Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

Розроблення системи прийняття рішень для керування автоматизованою роботизованою мобільною платформою
(тема)

Виконав:
студент _2_ курсу, групи КТРСм-21-1
Сосницький Максим Анатолійович
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)
Керівник доц. Хрустальов К. Л.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«02» грудня 2022 р. Сосницький М. А.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики та комп'ютеризованих технологій
Кафедра КІТАМ
Рівень вищої освіти Другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_»_____20_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Сосницькому Максиму Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи прийняття рішень для керування автоматизованою роботизованою мобільною платформою
Затверджена наказом по університету від 25.11.2022 р., № 1517 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 29.12.2022
3. Вихідні дані до роботи Система прийняття рішень,
машинне навчання,
мова Python,
середовище моделювання MatLab

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз систем штучного інтелекту та прийняття рішень роботизованої мобільної платформи
4.3 Вибір основних елементів системи керування роботизованої мобільної платформи
4.4 Розроблення програмного засобу для прийняття рішень та керування роботом
4.5 Моделювання та експериментальні дослідження розробленого програмного засобу у середовищі MatLab
4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій слайди у форматі Power Point у кількості 11 слайдів з розширенням .pptx 6 с., формату А4

6. Консультанти розділів роботи

Найменування Розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Аналіз технічного завдання	26.09.22	виконано
2	Опрацювання літератури за темою роботи.	05.10.22	виконано
3	Аналіз систем штучного інтелекту та прийняття рішень роботизованої мобільної платформи	20.11.22	виконано
4	Вибір основних елементів системи керування роботизованої мобільної платформи	27.11.22	виконано
5	Розроблення програмного засобу для прийняття рішень та керування роботом	5.12.22	виконано
6	Моделювання та експериментальні дослідження розробленого програмного засобу у середовищі MatLab	10.12.22	виконано
7	Оформлення пояснювальної записки	15.12.22	виконано
8	Оформлення презентації	20.12.22	виконано
9	Подання роботи на рецензію	27.12.22	виконано
10	Подання роботи на підпис зав. Кафедри	28.12.22	виконано
11	Подання кваліфікаційної роботи в ЕК	29.12.22	

Дата видачі завдання 26.09.2022 р.

Студент _____ Сосницький М.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Хрустальов К. Л.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 102 с., 13 табл., 25 рис., 3 дод., 21 джерело.

ПРИЙНЯТТЯ РІШЕНЬ, СИСТЕМА, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, Q-НАВЧАННЯ, PYTHON, MATLAB.

Об’єкт дослідження – система прийняття рішень.

Предмет дослідження – система прийняття рішень для керування мобільною роботизованою платформою.

Мета кваліфікаційної роботи магістра – розроблення системи прийняття рішень для керування автоматизованою роботизованою мобільною платформою з використанням методу Q-навчання.

Методи дослідження – методи теорії штучного інтелекту, методи навчання з підкріпленням, теорія алгоритмів та теорія прийняття рішення.

У кваліфікаційній роботі досліджено використання методів Q-навчання для системи прийняття рішень автоматизованої роботизованої платформи, а саме її переміщення у невизначеному просторі. Для прикладу такого простору було взято спіральний лабіринт.

Практична частина реалізована за допомогою мови Python та у середовищі MatLab було проведено експериментальні дослідження з імітаційною моделлю, що використовує у своєму складі принципи прийняття рішень на основі Q-навчання.

ABSTRACT

Explanatory note: 102 p., 13 tabl., 25 fig., 3 app., 21 sources.

DECISION-MAKING, SYSTEM, REINFORCEMENT LEARNING, Q-LEARNING, PYTHON, MATLAB.

The object of research is the decision-making system.

The subject of research is a decision-making system for controlling a mobile robotic platform.

The purpose of the master's qualification work is to develop a decision-making system for controlling an automated robotic mobile platform using the Q-learning method.

Research methods – artificial intelligence theory methods, reinforcement learning methods, algorithm theory and decision theory.

In the qualifying work, the use of Q-learning methods for the decision-making system of an automated robotic platform, namely its movement in an indefinite space, was investigated. A spiral labyrinth was taken as an example of such a space.

The practical part was implemented using the Python language and experimental studies were carried out in the MatLab environment with a simulation model that uses the principles of decision-making based on Q-learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ДП – динамічне програмування;

МК – метод Монте-Карло;

МН – машинне навчання;

НП – навчання з підкріпленням;

ПЗ – програмне забезпечення;

ППР – послідовне прийняття рішень;

ПР – прийняття рішень;

СОП – систем, що обслуговують продукти;

СППР – системи підтримки прийняття рішень;

ШІ – штучний інтелект;

SARSA – State-Action-Reward-State-Action, Стан-Дія-Нагорода-Стан-Дія.

ВСТУП

Прийняття рішень – це процес вибору шляхом визначення рішення, збирання інформації та оцінки альтернативних рішень. Використання послідовного процесу прийняття рішень може допомогти приймати більш обдумані рішення, систематизуючи відповідну інформацію та визначаючи альтернативи. Послідовне прийняття рішень (ППР) описує ситуацію, коли особа, яка приймає рішення (наприклад, мобільний роботизована платформа), послідовно спостерігає за процесом, перш ніж буде ухвалено остаточне рішення. Метою послідовного прийняття рішень є знайти правило зупинки, яке оптимізує рішення з погляду мінімізації втрат або максимізації вигоди, включаючи витрати на спостереження. Правило оптимальної зупинки також називають оптимальною стратегією та оптимальною політикою. Все написане вище і є частиною навчання з підкріпленням, частиною якого є Q-навчання.

Q-навчання – це метод вирішення завдань навчання із підкріпленням. Проблеми навчання з підкріпленням вимагають покращення поведінки на основі отриманих винагород. Він може зменшити складності в програмуванні роботів та збільшити здібності роботів діяти в будь-якому середовищі. Однак більшість сучасних систем Q-навчання не повністю застосовано у завданнях робототехніки. Виникають певні проблеми такі як: вони розглядають безперервні змінні, наприклад, швидкість або положення, як дискретні значення. Дискретність не забезпечує плавного управління та не повністю використовує сприйняту інформацію. Використання на практиці також повинно долати обмеження в реальному часі (затримки під час виконання дій), а також неправильні дані з датчиків.

Це дослідження описує алгоритм, який має справу з безперервним станом та дією – переміщення робота в невизначеному робочому просторі з використанням елементів прийняття рішень на основі Q-навчання.

У роботі використано здатність Q-навчання вчитися пасивно – при спостереженні за діями робота, а не обов'язково лише за його управлінням, що

приведе до скорочення тривалості навчальних експериментів, що й свідчить про актуальність дослідження за даною тематикою.

Отже, метою кваліфікаційної роботи магістра є розроблення системи прийняття рішень для керування автоматизованою роботизованою мобільною платформою з використанням методу Q-навчання.

Об’єкт дослідження – система прийняття рішень.

Предмет дослідження – система прийняття рішень для керування мобільною роботизованою платформою.

Методи дослідження – методи теорії штучного інтелекту, методи навчання з підкріпленням, теорія алгоритмів та теорія прийняття рішення.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз систем штучного інтелекту та прийняття рішень, що можуть бути використані при керуванні роботизованою мобільною платформою;

– провести вибір основних елементів системи керування роботизованою мобільною платформою;

– розробити програмний засіб для прийняття рішень та керування роботизованою мобільною платформою мовою Python;

– провести моделювання та експериментальні дослідження з розробленим програмним засобом для прийняття рішень та керування роботизованою мобільною платформою у середовищі MatLab;

– оформити пояснювальну записку згідно з [1] та [2].

1 АНАЛІЗ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПРИЙНЯТТЯ РІШЕНЬ РОБОТИЗОВАНОЇ МОБІЛЬНОЇ ПЛАТФОРМИ

1.1 Штучний інтелект у сучасному світі

1.1.1 Поняття штучного інтелекту

Штучний інтелект (ШІ) є областю обчислювальної техніки, в якій основна увага приділяється передачі інтелекту та мислення людини машинам, які можуть у багатьох відношеннях допомагати людям. ШІ повільно виник і став сильнішим у багатьох областях, таких як інженерія, математика, фізика, технології, і все це призвело до нинішнього величезного зрушення в цій галузі, яке ми спостерігаємо зараз [1].

Основною ідеєю ШІ є те, що машини можуть набувати інтелекту. Він охоплює таку галузь, як машини, що можуть навчатися самостійно, адаптуватися до конкретних обставин та самостійно виправляти свої помилки. Тобто машини можуть думати самостійно, не кодуючись командами.

Як правило, ШІ аналізує параметри середовища та вживає необхідних дій, щоб підвищити ймовірність успішного вирішення завдання. Цільові та корисні функції ШІ можуть бути простими для конкретної мети або складними під час виконання. В ньому можуть бути цілі, що включені явно чи неявно. Якщо мета встановлена для навчання з підкріпленням, то її можна встановити неявно, призначаючи заохочувальні бали або караючи інші фактори.

Так само еволюційна система може бути розроблена з цілями та функціями для досягнення цілей та відтворення системи ШІ для моделі, заснованої на задачі пошуку їжі тваринами.

Навпаки, система з ШІ, така як найближчий сусід, немає цілей чи працює з розумом і набором навчальних даних. У дослідницькому співтоваристві набирає популярності розробка бенчмарку для такої безцільової системи, завдання якої полягає у досягненні класифікації проблем. ШІ виконується на основі алгоритмів.

Алгоритм системи – це набір інструкцій, яким слідує машина. Розробляються складні алгоритми, за якими слідує простий алгоритм розв'язання задач.

Алгоритми на основі ШІ здатні обробляти дані та навчатися на них. Вони можуть вирішувати проблеми, вивчаючи нові речі, або самостійно розробити алгоритм. Деяка методологія розгортання за допомогою дерева рішень, байєсівських мереж та найближчого сусіда може теоретично навчитися апроксимувати математичні функції, які розроблять процес вирішення проблеми. Вони можуть отримати необхідні знання за допомогою всіх можливих гіпотез та зіставлення даних. Практично через явище комбінаторного вибуху неможливо оцінити всі можливості розв'язання задач, де час розв'язання задач зростає експоненційно. Дослідження ШІ включає пошук можливого рішення з широкого спектра можливостей, уникаючи при цьому можливостей, які не приносять користі для вирішення проблеми.

1.1.2 Основні напрямки використання ШІ

Існує безліч додатків, де використовується ШІ. Деякі з них обговорюються нижче та представлено на рис. 1.1.

ШІ для астрономії. Можна використовувати для вирішення універсальних завдань. Інструменти та методи ШІ можуть допомогти зрозуміти принципи роботи та походження Всесвіту.

ШІ для охорони здоров'я. В останні десятиліття системи та інструменти на основі ІІ все частіше використовуються в галузях охорони здоров'я. Він має намір створити значний сервіс при наданні медичних послуг. Алгоритми системи на основі штучного інтелекту можуть забезпечити якіснішу діагностику, ніж людина. Це може допомогти лікарям зрозуміти критичні випадки пацієнтів та дати їм сигнал про погіршення стану екстреної медичної допомоги.

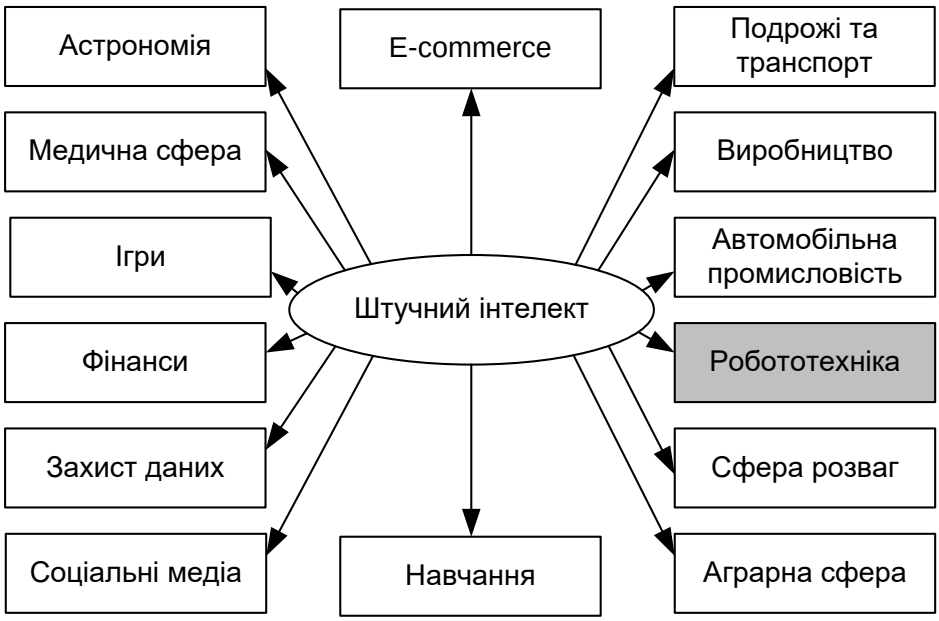


Рисунок 1.1 – Сфери застосування ШІ

ШІ для ігор. Система на його основі може брати участь в іграх. Наприклад, для шахової гри може розробити алгоритм пошуку різних варіантів кроків для певного кроку іншого гравця.

ШІ для фінансової сфери. Водночас ШІ та фінансові установи можуть надавати якісніші послуги при досягненні фінансових цілей. Розгортання алгоритмів для чат-бота, торгівлі, автоматизації та машинного навчання може відіграти життєво важливу роль.

ШІ захисту даних. Забезпечення безпеки даних стає важливим компонентом для різних галузей промисловості по всьому світу. Використання алгоритму штучного інтелекту визначення помилок у обробці програмного забезпечення та виявлення кібератак стає дедалі популярнішим серед бізнес-підрозділів.

ШІ для соціальних медіа/мереж. Сайти соціальних мереж містять безліч профілів користувачів та продуктів. Організація такого величезного набору даних є великою проблемою для людства. ШІ може керувати даними та впорядковувати їх відповідно до останніх тенденцій та вимог ринку.

ШІ для подорожей та транспорту. В індустрії подорожей та транспорту стає все більш популярним використання ШІ. Системи на його основі здатні

керувати замовленнями, пропонувати готелі та рейси, знаходити найкращий маршрут для користувачів. Бізнес-підрозділи розгортають чат-бота на основі штучного інтелекту для кращого інтерактивного спілкування зі своїми клієнтами.

ШІ для автомобільної промисловості. Багато відомих галузей використовують віртуального помічника для допомоги користувачам. TeslaBot від компанії Tesla надає користувачам послуги помічника в режимі реального часу. Багато організацій займаються розробкою автомобілів з автопілотом, які забезпечують безпечнішу та безпечнішу подорож, ніж традиційне водіння.

ШІ для робототехніки. За допомогою ШІ робототехніка може вирішити задачу, використовуючи свій минулий досвід. Проте традиційні звичайні роботи виконують завдання, що повторюються. Але інтеграція ШІ може покращити розумові здібності таких роботів. Розгортання алгоритму ШІ для гуманоїдів є ще одним прикладом обслуговування. Людиноподібний робот на ім'я Софія та Еріка може поводитися і говорити як людина.

ШІ для розваг. У сфері розваг алгоритми машинного навчання та ШІ можуть надати споживачеві якісніші послуги. У таких додатках, як NetFlix, Amazon Prime, такі алгоритми доставляють програми, що рекомендуються, на основі полів пошуку користувачів.

ШІ для аграрної сфери (сільського господарства). Традиційні методи сільського господарства потребують різних ресурсів, таких як гроші, робоча сила та час для кращого вирощування. Однак розгортання інструментів ШІ може покращити прогностичний аналіз фермера та механізм моніторингу врожаю.

ШІ для електронної комерції. У галузевих підрозділах електронної комерції інструменти на основі ШІ можуть забезпечити краще поєднання продуктів з розміром, кольором та брендом. Рецензенти коментують та аналізують, що може передбачити придатність продукту для конкретного користувача через Інтернет.

ШІ для освіти. Розгортання алгоритму ШІ для розробки чат-бота може покращити помічника викладача для учнів. У майбутньому він може працювати як особистий репетитор, який буде доступним для доступу в будь-який час [3].

1.1.3 Штучний інтелект, машинне навчання та прийняття рішень

Штучний інтелект та машинне навчання – це частини інформатики, які корелюють одна з одною. Ці дві технології є трендовими технологіями, які використовуються для створення інтелектуальних систем.

Хоча це дві пов'язані технології, і іноді люди використовують їх як синоніми один для одного, проте, у різних випадках це два різні терміни [4].

Штучний інтелект – це область інформатики, яка створює комп'ютерну систему, здатну імітувати людський інтелект.

Система штучного інтелекту не вимагає попереднього програмування, натомість вони використовують такі алгоритми, які можуть працювати з їхнім власним інтелектом. Він включає алгоритми машинного навчання, такі як алгоритм навчання з підкріпленням і нейронні мережі глибокого навчання. ІІ використовується в багатьох місцях, таких як Siri, Google AlphaGo, ШІ у грі в шахи тощо.

Машинне навчання (МН) – це область штучного інтелекту, яка дозволяє машинам навчатися на основі минулих даних або досвіду без програмування.

МН дозволяє комп'ютерній системі робити прогнози чи приймати певні рішення, використовуючи історичні дані, без явного програмування. Машинне навчання використовує величезну кількість структурованих та напівструктурованих даних, тому модель машинного навчання може генерувати точні результати або прогнозувати на основі цих даних.

МН працює за алгоритмом, що навчається самостійно, використовуючи попередньо отримані дані. Працює тільки для певних доменів, наприклад, якщо ми створюємо модель машинного навчання для виявлення зображень роботів, він дасть результат тільки для зображень роботів, але якщо ми дамо нові дані, такі як зображення верстату, МН перестане відповідати. Машинне навчання

використовується в різних місцях, таких як система онлайн-рекомендацій, алгоритми пошуку Google, фільтр спаму в електронній пошті, пропозиція позначки друзів Facebook Auto, тощо.

Його можна розділити на три види:

– контрольоване навчання;

– навчання з підкріпленням;

– неконтрольоване навчання.

Прийняття рішень за допомогою алгоритмів, розроблених за допомогою машинного навчання, все більше визначає наше життя. На жаль, повна непрозорість процесу є нормою.

Класифікація основних видів машинного навчання представлено на рис. 1.2 [5].

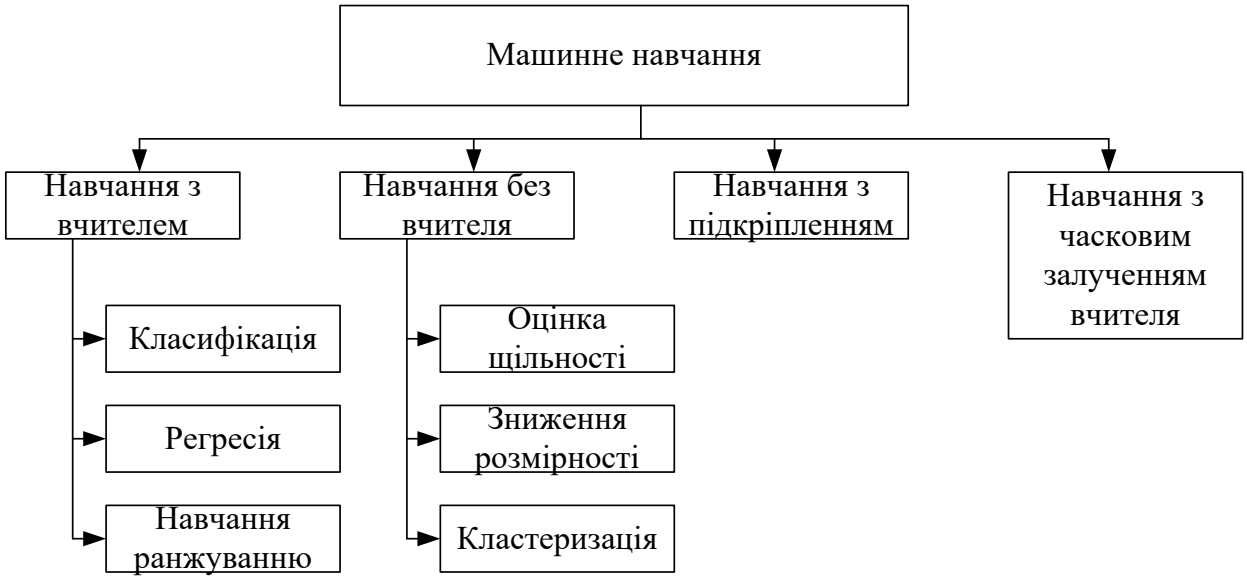


Рисунок 1.2 – Класифікація основних видів машинного навчання

Основне завдання навчання з учителем полягає в тому, щоб з даних витягти модель, яка дозволяє робити такі прогнози, що раніше не зустрічалися або на основі майбутніх даних. Під учителем розуміється або сама навчальна вибірка, або той, хто вказав на заданих об'єктах правильні відповіді.

Навчання без вчителя часто протиставляється навчанню з вчителем, коли для кожного навчального об'єкта задається так звана правильна відповідь, і потрібно знайти залежність між об'єктами та відповідями.

Завдання навчання з підкріпленням полягає у виробленні системи (агенту), яка покращує свою якість на основі взаємодій із середовищем. Оскільки інформація про поточний стан середовища, як правило, містить і так званий сигнал винагороди, навчання з підкріпленням можна подати як область, яка має відношення до навчання з учителем.

Навчання з підкріпленням – це навчання того, що треба робити, як слід відображати ситуації в дії, щоб максимізувати деякий сигнал заохочення (винагороди), що набуває числові значення.

Агенту не кажуть, яку дію слід зробити, як це має місце у більшості підходів до навчання машин. Натомість він, пробуючи виконувати різні дії, повинен знайти, які з них принесуть йому найбільшу винагороду. У найбільш цікавих і важливих випадках дії можуть впливати не тільки на винагороду, що отримується негайно, але також і на ситуацію, що виникає, а через неї на всі наступні заохочення.

З написаного вище і виникає необхідність використання систем прийняття рішень для агенту у невідомому середовищі, на динамічному виробництві, тощо.

1.2 Системи прийняття рішень при керуванні роботами та на виробництві

1.2.1 Системи прийняття рішень та Індустрія 4.0

Щоб відповідати більш складним очікуванням клієнтів, багато компаній повинні підвищувати ефективність і гнучкість. У цьому сенсі технології Індустрії 4.0 пропонують значні можливості для покращення як операційних, так і процесів прийняття рішень. Ці розробки дозволяють розглядати підвищення рівня операційних систем та автономності команд. Однак можливості посилення процесу ухвалення рішень за допомогою цих нових технологій у поточній літературі залишаються неясними.

Хмарні обчислення є основою для покращення всього процесу прийняття рішень. Однак деякі технології, такі як Інтернет речей та моделювання, мають великий потенціал лише для певних кроків у процесі прийняття рішень. Це дослідження також дає перше уявлення про перспективи, очікування та ризики менеджера, пов'язані з впровадженням нових способів прийняття рішень та автономії, що підтримуються технологіями Індустрії 4.0 [6].

Література з систем, що обслуговують продукти (СОП), Індустрії 4.0, обслуговування та прийняття рішень показує, що ці теми тісно пов'язані, якщо аналізувати їх з точки зору обслуговування. Перехід до економічно стійких пропозицій СОП вимагає визначення процесів, здатних виконувати запити клієнтів і водночас керувати ресурсами компанії, щоб бути ефективними та уникати економічних втрат.

Щоб підвищити ефективність надання послуг, компанії повинні переосмислити свої внутрішні процеси та керувати ними протягом усього життєвого циклу СОП, спираючись на сильні сторони та усуваючи слабкі сторони (наприклад, використовуючи мережі для спільної роботи).

Індустрія 4.0 робить значний внесок у поліпшення надання послуг, і діджиталізація послуг набуватиме все більшого значення в найближчі роки. Дійсно, можливість збору та обробки даних, що генеруються активом на етапі його середнього життєвого циклу, сприяє отриманню нових знань про об’єкти (агенти), що працюють у робочому просторі; моніторингу стан працездатності активів, найкраще планування робіт з технічного обслуговування для запобігання збоям та простоям та підвищення стійкості роботи системи.

У цьому сенсі обробка даних стає діяльністю з доданою вартістю для компанії лише тоді, коли дані перетворюються на корисні знання. У будь-якому випадку обробка даних не є тривіальною діяльністю і потребує структурованих процесів збирання, управління та аналізу даних. Для цього важливо встановити чітко визначені цілі та стратегії, а також правильно їх переслідувати.

Помилки під час збору даних впливають на подальший аналіз і можуть призвести до неправильних рішень, що впливають на результати технічного обслуговування.

Окрім проблем зі збором даних, готовність клієнтів ділитися операційними даними з постачальником послуг з технічного обслуговування також стає критично важливою для отримання нових знань та цінності для заінтересованих сторін.

Наявність системи підтримки прийняття рішень (СППР) у виробничому середовищі має основне значення підтримки операцій і дій протягом усього життєвого циклу продукту. Незважаючи на це, у виробничих компаніях рішення переважно приймаються людьми на основі їхнього досвіду, а не польових даних [7].

Люди, що керують процесом, зараз перебувають під більшим тиском, ніж у минулому, наприклад у минулому столітті. Це з зміною умов ведення бізнесу на світовому ринку. Вони зазнають тиску навколишнього середовища; цей тиск викликаний зовнішніми факторами, що постійно змінюються, впливають на всі підприємства та організації.

Глобалізація і технології та їх бурхливий розвиток є головними трендами цивілізації. В даний час конкуренція включає постійні ринкові зміни, при цьому частота цих змін неухильно зростає, прискорюються продуктові і технологічні інновації, сучасні системи управління, інновації в управлінні бізнесом тощо [8].

У такому складному середовищі людям, що керують процесом, доводиться приймати рішення. Можна сказати, що саме в такому середовищі вимоги і тиск на менеджерів для прийняття правильних рішень природно зростають. Опис дій у процесі прийняття рішень показано на рисунку 1.3.

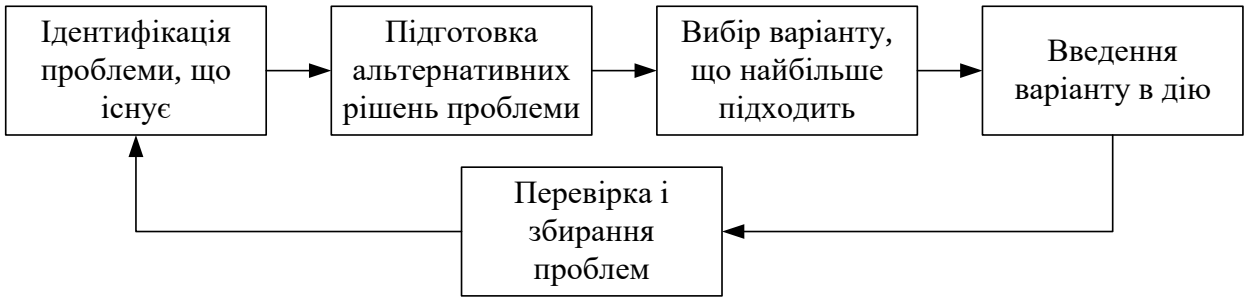


Рисунок 1.3 – Опис дій у процесі прийняття рішень

1.2.2 Прийняття рішень між людиною і роботом

У виробництві зростає бажання використовувати унікальні можливості людей та роботів для створення високоефективних команд людина-робот. Часто роботи не здатні виконувати ті ж завдання, що й їхні колеги-люди, і після появи робота працівники-люди перемикаються на виконання меншого набору завдань, що більш підходять для спритності та інтелекту людини працівника до свого оточення [9]. Належне функціонування команди з виробництва людино-роботів потребує суворої координації між роботою людини та робота, яка задовольняє жорстким тимчасовим та просторовим обмеженням. Як академічні дослідники, і галузеві практики розробили системи для планування чи планування роботи людини і робота, у яких люди або беруть участь у процесі прийняття рішень, або робота планується автономно.

Працівники-люди часто розвивають почуття ідентичності та безпеки завдяки своїй ролі або роботі на фабриці, і багато хто звикли мати певний ступінь автономії у прийнятті рішень. У результаті працівник-людина, перед яким стоїть завдання автоматизованого алгоритму планування, може почуватися знеціненим. Навіть якщо алгоритм спочатку підвищує ефективність процесу, відібрання контролю у людей може відштовхнути їх і, у свою чергу, зрештою завдають збитків загальної продуктивності. З іншого боку, працівники можуть вважати процес планування обтяжливим і віддати перевагу частині ефективної команди, а не брати участь у процесі планування, якщо збереження такої ролі знижує їх ефективність. У той час як автономні алгоритми планування можуть забезпечити майже оптимальні розклади протягом кількох секунд, ми також

хочемо визначити, які повноваження для прийняття рішень мають бути у людей у завданні.

Також існує зворотна залежність між ефективністю команди людина-робот та задоволеністю людей-працівників, залежно від того, наскільки люди-робітники мають контроль над розподілом завдань чи призначенням того, який працівник виконуватиме якесь завдання.

Автономні роботи повинні адаптувати свої рішення до довкілля. Вони використовують свої датчики та внутрішню пам'ять для аналізу та оновлення свого уявлення як про стан світу, так і про свій внутрішній стан. Щоб оцінити швидкість об'єкта, роботи зазвичай збирають дані датчика відстані. Спостерігаючи за зміною відстані, роботи зможуть оцінити швидкість об'єкта. Сприйняття навколишнього середовища роботами є складною проблемою, яка не розглядатиметься в даній дисертації. Однак важливо мати на увазі, що рішення, що приймаються роботами, засновані на частковій та зашумленій інформації.

1.2.3 Прийняття рішень при керуванні автономними мобільними роботами

Прийняття рішень (ПР) – це процес вибору дії, яку робот має виконати з урахуванням його внутрішнього уявлення про стан світу. Цей процес може просто полягати у застосуванні існуючої політики, тобто функції, що відображає кожен можливий стан світу в дію, але він також може включати симуляції для прогнозування результатів можливих дій у поточній ситуації. В обох випадках метою процесу прийняття рішень є максимізація винагороди, яку отримує робот [10-11].

Є три важливі аспекти, які слід враховувати під час прийняття рішень у робототехніці.

По-перше, автономні роботи повинні використовувати свою обчислювальну потужність через обмеження за вагою, обсягом та споживанням енергії, тому прийняття рішень має бути обчислювально ефективним.

По-друге, прийняття рішень має бути швидким, щоб уникнути затримок, тобто. Автономний автомобіль повинен швидко реагувати, щоб уникнути зіткнення.

Зрештою, важливо пам'ятати, що моделі, що використовуються в робототехніці, можуть бути наближеними до реального світу. Зазвичай існує значний розрив між оцінками, що забезпечуються фізичною моделлю (теорією), та наслідками її дій у реальному світі (практикою). Ця невідповідність є особливо великою для недорогих роботів того типу, який нас цікавить. Отже, оптимальна дія відповідно до моделі та оптимальна дія в реальному світі – це дві різні речі.

Розробка складних та ефективних моделей поведінки роботів зазвичай потребує використання симуляторів для оцінки продуктивності без ризику пошкодження робота. Моделювання дозволяє:

– оптимізація функцій управління шляхом експериментування з різними політиками можливо паралельно;

– моніторинг значень, які неможливо точно виміряти на роботах.

Моделювання засноване на фізичних рівняннях та специфікаціях роботів.

Між реальністю та симулятором може бути значний розрив. Деякі фізичні аспекти, такі як шоколад, важко точно змоделювати, особливо для недорогих роботів, побудованих за допомогою недорогого обладнання. Реальний робот може трохи відрізнятися від моделі через механічні недосконалості та неповне моделювання (наприклад, вага електричних проводів рідко включається в модель роботів, і кожен серводвигун має різну передавальну функцію). Хоча можна скоротити розрив між моделлю та реальністю, найчастіше це має суттєве значення вартості, оскільки потребує більш точного виготовлення.

Сприйняття довкілля роботами залежить від шуму. Всі цифрові датчики створюють шум кількісного визначення через роздільну здатність датчиків. Датчики також надають інформацію із заданою частотою, тому в будь-яку годину неможливо отримати доступ до стану робота. Як наслідок, датчики забезпечують не точні виміри фізичного параметра, а скоріше розподіл ймовірності за можливими значеннями.

Роботи не можуть повністю і точно фіксувати свій внутрішній стан, ні стан довкілля. Їм може не вистачати інформації про коефіцієнт тертя підлоги, точне положення енкодера або приблизної точності їхнього становища в довкіллі.

Отже, застосування однієї й тієї ж дії в тому самому вимірюваному стані може призвести до різних результатів залежно від невимірюваних змінних.

Оскільки прийняття рішень в середовищах, що частично спостерігаються, особливо ускладнене, ми вважаємо за краще розглядати роботизовані завдання як повністю спостерігаються, але зі стохастичними переходами. Хоча цей тип абстракції не відкидає реальність, він дозволяє використовувати складні алгоритми і, як і раніше, призводить до задоволених результатів.

Хоча підвищення точності апаратного забезпечення робота спрощує прийняття рішень та призводить до покращення здатності виконувати більшість завдань, це також збільшує вартість робота. Щоб досягти задовільних результатів через низькі витрати, необхідно враховувати шум у процесі прийняття рішень. Крім того, моделювання шуму робота та використання навчання для вирішення проблеми також надає рекомендації щодо апаратних змін, автоматично визначаючи, які аспекти шуму мають найбільше вплив на продуктивність.

На додаток до всіх труднощів, згаданих раніше, онлайн-рішення при створенні автономних роботів необхідно враховувати конкретні обмеження щодо обчислювальної потужності. Автономні роботи повинні вбудовувати свою обчислювальну потужність із трьома основними обмеженнями: обсяг, вага та енергоспоживання.

Більше того, оскільки роботи діють у реальному світі, їх рішення мають прийматись у реальному часі. Тому прийняття рішень в автономній робототехніці часто потребує зосередження уваги на наданні прийнятних стратегій за низькі обчислювальні витрати.

1.3 Навчання з підкріпленням та його використання

1.3.1 Основні поняття навчання з підкріпленням

Навчання з підкріпленням (НП) – це тип навчання, орієнтований на конкретну мету. Агент навчається, взаємодіючи з невідомим середовищем, зазвичай методом спроб і помилок. Це найпоширеніший спосіб навчання будь-якого агенту, який щось робить та спостерігає за тим, що відбувається. Він отримує зворотний зв'язок у вигляді винагороди (або покарання) від довкілля; потім він використовує цей зворотний зв'язок для самонавчання та накопичення досвіду та знань про навколишнє середовище (процес прийняття рішень).

Проблеми навчання з підкріпленням пов'язані з вивченням того, яку дію краще виконувати в залежності від ситуації, щоб максимізувати сукупну винагороду. Агент НП повинен вивчити політику (тобто повне зіставлення між ситуаціями та діями), пробуючи дії без будь-яких вказівок експерта у предметній галузі, як у багатьох інших формах машинного навчання [11].

Ще одна важлива характеристика проблеми НП полягає в тому, що в будь-якій ситуації агент повинен вибирати між використанням своїх поточних знань про навколишнє середовище (виконання дії, яка вже була випробувана раніше в цій ситуації) або вивченням дій, які ніколи раніше не робилися в цій ситуації.

Люди завжди намагаються покращити свою взаємодію з навколишнім середовищем, ґрунтуючись на своєму попередньому досвіді. Штучний агент у налаштуванні НП намагається поводитися так само.

Виходячи з концепції стану, дії, середовища, винагороди та політики, можна визначити мету агенту НП – знайти оптимальну політику для заданого набору станів, щоб максимізувати довгострокову винагороду. Загальний сценарій роботи НП показано на рис. 1.4. НП, що є найбільш цільовою гілкою МН, також включає відкладені винагороди в дуже складному налаштуванні відкладеної відповіді, де важко визначити, яка дія була корисною протягом великої кількості часових інтервалів.

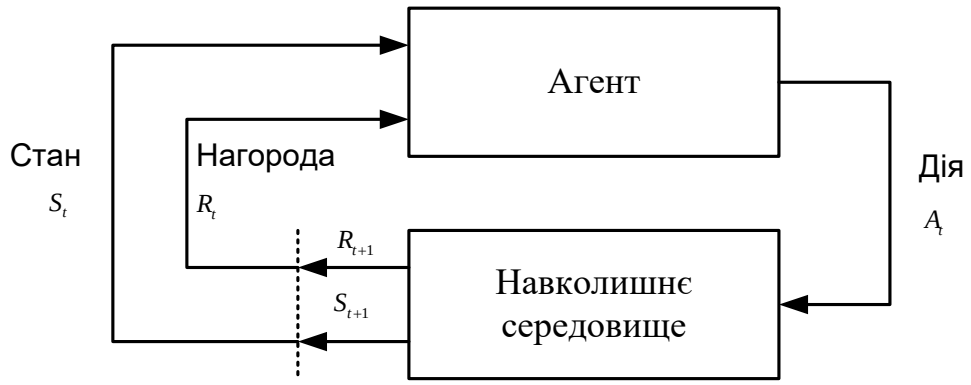


Рисунок 1.4 – Загальний сценарій роботи НП

Найпростіший випадок, який можна вирішити за допомогою НП – це завдання з так званими багаторукими бандитами, коли кожна дія, доступна агенту, має деяку нагороду, засновану на розподілі ймовірностей.

Наприклад, кілька однооруких бандитів (ігрові автомати) або один великий автомат з безліччю слотів, де мета полягає в тому, щоб отримати максимально бажану міру винагороди за певний період часу, граючи на ігрових автоматах (або бандитах), які мають невідомі та різні очікувані результати. Так само агент НП також намагається отримати максимально можливу нагороду за безліч епізодів.

Тонкий баланс між експлуатацією та дослідженням – одна з основних проблем у НП, якої немає в інших алгоритмах машинного навчання. У режимі експлуатації агент робить найкращу дію з уже відомих знань, у той час як в режимі дослідження агент може зробити стохастичну дію, щоб збільшити свою інформацію, щоб отримати більше винагороди. Наприклад, у задачі про ігровий автомат ви можете спробувати знайти нових бандитів при заданому розподілі ймовірностей, перш ніж завершити найкращі дії. Але в той же час цей пошук найкращого варіанта може відволікти вас від максимізації сукупної винагороди. Так само при експлуатації ви вибираєте звичайний шлях для переміщення з одного місця до іншого, але вам потрібно спробувати інші можливі шляхи для дослідження.

Функція значення стану та функція корисності, визначені раніше, однакові, і обидві використовуються значення стану. Цей термін називається функцією значення стану або просто функцією значення або функція корисності. Функція

цінності стану представлена $V(s)$, а згідно з політикою π представлена $V^{\pi}(s)$, тобто результатом, коли починають в стані s і після цього слідуєте політиці:

$$V^{\pi}(s) = (R_t|s_t). \tag{1.1}$$

Поки $Q^{\pi}(s,a)$, також відома як функція значення дії (ФЗД), задає значення при запуску зі стану s шляхом виконання дії a і після цього π і називається Q-значенням.

$$Q^{\pi}(s,a) = (R_t|s_t,a_t). \tag{1.2}$$

Для одного і того ж середовища функція значення може змінитися із зміною політики. Ключ до вирішення проблем НП, а також мета агенту НП буде полягати в тому, щоб знайти ці функції значень для конкретної проблеми. Рівняння Беллмана для функцій цінності стану та стану-дії наводиться далі відповідно.

$$V^{\pi}(s) = R(s) + \sum(\gamma V^{\pi}(s^{'})), \tag{1.3}$$

$$Q^{\pi}(s,a) = R(s) + \sum(\gamma Q^{\pi}(s^{'},a^{'})). \tag{1.4}$$

1.3.2 Алгоритми навчання з підкріпленням

Будь-який з алгоритмів НП може бути заснований на політиці чи цінності, чи комбінації того й іншого, тобто. на методі актор-критик. Алгоритми НП можуть бути класифіковані як методи без моделей, наприклад, Q-навчання та алгоритми на основі моделей, такі як динамічне програмування, моделі переходу та функція повернення. У алгоритмі з урахуванням моделі агент покладається на випробування, а використовує вже вивчену модель. У НП на основі моделей агент може робити прогнози про різні стани та відповідні винагороди після навчання.

Методи без моделей залежать від спроб і помилок, щоб оновити свій досвід та знання про дане середовище, оскільки у них немає знань про модель переходу та функцію винагороди. Вони повинні вивчати динаміку системи, взаємодіючи з довкіллям багато разів [12].

Динамічне програмування (ДП), представлене Річардом Беллманом, є математично надійним методом вирішення завдань оптимізації. Двома властивостями ДП є оптимальна підструктура і підзавдання, що перекриваються – це задовольняє цим двом властивостям. Складна проблема може бути послідовно розділена на простіші і дрібніші підзавдання. Як і людям, машині легко вчитися поетапно. Таким чином, ДП шукає всі можливі рішення та вибирає найкраще для кожного обчислення. Так само метою НП є пошук оптимальних політик чи дій, щоб агент діяв оптимально.

Алгоритм ДП, заснований на моделі, вимагає повних знань, що спостерігаються (модель переходу і функція винагороди). Таким чином, у деяких проблемах НП, де методи ДП (ітерація значення або ітерація політики) можуть використовуватися для отримання функції або політики оптимального значення. Іншими словами, ДП може застосовуватися для вирішення або завдання управління, або завдання прогнозування. Далі подано алгоритми ітерації політик та значень.

Іншими словами, ДП може застосовуватися для вирішення або завдання управління, або завдання прогнозування.

Метод Монте-Карло (МК) використовує випадковість на вирішення проблеми. МК – це метод без моделі, який навчається на повному епізоді (без початкового завантаження), використовуючи ідею середньої віддачі. Підхід МК можна розділити на МК під час першого відвідування та МК під час кожного відвідування. Перший являє собою середнє значення прибутковості, після першого відвідування стану за період епізодів, тоді як МК при кожному відвідуванні середнє значення засноване на всіх відвідуваннях стану.

Ми можемо використовувати метод МК для прогнозування та оцінки управління, і ми говоримо, що це пасивний та активний НП відповідно. У

пасивному НП мета агенту у тому, щоб вивчити функцію значення з допомогою політики, тоді як у активному НП агент намагається знайти оптимальну політику при взаємодії із середовищем. Переваги методів МК перед методом ДП полягають у наступному:

– може застосовуватись із зразками моделей (симуляціями);

– методи МК прості в реалізації та підходять для середовищ з невеликими підмножинами станів;

– МК знаходить оптимальні рішення через пряму взаємодію зі світом.

У завдання управління МК важливо вибирати початкові стани випадковим чином поряд з діями, що мають ненульову ймовірність. Хоча підходи МК можуть досягати оптимальної політики без знання моделі переходу, функції винагороди та заданої політики, але методи МК повинні чекати на завершення епізоду, щоб оновити функцію цінності. Це обмеження методу МК і рішенням цього обмеження є методи різниці в часі (РЧ), в яких оновлення може бути виконане після одного кроку.

Як правило, методи РЧ використовується для прогнозування кількості, яка залежить від наступних значень, але в задачі НП РЧ застосовується для прогнозування сукупної винагороди. РЧ ідентичний ДП, оскільки оновлюється на основі поточної оцінки та ідентичний методу МК під час вибірки [13]. Загальне правило методу РЧ наведено нижче:

$$\textit{Нова оцінка} \leftarrow \textit{Стара оцінка} + \alpha[\textit{Ціль} - \textit{Стара оцінка}]. \quad (1.5)$$

У правій частині рівняння (1.5) різницева частина є помилкою оцінки δ, а цільове значення являє собою реальне значення. Ціль полягає в тому, щоб зменшити δ. Символ α відомий як швидкість навчання або розмір кроку між нулем та одиницею. Агент НП враховує нещодавню винагороду для $\alpha = 1$ і нічого не дізнається, коли $\alpha = 0$.

Q-навчання та SARSA є найпопулярнішими методами РЧ, і представлені далі.

State-Action-Reward-State-Action (SARSA) – Стан-Дія-Нагорода-Стан-Дія – є модифікація Q-навчання. SARSA – це метод онлайн-навчання, при якому агент взаємодіє з даному середовищу та виконує оновлення політики на основі дії вибрано. Функція значення дії Q оновлюється помилка та коригування проводяться по α (швидкість навчання), як дано нижче.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \quad (1.6)$$

Псевдокод SARSA дано в алгоритмі 1, в якому агент першого кроку виконує дію та переміщається на крок уперед і спостерігає винагороду на другому кроці разом із новим станом та дією. Потім на третьому етапі SARSA оновлює функцію Q, але в останньому етапі політика оновлюється при кожному відвідуванні шляхом виконання дії з максимальним значенням Q.

Алгоритм псевдокоду SARSA має наступний вигляд (рис. 1.5).

1. Ініціалізація
Q обирається довільним чином
Q (термінальне) = 0
Повторити
    ініціалізація s
    обрати $a \in \varepsilon$-жадібне
    Повторити
        виконати дію a , спостерігати r, s'
        обрати $a' \in \varepsilon$-жадібне
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
        $s \leftarrow s'$
        $a \leftarrow a'$
    s є термінальним
до конвергенції

Рисунок 1.5 – Алгоритм псевдокоду SARSA

Конвергенція SARSA може бути прискорена за рахунок використання трас прийнятності для пар стан-дія, виконуваних для станів в РЧ(λ). Сліди прийнятності оновлюються в SARSA(λ), як показано нижче:

$$e_t(s,a) = \gamma\lambda et - 1(s,a) + 1, \textit{ якщо } s = s_t \textit{ і } a = a_t,$$

інакше,

$$e_t(s,a) = \gamma\lambda et - 1(s,a). \quad (1.7)$$

Отримане правило оновлення для алгоритму SARSA за допомогою трасування є:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \textit{ для всіх } s \in S. \quad (1.8)$$

Збіжність у SARSA гарантується, коли всі пари (стан-дія) спостерігаються протягом нескінченного часу. Щоб розглянути всі стани та дії, ми можемо використовувати ε -жадібну політику, яка випадковим чином виконує дію з невеликою ймовірністю та в іншому випадку вибирає дію з високими значеннями, як описано нижче:

$$\pi(s) = argmax_a\, Q\,(s,a), \textit{ якщо } \sigma > \varepsilon,$$
$$a \approx A(s), \textit{ якщо } \sigma \leq \varepsilon, \quad (1.9)$$

де $0 \leq \sigma \leq 1$. Якщо значення епсилон високе, SARSA сходиться повільно, оскільки він більше досліджує, тоді як з малим значенням відвідування всіх пар стан-дія не є гарантованим, і це центральна дилема експлуатації-дослідження в НП налаштування. Однак у більшості випадків хорошим вибором значення епсилон є 0,1.

Q-навчання – це алгоритм прямого навчання поза політикою та модель РЧ для керування [14]. Його правило оновлення подано як:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \underset{a}{max} Q(s_t,a_t)]\ . \quad (1.10)$$

Метод Q-навчання навчається знаходити оптимальну політику шляхом спостереження, тобто навчання поза політикою. Його псевдокод наведено на рис. 1.6.

1. Ініціалізація
Q обирається довільним чином
Q (термінальне) = 0
Повторити
    ініціалізація s
    Повторити
        обрати $a' \in \varepsilon$-жадібне
        виконати дію a , спостерігати r, s'
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
        $s \leftarrow s'$
    s є термінальним
до конвергенції

Рисунок 1.6 – Алгоритм псевдокоду Q-навчання

В алгоритмі Q-навчання майбутня дія a 0 виконується за допомогою жадібної політики, тобто вибирається дія, яка має максимальне значення Q наступного стану. Конвергенцію в Q-навчанні можна пришвидшити за допомогою трасування відповідності, які оновлюються в Q-навчанні(λ) наступним чином:

$$e_t(s,a) = Iss_t \cdot Iaa_t + \gamma\lambda et - 1(s,a), \text{ якщо } Q_{t-1} = \max_a Q_{t-1}(s_t, a),$$

інакше,

$$e_t(s,a) = 0. \quad (1.11)$$

Iss_t використовується тут для ідентифікації. Для $s = s_t$ Iss_t та Iaa_t дорівнюють 1. δ для Q-навчання визначається як:

$$\delta_s = r_{t+1} + r\max_a Q_{t-1}(s_{t+1}, a) - Q_t(s_t, a_t). \quad (1.12)$$

Отримане правило оновлення для алгоритму Q-навчання за допомогою трасування таке:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \text{ для всіх } s \in S. \quad (1.13)$$

1.4 Постановка задачі та висновки до розділу 1

Багато типів проблем в автономному управлінні, прийнятті рішень та робототехніці можна моделювати як проблему навчання з підкріпленням. Механізм проб і помилок архітектури НП допомагає роботу та системі керування автономно вивчати оптимальну поведінку, взаємодіючи з навколишнім середовищем [15-16].

Деякі з додатків НП для розв’язання роботизованих проблем за допомогою алгоритмів, щоб увімкнути робота для завдання пенальті, завдання навігації, завдання стикування мобільного робота на основі бачення та завдання уникнення перешкод відповідно.

У робототехніці кінцевою метою навчання з підкріпленням є наділення роботів здатністю вчитися, покращувати, адаптувати і відтворювати завдання з обмеженнями, що динамічно змінюються на основі дослідження та автономного навчання. Для цього використовується сучасне навчання з підкріпленням, що ґрунтується на максимізації очікувань, і для кожного завдання пропонуються та оцінюються різні уявлення політики. Політики пропонують життєздатні рішення шести проблем, що рідко розв'язуються в уявленнях політики: кореляції, адаптованість, множинний дозвіл, глобальність, багатовимірність і конвергенція. Обговорюються як успіхи, і практичні труднощі, що у цих прикладах. На основі аналізу цих конкретних випадків робляться висновки про сучасний стан та перспективні напрямки навчання з підкріпленням у робототехніці [17].

Навігаційне завдання для автономного робота у невідомому середовищі є складним завданням. У роботі пропонується підхід, який ґрунтується на

прийнятті рішень, а саме навчанні з підкріпленням, щоб допомогти роботу вийти з невідомого лабіринту.

Основна ідея роботизованої навігації полягає в тому, щоб знайти шлях без зіткнень з початкового стану до цільового стану. Для цього робот повинен збирати інформацію з навколишнього середовища за допомогою датчиків та приймати навігаційні рішення на основі сенсорних даних та своєї бази знань.

Завдяки своїй простоті та хорошій продуктивності в реальному часі, Q-навчання може бути дуже ефективним для створення робота, швидко вивчати навколишнє середовище та успішно орієнтуватися у ньому.

Аналіз розглянутих існуючих методів прийняття рішень підтверджує доцільність застосування навчання з підкріпленням.

Метою кваліфікаційної роботи магістра є розроблення системи прийняття рішень для керування автоматизованою роботизованою мобільною платформою з використанням методу Q-навчання.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз систем штучного інтелекту та прийняття рішень, що можуть бути використані при керуванні роботизованою мобільною платформою;

– провести вибір основних елементів системи керування роботизованою мобільною платформою;

– розробити програмний засіб для прийняття рішень та керування роботизованою мобільною платформою мовою Python;

– провести моделювання та експериментальні дослідження з розробленим програмним засобом для прийняття рішень та керування роботизованою мобільною платформою у середовищі MatLab.

2 ВИБІР ОСНОВНИХ ЕЛЕМЕНТІВ СИСТЕМИ КЕРУВАННЯ РОБОТИЗОВАНОЇ МОБІЛЬНОЇ ПЛАТФОРМИ

2.1 Вибір елементів конструкції

2.1.1 Вибір основних модулів системи

В якості плати керування можна використовувати або плати Raspberry Pi Beaglebone Black.

Raspberry Pi – це недорогий міні-комп'ютер розміром із кредитну картку. Він працює з більш ніж кількома різновидами Linux і може виконувати майже всі завдання, які може виконувати звичайний комп'ютер (рис. 2.1). Raspberry Pi додатково підтримує взаємодію датчиків та виконавчих механізмів через контакти введення-виведення загального призначення. Оскільки Raspberry Pi працює під керуванням операційної системи Linux, він підтримує Python з коробки. Крім того, він надає набір контактів введення-виведення загального призначення (GPIO), які дозволяють вам керувати цифровими компонентами для фізичних обчислень та відкривати для себе Інтернет речей (IoT).

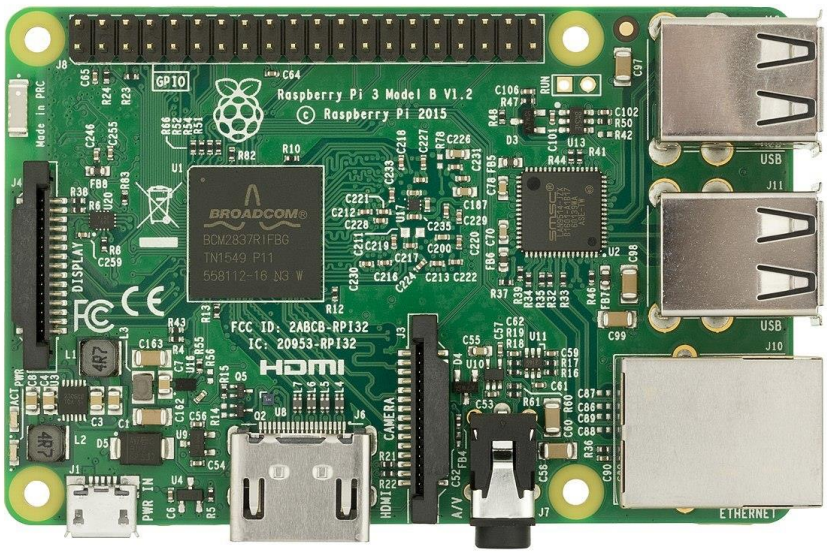


Рисунок 2.1 – Плата Raspberry Pi

Beaglebone Black – одноплатний комп'ютер з відкритим апаратним забезпеченням, розроблений Texas Instruments. Незважаючи на те, що він порівняний із Raspberry Pi, Beaglebone Black націлений на екстраординарний ринок. Замість орієнтуватися на любителів, це більше інженерна дошка (рис. 2.2).

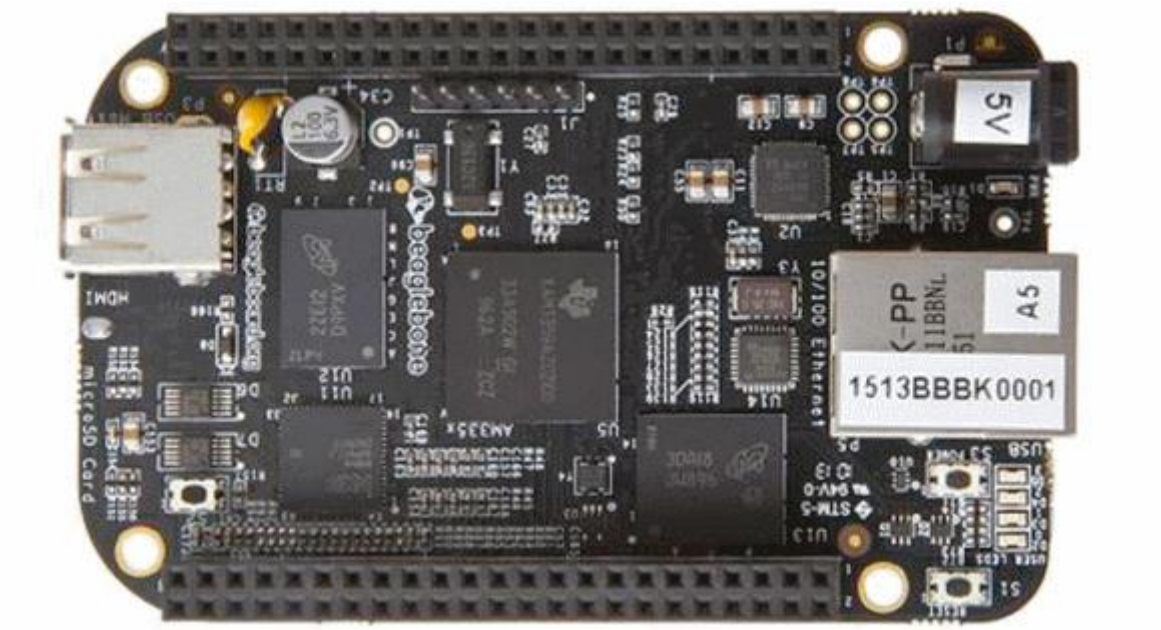


Рисунок 2.2 – Плата Beaglebone Black

З основними розбіжностями цих плат керування можна ознайомитись у таблиці 2.1.

Таблиця 2.1 – Різниця між Raspberry Pi і Beaglebone Black

Параметри	RASPBERRY PI	Beaglebone Black
1	2	3
Тип процесору	ARM11 процесор	ARM Cortex-A8 процесор
RAM	Для роботи raspberry pi використовується 512 Мб SDRAM	Для роботи beaglebone black використовується 512 Мб DDR3L
Швидкість процесору	використовує 700 МГц для обробки	для обробки використовує 1 ГГц

Продовження таблиці 2.1

1	2	3
Flash-пам’ять	Він має спеціальний роз'єм SD Card для завантаження операційної системи	Він використовує 4 Гб (мікро SD) для завантаження ОС і зберігання даних
Мінімальне живлення	Для цього потрібен джерело живлення 700 мА (3,5 Вт)	Для його функціонування потрібна мінімальна потужність 210 мА (1,05 Вт)
GPIO піни	Має 12 контактів GPIO	Має 69 контактів GPIO
Dev IDE	Він використовує IDLE, Scratch, Squeak/Linux для виконання завдань	Він використовує Python, Scratch, Squeak, Cloud9/Linux для виконання певного завдання
USB	На борту є 2 USB 2.0	На платі є 1 USB 2.0
Аудіо вихід	Підтримує HDMI, аналоговий аудіовихід	Використовує аналоговий вихід для аудіо
Відео вихід	Він підтримує HDMI, композитний вихід для відео	Немає такого специфічного відеовиходу
UART	Він використовує 1 UART для передачі та отримання послідовних даних	Він використовує 5 UART для передачі та отримання послідовних даних
Кількість пінів I/O	Він має 8 цифрових, 0 аналогових контактів	Має 65 цифрових і 7 аналогових контактів

Для побудови робототехнічної платформи було обрано плату Beaglebone під керуванням Linux як ОС, забезпечена виходами GPIO та PWM для керування двигунами постійного струму через Н-міст. Він також надає входи АЦП з максимальною аналоговою напругою 1,8 В для вибірки вихідних даних датчиків.

Шасі робота обрано простим – складається з двох рам та двох коліс, кожне з яких прикріплено до валу двигуна постійного струму. Для цього додатка не було необхідності в замкнутому контурі керування положенням валу двигунів постійного струму (рис. 2.3).

Також буде використано наступні модулі:

– три датчики наближення (ІЧ-датчики), здатні виявляти об'єкти, що знаходяться на відстані від 10 см до 80 см від датчика (рис. 2.4);

– драйвер L293d (подвійний Н-міст для керування двигунами постійного струму, рис. 2.5);

– лінійний регулятор на 5 В (рис. 2.6);

– акумулятор.

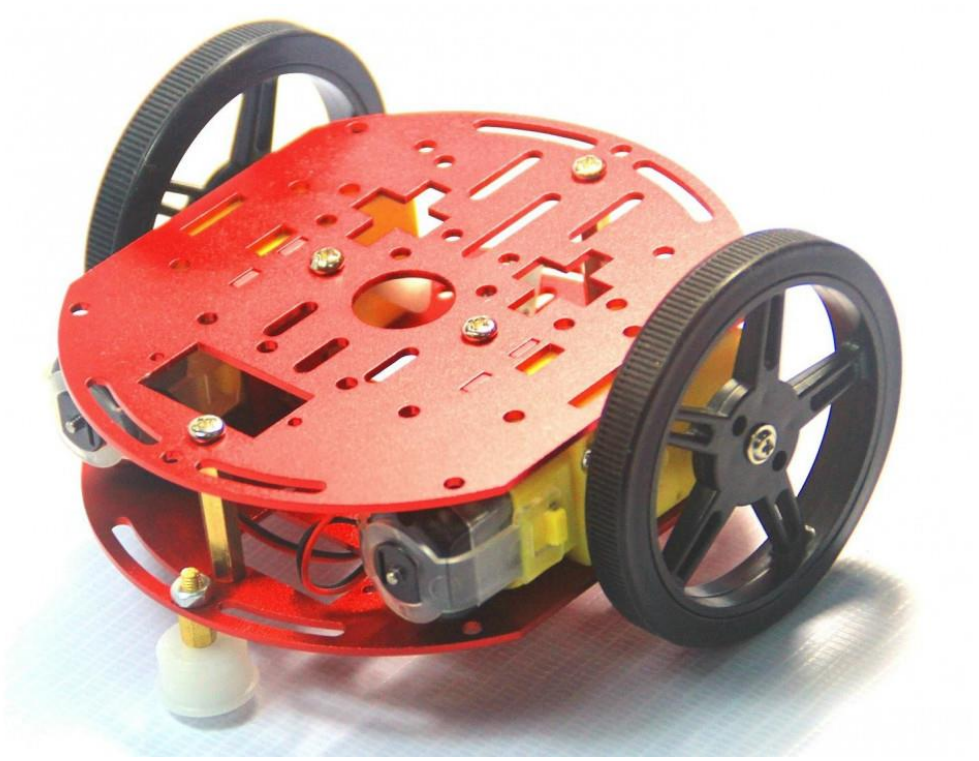


Рисунок 2.3 – Приклад мобільної платформи

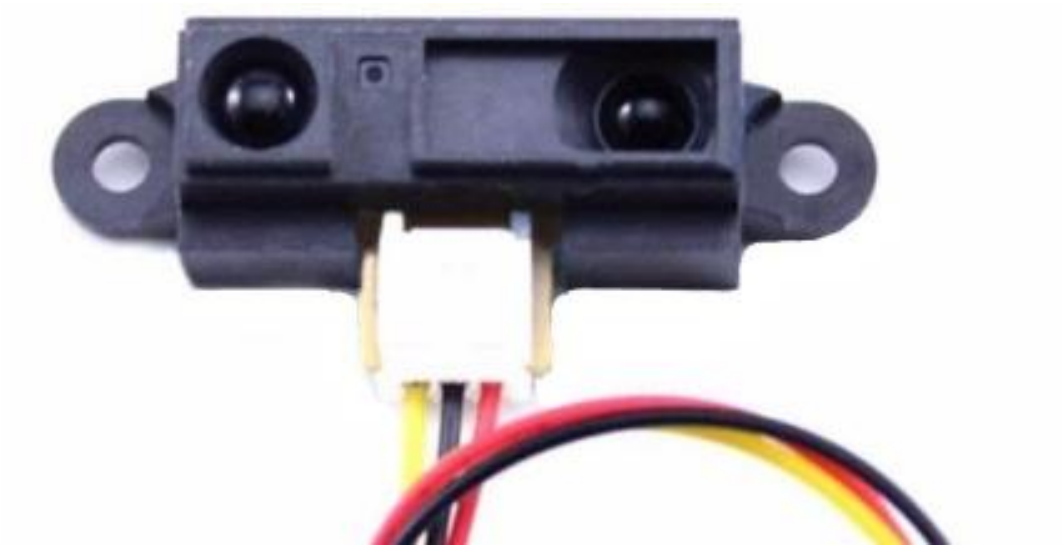


Рисунок 2.4 – ІЧ-датчик

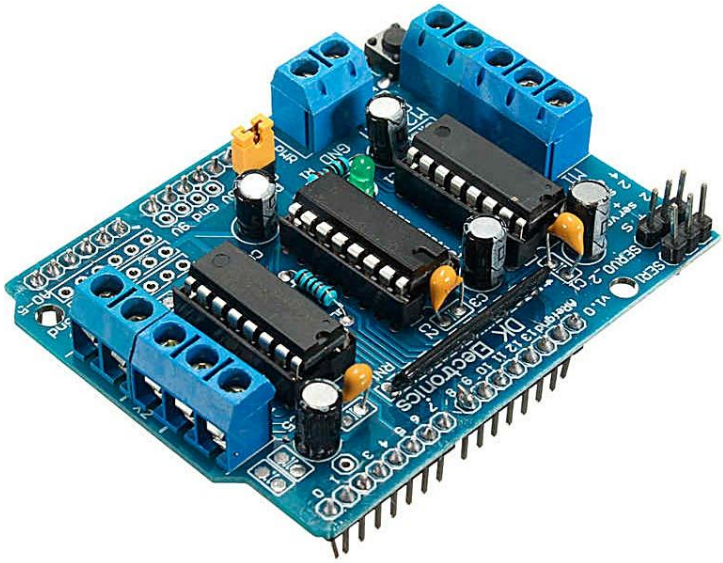


Рисунок 2.5 – Драйвер L293d

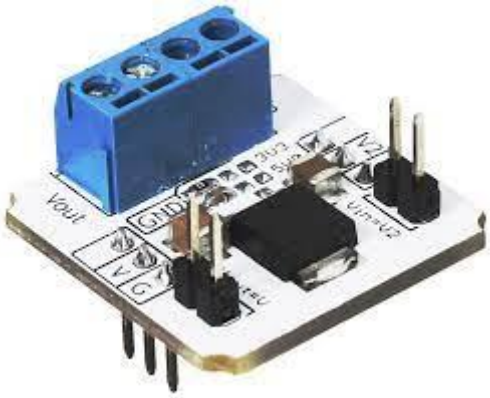


Рисунок 2.6 – Лінійний регулятор

2.1.2 Структурна схема розташування елементів конструкції

При створенні використовується підхід, заснований на Q-навчанні, щоб навчити робота уникати перешкод. Як різновид алгоритму навчання з підкріпленням, Q-навчання має наступні переваги:

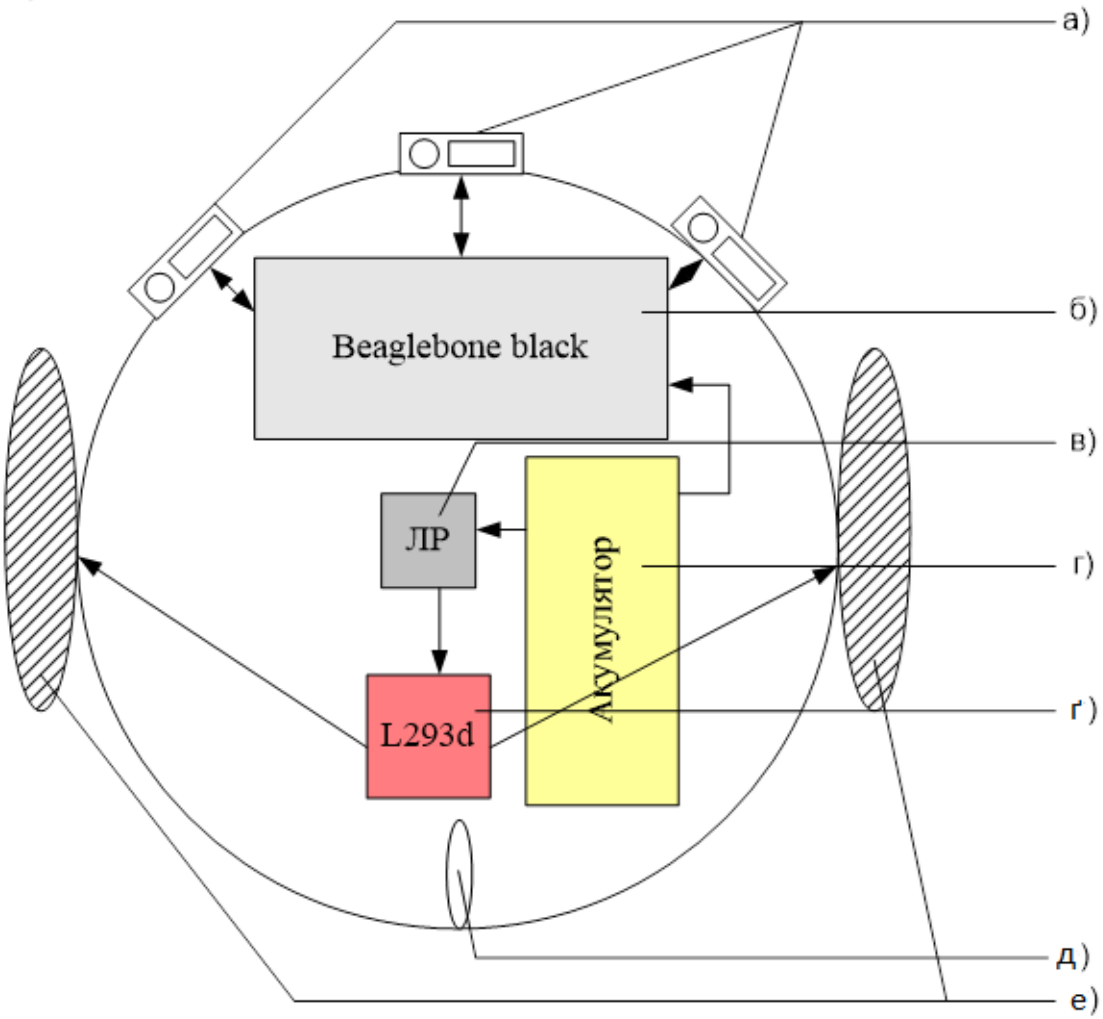
– низькі вимоги до обчислювальних ресурсів;

– хороша продуктивність у реальному часі;

– не вимагає навчальних вибірок.

Для визначення станів світу в алгоритмі Q-навчання використовуються три ІЧ-датчики наближення для визначення відстаней між роботом та найближчими перешкодами у трьох конкретних напрямках.

Структурна схема робота представлена на рис. 2.7.



а) ІЧ-датчики наближення, б) плата керування Beaglebone black, в) лінійний регулятор, г) акумулятор, ґ) драйвер двигуну, д) піддержуюче колесо, е) колеса з моторами

Рисунок 2.7 – Структурна схема робота

Об'єктом управління виступає робот із трьома ІЧ датчиками відстані. В роботі використовується комп'ютерна віртуальна модель фізичного пристрою, звана як агент.

Агент представлений у вигляді кількох матеріальних точок, що характеризують такі елементи:

– центр робота;

– лівий сенсор;

– правий сенсор;

– передній сенсор.

Агенту доступно три базові дії:

– рух вперед;

– поворот проти годинникової стрілки щодо центру робота;

– поворот за годинниковою стрілкою щодо центру робота.

2.2 Рух та стани мобільної роботизованої платформи

Робот повинен вивчити стратегію для автономного виходу з лабіринту – використати так звану політику для уникнення перешкод з використанням навчання із підкріпленням, а саме алгоритм Q-навчання.

У робота є здорові та нездорові стани, де перші – це ті, що входять до діапазонів вимірювання датчиків, а нездорові – які знаходяться за їх межами (рис. 2.8).

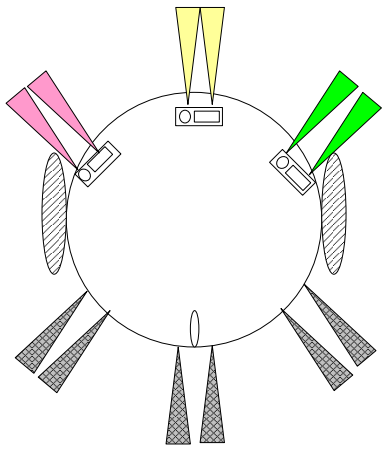


Рисунок 2.8 – Стани здоров’я та нездоров’я

Три конкретні напрямки були визначені як три різні колірні області (на рисунку кольорами показані здорові стани вимірювання датчиків, а сірим – нездорові).

Вектор стану визначається вихідним сигналом відносної відстані від трьох датчиків.

Стан може бути визначеним наступним чином:

$$s = [\textit{лівий правий передній}]. \quad (2.1)$$

Зокрема, три елементи вектора є відстань зліва, справа і спереду та їх відносні амплітуди.

Кожен елемент є цілим числом в діапазоні від 1 до 3, наприклад, вектор стану [1 2 3] представляє найбільшу відстань спереду та найменшу відстань зліва. Для цих станів доступні наступні дії:

– дія Вперед – просунутися вперед на 100 мм;

– дія Ліворуч – повернути ліворуч на 15°;

– дія Праворуч – повернути праворуч на 15°.

Крім того, для запобігання зіткненню робота з перешкодою в процесі навчання застосовується захисна дія: якщо одна з трьох відстаней менше 170 мм, робот рухатиметься назад на 100 мм.

Для кожної дії у конкретному стані визначається значення винагороди. Стан здоров'я (2.2) свідчить про наявність перешкод ще дуже далеко від робота, і вони не загрожуватимуть його руху. Отже, після виконання дії, якщо робот спостерігає за станом здоров'я, буде отримано позитивну винагороду.

З іншого боку, нездоровий стан (2.3) вказує на те, що принаймні одна з перешкод знаходиться дуже близько до роботи і роботу потрібно негайно відійти від нього. Таким чином, робот у нездоровому стані завжди буде отримати від'ємну винагороду.

Таким чином, значення винагороди за переміщення представлені у формулах (2.2) та (2.3). Докладніше про отримання винагород у розділах 3 та 4.

$$R = \begin{matrix} \textit{Стан/Дія} \\ 123 \\ 321 \\ 132 \\ 312 \\ 213 \\ 231 \end{matrix} \begin{matrix} \textit{Ліворуч} \\ \begin{bmatrix} 0 \\ 5 \\ 0 \\ 5 \\ 1 \\ 1 \end{bmatrix} \end{matrix} \begin{matrix} \textit{Праворуч} \\ \begin{bmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 5 \end{bmatrix} \end{matrix} \begin{matrix} \textit{Вперед} \\ \begin{bmatrix} 5 \\ 0 \\ 1 \\ 1 \\ 5 \\ 0 \end{bmatrix} \end{matrix}, \tag{2.2}$$

$$R = \begin{matrix} \textit{Стан/Дія} \\ 123 \\ 321 \\ 132 \\ 312 \\ 213 \\ 231 \end{matrix} \begin{matrix} \textit{Ліворуч} \\ \begin{bmatrix} -3 \\ 0 \\ -3 \\ 0 \\ -1 \\ -1 \end{bmatrix} \end{matrix} \begin{matrix} \textit{Праворуч} \\ \begin{bmatrix} -1 \\ -1 \\ 0 \\ -3 \\ -3 \\ 0 \end{bmatrix} \end{matrix} \begin{matrix} \textit{Вперед} \\ \begin{bmatrix} 0 \\ -3 \\ -1 \\ -1 \\ 0 \\ -3 \end{bmatrix} \end{matrix}. \tag{2.3}$$

2.3 Процес навчання агенту

Процес навчання побудований з урахуванням алгоритму Q-навчання. На основі одержуваної від середовища винагороди агент (мобільна роботизована платформа) формує функцію корисності Q, що згодом дає можливість вже невипадково вибирати стратегію поведінки, а враховувати досвід попередньої взаємодії із середовищем.

Одна з переваг Q-навчання полягає в тому, що вона може порівняти очікувану корисність доступних дій, не формуючи моделі навколишнього середовища.

Вибраний метод заснований на введенні функції, що відображає цінність кожної можливої дії агенту для поточного стану, в якому зараз симуляція. Ця функція визначає оцінку агентом тієї нагороди, яку він може отримати,

здійснивши в певний хід певну дію. А також вона включає оцінку того, яку нагороду агент може отримати у майбутньому. Процес навчання є ітераційним уточненням значення функції на кожному кроці агенту. Насамперед слід визначити величину нагороди, яку агент отримає у цей хід.

Навчання агенту відбувається епохами, кожна епоха задає нове оточення для агенту. Однак накопичений досвід агенту зберігається і з збільшенням числа епох навчання досвід зростає. На рис. 2.9 представлений алгоритм навчання та процес оновлення параметрів.

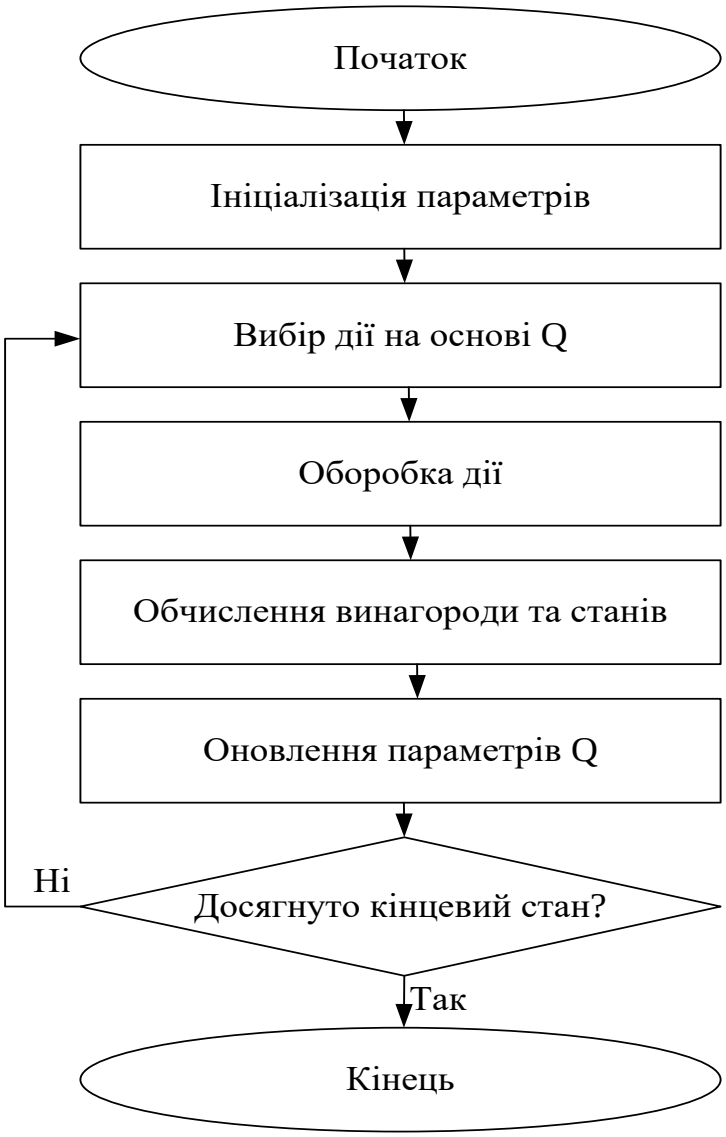


Рисунок 2.9 – Алгоритм навчання та процес оновлення параметрів

Кожна епоха представляє прохід робота по карті з коригуванням матриці вагових коефіцієнтів Q. Стартове положення робота задається вручну у програмі, вибираються всі точки, з яких можна зробити кроків уперед, і для кожної епохи випадково береться одна з них, після цього до кожної координаті додається випадкове ціле число з діапазону. Це робиться для того, щоб збільшити кількість початкових станів і перешкоджати процесу перенавчання.

2.5 Висновки до розділу 2

У другому розділі роботи було обрано основні модулі робота – його базу, датчики та модулі, що будуть використані у реальній моделі

Побудовано функціональну схему робота зі схематичним розташуванням обраних модулів.

Також було визначено, що у робота є здорові та нездорові стани і отримано їх значення.

Після цього було визначені дії для цих станів, а також визначено значення винагороди за переміщення.

Було отримано алгоритм процесу навчання з урахуванням алгоритму Q-навчання.

3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАСОБУ ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ ТА КЕРУВАННЯ РОБОТОМ

3.1 Алгоритм роботи програми

Алгоритм роботи програми керування представлено на рис. 3.1.

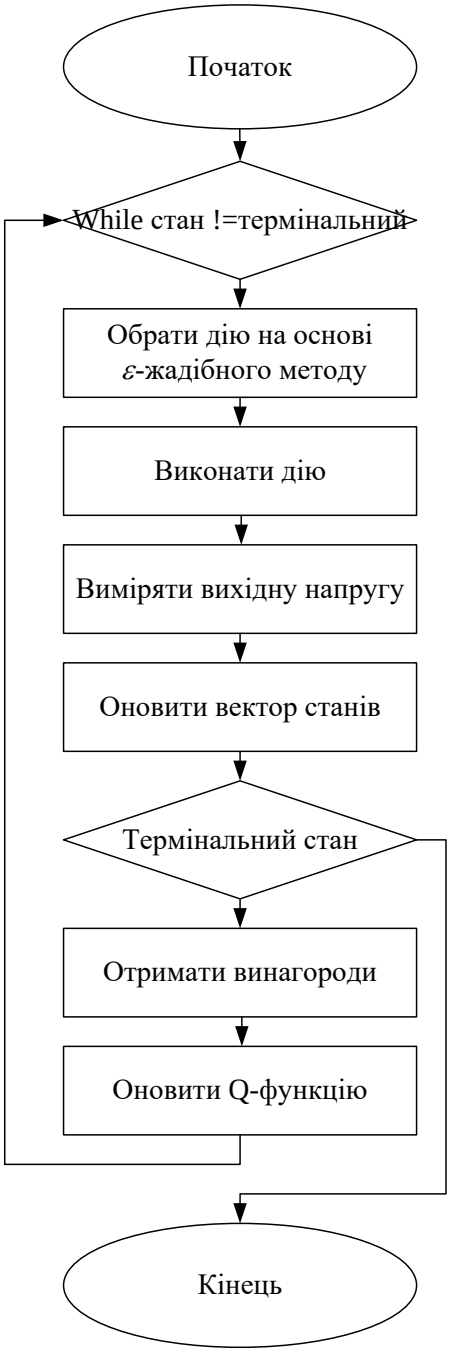


Рисунок 3.1 – Алгоритм роботи програми

Термінальний стан – аварійний стан, що може статися при виконанні алгоритму, може бути певний заданий параметр чи робототехнічна платформа досягла кінця лабіринту, будь-який із датчиків має відстань меншу за мінімальну дозволену для нездорового стану.

Для обирання дії за допомогою ε -жадібного методу використовується підхід до розвідки та експлуатації.

Під виконанням дії розуміється вперед, праворуч або ліворуч, увімкнувши обидва двигуни, лівий або правий двигун відповідно.

Вимірювання вихідної напруги відбувається з датчиків наближення.

Оновлення вектору станів має в своєму складі дії занесення даних до станів здоров’я та нездоров’я.

Отримання винагороди відбувається на основі попереднього стану та виконаної дії.

3.2 Розробленння програмного засобу прийняття рішень мовою Python

Розробка програмного засобу (ПЗ) прийняття рішень для керування починається з підключення бібліотек, де найголовнішими є бібліотеки для роботи з платою Beaglebone black, а саме аналогово-цифровий перетворювач (ADC), широтно-імпульстна модуляція (PWM) та портами введення-виведення (GPIO), а також для роботи з часом:

```
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.PWM as PWM
import Adafruit_BBIO.GPIO as GPIO
from time import sleep
```

Далі створюємо зміні, що відповідають за піни ІЧ-датчиків:

```
pin1 = "P9_35"
pin2 = "P9_37"
pin3 = "P9_39"
```

А також піни для керування моторами (лівого та правого коліс відповідно):

```
input_1A = "P8_12"
input_2A = "P8_14"
input_3A = "P8_8"
input_4A = "P8_10"
```

Встановлюємо піни на виведення:

```
GPIO.setup(input_1A, GPIO.OUT)
GPIO.setup(input_2A, GPIO.OUT)
GPIO.setup(input_3A, GPIO.OUT)
GPIO.setup(input_4A, GPIO.OUT)
```

Та змінні з пінами широтно-імпульстної модуляції:

```
motor1 = "P9_14"
motor2 = "P9_16"
```

Далі створюємо клас, що зберігає траєкторію:

```
class trajectory(object):
    def __init__(self, x_coord_origin, y_coord_origin, x_coord_s1, y_coord_s1, x_coord_s2, y_coord_s2, x_coord_s3,
                 y_coord_s3):
        self.x_coord_origin = x_coord_origin
        self.y_coord_origin = y_coord_origin
        self.x_coord_s1 = x_coord_s1
        self.y_coord_s1 = y_coord_s1
        self.x_coord_s2 = x_coord_s2
        self.y_coord_s2 = y_coord_s2
        self.x_coord_s3 = x_coord_s3
        self.y_coord_s3 = y_coord_s3
```

Наступним є клас, що зберігає перехід стану:

```
class state_transition(object):
    def __init__(self, reward, next_s_type, next_state, terminal, emergency):
        self.reward = reward
```

```
        self.next_s_type = next_s_type
        self.next_state = next_state
        self.terminal = terminal
        self.emergency = emergency
```

Потім створили функцію для читання інформації з сенсорів за допомогою АЦП (діапазон (0,96 В >= 10 см до 0,13 В > 70 см)):

```
def readSensors(pin1, pin2, pin3):
    sensor1Val = ADC.read(pin1)
    sensor1Volts = sensor1Val * 1.8
    sensor2Val = ADC.read(pin2)
    sensor2Volts = sensor2Val * 1.
    sensor3Val = ADC.read(pin3)
    sensor3Volts = sensor3Val * 1.8
    sensors = list()
    sensors.append(sensor1Volts)
    sensors.append(sensor2Volts)
    sensors.append(sensor3Volts)
    return sensors
```

Далі знаходимо евклідову відстань між датчиками та перешкодою:

```
def euclideanDistance2integers(sensors):
    state_vector = np.zeros(3)  # array of 3 elements
    state_vector[sensors.index(max(sensors))] = 1
    state_vector[sensors.index(min(sensors))] = 3
    for i in range(0, 3):
        if state_vector[i] == 0:
            state_vector[i] = 2
    return state_vector
```

Потрібно з’ясувати тип стану – здоров’я чи нездоров’я:

```
def state_type(sensors, critical_voltage):
```

```
    if sensors[0] > critical_voltage or sensors[1] > critical_voltage or sensors[2] > critical_voltage:
        s_type = 1
    else:
        s_type = 2
    return s_type
```

Створюємо функцію отримання винагороди:

```
def trans_reward(action, s_type, current_state):
    if action == 4:
        reward = 0
    else:
        if current_state.argmax() == action:
            if s_type == 1:
                reward = 0
            else:
                reward = 5
        elif current_state.argmin() == action:
            if s_type == 1:
                reward = -3
            else:
                reward = 0
        else:
            if s_type == 1:
                reward = -1
            else:
                reward = 1
    return reward
```

Створюються функції для переміщення вліво, вправо, вперед, назад, приклад одної з них наведено нижче:

```
def gotoRight():
```

```
    GPIO.output(input_1A, GPIO.HIGH)
    GPIO.output(input_2A, GPIO.LOW)
    GPIO.output(input_3A, GPIO.LOW)
    GPIO.output(input_4A, GPIO.LOW)
    duty = 30
    PWM.start(motor1, duty)
    sleep(0.2)
    PWM.stop(motor1)
    PWM.cleanup()
```

Наступною є функція виконання дії:

```
def execute_action(action):
    duty = 40
    if action == 1:
        goForward()
    elif action == 2:
        gotoLeft()
    elif action == 3:
        gotoRight()
    elif action == 4:
        goBackward()
    elif action == 5:
        gotoRight()
        goForward()
    else:
        gotoLeft()
        goForward()
```

Наступною є функція аналізу робочого простору:

```
def environment(action, current_state, s_type, critical_voltage, emergency_voltage, termination_voltage):
    execute_action(action)
```

```
    sensors = readSensors(pin1, pin2, pin3)
    if (sensors[1] >= emergency_voltage):
        emergency = 1
    else:
        emergency = 0
    next_s_type = state_type(sensors, critical_voltage)
    next_state = euclideanDistance2integers(sensors)  # vector of 3 integers from 1-3
    if sensors[0] <= termination_voltage and sensors[1] <= termination_voltage and sensors[2] <= termination_voltage:
        terminal = 1
    else:
        terminal = 0
    reward = trans_reward(action, s_type, current_state)
    return state_transition(reward, next_s_type, next_state, terminal, emergency)
```

Також потрібно створити функцію станів:

```
def state_index(s, s_type):
    ref_state1 = np.array([1, 2, 3])
    ref_state2 = np.array([1, 3, 2])
    ref_state3 = np.array([3, 2, 1])
    ref_state4 = np.array([3, 1, 2])
    ref_state5 = np.array([2, 3, 1])
    ref_state6 = np.array([2, 1, 3])
    if np.array_equal(s, ref_state1) == True:
        s_index = 1
    elif np.array_equal(s, ref_state2) == True:
        s_index = 2
    elif np.array_equal(s, ref_state3) == True:
        s_index = 3
    elif np.array_equal(s, ref_state4) == True:
```

```
            s_index = 4
        elif np.array_equal(s, ref_state5) == True:
            s_index = 5
        else:
            s_index = 6
        if s_type == 1:
            s_index = s_index + 6
        s_index = s_index - 1
        return s_index
```

Також задаємо у головній функції необхідні параметри:

```
if __name__ == "__main__":
    # масив координат траєкторії агента (координати початку, s1,s2,s3)
    robotTrajectory = trajectory([], [], [], [], [], [], [], [])  # define original coordinates
    Nepisodes = 1
    critical_voltage = 0.4  # ~15 см від лабіринту
    emergency_voltage = 0.85  # 10см або менше від лабіринту
    termination_voltage = 0.18  # [v]
    # дії
    a = [1, 2, 3]
    Nactions = 3
    # стани
    Nstates = 12  # 12 станів: 6 здоров'я і 6 нездоров'я
```

Функція значення дії Q:

```
    Q = np.zeros((Nstates, Nactions))
    Q[0, 2] = 100
    Q[1, 1] = 100
    Q[2, 0] = 100
    Q[3, 0] = 100
    Q[4, 1] = 100
```

```
Q[5, 2] = 100
Q[6, 2] = 100
Q[7, 1] = 100
Q[8, 0] = 100
Q[9, 0] = 100
Q[10, 1] = 100
Q[11, 2] = 100
Параметри Q-навчання:
gamma = 0.9  # дисконтний коефіцієнт
alpha = 0.1  # постійний розмір кроку
epsilon = 0.1  # епсілон для ε -жадібної політики
sensors = readSensors(pin1, pin2, pin3)
if (sensors[1] >= emergency_voltage):
    emergency = 1
else:
    emergency = 0
Отримати тип стану – здоров’я, нездоров’я:
s_type_start = state_type(sensors, critical_voltage)
print("s_type_start", s_type_start)
start_state = euclideanDistance2integers(sensors)
print("start_state=", start_state)
s_index = state_index(start_state, s_type_start)
print("s_index_start=", s_index)
for i in range(0, Nepisodes):
    terminal = 0
    s = start_state
    s_type = s_type_start
    while terminal == 0:
Обрати наступну дію (ε -жадібної політики)
        if (emergency == 1):
```

```
            a = 4
        else:
            Q_row = Q[s_index - 1]
            if random.random() < (1 - epsilon):
                if np.all(Q_row == 0):
                    a = random.randint(1, 3)
                else:
                    a = Q_row.argmax() + 1
            else:
                a = random.randint(1, 3)
        print( "action=", a)
```

Виконання обраної дії та перехід до наступного стану:

```
        s_transition = environment(a, s, s_type, critical_voltage, emergency_voltage, termination_voltage)
        s_index_next = state_index(s_transition.next_state, s_transition.next_s_type)
        if a != 4:
```

Оновити Q:

```
            Q[s_index, a - 1] = Q[s_index, a - 1] + alpha * (
                    s_transition.reward + gamma * max(Q[s_index_next, :]) - Q[s_index, a - 1])
```

Оновити стан оновлення, тип стану та розташування:

```
        s = s_transition.next_state
        s_type = s_transition.next_s_type
        s_index = s_index_next
        print("next_state=", s)
        print("next_s_type=", s_type)
        print("next_s_index=", s_index)
        print("next_Q=", Q)
        print("\n\n")
```

```
emergency = s_transition.emergency
terminal = s_transition.terminal
print("emergency=", emergency)
print("terminal=", terminal)
```

3.3 Висновки до розділу 3

В результаті написання 3 розділу було розроблено програмний засіб прийняття рішень для керування автоматизованою роботизованою мобільною платформою.

Апаратна реалізація має деякі обмеження, здебільшого пов'язані з обмеженнями апаратних компонентів.

Наприклад, обмежена вибірка лабіринту/перешкоди, що забезпечується вибраними ІЧ-датчиками (які можуть виявляти перешкоди лише на відстані від 10 см до 80 см від датчика) та їх кількість (рекомендується додати більше датчиків для більш точної вибірки).

Крім того, замкнутий контур керування положенням двигунів постійного струму забезпечить більш точне виконання дій.

На жаль, через неможливість розробити реальну модель автоматизованої роботизованої мобільної платформи і провести з нею експериментальні дослідження було прийнято рішення зробити комп’ютерну модель у середовищі MatLab для підтвердження роботи запропонованого алгоритму Q-навчання, що буде розглянуту у розділі 4.

4 МОДЕЛЮВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАСОБУ У СЕРЕДОВИЩІ MATLAB

4.1 Алгоритм роботи системи моделювання

Політика обходу перешкод, описана вище, була реалізована в MatLab і була побудована траєкторія руху агенту в часі.

Агент (мобільний робот) був змодельований як коло з 3 відповідними точками: головним колом та трьома точками на колі, що віддалені один від одного на 90°.

Ці кола представляють собою 3 датчики:

– передній;

– лівий;

– правий.

Відстань між перешкодами та роботом у будь-який момент часу вимірюється за допомогою евклідової відстані між точкою датчику та точкою перетину лінії.

Ця лінія з'єднує точку датчика і перешкоду (та сама лінія, що з'єднує точку датчика та початок кола).

В якості лабіринту, що було використано для моделювання алгоритму Q-навчання, було обрано складну траєкторію – спіральний лабіринт.

Алгоритм роботи цієї системи моделювання майже такий самий, як і у розділі 3 за невеликими відмінностями – виконання дій лише запускає переміщення розробленої моделі робота (коло с 3 малими колами) вперед, ліворуч та праворуч, а також обчислення евклідової відстані між кожним з датчиків та перешкодами.

Змінений алгоритм представлено на рис. 4.1.

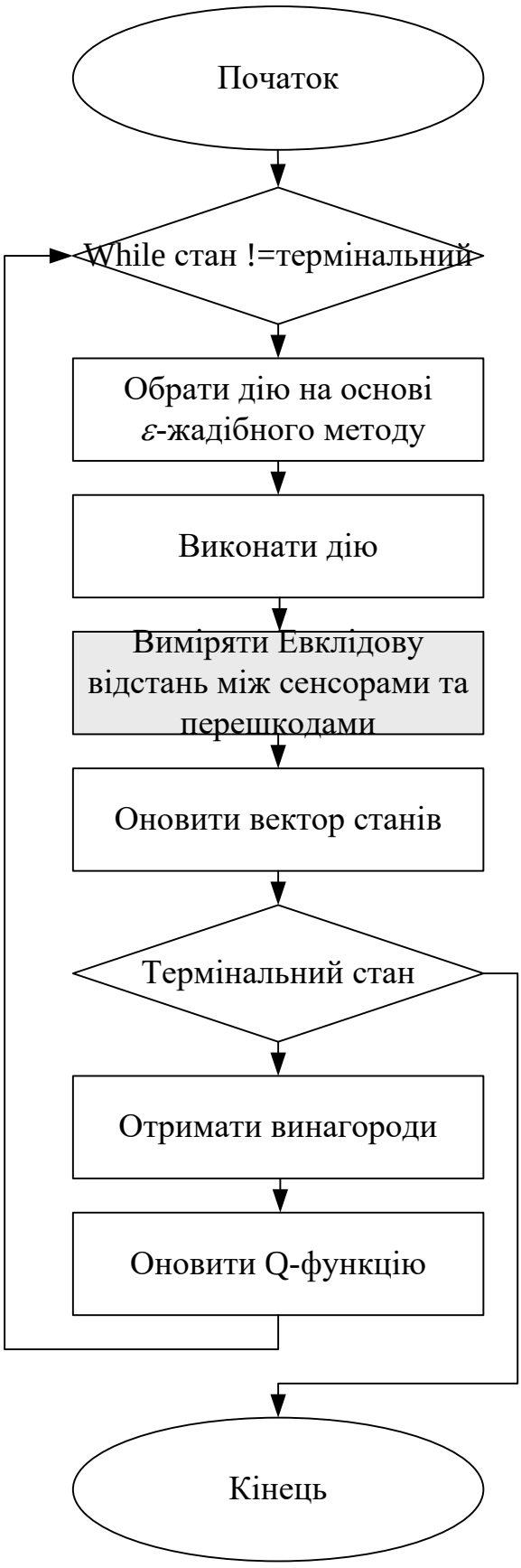


Рисунок 4.1 – Блок-схема алгоритму роботи програми

4.2 Розроблення системи моделювання

Аналогічно до попереднього розділу, тут використано схожі функції, але для зручності їх було рознесено до різних m-файлів.

Всього програма моделювання складається з 14 файлів:

– Main.m – головний файл з логікою;

– agent_location.m – для оцінки нового місця для наступної дії в залежності від типу самої дії (переміщення вперед, ліворуч, праворуч), а також малювання наступного положення агенту;

– circle.m – коло для побудови робота;

– environment.m – відповідає за оцінку нових координати агенту після виконання дії, оцінку нових станів (на основі нового розташування та вимірювань датчиків), та нагороди;

– euclideanDistance2integers.m – розрахунок евклідової відстані між роботом та перешкодами;

– InterX.m – функції для розрахунку перетину кривих;

– left_right_action.m – ця функція повертає координати агенте після обертання праворуч або ліворуч;

– maze.m – для побудови лабіринту;

– next_state.m – для отримання наступного стану переміщення;

– reward.m – функція отримання винагороди за переміщення;

– sonar.m – функція, що відповідає за віртуальні датчики і їх логіку;

– state.m – функція, що відповідає за вимірювання від датчиків, визначення типу стану (здоров’я чи нездоров’я), визначення термінального стану;

– state_index – визначає індекс станів з матриці станів;

– state_type.m – визначає тип стану – здоровий чи нездоровий.

Найголовнішим файлом є файл Main.m.

В цьому файлі спочатку оголошується функція для побудови спірального лабіринту:

[maze_coordinates]=maze();

Далі задається сам агент (мобільна роботизована платформа) та його основні параметри, такі як початкові координати робота (origin), початковий кут повороту (angle) та радіус самого робота (r):

origin=[0.0,-1.0];

angle=60;

r=0.7.

Місцезнаходження є next_location та current_location – масиви 4x2, які містять координати (x,y) з origin, sensor1, sensor2, sensor3 початкових координат розташування:

initial_location=left_right_action(angle, origin,r);

Масив траєкторій зберігає координати (x,y) початкової точки агенту, поки він знаходить вихід із лабіринту:

```
j=1;
trajectory(j,:)=initial_location(1,:);
```

Рисуємо початкове розташування агенту:

```
circle(initial_location(1,1),initial_location(1,2),r);
hold on
   for i=1:4
      plot(initial_location(i,1),initial_location(i,2),'o','MarkerSize',4)
   end
```

Далі потрібно отримати початкові стани. Коли будь-яка з евклідових відстаней (сенсорний лабіринт) опускається нижче критичної відстані, агент переходить у нездоровий стан:

critical_distance=1;

Якщо відстань нижче цієї умови, наступною дією є рух назад, щоб уникнути зіткнення:

emergency_distance=0.2; %

[terminal,type,start_state,emergency]=state(initial_location,maze,critical_distance,emergency_distance);

Індекс стану (для адресації стану в матриці Q є 12 станів: 6 здоров’я та 6 нездоров’я):

```
s_index=state_index(start_state,type);
```

Далі проводять дослідження і встановлюють певні параметри для них, так як кількість епізодів, набір дій, функція значення дії (12 станів: 6 нездорових і 6 здоров’я/3 дії), дисконтування, чим ближче до 1, тим швидше збіжність, константа розміру кроку, епсілон дляε -жадібної політики, ймовірність вибору випадкової дії:

```
Nepisodes=1;
a=[1 2 3];
Nactions = length(a);
Q=zeros(12,3);
gamma=0.9;
alpha=0.1;
epsilon=0.1;
```

Далі вже будується логіка Q-навчання:

```
for i=1:Nepisodes
   terminal=0;
   s=start_state;
   location=initial_location;
```

Запускається епізод, доки не знайдеться термінальний (аварійний) стан:

```
   while (terminal==0)
```

Вибрати дію next_state за політикою (ε -жадібне)

```
      if emergency==1
         a=4;
      else
         if rand<(1-epsilon)
            [q a] = max(Q(s_index,:));
         else
            a=randi(Nactions);
```

```
            end
        end
```

Далі потрібно виконати дію та досягти наступного стану: [agent_next_location,reward,next_s_type,next_state,terminal,emergency]=environment(a,s,type,location,maze_coordinates,critical_distance,emergency_distance);

Оновити траєкторію походження агенту, яку потрібно побудувати:

```
        j=j+1;
        trajectory(j,:)=agent_next_location(1,:);
        s_index_next=state_index(next_state,next_s_type);
        if a~=4 %emergency
```

Оновити параметри Q:

```
    Q(s_index,a)=Q(s_index,a)+alpha*(reward+gamma*max(Q(s_index_next,:))-Q(s_index,a));
        end
```

Оновити стан, місце розташування та тип стану:

```
        s=next_state;
        location=agent_next_location;
        type=next_s_type; %update type of state
        s_index=s_index_next;
         end
      end
```

4.3 Проведення експериментальних досліджень з розробленим ПЗ

Параметри Q-навчання були наступними:

– коефіцієнт дисконтування = 0,9;

– константа розміру кроку = 0,1;

– епсилон (ε -жадібний параметр) = 0,1.

Результати цієї реалізації представлені на наступних рисунках, на яких видно траєкторію походження агенту з початкової точки (вибраної випадковим

чином на початку спірального лабіринту) та кінцевої точки (кінець лабіринту) у часі.

На рис. 4.2 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 60°;

– початкова точка розташування робота (0.0, -1.0).

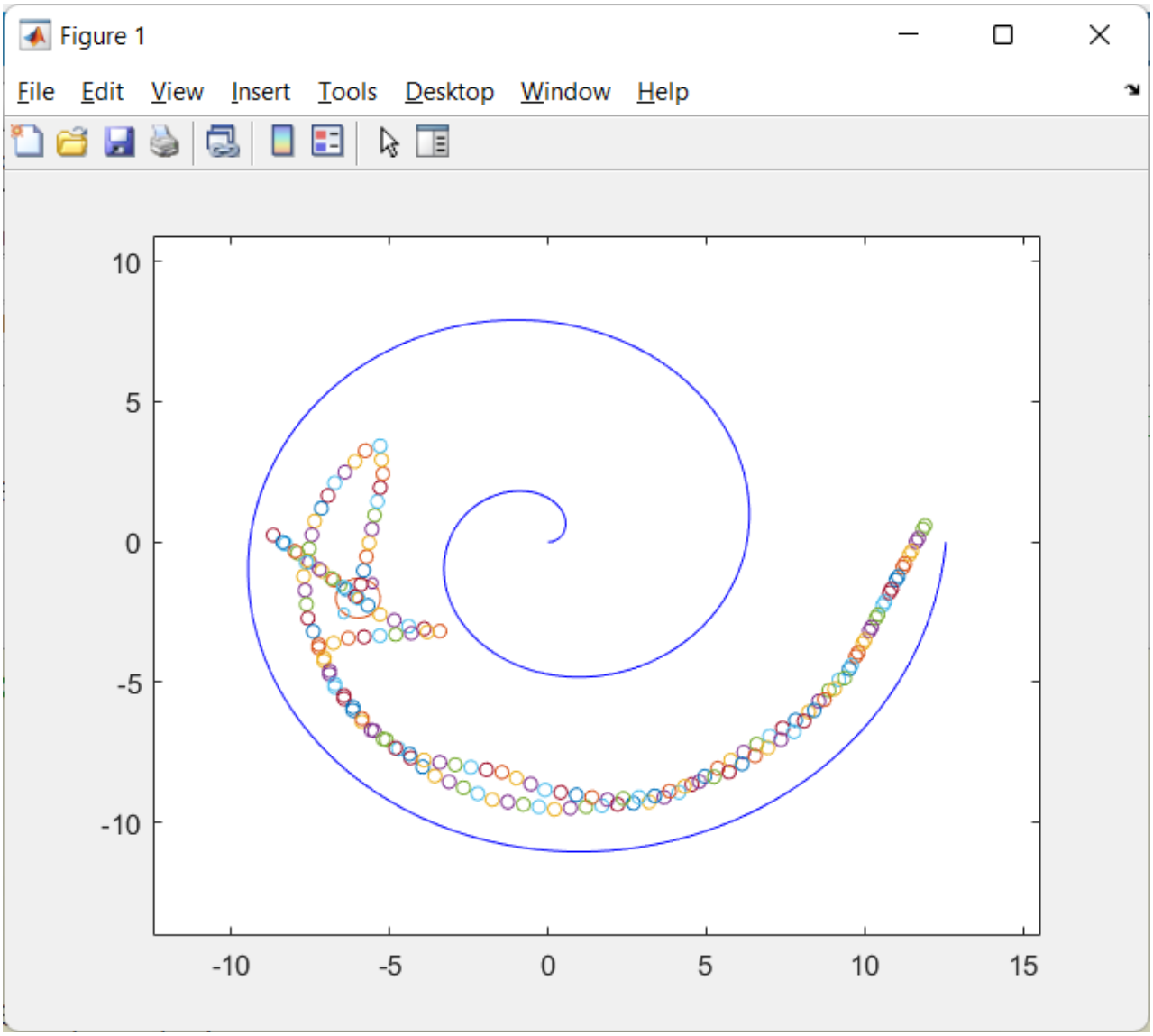


Рисунок 4.2 – Шлях роботу з використанням Q-навчання з параметрами кута 60°, початковою точкою (0.0, -1.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.1.

Таблиця 4.1 – Значення Q для кута 60°, точки (0.0, -1.0)

	Дії		
Стани	0.0855000000000000	0	0
	0.0330578100000000	0	0
	15.7110812094986	0.130068550000000	1.59490144398062
	5.91247952609442	0	0.411669383703029
	-0.237000449100000	11.9258229898298	2.14889959434705
	0.220546897385581	1.10229979835627	-0.283834939697525
	0.265846217269500	0	0
	-0.100000000000000	0.188723148620163	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 179 точок. Для наочності представлено деякі з них у табл. 4.2.

Таблиця 4.2 – Координати траєкторії переміщення

Координата x	Координата y
0	-1
-0.433012701892219	-0.750000000000000
-0.0866025403784438	-0.950000000000000
0.346410161513775	-1.20000000000000
0.779422863405995	-1.45000000000000
1.21243556529821	-1.70000000000000
1.64544826719043	-1.95000000000000
2.07846096908265	-2.20000000000000
2.51147367097487	-2.45000000000000
2.94448637286709	-2.70000000000000
…	…
9.12012981404473	-4.37154351474580
9.12012981404473	-3.87154351474580
9.12012981404473	-3.37154351474580
9.12012981404473	-2.87154351474580
9.12012981404473	-2.37154351474580
9.12012981404473	-1.87154351474580
9.12012981404473	-1.37154351474580
9.12012981404473	-0.871543514745802
9.12012981404473	-0.371543514745802
9.12012981404473	0.128456485254198

На рис. 4.3 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 50°;

– початкова точка розташування робота (0.0, -1.0).

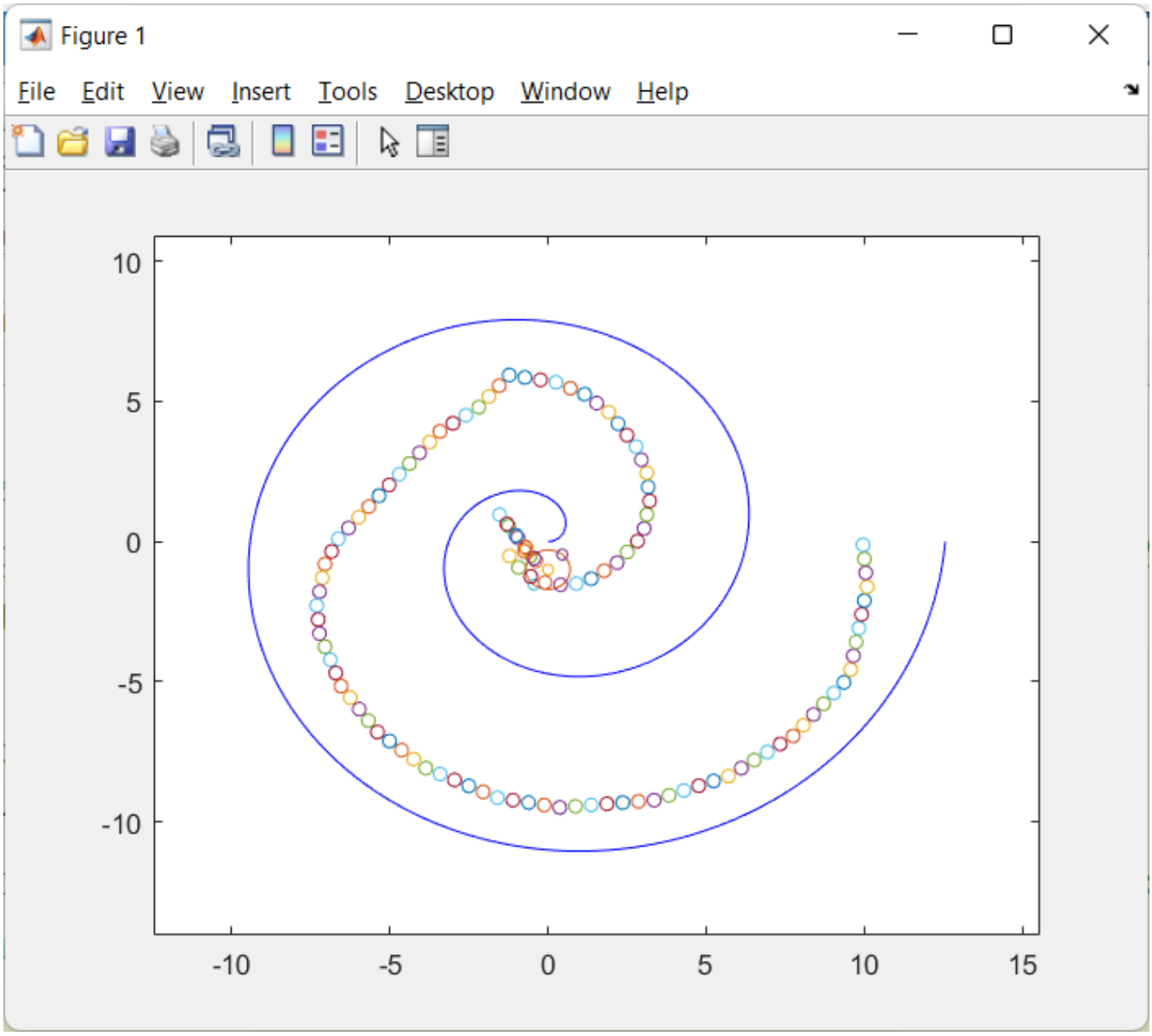


Рисунок 4.3 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (0.0, -1.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.3.

Таблиця 4.3 – Значення Q для кута 50°, точки (0.0, -1.0)

	Дії		
Стани	0	0	0
	0.00900000000000000	0	0
	20.6263010295260	1.88444553940712	3.78843618186482
	5.30441386682585	0	0
	-0.201000000000000	14.5060155733963	0.724324595092762
	0	6.66622847441407	0
	2.14721375323228	0	0.420970829914247
	-0.100000000000000	0.243552196755360	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 166 точок. Для наочності представлено деякі з них у табл. 4.4.

Таблиця 4.4 – Координати траєкторії переміщення

Координата x	Координата y
0	-1
-0.383022221559489	-0.678606195156730
-0.766044443118978	-0.357212390313461
-1.14906666467847	-0.0358185854701911
-1.53208888623796	0.285575219373079
-1.91511110779745	0.606969024216348
-1.60869333054985	0.349853980341732
-1.22567110899037	0.0284601754984628
-0.842648887430876	-0.292933629344807
-0.459626665871387	-0.614327434188076
…	…
9.84952518687494	-3.33156017526174
10.0608343177453	-2.87840628174342
10.2721434486156	-2.42525238822509
10.4834525794860	-1.97209849470677
10.6947617103563	-1.51894460118844
10.6947617103563	-1.51894460118844
10.6947617103563	-1.51894460118844
10.6947617103563	-1.51894460118844
10.5237516386935	-1.04909829079549
10.3527415670307	-0.579251980402532

На рис. 4.4 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 70°;

– початкова точка розташування робота (0.0, -1.0).

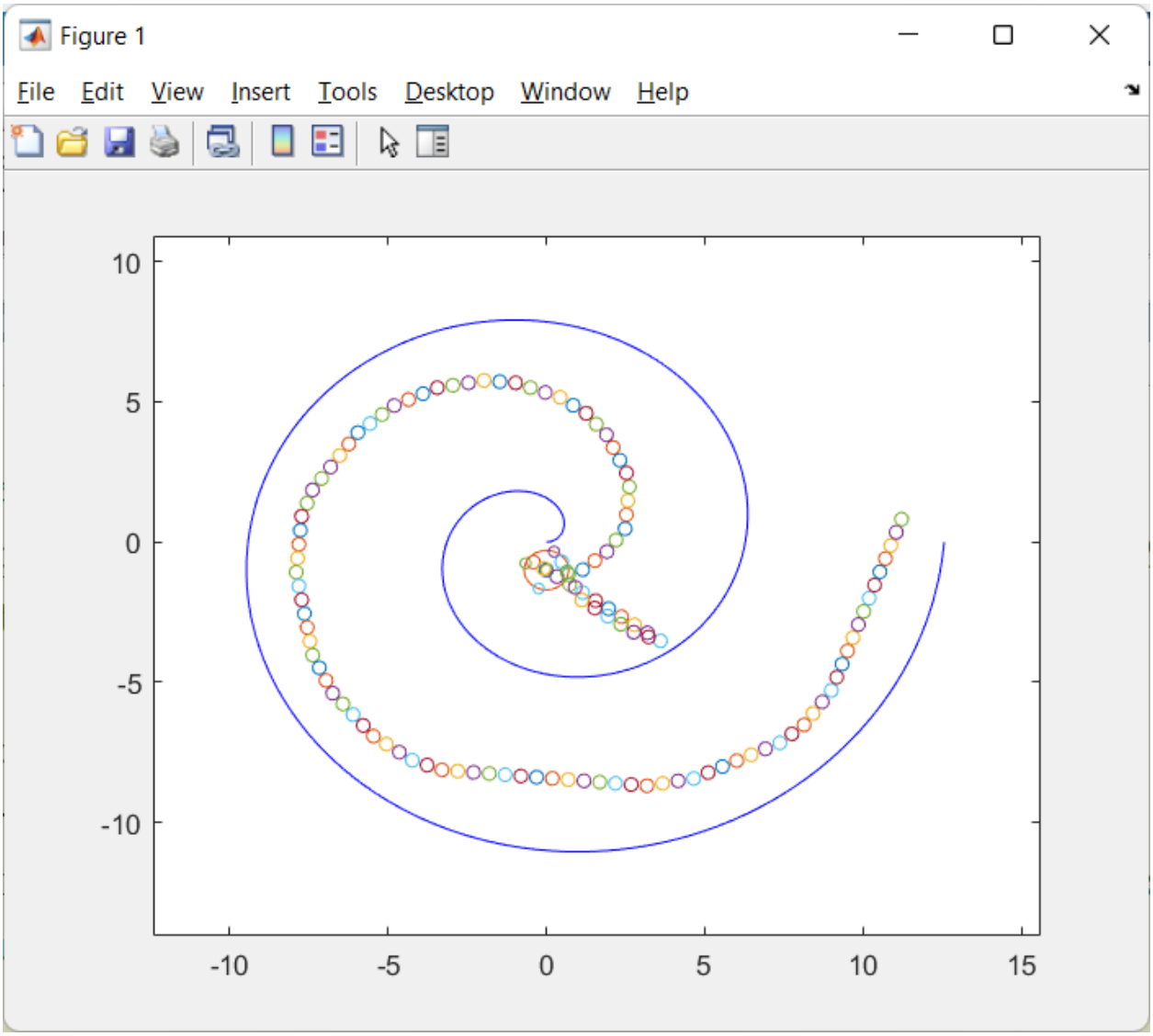


Рисунок 4.4– Шлях роботу з використанням Q-навчання з параметрами кута 70°, початковою точкою (0.0, -1.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.5.

Таблиця 4.5 – Значення Q для кута 70°, точки (0.0, -1.0)

	Дії		
Стани	0.0450000000000000	0	0
	0.0330578100000000	0	0
	17.2695209680933	1.00778502700599	0
	3.15018641027736	0.0175500000000000	0
	0.142751692880162	9.41182295914685	0
	2.26211354307207	-0.0891877954118326	-0.0707307275789237
	0.627815960670967	-0.281587245645000	0
	-0.241868439135729	0.315602346567560	-0.300000000000000
	-0.100000000000000	-0.300000000000000	0
	0	0	0
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 166 точок. Для наочності представлено деякі з них у табл. 4.6.

Таблиця 4.6 – Координати траєкторії переміщення

Координата x	Координата y
0	-1
0	-1
-0.409576022144496	-0.713211781824477
-0.0819152044288992	-0.942642356364895
0.327660817715597	-1.22943057454042
0.737236839860093	-1.51621879271594
1.14681286200459	-1.80300701089146
1.55638888414908	-2.08979522906699
1.96596490629358	-2.37658344724251
2.37554092843808	-2.66337166541803
…	…
9.68440011535390	-3.41128475175558
9.85541018701674	-2.94143844136263
10.0264202586796	-2.47159213096967
10.1974303303424	-2.00174582057672
10.3684404020052	-1.53189951018377
10.5394504736681	-1.06205319979081
10.7104605453309	-0.592206889397857
10.8814706169937	-0.122360579004902
11.0524806886566	0.347485731388052
11.2234907603194	0.817332041781006

На рис. 4.5 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 50°;

– початкова точка розташування робота (0.0, -2.0).

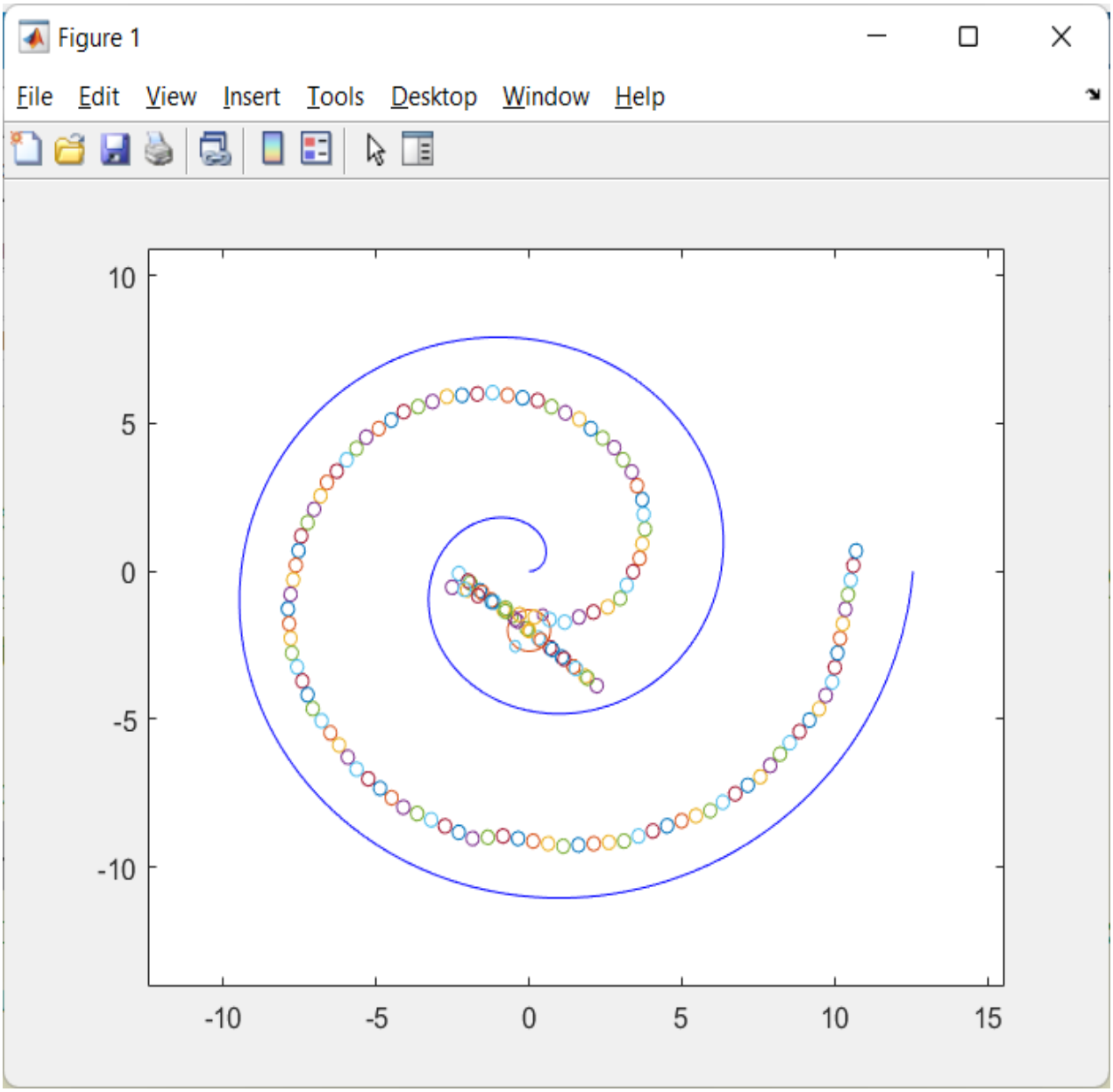


Рисунок 4.5 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (0.0, -2.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.7.

Таблиця 4.7 – Значення Q для кута 50°, точки (0.0, -2.0)

	Дії		
Стани	0	0.194382825362290	0
	0.0650393326080900	0	0
	23.3873161135306	1.22032828821705	7.48615015378449
	4.28168183525216	1.75890383222271	22.0488927990413
	5.06393363042731	22.1649443387632	8.68468502594941
	4.49361449074869	0.219217370650347	-0.541836669222486
	0	0	0
	-0.100000000000000	1.05539333040649	-0.281769362033602
	-0.0748810000000000	-0.813000000000000	0
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 526 точок. Для наочності представлено деякі з них у табл. 4.8.

Таблиця 4.8 – Координати траєкторії переміщення

Координата x	Координата y
0	-2
-0.383022221559489	-1.67860619515673
-0.766044443118978	-1.35721239031346
-1.14906666467847	-1.03581858547019
-1.53208888623796	-0.714424780626922
-1.91511110779745	-0.393030975783652
-2.29813332935693	-0.0716371709403822
-1.99171555210934	-0.328752214814998
-1.60869333054985	-0.650146019658268
-1.22567110899037	-0.971539824501537
9.92069664771547	-3.74094166812272
10.0075207365489	-3.24853779161662
10.0943448253824	-2.75613391511051
10.1811689142159	-2.26373003860441
10.2679930030493	-1.77132616209831
10.3548170918828	-1.27892228559220
10.4416411807163	-0.786518409086098
10.5284652695497	-0.294114532579994
10.6152893583832	0.198289343926110
10.7021134472167	0.690693220432214

На рис. 4.5 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 50°;

– початкова точка розташування робота (-4.0, -2.0).

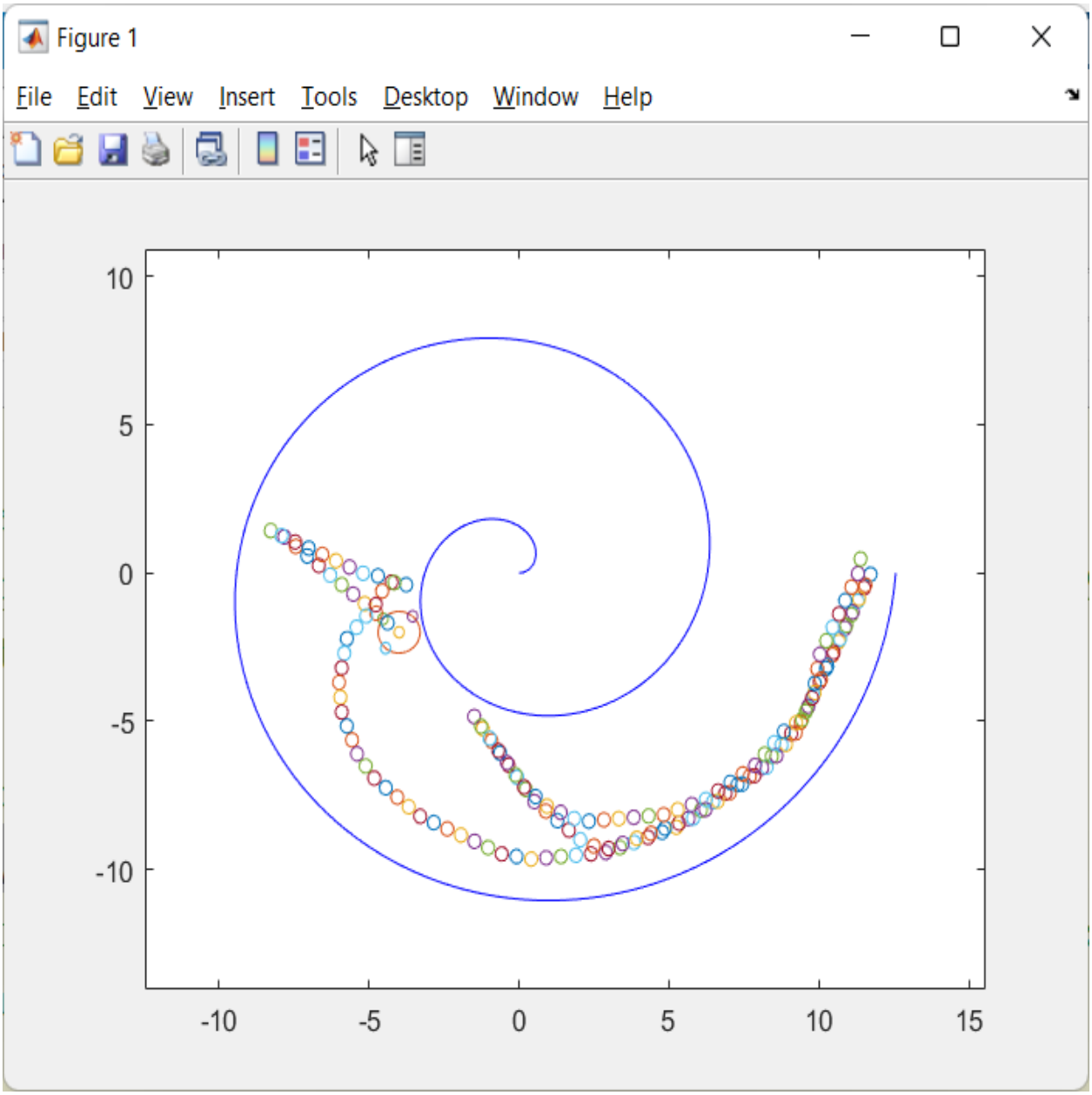


Рисунок 4.6 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (-4.0, -2.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.9.

Таблиця 4.9 – Значення Q для кута 50°, точки (-4.0, -2.0)

	Дії		
Стани	0	0	0
	2.43237804901476	19.8416384990860	3.12804251677455
	15.4098077859821	1.17160715969019	2.33185760332538
	5.23651675595671	1.80060848625343	17.6593923707103
	0.837909364065489	13.9127627168786	2.55072476706425
	5.54878630734741	1.17228417138362	0.682148644519073
	1.74362917590100	0	-0.0854038507171593
	-0.271000000000000	0.851821294835711	-0.300000000000000
	-0.0353638405000000	-0.300000000000000	0.0604601558310664
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 558 точок. Для наочності представлено деякі з них у табл. 4.10.

Таблиця 4.10 – Координати траєкторії переміщення

Координата x	Координата y
-4	-2
-4.38302222155949	-1.67860619515673
-4.76604444311898	-1.35721239031346
-5.14906666467847	-1.03581858547019
-5.53208888623796	-0.714424780626922
-5.91511110779745	-0.393030975783652
-6.29813332935694	-0.0716371709403822
-6.68115555091643	0.249756633902887
-7.06417777247592	0.571150438746157
-7.44719999403541	0.892544243589427
10.0369414221305	-2.74088582227206
10.0369414221305	-2.74088582227206
10.2482505530008	-2.28773192875374
10.4595596838712	-1.83457803523541
10.6708688147415	-1.38142414171709
10.8821779456119	-0.928270248198760
11.0934870764822	-0.475116354680435
11.3047962073525	-0.0219624611621102
11.3047962073525	-0.0219624611621102
11.3916202961860	0.470441415343994

На рис. 4.6 представлено шлях агенту з початковими умовами, такими як:

– початковий кут повороту робота 50°;

– початкова точка розташування робота (-6.0, -2.0).

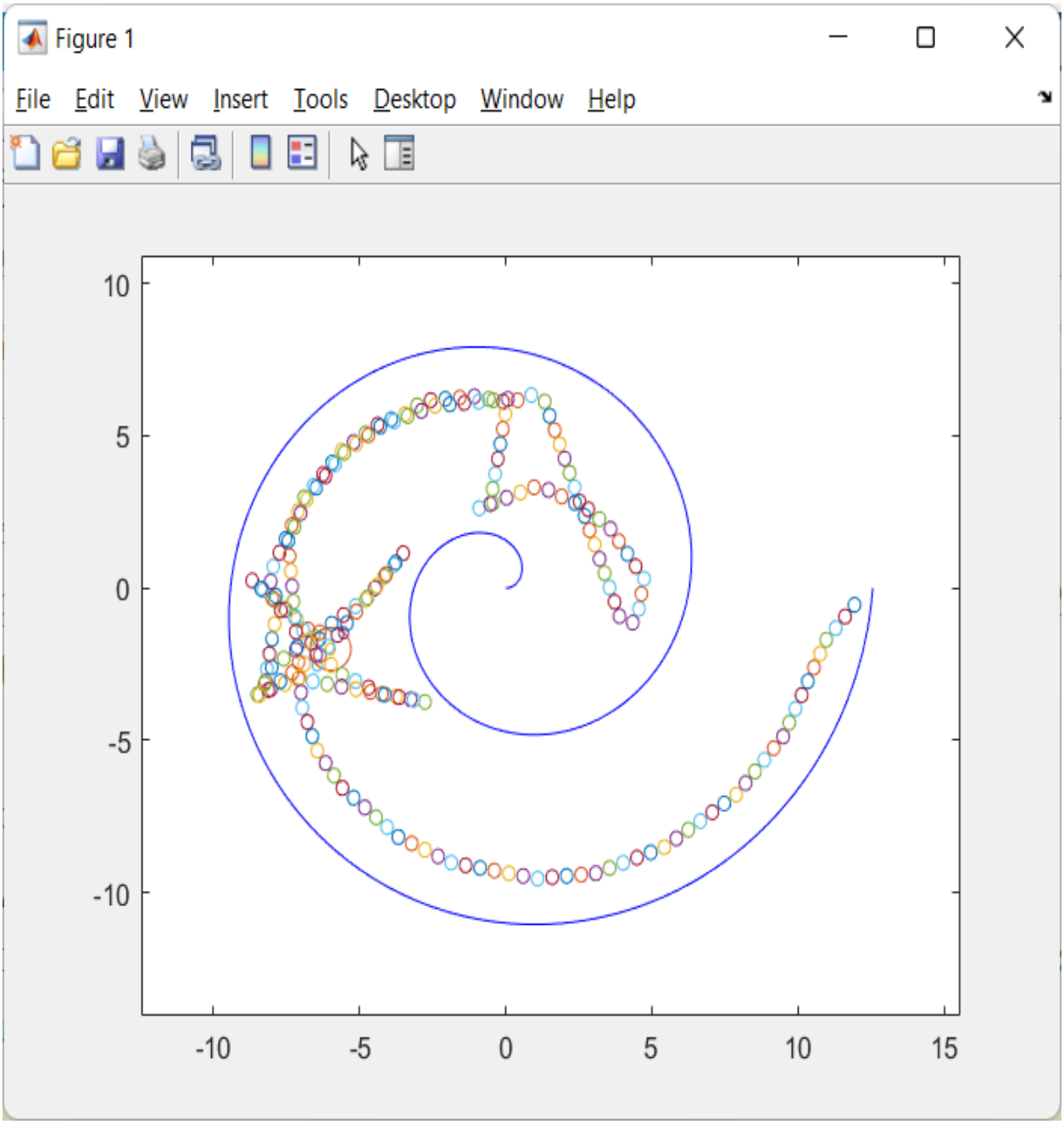


Рисунок 4.6 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (-6.0, -2.0)

Значення параметру Q мають значення (12 станів: 6 нездоров’я і 6 здоров’я/3 дії), записані у табл. 4.11.

Таблиця 4.11 – Значення Q для кута 50°, точки (-6.0, -2.0)

	Дії		
Стани	0.0171000000000000	0.781156766323702	0
	0.320889080106129	1.24130821788510	0
	13.3399037973426	1.01601370726382	0.613577893653769
	-0.300000000000000	1.69303222358303	0.296993689458990
	1.07163448769117	7.85823063324068	0
	1.65766028730240	0.313107099246907	0.240899697656432
	0.0501314016631813	0	-0.00506855721730222
	-0.146067562263126	1.60265812719952	0
	-0.100000000000000	-0.468321601317878	0.112110915151235
	0	0	0
	0	0	0
	0	0	0

Траєкторія переміщення має в своєму складі 309 точок. Для наочності представлено деякі з них у табл. 4.12.

Таблиця 4.12 – Координати траєкторії переміщення

Координата x	Координата y
-6	-2
-6.38302222155949	-1.67860619515673
-6.76604444311898	-1.35721239031346
-7.14906666467847	-1.03581858547019
-7.53208888623796	-0.714424780626922
-7.91511110779745	-0.393030975783652
-8.29813332935694	-0.0716371709403822
-8.68115555091643	0.249756633902887
-8.37473777366884	-0.00735840997172832
-7.99171555210935	-0.328752214814998
…	…
9.91964386775557	-3.96423653414183
10.1309529986259	-3.51108264062351
10.3422621294963	-3.05792874710518
10.5535712603666	-2.60477485358686
10.7648803912370	-2.15162096006853
10.9761895221073	-1.69846706655021

Координата x	Координата y
10.9761895221073	-1.69846706655021
11.2975833269506	-1.31544484499072
11.6189771317939	-0.932422623431231
11.9403709366371	-0.549400401871742

Також було збільшено кількість епізодів навчання до 2-4. Завдяки цьому отримали такий шлях агенту з початковими умовами (рис. 4.7-4.9), такими як:

– початковий кут повороту робота 50°;

– початкова точка розташування робота (-6.0, -2.0).

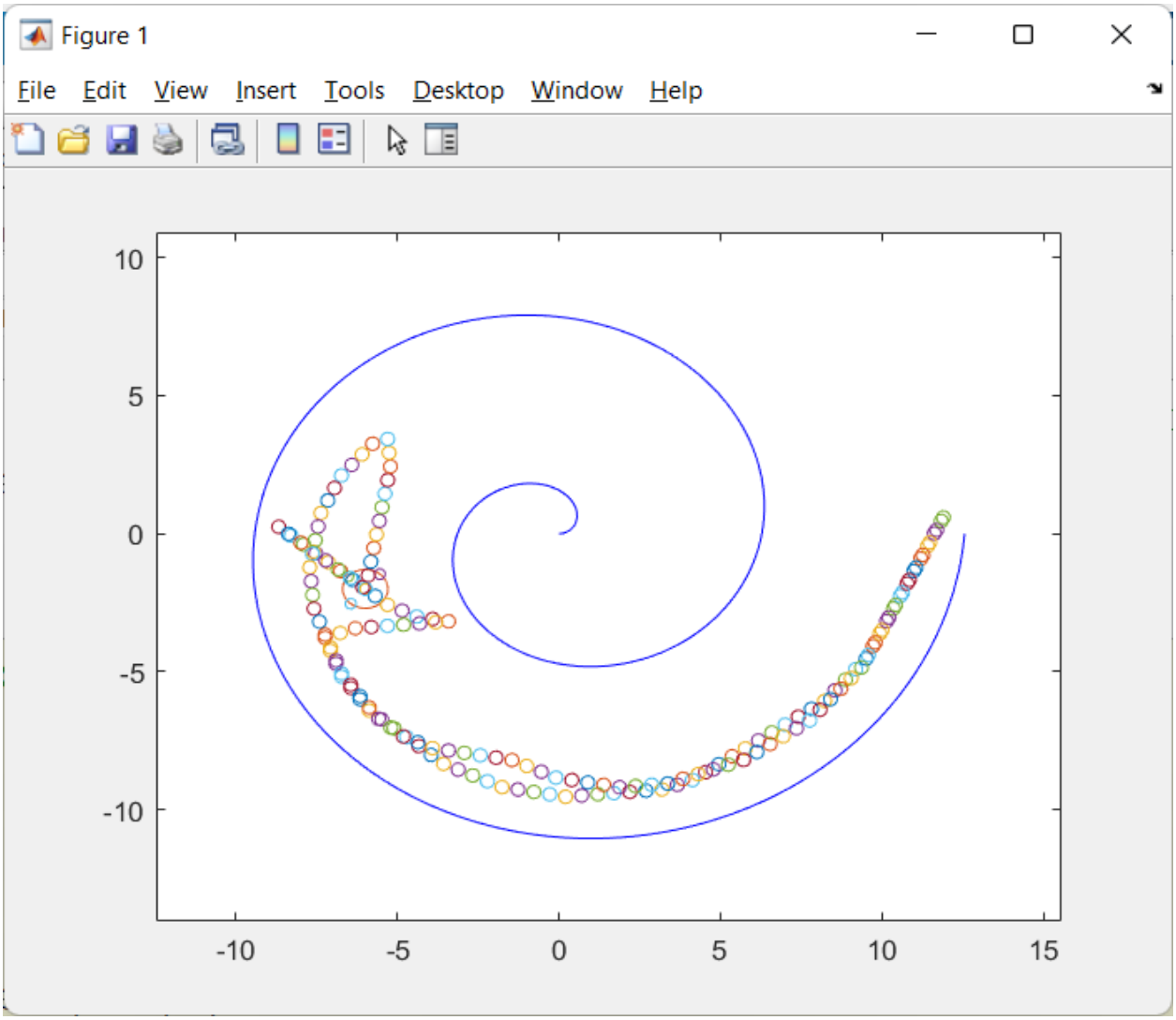


Рисунок 4.7 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (-6.0, -2.0) для 2 епізодів навчання

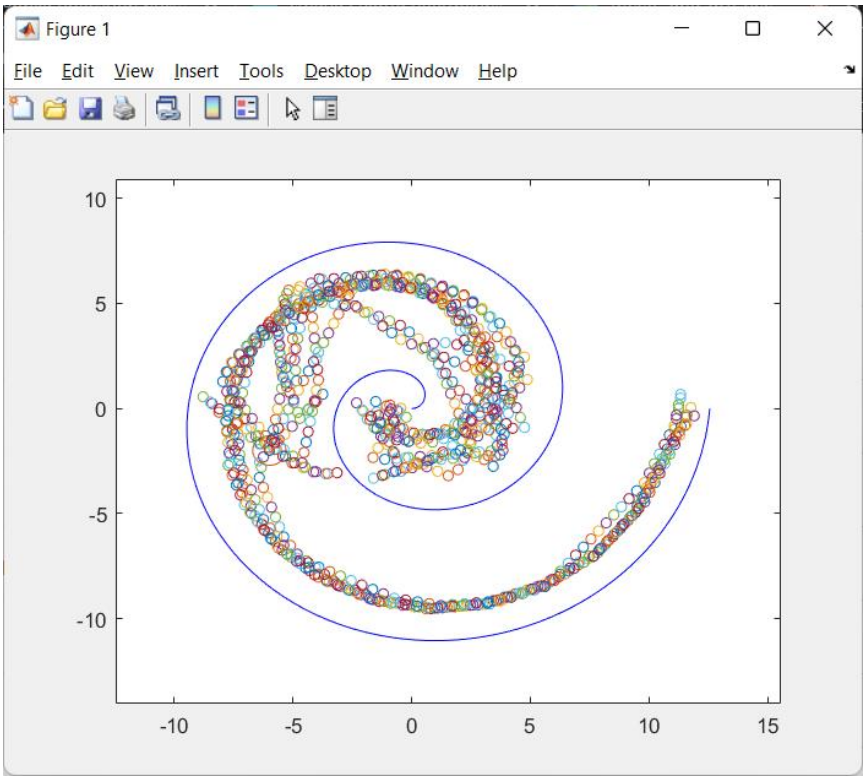


Рисунок 4.8 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (-6.0, -2.0) для 3 епізодів навчання

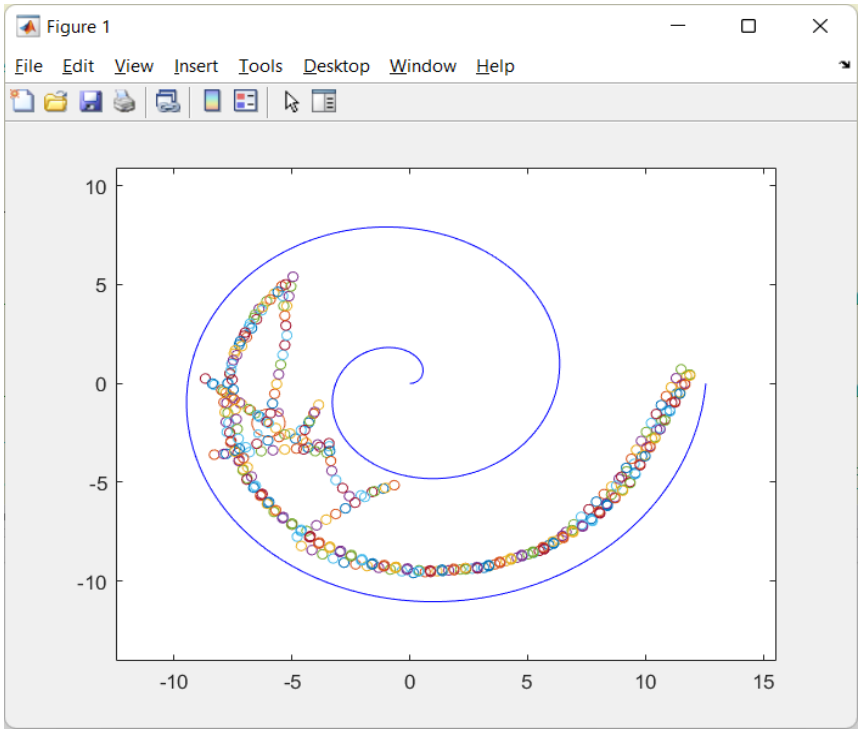


Рисунок 4.9 – Шлях роботу з використанням Q-навчання з параметрами кута 50°, початковою точкою (-6.0, -2.0) для 4 епізодів навчання

4.4 Висновки до розділу 4

Можна зробити висновок, що політика уникнення перешкод, заснована на Q-навчанні, дозволяє агенту успішно вийти з лабіринту. Виконання завдання займає змінний час від ітерації до ітерації. Відповідно до досвіду моделювання, агенту може знадобитися від кількох секунд до кількох хвилин, щоб пройти від заданої початкової точки до кінця лабіринту. Це для функції значення дії Q, ініціалізованої нульовим значенням та епсілон 0,1 (що дозволяє досліджувати дію у 10 % випадків та використовувати її у 90 % випадків).

Описаний вище підхід був змодельований у середовищі Matlab зі спіральним лабіринтом.

Шлях стає гладкішим у міру збільшення кількості епізодів. Вихідна відстань датчиків моделюється шляхом взяття евклідової відстані від перетину з лабіринтом.

5 ОХОРОНА ПРАЦІ

5.1 Пожежна безпека у комп'ютерній лабораторії

Відповідно до норм НАПБ Б.03.002-2007 за ступенем пожежо-вибухонебезпечності дане виробництво відноситься до категорії В. Згідно з ДБН В.1.1.7-2002 будівля відноситься до II ступеня вогнестійкості. Згідно з НПАОП 40.1-1.01-97 приміщення з пожежонебезпечності належить до класу П-Iіа [21].

Причиною пожежі у приміщенні можуть бути коротке замикання електропроводки, несправність електрообладнання, нагрівання провідників, куріння у непризначених для цього місцях, порушення правил пожежної безпеки та підвищена температура всередині приміщення.

Для запобігання пожежам пропонується проводити обов'язковий інструктаж з пожежної безпеки, стежити за доступністю та терміном придатності засобів пожежогасіння, виконувати встановлений режим експлуатації електричних мереж та обладнання, не курити у приміщеннях.

Відповідно до НАПБ Б.03.001 – 2004 у приміщенні розташовані первинні пристрої пожежогасіння – два вуглекислотні вогнегасники ВВК-1, 4 з розрахунку 1 вогнегасник на 3 ПК, але не менше 1 на приміщення. Їх використання обумовлене необхідністю гасіння електроустановок, які перебувають під напругою трохи більше 1000 В, розташованих у приміщенні. Також у кімнаті необхідний один димовий аналізатор відповідно до ДБН В.2.5-56:2010. Біля виходу з кімнати розташований план евакуації у разі екстреної ситуації.

5.2 Інструктаж з охорони праці на робочому місці

При виконанні робіт працівник перебуває за ПК та відстежує пересування маніпулятора робота за допомогою програмного забезпечення. При цьому працівнику доступно управління маніпулятором за допомогою ПК, а також огляд маніпулятора визначення його стану роботи.

До проведення робіт допускаються всі працівники, які досягли 18 років та навчені управлінню ПК на базовому рівні користувача. До огляду роботи маніпулятора робота допускаються робітники зі знаннями механіки та після проведення спеціального інструктажу.

Для захисту від шуму маніпулятора робочі місця перенесені в окрему кімнату, покриту шумопоглинаючими речовинами. Під час роботи з ПК встановлені спеціальні місця для зручного використання.

Вимоги безпеки перед початком робіт.

Робота починається з перевірки справності працездатності маніпулятора та програмного забезпечення на ПК.

Вимоги безпеки під час роботи.

Для безпечної роботи потрібно перебувати у відведеному приміщенні сидячи за ПК, за допомогою якого здійснюється все керування. За потреби можна оглянути роботу маніпулятора, при цьому не слід заходити за зони роботи маніпулятора. Якщо матеріал закінчився, потрібно призупинити роботу з допомогою ПК і зробити заміну матеріалів. Для забезпечення пожежної безпеки в приміщенні встановлено вогнегасник, також евакуаційний вихід знаходиться окремо від місця проведення робіт маніпулятором.

Вимоги безпеки після закінчення робіт.

Після закінчення робіт слід зупинити роботу маніпулятора. Для забезпечення гігієнічних норм слід оглянути робоче місце на наявність пилу та інших забруднюючих речовин та у разі їх наявності – усунути.

ВИСНОВКИ

У першому розділі було проведено аналіз літератури та виявлено, що навігаційне завдання для автономного робота у невідомому середовищі є складним завданням. У роботі пропонується підхід, який ґрунтується на прийнятті рішень, а саме навчанні з підкріпленням, щоб допомогти роботу вийти з невідомого лабіринту.

Основна ідея роботизованої навігації полягає в тому, щоб знайти шлях без зіткнень з початкового стану до цільового стану. Для цього робот повинен збирати інформацію з навколишнього середовища за допомогою датчиків та приймати навігаційні рішення на основі сенсорних даних та своєї бази знань.

У другому розділі роботи було обрано основні модулі робота, побудовано функціональну схему робота зі схематичним розташуванням обраних модулів, а також здорові та нездорові стани переміщення і отримано їх визначення з послідуючим визначенням дій для цих станів, а також визначено значення винагороди за переміщення. Було отримано алгоритм процесу навчання з урахуванням алгоритму Q-навчання.

В результаті написання 3 розділу було розроблено програмний засіб прийняття рішень для керування автоматизованою роботизованою мобільною платформою. Апаратна реалізація має деякі обмеження, здебільшого пов'язані з обмеженнями апаратних компонентів.

Крім того, замкнутий контур керування положенням двигунів постійного струму забезпечить більш точне виконання дій.

Політика уникнення перешкод, заснована на Q-навчанні, дозволяє агенту успішно вийти з лабіринту. Виконання завдання займає змінний час від ітерації до ітерації. Відповідно до досвіду моделювання, агенту може знадобитися від кількох секунд до кількох хвилин, щоб пройти від заданої початкової точки до кінця лабіринту. Це для функції значення дії Q, ініціалізованої нульовим значенням та епсілон 0,1 (що дозволяє досліджувати дію у 10 % випадків та використовувати її у 90 % випадків).

Описаний вище підхід був змодельований у середовищі Matlab зі спіральним лабіринтом.

Шлях стає гладкішим у міру збільшення кількості епізодів. Вихідна відстань датчиків моделюється шляхом взяття евклідової відстані від перетину з лабіринтом.

Для досягнення було вирішено такі завдання:

– проведено аналіз систем штучного інтелекту та прийняття рішень, що можуть бути використані при керуванні роботизованою мобільною платформою;

– проведено вибір основних елементів системи керування роботизованою мобільною платформою;

– розроблено програмний засіб для прийняття рішень та керування роботизованою мобільною платформою мовою Python;

– проведено моделювання та експериментальні дослідження з розробленим програмним засобом для прийняття рішень та керування роботизованою мобільною платформою у середовищі MatLab.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. 31 с.

3. Karthikeyan J Learning outcomes of classroom research / J. Karthikeyan, Ting Su Hie, Ng Yu Jin // Engineering and Technology, L Ordine Nuovo Publication – Tamil Nadu, India, 2021. 463 pp.

4. Bezboruah Tulshi. Artificial intelligence: The Technology, Challenges and Applications. / Tulshi Bezboruah, Abhijit Bora // Transactions on Machine Learning and Artificial Intelligence, Volume 8 No 5, 2020. pp: 44-51.

5. Shevchenko Anatolii. Regarding the Draft Strategy Development of Artificial Intelligence in Ukraine (2022 – 2030) / Anatolii Shavchenko and oth. // Artificial Intelligence, 2022. 74 p.

6. Difference between Artificial intelligence and Machine learning. URL: https://www.javatpoint.com/difference-between-artificial-intelligence-and-machine-learning (дата звернення: 25.11.2022).

7. Yuvalı, M. Classification Comparison of Machine Learning Algorithms Using Two Independent CAD Datasets / M. Yuvalı, B. Yaman, Ö. Tosun // Mathematics, 2022. 15 p.

8. Rosin, F. Enhancing the Decision-Making Process through Industry 4.0 Technologies / F. Rosin, P. Forget, S. Lamouri, R. Pellerin // Sustainability, 2022. 35 p.

9. Sala, R. Data-Driven Decision Making in Maintenance Service Delivery Process: A Case Study / R. Sala, F. Pirola, G. Pezzotta, Cavalieri // Appl. Sci., 2022. 17 p.

10. Litvaj, I. Decision-Making Procedures and Their Relation to Knowledge Management and Quality Management / I. Litvaj, O. Ponisciakova, D. Stancekova, J. Svobodova, J. Mrazik // Sustainability, 2022. 17 p.

11. Цимбал О.М. Методи та моделі інтелектуальної підтримки прийняття рішень у автоматизованому керуванні гнучким інтегрованим радіоелектронним виробництвом.- Дисертація д-ра техн. наук: 05.13.07, Харків. нац. ун-т радіоелектроніки. - Харків, 2015. 390 с.

12. Matthew C. Gombolay, Reymundo A. Gutierrez, Giancarlo F. Sturla, and Julie A. Shah Decision-Making Authority, Team Efficiency and Human Worker Satisfaction in Mixed Human-Robot Teams // Massachusetts Institute of Technology Cambridge, 2018. 9 p.

13. Hofer, Ludovic. Decision-making algorithms for autonomous robots / Ludovic Hofer // Robotics. Université de Bordeaux, 2018. 139 p.

14. Naeem, Muddasar. A Gentle Introduction to Reinforcement Learning and its Application in Different Fields / Muddasar Naeem, Syed Rizvi, Antonio Coronato // IEEE Access, Vol. 8, 2020. 25 p.

15. Miranda B. Combined model-free and model-sensitive reinforcement learning in non-human primates / B. Miranda, W. M. N. Malalasekera, T. E. Behrens, P. Dayan, and S. W. Kennerley // PLOS Comput. Biol., vol. 16, no. 6, 2020. 24 p.

16. Chebotar Y. Combining model-based and model-free updates for trajectory-centric reinforcement learning / Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, // arXiv, 2017. 13 p.

17. Kurdi B., Model-free and modelbased learning processes in the updating of explicit and implicit evaluations / B. Kurdi, S. J. Gershman, and M. R. Banaji, Proc. Nat. Acad. Sci. USA, vol. 116, no. 13, 2019. pp. 6035–6044.

18. Gu S., Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates/ S. Gu, E. Holly, T. Lillicrap, and S. Levine, // Proc. IEEE Int. Conf. Robot. Automat. (ICRA), Singapore, 2017. pp. 3389–3396.

19 P. Wang, A reinforcement learning based approach for automated lane change maneuvers / P. Wang, C. Chan, and A. de La Fortelle, // Proc. IEEE Intell. Vehicles Symp. (IV), Changshu, China, 2018. pp. 1379–1384.

20. Kormushev, Petar Reinforcement Learning in Robotics: Applications and Real-World Challenges / Petar Kormushev, Sylvain Calinon, Darwin Caldwell // Robotics, 2018. pp.122-148.

21. Методичні вказівки до виконання розділу "Охорона праці" у випускних роботах ОКР "бакалавр" усіх форм навчання / упоряд.: В. А. Айвазов. Т. Є. Стиценко., Н. Л. Березуцька ; М-во освіти і науки України, ХНУРЕ. Харків : ХНУРЕ, 2018. 28 с.