КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка підсистеми калібрування багатопараметричної системи на основі генетичного алгоритму»

ГАРАНСЬКИЙ КОСТЯНТИН ОЛЕКСІЙОВИЧ

(прізвище, ім’я та по батькові студента повністю)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А

„____" _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка підсистеми калібрування багатопараметричної системи на основі генетичного алгоритму»

Виконав: студент 4-го курсу, групи КН-20-1

Спеціальності: 122 «Комп`ютерні науки

Спеціалізація: «Інформаційні управляючі системи та технології»

(шифр і назва напряму підготовки, спеціальності)

Гаранський К.О.
(прізвище та ініціали)

Керівник к.т.н., доц. Горда О.В.
(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є.Є.
(прізвище та ініціали)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Спеціалізація: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

_____
„___” _____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Гаранський Костянтин Олексійович
Тема роботи: Розробка підсистеми калібрування багатопараметричної системи на основі генетичного алгоритму

затверджена наказом ректора КНУБА № 2650/2 від 18.11.2023.
2. Керівник роботи: Горда Олена Володимирівна, к.т.н, доцент кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Методи та моделі налаштування робота маніпулятора
Р.3. Програмна реалізація
Р.4. Ергономіка інформаційних технологій
5. Інформаційні слайди:
С.1. Етапи генетичного алгоритму
С.2. Рівні генетичного алгоритму
С.3. Налаштування генетичного алгортму
С.4. Датасет для тестування
С.5. Результат контрольного прикладу

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2024 р.
Р. 2. Алгоритмічне та математичне забезпечення	Лютий 2024 р.
Р. 3. Розробка програмного забезпечення	Березень 2024 р.
Тестовий приклад програми	Квітень 2024 р.
Р. 4. Ергономіка інформаційних технологій	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	доц. Ачкасов І.А.		
Прийом програмного продукту	доц. Рябчун Ю.В.		

8. Дата видачі завдання: 18.11.2023

Завідувач _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Керівник _____ Горда О.В.
(підпис) (прізвище та ініціали)

Студент _____ Гаранський К.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Гаранський К.О. Розробка підсистеми калібрування багатопараметричної системи на основі генетичного алгоритму.

Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки», спеціалізація: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2024.

Робота присвячена налаштуванню роботів-маніпуляторів з використанням генетичних алгоритмів. Описано методи налаштування, зокрема кінематичні моделі та генетичні алгоритми. Наведено результати тестування, які підтверджують високу точність та ефективність алгоритму.

Ключові слова: роботи-маніпулятори, генетичні алгоритми, налаштування робота, штучний інтелект, адаптивне управління, машинне навчання, алгоритми оптимізації.

SUMMARY

Haranskyi. K. O. Development of a calibration subsystem for a multiparameter system based on a genetic algorithm.

Bachelor's thesis for a bachelor's degree in specialty: 122 “Computer Science”, specialization: "Information Management Systems and Technologies - Kyiv National University of Construction and Architecture - Kyiv, 2024.

The work is devoted to the customization of robot manipulators using genetic algorithms. The tuning methods, including kinematic models and genetic algorithms, are described. The test results are presented, which confirm the high accuracy and efficiency of the algorithm.

Keywords: robotic manipulators, genetic algorithms, robot tuning, artificial intelligence, adaptive control, machine learning, optimization algorithms.

ЗМІСТ

ВСТУП

Перед різними галузями людської діяльності гостро стоїть питання зменшення виробничих витрат, полегшення роботи працівників. Особливо це завдання стосується трудомістких та шкідливих, а також, високоточних сфер діяльності. Бурхливий розвиток науково-технічного прогресу сприяє появі у нашому житті з'являється все більшої кількості складних високоточних цифрових інструментів, які допомагають вирішувати наші повсякденні проблеми, автоматизують робочі процеси та відкривають нові можливості. До таких автоматизованих систем відносяться пристрої, якими людина може керувати в ручному режимі і вони називаються роботами маніпуляторами.

На даний момент часу завдання розробки та впровадження робототехніки актуальне, оскільки застосування роботів-маніпуляторів дозволяє автоматизувати практично будь-який процес промисловості.

Сучасні роботи – високотехнологічні пристрої, що виконують виробничі, транспортні, сервісні та навчальні функції. Вони здатні замінити не тільки одну людину, а кілька десятків професіоналів, у багатьох галузях та працювати у високопродуктивному та інтенсивному режимі; забезпечують високу точність виконуваних операцій; можуть імітувати біокінетику тіла людини та тварин та допомагають повернути мобільність людям з обмеженими фізичними можливостями; ефективно виконують свою функцію у складних умовах експлуатації, наприклад, за наявності радіації та дії високих температур. Вже сьогодні знайшли своє застосування виробничі маніпулятори, роботи-безпілотники, інтерактивні іграшки, роботи-хірурги, побутові андроїди, роботи-собаки, дрони, роботи-укладачі цегли та інше.

Мета роботи: аналіз та розробка адаптивного алгоритму управління роботом маніпулятором на основі ланцюгових механізмів.

Об’єкт дослідження: система управління роботом маніпулятором.

Предмет дослідження: методи, моделі та алгоритми управління функціональною системою робота маніпулятора.

Методи дослідження: методи технічної кібернетики, методи системного аналізу, аналітична геометрія, евристичні алгоритми.

Робота присвячена розробці автоматизованої підсистеми калібрування роботів маніпуляторів за допомогою генетичних алгоритмів. Основною метою проекту є підвищення точності пересування маніпулятора шляхом корекції кутів шарнірів, що компенсують похибки у виконанні команд.

В першому розділі проведений аналіз сфер застосування роботів-маніпуляторів, проблем їх розробки та впровадження. Розглянуто значення штучного інтелекту для покращення характеристик маніпуляторів і зроблено висновок про необхідність розробки адаптивних методів управління.

Другий розділ присвячений аналізу методів та моделей налаштування маніпуляторів. Розглянуто кінематичні моделі, математичні та штучноінтелектуальні методи, зокрема генетичний алгоритм. Описано етапи генетичного алгоритму та його параметри.

У третьому розділі описана реалізація генетичного алгоритму для налаштування маніпулятора. Розглянуто побудову хромосом, формування дерев, оцінку пристосованості, схрещування та мутацію. Наведено приклади коду та результати тестування.

Четвертий розділ аналізує результати тестування генетичного алгоритму, досягнуті показники точності та ефективності. Зроблено порівняння з іншими методами та висновки про переваги генетичного алгоритму, а також рекомендації щодо його вдосконалення.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка й аналіз проблеми

Сучасні маніпулятори – це потужні та універсальні інструменти, які використовуються в найрізноманітніших галузях, починаючи від галузей виробництва, де маніпулятори досягають великих масштабів, закінчуючи мікроскопічними розмірами у сфері медицини.

З розвитком штучного інтелекту відкриваються нові можливості для покращення характеристик маніпуляторів, роблячи їх більш гнучкими, адаптивними та автономними. У жорсткому конкурентному середовищі використання маніпуляторів може стати ключовим фактором для підвищення продуктивності, гнучкості та конкурентоспроможності компаній. Також актуальною проблемою є дефіцит робочої сили, з яким в багатьох випадках вже може впоратись робот. За останні роки технологічного прогресу немає такої галузі промисловості, де б не використовували роботу маніпуляторів. Їхня точність та ефективність використовується як при збиранні та фарбуванні автомобілів так і при пакуванні продуктів харчування.

У найскладніших виробничих процесах роботи можуть виконувати керуючі та рухові функції. Завдяки застосуванню роботів-маніпуляторів на виробництві можна скоротити кількість помилок під час виробничого процесу, скоротити браковані вироби, а також зменшити втрати сировини та знизити рівень травмонебезпеки. За допомогою робототехніки підприємство можна оптимізувати під різні технології, забезпечити належний рівень праці та безпеки.

За словами відомого вченого-робототехніка Массачусетсьго технологічного університету Родні Брукса, робототехніка ще перебуває на початковій фазі розвитку. І щоб перевести її на якісно новий рівень, необхідно, щоб роботи, як мінімум, набули відмінних навичок розпізнавання мови та жестів людини, дрібної моторики, синтезу та аналізу зовнішнього середовища та соціального спілкування. Відповідно, розвиток робототехніки безпосередньо залежить від розвитку

суміжних галузей, адже тільки так можна розширити виконувані завдання та покращити якісні характеристики обладнання. Створення нових матеріалів, впровадження адаптивного програмного забезпечення, розробка нових джерел енергії – все це дозволить у найближчому майбутньому створити нові та ще більш досконалі роботи.

Адаптивне машинне навчання, як наука, що вивчає проблеми аналізу, обробки та подання даних у цифровому вигляді сьогодні являється одним з головних напрямків цифрових та комп'ютерних технологій. Ця сфера наукових досліджень швидко розвивається. В ІТ-технологіях виникли такі поняття, як «слабкий штучний інтелект» та «сильний штучний інтелект». Під першим розуміється системи, що навчаються, здатні вирішувати окремі завдання, під другим – розумні системи, під силу яким інтелектуальне мислення і вибудовування причинно-наслідкових зв'язків на основі обробки різних даних. Таким чином, машинне навчання дає можливість зробити з робота аналітика, здатного на основі закономірностей побудувати прогноз для заданої величини або проаналізувати та скоригувати ефективність застосовуваної методики лікування, розподілу логістичних потоків, завантаженості виробничих ліній, виявити помилки в технології та інше.

При керуванні маніпулятором важливою для вирішення задачею є точність рухів для досягнення потрібної точки.

В даний час, як правило, в основі принципів побудови систем управління маніпуляторами лежать методи розрахунку керуючого сигналу – обертаючого моменту, що дозволяє здійснювати переміщення маніпулятора з урахуванням деформацій і коливань, обумовлених пружністю складових його ланок. Сигнали керування крутним моментом, розраховуються для переміщення робочого органу маніпулятора по заданій траєкторії без його істотних коливань.

Виходячи з вище сказаного, можна зробити висновок, що розробка адаптивних методів та алгоритмів управління функціоналом робота маніпулятора є актуальним завданням.

1.2 Аналіз ринку застосування роботів маніпуляторів

Зазвичай в робототехніці виділяють два напрямки – промислова та сервісна робототехніка, але сьогодні роботи набули настільки широкого застосування, що такий поділ став не актуальним. Обидві галузі переживають зростання, проте причини цього дуже різні. Промислова робототехніка зростає (в середньому на 15% на рік) за рахунок стрімкої роботизації китайської економіки і, в той час як зростання сервісної робототехніки має глибші причини: більшість світової економіки є сервісною економікою. Саме тому сервісна робототехніка показує більш значне зростання вже зараз (на рівні 25% на рік) за відносно менших у абсолютному значенні цифрах порівняно з промисловою [1].

За даними американської компанії Mordor Intelligence ринок роботизованого оснащення в минулому році оцінювався в 5,3 мільярда доларів США (рис.1 1), і очікується, що середньорічний темп зростання становитиме 11,5% протягом прогнозованого періоду (2024-2029 рр.) і досягне 10,18 мільярда доларів США за наступних п'яти років.

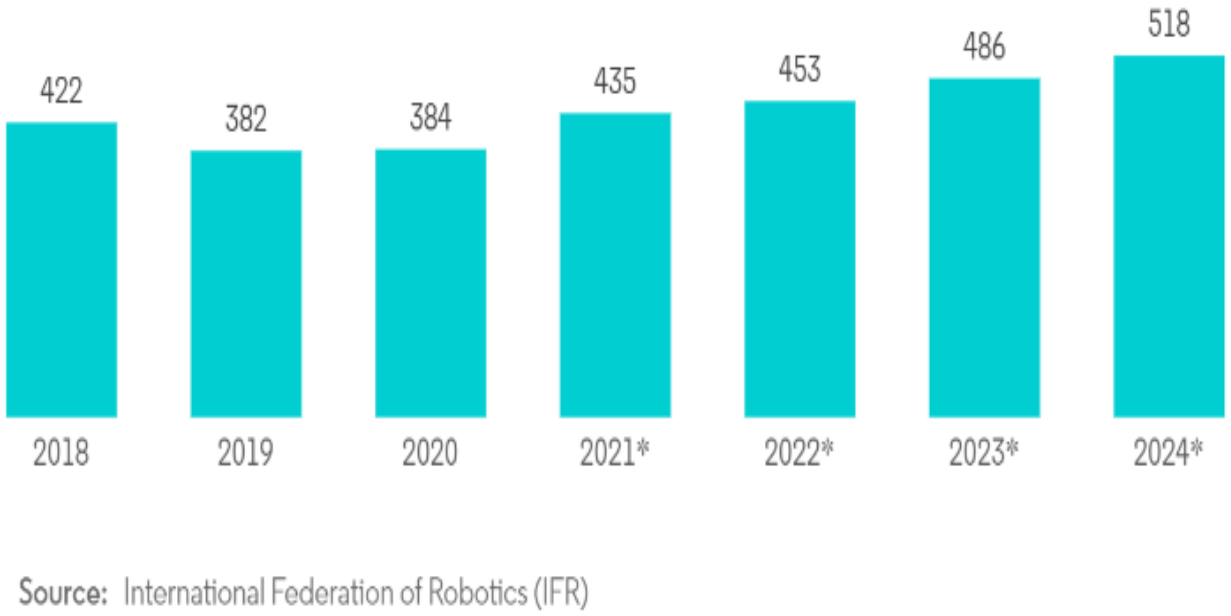


Рис. 1.1 Орієнтовне річне впровадження робототехніки в тис. одиниць (2018-2024)

Невпинно зростає і виробництво роботизовано техніки (рис. 1.2).

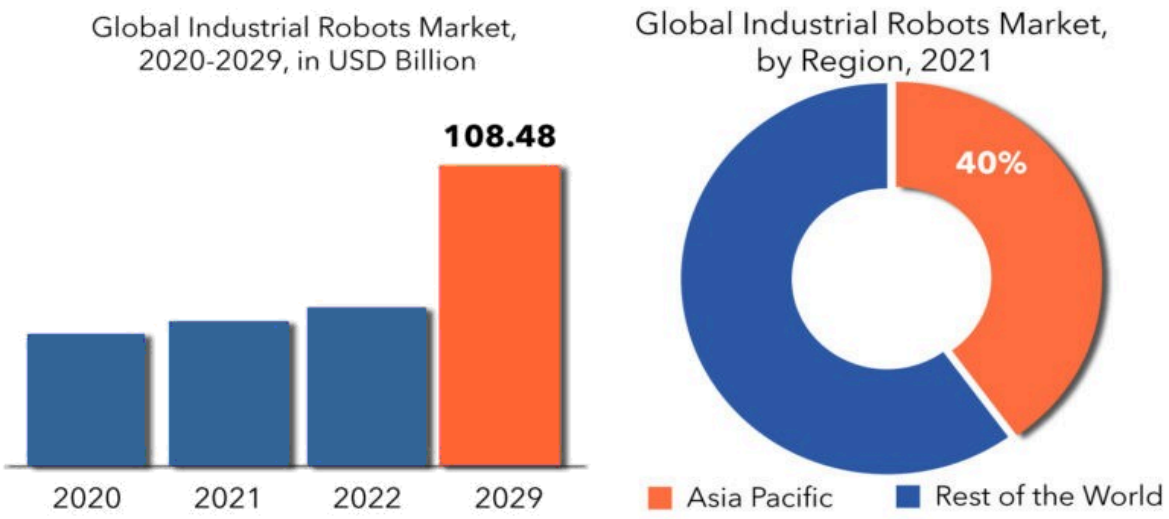
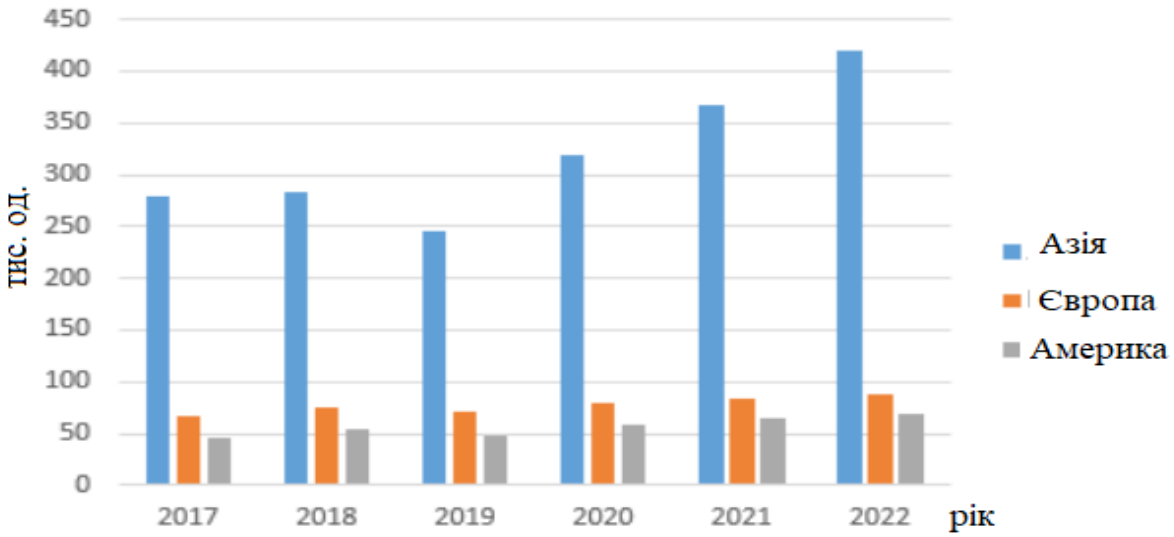


Рис. 1.2 Діаграмма виробництва робототехніки та його перспектив

За даними IFR (International Federation of Robotics) найбільша кількість встановлюваних промислових роботів припадає на країни Азії. Двісті вісімдесят п'ять тисяч штук у 2019 році. Кількість експортованих роботів у Європі в 2019 році складає 27% від азіатських поставок (або сімдесят чотири тисячі штук), Американські поставки займають 19% (або ж п'ятдесят чотири тисячі штук від азіатських постачань) [12]. Статистичні дані встановлюваних промислових роботів по регіонах світу представлені на (рис.1.3).



1.3 Статистичні дані встановлюваних промислових роботів по регіонах світу

За даними PwC (PricewaterhouseCoopers – міжнародна аудит –консалтингова корпорація), Японія та Німеччина лідирують за кількістю патентів та ліцензій на робототехніку та автономні системи (24 % та 22 % відповідно), у той час як у США зосереджено 17 % патентів усього світу, а в Китаї та Південній Кореї – 13 % та 10% відповідно[21].

Найбільш перспективними є наступні напрямки впровадження роботів:

- безпілотний пасажирський транспорт;
- роботи для комерційних просторів;
- роботи-співрозмовники (помічники);
- логістичні роботи та безпілотний вантажний транспорт;
- колаборативні роботи;
- промислові екзоскелети;
- роботи для сільського господарства.

Роботизація стає основним інноваційним процесом сучасної економіки, збільшуючи як обсяги виробництва розвинених країн, а й знижуючи витрати під час виготовлення промислової продукції. Згідно з дослідженнями групи Boston Consulting Group (BCG) відбувається процес зменшення витрат на робочу силу, і навпаки, збільшується рівень ККД співробітників компаній, де використовуються роботи.

1.3 Аналіз науково-практичних досліджень та напрацювань в галузі робототехніки

В галузі робототехніки можна виділити декілька напрямків досліджень:

- аналіз та конструювання механізмів;
- застосування роботів;
- розробка програмного забезпечення;
- впровадження методів штучного інтелекту.

Почнемо з першого пункту. В наш час важливо, щоб система не лише працювала, а й була адаптована до навколишнього середовища. Це питання стало головною темою роботи “Динамічна поведінка біоінспірованих структур: дизайн, механізми та моделі” [30]. В ході опису біоінспірованих структур, вчені Університету Північної Кароліни приводять у приклад конструкції, натхненні жуками. Їх крила легкі, але жорсткі, ефективно захищають носія від зовнішніх ударних навантажень (можуть витримувати ударне навантаження до 23 Н). У своїй статті автори кажуть, що незважаючи на відносно обмежену кількість компонентів, природні матеріали завжди служать набором інструментів для раціоналізації тонких архітектур і постійно захоплюють вчених та інженерів проектуванням штучних структур, натхненних природою, з видатними механічними характеристиками, такими як надвисока питома міцність, міцність, поглинання енергії та ударостійкість.

Альтернативний погляд на існуючі маніпулятори описали у своїй роботі Науковці Загребського університету. Задачею дослідження було виготовити та випробувати маніпулятор з одним ступенем свободи, приводи якого були представлені у вигляді пневматичних штучних м’язів (pneumatic artificial muscles) . У процесі дослідження було розроблено спрощену математичну модель системи, яка була використана в процедурі синтезу регулятора, а також проведено тестування практично реалізованої системи для завдання транспортування та сортування об'єктів. Контролер зі зворотним зв'язком за станом використовувався в контурі керування системою. На основі експериментів, проведених під час роботи було зроблено висновок, що PAM є дуже придатними приводами для нових типів промислових роботів і маніпуляторів. Їх характеристики дозволяють легко збирати їх, одночасно покращуючи продуктивність порівняно зі звичайними пневматичними приводами [7].

Наступним пунктом є застосування робототехніки. У своєму звіті про розмір ринку промислової робототехніки Grand View Research описують ринок кінцевого застосування роботів [11], де найбільшу частку зайняв електронний сегмент, який займає понад 25% усього ринку. Очікується, що сегмент кінцевого використання

хімікатів, гуми та пластмас зростатиме з найвищим CAGR понад 12% протягом прогнозованого періоду. Зростання сегмента можна пояснити необхідністю підтримувати узгодженість у таких завданнях, як вимірювання та тестування. Крім того, робототехніка може працювати з токсичними матеріалами без ризику погіршення здоров'я.

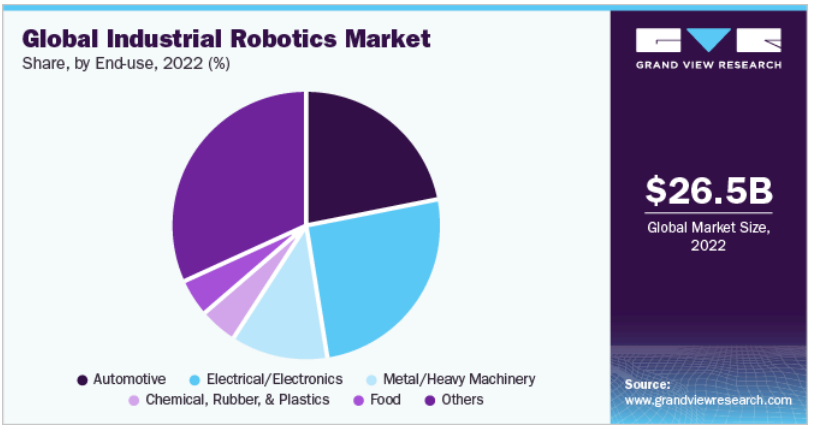


Рис. 1.4 Статистичні дані глобальної частки ринку промислової робототехніки, за кінцевим використанням

У звіті також прогнозується зростання обсягів і доходів на глобальному, регіональному та національному рівнях, а також надається аналіз останніх галузевих тенденцій у кожному з підсегментів з 2018 по 2030 рік. Для цього дослідження Grand View Research сегментував глобальний ринок промислової робототехніки (Рис. 1.5).

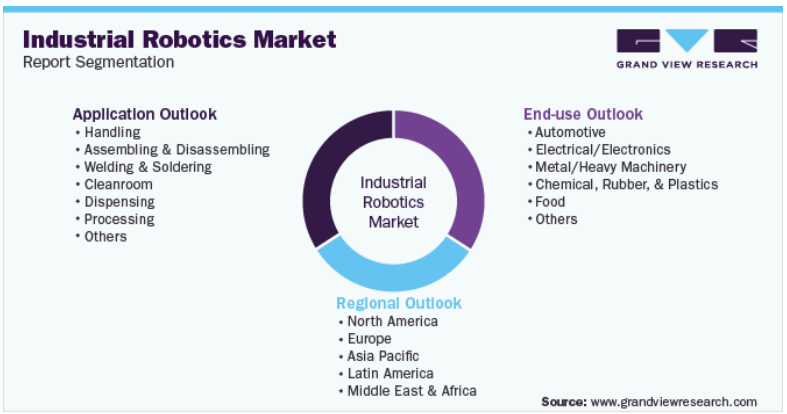


Рис. 1.5 Статистичні дані глобальної частки ринку промислової робототехніки, за кінцевим використанням

Програмне забезпечення робототехніки є основою, у становленні непов’язаної між собою електроніки у цілісну структуру, яка може виконувати певний набір функцій. Науковці Національного Університету Сан-Хуану у своїй роботі показали гарний приклад розробки програмного забезпечення, для взаємодії з існуючим маніпулятором [19]. Система з двох програм працює таким чином, що виконання алгоритму управління ізольоване від передачі сигналів між програмним забезпеченням та апаратурою робота. Це дозволяє скорочувати час і зусилля при реалізації різних алгоритмів управління. Результатами роботи стала відкрита система, яка дозволяє впроваджувати нові модулі, та застовувати їх на різного виду маніпуляторах. У процесі дослідження проведено позитивні оцінювання програми на роботах з використанням двох різних алгоритмів управління.

Останнім пунктом даної частини роботи є впровадження методів штучного інтелекту. "Міжнародний журнал з досліджень у прикладних науках та інженерії технологій" (IJRASET) опублікував огляд алгоритмів штучного інтелекту, що використовуються для роботизованих маніпуляторів [18]. У статті описується перелік проблем сучасного маніпулятора: планування траєкторії руху, управління рухом, ідентифікація динамічних параметрів та візійна система робота. Головна мета роботи, показати часткові вирішення цих проблем і показати тренд розвитку алгоритмів штучного інтелекту для застосування в роботизованих маніпуляторах.

1.4 Визначення цілей дослідження

Головною метою роботи є підвищення ефективності роботи робота маніпулятора за рахунок створення алгоритмів формування програмних траєкторій рухів адаптивних маніпуляційних роботів, призначених для широкого кола виробничих операцій. У складних системах до яких відносяться роботи маніпулятори загальна мета відокремлена від конкретних засобів її досягнення, тому вибір рішення в системі вимагає великої роботи з пов'язування загальної

мети із засобами її реалізації шляхом декомпозиції цілей і побудови «дерева цілей». В основу побудови дерева цілей можуть бути покладені два принципи – суто цільове та послідовне дезагрегування проблеми за схемою: цілі – заходи для досягнення цілей – шляхи реалізації заходів – окремі завдання.

Реалізація загальної мети пов'язана з досягненням низки внутрішніх цілей системи. Для технічної системи ці цілі називаються основними цілями системи і до них належать: цілі розвитку, функціонування, ефективності.

Ефективність роботи системи в даній роботі визначається за наступним переліком показників ефективності:

- похибка калібрування, яка включає в себе середнє значення абсолютної похибки, середнє квадратичне значення похибки та відносну похибку;

- якість калібрування, визначається за коефіцієнтом детермінації;

- час знаходження оптимального рішення;

Представимо мету роботи у вигляді дерева цілей взявши за основу другий принцип дезагрегування цілей (рис. 1.6).

Рис. 1.6 Дерево основних цілей створення адаптивного управління системи робот маніпулятор

Реалізація мети здійснюється вирішенням наступних основних завдань:

- провести аналіз проблем пов’язаних з впровадженням маніпуляторів, виявити “вузькі” місця;

- обрати базовий алгоритм для вирішення задачі налаштування (калібрування) маніпулятора, та виконати постановку задачі згідно до обраного алгоритму;

- побудувати математичну модель для вирішення задачі налаштування маніпулятора;

- розробити програмне забезпечення для реалізації задачі налаштування.

Структура основних задач наведена на рис. 1.7.

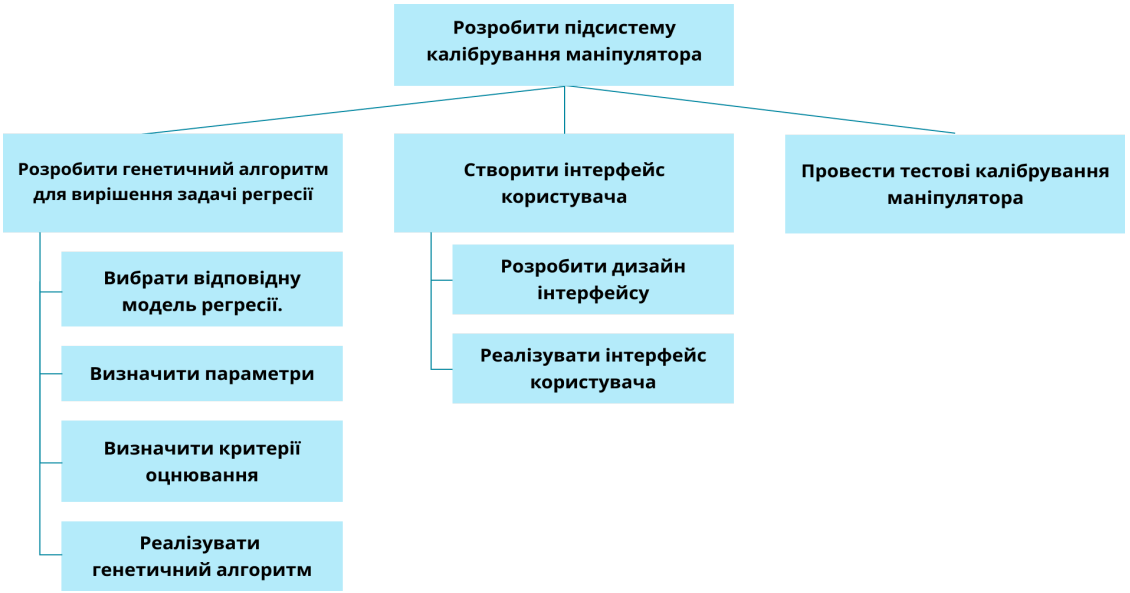


Рис. 1.7 Дерево основних задач налаштування робота маніпулятора

1.5 Аналіз системи “Робот маніпулятор”

1.5.1 Класифікація роботів маніпуляторів

Робот маніпулятор – складний пристрій, який класифікується за великою кількістю ознак. Умовно виділяють наступні види класифікацій роботів маніпуляторів, які об’єднуються за технологічним призначенням, особливостям конструкції та іншим параметрам:

Класифікація роботів-маніпуляторів за типом монтажу наведена на рис 1.8.

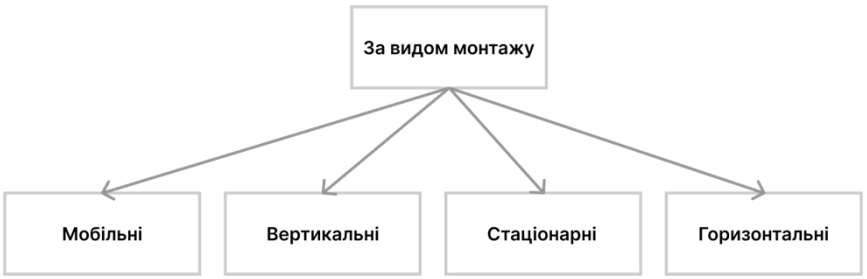


Рис. 1.8 Класифікація роботів-маніпуляторів за типом монтажу

Основною перевагою мобільних пристроїв є їхні невеликі розміри, вони не займатимуть багато місця при переміщенні, але при цьому з легкістю подолають перешкоди, а з мінусів – висока вартість та робота з невеликими вагами.

Найпопулярнішими роботами-маніпуляторами є стаціонарні. Вони відрізняються за типом закріплення (підлоговий, стельовий, настінний), серед переваг – широкий радіус застосування та збільшена вантажопідйомність. Горизонтальні можуть мати протяжність до кількох десятків метрів і використовуються в місцях експлуатації декількох об'єктів. Вертикальні – використовуються там, де мало місця.

Класифікація за типом застосування наведена на рис. 1.9.

Рис. 1.9 Класифікація роботів-маніпуляторів за типом застосування

Автономні роботи-маніпулятори (програмовані) – такі маніпулятори, які можна розмістити біля оброблюваних об'єктів, а також застосовувати поруч з іншими програмованими пристроями. Якщо оснастити робота колесами всенапрямленого руху, то такий пристрій може переміщатися в будь-які важкодоступні місця для людини без його допомоги. Роботи-маніпулятори з ручним керуванням забезпечують високу точність при виготовленні деталей. Колаборативні маніпулятори або коботи – маніпулятори, у процесі вироблення яких висуваються найжорсткіші умови до конструкційної безпеки та програмного забезпечення, такі заходи повинні мінімізувати травмування працівників.

На рис. 1.10 зображена класифікація роботів-маніпуляторів за типом виконуваних функцій.

Рис. 1.10 Класифікація роботів-маніпуляторів за типом виконуваних функцій

За допомогою збіркових маніпуляторів в можна збирати вироби будь-яких розмірів без безпосередньої участі людини, що зменшує необхідність використання пристроїв великої вантажопідйомності. Паяння та зварювання. Завдяки роботам можна досягти високого рівня зварювальних швів і стабільності дуги, збільшується швидкість роботи і можна застосувати дуже низький струм, те, чого не можна при ручному апаратному зварюванні. Обробка матеріалів. Маніпулятором можна досягти зовсім гладких поверхонь за умови підвищення міцності стінок металу та його зміцнення більше ніж на 30%, оскільки застосовується холодна ковка. Очищення, фарбування, дозування. Пристрій можна використовувати для очищення поверхонь водою під великим тиском, піскоструйної обробки, або фарбування вже готового товару. Різка та обробка. Така процедура дуже небезпечна для людини, для робота вона не принесе ніякої шкоди, також роботи дуже точно виконують різку, при цьому скорочується час обробки і мінімізується втрата матеріалу. Будівельні. У будівельній сфері роботи-маніпулятори дозволяють збільшувати швидкість будівництва до максимальної з збереженням максимальної точності. Пристрій працює цілодобово, не піддається зовнішнім погодним факторам та технічним особливостям процесу.

На рис. 1.11 зображена класифікація роботів-маніпуляторів за типом приводу.

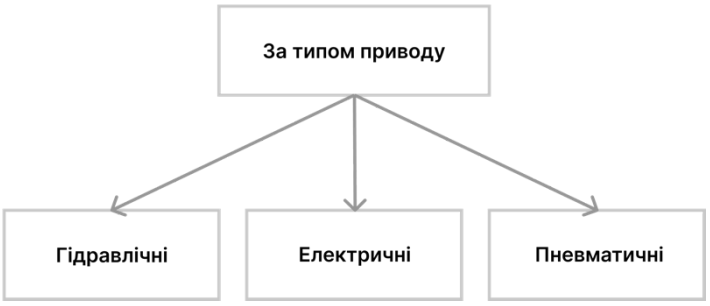


Рис. 1.11 Класифікація роботів-маніпуляторів за типом приводу

Гідравлічні роботи-маніпулятори. Для руху виконавчого органа застосовують рідину і використовують такі пристрої для вантажів масою більше 100 кг. Електричні. За допомогою електричного струму викликаються в рух, характеризується хорошою продуктивністю і точністю дій. Пневматичні. Робочий орган рухається за рахунок енергії стиснутого повітря, яке накачується компресором в пневмолінію.

Класифікація за корисним навантаженням наведена на рис. 1.12.

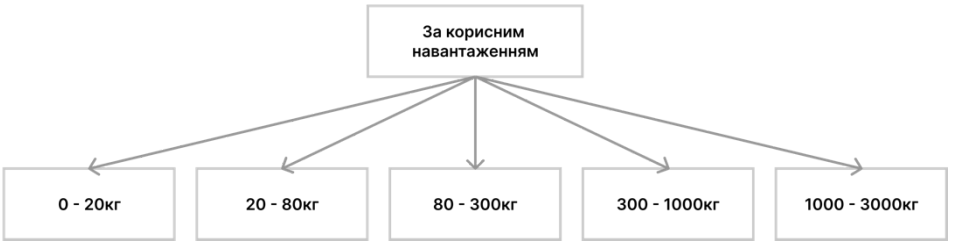


Рис. 1.12 Класифікація роботів-маніпуляторів за корисним навантаженням

- 0 – 20кг: такі роботи мають невеликі розміри, швидко рухаються та точні при невеликій вантажопідйомності і невеликій робочій зоні.
- 20 – 80кг: універсальні в своєму застосуванні та мають хорошу продуктивність, їх використовують для багаторазових швидких процесів.
- 80 – 300кг: придатні для важких умов виробництва, мало піддаються зношуванню, надійні та прості в своєму використанні.
- 300 – 1000кг: використовуються для складних умов, наприклад, ними повертають кузови автомобілів або застосовують у цехах для лиття.

- 1000 – 3000кг: легко справляються з дуже важкими вантажами, з точністю переміщують дуже великогабаритні вантажі, тому застосовні на завантаженні та розвантаженні.

1.5.2. Структурний аналіз

За своєю структурою маніпулятор – це багатоланкова система, між окремими елементами якої існують механічні зв'язки. Серед елементів робота маніпулятора варто виділяти наступні: стійка, ланки, зв’язки та функціональна кінцівка [5] (рис. 1.13).

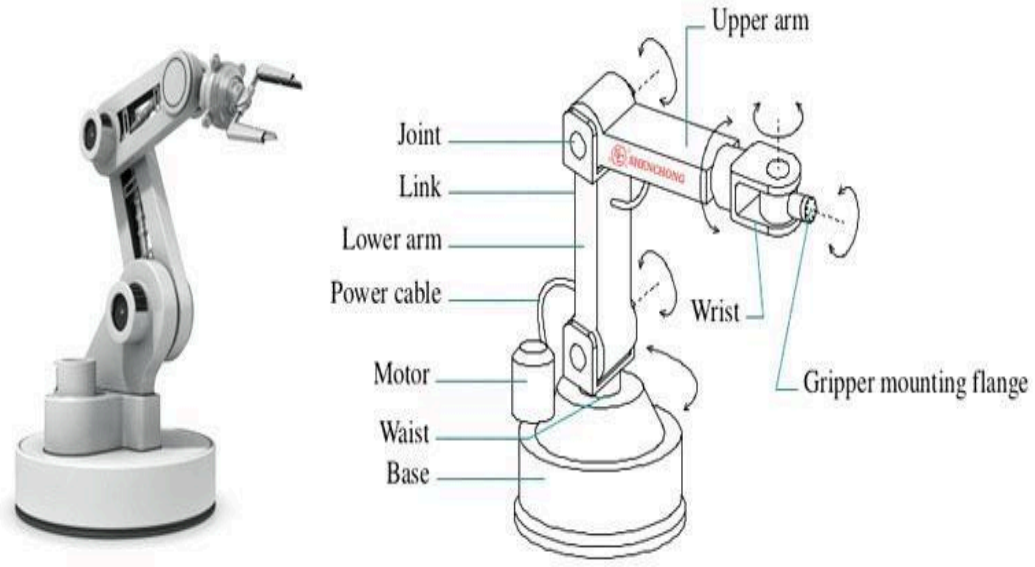


Рис. 1.13 Структура робота маніпулятора

Структурна схема такого механізму представляє собою графічне зображення, на якому показані всі його елементи, кінематичні пари та їх взаємне розташування. Кінематичною парою називають зв'язок між двома елементами маніпулятора, який дозволяє їм здійснювати певний вид руху відносно один одного [9]. Графічне зображення елементів схеми виконується з урахуванням прийнятих умовних позначень. На рисунку 1.14 наведені умовні позначення кінематичних пар.

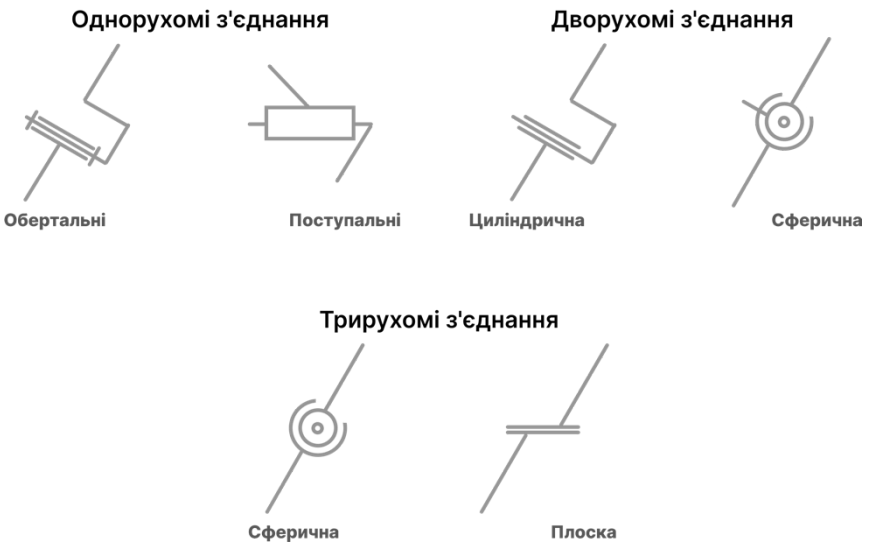


Рис. 1.14 Позначення кінематичних пар

Кінематичним ланцюгом називається система зв'язків, які утворюють між собою кінематичні пари. Ланцюг, в якому кожен зв'язок входить не більше, ніж у дві кінематичні пари, називається простим. Незамкненим називається такий кінематичний ланцюг, в якому є зв'язки, що входять лише в одну кінематичну пару.

Розглянемо структурну схему антропоморфного маніпулятора, тобто схему, яка відповідає механізму руки людини (рис. 1.15). Цей механізм складається з трьох рухомих вузлів і трьох кінематичних пар: двох трирухомих кульових з’єднань і одного однорухомого обертального з’єднання.

Структур комп (ланки, шарніри)

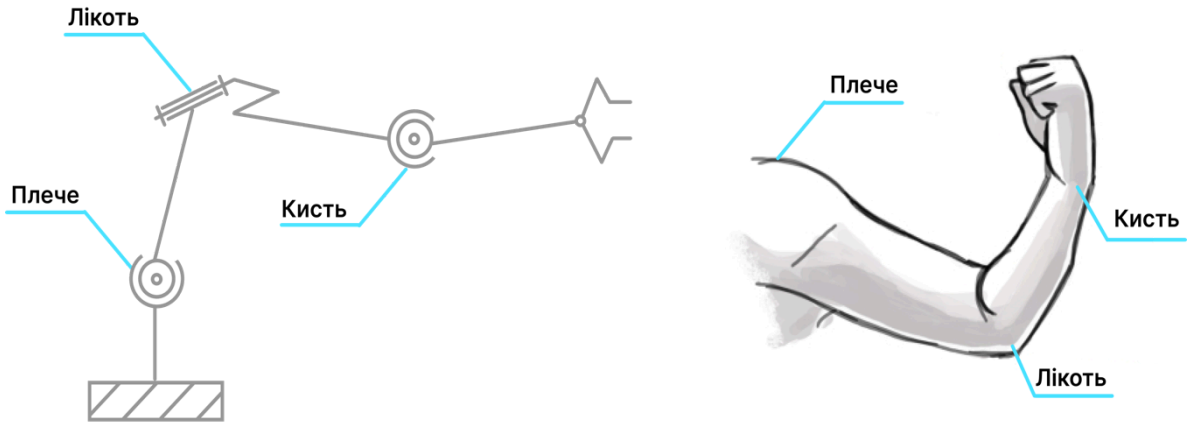


Рис. 1.15 Схема антропоморфного маніпулятора
у порівнянні з рукою людини

1.5.3. Функціональний аналіз

В залежності від області застосування можуть використовуватися різні схеми побудови механічної частини маніпулятора. Серед них розділяють «руку» та «зап’ясток».

Основна конструкція «руки» являє собою послідовність ланок, з'єднаних між собою обертальними та поступальними зв'язками. За характером та кількістю зв'язків можна виокремити наступні категорії організації просторових переміщень (рис. 1.16):

- роботи з декартовою системою координат;
- роботи з циліндричною системою координат;
- роботи зі сферичною системою координат;
- роботи з обертальною системою координат.

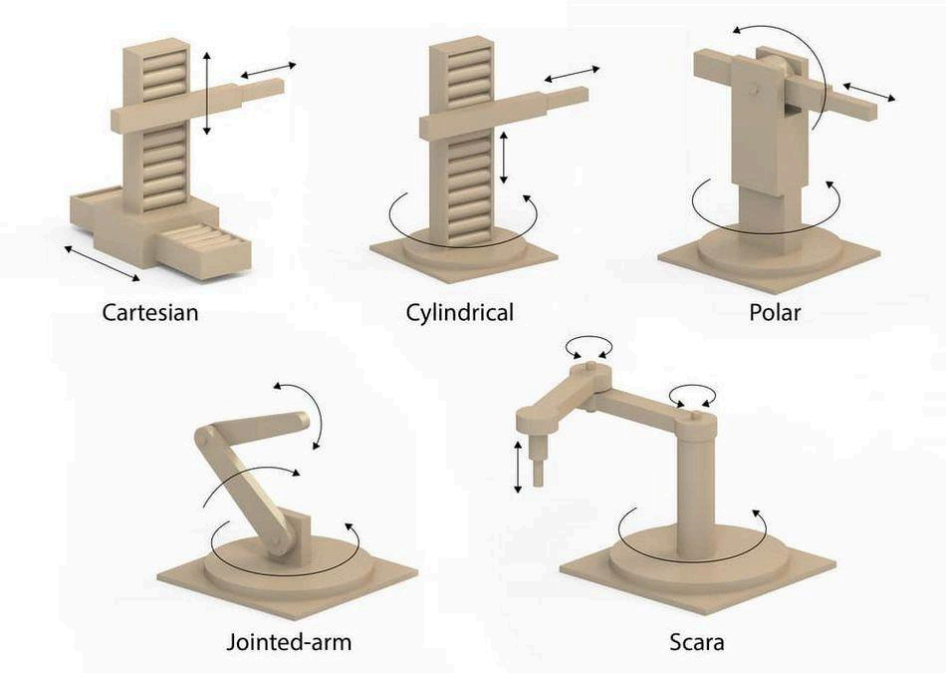


Рис. 1.16 Категорії організації просторових переміщень маніпулятора

Більшість маніпуляторів, що випускаються зараз, відносяться до числа роботів з обертальною системою координат. Вони забезпечують найбільший обсяг робочої зони, в якій може здійснюватися рух, їх структура дозволяє досягати

заданого положення і орієнтації робочого органу, в тому числі і при накладенні обмежень на можливі переміщення, що виникають при наявності перешкод в робочій зоні або потрібних для забезпечення безпеки експлуатації.

Перевагами роботів з обертальною системою координат є проста конструкція, висока надійність, широкий діапазон швидкостей та можливість роботи в умовах ударних навантажень. З мінусів можна відокремити складність управління, та обмежену точність позиціонування

Способи управління промисловими роботами підрозділяється на дві групи : програмний та адаптивний. Програмний метод – це найпростіша система управління з усіх можливих, в основному застосовується на промислових об'єктах для управління роботами. Моделі, які працюють під таким типом управління, часто не мають сенсорної частини в своєму агрегаті. Дії всі точні і одноманітні, змінити рухи робота можна лише перепрограмувавши його. Апаратом управління може виступати промисловий комп'ютер PC/104, рідше зустрічається MicroPC.

Адаптивний метод управління передбачає наявність сенсорної системи в середині робота. В систему надходять сигнали, сприйняті датчиками, встановленими по периметру робота. Вся отримана інформація аналізується і на основі результатів аналізу обирається певна відповідь на оточуючі умови (зупинка, рух, перехід до наступного етапу роботи). У такому випадку програмування, частіше за все, базується на методі використання штучного інтелекту. В системі можливе ручне управління людиною дистанційно або стаціонарно [24].

У промислових роботах застосовуються системи програмного управління різних видів. Абстрактно вони діляться на 3 види: циклові, позиційні і контурні, проте кожна з них має ряд видів залежно від характеру обслуговуваної технологічної операції, кінематичної структури робота або типу приводів[13].

Функціональна схема управління маніпулятором у загальному вигляді представлена на рис. 1.17.

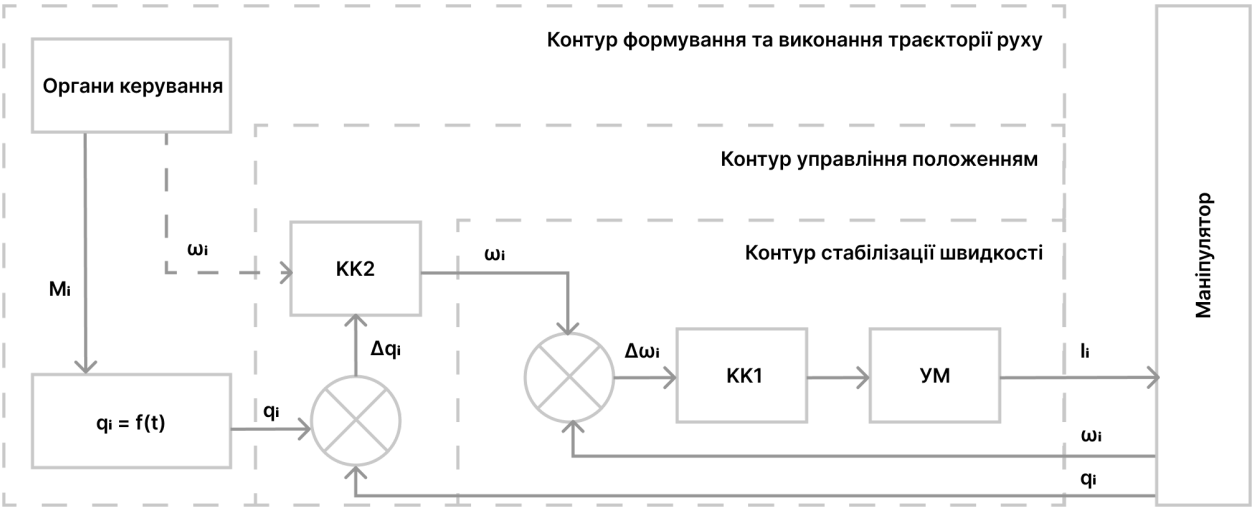


Рис. 1.17 Функціональна схема управління маніпулятором

На схемі (рис. 1.17) позначені наступні елементи:

- ККІ1 – коректуючий контур (регулятор) стабілізації швидкості обертання ланок маніпулятора;
- УМ – підсилювач потужності, що перетворює широтно-модульовані сигнали управління у вихідний струм виконавчих органів маніпулятора;
- ККІ2 – коректуючий контур (регулятор) положення ланок маніпулятора;
- *qi* – миттєві кути орієнтації ланок маніпулятора;
- *ωi* – миттєві кутові швидкості повороту ланок маніпулятора;
- *Ii* – струми у виконавчих електродвигунах маніпулятора.

Під життєвим циклом роботи маніпулятора розуміється певна кількість виконаних функцій за одинарний сеанс, тобто початок роботи, виконання повного комлексу переміщень та кінець робочого циклу. В процесі постійної роботи, у зв’язку з недосконалістю навколишнього середовища може виникати, або збільшуватись похибка точності переміщень, що негативно впливатиме на кінцевий результат роботи маніпулятора.

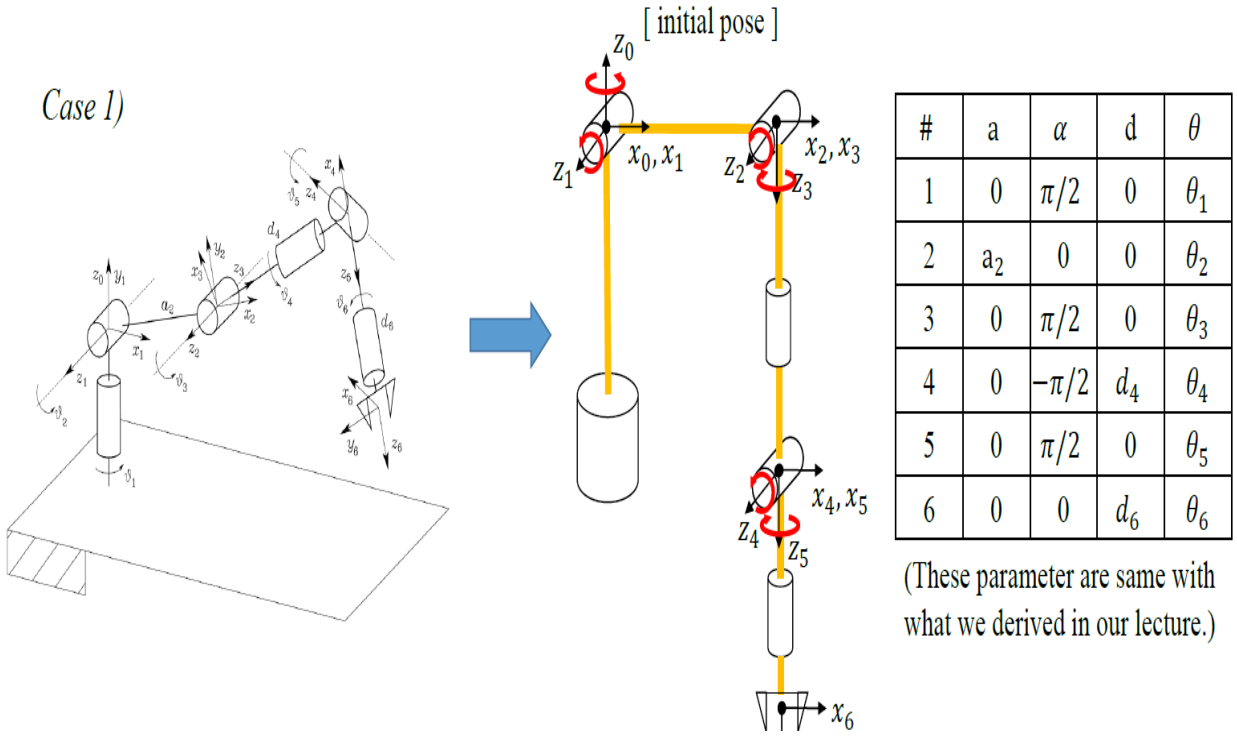
1.5.4 Параметричний аналіз

Параметричний аналіз роботів-маніпуляторів зосереджується на визначенні та аналізі параметрів, які характеризують кінематичні ланцюги маніпуляторів.

Одним із найпоширеніших методів представлення таких параметрів є метод Денавіта-Хартенберга (DH параметри) [4] [25].

Метод Денавіта-Хартенберга є стандартом для опису кінематики роботів. Він дозволяє визначати положення та орієнтацію кожного суглоба маніпулятора відносно попереднього суглоба через чотири параметри:

- θi: Кут обертання навколо осі zi−1. Це змінний параметр для обертових суглобів.
- di: Відстань вздовж осі zi−1 між попередньою та поточною ланкою. Це змінний параметр для призматичних суглобів.
- ai: Відстань між осями zi−1 та zi вздовж осі xi.
- αi : Кут між осями zi−1 та zi навколо осі xi.



#	a	α	d	θ
1	0	π/2	0	θ_1
2	a_2	0	0	θ_2
3	0	π/2	0	θ_3
4	0	−π/2	d_4	θ_4
5	0	π/2	0	θ_5
6	0	0	d_6	θ_6

(These parameter are same with what we derived in our lecture.)

Рис. 1.18 Кінематична схема маніпулятора

Постановка задачі

В результаті проведеного аналізу маємо проблему, яка полягає у постійному контролі точності пересувань маніпулятора.

Зі структурно функціонального аналізу системи випливає проблема, яка полягає у калібруванні та постійному контролі точності пересувань маніпулятора. Кути, які програма передає на шарніри маніпулятора на справді можуть зазнавати спотворень і виконуватись з роботом з похибками, через що точність системи може падати. Основна задача – розробка алгоритму для автоматизованого налаштування, або калібрування робота маніпулятора.

Вхідними даними буде кінцевий стан, який визначається положенням останньої ланки робота маніпулятора і описується наступними параметрами (x, y), як координати, відліком яких є центр нерухомої основи (бази) та кут нахилу β в цій площині.

Вхідні дані:

1. Кінематична модель робота: Вона описується DH параметрами, які включають довжини ланок (ai), кути між ланками (αi), зсуви вздовж осей (di) та кути обертання (θi). Ці параметри визначають геометрію робота і впливають на його рухи.
2. Задане положення та орієнтація кінцевого ефектора: Цільові координати та орієнтація, які маніпулятор має досягти.
3. Виміряні координати кінцевого ефектора: Реальні координати, які були досягнуті маніпулятором після виконання команд.

Вихідні дані:

1. Скориговані кути шарнірів (θi′): Відкориговані значення кутів, які повинні компенсувати спотворення і похибки у виконанні команд, забезпечуючи точне досягнення заданого положення та орієнтації кінцевого ефектора.
2. Модель корекції кутів: Математична модель, яка визначає, як саме потрібно коригувати вхідні кути для досягнення більшої точності.

2. МЕТОДИ ТА МОДЕЛІ НАЛАШТУВАННЯ РОБОТА МАНІПУЛЯТОРА

2.1 Аналіз методів та моделей кінематичної моделі робота маніпулятора

Виходячи з параметричного аналізу видно, що задача залежить від значної кількості параметрів, а отже є багатопараметричною [22]. У даному розділі ми розглянемо, як цю задачу можна звести до задачі регресії. Задача керування роботом-маніпулятором має на меті перевести маніпулятор у конкретну точку простору, що задається координатами x, y, z, проте робот приходить у координати x', y', z', які не дорівнюють початково заданим. Наша задача полягає у тому, щоб маючи експериментальні дані знайти залежність та оптимізувати прицільність робота маніпулятора [2].

З попереднього розділу вже знаємо, що існує залежність між координатами робота та кутами кожної ланки. Для переходу з координат в кути між ланками і навпаки маємо згадати про поняття прямої та зворотної кінематики. Пряма кінематика займається визначенням положення і орієнтації кінцевого ефектора робота, виходячи з відомих значень кутів його зчленувань (суглобів). У випадку роботів-маніпуляторів, це завдання полягає в обчисленні координат і орієнтації кінцевої ланки (ефектора) в просторі. Основою для таких обчислень є кінематична модель Денавіта-Хартенберг. Зворотня кінематика, навпаки, займається визначенням необхідних кутів зчленувань для досягнення заданого положення і орієнтації кінцевої ланки (ефектора). Це завдання часто буває складнішим, оскільки може мати декілька розв'язків або взагалі не мати розв'язку у випадку недосяжності цільової точки.

Почнімо з прямої кінематики. Загальна матриця перетворення для кожної ланки визначається як:

$$A_i = \begin{pmatrix} \cos\cos(\theta_i) & -\sin\sin(\theta_i)\cos\cos(\alpha_i) & \sin\sin(\theta_i)\sin\sin(\alpha_i) & a_i\cos\cos(\theta_i) \end{pmatrix}$$

,

де αi - кут між осями Zi та Zi+1, ai - відстань між осями Zi та Zi+1 вздовж осі Xi, di Xi та Xi+1 вздовж осі Zi+1, θi - кут обертання навколо осі Zi, що дозволяє перейти від Xi до Xi+1.

Маючи матриці перетворень для кожної ланки, можна визначити кінцеве положення кінцевого ефектора робота. Остаточна матриця перетворень T для кінцевого ефектора обчислюється як добуток всіх індивідуальних матриць перетворень:

$$T = A_1 A_2 A_3 A_4 A_5 A_6 ,$$

де T є загальною матрицею перетворення, що містить інформацію про положення та орієнтацію кінцевого ефектора в робочому просторі.

Позиція кінцевого ефектора P визначається як:

$$P = T(0\ 0\ 0\ 1\ )$$

де P - це вектор положення кінцевого ефектора в координатній системі бази робота.

Зворотня кінематика використовує чисельні методи, такі як метод Ньютона-Рафсона та метод градієнтного спуску, щоб знайти розв'язок.

Метод Ньютона-Рафсона використовується для знаходження коренів рівнянь. В контексті оберненої кінематики він може бути застосований для мінімізації різниці між бажаним положенням ефектора та тим, що обчислюється на основі поточних значень кутів зчленувань.

Процес ітераційно оновлює значення кутів зчленувань за формулою:

$$\theta new = \theta old - J-1(\theta old) \cdot f(\theta old)$$

де θ - вектор кутів зчленувань, J - матриця Якобі, $f(\theta)$ - вектор функцій, що описують різницю між бажаним та поточним положенням.

Метод градієнтного спуску мінімізує функцію втрат, яка в даному випадку є різницею між бажаним та поточним положенням кінцевого ефектора. Градієнт функції втрат використовується для оновлення значень кутів:

$$\theta new = \theta old - \alpha \cdot \nabla L(\theta old),$$

де α - швидкість навчання (learning rate), $\nabla L(\theta)$ - градієнт функції втрат.

Застосувавши вищеописані методи можемо звести задачу до задачі регресії [23], у якій будемо шукати залежності кутів, відповідно до координат які ми вводимо та кутів, які отримуємо в результаті роботи програми.

Для вирішення вже такої задачі можна використовувати кілька підходів, які включають математичні методи, методи штучного інтелекту та евристичні методи.

З математичних методів найбільш відомими є наступні:

1) Лінійне програмування. Метод використовує лінійні рівняння та нерівності для моделювання задачі. Цей метод підходить для задач, де цільова функція і обмеження є лінійними.
2) Нелінійне програмування. Цей метод підходить для більш складних задач, де обмеження або цільова функція нелінійні, використовуються методи нелінійного програмування. Це можуть бути методи градієнтного спуску, методи Лагранжа тощо.
3) Динамічне програмування. Метод розбиває задачу на підзадачі, які вирішуються рекурсивно. Використовується для задач оптимізації, які можуть бути розділені на взаємозалежні етапи.

З методів штучного інтелекту можемо згадати:

1) Машинне навчання. Алгоритми машинного навчання, такі як регресія, дерева рішень та нейронні мережі, можуть використовуватися для прогнозування та оптимізації багатопараметричних задач.
2) Глибинне навчання. Глибинні нейронні мережі ефективні для складних задач з великою кількістю параметрів, де традиційні методи можуть не впоратися.
3) Підкріплювальне навчання. Цей метод використовується для навчання агентів, які приймають рішення в динамічних середовищах з метою максимізації винагороди.

Остання група методів, евристичні методи, являє собою підходи до вирішення складних задач, що використовують правила, припущення або інтуїцію для знаходження прийнятних рішень у розумний час. Вони не гарантують

знаходження оптимального рішення, але часто дозволяють отримати хороші результати швидше та ефективніше, ніж традиційні методи. Евристики зазвичай застосовуються у випадках, коли пошук точного рішення є занадто складним або обчислювально затратним.

До популярних евристичних методів належать табу пошук, симуляційне моделювання, алгоритм рою частинок та генетичний алгоритм. Останній розглянемо у цій роботі більш детально.

Генетичний алгоритм (ГА) є методом оптимізації, натхненним принципами природного відбору та еволюції, які використовуються для знаходження рішень складних задач [10]. Ідея алгоритму полягає в еволюції популяції можливих рішень задачі з використанням операцій, аналогічних біологічним процесам, таким як селекція, кросовер (схрещування) і мутація. Цей алгоритм є частиною ширшої галузі, відомої як еволюційна обчислювальна техніка.

Основна мета генетичного алгоритму — знайти наближене або точне рішення для оптимізаційних задач шляхом багаторазового поліпшення початкового набору можливих рішень, які називаються індивідами. Кожен індивід характеризується набором параметрів, відомих як гени, які утворюють хромосому.

У контексті калібрування маніпулятора, генетичний алгоритм може бути використаний для пошуку формул розрахунку кутів для точного позиціонування функціональної кінцівки[28].

Переваги використання генетичного алгоритму для калібрування маніпулятора:

1) Автоматизація процесу: Генетичні алгоритми (ГА) дозволяють автоматизувати процес створення моделей корекції без потреби в людському втручанні. Це знижує ризик людських помилок і підвищує ефективність калібрування.
2) Символьна регресія: Використання символьної регресії замість числової дозволяє отримати не лише компенсацію похибок, а й символьний опис складних ефектів, пов'язаних зі з'єднаннями, що дозволяє подальший математичний аналіз.

3) Глобальна оптимізація: Завдяки своїй стохастичній природі, генетичні алгоритми мають потенціал знаходити глобально оптимальні моделі калібрування, що покращує точність маніпулятора.
4) Паралельна обробка: Можливість паралельної еволюції моделей корекції для кожного з'єднання робить процес більш ефективним, особливо при використанні розподілених обчислювальних ресурсів.
5) Гнучкість: Генетичні алгоритми не обмежуються фіксованими структурами моделей, що надає їм гнучкість у вирішенні різноманітних завдань калібрування.

Недоліки використання генетичного алгоритму для калібрування маніпулятора:

1) Високі вимоги до обчислювальних ресурсів: Генетичні алгоритми можуть бути обчислювально інтенсивними, особливо на пізніх етапах еволюції, коли швидкість збіжності зменшується. Це може обмежувати їхню застосовність у реальному часі.
2) Непередбачуваність процесу еволюції: Через стохастичну природу ГА, їхній прогрес може бути непередбачуваним. Це може призвести до варіацій у результатах різних запусків алгоритму, навіть при однакових параметрах налаштування.
3) Необхідність налаштування параметрів: Для досягнення оптимальних результатів ГА потребують ретельного налаштування параметрів, таких як розмір популяції та кількість поколінь, що може бути складним та часозатратним процесом.
4) Проблеми з надмірністю коду: На пізніх етапах еволюції може виникнути проблема надмірності коду (code bloat), що ускладнює застосування еволюційних операторів та знижує ефективність алгоритму.
5) Обмежена здатність до числової ідентифікації параметрів: На відміну від традиційних методів калібрування, генетичні алгоритми не завжди

можуть ефективно визначати числові параметри через високу нелінійність та дискретність еволюційних рівнянь.

Ці переваги та недоліки підкреслюють комплексний характер використання генетичних алгоритмів для калібрування маніпуляторів, що дозволяє значно підвищити точність, але вимагає високих обчислювальних ресурсів і ретельного налаштування.

2.2 Використання генетичних алгоритмів у різних галузях

Генетичні алгоритми (ГА) є потужним інструментом для вирішення широкого спектру задач в різних галузях. Їхня гнучкість і здатність знаходити оптимальні або наближені до оптимальних рішення роблять їх корисними в багатьох сферах.

Однією з основних галузей застосування генетичних алгоритмів є інженерія. Вони використовуються для оптимізації конструкцій, наприклад, в аерокосмічній інженерії для проєктування крил літаків, де необхідно враховувати багато параметрів, таких як аеродинамічні характеристики, вага та міцність матеріалів. ГА дозволяють знаходити оптимальні конфігурації, які забезпечують найкращі експлуатаційні характеристики при мінімальних витратах.

В галузі економіки та фінансів генетичні алгоритми застосовуються для оптимізації портфелів інвестицій, прогнозування фінансових ринків і розробки торгових стратегій. Вони дозволяють знаходити оптимальні комбінації активів, які максимізують дохід і мінімізують ризик. Крім того, ГА використовуються для автоматизації торгових систем, які можуть адаптуватися до змін ринкових умов в режимі реального часу.

У біоінформатиці генетичні алгоритми допомагають у вирішенні задач пов'язаних з аналізом генетичних даних, моделюванням еволюційних процесів і розробкою нових лікарських засобів. Наприклад, вони використовуються для вирішення задач секвенування генома, що дозволяє дослідникам знаходити нові гени і розуміти їх функції. Також ГА допомагають в оптимізації структури білків для створення ефективних лікарських препаратів.

У галузі штучного інтелекту генетичні алгоритми використовуються для навчання нейронних мереж і вдосконалення алгоритмів машинного навчання. Вони допомагають в автоматичній оптимізації гіперпараметрів моделей, що значно покращує їх продуктивність. ГА також застосовуються для розробки алгоритмів адаптивного навчання, які можуть змінювати свої стратегії на основі нових даних і змін у навколишньому середовищі [15].

В екології та сільському господарстві генетичні алгоритми допомагають в моделюванні екосистем і оптимізації агротехнічних процесів. Вони використовуються для прогнозування поведінки популяцій, оптимізації використання ресурсів і розробки стійких методів ведення сільського господарства. ГА допомагають знайти оптимальні стратегії для збереження біорізноманіття і ефективного використання природних ресурсів.

У сфері робототехніки генетичні алгоритми використовуються для навчання роботів виконувати складні завдання. Це може включати оптимізацію траєкторій руху, налаштування параметрів контролерів, а також навчання роботів виконувати завдання в динамічних і непередбачуваних середовищах. Використовуючи ГА, розробники можуть створювати роботів, здатних адаптуватися до нових умов і самостійно вдосконалювати свої навички.

У сфері телекомунікацій генетичні алгоритми застосовуються для оптимізації роботи мереж і розподілу ресурсів. Вони дозволяють покращити якість зв'язку, мінімізувати затримки та забезпечити ефективне використання мережевих ресурсів. ГА можуть допомагати в управлінні трафіком, плануванні частотних ресурсів та оптимізації розташування базових станцій[3].

Більше галузей застосування генетичного алгоритму зображено на рис. 2.1.

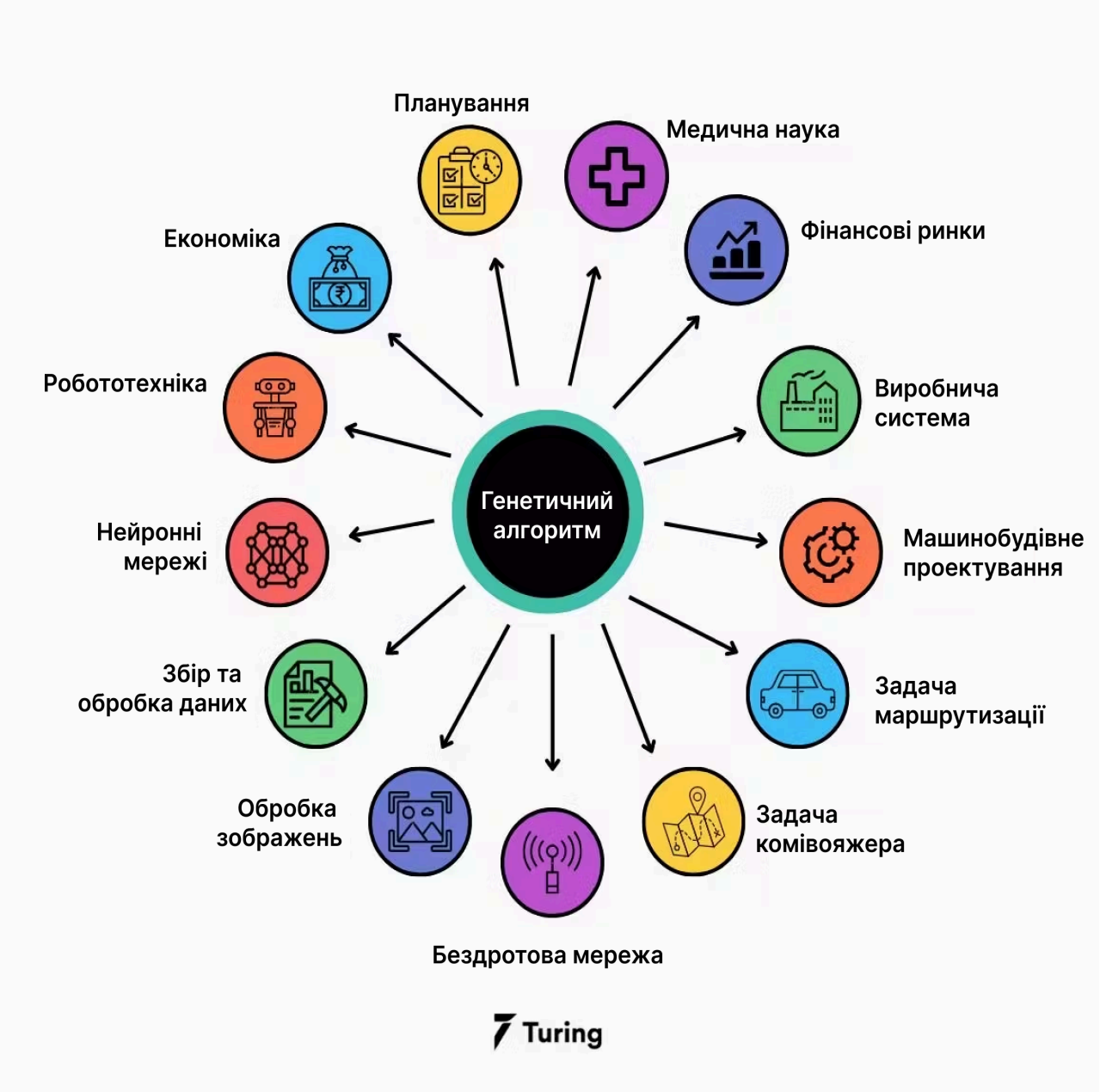


Рис. 2.1 Галузі застосування генетичного алгоритму

Отже, генетичні алгоритми є універсальними інструментами, які знаходять застосування в багатьох галузях науки і техніки. Їх здатність адаптуватися до складних і змінних умов робить їх незамінними для вирішення різноманітних задач оптимізації, пошуку і навчання. Завдяки постійному розвитку обчислювальних технологій, застосування генетичних алгоритмів продовжує розширюватися, відкриваючи нові можливості для інновацій та наукових досліджень.

2.3 Етапи генетичного алгоритму

Генетичний алгоритм складається з кількох ключових етапів, кожен з яких відіграє важливу роль у процесі еволюції популяції індивідів та досягненні оптимального рішення. Основні етапи генетичного алгоритму включають ініціалізацію, відбір, кросовер, мутацію, оцінку пристосованості та заміну поколінь. Кожен з цих етапів має свої особливості та важливість у контексті еволюційного процесу [14].

Основна структура генетичного алгоритму включає кілька ключових кроків, які можна узагальнити наступним чином:

1. Ініціалізація: Процес починається зі створення початкової популяції P індивідів. Ця популяція може бути згенерована випадковим чином або на основі попередніх знань про простір рішень.
2. Оцінка пристосованості: Кожен індивід у популяції оцінюється за рівнем пристосованості, що є мірою того, наскільки добре він виконує завдання в заданій проблемній області.
3. Селекція: На основі рівня пристосованості відбирається підмножина індивідів, які слугуватимуть батьками для наступного покоління.
4. Кросовер: Відібрані батьківські індивіди піддаються рекомбінації, де сегменти їхніх генотипів обмінюються для створення нових потомків. Ця операція спрямована на об'єднання корисних рис від різних батьків, що потенційно призводить до кращих потомків.
5. Мутація: Для підтримання генетичної різноманітності до деяких індивідів застосовується мутація. Це передбачає випадкові зміни їхніх генотипів, вводячи нові генетичні варіації, які можуть покращити пристосованість або допомогти досліджувати нові області простору рішень.
6. Заміна: Потомство, створене шляхом рекомбінації та мутації, замінює стару популяцію, створюючи нове покоління. Цикл оцінки, відбору,

рекомбінації та мутації триває до досягнення умови завершення, наприклад, досягнення задовільного рішення або завершення визначеної кількості поколінь.

7. Оцінка пристосованості: Пристосованість індивіда визначається його продуктивністю в проблемній області. У багатьох випадках це передбачає перетворення генотипу у фенотипічне представлення, а потім використання цільової функції для вимірювання його ефективності. Отримане значення пристосованості може бути скалярним або векторним, залежно від того, чи має задача один або кілька об'єктів.

Алгоритм продовжується поки не буде досягнуто критеріїв зупинки, наприклад, коли знайдено розв'язок з достатньо високою придатністю або після певної кількості поколінь.

Увесь процес більш наочно можна представити у вигляді схеми (рис. 2.2).

Рис. 2.2 Схема етапів генетичного алгоритму

Існують різні стратегії для відбору індивідів на основі їхньої пристосованості. Поширені методи включають пропорційний відбір, наприклад, рулетковий відбір, де індивіди обираються ймовірнісно на основі їхньої відносної

пристосованості, та ранговий відбір, де індивіди ранжуються і відбираються на основі їхнього порядку.

Існують три основні методи для створення нових індивідів:

1) Кросовер: Передбачає об'єднання частин генотипів двох або більше батьків (рис. 2.3 та рис. 2.4)

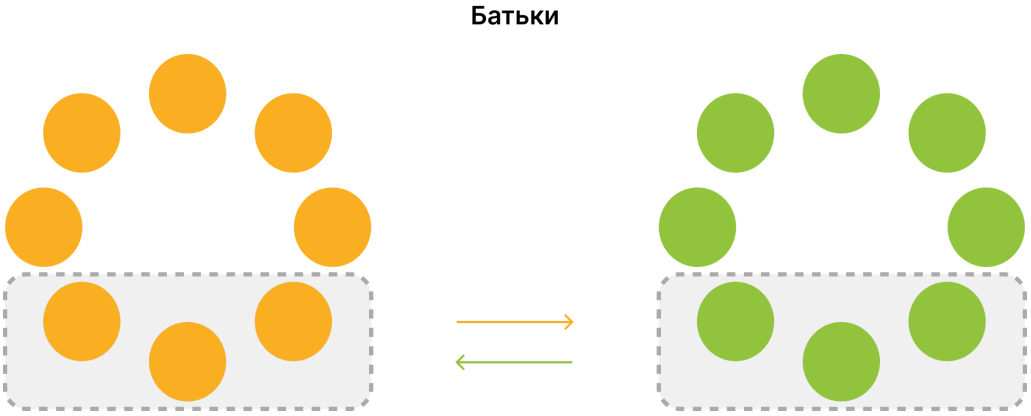


Рис. 2.3 Кросовер двох хромосом – батьки

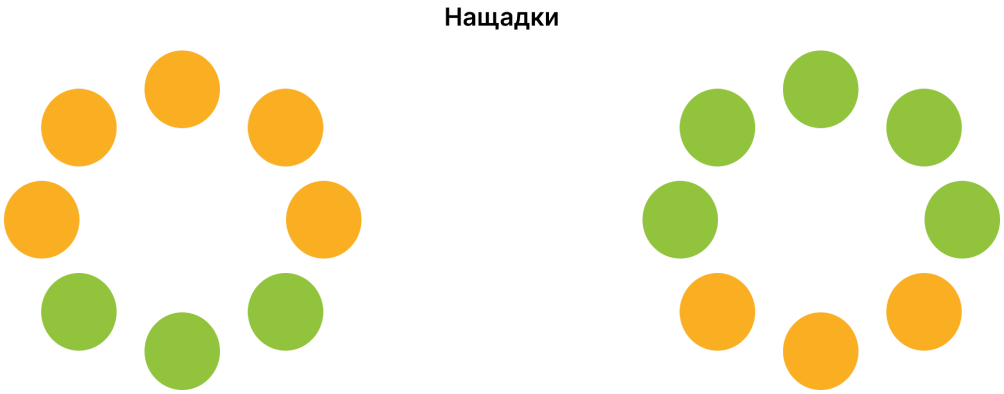


Рис. 2.4 Кросовер двох хромосом - нащадки

2) Мутація: Передбачає випадкові зміни генотипу індивіда (рис. 2.5).

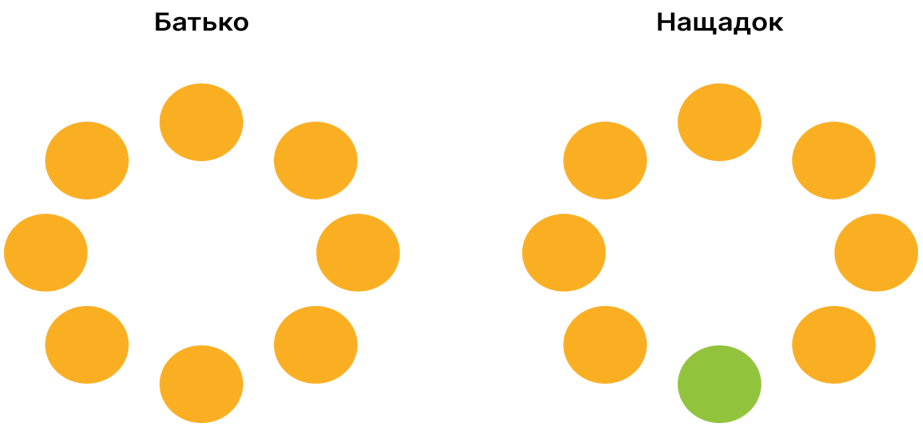


Рис. 2.5 Мутація хромосоми

3) Репродукція: Просто передбачає копіювання індивідів із поточної популяції в наступну без жодних змін, зберігаючи індивідів з високою пристосованістю.

Щоб запобігти втраті високоякісних рішень, часто використовується елітний відбір. Ця техніка забезпечує перенесення найкращих індивідів у наступне покоління без змін, підтримуючи високий стандарт продуктивності в популяції.

2.4 Параметри генетичного алгоритму

Параметри генетичного алгоритму (ГА) є ключовими для його ефективного функціонування та досягнення оптимальних результатів. Правильний вибір і налаштування цих параметрів впливають на продуктивність алгоритму, швидкість конвергенції та якість знайдених рішень. У цьому розділі розглядаються основні параметри генетичного алгоритму, їх роль та вплив на еволюційний процес.

Розмір популяції визначає кількість індивідів, що існують в кожному поколінні. Від розміру популяції залежить обсяг генетичного матеріалу, доступного для створення нових рішень, а також обчислювальні витрати. Велика популяція забезпечує більшу генетичну різноманітність, що може покращити здатність алгоритму знаходити глобальні оптимуми, але при цьому збільшує обчислювальні витрати. Занадто мала популяція може призвести до швидкої конвергенції до локальних оптимумів [16].

Кількість поколінь визначає, скільки разів алгоритм буде повторювати процес селекції, кросовера та мутації. Від кількості поколінь залежить, наскільки близько ГА може наблизитися до оптимального рішення. Якщо кількість поколінь занадто мала, алгоритм може не встигнути знайти оптимальне рішення, тоді як занадто велика кількість поколінь може призвести до надмірних обчислювальних витрат.

Ймовірність кросовера визначає, як часто генетичний матеріал буде обмінюватися між парами індивідів. Висока ймовірність кросовера сприяє більшому різноманіттю в популяції, що може покращити здатність алгоритму до глобального пошуку. Проте занадто висока ймовірність кросовера може призвести до втрати хороших рішень, вже знайдених у попередніх поколіннях.

Ймовірність мутації визначає, як часто випадкові зміни будуть вноситися в індивіди. Мутація допомагає запобігти передчасній конвергенції, додаючи нові варіації в популяцію. Низька ймовірність мутації може призвести до втрати різноманітності, а надто висока — до випадкового пошуку, що може знижувати ефективність алгоритму.

Розмір турніру використовується у турнірному відборі для визначення, скільки індивідів беруть участь у кожному турнірі. Більший розмір турніру збільшує селективний тиск, сприяючи вибору кращих індивідів. Проте занадто великий розмір турніру може призвести до швидкої конвергенції та втрати генетичної різноманітності.

$$p_t = \frac{1}{T}$$

де T — розмір турніру.

Максимальна глибина дерева визначає складність генетичних програм. Глибші дерева можуть представляти більш складні рішення, але вони також можуть призводити до перевантаження моделі та збільшення обчислювальних витрат. Занадто великі дерева можуть стати неефективними через ефект "засмічення" (bloat).

Еліта – це кількість найкращих індивідів, які без змін переносяться в наступне покоління. Це допомагає зберегти найкращі знайдені рішення та забезпечує стабільність еволюційного процесу. Занадто великий розмір еліти може знизити генетичне різноманіття в популяції.

Тип кросовера визначає метод, за допомогою якого відбувається обмін генетичним матеріалом між індивідами. Звичайний одноточковий або двоточковий кросовер є найпоширенішими методами. Вибір типу кросовера впливає на генетичне різноманіття та ефективність пошуку.

Одноточковий кросовер:

$$C1 = (P1[1:k], P2[k+1:end])C1 = (P1[1:k], P2[k+1:end])$$
$$C2 = (P2[1:k], P1[k+1:end])C2 = (P2[1:k], P1[k+1:end]),$$

де $C1$ і $C2$— потомки, $P1$ і $P2$ — батьки, k — випадкова точка кросовера.

Тип мутації визначає спосіб внесення випадкових змін у генетичний матеріал індивідів. Це можуть бути такі методи, як заміна вузла або зміна значення. Вибір типу мутації впливає на здатність алгоритму підтримувати генетичне різноманіття та уникати передчасної конвергенції.

Критерії зупинки визначають, коли генетичний алгоритм повинен зупинитися. Це може бути досягнення певного рівня пристосованості, досягнення максимальної кількості поколінь або інші умови. Правильний вибір критеріїв зупинки впливає на ефективність алгоритму.

2.5 Реалізація генетичного алгоритму

Реалізація генетичного алгоритму включає кілька основних етапів, що забезпечують ефективну еволюцію популяції індивідів і знаходження оптимальних рішень. У цьому розділі детально розглядаються основні етапи реалізації генетичного алгоритму, зокрема визначення генів та побудова хромосом, формування дерев, оцінка пристосованості, визначення точки схрещування та критерію схрещування, а також відбір наступного покоління.

2.5.1 Визначення генів та побудова хромосом

Гени є основними одиницями інформації у генетичному алгоритмі. Вони представляють мінімальні елементи, які можна комбінувати для створення більш складних рішень. Гени можуть бути визначені як окремі змінні, оператори або константи, які використовуються в побудові хромосом. У контексті генетичного алгоритму, гени можуть включати арифметичні оператори (наприклад, +, -, *, /),

логічні оператори (AND, OR, NOT), умовні оператори (if, else) та функції (sin, cos, exp, log). В нашому ж випадку у вигляді змінних будуть використовуватись кути θ.

Хромосоми в даному алгоритмі представляють собою структури даних, що складаються з генів. В роботі вони представлені у вигляді дерев, де кожен вузол дерева є геном. Дерева мають кореневий вузол, внутрішні вузли (нетермінали) та листові вузли (термінали). Термінальні вузли не мають дочірніх вузлів і часто є змінними або константами. Внутрішні вузли містять оператори, які визначають операції над дочірніми вузлами [20].

На прикладі еволюційних обчислень, генетичні алгоритми використовують хромосоми для представлення можливих рішень задачі. Кожна хромосома складається з набору генів, які визначають різні параметри або атрибути рішення.

Загальна схема популяції та її компонентів зображено на рис. 2.6.

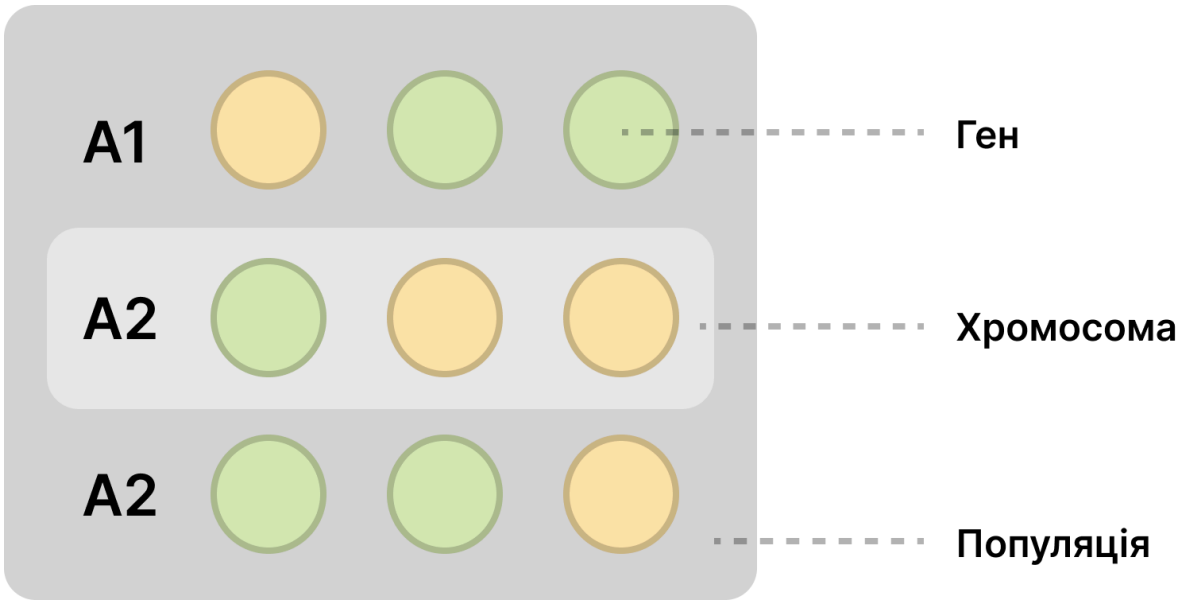


Рис. 2.6. Популяція генетичного алгоритму

2.5.2 Формування дерев

Побудова хромосом у вигляді дерев є поширеним методом у генетичному програмуванні. Дерева використовуються для представлення складних рішень, де кожен вузол відповідає певній операції або функції, а листові вузли - аргументам

або константам. Наприклад, дерево може представляти математичний вираз, де кореневий вузол є оператором, а його нащадки - операндами. Формула для побудови дерева може бути записана як:

Дерева дозволяють ефективно застосовувати операції кросовера та мутації, оскільки піддерева можуть бути легко замінені або змінені для створення нових варіантів рішень. Це забезпечує високу гнучкість та потужність генетичних алгоритмів у знаходженні оптимальних рішень для складних задач [27].

Початкова популяція дерев генерується випадковим чином або на основі певних правил. Існують різні методи ініціалізації, такі як метод "повного" дерева (FULL) та метод "зростання" (GROW).

Метод FULL: Всі вузли дерева на кожному рівні, крім листових вузлів, є нетерміналами. Цей метод забезпечує створення повного дерева певної глибини.

Метод GROW: Вузли на кожному рівні можуть бути як терміналами, так і нетерміналами. Цей метод дозволяє створювати дерева різної форми та глибини.

У прикладі, зображеному на рис. 2.4 кореневим вузлом є оператор “+”, який має два дочірні вузли: оператор “*”, який є внутрішнім вузлом, та константу 3, як листьовий вузол даного дерева. Вузол “*” також має два дочірні вузли: змінну “x” та константу “2” [8] [26].

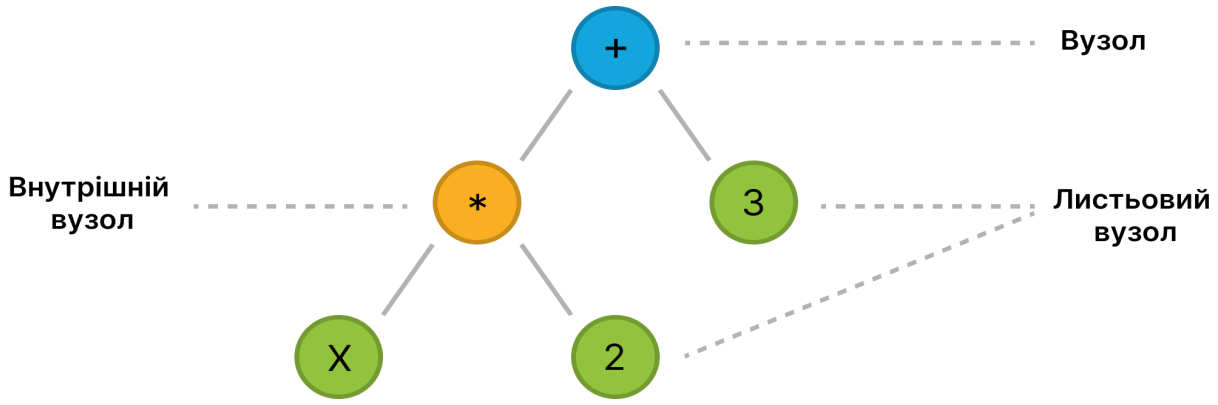


Рис. 2.7. Приклад побудови дерева

Таке дерево буде мати наступну математичну форму:

$$f(x) = 2x + 3$$

Алгоритм побудови хромосом можна представити у вигляді наступної архітектури:

```
Вхідні дані: Популяція індивідів P

1. Для кожного індивіда x в популяції P:
   a. Перетворити генотип x у фенотип y
   b. Обчислити значення функції помилки E(y)
   c. Обчислити значення
2. За необхідності нормалізувати значення пристосованості
3. Повернути значення пристосованості для всіх індивідів

Вихідні дані: Значення пристосованості для кожного індивіда в P
```

2.5.2 Оцінка пристосованості

Оцінка пристосованості дозволяє визначити якість кожного індивіда в популяції, що в свою чергу впливає на процес відбору, схрещування та мутації. Функція пристосованості (фітнес-функція) оцінює, наскільки добре кожен індивід вирішує поставлену задачу або задовольняє визначені критерії. У контексті задачі регресії, наша фітнес-функція вимірює різницю між прогнозованими та фактичними значеннями. Вибір правильної фітнес-функції є критично важливим для успішного виконання генетичного алгоритму, оскільки вона визначає, які індивіди будуть обрані для створення нового покоління.

Фітнес-функція в задачах регресії зазвичай базується на різних метриках помилок, таких як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE) або середня абсолютна відсоткова помилка (MAPE). Ці метрики використовуються для вимірювання відхилення прогнозів моделі від фактичних значень.

Наприклад, середньоквадратична помилка обчислюється як середнє значення квадратів різниць між прогнозованими та фактичними значеннями:

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

де yi - фактичне значення, $y\hat{}i$ - прогнозоване значення, n - кількість спостережень.

Середня абсолютна помилка обчислюється як середнє значення абсолютних різниць між прогнозованими та фактичними значеннями:

$$MAE=1n\sum i=1n|yi-y\hat{}i|$$

Середня абсолютна відсоткова помилка обчислюється як середнє значення абсолютних відсоткових різниць між прогнозованими та фактичними значеннями:

$$MAPE=100\%n\sum i=1n|yi-y\hat{}iyi|$$

Ці метрики допомагають визначити, наскільки добре модель передбачає результати. Чим менша ця помилка, тим кращою вважається модель. Фітнес-функція повинна бути такою, щоб індивіди з меншою помилкою отримували вищу оцінку, що сприятиме їх відбору для подальшого схрещування та мутації.

В нашому конкретному випадку задачі регресії, фітнес-функція обчислює середньоквадратичну помилку між прогнозованими значеннями, отриманими від моделі, і фактичними цільовими значеннями. Кожен індивід представляє певний математичний вираз, який використовується для прогнозування значень на основі вхідних даних. Мета полягає в тому, щоб знайти вираз, який мінімізує цю помилку. Таким чином, індивіди з меншою помилкою отримують вищу оцінку пристосованості і мають більше шансів бути обраними для створення нового покоління.

2.5.3 Визначення точки схрещування та критерію схрещування

У випадку деревоподібних структур, кожне дерево представляє собою можливе рішення задачі, де вузли дерева відповідають певним операціям або функціям, а листові вузли — аргументам або константам. Точка схрещування в таких структурах визначається вибором вузла в дереві, де буде виконано розрізання і обмін піддеревами між двома батьківськими деревами [6].

Вибір точки схрещування може бути виконаний випадковим чином або на основі певних критеріїв, таких як максимізація різноманітності потомків або мінімізація ймовірності розриву корисних підструктур (building blocks). Наприклад, можна використовувати ймовірнісний підхід, де кожен вузол дерева має певну ймовірність бути вибраним як точка схрещування.

Ймовірність кросовера визначає, як часто генетичний матеріал буде обмінюватися між парами індивідів. Висока ймовірність кросовера сприяє більшому різноманіттю в популяції, що може покращити здатність алгоритму до глобального пошуку. Однак занадто висока ймовірність може призвести до втрати хороших рішень, вже знайдених у попередніх поколіннях. Типовий діапазон ймовірності кросовера становить від 0.6 до 0.9.

На рис. 2.8 зображено кросовер між двома хромосомами представленими у вигляді дерев.

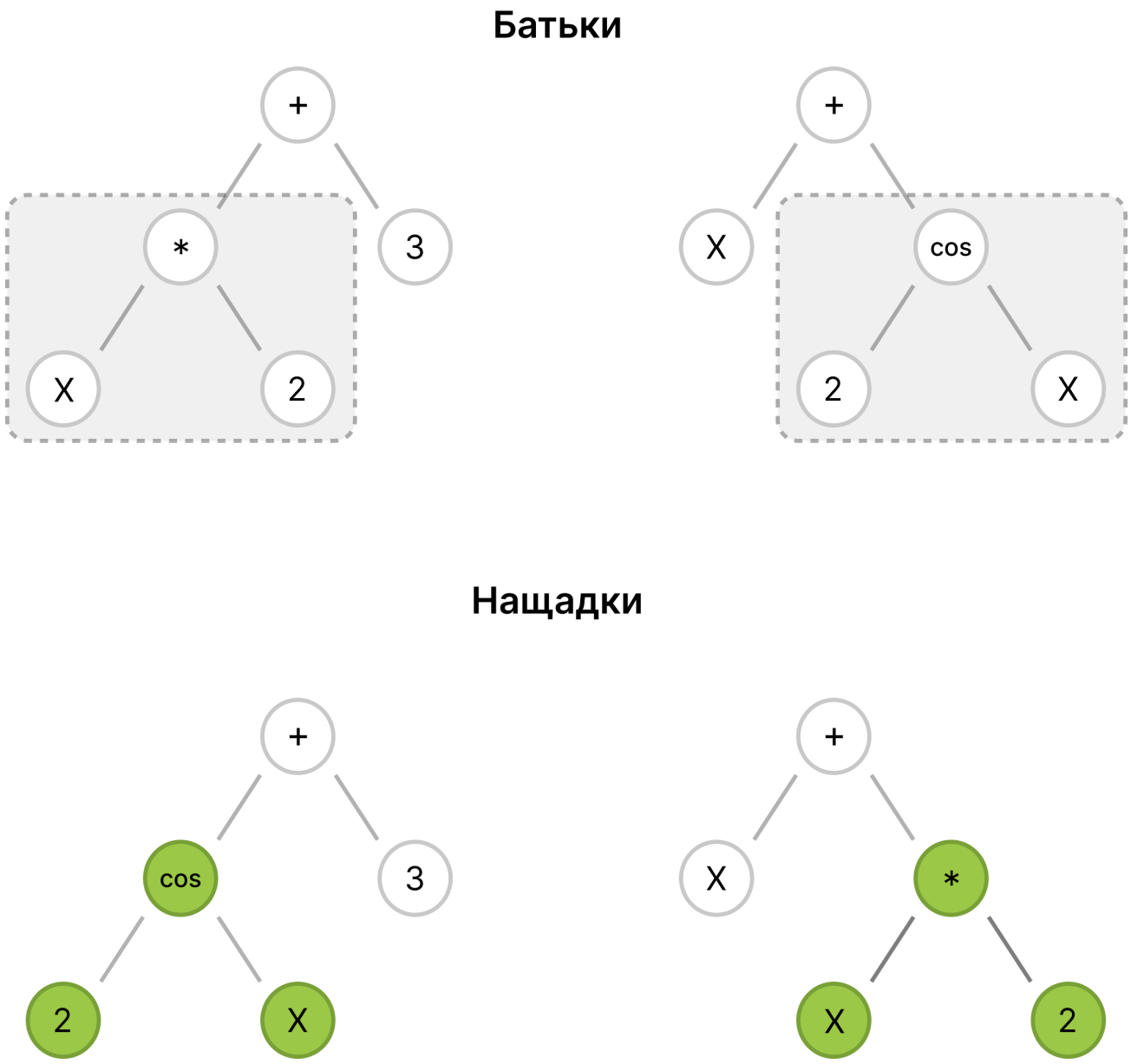


Рис. 2.8 Кросовер хромосом, представлених у вигляді дерев

2.5.4 Відбір наступного покоління

Відбір наступного покоління в контексті генетичного програмування (GP) є критичним етапом, який значно впливає на загальну ефективність алгоритму. Відбір забезпечує, що найкращі індивіди поточного покоління мають шанс передати свої гени наступному поколінню, що сприяє поступовому покращенню рішень.

Початково, кожен індивід в популяції оцінюється за допомогою функції пристосованості (фітнес-функції), яка визначає, наскільки добре він вирішує поставлену задачу. На основі значень пристосованості індивіди ранжуються, і на цьому етапі застосовуються різні методи відбору для визначення, які індивіди будуть використані для створення нового покоління.

Одним з найпоширеніших методів відбору є турнірний відбір. В цьому методі випадковим чином обираються кілька індивідів з популяції (розмір турніру зазвичай визначається параметром налаштувань), і найкращий з них (тобто той, що має найвищу пристосованість) обирається для подальшого схрещування. Турнірний відбір має декілька переваг: він простий у реалізації, дозволяє контролювати селективний тиск (змінюючи розмір турніру) і добре працює в умовах високої різноманітності популяції.

Іншим популярним методом є відбір за рулетковим принципом (пропорційний відбір), де ймовірність вибору індивіда пропорційна його пристосованості. Це дозволяє більш пристосованим індивідам мати більші шанси на відбір, але також зберігає ймовірність вибору менш пристосованих індивідів, що сприяє збереженню генетичної різноманітності.

Після відбору пар батьків відбувається процес схрещування (кросовера), де частини генетичного матеріалу батьків комбінуються для створення нових індивідів (нащадків). Одноточковий кросовер є одним з найпоширеніших методів, при якому хромосоми батьків розрізаються в одному випадковому місці, і частини після цієї точки обмінюються між собою.

Мутація є наступним етапом, який додає випадкові зміни до генів нових індивідів. Мутація важлива для збереження генетичної різноманітності популяції і допомагає уникнути передчасної конвергенції до локальних мінімумів. У генетичному програмуванні мутація може включати заміну одного з вузлів дерева на інший випадковий вузол.

Після схрещування і мутації нова популяція індивідів оцінюється знову за допомогою фітнес-функції. Найкращі індивіди з нової популяції обираються для

подальшого відбору, і процес повторюється до досягнення заданої кількості поколінь або досягнення критерію зупинки.

2.5.5 Аналіз результатів та продуктивність генетичних алгоритмів

Аналіз результатів і продуктивності генетичного алгоритму базується на оцінці точності обчислень, що здійснюються для визначення координат кінцевого ефектора маніпулятора. Після виконання обчислень за допомогою генетичного алгоритму, отримані результати порівнюються з виміряними координатами, і розраховується помилка для кожної координати.

Припустимо, що виміряні координати кінцевого ефектора становлять:

$$P_{meas} = \left(x_{meas}, y_{meas}, z_{meas}\right)$$

Обчислені координати будуть:

$$P_{calc} = \left(x_{calc}, y_{calc}, z_{calc}\right)$$

Для оцінки точності обчислень використовуються помилки в кожній координаті. Помилки розраховуються як різниця між виміряними та обчисленими значеннями координат:

Розрахунок помилок у кожній координаті:

$$e_x = x_{meas} - x_{calc},$$
$$e_y = y_{meas} - y_{calc},$$
$$e_z = z_{meas} - z_{calc}$$

Ці помилки дозволяють оцінити, наскільки обчислені значення відрізняються від виміряних.

Сумарна помилка обчислюється за допомогою формули, що враховує всі три координати:

$$e_{total} = \sqrt{e_x^2 + e_y^2 + e_z^2}$$

Цей показник дозволяє оцінити загальну точність обчислень.

Процес аналізу включає наступні етапи:

- Обчислення координат: Генетичний алгоритм генерує координати кінцевого ефектора, які потім використовуються для порівняння з виміряними значеннями.
- Розрахунок помилок: Різниця між виміряними та обчисленими координатами дозволяє розрахувати помилки для кожної координати окремо, а також сумарну помилку.
- Аналіз точності: Оцінка точності здійснюється на основі сумарної помилки, що дозволяє визначити, наскільки ефективно генетичний алгоритм виконує свої обчислення.
- Візуалізація результатів: Результати роботи генетичного алгоритму можуть бути візуалізовані у вигляді графіків або діаграм, що відображають точність обчислень та дозволяють краще зрозуміти продуктивність алгоритму.

Метод оцінки точності, описаний вище, дозволяє проводити детальний аналіз результатів, виявляти можливі проблеми та покращувати продуктивність генетичного алгоритму для задач регресії.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір стеку технологій

Для вирішення задачі регресії та оптимізації рухів робота-маніпулятора було обрано певний стек технологій. Вибір технологій базувався на їх можливостях, зручності використання та відповідності вимогам проекту. Основні інструменти і бібліотеки, використані в роботі:

- Python: Основна мова програмування, обрана для цієї роботи. Python є широко використовуваною мовою завдяки своїй простоті, читабельності та великій кількості доступних бібліотек для наукових розрахунків, обробки даних та машинного навчання.

- DEAP (Distributed Evolutionary Algorithms in Python): Ця бібліотека була обрана для реалізації генетичного програмування та символічної регресії. DEAP пропонує гнучкий і зручний інструментарій для побудови еволюційних алгоритмів, що дозволяє швидко і ефективно реалізувати складні обчислювальні задачі. Вона підтримує різні типи еволюційних алгоритмів, включаючи генетичне програмування, що було ключовою вимогою для нашої задачі.

- SciPy: Використовується для оптимізації та розв'язання задач нелінійної регресії. Зокрема, функції scipy.optimize.least_squares та scipy.optimize.minimize дозволяють ефективно знаходити оптимальні параметри для математичних моделей, що описують рухи робота. SciPy є стандартною бібліотекою для наукових розрахунків в Python і надає широкий спектр інструментів для чисельного аналізу.

- NumPy: Основна бібліотека для роботи з масивами даних і математичними операціями в Python. NumPy забезпечує ефективне зберігання і обробку великих масивів числових даних, що є критичним для задач регресії та обробки даних з сенсорів робота. Вона також тісно інтегрується з іншими бібліотеками, такими як SciPy та DEAP, що спрощує їх сумісне використання.

- IKPy: Бібліотека для обчислення прямої та зворотної кінематики роботів. IKPy надає зручні інструменти для моделювання та управління роботами-маніпуляторами, що дозволяє легко інтегрувати кінематичні розрахунки у загальний алгоритм управління роботом. Вона підтримує різні методи обчислення зворотної кінематики, включаючи методи оптимізації, що було використано в нашій задачі.

- Matplotlib: Використовується для візуалізації результатів та відображення графіків. Matplotlib дозволяє створювати високоякісні графіки і діаграми, що допомагає візуалізувати процес навчання моделі, оцінку помилок та результати роботи алгоритму.

- IPyWidgets: Бібліотека для створення інтерактивних віджетів в Jupyter Notebook. IPyWidgets була використана для створення зручного інтерфейсу користувача, що дозволяє налаштовувати параметри генетичного алгоритму та запускати обчислення безпосередньо з ноутбука. Це полегшує експериментування з різними параметрами та аналіз результатів.

- Jupyter Notebook: Інтерактивне середовище для розробки та виконання коду. Jupyter Notebook забезпечує зручний інтерфейс для написання, виконання і документування коду, що робить його ідеальним інструментом для наукових досліджень та аналізу даних.

- Google Colab: Платформа для реалізації проекту, яка надає безкоштовні ресурси для виконання коду в хмарі. Google Colab дозволяє працювати з Jupyter Notebook без необхідності налаштовувати локальне середовище, що значно спрощує процес розробки та тестування моделей. Colab забезпечує доступ до GPU та TPU, що дозволяє значно прискорити обчислення та навчання моделей.

Вибір цього стеку технологій був зумовлений необхідністю вирішувати складні задачі регресії та оптимізації рухів робота-маніпулятора, забезпечуючи при цьому високу продуктивність, зручність розробки та можливість інтерактивної взаємодії з моделлю [17]. Кожна з обраних технологій відіграє важливу роль у досягненні цих цілей, дозволяючи ефективно обробляти дані, будувати і оптимізувати моделі, а також візуалізувати результати та налаштовувати

параметри алгоритмів. Google Colab забезпечує необхідну обчислювальну потужність і зручність роботи, роблячи процес розробки більш ефективним та доступним.

3.2 Згальна архітектура та структура програми

Програма складається з трьох основних модулів: графічного, інтерфейсного та розрахункового. Кожен з них виконує свої специфічні функції та взаємодіє з іншими модулями для забезпечення комплексного вирішення задачі.

Графічний модуль відповідає за візуалізацію даних та результатів. Він використовує бібліотеки для побудови графіків, таких як Matplotlib, для відображення координат, кутів суглобів та інших важливих параметрів маніпулятора. Цей модуль також забезпечує візуалізацію процесу роботи алгоритмів, таких як генетичне програмування та ко-еволюція, що дозволяє користувачеві отримати наочне уявлення про ефективність та прогрес роботи алгоритмів.

Інтерфейсний модуль реалізує взаємодію користувача з програмою. Він використовує бібліотеки для створення інтерактивних віджетів, такі як IPyWidgets, що дозволяє налаштовувати параметри роботи алгоритмів безпосередньо з інтерфейсу. Користувач може встановлювати значення параметрів, запускати алгоритми та переглядати результати роботи у зручному форматі. Цей модуль забезпечує інтуїтивно зрозумілий та зручний спосіб управління програмою, що значно полегшує процес роботи з нею.

Розрахунковий модуль виконує основну обчислювальну роботу. Він включає в себе функції для ініціалізації та налаштування моделей, реалізації генетичних алгоритмів та методів машинного навчання. Цей модуль також відповідає за обчислення прямих та зворотних кінематичних задач для маніпулятора, використовуючи бібліотеку IKPy. Основні функції, такі як оцінка придатності моделей, символічна регресія та ко-еволюційна корекція, реалізовані в цьому модулі. Він забезпечує виконання всіх розрахунків, необхідних для корекції рухів

маніпулятора, та інтеграцію з іншими модулями для передачі результатів та вхідних даних.

Цілісна архітектура програми побудована таким чином, щоб кожен модуль виконував свої специфічні функції, забезпечуючи при цьому злагоджену роботу всієї системи. Графічний модуль отримує дані від розрахункового модуля та відображає їх у зручному форматі, інтерфейсний модуль дозволяє користувачеві налаштовувати параметри та взаємодіяти з програмою, а розрахунковий модуль виконує всю основну обчислювальну роботу. Така структура забезпечує гнучкість, масштабованість та зручність використання програми для вирішення задачі регресії та оптимізації рухів робота-маніпулятора.

Схему такої структури зображено на рис. 3.1.

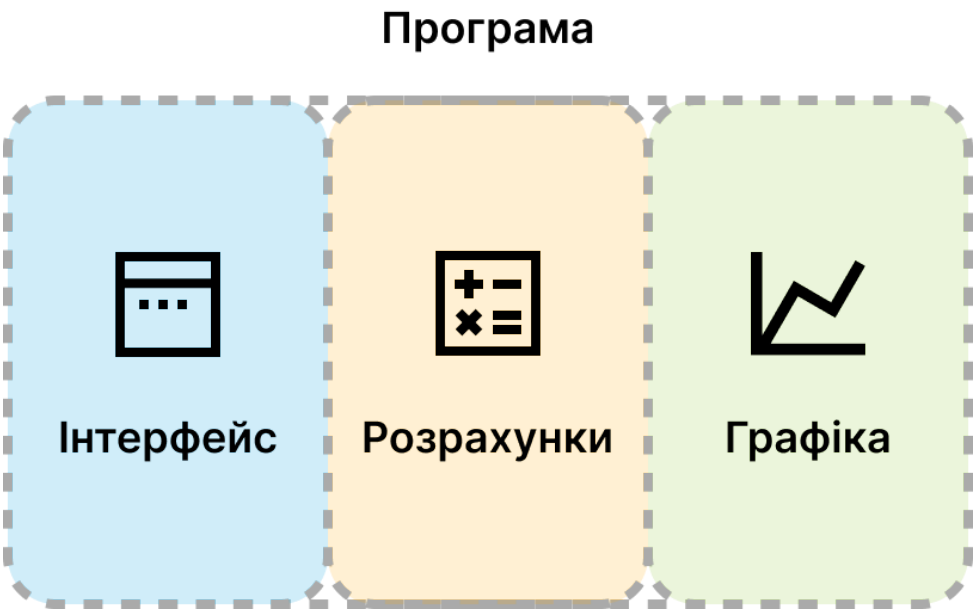


Рис. 3.1 Структура програми

3.2 Розробка компонентів програмного забезпечення

3.2.1 Підготовка вхідних даних

В якості наочного прикладу для цієї роботи було обрано застосовувати промисловий маніпулятор PUMA 762. Він є одним із класичних промислових

роботів, який широко використовується для виконання різноманітних завдань, таких як зварювання, збірка, пакування та інші операції. Щоб ефективно використовувати цей маніпулятор у задачах генетичного програмування та оптимізації, необхідно провести підготовку вхідних даних.

Маніпулятор PUMA 762 має шість ступенів свободи, що дозволяє йому виконувати складні рухи та маніпуляції в тривимірному просторі. Кінематичні параметри цього маніпулятора можна описати за допомогою параметрів Денавіта-Хартенберга (DH-параметри), які використовуються для визначення положення та орієнтації кожного з ланок маніпулятора відносно попередньої ланки. На рис 3.2 наведені DH-параметри для маніпулятора PUMA 762:

Joints	$\mathbf{d_i}$	$\boldsymbol{\alpha_i}$	$\mathbf{a_i}$	$\boldsymbol{\theta_i}$
1	0	-90	0	θ_1^*
2	0	0	650	θ_2^*
3	190	+90	0	$\theta_3^* + 90$
4	600	-90	0	θ_4^*
5	0	+90	0	θ_5^*
6	125	0	0	θ_6^*

Рис. 3.2 DH-параметри для маніпулятора PUMA 762

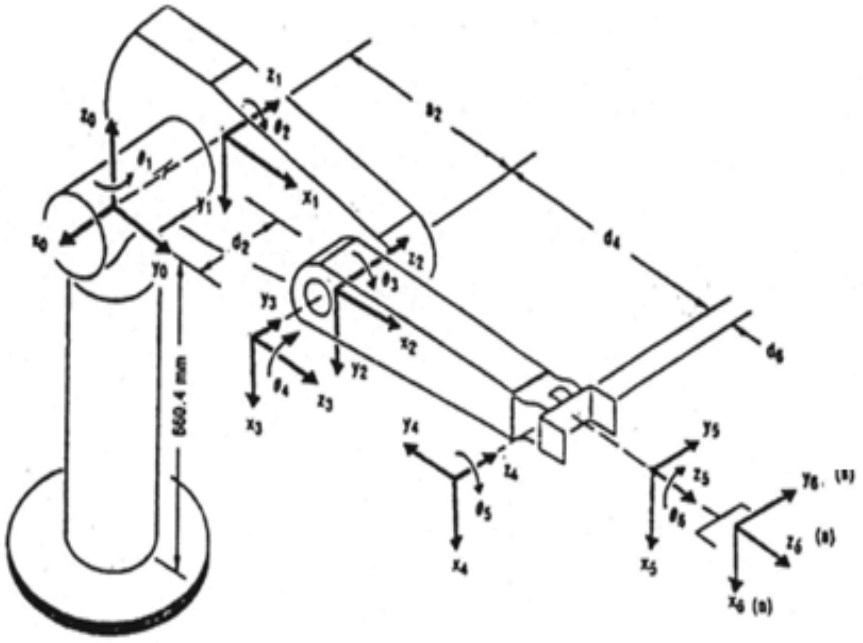


Рис. 3.3 Схематичне зображення параметрів маніпулятора PUMA 762

В коді маніпулятор буде мати наступний вигляд:

```
robot_chain = Chain(name='puma_762', links=[
    OriginLink(),
    URDFLink(
        name="link1",
        origin_translation=[0, 0, 0.675],
        origin_orientation=[0, 0, 0],
        rotation=[0, 0, 1]
    ),
    URDFLink(
        name="link2",
        origin_translation=[0.432, 0, 0],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link3",
        origin_translation=[0.02, 0, 0.120],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link4",
        origin_translation=[0.433, 0, 0],
```

```
        origin_orientation=[0, 0, 0],
        rotation=[1, 0, 0]
    ),
    URDFLink(
        name="link5",
        origin_translation=[0, 0, 0],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link6",
        origin_translation=[0, 0, 0.056],
        origin_orientation=[0, 0, 0],
        rotation=[1, 0, 0]
    )
])
```

Для роботи також знадобляться дані для навчання, тобто координати, які програмно задаються для руху маніпулятора та його фактичне положення після виконання програми.

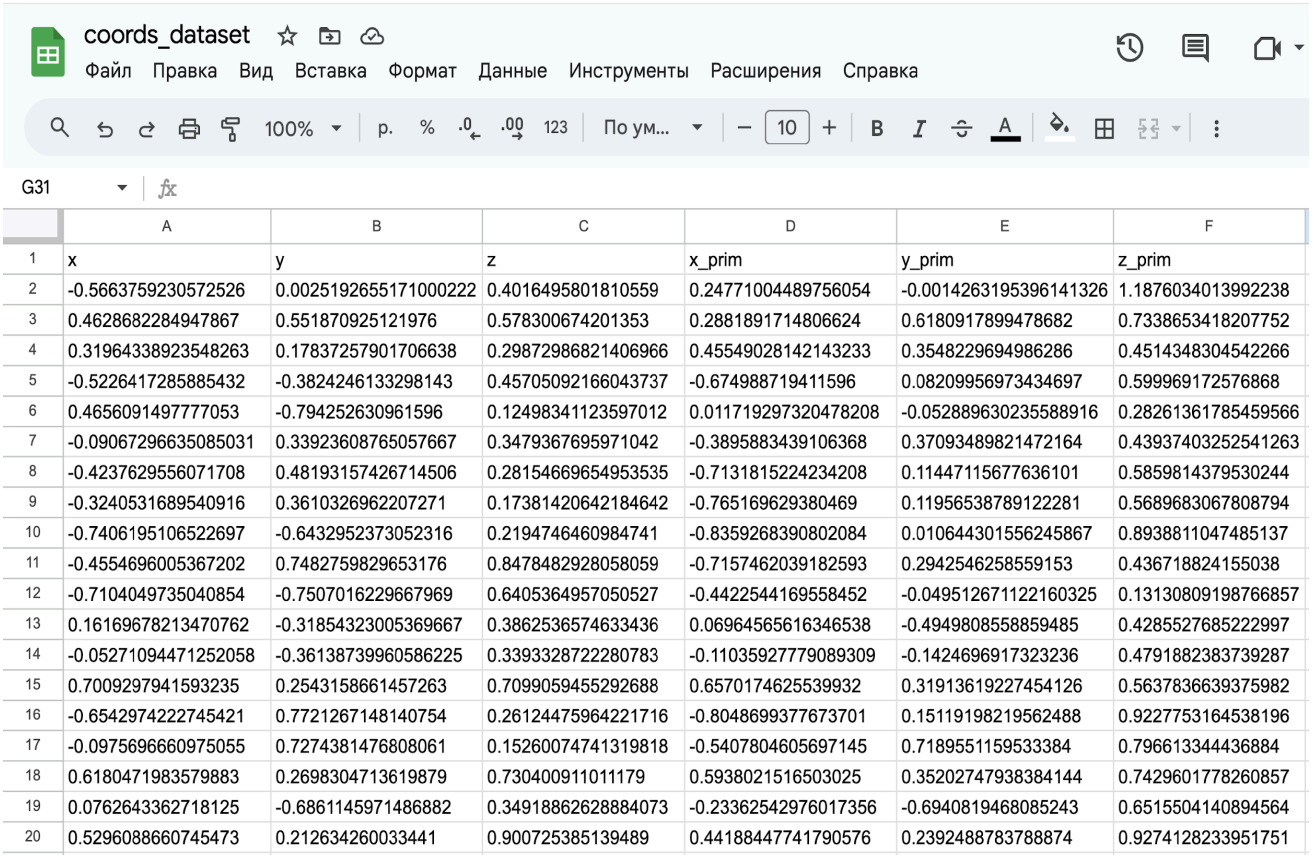
В роботі буде розглянуто завчасно сформований датасет координат (рис. 3.4), який включає похибку у розрахунку кожного з кутів відповідно:

$$\theta_1 = \theta_1 * 2$$

$$\theta_2 = \theta_2 - \sin(\theta_2)$$

$$\theta_3 = \theta_3 + \cos(\theta_3)$$

$$\theta_4 = \theta_4 / 2 + 0.5$$



	A	B	C	D	E	F
1	x	y	z	x_prim	y_prim	z_prim
2	-0.5663759230572526	0.0025192655171000222	0.4016495801810559	0.24771004489756054	-0.0014263195396141326	1.1876034013992238
3	0.4628682284947867	0.551870925121976	0.578300674201353	0.2881891714806624	0.6180917899478682	0.7338653418207752
4	0.31964338923548263	0.17837257901706638	0.29872986821406966	0.45549028142143233	0.3548229694986286	0.4514348304542266
5	-0.5226417285885432	-0.3824246133298143	0.45705092166043737	-0.674988719411596	0.08209956973434697	0.599969172576868
6	0.4656091497777053	-0.794252630961596	0.12498341123597012	0.011719297320478208	-0.052889630235588916	0.28261361785459566
7	-0.09067296635085031	0.33923608765057667	0.3479367695971042	-0.3895883439106368	0.37093489821472164	0.43937403252541263
8	-0.4237629556071708	0.48193157426714506	0.28154669654953535	-0.7131815224234208	0.11447115677636101	0.5859814379530244
9	-0.3240531689540916	0.3610326962207271	0.17381420642184642	-0.765169629380469	0.11956538789122281	0.5689683067808794
10	-0.7406195106522697	-0.6432952373052316	0.2194746460984741	-0.8359268390802084	0.010644301556245867	0.8938811047485137
11	-0.4554696005367202	0.7482759829653176	0.8478482928058059	-0.7157462039182593	0.2942546258559153	0.436718824155038
12	-0.7104049735040854	-0.7507016229667969	0.6405364957050527	-0.4422544169558452	-0.049512671122160325	0.13130809198766857
13	0.16169678213470762	-0.31854323005369667	0.3862536574633436	0.06964565616346538	-0.4949808558859485	0.4285527685222997
14	-0.05271094471252058	-0.36138739960586225	0.3393328722280783	-0.11035927779089309	-0.1424696917323236	0.4791882383739287
15	0.7009297941593235	0.2543158661457263	0.7099059455292688	0.6570174625539932	0.31913619227454126	0.5637836639375982
16	-0.6542974222745421	0.7721267148140754	0.26124475964221716	-0.8048699377673701	0.15119198219562488	0.9227753164538196
17	-0.0975696660975055	0.7274381476808061	0.15260074741319818	-0.5407804605697145	0.7189551159533384	0.796613344436884
18	0.6180471983579883	0.2698304713619879	0.730400911011179	0.5938021516503025	0.35202747938384144	0.7429601778260857
19	0.0762643362718125	-0.6861145971486882	0.34918862628884073	-0.23362542976017356	-0.6940819468085243	0.6515504140894564
20	0.5296088660745473	0.212634260033441	0.900725385139489	0.44188447741790576	0.2392488783788874	0.9274128233951751

Рис. 3.4 Файл з координатами для тестування програми

3.2.2 Інтерфейс програми

Інтерфейсна частина програмного забезпечення побудована на основі бібліотеки ipywidgets, яка дозволяє створювати інтерактивні віджети для Jupyter Notebook. Це забезпечує зручний та гнучкий спосіб взаємодії з алгоритмом генетичного програмування та його налаштуваннями.

Спершу визначаються основні елементи інтерфейсу, які будуть використовуватися для налаштування параметрів алгоритму. Використовується widgets.IntSlider для налаштування кількості поколінь (ngen_widget) та розміру популяції (population_size_widget). Ці елементи дозволяють користувачу вибрати відповідні значення за допомогою повзунків, що робить процес налаштування інтуїтивно зрозумілим. Кожен повзунок має визначені мінімальні, максимальні значення та крок, що дає змогу точно контролювати параметри.

```
ngen_widget = widgets.IntSlider(description='Generations', min=10, max=500,
step=10, value=100)
population_size_widget = widgets.IntSlider(description='Population Size',
min=50, max=500, step=10, value=200)
```

Наступним елементом є widgets.FloatSlider, який використовується для налаштування ймовірності схрещування (cxpb_widget) та ймовірності мутації (mutpb_widget). Ці повзунки працюють аналогічно до попередніх, але дозволяють встановлювати значення з плаваючою точкою, що є важливим для тонкого налаштування параметрів генетичного алгоритму.

```
cxpb_widget = widgets.FloatSlider(description='Crossover Prob', min=0.0,
max=1.0, step=0.1, value=0.5)
mutpb_widget = widgets.FloatSlider(description='Mutation Prob', min=0.0,
max=1.0, step=0.1, value=0.2)
```

Для вибору типу оператора схрещування використовується widgets.Dropdown. У випадаючому списку (cx_operator_widget) представлені різні типи операторів схрещування, такі як cxOnePoint, cxOnePointLeafBiased, cxSemantic. Користувач може вибрати один з них, що дозволяє змінювати поведінку генетичного алгоритму без потреби змінювати код програми.

```
cx_operator_widget = widgets.Dropdown(options=['cxOnePoint',
'cxOnePointLeafBiased'], description='Crossover')
```

Аналогічно, для вибору типу оператора мутації використовується ще один widgets.Dropdown. У випадаючому списку (mut_operator_widget) представлені різні типи операторів мутації, такі як mutShrink, mutEphemeral, mutInsert, mutNodeReplacement, mutUniform. Це дозволяє користувачу експериментувати з різними стратегіями мутації та обирати найбільш ефективну для конкретної задачі.

```
mut_operator_widget = widgets.Dropdown(options=['mutShrink', 'mutEphemeral',
'mutNodeReplacement', 'mutUniform'], description='Mutation')
```

Для завантаження вхідних даних використовується widgets.FileUpload. Користувач може завантажити CSV файл, який містить координати, необхідні для роботи алгоритму. Після завантаження файлу, він автоматично обробляється, і дані зберігаються у відповідних масивах (coords_array, coords_array_prim). Цей елемент інтерфейсу забезпечує зручний спосіб завантаження даних безпосередньо у програму, що робить процес більш інтерактивним та гнучким.

```
# Завантаження файлу з координатами
def load_file(change):
    global coords_array, coords_array_prim, angles_array, angles_array_prim
    uploaded_file = list(change['new'].values())[0]
    content = BytesIO(uploaded_file['content'])
    df_loaded = pd.read_csv(content)
    coords_array = df_loaded[['x', 'y', 'z']].values.tolist()
    coords_array_prim = df_loaded[['x_prim', 'y_prim',
'z_prim']].values.tolist()
    print("Файл завантажено успішно")

    angles_array = []
    angles_array_prim = []

    for coords in coords_array:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array.append(ik.tolist())
    for coords in coords_array_prim:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array_prim.append(ik.tolist())
```

Кнопка запуску алгоритму реалізована за допомогою widgets.Button. При натисканні на кнопку run_button, викликається функція run_gp, яка ініціалізує генетичне програмування з заданими параметрами.

Загальний інтерфейс програми зображений на рис. 3.5.

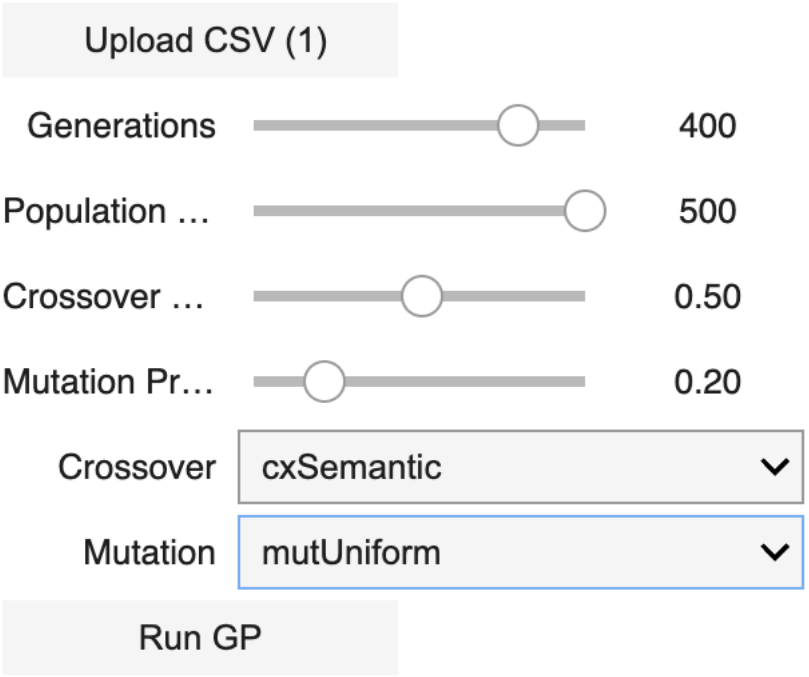


Рис. 3.5 Інтерфейс програми калібрування робота маніпулятора

Інтерфейс забезпечує інтерактивний та зручний спосіб налаштування і виконання генетичного алгоритму, дозволяючи користувачу легко змінювати параметри, завантажувати дані та аналізувати результати обчислень.

3.2.3 Опис розрахункової частини програми

Розрахункова частина програмного забезпечення відповідає за виконання символічної регресії та оптимізацію параметрів моделі робота-маніпулятора. Основними компонентами цієї частини є функції, що здійснюють обчислення пристосованості, виконання генетичних алгоритмів, обробку даних та реалізацію інструментів для генетичного програмування [29].

Функція eval_individual відповідає за оцінку пристосованості індивідуальних моделей. Вона приймає модель, дані та цільові значення, компілює модель у функцію та обчислює квадратичні помилки між прогнозованими та фактичними значеннями. Середнє значення цих помилок повертається як міра пристосованості.

```
def eval_individual(individual, data, target):
```

```
    func = toolbox.compile(expr=individual)
    try:
        sqerrors = [(func(x) - y)**2 for x, y in zip(data, target)]
    except ValueError:
        return float('inf'),
    return math.fsum(sqerrors) / len(data),
```

Функція symbolic_regression виконує символічну регресію, використовуючи генетичний алгоритм. Вона реєструє функцію оцінки пристосованості, ініціалізує популяцію моделей та виконує еволюційний алгоритм, використовуючи задані параметри. Після завершення алгоритму, функція повертає найкращу знайдену модель та логбук зі статистикою.

```
def symbolic_regression(data, target, ngen=500, population_size=100):
    toolbox.register("evaluate", partial(eval_individual, data=data,
target=target))
    population = toolbox.population(n=population_size)
    hof = tools.HallOfFame(1)

    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("mean", np.mean)
    stats.register("min", np.min)
    stats.register("max", np.max)

    logbook = tools.Logbook()
    logbook.header = ['gen', 'nevals'] + stats.fields

    population, log = algorithms.eaSimple(population, toolbox, cxpb=0.5,
mutpb=0.2, ngen=ngen, stats=stats, halloffame=hof, verbose=True)

    for entry in log:
        logbook.record(**entry)

    return hof[0], logbook
```

Для визначення набору примітивів для символічної регресії використовується об'єкт PrimitiveSet з бібліотеки DEAP. Цей набір включає основні арифметичні операції (додавання, віднімання, множення), абсолютне

значення, тригонометричні функції (синус, косинус) та ефемерні константи. Також додається примітив 'If' для семантичних операторів.

```
pset = gp.PrimitiveSet("MAIN", 1)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(operator.neg, 1)
pset.addPrimitive(abs, 1)
pset.addPrimitive(math.sin, 1)
pset.addPrimitive(math.cos, 1)
pset.addEphemeralConstant("rand101", lambda: random.uniform(-1, 1))
pset.renameArguments(ARG0='theta')
```

Ініціалізація інструментарію (toolbox) включає визначення методів генерації початкових виразів, створення індивідуумів та популяцій, компіляції виразів, операцій схрещування та мутації, а також функції селекції.

```
toolbox = base.Toolbox()
toolbox.register("expr", genHalfAndHalf, pset=pset, min_=1, max_=3)  # max_=3
для обмеження глибини дерева
toolbox.register("individual", tools.initIterate, creator.Individual,
toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", compile, pset=pset)
toolbox.register("select", tools.selTournament, tournsize=3)
```

Для завантаження файлу з координатами використовується функція load_file, яка читає дані з CSV файлу, розділяє їх на дві окремі матриці (coords_array, coords_array_prim) та обчислює відповідні кути за допомогою оберненої кінематики.

```
def load_file(change):
    global coords_array, coords_array_prim, angles_array, angles_array_prim
    uploaded_file = list(change['new'].values())[0]
    content = BytesIO(uploaded_file['content'])
    df_loaded = pd.read_csv(content)
    coords_array = df_loaded[['x', 'y', 'z']].values.tolist()
```

```
    coords_array_prim = df_loaded[['x_prim', 'y_prim',
'z_prim']].values.tolist()
    print("Файл завантажено успішно")

    angles_array = []
    angles_array_prim = []

    for coords in coords_array:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array.append(ik.tolist())
    for coords in coords_array_prim:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array_prim.append(ik.tolist())
```

Функція run_gp виконує основний алгоритм генетичного програмування, враховуючи налаштування користувача, включаючи вибір операторів схрещування та мутації. Вона також відповідає за обчислення прогнозованих кутів для кожного набору вхідних даних, обчислення помилок та збереження результатів.

```
def run_gp(b):
    results = []
    e_total_array = []
    with output:
        output.clear_output()
        ngen = ngen_widget.value
        population_size = population_size_widget.value
        cxpb = cxpb_widget.value
        mutpb = mutpb_widget.value

        cx_operator = cx_operator_widget.value
        mut_operator = mut_operator_widget.value

        # Вибір і реєстрація оператора схрещування
        if cx_operator == 'cxOnePoint':
            toolbox.register("mate", gp.cxOnePoint)
        elif cx_operator == 'cxOnePointLeafBiased':
            toolbox.register("mate", gp.cxOnePointLeafBiased, termpb=0.1)
        elif cx_operator == 'cxSemantic':
            toolbox.register("mate", gp.cxSemantic, pset=pset, min=0, max=2)
```

```
        # Вибір і реєстрація оператора мутації
        if mut_operator == 'mutShrink':
            toolbox.register("mutate", gp.mutShrink)
        elif mut_operator == 'mutEphemeral':
            toolbox.register("mutate", gp.mutEphemeral, mode='one')  # або
'all'
        elif mut_operator == 'mutNodeReplacement':
            toolbox.register("mutate", gp.mutNodeReplacement, pset=pset)
        elif mut_operator == 'mutUniform':
            toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr,
pset=pset)
        elif mut_operator == 'mutInsert':
            toolbox.register("mutate", gp.mutInsert, pset=pset)

        toolbox.decorate("mate",
gp.staticLimit(key=operator.attrgetter("height"), max_value=3))  # Обмеження
глибини дерева до 3
        toolbox.decorate("mutate",
gp.staticLimit(key=operator.attrgetter("height"), max_value=3))  # Обмеження
глибини дерева до 3

        theta_angles = rotate_matrix_clockwise(angles_array)
        theta_angles_prim = rotate_matrix_clockwise(angles_array_prim)

        logbooks = []

        for i in range(len(theta_angles)):
            best_expr_theta, logbook =
symbolic_regression(theta_angles_prim[i], theta_angles[i], ngen=ngen,
population_size=population_size, cxpb=cxpb, mutpb=mutpb)
            results.append(best_expr_theta)
            logbooks.append(logbook)

        for i in range(len(angles_array_prim)):
            calculated_angles = []
            for j in range(len(angles_array_prim[i])):
                func = toolbox.compile(expr=results[j])
                calculated_angles.append(func(angles_array_prim[i][j]))
```

```
            predictions =
robot_chain.forward_kinematics(calculated_angles)[:3, 3]
```

Ці функції забезпечують виконання символічної регресії та оптимізацію параметрів моделі робота-маніпулятора, що дозволяє досягти високої точності у визначенні положення та орієнтації кінцевого ефектора робота

Блок схему роботи розрахункової частини програми представлено на рис. 3.6

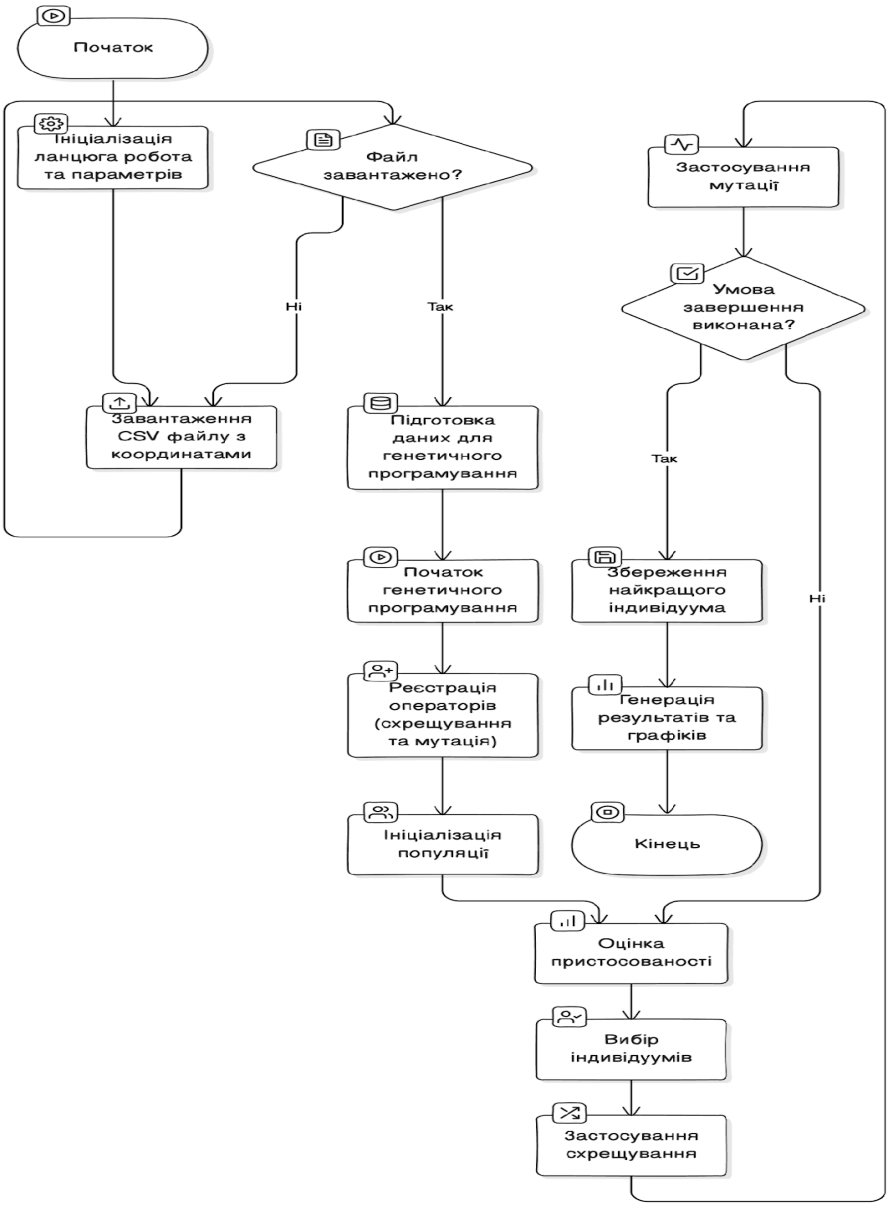


Рис. 3.6 Блок схема роботи розрахункової частини програми

3.2.3 Опис графічної частини програми

Графічна частина програмного забезпечення зосереджена на візуалізації результатів генетичного алгоритму та оцінці його ефективності. Вона включає кілька графіків, що допомагають аналізувати різні аспекти роботи алгоритму, включаючи фітнес-функції та абсолютні помилки позиціонування.

Один із ключових графіків відображає абсолютну позиційну помилку кінцевої точки робота до і після калібрування. Функція plot_absolute_positional_error обчислює помилки для початкових і скоригованих координат, а потім будує графік для порівняння цих значень. Це дозволяє візуально оцінити ефективність калібрування, яке виконується генетичним алгоритмом.

```
def plot_absolute_positional_error(coords_array, coords_array_prim,
e_total_array):
    e_total_array_prim = []

    for i in range(len(coords_array_prim)):
        e_x = coords_array_prim[i][0] - coords_array[i][0]
        e_y = coords_array_prim[i][1] - coords_array[i][1]
        e_z = coords_array_prim[i][2] - coords_array[i][2]
        e_total = math.sqrt(e_x**2 + e_y**2 + e_z**2)
        e_total_array_prim.append(e_total)

    plt.figure(figsize=(12, 6))
    plt.plot(e_total_array, label='Total Error After Calibration')
    plt.plot(e_total_array_prim, label='Positional Error Coords Array Prim',
linestyle='--')
    plt.xlabel('Sample')
    plt.ylabel('Error')
    plt.title('Absolute Positional Error of Robot Tool End Point')
    plt.legend()
    plt.show()
```

Наступний графік демонструє зміну фітнес-функції протягом поколінь для кожного суглоба. Функція plot_fitness_over_generations приймає на вхід логбуки,

що містять дані про мінімальну, середню та максимальну фітнес-значення в кожному поколінні. Ці дані використовуються для побудови графіків, які дозволяють стежити за прогресом алгоритму і його здатністю покращувати рішення з кожним поколінням.

```
def plot_fitness_over_generations(logbooks):
    rows = (len(logbooks) + 1) // 2
    fig, axes = plt.subplots(rows, 2, figsize=(15, 5 * rows))
    axes = axes.flatten()

    for i, logbook in enumerate(logbooks):
        gen = logbook.select("gen")
        fit_min = logbook.select("min")
        fit_mean = logbook.select("mean")
        fit_max = logbook.select("max")

        axes[i].plot(gen, fit_min, label="Min Fitness")
        axes[i].plot(gen, fit_mean, label="Mean Fitness")
        axes[i].plot(gen, fit_max, label="Max Fitness")
        axes[i].set_xlabel("Generation")
        axes[i].set_ylabel("Fitness")
        axes[i].set_title(f"Fitness over Generations for Joint {i+1}")
        axes[i].legend()

    plt.tight_layout()
    plt.show()
```

Інший важливий графік показує сумарну абсолютну помилку суглобів, яка є фітнес-функцією під час еволюції кожної індивідуальної корекційної моделі. Функція plot_summed_absolute_joint_error приймає логбуки та будує графіки для кожного суглоба, що дозволяє аналізувати, як зменшується помилка під час еволюційного процесу.

```
def plot_summed_absolute_joint_error(logbooks):
    fig, axes = plt.subplots((len(logbooks) + 1) // 2, 2, figsize=(15, 5 *
((len(logbooks) + 1) // 2)))
    axes = axes.flatten()
```

```
    for i, logbook in enumerate(logbooks):
        gen = logbook.select("gen")
        fit_sum = [logbook.select("min")[gen_idx] for gen_idx in
range(len(gen))]

        axes[i].plot(gen, fit_sum, label=f'Summed Absolute Joint Error for
Joint {i+1}')
        axes[i].set_xlabel('Generation')
        axes[i].set_ylabel('Summed Absolute Joint Error')
        axes[i].set_title(f'Summed Absolute Joint Error over Generations for
Joint {i+1}')
        axes[i].legend()

    plt.tight_layout()
    plt.show()
```

Ці графіки дозволяють користувачам бачити прогрес та ефективність генетичного алгоритму, оцінювати якість калібрування та визначати, наскільки добре алгоритм покращує моделі корекції суглобів робота. Кожен графік надає важливу інформацію про різні аспекти роботи алгоритму, що допомагає у подальшому аналізі та оптимізації.

3.3 Контрольний приклад

Контрольний приклад роботи програми починається з чіткого визначення мети тестування. Нашою метою є перевірка точності корекційних моделей для різних суглобів робота та оцінка здатності алгоритму зменшувати позиційні помилки.

Першим етапом є підготовка даних. Для цього ми використовуємо координати кінцевої точки робота до і після калібрування, які завантажуються з файлу CSV. Процес завантаження включає читання файлу та перетворення даних у відповідні масиви координат та кутів.

Наступним кроком є налаштування параметрів генетичного алгоритму. Ми обираємо кількість поколінь, розмір популяції, ймовірність схрещування та

мутації, а також типи операторів схрещування та мутації. Ці параметри можна налаштувати за допомогою інтерактивних віджетів, що дозволяє легко експериментувати з різними конфігураціями.

Налаштований інтерфейс з завантаженими даним зображено на рис. 3.7.

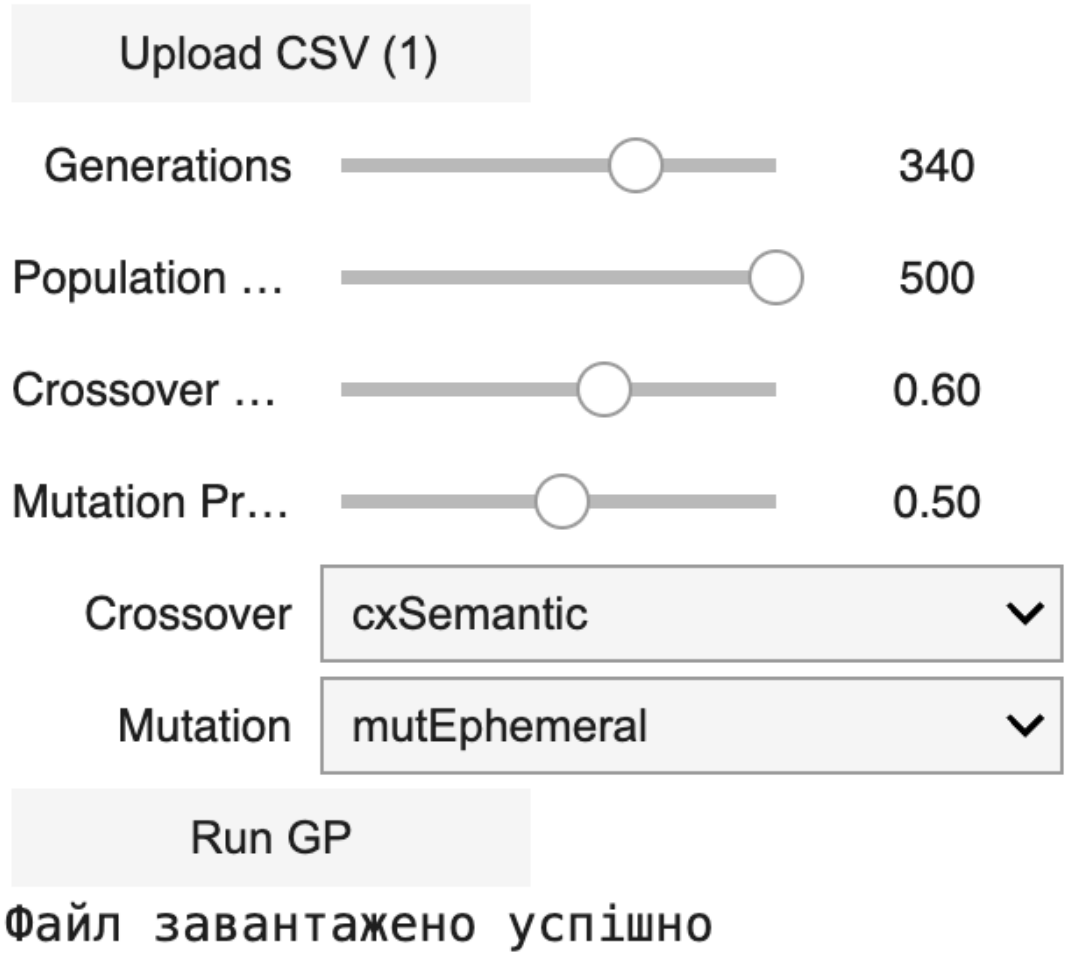


Рис. 3.7 Налаштування роботи генетичного алгоритму

Після налаштування параметрів запускаємо генетичний алгоритм з обраними налаштуваннями. Програма обирає оператори схрещування та мутації відповідно до налаштувань і виконує символічну регресію для кожного суглоба робота. Результати кожного покоління записуються у логбук для подальшого аналізу.

Коли алгоритм завершить своє виконання, переходимо до оцінки результатів. Для цього використовуємо різні графіки, що показують ефективність

роботи алгоритму. На рис. 3.8 зображено графік абсолютної позиційної помилки демонструє, наскільки зменшилася помилка після калібрування.

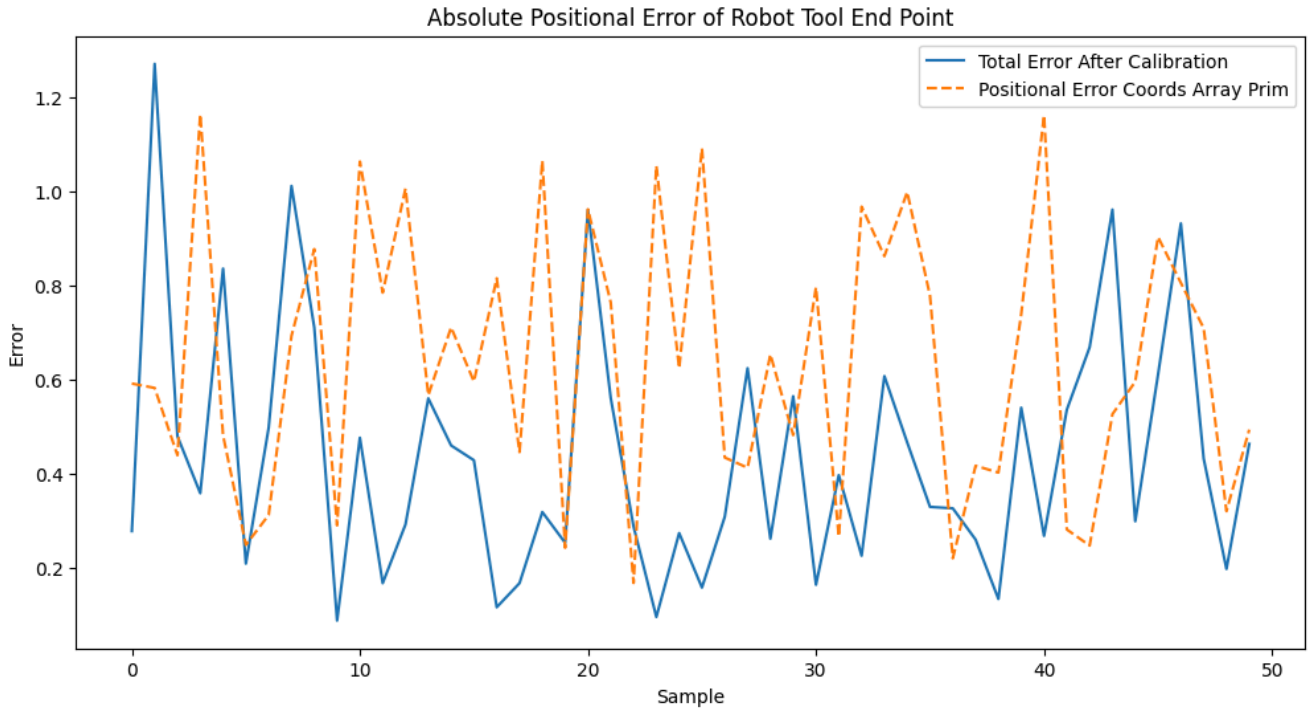


Рис. 3.8 Графік абсолютної позиційної помилки функціональної кінцівки робота маніпулятора

Кінцевий результат роботи програми, а саме набір найкращих генів, можна представити у вигляді таблиці, де кожен кут суглобу, θ, має власну функцію розрахунку.

Таблиця 3.1

Кут нахилу ланцюга	Аналітичне представлення
θ_1	$\lvert 0.2488\cdot\theta\rvert\cdot(\cos(-0.2009)+\lvert\theta\rvert)$
θ_2	$(\sin(\theta)+\sin(\theta))\cdot(\cos(0.4616)+\sin(0.4644))$
θ_3	$(-0.9998+\theta)\cdot\lvert-0.6550\rvert$
θ_4	$(0.9061-(-0.5868))+(0.4885+0.3681)$
θ_5	$-7.6623\times10-6+(-0.0722\cdot\theta)+(\cos(\theta)\cdot\theta)$
θ_6	$\cos(\theta)+(-0.9999+\theta)\cdot\lvert0.3340\rvert$
θ_7	$\sin(\theta)$

На рис. 3.9 зображено графік фітнес-функції, який показує зміну фітнесу протягом поколінь, що допомагає оцінити стабільність та швидкість збіжності алгоритму.

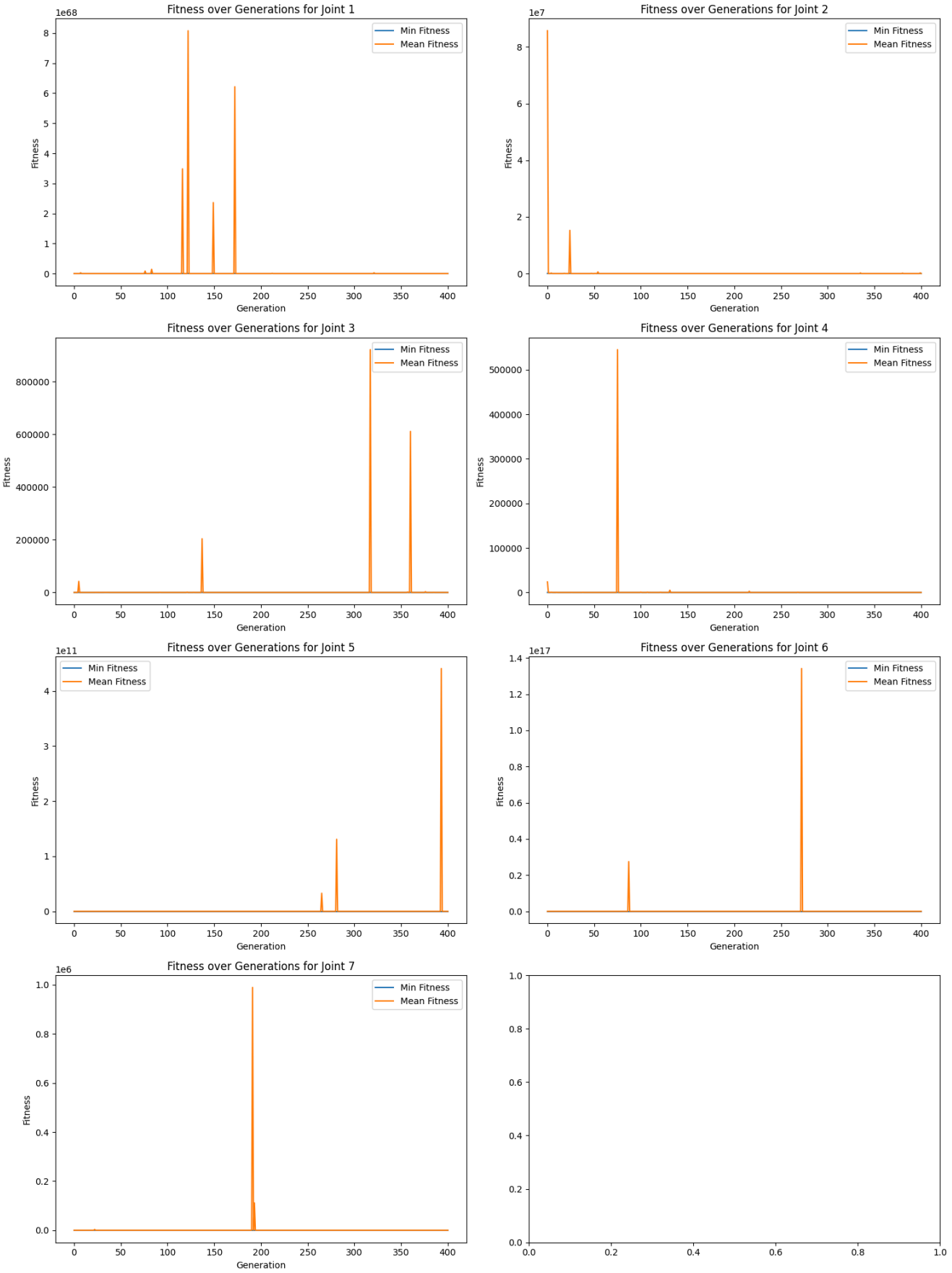


Рис. 3.9 Графік сумарної абсолютної помилки кожного суглобу

Одразу після нього, рис. 3.10, зображено графік сумарної абсолютної помилки суглобів дозволяє побачити, як зменшуються помилки для кожного суглоба окремо протягом поколінь.

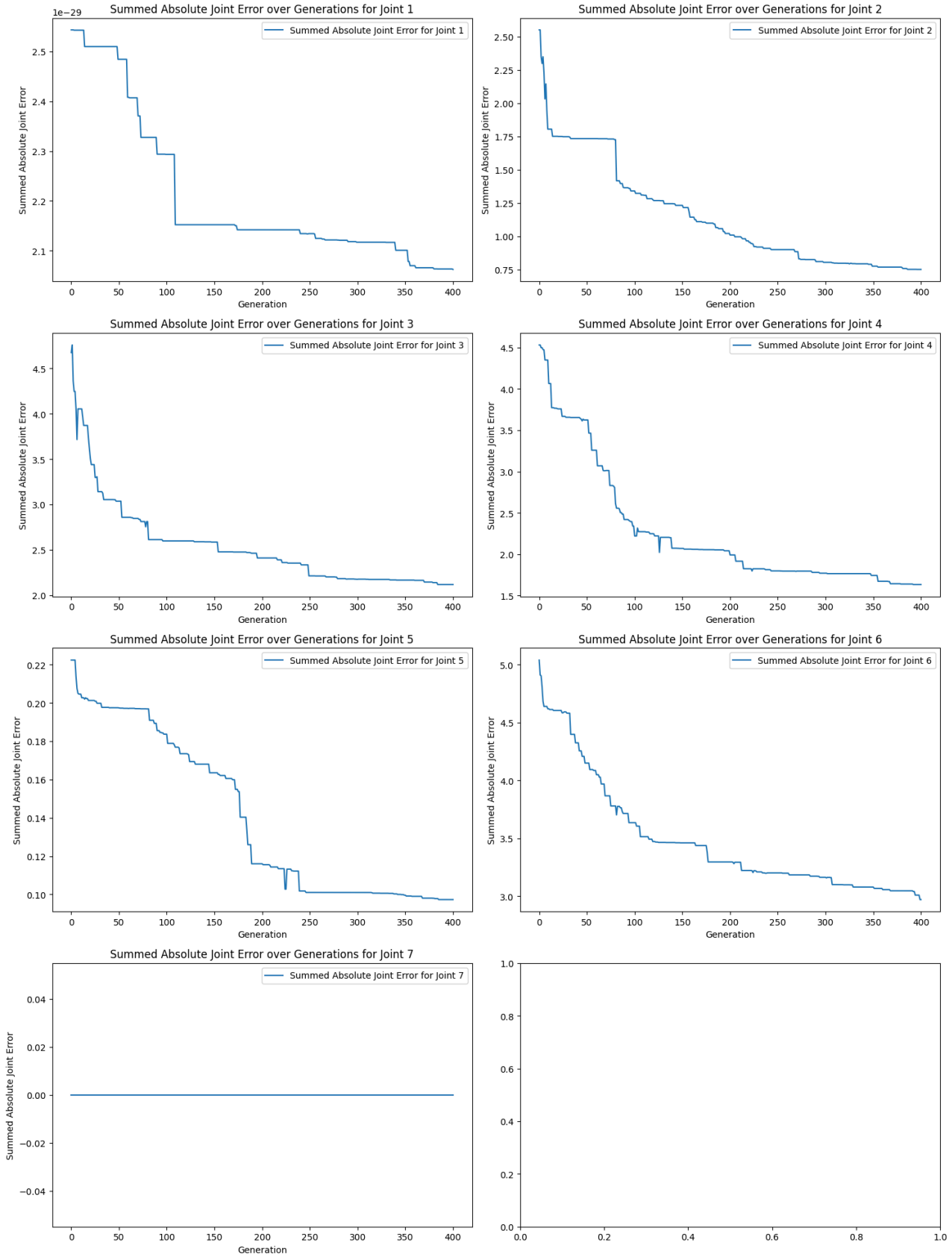


Рис. 3.10 Графік сумарної абсолютної помилки кожного суглобу

4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ ПІДСИСТЕМИ

4.1 Вступ

Розробка системи калібрування робота-маніпулятора методом генетичного алгоритму має на меті підвищення точності та ефективності роботи промислових роботів. Сучасні промислові роботи відіграють ключову роль у багатьох галузях, включаючи автомобільну, електронну, медичну та інші високотехнологічні галузі. Точність позиціювання маніпулятора є критично важливою для забезпечення високої якості продукції та зниження виробничих витрат. Використання генетичних алгоритмів для калібрування дозволяє автоматизувати процес налаштування робота, зменшити людський фактор та значно підвищити продуктивність.

4.2 Резюме

Назва проекту: "Підсистема калібрування роботів маніпуляторів методом генетичного алгоритму"

Місце розташування: Київ, Україна

Мета: Метою проекту є розробка інноваційної підсистеми калібрування для промислових роботів-маніпуляторів, яка базується на використанні генетичних алгоритмів. Цей підхід забезпечить високу точність та ефективність калібрування, що значно зменшить витрати на технічне обслуговування та підвищить продуктивність роботів.

Суть: Проект передбачає створення програмного забезпечення, яке дозволить автоматизувати процес калібрування промислових роботів-маніпуляторів. Основною технологією, що буде використана, є генетичні алгоритми, які відомі своєю здатністю вирішувати складні оптимізаційні задачі.

Це програмне забезпечення дозволить підприємствам, які використовують роботизовані системи, значно знизити витрати на обслуговування та підвищити ефективність виробничих процесів.

Програмне забезпечення буде включати інтуїтивно зрозумілий інтерфейс користувача, який дозволить завантажувати дані калібрування, налаштовувати параметри генетичного алгоритму та отримувати детальні звіти і графіки про результати калібрування. Система буде інтегруватися з різними типами промислових роботів, що робить її універсальним рішенням для різних галузей промисловості.

Проект також включає розробку спеціальних алгоритмів для обробки даних та оптимізації рухів робота. Це дозволить забезпечити максимальну точність та ефективність роботи маніпуляторів. Генетичні алгоритми будуть використовуватися для знаходження оптимальних параметрів калібрування, що дозволить зменшити похибки в роботі робота та підвищити його надійність.

Програмне забезпечення буде розроблятися з урахуванням сучасних вимог до безпеки та продуктивності. Це включає використання передових методів шифрування даних та захисту від несанкціонованого доступу. Система також буде мати можливість автоматичного оновлення, що дозволить постійно вдосконалювати її функціональність та забезпечувати підтримку нових моделей роботів.

Для реалізації проекту буде залучена команда висококваліфікованих спеціалістів у галузі робототехніки, програмування та машинного навчання. Вони будуть відповідати за розробку, тестування та впровадження системи. Планується також залучення експертів з промислових підприємств для отримання зворотного зв'язку та оптимізації продукту під реальні умови експлуатації.

Проект має високий потенціал для комерційного успіху завдяки інноваційному підходу та великому попиту на подібні рішення у промислових підприємствах. Впровадження підсистеми калібрування дозволить підприємствам підвищити ефективність виробничих процесів, знизити витрати на обслуговування та забезпечити високу якість продукції.

4.3 Опис проектованого продукту або вид послуг

Проект "Підсистема калібрування роботів маніпуляторів методом генетичного алгоритму" передбачає розробку програмного забезпечення для автоматичного калібрування промислових роботів-маніпуляторів. Основна функція цієї системи полягає у забезпеченні точного налаштування рухів робота для мінімізації похибок і підвищення ефективності його роботи. Програмне забезпечення буде використовувати методи генетичних алгоритмів для оптимізації параметрів калібрування.

Основні функції програмного забезпечення включають:

- Автоматичне калібрування: Програма автоматично проводить калібрування роботів, використовуючи дані, отримані з сенсорів та інших джерел. Це дозволяє зменшити похибки в роботі маніпулятора і забезпечити високу точність виконання завдань.
- Інтуїтивний інтерфейс користувача: Система буде мати зручний і простий у використанні інтерфейс, що дозволить користувачам легко завантажувати дані калібрування, налаштовувати параметри генетичного алгоритму, а також отримувати детальні звіти та графіки про результати калібрування.
- Підтримка різних моделей роботів: Програмне забезпечення буде сумісне з різними моделями промислових роботів-маніпуляторів, що робить його універсальним рішенням для багатьох підприємств.
- Аналіз і оптимізація рухів: Система буде використовувати передові алгоритми для аналізу рухів робота і їх оптимізації. Це дозволить значно підвищити ефективність роботи маніпулятора, зменшити знос його компонентів і збільшити тривалість експлуатації.
- Звіти і графіки: Користувачі зможуть отримувати детальні звіти про результати калібрування, а також переглядати графіки, що показують різні параметри роботи робота до і після калібрування.

- Можливість інтеграції: Програмне забезпечення буде мати можливість інтеграції з іншими системами управління виробництвом, що дозволить створити єдину інформаційну екосистему на підприємстві.

Переваги проектованого продукту:

- Точність і надійність: Використання генетичних алгоритмів дозволить досягти високої точності калібрування, що зменшить кількість дефектів у виробництві та підвищить якість продукції.
- Економія часу і ресурсів: Автоматизація процесу калібрування зменшить потребу в ручному налаштуванні роботів, що зекономить час і ресурси підприємства.
- Універсальність: Підтримка різних моделей роботів-маніпуляторів зробить систему корисною для широкого спектра промислових підприємств.
- Підвищення ефективності: Оптимізація рухів робота зменшить витрати енергії та знос компонентів, що підвищить загальну ефективність виробничих процесів.

4.4 Оцінка ринку збуту

Проект спрямований на промислові підприємства, які використовують роботів-маніпуляторів у своїх виробничих процесах. Ринок збуту для цього продукту охоплює широке коло галузей, включаючи автомобільну промисловість, електроніку, харчову промисловість, логістику, а також медичну галузь.

Основні сегменти ринку збуту:

- Автомобільна промисловість: У цій галузі роботів-маніпуляторів використовують для зварювання, фарбування, складання компонентів і інших виробничих процесів. Точне калібрування роботів дозволить зменшити кількість дефектів і підвищити якість продукції, що є критичним для автомобільних виробників.

- Електроніка: Виробництво електронних пристроїв вимагає високої точності і повторюваності рухів роботів. Наш продукт допоможе забезпечити ці вимоги, зменшивши кількість відходів і підвищивши ефективність виробництва.
- Харчова промисловість: У цій галузі роботи використовуються для пакування, сортування і обробки продуктів. Автоматичне калібрування дозволить забезпечити безперебійну роботу і високу продуктивність виробництва.
- Логістика: Роботи-маніпулятори активно використовуються для сортування і переміщення товарів на складах. Точне калібрування таких роботів дозволить збільшити швидкість і точність виконання логістичних операцій.
- Медична галузь: У медицині роботи використовуються для виконання делікатних операцій і маніпуляцій, де точність рухів є критичною. Наш продукт допоможе забезпечити необхідний рівень точності, що підвищить безпеку і ефективність медичних процедур.

За оцінками аналітиків, ринок промислових роботів зростає швидкими темпами. Очікується, що до 2026 року обсяг ринку досягне понад 70 мільярдів доларів США. Така тенденція пояснюється зростаючою автоматизацією виробничих процесів у всіх галузях промисловості. Це створює сприятливі умови для впровадження нашого продукту і дозволяє розраховувати на значний попит.

Проект орієнтований на міжнародний ринок, з акцентом на промислово розвинені країни, такі як США, Німеччина, Японія, Китай та інші. В цих країнах висока частка виробничих підприємств вже використовує роботів-маніпуляторів, що створює потенційний попит на нашу систему калібрування.

Потенційні клієнти:

- Великі промислові підприємства, що використовують роботів-маніпуляторів для автоматизації своїх виробничих процесів.
- Малі та середні підприємства, які прагнуть підвищити ефективність і якість своєї продукції шляхом впровадження автоматизованих систем.

- Інтегратори систем автоматизації, які можуть використовувати наш продукт для своїх клієнтів.

Переваги для клієнтів:

- Підвищення точності і надійності роботи роботів-маніпуляторів.
- Зниження витрат на ручне налаштування і обслуговування роботів.
- Зменшення кількості виробничих дефектів і підвищення якості продукції.
- Підвищення загальної продуктивності виробничих процесів.

4.5. Аналіз конкуренції

Наші переваги:

- Інноваційний підхід: Використання генетичних алгоритмів для калібрування роботів є новаторським підходом, який дозволяє автоматизувати процес налаштування з високою точністю і швидкістю. Це значно скорочує час і витрати на калібрування порівняно з традиційними методами.
- Висока точність: Генетичні алгоритми дозволяють знаходити оптимальні параметри калібрування з урахуванням всіх можливих варіантів налаштування, що забезпечує високу точність і надійність роботи роботів.
- Гнучкість і адаптивність: Наша система може бути легко адаптована для різних типів роботів і виробничих процесів, що робить її універсальним рішенням для широкого кола клієнтів.
- Зручність використання: Інтуїтивно зрозумілий інтерфейс і автоматизація процесу калібрування дозволяють користувачам легко налаштовувати і використовувати систему без потреби в спеціальних знаннях і навичках.
- Зниження витрат: Завдяки автоматизації і високій точності налаштування, наш продукт допомагає зменшити кількість

виробничих дефектів і знижує витрати на ручне налаштування і обслуговування роботів.

Стратегія конкуренції:

- Інновації і розвиток: Ми постійно працюємо над удосконаленням нашої системи, впроваджуючи нові алгоритми і технології для підвищення точності і ефективності калібрування роботів.
- Партнерство з виробниками: Співпраця з провідними виробниками промислових роботів дозволить інтегрувати нашу систему в їх рішення і розширити ринок збуту.
- Підтримка клієнтів: Ми надаємо високоякісну технічну підтримку і навчання для наших клієнтів, що допомагає їм максимально ефективно використовувати наш продукт.

4.6 Стратегія маркетингу

Ефективна маркетингова стратегія є ключем до успішного виходу на ринок та забезпечення стабільного зростання продажів нашого продукту. Враховуючи специфіку нашого ринку та конкурентне середовище, наша стратегія буде зосереджена на кількох основних напрямках.

Доступність: Ми пропонуємо конкурентні ціни на наш продукт, що робить його доступним для підприємств різного розміру. Цінова стратегія буде включати різні пакети послуг, щоб задовольнити потреби різних сегментів ринку.

Знижки та спеціальні пропозиції: Ми будемо надавати знижки для нових клієнтів, а також спеціальні умови для постійних клієнтів та навчальних закладів.

Просування продукту:

- Онлайн-просування: Ми активно використовуватимемо інтернет-маркетинг для просування нашого продукту. Це включатиме створення професійного веб-сайту, SEO-оптимізацію,

контент-маркетинг, а також використання соціальних мереж та платформи YouTube для демонстрації можливостей нашого продукту.

- Участь у виставках та конференціях: Ми плануємо брати участь у спеціалізованих виставках та конференціях, присвячених робототехніці та автоматизації, щоб презентувати наш продукт потенційним клієнтам та партнерам.
- Публікації у спеціалізованих виданнях: Ми будемо публікувати статті та огляди нашого продукту у професійних журналах та виданнях, що охоплюють тематику автоматизації та робототехніки.
- Реферальна програма: Ми розробимо реферальну програму, яка заохочуватиме наших клієнтів рекомендувати наш продукт своїм партнерам та колегам, пропонуючи їм знижки або бонуси за кожного нового клієнта.

Підтримка клієнтів:

- Технічна підтримка: Ми забезпечимо високий рівень технічної підтримки для наших клієнтів, надаючи консультації та допомогу у впровадженні та налаштуванні нашого продукту.
- Навчання та тренінги: Ми пропонуватимемо навчальні матеріали та тренінги для наших клієнтів, щоб вони могли ефективно використовувати наш продукт.
- Зворотний зв'язок: Ми будемо активно збирати зворотний зв'язок від наших клієнтів для постійного покращення нашого продукту та задоволення їхніх потреб.

Позиціонування продукту:

- Інноваційність та ефективність: Ми підкреслюватимемо інноваційний підхід нашого продукту, який базується на використанні генетичного алгоритму для калібрування роботів, що дозволяє досягти високої точності та ефективності.

- Гнучкість та універсальність: Ми будемо акцентувати увагу на гнучкості нашого рішення, яке може бути адаптоване для різних типів роботів та виробничих процесів.
- Доступність: Ми позиціонуватимемо наш продукт як доступне рішення для підприємств різного розміру, що дозволяє їм підвищити продуктивність та знизити витрати.

Реалізація цієї маркетингової стратегії дозволить нам ефективно вийти на ринок, залучити нових клієнтів та забезпечити стабільне зростання продажів нашого продукту.

4.7 План виробництва

Враховуючи специфіку проекту, наш виробничий процес буде включати кілька ключових етапів та заходів.

Етапи виробничого процесу:

1. Розробка програмного забезпечення:

- Аналіз вимог: Спільно з потенційними клієнтами та експертами з робототехніки ми збираємо та аналізуємо вимоги до підсистеми калібрування.

- Проектування системи: На основі зібраних вимог розробляємо архітектуру системи, визначаємо основні модулі та їх взаємодію.

2. Розробка коду: Виконуємо розробку програмного забезпечення, використовуючи мову програмування Python та відповідні бібліотеки для генетичного алгоритму та кінематики робота.

3. Тестування: Проводимо ретельне тестування системи, включаючи модульні тести, інтеграційні тести та функціональні тести, щоб забезпечити якість і надійність продукту.

4. Інтеграція:

- Інтеграція з роботами-маніпуляторами: Взаємодіємо з різними моделями роботів-маніпуляторів для забезпечення сумісності нашого програмного забезпечення.

5. Налаштування параметрів: Налаштовуємо параметри генетичного алгоритму відповідно до вимог конкретних клієнтів та їх виробничих процесів.

Впровадження та навчання:

- Встановлення програмного забезпечення: Забезпечуємо встановлення та налаштування нашої підсистеми на обладнанні клієнтів.
- Навчання персоналу: Проводимо тренінги та навчальні семінари для персоналу клієнтів, щоб вони могли ефективно використовувати нашу підсистему для калібрування роботів.

Необхідні ресурси:

1. Людські ресурси:
   - Команда розробників: Складається з програмістів, інженерів з робототехніки та фахівців з генетичних алгоритмів.
   - Тестувальники: Фахівці, відповідальні за тестування та забезпечення якості продукту.
   - Технічні консультанти: Експерти з робототехніки та автоматизації, які допомагають у налаштуванні та впровадженні системи у клієнтів.
2. Технічні ресурси:
   - Обладнання для розробки: Потужні комп'ютери та сервери для розробки та тестування програмного забезпечення.
   - Роботи-маніпулятори: Різні моделі роботів-маніпуляторів для забезпечення сумісності та тестування.
   - Програмне забезпечення: Ліцензії на необхідні інструменти та бібліотеки для розробки та тестування.

4.8 Організаційний план

1. Структура управління:
   - Керівник проекту: Відповідає за загальне керівництво проектом, координацію між різними відділами та забезпечення досягнення стратегічних цілей проекту.

- Команда розробників: Складається з програмістів, інженерів з робототехніки та фахівців з генетичних алгоритмів. Відповідає за розробку, тестування та впровадження програмного забезпечення.
- Тестувальники: Відповідальні за тестування продукту, виявлення та усунення помилок, забезпечення якості.
- Технічні консультанти: Надають експертну підтримку у питаннях робототехніки та автоматизації, допомагають клієнтам у налаштуванні та використанні системи.
- Відділ підтримки клієнтів: Відповідає за взаємодію з клієнтами, надання технічної підтримки та допомоги у вирішенні будь-яких проблем.

2. Кадровий склад:
   - Керівник проекту: 1 особа.
   - Розробники: 5 осіб.
   - Тестувальники: 3 особи.
   - Технічні консультанти: 2 особи.
   - Відділ підтримки клієнтів: 3 особи.
3. Основні функції та обов'язки:
   - Керівник проекту: Керування проектом, контроль за виконанням завдань, координація роботи команди, зв'язок з клієнтами.
   - Розробники: Розробка та тестування програмного забезпечення, написання коду, реалізація нових функцій.
   - Тестувальники: Проведення тестування, виявлення та виправлення помилок, забезпечення якості продукту.
   - Технічні консультанти: Надання консультацій, підтримка клієнтів у налаштуванні та використанні системи.
   - Відділ підтримки клієнтів: Забезпечення зворотного зв'язку з клієнтами, технічна підтримка, вирішення проблем.

4.9 Юридичний план

1. Форма власності:
    - Компанія буде зареєстрована як товариство з обмеженою відповідальністю (ТОВ), що забезпечить гнучкість управління та обмеження відповідальності власників.
2. Юридичні аспекти:
    - Реєстрація компанії: Реєстрація юридичної особи у відповідних органах державної влади.
    - Отримання ліцензій та дозволів: Отримання необхідних ліцензій та дозволів на розробку та впровадження програмного забезпечення.
    - Дотримання законодавства: Забезпечення відповідності всім законодавчим вимогам, включаючи податкове законодавство, трудове законодавство, закони про захист інтелектуальної власності.
3. Захист інтелектуальної власності:
    - Патенти та авторські права: Реєстрація патентів та авторських прав на розроблене програмне забезпечення для захисту від копіювання та несанкціонованого використання.
    - Ліцензійні угоди: Розробка та підписання ліцензійних угод з клієнтами, що визначають умови використання програмного забезпечення та права на його подальше використання.
4. Контракти та угоди:
    - Трудові договори: Укладення трудових договорів з працівниками, що визначають їх обов'язки, права та умови роботи.
    - Договори з клієнтами: Укладення договорів з клієнтами, що визначають умови надання послуг, гарантії та відповідальність сторін.

4.10 Фінансовий план

Оцінка витрат:

- Заробітна плата персоналу: Витрати на оплату праці всіх членів команди, включаючи розробників, тестувальників, технічних консультантів та працівників відділу підтримки клієнтів.
- Оренда офісу: Витрати на оренду приміщення для роботи команди.
- Обладнання та програмне забезпечення: Витрати на закупівлю комп'ютерного обладнання, ліцензій на використання необхідного програмного забезпечення та інших технічних засобів.
- Розробка та тестування: Витрати на розробку, тестування та впровадження програмного забезпечення, включаючи оплату за використання тестових середовищ та інструментів.
- Маркетинг та реклама: Витрати на проведення маркетингових кампаній, рекламних заходів та просування продукту на ринку.
- Адміністративні витрати: Витрати на адміністративну підтримку, включаючи послуги бухгалтерії, юридичні послуги та інші операційні витрати.

Прогноз доходів:

- Продаж програмного забезпечення: Очікувані доходи від продажу ліцензій на використання програмного забезпечення для калібрування роботів.
- Консультаційні послуги: Додаткові доходи від надання консультаційних послуг з налаштування та використання системи.
- Підписка на оновлення: Доходи від підписки на регулярні оновлення програмного забезпечення та технічну підтримку.

Аналіз рентабельності:

- Розрахунок чистого прибутку: Визначення чистого прибутку шляхом віднімання загальних витрат від загальних доходів.
- Показники рентабельності: Обчислення показників рентабельності, таких як рентабельність продажів (ROS) та рентабельність активів (ROA), для оцінки ефективності використання ресурсів.

4.11 Стратегія фінансування

- Інвестори: Залучення зовнішніх інвесторів, які зацікавлені у підтримці інноваційних технологій у сфері робототехніки. Це можуть бути венчурні капіталісти, бізнес-ангели або спеціалізовані інвестиційні фонди.
- Кредити та гранти: Отримання кредитів у банках на вигідних умовах для фінансування розвитку проекту. Також можливе отримання грантів від державних та міжнародних організацій, які підтримують інноваційні проекти.
- Доходи від попередніх продажів: Залучення коштів від попередніх продажів або передзамовлень на програмне забезпечення. Це дозволить отримати початкові кошти для розвитку проекту та забезпечити стабільний потік доходів.

4.12 Оцінка ризику

Технічні ризики:

- Неочікувані технічні проблеми:Можливість виникнення технічних проблем під час розробки або впровадження системи, які можуть затримати проект або збільшити його вартість.
- Збої в роботі програмного забезпечення: Можливість збоїв або помилок у програмному забезпеченні, що може призвести до некоректного калібрування робота та негативного впливу на продуктивність.
- Сумісність із обладнанням: Можливість проблем з інтеграцією системи з існуючими роботами та обладнанням, що може вимагати додаткових ресурсів для вирішення.

Фінансові ризики:

- Перевищення бюджету: Ризик перевищення запланованого бюджету через неочікувані витрати або помилки в плануванні.
- Недостатнє фінансування: Можливість недостатнього фінансування проекту через невдачі у залученні інвесторів або отриманні кредитів.

Ринкові ризики:

- Зміна попиту: Ризик зменшення попиту на продукт через зміну ринкових умов або появу нових конкурентів.
- Конкуренція: Поява нових конкурентів або поліпшення існуючих продуктів на ринку, що може зменшити частку ринку та вплинути на прибутковість проекту.

Юридичні ризики:

- Порушення інтелектуальної власності: Ризик порушення прав інтелектуальної власності інших компаній, що може призвести до юридичних проблем та фінансових витрат.
- Зміни в законодавстві: Можливість змін у законодавстві, які можуть вплинути на розробку, впровадження або використання системи.

4.13 Страхування

Страхування від технічних ризиків:

- Поліси страхування від збоїв в роботі програмного забезпечення, які покривають витрати на виправлення помилок та технічну підтримку.
- Страхування обладнання, яке використовується для розробки та тестування системи, від поломок та нещасних випадків.

Страхування від фінансових ризиків:

- Поліси страхування від фінансових втрат, пов'язаних з перевищенням бюджету або недостатнім фінансуванням.
- Страхування кредитів, отриманих для фінансування проекту, для захисту від ризику неспроможності виплат.

Страхування від ринкових ризиків:

- Поліси страхування від ринкових втрат, пов'язаних зі зменшенням попиту на продукт або появою нових конкурентів.
- Страхування від втрат через зміну ринкових умов, що можуть негативно вплинути на прибутковість проекту.

Юридичне страхування:

- Поліси страхування від юридичних ризиків, пов'язаних з порушенням інтелектуальної власності або змінами в законодавстві.
- Страхування від витрат на юридичні послуги та захист в суді у випадку виникнення юридичних спорів.

ВИСНОВКИ

У даній дипломній роботі було проведено дослідження та розробку підсистеми калібрування роботів маніпуляторів методом генетичного алгоритму. Розглянуто основні етапи реалізації, а також проведено тестування та оцінку ефективності запропонованого підходу.

Виявлено ключову проблему, пов'язану з точністю пересування маніпулятора. Кути, які програма передає на шарніри маніпулятора, можуть зазнавати спотворень і виконуватися з похибками, через що точність системи може падати.

Запропоновано та реалізовано алгоритм автоматизованого налаштування маніпуляторів, який використовує генетичні алгоритми для пошуку оптимальних корекційних функцій. Цей підхід дозволяє значно зменшити похибки в роботі маніпулятора, покращуючи точність виконання завдань.

Проведено тестування розробленої системи на прикладі робота маніпулятора PUMA 762. Використано симуляційні дані для оцінки ефективності алгоритму. Порівняння результатів показало, що запропонований алгоритм значно зменшує похибки в роботі маніпулятора.

Запропонована система показала високу ефективність у зменшенні похибок маніпулятора. Графік абсолютної позиційної помилки продемонстрував, що після калібрування загальна помилка значно зменшується, що свідчить про ефективність використання генетичних алгоритмів для корекції кутів шарнірів маніпулятора.

Запропонований підхід має ряд переваг порівняно з існуючими системами калібрування:

- Використання генетичних алгоритмів дозволяє адаптивно налаштовуватися під різні умови та завдання.
- Висока точність корекції забезпечується завдяки оптимізації на основі реальних даних.

- Гнучкість налаштувань та інтерактивний інтерфейс забезпечують зручність використання та можливість швидкого коригування параметрів.

На основі отриманих результатів рекомендується використовувати розроблену систему для калібрування роботів маніпуляторів у виробничих середовищах, де висока точність є критично важливою. Система може бути впроваджена на підприємствах, що займаються автоматизацією виробничих процесів, зокрема в автомобільній та електронній промисловості. Результати даної роботи можуть бути використані для подальшого розвитку систем автоматизованого калібрування роботів маніпуляторів.

Таким чином, дана дипломна робота зробила значний внесок у розвиток методів автоматизованого калібрування роботів маніпуляторів, показавши високу ефективність запропонованого підходу та його переваги над існуючими аналогами.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. AN OVERVIEW AND ANALYSIS OF THE CURRENT STATE OF THE MARKET FOR INDUSTRIAL ROBOT MANIPULATORS. *International scientific research journal*. URL: https://research-journal.org/media/articles/3491.pdf (date of access: 07.06.2024).
2. Artificial Intelligence and Machine Learning. AIML-4-3-6 Zadacha rehressyy dlia kalybrovky manypuliatora, 2015. *YouTube*. URL: https://www.youtube.com/watch?v=JXmcHmiSw5M (data zvernennia: 07.06.2024).
3. Chaudhary S. Genetic Algorithm Applications in Machine Learning. *AI-Powered Engineering Services, LLM Training, Teams | Turing*. URL: https://www.turing.com/kb/genetic-algorithm-applications-in-ml (date of access: 07.06.2024).
4. Contributors to Wikimedia projects. Denavit—Hartenberg parameters - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Denavit—Hartenberg_parameters (date of access: 07.06.2024).
5. Craig J. J. Introduction to robotics: Mechanics and control. Upper Saddle River, NJ, USA : Pearson Education, 2005. 408 p.
6. de Franca F. O., Aldeia G. S. I. Interaction-Transformation Evolutionary Algorithm for Symbolic Regression. *Evolutionary Computation*. 2020. P. 1—25. URL: https://doi.org/10.1162/evco_a_00285 (date of access: 07.06.2024).
7. Design, Construction and Control of a Manipulator Driven by Pneumatic Artificial Muscles. *MDPI*. URL: https://www.mdpi.com/1424-8220/23/2/776 (date of access: 07.06.2024).

8. Dolinsky J. U., Jenkinson I. D., Colquhoun G. J. Application of genetic programming to the calibration of industrial robots. *Computers in Industry*. 2007. Vol. 58, no. 3. P. 255—264. URL: https://doi.org/10.1016/j.compind.2006.06.003 (date of access: 07.06.2024).
9. Dynamics and Control of Robot Manipulators / ed. by M. Carricato, E. Idà. MDPI, 2023. URL: https://doi.org/10.3390/books978-3-0365-8427-0 (date of access: 07.06.2024).
10. Evolutionary Algorithms in Engineering Design Optimization. MDPI, 2022. URL: https://doi.org/10.3390/books978-3-0365-2715-4 (date of access: 07.06.2024).
11. Global Industrial Robotics Market Size & Share Report, 2030. *Market Research Reports & Consulting | Grand View Research, Inc.* URL: https://www.grandviewresearch.com/industry-analysis/industrial-robotics-market (date of access: 07.06.2024).
12. IFR Press Conference. *International Federation of Robotics*. URL: https://ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf (date of access: 07.06.2024).
13. Industrial Robotics: Programming, Simulation and Applications / ed. by L. Kin. Pro Literatur Verlag, Germany / ARS, Austria, 2006. URL: https://doi.org/10.5772/40 (date of access: 07.06.2024).
14. Introduction to Genetic Algorithms. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. URL: https://doi.org/10.1007/978-3-540-73190-0 (date of access: 07.06.2024).
15. Joshi G. Optimization Algorithms for Distributed Machine Learning. Cham : Springer International Publishing, 2023. URL: https://doi.org/10.1007/978-3-031-19067-4 (date of access: 07.06.2024).
16. Lamont G. B., Coello C. C., Veldhuizen D. A. v. Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, 2014. 800 p.

17. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. O'Reilly Media, 2022. 550 p.
18. Ngo X. K. A Review of Artificial Intelligence Algorithms used for Robotic Manipulator. *International Journal for Research in Applied Science and Engineering Technology*. 2020. Vol. 8, no. 9. P. 410—416. URL: https://doi.org/10.22214/ijraset.2020.31375 (date of access: 07.06.2024).
19. Open Software Structure for Controlling Industrial Robot Manipulators. *intechopen*. URL: https://cdn.intechopen.com/pdfs/10496/InTech-Open_software_structure_for_controlling_industrial_robot_manipulators.pdf (date of access: 07.06.2024).
20. Poli R. A field guide to genetic programming. [S.I.] : [Lulu Press], lulu.com, 2008. 233 p.
21. PwC's 2019 actuarial robotic process automation (RPA) survey report. *PwC*. URL: https://www.pwc.com/gx/en/industries/financial-services/publications/pwc-2019-actuarial-robotic-process-automation-survey-report.html (date of access: 07.06.2024).
22. Schütze O., Hernández C. Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms. Cham : Springer International Publishing, 2021. URL: https://doi.org/10.1007/978-3-030-63773-6 (date of access: 07.06.2024).
23. Seth Chandler. Symbolic Regression. Video 32., 2021. *YouTube*. URL: https://www.youtube.com/watch?v=eMcBMvy_tlk (date of access: 07.06.2024).
24. Siciliano B. Robotics: Modelling, planning and control. London : Springer, 2009. 632 p.
25. Spong M. W. Robot modeling and control. Hoboken, NJ : John Wiley & Sons, 2006. 478 p.
26. Steve Brunton. Machine Learning Control: Genetic Programming, 2018. *YouTube*. URL: https://www.youtube.com/watch?v=K2Hl7m2Ty_4 (date of access: 07.06.2024).

27. Symbolic Regression on Noisy Data with Stepwise Genetic Programming Algorithm / H. R. Zhang et al. *Applied Mechanics and Materials*. 2014. Vol. 530-531. P. 625—628. URL: https://doi.org/10.4028/www.scientific.net/amm.530-531.625 (date of access: 07.06.2024).
28. Symvolna rehresiia. *Khabr*. URL: https://habr.com/ru/articles/163195/ (data zvernennia: 07.06.2024).
29. Wirsansky E. Hands-On Genetic Algorithms with Python: Applying Genetic Algorithms to Solve Real-World Deep Learning and Artificial Intelligence Problems. Packt Publishing, Limited, 2020. 346 p.
30. Zhang W., Xu J., Yu T. X. Dynamic behaviors of bio-inspired structures: Design, mechanisms, and models. *Engineering Structures*. 2022. Vol. 265. P. 114490. URL: https://doi.org/10.1016/j.engstruct.2022.114490 (date of access: 07.06.2024).

Додаток А

УДК 004.8, 004.9

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ГЕНЕТИЧНОГО АЛГОРИТМУ

Гаранський К., студент, КН-31, КНУБА
Горда О., к. т. н., доцент, *ORCID: 0000-0001-9548-1959*

Вступ. Сьогодні проблема автоматизації та оптимізації процесів дуже актуальна. Ми часто стикаємось з ними у різного типу галузях, починаючи з інженерних, економічних, логістичних задач, закінчуючи задачами штучного інтелекту, включаючи навіть задачі військового характеру. Генетичний алгоритм (ГА) – це адаптивний евристичний метод оптимізаційного пошуку, заснований на механізмі генетичної спадковості, що запозичений у природи. У сучасних роботах наведена повна перевага ГА над алгоритмами повного перебору у випадках, коли розмір популяції більше 20 [1].

Мета. Розробити програмну реалізацію генетичного алгоритму з метою впровадження у навчальний процес для вивчення та дослідження сучасних інтелектуальних методів оптимізації та прийняття рішень. В якості прикладу розглядається задача комівояжера з використанням на графів для візуалізації рішення проблеми.

Виклад основного матеріалу. В реалізації Га можна виділити чотири основних кроки (рис. 1):

1) створення популяції – за заданою кількістю популяцій та кількістю індивідів створюються шляхи (послідовності ребер, за якими буде проходити алгоритм);
2) селекція – вибір двох батьків для схрещування;
3) кросовер – схрещування батьків та мутація (частина одного батька йде до іншого і навпаки);
4) знаходження найкращого індивіда.

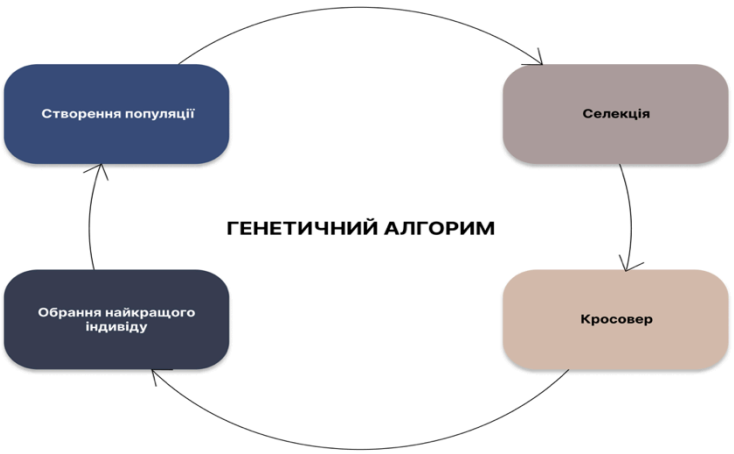


Рис. 1 Схема генетичного алгоритму

Причому, кожний крок алгоритму може мати різні способи реалізації. З метою створення гнучкого, еволюційного програмного додатку в його основу покладене управляюче ядро, яке служить каркасом системи у якому передбачені ніші для розміщення бібліотек функцій, що реалізують відповідний крок ГА. В результаті отримаємо додаток, який легко доповнювати та редагувати.

Так як, програмний додаток створюється з метою вивчення студентами ГА, то особлива увага приділена візуалізації окремих кроків. У інтерфейсній частині відображається процес створення самого графу, його ребер та результат роботи алгоритму (найкоротший шлях) (рис. 2).

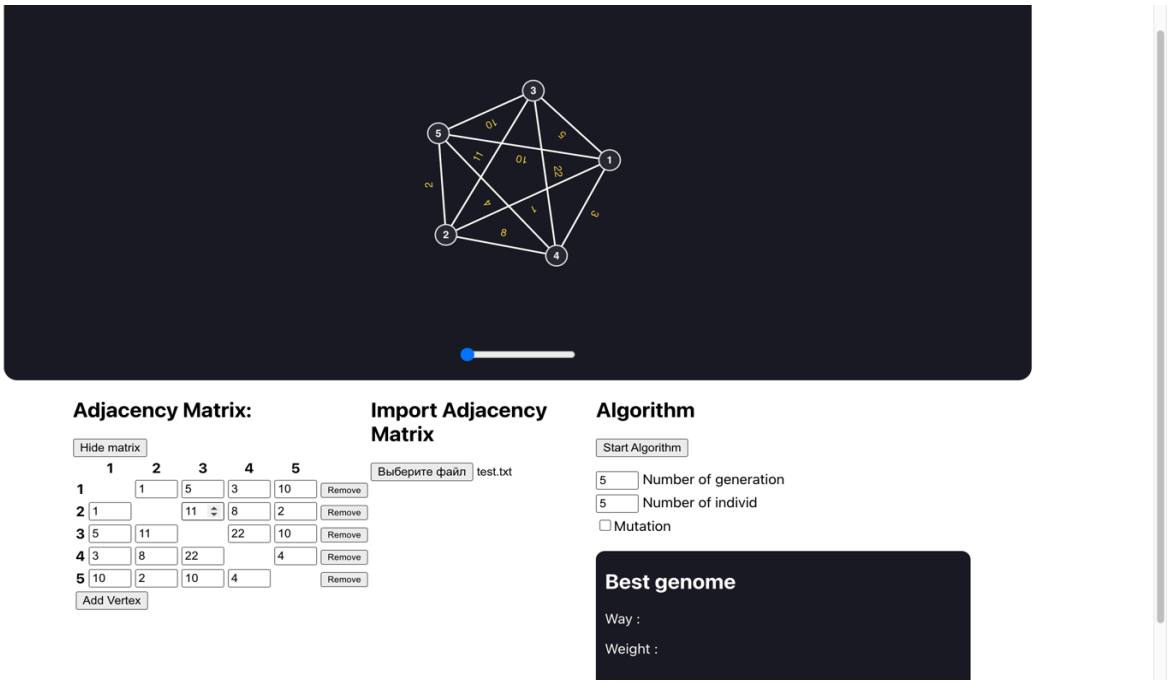


Рис. 2 Інтерфейсна частина програмного додатку

За допомогою спеціальних інпутів надається можливість змінювати параметри роботи алгоритму: кількість популяцій, кількість осіб у популяції, режим роботи мутації (увімкнений або вимкнений). Також передбачена можливість редагування графа за допомогою корегування матриці суміжності.

Висновок. Розроблений та реалізований на мові C++ програмний додаток для ГА. З метою покращення когнітивних характеристик додатку для студентів, передбачена візуалізація основних кроків алгоритму.

Використана література:

1. Федоров Е. А. Исследование скорости работы генетического алгоритма и алгоритма полного перебора // Сборник избранных статей научной сессии ТУСУР. №1. 2019. – С. 107-109.

Додаток Б

```
import random
import numpy as np
import pandas as pd
from deap import base, creator, gp, tools, algorithms
from deap.gp import compile, genHalfAndHalf, PrimitiveTree, cxOnePoint,
cxOnePointLeafBiased, mutUniform, mutEphemeral, mutInsert, mutNodeReplacement,
mutShrink
import operator
import math
from functools import partial
import ipywidgets as widgets
from IPython.display import display, clear_output
import matplotlib.pyplot as plt
from collections import defaultdict
from inspect import isclass
from io import BytesIO
from ikpy.chain import Chain
from ikpy.link import URDFLink, OriginLink

# Ініціалізація цепи робота PUMA 762
robot_chain = Chain(name='puma_762', links=[
    OriginLink(),
    URDFLink(
        name="link1",
        origin_translation=[0, 0, 0.675],
        origin_orientation=[0, 0, 0],
        rotation=[0, 0, 1]
    ),
    URDFLink(
        name="link2",
        origin_translation=[0.432, 0, 0],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link3",
        origin_translation=[0.02, 0, 0.120],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link4",
        origin_translation=[0.433, 0, 0],
        origin_orientation=[0, 0, 0],
        rotation=[1, 0, 0]
    ),
    URDFLink(
        name="link5",
        origin_translation=[0, 0, 0],
        origin_orientation=[0, 0, 0],
        rotation=[0, 1, 0]
    ),
    URDFLink(
        name="link6",
        origin_translation=[0, 0, 0.056],
        origin_orientation=[0, 0, 0],
        rotation=[1, 0, 0]
    )
```

```
])

# Ініціалізація ланцюга робота
num_joints = len(robot_chain.links)
results = []
e_total_array = []
coords_array = []
coords_array_prim = []
angles_array = []
angles_array_prim = []

# Функція для повороту матриці за годинниковою стрілкою
def rotate_matrix_clockwise(matrix):
    transposed_matrix = list(zip(*matrix))
    rotated_and_reversed_matrix = [list(row)[::-1] for row in
transposed_matrix]
    return rotated_and_reversed_matrix

# Функція для повороту матриці проти годинникової стрілки
def rotate_matrix_counterclockwise(matrix):
    transposed_matrix = list(zip(*matrix))
    rotated_matrix = [list(row) for row in transposed_matrix[::-1]]
    return rotated_matrix

# Завантаження файлу з координатами
def load_file(change):
    global coords_array, coords_array_prim, angles_array, angles_array_prim
    uploaded_file = list(change['new'].values())[0]
    content = BytesIO(uploaded_file['content'])
    df_loaded = pd.read_csv(content)
    coords_array = df_loaded[['x', 'y', 'z']].values.tolist()
    coords_array_prim = df_loaded[['x_prim', 'y_prim',
'z_prim']].values.tolist()
    print("Файл завантажено успішно")

    angles_array = []
    angles_array_prim = []

    for coords in coords_array:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array.append(ik.tolist())
    for coords in coords_array_prim:
        ik = robot_chain.inverse_kinematics(target_position=coords)
        angles_array_prim.append(ik.tolist())

# Опис функції для обчислення пристосованості
def eval_individual(individual, data, target):
    func = toolbox.compile(expr=individual)
    sqerrors = ((func(x) - y)**2 for x, y in zip(data, target))
    return math.fsum(sqerrors) / len(data),

# Функція для виконання символічної регресії
def symbolic_regression(data, target, ngen=500, population_size=1000,
cxpb=0.5, mutpb=0.6):
    toolbox.register("evaluate", partial(eval_individual, data=data,
target=target))
    population = toolbox.population(n=population_size)
    hof = tools.HallOfFame(1)

    stats = tools.Statistics(lambda ind: ind.fitness.values)
```

```
    stats.register("mean", np.mean)
    stats.register("min", np.min)
    stats.register("max", np.max)

    with output:
        clear_output()
        print("Running Genetic Programming...")

    population, logbook = algorithms.eaSimple(population, toolbox, cxpb=cxpb,
mutpb=mutpb, ngen=ngen, stats=stats, halloffame=hof, verbose=False)
    return hof[0], logbook

# Опис примітивного набору для символічної регресії
pset = gp.PrimitiveSet("MAIN", 1)  # Один вхід x (кут theta)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(abs, 1)
pset.addPrimitive(math.sin, 1)
pset.addPrimitive(math.cos, 1)
pset.addEphemeralConstant("rand101", partial(random.uniform, -1, 1))
pset.renameArguments(ARG0='theta')

# Додаємо примітив 'lf' для семантичних операторів
pset.addPrimitive(lambda x: x, 1, name="lf")

# Визначення типів
if "FitnessMin" not in creator.__dict__:
    creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
if "Individual" not in creator.__dict__:
    creator.create("Individual", PrimitiveTree, fitness=creator.FitnessMin)

# Ініціалізація toolbox
toolbox = base.Toolbox()
toolbox.register("expr", genHalfAndHalf, pset=pset, min_=1, max_=3)  # max_=3
для обмеження глибини дерева
toolbox.register("individual", tools.initIterate, creator.Individual,
toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", compile, pset=pset)
toolbox.register("select", tools.selTournament, tournsize=3)

# Виджети для налаштування параметрів
ngen_widget = widgets.IntSlider(description='Generations', min=10, max=500,
step=10, value=100)
population_size_widget = widgets.IntSlider(description='Population Size',
min=50, max=500, step=10, value=200)
cxpb_widget = widgets.FloatSlider(description='Crossover Prob', min=0.0,
max=1.0, step=0.1, value=0.5)
mutpb_widget = widgets.FloatSlider(description='Mutation Prob', min=0.0,
max=1.0, step=0.1, value=0.2)
cx_operator_widget = widgets.Dropdown(options=['cxOnePoint',
'cxOnePointLeafBiased', 'cxSemantic'], description='Crossover')
mut_operator_widget = widgets.Dropdown(options=['mutShrink', 'mutEphemeral',
'mutInsert', 'mutNodeReplacement', 'mutUniform'], description='Mutation')
file_upload = widgets.FileUpload(description="Upload CSV")
run_button = widgets.Button(description="Run GP")

output = widgets.Output()
```

```
def plot_absolute_positional_error(coords_array, coords_array_prim,
e_total_array):
    e_total_array_prim = []

    for i in range(len(coords_array_prim)):
        e_x = coords_array_prim[i][0] - coords_array[i][0]
        e_y = coords_array_prim[i][1] - coords_array[i][1]
        e_z = coords_array_prim[i][2] - coords_array[i][2]
        e_total = math.sqrt(e_x**2 + e_y**2 + e_z**2)
        e_total_array_prim.append(e_total)

    plt.figure(figsize=(12, 6))
    plt.plot(e_total_array, label='Total Error After Calibration')
    plt.plot(e_total_array_prim, label='Positional Error Coords Array Prim',
linestyle='--')
    plt.xlabel('Sample')
    plt.ylabel('Error')
    plt.title('Absolute Positional Error of Robot Tool End Point')
    plt.legend()
    plt.show()

# Генерація додаткових графіків
def generate_additional_plots(e_total_array, logbooks):
    plot_absolute_positional_error(coords_array, coords_array_prim,
e_total_array)
    plot_fitness_over_generations(logbooks)
    plot_summed_absolute_joint_error(logbooks)

def plot_fitness_over_generations(logbooks):
    rows = (len(logbooks) + 1) // 2
    fig, axes = plt.subplots(rows, 2, figsize=(15, 5 * rows))
    axes = axes.flatten()

    for i, logbook in enumerate(logbooks):
        gen = logbook.select("gen")
        fit_min = logbook.select("min")
        fit_mean = logbook.select("mean")
        fit_max = logbook.select("max")

        axes[i].plot(gen, fit_min, label="Min Fitness")
        axes[i].plot(gen, fit_mean, label="Mean Fitness")
        axes[i].set_xlabel("Generation")
        axes[i].set_ylabel("Fitness")
        axes[i].set_title(f"Fitness over Generations for Joint {i+1}")
        axes[i].legend()

    plt.tight_layout()
    plt.show()

def plot_summed_absolute_joint_error(logbooks):
    fig, axes = plt.subplots((len(logbooks) + 1) // 2, 2, figsize=(15, 5 *
((len(logbooks) + 1) // 2)))
    axes = axes.flatten()

    for i, logbook in enumerate(logbooks):
        gen = logbook.select("gen")
        fit_sum = [logbook.select("min")[gen_idx] for gen_idx in
range(len(gen))]
```

```
        axes[i].plot(gen, fit_sum, label=f'Summed Absolute Joint Error for
Joint {i+1}')
        axes[i].set_xlabel('Generation')
        axes[i].set_ylabel('Summed Absolute Joint Error')
        axes[i].set_title(f'Summed Absolute Joint Error over Generations for
Joint {i+1}')
        axes[i].legend()

    plt.tight_layout()
    plt.show()

# Функція запуску генетичного програмування
def run_gp(b):
    results = []
    e_total_array = []
    with output:
        output.clear_output()
        ngen = ngen_widget.value
        population_size = population_size_widget.value
        cxpb = cxpb_widget.value
        mutpb = mutpb_widget.value

        cx_operator = cx_operator_widget.value
        mut_operator = mut_operator_widget.value

        # Вибір і реєстрація оператора схрещування
        if cx_operator == 'cxOnePoint':
            toolbox.register("mate", gp.cxOnePoint)
        elif cx_operator == 'cxOnePointLeafBiased':
            toolbox.register("mate", gp.cxOnePointLeafBiased, termpb=0.1)
        elif cx_operator == 'cxSemantic':
            toolbox.register("mate", gp.cxSemantic, pset=pset, min=0, max=2)

        # Вибір і реєстрація оператора мутації
        if mut_operator == 'mutShrink':
            toolbox.register("mutate", gp.mutShrink)
        elif mut_operator == 'mutEphemeral':
            toolbox.register("mutate", gp.mutEphemeral, mode='one')  # або
'all'
        elif mut_operator == 'mutNodeReplacement':
            toolbox.register("mutate", gp.mutNodeReplacement, pset=pset)
        elif mut_operator == 'mutUniform':
            toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr,
pset=pset)
        elif mut_operator == 'mutInsert':
            toolbox.register("mutate", gp.mutInsert, pset=pset)

        toolbox.decorate("mate",
gp.staticLimit(key=operator.attrgetter("height"), max_value=3))  # Обмеження
глибини дерева до 3
        toolbox.decorate("mutate",
gp.staticLimit(key=operator.attrgetter("height"), max_value=3))  # Обмеження
глибини дерева до 3

        theta_angles = rotate_matrix_clockwise(angles_array)
        theta_angles_prim = rotate_matrix_clockwise(angles_array_prim)

        logbooks = []

        for i in range(len(theta_angles)):
```

```
            best_expr_theta, logbook =
symbolic_regression(theta_angles_prim[i], theta_angles[i], ngen=ngen,
population_size=population_size, cxpb=cxpb, mutpb=mutpb)
            results.append(best_expr_theta)
            logbooks.append(logbook)

        for i in range(len(angles_array_prim)):
            calculated_angles = []
            for j in range(len(angles_array_prim[i])):
                func = toolbox.compile(expr=results[j])
                calculated_angles.append(func(angles_array_prim[i][j]))

            predictions =
robot_chain.forward_kinematics(calculated_angles)[:3, 3]
            print(f'predictions = ({predictions[0]}, {predictions[1]},
{predictions[2]})')
            print(f'actual x = {coords_array[i][0]}, y = {coords_array[i][1]},
z = {coords_array[i][2]}')
            e_x = coords_array[i][0] - predictions[0]
            e_y = coords_array[i][1] - predictions[1]
            e_z = coords_array[i][2] - predictions[2]

            print(f'x = {e_x}, y = {e_y}, z = {e_z}')

            # Загальна помилка
            e_total = math.sqrt(e_x**2 + e_y**2 + e_z**2)
            e_total_array.append(e_total)

            # Відносна помилка у відсотках
            relative_error = (e_total / math.sqrt(coords_array[i][0]**2 +
coords_array[i][1]**2 + coords_array[i][2]**2)) * 100
            print(f'Relative error: {relative_error:.2f}%')

        print(e_total_array)
        print(math.fsum(e_total_array) / len(e_total_array))

        for i in range(len(results)):
            print(f'Best expression for theta_{i} -> theta_{i}_prime:
{results[i]}')

        # Додаткові графіки
        generate_additional_plots(e_total_array, logbooks)

run_button.on_click(run_gp)
file_upload.observe(load_file, names='value')

# Відображення віджетів
params_widgets = widgets.VBox([file_upload, ngen_widget,
population_size_widget, cxpb_widget, mutpb_widget, cx_operator_widget,
mut_operator_widget, run_button])
display(params_widgets, output)
```