

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)
Кафедра _____ Штучного інтелекту _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський) _____

Метод синтезу розв'язків математичних задач в аналітичному
вигляді з використанням алгоритму Proximal Policy Optimization
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-3 _____

Віктор НЕМЧЕНКО
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки _____

(код і повна назва спеціальності)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____

(повна назва освітньої програми)

Керівник _____ ас. Максим ПОЛІТ _____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____

(підпис)

Лариса ЧАЛА
(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Немченку Віктору Віталійовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Метод синтезу розв'язків математичних задач в аналітичному вигляді з використанням алгоритму Proximal Policy Optimization _____

затверджена наказом університету від 8 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 11 липня 2026 р.

3. Вихідні дані до роботи Науково-технічні публікації та дані Інтернет-джерел у сфері символічної регресії, навчання з підкріпленням (Reinforcement Learning) та алгоритмів оптимізації політики PPO, офіційна технічна документація фреймворків Stable-Baselines3 та sb3 contrib, бібліотеки Gymnasium, а також інструментів обчислювального аналізу NumPy, SciPy та Pandas, збалансований синтетичний датасет числових умов та коефіцієнтів для перевірки коректності аналітичного синтезу формули дискримінанта квадратного рівняння

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та постановка задачі

2) Теоретико-методологічні основи символічної регресії та методів навчання з підкріпленням

3) Проєктування та алгоритмічне моделювання компонентів середовища символічного синтезу

4) Практична реалізація системи автоматизації

5) Експериментальні дослідження та аналіз процесів навчання моделі

6) Рекомендації щодо подальшої роботи в даному напрямку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі	09.06.2026	виконано
3	Огляд літературних джерел	15.06.2026	виконано
4	Обґрунтування та проектування середовища символічного синтезу	15.06.2026–17.06.2026	виконано
5	Практична реалізація системи аналітичного символічного синтезу	17.06.2026–19.06.2026	виконано
6	Проведення експериментальних досліджень	20.06.2026–25.06.2026	виконано
7	Написання пояснювальної записки	26.06.2026–11.07.2026	виконано
8	Перевірка на академічний плагіат	12.07.2026	виконано
9	Нормоконтроль	12.07.2026	виконано
10	Підготовка презентації та доповіді	12.07.2026	виконано
11	Попередній захист	12.07.2026	виконано
12	Рецензування	12.07.2026	виконано
13	Захист перед ЕК	14.07.2026	

Дата видачі завдання 8 червня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____
(підпис)

ас. Максим ПОЛІТ
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 61 с., 6 рис., 1 дод., 21 джерело.

АНАЛІТИЧНИЙ ВИГЛЯД, ГЛИБОКЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ГРАДІЄНТ СТРАТЕГІЇ, ЗВОРОТНА ПОЛЬСЬКА НОТАЦІЯ, МАСКУВАННЯ ДІЙ, НЕЙРОННІ МЕРЕЖІ, СИМВОЛЬНА РЕГРЕСІЯ, СИНТЕЗ РОЗВ'ЯЗКІВ, ФУНКЦІЯ ВИНАГОРОДИ, ШТУЧНИЙ ІНТЕЛЕКТ, GYMNASIUM, MASKABLE PPO, NUMPY, PANDAS, PROXIMAL POLICY OPTIMIZATION, PYTHON, SB3-CONTRIB, SCIPY, STABLE-BASELINES3.

Об'єктом дослідження кваліфікаційної роботи є процес автоматизованого структурно-параметричного пошуку, виведення та синтезу нелінійних математичних виразів і функціональних залежностей в аналітичному вигляді на основі багатовимірних емпіричних наборів даних.

Метою роботи є розробка, реалізація та дослідження програмно-алгоритмічного комплексу символічної регресії на засадах глибокого навчання з підкріпленням, який використовує оптимізований маскований алгоритм Proximal Policy Optimization («Maskable PPO») для генерації синтаксично коректних математичних рівнянь мінімальної складності з високою точністю аналітичної апроксимації.

Методами дослідження є теоретичні підходи системного аналізу, теорія штучних нейронних мереж, концепції градієнтних методів оптимізації політики PPO, математичний апарат лінеаризації синтаксичних дерев через зворотну польську нотацію RPN та принципи багатofакторного формування функцій винагороди («Reward Shaping») й анти-блоат регуляризації. Практична складова базується на об'єктно-орієнтованому та векторизованому програмуванні мовою Python із використанням фреймворків «Gymnasium», «Stable-Baselines3» та «sb3-contrib».

ABSTRACT

Bachelor's thesis contains: 61 pp., 6 fig., 1 ann., 21 references.

ACTION MASKING, ANALYTICAL FORM, ARTIFICIAL INTELLIGENCE, DEEP REINFORCEMENT LEARNING, GYMNASIUM, MASKABLE PPO, NEURAL NETWORKS, NUMPY, PANDAS, POLICY GRADIENT, PROXIMAL POLICY OPTIMIZATION, PYTHON, REVERSE POLISH NOTATION, REWARD FUNCTION, SB3-CONTRIB, SCIPY, STABLE-BASELINES3, SYMBOLIC REGRESSION, SYNTHESIS OF SOLUTIONS.

The object of research of this qualification work is the process of automated structural-parametric search, derivation, and synthesis of non-linear mathematical expressions and functional dependencies in analytical form based on multidimensional empirical datasets.

The purpose of the work is the development, implementation, and research of a software-algorithmic complex for symbolic regression based on deep reinforcement learning, which uses an optimized Maskable Proximal Policy Optimization (Maskable PPO) algorithm to generate syntactically correct mathematical equations of minimal complexity with high accuracy of analytical approximation.

The research methods include theoretical approaches of systems analysis, the theory of artificial neural networks, concepts of policy gradient optimization methods (PPO), the mathematical apparatus of linearizing syntax trees via Reverse Polish Notation (RPN), and the principles of multi-factor reward shaping and anti-bloat regularization. The practical component is based on object-oriented and vectorized programming in Python using the Gymnasium, Stable-Baselines3, and sb3-contrib frameworks.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	9
Вступ.....	12
1 Аналіз предметної галузі та постановка задачі.....	14
1.1 Опис предметної галузі	14
1.2 Опис проблем та викликів.....	15
1.3 Огляд існуючих рішень	16
1.3.1 Eureqa (Genetic Programming)	16
1.3.2 AI Feynman (Neural-Symbolic Decomposition).....	17
1.3.3 FFX (Fast Function Extraction).....	18
1.3.4 SINDy (Sparse Identification of Non-linear Dynamics)	18
1.3.5 NeSym (Neural Symbolic Regression that Scales)	19
1.3.6 SymbolicGPT (Generative Pre-trained Transformer)	19
1.3.7 BSR (Bayesian Symbolic Regression)	20
1.3.8 DSR (Deep Symbolic Regression)	20
1.3.9 DSO (Deep Symbolic Optimization).....	21
1.4 Інноваційність та переваги розроблюваного рішення	22
1.5 Постановка задачі.....	23
2 Теоретико-методологічні основи символної регресії та методів навчання з підкріпленням	26
2.1 Концепція аналітичного синтезу математичних виразів у формі символної регресії	26
2.2 Математичний апарат базового алгоритму «Proximal Policy Optimization» (PPO).....	27
3 Проектування та алгоритмічне моделювання компонентів середовища символного синтезу.....	29
3.1 Представлення математичних виразів та обчислювальної логіки у вигляді стека «Зворотної польської нотації» за допомогою компонентів «RPNExpression» та «RPNEnvStack»	29

3.2	Формалізація алфавіту математичних токенів та примусова типізація структур даних через перерахування «SymbolTypes», іменовані кортежі «StackSymbol» та «SymbolSequence»	30
3.3	Формування векторизованого простору спостережень об'єкта стану середовища та інтеграція динамічних масок дій («Action Masking») для виключення синтаксично некоректних операцій агента	31
3.4	Математичне обґрунтування компонентів комплексної багатоцільової функції винагороди («Reward Shaping»).....	32
3.4.1	Оцінка функціональної близькості обчисленого виразу до цільового тренду на основі раціональної функції втрат («Rational Fitness Loss»).....	32
3.4.2	Механізми придушення структурного розростання виразів («Anti-bloat Complexity Penalty») та максимізації покриття змінних («Exploration Diversity Score»).....	34
3.4.3	Динамічне коригування та фільтрація винагороди залежно від етапу навчання, частоти повторення виразів та статистичних коефіцієнтів кореляції Пірсона і Спірмена	35
4	Практична реалізація системи автоматизації.....	37
4.1	Вибір та обґрунтування стека технологій	37
4.2	Алгоритм префіксної ін'єкції цільових токенів змінних на стек середовища («Prefix Injection Logic») для оптимізації початкових пошукових траєкторій	38
4.3	Реалізація кастомного рангового колбеку «CustomRankRewardCallback» для нелінійного сортування, нормалізації та нелінійного масштабування винагород елітних траєкторій у буфері «rollout»	39
4.4	Архітектурне конфігурування багатошарової актор-критик нейромережевої політики («MlpPolicy», «net_arch») та інтеграція з моделями «MaskablePPO»	41

4.5 Розробка програмного модуля генерації, санітизації та збалансованого розподілу синтетичного датасету за математичними умовами дискримінанта квадратного рівняння	42
5 Експериментальні дослідження та аналіз процесів навчання моделі.....	44
5.1 Конфігурація експериментального середовища, паралелізація обчислювальних потоків та нормалізація векторів	44
5.2 Дослідження впливу бар'єра елітності («excellence_threshold») та обмежень ранжування винагороди на стабілізацію функції цінності критика («Value Function»).....	45
5.3 Оцінка ефективності механізму збереження та дешифрації знайдених аналітичних рішень у контейнері виразів («RPNEexprContainer») з автоматичною трансляцією в інфіксну математичну форму	46
5.4 Результати експериментів	47
5.5 Аналіз результатів експериментів	48
5.6 Компроміс між автономністю та контрольованістю	52
6 Рекомендації щодо подальшої роботи в даному напрямку	54
Висновки	56
Перелік джерел посилання	58
Додаток А Відомість кваліфікаційної роботи	61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Анти-блоат регуляризація – метод обмеження надмірного структурного розростання довжини та складності математичного виразу без підвищення його апроксимаційної точності;

Зворотна польська нотація – форма запису математичних виразів, у якій оператори розташовуються після своїх операндів, що дозволяє обчислювати вирази лінійно без використання дужок;

Імунітет формул – механізм повного скасування штрафів за повторення виразів у буфері середовища у випадку досягнення ними максимальних показників статистичної кореляції;

Локальна оптимізація констант – процес чисельного пошуку оптимальних значень неперервних числових коефіцієнтів у структури виразу з використанням безградієнтних методів мінімізації функцій;

Маскування дій – архітектурний підхід у навчанні з підкріпленням, який динамічно блокує вибір синтаксично недопустимих токенів на кожному кроці генерації залежно від поточного стану стека;

Префіксна ін'єкція – евристичний метод примусового додавання визначеного базового токена на стек середовища під час ініціалізації епізоду для прискорення сходження політики агента;

Рангове нормалізування винагород – нелінійне перетворення значень винагороди у буфері роллаутів на основі їхнього порядкового рангу для посилення сигналу підкріплення та стабілізації Критика;

Символьна регресія – задача пошуку математичної залежності у вигляді аналітичної формули, яка найкраще описує заданий набір даних;

Синтез розв'язків – процес автоматичного конструювання цільової аналітичної структури математичного виразу з базового набору доступних токенів;

Функція винагороди – математичний критерій оцінки дій агента, що визначає цільову якість згенерованого виразу на основі його точності, складності та різноманітності;

Arcsinh – зворотний гіперболічний синус – математична функція, яка використовується для нелінійного згладжування великих значень помилки апроксимації у функції винагороди з метою захисту градієнта від вибуху;

DSO – Deep Symbolic Optimization – напрям глибокого навчання, орієнтований на пошук оптимальних символьних і математичних структур;

Gymnasium – стандартизований програмний інтерфейс для розробки, тестування та взаємодії із середовищами навчання з підкріпленням;

Maskable PPO – модифікація алгоритму Proximal Policy Optimization, що підтримує інтеграцію динамічних масок доступних дій на рівні нейромережевої політики;

MSE – Mean Squared Error – середньоквадратична помилка, метрика оцінки відхилення передбачених значень від цільового розподілу даних;

Nelder-Mead – алгоритм Нелдера-Міда – метод чисельної безградієнтної оптимізації багатовимірних функцій, що використовується для безперервного локального підбору числових констант у згенерованих формулах;

NumPy – бібліотека мови Python для високопродуктивних векторизованих обчислень та роботи з багатовимірними числовими масивами;

Pandas – бібліотека мови Python для аналізу, маніпулювання, фільтрації та структурування табличних даних;

PPO – Proximal Policy Optimization – градієнтний алгоритм навчання з підкріпленням, заснований на обмеженні кроку оновлення параметрів політики;

Python – високорівнева мова програмування, що використовується для розробки та навчання систем штучного інтелекту;

Reward Shaping – формування функції винагороди – метод проектування та комбінування багатофакторних джерел винагород і штрафів для прискорення навчання інтелектуального агента;

Rollout Buffer – буфер траєкторій – тимчасове сховище даних алгоритму PPO для накопичення кроків, дій, станів середовища та логів винагород агента під час ітерацій роллауту;

RL – Reinforcement Learning – навчання з підкріпленням, парадигма машинного навчання, де агент навчається приймати рішення шляхом взаємодії із середовищем;

RPN – Reverse Polish Notation – зворотна польська нотація, лінійна форма запису математичних операцій;

sb3-contrib – розширення бібліотеки Stable-Baselines3, що містить додаткові експериментальні алгоритми та механізми маскуванню дій;

SciPy – бібліотека мови Python для наукових і математичних обчислень, що містить інструменти для чисельної оптимізації функцій;

Stable-Baselines3 – бібліотека надійних реалізацій стандартних алгоритмів глибокого навчання з підкріпленням мовою Python.

ВСТУП

Тривалий час домінуючим підходом у сфері символної регресії залишалося генетичне програмування, яке здійснює пошук математичних формул за допомогою еволюційних евристик. Проте еволюційні алгоритми мають виражені фундаментальні обмеження, пов'язані з комбінаторним вибухом простору пошуку при ускладненні цільових залежностей та схильністю до неконтрольованого розростання розміру виразів. Як альтернатива виникла концепція глибокої символної оптимізації, яка формулює синтез формул як задачу навчання з підкріпленням. За такої постановки послідовне конструювання виразу виконується інтелектуальним агентом за допомогою градієнтних методів оптимізації політики, зокрема – алгоритму PPO. Незважаючи на вищу стабільність градієнтного сходження, традиційні архітектури навчання з підкріпленням залишаються обчислювально неефективними, оскільки витрачають значний ресурс на невалідні, синтаксично деструктивні послідовності символів.

Актуальність даної роботи зумовлена необхідністю подолання синтаксичного хаосу в просторі пошуку шляхом інтеграції динамічних обмежень безпосередньо в обчислювальне ядро нейромережевої політики агента. Об'єктивною підставою для виконання дослідження є потреба у створенні синтаксично захищеного середовища штучного інтелекту, яке повністю виключає генерацію математично некоректних формул на етапі вибору дій. Поєднання лінійної зворотної польської нотації (RPN) для представлення математичних дерев із технологією динамічного маскування дій («Action Masking») дозволяє заблокувати невалідні токени на основі поточного стану обчислювального стека та відомої арності операцій.

Метою кваліфікаційної роботи є дослідження градієнтних методів глибокого навчання з підкріпленням у задачах символної регресії та розробка стійкого програмно-алгоритмічного комплексу на базі алгоритму «Maskable PPO» для автоматизованого структурно-параметричного синтезу

синтаксично коректних нелінійних математичних виразів мінімальної складності. Головним завданням є пошук оптимального архітектурного балансу між гнучкістю дослідження математичного простору («exploration») та жорстким детермінованим синтаксичним контролем за станом обчислювального RPN-стека агента. Можливими сферами застосування розроблюваної системи є автоматизовані інструменти пошуку прихованих функціональних закономірностей у наукових та інженерних дослідженнях, системи аналізу експериментальних числових розподілів, а також аналітичні компоненти сучасних систем підтримки прийняття рішень..

Історичний аналіз розвитку символної регресії показує чітку еволюцію від жорстко заданих шаблонів класичної регресії до гнучких стохастичних графів. Перехід до парадигми навчання з підкріпленням із маскуванням дій знаменує новий етап, де агент оперує в суворо детермінованому синтаксичному коридорі.

Результати роботи мають як дослідницький, так і прикладний характер. Виконане дослідження включає всебічний аналіз сучасних архітектур глибокої символної оптимізації, програмну реалізацію експериментального маскованого середовища на базі фреймворку Gymnasium та інтеграцію чисельних методів локальної оптимізації констант. Отримані практичні результати у вигляді відносно стійкої нейромережевої моделі, яка пройшла певну валідацію на задачі автоматичного аналітичного синтезу формули дискримінанта квадратного тричлена, можуть бути безпосередньо використані під час подальших досліджень у сфері автономного генерації знань, автоматизації математичного моделювання складних нелінійних систем та розробки інтелектуальних інструментів комп'ютерної алгебри.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Розвиток методів штучного інтелекту та обчислювальної математики відкриває людям нові можливості для автоматизації процесів наукового пошуку. Одним із найбільш перспективних напрямів у цій сфері є автоматичний синтез розв'язків математичних задач в аналітичному (символьному) вигляді. На відміну від традиційного чисельного моделювання або класичних методів глибокого навчання, які створюють непрозорі моделі типу «чорна скринька», символьний синтез дозволяє знаходити явні математичні залежності. Результатом роботи таких систем є зрозумілі людині формули, які відображають фундаментальні закони досліджуваних процесів, забезпечують високу інтерпретованість та спроможність до екстраполяції за межі навчальної вибірки.

1.1 Опис предметної галузі

Останнім часом для вирішення задач символьного пошуку активно застосовуються підходи на основі навчання з підкріпленням («Reinforcement Learning», RL). У межах цієї парадигми процес побудови математичного виразу моделюється як послідовність дій інтелектуального агента, який взаємодіє зі спеціально спроектованим середовищем. Агент крок за кроком вибирає математичні елементи – змінні, константи та оператори, формуючи підсумкове «вираз-дерево» або лінійну послідовність у форматі зворотної польської нотації (RPN). За кожен згенерований вираз агент отримує зворотний зв'язок у вигляді винагороди, що визначається точністю відповідності синтезованої формули заданим математичним умовам чи набору даних.

Ключовим елементом такої архітектури є алгоритм оптимізації стратегії агента, серед яких особливе місце посідає «Proximal Policy Optimization» (PPO). Алгоритм PPO виступає математичним рушієм, який

керує оновленням параметрів нейромережевої моделі агента. Він забезпечує баланс між дослідженням нових варіантів побудови формул та експлуатацією вже знайдених успішних патернів. Завдяки інтеграції парадигми RL та алгоритму PPO стає можливим ефективне навчання моделі, яка здатна адаптуватися до різноманітних математичних бенчмарків, поступово накопичувати внутрішні закономірності побудови функцій та успішно синтезувати лаконічні й точні аналітичні розв'язки.

1.2 Опис проблем та викликів

Розробка та практичне впровадження систем аналітичного синтезу математичних розв'язків пов'язані з низкою серйозних теоретичних та обчислювальних складностей. Головним викликом є «комбінаторний вибух» простору пошуку. Оскільки кількість можливих комбінацій математичних символів зростає експоненціально зі збільшенням довжини виразу та кількості доступних операторів, прямий перебір або випадковий пошук стають абсолютно неефективними навіть для відносно простих залежностей.

Іншою важливою проблемою є надзвичайна дискретна чутливість математичних виразів. На відміну від безперервних просторів, де невеликі зміни параметрів призводять до пропорційних змін результату, у символьному просторі заміна навіть одного оператора (наприклад, знака додавання на віднімання або множення на ділення) може повністю спотворити зміст формули або зробити її синтаксично некоректною. Це створює середовище з високим рівнем математичного шуму, де випадкові помилки на початку генерації призводять до отримання беззмістовних чи некоректних результатів, таких як ділення на нуль або обчислення квадратного кореня від від'ємного числа.

Крім того, під час використання методів навчання з підкріпленням виникає проблема розрідженості винагороди та високої дисперсії градієнтів.

Агент часто отримує фінальну оцінку лише після того, як вираз побудовано повністю, що значно ускладнює визначення того, які саме проміжні кроки були правильними. У класичних градієнтних підходах RL це призводить до деструктивно великих кроків оновлення політики, через що модель втрачає стабільність, деградує та потрапляє в локальні екстремуми, генеруючи або тривіальні вирази, або надмірно ускладнені конструкції.

Саме для подолання цих викликів критично важливим є використання алгоритму PPO. Завдяки специфічній цільовій функції з механізмом обмеження («clipping»), PPO жорстко контролює масштаб оновлення стратегії агента на кожній ітерації навчання. Це запобігає руйнівним змінам параметрів нейронної мережі під впливом випадкових аномальних виразів та гарантує монотонне, стабільне покращення якості генерації аналітичних розв'язків.

1.3 Огляд існуючих рішень

Для огляду та аналізу вже існуючих рішень чи досліджень у сфері автоматичного виведення аналітичних залежностей необхідно розглянути ключові інструменти та алгоритми символічної регресії. Кожен із підходів базується на власній алгоритмічній парадигмі, що визначає його математичні можливості та межі застосування.

1.3.1 Eureqa (Genetic Programming)

Опис: класичний комерційний та науковий інструментарій для символічної регресії, побудований на принципах еволюційного моделювання.

Принцип роботи: простір пошуку аналітичних виразів досліджується за допомогою генетичного програмування, де математичні формули представляються у вигляді деревоподібних структур. Процес включає

випадкове схрещування, мутації структурних елементів та природний відбір найкращих кандидатів на основі точності апроксимації даних.

Переваги та обмеження: головною перевагою є здатність знаходити точні фізичні закони без жодних попередніх припущень щодо їхньої структури. Проте метод схильний до комбінаторного вибуху простору пошуку, часто застрягає в локальних екстремумах та генерує надмірно складні вирази, які важко інтерпретувати.

Масштабованість: масштабованість є низькою, оскільки з ростом кількості вхідних змінних або глибини формули час обчислень зростає експоненціально.

1.3.2 AI Feynman (Neural-Symbolic Decomposition)

Опис: алгоритм символічної регресії, розроблений для відкриття фундаментальних фізичних законів шляхом поєднання нейронних мереж та аналітичних спрощень.

Принцип роботи: використовує стратегію «розділяй і володарюй». Нейронна мережа спочатку навчається як сурогатна модель для інтерполяції даних, після чого аналізуються її математичні властивості, такі як розмірність, симетрія, роздільність змінних та періодичність, що дозволяє розбити одну складну задачу на кілька простіших підзадач.

Переваги та обмеження: метод надзвичайно ефективний для пошуку складних формул із багатьма змінними, якщо вони мають фізичні симетрії чи структурну роздільність. Обмеженням є жорстка прив'язка до специфічних математичних властивостей формул та необхідність великого обсягу чистих даних для точного попереднього навчання нейронної мережі.

Масштабованість: висока для структурно роздільних задач, але суттєво знижується на нероздільних математичних виразах.

1.3.3 FFX (Fast Function Extraction)

Опис: детермінований алгоритм символної регресії, орієнтований на надзвичайно швидку генерацію аналітичних моделей.

Принцип роботи: замість стохастичного пошуку алгоритм використовує техніку згенериування великої фіксованої бази нелінійних базисних функцій з наступним застосуванням розрідженої лінійної регресії (такої як еластична сітка або «Lasso») для вибору більш значущих членів.

Переваги та обмеження: основними перевагами є повна детермінованість, відсутність випадкового пошуку та колосальна швидкість обчислень. Обмеження полягають у неможливості генерації глибоких вкладених або раціональних функцій, які не були заздалегідь закладені в початковий лінійний базис.

Масштабованість: висока щодо кількості вхідних змінних, але низька щодо глибини та складності нелінійних зв'язків.

1.3.4 SINDy (Sparse Identification of Non-linear Dynamics)

Опис: спеціалізований алгоритм для ідентифікації нелінійних динамічних систем та диференціальних рівнянь на основі даних.

Принцип роботи: створюється велика бібліотека потенційних нелінійних функцій (поліноми, тригонометричні елементи, похідні), після чого застосовуються послідовні алгоритми розрідженої лінійної оптимізації для відбору мінімального набору членів, що описують динаміку системи.

Переваги та обмеження: метод ідеально підходить для фізичних процесів, забезпечуючи високу точність, фізичну коректність та стійкість до невеликих шумів. Головне обмеження полягає в необхідності точного ручного налаштування базисних функцій у бібліотеці, без яких система не зможе знайти правильне рішення.

Масштабованість: висока для динамічних систем помірної розмірності, якщо шукана залежність може бути лінійно комбінована з елементів бібліотеки.

1.3.5 NeSym (Neural Symbolic Regression that Scales)

Опис: сучасний підхід до символної регресії, що використовує великі нейромережеві архітектури для масштабування процесу пошуку.

Принцип роботи: модель базується на попередньому навчанні (pre-training) трансформера на величезному процедурно згенерованому наборі математичних виразів. Під час тестування модель приймає числовий набір точок даних і безпосередньо генерує символну формулу у вигляді послідовності токенів за один прохід.

Переваги та обмеження: головна перевага – миттєва генерація рішень під час тестування завдяки перенесенню обчислювальних витрат на етап попереднього навчання. Обмеження полягають у високій чутливості до даних, що виходять за межі розподілу навчання, та неможливості гнучкого налаштування під унікальні аномалії даних.

Масштабованість: висока за часом виконання на етапі інференсу, але обмежена за складністю формул архітектурою контекстного вікна моделі.

1.3.6 SymbolicGPT (Generative Pre-trained Transformer)

Опис: генеративна модель на основі архітектури Transformer («Decoder-only»), яка розглядає символну регресію як задачу мовного моделювання.

Принцип роботи: математичні вирази кодуються як послідовності у зворотній польській або префіксній нотації. Модель навчається передбачати наступний математичний токен (оператор, змінну або константу) на основі послідовності числових значень вхідних даних.

Переваги та обмеження: перевагою є використання потужних ймовірнісних механізмів сучасних мовних моделей для виявлення прихованих закономірностей у математичних структурах. Основним обмеженням є ризик генерації синтаксично некоректних або математично безглузвих виразів через відсутність жорстких логічних фільтрів під час генерації.

Масштабованість: добре масштабується за обсягом даних, проте потребує значних обчислювальних ресурсів для навчання та подальшої точної оптимізації констант.

1.3.7 BSR (Bayesian Symbolic Regression)

Опис: ймовірнісний підхід до символічної регресії, заснований на баєсівському виведенні та вибірці Марковських ланцюгів.

Принцип роботи: математичні дерева виразів розглядаються як випадкові об'єкти з певним апіорним розподілом ймовірностей, який карає надмірну складність структури. Пошук здійснюється за допомогою вибірки за алгоритмом Метрополіса-Гастінгса (MCMC), що дозволяє досліджувати простір формул та знаходити апостеріорний розподіл рішень.

Переваги та обмеження: надає не просто одну формулу, а ймовірнісну оцінку різних варіантів структурування моделі, що дозволяє кількісно оцінювати невизначеність даних. Недоліком є висока обчислювальна складність MCMC-семплювання та повільна збіжність алгоритму.

Масштабованість: низька масштабованість для великих наборів даних та великої кількості змінних через дискретну природу баєсівського пошуку на графах.

1.3.8 DSR (Deep Symbolic Regression)

Опис: метод символічної регресії, який використовує навчання з підкріпленням для керування та оптимізації процесу генерації математичних виразів.

Принцип роботи: рекурентна нейронна мережа виступає в ролі агента, який покроково генерує математичні вирази у вигляді префіксного дерева дій. Навчання відбувається за допомогою градієнта стратегії, орієнтованого на ризик («risk-seeking policy gradient»), що оптимізує модель на пошук найкращого екстремального виразу.

Переваги та обмеження: забезпечує стабільний пошук аналітичних рішень та інтеграцію синтаксичних обмежень безпосередньо під час генерації токенів. Обмеженням є можливість передчасної збіжності політики агента до локальних мінімумів (колапс стратегії), що зупиняє подальше дослідження простору пошуку.

Масштабованість: середня масштабованість, яка суттєво залежить від ефективності побудови функції винагороди та загальної довжини траєкторії генерації виразу.

1.3.9 DSO (Deep Symbolic Optimization)

Опис: уніфікований фреймворк глибокого навчання, що розширює концепцію DSR та поєднує градієнтні підходи навчання з підкріпленням із генетичним програмуванням.

Принцип роботи: генерація здійснюється нейромережевим агентом на базі RL, але найкращі згенеровані вирази додатково використовуються як початкова популяція («насіння») для локального генетичного алгоритму. Це створює синергетичний ефект глобального нейронного керування та локального еволюційного пошуку.

Переваги та обмеження: демонструє найвищу точність серед сучасних методів на складних математичних бенчмарках символічної регресії завдяки гібридизації підходів. Обмеженням є висока чутливість до гіперпараметрів оптимізації та складність балансування між нейромережевим агентом та еволюційним ядром.

Масштабованість: вища за класичне генетичне програмування, але все ще обмежена при роботі з системами дуже великої розмірності без попередньої декомпозиції простору дій.

1.4 Інноваційність та переваги розроблюваного рішення

Інноваційність запропонованого методу полягає в докорінній зміні парадигми пошуку аналітичних розв'язків: замість використання евристичних еволюційних алгоритмів або класичних методів підбору функцій, задача символічного синтезу інтегрується у фреймворк навчання з підкріпленням (RL) із застосуванням алгоритму «Proximal Policy Optimization» (PPO). Основне технологічне нововведення полягає в адаптації математичного апарату PPO, який традиційно використовується для управління безперервними фізичними системами, до дискретного комбінаторного простору генерації математичних виразів, представлених у вигляді послідовностей зворотньої польської нотації (RPN).

На відміну від традиційних підходів глибокої символічної регресії, які спираються на оптимізацію градієнта стратегії з орієнтацією на ризик, розроблюване рішення впроваджує механізм обмеження оновлення політики («clipping mechanism»), притаманний саме PPO. Це дозволяє повністю уникнути критичного явища колапсу стратегії навчання, коли нейромережевий агент передчасно втрачає здатність до дослідження простору рішень та застрягає в локальних математичних мінімумах. Такий підхід гарантує стабільне та монотонне покращення якості генерації формул протягом усього процесу оптимізації.

Іншим ключовим аспектом інноваційності є розробка кастомного RL-середовища з підтримкою динамічного маскування дій агента. Попередні генеративні архітектури та мовні моделі символічної регресії витрачають значні обчислювальні ресурси на створення, обчислення та подальше відсіювання синтаксично некоректних формул (наприклад, виразів із

незакритими дужками або подвійними операторами). Впроваджений у даній роботі механізм жорсткої граматичної фільтрації на кожному кроці генерації токена динамічно блокує завідомо помилкові дії в дискретному просторі дій, змушуючи всю обчислювальну потужність RL зосереджуватися виключно на пошуку математично змістовних аналітичних структур.

Переваги розроблюваного програмно-алгоритмічного комплексу над існуючими аналогами можна звести до кількох фундаментальних аспектів:

- висока стабільність навчання: завдяки математичним обмеженням цільової функції PPO, процес пошуку аналітичних законів захищений від різких коливань градієнтів, що забезпечує надійне відновлення формул навіть за наявності шумів у вхідних даних;

- оптимальне співвідношення точності та складності: використання багатофакторної функції винагороди на засадах Парето-оптимальності дозволяє агенту самостійно балансувати між мінімізацією похибки апроксимації та лаконічністю виразу, запобігаючи неконтрольованому структурному розростанню формули;

- ефективність дослідження простору дій: на відміну від генетичного програмування, яке виконує випадкові мутації та схрещування дерев виразів, RL-агент на базі PPO накопичує узагальнений досвід та формує структурну «математичну інтуїцію», що суттєво пришвидшує збіжність алгоритму під час роботи зі складними вкладеними структурами, такими як повний аналітичний синтез формули дискримінанту та коренів квадратного рівняння.

1.5 Постановка задачі

Головною метою цієї кваліфікаційної роботи є розробка, реалізація та дослідження методу автоматичного синтезу розв'язків математичних задач

в аналітичному вигляді на основі парадигми навчання з підкріпленням (RL) із застосуванням алгоритму оптимізації проксимальної політики (PPO).

Об'єктом дослідження є процеси автоматичного виведення аналітичних залежностей, фізичних законів та математичних формул за допомогою інтелектуальних комп'ютерних систем. Предметом дослідження є моделі глибокого навчання з підкріпленням та алгоритми PPO, пристосовані для оптимізації послідовностей дій у дискретних комбінаторних просторах пошуку.

Формально задача символічного синтезу в межах роботи розглядається як марковський процес прийняття рішень (MDP). Нехай задано числовий набір даних або опис математичної системи, а також фіксований базис операцій, який складається зі змінних, констант та математичних операторів. Нейромережевий агент RL повинен на кожному дискретному кроці обирати токен із базису для формування математичного виразу у вигляді послідовності зворотньої польської нотації (RPN). Задача зводиться до знаходження такої оптимальної стратегії генерації за допомогою градієнтних методів PPO, яка максимізує очікувану багатofакторну винагороду, що відображає точність апроксимації та структурну лаконічність знайденої формули.

Для досягнення поставленої мети та розв'язання глобальної науково-практичної задачі необхідно виконати такі завдання:

- спроектувати та програмно формалізувати кастомне RL-середовище на базі фреймворку «Gymnasium», яке трансформує процес генерації математичних виразів у послідовність кроків інтелектуального агента;
- реалізувати алгоритм «Proximal Policy Optimization» за допомогою бібліотеки «Stable-Baselines3», адаптувавши його математичну цільову функцію з механізмом обмеження («clipping») для стабілізації оновлення параметрів політики в дискретному просторі математичних токенів;
- розробити та впровадити алгоритм динамічного маскування недопустимих дій на кожному кроці генерації, що спирається на жорсткі

правила математичної граматики та повністю блокує створення синтаксично некоректних формул;

- розробити багатофакторну функцію винагороди на засадах Парето-оптимальності, яка математично збалансує заохочення за мінімізацію похибки та штрафи за надмірну довжину чи складність виразу;

- інтегрувати аналітичне ядро на базі бібліотеки «SymPy» для автоматичного фінального спрощення, розкриття дужок та символної перевірки згенерованих розв'язків;

- провести комплексне експериментальне дослідження розробленого комплексу на наборі математичних бенчмарків, включаючи повний аналітичний синтез формули дискримінанту та коренів квадратного рівняння, з подальшим аналізом швидкості збіжності та стійкості алгоритму PPO.

2 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СИМВОЛЬНОЇ РЕГРЕСІЇ ТА МЕТОДІВ НАВЧАННЯ З ПІД КРІПЛЕННЯМ

2.1 Концепція аналітичного синтезу математичних виразів у формі символічної регресії

Аналітичний синтез математичних виразів у формі символічної регресії є підходом, спрямованим на автоматичний пошук функціональних залежностей у масивах числових даних. На відміну від класичних методів регресійного аналізу, де структура математичної моделі задається дослідником заздалегідь, а оптимізації підлягають лише її числові коефіцієнти, символічна регресія одночасно шукає як оптимальну структуру виразу, так і значення його внутрішніх констант.

Основна суть цієї концепції полягає у відображенні вхідних змінних на цільову величину за допомогою скінченного набору математичних токенів, які включають змінні, математичні оператори та константи. У межах досліджуваного методу як цільова залежність розглядається обчислення дискримінанта квадратного рівняння на основі числових значень коефіцієнтів. Простір пошуку формується з дискретних елементів, які об'єднуються у послідовності для створення синтаксично правильних математичних законів.

Для обчислення та оцінки побудованих виразів використовується концепція Зворотної польської нотації. Вона дозволяє представляти математичні операції без використання дужок, послідовно накопичуючи операнди та оператори на обчислювальному стеку. Ефективність синтезованого виразу визначається на основі багатокритеріального аналізу, який балансує між точністю наближення цільового тренду та простотою структури формули. Головною проблемою при такому синтезі є схильність виразів до надмірного розростання (ефект «bloat»), через що концепція передбачає обов'язкове призначення індивідуальних вагових коефіцієнтів

для кожного типу оператора для стримування підсумкової складності виразу.

2.2 Математичний апарат базового алгоритму «Proximal Policy Optimization» (PPO)

Алгоритм «Proximal Policy Optimization» (PPO) належить до методів градієнта стратегії («Policy Gradient») у навчанні з підкріпленням і використовується для оптимізації поведінки інтелектуальних агентів у дискретних або неперервних просторах дій. Головною математичною особливістю PPO є стабілізація процесу навчання нейромережевої політики шляхом обмеження величини її оновлення на кожному кроці, що запобігає руйнівним змінам стратегії.

Математичний апарат алгоритму базується на концепції сурогатної цільової функції («Surrogate Objective»). Замість прямої максимізації очікуваної винагороди, PPO обчислює відношення ймовірностей виконання певної дії за нової та старої стратегій. Щоб уникнути занадто великих кроків оновлення ваг мережі, це відношення штучно обмежується («clipping») у межах фіксованого інтервалу за допомогою спеціального параметра ϵ . Механізм відсікання гарантує, що нова стратегія не отримає надлишкового градієнтного стимулу, якщо дія призводить до неочікувано великого стрибка винагороди.

Для оцінки доцільності обраної дії використовується функція переваги («Advantage Function»). Вона визначає, наскільки поточний вибір агента є ефективнішим порівняно з середнім очікуваним результатом для даного стану. Розрахунок цієї функції спирається на дворівневу архітектуру «Актор – Критик» («Actor-Critic»):

– мережа актора відповідає безпосередньо за формування стратегії (генерацію розподілу ймовірностей дій) та визначається параметрами архітектури;

– мережа критика оцінює загальну цінність поточного стану середовища («Value Function») і конфігурується параметрами архітектури.

Під час градієнтного спуску оптимізація параметрів виконується ітеративно протягом кількох епох навчання на основі накопичених пакетів даних. Додатково в математичну функцію втрат вводиться коефіцієнт ентропії («ent_coef»), який діє як регуляризатор: він максимізує випадковість дій на початкових етапах, утримуючи вікно пошуку відкритим та стимулюючи агента досліджувати нові комбінації токенів.

3 ПРОЄКТУВАННЯ ТА АЛГОРИТМІЧНЕ МОДЕЛЮВАННЯ КОМПОНЕНТІВ СЕРЕДОВИЩА СИМВОЛЬНОГО СИНТЕЗУ

3.1 Представлення математичних виразів та обчислювальної логіки у вигляді стека «Зворотної польської нотації» за допомогою компонентів «RPNEExpression» та «RPNEEnvStack»

Для ефективного навчання градієнтного агента простору пошуку аналітичних рішень необхідна лінійна та однозначна форма представлення математичних операцій. Зворотна польська нотація (RPN) ідеально підходить для інтеграції в алгоритми навчання з підкріпленням, оскільки вона виключає потребу у використанні дужок та складних синтаксичних дерев, перетворюючи процес генерації формули на послідовний вибір токенів із дискретного простору дій.

На програмному рівні обчислювальна логіка взаємодії зі стеком та збереження результатів інкапсульована у двох базових класах:

- клас «RPNEEnvStack» виконує роль безпосереднього операційного середовища для моделювання математичного стека протягом поточного епізоду навчання; він підтримує динамічну історію доданих токенів і розраховує віртуальну довжину стека, яка збільшується при додаванні операндів та зменшується відповідно до арності при виклику математичних операторів; цей компонент забезпечує базові функції валідації синтаксису, трансляції накопиченої послідовності в інфіксну (класичну) форму, а також безпосереднє обчислення значень виразу на масивах даних із підстановкою оптимізованих констант;

- клас «RPNEExpression» є незмінною структурою даних, яка слугує контейнером для фіксації та зберігання вже повністю сформованих математичних виразів; він створюється наприкінці успішного епізоду або при виявленні перспективних аналітичних траєкторій; завдяки використанню оптимізованого механізму слотів для економії пам'яті, цей

об'єкт надійно зберігає копію послідовності символів, довжину виразу, його логічний статус валідності, а також зафіксовані аналітичні метрики складності, різноманітності та глибини обчислювального дерева.

Такий розподіл обов'язків дозволяє ізолювати процес покрокового конструювання формули від механізмів довгострокового збереження елітних рішень, забезпечуючи високу швидкість обчислень під час взаємодії агента з середовищем штучного інтелекту.

3.2 Формалізація алфавіту математичних токенів та примусова типізація структур даних через перерахування «SymbolTypes», іменовані кортежі «StackSymbol» та «SymbolSequence»

Для побудови стійкого процесу синтезу виразів необхідно чітко типізувати кожен елемент алфавіту, яким оперує інтелектуальний агент. Це запобігає виникненню помилок невідповідності типів на рівні обчислювального ядра та дозволяє системі однозначно інтерпретувати семантичне призначення кожного обраного токена. Алфавіт доступних дій у розроблюваній системі складається зі змінних (a, b, c), базових лінійних операторів (+, -), нелінійних операцій (*, ^2), придатної до оптимізації числової константи (C) та спеціального термінального символу (=), що сигналізує про завершення формування виразу.

Строга типізація та організація структури даних у системі реалізується за допомогою трьох взаємопов'язаних базових компонентів:

- перерахування «SymbolTypes» фіксує п'ять легітимних категорій елементів у середовищі: «CONST_TYPE» (для констант), «SYMB_TYPE» (для математичних операторів), «NUM_TYPE» (для фіксованих чисел), «VAR_TYPE» (для вхідних змінних) та «TERM_TYPE» (для маркера виходу з епізоду);

- іменований кортеж «StackSymbol» представляє атомарну одиницю обчислень, об'єднуючи в собі посилання на конкретний тип із

перерахування «SymbolTypes» та безпосереднє числове або текстове значення токена;

– іменований кортеж «SymbolSequence» виступає високорівневою обгорткою над масивом символів, зберігаючи підсумкову послідовність об'єктів «StackSymbol», загальну кількість елементів у ній та булевий прапорець «is_valid», який відображає синтаксичну коректність усього виразу.

Завдяки такому підходу алфавіт математичних токенів повністю ізолюється від низькорівневих динамічних типів мови програмування. Це суттєво спрощує логіку фільтрації дій агента та підвищує загальну надійність і швидкість роботи моделі штучного інтелекту під час ітеративного перебору обчислювальних варіантів.

3.3 Формування векторизованого простору спостережень об'єкта стану середовища та інтеграція динамічних масок дій («Action Masking») для виключення синтаксично некоректних операцій агента

Для успішного навчання агента алгоритму PPO необхідно забезпечити його повною інформацією про поточний стан обчислювального середовища, а також запобігти генерації синтаксично помилкових послідовностей токенів. Простір спостережень (observation space) у розробленому середовищі реалізовано у вигляді плоского вектора фіксованого розміру, що поєднує структурну матрицю історії та додаткові нормалізовані скаляри. Матрична частина послідовно відображає всі токени, які агент уже встиг помістити на обчислювальний стек протягом поточного епізоду.

З метою стабілізації обчислень та вирівнювання ознак вектор спостережень розширюється чотирма додатковими скалярними величинами, які динамічно масштабуються в межах від 0.0 до 1.0:

- масштабований номер поточного кроку епізоду, що відображає наближення агента до ліміту дозволених дій;
- масштабована віртуальна довжина стека, яка вказує на поточне співвідношення між накопиченими числами та операторами;
- масштабований залишок доступних для використання числових констант, що контролює використання параметрів у формулі;
- масштабована поточна глибина обчислювального дерева, яка допомагає агенту оцінювати структурну складність виразу на кожному кроці.

Проте сама лише видача детального стану не захищає від випадкового вибору математичного оператора за відсутності чисел на стеку. Для розв'язання цієї проблеми в архітектуру інтегровано механізм динамічного маскування дій («Action Masking»). На кожному кроці середовище прораховує логічну маску валідності для всього алфавіту дій. Математичні оператори примусово блокуються, якщо поточна віртуальна довжина стека є меншою за їхню арність; символ константи стає недоступним при досягненні ліміту; а термінальний символ виходу дозволяється до вибору виключно тоді, коли стек містить рівно один фінальний результат обчислень. Таке відсікання звужує простір пошуку, змушуючи градієнтні алгоритми РРО фокусуватися виключно на змістовній математичній якості виразів.

3.4 Математичне обґрунтування компонентів комплексної багатоцільової функції винагороди («Reward Shaping»)

3.4.1 Оцінка функціональної близькості обчисленого виразу до цільового тренду на основі раціональної функції втрат («Rational Fitness Loss»)

Критично важливим елементом навчання з підкріпленням у задачах символної регресії є математичне визначення метрики відхилення згенерованого виразу від цільових значень. Для оцінки функціональної

близькості передбачено розрахунок базової помилки відповідності («fitness loss») між вектором передбачених агентом значень та вектором цільових ознак із тренувального масиву даних.

Для стабілізації обчислень та запобігання надмірним стрибкам функції втрат при роботі з великими числовими розбіжностями у системі застосовується попереднє нелінійне перетворення векторів за допомогою гіперболічного арксинуса $\operatorname{arcsinh}$. Базова оцінка втрат розраховується як корінь із середньоквадратичного відхилення між трансформованими значеннями передбачень і цілей. Якщо обчислені значення містять математичні невизначеності або нескінченності, функція примусово повертає фіксоване велике штрафне значення, сигналізуючи про непридатність виразу.

Класичні підходи часто використовують суто експоненційну форму для переведення помилки у фіксований діапазон винагороди, проте це призводить до швидкого згасання градієнтів, коли через значну початкову помилку винагорода агента миттєво падає до абсолютного нуля. Для подолання цієї проблеми у роботі реалізовано перехід до раціональної функції оцінки близькості (формула 3.1).

$$rational_fitness = \frac{1.0}{1.0 + base_fitness_loss}. \quad (3.1)$$

Цей раціональний місток забезпечує плавний і безперервний потік градієнтів під час оптимізації політики. Він гарантує, що перспективні структури формул, які ще не досягли ідеальної точності, все одно отримують вагоме проміжне обчислювальне стимулювання, що дозволяє алгоритму РРО стабільно розвивати пошукову траєкторію без потрапляння в локальні тупики.

3.4.2 Механізми придушення структурного розростання виразів («Anti-bloat Complexity Penalty») та максимізації покриття змінних («Exploration Diversity Score»)

Для уникнення неконтрольованого структурного розростання формул (ефекту «bloat») у функцію винагороди інтегрується штраф за зважену складність виразу. Кожному токenu в алфавіті призначається індивідуальна вага: змінні (a, b, c) та константа (C) мають базову вагу 1, лінійні оператори додавання та віднімання – вагу 1, а оператори множення (*) та піднесення до квадрата (^ 2) – підвищену вагу 2, що дозволяє ефективно контролювати глибину обчислювального дерева. Штраф за складність розраховується за експоненціальним законом як $\exp(-\alpha \cdot \text{complexity})$, де коефіцієнт α визначає ступінь придушення надлишкової довжини формули та забезпечує плавне зниження винагороди для громіздких математичних структур.

Для стимулювання дослідницької поведінки агента та розширення простору пошуку в процесі генерації послідовностей застосовується оцінка різноманітності виразу («diversity score»). Розрахунок цієї метрики базується на двох ключових показниках:

- покриття змінних, що відображає відношення кількості унікальних вхідних змінних, присутніх у поточному стеку, до загального обсягу очікуваних або дозволених у системі змінних;
- різноманітність токенів, що оцінює відношення кількості всіх унікальних символів виразу до його повної довжини, запобігаючи циклічному повторенню однакових операцій.

Зазначені показники поєднуються у зважену суму на основі лінійних коефіцієнтів та інтегруються безпосередньо у функцію винагороди як позитивний бонус різноманітності, масштабований параметром β . У підсумку, оцінка точності та штраф за довжину поєднуються як добуток раціональної точності та експоненціального згасання складності, після чого

до результату додається лінійна надбавка різноманітності, що дозволяє збалансувати пошукову поведінку моделі штучного інтелекту.

3.4.3 Динамічне коригування та фільтрація винагороди залежно від етапу навчання, частоти повторення виразів та статистичних коефіцієнтів кореляції Пірсона і Спірмена

У процесі тривалого навчання з підкріпленням виникає значний ризик зациклення агента на локально успішних, але субоптимальних математичних паттернах, що утворюють так звану «сіру масу» виразів. Для придушення таких повторюваних структур і динамічного стимулювання пошуку нових рішень в архітектуру функції винагороди інтегровано спеціалізований мультиплікатор «pearson_reward_multiplier». Його робота базується на одночасному оцінюванні статистичного взаємозв'язку між передбаченими та цільовими даними, частоти зустрічальності поточної формули в історії епізодів, а також глобального прогресу навчання моделі.

Розрахунок статистичної відповідності трендів виконується шляхом порівняння коефіцієнтів лінійної кореляції Пірсона та рангової кореляції Спірмена, після чого для подальшого аналізу алгоритм автоматично обирає максимальне з двох отриманих значень. Отриманий узагальнений коефіцієнт кореляції передається у функцію фільтрації, яка розподіляє обчислювальний процес за кількома регуляризаційними зонами залежно від поточної ітерації навчання:

- зона прецизійного імунітету активується одразу, якщо коефіцієнт кореляції перевищує суворий поріг 0.975, що повністю звільняє досконалу або максимально точну формулу від будь-яких частотних штрафів задля її закріплення у вагах мережі;

- динамічний поріг кореляції лінійно збільшується в міру проходження загального часу навчання від початкового значення 0.2 до

фінальної планки 0.85, відображаючи постійне зростання вимог до якості виразів із кожною тисячею кроків моделі;

– зона швидкого придушення (умовна «Зона А») застосовується до виразів, які відстають від динамічного порогу прогресу, піддаючи їх суворому згасанню та суттєво обмежуючи підсумкову винагороду при перевищенні ліміту в 100 повторень;

– зона лагідної фільтрації (умовна «Зона Б») діє як інкубатор для перспективних формул, що встигають за загальним математичним прогресом, плавно знижуючи їхню винагороду при частих повтореннях для стимулювання подальшого дослідження простору дій.

Така багатоступенева фільтрація за рахунок динамічного зворотного зв'язку дозволяє ефективно очищати обчислювальний буфер від однотипних закономірностей, не порушуючи при цьому стабільність розвитку унікальних аналітичних рішень на пізніх етапах оптимізації політики РРО.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ

4.1 Вибір та обґрунтування стека технологій

Для практичної реалізації розробленого методу аналітичного синтезу математичних виразів було сформовано спеціалізований технологічний стек на базі мови програмування Python 3.10. Вибір цієї версії мови зумовлений її високою стабільністю, швидкістю обчислень та повною сумісністю з сучасними інструментами машинного навчання та аналізу даних.

Основним інструментом для побудови архітектури навчання з підкріпленням обрано екосистему фреймворків «Stable-Baselines3» та її офіційне прикладне розширення «sb3_contrib». Ключовим фактором вибору цього інструментарію є наявність готової, оптимізованої та стабільної реалізації алгоритму «MaskablePPO». Оскільки базовий алгоритм PPO не має вбудованої підтримки динамічного маскуванню дискретних дій, використання спеціалізованої бібліотеки дозволяє надійно відсікати синтаксично некоректні кроки агента без руйнування градієнтної логіки оновлення ваг актор-критик мережі.

Для моделювання логіки взаємодії інтелектуального агента з обчислювальним стеком використано інструментарій «Gymnasium» від Farama Foundation. Він виступає сучасним стандартом для проектування кастомних середовищ навчання і надає готові інтерфейси, базові класи та детерміновані структури для конфігурування просторів спостережень і дискретних просторів дій.

Математичне ядро системи розроблено із залученням бібліотеки «NumPy», яка забезпечує швидке виконання масивних векторних операцій при підстановці змінних та обчисленні значень формул. Для вирішення супутніх обчислювальних задач інтегровано компоненти бібліотеки «SciPy». Зокрема, модуль «scipy.optimize» використовується для чисельного підбору оптимальних значень вільних констант усередині формули за

допомогою алгоритму Нелдера-Міда, а модуль «scipy.stats» забезпечує розрахунок коефіцієнтів кореляції Пірсона та Спірмена для динамічного оцінювання трендів.

Керування тренувальними та валідаційними даними реалізовано за допомогою бібліотеки Pandas. Вона забезпечує швидке зчитування сформованих табличних масивів, зручне відображення колонок змінних та формування випадкових пакетів даних для валідації математичної точності виразів на кожному кроці епізоду. Сформований стек технологій забезпечує високу швидкість ітерацій навчання та надійну ізоляцію логіки середовища від алгоритмів нейромережевої оптимізації.

4.2 Алгоритм префіксної ін'єкції цільових токенів змінних на стек середовища («Prefix Injection Logic») для оптимізації початкових пошукових траєкторій

На початкових етапах навчання з підкріпленням інтелектуальний агент здійснює випадковий перебір доступних дій, що у задачах символічної регресії призводить до тривалого пошуку хоча б частково правильних математичних структур. Особливо гостро ця проблема постає при синтезі специфічних функціональних залежностей, де певна змінна відіграє домінуючу роль у формуванні фінального результату (як-от коефіцієнт b у формулі дискримінанта квадратного рівняння). Для прискорення збіжності градієнтних алгоритмів та звуження простору випадкового пошуку в архітектуру взаємодії було інтегровано метод префіксної ін'єкції цільових токенів («Prefix Injection Logic»).

Цей алгоритм реалізовано безпосередньо всередині базового методу скидання стану середовища `reset` класу «RPNMaskableEnv». Логіка роботи префіксної ін'єкції базується на таких положеннях:

- середовище оперує базовою ймовірністю примусового додавання символу, яка за замовчуванням встановлена на рівні 60% для всіх тренувальних епізодів;

- передбачено можливість явного примусового виклику ін'єкції через зовнішні конфігураційні параметри за допомогою булевого прапорця «inject_b», що передається в словнику опцій;

- для збереження суворого детермінізму обчислень та повної сумісності з механізмом відтворюваності експериментів («seeds») вибір випадкового виконання ін'єкції здійснюється через вбудований у «Gymnasium» генератор «self.np_random».

Якщо умови для виконання алгоритму виконуються, система автоматично знаходить об'єкт «StackSymbol» для токена змінної b і поміщає його на обчислювальний стек середовища до того, як агент зробить свій перший самостійний вибір. Одночасно з цим лічильник поточних кроків епізоду примусово встановлюється в значення 1. Таким чином, агент починає генерацію виразу не з порожнього стека, а з уже наявним базовим операндом, що суттєво зменшує проблему «холодного старту» і спрямовує оптимізаційні траєкторії PPO у бік математично змістовних розв'язків.

4.3 Реалізація кастомного рангового колбеку «CustomRankRewardCallback» для нелінійного сортування, нормалізації та нелінійного масштабування винагород елітних траєкторій у буфері «rollout»

Для підвищення стабільності навчання та цілеспрямованого стимулювання агента генерувати винятково точні математичні вирази стандартний процес збору траєкторій було розширено за допомогою спеціалізованого механізму зворотного зв'язку. На програмному рівні цю логіку реалізовано у вигляді кастомного класу «CustomRankRewardCallback», який успадковує функціональність базового компонента «BaseCallback» фреймворку «Stable-Baselines3». Головне

завдання цього модуля – виконувати аналітичну пост-обробку та модифікацію масиву винагород безпосередньо всередині буфера «rollout» перед початком кожного етапу оновлення градієнтів нейронної мережі.

Робота розробленого колбеку розділена на два ключові етапи обробки кроків агента:

- управління пам'яттю середовища на проміжних кроках: під час виконання методу «_on_step» колбек безперервно відстежує глобальний лічильник кроків моделі та з фіксованою частотою ініціює очищення накопиченої середовищем історії повторень для уникнення зациклення; очищення виконується за допомогою вбудованого методу «env_method», який викликає функцію «clear_expression_history» паралельно на всіх активних воркерах векторизованого середовища;

- ранжування та нелінійне масштабування елітних рішень: після завершення повної ітерації збору досвіду викликається метод «_on_rollout_end», який зчитує копію сирих винагород із буфера моделі; якщо максимальна винагорода в поточному пакеті проходить встановлений поріг елітності «excellence_threshold», алгоритм виділяє індекси перспективних формул і піддає їх дворівневому захисному та оптимізаційному аналізу.

Захисний аналіз колбеку запобігає викривленню функції цінності критика через однотипні результати обчислень. Перед початком сортування впроваджено обов'язкову перевірку на дисперсію: якщо різниця між найкращим та найгіршим виразом серед відібраних кандидатів є меншою за фіксоване значення 0.05, то такий набір вважається «сірою масою» однакових за якістю спроб, і буфер «rollout» залишається недоторканим.

Якщо всі захисні бар'єри успішно пройдено, система виконує фірмове нелінійне ранжування. За допомогою подвійного сортування індексів обчислюється порядковий ранг кожного успішного виразу, який спочатку переводиться в лінійний нормалізований вигляд у діапазоні від 0.0 до 1.0, а потім підноситься до квадрата для реалізації нелінійного масштабування.

Отримані квадратичні оцінки відображаються на цільовий робочий інтервал між мінімальним та максимальним ранговими значеннями винагороди, після чого скориговані елітні оцінки примусово перезаписуються назад у буфер моделі PPO. Таке штучне «роздування» винагород для кращих формул дозволяє чітко відокремити високоефективні пошукові траєкторії від середньостатистичних обчислювальних кроків.

4.4 Архітектурне конфігурування багатошарової актор-критик нейромережевої політики («MlpPolicy», «net_arch») та інтеграція з моделями «MaskablePPO»

Ефективність пошуку оптимальних аналітичних виразів безпосередньо залежить від структурної складності та параметрів конфігурації штучних нейронних мереж, які апроксимують стратегію поведінки агента та цінність поточних обчислювальних станів середовища. Оскільки математичні формули утворюють дискретні послідовності, нейромережева політика повинна володіти високою ємністю для точного розпізнавання прихованих закономірностей у векторі спостережень.

У межах розробленого методу інтеграція інтелектуального ядра реалізується за допомогою алгоритму «MaskablePPO» на базі стандартної багатошарової політики перцептрона «MlpPolicy». Налаштування внутрішньої архітектури мереж виконується через спеціалізований конфігураційний словник «policy_kwargs», який дозволяє явно розділити обчислювальні шари актора та критика для підвищення загальної стабільності градієнтного спуску:

– мережа актора (компонент «pi») конфігурується у вигляді двох послідовних прихованих шарів по 256 нейронів у кожному; це забезпечує оптимальний баланс між швидкістю розрахунку ймовірностей дій та здатністю мережі формувати складні логічні зв'язки між поточним станом стека й доступними математичними токенами;

– мережа критика (компонент «vf») вимагає потужнішої обчислювальної структури, через що її архітектура моделюється за допомогою трьох послідовних шарів ємністю 512, 256 та 256 нейронів відповідно; розширення першого прихованого шару до 512 елементів зумовлене високою складністю апроксимації функції цінності математичних станів, оскільки навіть незначна зміна порядку символів на стеку може кардинально змінити фінальну точність формули.

Під час інтеграції середовища з моделлю «MaskablePPO» встановлюється набір гіперпараметрів навчання, оптимізованих під специфіку символної регресії. Швидкість навчання («learning_rate») фіксується на рівні $2.5 \cdot 10^{-4}$ при розмірі пакета оновлення («batch_size») у 256 елементів та тривалості одного циклу збору досвіду («n_steps») в 512 кроків, що гарантує плавну модифікацію ваг без критичних розбіжностей градієнтів. Коефіцієнт дисконтування gamma налаштовується на високе значення 0.995, оскільки формування довгострокових математичних структур вимагає від агента врахування віддалених наслідків перших кроків на стеку.

Особливе місце в конфігурації посідає коефіцієнт ентропії («ent_coef»), який примусово збільшено до значення 0.08. Таке архітектурне рішення дозволяє штучно підтримувати високий рівень випадковості у діях агента, утримуючи вікно пошуку відкритим протягом тривалого часу та активно стимулюючи нейронну мережу досліджувати складні комбінації символів, зокрема доцільність впровадження числової константи C у структуру синтезованого виразу.

4.5 Розробка програмного модуля генерації, санітизації та збалансованого розподілу синтетичного датасету за математичними умовами дискримінанта квадратного рівняння

Для навчання інтелектуального агента задачі символної регресії розроблено та реалізовано автономний інструмент генерації синтетичних

даних, інкапсульований у модулі «dataset_generator.py». Головною метою цього модуля є створення збалансованого та репрезентативного числового масиву, який охоплює всі можливі математичні випадки та екстремальні стани при обчисленні дискримінанта квадратного рівняння.

Процес формування та очищення даних базується на кількох архітектурних рішеннях:

- загальний обсяг датасету жорстко зафіксовано на рівні 5120 зразків, які збалансовано розділені за типами даних у суворій пропорції: 60% цілих чисел та 40% чисел із плаваючою точкою;
- структура генерації має дворівневу ієрархію розподілу записів, де на першому рівні числа фільтруються за знаком дискримінанта (більше, менше або дорівнює нулю), а на другому – за індивідуальними знаками чи нульовими значеннями окремих коефіцієнтів a , b та c ;
- програмний модуль містить три спеціалізовані математичні функції розбиття для автоматичного та пропорційного квантування заданої кількості рядків між усіма логічними підрівнями умов;
- базові межі значень для випадкових коефіцієнтів обмежені інтервалом від -192 до 192, проте для параметра b простір генерації штучно розширено до меж від -500 до 500, що дозволяє збалансувати вибірку та спростити пошук рішень для агента;
- для усунення системних похибок округлення дробових чисел, які притаманні базовим типам даних, у логіку санітизації інтегровано використання об'єктів «Decimal» із примусовим конфігуруванням режимів точного математичного округлення.

Усі сформовані числові комбінації проходять циклічну валідацію на унікальність усередині динамічних списків, що повністю виключає появу дублікатів у фінальному наборі. Після успішного збору унікальних рядків для кожного зразка розраховується фінальне цільове значення ознаки «D_value» на основі формули дискримінанта. Підсумкова матриця автоматично записується у структурований CSV-файл, що забезпечує швидке зчитування та стабільну роботу шару доступу до даних під час тренувальних епізодів РРО.

5 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРОЦЕСІВ НАВЧАННЯ МОДЕЛІ

5.1 Конфігурація експериментального середовища, паралелізація обчислювальних потоків та нормалізація векторів

Для проведення обчислювальних експериментів та оцінки ефективності розробленого методу символьного синтезу аналітичних рішень було налаштовано інтегроване тестове середовище на базі алгоритму навчання з підкріпленням «MaskablePPO». Процес тренування моделі та збору траєкторій досвіду організовано за допомогою спеціалізованих інструментів векторизації, що дозволяє оптимізувати використання обчислювальних ресурсів комп'ютера та підвищити статистичну різноманітність накопичуваних даних.

Організація експериментального обчислювального процесу спирається на такі архітектурні рішення:

- для паралелізації збору досвіду застосовується векторизована обгортка «DummyVecEnv», яка координує одночасну роботу двох паралельних потоків взаємодії з кастомним середовищем «RPNMaskableEnv»;

- з метою запобігання дестабілізації градієнтів нейронної мережі векторизоване середовище додатково розширене за допомогою інструменту «VecNormalize»;

- у конфігурації «VecNormalize» пряму нормалізацію векторів спостережень та сирих винагород було вимкнено, проте активовано механізм жорсткого обмеження («clipping») абсолютного значення фінальної винагороди агента на рівні 10.0.

Таке обмеження амплітуди ревордів є критично важливим для стабілізації навчання актор-критик архітектури, оскільки воно захищає критичну функцію цінності («Value Function») мережі критика від раптових

деструктивних коливань градієнтів при виявленні агентом аномально успішних або елітних математичних формул у процесі випадкового пошуку. Загальний обсяг експериментального навчання зафіксовано на позначці 75000–100000 глобальних кроків, що забезпечує достатній час для повної збіжності алгоритму та формування точних аналітичних виразів.

5.2 Дослідження впливу бар'єра елітності («`excellence_threshold`») та обмежень ранжування винагороди на стабілізацію функції цінності критика («`Value Function`»)

Під час проведення обчислювальних експериментів особливу увагу було приділено аналізу динаміки навчання мережі критика, яка відповідає за апроксимацію функції цінності («`Value Function`») станів обчислювального середовища. Встановлено, що пряме використання сирих винагород або неконтрольоване лінійне розгоняння оцінок успішних траєкторій призводить до критичного зростання помилки передбачення критика. Для стабілізації цього процесу було досліджено вплив параметрів та захисних бар'єрів кастомного колбеку «`CustomRankRewardCallback`».

Емпіричний аналіз показав високу ефективність упровадження таких інженерних рішень:

- зниження верхньої планки ранжування: зменшення максимального значення відранжованої винагороди «`max_rank_val`» з 1.75 до 0.95 дозволило суттєво звузити амплітуду штучно модифікованих ревордів. Це безпосередньо стабілізувало функцію цінності критика, усунувши каскадні коливання градієнтів у шарах мережі «`vf`» при раптовому виявленні агентом вдалих математичних послідовностей;

- підняття бар'єра елітності: встановлення параметра «`excellence_threshold`» на рівні 0.42 заблокувало передчасний перехід субоптимальних рішень до категорії елітних. Прості вирази, такі як квадрат змінної b (що мають сиру оцінку близько 0.36), більше не отримували

додаткового рангового стимулювання, що змусило агента відмовитися від простих паттернів і залучати числовий токен константи C для отримання високого реворду;

– захисний екран дисперсії: впровадження обов'язкової перевірки варіативності оцінок (мінімальний поріг розбіжності 0.05 між кращим і гіршим виразом у пакеті) дозволило вчасно виявляти та ігнорувати однотипну «сіру масу» формул. Коли кандидати в буфері виявлялися занадто близькими за якістю, колбек автоматично вимикав ранжування, залишаючи недоторканими сирі градієнти РРО.

Завдяки поєднанню жорсткого відбору за якістю та обмеженню максимального діапазону винагород, лос функції цінності критика продемонстрував стійке, плавне згасання. Це забезпечило агенту надійну основу для точного розрахунку переваг дій на пізніх етапах оптимізації стратегії.

5.3 Оцінка ефективності механізму збереження та дешифрації знайдених аналітичних рішень у контейнері виразів («RPNExprContainer») з автоматичною трансляцією в інфіксну математичну форму

Фінальним етапом перевірки працездатності розробленого методу є оцінка механізму довгострокового збереження та дешифрації знайдених математичних рішень. Оскільки агент взаємодіє з середовищем виключно через лінійні послідовності токенів Зворотної польської нотації, виникає об'єктивна потреба в автоматизованому виділенні, накопиченні та перекладі найкращих результатів обчислень у зрозумілий для людини алгебраїчний вигляд. Для вирішення цієї задачі в архітектуру системи впроваджено автономний сховище-компонент «RPNExprContainer».

Робота цього механізму інтегрована в логіку виконання кроків інтелектуального агента в середовищі «RPNMaskableEnv». Процес відбору елітних рішень організовано за такою схемою:

– коли епізод обчислень завершується вибором термінального символу, середовище розраховує підсумкову багатоцільову винагороду за згенерований вираз;

– якщо поточна винагорода досягає строгого прохідного порогу «container_reward_threshold», який у фінальній конфігурації дорівнює 0.89, система ініціює процес створення об'єкта «RPNExpression»;

– об'єкт «RPNExpression» фіксує синтаксичну структуру формули, а також автоматично зберігає прораховані метрики її глибини, зваженої складності та різноманітності токенів;

– новий вираз успішно додається до сховища «RPNExprContainer» лише за умови, що його винагорода перевищує раніше зафіксований історичний максимум «__prev_max_reward», а сам вираз ще не присутній у пам'яті контейнера.

Ефективність дешифрації знайдених аналітичних рішень забезпечується вбудованим у обчислювальний стек «RPNEnvStack» алгоритмом «to_infix». Цей метод виконує зворотне покрокове читання токенів і реконструює класичний алгебраїчний вираз із правильним розташуванням операндів та дужок. Бінарні оператори автоматично розміщуються між своїми аргументами, а для специфічного унарного оператора піднесення до квадрата генерується класичний запис у форматі N^2 . Сам контейнер здійснює індексацію формул за внутрішніми словниками, що виключає логічне дублювання за їхньою інфіксною чи строковою формою та забезпечує фінальне виведення унікальних знайдених рішень.

5.4 Результати експериментів

Експериментальне дослідження розробленого методу аналітичного синтезу математичних виразів базувалося на тренуванні моделі протягом 75000 глобальних кроків у векторизованому середовищі, що

складалося з двох паралельних процесів. Основним індикатором успішності експерименту виступала динаміка зростання середньої винагороди за епізод та здатність агента стабільно генерувати точні математичні рішення, які задовольняють встановлений поріг елітності.

У результаті проведення обчислювальних симуляцій було зафіксовано такі підсумкові результати:

- процес навчання продемонстрував стійку збіжність градієнтних алгоритмів PPO завдяки поєднанню динамічного маскування дій та нелінійного ранжування винагород у буфері «rollout»;

- завдяки підвищеному коефіцієнту ентропії агент активно досліджував простір дій на перших етапах обчислень, що дозволило уникнути передчасної стабілізації на спрощених, але помилкових підгрупах токенів;

- механізм префіксної ін'єкції токена змінної b на початкових кроках епізоду дозволив моделі суттєво скоротити час пошуку перших валідних залежностей;

- контейнер найкращих виразів успішно зафіксував цільові математичні структури, які пройшли встановлений суворий прохідний бар'єр винагороди;

- обчислювальний стек середовища забезпечив точну дешифрацію та виведення фінальних результатів у класичній інфікській формі, підтвердивши знаходження шуканої структури дискримінанта квадратного рівняння.

Отримані числові показники та збережені аналітичні форми повністю підтверджують спроможність розробленої архітектури вирішувати задачі символної регресії в автоматичному режимі.

5.5 Аналіз результатів експериментів

Аналіз результатів фінального етапу обчислювального експерименту підтверджує працездатність та високу точність розробленого методу

символьного синтезу. Згідно з отриманими даними метрик термінала, повний цикл навчання агента завершився на позначці 75776 глобальних кроків, що тривало 1603 секунди (близько 26.7 хвилин) при середній швидкості обробки кадрів середовища на рівні 47 FPS.

Аналіз числових показників фінального пакета оновлень нейронної мережі дозволяє оцінити якість градієнтної оптимізації політики PPO:

- метрика втрат критика («value_loss») зафіксувалася на винятково низькому рівні 0.00409, що емпірично доводить ефективність застосування кастомного рангового колбеку для стабілізації функції цінності стану середовища;

- показник втрат стратегії («policy_gradient_loss») склав -0.00512 при значенні коефіцієнта відсікання «clip_fraction» на рівні 0.00586, що свідчить про плавне, контрольоване оновлення ваг актора без деструктивних стрибків пошукових траєкторій;

- значення втрат ентропії («entropy_loss») наприкінці навчання склало -0.535, підтверджуючи, що мережа актора зберегла необхідний рівень випадковості дій для утримання вікна дослідження відкритим до моменту повної збіжності алгоритму;

- показник поясненої дисперсії («explained_variance») склав -1.24, що є типовим наслідком функціонування агента в середовищах із нестационарними, складно зваженими функціями винагороди та динамічним частотним згасанням, проте це не завадило успішній конвергенції актора.

Важливим підтвердженням теоретичних положень є лог проміжних траєкторій епізодів у верхній частині знімка екрана (рисунок 5.1). Там чітко зафіксовано накопичення лічильників повторень (наприклад, 68 та 73 зустрічей) для локально успішних, але неповних математичних паттернів виду « $b^2 \cdot c$ » із проміжною винагородою 0.4044. Поступове збільшення кількості цих записів активувало штрафні коефіцієнти модуля «pearson_reward_multiplier», що знижувало привабливість субоптимальних кроків та змушувало агента продовжувати пошук нових комбінацій. Також

лог відображає роботу захисного механізму колбеку: повідомлення про скидання через низький реворд буфера (0.415) нижче планки 0.553 демонструє успішне відсікання «сірої маси» виразів від процесу штучного ранжування.

```

b ^ 2 C * c - = (rw: 0.3631) (pc: 0.8747) | pc_0.87: 51
b ^ 2 C * c - = (rw: 0.4044) (pc: 0.8837) | pc_0.88: 68
b ^ 2 ^ 2 C * ^ 2 = (rw: 0.3625) (pc: 0.7819) | pc_0.78: 6
b ^ 2 C * a + = (rw: 0.3771) (pc: 0.8658) | pc_0.87: 38
b ^ 2 C * c - = (rw: 0.4044) (pc: 0.8837) | pc_0.88: 73

[RankCallback] Скидання: Макс реворд буфера (0.415) нижче планки 0.553

time/
fps              47
iterations       74
time_elapsed     1603
total_timesteps  75776
train/
approx_kl        0.0016778517
clip_fraction    0.00586
clip_range       0.2
entropy_loss     -0.535
explained_variance -1.24
learning_rate    0.00025
loss             -0.0509
n_updates        285
policy_gradient_loss -0.00512
value_loss       0.00409

Навчання завершено!

--- Найкращі знайдені вирази ---
1. Infix form of expression: D = ((b ** 2) - ((C ** 2) * (c * a)));

```

Рисунок 5.1 – Знімок екрану з фінальними результатами роботи розробленого програмного рішення

Останнім головним практичним результатом експерименту є успішний синтез цільового математичного виразу та його безпомилкова дешифрація обчислювальним стеком в інфіксну форму: $D = \left((b^2) - ((C^2) * (c * a)) \right)$. Оскільки числова константа C піддавалася локальній чисельній оптимізації алгоритмом Нелдера-Міда на базі згенерованого математичного датасету квадратних рівнянь, її збіжність до значення, квадрат якого дорівнює 4 ($C \approx \pm 2$), є прямим доказом того, що інтелектуальний агент майже повністю самостійно відновив приблизну аналітичну структуру формули дискримінанта ($b^2 - 4ac$).

Результати деяких проміжних пусків розробленого програмного рішення представлі на знімках екрану нижче (рисунок 5.2–5.6).

```
c a * C * b C + + = (0.28217368684763955)
a c * C * C + b ^ 2 + = (0.9825950300111503)
a c * C * C + b + = (0.28217368684763955)
```

Рисунок 5.2 – Знімок екрану з проміжними результатами роботи розробленого програмного рішення

```
a c * C * C b + ^ 2 + = (0.9825554105660512)
a c * C * C b b + + + = (0.28161648034376335)
c a * C * C b ^ 2 + + = (0.9825950300111503)
c a * C * C b + ^ 2 + = (0.9825554105660512)
a c * C * C b ^ 2 + + = (0.9825950300111503)
a c * C * C b + ^ 2 + = (0.9825554105660512)
a c * C * C b + ^ 2 + = (0.9825554105660512)
c a * C * C b + ^ 2 + = (0.9825554105660512)
c a * C * C b ^ 2 + + = (0.9825950300111503)
a c * C * C b ^ 2 + + = (0.9825950300111503)
a c * C * C b b + + + = (0.2816164803437633)
a c * C * C b + ^ 2 + = (0.9825554105660512)
a c * C * C b ^ 2 + + = (0.9825950300111503)
a c * C * C b ^ 2 + + = (0.9825950300111503)
a c * C * C b ^ 2 + + = (0.9825950300111503)
```

Рисунок 5.3 – Знімок екрану з проміжними результатами роботи розробленого програмного рішення

```
65. Infix form of expression: D = (((a * c) * C) + (C + c));
66. Infix form of expression: D = (((c * a) * C) + ((C - b) + a));
67. Infix form of expression: D = (((a * c) * C) + ((C - b) + b));
68. Infix form of expression: D = (((a * c) * C) + (b - C));
69. Infix form of expression: D = (((a * c) * C) + ((C + b) ** 2));
70. Infix form of expression: D = (((c * a) * C) + (C - b));
71. Infix form of expression: D = (((c * a) * C) + (C ** 2));
72. Infix form of expression: D = (((a * c) * C) - (C + (b + b)));
73. Infix form of expression: D = (((a * c) * C) + (C - (b + c)));
74. Infix form of expression: D = (((a * c) * C) + ((C - b) ** 2));
75. Infix form of expression: D = (((c * a) * C) + ((C + b) ** 2));
76. Infix form of expression: D = (((c * a) * C) + ((C - b) ** 2));
77. Infix form of expression: D = (((c * a) * C) + b) + C;
78. Infix form of expression: D = (((c * a) * C) + ((C + b) + b));
79. Infix form of expression: D = (((c * a) * C) - (C + (b + b)));
80. Infix form of expression: D = (((c * a) * C) + (C + (b ** 2)));
81. Infix form of expression: D = (((a * c) * C) + ((C ** 2) + b));
82. Infix form of expression: D = (((c * a) * C) + (C + (b + c)));
83. Infix form of expression: D = (((a * c) * C) + ((C + c) + b));
84. Infix form of expression: D = (((a * c) * C) + ((C ** 2) ** 2));
85. Infix form of expression: D = (((c * a) * C) + ((C - b) + b));
86. Infix form of expression: D = (((c * a) * C) + (C * (b ** 2)));
87. Infix form of expression: D = (((c * a) * C) + ((C ** 2) ** 2));
88. Infix form of expression: D = (((a * c) * C) + b) - C;
89. Infix form of expression: D = (((c * a) * C) + ((C * b) ** 2));
90. Infix form of expression: D = (((a * c) + b) * C) + C;
91. Infix form of expression: D = (((a * c) * C) + (C + (b + a)));
92. Infix form of expression: D = (((a * c) * C) + ((b + C) ** 2));
```

Рисунок 5.4 – Знімок екрану з проміжними результатами роботи розробленого програмного рішення

```

--- Найкращі знайдені вирази ---
1. Infix form of expression: D = (b - (C + (a * c)));
2. Infix form of expression: D = (C - ((c * a) - b));

```

Рисунок 5.5 – Знімок екрану з проміжними результатами роботи розробленого програмного рішення

```

--- Найкращі знайдені вирази ---
1. Infix form of expression: D = ((b ** 2) - (a * (c * C)));
2. Infix form of expression: D = ((b ** 2) - ((a * c) * C));
3. Infix form of expression: D = ((b ** 2) - (a * (c - (C * c))));
4. Infix form of expression: D = ((b ** 2) - (a * (C * c)));
5. Infix form of expression: D = ((b ** 2) + (a * (c * C)));
6. Infix form of expression: D = ((b ** 2) + ((a * c) * C));
7. Infix form of expression: D = ((b ** 2) + (a * (c - (C * c))));
8. Infix form of expression: D = ((b ** 2) - (a * ((c * C) - c)));
9. Infix form of expression: D = ((b ** 2) - (a * (c - (c * C))));
10. Infix form of expression: D = ((b ** 2) + (a * (C * c)));
11. Infix form of expression: D = ((b ** 2) - ((a * c) * c));

```

Рисунок 5.6 – Знімок екрану з проміжними результатами роботи розробленого програмного рішення

5.6 Компромід між автономністю та контрольованістю

Розробка методів символьного синтезу математичних виразів на основі навчання з підкріпленням вимагає знаходження балансу між автономією пошукових дій інтелектуального агента та детермінованим контролем над коректністю одержаних результатів. З одного боку, високий рівень автономії є необхідним для вільного дослідження великого простору дискретних дій та виявлення нестандартних нелінійних закономірностей у числових масивах. З іншого боку, абсолютна свобода дій зазвичай призводить до комбінаторного вибуху, швидкого накопичення синтаксичних помилок та неконтрольованого розростання структури формул.

У межах даної роботи автономність агента забезпечується за рахунок гнучкої актор-критик архітектури нейронної мережі та примусового збільшення коефіцієнта ентропії політики до рівня 0.05 – 0.1 (0.08). Це

підтримує високу випадковість вибору токенів на початкових етапах навчання, спонукаючи агента самостійно комбінувати змінні, оператори та константи у пошуках оптимальної обчислювальної траєкторії.

Контрольованість системи реалізується через багаторівневу систему обмежень, інтегровану безпосередньо в обчислювальне середовище «RPNMaskableEnv». Головним інструментом регулювання виступає механізм динамічного маскуванню дій («Action Masking»), який аналізує стан стека на кожному кроці епізоду й примусово блокує виклик математичних операторів, що суперечать правилам арифметичності Зворотної польської нотації. Додатковий контроль забезпечується суворими інфраструктурними лімітами середовища, такими як обмеження на максимальну довжину виразу, глибину обчислювального дерева та ліміт використання вільних констант тощо.

Знайдений у роботі компроміс полягає в тому, що середовище повністю позбавляє агента можливості припуститися синтаксичної помилки через видачу від'ємної винагороди, але залишає йому абсолютну свободу в семантичному конструюванні математичної логіки виразу. При цьому комплексна функція винагороди та кастомний ранговий колбек діють як м'які аналітичні регулятори, які не обмежують пошук агента жорсткими правилами, а через динамічне коригування ревордів у буфері «rollout» природно спрямовують його автономну поведінку у бік точних, компактних та різноманітних аналітичних рішень.

6 РЕКОМЕНДАЦІЇ ЩОДО ПОДАЛЬШОЇ РОБОТИ В ДАНОМУ НАПРЯМКУ

Подальший розвиток розробленого методу синтезу розв'язків математичних задач в аналітичному вигляді доцільно спрямувати на розширення його функціональних можливостей, оптимізацію обчислювальної швидкості та підвищення масштабованості системи. На основі проведених експериментів та аналізу архітектури середовища навчання можна виділити такі ключові рекомендації для майбутніх досліджень у цьому напрямку:

- розширення алфавіту математичних токенів: доцільно включити до простору дій нові типи операцій, зокрема тригонометричні ($\sin$, $\cos$), показникові, логарифмічні функції, а також оператори ділення; це дозволить агенту виходити за межі поліноміальних залежностей і синтезувати аналітичні рішення для значно ширшого класу математичних та фізичних задач;

- оптимізація підбору числових констант: поточна реалізація використовує алгоритм Нелдера-Міда для оптимізації значень вільних констант наприкінці кожного успішного епізоду, що може створювати обчислювальне «вузьке місце» при збільшенні обсягу даних; перспективним кроком є інтеграція швидших градієнтних методів локального пошуку або використання окремої нейромережевої підсистеми для миттєвого передбачення значень коефіцієнтів;

- масштабування для багатовимірних задач: для адаптації методу до роботи з великою кількістю вхідних змінних рекомендується дослідити архітектури з використанням механізмів семантичного аналізу або технологій «Retrieval-Augmented Generation» (RAG); це дозволить агенту динамічно фокусуватися лише на релевантних змінних, не перевантажуючи простір спостережень і зберігаючи стабільність збіжності алгоритму PPO;

– динамічний розклад параметрів винагороди («Reward Scheduling»): доцільно впровадити механізм автоматичної зміни вагових коефіцієнтів альфа та бета безпосередньо в процесі тренування; наприклад, на початкових етапах навчання пріоритет може віддаватися різноманітності токенів та дослідженню простору, а на пізніх – жорсткому придушенню складності та максимізації точності формули;

– перехід до мультиагентних архітектур: перспективним напрямком є розподіл функціональних ролей між кількома спеціалізованими агентами; у такій системі один агент може відповідати суто за генерацію скелета (структури) математичного виразу, другий – за локальне чисельне припасування констант, а третій – за перевірку математичної безпеки (виключення ділення на нуль чи отримання комплексних коренів).

Реалізація зазначених рекомендацій дозволить трансформувати розроблену систему з експериментального середовища символічної регресії у повноцінний, масштабований інструментарій автоматизованого наукового пошуку та аналітичного моделювання складних динамічних процесів.

ВИСНОВКИ

У кваліфікаційній роботі розроблено, програмно реалізовано та всебічно досліджено метод автоматичного синтезу розв'язків математичних задач в аналітичному вигляді з використанням алгоритму навчання з підкріпленням «Proximal Policy Optimization» (PPO).

В межах дослідження побудовано кастомне обчислювальне середовище «RPNMaskableEnv» на базі фреймворку «Gymnasium», яке реалізує покрокове моделювання обчислень у форматі «Зворотної польської нотації» за допомогою компонентів «RPNEnvStack» та «RPNExpression». Інтегрований механізм динамічного маскування дій («Action Masking») дозволив повністю виключити ймовірність генерації синтаксично некоректних виразів, що суттєво звузило дискретний простір випадкового пошуку та спрямувало градієнтні алгоритми на оптимізацію суто семантичної якості формул. Спроектowana комплексна багатоцільова функція винагороди через раціональну функцію втрат («Rational Fitness Loss») забезпечила безперервний потік градієнтів, а через експоненціальний штраф за складність та бонус різноманітності ефективно придушила ефект структурного розростання формул («bloat»). Додатково впроваджений динамічний прецизійний частотний фільтр «pearson_reward_multiplier» на основі статистичних коефіцієнтів кореляції Пірсона та Спірмена дозволив успішно очищати обчислювальний буфер від повторюваної «сірої маси» однотипних формул без шкоди для розвитку унікальних аналітичних траєкторій.

Стабілізація функції цінності («Value Function») мережі критика із фіксацією фінального лосу на винятково низькому рівні 0.00409 була досягнута завдяки розробці кастомного рангового колбеку «CustomRankRewardCallback», який виконує нелінійне масштабування винагород елітних рішень та обов'язкову перевірку дисперсії. Надійну інформаційну базу для експериментів забезпечив автономний програмний

модуль генерації, санітизації та квантування синтетичного датасету за математичними умовами дискримінанта, що дозволило сформувати тренувальну вибірку з 5120 унікальних зразків без системних похибок округлення.

Фінальні обчислювальні експерименти підтвердили стійку збіжність актор-критик архітектури протягом 75776 кроків навчання за 1603 секунди при середній швидкості 47 FPS. Головним практичним результатом роботи став успішний автоматичний синтез аналітичного виразу близького до «цільового» та його безпомилкова дешифрація в інфіксну форму $D = ((b^2) - ((C^2) * (c * a)))$.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. AI Feynman 2.0: pareto-optimal symbolic regression exploiting graph modularity / S.-M. Udrescu et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2006.10782> (date of access: 15.06.2026).
2. A neural-guided dynamic symbolic network for exploring mathematical expressions from data / W. Li et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2309.13705> (date of access: 16.06.2026).
3. Bayesian symbolic regression / Y. Jin et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.1910.08892> (date of access: 17.06.2026).
4. Brunton S. L., Proctor J. L., Kutz J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*. 2016. Vol. 113, no. 15. P. 3932–3937. URL: <https://doi.org/10.1073/pnas.1517384113> (date of access: 18.06.2026).
5. Complexity-Aware deep symbolic regression with robust risk-seeking policy gradients / Z. Bastiani et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2406.06751> (date of access: 19.06.2026).
6. Deep symbolic optimization: reinforcement learning for symbolic mathematics / C.F. Hayes et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2505.10762> (date of access: 20.06.2026).
7. Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients / B.K. Petersen et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.1912.04871> (date of access: 21.06.2026).
8. Discovering faster matrix multiplication algorithms with reinforcement learning / A. Fawzi et al. *Nature*. 2022. Vol. 610, no. 7930. P. 47–53. URL: <https://doi.org/10.1038/s41586-022-05172-4> (date of access: 22.06.2026).
9. Discovering mathematical formulas from data via gpt-guided monte carlo tree search / Y. Li et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2401.14424> (date of access: 23.06.2026).

10. Martius G., Lampert C.H. Extrapolation and learning equations. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.1610.02995> (date of access: 24.06.2026).
11. Memar F., Zhe T., Wang D. Neural structure embedding for symbolic regression via continuous structure search and coefficient optimization. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2603.22429> (date of access: 25.06.2026).
12. Neural symbolic regression that scales / L. Biggio et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2106.06427> (date of access: 26.06.2026).
13. Proximal policy optimization algorithms / J. Schulman et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.1707.06347> (date of access: 27.06.2026).
14. Schmidt M., Lipson H. Distilling free-form natural laws from experimental data. *Science*. 2009. Vol. 324, no. 5923. P. 81–85. URL: <https://doi.org/10.1126/science.1165893> (date of access: 28.06.2026).
15. Dabelow L., Ueda M. Symbolic equation solving via reinforcement learning. *Neurocomputing*. 2025. Vol. 613. P. 128732. URL: <https://doi.org/10.1016/j.neucom.2024.128732> (date of access: 29.06.2026).
16. SymbolicGPT: a generative transformer model for symbolic regression / M. Valipour et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2106.14131> (date of access: 30.06.2026).
17. Symbolic regression via neural-guided genetic programming population seeding / T.N. Mundhenk et al. *arXiv.org*. URL: <https://doi.org/10.48550/arXiv.2111.00053> (date of access: 01.07.2026).
18. Udrescu S.-M., Tegmark M. AI Feynman: a physics-inspired method for symbolic regression. *Science advances*. 2020. Vol. 6, no. 16. P. eaay2631. URL: <https://doi.org/10.1126/sciadv.aay2631> (date of access: 02.07.2026).
19. Vaddireddy H., San O. Equation discovery using fast function extraction: a deterministic symbolic regression approach. *Fluids*. 2019. Vol. 4,

no. 2. P. 111. URL: <https://doi.org/10.3390/fluids4020111> (date of access: 03.07.2026).

20. Williams R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*. 1992. Vol. 8, no. 3-4. P. 229–256. URL: <https://doi.org/10.1007/bf00992696> (date of access: 04.07.2026).

21. GitHub - viktornemchenko-nure-ops/NURE---Thesis-RL-PPO-RPN. *GitHub*. URL: <https://github.com/viktornemchenko-nure-ops/NURE---Thesis-RL-PPO-RPN> (date of access: 11.07.2026).