Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Математичні моделі та методи
вирішення оптимізаційних фінансових задач
на основі нейронних мереж
(тема)

Виконав:
здобувач _2_ року навчання, групи _САУм-23-1
Петришин А.Ю.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз і управління
(повна назва освітньої програми)
Керівник доц. Єсілевський В.С.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _124 Системний аналіз
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ _25_ ” _листопада_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Петришину Андрію Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Математичні моделі та методи вирішення оптимізаційних фінансових задач на основі нейронних мереж
затверджена наказом по університету від 22 листопада 2024 р. № 1228 Ст

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

3. Вихідні дані до роботи _задача портфелю Мертона

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Єсілевський В.С
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 59 с., 11 рис., 1 дод., 19 джерел.

ОПТИМАЛЬНЕ КЕРУВАННЯ, ЗАДАЧА ПОРТФЕЛЮ МЕРТОНА, ФІЗИКО-ІНФОРМОВАНІ НЕЙРОННІ МЕРЕЖІ, РІВНЯННЯ ГАМІЛЬТОНА-ЯКОБІ-БЕЛЛМАНА, ОПТИМІЗАЦІЯ У ФІНАНСАХ, ДИФЕРЕНЦІЙНІ РІВНЯННЯ З ЧАСТИННИМИ ПОХІДНИМИ.

Об’єкт дослідження – задача портфелю Мертона, що зводиться до диференційних рівняння в частинних похідних.

Мета роботи – дослідження розв’язку задачі Мертона з використанням нейронних мереж для оптимізації фінансових стратегій.

Методи дослідження – фізико-інформовані нейронні мережі.

В даній кваліфікаційній роботі було розглянуто фізико-інформовану нейронну мережу, що була змодельована і навчена під вирішення задачі портфелю Мертона, що зводиться до диференційного рівняння в частинних похідних. За результатом дослідження було отримано, що нейромережа гарно наближає шукану функцію. Отримані результати можна застосувати як основу для подальшого дослідження фізико-інформованих нейронних мереж для вирішення задач, пов’язаних із диференційними рівняннями. Так, побудовану нейромережу можна застосувати для подальшого вирішення інших диференційних рівнянь, зокрема, тих, які не мають аналітичного розв’язку. В наступних дослідженнях можна розглянути збіжність даного виду нейромереж, що дасть обґрунтованість застосування цієї нейромережі і підвищить точність результатів.

ABSTRACT

Introductory note: 59 pages, 11 figures, 1 appendix, 19 sources.

OPTIMAL CONTROL, MERTON'S PORTFOLIO PROBLEM, PHYSICS-INFORMED NEURAL NETWORK, HAMILTON-JACOBIAN-BELLMAN EQUATION, FINANCE OPTIMIZATION, PARTIAL DIFFERENTIAL EQUAION.

Object of research – Merton's portfolio problem, which reduces to a partial differential equation.

Purpose of work – to study the solution of Merton's problem using neural networks to optimize financial strategies.

Methods of research – physics-informed neural networks.

In this qualification work, physically-informed neural network was considered, which were modeled and trained to solve the Merton portfolio problem, a reducible partial differential equation. The study shows that the neural network approximates the desired function well. The obtained results can be used as a basis for further study of physically-informed neural networks for solving problems related to differential equations. Thus, the constructed neural network can be used to further solve other differential equations, in particular, those that do not have an analytical solution. In future studies, we can consider the convergence of this type of neural network, which will give the validity of the use of this neural network and increase the accuracy of the results.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

PINN – Physics-informed neural network;

HJB – Hamilton-Jacobi-Bellman.

ВСТУП

Актуальність теми. В сучасному світі дедалі більше і частіше виникає необхідність у вирішенні диференційних рівнянь із частинними похідними у фізиці, біології, фінансах тощо. Часто, для цих диференційних рівнянь складно або неможливо знайти точний аналітичний розв'язок, що призводить до необхідності застосування чисельних методів, як от метод сіток, метод скінченних елементів тощо. Та ці чисельні методи потребують значних обчислювальних ресурсів і, відповідно, часу для вирішення конкретної задачі, особливо якщо система, яку вирішують є дуже динамічною і розв'язок треба шукати постійно новий. Зважаючи на це, пошук методу, який перевершить чисельні методі є актуальною темою сьогодення.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є дослідження методів розв'язання задачі портфелю Мертона [1, 2] з використанням нейронних мереж для оптимізації фінансових стратегій. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі «оптимізації у фінансах»;

– дослідити існуючі методи розв'язання задачі оптимального контролю в контексті фінансових стратегій;

– розробити та апробувати нейромережевий підхід для вирішення задачі портфелю Мертона;

– оцінити ефективність розробленої моделі на основі числових експериментів.

Об'єктом дослідження є задача портфелю Мертона, що зводиться до диференційного рівняння в частинних похідних.

Предметом дослідження є математичні моделі та методи, зокрема нейронні мережі, для вирішення задачі оптимального контролю в задачі портфелю Мертона.

Методи дослідження. У кваліфікаційній роботі використовуються фізико-інформовані нейронні мережі [3], що застосовуються для розв'язання рівняння Гамільтона-Якобі-Беллмана [4], яке витікає з задачі портфелю Мертона.

Публікації. Результати, отримані у кваліфікаційній роботі, було представлено на Information Technology and Implementation (Satellite) (Kyiv, 21 November 2024) [5], 13-й Міжнародній науково-технічної конференції ІСТ-2024 (м. Харків 26 – 28 листопада 2024 р.) [6].

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз задачі оптимізації у фінансах

Оптимізація у фінансах є ключовим процесом для ефективного управління ресурсами з метою досягнення певних фінансових цілей. У контексті фінансової оптимізації, цей процес охоплює розподіл ресурсів таким чином, щоб максимально збільшити дохідність, мінімізувати ризики або знайти баланс між дохідністю та ризиком. Фінансова оптимізація включає різні аспекти, зокрема управління інвестиційними портфелями, планування грошових потоків, побудову структури капіталу, управління ризиками та інші ключові елементи фінансового менеджменту.

Фінансова оптимізація передбачає розробку та застосування математичних моделей, які допомагають визначити оптимальну комбінацію фінансових інструментів або активів з урахуванням обмежень та поставлених цілей. Моделі враховують важливі критерії, такі як дохідність, ризик, ліквідність, диверсифікація та відповідність стратегічним цілям компанії або інвестора. Процес оптимізації передбачає детальний аналіз різних сценаріїв розвитку подій, оцінку макроекономічних умов та ринкових тенденцій, а також застосування сучасних методів, таких як лінійне та нелінійне програмування, стохастичні моделі, методи машинного навчання та нейронні мережі для обробки великих обсягів даних.

Задачу оптимізації у фінансах можна розбити на кілька ключових елементів, які визначають оптимальні фінансові рішення:

– цілі оптимізації – показники, досягнення яких прагне фінансовий суб'єкт, включають максимізацію прибутку, мінімізацію ризику, забезпечення ліквідності або збалансування ризику й дохідності;

– фінансові інструменти та активи – різноманітні активи (акції, облігації, деривативи тощо), які використовуються для досягнення цілей оптимізації;

– обмеження – зовнішні та внутрішні фактори, що обмежують досягнення оптимальних рішень, зокрема політика компанії, структура капіталу, ринкові регуляції;

– критерії оптимальності – параметри, на основі яких оцінюється ефективність рішень, такі як дохідність, рівень ризику, ліквідність, коефіцієнт Шарпа [7];

– методи оптимізації – підходи для знаходження оптимальних рішень, включаючи математичні моделі, лінійне та нелінійне програмування, методи машинного навчання;

– ринкові та економічні фактори – макроекономічні та ринкові змінні, які впливають на результати оптимізації, наприклад інфляція, відсоткові ставки;

– моделі прогнозування – інструменти для оцінки майбутньої поведінки фінансових інструментів та економічних умов;

– аналіз результатів та прийняття рішень – оцінка отриманих даних та розробка практичних стратегій.

При вирішенні задачі оптимізації у фінансах, сама задача може зводитися до задачі оптимального управління. Розглянемо ключові елементи цієї задачі:

– змінні стану – це змінні, які описують поточний стан фінансової системи або портфеля. Прикладами можуть бути вартість активів, рівень боргу, процентні ставки тощо;

– змінні керування – змінні, які можна регулювати для впливу на систему. У фінансовому контексті це можуть бути обсяги інвестицій у різні активи, ставки реінвестування, рівні споживання та інші рішення управління;

– динаміка системи – ці рівняння описують, як станові змінні змінюються з часом під впливом керуючих змінних та випадкових факторів. У фінансах це часто диференціальні рівняння або стохастичні процеси, які моделюють поведінку ринків та активів;

– цільова функція – це функція, яку необхідно оптимізувати (максимізувати або мінімізувати). Це може бути максимізація прибутковості, мінімізація ризику або оптимізація співвідношення ризику та прибутку.

– обмеження – це умови, які повинні виконуватися під час оптимізації. Вони можуть включати бюджетні обмеження, регуляторні вимоги, обмеження на ризик, а також початкові та кінцеві умови для станових змінних;

– невизначеність та ризик – часто, у фінансах, враховуються випадкові фактори, такі як волатильність ринку та непередбачувані зміни цін. Це може вимагати використання стохастичних методів оптимального контролю;

– методи вирішення – алгоритми та математичні методи, які використовуються для знаходження оптимального контролю. Це можуть бути аналітичні методи, чисельні алгоритми, динамічне програмування, методи оптимізації тощо.

Ці елементи формують комплексну структуру задачі оптимального управління у фінансах. Розуміння та аналіз кожного з них є критично важливим при проведенні системного аналізу, оскільки вони впливають на побудову моделі, вибір методів вирішення та інтерпретацію отриманих результатів.

1.2 Аналіз сценаріїв вирішення задачі оптимального управління у фінансах

При аналізі фінансового ринку, виникають різні задачі оптимального управління у фінансах. Розглянемо конкретніше які саме задачі оптимального управління виникають.

Управління інвестиційним портфелем. Основна задача в цьому контексті полягає у виборі такої стратегії розподілу активів, яка забезпечує оптимальне співвідношення між дохідністю та ризиком. Методологія Марковіца [8] є класичним підходом до розв’язання цієї задачі, що дозволяє побудувати ефективний портфель шляхом мінімізації дисперсії його доходності за умови досягнення заданого рівня очікуваної дохідності. У багатоперіодному контексті для вирішення задач управління портфелем широко застосовується динамічне програмування, яке дозволяє враховувати зміну ринкових умов із часом. Крім того,

стохастичний контроль, заснований на моделях типу рівняння Блека-Шоулза [9], використовується для моделювання динаміки активів і визначення оптимальних стратегій. Однак основними викликами в цьому напрямі залишаються висока невизначеність ринку, нелінійність фінансових процесів і залежність від точності історичних даних.

Управління боргом та ліквідністю. У цьому випадку метою є розробка стратегії, яка забезпечує баланс між виконанням боргових зобов'язань і підтриманням достатнього рівня ліквідності для покриття поточних фінансових потреб. Для цього широко використовуються сценарне планування і моделі контролю запасів ліквідності, які дозволяють розглядати можливі варіанти розвитку подій і визначати оптимальний рівень резервів. Крім того, машинне навчання і нейронні мережі застосовуються для прогнозування ризиків і оптимізації управлінських рішень. Основними викликами в цій сфері є значна залежність від макроекономічних факторів, таких як процентні ставки і ринкова кон'юнктура, а також регуляторні обмеження.

Управління фінансовими ризиками. До таких ризиків відносяться валютні, кредитні, ринкові тощо. Основною метою є зниження ймовірності негативних фінансових наслідків через несприятливі події на ринку. Серед методів вирішення таких задач найбільш поширеними є хеджування за допомогою фінансових інструментів, таких як опціони, ф'ючерси або свопи, а також застосування стохастичних моделей і «Value at Risk» [10] для оцінки й управління ризиками. Однак для успішного впровадження таких підходів необхідно враховувати складність у побудові адекватних моделей, залежність від ринкових даних і невизначеність у поведінці контрагентів.

Оптимізація фінансових потоків у корпоративному секторі. У цьому контексті широко застосовуються методи лінійного та нелінійного програмування, алгоритми генетичної оптимізації і підкріплюючого навчання, які дозволяють визначити оптимальну стратегію управління фінансовими ресурсами. Багатокритеріальна оптимізація також грає важливу роль, оскільки дозволяє враховувати одночасно кілька показників, таких як рентабельність, ризик і ліквідність.

Серед основних викликів є невизначеність макроекономічного середовища і залежність від зовнішніх факторів, таких як податкове регулювання і валютна політика.

Управління страхуванням та пенсійними фондами. У цьому випадку метою є забезпечення достатньої дохідності активів для покриття довгострокових зобов'язань перед клієнтами. Для цього використовуються актуарні моделі, методи «Asset-Liability Management» [11] і стохастичний контроль. Однак такі задачі мають ряд специфічних викликів, зокрема необхідність довгострокового прогнозування демографічних і економічних змін, а також дотримання суворих регуляторних вимог.

Таким чином, оптимальний контроль у фінансах є багатогранним інструментом, який дозволяє ефективно вирішувати складні задачі, пов'язані з управлінням активами, ризиками, фінансовими потоками і зобов'язаннями. Незважаючи на існуючі виклики, розвиток математичних методів, машинного навчання і обчислювальних потужностей відкриває нові можливості для впровадження оптимального контролю в різних сферах фінансової діяльності, сприяючи підвищенню ефективності прийняття управлінських рішень і забезпеченню стабільності фінансових систем.

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Розглянемо інвестора, який хоче вкласти гроші для збільшення власних статків. В цього інвестора є певна сума грошей. Ці гроші можна покласти на депозит, який приносить стабільний, але невеликий дохід, або вкласти в акції, які можуть принести великий прибуток, але з ризиком втрат. Завдання інвестора – розподілити гроші між цими двома варіантами так, щоб у кінцевому підсумку отримати найбільшу вигоду (рис. 1.1).

Складність задачі полягає в постійній зміні ринку: ціни на акції можуть різко зростати або падати, і ніхто не знає напевно, що буде далі. Тому інвестору потрібно приймати рішення не лише один раз, а постійно змінювати свою стратегію, враховуючи поточну ситуацію.

Цю задачу називають задачею портфеля Мертона і вона є задачею оптимальногой керування, бо інвестору потрібно не просто щось зробити, а знайти найкращий спосіб діяти в умовах невизначеності. Це як керувати автомобілем: водій не просто їде прямо, а постійно реагує на повороти дороги, щоб дістатися до пункту призначення з найменшими втратами.

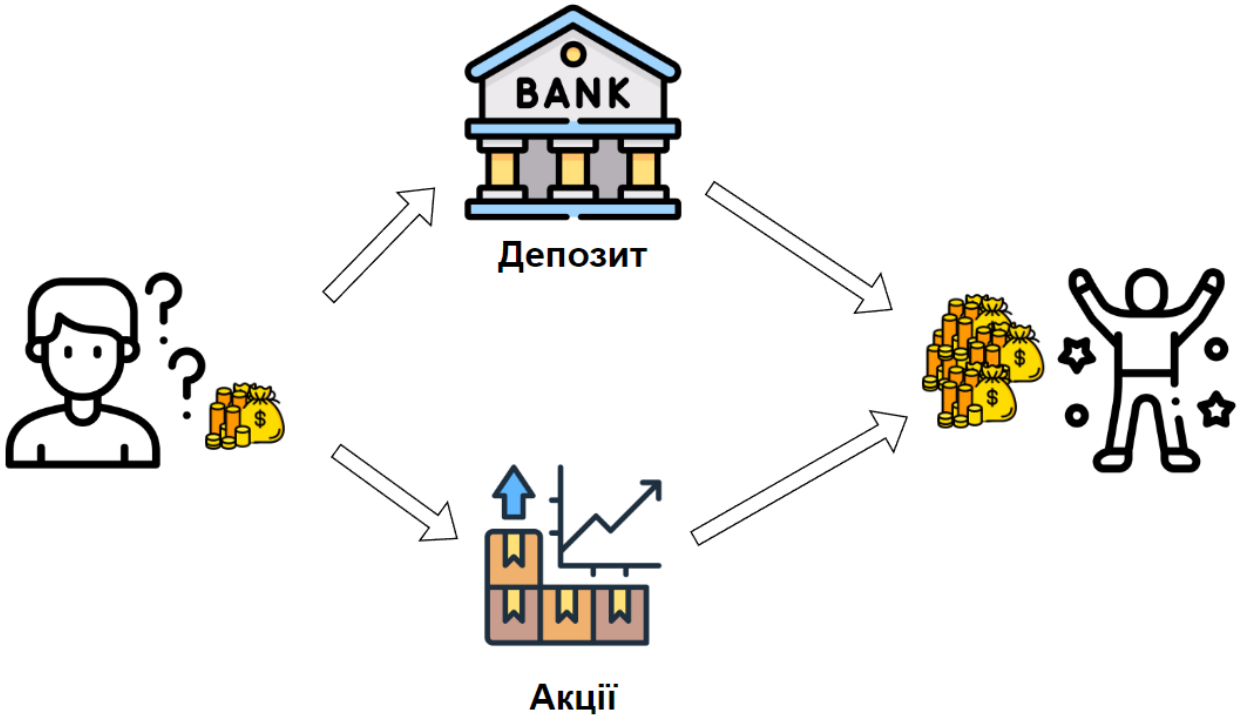

Рисунок 1.1 – Схематична постановка задачі

1.3.2 Формальна постановка задачі

Нехай інвестор зараз має капітал $W_0 > 0$ і відомо, що він розглядає інвестування протягом часу T років. Інвестор має змогу інвестувати в один ризиковий та один безризиковий активи. Задача інвестора полягає у визначенні най-

кращого плану інвестицій, тобто пошук найкращої стратегії інвестиції, так, щоб наприкінці часу T він зміг максимізувати прибуток від капіталовкладень. Тож постає питання як саме інвестор має вкладати свої статки в кожен момент часового горизонту щоб досягти його цілі.

Мертон запропонував наступну модель зміни капіталу:

$$dW_t = \left(\left(\pi_t \cdot (\mu - r) + r\right) \cdot W_t - c_t\right) \cdot dt + \pi_t \cdot \sigma \cdot W_t \cdot dB_t, \qquad (1.1)$$

де $\pi_t = \pi(t, W_t)$ – частка інвестицій, що йде на ризиковий актив в момент часу t;

$1 - \pi(t, W_t)$ – частка інвестицій на безризоковий актив в момент часу t;

$W_t = W(t) > 0$ – капітал;

$B_t = B(t)$ – геометричний броунівський рух;

$c_t = c(t, W_t) \geq 0$ – споживання інвестицій;

σ – волатильність ризикового активу;

μ – очікувана дохідність ризикового активу;

r – постійна відсоткова ставка безризикового активу.

Для даної моделі необхідно вирішити задачу оптимального управління:

$$\max_{\pi_t, c_t} E\left[U\left(W(T)\right)\right], \qquad (1.2)$$

де $U(W) = \dfrac{W^{1-\gamma}}{1-\gamma}$ – очікувана корисність інвестицій при γ – коефіцієнт відносно ухилення від ризику;

$E[...]$ – функція, що описує математичне сподівання.

Розв'язок задачі (2.1) – (2.2) необхідно шукати за допомогою нейромереж, спростивши задані рівняння до диференційних рівнянь із частинними похідними.

1.4 Постановка задач дослідження

Метою дослідження є розробка математичної моделі на основі нейромереж для пошуку розв’язку задачі (1.1). Для досягнення поставленої мети необхідно виконати наступні задачі:

– провести огляд, аналіз сучасного стану задачі оптимального управління;

– оглянути методи вирішення задачі оптимального управління;

– дослідити обраний підхід до рішення задачі;

– побудувати модель на основі нейромереж для вирішення задачі;

– програмно реалізувати нейромережу для вирішення проблеми;

– провести аналіз натренованої моделі;

– на основі отриманих даних зробити висновки про проведену роботу.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Методи вирішення задачі оптимального управління

2.1.1 Принцип максимуму Понтрягіна

Принцип максимуму Понтрягіна – один із ключових інструментів в теорії оптимального керування, розроблений Левом Понтрягіним і його колегами. Цей принцип формалізує необхідні умови оптимальності для динамічних систем, що перебувають під керуванням, і допомагає знайти оптимальні траєкторії для керованих процесів [12].

Принцип максимуму Понтрягіна стверджує, що для оптимального керування необхідно максимізувати (або мінімізувати, залежно від задачі) певний функціонал (зазвичай, це функціонал Гамільтона) при заданих обмеженнях на динамічну систему. Це означає, що оптимальне керування вибирається таким чином, щоб функціонал Гамільтона був максимальним у кожен момент часу вздовж траєкторії оптимального руху.

Розглянемо динамічну систему із вектором станів $x(t)$ і вектором керування $u(t)$, де динамічне рівняння системи описується як:

$$\frac{dx}{dt} = f\left(x(t), u(t)\right), \tag{2.1}$$

де $f\left(x(t), u(t)\right)$ – векторна функція, що визначає зміну стану системи залежно від керування $u(t)$ та часу t, а функціонал оптимізації, який потрібно максимізувати чи мінімізувати має вигляд:

$$J = \int_{t_0}^{T} L\left(x(t), u(t)\right) dt + \phi\left(x(T)\right), \tag{2.2}$$

де $L\left(x(t),u(t)\right)$ – миттєва функція витрат;

$\phi\left(x(T)\right)$ – функція від кінцевого стану.

Для застосування принципу максимуму вводять допоміжний вектор спряжених змін $\lambda(t)$ і визначають функцію Гамільтона:

$$H\left(x(t),u(t),\lambda(t)\right)=\lambda^{T}(t)f\left(x(t),u(t)\right)+L\left(x(t),u(t)\right). \tag{2.3}$$

Принцип максимуму Понтрягіна стверджує, що для оптимального керування $u^{*}(t)$ і оптимальної траєкторії $x^{*}(t)$ повинні виконуватися граничні умови задачі і поверх цього такі умови:

$$H\left(x^{*}(t),u^{*}(t),\lambda(t)\right)\geq H\left(x^{*}(t),u(t),\lambda(t)\right),\ \forall u(t) \tag{2.4}$$

$$\frac{dx^{*}}{dt}=\frac{\partial H}{\partial\lambda}, \tag{2.5}$$

$$\frac{d\lambda}{dt}=-\frac{\partial H}{\partial x}. \tag{2.6}$$

2.1.2 Динамічне програмування

Динамічне програмування (також динамічне програмування Беллмана) – це метод, що застосовується для вирішення задач оптимального керування і прийняття рішень, які можна розділити на послідовні етапи. Основна ідея методу полягає в тому, щоб розбити складну задачу на підзадачі, які легше розв’язувати, а потім рекурсивно об’єднати їх для отримання остаточного рішення. Метод базується на принципі оптимальності Беллмана, який стверджує, що «незалежно від початкового стану і початкового рішення, подальші рішення повинні утворювати оптимальну політику стосовно стану, до якого привело перше рішення [13].

Розглянемо процес прийняття рішень у дискретні моменти часу, де в кожен момент часу система перебуває у стані та виконується певна дія, яка змінює стан системи до наступного моменту часу. Динаміка системи описується як залежність стану в наступний момент часу від поточного стану та обраної дії. Кожна дія має певну «вартість» чи «виграш», який характеризується функцією витрат. Метою є знайти таку послідовність дій (політику), яка мінімізує сумарні витрати на всіх етапах від початкового до кінцевого стану.

Основним інструментом у динамічному програмуванні є функція вартості (функція Беллмана), яка представляє мінімальну суму витрат від поточного стану до кінцевого. Вона обчислюється рекурсивно за допомогою принципу оптимальності Беллмана. Функція вартості для кожного стану визначається як мінімум суми витрат за поточний крок і вартості наступного стану, куди система перейде залежно від обраної дії. Задача вирішується «зворотним індукційним методом»: починаючи з кінцевого моменту часу і поступово обчислюючи функцію вартості для всіх попередніх станів, ми можемо знайти оптимальну політику для всіх моментів часу.

Алгоритм роботи динамічного програмування Беллмана виглядає так: спочатку обчислюються значення функції вартості для кінцевих станів. Далі, для кожного попереднього моменту часу обчислюються значення функції вартості для всіх можливих станів, обираючи при цьому оптимальну дію для кожного з них. Після цього, на основі обчислених значень функції вартості, визначається оптимальна послідовність дій для всієї задачі.

Метод динамічного програмування Беллмана широко застосовується в задачах оптимального керування та прийняття рішень у таких сферах, як контрольні системи (навігація, авіація, робототехніка), економіка та фінансове управління (інвестиції, ризик-менеджмент), обчислювальна техніка та штучний інтелект (планування, навчання з підкріпленням). Головною перевагою методу є можливість вирішення багатокрокових задач з точним розв’язком, проте недоліком є експоненційне зростання складності обчислень з кількістю станів і дій, що призводить до проблеми «прокляття розмірності».

2.1.3 Метод сіток

Метод сіток (або метод дискретизації) – це підхід до чисельного розв’язання диференціальних рівнянь і задач оптимального керування, коли неперервний час, простір або обидва дискретизуються, тобто розбиваються на сітку з кінцевою кількістю вузлів. Цей метод дозволяє перетворити задачі, які в початковому вигляді є неперервними (наприклад, диференціальні рівняння), на кінцеві системи алгебраїчних рівнянь, які можна розв’язувати чисельно. Дискретизація є особливо корисною при розв’язанні задач оптимального керування, оскільки дозволяє використовувати алгоритми, розроблені для скінченновимірних задач [14].

Розглянемо етапи методу сіток.

Дискретизація простору та часу: Неперервний інтервал часу розбивається на рівномірні або нерівномірні проміжки. Точки дискретизації часу називаються вузлами сітки. Так само простір станів може бути розбитий на окремі комірки або сегменти, створюючи кінцеву кількість дискретних станів (рис. 2.1).

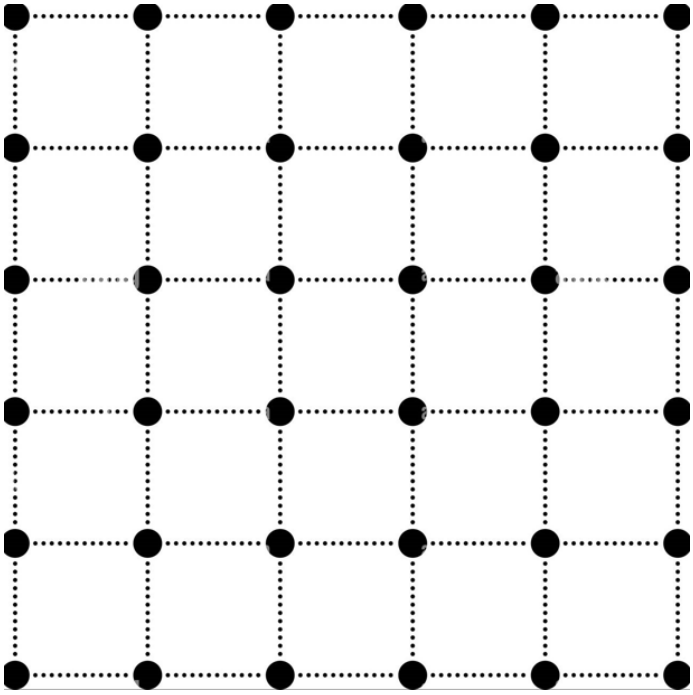

Рисунок 2.1 – Дискретизація часу та простору

Наближення (апроксимація) диференціальних рівнянь: Оскільки початкові рівняння можуть бути диференціальними, їх перетворюють на різницеві рівняння, що апроксимують зміни змінних стану та керування в кожному вузлі сітки. Наприклад, замість похідної можна використовувати різницеве співвідношення, наближене кінцевими різницями.

Побудова дискретної задачі: Після дискретизації рівняння, що описують динаміку системи, а також функціонал оптимізації (якщо це задача оптимального керування), перетворюються на систему алгебраїчних рівнянь. Ця система алгебраїчних рівнянь описує поведінку системи в дискретні моменти часу.

Чисельне розв’язання дискретної задачі: Отриману дискретну систему рівнянь можна розв’язувати чисельними методами, такими як метод Ньютона, метод простих ітерацій, метод найшвидшого спуску тощо. Якщо це задача оптимального керування, шукається така послідовність керувань на сітці, яка мінімізує або максимізує заданий функціонал витрат.

Переваги методу сіток:

– універсальність – метод сіток може застосовуватись до широкого класу диференціальних рівнянь і задач оптимального керування;

– чисельна реалізація – після дискретизації задача стає скінченновимірною, і для її розв'язання можна використовувати чисельні алгоритми, доступні для комп'ютерних обчислень;

– гнучкість: сітка може бути побудована нерівномірно, що дозволяє отримати більшу точність в областях із більш різкими змінами функції.

Недоліки методу сіток:

– прокляття розмірності – зі збільшенням розмірності задачі кількість вузлів сітки зростає експоненційно, що призводить до значних обчислювальних витрат;

– помилки апроксимації – дискретизація призводить до втрати точності, оскільки неперервні функції та їх похідні замінюються скінченними різницями.

2.1.4 Метод скінченних елементів

Метод скінченних елементів (МСЕ) – це чисельний метод для розв’язання диференціальних рівнянь у часткових похідних, що широко застосовується для аналізу та моделювання фізичних процесів, таких як механічна напруга, теплообмін, електромагнітні поля та гідродинаміка. Ідея МСЕ полягає в тому, щоб розділити складну геометрію області, в якій розв’язується задача, на менші, простіші частини (кінцеві елементи) і створити систему рівнянь, які апроксимують поведінку процесу в кожному з них [15].

Розглянемо основні етапи даного методу.

Розбиття області на кінцеві елементи. Спочатку область, у якій розв’язується задача, розбивається на під-області – кінцеві елементи (трикутники, квадрати, тетраедри тощо), утворюючи так звану сітку кінцевих елементів. Вузли сітки розташовуються в кутах (а інколи – на гранях чи всередині) елементів.

Вибір базових функцій. У кожному кінцевому елементі вибираються базові функції (зазвичай це поліноми низького ступеня), які визначають, як шуканий розв’язок змінюється всередині елемента. Базові функції використовуються для апроксимації розв’язку в кожному елементі сітки. Для більшості задач застосовують лінійні або квадратичні функції.

Формулювання локальних рівнянь для кожного елемента. Диференціальне рівняння перетворюється на систему рівнянь для кожного кінцевого елемента. Це часто робиться за допомогою методу Галеркіна, що зводить задачу до інтегральної форми, або за допомогою варіаційного підходу, коли мінімізується відповідний функціонал.

Збирання глобальної системи рівнянь. Локальні рівняння для кожного елемента об’єднуються у велику систему рівнянь для всієї сітки. Ця глобальна система лінійних (або нелінійних) рівнянь описує поведінку всього фізичного процесу в області.

Застосування граничних умов. Задані граничні умови додаються до глобальної системи рівнянь, що дозволяє адаптувати розв’язок до специфічних

фізичних умов на краях області (наприклад, температура або тиск на границях тіла).

Розв'язання системи рівнянь. Отримана система рівнянь розв'язується чисельними методами (наприклад, методом Гаусса, ітераційними методами) для отримання значень шуканих величин у вузлах сітки. Це значення апроксимують розв'язок диференціального рівняння в межах області.

До переваг методу кінцевих елементів відносяться такі:

– гнучкість у роботі зі складними геометричними формами;

– можливість роботи із матеріалами (фізичними) та неоднорідностями;

– можливість високої точності (при розбитті на достатньо дрібні елементи основної області).

Недоліки методу є наступні:

– висока обчислювальна складність;

– труднощі у побудові сіток;

– чутливість до якості.

2.1.5 Метод градієнтного спуску

Будемо розглядати задачу (2.1) – (2.6) описану раніше із початковим станом $x_0(t) = x_0$.

Метод градієнтного спуску застосовується для поступової корекції керування $u(t)$ таким чином, щоб зменшити значення функціоналу J. Основна ідея полягає в тому, щоб змінювати керування у напрямку, протилежному градієнту функціоналу витрат за керуванням. Градієнт функціоналу $\nabla_u J$ обчислюється як похідна функціоналу витрат по керуванню.

Опишемо ітеративний процес.

Крок 1. Початкове наближення керування. Починається з певного початкового керування $u_0(t)$, яке може бути обрано, наприклад, випадково або як

постійне значення.

Крок 2. Розв’язання прямої та спряженої задачі. У класичному підході до задач оптимального контролю з використанням градієнтного спуску спочатку обчислюється пряма задача, тобто знаходяться траєкторії стану $x(t)$, розв'язуючи рівняння динаміки. Потім обчислюється спряжена траєкторія $\lambda(t)$, що задовольняє спряжене рівняння (2.6).

Крок 3. Обчислення градієнта витрат за керуванням. Використовуючи значення $x(t)$ та $\lambda(t)$, обчислюється градієнт витрат по керуванню:

$$\nabla_u J = \frac{\partial H}{\partial u}.$$

Крок 4. Оновлення керування. Керування коригується за правилом градієнтного спуску:

$$u_{new}(t) = u_{current}(t) - \alpha \nabla_u J,$$

де α – це крок навчання (або коефіцієнт швидкості спуску), що визначає розмір змін у керуванні на кожному етапі.

Кроки 2 – 4 повторюються допоки значення функціоналу J не перестане значно змінюватися або не буде досягнуто максимальної кількості ітерацій [16].

На рис. 2.2 наведено приклад застосування градієнтного спуску для пошуку мінімуму (оптимуму) на множині рівнів.

Метод градієнтного спуску для оптимального контролю є потужним інструментом, що дозволяє знайти розв’язки для складних, нелінійних систем, коли аналітичне розв’язання важко або неможливо знайти. Однією з головних переваг є те, що метод дозволяє працювати з гнучким вибором початкового керування та адаптувати його в процесі.

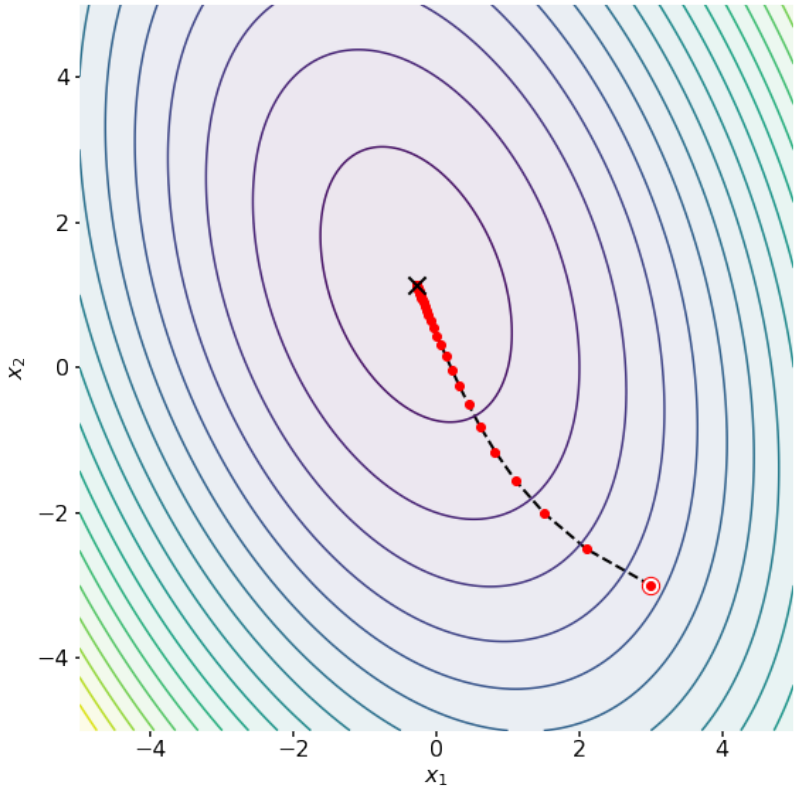

Рисунок 2.2 – Приклад роботи методу градієнтного спуску на множині рівнів (обведена колом точка – початкова; хрестик – оптимум)

Проте метод має і обмеження. Він може застрягнути в локальних мінімумах, особливо якщо функціонал витрат має багато екстремумів (рис 2.3). Крім того, ефективність алгоритму сильно залежить від вибору кроку α: занадто великий крок може призвести до нестабільності, а занадто малий – до повільної збіжності. Цей метод також потребує обчислення градієнта функціоналу, що може бути складним у випадку великих або нелінійних систем.

2.1.6 Генетичні алгоритми

Генетичні алгоритми використовують популяційний підхід, у якому розглядається група можливих розв’язків, що еволюціонують із покоління в покоління. Кожен розв’язок (або індивід) у популяції представляється у вигляді

хромосоми, яка є набором параметрів (генів) і кодує потенційний розв'язок задачі. Процес еволюції передбачає такі основні етапи: ініціалізацію, відбір, кросовер (схрещування), мутацію та заміну [17].

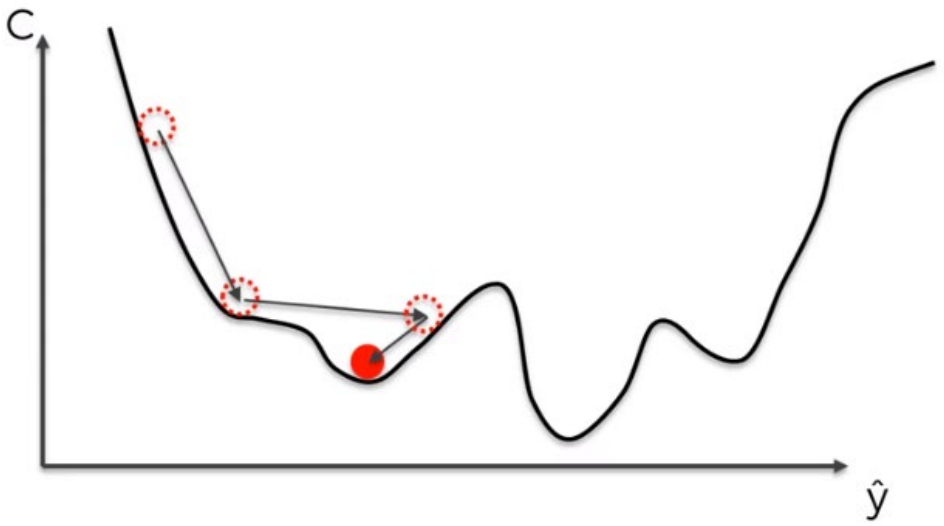

Рисунок 2.3 – Приклад градієнтного спуску, що знайшов екстремум, замість оптимуму

Розглянемо основні етапи генетичного алгоритму.

Ініціалізація популяції. На першому етапі створюється початкова популяція випадкових хромосом, де кожна хромосома кодує можливий розв'язок. Цей початковий набір розв'язків формує популяцію, з якою буде працювати алгоритм.

Оцінка пристосованості (фітнес-функція). Кожна хромосома (розв'язок) оцінюється за допомогою фітнес-функції, яка визначає, наскільки добрим є даний розв'язок для поставленої задачі. Фітнес-функція залежить від конкретної задачі та обчислює значення, яке алгоритм намагатиметься максимізувати (або мінімізувати, залежно від задачі).

Відбір. Після оцінки фітнесу проводиться відбір хромосом для створення нової популяції. Хромосоми з вищим значенням фітнесу мають більше шансів бути обраними для відтворення, що імітує природний відбір у живій природі. Існує кілька методів відбору, серед яких: турнірний відбір, рулетка та відбір

елітних особин.

Кросовер (схрещування). На цьому етапі обрані пари хромосом обмінюються частинами генів, створюючи нові хромосоми (нащадки). Кросовер дозволяє новим хромосомам комбінувати риси батьків, що допомагає знаходити нові розв’язки. Існують різні типи кросоверу, такі як одноточковий, двоточковий і рівномірний.

Мутація. Для підтримки генетичного різноманіття і запобігання застою в локальному мінімумі до деяких генів нових хромосом застосовується мутація, що випадково змінює їх значення. Мутація є важливим процесом для підтримання різноманітності популяції та для того, щоб алгоритм не «застряг» в одній області простору рішень.

Заміна популяції. Після кросоверу і мутації нові хромосоми замінюють стару популяцію або частину її, і процес повторюється. Таким чином, кожне нове покоління намагається покращити розв’язки завдяки еволюційному процесу.

Алгоритм продовжує генерувати нові покоління, поки не буде досягнуто певного критерію зупинки, наприклад, встановленої кількості поколінь, досягнення бажаного значення фітнес-функції або відсутності значного покращення у фітнес-функції протягом кількох поколінь.

Генетичні алгоритми мають кілька переваг:

– вони є потужними для задач, у яких неможливо обчислити градієнт цільової функції, або для оптимізації функцій із багатьма локальними екстремумами, де інші методи можуть застрягати в локальних мінімумам;

– вони гнучкі й можуть бути адаптовані до різних типів задач.

Однак вони мають і певні недоліки:

– генетичні алгоритми можуть бути обчислювально-затратними, оскільки потребують багато ітерацій для досягнення оптимального розв’язку, особливо для задач великої розмірності;

– ефективність алгоритму сильно залежить від вибору параметрів (швидкості мутації, типу відбору тощо), а неправильний вибір може призвести до по-

вільної збіжності або передчасної зупинки на локальних екстремумах.

2.1.7 Метод рою частинок

Метод рою частинок (англ. Particle Swarm Optimization, PSO) – це алгоритм оптимізації, натхненний поведінкою рою в природі, зокрема зграями птахів або косяками риб, які рухаються у просторі в пошуках їжі. Метод PSO належить до еволюційних алгоритмів оптимізації і використовується для розв'язання задач, в яких потрібно знайти глобальний мінімум або максимум цільової функції в багатовимірному просторі. У PSO група частинок (агентів) рухається у просторі розв’язків, дотримуючись певних правил, і кожна частинка поступово оновлює своє положення на основі власного досвіду та досвіду інших частинок у рої [18].

У PSO кожна частинка має своє поточне положення і швидкість, які оновлюються на кожній ітерації. Положення частинки відповідає потенційному розв'язку задачі, а швидкість визначає, як швидко і в якому напрямку частинка переміщується. Кожна частинка «пам'ятає» найкраще положення, у якому вона перебувала (особистий досвід), а також знає найкраще положення, знайдене всіма частинками у рої (колективний досвід).

Основні етапи методу.

Ініціалізація. Визначається початкова популяція частинок. Кожна частинка отримує початкове випадкове положення та швидкість у просторі розв’язків. Також кожній частинці присвоюється значення фітнес-функції, що визначає якість цього положення відповідно до поставленої задачі. Припустимо, що час $t \in N$.

Оновлення швидкості. На кожній ітерації швидкість частинки оновлюється за формулою:

$$v_i(t+1) = wv_i(t) + c_1 r_1 (p_i - x_i) + c_2 r_2 (g - x_i),$$

де $v_i(t+1)$ – швидкість частинки i на наступній ітерації;

w – коефіцієнт інерції, що визначає вплив попередньої швидкості на поточний рух;

c_1 і c_2 – коефіцієнти когнітивного та соціального впливу відповідно (вони визначають, наскільки частинка притягується до найкращого особистого і найкращого глобального положення);

r_1 і r_2 – випадкові числа у межах від 0 до 1, що додають елемент випадковості;

p_i – найкраще положення, яке займає частинка i за весь час (особистий досвід);

g – найкраще положення, знайдене всіма частинками у рої (глобальний досвід);

x_i – поточне положення частинки i.

Оновлення положення. Нове положення частинки визначається як її попереднє положення, зміщене на величину оновленої швидкості:

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

що дозволяє частинці рухатися в напрямку, визначеному її швидкістю.

Оцінка фітнес-функції. Для кожної частинки обчислюється значення фітнес-функції (функціонал J) в її новому положенні. Якщо нове положення частинки є кращим за попереднє найкраще особисте положення, то воно оновлюється. Якщо положення частинки є кращим за глобальне найкраще положення (колективний досвід), то воно також оновлюється.

Ці етапи ітеративно повторюються (від оновлення швидкості до оцінки фітнес функції) доти, поки не буде досягнуто певного критерію зупинки, наприклад, заданої кількості ітерацій або певного значення фітнес-функції.

Переваги методу:

– простота реалізації та налаштування параметрів;

– здатність працювати з нелінійними функціями та задачами великої розмірності;

– ефективність у випадках, коли градієнт цільової функції недоступний або важко обчислюваний.

Недоліки методу:

– схильність до передчасної збіжності, особливо якщо параметри налаштовані неправильно;

– можливість застрягти в локальному мінімумі у складних багатовимірних задачах;

– вибір параметрів може значно впливати на ефективність алгоритму.

2.1.8 Машинне навчання

Методи машинного навчання, зокрема нейронні мережі, навчання з підкріпленням та фізико-інформовані нейронні мережі (PINN, Physics-Informed Neural Networks), мають широке застосування в задачах оптимального керування. Ці методи дозволяють знаходити оптимальні стратегії управління в умовах невизначеності та складної динаміки системи. Оптимальне керування передбачає пошук такого управління, яке мінімізує або максимізує певний цільовий функціонал, забезпечуючи виконання динамічних і функціональних обмежень. У цьому контексті машинне навчання допомагає створювати алгоритми, які навчаються на основі наявних даних або в процесі експериментів, підбираючи оптимальні керування для досягнення поставленої мети [19].

Одним із ключових підходів у машинному навчанні для оптимального керування є навчання з підкріпленням, де агент навчається приймати рішення на основі винагороди, отриманої за кожну дію. Навчання з підкріпленням дозволяє агенту самостійно знаходити стратегії управління, поступово вдосконалюючи їх шляхом експериментів і накопичення досвіду. Однак у випадках, коли динаміка системи є складною або частково невідомою, фізико-інформовані

нейронні мережі (PINN) надають унікальні можливості для інтеграції фізичних законів у процес навчання.

PINN поєднують переваги глибокого навчання з відомими фізичними законами, які описують динаміку системи. Це дозволяє ефективно розв'язувати диференціальні рівняння, зокрема рівняння в частинних похідних, які описують поведінку системи, і забезпечувати точні моделі навіть за відсутності великих обсягів даних. Наприклад, PINN дозволяють працювати з задачами, де динаміка системи частково невідома або змінюється з часом, а класичні методи оптимізації виявляються малоефективними через багатовимірність і нелінійність системи.

Нейронні мережі та PINN можуть використовуватися для апроксимації цільової функції або обмежень, а також для побудови моделей складних систем. Вони здатні навчатися на основі реальних даних, враховуючи при цьому фізичні закони, що значно підвищує точність і адаптивність моделювання. Зокрема, PINN дозволяють створювати наближені моделі динаміки, які відповідають фізичним обмеженням і можуть використовуватися для оптимізації управління. Це особливо корисно у високовимірних задачах, де традиційні чисельні методи є обчислювально затратними.

Окрім того, використання PINN допомагає прискорити обчислення, зокрема у випадках, коли оцінка цільової функції є складною або потребує великих обсягів даних. Завдяки інтеграції фізичних законів, PINN можуть значно зменшити обчислювальні витрати, що відкриває нові можливості для вирішення задач оптимального керування в реальному часі.

Машинне навчання, у поєднанні з підходами PINN, дозволяє створювати адаптивні стратегії управління, які змінюються в режимі реального часу, враховуючи поточний стан системи або зовнішні умови. Це є особливо важливим для динамічних задач, у яких традиційні методи оптимального керування, побудовані на жорстко заданих параметрах, не можуть забезпечити адекватну реакцію на змінні фактори середовища.

2.2 Застосування нейромереж PINN при розв'язку задачі портфелю Мертона

При розв'язку задачі (1.1) – (1.2), її (задачу) можна звести до рівняння Гамільтона-Якобі-Беллмана (HJB):

$$\rho \cdot V^*(t,W_t) = \max_{\pi_t, c_t}\left[\frac{\partial V^*}{\partial t} + \frac{\partial V^*}{\partial W}\left(\left(\pi_t(\mu - r) + r\right)W_t - c_t\right) + \frac{\partial^2 V^*}{\partial W^2}\cdot\frac{\pi_t^2\sigma^2 W_t^2}{2} + \frac{c_t^{1-\gamma}}{1-\gamma}\right],$$

де $\rho \geq 0$ – дисконт для очікуваної корисності;

$V^* = V^*(W,t)$ – оптимальна функція цінності.

Це рівняння HJB можна спростити далі до диференційного рівняння в частинних похідних:

$$\frac{\partial V^*}{\partial t} - \frac{(\mu - r)^2}{2\sigma^2}\cdot\frac{\left(\frac{\partial V^*}{\partial W_t}\right)^2}{\frac{\partial^2 V^*}{\partial W_t^2}} + \frac{\partial V^*}{\partial W_t}\cdot r\cdot W_t + \frac{\gamma}{1-\gamma}\cdot\left(\frac{\partial V^*}{\partial W_t}\right)^{\frac{\gamma-1}{\gamma}} = \rho\cdot V^*, \qquad (2.7)$$

$$V^*(T,W_T) = \frac{W_T^{1-\gamma}}{1-\gamma}, \qquad (2.8)$$

де $\pi_t^* = \dfrac{-\dfrac{\partial V^*}{\partial W_t}\cdot(\mu - r)}{\dfrac{\partial^2 V^*}{\partial W_t^2}\cdot\sigma^2\cdot W_t}$ – оптимальне частка інвестицій, що піде на ризиковий актив;

$c_t^* = \left(\dfrac{\partial V^*}{\partial W_t}\right)^{-\frac{1}{\gamma}}$ – оптимальне споживання інвестицій.

Для пошуку розв'язку системи (2.7) – (2.8) ми застосуємо нейромережі, а

саме фізико-інформовані нейромережі (PINN), які є популярними останніми роками.

Як відомо, під час тренування нейромереж, одним із останніх кроків однієї епохи тренування є зміна вагових коефіцієнтів на основі втрат. Часто значення цих втрат визначається як квадратичне відхилення між отриманим результатом і шуканим. У випадку із PINN функція втрат обчислюється так само: різниця між отриманим і очікуваним результатом, тільки очікуваний результат є значення диференційного рівняння в частинних похідних в конкретній точці, точки із крайових умов і, можливо, певні точки в середині області пошуку розв'язку, які були знайдені раніше, наприклад, чисельним методом.

Для нашого рівняння ми маємо тільки крайову умову і саме диференційне рівняння. Тобто втрата обчислюється так:

$$Loss = MSE\left(HJB_{residual}\right) + MSE\left(BC_{residual}\right), \tag{2.9}$$

де $HJB_{residual}$ – різниця в рівнянні (2.7);

MSE – середнє квадратичне відхилення;

$BC_{residual}$ – різниця в крайовій умові (2.8).

Цей метод є ефективнішим в порівнянні з чисельними методами такими як сітковий чи метод спряжених градієнтів, оскільки він пропонує гнучкіший метод для вирішення нелінійних задач або складних фізичних систем. Він є простим у побудові та імплементації. Він зручний для застосування в багатовимірних системах, оскільки тільки лінійно росте навантаження на обчислення, на відміну від багатьох чисельних методів.

Серед недоліків є те, що він не обов'язково буде сходитися до оптимального рішення, потребує, можливо, правильного підбору нейромережі для конкретної задачі.

Висновки за розділом 2

В даному розділі були виконано:

– огляд методів вирішення задачі оптимального управління, включаючи, метод роб часток, сіткові методи, машинне навчання тощо;

– дослідження обраного підходу до рішення задачі, а саме машинне навчання із PINN.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Екосистема універсальної мови програмування Python

Python — одна з найпопулярніших універсальних мов програмування, яка відзначається простотою синтаксису, широким спектром застосування та величезною екосистемою інструментів і бібліотек. Ця мова широко використовується в наукових дослідженнях, машинному навчанні, розробці веб-додатків, автоматизації завдань, аналізі даних та багатьох інших галузях. Вона є мультипарадигмальною, підтримуючи об'єктно-орієнтоване, функціональне та процедурне програмування, що дозволяє розробникам обирати найбільш підходящий стиль програмування для конкретних задач.

Однією з ключових переваг Python є потужна екосистема бібліотек. Наприклад, NumPy є базовою бібліотекою для чисельних обчислень, яка надає багатовимірні масиви та функції для виконання операцій над ними. NumPy забезпечує високу продуктивність завдяки використанню оптимізованого коду на рівні C, що робить її основою для багатьох інших бібліотек.

Для роботи з машинним навчанням та глибинними нейронними мережами широко застосовується бібліотека PyTorch, яка забезпечує високий рівень гнучкості та інтерактивності. PyTorch дозволяє легко створювати, тренувати та тестувати складні моделі, а також підтримує автоматичне обчислення похідних, що є критично важливим для оптимізації моделей. Її інтеграція з апаратним прискоренням, таким як GPU, дозволяє виконувати розрахунки значно швидше.

Графічна бібліотека Matplotlib є одним із найпотужніших інструментів для візуалізації даних. Вона дозволяє створювати різноманітні типи графіків і діаграм, включаючи лінійні, стовпчикові, точкові та тривимірні графіки. Matplotlib також підтримує високий рівень налаштувань, що дозволяє створювати візуалізації, адаптовані до конкретних потреб користувачів.

Разом ці бібліотеки, як і багато інших, формують потужну екосистему Python, яка забезпечує розробників інструментами для ефективного розв’язання

складних завдань у різних галузях. Завдяки активній спільноті та великій кількості доступних ресурсів Python залишається ідеальним вибором для як новачків, так і досвідчених розробників.

3.2 Алгоритм розв’язання задачі портфеля Мертона

Задача оптимального керування, яка визначена формулами (1.1) – (1.2) і спрощена до часткового диференційного рівняння з кінцевою умово (2.7) – (2.8) буде вирішуватися шляхом безпосереднього застосування нейромереж типу PINN.

Розглянемо по-кроковий план вирішення даної задачі портфеля Мертона.

Крок 1. Визначити вхідні параметри до задачі. До цих параметрів належать такі:

– волатильність ризикового активу;

– очікувана дохідність ризикового активу;

– постійна відсоткова ставка без-ризикового активу;

– дисконт для очікуваної корисності;

– коефіцієнт відносного ухилення від ризику.

Крок 2. Побудувати нейромережу на основі PINN.

Крок 3. Натренувати мережу за допомогою оптимізатора Adam, де втрата визначається формулою (2.9).

Крок 4. Провести аналіз натренованої моделі. Для кращого розуміння результатів, провести порівняння із аналітичним точним розв'язком.

Із виконанням цим кроків, задача буде вважатися виконаною.

3.3 Опис програми

Програма написана мовою програмування Python із застосування бібліо-

тек Numpy для роботи із масивами даних, Pytorch для розробки нейромережі і Matplotlib для зручної побудови графіків.

Код програми поділяється на декілька частин.

В першій визначаються вхідні параметри (крок 1 плану вирішення задачі).

Далі визначаються функції для безпосереднього обчислення значень рівнянь (2.7) – (2.9). Серед них є обчислення крайової умови, функція очікуваної корисності інвестицій, обчислення різниці за рівняння (2.7).

Згодом йде визначення нейромережі (крок 2 плану), показаний на рис. 3.1. Як видно із рисунка, нейромережа складається із декількох шарів, що включають лінійне перетворення та нелінійне (гіперболічний тангенс). На рисунку 3.2 зображено повну модель нейромережі (вже натренованої).

```
class HJBNet(nn.Module):
    def __init__(self):
        super(HJBNet, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(2, 512),  # Input layer: wealth W and time t
            nn.Tanh(),
            nn.Linear(512, 1024),
            nn.Tanh(),
            nn.Linear(1024, 512),
            nn.Tanh(),
            nn.Linear(512, 1),
        )

    def forward(self, W, t):
        inputs = torch.cat([W, t], dim=1)
        V = self.net(inputs)
        return V
```

Рисунок 3.1 – Модель нейромережі

Наступними блоками коду йдуть безпосереднє навчання моделі (крок 3 плану). Навчання моделі відбувається за допомогою оптимізатора Adam. Безпо-

середнє тренування розділене на декілька блоків: навчання суто на граничній умові, щоб ця крайова умова була наближено якомога найкраще; гранична умова і невеликі відрізки за часом від крайової умови до нуля. Це зроблено, на сам перед, щоб спробувати зберегти збіжність. Ідея, що тренування має бути точним на граничній умові і наближене якомога краще на невеликих проміжках від крайової умови послідовно.

Далі в програмі йде безпосереднє порівняння результатів натренованої моделі із аналітичним розв’язком (крок 4 плану).

Таким чином, програма відповідає чітко визначеному плану з п. 3.2.

Варто зазначити, що в ході експерименту, було використано потужності графічної карти, які теж враховані в програмі, якщо такий прилад доступний для використання.

Висновки за розділом 3

В розділі 3 було розглянуто детально середовище виконання практичної частини роботи, що включає опис алгоритму розв’язку задачі і розробку програмного забезпечення під алгоритм розв’язку задачі.

Програма написана на універсальній мові програмування Python із застосування широковідомих і потужних бібліотек для роботи із даними, побудови і навчання нейромереж і аналізу результатів шляхом їх візуалізації.

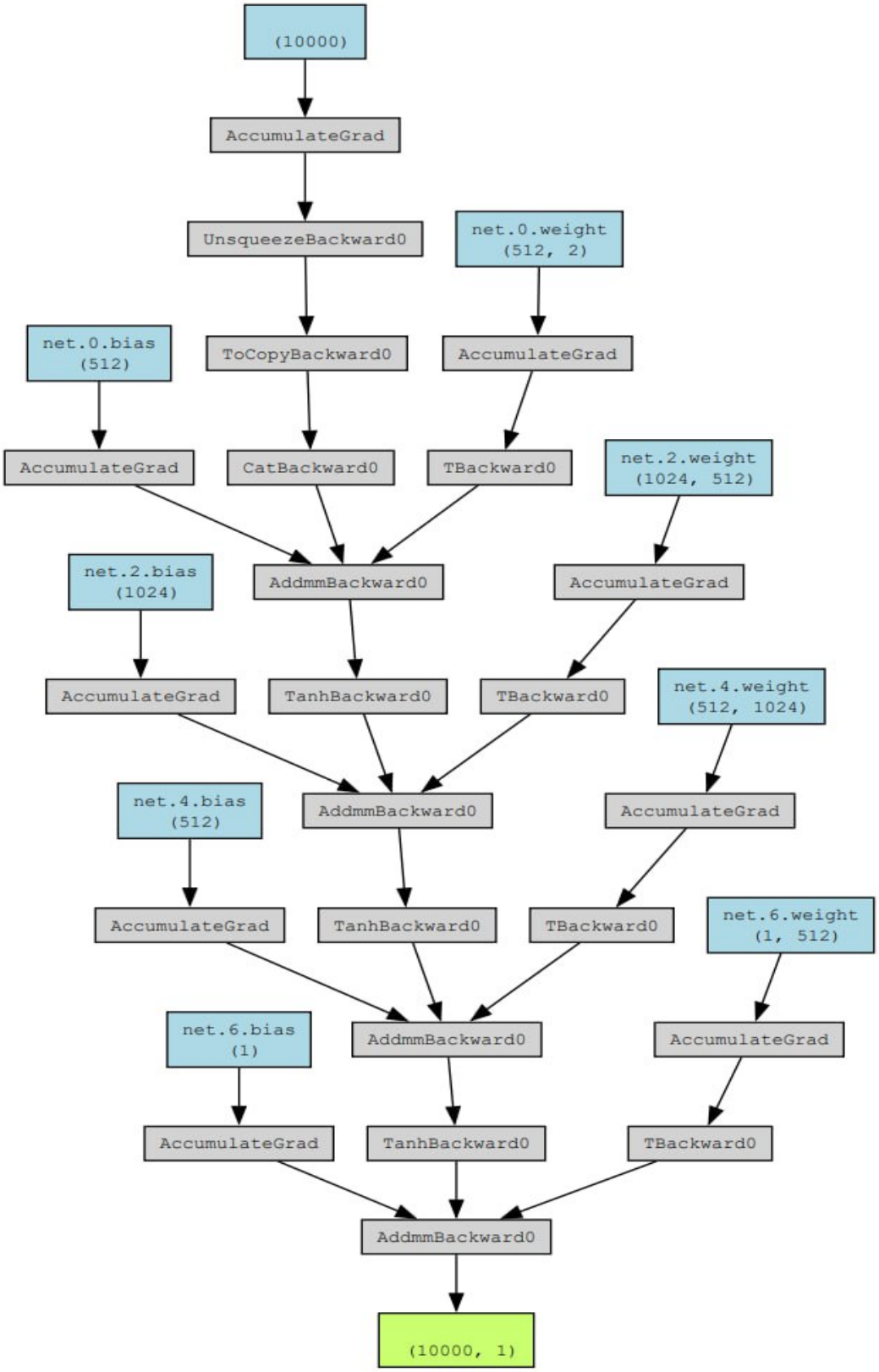

Рисунок 3.2 – Модель нейромережі

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Навчання моделі

Вхідні дані для тренування моделі обрані наступним чином:

– волатильність ризикового активу – 0.2;

– очікувана дохідність ризикового активу – 0.07;

– постійна відсоткова ставка без-ризикового активу – 0.03;

– дисконт для очікуваної корисності – 0.04;

– коефіцієнт відносного ухилення від ризику – 0.5;

– інвестиційний капітал – розглядається інтервал $[0.1;1]$;

– часовий інтервал – розглядається інтервал $[0;1]$.

Тренування моделі, визначеної в пункті 3.3, розділене на декілька етапів. Розглянемо їх детальніше.

Етап 1. Тренування моделі на крайовій умові. На цьому етапі, модель тренувалася 13000 епох, допоки не досягла втрати в 0.000097 (менше – краще).

Етап 2. Тренування на крайовій умові і додатково невеликий відступ від крайової умови в часовому вимірі. Це тренування відбувається протягом 20000 епох. Втрата наприкінці досягла 0.00055.

Етап 3. До етапу 2 додається іще невеликий проміжок на якому тренується модель. На цьому етапі проходить 2000 епох.

Етап 4. Повторюється Етап 3 із новими проміжками допоки весь часовий інтервал не буде розглянутий.

Таким чином, модель поступово наближається до цільової функції, що дозволяє гнучкіше керування із точністю і швидкістю навчання. Наприкінці всього навчання, модель має втрату навчання 0.0028.

4.2 Оцінка якості моделі

По закінченню навчання моделі, було проведено аналіз якості моделі шляхом безпосереднього порівняння отриманих результатів із точним розв’язком задачі. Розглянемо результати.

Першим елементом порівняння є цільова функція із задачі HJB. На рис. 4.1 наведено абсолютну помилку по модулю в значеннях цільової функції в моделі та точного розв’язку (холодніше – краще). Як видно, найбільша різниця сягає 0.045, що є доволі низькою помилкою, при тому, загальний розв’язок виглядає доволі якісним, оскільки помилка росте лише при наближенні до крайових значень інтервалу інвестицій.

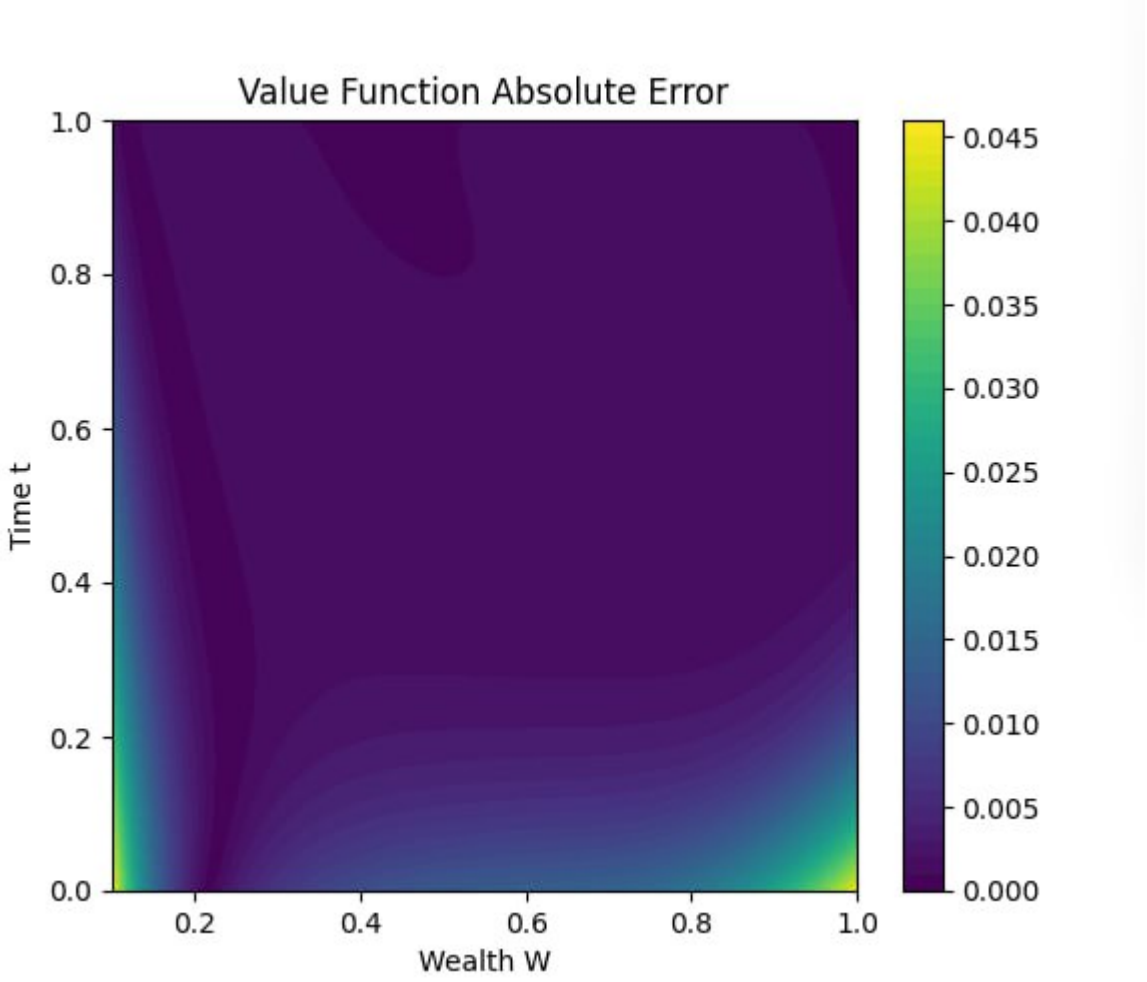

Рисунок 4.1 – Абсолютна помилка цільової функції моделі в порівнянні із точним розв’язком (холодніше – краще)

На рисунку 4.2 наведено фактичні значення цільових функцій моделі і точного розв'язку. По цьому рисунку, вони дуже подібні.

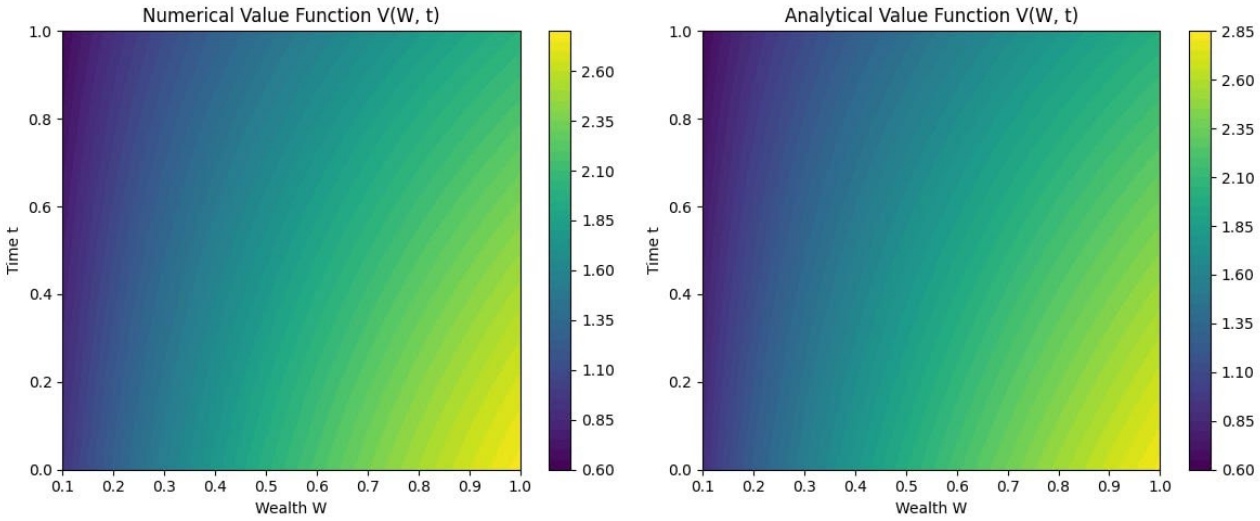

Рисунок 4.2 – Значення цільової функції моделі (ліворуч)
і аналітичного розв’язку (праворуч)

Наступним елементом порівняння є оптимальне споживання інвестицій $c^*(t,W)$. На рисунку 4.3 Наведено абсолютну помилку між точним розв’язком і результатом моделі. Абсолютна похибка не перевищує 0.225. Загально видно, що помилка росте лише близько до крайових умов. На рисунку 4.4 наведено фактичні результати моделі та точного розв’язку для оптимального споживання інвестицій.

Останнім елементом порівняння є оптимальна частка інвестицій, що піде на ризиковий актив $\pi^*(t,W)$. Результати наведені на рис. 4.5. З нього видно, що результат моделі сильно не співпадає із точним розв'язком.

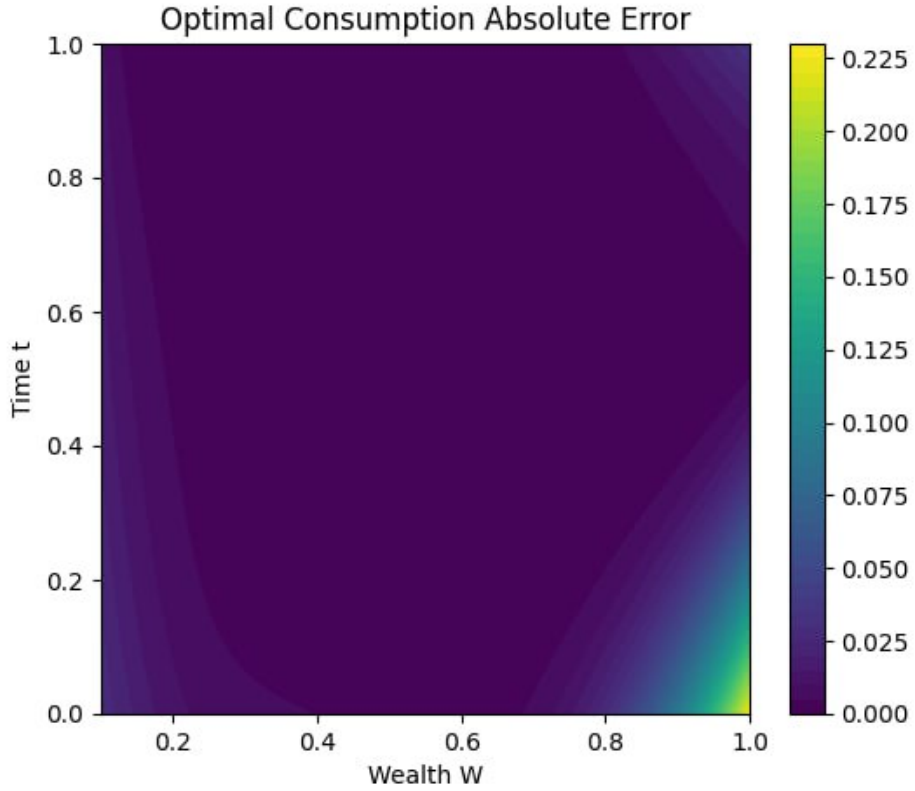

Рисунок 4.3 – Абсолютна помилка функції оптимального споживання інвестицій моделі в порівнянні із точним розв’язком (холодніше – краще)

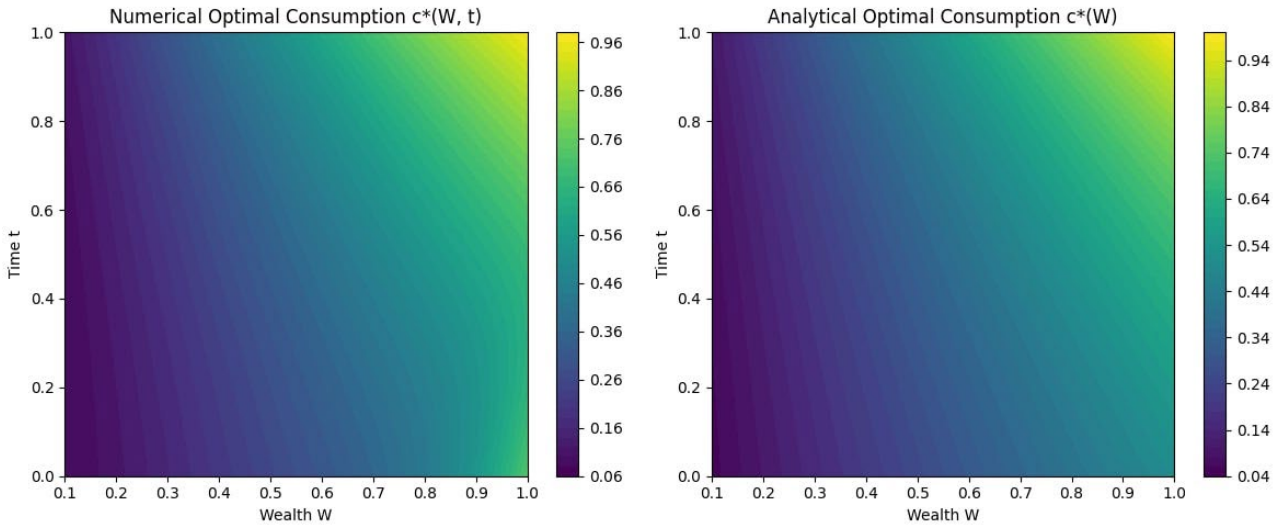

Рисунок 4.4 – Значення оптимального споживання інвестицій моделі (ліворуч) і точного розв’язку (праворуч)

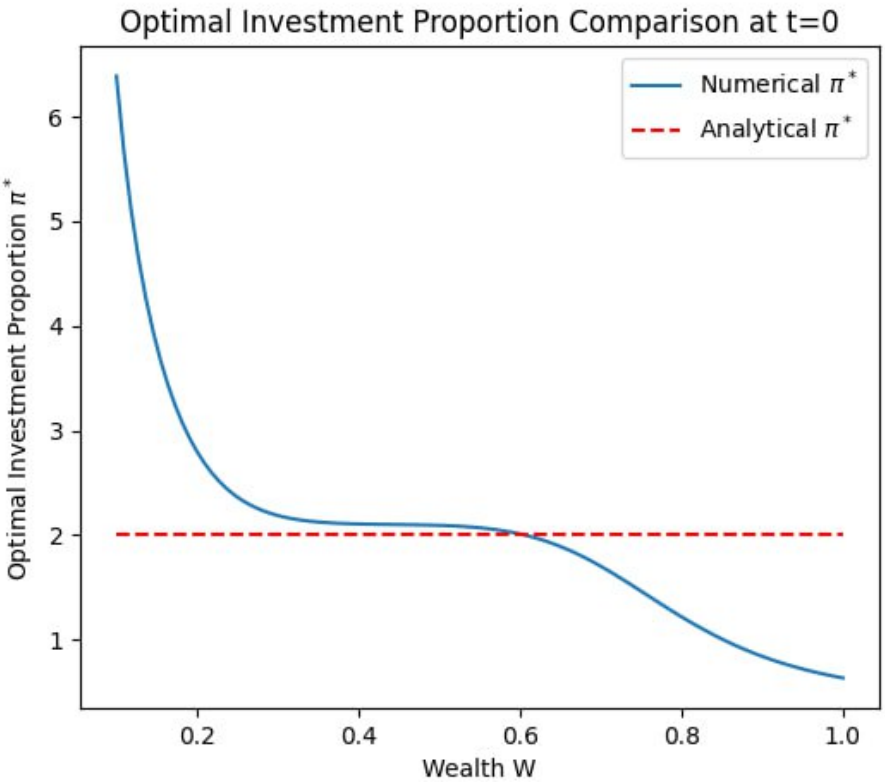

Рисунок 4.5 – Графік зміни оптимальної частки інвестицій ризикового активу для моделі (синя лінія) і точного розв’язку (червона штрихована лінія)

Висновки за розділом 4

В даному розділі було проведено аналіз результатів обчислювального експерименту. Із результатів видно, що модель натренована загально добре, але є місця, де сама модель є дуже не точно.

З цих результатів можна зробити висновки, що ідея PINN заслуговує уваги, особливо для вирішення задач із частковими диференційними рівняннями, зокрема тими, де аналітичний розв’язок дуже важко обчислити. Сама по собі технологія PINN дозволяє швидко вирішувати такі задачі, хоча недоліком є те, що не відома форма шуканої функції і збіжність нейромережі PINN не доведена, тобто вона не завжди сходиться.

ВИСНОВКИ

Результати отримані у ході проведення дослідженого, описаного у кваліфікаційній роботі показують, що при розв’язку оптимізаційної задачі, зокрема задачі оптимального управління, де є диференційні рівняння в частинних похідних, застосування нейромереж типу PINN для апроксимації шуканої функції дає гарні результати. При цьому не потрібні великі обчислювальні можливості, як при застосуванні чисельних методів.

Даний тип нейромереж можна застосовувати для вирішення диференційних рівнянь в частинних похідних, в особливості ті, які не мають аналітичного розв’язку. Такі рівняння часто виникають у фізиці (динаміка рідин), біології тощо. Тобто сфера застосування PINN є широкою.

Ця робота є науково значуща, оскільки вона досліджує фізико-інформовані нейромережі для наближення функції, що описана диференційним рівняння в частинних похідних, що є широко затребуваною темою дослідження, оскільки сучасні чисельні методи потребують велику кількість обчислювань, що затримує пошук розв’язку, особливо на фінансовому ринку, де динаміка швидко змінюється.

Дослідження застосування нейромережі PINN і далі є доцільною темою, оскільки є багато моментів, пов’язані із цим типом нейромережі, які потребують глибшого дослідження і покращення: збіжність нейромережі, адаптація нейромережі під загальну форму тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. HJB Equation and Merton’s Portfolio Problem. URL: https://stanford.edu/~ashlearn/RLForFinanceBook/MertonPortfolio.pdf (дата звернення: 08.11.2024).

2. Risk-averse Merton’s Portfolio Problem. URL: https://www.sciencedirect.com/science/article/pii/S2405896316306607?ref=cra_js_challenge&fr=RR-1 (дата звернення: 08.11.2024).

3. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. URL: https://arxiv.org/abs/1711.10561 (дата звернення: 08.11.2024).

4. HJB equation. URL: https://liberzon.csl.illinois.edu/teaching/cvoc/node95.html (дата звернення: 08.11.2024).

5. Petryshyn Andrii. Neural Network Solution For Partial Differential Equation. Information Technology and Implementation (Satellite) (Kyiv, 21 November 2024). 2024. Pp 97 – 98.

6. Єсілевський В. С, Петришин А. Ю. Розв’язок часткових диференційних рівнянь із застосуванням нейромереж. *Інформаційні системи та технології ІСТ-2024: зб. матеріалів 13-ї Міжнародної науково-технічної конференції* (м. Харків 26 – 28 листопада 2024р.). Частина 2. Харків, 2024. С. 60 – 61.

7. Sharpe Ratio. URL: https://corporatefinanceinstitute.com/resources/career-map/sell-side/risk-management/sharpe-ratio-definition-formula/ (дата звернення: 05.01.2025).

8. Investments: Lecture 3 Mean-variance theory. URL: https://dybfin.wustl.edu/teaching/inv/slides/invl3.html (дата звернення: 05.01.2025).

9. Black-Scholes Model: What It Is, How It Works, and Options Formula. URL: https://www.investopedia.com/terms/b/blackscholes.asp (дата звернення: 05.01.2025).

10. Value at Risk (VaR). URL: https://corporatefinanceinstitute.com/resources/career-map/sell-side/risk-management/value-at-risk-var/ (дата звернення: 05.01.2025).

11. Asset/Liability Management: Definition, Meaning, and Strategies. URL: https://www.investopedia.com/terms/a/asset-liabilitymanagement.asp (дата звернення: 05.01.2025).

12. Pontryagin's maximum principle. URL: https://homes.cs.washington.edu /~todorov/courses/amath579/Maximum.pdf (дата звернення: 08.11.2024).

13. Lecture 3 | MIT 6.832 (Underactuated Robotics), Spring 2020 | Dynamic Programming I. URL: https://www.youtube.com/watch?v=GPvw92IKO44&list= PLkx8KyIQkMfU5szP43GlE_S1QGSPQfL9s&index=3 (дата звернення: 08.11.2024).

14. A multigrid method for constrained optimal control problems. URL: https://www.sciencedirect.com/science/article/pii/S0377042711001786 (дата звернення: 08.11.2024).

15. The Finite Element Method (FEM). URL: https://www.comsol.com /multiphysics/finite-element-method (дата звернення: 08.11.2024).

16. Gradient Descent Methods on Optimal Control Problems. URL: https://dcn.nat.fau.eu/wp-content/uploads/25.DyCon-Toolbox.pdf (дата звернення: 08.11.2024).

17. A modified genetic algorithm for optimal control problems. URL: https://www.sciencedirect.com/science/article/pii/089812219290094X (дата звернення: 08.11.2024).

18. Two Optimal Control Strategies Based on Particle Swarm Optimization For a Hybrid Two-Tank System. URL: https://www.researchgate.net/publication/ 282857662_Two_Optimal_Control_Strategies_Based_on_Particle_Swarm_Optimization_For_a_Hybrid_Two-Tank_System (дата звернення: 08.11.2024).

19. Machine Learning and Optimal Control. URL: https://www.youtube.com/ watch?v=jZDh6Aulg30 (дата звернення: 08.11.2024).