Міністерство освіти і науки
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська кваліфікаційна робота на тему:
"Програмний засіб ідентифікації рухомого об'єкта"

Виконав: студент 4 курсу групи
БКС-21б спеціальності 125
«Кібербезпека»
Кирил МАТВЄЄВ

Керівник: к.т.н., професор каф. ЗІ
Наталія КОНДРАТЕНКО
«16» червня 2025 р.

Рецензент: к.т.н., доц. каф. ПЗ
Олена КОВАЛЕНКО
«16» червня 2025 р.

Допущено до захисту

Завідувач кафедри ЗІ

д. т. н., проф.

Володимир ЛУЖЕЦЬКИЙ

«16» червня 2025 р

Вінниця ВНТУ - 2025

Міністерство освіти і науки України Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії Кафедра захисту інформації
Рівень вищої освіти I (бакалаврський)
Галузь знань – 12 Інформаційні технології Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Кібербезпека критичних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри ЗІ, д.т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кирилу МАТВЄЄВУ

1. Тема роботи: «Програмний засіб ідентифікації рухомого об'єкта», керівник роботи: Наталія КОНДРАТЕНКО, к. т. н., проф. кафедри ЗІ, затверджені наказом ректора ВНТУ від 20 березня 2025 року за №97.
2. Строк подання студентом роботи 16 червня 2025 р.
3. Вихідні дані до роботи:
   - галузь застосування: Забезпечення безпеки шляхом превентивно-профілактичних мір протидії несанкціонованому вторгненню;
   - використовуванні інструменти: Windows 11, Pycharm.
4. Зміст текстової частини: Вступ. Аналіз предметної області. Розробка програмного засобу ідентифікації рухомого об'єкту з використанням нечіткої класифікації. Програмна реалізація та тестування класифікації параметрів руху об'єкту типу БПЛА на основі даних відеоспостереження. Висновки. Список використаних джерел. Додатки.
5. Ілюстративна частина: Приклад графіків належності. Алгоритм роботи програми. Оптимізовані графіки належності. Приклад роботи програми. Схема генетичного алгоритму. Алгоритм визначення фінального інтервалу.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Н. КОНДРАТЕНКО, професор, каф.ЗІ	14.03.25	10.06.25
2	Н. КОНДРАТЕНКО, професор, каф.ЗІ	14.05.25	10.06.25
3	Н. КОНДРАТЕНКО, професор, каф.ЗІ	31.05.25	10.06.25

7. Дата видачі завдання 21.03 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	14.03.2025 – 20.03.2025	
2	Аналіз літературних джерел за напрямком БКР	20.03.2025 – 30.03.2025	
3	Розробка рішень, моделей, алгоритмів	30.03.2025 – 19.04.2025	
4	Практична реалізація, моделювання, експериментування, результати	19.04.2025 – 02.05.2025	
5	Оформлення пояснювальної записки	02.05.2025 – 30.05.2025	
6	Попередній захист БКР	30.05.2025 – 07.06.2025	
7	Виправлення зауважень, підготовка ілюстративного матеріалу	07.06.2025 – 07.06.2025	
8	Перевірка на наявність текстових запозичень	09.06.2025 – 10.06.2025	
9	Представлення БКР до захисту, рецензування	10.06.2025 – 18.06.2025	
10	Захист БКР	19.06.2025	

Студент _____ Кирил МАТВЄЄВ

Керівник роботи _____ Наталія КОНДРАТЕНКО

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 82 сторінок формату А4, на яких є 17 рисунків, 4 таблиць, список використаних джерел містить 20 найменувань.

Бакалаврська робота присвячена розробці програмного засобу для ідентифікації та класифікації параметрів руху рухомого об’єкта. В результаті аналізу існуючих способів класифікації обрано такий метод, як класифікація за допомогою нечіткої логіки. Така класифікації передбачає використання графіків належності та математичних інтервальних операцій для визначенну фінального класу параметрів об’єкта. Після розробки схем функціонування програмного засобу в цілому і алгоритмів його окремих складових здійснено програмну реалізацію за допомогою технологій програмування.

Ключові слова: класифікація, параметри руху, графіки належності, інтервали, нечітка логіка, рухомий об’єкт, генетичний алгоритм, лінгвістична ознака, база правил

ABSTRACT

The bachelor's thesis consists of 85 A4 pages, which have 17 figures, 5 tables, the list of sources used contains 20 items.

Baccalaureate work is devoted to the development of software for identifying and classifying the motion parameters of a moving object. As a result of analyzing existing classification methods, a method such as classification using fuzzy logic was chosen. This classification involves the use of membership graphs and mathematical interval operations to determine the final class of object parameters. After developing the overall functioning of the software tool and the algorithms of its individual components, the software was implemented using programming technologies.

Keywords: classification, motion parameters, membership graphs, intervals, fuzzy logic, moving object, genetic algorithm, linguistic feature, rule base

ЗМІСТ

ВСТУП

Моніторинг рухомих об’єктів за допомогою відеоспостереження є важливим елементом сучасних систем безпеки. Особливої уваги потребує виявлення та класифікація об’єктів типу безпілотних літальних апаратів (БПЛА), які можуть становити серйозну загрозу для об’єктів критичної інфраструктури, державних установ, промислових підприємств або приватної власності.

Захист таких об’єктів від несанкціонованого проникнення потребує застосування ефективних методів виявлення і аналізу руху. Для цього використовуються превентивно-профілактичні заходи, системи дрон-детекції, а також засоби нейтралізації БПЛА. Однією з перспективних технологій для автоматичного аналізу динаміки руху об’єкта є використання методів інтервального нечіткого моделювання, що дозволяють здійснювати точну класифікацію параметрів руху навіть в умовах невизначеності або часткової втрати даних.

Такі моделі враховують критичні параметри руху об’єкта, що мають важливе значення для забезпечення кібербезпеки. Відеоспостереження виступає джерелом первинних даних, які надалі обробляються для виявлення потенційно небезпечних рухомих об’єктів та визначення їхніх характеристик.

Метою бакалаврської кваліфікаційної роботи є підвищення якості моніторингу контрольованої зони об’єкта критичної інфраструктури шляхом створення програмного засобу класифікації параметрів руху рухомого об’єкта на основі даних відеоспостереження. У рамках роботи було передбачено вирішення кількох ключових завдань. Насамперед — ознайомлення з підходами до нечіткого та інтервального моделювання, що дозволяють враховувати невизначеність у вхідних даних. Далі — побудова математичної моделі для класифікації параметрів руху об’єктів на основі відповідних ознак. Завершальним етапом стало створення програмного засобу, призначеного для моніторингу та аналізу відеоданих із застосуванням розробленої моделі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загроза рухомих об’єктів типу БПЛА для кібербезпеки

Останні роки характеризуються стрімким зростанням кількості інцидентів, пов’язаних із проникненням безпілотних літальних апаратів (БПЛА) у контрольований або обмежений повітряний простір, що суттєво підвищує рівень загроз для об'єктів критичної інфраструктури, державних установ та громадської безпеки. Така динаміка актуалізує потребу у перегляді існуючих підходів до протидії дронам та розробці нових комплексних рішень, здатних ефективно реагувати на широкий спектр потенційних загроз.

Аналіз зафіксованих випадків порушення повітряного простору з боку БПЛА дозволяє виявити основні недоліки в наявних системах виявлення та нейтралізації таких апаратів. Ці інциденти свідчать про необхідність удосконалення як технічних засобів, так і процедурних механізмів реагування. Особливої уваги потребують новітні форми застосування дронів, зокрема автономні ройові атаки, використання штучного інтелекту для навігації, а також нелінійні методи доставки вантажів або вибухових пристроїв.

У цьому контексті питання ефективності систем протидії БПЛА набуває не лише технологічного, а й стратегічного значення. Сучасна парадигма безпеки передбачає не лише реагування на вже здійснений інцидент, а й формування системи завчасного виявлення та адаптивного управління ризиками. Отже, першочерговим стає не питання ймовірності вторгнення, а здатність системи оперативно і точно відреагувати на загрозу.

Однією з ключових загроз, які постають у зв’язку з поширенням безпілотних літальних апаратів (БПЛА), є їх здатність здійснювати приховане спостереження за фізичними особами та організаціями. Завдяки компактним розмірам і високій мобільності, дрони можуть бути оснащені засобами аудіозапису, здатними вловлювати розмови на значній відстані без візуального контакту. Технології, що забезпечують чітке захоплення звуку, постійно вдосконалюються, а самі пристрої –

доступні на відкритому ринку, що створює реальні передумови для несанкціонованого збору конфіденційної інформації.

Не менш значущою є загроза, пов’язана з використанням БПЛА для фото- та відеозйомки. Масове використання дронів у сфері дозвілля, журналістики та комерційних зйомок призвело до нормалізації практики повітряного спостереження, що водночас створило можливість для його зловмисного застосування. Завдяки здатності наближатися до приватних або охоронюваних об'єктів без привернення уваги, дрони можуть фіксувати зображення, які містять особисту, корпоративну або державну таємницю. Такий візуальний контент потенційно може бути використаний для шантажу, викрадення інтелектуальної власності або підготовки до фізичного вторгнення.

Окрему категорію становлять загрози кіберфізичного характеру. Сучасні дрони можуть бути платформами для реалізації атак на цифрову інфраструктуру: наближення до локальних мереж, серверів або пристроїв з бездротовими інтерфейсами відкриває можливості для несанкціонованого підключення, впровадження шкідливого програмного забезпечення чи зчитування даних. Такі атаки особливо небезпечні для об’єктів, що обробляють або зберігають критичну інформацію, оскільки компрометація інформаційних систем може призвести до значних операційних і фінансових втрат, а також до порушення національної безпеки.

У грудні 2024 року в окрузі Салем (штат Нью-Джерсі) було зафіксовано проліт безпілотного літального апарата поблизу об’єктів критичної інфраструктури, зокрема атомних електростанцій компанії PSE&G, що викликало занепокоєння щодо потенційного несанкціонованого спостереження або розвідки. Подібна активність свідчить про вразливість стратегічно важливих енергетичних об’єктів до технологічного моніторингу з повітря [1].

Інший випадок стався 6 грудня 2024 року поблизу водосховища Раунд-Веллі в населеному пункті Клінтон Тауншип (Нью-Джерсі), де було зафіксовано підозрілу активність дронів у безпосередній близькості до гідротехнічної інфраструктури. Така поведінка літальних апаратів викликала побоювання щодо можливих спроб збору

інформації або підготовки до деструктивних дій, враховуючи важливість водних об’єктів для регіонального забезпечення [1].

5 грудня 2024 року на території Стейтен-Айленда (Нью-Йорк) було зафіксовано серію польотів невідомих дронів поблизу мосту Гьоталс. Незрозумілий характер цієї активності змусив місцеву владу звернутися за допомогою до федеральних агентств, що підкреслює загальнонаціональне значення загроз, пов’язаних із використанням БПЛА в урбанізованому середовищі [1].

Такі випадки є не рідкісними у нашому світі, а це підвищує степінь важливості розробки рішень для протидії.

1.2 Основні параметри руху об’єкту для відеоспостереження

Безпілотні літальні апарати (БПЛА), відомі також як дрони, є однією з найактуальніших сучасних технологій. Це літальні пристрої, які не потребують присутності пілота на борту [2]. Завдяки стрімкому технологічному розвитку та широким можливостям застосування, дрони відкривають нові горизонти в різних сферах діяльності.

Безпілотники відрізняються за конструкцією та розмірами. Сучасні БПЛА мають вражаючі технічні характеристики: вони здатні здійснювати автономні польоти на значні відстані, оснащені системами автоматичного керування, GPS-навігацією, модулями уникнення перешкод, акумуляторним живленням та інтегрованими системами керування, що робить їх використання зручним та ефективним.

Прикладом сучасного безпілотного літального апарата з високим технічним потенціалом є модель DJI FPV. Цей дрон, створений однією з провідних приватних компаній Китаю, яка спеціалізується на виробництві БПЛА, систем керування польотом і обладнання для стабілізації відеозйомки, вирізняється інноваційними конструктивними та функціональними рішеннями. Основні технічні характеристики DJI FPV наведено в таблиці 1.1 [3]. Їх аналіз свідчить про те, що дана модель належить до високотехнологічних, багатофункціональних пристроїв, здатних виконувати

широкий спектр завдань. Висока маневровість досягається завдяки компактному форм-фактору та невеликій масі дрона. Максимальна швидкість польоту, яка сягає 140 км/год, а також можливість підйому на висоту до 6000 метрів, ускладнюють його виявлення та перехоплення засобами традиційного спостереження або протидії.

Таблиця 1.1 – Технічні характеристики DJI FPV

Характеристика	Значення
Розмір	255 x 312 x 127 мм
Вага	795 г (з акумулятором)
Максимальна швидкість	140 км/год (в режимі Manual)
Максимальна висота польоту	6000 метрів
Час польоту	до 20 хвилин
Діапазон робочих частот	2,4-2,48 ГГц 5,725-5,850 ГГц
Максимальна дальність польоту	до 16.8 км (залежить від умов)
Сенсор камери	1/2.3" CMOS, 12 МП
Об'єктив камери	FOV: 150 , діафрагма: f/2.8
Відео	4K/60fps, 1080p/120fps
Стабілізація камери	Електронна стабілізація зображення (EIS)
Система передачі відео	DJI O3 (OcuSync 3.0)
Дальність передачі відео	До 10 км (FCC), до 6 км (CE/SRRC/MIC)
Тип акумулятора	LiPo 6S
Ємність акумулятора	2000 мА·годин

Відносно великий час польоту для такого типу БпЛА (до 20 хвилин), а також велика дальність польоту (до 16.8 км), дозволяють виконувати доволі тривалі польоти.

Головним фактором є, що на швидкості 140км/год об’єкту, людське око не зможе розпізнати його і яку загрозу він представляє. Натомість, є можливість створення методу розпізнавання об’єкта за допомогою камери.

Під час супроводу або спостереження за об’єктом на екрані можуть відображатися критично важливі параметри, що забезпечують контроль, точність навігації та аналіз ефективності бойового завдання. Ці параметри умовно поділяються на позиційні, геометричні, візуальні та сигнальні.

До ключових позиційних параметрів належать географічні координати, що включають широту та довготу і визначають точне місцезнаходження дрона або цілі. Це дає змогу координувати дії з артилерією, уточнювати положення об’єкта для корекції траєкторії польоту об’єкту, а також передавати точні дані іншим підрозділам у реальному часі. Важливим параметром є висота над рівнем моря або над землею, яка враховується для зменшення ризику зіткнення з перешкодами, зокрема на пересіченій місцевості. Швидкість руху дрона або об’єкта дозволяє оцінити динаміку цілі та її характер – стаціонарна вона чи мобільна.

Окрім координат, система візуального спостереження часто обробляє геометричні характеристики об’єкта на зображенні. Визначення площі об’єкта у пікселях або відносних одиницях дозволяє судити про його розмір у кадрі та дає можливість відстежувати зміни положення або наближення/віддалення цілі. Ці параметри є основою для автоматизованого розпізнавання типу об’єкта, такого як танк, бронетранспортер або автомобіль. Також візуальна система може аналізувати контурну площу або периметр об’єкта, що застосовується в алгоритмах комп’ютерного зору для точного розпізнавання форм. Координати центроїда об’єкта в межах кадру (координати X і Y) дозволяють точно визначити його положення для візуального супроводу або використання функції автозахоплення.

Візуальні параметри є невід’ємною частиною спостереження. Поточне відео з дрона може передаватися як у вигляді «від першої особи», так і з боку, що дає оператору можливість повноцінно контролювати обстановку. Оптичне збільшення та кут огляду камери суттєво впливають на рівень деталізації зображення. Тип зображення – RGB чи тепловізійне – залежить від типу сенсора і вибирається відповідно до умов спостереження, наприклад, для виявлення вночі або крізь димову завісу.

1.3 Основні підходи до задач класифікації

Класифікація – це процес розподілу об’єктів, явищ або процесів на групи за певними ознаками з метою впорядкування та спрощення їхнього аналізу. Вона широко використовується в різних сферах діяльності – науці, економіці, медицині, техніці, сільському господарстві, промисловості. Суть класифікації полягає у виявленні подібностей між об’єктами та об’єднанні їх у групи на основі спільних характеристик.

Процес класифікації базується на чітко визначених правилах: при кожному поділі слід враховувати лише одну основу; обсяг отриманих підгруп має відповідати обсягу початкового поняття; підгрупи не повинні перетинатися; а сам поділ має бути логічно послідовним. За характером ознак класифікація може бути штучною, якщо вона базується на зовнішніх, формальних характеристиках, або природною, коли враховуються внутрішні, суттєві ознаки, що вказують на глибші закономірності.

Залежно від підходу класифікація буває простою, коли об’єкти поділяються за однією ознакою, або складною, коли враховується кілька основ і комбінується кілька поділів. У прикладних задачах класифікація часто використовується для прогнозування значення категоріальної змінної. Це може бути бінарна класифікація, коли результат має два можливі варіанти, або мультикласова – коли передбачуваний результат належить до одного з кількох можливих класів.

Класифікація може здійснюватися за однією ознакою (одновимірна) або за кількома одночасно (багатовимірна). Прикладами практичного застосування класифікації є визначення кредитоспроможності клієнтів, діагностика захворювань, розпізнавання образів, поведінковий аналіз відвідувачів вебсайтів.

Процес класифікації базується на фундаментальних концепціях подібності та відмінності між об’єктами. Методологічною основою класифікаційного підходу є два ключові положення: по-перше, до одного класу відносяться елементи, що мають спільні або схожі характеристики; по-друге, рівень подібності між об’єктами в межах одного класу має бути вищим, ніж між об’єктами, що належать до різних класів.

Оцінювання подібності відбувається з урахуванням однієї або кількох ознак, які, на думку фахівців, є визначальними для формування характерного «образу» класу. У класичному підході до класифікації ці ознаки мають ієрархічну структуру залежно від їхньої значущості. Поділ множини об’єктів здійснюється поетапно – на кожному рівні класифікації враховується лише одна ознака, що забезпечує послідовну деталізацію структури класів. Така схема ефективна для невеликих вибірок, однак її застосування в масштабніших або складніших системах є обмеженим.

Альтернативний підхід передбачає одночасне використання кількох класифікаційних ознак. У цьому випадку кожен об’єкт розглядається як точка в багатовимірному просторі ознак, а ступінь подібності між об’єктами визначається на основі їхньої просторової близькості. Реалізація цього підходу можлива у двох основних формах: перша полягає у створенні комплексних оцінок (індексів або рейтингів), що згодом використовуються в традиційній класифікації; друга – у застосуванні методів кластерного аналізу, де критерії однорідності формалізуються через вибрані метрики.

Необхідно наголосити, що незалежно від застосованої схеми, будь-яка класифікація має певний ступінь суб’єктивності, оскільки її результат значною мірою залежить від обраних ознак і способу їх інтерпретації як розмежувальних критеріїв.

Кластеризація [4] — це метод аналізу даних, що полягає у групуванні об'єктів дослідження на основі їх подібності або відстані між ними. Основна ідея цього підходу полягає в тому, щоб розподілити множину різнорідних об’єктів на відносно однорідні підгрупи, які називаються кластерами. Кожен кластер об’єднує елементи, що максимально схожі один на одного за низкою характеристик, при цьому відмінність між об’єктами з різних кластерів є суттєвою. Такий підхід широко використовується в різних галузях — від біоінформатики, маркетингу й соціології до комп’ютерного зору та машинного навчання.

Кластерний аналіз є невід'ємною частиною неконтрольованого навчання, адже на відміну від класифікації, тут не вимагається попередньо маркованих даних. Алгоритм самостійно виявляє приховані структури у даних, спираючись на внутрішні

зв’язки між об’єктами. Ключовим етапом такого аналізу є визначення міри схожості або відмінності між елементами. Залежно від типу ознак (числові, категоріальні, бінарні), можуть використовуватись різні метрики — евклідова відстань, мангеттенська, косинусна подібність, коефіцієнт Жаккара та інші.

На основі числових значень ознак для кожної пари об'єктів формується матриця відстаней, яка є симетричною та дозволяє оцінити ступінь близькості між усіма елементами вибірки. Чим менша відстань між двома об’єктами, тим вища їх схожість. У разі застосування бінарних ознак (наприклад, "є/немає", "так/ні"), використовуються спеціальні коефіцієнти подібності, які враховують кількість однакових та протилежних значень.

Методи кластеризації поділяються на дві основні категорії: ієрархічні та ітераційні (розділові). Ієрархічні підходи формують структуру кластерів шляхом послідовного злиття або розділення груп. Зазвичай реалізується агломеративний підхід, який починається з того, що кожен об'єкт вважається окремим кластером. На кожному наступному кроці об'єднуються ті кластери, між якими найменша відстань, до моменту, поки не буде досягнуто певного критерію зупинки — наприклад, кількість кластерів або відстань між ними.

Результатом агломеративного аналізу є дендрограма — дерево подібності, де гілки вказують, які об’єкти об’єднувалися та на яких рівнях відстані це відбувалося. Аналізуючи дендрограму, дослідник може візуально визначити логічні межі між групами, відсікаючи дерево на певному рівні. Часто встановлюється поріг відстані, перевищення якого забороняє подальше об’єднання кластерів. Це дає змогу контролювати ступінь внутрішньої однорідності кластерів і запобігає формуванню надто загальних груп.

Однак, незважаючи на свою зрозумілість і візуальну привабливість, ієрархічні методи мають певні обмеження. Вони погано масштабуються при роботі з великою кількістю об’єктів, оскільки потребують обчислення повної матриці відстаней та складної обробки деревоподібної структури.

Ітераційні методи зазвичай не потребують побудови повної матриці відстаней і працюють безпосередньо з вхідними даними. Вони починають із випадкового або заданого розподілу центрів кластерів, після чого в кілька кроків переміщують ці центри до положень, які мінімізують внутрішньокластерну варіацію. Результати таких методів можуть відрізнятись при різних запусках, однак вони значно ефективніші при роботі з великими наборами даних.

У підсумку, вибір між ієрархічними та ітераційними підходами до кластеризації залежить від поставленої задачі, розміру вибірки, типу ознак і вимог до інтерпретації результатів. У деяких випадках ці методи навіть поєднують, використовуючи ієрархічний підхід для попередньої оцінки кількості кластерів, а ітераційний — для уточнення їх меж.

Серед методів розпізнавання образів дискримінантний аналіз займає особливе місце. Його головною метою є не створення нових груп, як у випадку кластерного аналізу, а виявлення відмінностей між уже відомими класами та визначення, до якого з них належить новий об’єкт, базуючись на принципі максимальної подібності. Цей підхід широко застосовується, наприклад, у медичній діагностиці або при оцінюванні ймовірності відмови технічних пристроїв. Основним завданням при цьому є мінімізація ймовірності помилки класифікації.

Дискримінантний аналіз [5] широко використовується в практичних задачах, зокрема у медичній діагностиці, фінансовій аналітиці, прогнозуванні технічних збоїв, контролі якості, а також в соціальних науках, де потрібно розподілити об’єкти (наприклад, пацієнтів, клієнтів, пристрої) за класами на основі виміряних параметрів. Його застосування забезпечує прогнозування, оцінку ризиків і виявлення аномалій, що є критично важливим у системах підтримки прийняття рішень.

Суть методу полягає у побудові спеціальної функції, яка являє собою певну комбінацію числових характеристик об’єктів, на основі яких і проводиться класифікація. Такі характеристики, або ознаки, мають бути виміряними у кількісному вигляді. Щоб уникнути спотворення результатів, до аналізу не включаються ті ознаки, які надто тісно пов’язані між собою або дублюють одна одну.

Формально це виглядає як:

$$D(x) = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b \quad (1.1)$$

де x_1, x_2, ..., x_n — числові значення ознак об’єкта, w — вагові коефіцієнти (оцінюють внесок кожної ознаки), b — зсув (порогове значення). Значення дискримінантної функції використовується для прийняття рішення: до якого з класів належить об'єкт.

Щоб уникнути спотворення результатів, до моделі не включають надмірно корельовані ознаки (які дублюють інформацію одна одної), оскільки це може призвести до нестійкості функції. Перед побудовою моделі часто виконують нормалізацію або стандартизацію даних, щоб усі ознаки були порівнянними за масштабом.

У просторовому уявленні кожен клас асоціюється з певною точкою в багатовимірному просторі, координати якої відображають середні значення його ознак. Ці точки, або центроїди, виступають орієнтирами при віднесенні нових об’єктів до того чи іншого класу. Процес класифікації базується на тому, наскільки близьким є об’єкт до того чи іншого центроїда, що визначається за спеціальною метрикою, яка враховує як розташування, так і поширення даних у кожному класі.

Ціль дискримінантного аналізу – максимально чітко відокремити класи один від одного, при цьому забезпечивши мінімальну варіативність усередині кожного класу. Найкращим вважається поділ, за якого відмінності між класами є найбільшими, а внутрішні відхилення – найменшими.

Щоб оцінити ефективність побудованої дискримінантної функції, використовують певні статистичні показники. Вони дають змогу зрозуміти, наскільки функція здатна точно розпізнавати належність об’єктів до класів. Якщо ці показники наближуються до певних граничних значень, це свідчить про високу точність розпізнавання.

Також аналізуються матриці класифікації (confusion matrices), які показують кількість правильно/неправильно класифікованих об’єктів, та розраховується відсоток точності.

У результаті, коли з’являється новий об’єкт, алгоритм обчислює значення дискримінантної функції для кожного класу, порівнює їх між собою і обирає той клас, значення якого найбільш наближене або максимальне. Саме до цього класу і буде віднесено об’єкт:

```
if D₁(x) > D₂(x):  → клас 1
else:              → клас 2
```

У підсумку, коли виникає необхідність класифікувати новий об’єкт, він за допомогою дискримінантної функції потрапляє до того класу, з яким має найбільшу подібність.

Іншим популярним методом для класифікації та прогнозування є метод дерев рішень [6]. Це потужний інструмент аналізу даних, що дозволяє не лише визначити належність об’єкта до певного класу, а й здійснювати передбачення на основі вхідних параметрів. Завдяки своїй простоті та прозорості логіки, дерева рішень широко використовуються у фінансах, медицині, управлінні ризиками, маркетингу, а також у бізнес-аналітиці, де важливо не лише приймати точні рішення, а й пояснювати їх суть фахівцям, які не мають математичної підготовки.

У процесі побудови дерево формує ієрархічну структуру з вузлів і гілок, яка представляє собою послідовність логічних перевірок. Аналіз починається з кореневого вузла (root node), який містить основне правило розділення. Далі, залежно від результату перевірки умови, об’єкт рухається по одному з напрямків — до лівої чи правої гілки. Цей процес повторюється на кожному наступному вузлі, доки об’єкт не досягне листового вузла (leaf node), де вже міститься остаточне рішення: клас, значення чи прогноз.

На кожному кроці дерево перевіряє певну умову, яка стосується одного з параметрів об’єкта. Наприклад, система може порівняти, чи значення ознаки «вік» більше за 35, або чи дохід перевищує певний поріг. Якщо умова виконується — об’єкт

спрямовується в одну гілку, якщо ні — в іншу. Таким чином, дерево будує гнучку послідовність рішень, яка добре адаптується до структури даних.

Однією з ключових переваг методу дерев рішень є його інтерпретованість. Кожен крок дерева легко пояснити: видно, які саме параметри і які порогові значення вплинули на прийняте рішення. Це надзвичайно важливо в сферах, де потрібно не просто надати прогноз, але й обґрунтувати його (наприклад, у медицині або судовій експертизі).

Проте цей метод має і свої недоліки. Одним з найпоширеніших є проблема переобучення (overfitting). Це трапляється тоді, коли дерево стає надто глибоким і надмірно адаптується до навчальної вибірки, втрачаючи здатність узагальнювати. На нижніх рівнях дерева дані часто діляться на малі підмножини, які можуть бути статистично незначущими. Це означає, що результати, отримані на основі таких підгруп, не можна з високою впевненістю переносити на нові, невідомі дані.

Крім того, дерева рішень чутливі до шуму у даних. Навіть невеликі зміни у вхідних значеннях можуть призвести до суттєво іншої структури дерева. Також вони погано працюють з параметрами, що мають сильну кореляцію або зміщення, якщо не виконано попередню обробку даних.

У відповідь на ці проблеми були розроблені вдосконалення методу — зокрема, методи ансамблевого навчання, такі як Random Forest або градієнтний бустинг. Вони створюють набір дерев (ліс), які працюють разом, компенсуючи недоліки окремих моделей. У таких системах кожне дерево вносить свій «голос» у фінальне рішення, що дозволяє отримати більш стабільний і точний результат.

Також у сучасній практиці застосовуються різні методи обрізання дерева (pruning) — як до побудови (pre-pruning), так і після (post-pruning). Це дозволяє зменшити глибину дерева, видаливши слабкі або нерепрезентативні гілки, що підвищує його здатність до узагальнення.

Підсумовуючи, дерева рішень — це зручний і потужний інструмент для прийняття рішень у задачах класифікації та регресії. Їхня головна сила — у зрозумілості структури та здатності працювати як із числовими, так і з

категоріальними даними. Проте для складних задач із великою кількістю параметрів або змінною структурою даних рекомендується використовувати модифіковані або комбіновані підходи для досягнення більшої точності та надійності.

1.4 Використання інтелектуальних технологій для розв'язання задач класифікації параметрів руху рухомого об’єкта

У сучасних системах аналізу, завдання класифікації часто ускладнюється високим рівнем невизначеності вхідних даних, нечіткістю ознак, а також динамічністю поведінки самих об’єктів. У таких умовах ефективність класичних підходів до класифікації, заснованих на жорстких правилах або глибокому навчанні, значно знижується, оскільки ці методи здебільшого орієнтовані на чіткі межі між класами, стабільність ознакових структур і повноту вхідної інформації. У випадках, коли дані надходять від різнорідних сенсорів, частково втрачаються або містять суттєві флуктуації, більш прийнятним є використання концепцій нечіткої логіки. Порівняння з чіткою логікою зображена на таблиці 1.2

Таблиця 1.2 – Порівняння класифікацій

Характеристика	Класична класифікація	Нечітка класифікація
Тип рішень	Жорсткі (однозначні)	Гнучкі (ступінчасті)
Робота з неповними даними	Погана	Хороша
Інтерпретованість	Зазвичай обмежена	Висока
Навчання на даних	Обов’язкове	Необов’язкове

Нечітка логіка як формалізована система дозволяє моделювати знання у вигляді нечітких лінгвістичних конструкцій, які відображають реальний підхід людини до прийняття рішень в умовах невизначеності. Замість того, щоб оперувати категоріями «належить / не належить», система нечіткої класифікації дозволяє оцінювати ступінь належності об’єкта до кількох класів одночасно, формуючи при цьому ймовірнісний або коефіцієнтний розподіл упевненості. Це надзвичайно важливо при класифікації об’єктів, які мають схожі ознаки або демонструють поведінку, притаманну відразу

кільком класам. Наприклад, у системах моніторингу безпілотних об'єктів Fuzzy-класифікатори дозволяють враховувати фактори, що постійно змінюються – наприклад, коливання швидкості, неповні зображення або часткове перекриття цілей. Завдяки цьому система може продовжувати функціонувати навіть тоді, коли класичний алгоритм просто відкидає нечіткі чи неповні дані.

Процес нечіткої класифікації базується на формуванні системи правил, у яких вхідні змінні представлені у вигляді термів лінгвістичного типу – таких як «висока швидкість», «мала площа силуету», «низька температура». У такому підході формуються лінгвістичні правила типу "Якщо відстань мала, тоді швидкість низька", що є основою роботи Fuzzy Inference System (FIS) [7]. Така система містить чотири основні компоненти – фазифікатор, базу знань (правил), модуль логічного виводу та дефазифікатор, що забезпечують повний цикл обробки нечітких даних. Значення цих термів задаються через функції належності, які відображають, наскільки конкретна величина відповідає певному описовому рівню. Наприклад, якщо система аналізує температуру об’єкта, значення 42 С може одночасно відповідати двом термам – «середня температура» зі ступенем належності 0.7 та «висока температура» зі ступенем 0.3. Це дає змогу активувати кілька правил, які реагують на різні лінгвістичні оцінки, і забезпечує гнучкість аналізу навіть у випадках, коли числові значення потрапляють у перехідні зони між класами. Далі, застосовуючи механізм нечіткого логічного виводу, система активує відповідні правила й обчислює, з якою інтенсивністю кожен клас може бути актуальним для поточного об’єкта. Це дозволяє створити не одномірне рішення, а багатовекторний портрет імовірностей, що уможливлює подальший гнучкий вибір дій, оцінку ризику або перенесення рішення на вищий рівень обробки.

У нечітких системах застосовуються різні типи функцій належності – трикутні, трапецеїдальні, гаусові, сигмоїдні тощо. Вибір форми функції визначає чутливість системи до змін вхідних даних та ступінь перекриття між класами. Наприклад, трикутні функції прості в реалізації, а гаусові – більш м’які в поведінці на краях. На рисунку 1.1 та 1.2 зображені трикутні функції та гаусові.

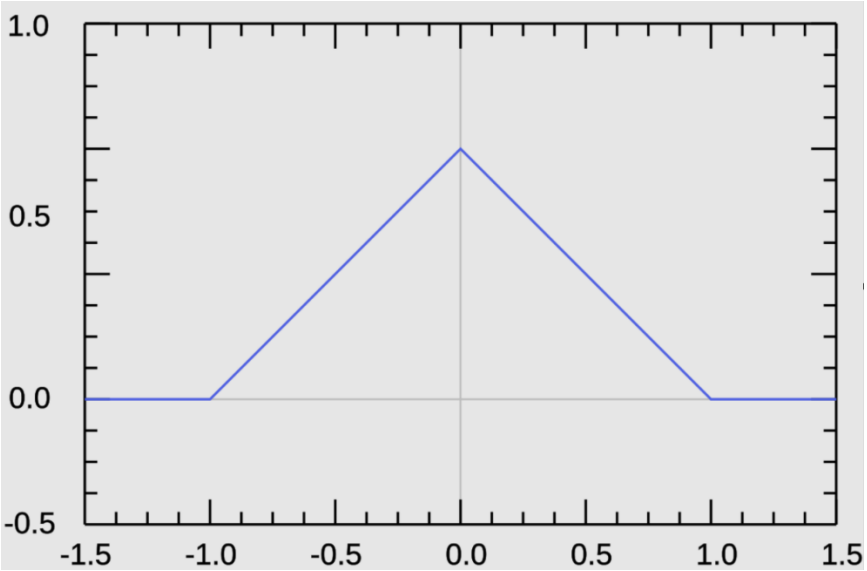


Рисунок 1.1 – Приклад трикутної функції

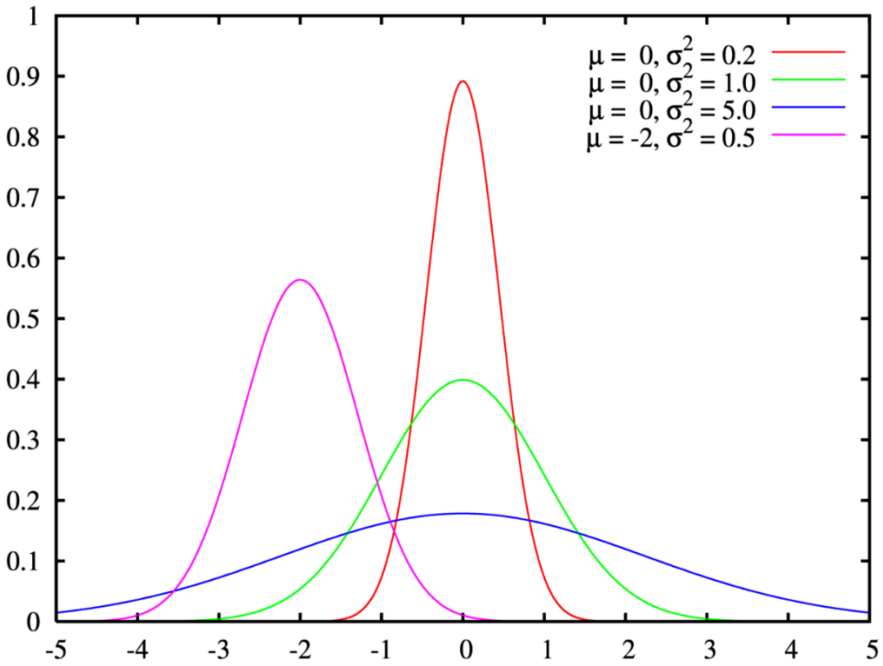


Рисунок 1.2 – Приклад гаусової функції

Особливої ефективності нечітка логіка досягає в ситуаціях, коли інформація про об’єкт представлена не в числовій, а в якісній формі або коли між класами немає чіткої межі. Наприклад, рухомий об’єкт, що пересувається зі швидкістю 35 км/год, може частково відповідати як ознаці «низька швидкість», так і «середня швидкість», при цьому його інші характеристики, такі як температурна сигнатура або форма, можуть також мати перехресне накладання на кілька класів. У таких випадках нечіткий підхід дозволяє уникнути жорсткого розмежування та надає рішення, яке найбільше відповідає сукупності умов.

Нечітка класифікація є особливо корисною у складних багатофакторних системах, де кожна ознака окремо не має вирішального значення, проте їхня комбінація утворює узагальнену картину. Наприклад, об’єкт із середньою швидкістю, великою площею силуету та високою температурою може з великою імовірністю бути класифікованим як військова техніка, тоді як ті самі ознаки, але з нижчими значеннями температури, зміщують класифікацію в бік цивільного транспортного засобу. Саме такі переходи між класами, які неможливо описати логікою «або–або», і є природним середовищем для застосування Fuzzy-алгоритмів.

Окрім того, нечітка логіка дозволяє інтегрувати у класифікаційну систему експертні знання, які можуть бути виражені у вигляді умовно-природних правил. Цей підхід особливо корисний при використанні нечітких моделей у медичних, військових та робототехнічних системах, де правила приймаються на основі досвіду операторів або аналітиків. Наприклад, при контролі анестезії або класифікації фаз епілептичного нападу використання Fuzzy SVM на основі IT2 FLS[8] дозволяє значно підвищити точність у складних умовах. Це відкриває можливість створення моделей, що поєднують як формальні ознаки, так і практичний досвід операторів систем спостереження, аналітиків чи військових фахівців. Така інтеграція знань стає фундаментом для створення пояснюваних інтелектуальних систем, де кожне рішення можна відстежити та зрозуміти з позиції логіки людини.

З технічної точки зору реалізація нечіткої класифікації передбачає формування ядра у вигляді бази правил, побудованої на основі узагальнених типових ситуацій. Загальна структура типової нечіткої системи включає чотири основні модулі: фазифікацію (перетворення вхідних значень у ступені належності), базу правил (записані у формі лінгвістичних умов), блок логічного виводу (для активації правил) та дефазифікатор (який формує остаточне рішення або оцінку). Така архітектура забезпечує адаптивність до широкого спектра прикладних задач. Вхідні дані проходять фазифікацію, тобто перетворення на нечіткі множини, які надалі обробляються у системі правил, після чого виконується агрегування ймовірностей і перехід до дефазифікації, якщо потрібен однозначний результат. Проте в багатьох випадках системи працюють у «м’якому режимі», коли рішення подається у вигляді

оцінки ступеня належності до кожного класу, що надзвичайно важливо для ідентифікації рухомих об’єктів у режимі реального часу.

Генетичні алгоритми (ГА) – це обчислювальні методи оптимізації, які наслідують принципи природного добору та генетичних процесів [9]. Вони імітують механізми відбору, схрещування та мутації, дозволяючи системі шукати оптимальне рішення навіть у просторах безперервних чи дискретних параметрів. Важливо, що ГА не потребують інформації про градієнти або інші похідні функцій, що робить їх особливо корисними для задач, де класичні методи не застосовні. Вони призначені для розв’язання складних задач шляхом моделювання генетичного процесу, що поступово вдосконалює набір можливих рішень. У таких алгоритмах потенційні варіанти рішень представляються у вигляді закодованих структур – наприклад, бінарних рядків або інших форматів даних.

Основною передумовою застосування генетичного алгоритму є наявність множини спостережень за рухомим об’єктом, представлених у вигляді вектора параметрів – наприклад: координат, швидкості, прискорення, змін траєкторії, радіолокаційних або оптичних ознак. ГА працює як оптимізаційний механізм, що дозволяє виявити найкраще наближення до вже відомих моделей поведінки або формальних описів різних типів об’єктів. Наприклад, у охоронних системах ГА може використовуватись для розпізнавання типу транспортного засобу за сукупністю параметрів його руху – таких як прискорення, напрямок повороту, час між змінами траєкторії. Алгоритм поступово підлаштовується до різних моделей поведінки, дозволяючи виявити, чи об’єкт рухається за цивільною або типовою бойовою схемою.

Процес роботи алгоритму починається зі створення початкової популяції, яка зазвичай формується випадковим чином. Ця початкова популяція, що формується випадково, може бути представлена у вигляді бінарних хромосом або масивів дійсних чисел, залежно від типу задачі. Наприклад, у задачах з неперервними параметрами використовують так звані continuous GA, де кожен хромосом кодується як масив реальних чисел. Далі відбувається послідовність ітерацій (так званих поколінь), під час яких застосовуються основні еволюційні операції – відбір, схрещування

(кросовер) та мутація [10]. Ці операції імітують біологічні процеси, зокрема природний добір, розмноження та генетичні зміни.

На етапі відбору кожен індивід оцінюється згідно з функцією пристосованості, яка визначає, наскільки вдало запропоноване рішення виконує поставлену задачу. Функція пристосованості є критичною – саме вона визначає напрям еволюції популяції. Як відзначено у літературі, неправильне визначення функції може спричинити "експлуатацію" недоліків системи або її передчасну конвергенцію до локального, а не глобального оптимуму Рішення з вищою придатністю мають вищу ймовірність бути обраними для подальшого розвитку, подібно до виживання найсильніших у природі.

Схрещування полягає в обміні генетичною інформацією між двома відібраними індивідами з метою створення нових – нащадків. Це нагадує процес статевого розмноження, у результаті якого виникають нові, генетично різноманітні особини.

Мутація ж передбачає внесення невеликих випадкових змін до генів, що дозволяє підтримувати різноманітність популяції та досліджувати нові ділянки простору можливих рішень. Попри свою низьку ймовірність, мутація відіграє ключову роль у підтриманні різноманітності популяції та запобіганні застою на локальних максимумах. Наприклад, при ймовірності 0.001, з великої кількості бітових позицій зазвичай змінюється лише 1-2, але навіть це може спричинити значні зрушення у загальній динаміці пошуку

Після виконання генетичних операцій формується нове покоління популяції, яке замінює попереднє. Такий цикл повторюється або визначену кількість разів, або до моменту досягнення заданого критерію зупинки – наприклад, до досягнення необхідного рівня ефективності або вичерпання максимальної кількості ітерацій.

На рисунку 1.3 зображено блок-схему яка окреслює основні кроки в генетичному алгоритмі.

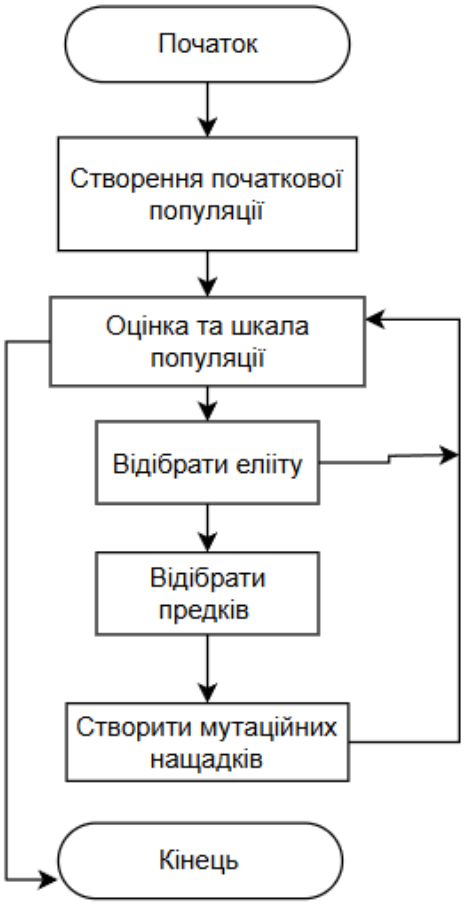


Рисунок 1.3 – Блок-схема генетичного алгоритму

У сучасних інтелектуальних системах ГА часто комбінуються з іншими методами – наприклад, із нейронними мережами або нечіткою логікою. Такі гібридні підходи можуть включати поєднання ГА з нейронними мережами (наприклад, для налаштування їхньої архітектури) або з методами градієнтного спуску, де ГА виконує глобальний пошук, а градієнтні методи – локальне уточнення Такі гібридні підходи дозволяють автоматично формувати правила або навчатись на даних, при цьому зберігаючи здатність до адаптивної оптимізації параметрів і структури моделей.

Реалізація оптимізації параметрів рухомого об’єкта на основі генетичного алгоритму також дозволяє створити навчальну систему, здатну постійно вдосконалювати свою точність. Після кожного епізоду класифікації результат може бути оцінений вручну або автоматично, і, у разі помилки, включений до бази знань для подальших ітерацій. Таким чином, ГА може слугувати ядром адаптивної системи класифікації, яка самонавчається в процесі експлуатації.

2 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ІДЕНТИФІКАЦІЇ РУХОМОГО ОБ’ЄКТУ З ВИКОРИСТАННЯМ НЕЧІТКОЇ КЛАСИФІКАЦІЇ

2.1 Метод лінгвістичної інтерпретації параметрів руху об’єкту

У тих випадках, коли вихідний інтервал, сформований нечіткою системою, не дозволяє однозначно прийняти рішення – наприклад, коли межі інтервалу охоплюють декілька класів або частково перетинають їхні кордони, – доцільним є використання методу лінгвістичної інтерпретації. Цей підхід базується на перетворенні числових або нечітких оцінок у зрозумілі для людини мовні категорії, що виражають якісні характеристики об’єкта чи ситуації. Такий механізм особливо корисний у задачах, де точна числова відповідь є або недосяжною, або неінформативною з погляду прийняття рішень, зокрема у випадках, коли результат необхідно узгодити з експертним висновком, нормативними вимогами або когнітивним сприйняттям операторів.

Лінгвістична інтерпретація дозволяє оцінювати складні об'єкти на основі набору факторів, кожен з яких подається у вигляді лінгвістичної змінної – терму, що описує якісний стан ознаки. Наприклад, рівень ризику може бути описаний як «низький», «середній» або «високий» замість конкретних числових значень. Така форма подання сприяє кращому розумінню та інтуїтивному сприйняттю результату як експертом, так і кінцевим користувачем. Це особливо актуально в умовах обмеженого часу на ухвалення рішень або в середовищах, де працюють користувачі без глибоких технічних знань.

Формально лінгвістична інтерпретація здійснюється через зіставлення інтервалу належності до відповідних мовних категорій, що попередньо визначені для кожної з характеристик. Якщо інтервал повністю потрапляє в область одного класу, то об’єкту однозначно присвоюється відповідна лінгвістична оцінка. У випадку ж перетину кількох класів, інтерпретація може включати комбінацію термів, наприклад: «від середнього до високого» або «ймовірно середній». Така гнучкість дозволяє моделі адаптувати висновок до умов неповноти, суперечливості або розмитості інформації.

Крім того, метод лінгвістичної інтерпретації створює основу для формування логічних умов у правилах продукційного типу, які широко застосовуються в експертних системах. Наприклад, на підставі лінгвістичних оцінок можна будувати умовні конструкції типу: «Якщо швидкість висока, а площа об’єкта велика, тоді рівень загрози – високий». Таким чином, числові значення трансформуються у зрозумілі категорії, які зручно аналізувати, перевіряти та коригувати під час налагодження системи.Формально, нечітка модель, що реалізує цей підхід, описується у вигляді кортежу двійки (W, Q), де:

$$W = x_1 \times x_2 \times \ldots \times x_h \qquad (2.1)$$

множина всіх можливих комбінацій ознак-факторів, утворена через операцію декартового добутку. Кожна змінна x_i (де i = 1, 2, ..., h) – це окрема ознака об'єкта, що може набувати лінгвістичних значень. А Q – інтервальний показник, що відображає діапазон значень від одного класу до іншого.

Множина W розбивається на k нечітких класів:

$$W = \{l_1, l_2, \ldots, l_k\} \qquad (2.2)$$

кожен з яких відповідає певному узагальненому стану або категорії системи. Для опису приналежності конкретного об'єкта до класу l_j використовується функція належності M_{L_j}:

$$M_{L_j}(x_1, x_2, \ldots, x_r) = \cup_{(a_1, a_2, \ldots, a_5) \in L_j} W_{jp}(M_{a_1}(x_1) \cap M_{a_2}(x_2) \cap \ldots \cap M_{a_5}(x_5)) \qquad (2.3)$$

де $a_1, a_2, \ldots, a_5$ – значення лінгвістичних змін. W_{jp} – вага правила з номером $jp(p = \overline{1, k_j})$. W_{jp}=[0,_,1], k_j-кількість правил для класа h_l.

У рамках реалізованої моделі було сформовано нечіткі еталонні класи, що представляють собою узагальнені оцінки об’єкта за сукупністю факторних ознак. Така класифікаційна структура дозволяє відображати різні рівні відповідності об'єкта до заданих умов безпечного або небезпечного розташування. Для кожного класу було визначено набір характерних комбінацій лінгвістичних змінних, які відповідають

певному якісному стану та узгоджуються з експертними уявленнями про ступінь ризику або загрози. Формування таких класів забезпечує адаптацію системи до неоднозначності, що притаманна поведінці реальних об’єктів у складних умовах середовища.

Усього було сформовано 5 еталонних класів, зокрема:

L_1 – "Н" – Параметри руху об’єкта повністю відповідають умовам безпечного розташування (за думкою експерта). Цей клас описує об'єкти з мінімальними значеннями ознак, які не викликають занепокоєння. Вони мають низьку швидкість, малу площу, стабільну траєкторію та інші показники, що відповідають фоновому або нейтральному рівню активності. Такий клас вважається еталоном безпечного функціонування об’єкта в контрольованому середовищі.

L_2 – "М" (маленький) – Параметри руху об’єкта здебільшого відповідають умовам безпечного розташування. Вони характеризують об’єкти, поведінка яких є дещо активнішою, ніж у класі L_1, проте все ще в межах допустимої норми. Це можуть бути об'єкти, які демонструють незначне підвищення швидкості або площі, але не виходять за межі ризикованих зон. Така поведінка розцінюється як потенційно безпечна, але вимагає періодичного спостереження.

L_3 – "С" (середній) – Параметри руху об’єкта частково відповідають умовам безпечного розташування. Цей клас є перехідним і описує об'єкти з помірною активністю та нестабільністю. Значення ознак знаходяться в центральних ділянках функцій належності, що означає підвищену динамічність, змішану поведінку або варіативність траєкторії. Клас "С" є критичним для подальшого аналізу, оскільки саме тут найчастіше виникає потреба в застосуванні додаткових експертних або автоматизованих процедур розпізнавання.

L_4 – "ВС" (вище середнього) – Параметри руху об’єкта слабо відповідають умовам безпечного розташування. Об'єкти цього класу демонструють активні або нестандартні дії, які можуть сигналізувати про зміну режиму, підготовку до небезпечної поведінки або потенційну загрозу. Ознаки, притаманні цьому класу,

наближаються до високих значень, і саме їх наявність вимагає посиленого моніторингу та автоматичного сповіщення операторів.

L_5 – "В" (високий) – Параметри руху об’єкта не відповідають умовам безпечного розташування. Це найбільш ризикований клас, який відповідає об'єктам з максимальними значеннями ознак — високою швидкістю, великою площею проєкції, непередбачуваною траєкторією або надмірною варіативністю у поведінці. Такі об'єкти можуть свідчити про загрозливу активність, зокрема – підготовку до перетину контрольованих меж, приховану розвідку чи деструктивну дію.

У моделі було сформовано п’ять нечітких еталонних класів L_1–L_5 за оцінкою експерта, кожен з яких відображає певний рівень оцінюваної характеристики. Для кожного класу задано набір лінгвістичних правил, що складаються з п’яти ознак-факторів (наприклад: координата, площа, радіус, напрямок, швидкість), кожна з яких приймає одне з лінгвістичних значень: Н (низький), М (маленький), С (середній), ВС (вище середнього), В (високий). Таке правило описує один із типових сценаріїв поведінки об'єкта і використовується в процесі логічного виводу при класифікації.

Система лінгвістичних правил виконує роль бази знань, яка формалізує експертний досвід та дозволяє ефективно інтерпретувати параметри об’єкта в умовах неповноти або варіативності даних. Саме завдяки такій структурі модель забезпечує не лише класифікацію, а й пояснюваність результату, що є важливою вимогою до сучасних інтелектуальних систем. База правил зображена на таблиці 2.1.

Таблиця 2.1 – База правил

Ознака 1	Ознака 2	Ознака 3	Ознака 4	Ознака 5	Клас
ВС	ВС	М	Н	Н	Низький
ВС	Н	М	М	Н	
Н	М	М	Н	Н	
Н	ВС	М	Н	Н	
М	С	ВС	М	М	Маленький
М	С	В	С	М	
ВС	С	ВС	С	М	
ВС	С	ВС	ВС	С	Середній
С	М	С	С	С	
ВС	М	М	С	С	
ВС	М	М	ВС	ВС	Вище середнього
С	ВС	ВС	ВС	ВС	
ВС	ВС	ВС	В	В	Високий
С	С	ВС	В	В	

2.2 Побудова нечітких моделей ідентифікацій параметрів руху об’єкти типу БПЛА на основі інтервальних функцій належності

У контексті побудови нечітких моделей важливим поняттям є універсальна множина, яка охоплює всю предметну область, що розглядається у тій чи іншій задачі. На основі цієї множини формується нечітка множина, яка визначається як сукупність впорядкованих пар, де кожній змінній універсуму відповідає певне значення функції належності. Функція належності — це відображення з універсальної множини у відрізок [0;1], де значення 0 означає повну відсутність належності елемента до множини, а значення 1 — повну належність. Якщо ж функцію належності обмежити лише двома значеннями — 0 або 1 — то така множина перестає бути нечіткою і стає звичайною, тобто чіткою, а сама функція — характеристичною.

Наприклад задано універсальну множину , яка охоплює всю розглядувану предметну область. Нечітка множина в цьому контексті розглядається як сукупність пар виду

Функція належності набуває значень у діапазоні [0,1], де 0 означає повну відсутність належності, а 1 – повну належність елемента до нечіткої множини. Якщо ж у цьому контексті замінити інтервал [0,1] на дискретну множину {0,1}, то така функція перетворюється на звичайну характеристичну функцію чіткої множини.

Нечітка множина $A \subseteq X$ являє собою набір пар $\{(x, \mu_A(x))\}$, де $x \in X$ і $\mu_A: X \to [0,1]$ – функція належності, що являє собою деяку суб'єктивну міру відповідності елемента x нечіткій множині A.

Для випадку, коли нечітка множина А задана на скінченному універсумі її доцільно представляти у вигляді:

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n = \sum_{i=1}^{n} \mu_A(x_i)/x_i \qquad (2.4)$$

де $\mu_A(x_i)/x_i$ кожна пара називається синглтоном, а символ “+” позначає об’єднання таких пар.

Якщо ж універсум X є неперервним, використовують запис:

$$A = \int_X \mu_A(x)/x \; . \qquad (2.5)$$

Під носієм нечіткої множини А мається на увазі множина елементів A_S, для яких значення функції належності відмінне від нуля. Це можна виразити так:

$$A_S = \{x \in X | \mu_A(x) > 0\}. \qquad (2.6)$$

Нечітка множина вважається скінченною, якщо її носій є скінченною множиною. Потужність такої множини визначається числом елементів її носія як звичайної (чіткої) множини.

Особливу роль у нечітких множинах відіграє поняття носія множини, тобто множини таких елементів, для яких функція належності є більшою за нуль. Цей носій визначає область, у якій множина фактично «існує», адже за межами носія її вплив дорівнює нулю. Якщо носій є скінченною множиною, то й сама нечітка множина

вважається скінченною, а її потужність можна визначити як кількість елементів у носії. У порівнянні зі звичайними множинами, де елемент або належить, або не належить до множини, у нечітких множинах допускається часткове належність. Це дає змогу моделювати реальні ситуації, де класи не мають чітких меж або розмежування є умовним. Такий підхід є надзвичайно корисним у тих галузях, де інформація подається у формі експертних оцінок, описових характеристик або природної мови — наприклад, у системах прийняття рішень, інтелектуальному аналізі даних, управлінні ризиками або розпізнаванні образів. Саме тому поняття нечітких множин та відповідні функції належності стали невід'ємною частиною сучасних методів штучного інтелекту, fuzzy-систем і експертних систем.

Одним із найважливіших етапів у побудові таких моделей є визначення форми функцій належності. Вони можуть бути трикутними, трапецеїдальними, гаусовими або іншими. Вибір форми залежить від особливостей прикладної задачі та характеру даних. У багатьох випадках доцільно використовувати гаусоподібні функції, які дозволяють м'яко відображати зміну ступеня належності поблизу межових значень. На відміну від звичайної (чіткої) множини, де кожен елемент або належить множині, або не належить (тобто оцінюється значенням 1 або 0), у нечіткій множині допускається часткове належність елементів. Це особливо важливо у тих сферах, де кордони між класами не є строго визначеними або носять умовний характер. У застосуваннях, пов'язаних із прийняттям рішень, обробкою природномовних даних або описових оцінок, де значення не завжди можна чітко класифікувати, використовується поняття нечітких множин та відповідних функцій належності. Їх побудова — один із ключових етапів при створенні нечітких моделей, зокрема у fuzzy-системах, експертних системах, інтелектуальному аналізі даних, системах контролю тощо. На таблиці 2.2 зображені методи побудов функцій належності.

Таблиця 2.2 – Методи побудов функцій належності

Метод	Суть методу	Особливості
Метод парних порівнянь	Експерти послідовно порівнюють об’єкти між собою, оцінюючи ступінь їх відносної переваги або належності до класу	Забезпечує ранжування за відносною важливістю; підходить для складних оціночних задач
Метод експертних оцінок	Ступінь належності визначається безпосередньо фахівцями на основі досвіду та інтуїції, без порівнянь	Простий у реалізації; можливі суб’єктивні похибки; часто використовується на ранніх етапах
Параметричний підхід	Функції належності задаються аналітично, наприклад, у вигляді трикутних, трапецієподібних або гаусових кривих	Дає змогу будувати гладкі та контрольовані функції; добре підходить для автоматизованих систем
Інтервальний метод	Значення належності не є точкою, а інтервалом, який відображає діапазон можливих значень і невизначеність	Особливо корисний у нечітких інтерпретаціях, де важлива гнучкість і урахування похибок

Функції належності [11] виконують роль перетворювачів числових значень параметрів у ступінь належності до відповідних лінгвістичних термів. Це особливо актуально у випадках, коли необхідно працювати з розмитими межами між поняттями. Наприклад, температура 30°C не є однозначно "середньою" чи "високою" — вона може бути одночасно частково середньою та трохи високою. У такому випадку функції належності дають змогу чисельно описати, з якою силою це значення належить до кожної категорії.

У випадку, коли ми працюємо з вектором параметрів, кожен з них обробляється окремо через свою функцію належності, яка визначає ступінь участі у кількох класах одночасно. Наприклад, якщо оцінюється не лише температура, а й вологість, швидкість, площа, напрямок тощо — для кожного параметра будується своя множина та відповідні криві належності.

Результатом такого аналізу не є просто вибір одного класу, а розподіл належності — тобто набір значень, який показує ступінь відповідності кожному класу. Це дає змогу отримати багатоваріантні висновки, що є надзвичайно важливим у складних системах прийняття рішень, де важлива не тільки категорія, але й упевненість у виборі. У системах штучного інтелекту, нечіткої логіки та машинного навчання такий підхід дозволяє забезпечити гнучкість і толерантність до нечітких, неповних або суперечливих даних. Нечітке класифікування дозволяє уникати жорстких рішень у прикордонних ситуаціях, забезпечуючи плавний перехід між класами — подібно до того, як це відбувається у людському мисленні. Завдяки своїм властивостям, функції належності стали ключовим інструментом у розробці fuzzy-контролерів, діагностичних систем, нечітких класифікаторів, інтервальних моделей, де класична логіка не справляється через свою надмірну жорсткість. Вони поєднують у собі формалізовану структуру з елементами "людської" нечіткості, що робить такі системи гнучкими, адаптивними та більш природними в інтерпретації. На рисунку 2.1 зображений графік належності для 3 параметрів.

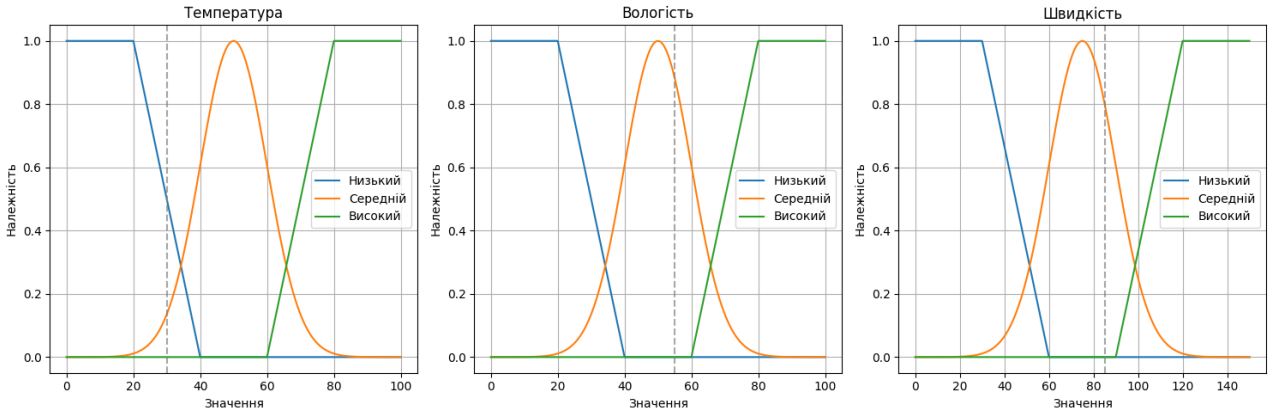


Рисунок 2.1 – Графіки належності

Для більш точного опису об’єкта іноді використовують інтервальні функції належності, які для кожного значення аргументу визначають інтервал можливих значень належності, а не одне фіксоване число. Такий інтервал зазвичай лежить у межах від 0 до 1, де нижня межа визначає мінімальний можливий ступінь належності елемента до класу, а верхня — максимальний. Ширина цього інтервалу безпосередньо залежить від варіативності оцінок, наданих експертами, або від

ступеня достовірності вимірювань. Чим більша невизначеність або розбіжність у думках, тим ширшим буде інтервал.

На відміну від класичних (одновимірних) функцій належності, інтервальні функції [12] дозволяють не лише визначити ступінь належності вхідного значення до певного класу, але й описати рівень довіри до цієї належності у вигляді інтервалу. Це особливо важливо у випадках, коли система оперує неточними, частковими або суперечливими даними. Наприклад, у системі, де дані надходять з декількох джерел або обробляються різними аналітиками, можуть виникати різні погляди на те, наскільки певне значення параметра належить до того чи іншого класу. У таких випадках представлення оцінки у вигляді діапазону значень, а не одного числа, дає змогу більш повно врахувати розкид думок або похибки.

На практиці це означає, що замість одного числа, яке б вказувало на "наскільки сильно" об’єкт належить до певної категорії, ми отримаємо діапазон значень – від нижньої до верхньої межі належності.

З технічної точки зору, замість фіксованого центру гаусової функції, у таких випадках використовується інтервал [c_min, c_max], який відображає допустимий діапазон зміни центрального значення (середини кривої). Це дозволяє врахувати непевність у виборі оптимального параметра і забезпечити більшу гнучкість у моделюванні. На основі цього інтервалу формуються дві межі функції належності – нижня та верхня, які разом утворюють зону нечіткості, тобто область, у якій неможливо чітко визначити ступінь належності без додаткових уточнень або інформації.

Така зона нечіткості між верхньою та нижньою кривими функції належності виконує роль "буфера невизначеності", в межах якого система приймає рішення з урахуванням можливих коливань. Це суттєво знижує ризик прийняття помилкових рішень у прикордонних або спірних ситуаціях. Зокрема, якщо значення належності потрапляє в таку зону, система може або перенаправити об’єкт на додатковий аналіз, або враховувати альтернативні сценарії дій.

На рисунку 2.2 зображено приклад інтервальної функції належності типу-2, яка має гаусову форму. Така функція одночасно відображає плавність переходів між класами (завдяки гаусовій формі) та невизначеність у знаннях (завдяки інтервальному підходу). Саме поєднання цих двох властивостей забезпечує високу точність та адаптивність моделі в умовах складних і динамічних задач, зокрема у системах розпізнавання поведінки об'єктів, відеоаналітики, ситуаційного моніторингу та кіберфізичної безпеки.

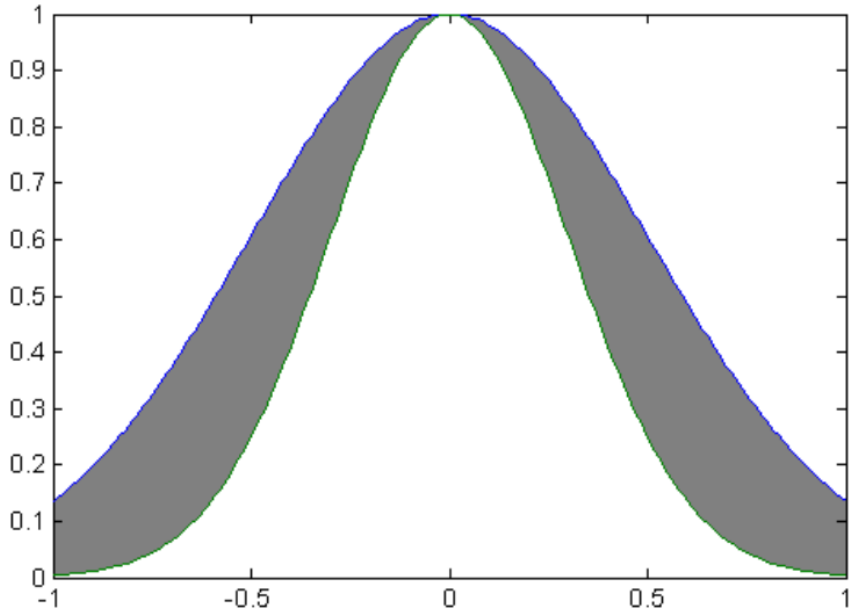


Рисунок 2.2 – Інтервальна функція належності

Застосування інтервальних функцій належності дозволяє створювати більш реалістичні, стійкі до шуму та експертно узгоджені моделі, що особливо важливо в умовах неповноти, нечіткості та варіативності вхідної інформації.

Інтервальна функція належності описується як:

$$\mu_A(x) = \left[\underline{\mu_A}(x), \overline{\mu_A}(x)\right] \tag{2.7}$$

де $\underline{\mu_A}(x)$— нижня межа значення функції належності, а $\overline{\mu_A}(x)$ – верхня межа. Значення верхньої межі інтервальної функції належності завжди не менше за відповідне нижнє значення. У випадку, коли обидві межі збігаються, така функція

належності називається виродженою і фактично еквівалентна звичайній (традиційній) функції належності.

2.3 Настроювання параметрів інтервальних функцій належності для ідентифікації параметрів руху об’єкту.

У процесі побудови інтервальних функцій належності особлива увага приділяється їхньому параметричному налаштуванню. Одним із найпоширеніших способів формалізації таких функцій є використання гаусового закону, що забезпечує гладкість і симетричність кривої належності. Загальний вигляд функції задається виразом:

$$\mu(x) = e^{-\left(\frac{x-m}{\partial}\right)^2} \tag{2.8}$$

де δ (дельта) виступає в ролі параметра стиснення функції та визначає ширину області, в якій значення функції наближаються до 1, а m – це параметр зміщення, що відповідає центру або моді функції, тобто значенню, якому надається найбільший ступінь належності. Змінна xxx при цьому належить до універсальної множини значень, які потенційно можуть бути оброблені нечіткою системою. Інтервальний характер такої функції проявляється в тому, що параметри m і δ можуть змінюватися в певних допустимих межах, які обираються з урахуванням збереження адекватності моделі відносно експериментальних даних.

Параметри функцій належності підбирались за оцінкою експера окремо для кожного з вхідних показників: координат (X, Y), площі об’єкта, радіуса, напрямку. Найважливішими серед них були центр функції (mean), який вказує на характерне або типове значення для кожного нечіткого класу, та ширина функції (delta), яка визначає, наскільки «розпливчатою» або нечіткою є межа цього класу. Такий підхід дозволяє досягти оптимального балансу між чутливістю до варіацій і толерантністю до шуму в реальних даних.

Наприклад, для координати по осі X було проаналізовано всі значення, які зустрічались у експериментальному наборі. Наприклад, кількість об’єктів

фіксувалась поблизу значень 0, 70, 140, 210 і 280 одиниць. Саме ці значення були обираються як центри для лінгвістичних класів: «Низький», «Малий», «Середній», «Вище середнього» та «Високий». Оскільки координата X мала широку дисперсію, ширини відповідних гаусових функцій (дельта) підбиралися так, щоби забезпечити перекриття суміжних класів, створюючи плавний перехід між ними. Завдяки цьому, навіть якщо значення потрапляє між двома класами, система не змушена робити жорсткий вибір, а може врахувати часткову належність до обох.

Аналогічний підхід можна застосовувати до параметра площі об’єкта (Area). Цей показник відзначався значною варіативністю — від кількох десятків до понад 4000 одиниць. У таких умовах фіксовані або надто вузькі функції належності призвели б до ігнорування важливих, але крайових значень. Тому центри класів розміщувалися на рівнях 0, 1000, 2000, 3100 і 4100, з ширинами (дельта) до 1400, що дозволяло охопити як типові, так і рідкісні значення, не втрачаючи інформацію про крайові випадки.

У свою чергу, параметри з меншою дисперсією, як-от радіус об’єкта, вимагали більш точного підходу. Для цього показника характерні значення в діапазоні 0–40 одиниць. Центри класів задавалися з кроком 9 (0, 9, 18, 27, 36), а ширина функцій змінювалася відповідно до позиції центру — функції з центрами на краях були ширшими, щоб уникнути втрати об’єктів із крайовими значеннями. Таким чином, було досягнуто компромісу між локальною чутливістю та загальною охоплюваністю простору ознак.

Ще один приклад — координата по осі Y, де функції належності були побудовані з урахуванням фактичної щільності розподілу значень у даних. Наприклад, значення Y = 80 може одночасно належати до класів «Малий» та «Середній» із різним ступенем, залежно від конкретного положення пік-функцій і ширини кривих. Завдяки цьому система може враховувати граничні випадки та ситуації невизначеності, що характерні для реальних процесів, зокрема в спостереженнях за рухомими об’єктами.

Такий метод побудови функцій належності дозволяє гнучко адаптувати модель до різних типів вхідних параметрів і забезпечує високу точність під час інтерпретації

неоднозначних чи перехідних значень. Ключовою перевагою використання гаусових функцій є їх здатність забезпечувати плавні зміни значення належності, що повністю відповідає природі нечіткої логіки.

Крім того, у даній системі використано інтервальні функції належності, які, на відміну від класичних (одновимірних) функцій першого типу, формують не одне значення належності, а цілий інтервал можливих значень. Це дає змогу враховувати неповноту або неточність вхідних даних, а також статистичну невизначеність, яка часто виникає в умовах шуму або обмежених спостережень. Наприклад, при спостереженні за об’єктом через камеру може бути похибка у вимірі площі, і замість фіксованого значення функція повертає діапазон, у межах якого об’єкт може бути класифікований.

У підсумку, застосування такого адаптивного, параметризованого і інтервального підходу до побудови функцій належності дозволяє суттєво підвищити достовірність та гнучкість нечіткої логічної системи, особливо в умовах динамічних, неточних або частково відомих даних. Це забезпечує більш стійку поведінку моделі, кращу інтерпретованість результатів та підвищує рівень обґрунтованості рішень, які вона генерує.

2.4 Модель математичних інтервальних операцаій

У контексті інтервальної нечіткої логіки важливим є не лише побудова функцій належності, але й можливість виконання над ними основних множинних операцій – об’єднання [13] та перетину [13]. Оскільки кожне значення належності подано у вигляді інтервалу, операції повинні враховувати як нижню $\underline{M}$, так і верхню $\overline{M}$ межі для кожної точки універсальної множини.

Операція об’єднання двох інтервальних нечітких множин A і B, і визначається як:

$$A \cup B = \{(M_{A \cup B}(x)/x)/x \in X) \tag{2.9}$$

де верхня межа інтервалу визначається:

$$M_{A\cup B}(x) = [bor_{\cup}(M_A(x), M_B(x))], \max\left[\{, \overline{M_A}(x), \overline{M_B}(x)\}\right] \quad (2.10)$$

а нижня межа:

$$bor_{\cup}(M_A(x), M_B(x)) = \begin{cases} \underline{M_A}(x), \overline{M_A}(x) > \overline{M_B}(x) \\ \underline{M_B}(x), \overline{M_A}(x) \leq \overline{M_B}(x) \end{cases} \quad (2.11)$$

Операція перетину множин гарантує вибір найменшої з можливих верхніх меж:

$$A\cap B = \{(M_{A\cap B}(x)/x)/x \in X)\} \quad (2.12)$$

де верхня межа інтервалу визначається:

$$M_{A\cap B}(x) = [bor_{\cap}(M_A(x), M_B(x)], min\left[\{, \overline{M_A}(x), \overline{M_B}(x)\}\right].$$

а нижня межа:

$$bor_{\cap}(M_A(x), M_B(x)) = \begin{cases} \underline{M_A}(x), \overline{M_A}(x) < \overline{M_B}(x) \\ \underline{M_B}(x), \overline{M_A}(x) \geq \overline{M_B}(x) \end{cases}; \quad (2.13)$$

У практичних системах класифікаціх рухомих об’єктів нерідко виникає ситуація, коли один і той самий об’єкт може частково належати одночасно до кількох різних класів. Така неоднозначність особливо характерна для задач відеоспостереження та ситуаційного аналізу, де характеристики об'єкта – як-то швидкість, напрямок руху, площа проєкції чи інші ознаки – не дають змоги провести чітку межу між класами. Наприклад, дрон, що летить зі змінною швидкістю на середній висоті у складному рельєфі, може відповідати як типовій траєкторії цивільного апарата, так і виявляти ознаки потенційної загрози. У таких випадках жорстке правило належності або відкидає корисну інформацію, або призводить до помилкових висновків. Саме тому у нечіткій логіці застосовуються операції над множинами, які дозволяють враховувати часткову належність — зокрема операції об’єднання та перетину.

Операція об’єднання у контексті інтервальних нечітких множин виконує роль агрегатора всіх можливих ступенів належності. Вона дозволяє сформувати узагальнене уявлення про потенційну приналежність об’єкта принаймні до одного з

класів. Такий підхід особливо доцільний на етапі первинного фільтрування даних або при оцінці загального ризику, коли навіть незначна часткова відповідність підозрілому профілю вимагає реакції з боку системи — наприклад, активації сигналу тривоги, переходу до точнішого режиму сканування або залучення додаткових модулів обробки. У цьому сенсі об’єднання допомагає системі бути чутливою до всіх можливих сценаріїв і вчасно ініціювати додаткову перевірку.

Операція перетину, на відміну від об’єднання, слугує інструментом посилення селекції. Вона дозволяє встановити, чи відповідає об’єкт водночас усім ключовим критеріям належності до певного класу. Це надзвичайно корисно у задачах, де пріоритетом є надійність класифікації та мінімізація хибнопозитивних спрацьовувань. Наприклад, щоб розпізнати поведінку як загрозливу, недостатньо лише високої швидкості — об’єкт також має мати характерну геометрію, напрямок руху та розмір, що відповідає заданому шаблону. Перетин дозволяє врахувати всі ці параметри комплексно, забезпечуючи суворіші критерії прийняття рішення. В умовах шумових впливів або неповних даних це дозволяє досягти більшої стабільності у роботі системи.

Інтервальні функції належності відіграють ключову роль у такій формі логічного висновку. Вони дають змогу не лише виразити сам факт належності елемента до певного класу, а й оцінити діапазон довіри до цього твердження. Такий підхід особливо цінний у ситуаціях, де дані мають розбіжності в джерелах, недостовірність у вимірюваннях або суперечливі оцінки експертів. У поєднанні з операціями об’єднання та перетину інтервальні функції дозволяють створити багатовимірну модель ухвалення рішень, що враховує як кількісні, так і якісні аспекти аналізу.

Таким чином, включення інтервальних логічних операцій у процес класифікації не лише покращує її точність, а й надає системі властивості адаптивності, гнучкості та стійкості до змін зовнішніх умов. Це робить такі підходи ефективними інструментами в задачах розпізнавання поведінки, оцінки загроз, ситуаційного моніторингу та кіберфізичної безпеки, де важливо не лише виявити об’єкт, а й правильно інтерпретувати його дії в режимі реального часу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КЛАСИФІКАЦІЇ ПАРАМЕТРИ РУХУ ОБ'ЄКТА ТИПУ БПЛА НА ОСНОВІ ДАНИХ ВІДЕОСПРОСТЕРЕЖЕННЯ

3.1 Обґрунтування вибору програмних засобів

Python – це високорівнева мова програмування загального призначення, яка відзначається простою та зрозумілою синтаксичною структурою. Вона була створена у 1991 році Гвідо ван Россумом [14] і з того часу набула великої популярності серед розробників завдяки своїй ігнучкості, потужності та легкості у вивченні. Python активно використовується в різних сферах – від розробки вебдодатків та автоматизації до наукових обчислень, штучного інтелекту, аналізу даних та побудови програмних інтерфейсів.

Python є чудовим вибором для реалізації криптографічних протоколів з кількох причин. По-перше, мова забезпечує високу читабельність і дозволяє швидко писати та перевіряти код, що особливо важливо в галузях, де пріоритетом є надійність та безпека. По-друге, Python підтримує велику кількість бібліотек, зокрема для криптографії, які дозволяють швидко реалізовувати складні алгоритми без необхідності писати їх з нуля. Крім того, Python є кросплатформним, що робить його придатним для створення додатків, які працюють на різних операційних системах.

Хоча Python не завжди забезпечує найвищу продуктивність порівняно з такими мовами як C++ чи Java, його переваги в гнучкості, простоті розробки та великій кількості готових інструментів роблять його привабливим варіантом для багатьох типів проєктів, у тому числі і в криптографії. Завдяки активній спільноті та постійному розвитку, Python продовжує залишатися однією з провідних мов для прототипування, досліджень і створення надійного програмного забезпечення.

Для створення графічного інтерфейсу користувача Python часто використовує фреймворки, такі як Tkinter[15]. Цей інструмент дозволяє створювати прості й водночас функціональні інтерфейси, інтегруючи графіку, кнопки, поля введення та інші елементи управління. Tkinter дотримується об'єктно-орієнтованої моделі, що дає змогу структурувати інтерфейси як набори взаємодіючих компонентів. Завдяки

цьому розробник має можливість створювати масштабовані та зручні програми для настільних платформ.

Для розробки на Python одним із найзручніших середовищ є PyCharm[16] – потужна інтегрована середа розробки, що надає великий набір інструментів для написання, тестування та налагодження коду. Вона підтримує автодоповнення, інтеграцію з системами контролю версій, відлагодження коду, а також має зручний візуальний інтерфейс. PyCharm сприяє підвищенню продуктивності під час розробки Python-проєктів.

3.2 Розробка основних складових програмного засобу

Програмний засіб ідентифікації параметрів рухмого об’єкту код реалізує аналіз даних за допомогою нечіткої логіки. Він написаний мовою Python із використанням бібліотек NumPy, Pandas, Matplotlib, Tkinter та SciPy. Основне вікно програми реалізоване через `Tkinter`, яке дозволяє взаємодіяти з користувачем у графічному інтерфейсі. У ньому користувач може завантажити CSV-файл з даними про рухомі об’єкти, які містять параметри, як-от напрямок руху, координати, радіус і площа контуру. Після завантаження даних вони проходять обробку, під час якої кожен об’єкт аналізується через набір нечітких правил. На рисунку 3.1 зображений алгоритм роботи програми. 2Для кожного параметра створено універсуми визначення, на які накладаються гаусівські функції належності, що моделюють нечіткі терміни – від низького до високого значення. Для визначення рівнів належності використано інтерполяцію, яка дозволяє розраховувати відповідні інтервали на основі вхідних значень. Кожне правило поєднує п’ять параметрів, і з усіх правил обирається найсильніше, тобто те, яке дає найбільшу верхню межу інтервалу (або нижню, якщо значення верхніх меж збігаються). Таким чином, об’єкт класифікується в один із п’яти експертних класів – від незадовільного стану до найкращого.

Окрім цього, в коді реалізовано можливість виведення графіків для кожного параметра, які відображають області належності до різних лінгвістичних категорій. Програма також дозволяє експортувати результати аналізу у форматі Excel, що

зручно для подальшої обробки. Усі взаємодії з користувачем – від завантаження даних до виводу результатів – відбуваються в зручному візуальному середовищі.

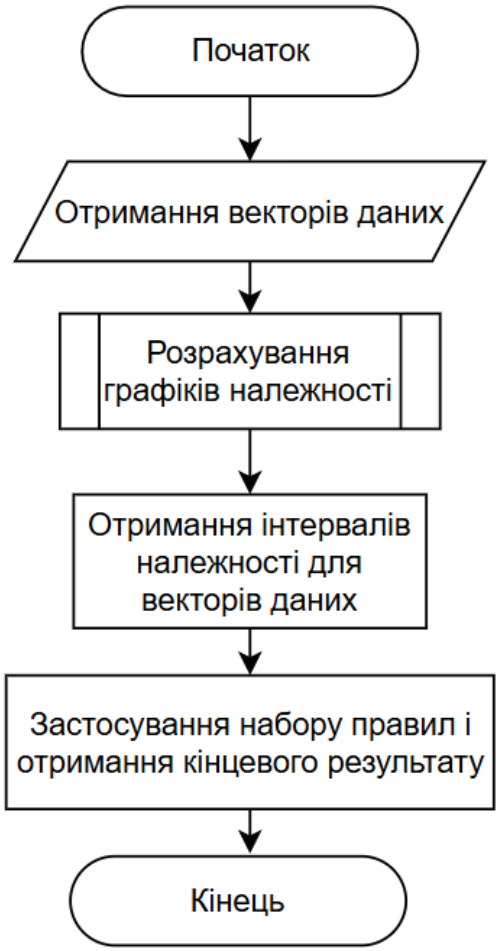


Рисунок 3.1 – Схема алгоритма роботи програми

3.3 Отримання даних

Отримання даних програмі реалізується через інтерфейс користувача, побудований за допомогою бібліотеки Tkinter, що дозволяє користувачеві обрати CSV-файл із даними про рухомі об’єкти. Після вибору файла дані завантажуються у таблицю pandas. DataFrame, фільтруються і перетворюються у зручний формат для подальшої обробки.

Ключова функція, яка відповідає за завантаження вхідних даних, має назву load_csv(). Вона реалізує повний цикл первинної обробки даних — від вибору файлу до підготовки об’єктів для подальшого аналізу. На рисунку 3.2 – зображено схему алгоритму отримання даних у програмі.

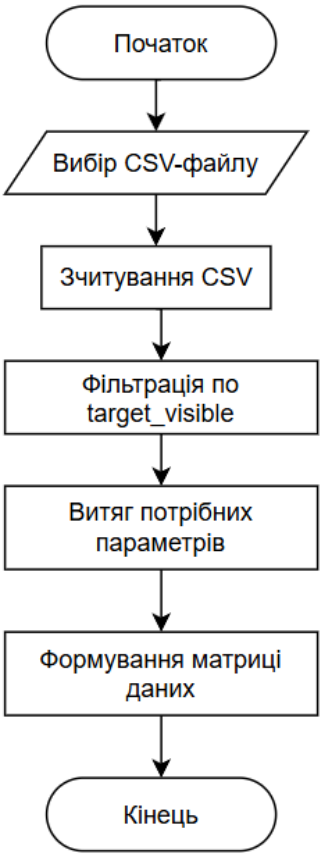


Рисунок 3.2 – Схема алгоритма отримання даних з таблиці

Робота функції починається з відкриття діалогового вікна вибору файлу за допомогою стандартного модуля filedialog. Користувачеві пропонується вибрати файл формату .csv, у якому містяться результати попередніх спостережень або сенсорних вимірювань:

```
file = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
df = pd.read_csv(file)
```

Наступним кроком виконується фільтрація даних, щоб залишити лише ті записи, де об'єкт є видимим. Це реалізується через умову df['target_visible'] == 1, яка означає, що в обробку надходять тільки ті об'єкти, які фактично були зафіксовані сенсорами або системою комп'ютерного зору:

```
df = df[df['target_visible'] == 1]
```

Цей етап є критично важливим, оскільки дозволяє відкинути шумові або неповні спостереження, що не містять корисної інформації. Таким чином, модель

працює тільки з об'єктами, які мають повний набір характеристик і дійсно присутні у спостережуваному середовищі.

Після цього із загальної таблиці вибираються лише необхідні параметри, які використовуються в аналізі: heading, target_x, target_y, target_radius, contour_area. Ці ознаки були попередньо відібрані як найбільш інформативні для класифікації об'єктів за поведінковими або геометричними характеристиками. Обрані стовпці перетворюються у матрицю NumPy, що забезпечує швидкий доступ до даних і сумісність з подальшими математичними операціями:

```
data_matrix = df[['heading', 'target_x', 'target_y', 'target_radius',
'contour_area']].to_numpy()
```

Цей підхід дозволяє швидко перемикатися між об’єктами без необхідності вручну вводити ідентифікатори або номери — усе відбувається в інтерактивному режимі з максимальним комфортом для користувача. Насамкінець, якщо завантаження даних пройшло без помилок, користувач отримує інформаційне повідомлення, яке підтверджує кількість об'єктів, готових до аналізу:

```
combo['values'] = list(range(len(data_matrix)))
combo.current(0)
```

Якщо завантаження даних пройшло успішно, користувач отримує повідомлення з інформацією про кількість об’єктів:

```
messagebox.showinfo("OK", f"Завантажено {len(data_matrix)} об'єктів.")
```

Функція load_csv() виконує важливу роль у процесі ініціалізації аналізу: вона не лише забезпечує зв’язок із джерелами даних, а й фільтрує, структурує та готує інформацію до подальшої обробки в системі нечіткої класифікації. Завдяки зрозумілому та інтерактивному інтерфейсу, користувач може зосередитись на оцінці поведінки об'єктів, не витрачаючи час на попередню підготовку даних вручну. Це робить інструмент зручним, надійним і ефективним нав.іть для користувачів без досвіду роботи з великими наборами спостережень.

3.4 Побудова графіків належності

Розрахунок графіків належності реалізовано за допомогою функції plot_gui(param_idx), яка відповідає за побудову графіків нечітких функцій належності для кожного з параметрів об'єкта. Побудова виконується з використанням бібліотеки Matplotlib у поєднанні з Tkinter, що дозволяє відображати графіки безпосередньо у вікні програми. На рисунку 3.3 зображений алгоритм побудови.

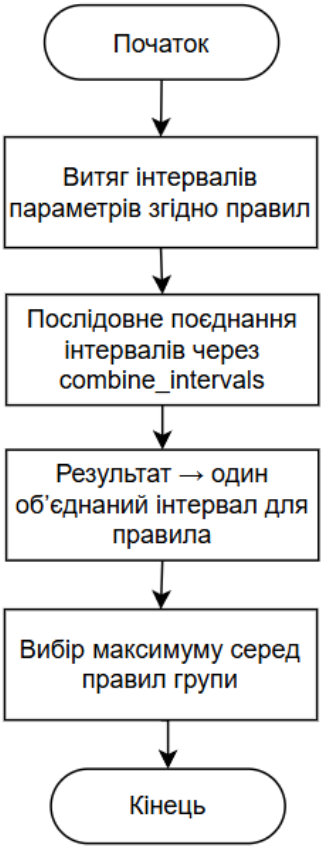


Рисунок 3.3 – Схема алгоритму побудови графіків належності

Кожен параметр у системі має свій унікальний діапазон значень, який визначається масивом universes. Цей масив задає допустимі межі для побудови функцій належності та графіків, відображаючи розподіл значень у відповідному просторі. Параметри можуть мати різну природу — швидкість, розмір, площу проєкції тощо — тому діапазон кожного з них визначається індивідуально. Для кожного параметра формується набір нечітких правил, що відповідають певним якісним характеристикам, наприклад: «низький», «середній», «високий».

Для кожного правил використовується гаусова функція належності, яка описує плавну зміну ступеня належності елемента до нечіткої множини. Такі функції є найбільш придатними для моделювання реальних процесів, оскільки не мають жорстких меж і добре відображають поступовий перехід між категоріями. Формула такої функції реалізується через:

```
def gaussian_mf(x, mean, delta):
return np.exp(-((x - mean) / delta) ** 2)
```

У цій функції mean визначає центр (найвищу точку) функції належності, а delta — ширину розсіювання, що відображає ступінь невизначеності або варіативності терма. Залежно від вибраних параметрів mean і delta, форма графіка може бути вужчою або ширшою, що впливає на те, скільки значень потрапляє до тієї чи іншої нечіткої категорії.

Під час візуалізації функцій належності спочатку створюється графічний об’єкт з використанням бібліотеки matplotlib. Стандартний розмір вікна та роздільна здатність задаються через:

```
fig = Figure(figsize=(6, 4), dpi=100)
ax = fig.add_subplot(111)
```

На цьому графіку кожен терм відображається окремою кривою, яка супроводжується інтервалами належності. Для побудови інтервальної області програма обчислює нижню та верхню межі належності для кожної точки простору x. Це виконується за допомогою локального аналізу околу точки: для кожного xi визначається вікно навколо нього (left, right), у межах якого аналізується поведінка функції:

```
left = xi - deltas[param_idx]
right = xi + deltas[param_idx]
mask = (x >= left) & (x <= right)
vals = mf[mask]
lower_bound.append(np.min(vals) if len(vals) else 0
upper_bound.append(np.max(vals) if len(vals) else 0)
```

Програма «сканує» функцію по всій довжині та будує відповідні межі нечіткої області, ураховуючи розсіювання (невизначеність) терма. Отримані масиви

lower_bound і upper_bound використовуються для створення зафарбованої області на графіку за допомогою методу fill_between, який візуально демонструє ступінь варіативності:

```
ax.fill_between(x, lower_bound, upper_bound, alpha=0.2, color=colors[j
% len(colors)])
```

Цей ефект дозволяє користувачу інтуїтивно оцінити нечіткість кожної категорії — чим ширша область, тим більше розмиття у визначенні терма.

Після побудови графіка він відображається у новому вікні, створеному на базі Toplevel у Tkinter. Завдяки використанню модуля FigureCanvasTkAgg з бібліотеки matplotlib.backends.backend_tkagg, графік інтегрується у GUI програми:

```
canvas_frame = tk.Toplevel(root)
canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
canvas.draw()
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
```

Кожен параметр має власну візуалізацію з окремими графіками для кожного терма. Це дозволяє користувачеві краще зрозуміти, як саме значення об'єкта розподіляється по нечітких множинах, як формуються інтервали належності, і яку інтерпретацію система застосує при подальшому аналізі. Таке графічне представлення значно покращує пояснюваність моделі, що є важливою вимогою для інтелектуальних систем прийняття рішень, особливо в умовах неоднозначності або при необхідності експертної перевірки.

3.5 Інтервали належності та математичні операції з ними

Інтервали належності для параметрів беруться безпосередньо з графіків нечітких функцій, які попередньо побудовані для кожного параметра. Цей процес реалізовано через обчислення інтервалів належності, які враховують не лише саму функцію, але й її поведінку на певному відрізку, що визначається параметром delta. На рисунку 3.4 зображено схему алгоритму отримання інтервалів з графіків.

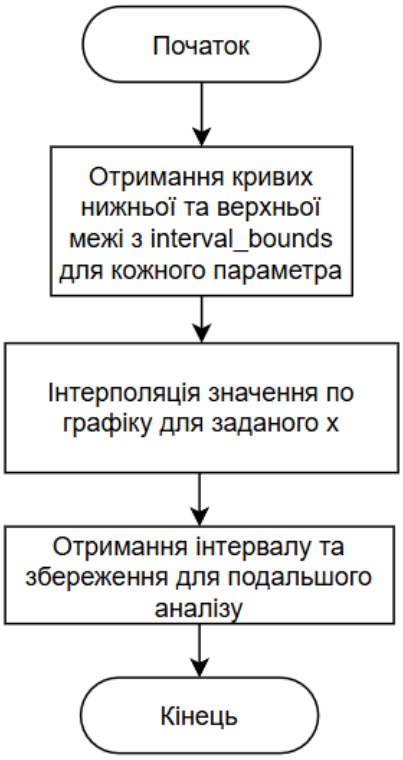


Рисунок 3.4 – Схема алгоритму обрахунку інтервалів з графіків належності

Першим кроком є формування повного набору нижніх і верхніх меж для кожної функції належності. Для цього використовується функція compute_interval_bounds, яка для кожного параметра та кожного класу розраховує інтервали:

```
group_ints = []
for rule in rules:
    a = all_intervals[0][rule[0]]
    b = all_intervals[1][rule[1]]
    c = all_intervals[2][rule[2]]
    d = all_intervals[3][rule[3]]
    f = all_intervals[4][rule[4]]
    comb = combine_intervals(a, b)
    comb = combine_intervals(comb, c)
    comb = combine_intervals(comb, d)
    comb = combine_intervals(comb, f)
    group_ints.append(comb)
```

Ці межі зберігаються в словнику interval_bounds і слугують базою для подальшого аналізу.

Коли виникає необхідність визначити інтервал належності для конкретного значення параметра, наприклад, t_x = 80, у програмі викликається спеціальна функція interval_membership_from_graph(). Ця функція дозволяє «вичитати» з графіка функції

належності діапазон значень належності, у якому може перебувати цей параметр. Для реалізації такого підходу використовується лінійна інтерполяція — тобто значення на кривій не обчислюються аналітично, а визначаються як зважене значення між найближчими точками на графіку:

```
interp_lower = interp1d(x_universe, lower_bound, bounds_error=False)
interp_upper = interp1d(x_universe, upper_bound, bounds_error=False)
return [float(interp_lower(value)), float(interp_upper(value))]
```

Таким чином, для будь-якого значення параметра система обчислює інтервал можливих значень належності до певного нечіткого терма. Наприклад, якщо параметр швидкість має значення 80, а функція належності терма "висока швидкість" у цій точці дає інтервал [0.2, 0.7], це означає, що ступінь належності цього значення до терма "висока" лежить у цьому діапазоні, з урахуванням варіативності та невизначеності вхідних даних.

Під час аналізу об’єкта ці інтервали обчислюються для кожного параметра та кожного лінгвістичного терма, що входить до правил. Це реалізовано у функції analyze_object(), де для кожної змінної відбувається доступ до відповідних меж інтервалів та виклик функції інтерполяції:

```
for label, mf in membership_functions[i].items():
 lower_bound, upper_bound = interval_bounds[i][label]
 inter = interval_membership_from_graph(x, lower_bound, upper_bound,
val)
 mf_ints[label] = inter
```

Кожне правило класифікації включає набір умов, які перевіряють значення параметрів на відповідність лінгвістичним термам. Щоб визначити, наскільки вся сукупність параметрів задовольняє конкретне правило, необхідно поєднати інтервали належності кожного з параметрів. Це реалізується як послідовний перетин інтервалів, що математично відображає принцип мінімальної підтримки умов у нечіткій логіці: рівень відповідності правила визначається за найслабшим елементом.

Операція перетину реалізується через функцію combine_intervals(), яка приймає два інтервали a та b у вигляді списків із двох чисел: [нижня межа, верхня межа]. У межах цієї функції визначається новий інтервал як "перетин" двох попередніх: верхня

межа вибирається як мінімум із обох верхніх меж, а нижня — як така, що відповідає більш вузькому інтервалу:

```
def combine_intervals(a, b):
    upper_a = a[1]
    upper_b = b[1]
    lower_a = a[0]
    lower_b = b[0]
    lower = lower_a if upper_a < upper_b else lower_b
    upper = min(upper_a, upper_b)
    return [lower, upper]
```

Цей принцип застосовується послідовно – спочатку поєднуються два перші інтервали, потім результат з третім, і так далі, поки не буде оброблено всі п’ять параметрів одного правила:

```
comb = combine_intervals(a, b)
comb = combine_intervals(comb, c)
comb = combine_intervals(comb, d)
comb = combine_intervals(comb, f)
```

На рисунку 3.5 зображено алгоритм визначення класу для вектора даних.

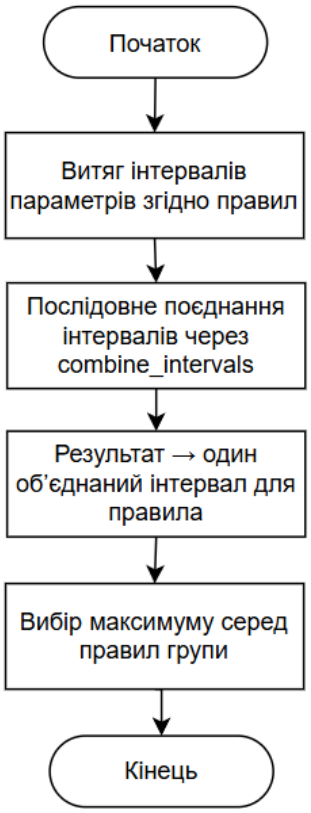


Рисунок 3.5 – Схема алгоритму визначення фінального інтервалу

У результаті цього багатоетапного комбінування для кожного правила формується один підсумковий інтервал належності, який відображає, наскільки об'єкт відповідає умовам саме цього правила. У разі, коли є кілька правил, що належать до одного класу, всі їх результати об’єднуються за допомогою пошуку найсильнішого інтервалу, тобто того, що має найвищу верхню межу (а при однаковій верхній — більшу нижню межу):

```
h = group_ints[0]
for interval in group_ints[1:]:
  h = max(h, interval)
```

Ця логіка реалізує принцип агрегації нечіткої інформації за домінуючим сценарієм: якщо існує декілька способів, якими об’єкт може задовольнити умови класу, обирається той варіант, який демонструє найвищу ймовірність належності. У підсумку, після обробки всіх класів, порівнюються між собою підсумкові інтервали (h_1, h_2, …), які були обчислені для кожного класу, і обирається той, який найбільше відповідає поведінці аналізованого об’єкта:

```
best_h = max(h_results, key=lambda k: (h_results[k][1],
h_results[k][0]))
```

Через комбінацію інтервалів, перетини та порівняння, реалізується нечіткий логічний висновок — математичне обґрунтування оцінки об'єкта на основі його параметрів.

3.6Тестування програмного засобу

При запуску користувач може вибрати декілька опцій роботи програми (рис. 3.6), але для початку потрібно вибрати потрібну базу даних у форматі .csv (рис. 3.7) і результат імпорту (рис. 3.8). Також можна подивитись на графіки належності (рис. 3.9).

Рисунок 3.6 – Опції роботи програми

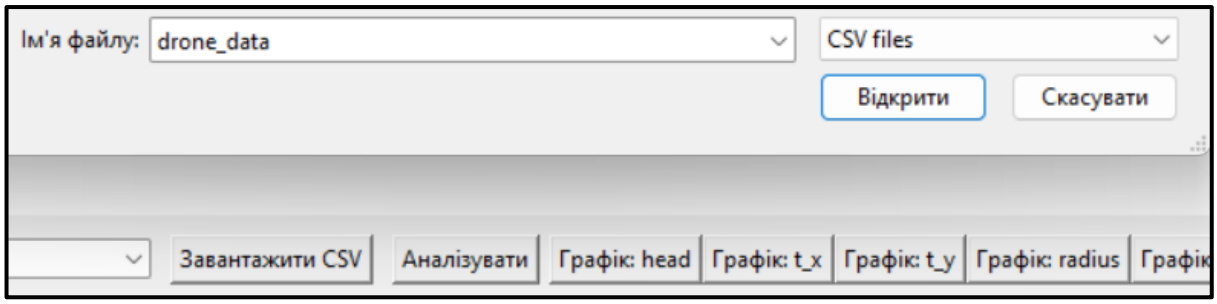


Рисунок 3.7 – Імпорт таблиці

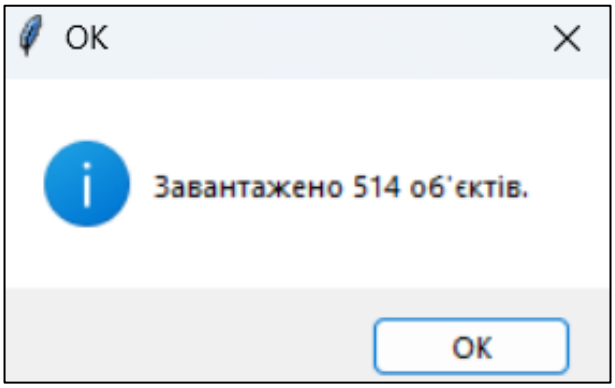


Рисунок 3.8 – Імпорт успішний

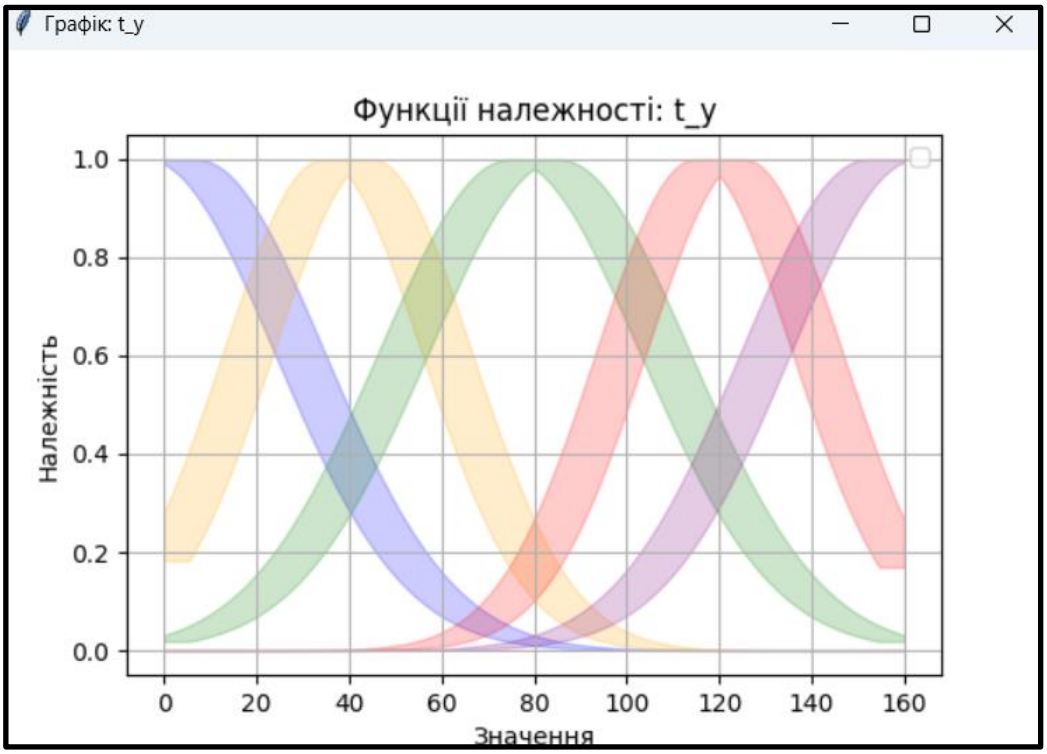


Рисунок 3.9 – Графік належності для параметра y

Далі користувач може проаналізувати певний вектор даних у самій програмі(рис 3.9) або експортувати у таблицю Excel для кращого аналізу (табл. 3.1)

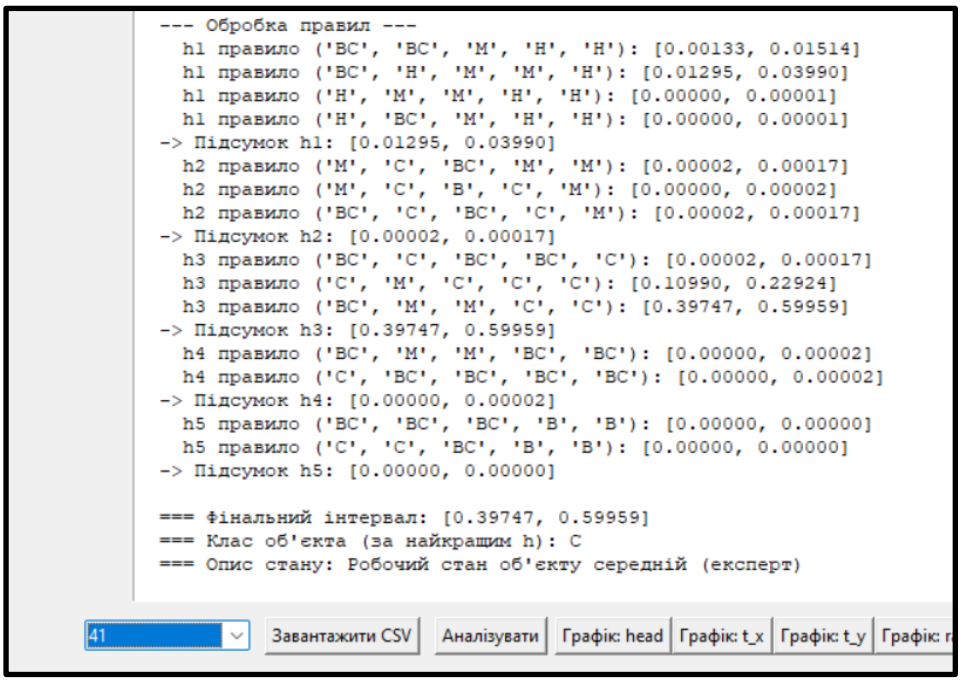


Рисунок 3.10 – Аналіз параметрів сорок першого об’єкта

Таблиця 3.1 – Приклад виводу

Index у таблиці	Head	T_x	T_y	T_r	Area	Клас за контуром	Фінальний клас (h)	Фінальний інтервал
625	274	132	96	14,42826	654	Маленький	М	[0.38334, 0.68367]
277	318	176	29	0,977205	3	Низький	Н	[0.55446, 0.76099]
412	106	154	133	12,73495	509,5	Маленький	М	[0.24810, 0.54290]
421	133	150	144	15,62	766,5	Маленький	М	[0.64115, 0.93088]
985	231	112	81	18,14201	1034	Середній	С	[0.16987, 0.31009]
626	274	132	96	14,42826	654	Маленький	М	[0.38334, 0.68367]
627	274	147	101	24,84677	1939,5	Середній	С	[0.51655, 0.81652]
631	278	233	106	35,99777	4071	Високий	В	[0.72912, 0.90455]
670	21	231	78	3,90882	48	Низький	Н	[0.12516, 0.30941]

3.7 Оптимізація за допомогою генетичного алгоритму

Оптимізація функцій належності в програмі виконується за допомогою генетичного (генетичного) алгоритму, який автоматизує процес підбору найкращих

параметрів для гаусової функції – центру (mean) та ширини (delta). Метою цієї оптимізації є побудова таких функцій належності, які найточніше відтворюють форму бажаного профілю — так званої таргет-функції (еталону), що задається на основі попередньої експертної оцінки або апріорного графіка.

На першому етапі для кожної лінгвістичної змінної (наприклад, «низький», «середній», «високий») формується еталонна функція належності. Вона створюється на основі початково заданих параметрів mean_init та delta_init і представляє собою ціль, яку треба наблизити. Це реалізується через функцію:

```
y_target = generate_target(x, mean_init, delta_init)
```

де `x` — це масив значень універсальної множини, а `y_target` — відповідні значення функції належності.

Далі визначається функція пристосованості (fitness function), яка обчислює середньоквадратичну помилку (MSE) між значеннями еталонної функції та поточними значеннями функції, побудованої з певними mean і delta. Ця помилка і є цільовою метрикою, яку алгоритм намагається мінімізувати, тобто знайти такі параметри, за яких помилка буде найменшою. Щоб алгоритм працював, визначаються структура індивіда та тип його оцінки:

```
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
```

Один індивід представляє собою пару параметрів — значення mean і delta. Початкові значення індивідів генеруються випадковим чином у межах допустимих діапазонів: центр вибирається в межах x.min() і x.max(), а ширина — у межах від дуже малого значення до половини ширини універсуму:

```
toolbox.register("individual", tools.initCycle, creator.Individual,
(lambda: np.random.uniform(x.min(), x.max()), lambda:
np.random.uniform(1e-3, (x.max() - x.min()) / 2)), n=1)
```

Після цього формується початкова популяція, над якою виконуються типові для еволюційних алгоритмів операції: схрещування (crossover), мутація (mutation) та відбір (selection). Еволюція відбувається упродовж кількох поколінь — у кожному

новому поколінні особини з кращими характеристиками мають більше шансів на виживання:

```
algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.3, ngen=40,
verbose=False)
```

Після завершення всіх ітерацій з усієї популяції обирається найкращий індивід — тобто та комбінація mean і delta, яка дає найменшу середньоквадратичну помилку порівняно з еталонною функцією:

```
best = tools.selBest(pop, 1)[0]
return best[0], best[1]
```

Ці знайдені оптимальні значення mean_opt та delta_opt використовуються для побудови вже оптимізованої гаусової функції належності. Вона зберігається у загальну структуру, яка містить функції для всіх класів і всіх параметрів:

```
optimized_membership_functions[param_idx][class_name] = gaussian_mf(x,
mean_opt, delta_opt)
```

Наприкінці, після всіх оптимізацій, кожна функція проходить повторну оцінку, де за її графіком визначається нове значення центру – це точка максимуму кривої:

```
peak_idx = np.argmax(y)
imean = x[peak_idx]
```

Ширина функції визначається як півширина на рівні значення `e^(-1)`, тобто на висоті ≈ 0.3679 від максимуму. Відстань між точками падіння функції ліворуч і праворуч від піку дає нову delta:

```
delta = (right_x - left_x) / 2 if right_x > left_x else 1.0
```

Отримані оптимізовані графіки належності зображені на рисунках 3.10-3.12

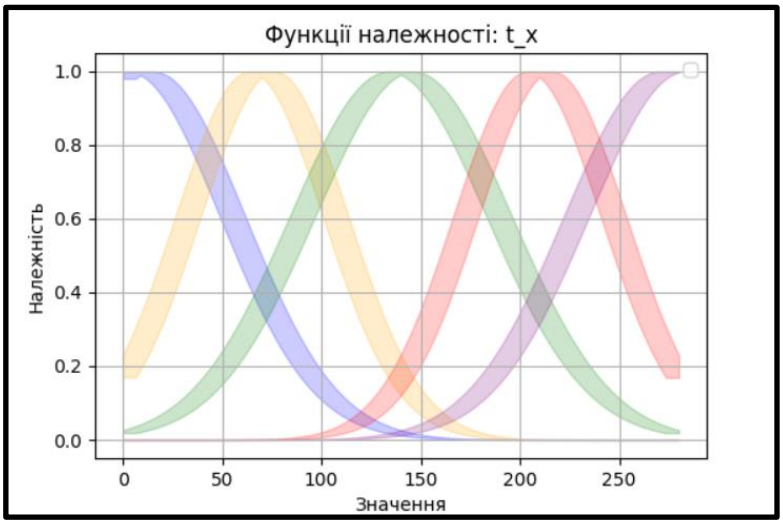


Рисунок 3.10 – Оптимізований графік належності для параметру x

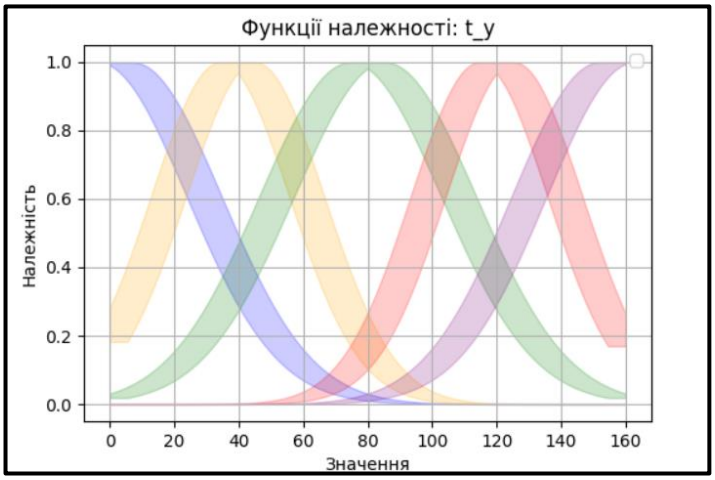


Рисунок 3.11 – Оптимізований графік належності для параметру y

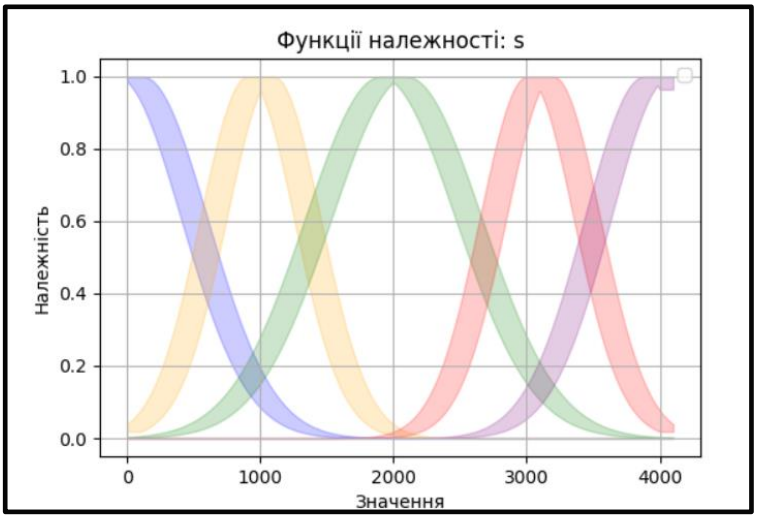


Рисунок 3.12 – Оптимізований графік належності для параметру S(area)

У цьому розділі роботи було реалізовано повнофункціональний програмний засіб, побудований на мові Python із використанням графічного інтерфейсу, розрахований на роботу з інтервальними нечіткими моделями. Основна увага була приділена інтеграції математичної моделі з візуально зрозумілим користувацьким інтерфейсом. Програма дозволяє завантажити вхідні дані у форматі таблиць, розрахувати інтервали належності для кожного з параметрів, виконати класифікацію об'єкта згідно з наперед заданими правилами та отримати підсумковий результат у зручному для сприйняття вигляді. Усі обчислення виконуються у фоновому режимі, а користувач взаємодіє з інтерфейсом через кнопки, випадаючі списки та діалогові вікна, що робить систему інтуїтивно зрозумілою навіть для користувачів без спеціальної технічної підготовки.

Окремо була реалізована можливість візуалізації функцій належності у вигляді графіків, що дозволяє оцінити поведінку кожного параметра в системі нечітких множин. Це забезпечує додаткову прозорість у прийнятті рішень і дозволяє експертам або аналітикам перевірити, наскільки коректно сформовані інтервали та як поводяться функції при зміні вхідних значень. Завдяки цьому розроблений інструмент може використовуватись як у навчальних цілях, так і для демонстрації принципів нечіткого виводу на реальних прикладах.

Програмна архітектура побудована модульно — кожна логічна частина (аналіз, графіка, збереження даних, оптимізація) реалізована окремими функціями або класами, що сприяє масштабованості, спрощує тестування окремих компонентів, а також дозволяє легко додавати новий функціонал у майбутньому. Такий підхід також значно покращує адаптацію системи під зміну вхідних вимог або специфіку нових прикладних задач.

Для підвищення точності класифікації було додатково впроваджено модуль оптимізації параметрів функцій належності з використанням генетичного алгоритму. Цей модуль забезпечує автоматичний підбір оптимальних значень центрів та ширин функцій (mean, delta), мінімізуючи похибки, що виникають при ручному налаштуванні. Завдяки цьому система демонструє здатність адаптуватися до специфіки нових вхідних даних, підвищуючи ефективність класифікації навіть у

випадках, коли розподіл параметрів суттєво відрізняється від попереднього навчального набору.

Результати класифікації можна експортувати у формат Excel, що значно полегшує подальший аналіз, зберігання звітності або передачу результатів зовнішнім системам. Кожен рядок у таблиці супроводжується не лише фінальним класом, а й інтервалами належності, що дозволяє детально простежити логіку прийнятого рішення. Така реалізація поєднує аналітичну точність із зручністю взаємодії, забезпечуючи ефективне застосування створеного інструменту в умовах практичних задач, включаючи автоматизоване спостереження, аналіз поведінки об'єктів та ситуаційний моніторинг у системах безпеки.

Крім того, передбачено можливість роботи з новими наборами даних без необхідності модифікації коду, що робить програму придатною для повторного використання у різних сценаріях. Враховуючи підтримку нечіткої логіки, інтервального підходу та модулів оптимізації, розроблений засіб становить собою універсальну платформу для побудови гнучких класифікаційних моделей із високим рівнем пояснюваності результатів.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено програмний засіб на мові Python, призначений для ідентифікації параметрів рухомого об’єкта на основі інтервальної нечіткої логіки. Основною метою проєкту було створення системи, здатної класифікувати характеристики руху об’єктів, зокрема безпілотних літальних апаратів (БПЛА), на основі аналізу даних з відеоспостереження.

У першому розділі було проаналізовано сучасні загрози, пов’язані з використанням БПЛА, та обґрунтовано необхідність створення автоматизованих систем моніторингу. Також розглянуто основи нечіткої логіки, особливості іінтервальних функцій належності та переваги їх застосування в умовах невизначеності. Детально описано математичні підходи до побудови моделі класифікації, формування бази правил і методи агрегації нечітких оцінок.

Особливу увагу приділено алгоритму оптимізації параметрів функцій належності за допомогою генетичного алгоритму. Було успішно протестовано роботу програми на експериментальних даних, що підтвердило її ефективність. Серед основних переваг розробленого програмного засобу варто відзначити його здатність адаптуватися до нечітких та неповних вхідних даних, що є критично важливим у задачах, пов’язаних із аналізом поведінки рухомих об’єктів. Інтерфейс програми побудований таким чином, щоб бути максимально зрозумілим і зручним для користувача, що спрощує роботу навіть для тих, хто не має досвіду в нечіткій логіці. Водночас реалізована структура забезпечує гнучкість у зміні правил, термів і параметрів функцій належності, що дозволяє адаптувати систему під конкретні умови або сценарії застосування. Окремою перевагою є автоматизоване виконання розрахунків — інтервали належності та результати класифікації формуються без втручання користувача, що забезпечує швидкість і стабільність роботи засобу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tracking the Threat: Key Takeaways from Recent Drone Incidents - D-Fend Solutions. *D-Fend Solutions*. URL: https://d-fendsolutions.com/blog/key-takeaways-from-recent-drone-incidents/ (дата звернення: 12.04.2025).
2. Михальов Д.В., Сокол Г.І. Розрахунок характеристик акустичного поля гвинтів БПЛА: XXIII Міжнародна молодіжна науково-практична конференція «Людина і космос»: Збірник тез. Дніпро, 2021. С.10. (дата звернення: 21.05.2025).
3. Support für DJI FPV - DJI. DJI. URL: https://www.dji.com/global/support/product/dji-fpv (дата звернення: 21.05.2025).
4. Що таке кластеризація? | Солікс. *Solix Technologies, Inc.* URL: https://www.solix.com/uk/kb/clustering/ (дата звернення: 22.07.2025).S
5. Дискримінантний аналіз - що це таке, визначення та поняття. *Economy-Pedia.com*. URL: https://uk.economy-pedia.com/11040389-discriminant-analysis (дата звернення: 23.05.2025).
6. Довідник по machine learning – decision tree in machine learning | база знань IT. *База знань IT технологій*. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/decision-tree-in-machine-learning (дата звернення: 24.05.2025).
7. Fuzzy Logic Inference System. *Tutorials on Technical and Non Technical Subjects*. URL: https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_inference_system.htm (дата звернення: 21.04.2025).
8. Type-1 and Interval Type-2 Fuzzy Systems. *Type-1 and Interval Type-2 Fuzzy Systems*. URL: https://fuzzysystem.github.io/ (дата звернення: 21.05.2025).
9. GeeksforGeeks. Genetic Algorithms: https://www.geeksforgeeks.org/genetic-algorithms/ (дата звернення: 21.04.2025).
10. Олександр Тартачний. Як працюють генетичні алгоритми. *robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс*. URL: https://robotdreams.cc/uk/blog/186-kak-rabotayut-geneticheskie-algoritmy (дата звернення: 21.04.2025).
11. Yepifanova I. Yu., Dzhedzhula V. V. (2021) Modelling of potential level of industrial enterprises. WSEAS Transactions on Environment and Development. Vol. 17, pp. 556-565
12. Кондратенко Н.Р. Використання нечітких баз знань з функціями належності типу-2 у медичній діагностиці / Н.Р. Кондратенко// Матеріали статей Міжнародної науково-практичної конференції«Актуальні задачі медичної, біологічної фізики таінформатики».-2022р.-Вінниця

13. Операції над множинами. *Stud*. URL: https://stud.com.ua/119335/prirodoznavstvo/operatsiyi_mnozhinami (дата звернення: 24.05.2025).
14. Welcome to python.org. *Python.org*. URL: https://www.python.org/ (дата звернення: 25.05.2025).
15. Python interface to Tcl/Tk. *Python documentation*. URL: https://docs.python.org/uk/3.13/library/tkinter.html (дата звернення: 25.05.2025).
16. JetBrains. PyCharm: єдине середовище розробки Python, яке вам потрібне. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 15.04.2025).
17. GeeksforGeeks. Top 6 Machine Learning Classification Algorithms URL: https://www.geeksforgeeks.org/top-6-machine-learning-algorithms-for-classification/ (дата звернення: 21.05.2025).
18. Belcic I. What is Classification in Machine Learning? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/classification-machine-learning (дата звернення: 21.05.2025).
19. Krivonos A. Класифікація дронів: які види та типи бувають? Частина перша. Bezpeka-Shop.com.URL: https://www.bezpeka-shop.com/ua/blog/poleznye-sovety/klassifikatsiya-dronov-kakie-vidy-i-tipy-byvayut-chast pervaya/?srsltid=AfmBOoqB5kb3dJVl4GkQNS9Zj0oD3l0zNTvsJ5AWdpzudEyiTuuEupCJ (дата звернення: 21.05.2025).
20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 125 «Кібербезпека» освітньо-професійні програми «Безпека інформаційних і комунікаційних систем» та «Кібербезпека критичних систем» / Уклад. В. А. Каплун, О. П. Войтович. – Вінниця: ВНТУ, 2023. – 61 с.

ДОДАТКИ

4

Додаток А
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний засіб ідентифікації рухомого об'єкта

Автор роботи: Матвєєв Кирил Костянтинович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ , група 1 БКС-21 б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **1,65** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д.т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Кібербезпека критичних систем» к. т. н., доц. _____ Юрій БАРИШЕВ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Кондрабенко Н.Р.

Здобувач _____ Матвєєв К.К

Додаток Б

ЛІСТИНГ ПРОГРАМИ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from scipy.interpolate import interp1d

root = tk.Tk()
root.title("Fuzzy Analyzer")
root.geometry("1000x800")

output = tk.Text(root, height=45, width=95)
output.pack(pady=10)

frame = tk.Frame(root)
frame.pack()

combo = ttk.Combobox(frame, state="readonly", width=15)
combo.pack(side=tk.LEFT, padx=5)

def gaussian_mf(x, mean, delta):
    return np.exp(-((x - mean) / delta) ** 2)


def interval_membership_from_graph(x_universe, lower_bound,
upper_bound, value):
    if value <= x_universe[0]:
        return [lower_bound[0], upper_bound[0]]
    elif value >= x_universe[-1]:
        return [lower_bound[-1], upper_bound[-1]]
    interp_lower = interp1d(x_universe, lower_bound,
bounds_error=False)
    interp_upper = interp1d(x_universe, upper_bound,
bounds_error=False)
    return [float(interp_lower(value)), float(interp_upper(value))]


def combine_intervals(a, b):
    upper_a = a[1]
    upper_b = b[1]
    lower_a = a[0]
    lower_b = b[0]
    lower = lower_a if upper_a < upper_b else lower_b
    upper = min(upper_a, upper_b)
    return [lower, upper]

param_names = ['head', 't_x', 't_y', 't_r', 's']
```

```
universes = {
    0: np.linspace(0, 450, 1000),
    1: np.linspace(0, 280, 1000),
    2: np.linspace(0, 160, 1000),
    3: np.linspace(0, 36, 100),
    4: np.linspace(0, 4100, 6000)
}
deltas = {
    0: 7.5,    # head
    1: 6.5,    # t_x
    2: 5.5,    # t_y
    3: 2.2,  # t_r
    4: 100   # s (contour_area)
}


membership_functions = {
    0: {
        'H': gaussian_mf(universes[0], 5.86, 87.39),
        'M': gaussian_mf(universes[0], 112.61, 109.91),
        'C': gaussian_mf(universes[0], 224.77, 90.32),
        'BC': gaussian_mf(universes[0], 312.16, 110.14),
        'B': gaussian_mf(universes[0], 448.20, 89.19),
    },
    1: {
        'H': gaussian_mf(universes[1], 9.01, 65.32),
        'M': gaussian_mf(universes[1], 69.97, 52.55),
        'C': gaussian_mf(universes[1], 139.94, 70.12),
        'BC': gaussian_mf(universes[1], 209.91, 52.55),
        'B': gaussian_mf(universes[1], 276.58, 68.17),
    },
    2: {
        'H': gaussian_mf(universes[2], 1.70, 39.05),
        'M': gaussian_mf(universes[2], 39.65, 30.38),
        'C': gaussian_mf(universes[2], 79.98, 39.99),
        'BC': gaussian_mf(universes[2], 119.97, 30.04),
        'B': gaussian_mf(universes[2], 157.24, 38.54),
    },
    3: {
        'H': gaussian_mf(universes[3], 1.62, 8.28),
        'M': gaussian_mf(universes[3], 8.89, 7.27),
        'C': gaussian_mf(universes[3], 18.18, 9.09),
        'BC': gaussian_mf(universes[3], 27.07, 7.07),
        'B': gaussian_mf(universes[3], 36.77, 9.49),
    },
    4: {
        'H': gaussian_mf(universes[4], 18.22, 688.08),
        'M': gaussian_mf(universes[4], 1000.11, 500.06),
        'C': gaussian_mf(universes[4], 1999.78, 800.31),
        'BC': gaussian_mf(universes[4], 3099.90, 500.06),
        'B': gaussian_mf(universes[4], 3980.00, 632.96),
    },
}
```

```
def compute_interval_bounds(universes, membership_functions, deltas):
    interval_bounds = {}
    for param_idx, mfs in membership_functions.items():
        x = universes[param_idx]
        delta = deltas[param_idx]
        interval_bounds[param_idx] = {}

        for class_name, mf in mfs.items():
            lower_bound = []
            upper_bound = []
            for xi in x:
                left = xi - delta
                right = xi + delta
                mask = (x >= left) & (x <= right)
                vals = mf[mask]
                lower_bound.append(np.min(vals) if len(vals) else 0)
                upper_bound.append(np.max(vals) if len(vals) else 0)
            interval_bounds[param_idx][class_name] = (lower_bound,
upper_bound)
    return interval_bounds
grouped_rules = {
     'h1': [
        ('BC', 'BC', 'M','H', 'H'),
        ('BC', 'H', 'M','M', 'H'),
        ('H', 'M', 'M','H', 'H'),
        ('H', 'BC', 'M','H', 'H')
    ],
    'h2': [
        ('M', 'C', 'BC','M', 'M'),
        ('M', 'C', 'B', 'C', 'M'),
        ('BC', 'C', 'BC','C', 'M')
    ],
    'h3': [
        ('BC', 'C', 'BC', 'BC', 'C'),
        ('C', 'M', 'C', 'C', 'C'),
        ('BC', 'M', 'M', 'C', 'C')
    ],
    'h4': [
        ('BC', 'M', 'M', 'BC', 'BC'),
        ('C', 'BC', 'BC', 'BC', 'BC')
    ],
    'h5': [
        ('BC', 'BC', 'BC', 'B', 'B'),
        ('C', 'C', 'BC', 'B', 'B')
    ]
}

def show_context_menu(event):
    context_menu = tk.Menu(root, tearoff=0)
    context_menu.add_command(label="Копіювати", command=lambda:
output.event_generate("<<Copy>>"))
    context_menu.tk_popup(event.x_root, event.y_root)
```

```
output.bind("<Button-3>", show_context_menu)  # ПКМ по вікну

def plot_gui(param_idx):
    x = universes[param_idx]
    colors = ['blue', 'orange', 'green', 'red', 'purple']
    fig = Figure(figsize=(6, 4), dpi=100)
    ax = fig.add_subplot(111)

    for j, (class_name, mf) in
enumerate(membership_functions[param_idx].items()):
        lower_bound = []
        upper_bound = []
        for xi in x:
            left = xi - deltas[param_idx]
            right = xi + deltas[param_idx]
            mask = (x >= left) & (x <= right)
            vals = mf[mask]
            lower_bound.append(np.min(vals) if len(vals) else 0)
            upper_bound.append(np.max(vals) if len(vals) else 0)
        # ax.plot(x, lower_bound, '--', label=f"{class_name} (низ)",
color=colors[j % len(colors)])
        # ax.plot(x, upper_bound, '-', label=f"{class_name} (верх)",
color=colors[j % len(colors)])
        ax.fill_between(x, lower_bound, upper_bound, alpha=0.2,
color=colors[j % len(colors)])

    ax.set_title(f"Функції належності: {param_names[param_idx]}")
    ax.set_xlabel("Значення")
    ax.set_ylabel("Належність")
    ax.grid(True)
    ax.legend()


    canvas_frame = tk.Toplevel(root)
    canvas_frame.title(f"Графік: {param_names[param_idx]}")
    canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
def classify_area(area):
    if area < 400:
        return 'Низький'
    elif area < 1000:
        return 'Маленький'
    elif area < 2100:
        return 'Середній'
    elif area < 3100:
        return 'Вище середнього'
    else:
        return 'Високий'
def export_to_excel():
    try:
        results = []
```

```
        interval_bounds = compute_interval_bounds(universes,
membership_functions, deltas)

        for idx in range(len(data_matrix)):
            row = data_matrix[idx]
            all_intervals = {}

            for i, val in enumerate(row):
                mf_ints = {}
                for label in membership_functions[i].keys():
                    lower, upper = interval_bounds[i][label]
                    inter =
interval_membership_from_graph(universes[i], lower, upper, val)
                    mf_ints[label] = inter
                all_intervals[i] = mf_ints

            h_results = {}
            for h_key, rules in grouped_rules.items():
                group_ints = []
                for rule in rules:
                    a = all_intervals[0][rule[0]]
                    b = all_intervals[1][rule[1]]
                    c = all_intervals[2][rule[2]]
                    d = all_intervals[3][rule[3]]
                    f = all_intervals[4][rule[4]]

                    comb = combine_intervals(a, b)
                    comb = combine_intervals(comb, c)
                    comb = combine_intervals(comb, d)
                    comb = combine_intervals(comb, f)

                    group_ints.append(comb)

                h = group_ints[0]
                for interval in group_ints[1:]:
                    h = max(h, interval)
                h_results[h_key] = h

            h_classes = {
                'h1': 'H',
                'h2': 'M',
                'h3': 'C',
                'h4': 'BC',
                'h5': 'B'
            }

            best_h = max(h_results, key=lambda k: (h_results[k][1],
h_results[k][0]))
            final_class = h_classes[best_h]

            class_descriptions = {
            'H': "Параметри руху об’єкта повністю відповідають умовам
безпечного розташування (за думкою експерта)",
```

```
            'M': "Параметри руху об’єкта здебільшого відповідають
умовам безпечного розташування (за думкою експерта)",
            'C': "Параметри руху об’єкта частково відповідають умовам
безпечного розташування (за думкою експерта)",
            'BC': "Параметри руху об’єкта слабо відповідають умовам
безпечного розташування (за думкою експерта)",
            'B': "Параметри руху об’єкта не відповідають умовам
безпечного розташування (за думкою експерта)"

        }
            description = class_descriptions.get(final_class,
"Невідомий стан")

            # Клас за площею
            real_index = df.index[idx]
            area_value = df.loc[real_index, 'contour_area']
            area_class = classify_area(area_value)

            results.append({
                "Index у таблиці": real_index,
                "Head": row[0],
                "T_x": row[1],
                "T_y": row[2],
                "T_r": row[3],
                "Area": area_value,
                "Клас за контуром": area_class,
                "Фінальний клас (h)": final_class,
                "Фінальний інтервал": f"[{h_results[best_h][0]:.5f},
{h_results[best_h][1]:.5f}]",
                "Опис стану": description
            })

        df_out = pd.DataFrame(results)
        save_path =
filedialog.asksaveasfilename(defaultextension=".xlsx",
filetypes=[("Excel files", "*.xlsx")])
        if save_path:
            df_out.to_excel(save_path, index=False)
            messagebox.showinfo("Успіх", "Експорт завершено успішно!")

    except Exception as e:
        messagebox.showerror("Помилка при експорті", str(e))


def analyze_object():
    output.delete("1.0", tk.END)
    try:
        idx = int(combo.get())
        row = data_matrix[idx]
        real_index = df.index[idx]
        area_value = df.loc[real_index, 'contour_area']
        area_class = classify_area(area_value)
```

```
        output.insert(tk.END, f"\n  -> Площа контуру: {area_value:.2f}
→ Клас площі: {area_class}\n")
        output.insert(tk.END, f"=== Об'єкт {idx} (індекс у таблиці:
{df.index[idx]}) ===\n")
        all_intervals = {}
        interval_bounds = compute_interval_bounds(universes,
membership_functions, deltas)
        for i, val in enumerate(row):
            output.insert(tk.END, f"  {param_names[i]}: {val}\n")
            mf_ints = {}
            x = universes[i]

            for label, mf in membership_functions[i].items():
                lower_bound, upper_bound = interval_bounds[i][label]
                inter = interval_membership_from_graph(x, lower_bound,
upper_bound, val)
                mf_ints[label] = inter
                output.insert(tk.END, f"  '{label}': [{inter[0]:.5f},
{inter[1]:.5f}]\n")
            all_intervals[i] = mf_ints
        output.insert(tk.END, "\n  --- Обробка правил ---\n")
        h_results = {}
        for h_key, rules in grouped_rules.items():
            group_ints = []
            for rule in rules:
                a = all_intervals[0][rule[0]]
                b = all_intervals[1][rule[1]]
                c = all_intervals[2][rule[2]]
                d = all_intervals[3][rule[3]]
                f = all_intervals[4][rule[4]]
                comb = combine_intervals(a, b)
                comb = combine_intervals(comb, c)
                comb = combine_intervals(comb, d)
                comb = combine_intervals(comb, f)
                group_ints.append(comb)
                output.insert(tk.END, f"    {h_key} правило {rule}:
[{comb[0]:.5f}, {comb[1]:.5f}]\n")
            h = group_ints[0]
            for interval in group_ints[1:]:
                h = max(h, interval)
            h_results[h_key] = h
            output.insert(tk.END, f"  -> Підсумок {h_key}:
[{h[0]:.5f}, {h[1]:.5f}]\n")
        h_classes = {
            'h1': 'H',
            'h2': 'M',
            'h3': 'C',
            'h4': 'BC',
            'h5': 'B'
        }

        # Вибір найкращого h: спочатку по верхній межі, при рівності —
по нижній
```

```
        best_h = max(h_results, key=lambda k: (h_results[k][1],
h_results[k][0]))
        final_class = h_classes[best_h]

        # Вивід
        output.insert(tk.END, f"\n  === Фінальний інтервал:
[{h_results[best_h][0]:.5f}, {h_results[best_h][1]:.5f}]\n")
        class_descriptions = class_descriptions = {
            'H': "Параметри руху об’єкта повністю відповідають умовам
безпечного розташування (за думкою експерта)",
            'M': "Параметри руху об’єкта здебільшого відповідають
умовам безпечного розташування (за думкою експерта)",
            'C': "Параметри руху об’єкта частково відповідають умовам
безпечного розташування (за думкою експерта)",
            'BC': "Параметри руху об’єкта слабо відповідають умовам
безпечного розташування (за думкою експерта)",
            'B': "Параметри руху об’єкта не відповідають умовам
безпечного розташування (за думкою експерта)"

        }
        description = class_descriptions.get(final_class, "Невідомий
стан")

        output.insert(tk.END, f"  === Клас об'єкта (за найкращим h):
{final_class}\n")
        output.insert(tk.END, f"  === Опис стану: {description}\n")

    except Exception as e:
        messagebox.showerror("Помилка", str(e))

def load_csv():
    file = filedialog.askopenfilename(filetypes=[("CSV files",
"*.csv")])
    if not file:
        return
    global df, data_matrix
    df = pd.read_csv(file)
    df = df[df['target_visible'] == 1]
    # data_matrix = df[['heading', 'target_x', 'target_y']].to_numpy()
    data_matrix = df[['heading', 'target_x', 'target_y',
'target_radius', 'contour_area']].to_numpy()
    combo['values'] = list(range(len(data_matrix)))
    combo.current(0)
    messagebox.showinfo("OK", f"Завантажено {len(data_matrix)}
об'єктів.")

tk.Button(frame, text="Завантажити CSV",
command=load_csv).pack(side=tk.LEFT, padx=5)
tk.Button(frame, text="Аналізувати",
command=analyze_object).pack(side=tk.LEFT, padx=5)
tk.Button(frame, text="Графік: head", command=lambda:
plot_gui(0)).pack(side=tk.LEFT)
```

```
tk.Button(frame, text="Графік: t_x", command=lambda:
plot_gui(1)).pack(side=tk.LEFT)
tk.Button(frame, text="Графік: t_y", command=lambda:
plot_gui(2)).pack(side=tk.LEFT)
tk.Button(frame, text="Графік: radius", command=lambda:
plot_gui(3)).pack(side=tk.LEFT)
tk.Button(frame, text="Графік: area", command=lambda:
plot_gui(4)).pack(side=tk.LEFT)
tk.Button(frame, text="Експортувати в Excel",
command=export_to_excel).pack(side=tk.LEFT, padx=5)

root.mainloop()
```

ІЛЮСТРАТИВНА ЧАСТИНА

Програмний засіб ідентифікації рухомого об’єкта

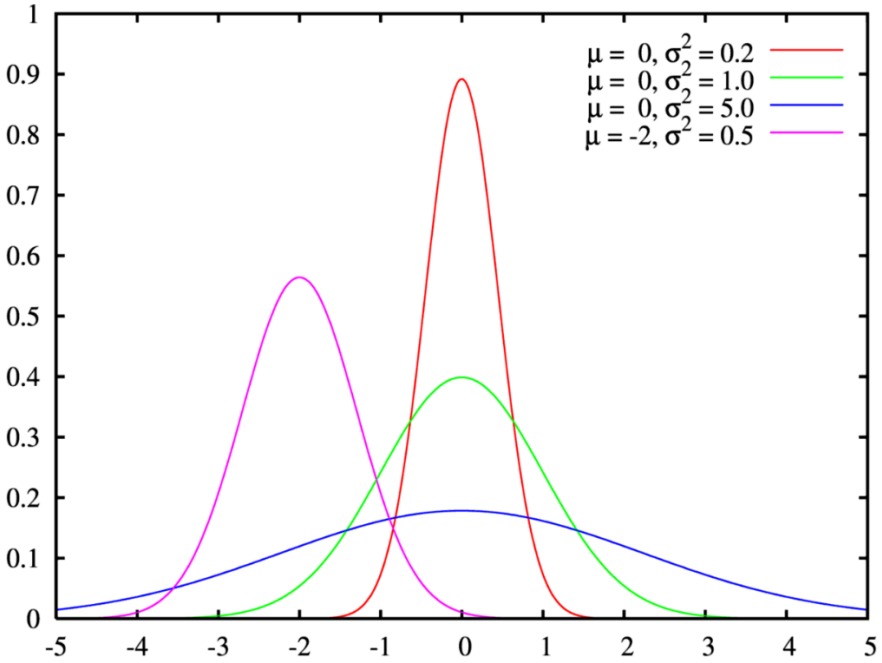
Виконав: студент 4 курсу групи
БКС-21б спеціальності 125
«Кібербезпека»

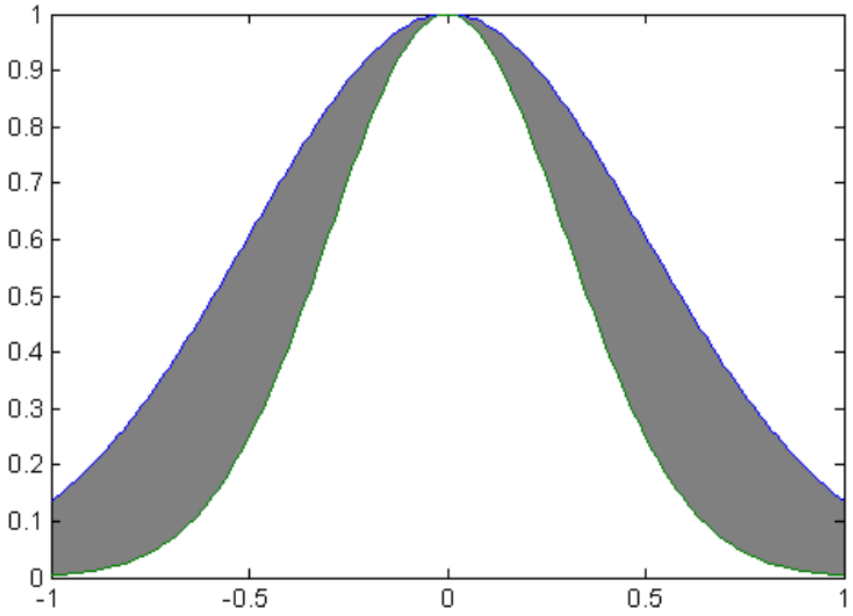
Кирил МАТВЄЄВ
«16» червня 2025 р.

Керівник: к.т.н., професор каф. ЗІ

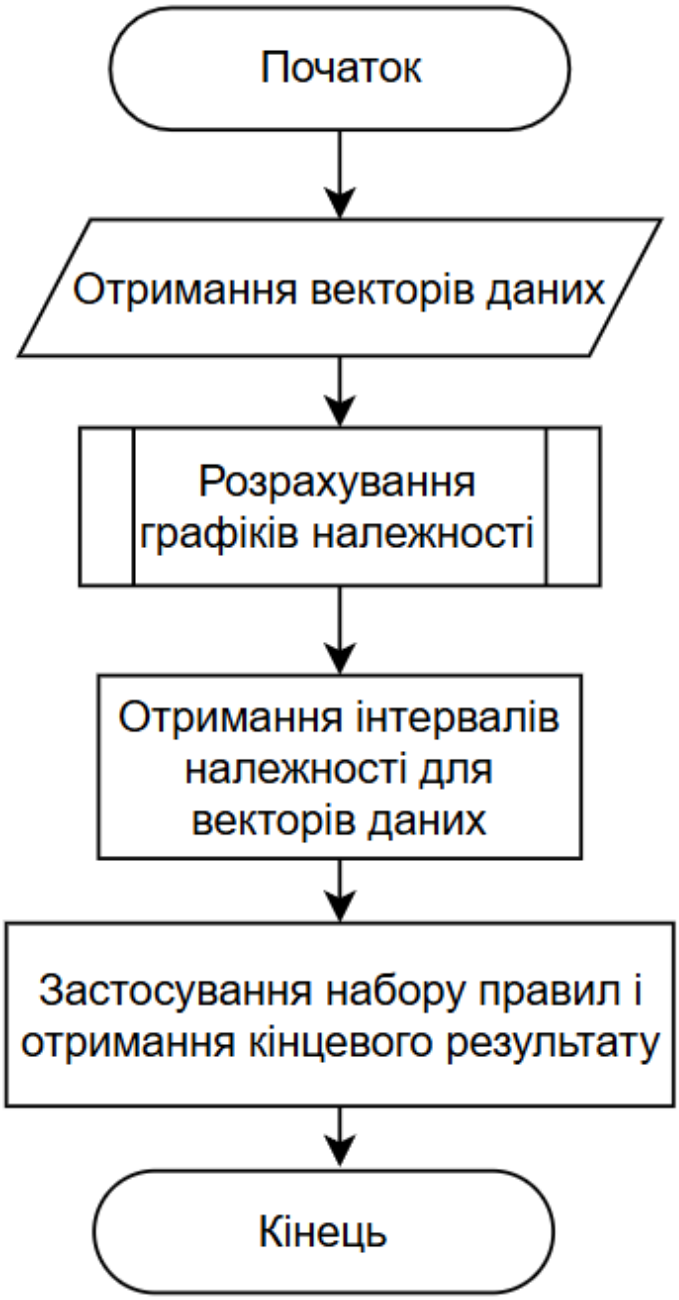
Наталія КОНДРАТЕНКО
«16» червня 2025 р.

ПРИКЛАД ГРАФІКІВ НАЛЕЖНОСТІ

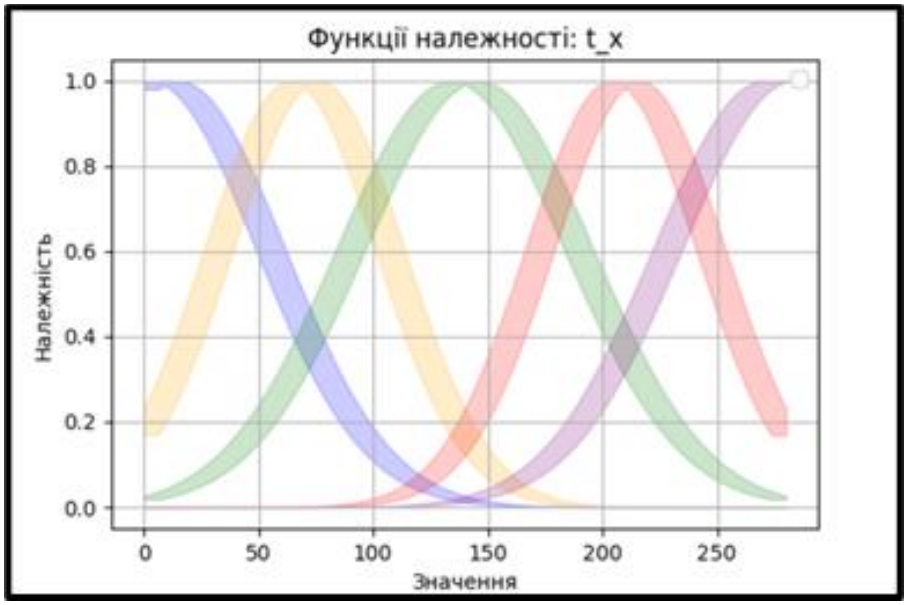


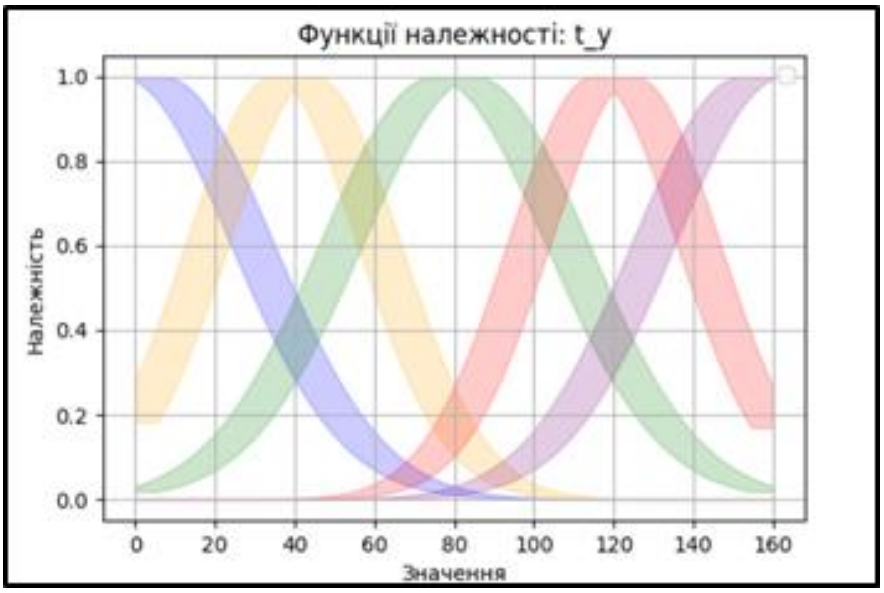


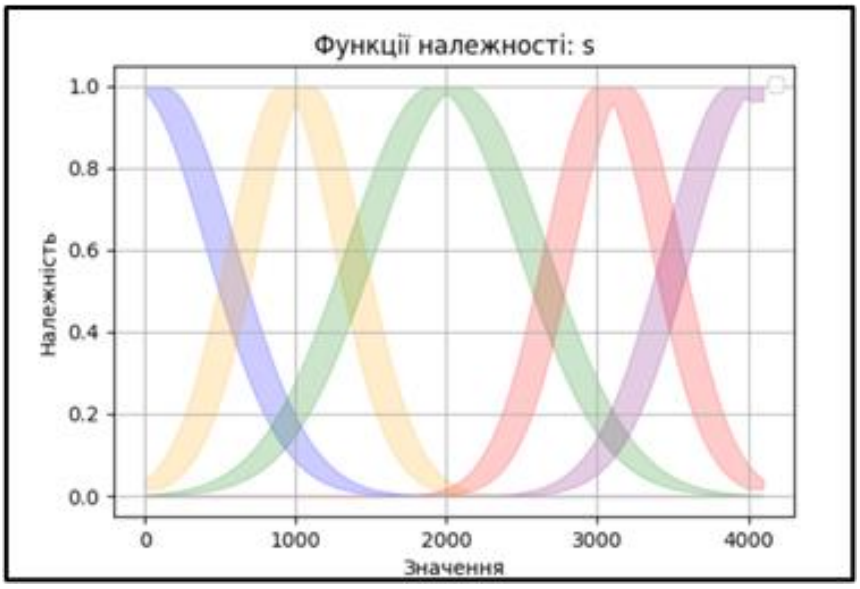
АЛГОРИТМ РОБОТИ ПРОГРАМИ



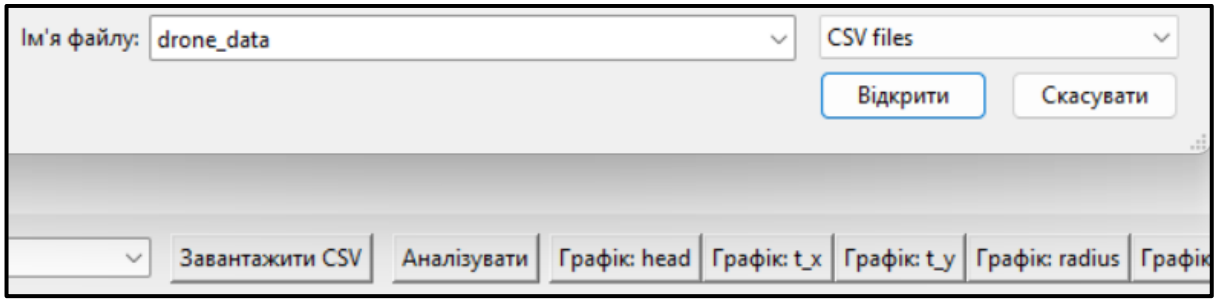
ОПТИМІЗОВАНІ ГРАФІКИ НАЛЕЖНОСТІ







ПРИКЛАД РОБОТИ ПРОГРАМИ



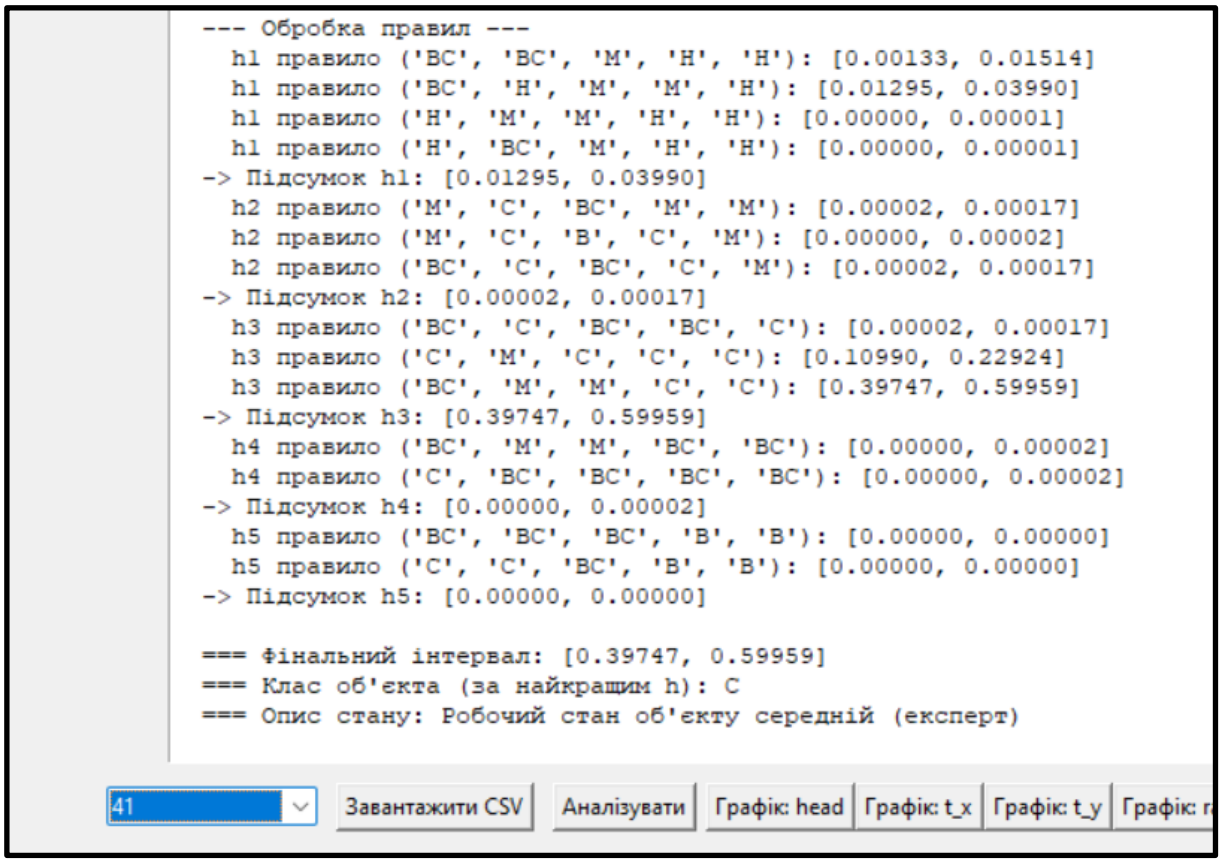
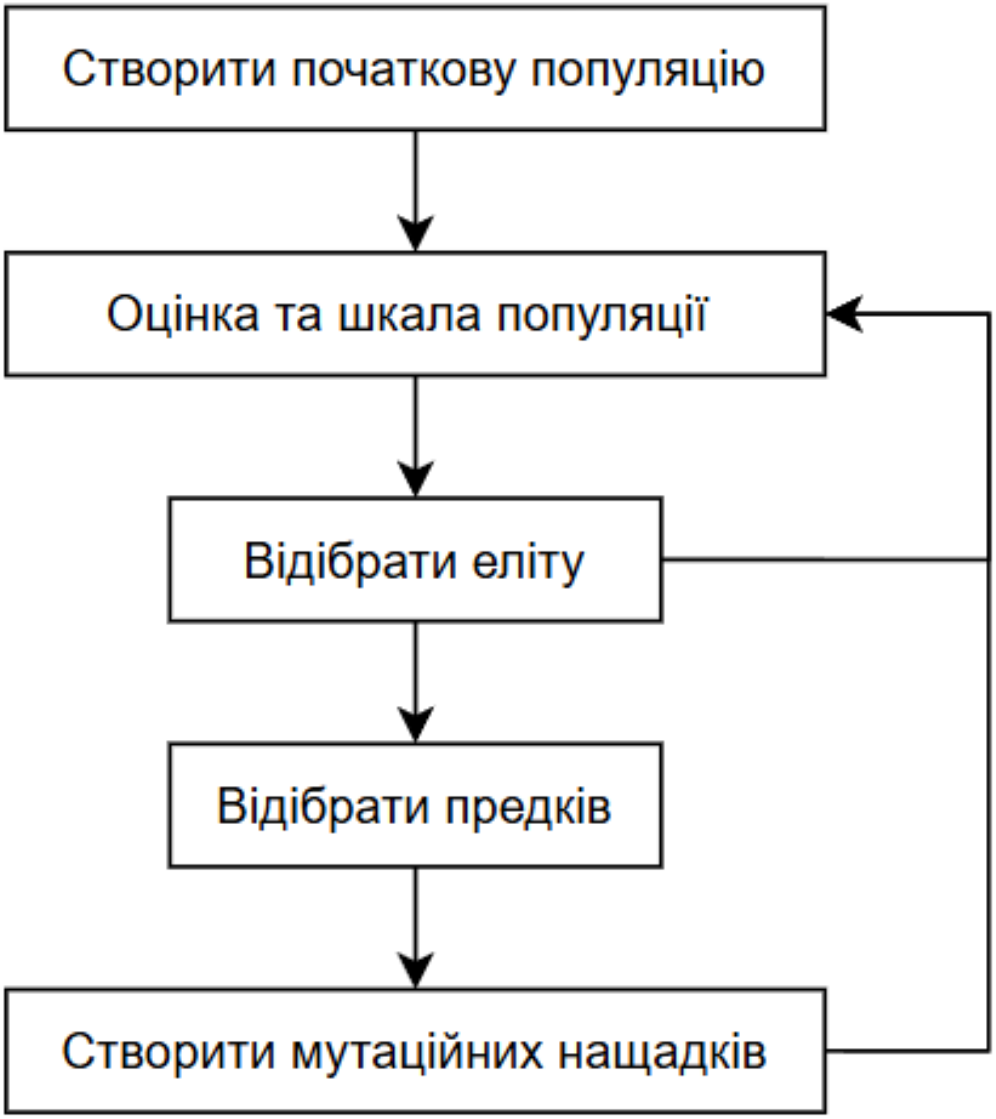


СХЕМА ГЕНЕТИЧНОГО АЛГОРИТМУ



АЛГОРИТМ ВИЗНАЧЕННЯ ФІНАЛЬНОГО ІНТЕРВАЛУ