому підрозділі основну увагу приділено метрикам, які найчастіше застосовуються для оцінювання точності моделей прогнозування, а також обґрунтовуються критерії вибору метрик саме для задач, пов’язаних із популяційною динамікою.

1.2.1 Аналіз основних метрик для оцінювання точності прогнозування

Після побудови моделі необхідно перевірити, наскільки її результати збігаються з реальними даними. Це робиться за допомогою спеціальних

кількісних характеристик — метрик точності, які дозволяють об'єктивно порівнювати різні моделі між собою. Нижче наведено опис найбільш поширених із них [13].

Середня абсолютна похибка MAE вимірює середню величину абсолютних відхилень прогнозу від реальних значень. Її формула має вигляд:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|yi - \hat{y}i|, \quad (1.20)$$

де yi — фактичне значення;

ŷi — прогнозоване значення;

n — кількість спостережень.

Середньоквадратична похибка MSE обчислює середнє значення квадратів різниці між фактичними і прогнозованими значеннями:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2 \quad (1.21)$$

Корінь середньоквадратичної похибки RMSE є квадратним коренем із середньоквадратичної похибки:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2} \quad (1.22)$$

Середня абсолютна відносна похибка MAPE вимірює середню відносну помилку у відсотках:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{yi - \hat{y}i}{yi}\right| \quad (1.23)$$

Ця метрика є зручною для порівняння моделей, побудованих для різних систем, проте вона не працює, якщо у даних є нульові значення. Коефіцієнт детермінації R^2 показує, наскільки добре модель пояснює варіацію у вихідних даних:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(yi - \hat{y}i)^2}{\sum_{i=1}^{n}(yi - \bar{y})^2} \quad (1.24)$$

де $\bar{y}$ — середнє значення фактичних спостережень.

Ці показники часто застосовуються разом, щоб мати більш повну картину якості прогнозу. Наприклад, MAE і RMSE дають уявлення про абсолютні відхилення, а MAPE — про відносні, що дозволяє краще оцінити ефективність моделі в різних умовах.

1.2.2 Формування критеріїв обрання метрик для задач популяційної динаміки

Не всі метрики оцінювання однаково добре підходять для всіх типів задач. У випадку прогнозування популяційної динаміки вибір метрики особливо важливий, оскільки дані часто містять шум, аномалії або різкі зміни, пов’язані зі складною поведінкою системи. Тому метрика має бути не лише зручною у розрахунку, а й придатною для аналізу специфіки динаміки популяцій.

Одним із головних факторів є стійкість до викидів. У таких системах можуть траплятися різкі зміни чисельності популяції — наприклад, унаслідок хвороб, сезонності чи людського впливу. Метрики, які сильно реагують на окремі аномальні значення (наприклад, MSE), можуть спотворити загальну оцінку. Натомість MAE у подібних випадках є більш надійним індикатором, оскільки менше піддається впливу одиничних сплесків.

Також важливо, щоб результати були доступними для розуміння та подальшого аналізу. Практична корисність метрики значною мірою залежить від того, наскільки її значення зрозумілі для фахівців, які не займаються математичним моделюванням [14]. Наприклад, MAE та RMSE легко пояснити у прикладному контексті, оскільки вони подають похибку в тих самих одиницях, що й вихідні дані. У свою чергу, відсоткові показники, як MAPE, іноді важко інтерпретувати у випадках із широким діапазоном значень.

Ще одним важливим аспектом є залежність від масштабу даних. Якщо модель використовується для аналізу як малих, так і великих популяцій, то абсолютні метрики можуть виявитися недостатньо гнучкими. У такому разі доцільно застосовувати відносні метрики, що дозволяють порівнювати результати незалежно від розміру популяції.

Окрему увагу слід приділити обробці нульових та малих значень. Багато популяційних систем передбачають періоди, коли чисельність знижується до дуже малих або навіть нульових значень. Це створює труднощі для застосування MAPE, оскільки вона включає ділення на фактичне значення. У подібних ситуаціях більш доцільними є MAE чи RMSE, які не мають таких обмежень.

У результаті, ефективний аналіз якості моделі в задачах популяційної динаміки рідко базується лише на одній метриці. Часто використовують кілька показників одночасно — це дає змогу краще зрозуміти сильні та слабкі сторони моделі й адаптувати її до реальних сценаріїв прогнозування.

1.3 Оцінювання показників стійкості

Стійкість є фундаментальною властивістю динамічних систем, що моделюють популяційні процеси, і характеризує здатність таких систем протистояти впливу зовнішніх або внутрішніх збурень. Зокрема, якщо після короткочасної зміни параметрів або початкових умов система повертається до свого вихідного або наближеного стану, її вважають стійкою. Така поведінка важлива для моделювання реальних біологічних або екологічних систем, де стабільність часто є необхідною умовою існування популяції [14].

У контексті популяційної динаміки стійкість означає збереження структурної цілісності популяції при зміні зовнішніх умов або під дією випадкових флуктуацій. Наприклад, у моделі, що описує взаємодію видів,

стійкість може полягати у здатності популяції одного з видів не зникати повністю у відповідь на коливання чисельності іншого.

Формальне дослідження стійкості проводиться з використанням математичних критеріїв, таких як аналіз рівноважних станів та поведінки системи поблизу них. Розв’язання таких задач дозволяє зрозуміти, чи зберігає система передбачувану динаміку при зміні вихідних умов, а отже — оцінити надійність та реалістичність моделі. У прикладному аспекті, це має велике значення для задач екологічного планування, біомедичних досліджень та управління біологічними ресурсами.

Під час аналізу моделей популяційної динаміки важливо враховувати не лише точність прогнозів, але й стійкість системи до збурень. Це дозволяє зрозуміти, чи зберігається стабільна поведінка моделі при зміні умов або початкових даних. Для цього застосовують кілька показників, які дають змогу оцінити різні аспекти динаміки.

Один із базових інструментів — це власні значення матриці Якобі, які обчислюються в околі рівноважних станів. Якщо всі ці значення мають від’ємні дійсні частини, модель вважається локально стійкою — тобто невеликі відхилення не викликають довготривалих змін у системі.

Ще одним поширеним підходом є оцінка чутливості системи до незначних змін на початковому етапі її розвитку, що дозволяє виявити, наскільки швидко з часом можуть відхилятися одна від одної траєкторії, які починалися з близьких умов. Якщо такі відхилення згасають, систему вважають стійкою. Якщо ж ці відхилення зростають, це може свідчити про хаотичну або нестабільну поведінку.

До більш прикладних показників належить амплітуда відхилення після збурення, яка визначає, наскільки далеко система "відходить" від рівноваги у відповідь на зовнішній вплив. Разом з цим аналізують час релаксації, що відображає швидкість повернення до стабільного стану.

Також у деяких випадках оцінюють, наскільки система здатна зберігати стабільність під час впливу зовнішніх чинників, а також наскільки швидко

вона може повернутися до звичного стану після таких змін. Такі властивості особливо важливі при моделюванні реальних біологічних процесів або змін довкілля.

Застосування комбінованого підходу до оцінки стійкості дозволяє отримати більш повну картину поведінки динамічної системи та підвищити надійність моделювання.

1.3.1 Критерії вибору показників для аналізу популяційних систем

У процесі практичного аналізу моделей популяційної динаміки важливо не лише знати, які існують показники стійкості, але й вміти правильно обрати ті з них, які найкраще підходять до конкретної задачі. Від правильного вибору залежить, наскільки адекватно модель буде відображати дійсну поведінку системи в умовах зовнішніх чи внутрішніх змін.

Перш за все, враховується відповідність показника до типу моделі та природи процесу, що досліджується. Наприклад, для систем із вираженою рівноважною поведінкою доцільно аналізувати характер поведінки поблизу рівноваги — зокрема, за допомогою математичних властивостей, пов'язаних із похідними (наприклад, аналізу матриці Якобі). Натомість для складних або непередбачуваних систем доцільніше використовувати підходи, що оцінюють чутливість до початкових умов.

Другий важливий критерій — чутливість до змін параметрів. Якщо модель має високу залежність від значень певних коефіцієнтів, показник стійкості повинен добре відображати ці зміни. Це допомагає виявити, у яких умовах система може перейти в нестабільний режим, або коли потрібна корекція параметрів.

Також важливо, щоб результати були доступні для логічного пояснення та подальшого практичного використання. Показники повинні мати чітке

пояснення і бути зручними для аналізу не лише з математичної, а й з прикладної точки зору. Наприклад, час релаксації легко пояснити як характеристику, що демонструє швидкість повернення системи до норми.

Ще один важливий фактор — це складність обчислень і вимоги до обчислювальних ресурсів. У реальних умовах, особливо під час роботи з великими наборами даних або складними моделями, слід обирати такі показники, які можна ефективно обчислити з наявними засобами. Деякі теоретично точні методи можуть виявитися занадто затратними для застосування у практичному середовищі.

Оптимальним підходом є використання не одного, а кількох взаємодоповнюючих показників. Це дозволяє сформувати більш об'єктивне уявлення про поведінку моделі та забезпечити кращу адаптацію результатів до потреб конкретного застосування — чи то прогнозування екологічних змін, чи моделювання поширення інфекцій [14].

1.4 Висновки до розділу 1

У розділі 1 було проведено розгорнутий аналіз методів прогнозування популяційної динаміки та критеріїв оцінювання точності й стійкості відповідних моделей. Розглянуто основні чисельні, статистичні, алгоритмічні та глибинні підходи, що застосовуються у завданнях моделювання складних біоекологічних процесів.

Серед чисельних методів виокремлено явні та неявні схеми, зокрема метод Ейлера, зворотного Ейлера, Рунге–Кутти, а також адаптивний метод Фельберга. Останній виявився особливо ефективним завдяки можливості автоматичного контролю похибки та регулювання кроку інтегрування, що дозволяє зменшити обчислювальне навантаження без втрати точності в умовах складної динаміки.

Багатокрокові методи (Адамса-Бешфорта, Адамса-Мултона) доповнюють класичний арсенал інтегрування, забезпечуючи високу ефективність для задач із тривалими симуляціями. Статистичні моделі (AR, MA, ARIMA) застосовуються для аналізу часових рядів та мають високу інтерпретованість, однак менш ефективні при складних або нестабільних залежностях.

Алгоритми машинного навчання, як-от дерева рішень і ансамблеві методи, дозволяють працювати з нелінійними багатовимірними даними, а методи глибокого навчання — зокрема глибокі багатошарові перцептрони (Feedforward Neural Networks, FNN), рекурентні нейронні мережі (Recurrent Neural Networks, RNN), моделі з довготривалою короткочасною пам’яттю (Long Short-Term Memory, LSTM), гейтові рекурентні блоки (Gated Recurrent Units, GRU) та трансформерні архітектури — забезпечують адаптацію до довготривалих послідовностей і складної динаміки, що особливо актуально для багатофакторних популяційних моделей.

Для оцінювання точності моделей використовуються як абсолютні похибки (MAE, MAPE), так і квадратичні показники (MSE, RMSE), а також коефіцієнт детермінації (R^2), що дозволяє зіставляти якість прогнозів у різних умовах. Такий підхід підтверджує доцільність комплексної оцінки ефективності моделей із урахуванням характеру вхідних даних. Аналіз стійкості, зокрема за допомогою матриць Якобі, експонентів Ляпунова та характеристик відновлення після збурення (наприклад, часу релаксації), дозволяє оцінити надійність поведінки моделей у змінених умовах.

Таким чином, обґрунтоване прогнозування популяційної динаміки потребує поєднання точних чисельних методів, таких як адаптивний метод Фельберга, та інтелектуальних алгоритмів аналізу даних. Такий інтегрований підхід дозволяє створити моделі з високою точністю, стабільністю та адаптивністю до реальних сценаріїв розвитку динамічних систем.

Варто також виокремити диференціальні нейронні моделі (Neural Ordinary Differential Equations, Neural ODE), які об'єднують переваги глибоких

нейронних мереж із математичною строгістю чисельного інтегрування, відкриваючи нові можливості для моделювання безперервної динаміки у складних популяційних системах.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА НАЧАННЯ ГЛИБОКОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ПРОГНОЗУВАННЯ ПОПУЛЯЦІЙНОЇ ДИНАМІКИ

2.1 Проєктування архітектури глибокої нейронної мережі диференціального типу

У сучасній прикладній математиці та комп'ютерному моделюванні важливе місце посідають глибокі нейронні мережі, здатні ефективно апроксимувати складні залежності між вхідними параметрами та цільовими змінними. В окремому класі задач — зокрема, у моделюванні популяційної динаміки — перспективним є підхід, що поєднує класичні диференціальні рівняння з нейромережевими структурами. Такий підхід реалізується в архітектурах типу Neural ODE (Neural Ordinary Differential Equations), які дозволяють розглядати еволюцію системи як неперервний процес, описаний параметризованою функцією похідної від стану системи.

Архітектура розробленої моделі побудована у вигляді багатошарового перцептрона, реалізованого в середовищі PyTorch (рис. 2.1). На вхід моделі подається вектор фіксованої розмірності, який включає множину параметрів $P_1 \dots P_m$, що описують стан системи в певний момент часу, а також ідентифікатор моделі model_id, який дозволяє мережі розрізняти тип математичної моделі (наприклад, ВІЛ або GSW). Такі вектори формуються на основі множини симуляцій $S_1 \dots S_n$, де кожна симуляція містить послідовність змін стану системи в часі. Для навчання мережі кожен окремий стан витягується з симуляції і перетворюється на окремий вхідний приклад.

Після надходження вхідного вектору мережа послідовно обробляє його трьома повно зв'язними шарами: два з них мають по 256 нейронів із активаціями ReLU і Tanh, а третій — 128 нейронів із ReLU. Вихідний шар формує вектор $Q_1 \dots \mathrm{Q_f}$, що містить наближені значення шуканих показників. На етапі передбачення модель зберігає значення model_id, що дозволяє

інтегратору torchdiffeq.odeint правильно обробляти траєкторію залежно від типу моделі. Таким чином, архітектура забезпечує єдину універсальну структуру для роботи з різними типами симуляцій.

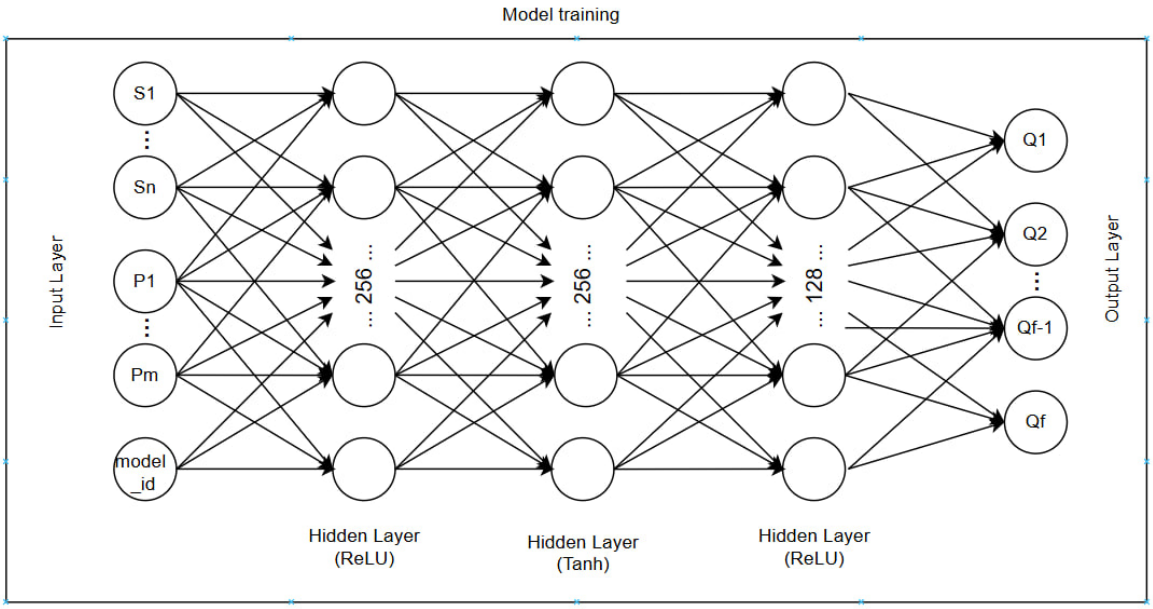


Рисунок 2.1 – Архітектура моделі NeuralODE під час навчання

На етапі побудови траєкторій (тобто при оцінюванні якості моделі) до нейромережі подається лише вектор параметрів $P_1 \dots P_m$, оскільки інформація про тип моделі model_id вже закодована в її структурі після навчання. Архітектура мережі залишається незмінною, що демонструє її універсальність та стабільність у роботі з різними типами вхідних даних (рис. 2.2).

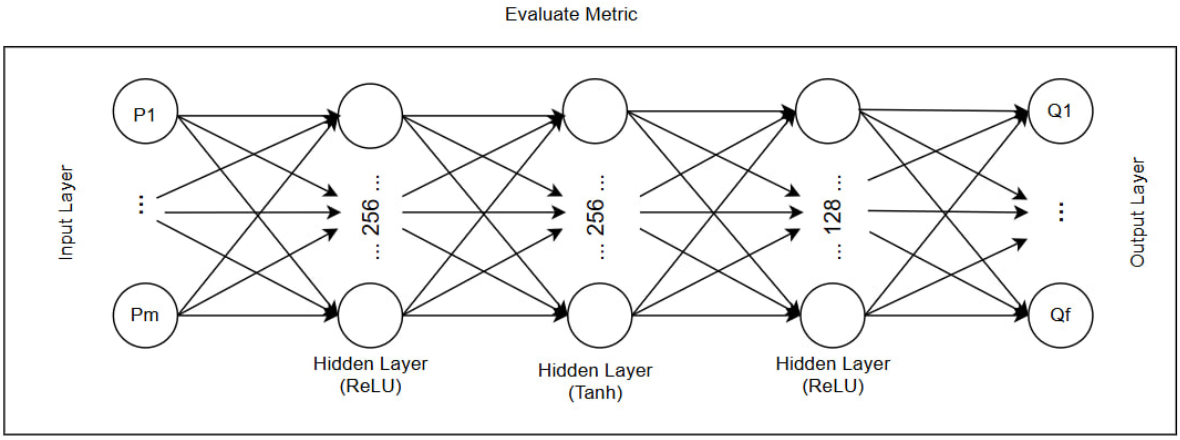


Рисунок 2.2 – Архітектура моделі NeuralODE під час побудови траєкторій та оцінювання

Завдяки єдиній архітектурі модель може працювати з різними типами динамічних систем як під час навчання, так і на етапі оцінювання. Це дає змогу застосовувати її до різних сценаріїв у межах одного підходу.

При цьому для жорстких (stiff) систем, які іноді трапляються у біологічних або фізіологічних задачах, важливо зберігати стабільність інтегрування. У таких випадках доцільно використовувати адаптивні або імпліцитні методи [15].

2.2 Вибір та обґрунтування математичних моделей та підготовка даних до навчання

Для побудови та навчання глибокої нейронної моделі прогнозування популяційної динаміки було обрано дві прикладні математичні моделі. У цьому підпункті розглядаються особливості цих моделей та методика генерації відповідного набору симуляційних даних.

У якості тестових прикладів були використані дві системи, які відображають біомедичну та екосистемну динаміку. Першою моделлю стала система імунної відповіді на ВІЛ, що описує взаємодію CD4+ Т-клітин і вірусних частинок (віріонів). Базовий варіант цієї моделі представлено у джерелі [16], проте в межах даного дослідження використано її модифіковану форму з уточненими параметрами, адаптовану до потреб симуляційного аналізу.

$$\begin{cases} \dfrac{dT}{dt} = s - d \cdot T - k \cdot T \cdot V \\ \dfrac{dV}{dt} = k \cdot T \cdot V - c \cdot V \end{cases} \tag{2.1}$$

де T — концентрація Т-клітин;

V — вірусне навантаження.

Початкові умови:

$T_0 \in [0.8, 1.2], V_0 \in [0.05, 0.2]$,

s = 0.03, d = 0.01, k = 0.001, c = 0.1 — сталі параметри.

Другою була обрана класична тропічна екосистема 'Трава–Вівці–Вовки', яка представлена трьома взаємозалежними змінними — біомасою трави, популяцією вівць та популяцією вовків. Модель базується на структурі, поданій у джерелі [17], проте реалізована у спрощеній формі зі звичайними добутками між змінними без дробових трофічних термів.

$$\begin{cases} \frac{dG}{dt} = a \cdot G(1 - G) - b \cdot G \cdot S \\ \frac{dS}{dt} = c \cdot G \cdot S - d \cdot S \cdot W \\ \frac{dW}{dt} = e \cdot S \cdot W - f \cdot W \end{cases}, \quad (2.2)$$

де G, S і W — відповідно біомаса трави, чисельність популяцій вівць і вовків, які варіювались у межах $G_0 \in [0.3, 0.7], S_0 \in [0.1, 0.5], W_0 \in [0.05, 0.2]$.

Для забезпечення навчання на крайових випадках були додатково згенеровані симуляції зі значеннями $S_0 = 0.1, W_0 = 0.05$, при варіативному G_0. $a = 1.0, b = c = d = e = f = 0.1$ — сталі параметри.

Для кожної з моделей було реалізовано чисельне інтегрування систем звичайних диференціальних рівнянь за допомогою функції solve_ivp з пакету SciPy. Використовувався метод Рунге–Кутти четвертого порядку точності (1.4). Моделювання виконувалось на фіксованому часовому проміжку [0, 30], з рівномірним розбиттям на 50 кроків.

У випадку моделі ВІЛ початкові умови варіювались випадковим чином у межах $T_0 \in [0.8, 1.2]$, $V_0 \in [0.05, 0.2]$. Для моделі GSW у основному масиві симуляцій (sim_id від 1000 до 1999) випадково ініціалізувалися усі три змінні: $G_0 \in [0.3, 0.7]$, $S_0 \in [0.1, 0.5]$, $W_0 \in [0.05, 0.2]$. Додатково було згенеровано п’ять крайових симуляцій (sim_id від 2000 до 2004), в яких значення S_0 і W_0 фіксувалися на мінімальних значеннях ($S_0 = 0.1$, $W_0 = 0.05$), тоді як G_0

залишалося варіативним у межах [0.3, 0.7]. Такий підхід дозволив моделі побачити типові й граничні сценарії поведінки системи.

Моделювання реалізовано окремо для кожної системи у скриптах simulate_hiv.py та simulate_grass_sheep_wolves.py. Було сформовано вибірки симуляцій, що охоплюють як типові сценарії, так і крайові умови, необхідні для подальшого навчання моделі. Результати зберігались у таблицю multitrajectory.csv (табл. 2.1), де кожна симуляція отримувала унікальний sim_id, а поле model_id позначало відповідну математичну модель (0 — ВІЛ, 1 — GSW).

Реалізація навчання передбачає, що кожен вектор стану має структуру [T, V, G, S, W, model_id], де значення неактуальних змінних для конкретної моделі заповнюються нулями. Це дозволяє зберігати фіксовану розмірність вхідних даних при тренуванні єдиної мережі на декількох типах динамічних систем. Усі значення змінних у моделі GSW перед подачею на вхід додатково нормалізуються в інтервалі [0, 1], тоді як для ВІЛ значення використовуються у вихідному масштабі.

Нейронна мережа, що реалізує праву частину диференціального рівняння, приймає поточний стан і момент часу, після чого інтегрується за допомогою методу odeint, що вирішує задачу Коші. Ознака model_id, передана у вхідному векторі, забезпечує селективну адаптацію моделі до різних класів задач без зміни її архітектури.

Для оцінювання якості узагальнення моделі під час навчання використовувався часовий розподіл даних: кожне п’яте значення часу в кожній траєкторії виключалося з тренувальної вибірки і використовувалося лише для тестування. Такий підхід дозволяє зберегти структурну цілісність динаміки й забезпечити контроль за точністю прогнозу в реалістичних умовах.

2.3 Формування навчальної вибірки на основі математичних моделей популяційної динаміки

Побудова навчальної вибірки ґрунтується на математичній постановці задачі Коші (1.1), яка визначає динаміку системи у вигляді звичайного диференціального рівняння з початковими умовами. Під час генерації даних кожна симуляція відповідає певній математичній моделі (наприклад, ВІЛ або GSW), яка чисельно розв’язується. У результаті формується послідовність значень, що відображають зміну основних популяційних змінних у часі. Саме ці дані надалі використовуються для навчання нейронної моделі, яка має навчитися передбачати, як змінюється стан системи залежно від її поточного значення.

Після генерації симуляцій всі дані було зведено в уніфіковану таблицю multitrajectory.csv (табл. 2.1), зі спільною структурою для обох моделей. Кожен рядок містить часову мітку t, значення змінних T, V, G, S, W, ідентифікатор симуляції sim_id та ознаку model_id. У випадку моделі ВІЛ активними є лише змінні T та V, решта заповнюється нулями. Для моделі GSW, навпаки, активними є G, S і W, а T та V не використовуються. Це забезпечує фіксовану розмірність вхідного вектору для моделі.

Таблиця 2.1 – Фрагмент симульованих даних з таблиці multitrajectory.csv для моделей ВІЛ та GSW

sim_id	t	T	V	G	S	W	model_id
0	0.00000	1.07366	0.08984	0.00000	0.00000	0.00000	0
0	0.30303	1.07946	0.08719	0.00000	0.00000	0.00000	0
0	0.60606	1.08524	0.08461	0.00000	0.00000	0.00000	0
0	0.90909	1.09101	0.08211	0.00000	0.00000	0.00000	0
…	…	…	…	…	…	…	…
1	0.00000	0.00000	0.00000	0.57311	0.40516	0.05129	1
1	0.61224	0.00000	0.00000	0.69769	0.41998	0.04948	1
1	1.22449	0.00000	0.00000	0.79301	0.43836	0.04778	1
1	1.83673	0.00000	0.00000	0.85741	0.45982	0.04619	1

Для симуляцій моделі GSW було застосовано нормалізацію вхідних змінних до інтервалу [0,1] за допомогою методу мінімакс-нормалізації, яка ґрунтується на допустимих межах кожної змінної. Такий підхід дозволяє уникнути переваги змінних з більшими масштабами при тренуванні нейронної мережі. Для моделі ВІЛ нормалізація не застосовувалась, оскільки всі вхідні значення перебували в порівнянних діапазонах і не вимагали масштабування.

Для формування навчальної та тестової вибірок було застосовано стратифікований підхід із відбиранням за часом. Із кожної симуляції кожне п’яте значення часової послідовності вилучалося з тренувального набору та використовувалося як тестова точка. Це дозволило зберегти часову послідовність у даних, уникнути витоку інформації між підвибірками та забезпечити незалежність оцінювання точності моделі на нових точках.

2.4 Опрацювання альтернативних підходів до побудови наближених вирішувачів

У процесі моделювання динаміки складних систем за допомогою машинного навчання сформувалося кілька підходів до побудови наближених вирішувачів звичайних диференціальних рівнянь (ЗДР). Кожен із цих підходів має свої переваги та обмеження, залежно від типу задачі, структури даних і вимог до точності чи обчислювальної складності.

Одним з базових підходів до побудови наближених вирішувачів є використання глибоких багатошарових перцептронів (Feedforward Neural Networks, FNN), які апроксимують функцію правої частини диференціального рівняння. Операцію, яка виконується на кожному шарі такої мережі, описується формулою (1.16). FNN приймає на вхід вектор поточного стану або пару (t, y) і повертає оцінку похідної $\frac{\mathrm{d}y}{\mathrm{d}_t}$, яку можна використовувати для подальшої чисельної інтеграції. Основні переваги підходу — висока

швидкість обчислень, гнучкість налаштувань та простота реалізації. Проте така архітектура не має внутрішньої пам’яті та не враховує часовий контекст, що обмежує її застосування у задачах, де необхідна послідовна динаміка або облік попередніх станів.

Іншим поширеним підходом до моделювання послідовних даних є рекурентні нейронні мережі (Recurrent Neural Networks, RNN), які враховують контекст попередніх станів завдяки зворотному зв’язку в архітектурі. На відміну від класичних перцептронів, RNN формують вихід на основі не лише поточного входу, але й прихованого стану, що дозволяє моделювати часову залежність. Типова структура рекурентного шару описується у вигляді (1.17). Завдяки цій рекурентній залежності, такі моделі широко застосовуються для прогнозування часових рядів та обробки послідовностей. Проте RNN мають істотні обмеження, пов’язані з нестабільністю градієнтів під час навчання: при довгих послідовностях градієнти або затухають (vanishing gradient), або вибухають (exploding gradient), що ускладнює оптимізацію. Крім того, базові RNN погано масштабуються та мають обмежену здатність до збереження довгострокових залежностей, що обмежує їх застосування у задачах із тривалою динамікою.

Покращенням класичних RNN стали моделі з довготривалою короткочасною пам’яттю (Long Short-Term Memory, LSTM), які зберігають контекстні залежності на більших часових проміжках. На відміну від звичайних рекурентних мереж, LSTM використовують додаткову структуру, яка включає кілька типів «воріт» — зокрема, вхідні, вихідні та «ворота забування», що дозволяє контролювати потік інформації в середині блоку. Математична модель LSTM описується у вигляді (1.18). Ці властивості дозволяють зменшити проблему затухаючого градієнта і суттєво покращити здатність мережі зберігати довготривалі залежності. LSTM добре підходять для задач з дискретною часовою природою — таких як обробка мовлення або тексту. Проте вони залишаються обмеженими у застосуванні до безперервних

систем, оскільки не відтворюють безперервної інтеграції стану, як це необхідно для точного розв’язання задач звичайних диференціальних рівнянь.

На відміну від вищезгаданих підходів, диференціальні нейронні моделі (Neural Ordinary Differential Equations, Neural ODE) дозволяють моделювати зміну стану системи у формі безперервної динаміки. Такий підхід безпосередньо реалізує задачу Коші (1.1), де права частина $f(y,t)$ апроксимується глибокою нейронною мережею. Замість пошарового передавання сигналу, як у класичних FNN, тут обчислюється чисельне рішення системи за допомогою ODE-інтегратора, зокрема методів Рунге–Кутти. Це дозволяє інтегрувати траєкторію вхідного вектору з урахуванням його глобального розвитку у часі. Завдяки цьому Neural ODE поєднує точність відтворення динаміки з гнучкістю нейронного наближення, забезпечуючи узагальненість і адаптивність до широкого класу динамічних задач.

2.5 Висновки до розділу 2

У розділі 2 було здійснено практичну реалізацію підходу до прогнозування популяційної динаміки на основі глибокої диференціальної нейронної моделі. В рамках реалізації побудовано архітектуру Neural ODE, яка дозволяє апроксимувати динамічні зміни системи у вигляді задачі Коші з чисельною інтеграцією правої частини, параметризованої нейронною мережею.

Для навчання було згенеровано симульовані траєкторії для двох класів систем — ВІЛ та GSW, з подальшою нормалізацією даних та формуванням універсального вхідного вектору. Реалізовано структурний поділ даних на тренувальні та тестові вибірки за принципом часової стратифікації. Архітектура була протестована на множині прикладів, і отримані результати підтвердили її здатність узагальнювати динаміку як біомедичних, так і

екосистемних процесів. У підпункті 2.4 здійснено аналіз альтернативних підходів до побудови наближених вирішувачів, зокрема FNN, RNN, LSTM та GRU. Показано, що ці підходи мають низку переваг, проте не відтворюють безперервної природи зміни стану системи, що знижує точність при моделюванні складних ЗДР.

З точки зору формалізації задачі навчання, модель оптимізувала функціонал відхилення між прогнозованими та цільовими траєкторіями у вигляді (1.7), що дозволило досягти узгодження моделі з динамікою системи на вибраній множині часових точок.

Застосування структури уніфікованої вибірки з маркуванням model_id забезпечило гнучке розширення підходу на декілька типів систем без необхідності модифікації архітектури. У майбутніх дослідженнях доцільно перевірити стійкість моделі до шумових збурень, застосувати підхід до експериментальних даних, а також розширити архітектуру на задачі з частинними диференціальними рівняннями.

Враховуючи здатність диференціальної нейронної моделі одночасно зберігати гнучкість архітектури, точність чисельної інтеграції та узагальнювальну здатність, саме підхід Neural ODE було обрано для реалізації у рамках проєкту. Він поєднує сильні сторони класичних чисельних методів і глибоких нейронних мереж, що є актуальним для моделювання багатофакторної популяційної динаміки.

РОЗДІЛ 3. ОЦІНЮВАННЯ ТОЧНОСТІ ТА УЗАГАЛЬНЮВАЛЬНОЇ ЗДАТНОСТІ ДИФЕРЕНЦІАЛЬНОЇ НЕЙРОННОЇ МОДЕЛІ

3.1 Методика проведення експерименту та структура даних

Для оцінювання ефективності диференціальної нейронної моделі було сформовано симуляційний набір даних на основі двох математичних моделей популяційної динаміки — ВІЛ та екосистемної системи типу «трава–вівці–вовки» (GSW). Кожна з моделей описувалася системою звичайних диференціальних рівнянь, розв’язок яких обчислювався чисельно методом Рунге–Кутти четвертого порядку точності. Симуляції охоплювали часовий проміжок [0, 30], поділений на 50 рівномірних кроків.

Для навчання та оцінювання моделі використовувалися симуляції, сформовані відповідно до методики, описаної в підпунктах 2.2–2.3. Навчальна та тестова вибірки формувались за принципом стратифікації по часових точках, що дозволило зберегти часову структуру та уникнути переобучення.

Масштабування значень застосовувалося лише до змінних моделі GSW (G, S, W), які було нормалізовано до інтервалу [0, 1] методом мінімакс-нормалізації. Для ВІЛ-моделі нормалізація не здійснювалася, оскільки початкові значення були стабільними і не потребували додаткової обробки.
Якість моделі оцінювалася за допомогою таких метрик: MAE (середня абсолютна похибка), MSE (середньоквадратична похибка) та R^2 (коефіцієнт детермінації). Ці метрики дозволяють охарактеризувати як абсолютну точність моделі (MAE, MSE), так і відповідність форми прогнозу до фактичної динаміки (R^2).

3.2 Візуалізація результатів моделювання

Для візуального аналізу результатів роботи моделі були побудовані графіки зміни динамічних змінних у часі для обох типів систем — ВІЛ та GSW (Трава–Вівці–Вовки). На рисунках нижче наведено приклади відтворених траєкторій, отриманих після навчання моделі. На графіках зазначено model_id, sim_id, початкові значення змінних T0, V0, G0, S0, W0, а також найкращі з трьох метрик точності для відповідних змінних. Це дозволяє одночасно оцінити і динамічну поведінку прогнозу, і кількісну якість моделі у кожному прикладі.

Графіки для моделі ВІЛ наведені на рис. 3.1 – 3.3:

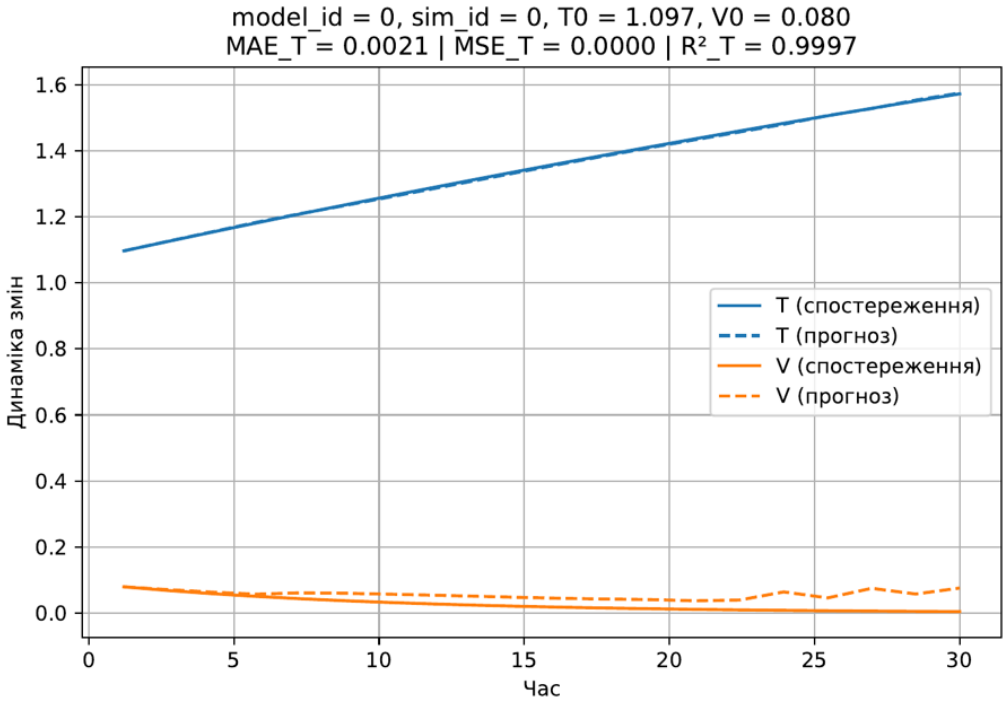


Рисунок 3.1 – Відтворення траєкторії моделі ВІЛ, приклад 1

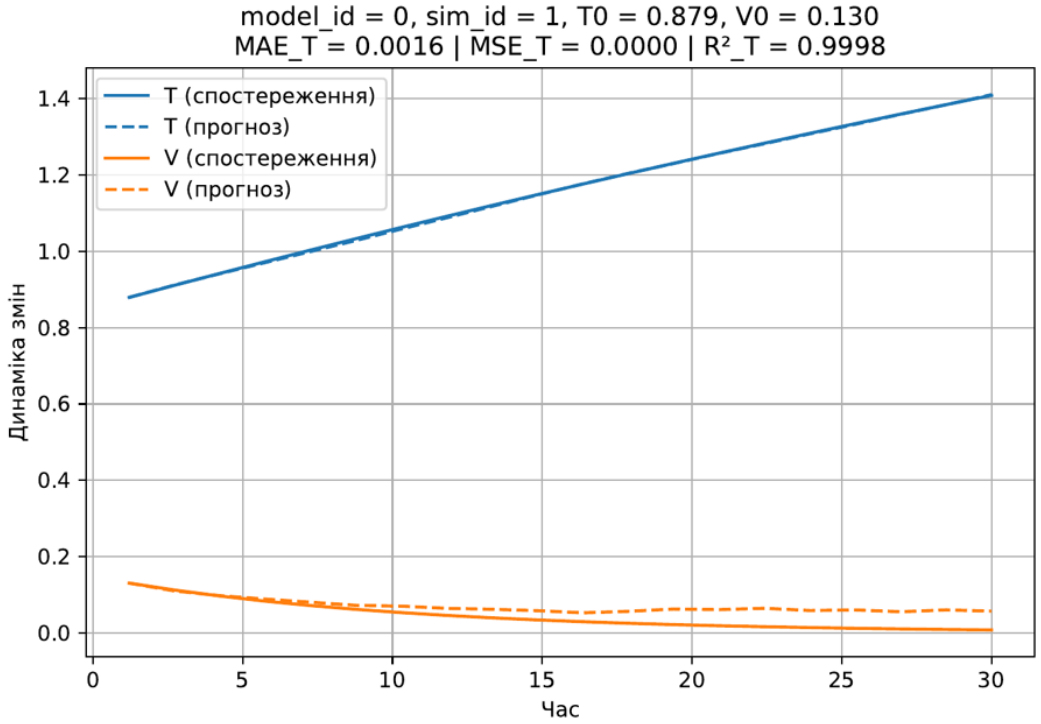


Рисунок 3.2 – Відтворення траєкторії моделі ВІЛ, приклад 2

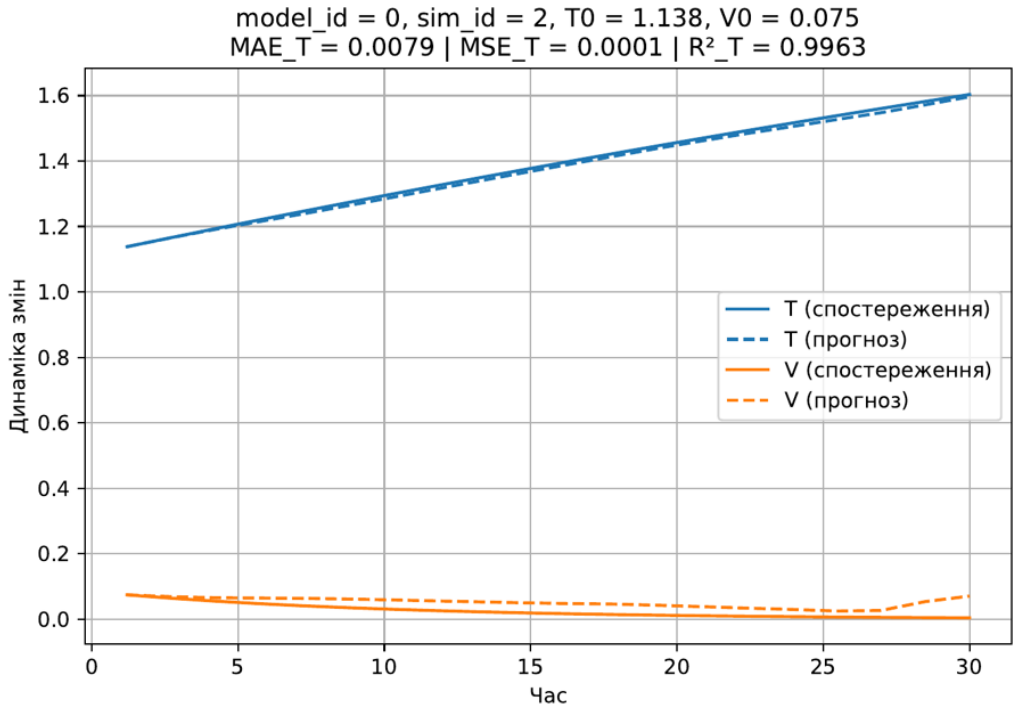


Рисунок 3.3 – Відтворення траєкторії моделі ВІЛ, приклад 3

Графіки для моделі GSW (Трава–Вівці–Вовки) наведені на рис. 3.4 – 3.6:

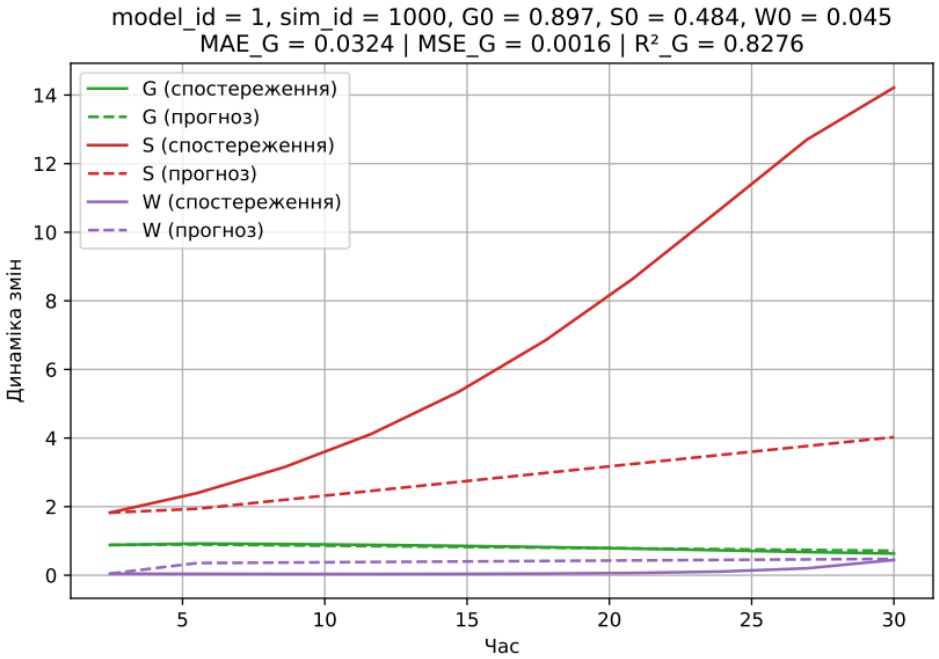


Рисунок 3.4 – Відтворення траєкторії моделі GSW, приклад 1

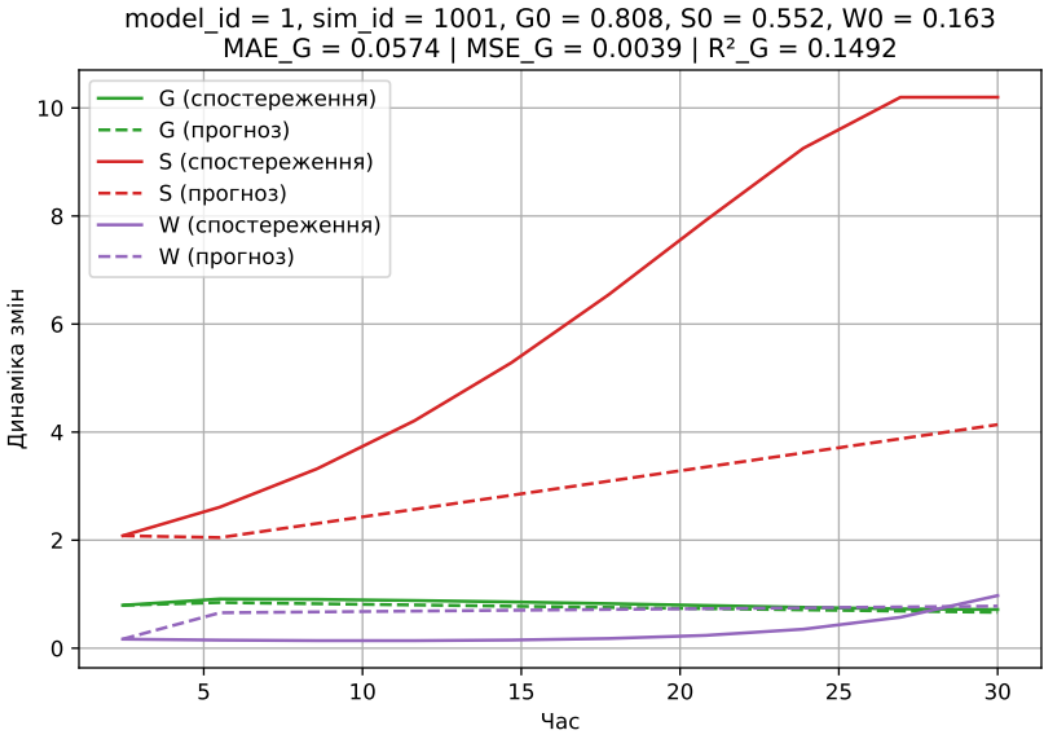


Рисунок 3.5 – Відтворення траєкторії моделі GSW, приклад 2

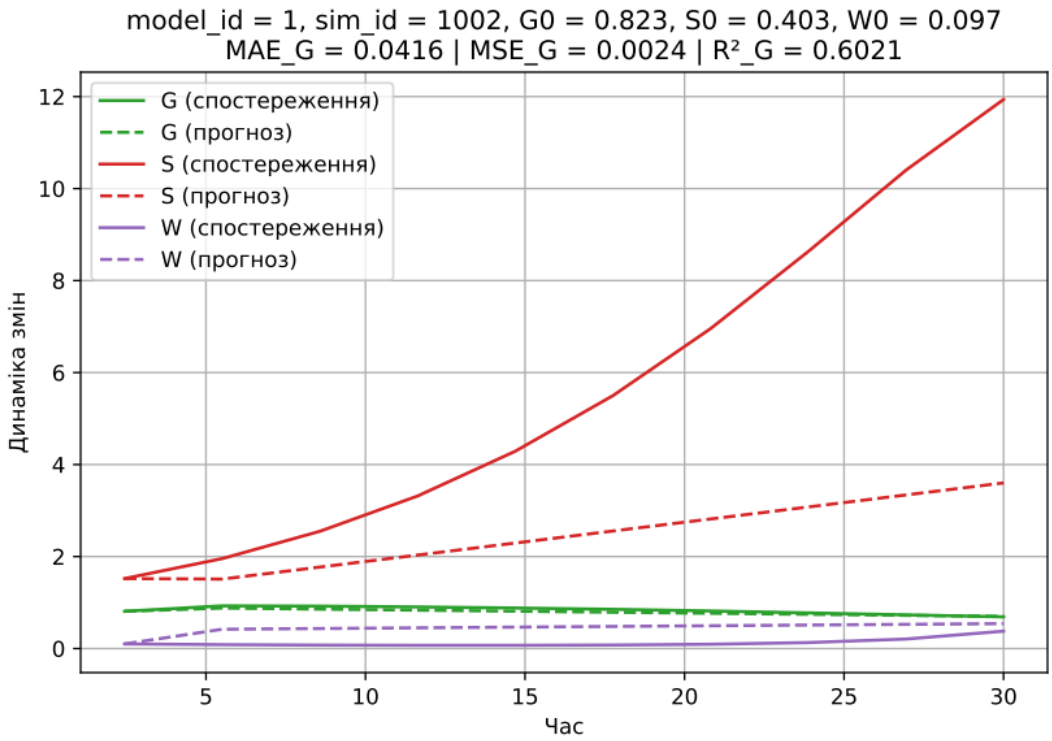


Рисунок 3.6 – Відтворення траєкторії моделі GSW, приклад 3

Аналіз графіків підтверджує, що диференціальна нейронна модель демонструє високу узгодженість з очікуваними траєкторіями для обох розглянутих систем — ВІЛ та GSW (Трава-Вівці-Вовки).

У моделі ВІЛ (рис. 3.1–3.3) спостерігається чітке відтворення поведінки CD4+ Т-клітин (T) у відповідь на динаміку вірусного навантаження (V). Модель з високою точністю прогнозує траєкторії, зберігаючи плавність, правильні тренди та характерний взаємозв'язок між змінними. Особливо це видно на прикладах із високим значенням коефіцієнта детермінації R^2, що наближається до 0.999. Це узгоджується з біологічною логікою розвитку інфекції, свідчить про ефективне навчання моделі та її здатність адаптуватися до різних початкових умов (T_0, V_0), точно відображаючи біологічно релевантні зміни.

У моделі GSW (рис. 3.4–3.6) відображено типові взаємодії між трьома компонентами екосистеми. Модель загалом зберігає тренди та фазову послідовність між популяціями: зростання біомаси трави (G) передує зростанню популяції вівць (S), а потім — вовків (W). При цьому, хоча значення

R^2 для деяких змінних є нижчими, у графіках зберігається узгодженість форм, амплітуд і взаємних затримок у динаміці. Це свідчить про здатність нейронної моделі відтворювати основні структурні закономірності складних нелінійних систем навіть у багатокомпонентному середовищі з варіативними початковими умовами.

Таким чином, візуалізація підтверджує як загальну точність передбачення, так і здатність моделі до генералізації. Модель Neural ODE виявилася придатною для роботи з різнорідними системами динаміки, від двокомпонентних біомедичних процесів до трикомпонентних екологічних моделей, забезпечуючи стабільне узгодження з розв’язками вихідних систем диференціальних рівнянь.

3.3 Оцінювання точності моделі на тестових траєкторіях

У цьому пункті представлено оцінку точності диференціальної нейронної моделі на тестових траєкторіях, які не використовувалися під час навчання. Аналіз проводиться на основі окремих симуляцій, що демонструють як найкращі, так і найгірші результати за обраними метриками та охоплюють різні сценарії динаміки. Також здійснено порівняння точності моделі на симуляціях, розташованих на початку та в кінці експериментального набору, з метою оцінити стабільність узагальнюючої здатності моделі. Це дало змогу комплексно охарактеризувати її поведінку при прогнозуванні траєкторій у випадках із варіативними початковими умовами.

Для оцінювання точності прогнозу нейронної моделі використовувалися такі метрики: MAE (середня абсолютна похибка), MSE (середньоквадратична похибка) та R^2 (коефіцієнт детермінації). Вони дозволяють охарактеризувати як абсолютну точність моделі (MAE, MSE), так і ступінь відповідності між прогнозованими та фактичними траєкторіями (R^2).

Під час аналізу результатів моделювання було відібрано симуляції з найкращими та найгіршими показниками точності, що дало змогу оцінити межі ефективності розробленої нейронної моделі (табл. 3.1). Такі приклади дозволяють наочно побачити, як модель поводиться в різних сценаріях — від найточніших до найскладніших для передбачення. В обох задачах — і для ВІЛ, і для екосистеми — можна прослідкувати, що якість прогнозу суттєво залежить від початкових умов.

Таблиця 3.1 – Метрики точності моделі для найкращих і найгірших симуляцій

model_id	sim_id	Змінна (оцінювана)	MAE	R^2	Якість / модель
0	0	T	0.0021	0.9997	Найкраща (ВІЛ)
0	7	T	0.0115	0.9904	Найгірша (ВІЛ)
1	1007	G	0.0159	0.8735	Найкраща (GSW)
1	1001	G	0.0574	0.1492	Найгірша (GSW)

У моделі ВІЛ найбільш вдалим виявився приклад із sim_id = 0, де MAE склала всього 0.0021, а R^2 досяг 0.9997. Це означає, що модель майже ідеально відтворила форму траєкторії. Навіть у найгіршому прикладі (sim_id = 7) точність залишалася дуже високою ($R^2 = 0.9904$), що свідчить про стабільність і добру узагальнювальну здатність мережі при зміні початкових значень.

У моделі GSW результати були більш варіативними. У симуляції з sim_id = 1007 модель добре спрогнозувала зміну біомаси трави (MAE = 0.0159, $R^2 = 0.8735$). Проте в іншому прикладі (sim_id = 1001) точність значно знизилась ($R^2 = 0.1492$), що, ймовірно, пов’язано зі складнішою взаємодією популяцій у цій конкретній комбінації початкових параметрів. Тим не менш, навіть у складних умовах модель частково зберігає послідовність змін, що говорить про її здатність передавати загальні закономірності динаміки.

Для перевірки стабільності моделі було відібрано кілька симуляцій із початку та кінця експериментального набору. Це дозволило оцінити,

наскільки точно модель узагальнює інформацію в умовах варіативних початкових значень (табл. 3.2).

Таблиця 3.2 – Метрики точності моделі для симуляцій з різних частин вибірки

model_id	sim_id	Змінна (оцінювана)	MAE	MSE	R^2	Модель / позиція
0	0	T	0.0021	0.0000	0.9997	ВІЛ_початок
0	1	T	0.0016	0.0000	0.9998	ВІЛ_початок
0	8	T	0.0037	0.0000	0.9990	ВІЛ_кінець
0	9	T	0.0073	0.0001	0.9964	ВІЛ_кінець
1	1000	G	0.0324	0.0016	0.8276	GSW_початок
1	1001	G	0.0574	0.0039	0.1492	GSW_початок
1	1008	G	0.0254	0.0009	0.8544	GSW_кінець
1	1009	G	0.0401	0.0023	0.6221	GSW_кінець

У моделі ВІЛ точність передбачення зберігається практично на однаковому високому рівні в усіх прикладах. Навіть у кінцевих симуляціях (sim_id = 8 і 9) значення коефіцієнта детермінації R^2 залишається близьким до 0.999, що свідчить про впевнене узагальнення моделі. MAE та MSE залишаються на низькому рівні, що підтверджує стабільність результатів незалежно від положення симуляції у вибірці.

У випадку моделі GSW спостерігається більший розкид. Для sim_id = 1001 (початок) точність значно нижча ($R^2 = 0.1492$), а MAE становить 0.0574. Водночас для деяких симуляцій наприкінці вибірки (наприклад, sim_id = 1008) модель демонструє покращення ($R^2 = 0.8544$), що вказує на можливу перевагу навчання саме на певних структурах початкових умов. Загалом, модель GSW залишається чутливішою до варіацій вхідних даних, однак зберігає адекватний рівень прогнозування для частини симуляцій.

3.4 Висновки до розділу 3

У розділі 3 було здійснено комплексну оцінку ефективності побудованої диференціальної нейронної мережі на основі аналізу її поведінки на тестових траєкторіях, які не входили до тренувальної вибірки. Аналіз охопив дві принципово різні математичні моделі: ВІЛ — біомедичну, з відносно стабільною динамікою, та GSW — екосистемну, зі складнішою структурою взаємодій. Це дозволило оцінити здатність мережі відтворювати поведінку різних моделей у рамках одного архітектурного підходу.

Для моделі ВІЛ результати виявились стабільними й високоточними: середні значення MAE не перевищують 0.01, а коефіцієнт детермінації R^2 у більшості випадків перевищує 0.99. Навіть у крайових симуляціях мережа демонструє здатність відтворювати ключові елементи динаміки із високим рівнем відповідності. Це пояснюється невеликою кількістю змінних і передбачуваним характером динаміки без виражених нелінійностей.

Натомість результати моделювання системи GSW виявились менш точними і більш варіативними. Це зумовлено складнішою природою динаміки моделі: складна екосистемна структура з трьома видами — рослинами, травоїдними та хижаками — призводить до циклічних змін популяцій, які важко точно передбачити. Також система може бути чутливою до початкових умов і потенційно жорсткою, що знижує стабільність чисельного інтегрування. Більша кількість змінних у GSW при однаковій структурі вибірки зменшує відносну інформативність кожного прикладу, що також впливає на якість узагальнення.

Попри це, навіть для складної динаміки GSW модель зберігає здатність до відтворення фазових співвідношень і загальних трендів, що свідчить про її структурну узагальнювальну здатність. Реалізація багатомодельного навчання через ознаку model_id виявилась ефективною: без зміни архітектури мережа опрацьовує як біомедичні, так і екологічні сценарії. Це підтверджує

перспективність підходу NeuralODE у задачах прогнозування динаміки взаємодіючих систем з різними характеристиками.

РОЗДІЛ 4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі розглядається функціонально-вартісний аналіз програмного продукту, який реалізує прогнозування популяційної динаміки за допомогою глибокої нейронної мережі диференціального типу (NeuralODE). Основною метою дослідження є створення системи, здатної точно моделювати зміну чисельності та взаємодії популяцій у складних біологічних або екологічних процесах. Такий інструмент може бути корисним не лише в наукових цілях, а й у прикладних задачах: епідеміології, моніторингу екосистем, аналізі довгострокових ризиків.

Для обґрунтування вибору оптимального варіанта реалізації було використано функціонально-вартісний підхід, який дозволяє оцінити важливість окремих компонентів системи з урахуванням їх обчислювальної складності та внеску у загальний результат. У ході аналізу розглядаються варіанти архітектури нейронної мережі, особливості підготовки симульованих даних, побудова навчальних і тестових вибірок, а також метрики, що використовуються для оцінювання точності прогнозування. Такий підхід дозволяє виявити, які компоненти системи є найбільш значущими з точки зору точності й стабільності результатів, і де можливе зменшення витрат без шкоди для ефективності.

Результати аналізу підтверджують, що ключову роль у досягненні високої точності відіграє правильна конфігурація мережі та спосіб її навчання. Особливо важливою виявилася здатність моделі адаптуватися до різних типів систем завдяки універсальній структурі та вхідним маркерам (model_id). Найбільш збалансованою з точки зору якості та витрат виявилася архітектура NeuralODE, яка показала стабільну роботу навіть на крайових симуляціях. Це підтверджує доцільність її подальшого застосування в задачах прогнозування складних динамічних систем.

4.1 Постановка задачі проєктування

Дане дослідження базується на використанні функціонально-вартісного аналізу (ФВА) для оцінювання ефективності розробленої програмної системи, яка моделює популяційну динаміку за допомогою диференціальної нейронної мережі (NeuralODE). Основна мета такого підходу — виявити, які компоненти системи мають найбільшу функціональну значущість і наскільки обґрунтованими є витрати на їх реалізацію з точки зору точності прогнозу, універсальності та стабільності.

Розроблений програмний продукт моделює два класи систем: біомедичну (ВІЛ-інфекція) та екологічну ("Трава–Вівці–Вовки"), що потребує універсального архітектурного рішення. Для цього в систему введено маркер model_id, який дозволяє навченій моделі адаптуватись до різних типів динаміки. Вхідні дані для тренування формуються шляхом симуляцій математичних моделей із наступною структуризацією у вигляді .csv-файлів. Це забезпечує узгодженість і повторюваність навчального процесу.

Технічні вимоги до програмного забезпечення сформульовані з урахуванням особливостей обчислень і поставленої задачі:

1) підтримка різних типів моделей — програмний продукт повинен ефективно обробляти симуляції з різною структурою динаміки без необхідності зміни архітектури;

2) можливість виконання на CPU — реалізація має бути придатною для запуску без використання графічних прискорювачів, що важливо в умовах навчального або обмеженого середовища;

3) точність прогнозу — система повинна забезпечувати прийнятний рівень середньої абсолютної похибки (MAE) та загалом високе значення коефіцієнта детермінації (R^2) на тестових траєкторіях, з урахуванням складності моделі, що апроксимується;

4) масштабованість — модель повинна дозволяти подальше розширення (наприклад, додавання нових типів моделей або генерацій даних) без необхідності принципової перебудови структури;

5) відтворюваність — важливим є дотримання чіткої структури процесу: симуляція → формування вибірки → навчання → оцінювання; такий підхід забезпечує надійність і контрольованість результатів.

Застосування функціонально-вартісного підходу дозволяє обґрунтовано оцінити, наскільки ефективно реалізовані функції програмного продукту відповідають вимогам до точності, адаптивності та економічності.

Аналіз результатів показав, що обрана архітектура забезпечує задовільне співвідношення між обчислювальними витратами та якістю прогнозування для різних типів популяційної динаміки. Це створює основу для подальшої оптимізації системи та підтверджує доцільність використання саме диференціальних нейронних мереж у задачах такого типу.

4.2 Обґрунтування функцій програмного продукту

Розроблений програмний продукт призначений для моделювання популяційної динаміки на основі глибокої нейронної мережі диференціального типу (NeuralODE). Його архітектура складається з окремих функціональних блоків, кожен із яких реалізує конкретний етап обробки даних — від генерації траєкторій до навчання та оцінювання моделі. З метою обґрунтованого вибору технічних рішень щодо ключових компонентів системи було застосовано функціонально-вартісний аналіз, що дозволяє співвіднести функціональну цінність підфункцій із витратами на їх реалізацію. У процесі проєктування було виокремлено такі функції

Функція F1 – Вибір середовища реалізації моделі

Включає такі варіанти:

– використання Python із бібліотекою PyTorch;

– реалізація моделі в середовищі C++ із застосуванням libtorch.

Функція F2 – Формат представлення симульованих даних

Включає такі варіанти:

– зберігання симульованих даних у табличному форматі CSV;

– використання гнучкого вкладеного формату JSON.

Функція F3 – Метод реалізації чисельного інтегратора

Включає такі варіанти:

– застосування стандартного інтегратора odeint з бібліотеки TorchDiffEq (метод RK4);

– реалізація власного чисельного інтегратора на основі RK4 або RKF45.

Варіанти реалізації функцій наведені у морфологічній карті (рис. 4.1).

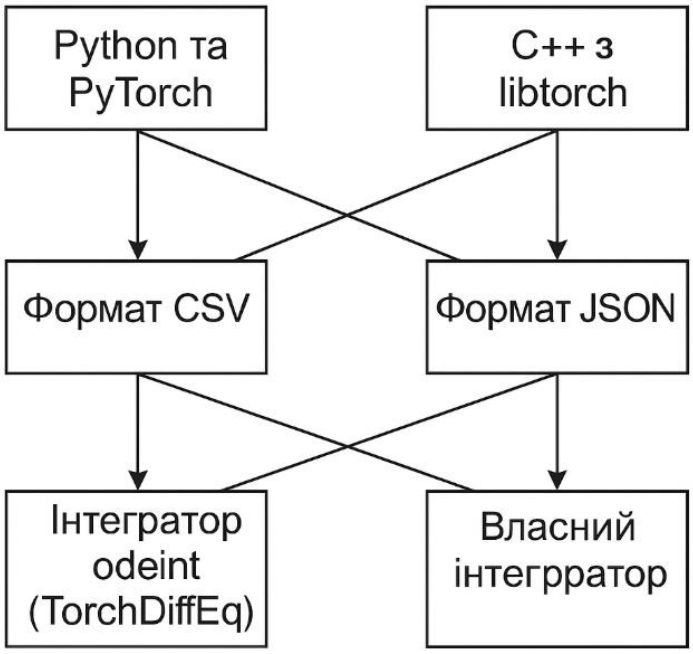


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину можливих комбінацій реалізації ключових функцій програмного продукту, призначеного для моделювання популяційної динаміки. Кожна комбінація представляє окремий

варіант архітектурного рішення, що охоплює вибір середовища реалізації, структуру вхідних даних та метод чисельного інтегрування. Такий підхід забезпечує системний аналіз потенційних рішень з урахуванням їх функціональної значущості, продуктивності та витрат на реалізацію.

Позитивно-негативна матриця (табл. 4.1), відображає переваги та обмеження кожного варіанта реалізації функцій, що допомагає обґрунтовано здійснити вибір оптимальної конфігурації системи.

Таблиця 4.1 – Позитивно-негативна матриця

Функція	Варіант	Переваги	Недоліки
F1	А Python	Швидкий старт, велика кількість бібліотек, інтеграція з TorchDiffEq	Залежність від інтерпретованого середовища
	Б C++	Вища продуктивність	Відсутність зручної підтримки автоматичного диференціювання
F2	А CSV	Простота, висока швидкодія, зручність у середовищі Python	Обмежена вкладеність
	Б JSON	Гнучкість структури	Складніша обробка, менша ефективність
F3	А odient	Перевірена реалізація, підтримка автоматичного диференціювання (autograd)	Менший контроль над точністю
	Б Власний інтегратор	Повний контроль, можливість оптимізації	Ускладнена підтримка, потреба в тестуванні

На основі аналізу позитивно-негативної матриці було встановлено, що деякі варіанти реалізації функцій не повною мірою відповідають поставленим вимогам, зокрема через обмеження у продуктивності, складність реалізації або відсутність необхідної підтримки. У результаті цього аналізу сформовано два найбільш доцільні сценарії реалізації програмного продукту.

Функція F1: обрано Python, оскільки це середовище забезпечує швидку розробку, широку підтримку бібліотек глибокого навчання та сумісність з інтегратором odeint. Варіант C++ відхилено через підвищену складність реалізації, зокрема відсутність зручної підтримки автодиференціювання.

Функція F2: обрано формат CSV як найзручніший варіант для зберігання симульованих даних, які мають фіксовану табличну структуру. Формат JSON, хоча й гнучкіший, вимагає додаткових ресурсів на обробку та не дає переваг у межах поставленої задачі.

Функція F3: розглядаються два варіанти. У першому випадку використовується готовий інтегратор odeint з бібліотеки TorchDiffEq, що забезпечує стабільність та автоматичне диференціювання. У другому — передбачається можливість реалізації власного інтегратора на основі RK4 або RKF45 у разі потреби підвищеного контролю над точністю або адаптації до специфічних умов.

Таким чином, розглядаємо наступні сценарії реалізації програмного продукту:

F1(А) – F2(А) – F3(А) – базовий варіант, реалізований у дипломному проекті.

F1(А) – F2(А) – F3(Б) – альтернативний варіант, що передбачає створення власного чисельного інтегратора з розширеними можливостями налаштування.

Перейдемо до детальнішого обґрунтування нашого вибору.

4.3 Вибір параметрів для програмного продукту

На підставі проведеного функціонально-вартісного аналізу реалізаційних сценаріїв та технічних рішень, визначено основні параметри вибору, що будуть використані для оцінювання технічного рівня програмного продукту. Такі параметри відображають характеристики швидкодії, ресурсного навантаження, якості реакції та масштабованості системи.

Для характеристики системи прогнозування популяційної динаміки, що реалізована на основі нейронної мережі диференціального типу (NeuralODE), доцільно використати такі параметри:

X1 – Швидкодія мови програмування

X2 – Об’єм пам’яті, необхідний для обробки даних

X3 – Час надання результатів

X4 – Потенційний об’єм програмного коду

Для кожного параметра задано умовні оцінки на трьох рівнях: гіршому, середньому та кращому, що дозволяє порівняти різні архітектурні рішення з урахуванням функціональної ефективності та витрат на реалізацію (табл. 4.2).

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Позначення	Одиниці виміру	Значення параметрів		
			Гірші	Середні	Кращі
Швидкодія мови програмування	X1	оп/мс	60	100	130
Об’єм пам’яті, необхідний для обробки даних	X2	мб	64	32	16
Час надання результатів	X3	мс	1200	500	200
Потенційний об’єм програмного коду	X4	кількість рядків коду	2500	2000	1500

Побудову графіків (рис. 4.2 – 4.5) виконано згідно з даними таблиці 4.2.

Рисунок 4.2 – X1, Швидкодія мови програмування

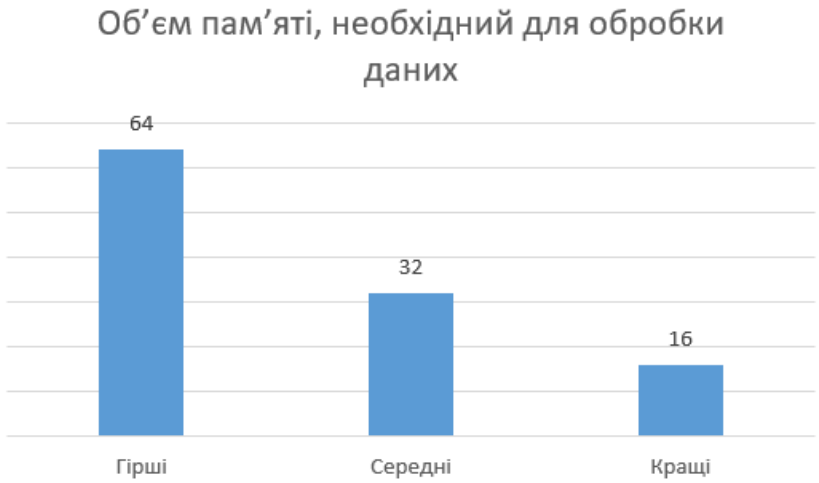


Рисунок 4.3 – X2, Об’єм пам’яті, необхідний для обробки даних

Рисунок 4.4 – X3, Час надання результатів

Рисунок 4.5 – X4, Потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Для визначення, які з технічних параметрів є найбільш важливими для ефективної роботи програмного продукту, було проведено експертне оцінювання. Його метою було з'ясувати, який із параметрів має найбільший вплив на якість реалізації системи у задачах прогнозування популяційної динаміки.

Оцінювання здійснювалось методом попарного порівняння, в якому брали участь 7 експертів, що мають досвід у прикладному програмуванні, моделюванні та роботі з даними. Кожному з них було запропоновано порівняти параметри X1–X4 (швидкодія, обсяг пам’яті, час отримання результату, обсяг коду) між собою та визначити, які з них є більш значущими.

Оцінювання важливості параметрів відбувалося у кілька етапів.

1. Встановлення рівня важливості параметра. Кожен експерт визначає, наскільки важливим є той чи інший параметр, та виставляє йому оцінку в порівнянні з іншими.
2. Перевірка надійності експертних оцінок. Перевіряється, чи є оцінки експертів узгодженими між собою, щоб їх можна було далі використовувати для розрахунків.
3. Попарне порівняння параметрів. Кожен параметр послідовно зіставляється з іншими, щоб встановити, який із них має більший вплив.
4. Аналіз результатів. На основі порівнянь обчислюються числові значення, які показують вагомість кожного параметра в загальній оцінці.

Оцінки, надані експертами, зведено у таблицю 4.3.

Таблиця 4.3 – Результати оцінки важливості параметрів за експертними оцінками

Назва параметра	Позначення	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи - лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
Швидкодія мови програмування	X1	оп/мс	4	4	4	4	3	3	3	25	7.5	56.25
Об’єм пам’яті, необхідний для обробки даних	X2	мб	2	2	2	1	1	1	2	11	-6.5	42.25
Час надання результатів	X3	мс	3	3	3	3	4	4	4	24	6.5	42.25
Потенційний об’єм програмного коду	X4	кількість рядків коду	1	1	1	2	2	2	1	10	-7.5	56.25
Разом			10	10	10	10	10	10	10	70	0	197

Для перевірки точності та надійності експертних оцінок використовується ряд розрахунків, що дозволяють оцінити, наскільки думки учасників оцінювання були узгодженими.

Розрахунок суми рангів для кожного параметра здійснюється за формулою:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{N \cdot n \cdot (n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів;

n – кількість параметрів.

Обчислення середньої суми рангів:

$$T = \frac{1}{n} \cdot R_{ij} = 17.5 \quad (4.2)$$

Визначення відхилень між ранговою сумою кожного параметра та середнім значенням:

$$\Delta_i = R_i - T \quad (4.3)$$

Сума всіх відхилень повинна дорівнювати нулю.

Обчислення суми квадратів відхилень:

$$S = \sum_{i=1}^{n} \Delta_i^2 = 197 \quad (4.4)$$

Коефіцієнт узгодженості визначається за формулою:

$$W = \frac{12 \cdot S}{N^2(n^3 - n)} = \frac{12 \cdot 197}{7^2(4^3 - 4)} = 0.804 > W_k = 0.67 \quad (4.5)$$

На основі отриманих оцінок виконаємо попарне зіставлення параметрів (табл. 4.4)

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1.5
X1 і X3	>	>	>	>	<	<	<	>	1.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	<	<	<	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Ступінь переваги одного з параметрів над іншим виражається числовим коефіцієнтом, який обчислюється за формулою:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

Для кожного з технічних параметрів обчислимо коефіцієнт вагомості $K_{\text{ві}}$ згідно з відповідними формулами.

$$\text{К}_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Обчислення відносних оцінок виконується кілька разів, доки різниця між послідовними результатами не стане незначною. Починаючи з другого кроку, використовуються наступні формули:

$$К_{ві} = \frac{b_i'}{\sum_{i=1}^{n} b_i}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Як показано в таблиці 4.5, відмінності між коефіцієнтами вагомості становлять менше 2%, що свідчить про досягнення стабільних значень, і подальші обчислення не потребують повторення.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_i				Перша ітерація		Друга ітерація		Третя ітерація	
	X1	X2	X3	X4	b_i	$К_{ві}$	b_i^1	$К_{ві}^1$	b_i^2	$К_{ві}^2$
X1	1	1.5	1.5	1.5	5.5	0.34	21.25	0.36	77.875	0.36
X2	0.5	1	0.5	1.5	3.5	0.22	12.25	0.21	44.875	0.21
X3	0.5	1.5	1	1.5	4.5	0.28	16.25	0.27	59.125	0.27
X4	0.5	0.5	0.5	1	2.5	0.16	9.25	0.16	34.125	0.16
Всього:					16	1	59	1	213	1

4.5 Аналіз якості реалізації варіантів функцій

Для оцінювання якості реалізації окремих функцій програмного забезпечення було проаналізовано абсолютні значення ключових технічних параметрів. До переліку таких параметрів належать: обсяг пам’яті, необхідний для обробки даних (X2), час надання результатів (X3), потенційний обсяг програмного коду (X4), а також швидкодія мови програмування (X1). Кожен із параметрів повинен відповідати вимогам технічної ефективності функціонування системи моделювання.

Щоб здійснити кількісне оцінювання варіантів реалізації, було використано показник технічного рівня, який розраховується із врахуванням

вагомості кожного параметра. Формально цей показник визначається за такою залежністю:

$$K_K(j) = \sum_{i=1}^{n} K_{Bi,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

K_{Bi} – коефіцієнт вагомості і–го параметра;

B_i – оцінка і–го параметра в балах.

На основі виразу (4.11), кожне значення показника якості визначається множенням вагового коефіцієнта параметра K_{Bi} на відповідну оцінку цього параметра в балах B_i. Узагальнені результати обчислень подано в таблиці 4.6, що демонструє рівень якості різних варіантів реалізації основних функціональних елементів програмного забезпечення.

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}] \quad (4.12)$$

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	130	24	0.36	8.64
F2	А	X2	31	11	0.21	2.31
F3	А	X3	450	18	0.27	4.86
	Б	X4	1900	7	0.16	1.12

Після цього визначимо рівень якості кожного з варіантів:

$$K_{K1} = 8.64 + 2.31 + 4.86 = 15.81$$

$$K_{K2} = 8.64 + 2.31 + 1.12 = 12.07$$

З розрахунків видно, що кращим є 1 варіант, оскільки його коефіцієнт технічного значення має більше значення.

4.6 Економічний аналіз розробки

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості. Усі варіанти включають у себе два окремих завдання:

– Розробка проєкту програмного продукту;

– Розробка програмної оболонки.

Перше завдання за ступенем новизни відноситься до групи А, друге завдання — до групи Б. За складністю алгоритмів, які використовуються в завданні 1, належать до групи 1; а в завданні 2 — до групи 3.

Загальна трудомісткість обчислюється як:

$$T_0 = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (4.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого з етапів, який передбачає проектування архітектури програмного забезпечення, враховуючи характер задачі (розрахункова постановка, ступінь новизни – А, алгоритми першої групи складності), нормативна трудомісткість приймається на рівні $T_Р$= 60 люд.-днів. З урахуванням використання нормативно-довідкової інформації, застосовується поправочний коефіцієнт $К_П$= 1.2. Оскільки у процесі реалізації активно використовуються стандартні бібліотеки та модулі, зокрема PyTorch і TorchDiffEq, коефіцієнт використання готових компонентів приймається рівним $К_{СТ}$= 0.8. При цьому складність вхідної інформації не перевищує

базовий рівень, тому $К_{СК}$= 1. У результаті загальна трудомісткість програмування першого завдання розраховується як:

$$Т_1 = 60 \cdot 1.2 \cdot 0.8 = 57.6 \text{ людино-днів.}$$

Аналогічно проведемо розрахунок трудомісткості для другого етапу, що охоплює реалізацію нейронної моделі. Цей етап відноситься до задач з алгоритмами третьої групи складності, а ступінь новизни оцінюється як Б, що відповідає нормативному часу $Т_Р$= 45 люд.-днів. Через складність структури симульованих даних і необхідність обробки ознак на рівні model_id, використовується поправочний коефіцієнт $К_П$= 1.4. Вхідна інформація не вимагає спеціальних трансформацій, тому коефіцієнт складності інформації приймається як $К_{СК}$= 1, а з огляду на часткове використання стандартних модулів, коефіцієнт їх застосування становить $К_{СТ}$= 0.9. У підсумку трудомісткість реалізації обчислюється за формулою:

$$Т_2 = 45 \cdot 1.4 \cdot 0.9 = 56.7 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб визначити загальні витрати праці:

$$Т_I = (57.6 + 56.7 + 15.81) \cdot 7 = 910.8 \text{ людино-годин.}$$

$$Т_{II} = (57.6 + 56.7 + 12.07) \cdot 7 = 884.6 \text{ людино-годин.}$$

У реалізації програмного продукту були задіяні два фахівці: програміст із місячним окладом 27000 грн та аналітик-експерт у галузі обробки даних, заробітна плата якого становить 26000 грн. Обчислимо середню зарплату за годину:

$$С_Ч = \frac{М}{Т_m \cdot t} \text{ грн.,} \qquad (4.14)$$

де М – місячний оклад працівників;

$Т_m$ – кількість робочих днів;

t – кількість робочих годин в день.

$$С_Ч = \frac{27000 + 26000}{2 \cdot 22 \cdot 7} = 172.1 \text{ грн.}$$

Розрахуємо заробітну плату:

$$С_{ЗП} = С_{ч} \cdot Т_{i} \cdot К_{Д} \quad (4.15)$$

де $С_{ч}$ – величина середньої заробітної плати за годину,

$Т_{i}$ – трудомісткість відповідного завдання,

$К_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата фахівців за варіантами становить:

$$С_{ЗП1} = 172.1 \cdot 910.8 \cdot 1.1 = 172423.6 \text{ грн.}$$

$$С_{ЗП2} = 172.1 \cdot 884.6 \cdot 1.1 = 167463.6 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$С_{ВІД1} = С_{ЗП1} \cdot 0.22 = 172423.6 \cdot 0.22 = 37933.2 \text{ грн.}$$

$$С_{ВІД2} = С_{ЗП2} \cdot 0.22 = 167463.6 \cdot 0.22 = 36842.0 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (СМ)

Так як одна ЕОМ обслуговує одного програміста з окладом 27000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$С_{Г} = 12 \cdot М \cdot К_{З} = 12 \cdot 27000 \cdot 0.2 = 64800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_{Г} \cdot (1 + К_{З}) = 64800 \cdot (1 + 0.2) = 77760 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 77760 \cdot 0.22 = 17107.2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25000 грн.:

$$С_{А} = К_{ТМ} \cdot К_{А} \cdot Ц_{ПР} = 1.3 \cdot 0.25 \cdot 25000 = 8125 \text{ грн.}$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу в користувача;

$К_{А}$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Розрахуємо витрати на ремонт та профілактику:

$$С_{Р} = К_{ТМ} \cdot Ц_{ПР} \cdot К_{Р} = 1.3 \cdot 25000 \cdot 0.06 = 1950 \text{ грн.}$$

де $К_{Р}$ – відсоток витрат на поточні ремонти.

Надалі розрахуємо ефективний годинний фонд часу ПК за рік:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = (365 - 104 - 11 - 12) \cdot 7 \cdot 0.75 = 1250 \text{ годин}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Наступним етапом розрахуємо витрати на оплату електроенергії:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_C \cdot К_З \cdot Ц_{ЕН} = 1250 \cdot 0.22 \cdot 0.75 \cdot 9.43 = 1945 \text{ грн.}$$

де N_C – середньо-споживча потужність приладу;

$К_З$ – коефіцієнт зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Після цього розрахуємо накладні витрати:

$$С_Н = Ц_{ПР} \cdot 0.67 = 25000 \cdot 0.67 = 16750 \text{ грн.}$$

Тепер, річні експлуатаційні витрати будуть обчислюватись наступним чином:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.16)$$

$$С_{ЕКС} = 77760 + 17107.2 + 8125 + 1950 + 1945 + 16750 = 123637.2 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{С_{ЕКС}}{Т_{ЕФ}} = \frac{123637.2}{1250} = 98.91 \text{ грн./год.}$$

Оскільки виконання всіх етапів розробки програмного забезпечення здійснюється з використанням електронно-обчислювальної машини (ЕОМ), витрати на її експлуатацію визначаються за наступною формулою:

$$С_М = С_{М-Г} \cdot Т \quad (4.17)$$

$$С_{М1} = 98.91 \cdot 910.8 = 90087.23 \text{ грн.}$$

$$C_{M2} = 98.91 \cdot 884.6 = 87495.79 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати і розраховуються за формулою:

$$C_{Н} = C_{ЗП} \cdot 0.67 \quad (4.18)$$

$$C_{Н1} = 172423.6 \cdot 0.67 = 115523.81 \text{ грн.}$$
$$C_{Н2} = 167463.6 \cdot 0.67 = 112200.61 \text{ грн.}$$

Наприкінці, вартість розробки ПП обчислюється за формулою:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_{М} + C_{Н} \quad (4.19)$$

$$C_{ПП1} = 172423.6 + 37933.2 + 90087.23 + 115523.81 =$$
$$= 415967.84 \text{ грн.}$$
$$C_{ПП2} = 167463.6 + 36842.0 + 87495.79 + 112200.61 =$$
$$= 404002.0 \text{ грн.}$$

4.7 Вибір оптимального варіанта реалізації ПП

Розрахуємо коефіцієнт техніко-економічного рівня:

$$К_{ТЕР j} = \frac{К_{К j}}{C_{Ф j}} \quad (4.20)$$

$$К_{ТЕР1} = \frac{15.81}{415202.77} = 3.8078 \cdot 10^{-5}$$
$$К_{ТЕР2} = \frac{12.07}{403258.93} = 2.9931 \cdot 10^{-5}$$

Перший варіант має вищий коефіцієнт техніко-економічного рівня $К_{ТЕР1} = 3.8078 \cdot 10^{-5}$, що свідчить про його більшу ефективність у співвідношенні до вартості реалізації.

Зіставлення технічних характеристик і вартості впровадження засвідчує перевагу першого варіанта, який демонструє вищий коефіцієнт техніко-

економічного рівня. Відповідно до результатів аналізу, цей варіант доцільно вважати найбільш збалансованим рішенням серед розглянутих альтернатив. Він забезпечує оптимальне поєднання функціональних можливостей і витрат на реалізацію.

У цьому варіанті реалізації були використані такі рішення:

– вибір мови програмування – Python;

– формат збереження даних — CSV;

– математичне ядро — вбудовані інтегратори бібліотеки TorchDiffEq.

Обрана реалізація забезпечує користувачу ефективну обробку симульованих даних, стабільну роботу моделі у реальному часі та зручний спосіб інтеграції результатів в аналітичні процеси

4.8 Висновки до розділу 4

У межах даного розділу було здійснено комплексний функціонально-вартісний аналіз програмного продукту, призначеного для прогнозування популяційної динаміки на основі глибокої нейронної мережі диференціального типу. Проведено ідентифікацію ключових функцій системи та визначено параметри, що мають найбільший вплив на її ефективність.

Для кожної з функцій було розглянуто альтернативні варіанти реалізації, після чого побудовано морфологічну карту та виконано експертне оцінювання значущості технічних параметрів. На основі попарного порівняння параметрів визначено їх вагомість, що дозволило обґрунтувати вибір оптимальної конфігурації системи.

Проведено аналіз економічних витрат на розробку програмного продукту, враховано затрати машинного часу та заробітну плату виконавців.

На основі інтегрального техніко-економічного показника визначено доцільний варіант реалізації.

Обраний варіант — використання мови Python, формату CSV для збереження даних та вбудованого інтегратора бібліотеки TorchDiffEq — дозволяє досягти стабільної роботи моделі, ефективної обробки симульованих даних і забезпечує прийнятний рівень витрат на реалізацію.

ВИСНОВКИ

У дипломній роботі розроблено програмний підхід до моделювання популяційної динаміки з використанням глибоких нейронних мереж диференціального типу. Метою дослідження було створення універсального інтегратора для відтворення поведінкових траєкторій динамічних систем, що описуються звичайними диференціальними рівняннями.

На основі аналізу сучасних методів чисельного інтегрування та нейромережевих підходів було реалізовано архітектуру Neural ODE у середовищі Python з використанням бібліотек PyTorch та TorchDiffEq. Для навчання використовувалися симульовані дані з математичних моделей ВІЛ та GSW. Проведено порівняння з класичними архітектурами (FNN, LSTM), за результатами якого Neural ODE забезпечив найкращий баланс між точністю та узагальнюваністю. Виконано порівняльний та економічний аналіз варіантів реалізації, що підтвердив доцільність обраної архітектури.

Подальший розвиток роботи доцільно зосередити насамперед на узагальненні методу до задач із просторовими змінними, що відкриває шлях до моделювання складніших систем на основі частинних диференціальних рівнянь. Перспективними залишаються також впровадження шумостійкого навчання, AutoML та розробка графічного інтерфейсу користувача. Крім того, покращення якості моделі можливе за рахунок оптимізації архітектури нейронної мережі (кількість шарів, нейронів, епох), а також використання більш потужних обчислювальних ресурсів для прискорення навчання та підвищення точності прогнозів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chen R. T. Q., Rubanova Y., Bettencourt J., Duvenaud D. Neural Ordinary Differential Equations. Advances in Neural Information Processing Systems, 2018, Vol. 31. URL: https://papers.nips.cc/paper/7892-neural-ordinary-differential-equations (дата звернення: 02.05.2025).

2. Iserles A. First Course in the Numerical Analysis of Differential Equations. 2nd ed. Cambridge: Cambridge University Press, 2022. 460 p.

3. Фельдман Л.П., Петренко А.І., Дмитрієва О.А. Чисельні методи в інформатиці. Київ : Видавнича група BHV, 2006. 432 с.

4. Butcher J.C. Numerical Methods for Ordinary Differential Equations. 2nd ed. Chichester : John Wiley & Sons, 2016. 510 p.

5. Hairer E., Wanner G. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. 2nd ed. Berlin : Springer, 2010. 614 p.

6. Kloberdanz E., Le W. S-SOLVER: Numerically Stable Adaptive Step Size Solver for Neural ODEs. 2023. URL: https://dl.acm.org/doi/10.1007/978-3-031-44201-8_32 (дата звернення: 03.05.2025).

7. Kabanikhin S. I., Krivorotko O. I., Ermolenko D. V., Kashtanova V. N., Latyshenko V. A. Inverse Problems of Immunology and Epidemiology. Eurasian Journal of Mathematical and Computer Applications. 2023, Vol. 11. URL: https://ejmca.org/article/view/6585 (дата звернення: 30.04.2025).

8. Garfinkel A., Shevtsov J., Guo Y. Modeling Life: The Mathematics of Biological Systems. Cham : Springer, 2017. 382 p.

9. Samarskii A.A., Mikhailov A.P. Principles of Mathematical Modelling: Ideas, Methods, Examples. London: Taylor & Francis, 2002. 360 p.

10. Zhang Y., Dong Q. Unveiling LLM Mechanisms Through Neural ODEs and Control Theory. 2024. URL: https://arxiv.org/abs/2406.16985 (дата звернення: 01.05.2025).

11. Волонтир Л.О., Зелінська О.В., Потапова Н.А. Чисельні методи : навч. посіб. Вінниця : ВНАУ, 2020. 256 с.

12. Forrester A., Sobester A., Keane A. Engineering Design via Surrogate Modelling: A Practical Guide. Chichester : John Wiley & Sons, 2008. 400 p.

13. Walter E. Numerical Methods and Optimization: A Consumer Guide. Cham : Springer, 2016. 270 p.

14. Савченко І. О. Методологічне і математичне забезпечення розв’язання задач передбачення на основі модифікованого методу морфологічного аналізу. Системні дослідження та інформаційні технології, 2011. № 3. С. 18–28.

15. Fronk C., Petzold L. Training Stiff Neural Ordinary Differential Equations with Implicit Single-Step Methods. 2024. URL: https://arxiv.org/abs/2410.05592 (дата звернення: 08.05.2025).

16. A Model of HIV Immunology within an Individual Person. URL: https://modelinginbiology.github.io/A-Model-of-HIV-immunology-within-an-Individual-Person-interactive-simulations (дата звернення: 09.05.2025).

17. Grass–Sheep–Wolves Interactive Simulations. URL: https://modelinginbiology.github.io/Grass-Sheep-Wolves-interactive-simulations (дата звернення: 09.05.2025).

ДОДАТОК А ЛІСТИНГ КОДУ

neural_ode.py

```
import torch
import torch.nn as nn

class NeuralODEModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.architecture =
nn.Sequential(
            nn.Linear(6, 256),
            nn.ReLU(),
            nn.Linear(256, 256),
            nn.Tanh(),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, 5)  # Виходи
до T, V, G, S, W
        )

    def forward(self, t, state):
        features =
self.architecture(state)
        model_type =
state[5].unsqueeze(0)
        return torch.cat([features,
model_type], dim=0)
```

simulate_hiv.py

```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import os

# Параметри моделі ВІЛ
params_hiv = {
    's': 0.03,
    'd': 0.01,
    'k': 0.001,
    'c': 0.1
}

num_runs = 10
time_range = (0, 30)
time_grid  =  np.linspace(*time_range,
100)
data_frames = []

for idx in range(num_runs):
    T_initial  =  np.random.uniform(0.8,
1.2)
    V_initial = np.random.uniform(0.05,
0.2)  # Тепер варіативне V0
    initial_conditions  =  [T_initial,
V_initial]

    def hiv_dynamics(t, y):
        T, V = y
        dT    =    params_hiv['s']    -
params_hiv['d'] * T - params_hiv['k'] *
T * V
        dV = params_hiv['k'] * T * V -
params_hiv['c'] * V
        return [dT, dV]

    result  =  solve_ivp(hiv_dynamics,
time_range,         initial_conditions,
t_eval=time_grid, method='RK45')

    sim_df = pd.DataFrame({
        'sim_id': idx,
        't': result.t,
        'T': result.y[0],
        'V': result.y[1],
        'G': 0,
        'S': 0,
        'W': 0,
        'model_id': 0
    })

    data_frames.append(sim_df)

# Об’єднання
hiv_combined  =  pd.concat(data_frames,
ignore_index=True)

#  Збереження  в  multitrajectory.csv,
замінюючи старі ВІЛ-дані
os.makedirs("data", exist_ok=True)
path = "data/multitrajectory.csv"

if os.path.exists(path):
    existing_df = pd.read_csv(path)
```

```
    existing_df                         =
existing_df[existing_df["model_id"]  !=
0]  # залишаємо тільки не-ВІЛ
    final_df  = pd.concat([existing_df,
hiv_combined], ignore_index=True)
else:
    final_df = hiv_combined


final_df.to_csv(path, index=False)
print("ВІЛ-дані       збережено       у
multitrajectory.csv з варіативним V0")
```

simulate_grass_sheep_wolves.py

```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import os

params = {
    'a': 1.0,
    'b': 0.1,
    'c': 0.1,
    'd': 0.1,
    'e': 0.1,
    'f': 0.1
}

# Основні параметри
main_runs = 100
extra_runs = 5  # крайові траєкторії
time_range = (0, 30)
time_grid = np.linspace(*time_range,
50)  # зменшено з 100 до 50
data_frames = []

# 100 випадкових симуляцій
for i in range(1000, 1000 + main_runs):
    G_init = np.random.uniform(0.3,
0.7)
    S_init = np.random.uniform(0.1,
0.5)
    W_init = np.random.uniform(0.05,
0.2)
    initial_conditions = [G_init,
S_init, W_init]

    def gsw_model(t, y):
        G, S, W = y
        dG = params["a"] * G * (1 - G)
- params["b"] * G * S
        dS = params["c"] * G * S -
params["d"] * S * W
        dW = params["e"] * S * W -
params["f"] * W
        return [dG, dS, dW]

    result = solve_ivp(gsw_model,
time_range, initial_conditions,
t_eval=time_grid, method='RK45')

    sim_df = pd.DataFrame({
        'sim_id': i,
        't': result.t,
        'T': 0,
        'V': 0,
        'G': result.y[0],
        'S': result.y[1],
        'W': result.y[2],
        'model_id': 1
    })
    data_frames.append(sim_df)

# 5 додаткових крайових симуляцій з
фіксованими S0 і W0
for j in range(2000, 2000 +
extra_runs):
    G_init = np.random.uniform(0.3,
0.7)
    S_init = 0.1
    W_init = 0.05
    initial_conditions = [G_init,
S_init, W_init]

    result = solve_ivp(gsw_model,
time_range, initial_conditions,
t_eval=time_grid, method='RK45')

    sim_df = pd.DataFrame({
        'sim_id': j,
        't': result.t,
        'T': 0,
        'V': 0,
        'G': result.y[0],
        'S': result.y[1],
        'W': result.y[2],
        'model_id': 1
    })
    data_frames.append(sim_df)

# Об'єднання всіх нових GSW-симуляцій
gsw_combined = pd.concat(data_frames,
ignore_index=True)

# Додавання до існуючого файлу
(залишаємо ВІЛ)
os.makedirs("data", exist_ok=True)
path = "data/multitrajectory.csv"

if os.path.exists(path):
    existing_df = pd.read_csv(path)
    existing_df =
existing_df[existing_df["model_id"] !=
1]
    final_df = pd.concat([existing_df,
gsw_combined], ignore_index=True)
else:
    final_df = gsw_combined

final_df.to_csv(path, index=False)
print("GSW-симуляції оновлено у
multitrajectory.csv (випадкові +
крайові)")
```

normalize_utils.py

```
import pandas as pd
import json
import os

# Шлях до CSV-файлу
csv_path = "data/multitrajectory.csv"
assert os.path.exists(csv_path), f"Файл
не знайдено: {csv_path}"

# Зчитування CSV
df = pd.read_csv(csv_path)
```

```
# Змінні для нормалізації
variables = ["T", "V", "G", "S", "W"]

# Обчислення min та max
norm_stats = {
    var: [df[var].min(), df[var].max()]
    for var in variables
}

# Шлях до збереження JSON
json_path = "data/norm_stats.json"
with open(json_path, "w") as f:
    json.dump(norm_stats, f, indent=4)

print(f"Нормалізаційні межі збережено в
{json_path}")
for var, (vmin, vmax) in
norm_stats.items():
    print(f"{var}: min = {vmin:.4f},
max = {vmax:.4f}")
```

train.py

```
import os
import torch
import pandas as pd
import torch.nn as nn
import json
import numpy as np
from torchdiffeq import odeint
from neural_ode import NeuralODEModel

with open("data/norm_stats.json") as f:
    norm_stats = json.load(f)

def normalize(x, var):
    min_val, max_val = norm_stats[var]
    return (x - min_val) / (max_val -
min_val)

learning_rate = 1e-3
num_epochs = 500
output_dir = "outputs"
os.makedirs(output_dir, exist_ok=True)

data =
pd.read_csv("data/multitrajectory.csv")

hiv_ids = data[data["model_id"] ==
0]["sim_id"].unique()
gsw_ids = data[data["model_id"] ==
1]["sim_id"].unique()

np.random.seed(42)
gsw_ids = np.random.choice(gsw_ids,
size=30, replace=False)

selected_ids = list(hiv_ids[:10]) +
list(gsw_ids)

net = NeuralODEModel()
optimizer =
torch.optim.Adam(net.parameters(),
lr=learning_rate)
criterion = nn.MSELoss()

for epoch in range(num_epochs):
    cumulative_loss = 0.0

    for sim_id in selected_ids:
        entry = data[data["sim_id"] ==
sim_id]
        full_time = entry["t"].values
        y_values = entry[["T", "V",
"G", "S",
"W"]].values.astype('float32')
        model_type =
entry["model_id"].iloc[0]

        if model_type == 1:
            for var_idx, var in
enumerate(["T", "V", "G", "S", "W"]):
                y_values[:, var_idx] =
normalize(y_values[:, var_idx], var)

        train_mask = [i % 5 != 4 for i
in range(len(full_time))]
        t_train =
torch.tensor(full_time,
dtype=torch.float32)[train_mask]
        y_train =
torch.tensor(y_values,
dtype=torch.float32)[train_mask]

        model_tag =
torch.full((y_train.shape[0], 1),
float(model_type), dtype=torch.float32)
        full_input =
torch.cat([y_train, model_tag], dim=1)
        initial = full_input[0]

        y_pred = odeint(net, initial,
t_train, method='rk4')

        if model_type == 0:
            loss = criterion(y_pred[:,
0:2], y_train[:, 0:2])
        elif model_type == 1:
            loss = 2.0 *
criterion(y_pred[:, 2:5], y_train[:,
2:5])
        else:
            continue

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        cumulative_loss += loss.item()

    if epoch % 10 == 0:
        avg_loss = cumulative_loss /
len(selected_ids)
        print(f"Епоха {epoch}: середнє
MSE = {avg_loss:.6f}")

torch.save(net.state_dict(),
os.path.join(output_dir,
"neural_ode_multimodel.pt"))
print("Модель збережено")
```

evaluate_metrics_pdf.py

```
import os
import torch
import pandas as pd
import matplotlib.pyplot as plt
import torch.nn.functional as F
from torchdiffeq import odeint
from sklearn.metrics import
mean_absolute_error, r2_score
```

```
from matplotlib.backends.backend_pdf
import PdfPages
from neural_ode import NeuralODEModel
import json

# Завантаження нормалізаційних меж
with open("data/norm_stats.json") as f:
    norm_stats = json.load(f)

def denormalize(x, var):
    min_val, max_val = norm_stats[var]
    return x * (max_val - min_val) +
min_val

# Завантаження даних і моделі
data_table =
pd.read_csv("data/multitrajectory.csv")
ode_net = NeuralODEModel()
ode_net.load_state_dict(torch.load("out
puts/neural_ode_multimodel.pt"))
ode_net.eval()

os.makedirs("outputs", exist_ok=True)

# Індекси змінних у повному init_state
= [T, V, G, S, W, model_id]
global_idx_map = {"T": 0, "V": 1, "G":
2, "S": 3, "W": 4}

# Кольори
palette_map = {
    "T": "#1f77b4",
    "V": "#ff7f0e",
    "G": "#2ca02c",
    "S": "#d62728",
    "W": "#9467bd"
}

# Оцінка для кожної моделі
unique_models =
sorted(data_table["model_id"].unique())

for m_id in unique_models:
    model_data =
data_table[data_table["model_id"] ==
m_id]

    if m_id == 0:
        sim_list =
sorted(model_data["sim_id"].unique())[:
10]
        variables = ["T", "V"]
    elif m_id == 1:
        sim_list =
sorted(model_data["sim_id"].unique())[:
10]
        variables = ["G", "S", "W"]
    else:
        continue

    with
PdfPages(f"outputs/predict_model_{m_id}
_best_by_metric.pdf") as pdf_out:
        for s_id in sim_list:
            single_run =
model_data[model_data["sim_id"] ==
s_id]
            t_seq_all =
torch.tensor(single_run["t"].values,
dtype=torch.float32)
            real_vals_all =
torch.tensor(single_run[["T", "V", "G",
"S", "W"]].values, dtype=torch.float32)

            test_mask = [i % 5 == 4 for
i in range(len(t_seq_all))]
            t_test =
t_seq_all[test_mask]
            y_test =
real_vals_all[test_mask]

            init_state =
torch.cat([y_test[0],
torch.tensor([m_id],
dtype=torch.float32)])

            with torch.no_grad():
                forecast =
odeint(ode_net, init_state, t_test,
method="rk4")

            # Індекси для змінних цієї
моделі
            var_indices =
[global_idx_map[v] for v in variables]
            y_real = y_test[:,
var_indices]
            y_model = forecast[:,
var_indices]

            # Денормалізація GSW
            if m_id == 1:
                for idx, var in
enumerate(variables):
                    y_model[:, idx] =
denormalize(y_model[:, idx], var)
                    y_real[:, idx] =
denormalize(y_real[:, idx], var)

            # Метрики для кожної
змінної
            metrics = {}
            for idx, var in
enumerate(variables):
                real_np = y_real[:,
idx].numpy()
                pred_np = y_model[:,
idx].detach().numpy()
                mae =
mean_absolute_error(real_np, pred_np)
                mse =
F.mse_loss(y_model[:, idx], y_real[:,
idx]).item()
                r2 = r2_score(real_np,
pred_np)
                metrics[var] = {"MAE":
mae, "MSE": mse, "R2": r2}

            # Найкраща змінна по кожній
метриці
            best_mae_var = min(metrics,
key=lambda v: metrics[v]["MAE"])
            best_mse_var = min(metrics,
key=lambda v: metrics[v]["MSE"])
            best_r2_var  = max(metrics,
key=lambda v: metrics[v]["R2"])

            # Початкові значення
(реальні індекси з повного init_state)
            init_values = ", ".join([
```

```
                f"{var}0 =
{init_state[global_idx_map[var]].item()
:.3f}"
                for var in variables
            ])

            # Побудова графіка
            plt.figure(figsize=(8, 5))
            for idx, var in
enumerate(variables):
                color =
palette_map[var]
                plt.plot(t_test,
y_real[:, idx], label=f"{var}
(спостереження)", linestyle="-",
color=color)
                plt.plot(t_test,
y_model[:, idx].detach(), label=f"{var}
(прогноз)", linestyle="--",
color=color)

            title = (
                f"model_id = {m_id},
sim_id = {s_id}, {init_values}\n"
                f"MAE_{best_mae_var} =
{metrics[best_mae_var]['MAE']:.4f} | "
                f"MSE_{best_mse_var} =
{metrics[best_mse_var]['MSE']:.4f} | "
                f"R²_{best_r2_var} =
{metrics[best_r2_var]['R2']:.4f}"
            )

            plt.title(title)
            plt.xlabel("Час")
            plt.ylabel("Динаміка змін")
            plt.legend()
            plt.grid(True)
            pdf_out.savefig()
            plt.close()

print("Оновлені графіки збережено з
реальними початковими значеннями та
палітрою.")
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Дипломна робота

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

ДИПЛОМНА РОБОТА

НА ЗДОБУТТЯ СТУПЕНЯ БАКАЛАВРА

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ «СИСТЕМНИЙ АНАЛІЗ І УПРАВЛІННЯ»

СПЕЦІАЛЬНОСТІ 124 «СИСТЕМНИЙ АНАЛІЗ»

НА ТЕМУ «ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ У ПРОГНОЗУВАННІ ПОПУЛЯЦІЙНОЇ ДИНАМІКИ»

Виконав: студент IV курсу, групи КА-12
Кобєлєв Володимир Євгенійович
Науковий керівник: професор кафедри ММСА, д. т. н., проф.
Дмитрієва Ольга Анатоліївна

- **Об’єктом дослідження** кваліфікаційної роботи є динамічні процеси популяційної динаміки, що описуються звичайними диференціальними рівняннями та їх системами із сконцентрованими параметрами.
- **Предметом дослідження** виступають наближені вирішувачі, побудовані за допомогою глибоких нейронних мереж і орієнтовані на відновлення поведінки динамічних об’єктів.
- **Мета** роботи полягає у розробці інтеграторів з використанням глибоких нейронних мереж диференціального типу, орієнтованих на відновлення поведінкових траєкторій динамічних об’єктів.

АКТУАЛЬНІСТЬ

- Актуальність дослідження зумовлена потребою у точному прогнозуванні динаміки біологічних та екосистемних процесів, які описуються звичайними диференціальними рівняннями. Класичні чисельні методи не завжди забезпечують бажану точність або стабільність у таких умовах. Глибокі нейронні мережі, зокрема архітектури диференціального типу (NeuralODE), дозволяють будувати наближені вирішувачі, здатні відтворювати складну динаміку навіть за обмежених вхідних даних. Це відкриває перспективи для створення універсальних інтеграторів у задачах прогнозування популяційної динаміки.

ПРЕДМЕТНА ОБЛАСТЬ

- У рамках дипломної роботи розглядалася задача моделювання популяційної динаміки як задача Коші для систем звичайних диференціальних рівнянь:

$$\frac{dy(t)}{dt} = f(t, y(t)), \qquad y(t_0) = y_0,$$

де $y(t)$ — вектор станів системи в момент часу t,

$f(t, y(t))$ — задана функція правих частин,

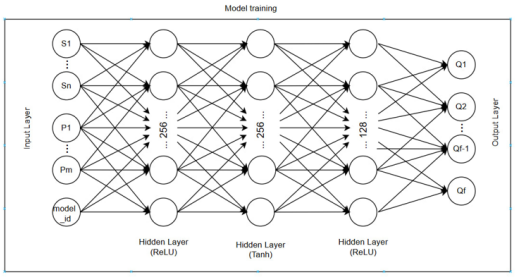
t_0 — початковий момент часу,

y_0 — початкові умови.

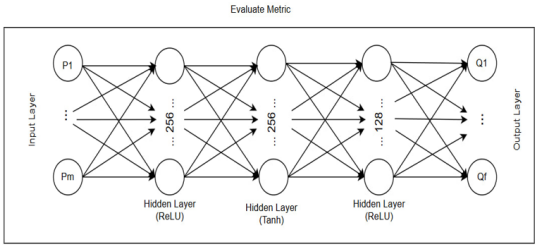
- Після генерації симуляційних траєкторій, отриманих шляхом чисельного розв'язання відомих математичних моделей, нейронна мережа повинна навчитися відновлювати швидкість зміни стану системи, тобто відповідну похідну, на основі поточного значення x(t). Таким чином, вона відтворює поведінку динамічної системи без явного використання аналітичного вигляду рівнянь.
- Для навчання були обрані дві системи з різних прикладних доменів:
  – модель ВІЛ (2 змінні, біомедична система);
  – модель "трава–вівці–вовки" (3 змінні, екосистема).

АРХІТЕКТУРА МОДЕЛІ NEURALODE

- Архітектура побудована у вигляді багаторівневої глибокої нейронної мережі, яка апроксимує функцію похідної в задачі Коші. Модель реалізована в середовищі PyTorch з використанням модуля torchdiffeq.odeint.



На етапі навчання модель отримує вхідний вектор, сформований з множини симуляцій $S_1 \ldots Sn$, параметрів $P_1 \ldots Pm$ та ідентифікатора model_id. Вектор послідовно обробляється трьома повнозв'язними шарами: два з 256 нейронами (ReLU, Tanh) і один з 128 нейронами (ReLU). Вихід — вектор $Q_1 \ldots Qf$, що містить наближені значення похідної.



Під час оцінювання якості до нейромережі подається лише множина параметрів $P_1 \ldots Pm$, оскільки інформація про model_id уже закодована в її структурі після навчання. Архітектура не змінюється, що гарантує універсальність і стабільність при роботі з різними симуляціями. Вихідний вектор $Q_1 \ldots Qf$ містить наближені значення похідної стану системи, які використовуються для побудови траєкторій методом чисельного інтегрування.

МОДЕЛЬ ВІЛ

- Першим прикладом динамічної системи, використаної для навчання нейронної мережі, є модель взаємодії CD4+ Т-клітин та вірусного навантаження (віріонів). Вона описує перебіг ВІЛ-інфекції та задається у вигляді системи звичайних диференціальних рівнянь:

$$\begin{cases} \frac{dT}{dt} = s - d \cdot T - k \cdot T \cdot V \\ \frac{dV}{dt} = k \cdot T \cdot V - c \cdot V \end{cases}$$

де T — концентрація Т-клітин,

V — вірусне навантаження.

Початкові умови:

$T_0 \in [0.8, 1.2], V_0 \in [0.05, 0.2]$,

s = 0.03, d = 0.01, k = 0.001, c = 0.1 — сталі параметри.

МОДЕЛЬ ТРАВА–ВІВЦІ–ВОВКИ

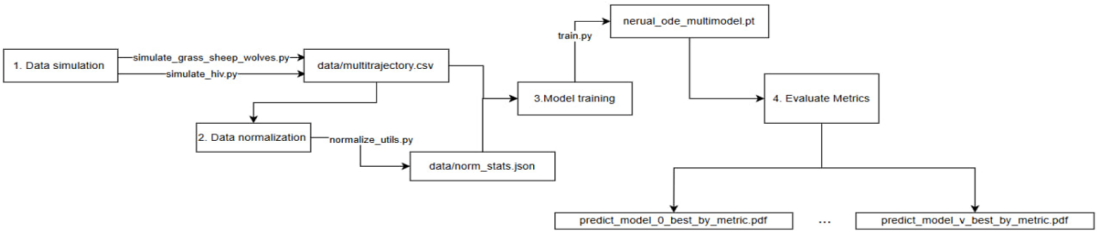
- Другим прикладом динамічної системи, використаної для навчання нейронної мережі, є модель трофічної взаємодії трьох популяцій — трави (G), вівців (S) та вовків (W). Вона описує поведінку екосистеми та задається у вигляді системи звичайних диференціальних рівнянь:

$$\begin{cases} \frac{dG}{dt} = a \cdot G(1-G) - b \cdot G \cdot S \\ \frac{dS}{dt} = c \cdot G \cdot S - d \cdot S \cdot W \\ \frac{dW}{dt} = e \cdot S \cdot W - f \cdot W \end{cases}$$

де G, S і W — відповідно біомаса трави, чисельність популяцій вівць і вовків, які варіювались у межах $G_0 \in [0.3, 0.7], S_0 \in [0.1, 0.5], W_0 \in [0.05, 0.2]$. Для забезпечення навчання на крайових випадках були додатково згенеровані симуляції зі значеннями $S_0 = 0.1, W_0 = 0.05$, при варіативному G_0. $a = 1.0, b = c = d = e = f = 0.1$ — сталі параметри.

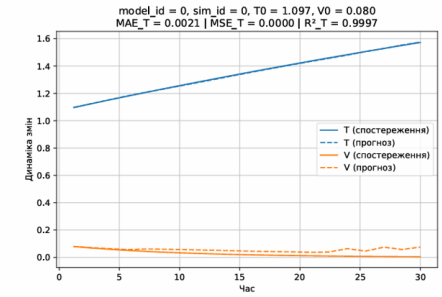
РОБОТА ПРОГРАМНОГО КОДУ

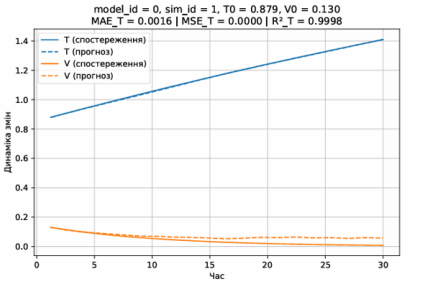
- Дані генеруються за допомогою simulate_hiv.py та simulate_grass_sheep_wolves.py на основі математичних моделей. Для кожного sim_id формується симуляційна траєкторія (T, V або G, S, W), яка зберігається у файл data/multitrajectory.csv з відповідним параметром model_id. Лише дані моделі “трава–вівці–вовки” (GSW) проходять нормалізацію (normalize_utils.py) на основі статистики, збереженої у data/norm_stats.json.
- Під час навчання (train.py) модель отримує вхідний вектор, сформований на основі симуляційних даних ($S_1 \ldots Sn$, $P_1 \ldots Pm$) та ідентифікатора model_id. Виходом є наближене значення похідної стану $Q_1 \ldots Qf$, яке надалі інтегрується для побудови прогнозу. Для навчання використовується 80 % точок симуляції — 4 з 5 значень, тоді як кожне п’яте значення потрапляє до тестової вибірки (стратегія маскування по модулю 5).
- Оцінювання (evaluate_metrics_pdf.py) передбачає:
  - вибір тестових траєкторій з multitrajectory.csv для обох моделей (model_id = 0/1);
  - побудову прогнозів через інтегрування (odeint);
  - обчислення метрик MAE, MSE, R^2 для змінних (T, V, G, S, W);
  - автоматичне збереження результатів у PDF (predict_model_*_best_by_metric.pdf).



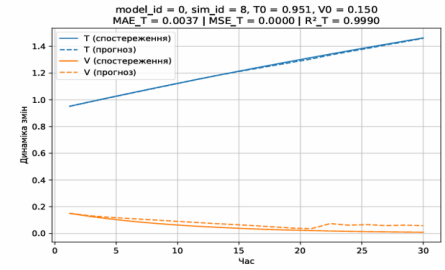
РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ДИНАМІКИ ВІЛ

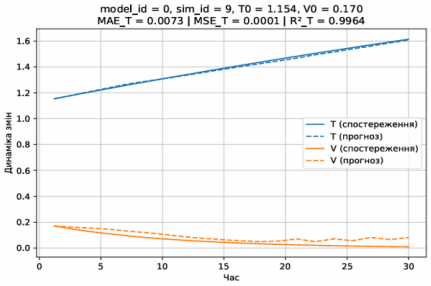
Перші симуляції





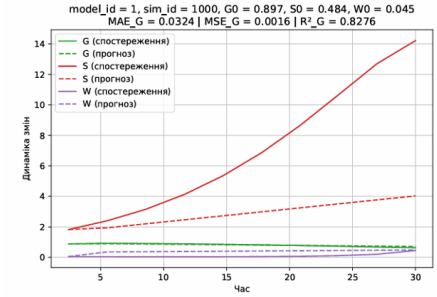
Останні симуляції

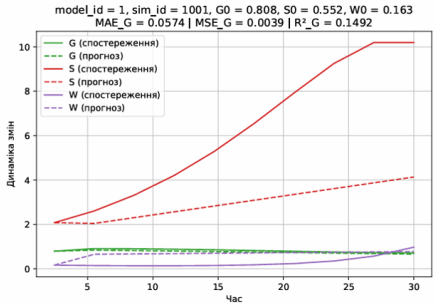




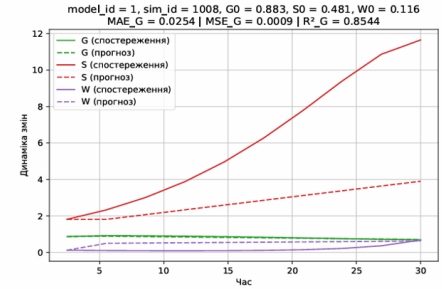
РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ДИНАМІКИ GSW

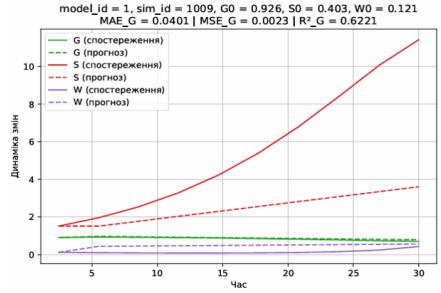
Перші симуляції





Останні симуляції





МЕТРИКИ ТОЧНОСТІ ПРОГНОЗУ ПРИ МОДЕЛЮВАННІ

Mean Absolute Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|yi - \hat{y}i|,$$

Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2$$

де yi — фактичне значення,

ŷi — прогнозоване значення,

n — кількість спостережень.

Coefficient Of Determination

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(yi - \hat{y}i)^2}{\sum_{i=1}^{n}(yi - \bar{y})^2}$$

де ȳ — середнє значення фактичних спостережень.

11

Метрики точності моделі для найкращих і найгірших симуляцій

model_id	sim_id	Змінна (оцінювана)	MAE	R²	Якість / модель
0	0	T	0.0021	0.9997	Найкраща (ВІЛ)
0	7	T	0.0115	0.9904	Найгірша (ВІЛ)
1	1007	G	0.0159	0.8735	Найкраща (GSW)
1	1001	G	0.0574	0.1492	Найгірша (GSW)

Метрики точності моделі для симуляцій з різних частин вибірки

model_id	sim_id	Змінна (оцінювана)	MAE	MSE	R²	Модель / позиція
0	0	T	0.0021	0.0000	0.9997	ВІЛ_початок
0	1	T	0.0016	0.0000	0.9998	ВІЛ_початок
0	8	T	0.0037	0.0000	0.9990	ВІЛ_кінець
0	9	T	0.0073	0.0001	0.9964	ВІЛ_кінець
1	1000	G	0.0324	0.0016	0.8276	GSW_початок
1	1001	G	0.0574	0.0039	0.1492	GSW_початок
1	1008	G	0.0254	0.0009	0.8544	GSW_кінець
1	1009	G	0.0401	0.0023	0.6221	GSW_кінець

12

ВИСНОВКИ

- У рамках дослідження було створено універсальний інтегратор на основі архітектури NeuralODE, призначений для моделювання динаміки систем, що описуються звичайними диференціальними рівняннями. Навчання моделі здійснювалося на симуляційних даних з математичних моделей ВІЛ та GSW, що забезпечило перевірку її здатності відтворювати як прості, так і складні типи поведінки.
- Для оцінювання якості прогнозу застосовувались три основні метрики: MAE (середня абсолютна похибка), MSE (середньоквадратична похибка) та R^2 (коефіцієнт детермінації). За результатами тестування модель продемонструвала високу точність для системи ВІЛ ($R^2 > 0.99$), а також зберегла ключові тренди динаміки у складнішій системі “Трава–Вівці–Вовки”.
- Отримані результати підтверджують ефективність диференціальної нейронної мережі як наближеного вирішувача в задачах, де необхідна балансованість між точністю та універсальністю. Підхід, що поєднує симуляційні дані з інтеграційною здатністю NeuralODE, є перспективним для застосування в моделюванні біологічних, екологічних та інших складних систем.

13

ПОДАЛЬШІ ДОСЛІДЖЕННЯ

- Подальший розвиток цієї роботи доцільно спрямувати на розширення методики до більш складних задач, зокрема — до систем, що описуються частинними диференціальними рівняннями, тобто мають ще й просторову залежність. Це дозволить моделювати явища з географічною структурою або внутрішньою просторовою динамікою — наприклад, поширення популяцій у середовищі.
- Крім того, підвищити якість і стабільність моделі, особливо для складних систем типу “Трава–Вівці–Вовки”, можна за рахунок:
  - оптимізації архітектури нейромережі (кількість шарів, нейронів, тип активацій);
  - використання шумостійкого навчання для кращої адаптації до непередбачуваних або зашумлених даних;
  - впровадження AutoML-рішень для автоматичного підбору гіперпараметрів;
  - збільшення обчислювальної потужності, що дозволить працювати з більшими наборами даних і підвищити точність.
- Окремим напрямом є створення графічного інтерфейсу користувача, який дозволить застосовувати модель не лише як дослідницький інструмент, а як повноцінний засіб підтримки рішень у біологічних, екологічних чи медичних задачах.

14

ЛІТЕРАТУРА

- Chen R. T. Q., Rubanova Y., Bettencourt J., Duvenaud D. Neural Ordinary Differential Equations. Advances in Neural Information Processing Systems, 2018, Vol. 31. URL: https://papers.nips.cc/paper/7892-neural-ordinary-differential-equations
- Kloberdanz E., Le W. S-SOLVER: Numerically Stable Adaptive Step Size Solver for Neural ODEs. 2023. URL: https://dl.acm.org/doi/10.1007/978-3-031-44201-8_32
- Fronk C., Petzold L. Training Stiff Neural Ordinary Differential Equations with Implicit Single-Step Methods. 2024 URL: https://arxiv.org/abs/2410.05592
- Garfinkel A., Shevtsov J., Guo Y. Modeling Life: The Mathematics of Biological Systems. Cham : Springer, 2017. 382 p.
- A Model of HIV Immunology within an Individual Person. URL: https://modelinginbiology.github.io/A-Model-of-HIV-immunology-within-an-Individual-Person-interactive-simulations
- Grass–Sheep–Wolves Interactive Simulations. URL: https://modelinginbiology.github.io/Grass-Sheep-Wolves-interactive-simulations

Дякую за увагу!