ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Розробка навчального програмного забезпечення з теми "Операції з нечіткими числами"»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня бакалавра

Виконавець роботи Квіта Єлизавета Андріївна

_____ «___» _____ 202_ р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Чілікіна Тетяна Василівна

_____ «___» _____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2025

РЕФЕРАТ

Записка: 72 с., 14 рис., 2 таблиці, 3 додатки, 12 джерел

ОПЕРАЦІЇ З НЕЧІТКИМИ ЧИСЛАМИ

Об’єкт розробки – система для виконання арифметичних операцій з нечіткими числами, орієнтована на використання в задачах, де присутня невизначеність даних.

Мета роботи – розробка навчального програмного забезпечення з теми "Операції з нечіткими числами»

Методи дослідження – методи теорії нечітких множин, алгоритми для виконання арифметичних операцій з нечіткими числами, середовище розробки Visual Studio Code, мова програмування Java.

У роботі проведено огляд існуючих підходів до роботи з нечіткими числами, проаналізовано їх переваги та недоліки. Розроблено алгоритми для виконання основних арифметичних операцій над нечіткими числами, включаючи додавання, віднімання, множення та ділення. Описано методи побудови функцій належності для представлення нечітких чисел і правила виконання операцій з такими числами. На основі створених алгоритмів побудовано блок-схеми, які забезпечують зручне відображення основних етапів обчислень.

Здійснена програмна реалізація алгоритмів для роботи з нечіткими числами, а також проведено тестування реалізованого програмного забезпечення для перевірки коректності роботи та оцінки ефективності алгоритмів у різних сценаріях.

Результати роботи можуть бути використані для розв’язання задач з невизначеними вхідними даними, включаючи задачі в галузях управління, прогнозування, аналізу даних та штучного інтелекту.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
A	Нечітке число, представлене в вигляді нечіткої множини
Ã	Нечітка множина, що представляє нечітке число A.
$\mu_A(x)$	Функція належності нечіткої множини A~A~, яка визначає ступінь належності елемента xx до множини A~.
α-рівень	Підмножина нечіткої множини, яка містить всі елементи з належністю не менше, ніж α.
+ –	Операція додавання нечітких чисел
− –	Операція віднімання нечітких чисел
× –	Операція множення нечітких чисел
ò	Нечіткий нуль, який використовується для визначення відсутності кількісного значення в нечітких обчисленнях.
[a,b]	Інтервальний вигляд нечіткого числа, де a та b представляють нижню та верхню границі відповідно.
core(A~)	Ядро нечіткої множини A~, що є множиною точок з максимальною функцією належності.
supp(A~)	Носій нечіткої множини A~, який охоплює всі елементи з ненульовою функцією належності.
Fuzzy Logic (FL)	Нечітка логіка, математична логіка, яка дозволяє оперувати з невизначеними або нечіткими даними.
R –	Множина дійсних чисел.
0,1]	Інтервал значень функції належності, що показує ступінь належності елемента до нечіткої множини (від 0 до 1).

ВСТУП

Актуальність. У світі, де інформація й технології швидко розвиваються, дедалі більшого значення набувають методи обробки та аналізу неточних і неповних даних. В умовах невизначеності, коли традиційні методи не можуть забезпечити необхідної точності, актуальними стають методи нечіткої логіки. Нечіткі числа та операції з ними є важливими елементами в системах штучного інтелекту, прийняття рішень та автоматизації, що сприяє розробці адаптивних і гнучких систем управління та прогнозування.

Метою роботи є дослідження та програмна реалізація операцій з нечіткими числами для задач прийняття рішень в умовах невизначеності.

Об’єктом розробки в даній роботі є нечіткі системи, що дозволяють обробляти нечіткі дані для отримання інформативних рішень.

Предметом розробки є алгоритми й методи реалізації операцій з нечіткими числами та їх програмна реалізація в середовищі програмування.

Головне завдання – розробка навчального програмного забезпечення з теми "Операції з нечіткими числами» розробити програмний модуль для роботи з нечіткими числами, який може виконувати основні математичні операції над ними.

Перелік **використаних методів** полягає у застосуванні теорії нечітких множин, алгоритмів для роботи з нечіткими числами, а також принципів об’єктно-орієнтованого програмування. При реалізації модуля використано мову програмування Python, що дозволяє створити функціональний і надійний програмний інструмент для обробки нечітких даних.

Робота складається з трьох основних розділів. У першому розділі наведено інформаційний огляд основних підходів до роботи з нечіткими числами, їх переваг та недоліків. У другому розділі розглянуто теоретичні основи операцій з нечіткими числами, основні властивості та методи обчислень. У третьому розділі представлено алгоритми для виконання операцій з нечіткими числами,

обґрунтовано вибір програмного середовища та реалізовано розроблений алгоритм.

Обсяг пояснювальної записки: 60 стор., з них основна частина – 50 стор., джерела – 15 назв.

ПОСТАНОВКА ЗАДАЧІ

Основною задачею даної роботи є розробка методів і алгоритмів для виконання математичних операцій з нечіткими числами, що мають різні типи представлення, зокрема трикутні та трапецієподібні нечіткі числа. Це включає реалізацію базових арифметичних операцій — додавання, віднімання, множення та ділення. Основна мета розробки — створення програмного модуля, здатного обробляти нечіткі числа та забезпечувати надійні результати для прийняття рішень в умовах невизначеності. Даний модуль повинен забезпечити простий інтерфейс для користувачів, щоб спростити розрахунки з нечіткими даними.

У багатьох реальних задачах, таких як прогнозування ризиків, керування складними системами та прийняття рішень у невизначених ситуаціях, часто доводиться працювати з неповними або нечіткими даними. Наприклад, у фінансовій аналітиці для оцінки майбутньої прибутковості інвестицій часто використовують нечіткі значення ризику та прибутку. В екологічному моделюванні для оцінки впливу на довкілля параметри системи також можуть бути нечіткими через природні варіації та складність точного вимірювання. Проблема нечітких чисел полягає в тому, що вони не мають точного числового значення, а їх значення задається у вигляді діапазону можливих значень або функції належності. Це створює певні складнощі при виконанні математичних операцій, оскільки класичні методи не підходять для роботи з нечіткими величинами. Отже, необхідне спеціальне математичне й програмне забезпечення для роботи з ними.

Для ефективного вирішення задачі роботи з нечіткими числами необхідно враховувати такі критерії та методи:

1. Необхідно забезпечити максимально можливу точність у виконанні операцій над нечіткими числами, незважаючи на їхню природу невизначеності.

2. Операції над нечіткими числами повинні бути узгоджені з класичними арифметичними операціями, щоб забезпечити можливість комбінування нечітких і точних чисел у розрахунках.

3. Для роботи з нечіткими числами використовуються такі методи, як метод α-рівнів та метод інтервального аналізу, які дозволяють представляти нечітке число у вигляді набору інтервалів, що спрощує виконання арифметичних операцій.

4. Основою для роботи з нечіткими числами є функції належності, які визначають ступінь приналежності елемента до нечіткої множини. У даній роботі застосовуються трикутні та трапецієподібні функції належності для представлення нечітких чисел, оскільки вони є найбільш поширеними та зручними для математичних обчислень.

5. Програмна реалізація алгоритмів для операцій над нечіткими числами потребує вибору відповідної мови програмування. У даній роботі обрано мову Python, яка має розвинуті бібліотеки для роботи з нечіткими системами й дозволяє легко реалізувати необхідні алгоритми.

РОЗДІЛ 1.
ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Історія розвитку нечітких чисел і теорії нечітких множин

Теорія нечітких множин та нечітких чисел з'явилася як спроба розв'язати задачі, пов'язані з невизначеністю та неоднозначністю в реальних системах, де традиційна двійкова логіка не була ефективною. Основи цієї теорії заклав американський математик і інженер Лотфі Заде, який у 1965 році вперше опублікував статтю на тему нечітких множин. У цій статті Заде запропонував новий підхід до опису невизначеності, використовуючи математичну структуру, яка дозволяє елементам належати до множин не повністю, а частково, з певним ступенем належності. Це дозволило описувати різноманітні аспекти реального світу, в яких неможливо чітко визначити межі понять, наприклад, такі як "тепло", "вологість" або "швидкість". Ідеї Лотфі Заде мали великий вплив на подальший розвиток нечіткої логіки, хоча спершу вони не здобули широкого визнання у науковій спільноті. Однак, згодом, у 1970-80-х роках, нечітка логіка стала все більше привертати увагу завдяки своїм практичним застосуванням, особливо в Японії. Японські дослідники та інженери активно розробляли та впроваджували системи, які базувалися на нечітких множинах для управління промисловими процесами, транспортними системами та побутовою технікою. Японські компанії, такі як Matsushita, Hitachi та Mitsubishi, стали піонерами у використанні нечіткої логіки для автоматичного управління складними системами, такими як кондиціонери, пральні машини та навіть системи автоматичного гальмування для транспорту.

До 1990-х років теорія нечітких множин та нечітка логіка отримали визнання в усьому світі. Нечіткі множини почали застосовуватися не лише в промисловості, але й у різних галузях, таких як медицина, економіка, соціологія, екологія та інформатика. Завдяки можливості опрацьовувати нечітку інформацію, цей

підхід став корисним інструментом для моделювання складних процесів, де класичні математичні методи виявилися менш ефективними. Нечіткі числа, як розширення теорії нечітких множин, забезпечили ще більші можливості для опису і роботи з невизначеними або нечіткими даними.

Розвиток теорії нечітких множин привів до створення різних видів нечітких чисел і функцій належності, які використовуються для опису нечітких значень у різних системах. Важливо зазначити, що на початкових етапах нечітка логіка мала обмежене застосування через відсутність відповідної обчислювальної потужності для обробки великих обсягів нечітких даних. З розвитком комп'ютерної техніки та математичного апарату можливості застосування нечіткої логіки розширилися, що призвело до її активного впровадження в багатьох прикладних областях.

Науковий розвиток теорії нечітких множин також включає вивчення властивостей нечітких чисел та їхньої поведінки у різних умовах. Окрім основоположних праць Лотфі Заде, значний внесок у розвиток теорії нечітких чисел зробили дослідники таких як Дітмар Гельман, Енріке Мадрід, Доріан Ченг, які вивчали можливості розширення класичних математичних операцій на нечіткі множини. Також були розроблені підходи для створення нечітких експертних систем, заснованих на знаннях спеціалістів, де нечіткі числа дозволяють описувати досвід і знання експертів, враховуючи всі тонкощі невизначеності та варіативності.

Отже, історія розвитку нечітких чисел та теорії нечітких множин — це послідовний процес, що йшов від теоретичних основ до практичних додатків, і від вузькоспеціалізованих досліджень до широкого використання у багатьох галузях. Завдяки нечіткій логіці та її здатності описувати неоднозначні та неточні ситуації,

вона стала цінним інструментом для наукових досліджень, інженерії, економіки та багатьох інших сфер.

1.2 Основні поняття нечіткої логіки та нечітких чисел

Нечітка логіка — це розділ математики та штучного інтелекту, що досліджує способи моделювання та управління системами, які працюють з невизначеністю або нечіткістю даних. На відміну від класичної двійкової логіки, де кожне твердження має лише два значення — "істина" або "хиба", нечітка логіка дозволяє оперувати значеннями між цими крайніми межами. Основний принцип нечіткої логіки полягає у використанні ступенів належності, що дозволяє моделювати ситуації, в яких явища не мають точних або чітких меж. Це робить нечітку логіку особливо корисною в системах, де потрібно обробляти складні та неоднозначні дані, такі як температурні режими, рівень комфорту, управління промисловими процесами.

Нечіткі множини — це основне поняття нечіткої логіки, яке було вперше запропоновано Лотфі Заде. На відміну від класичних множин, де кожен елемент або належить множині, або не належить їй, нечітка множина дозволяє кожному елементу належати до множини з певним ступенем належності, що виражається числом від 0 до 1. Наприклад, якщо класична множина "гаряча температура" визначає конкретний діапазон значень, то нечітка множина дозволяє, щоб температури 25°C, 30°C та 35°C належали цій множині з різними ступенями належності.

Функція належності є ключовим елементом у визначенні нечітких множин. Вона вказує на ступінь, з яким певний елемент належить до множини. Функції належності можуть набувати різних форм, залежно від типу нечіткої множини та специфіки задачі. Найпоширенішими є трикутна, трапецієвидна та гаусова функції належності, кожна з яких використовується для моделювання різних типів невизначеності.

Нечіткі числа — це розширення поняття звичайних чисел у випадках, коли значення є нечіткими або варіативними. Нечітке число може бути представлене як

нечітка множина, де кожне значення має певний ступінь належності. Наприклад, нечітке число "приблизно 10" може містити значення 8, 9, 10, 11 та 12, кожне з яких має різний ступінь належності. Це дозволяє моделювати числові значення, які не є точно відомими, але можуть змінюватися в певному діапазоні.

Основні операції з нечіткими числами. Як і звичайні числа, нечіткі числа можуть бути об’єктом математичних операцій, таких як додавання, віднімання, множення та ділення. Однак через нечіткість даних ці операції виконуються з використанням спеціальних правил, що враховують нечіткість значень. Наприклад, при додаванні двох нечітких чисел результатом є нове нечітке число, яке враховує суму ступенів належності обох чисел. Подібні операції допомагають створювати та управляти моделями, що працюють з нечіткими значеннями у реальних задачах.

Нечіткі правила та системи висновку є невід’ємною частиною нечіткої логіки. Нечіткі правила формулюються у вигляді логічних висловлювань типу "якщо-тоді", де кожне правило має нечіткі передумови та нечіткий висновок. Наприклад, правило може бути таким: "Якщо температура висока, то швидкість вентилятора повинна бути великою." Використовуючи нечіткі правила, системи нечіткої логіки можуть генерувати відповіді на запити та приймати рішення навіть за умов невизначеності. Системи нечіткого висновку, такі як метод Мамдані та метод Сугено, широко використовуються для прийняття рішень в умовах нечітких даних. Ці методи застосовуються в експертних системах, контролерах та інших обчислювальних системах, які опрацьовують нечіткі значення, дозволяючи моделювати процеси з високою точністю та врахуванням невизначеності.

Отже, основні поняття нечіткої логіки та нечітких чисел забезпечують ефективні інструменти для аналізу та моделювання невизначених ситуацій у різних сферах — від технічних систем до соціальних процесів. Вони дозволяють обробляти нечітку інформацію, приймати зважені рішення та створювати моделі для реальних задач, що містять неповну або неоднозначну інформацію.

1.3 Огляд методів операцій з нечіткими числами

Операції з нечіткими числами є одним з ключових аспектів застосування нечіткої логіки. Вони дозволяють виконувати базові математичні операції, такі як додавання, віднімання, множення і ділення, з урахуванням невизначеності даних. Оскільки нечіткі числа представляють значення, що можуть змінюватися в певному діапазоні з різними ступенями належності, виконання таких операцій потребує спеціальних методів, які б забезпечили коректність обчислень.

Основні методи операцій з нечіткими числами:

1. Арифметичні операції з використанням інтервальних чисел

Одним з найбільш популярних способів роботи з нечіткими числами є представлення їх у вигляді інтервалів. Наприклад, нечітке число може бути задане інтервалом [a, b], де a та b представляють нижню та верхню межі можливого значення. Виконуючи арифметичні операції з такими інтервалами, результатом буде новий інтервал, який представляє діапазон можливих значень результату. Цей підхід досить простий, але може давати надмірно розмиті результати, особливо у випадках множення та ділення.

2. Альфа-рівневий метод.

Для зменшення нечіткості в результаті використовується альфа-рівневий метод, в якому нечітке число розбивається на кілька інтервалів, кожен з яких має певний рівень належності (α-рівень). Кожний інтервал відповідає певному рівню належності в межах від 0 до 1. Потім операції виконуються для кожного інтервалу окремо, після чого результати об'єднуються, формуючи нове нечітке число. Цей метод є більш точним і дозволяє отримати більш "концентрований" результат, проте він є обчислювально складнішим.

3. Метод розширення Цаде.

Цей метод, запропонований засновником нечіткої логіки Лотфі Заде, дозволяє виконувати арифметичні операції на основі функцій належності. Згідно з методом розширення, для виконання операції між двома нечіткими числами

будується нова функція належності для результату, що враховує всі можливі значення цих чисел. Метод розширення є досить точним, але також вимагає великих обчислювальних ресурсів, що обмежує його застосування в реальних системах з великим обсягом даних.

4. Трикутні і трапецієподібні наближення

Трикутні та трапецієподібні нечіткі числа — це спрощені форми функцій належності, що дозволяють зменшити складність обчислень. Такі числа задаються трьома або чотирма точками, які визначають "основу" та "вершину" функції належності. Використання трикутних та трапецієподібних наближень є популярним в інженерних додатках, оскільки вони дозволяють швидко виконувати арифметичні операції з мінімальною втратою точності.

5. Метод Монте-Карло для нечітких чисел

Цей підхід використовує статистичні методи для оцінки результатів арифметичних операцій з нечіткими числами. Метод Монте-Карло полягає в генерації численних випадкових вибірок з розподілів, що описують нечіткі числа, та обчисленні результатів операцій для кожної вибірки. Цей метод забезпечує високу точність результатів, але є досить обчислювально затратним, оскільки потребує великої кількості вибірок.

6. Наближений метод центрів ваги

Один з найбільш простих методів для обчислення результатів з нечіткими числами полягає у використанні центрів ваги кожного нечіткого числа. Цей метод полягає у заміні кожного нечіткого числа його "центром" — значенням, що має найбільшу вагу або найвищу належність. Потім обчислення виконуються як з чіткими числами. Хоча цей метод є дуже простим та швидким, він може призводити до великих помилок в умовах складних розподілів.

Сфери застосування методів операцій з нечіткими числами є особливою. Кожен з наведених методів має свої переваги та недоліки і підходить для різних типів задач. Наприклад, для завдань реального часу часто використовуються

трикутні або трапецієподібні наближення, оскільки вони є простими у реалізації та дозволяють отримати швидкі результати. Методи розширення або Монте-Карло частіше застосовуються у наукових дослідженнях та моделях, де важлива висока точність, і обчислювальні ресурси не є обмеженням. Отже, вибір методу для роботи з нечіткими числами залежить від вимог конкретної задачі: швидкість, точність, ресурси. Розуміння особливостей кожного методу дозволяє ефективно використовувати їх у різних областях, таких як автоматичне управління, моделювання складних систем та експертні системи.

1.4 Використання нечітких чисел у прикладних задачах

Нечіткі числа знаходять широке застосування в різноманітних прикладних задачах, де необхідно враховувати невизначеність або неповні знання про систему. Технології нечіткої логіки і нечітких чисел дозволяють моделювати реальні процеси з високою точністю, навіть коли дані мають неповну або нечітку природу. Вони використовуються в таких сферах, як автоматичне управління, робототехніка, медичні діагностики, економіка, екологія, а також в інженерії та науці. Розглянемо деякі з основних прикладних напрямків: автоматичне управління, робототехніка та штучний інтелект, медична діагностика, фінансові та економічні моделі, Екологічне моделювання, прогнозування, управління якістю та оцінка ризиків.

У системах автоматичного управління, де необхідно враховувати численні фактори з невизначеністю, нечіткі числа допомагають розв'язувати складні задачі, що виникають під час проектування і контролю. Наприклад, у керуванні температурними режимами в промислових процесах, де точні значення параметрів важко виміряти, нечіткі числа використовуються для створення нечітких контролерів. Це дозволяє знижувати вплив похибок, що виникають через неточності вимірювань, і забезпечує стабільну роботу системи.

У робототехніці та штучному інтелекті нечіткі числа застосовуються для моделювання прийняття рішень в умовах невизначеності. Наприклад, роботи, що

взаємодіють з реальним світом, часто стикаються з нечіткими даними, такими як нечіткі показники відстані, швидкості, температури, руху або навіть шуму навколишнього середовища. За допомогою нечіткої логіки робот може оцінити ситуацію та прийняти рішення, які відповідають умовам середовища, навіть коли точність сенсорних даних залишає бажати кращого. В рамках цієї технології розвиваються такі підсистеми, як навігація, маніпуляція об'єктами, діагностика та адаптація.

Нечіткі числа застосовуються для аналізу медичних даних, де часто немає точних меж для класів захворювань або їх симптомів. Наприклад, нечіткі числа можуть бути використані для оцінки рівня ризику розвитку захворювання на основі множини симптомів, де точні значення можуть варіюватися. Нечітка логіка дозволяє моделювати складні системи прийняття медичних рішень, такі як експертні системи для діагностики, де різні фактори (як-от вік пацієнта, рівень холестерину, наявність супутніх захворювань) можуть мати різні ступені впливу на результат, що дає лікарям інструмент для точної оцінки.

У фінансах і економіці нечіткі числа використовуються для оцінки ризиків, прогнозування економічних тенденцій та прийняття рішень в умовах невизначеності. Наприклад, нечіткі моделі дозволяють враховувати різні економічні фактори (відсоткові ставки, інфляцію, попит на ринку) у фінансовому плануванні або аналізі інвестицій. Моделювання фінансових ринків часто вимагає врахування нечітких критеріїв, оскільки точні прогнози зазвичай неможливі. Такі методи можуть бути застосовані для створення систем управління ризиками, оптимізації портфелів або навіть в процесі кредитування.

Нечіткі числа використовуються в екології для моделювання складних систем, таких як прогнозування забруднення довкілля або аналіз зміни клімату. Екологічні процеси, такі як забруднення повітря чи води, можуть бути дуже складними для точного прогнозування через багато змінних, що впливають на систему (наприклад, метеорологічні умови, рівень промислових викидів, екологічна політика). Використання нечіткої логіки дозволяє ефективно описувати ці процеси та робити прогнози на основі невизначених або часткових

даних, що дозволяє створювати більш точні екологічні моделі.

Нечіткі числа використовуються для прогнозування та планування в багатьох галузях. Наприклад, в управлінні виробництвом нечітка логіка може застосовуватись для оптимізації ресурсів, управління запасами та планування виробничих процесів. Це дає можливість врахувати невизначеність попиту, змінні поставки або коливання ціни на сировину. Аналогічно, в управлінні проектами нечіткі числа дозволяють оцінювати ризики і потенційні результати на основі нечітких або недостовірних даних, що може значно підвищити ефективність процесу прийняття рішень. Нечітка логіка активно використовується для управління якістю в різних індустріях, де оцінка якості продукції чи послуг потребує врахування різних нечітких критеріїв. Наприклад, при оцінці якості виробів у машинобудуванні чи харчовій промисловості часто використовуються нечіткі методи для врахування таких факторів, як зовнішній вигляд, смакові якості, органолептичні характеристики. Це дозволяє створювати системи управління якістю, які точніше відображають реальний стан продукції.

Отже, використання нечітких чисел у прикладних задачах відкриває широкі можливості для моделювання і прийняття рішень в умовах невизначеності та неповноти інформації. Ці методи забезпечують більшу гнучкість і точність у складних ситуаціях, що робить їх надзвичайно корисними у багатьох наукових і практичних сферах.

РОЗДІЛ 2.
ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Основні принципи операції над нечіткими числами

Операції над нечіткими числами відрізняються від традиційних арифметичних операцій через необхідність врахування невизначеності, яка притаманна нечітким множинам. Основою для здійснення цих операцій є використання функцій належності, які описують ступінь приналежності певного значення до нечіткої множини. Завдяки цьому можна працювати з даними, які мають різні ступені точності чи достовірності, і відображати нечіткі значення, що не можуть бути строго визначені.

Таблиця 1 – Основні принципи виконання операцій над нечіткими числами

№		Опис
1	Функція належності як основа операцій	описуються функціями належності, які визначаються як відображення ступеня приналежності числа до певної множини на шкалі від 0 до 1. Наприклад, число може належати до множини "високий рівень доходу" з належністю 0.7, що відображає високу, але

		не абсолютну впевненість у тому, що воно відповідає цьому критерію. Виконуючи операції з такими числами, важливо враховувати ці функції, оскільки вони впливають на результативні ступені належності.
2	Застосування операцій Ларсена і Цаде:	застосовуються для обчислення об'єднання, перетину та доповнення нечітких множин. Для об'єднання використовується максимум двох функцій належності, тоді як для перетину застосовується мінімум. Ці оператори дозволяють отримати нечітке число як результат операції, враховуючи всі можливі рівні належності значень, які взаємодіють.
3	Арифметичні операції з нечіткими числами:	при виконанні арифметичних операцій над такими числами, необхідно враховувати всі можливі значення в межах цього інтервалу. Зазвичай, для цього використовуються такі методи, як альфа-рівні, які дозволяють розбити нечітке число на підмножини з фіксованою впевненістю, та методи згортання, що об'єднують результати операцій з нечіткими числами у єдине значення
4	Альфа-рівні та їх застосування	підмножини нечіткої множини, для яких ступінь належності до нечіткої множини перевищує певний поріг. В межах кожного альфа-рівня нечітке число розглядається як чіткий інтервал, і всі операції виконуються для кожного з цих інтервалів. Після завершення операцій результати інтервалів об'єднуються для отримання остаточного нечіткого результату. Цей підхід дозволяє зберігати нечіткість і

		невизначеність при виконанні операцій, зберігаючи точність результатів.
5	Інтервальне представлення результатів	виконуються операції додавання чи множення над нечіткими числами, результат виражається у вигляді інтервалу можливих значень, що відображає усі можливі результати з урахуванням невизначеності вхідних даних. Цей інтервальний підхід дозволяє отримати результати, які зберігають нечіткість, що є важливою вимогою для роботи з нечіткими даними у складних обчислювальних моделях, де необхідна гнучкість та адаптивність.
6	Метод розмитої логіки Мамдан	дозволяє перетворити чіткі входи в нечіткі множини, застосувати правила для формування нечітких результатів і повернути чітке значення через фазифікацію. Це є ключовим для багатьох систем керування, які потребують аналізу нечітких даних для прийняття рішення.

Описані принципи в табл. 1 формують основу для роботи з нечіткими числами та їх використання у різноманітних прикладних задачах, де потрібна обробка неточних, нечітких чи неповних даних. Таблиця 2 з прикладами використання операцій над нечіткими числами в різних прикладних задачах. У ній наведено приклади задач, вид операції та конкретний опис того, як ці операції застосовуються у реальних ситуаціях.

Таблиця 2 – Приклади використання операцій над нечіткими числами

Приклад задачі	Операція з нечіткими	Опис застосування

	числами	
Керування температурою в приміщенні	Додавання	У системах "розумного будинку" використовують нечітке додавання для визначення оптимальної температури на основі зовнішньої температури та поточної температури в приміщенні, з урахуванням нечітких факторів, як-от "тепло" чи "прохолодно"
Медичне діагностування	Множення	Для оцінки ймовірності захворювання можуть застосовувати множення нечітких чисел, яке відображає вплив кількох симптомів одночасно. Наприклад, на основі симптому "висока температура" (0.7) і "кашель" (0.6) оцінюється загальна ймовірність інфекції
Фінансовий аналіз ризиків	Віднімання	У фінансовому аналізі ризиків віднімання нечітких чисел застосовується для визначення "непередбачуваного ризику", зважаючи на нечіткі фактори, як-от нестабільність ринку та інфляцію. Це дає можливість точніше оцінити відхилення від прогнозованих показників
Оцінка якості продукції	Порівняння	Порівняння нечітких чисел використовується для визначення відповідності продукції стандартам якості, якщо критерії, наприклад, "висока якість", нечітко визначені. Це дозволяє оцінити, наскільки продукція відповідає необхідним нормам
Транспортна логістика	Максимізація	У транспортних задачах максимізація нечітких значень допомагає визначити оптимальні маршрути доставки, враховуючи нечіткі фактори, як-от "коротший шлях", "менші затримки" Максимізація забезпечує оптимальний

		вибір маршруту з урахуванням всіх умов
Система підтримки прийняття рішень	Мінімізація	У системах прийняття рішень мінімізація дозволяє обирати альтернативу з найменшою невизначеністю, зокрема в питаннях керування проєктами, де нечіткі фактори, як-от "низький ризик", визначають вибір проекту для інвестування або розвитку
Енергетичний моніторинг	Альфа-рівні	У контролі за енергоспоживанням використовують альфа-рівні для розподілу енергетичних ресурсів на різних рівнях попиту. Наприклад, обсяг енергоспоживання визначають за рівнями "високий", "середній" і "низький", щоб ефективно керувати енергетичними ресурсами

Табл. 2 дозволяє наочно представити, як нечіткі операції знаходять застосування у практичних задачах і забезпечують адаптивність, точність та ефективність при роботі з неповними або неточними даними.

2.2 Арифметичні операції з нечіткими числами

Арифметичні операції з нечіткими числами — це один з основних інструментів у нечіткій математиці, який дозволяє обробляти дані, що мають неточні, розмиті значення. На відміну від традиційних чисел, нечіткі числа виражаються у вигляді функцій приналежності та враховують ступінь невизначеності даних, що є особливо корисним при моделюванні складних систем із частковими або нечіткими вхідними даними. Основні арифметичні операції з нечіткими числами включають додавання, віднімання, множення та ділення. Кожна з цих операцій має свої особливості й обмеження, які враховуються при розв’язанні прикладних задач. Арифметичні операції з нечіткими числами

дозволяють враховувати всі можливі варіації даних у межах заданих розмитих множин, що є важливим для точності та адаптивності моделей. Вони також застосовуються у багатьох галузях, включаючи економіку, інженерію, медицину та екологію, для побудови гнучких моделей, що здатні працювати з неточними або неповними даними.

Ось приклади арифметичних операцій з нечіткими числами, де розглянемо числа як інтервали для кожної операції:

Нехай маємо два нечіткі числа A=[1,3]A=[1,3] та B=[2,4]B=[2,4]. Для додавання нечітких чисел розраховуємо мінімальну та максимальну границі результату:

1. Операція додавання нечітких чисел використовується для визначення загального ефекту двох або більше нечітких значень. Для реалізації цієї операції до кожного значення нечіткого числа застосовуються функції приналежності, що дозволяє отримати нову нечітку множину, яка відображає сумарний результат. Наприклад, якщо є два нечіткі числа, що описують "низький рівень вологості" і "середній рівень температури", то їхнє додавання дозволить знайти сумарний вплив цих параметрів на кліматичний стан приміщення.

Нехай маємо два нечіткі числа A= [1,3] A= [1,3] та B = [2,4]. Для додавання нечітких чисел розраховуємо мінімальну та максимальну границі результату:

$$A + B = [1 + 2,3 + 4] = [3,7]$$

Отже, нечітке число, отримане шляхом додавання A і B, буде [3,7].

2. Віднімання нечітких чисел використовується для визначення різниці між двома нечіткими значеннями. Це особливо корисно в задачах порівняння, коли необхідно визначити, наскільки один показник відрізняється від іншого. У таких випадках виконуються операції над відповідними функціями приналежності для обчислення різниці. Застосування віднімання можна зустріти в фінансовому

аналізі ризиків, коли визначається різниця між запланованим і фактичним значенням доходу або витрат з урахуванням нечітких змін.

Припустимо, що маємо нечіткі числа $A = [5,7]$ та $B = [1,3]$. Віднімання дає такий результат:

$$A - B = [5 - 3,7 - 1] = [2,6]$$

Таким чином, результат віднімання A і B є нечітке число [2,6]

3. Множення нечітких чисел дозволяє визначити комбінований вплив двох розмитих показників, особливо коли йдеться про фактори, що залежать один від одного. У реальних умовах ця операція застосовується, наприклад, для оцінки ймовірності настання події за наявності кількох ризиків одночасно, що можуть бути нечітко визначеними. У такому випадку значення функції приналежності обчислюються для кожного нечіткого числа, після чого множаться для отримання результату.

Нехай маємо нечіткі числа A=[2,4] та B=[1,3]. Для множення розглянемо всі комбінації меж інтервалів, щоб отримати нечітке число:

$$A \cdot B = [min(2 \cdot 1,2 \cdot 3,4 \cdot 1,4 \cdot 3), max(2 \cdot 1,2 \cdot 3,4 \cdot 1,4 \cdot 3)] \ = \ [2,12]$$

Отже, результат множення A і B є нечітке число [2,12].

4. Ділення нечітких чисел застосовується для задач, де необхідно поділити один нечіткий показник на інший. Це може бути корисно у випадках, коли потрібно визначити середнє значення або співвідношення між двома нечіткими значеннями. Наприклад, у задачах економічного прогнозування ділення може використовуватися для оцінки ефективності витрат у розмитих умовах. Через специфіку нечітких чисел, операція ділення є більш складною, ніж інші арифметичні операції, і вимагає особливої уваги для забезпечення стабільності результату.

Візьмемо нечіткі числа A=[4,8] та B=[2,3]. Для ділення також враховуємо всі можливі комбінації меж:

$\frac{A}{B}$ = [min($\frac{4}{3}, \frac{4}{2}, \frac{8}{3}, \frac{8}{2}$),max($\frac{4}{3}, \frac{4}{2}, \frac{8}{3}, \frac{8}{2}$)] =[1.33,4]

Таким чином, результат ділення A і B буде [1.33,4].

5. Часто нечіткі числа подаються у вигляді інтервалів, де є мінімальне і максимальне значення, що представляє розмах можливих значень. Арифметичні операції з інтервальними нечіткими числами дозволяють виконувати розрахунки з урахуванням повного діапазону можливих значень, що особливо актуально для прогнозів із великою часткою невизначеності. Наприклад, при прогнозуванні кліматичних умов на певний період інтервальні нечіткі числа можуть допомогти врахувати всі можливі зміни температури або вологості в межах заданого інтервалу.

Припустимо, що маємо інтервальні нечіткі числа A=[3,6]] і B=[1,2]. Якщо розглядаємо операцію додавання та віднімання, можемо одержати:

Додавання: $A + B = [3 + 1{,}6 + 2] = [4{,}8]$

Віднімання: $A - B = [3 - 2{,}6 - 1] = [1{,}5]$

Ці приклади демонструють, як нечіткі числа обчислюються, зберігаючи варіативність значень у межах інтервалів, що дозволяє працювати з розмитістю даних

2.3 Операції порівняння нечітких чисел

Порівняння нечітких чисел є важливим аспектом у застосуванні нечіткої логіки, оскільки дає змогу оцінити відносні значення розмитих даних, які не завжди мають чіткі межі, як у традиційній логіці. В реальних задачах, таких як управління, прогнозування та ухвалення рішень, порівняння нечітких чисел дозволяє обробляти дані з невизначеністю та варіативністю. Оскільки нечіткі числа представлені інтервалами або функціями належності, класичні методи порівняння чисел стають неефективними, і для цього використовують спеціальні методи. Основні підходи до порівняння нечітких чисел: порівняння на основі α-рівнів, метод центру тяжіння, метод максимальних і мінімальних значень, метод ступеня домінування, метод перетинів і спільної площі.

Метод порівняння на основі α-рівнівзасноваий на представленні нечіткого числа як набору інтервалів для різних рівнів α ($0 \leq \alpha \leq 1$). Кожен α-рівень (α-cut) визначає інтервал, де функція належності нечіткого числа набуває значень не менше α. Для порівняння двох нечітких чисел порівнюють інтервали, які відповідають одному й тому ж рівню α. Якщо одне число має більший інтервал при кожному рівні α, то його вважають більшим. Цей метод широко використовується, оскільки він забезпечує надійність при порівнянні. Наприклад, нехай A=[2,5] та B=[3,6]. Для α = 0.5 інтервал для A буде [2.5, 4.5], а для B — [3.5, 5.5]. Якщо для всіх рівнів α інтервал для B більше, ніж для A, то B вважається більшим за A.

Метод центру тяжінняпередбачає обчислення центру тяжіння (центроїда) функції належності нечіткого числа. Центр тяжіння визначає центральне значення нечіткого числа і дозволяє порівнювати нечіткі числа як більш-менш "центральні". Число з більшим значенням центру тяжіння вважається більшим. Наприклад, нехай для нечітких чисел A і B функції належності визначені так, що центри тяжіння для них становлять 3.5 та 4.0 відповідно. У такому разі B вважається більшим за A.

У цьому методі порівняння нечітких чисел базується на максимальних і мінімальних значеннях їх інтервалів. Якщо максимальне значення одного числа більше за максимальне значення іншого, то перше число вважається більшим. У разі, якщо максимальні значення однакові, порівнюють мінімальні значення. Наприклад, нехай A=[2,5] та B=[3,6]. Оскільки максимальне значення B (6) більше за максимальне значення A (5), то B більше за A.

Метод ступеня домінування розглядає відносний ступінь домінування одного нечіткого числа над іншим. Ступінь домінування визначається як інтеграл, який показує, наскільки одне нечітке число переважає інше у вигляді ймовірнісної оцінки. Зазвичай використовується в задачах, де потрібно визначити ступінь переваги одного варіанту над іншим. Наприклад, нехай нечітке число A має ступінь домінування над B, що дорівнює 0.7 (тобто ймовірність, що A більше за B, становить 70%). У цьому випадку A переважає над B з визначеним ступенем

домінування.

У цьому методі перетинів і спільної площі розглядається площа спільної частини функцій належності двох нечітких чисел. Число з більшою площею, яка не перекривається з іншим числом, вважається більшим. Цей підхід дозволяє отримати уявлення про міру переваги одного нечіткого числа над іншим. Наприклад, нехай функції належності двох нечітких чисел A та B мають площу, що не перекривається. Якщо площа функції належності для A більша за площу для B, то A вважається більшим.

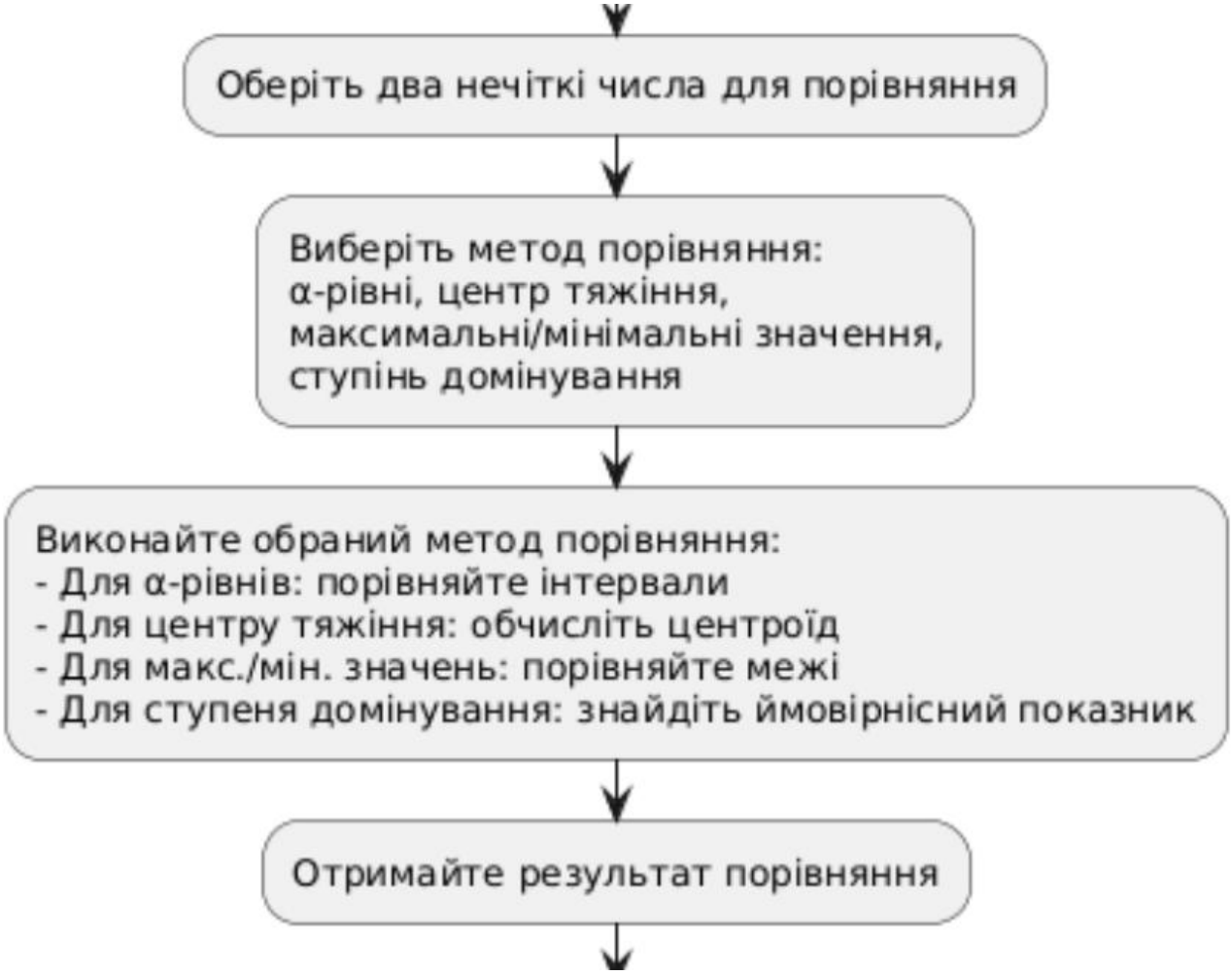

Рисунок 1 – Основні кроки для порівняння нечітких чисел різними методами

На схемі (рис. 1) показано основні етапи: вибір чисел для порівняння, вибір методу, виконання обраного методу для порівняння нечітких чисел, а також

отримання результату. Дана блок-схема допоможе систематизувати процес порівняння нечітких чисел для практичних застосувань.

2.4 Методи обробки нечітких даних: розмиті множини та функції належності

Обробка нечітких даних є важливим аспектом нечіткої логіки, що дозволяє працювати з інформацією, яка не може бути точно представлена в термінах традиційної бінарної логіки (чіткої). Нечіткі числа й множини використовують теорії, які дозволяють описувати й обробляти невизначеність, неточність та неповноту даних, що мають місце в реальному світі. У цьому контексті важливими інструментами є розмиті множини та функції, які є основою для більшості методів обробки нечітких даних. Розмита множина — це множина елементів, де кожен елемент має не просто чітке належність (як у класичних множинах), а певну ступінь належності, який вимірюється в інтервалі від 0 до 1. Традиційні множини з чіткою належністю визначають елементи, які або належать, або не належать до множини. Водночас у розмитих множинах кожен елемент може мати ступінь належності, що вказує на "нечіткість" його статусу. Наприклад, множина "високі люди" може бути описана не лише через чітке значення зросту (наприклад, більше ніж 180 см), а через функцію належності, яка вказує на ймовірність того, що людина належить до цієї категорії при різних значеннях її зросту. Розмита множина A для змінної x може бути описана за допомогою функції належності $\mu_A(x)$, де $\mu_A(x) = 0$, якщо x точно не належить до множини; $\mu_A(x) = 1$, якщо x повністю належить до множини; $0 < \mu_A(x) < 1$, якщо x має часткову належність до множини.

Розмита функція — це функція, що має нечіткий або розмитий характер і описується через нечітке значення замість точного числового результату. Замість того щоб використовувати конкретні значення, розмиті функції оперують з множинами, де результат залежить від ступеня належності вхідних даних до розмитих множин. Такий підхід застосовується в багатьох сферах, зокрема в обробці сигналів, системах автоматичного управління та нечітких системах. Наприклад, функція належності для температури в діапазоні від 20°C до 30°C

може мати значення близько 1 при 25°C (повністю належить до категорії "тепло") і значення близько 0 при 10°C (не належить до категорії "тепло").

Розмиті функції мають широке застосування у нечітких логічних системах для прогнозування та моделювання процесів, де важливо не лише мати чітке значення, але й враховувати ступінь невизначеності або нечіткості даних.

Існують кілька основних методів обробки нечітких даних, зокрема:

1. Розмиті операції над множинами (операції, такі як об'єднання, перетин і доповнення, що застосовуються до нечітких множин, замість стандартних операцій, де визначається чітке належність елементів, тут визначається ступінь належності для кожного елемента на основі функцій належності).

2. Розмиті логічні операції (базуються на операціях нечіткої логіки, де замість бінарних значень (істина/хибність) використовуються нечіткі значення в інтервалі [0, 1], що є основою нечітких логічних систем для прийняття рішень).

3. Моделі нечітких множин (використовуються для побудови математичних моделей реальних об'єктів або процесів, де традиційні методи моделювання є недостатніми через наявність високої невизначеності).

4. Алгоритми нечітких систем (дозволяють проводити обчислення на основі нечітких чисел, таких як методи нечіткої регресії або нечітких класифікацій, використовуються в аналізі даних, де дані можуть бути нечіткими або неповними).

Розмиті множини і функції знайшли широке застосування в багатьох галузях (рис. 2)

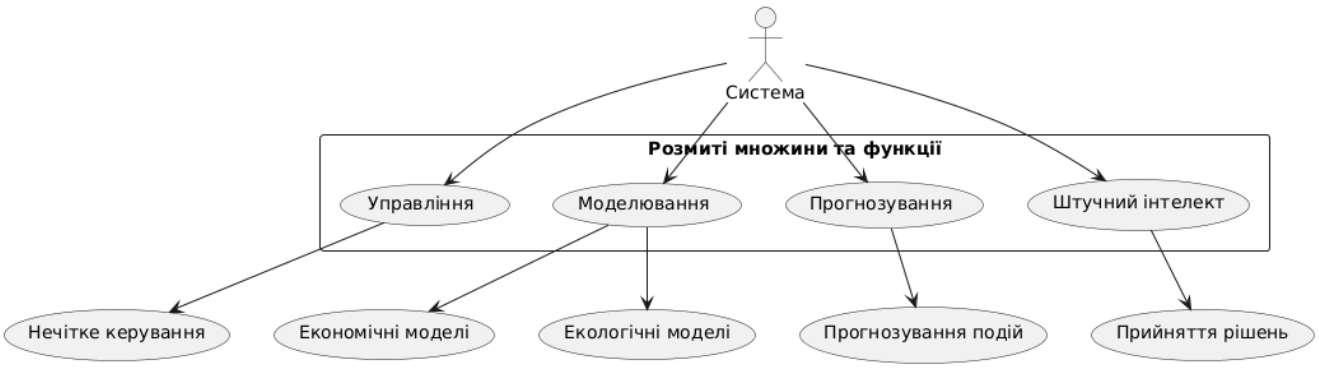

Рисунок 2 – Використання розмитих множин і функцій в різних галузях

2.5 Алгоритми для виконання операцій над нечіткими числами

Алгоритми для виконання операцій над нечіткими числами є важливою складовою нечіткої логіки та нечітких систем, оскільки вони дозволяють ефективно обробляти інформацію, що містить невизначеність або нечіткість. Оскільки нечіткі числа мають ступінь належності до множини, ці алгоритми працюють із такими даними, де результат операцій залежить від певного ступеня належності елементів до множини.

Нечіткі числа вимагають застосування спеціальних методів, оскільки звичайні арифметичні операції (як-от додавання, віднімання) не застосовуються до нечітких значень без адаптації. Для цього було розроблено низку алгоритмів, які забезпечують обробку нечітких чисел у рамках нечіткої логіки та системи. Нижче представлені основні типи алгоритмів для виконання операцій над нечіткими числами.

Для додавання нечітких чисел застосовується операція, де результат є нечітким числом з функцією належності, яка розраховується як комбінація функцій належності двох операндів. Ось базова ідея алгоритму: якщо $A(x)$ та $B(x)$ — це функції належності нечітких чисел A та B відповідно, то функція належності їхньої суми $C(x) = A(x) + B(x)$ обчислюється через операцію складання для кожної пари елементів. Потрібно враховувати, що результат не буде мати чітко визначеного значення, а лише ступінь належності до розмитої множини.

Множення нечітких чисел зазвичай визначається через операцію на функціях належності: якщо $A(x)$ та $B(x)$ — це функції належності нечітких чисел A і B, то функція належності для результату множення C(x) визначається як:

$$C(x) = A(x) \cdot B(x)$$

Вищевкзане означає, що для кожної пари значень ступінь належності до результату визначається добутком ступеня належності окремих елементів. Алгоритм для поділу нечітких чисел аналогічний до множення, однак додатково враховуються складнощі з діленням, коли один з елементів може мати значення, близьке до нуля. Алгоритм полягає в обчисленні відношення двох нечітких чисел шляхом поділу їх функцій належності: $C(x) = \frac{A(x)}{B(x)}$, де $B(x)$ не може бути нульовим, інакше треба застосувати додаткові заходи для обробки таких випадків.

Для порівняння нечітких чисел найчастіше використовуються алгоритми, які оцінюють ступінь належності елементів до заданих множин. Наприклад, якщо A(x) та B(x) — це функції належності нечітких чисел, то можна порівнювати їх шляхом оцінки значень функцій належності в певних точках або через агреговані значення функцій належності. Просте порівняння можна здійснити через умови типу: A(x)>B(x), якщо для всіх x, A(x)>B(x) Це дозволяє визначити, яке з нечітких чисел є більшим або меншим на основі їхніх функцій належності. Для нечітких чисел також існують алгоритми для порівняння на основі мінімуму та максимуму їхніх функцій належності:

Максимум: для двох нечітких чисел A(x) і B(x) можна визначити максимум: x

$$\mu_{max}(x) = \max(\mu_A(x), \mu_B(x))$$

Мінімум: для мінімуму використовується:

$$\mu_{min}(x) = \min(\mu_A(x), \mu_B(x))$$

Такі операції застосовуються для порівняння результатів, де для кожної точки xx обирається максимум або мінімум з відповідних функцій належності. Агрегація нечітких чисел полягає в поєднанні кількох нечітких чисел в одне за певними правилами. Це використовується в задачах, де необхідно об'єднати результати кількох

нечітких вимірювань чи оцінок. Для обчислення середнього значення кількох нечітких чисел часто застосовується середнє значення їхніх функцій належності:

C(x) = $\frac{1}{n}\sum_{n=1}^{n} A_i(x)$, де $A_i(x)$ - це функції належності кожного нечіткого числа, а n - кількість чисел.

В деяких випадках для агрегації нечітких чисел використовуються алгоритми, які поєднують мінімум або максимум значень для кожної точки x:

Для мінімальної агрегації:

$$C(x) = min(A_1(x), A_2(x), \dots, A_n(x))$$

Для максимальної агрегації:

$$C(x) = max(A_1(x), A_2(x), \dots, A_n(x))$$

У системах нечіткого висновку (Fuzzy Inference Systems) використовуються спеціалізовані алгоритми, такі як методи Мамдані та Сугено, для оцінки результатів на основі нечітких правил. Основні етапи таких алгоритмів включають застосування нечітких правил для отримання нечітких висновків та перетворення нечітких висновків в чітке значення, що може бути використано в реальному додатку.

Алгоритми для виконання операцій над нечіткими числами є основою для обробки інформації в нечітких системах. Вони дозволяють ефективно працювати з даними, які мають невизначену або нечітку природу, застосовуючи спеціалізовані методи для арифметичних операцій, порівняння, агрегації та висновків. Ці алгоритми широко використовуються в різних галузях, таких як системи управління, моделювання, прогнозування та штучний інтелект.

РОЗДІЛ 3.
ТЕОРЕТИЧНА ЧАСТИНА

3.1 Опис реалізації блок операцій з нечіткими числами

Задача полягає у розробці програмного забезпечення для виконання операцій над нечіткими числами, що представляють собою пари значень: основне значення (значення вимірювання) та невизначеність (похибка або діапазон можливих коливань цього значення). Операції, які необхідно реалізувати, включають:

1. Додавання нечітких чисел – виконання арифметичної операції додавання двох нечітких чисел. Для цього потрібно додати основні значення та обчислити нову невизначеність як суму невизначеностей обох чисел.

2. Віднімання нечітких чисел – виконання операції віднімання для двох нечітких чисел. Під час віднімання основних значень результату, невизначеність обчислюється як сума невизначеностей обох чисел.

3. Множення нечітких чисел – для множення нечітких чисел потрібно обчислити нове основне значення як добуток основних значень двох чисел, а невизначеність результату визначається через вираз, що залежить від основних значень та невизначеностей чисел.

4. Порівняння нечітких чисел – здійснюється порівняння двох нечітких чисел, з врахуванням невизначеності. Результатом може бути виведення одного з трьох варіантів: перше число більше, друге більше або числа рівні (при перекритті діапазонів невизначеності).

Для кожної з цих операцій необхідно правильно обчислити нові значення результату з урахуванням як основних значень, так і невизначеностей. Оскільки нечіткі числа враховують варіації та похибки, важливо забезпечити математичну коректність кожної операції і відображати результат у вигляді, що чітко показує не тільки результат арифметичної операції, але і його точність (невизначеність).

Всі операції повинні бути реалізовані в програмі, де користувач може вводити значення нечітких чисел через графічний інтерфейс, вибирати потрібні операції, а також переглядати результати обчислень разом з їх невизначеністю.

3.2 Алгоритм роботи програми

Алгоритм для виконання операцій з нечіткими числами включає кілька основних етапів: введення значень, вибір операції, виконання математичних операцій та виведення результату.

Першим кроком є введення двох нечітких чисел. Користувач вводить основні значення та відповідні їм невизначеності. Для кожного нечіткого числа користувач вводить два параметри: основне значення та невизначеність. Після введення програма перевіряє, чи є введені дані дійсними числами. Якщо введено

некоректні значення, користувач отримує повідомлення про помилку та запит на введення правильних значень.

Далі користувач обирає одну з операцій, які можуть бути виконані над нечіткими числами. Операції включають додавання, віднімання, множення та порівняння. Користувач вказує, яку операцію потрібно виконати, і програма переходить до відповідного етапу.

В залежності від вибраної операції програма виконує обчислення:

1. Додавання: нове основне значення обчислюється як сума основних значень двох нечітких чисел, а нова невизначеність — як сума невизначеностей обох чисел.
2. Віднімання: нове основне значення обчислюється як різниця основних значень, а невизначеність обчислюється аналогічно до додавання.
3. Множення: нове основне значення обчислюється як добуток основних значень двох чисел. Невизначеність обчислюється як сума абсолютних значень, які враховують вплив невизначеностей кожного числа на результат множення.
4. Порівняння: програма порівнює два нечітких числа, враховуючи їх основні значення та невизначеності, щоб визначити, яке число більше, менше або рівне.

Після виконання обчислень програма виводить результат. Якщо виконано арифметичну операцію (додавання, віднімання, множення), то результат виводиться у вигляді нового нечіткого числа з основним значенням і невизначеністю. Якщо виконано операцію порівняння, то результат виводиться у вигляді порівняння двох чисел (наприклад, "FuzzyNumber1 > FuzzyNumber2").

Після виведення результату користувач може вибрати нову операцію або ввести нові значення для обчислень. Якщо програма досягла кінця і користувач більше не хоче працювати, вона завершується. Алгоритм допомагає здійснювати базові операції з нечіткими числами, обчислюючи нові нечіткі числа або порівнюючи їх з іншими, при цьому враховуючи невизначеність, яка є складовою частиною нечітких чисел.

Алгоритм:

1. Введення значень для FuzzyNumber1 і FuzzyNumber2.

2. Перевірка коректності введених значень.

3. Користувач вибирає операцію.

4. В залежності від обраної операції:

   4.1. Для додавання:

      - Обчислити new_value = value1 + value2.

      - Обчислити new_uncertainty = uncertainty1 + uncertainty2.

   4.2. Для віднімання:

      - Обчислити new_value = value1 - value2.

      - Обчислити new_uncertainty = uncertainty1 + uncertainty2.

   4.3. Для множення:

      - Обчислити new_value = value1 * value2.

      - Обчислити new_uncertainty = |value1 * uncertainty2| + |value2 * uncertainty1|.

   4.4. Для порівняння:

      - Порівняти значення з урахуванням невизначеностей.

5. Вивести результат.

6. Повторити або завершити.

3. 3 Обгрунтування вибору програмних засобів для створення програмного продукту

Для створення навчального програмного забезпечення з теми "Операції над нечіткими числами" необхідно було обрати відповідні програмні засоби, які б забезпечували ефективну реалізацію теоретичних і практичних аспектів нечіткої логіки. Враховуючи вимоги до функціональності, простоти використання та інтеграції різних компонентів, були обрані такі основні інструменти та технології:

HTML

Простота вивчення
Базова синтаксична структура зрозуміла навіть новачкам (теги, атрибути).

Універсальність
Працює на всіх пристроях і браузерах (Chrome, Firefox, Safari тощо).

Семантичність (HTML5)
Теги `<header>`, `<section>`, `<article>` покращують SEO та доступність для скрінрідерів.

Інтеграція з іншими технологіями
Легко поєднується з CSS (стилі) та JavaScript (інтерактивність).

Відкритий стандарт
Не потребує ліцензій або плати — безкоштовний для використання.

Швидкість розробки
Можна створити просту сторінку за лічені хвилини.

Переваги Flask

Гнучкість — можна використовувати лише потрібні модулі.

Легка інтеграція з іншими бібліотеками (наприклад, Pandas для аналітики).

Ідеально для MVP — швидкий старт для прототипів.

Документація — детальна та зрозуміла.

Переваги CSS

Розділення структури (HTML) та стилів (CSS) — легше супроводжувати.

Реюзабельність — один стиль можна застосовувати до багатьох елементів.

Сучасні макети — Flexbox/Grid спрощують створення складних layout-ів.

Адаптивний дизайн — можливість оптимізувати сторінку для мобільних пристроїв.

Підтримка анімацій — без необхідності використовувати JavaScript.

.

Причини вибору інструментів

Вибір зазначених програмних засобів базується на їхній здатності задовольнити основні вимоги до функціональності та якості програмного продукту. Java була обрана завдяки своїй популярності, стабільності та можливості крос-платформенного виконання, що дозволяє запускати програму на будь-якій операційній системі. Swing забезпечує створення якісного графічного інтерфейсу, що є важливим для взаємодії з користувачем. Maven спрощує керування залежностями, а JUnit дозволяє проводити тестування для забезпечення якості програмного забезпечення. Використання Git дозволяє ефективно керувати кодом та координувати роботу в команді.

Цей комплекс інструментів дозволяє створити надійне та ефективне програмне забезпечення, яке задовольнить потреби користувачів у навчанні теорії та практиці роботи з нечіткими числами

3.5. Проєктування структури програмного забезпечення

На етапі проєктування було визначено загальну архітектуру навчального програмного забезпечення "Операції з нечіткими числами", яка складається з клієнтської частини (frontend), серверної частини (backend), а також десктопної

версії на основі PyInstaller. Такий підхід забезпечує кросплатформенність і доступність як через браузер, так і у вигляді локальної програми.

Логічна структура сайту

Для кращого розуміння взаємозв’язку сторінок було побудовано схему логічної структури веб-додатку (див. рис. 3.1). Вона демонструє основні модулі системи та навігаційні переходи між ними.

Рисунок 4.1 – Блок-схема логічної структури сайту

Архітектурна модель

Програмне забезпечення побудовано за моделлю "клієнт–сервер", де:

- Клієнт відповідає за відображення інтерфейсу, взаємодію з користувачем та візуалізацію результатів.
- Сервер обробляє логіку обчислень, операції з нечіткими числами, генерує графіки та обслуговує API-запити.

Основні компоненти:

- Flask використовується як основний веб-фреймворк для створення маршрутизації, обробки запитів, рендерингу HTML-шаблонів через Jinja2.
- NumPy забезпечує математичні обчислення з масивами, які описують функції належності.
- Plotly відповідає за побудову інтерактивних графіків як на стороні сервера (через Python), так і на стороні клієнта (через plotly.js).
- HTML/CSS/JavaScript реалізують клієнтський інтерфейс користувача.
- PyInstaller формує десктопну версію програми у вигляді виконуваного файлу .exe.

Структура каталогів та файлів

Проєкт

├── app.py # Основний Flask-додаток

├── fuzzy.py # Логіка обчислень з нечіткими числами

├── fuzzy_app.py # Десктопна версія

├── static/ # Статичні файли (CSS, зображення)

├── templates/ # HTML-шаблони для сторінок

├── Програма exe/ # Скомпільована десктопна програма

├── build/, dist/, __pycache__/ # Технічні каталоги збірки

└── README.md # Документація

Розподіл функцій між модулями:

- fuzzy.py — реалізує класи, функції та операції з нечіткими числами.
- app.py — містить усі маршрути Flask, які обробляють запити від клієнта.
- templates/ — містить сторінки інтерфейсу: теорія, калькулятор, тести, тренажер, тощо.
- static/ — зберігає CSS-стилі, скрипти та медіафайли.
- fuzzy_app.py — дозволяє запускати програму локально без сервера.

Таким чином, структура проєкту побудована з урахуванням зручності розгортання, підтримки та масштабування.

РОЗДІЛ 4 .ПРАКТИЧНА ЧАСТИНА

4.1. Реалізація модуля теоретичного навчання

Модуль теоретичного навчання є базовим компонентом програмного забезпечення та виконує роль джерела знань для користувачів, які тільки починають знайомство з темою нечітких чисел. Його реалізація охоплює зручне представлення текстової інформації, структуровану навігацію, а також забезпечує стилістичну узгодженість з усією платформою.

Весь теоретичний матеріал згруповано на одній сторінці (theory.html), яка автоматично підключається через Flask-маршрут:

```
@app.route('/theory')
def theory():
    return render_template('theory.html')
```

Шаблон theory.html наслідує базову структуру з base.html, що дозволяє зберігати однакову навігаційну панель, футер та загальний стиль.

Структура контенту

Усі теоретичні розділи розміщуються в HTML у вигляді <section> та <article> з відповідною семантикою. Кожен блок включає заголовок, опис, приклади та (у разі потреби) формули або візуалізації.

Наприклад, базовий розділ теорії реалізується так:

<section class="theory-section">

<h1>Що таке нечітке число?</h1>

Нечітке число — це множина з елементами, які мають ступінь належності в інтервалі від 0 до 1.

Така форма представлення дозволяє моделювати нечіткі або приблизні значення.

</section>

Для створення блоків з ілюстраціями або математичними виразами можуть використовуватись img, code, pre або MathJax для LaTeX-форматів.

Візуальне оформлення

Сторінка теорії має власний файл стилів theory.css, який задає кольорову палітру, відступи, оформлення блоків, а також анімації для покращення сприйняття.

```
.theory-section {
    background-color: var(--bg-secondary);
    padding: 40px;
    border-radius: 12px;
    box-shadow: var(--shadow-sm);
}
```

```
.theory-section h1, .theory-section h2 {
    color: var(--primary-color);
    margin-bottom: 15px;
}

.theory-section p {
    font-size: 1.1rem;
    line-height: 1.7;
}
```

Анімації використовуються для плавної появи текстових блоків при завантаженні:

```
@keyframes fadeIn {
    from { opacity: 0; transform: translateY(10px); }
    to { opacity: 1; transform: translateY(0); }
}

.theory-section {
    animation: fadeIn 0.6s ease-out;
}
```

Зміст сторінки

Навчальні матеріали створено вручну у вигляді статичних блоків. До них входять такі ключові теми:

- Основи нечіткої логіки: відмінність між класичною та нечіткою множиною;
- Функції належності: типи графіків — трикутна, трапецієподібна, гаусова;
- Операції з нечіткими числами: поняття α-рівнів, приклади обчислень;
- Типові задачі: приклади використання в реальному світі.

Теорія подана мовою, адаптованою під студентів, із великою кількістю візуальних прикладів, що дозволяє краще засвоїти складні математичні поняття.

4.2. Розробка модуля навчальної програми

Модуль навчальної програми слугує основною частиною системи, яка забезпечує поступове проходження теоретичних блоків, прикладів і завдань. Він реалізований як окрема сторінка вебзастосунку, на якій користувач може ознайомитись із матеріалом поетапно: від базових понять до розв’язання практичних задач.

Маршрут для цієї сторінки описано у Flask-програмі:

```
@app.route('/trainer', methods=['GET', 'POST'])
def trainer():
    return render_template('learning_program.html')
```

Шаблон learning_program.html генерується на основі структури з base.html, що забезпечує єдність стилю між усіма сторінками сайту.

Формат подачі

Навчальна програма структурована у вигляді трьох модулів, кожен з яких містить:

- короткий вступ до теми;
- приклади;
- інтерактивні завдання або питання.

Інформація зберігається у вигляді словників Python, що передаються до шаблону через Flask. Наприклад:

```
module1_content = {
    'title': 'Модуль 1: Основи нечітких чисел',
    'topics': [
        {
            'title': 'Що таке нечітке число?',
            'theory': 'Нечітке число визначається функцією належності...',
            'questions': [
                {
                    'q': 'Яке значення може мати ступінь належності?',
                    'options': ['0 або 1', 'лише 1', 'від 0 до 1', 'від -1 до 1'],
                    'correct': 2
                }
            ]
        }
    ]
}
```

Цей підхід дозволяє легко масштабувати програму, додаючи нові модулі без переписування HTML-шаблонів.

Відображення на сторінці

У HTML-коді передбачено генерацію модулів циклічно через Jinja2:

```
{% for module in modules %}
  <div class="module-card">
    <h2>{{ module.title }}</h2>
    {% for topic in module.topics %}
      <div class="topic-block">
        <h3>{{ topic.title }}</h3>
        <p>{{ topic.theory }}</p>
      </div>
    {% endfor %}
  </div>
{% endfor %}
```

Візуальне оформлення

Кожен модуль візуально виділяється кольором, іконкою та заголовком. Для стилізації використано CSS-файл style.css:

```
.module-card {
    border: 2px solid var(--primary-color);
    border-radius: var(--border-radius);
    padding: 20px;
    margin-bottom: 30px;
    background: var(--bg-secondary);
```

```
    box-shadow: var(--shadow-md);
    transition: var(--transition);
}

.module-card:hover {
    transform: scale(1.02);
}
```

Інтерактивність

Навчальний модуль інтегрується з JavaScript-логікою, яка дозволяє:

- розгортати/згортати теми;
- фіксувати прогрес користувача;
- перевіряти відповіді на прості питання в режимі реального часу (без перезавантаження сторінки).

Модуль навчальної програми поєднує в собі як теоретичні, так і практичні компоненти, стимулюючи користувача до активного навчання та самоперевірки знань у зручному візуальному форматі.

4.3. Розробка модуля навчального калькулятора

Інтерактивний калькулятор є центральним елементом практичного блоку навчального програмного забезпечення. Його основна мета — дати користувачу можливість самостійно виконувати арифметичні операції над нечіткими числами, аналізувати результат і візуально спостерігати функцію належності та α-рівні.

Архітектура функціонування

Калькулятор реалізовано у вигляді окремої HTML-сторінки calculator.html, яка підключається за маршрутом:

```
@app.route('/calculator')
def calculator():
    return render_template('calculator.html')
```

Обчислення виконуються асинхронно через POST-запити до Flask-серверу на маршрут /calculate:

```
@app.route('/calculate', methods=['POST'])
def calculate():
    data = request.get_json()

    fuzzy_a = FuzzyNumber(data['a_x'], data['a_mu'], "A")
    fuzzy_b = FuzzyNumber(data['b_x'], data['b_mu'], "B")
    operation = data['operation']

    result = fuzzy_arithmetic_operation(fuzzy_a, fuzzy_b, operation)
    chart = create_operation_chart(fuzzy_a, fuzzy_b, result, operation)

    return jsonify({
        'success': True,
        'result': result.to_dict(),
        'chart': chart.to_json()
```

```
})
```

Інтерфейс користувача

Форма введення передбачає два блоки:

- введення точок x та значень функції належності μ для двох нечітких чисел (A і B);
- вибір операції: додавання, віднімання, множення, ділення.

Розмітка HTML:

```
<div class="main-panel">
  <div class="number-block" id="inputA">
    <h3>Число A</h3>
    <textarea name="a_values" rows="5"></textarea>
  </div>

  <div class="number-block" id="inputB">
    <h3>Число B</h3>
    <textarea name="b_values" rows="5"></textarea>
  </div>

  <div class="operations-block">
    <button class="big-btn add-btn" data-op="add">+</button>
    <button class="big-btn sub-btn" data-op="sub">−</button>
    <button class="big-btn mul-btn" data-op="mul">×</button>
```

```
    <button class="big-btn div-btn" data-op="div">÷</button>
  </div>
</div>
```

Візуалізація результату

Після виконання операції користувач бачить:

- графіки обох вхідних чисел A та B;
- графік результуючого нечіткого числа;
- таблицю з α-рівнями.

Це забезпечується функцією create_operation_chart з бібліотеки Plotly:

```
def create_operation_chart(A, B, result, operation):
    fig = make_subplots(rows=2, cols=2, ...)
    fig.add_trace(go.Scatter(x=A.x, y=A.mu, name="A"), row=1, col=1)
    fig.add_trace(go.Scatter(x=B.x, y=B.mu, name="B"), row=1, col=2)
    fig.add_trace(go.Scatter(x=result.x, y=result.mu, name="Результат"), row=2, col=1)
    return fig
```

Стилізація та UX

Калькулятор оформлено з використанням адаптивного CSS (calculator.css), що забезпечує:

- підтримку десктопів і мобільних пристроїв;
- плавні анімації при виборі операцій;
- кольорове кодування кнопок (зелений — додавання, червоний — віднімання тощо).

```
.big-btn {
```

```
  padding: 20px;
  border-radius: 12px;
  font-size: 1.5rem;
  color: white;
  cursor: pointer;
  transition: transform 0.2s ease;
}
.big-btn:hover {
  transform: scale(1.05);
}
```

Обробка помилок

Калькулятор передбачає перевірку введених даних на коректність:

- відповідність кількості x і μ;
- перевірка допустимості ділення (ділення на нечітке число, яке містить 0, заборонено);
- повідомлення про помилки виводяться через alert або в інтерфейсі.

Таким чином, модуль калькулятора поєднує в собі математичну точність та зручний візуальний інтерфейс, що робить його важливим інструментом для формування глибшого розуміння операцій з нечіткими числами.

4.4. Розробка система тестування знань

Система тестування знань є важливим компонентом навчального процесу, що дозволяє користувачеві перевірити засвоєння матеріалу з теми "Операції з

нечіткими числами". Вона реалізована як інтерактивна веб-сторінка з підтримкою автоматичної перевірки відповідей, поясненнями, а також виведенням підсумкового результату.

Структура та маршрутизація

У програмному коді передбачено два основні тести:

- quiz1 — з теорії нечітких чисел;
- quiz2 — з арифметичних операцій.

Кожен тест має свій маршрут у Flask-додатку:

```
@app.route('/quiz1', methods=['GET', 'POST'])
def quiz1():
    ...
@app.route('/quiz2', methods=['GET', 'POST'])
def quiz2():
    ...
```

Формат запитань

Запитання зберігаються у вигляді масивів Python-словників:

```
questions_pool = [
    {
        'question': 'Яке з наведених визначень є вірним для нечіткого числа?',
        'options': [
            'Число, що має точне значення',
            'Набір чисел з однаковою імовірністю',
            'Множина зі ступенем належності для кожного елемента',
```

```
            'Гаусова випадкова величина'
        ],
        'correct': 2,
        'explanation': 'Нечітке число — це множина, де кожен елемент має значення функції належності.'
    }
]
```

У режимі GET користувачеві показується випадково відібраний набір запитань, а при POST — обробляються відповіді й обчислюється бал:

```
if request.method == 'POST':
    user_answers = request.form.getlist('answers[]')
    score = calculate_score(user_answers, questions_pool)
    return render_template('quiz1.html', score=score, completed=True)
```

Шаблон сторінки тесту

Форма тестування генерується за допомогою Jinja2. Вона включає запитання з варіантами відповідей (радіокнопки) і кнопку "Перевірити":

```
<form method="POST">
  {% for question in questions %}
    <div class="question-block">
      <p><strong>{{ loop.index }}. {{ question.question }}</strong></p>
      {% for option in question.options %}
        <label>
```

```
            <input type="radio" name="answers[]" value="{{ loop.index0 }}"> {{ option }}
            </label><br>
          {% endfor %}
        </div>
      {% endfor %}
      <button type="submit">Перевірити</button>
    </form>
```

Обчислення результату

Функція calculate_score порівнює відповіді з правильними, підраховує бал і формує пояснення:

```
def calculate_score(user_answers, questions):
    correct = 0
    detailed_results = []

    for i, user_answer in enumerate(user_answers):
        is_correct = int(user_answer) == questions[i]['correct']
        if is_correct:
            correct += 1
        detailed_results.append({
            'question': questions[i]['question'],
            'user_answer': questions[i]['options'][int(user_answer)],
```

```
            'correct_answer': questions[i]['options'][questions[i]['correct']],
            'explanation': questions[i]['explanation'],
            'is_correct': is_correct
        })

    return correct, detailed_results
```

Вивід результатів

Після перевірки відповідей користувач бачить свій бал та розгорнуті пояснення по кожному питанню. Це стимулює не просто вгадати, а й зрозуміти, чому правильна саме така відповідь.

Система тестування сприяє формуванню зворотного зв’язку та самоперевірки, дозволяє оцінити рівень знань користувача та рекомендувати повторне проходження модулів за потреби.

4.5. Інструкція для користувача

У цьому розділі подано покрокову інструкцію з користування веб-платформою FuzzyLearn. Інтерфейс системи розроблений таким чином, щоб забезпечити швидкий доступ до навчальних модулів, інструментів обчислення та перевірки знань.

1. Головна сторінка (див. рис. 4.2)

Після запуску сайту відкривається головна сторінка з назвою платформи, коротким описом функціоналу та кнопками переходу до навчання й тестування. Блок «Навчальні модулі» дозволяє швидко перейти до основних розділів: навчальна програма, інтерактивний калькулятор, система тестування.

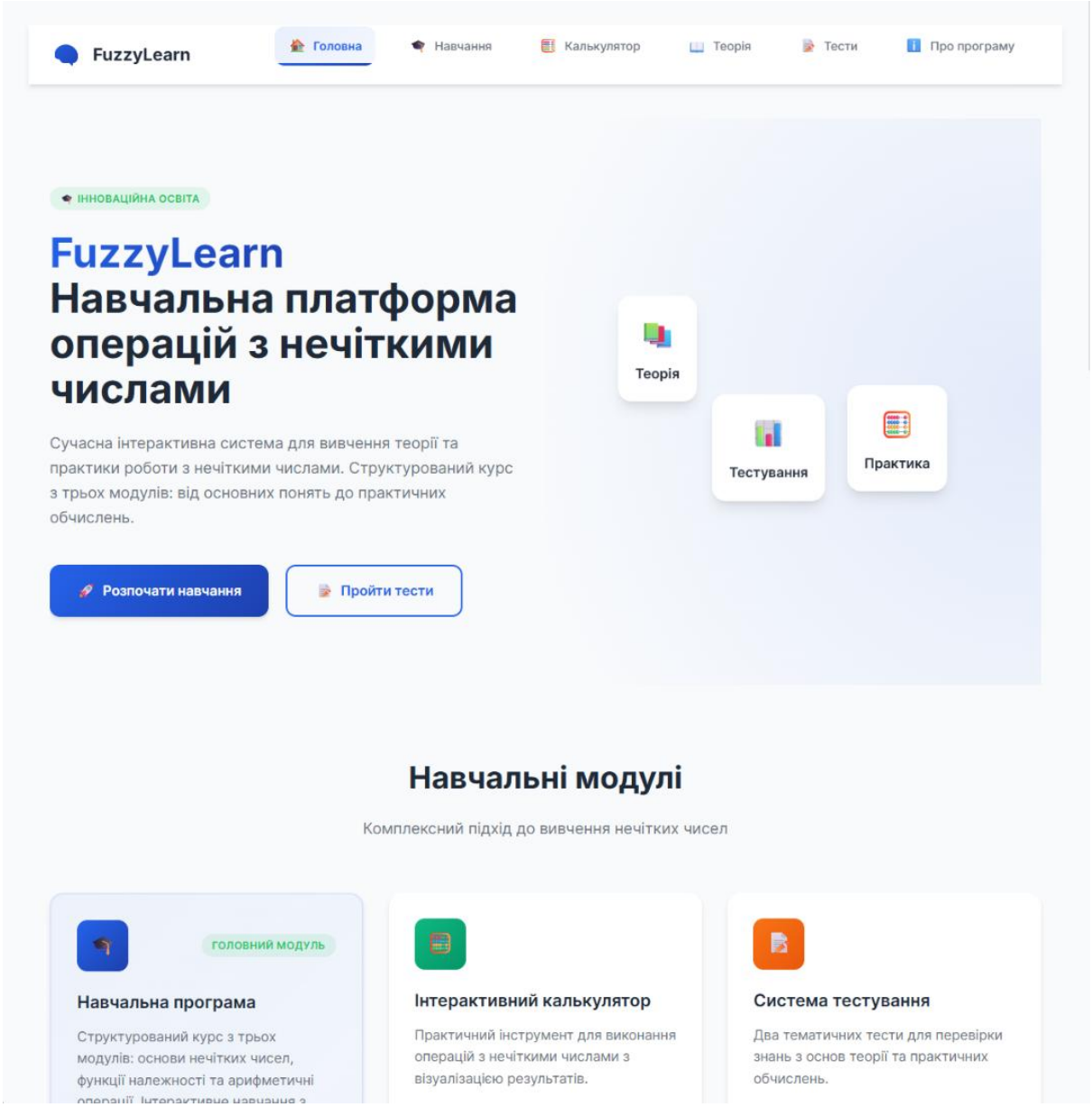

Рисунок 4.2 – Головна сторінка платформи FuzzyLearn

2. Навчальна програма (див. рис. 4.3)

Цей розділ містить три послідовно побудовані модулі:

- Основи нечітких чисел
- Функції належності
- Арифметичні операції

Кожен модуль супроводжується інформацією про теми, кількість запитань, орієнтовну тривалість та кнопкою старту. Користувач може обрати будь-який модуль залежно від рівня підготовки.

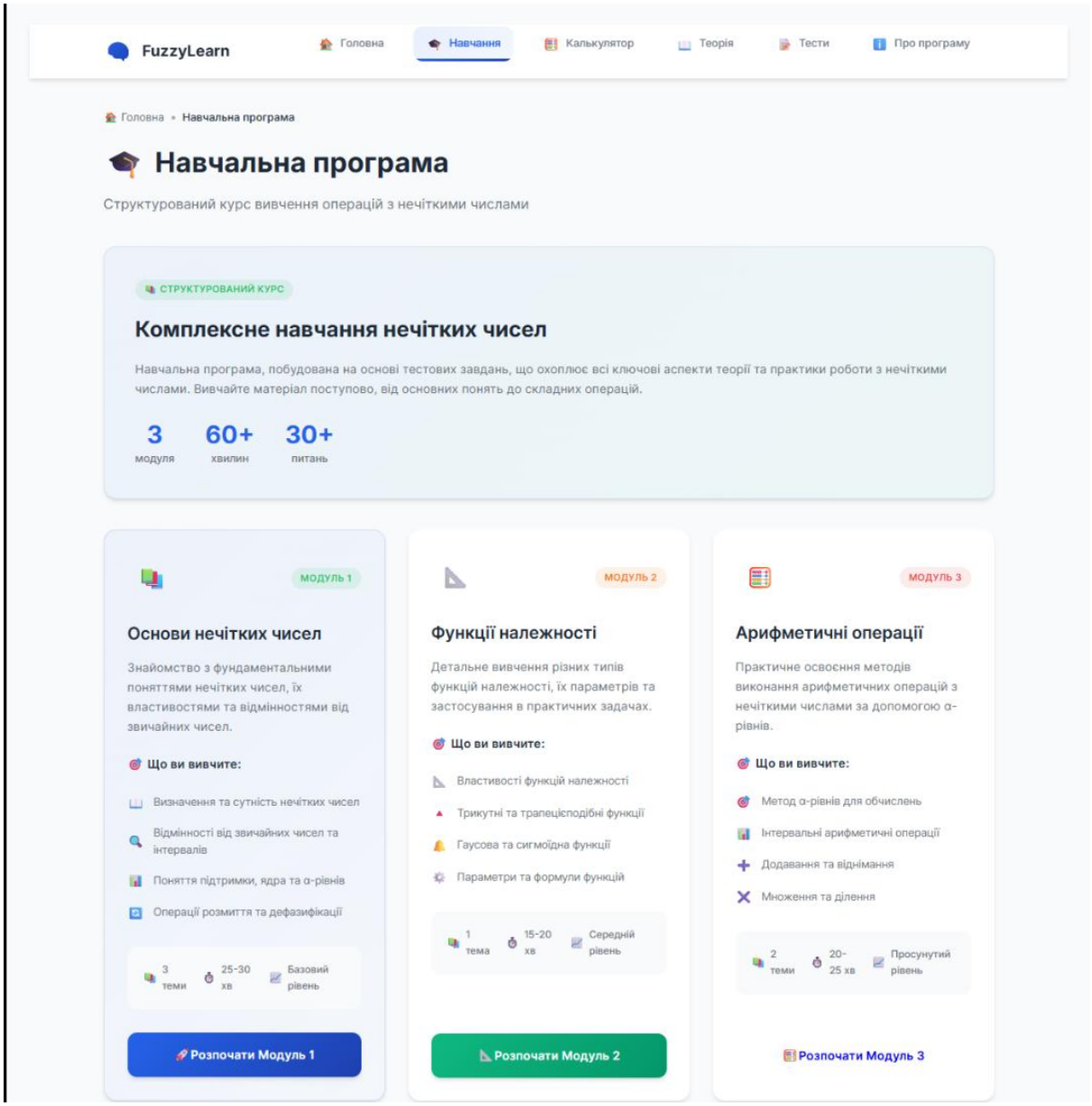

Рисунок 4.3 – Розділ «Навчальна програма»

3. Структура модуля (див. рис. 4.4)

Усередині кожного модуля матеріал подається у логічній послідовності:

- Теоретичний опис поняття
- Приклади
- Контрольні питання з варіантами відповідей

Навчання побудоване так, щоб після кожного блоку теорії йшло миттєве закріплення знань.

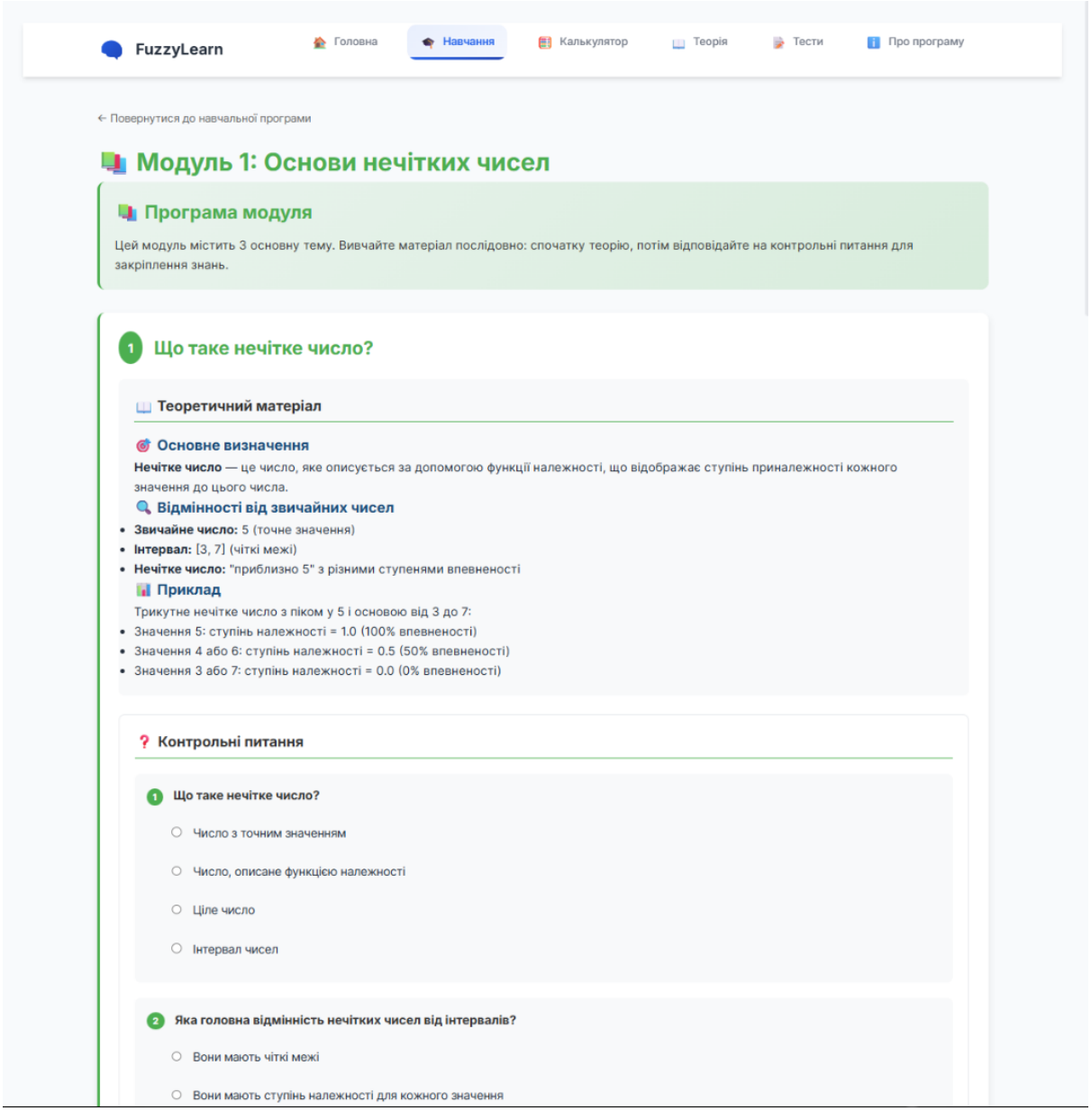

Рисунок 4.4 – Сторінка модуля 1: Основи нечітких чисел

4. Інтерактивний калькулятор (див. рис. 4.5)

У розділі «Калькулятор» користувач вводить координати нечіткого числа A і B, ступені належності, обирає одну з чотирьох операцій: додавання, віднімання, множення або ділення. Передбачено приклади стандартних функцій (трикутна, трапецієподібна). Кнопка «Довідка» дозволяє уточнити правила введення.

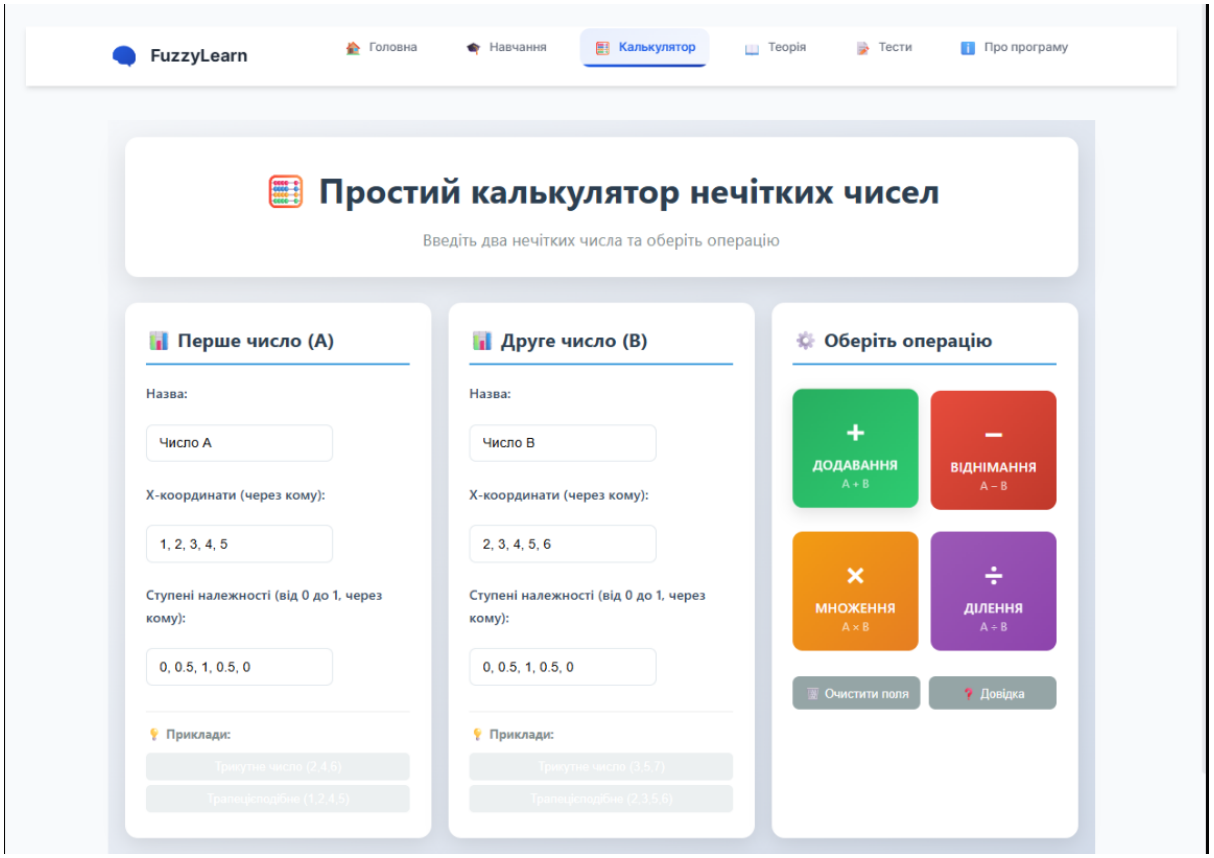

Рисунок 4.5 – Калькулятор нечітких чисел

5. Результати обчислень (див. рис. 4.6)

Після виконання операції користувач бачить:

- графік нечіткого результату
- підтримку, ядро, висоту та центроїд
- інтерактивну побудову графіка (через Plotly)

- зведену таблицю параметрів результату

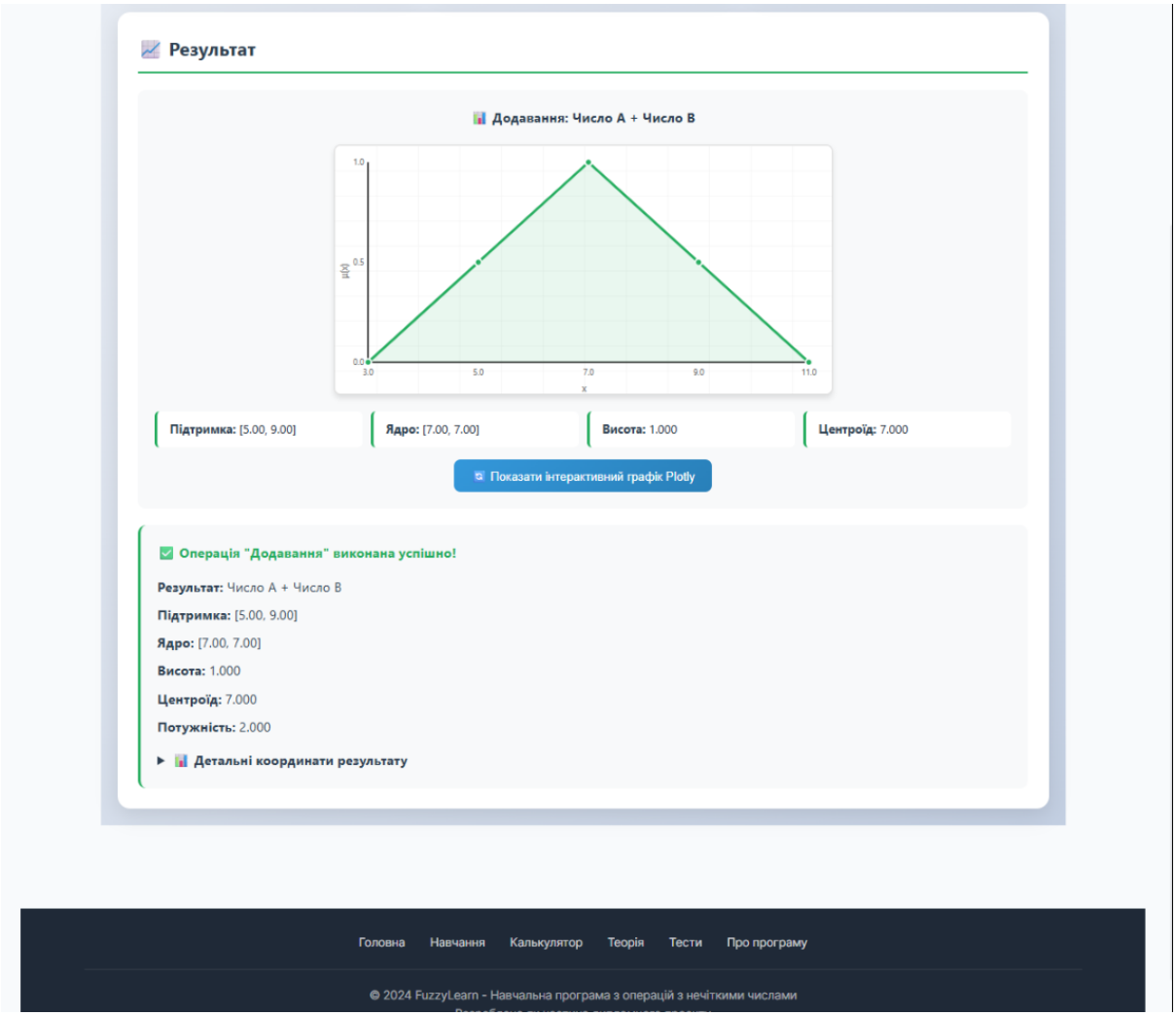

Рисунок 4.6 – Результати обчислень у калькуляторі

6. Теоретичний розділ (див. рис. 4.7)

Окремий розділ «Теорія» містить повний курс із теми нечітких множин. Матеріал структуровано за категоріями: від загальних понять до прикладного використання. Користувач може обирати тему за кнопками у верхній панелі або проходити їх послідовно.

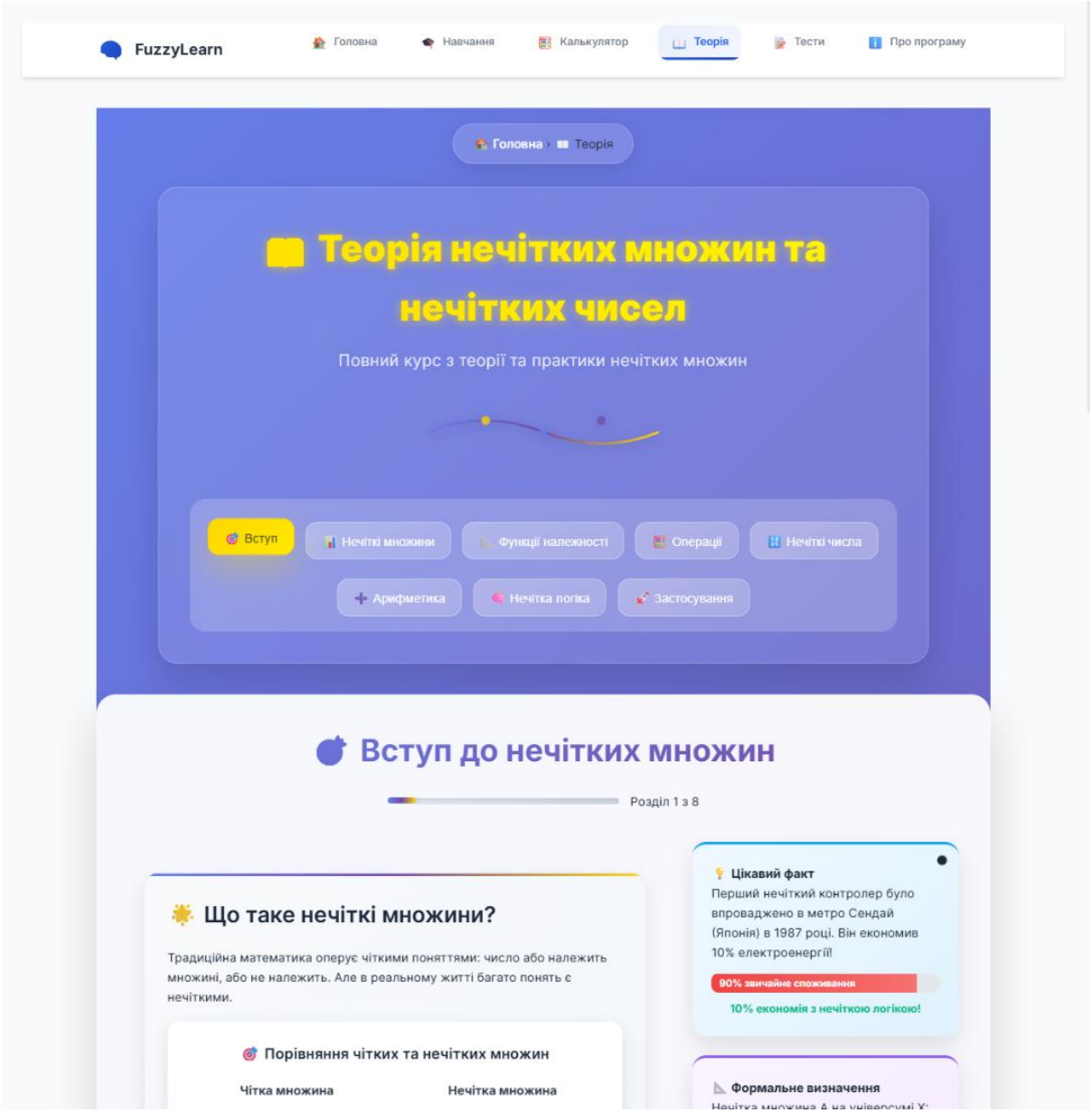

Рисунок 4.7 – Теоретичний блок «Нечіткі множини та числа»

7. Сторінка тестування (див. рис. 4.8)

Розділ «Тести» пропонує два тематичні блоки:

- Тест 1: нечіткі числа і функції належності (25 питань)
- Тест 2: арифметичні операції (13 питань)

Під кожним тестом відображено тематику, кількість питань, тривалість проходження та кнопка початку.

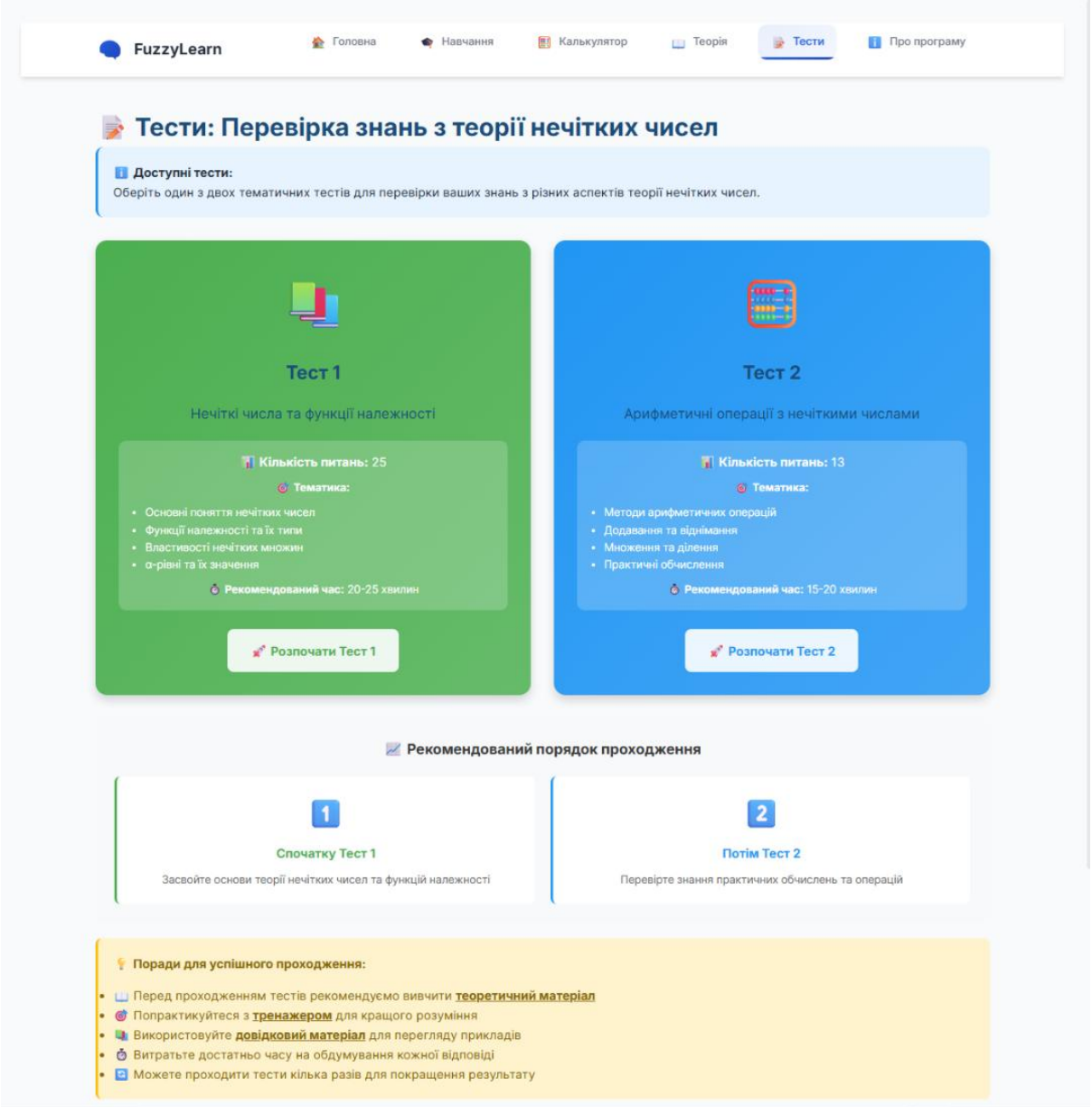

Рисунок 4.8 – Тематичне тестування знань

Інтерфейс платформи є інтуїтивно зрозумілим, адаптивним та візуально комфортним. Він дозволяє користувачам самостійно засвоювати теорію, практикувати навички обчислень і перевіряти рівень знань без сторонньої допомоги. Платформа підходить як для індивідуального навчання, так і для використання в аудиторних умовах.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено дослідження та розроблено програму для виконання операцій з нечіткими числами. Це дозволило отримати глибше розуміння теорії нечіткої логіки та її можливостей для моделювання ситуацій із невизначеністю. У ході роботи вивчено історію розвитку нечітких чисел, основні поняття, методи і алгоритми для виконання операцій, що слугувало базою для створення власного програмного продукту.

Розроблений алгоритм і програмна реалізація охоплюють базові операції з нечіткими числами, такі як додавання, віднімання, множення і порівняння, що дозволяє здійснювати обчислення з урахуванням невизначеності вхідних даних. Програму було реалізовано з використанням мови Python, що обґрунтовується її простотою, доступністю бібліотек для математичних розрахунків і високою ефективністю при роботі з числовими значеннями.

Тестування програми показало, що алгоритм працює коректно і надійно, видаючи очікувані результати на різних наборах даних, включаючи граничні випадки. Це підтверджує правильність вибору алгоритмічних рішень та обраного методу реалізації. З практичної точки зору, результати роботи є важливими для подальшого розвитку методів обробки невизначених даних і можуть бути використані для створення прикладних рішень у галузях, де необхідно враховувати неточність або неоднозначність даних. Розроблена програма може служити основою для подальших наукових досліджень і практичних застосувань у сферах прогнозування, управління ризиками та оптимізації процесів.

Загалом, проведене дослідження і реалізація продемонстрували можливості нечіткої логіки для створення адаптивних рішень, що враховують невизначеність, і відкрили перспективи для подальшого використання нечітких чисел у прикладних задачах.

СПИСОК ЛІТЕРАТУРИ

1. Бутко А. І. Математичні методи нечіткої логіки та їх застосування в штучному інтелекті. Київ : Академвидав, 2015. 256 с.

2. Іванченко В. О., Сидоренко М. П. Нечіткі множини та їх застосування в задачах управління : навч. посіб. Київ : Наукова думка, 2018. 312 с.

3. Губернаторов Ю. А., Ситников С. В., Бурлакова Т. Л. Основи теорії нечітких множин. Харків : ХНУРЕ, 2019. 220 с. Петренко О. М., Ковальчук В. І., Самойленко О. В. Нечіткі множини та нечітка логіка. Київ : Видавництво НПУ ім. М. П. Драгоманова, 2020. 198 с.

4. Гаврилюк І. М., Маляренко О. П. Основи нечіткої математики : навч. посіб. Львів : Видавництво ЛНУ, 2017. 284 с.

5. Твердохліб П. С., Зубенко І. В., Ляшенко О. О. Нечіткі системи та їх застосування в технічних та економічних системах. Одеса : ОНПУ, 2016. 190 с.

6. Дубовик В. О. Математичні моделі на основі нечіткої логіки : монографія. Київ : Політехніка, 2018. 278 с. Ситников С. В. Основи нечіткої логіки та нечітких систем управління. Харків : ХАІ, 2021. 210 с.

7. Гавриш М. О., Плотніков М. Г. Застосування нечітких множин в інтелектуальних системах. Київ : Видавництво КНУ, 2015. 215 с.

8. Вовк О. М., Бабак П. М., Палій О. В. Методи та засоби нечіткої логіки. Дніпро : ДНУ, 2019. 270 с.

9. Бандура С. О., Павленко В. Л., Коваль Л. І. Теорія нечітких множин та її застосування. Харків : Видавництво ХНУ ім. В. Н. Каразіна, 2020. 230 с.

10. Литвин О. С., Яремчук І. М., Микитюк Ю. В. Моделі нечітких систем у системах підтримки прийняття рішень. Київ : Видавничий центр «Київський університет», 2018. 245 с.

11. Козіна Ж. Л. Теоретичні основи нечіткої логіки в задачах класифікації та прогнозування. Теорія та методика інтелектуальних систем. 2016. № 4. С. 12—15.

12. Кравченко О. В. Порівняння традиційних і нечітких підходів до управління. Вісник університету «Україна». 2017. № 3. С. 21—24.

13. Flask Documentation – The Pallets Projects [Електронний ресурс] – Режим доступу: https://flask.palletsprojects.com

14. Plotly Official Documentation – Interactive Graphing & Data Visualization [Електронний ресурс] – Режим доступу:https://plotly.com/python

15. MDN Web Docs – HTML, CSS та JavaScript документація [Електронний ресурс] – Режим доступу:https://developer.mozilla.org

16. PyInstaller Manual – Packaging Python Applications [Електронний ресурс] – Режим доступу:https://pyinstaller.org/en/stable

17. Джон Дакетт: «HTML и CSS. Разработка и дизайн веб-сайтов» [Електронний ресурс] – 18https://kupichitay.com.ua/product/html-i-css-razrabotka-i-dizayn-veb-saytov-dzh/

18. Ольховська О. В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська,. – Полтава : ПУЕТ, 2025. – 67 с. –

ДОДАТКИ

Додаток А. Специфікація ПЗ

Основні функції:

1 Операція додавання двох нечітких чисел.

Результат: нове нечітке число, яке є сумою значень і невизначеностей двох чисел.

2. Операція віднімання двох нечітких чисел.

Результат: нове нечітке число, яке є різницею значень і невизначеностей двох чисел.

3. Операція множення двох нечітких чисел.

Результат: нове нечітке число, яке є результатом множення значень і сумою невизначеностей, враховуючи їх вплив на результат.

4. Операція порівняння двох нечітких чисел.

Результат: виведення результату порівняння:

Якщо перше число більше за друге, вивести: "Більше".

Якщо перше число менше за друге, вивести: "Менше".

Якщо числа наближаються одне до одного, вивести: "Приблизно рівні".

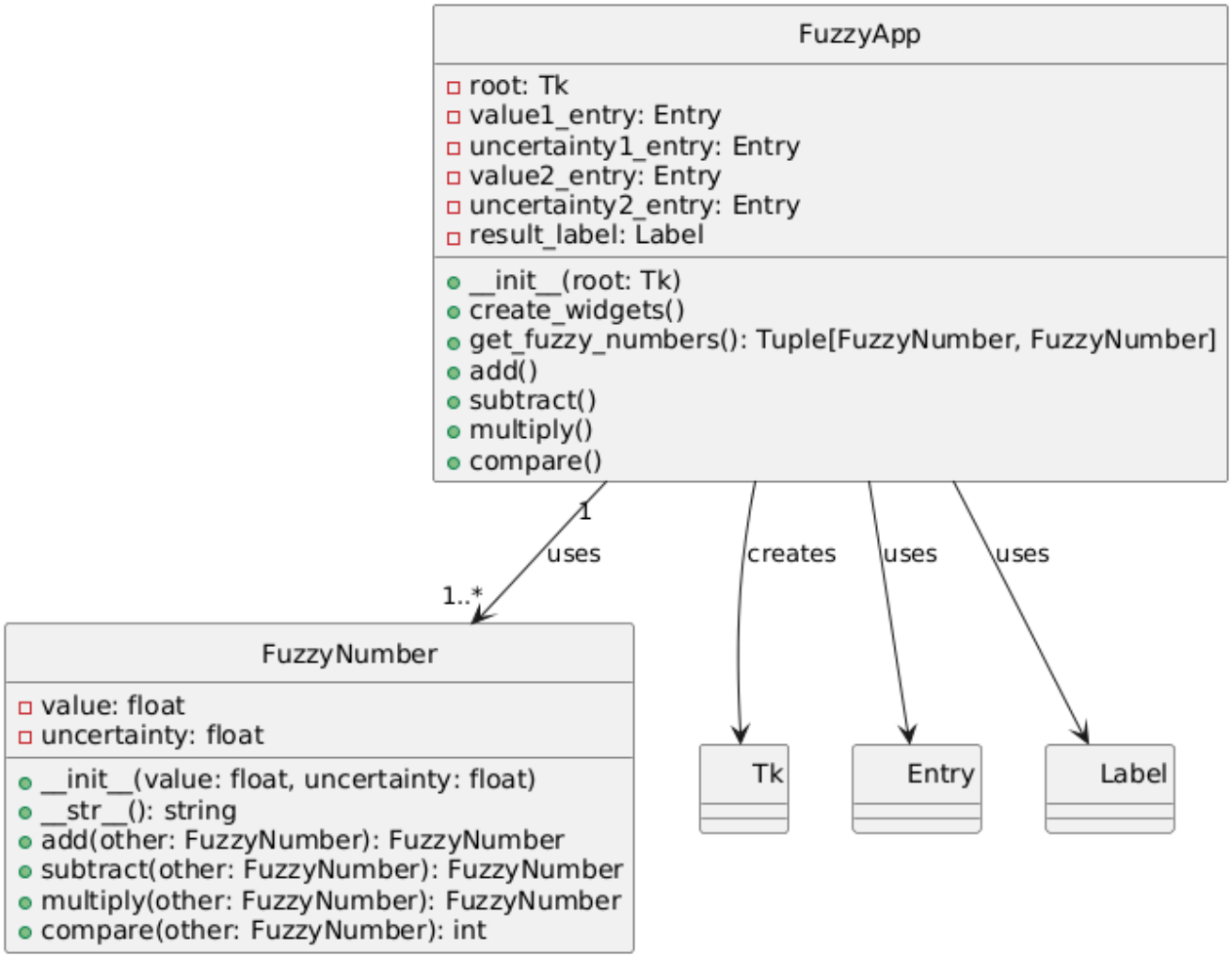

Рисунок 3.1 – Діаграма класів

Додаток Б. Програмний код

```
import tkinter as tk
from tkinter import messagebox

class FuzzyNumber:
    def __init__(self, value, uncertainty):
        self.value = value
        self.uncertainty = uncertainty

    def __str__(self):
        return f"({self.value} ± {self.uncertainty})"

    def add(self, other):
        new_value = self.value + other.value
        new_uncertainty = self.uncertainty + other.uncertainty
        return FuzzyNumber(new_value, new_uncertainty)

    def subtract(self, other):
        new_value = self.value - other.value
        new_uncertainty = self.uncertainty + other.uncertainty
        return FuzzyNumber(new_value, new_uncertainty)

    def multiply(self, other):
        new_value = self.value * other.value
```

```
        new_uncertainty = abs(self.value * other.uncertainty) + abs(other.value *
self.uncertainty)
        return FuzzyNumber(new_value, new_uncertainty)

    def compare(self, other):
        if self.value - self.uncertainty > other.value + other.uncertainty:
            return 1
        elif self.value + self.uncertainty < other.value - other.uncertainty:
            return -1
        else:
            return 0

class FuzzyApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Fuzzy Number Operations")
        self.create_widgets()

    def create_widgets(self):
        # Labels and entries for the first fuzzy number
        tk.Label(self.root, text="Fuzzy Number 1").grid(row=0, column=0,
columnspan=2)
        tk.Label(self.root, text="Value:").grid(row=1, column=0)
        self.value1_entry = tk.Entry(self.root)
        self.value1_entry.grid(row=1, column=1)

        tk.Label(self.root, text="Uncertainty:").grid(row=2, column=0)
        self.uncertainty1_entry = tk.Entry(self.root)
        self.uncertainty1_entry.grid(row=2, column=1)

        # Labels and entries for the second fuzzy number
        tk.Label(self.root, text="Fuzzy Number 2").grid(row=3, column=0,
columnspan=2)
        tk.Label(self.root, text="Value:").grid(row=4, column=0)
        self.value2_entry = tk.Entry(self.root)
        self.value2_entry.grid(row=4, column=1)

        tk.Label(self.root, text="Uncertainty:").grid(row=5, column=0)
        self.uncertainty2_entry = tk.Entry(self.root)
        self.uncertainty2_entry.grid(row=5, column=1)

        # Buttons for operations
        tk.Button(self.root, text="Add", command=self.add).grid(row=6, column=0)
        tk.Button(self.root, text="Subtract", command=self.subtract).grid(row=6,
column=1)
        tk.Button(self.root, text="Multiply", command=self.multiply).grid(row=7,
column=0)
        tk.Button(self.root, text="Compare", command=self.compare).grid(row=7,
column=1)

        # Result display
```

```
        self.result_label = tk.Label(self.root, text="Result: ")
        self.result_label.grid(row=8, column=0, columnspan=2)

    def get_fuzzy_numbers(self):
        try:
            value1 = float(self.value1_entry.get())
            uncertainty1 = float(self.uncertainty1_entry.get())
            fuzzy1 = FuzzyNumber(value1, uncertainty1)

            value2 = float(self.value2_entry.get())
            uncertainty2 = float(self.uncertainty2_entry.get())
            fuzzy2 = FuzzyNumber(value2, uncertainty2)

            return fuzzy1, fuzzy2
        except ValueError:
            messagebox.showerror("Invalid Input", "Please enter valid numbers.")
            return None, None

    def add(self):
        fuzzy1, fuzzy2 = self.get_fuzzy_numbers()
        if fuzzy1 and fuzzy2:
            result = fuzzy1.add(fuzzy2)
            self.result_label.config(text=f"Result: {fuzzy1} + {fuzzy2} = {result}")

    def subtract(self):
        fuzzy1, fuzzy2 = self.get_fuzzy_numbers()
        if fuzzy1 and fuzzy2:
            result = fuzzy1.subtract(fuzzy2)
            self.result_label.config(text=f"Result: {fuzzy1} - {fuzzy2} = {result}")

    def multiply(self):
        fuzzy1, fuzzy2 = self.get_fuzzy_numbers()
        if fuzzy1 and fuzzy2:
            result = fuzzy1.multiply(fuzzy2)
            self.result_label.config(text=f"Result: {fuzzy1} * {fuzzy2} = {result}")

    def compare(self):
        fuzzy1, fuzzy2 = self.get_fuzzy_numbers()
        if fuzzy1 and fuzzy2:
            comparison = fuzzy1.compare(fuzzy2)
            if comparison == 1:
                result_text = f"{fuzzy1} > {fuzzy2}"
            elif comparison == -1:
                result_text = f"{fuzzy1} < {fuzzy2}"
            else:
                result_text = f"{fuzzy1} ≈ {fuzzy2}"
            self.result_label.config(text=f"Result: {result_text}")

if __name__ == "__main__":
    root = tk.Tk()
    app = FuzzyApp(root)
```

```
root.mainloop()
```

Додаток В. Схемиа та діаграми

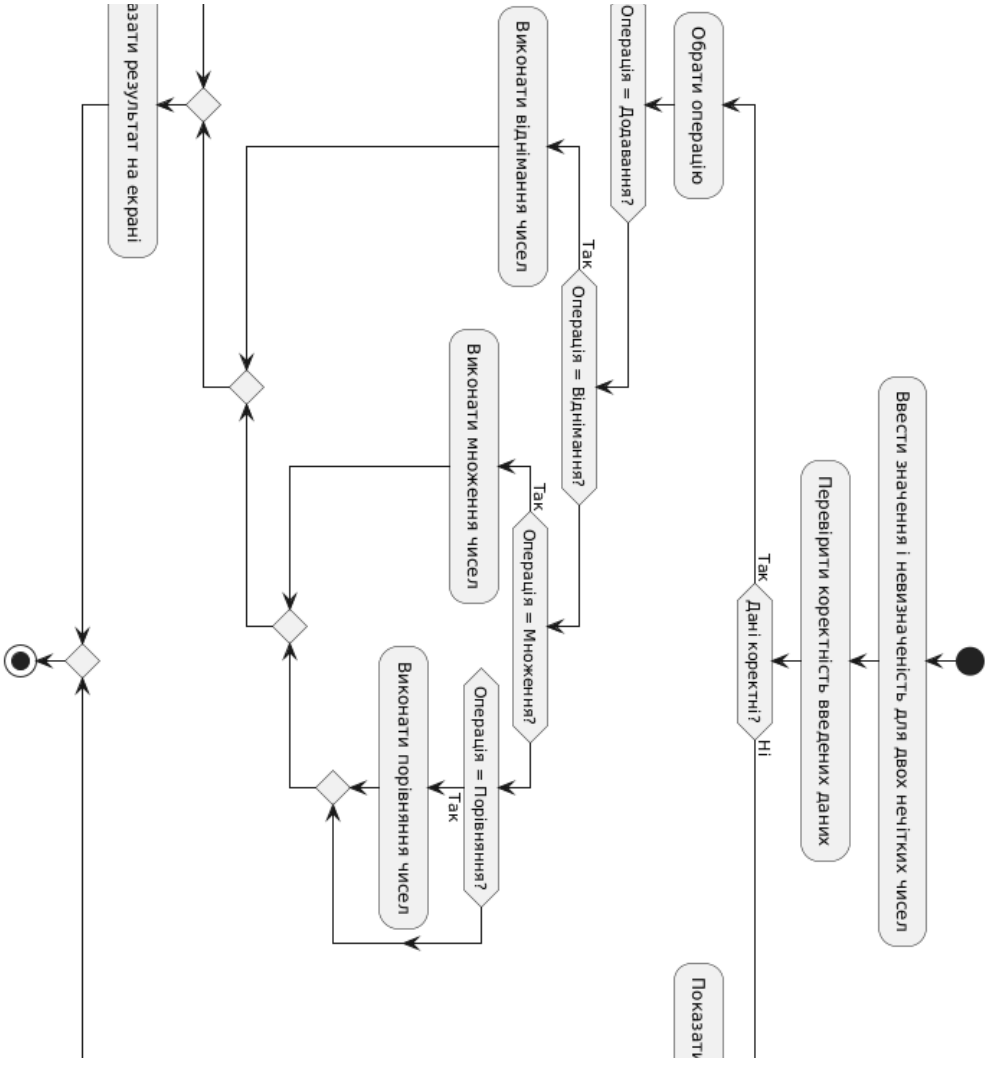

Рисунок 3.1 – Алгоритм роботи програми

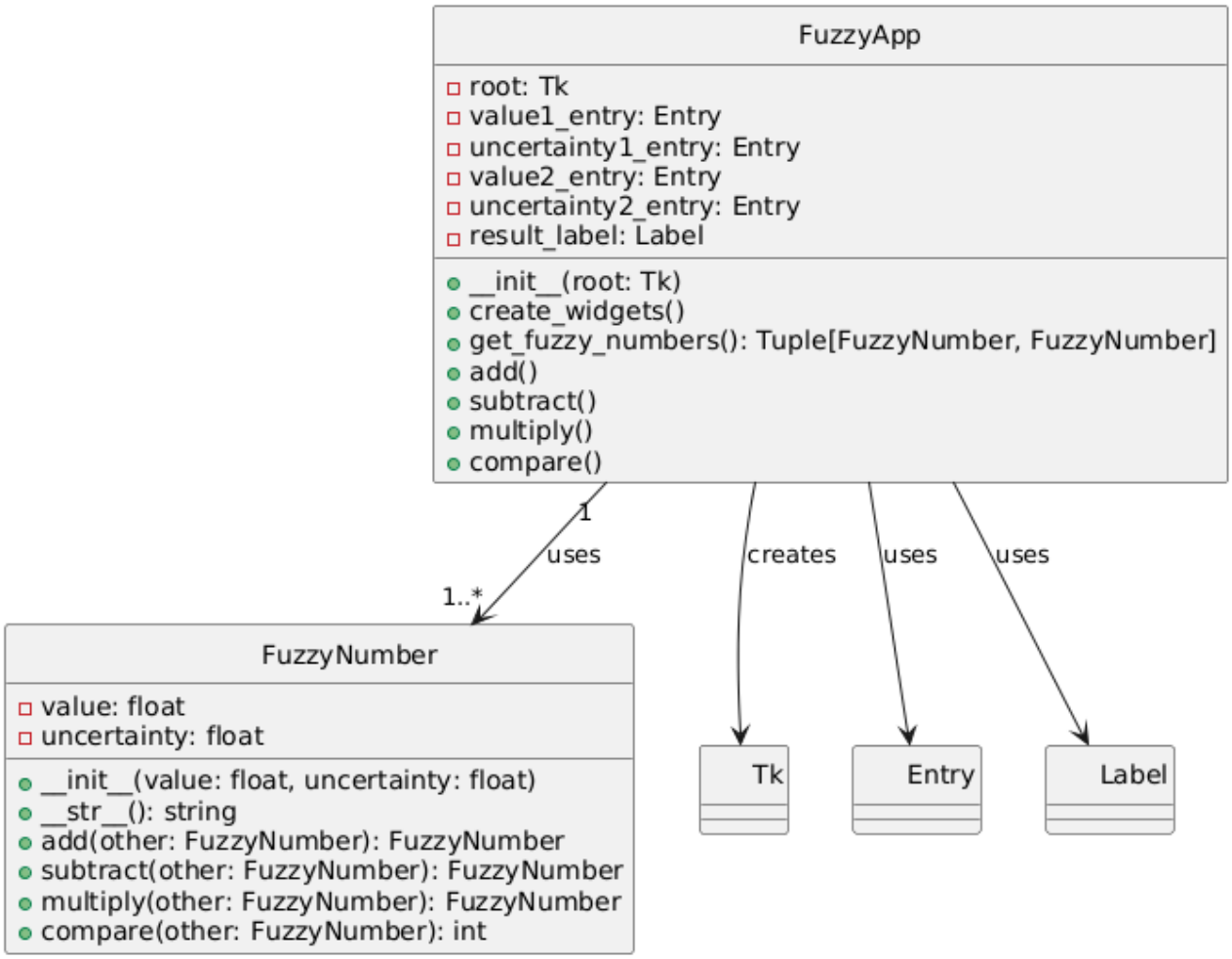

Рисунок 3.3 – Діаграма класів

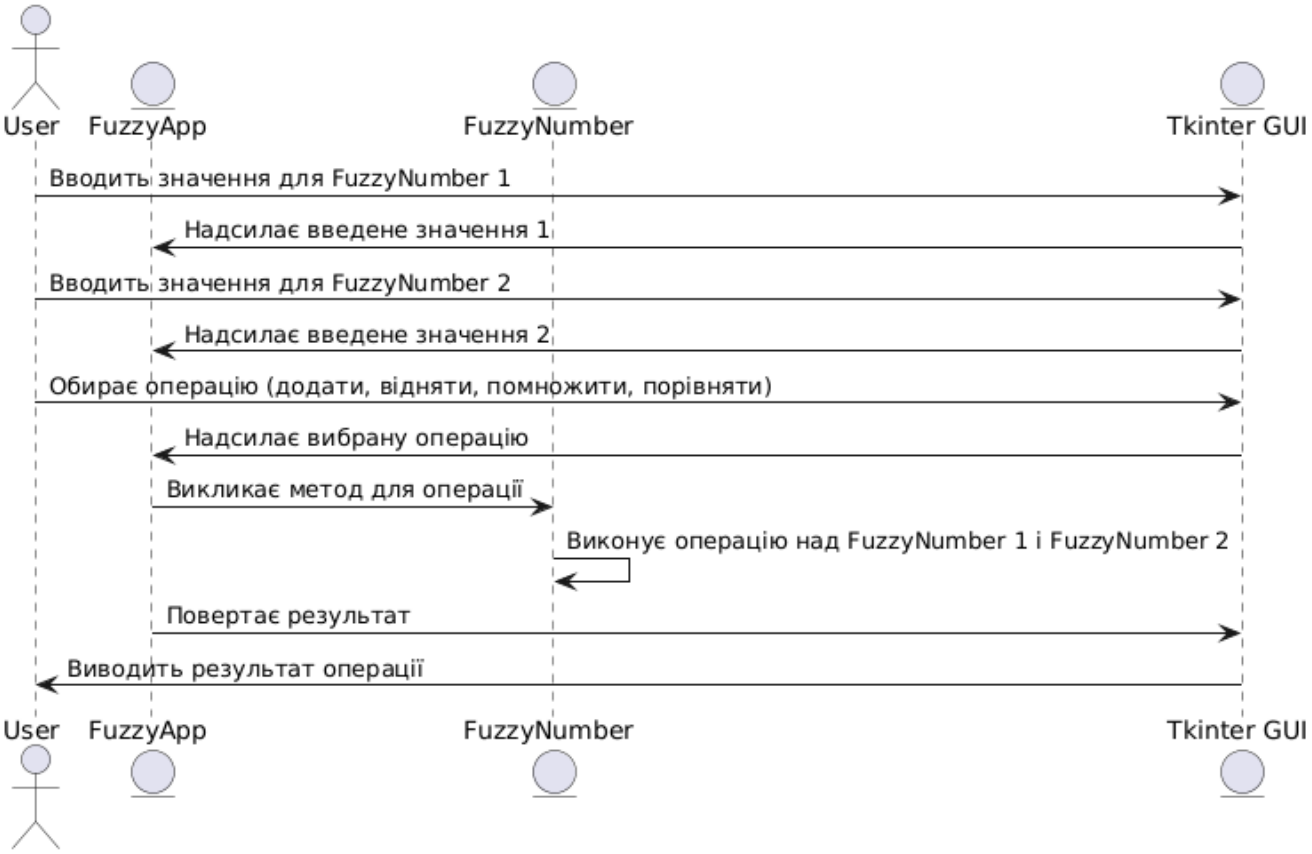

Рисунок 3.1 – Діаграма алгоритму роботи програми