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)

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

РОЗРОБКА ІНТЕРАКТИВНОГО ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ НАВЧАННЯ МЕТОДАМ ЧИСЕЛЬНОЇ ОПТИМІЗАЦІЇ

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Швирид Тарас Іванович
_____ «____»_____ 2025 р.
(підпис)

Науковий керівник к.ф.-м.н., доцент, Чілікіна Тетяна Василівна
_____ «____»_____ 2025 р.
(підпис)

Рецензент

ПОЛТАВА 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка інтерактивного програмного комплексу для навчання методам чисельної оптимізації**»**
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Швирид Тарас Іванович
Затверджена наказом ректора № ____ _____ 2024 р..
Термін подання студентом роботи «___»_____ 2025 р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки сервісів для розробки навчальних декстопних застосунків.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Огляд сучасних навчальних програм з чисельних методів і оптимізації
2.2. Порівняльний аналіз програмних засобів для моделювання оптимізаційних процесів
2.3. Аналіз інтерфейсів та дидактичних можливостей подібних систем
2.4. Визначення недоліків існуючих рішень і шляхів їх усунення
2.5. Обґрунтування необхідності розробки інтерактивного навчального комплексу
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Класифікація методів чисельної оптимізації (градієнтні, дихотомічні, золотого перетину, Ньютона, генетичні тощо)
3.2. Математичне обґрунтування вибраних методів оптимізації
3.3. Алгоритмічні схеми методів та їх порівняльний аналіз за критеріями точності та швидкодії
3.4. Проектування архітектури інтерактивного навчального комплексу та обгрунтування обраних іструментів розробки
3.5. Побудова блок-схем, діаграм послідовності, компонентів і класів
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка структури програмного комплексу та модулів взаємодії
4.2. Реалізація навчального модуля з візуалізацією процесу оптимізації та модуля методу дихотомії
4.3. Побудова інтерактивного інтерфейсу користувача
4.4. Інструкція з використання програмного комплексу
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А
Перелік графічного матеріалу: 19 ілюстрацій.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Чілікіна Т.В.		
Інформаційний огляд	Чілікіна Т.В.		
Теоретична частина	Чілікіна Т.В.		
Практична частина	Чілікіна Т.В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
Вступ		
Вивчення методичних рекомендацій та стандартів та звіт керівнику		
Постановка задачі		
Інформаційний огляд джерел бібліотек та інтернету		
Теоретична частина		
Практична частина		
Закінчення оформлення		
Доповідь студента на кафедрі		
Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 2024 р.

Здобувач вищої освіти _____ Швирид Тарас Іванович
(підпис)

Науковий керівник _____ к.ф.-м.н., доц. Чілікіна Т.В.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на
_____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2025 р.

Секретар ЕК_____ _____
(підпис)
(ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 2024 р.

Погоджено
Науковий керівник _____
к.пед.н., Тетяна ЧІЛІКІНА
«____» _____ 2024 р.

План

дипломного проекту з фаху
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
на тему «Розробка інтерактивного програмного комплексу для навчання методам чисельної оптимізації»
Прізвище, ім'я, по батькові Швирид Тарас Іванович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Огляд сучасних навчальних програм з чисельних методів і оптимізації
2.2. Порівняльний аналіз програмних засобів для моделювання оптимізаційних процесів
2.3. Аналіз інтерфейсів та дидактичних можливостей подібних систем
2.4. Визначення недоліків існуючих рішень і шляхів їх усунення
2.5. Обґрунтування необхідності розробки інтерактивного навчального комплексу
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Класифікація методів чисельної оптимізації (градієнтні, дихотомічні, золотого перетину, Ньютона, генетичні тощо)
3.2. Математичне обґрунтування вибраних методів оптимізації
3.3. Алгоритмічні схеми методів та їх порівняльний аналіз за критеріями точності та швидкодії
3.4. Проектування архітектури інтерактивного навчального комплексу та обґрунтування обраних іструментів розробки
3.5. Побудова блок-схем, діаграм послідовності, компонентів і класів
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка структури програмного комплексу та модулів взаємодії
4.2. Реалізація навчального модуля з візуалізацією процесу оптимізації та модуля методу дихотомії
4.3. Побудова інтерактивного інтерфейсу користувача
4.4. Інструкція з використання програмного комплексу
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ Т.І. Швирид
(підпис)
«____» _____ 2024 р.

РЕФЕРАТ

Записка: 86 сторінок, 19 рисунків, 1 додаток, 12 літературних джерел.

Метою даного дослідження є розробка інтерактивного програмного комплексу для навчання методам чисельної оптимізації, який забезпечує можливість моделювання, аналізу та візуалізації результатів роботи класичних алгоритмів оптимізації на практичних прикладах.

Об’єктом дослідження є процес навчання методам чисельної оптимізації з використанням комп’ютерних технологій.

Предметом дослідження виступають методи, алгоритми та програмні засоби чисельної оптимізації, а також принципи побудови інтерактивних навчальних систем.

У процесі розробки застосовувалися такі методи: модульне програмування, об’єктно-орієнтоване проектування, візуальне моделювання, а також методи математичного аналізу та чисельних обчислень. Методологічну основу роботи становив структурно-модульний підхід, який забезпечує незалежність функціональних частин системи, зручність тестування та можливість подальшого розширення комплексу.

Інструментарій реалізації включав мову програмування C# і фреймворк Windows Presentation Foundation (WPF) у складі .NET Framework 4.8. Використання WPF зумовлене його здатністю до побудови масштабованих інтерфейсів із векторною графікою, підтримкою дво- та тривимірної візуалізації, анімацій і даних, що динамічно оновлюються. Розробка здійснювалася в середовищі Visual Studio. Для зберігання та обміну даними застосовано формат XML, який забезпечує сумісність і зручність структурування результатів.

У межах роботи обґрунтовано доцільність використання методу дихотомії як базового для демонстрації принципів чисельного пошуку мінімуму

функції; розроблено архітектуру програмного комплексу, що включає модулі введення/виведення даних, обчислень і візуалізації; створено навчальний модуль, який відображає динаміку процесу оптимізації в графічному вигляді; побудовано алгоритмічні схеми та діаграми компонентів, що демонструють взаємозв’язок між структурними частинами системи; розроблено інтерактивний інтерфейс користувача, який забезпечує введення функції, параметрів пошуку та перегляд результатів обчислень.

Розроблений програмний комплекс є автономним десктопним застосунком, який не потребує підключення до мережі Інтернет, має невеликий обсяг, простий у встановленні й орієнтований на використання у навчальному процесі. Система може бути інтегрована в електронні навчальні курси, лабораторні заняття та платформи дистанційного навчання.

Практична цінність роботи полягає у створенні інструменту, що поєднує точність чисельних методів, наочність графічного представлення та зручність інтерактивної взаємодії користувача з алгоритмами. Програмний комплекс може використовуватися як демонстраційний засіб у навчальних дисциплінах, а також як базова платформа для подальшої розробки систем моделювання оптимізаційних процесів.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 11
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ 14
2.1. Огляд сучасних навчальних програм з чисельних методів і оптимізації . 14
2.2. Порівняльний аналіз програмних засобів для моделювання оптимізаційних процесів 22
2.3. Аналіз інтерфейсів та дидактичних можливостей подібних систем 26
2.4. Визначення недоліків існуючих рішень і шляхів їх усунення 30
2.5. Обґрунтування необхідності розробки інтерактивного навчального комплексу 35
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА 38
3.1. Класифікація методів чисельної оптимізації 38
3.2. Математичне обгрунтування вираних методів оптимізації 43
3.3. Алгоритмічні схеми методів та їх порівняльний аналіз за критеріями точності та швидкодії 45
3.4. Проєктування архітектури інтерактивного навчального комплексу та обгрунтування обраних іструментів розробки 48
3.5. Побудова блок-схем, діаграм послідовності, компонентів і класів 52
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА 59
4.1. Розробка структури програмного комплексу та модулів взаємодії 59
4.2. Реалізація навчального модуля з візуалізацією процесу оптимізації та модуля методу дихотомії 62
4.3. Побудова інтерактивного інтерфейсу користувача 66
4.4. Інструкція з використання програмного комплексу 70
ВИСНОВКИ 78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 80
ДОДАТОК А 82

ВСТУП

Актуальність. У сучасному інформаційному суспільстві цифровізація освіти та інтенсивний розвиток комп’ютерних технологій призводять до активного впровадження інноваційних інструментів у навчальний процес. Особливої уваги заслуговують інтерактивні системи, що поєднують теоретичне навчання з практичними обчисленнями та візуалізацією результатів. Одним із важливих напрямів є створення навчальних програмних комплексів для вивчення методів чисельної оптимізації, які широко застосовуються у науці, інженерії, економіці, машинному навчанні та інших галузях, де необхідно знаходити екстремуми функцій.

Використання сучасних засобів програмування дозволяє реалізовувати динамічні, наочні моделі процесів оптимізації, що забезпечує глибше розуміння принципів роботи алгоритмів, підвищує мотивацію студентів і розвиває їх аналітичне мислення. Проте більшість існуючих навчальних програм обмежується текстовими демонстраціями або статичними прикладами без інтерактивного аналізу проміжних результатів. Це зумовлює потребу в розробці програмного комплексу, який би поєднував обчислювальні модулі з візуалізацією, моделюванням та можливістю взаємодії користувача з процесом оптимізації в реальному часі.

Метою даного дослідження є розробка інтерактивного програмного комплексу для навчання методам чисельної оптимізації, який забезпечує можливість моделювання, аналізу та візуалізації результатів роботи класичних алгоритмів оптимізації на практичних прикладах.

Об’єктом дослідження є процес навчання методам чисельної оптимізації з використанням комп’ютерних технологій.

Предметом дослідження виступають методи, алгоритми та програмні засоби чисельної оптимізації, а також принципи побудови інтерактивних навчальних систем.

Завдання дослідження полягають наступному:

– провести аналіз існуючих навчальних систем і програмних засобів для візуалізації процесів оптимізації;

– дослідити математичні основи класичних методів чисельної оптимізації (дихотомії, золотого перетину, градієнтного спуску, методу Ньютона тощо);

– створити архітектуру програмного комплексу, що передбачає модульну структуру та можливість розширення;

– реалізувати засоби імпорту даних, запуску оптимізаційних алгоритмів і збереження результатів;

– розробити інтерактивний графічний інтерфейс для візуального спостереження за процесом оптимізації;

– здійснити тестування системи на прикладах задач із відомими аналітичними розв’язками.

Розроблення комплексу базується на результатах попередніх робіт, у яких було створено консольний застосунок для пошуку мінімуму функції методом дихотомії. Програма реалізує обробку даних у форматі XML, роботу з поліноміальними функціями, визначення мінімуму на заданому проміжку та збереження результатів у звітному файлі. Ця система стала основою для подальшого вдосконалення - переходу до інтерактивного середовища навчання з розширеною підтримкою методів оптимізації та зручним візуальним інтерфейсом.

Запропонований комплекс базується на принципах об’єктно-орієнтованого проєктування та модульної архітектури, що забезпечує можливість масштабування, доповнення новими алгоритмами та інтеграції з навчальними платформами. У процесі розробки застосовуються методи модульного тестування, побудови UML-діаграм, алгоритмічного моделювання та візуалізації даних.

Інструментарій реалізації включає мову програмування C# та середовище Visual Studio, бібліотеку System.Xml для зчитування даних, засоби візуалізації функцій, а також власну бібліотеку Library, що містить класи Data, Container, Dich, Point для організації математичних обчислень.

Пояснювальна записка складається з чотирьох основних розділів: постановки задачі, огляду сучасних систем аналогічного призначення, теоретичної частини з описом математичних методів і архітектури програмного комплексу, а також практичної частини, у якій наведено процес розробки, тестування та аналізу роботи системи. Структура роботи орієнтована на послідовне розкриття проблеми та практичну реалізацію ефективного навчального інструменту для вивчення методів чисельної оптимізації.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

У процесі підготовки фахівців у галузі комп’ютерних наук значну роль відіграє вивчення методів чисельної оптимізації, які застосовуються в різних сферах людської діяльності - від математичного моделювання та економічного прогнозування до машинного навчання, технічного проєктування й управління складними системами. Ці методи дозволяють знаходити екстремальні значення функцій, коли аналітичний розв’язок є складним або неможливим. Попри широку сферу застосування, їх практичне вивчення в освітньому процесі часто ускладнене відсутністю інтерактивних засобів, які б наочно демонстрували послідовність і логіку роботи алгоритмів оптимізації.

Завдання полягає у створенні інтерактивного програмного комплексу, який забезпечує можливість моделювання, аналізу та візуалізації процесів чисельної оптимізації. Розроблюваний комплекс має бути навчальним інструментом, що поєднує теоретичні аспекти з практичними експериментами, дозволяє користувачеві змінювати параметри обчислень, спостерігати результати на кожному етапі та досліджувати вплив різних алгоритмічних підходів на точність і швидкодію розв’язку.

Основна мета полягає у створенні програмного засобу, який надає користувачу можливість дослідження процесу пошуку екстремуму функцій з використанням чисельних методів. Програмний комплекс повинен мати зручний інтерфейс, який дає змогу задавати початкові дані: вид функції, межі інтервалу пошуку, параметри точності та отримувати візуальне представлення результатів. Інтерактивна взаємодія з алгоритмом забезпечить краще розуміння принципів його роботи, а також сприятиме формуванню навичок практичного застосування методів оптимізації у майбутній професійній діяльності.

Одним із центральних компонентів системи є реалізація методу дихотомії, який належить до класу одновимірних методів чисельної оптимізації. Цей метод базується на поступовому звуженні інтервалу, що

містить точку мінімуму або максимуму функції. На кожній ітерації відрізок [a,b][a, b][a,b] ділиться на дві частини за допомогою двох симетричних точок, розташованих на відстані δ від середини інтервалу. Після обчислення значень функції у цих точках порівнюються результати, і новий інтервал визначається таким чином, щоб у ньому залишалася область із меншим значенням функції. Процес повторюється доти, поки довжина інтервалу не стане меншою за задану точність ε.

Метод дихотомії відзначається простотою реалізації, логічною прозорістю та стабільною збіжністю для неперервних функцій. Саме ці властивості роблять його доцільним для використання в навчальних цілях. Він наочно демонструє сутність ітераційних методів оптимізації - поступове уточнення області пошуку мінімуму або максимуму - та дозволяє візуально простежити процес наближення до розв’язку. У програмному комплексі метод дихотомії буде реалізовано як базовий алгоритм, на основі якого здійснюється подальше вивчення та порівняння ефективності інших чисельних методів.

З технічної точки зору, реалізація програмного комплексу передбачає використання сучасних інструментів і технологій розробки програмного забезпечення. Для створення обчислювальної частини планується використати мову програмування C#, яка забезпечує високу продуктивність, підтримку об’єктно-орієнтованої парадигми та широкий набір бібліотек для математичних обчислень. Як середовище розробки обрано Microsoft Visual Studio, що дозволяє ефективно організувати структуру проєкту, здійснювати налагодження та тестування.

Для побудови графічного інтерфейсу користувача буде використано технологію Windows Presentation Foundation (WPF), яка забезпечує гнучкі можливості візуалізації даних, підтримує інтерактивну взаємодію з користувачем і дозволяє створити зручне навчальне середовище. Модуль візуалізації передбачає відображення графіків функцій, демонстрацію проміжних точок пошуку мінімуму та динамічну зміну інтервалу обчислень у

процесі роботи алгоритму. Для збереження вхідних та вихідних даних планується використання формату XML, що забезпечує структуроване представлення інформації й зручну взаємодію між модулями системи.

Архітектура комплексу буде модульною і включатиме три основні частини: обчислювальне ядро, графічний модуль візуалізації та модуль введення-виведення даних. Такий підхід дозволить у подальшому розширювати функціональні можливості системи, додавати нові алгоритми оптимізації та інтегрувати програмний продукт із зовнішніми навчальними платформами.

Розроблюваний програмний комплекс поєднає точність математичних обчислень із наочністю представлення процесів оптимізації, що забезпечить ефективне поєднання теоретичної підготовки студентів із практичними навичками моделювання та аналізу алгоритмів у контексті сучасної комп’ютерної освіти.

РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

2.1. Огляд сучасних навчальних програм з чисельних методів і оптимізації

Сучасний розвиток обчислювальної математики, штучного інтелекту та хмарних технологій суттєво розширив можливості реалізації чисельних методів оптимізації у прикладних і навчальних системах. Оптимізація стала невід'ємним інструментом аналізу, прогнозування, керування та навчання, у сферах економіки, інженерії, енергетики, транспорту, біоінформатики й машинного навчання. Це зумовило активне створення програмних продуктів, здатних не лише виконувати обчислення, але й забезпечувати інтерактивну взаємодію користувача з алгоритмічними процесами.

У загальному вигляді сучасні системи чисельної оптимізації можна класифікувати за кількома критеріями: за призначенням (навчальні, науково-дослідні, прикладні, інженерні), за типом реалізації (настільні програми, бібліотеки, веб-платформи, хмарні сервіси) та за алгоритмічною базою (градієнтні, стохастичні, еволюційні, комбінаторні, гібридні методи). Кожна група має свої особливості щодо гнучкості, точності, обчислювальної ефективності та зручності інтеграції у навчальний процес.

Історично одним із найвідоміших середовищ, у якому реалізовано чисельну оптимізацію, є MATLAB, розроблений компанією MathWorks. MATLAB поєднує високу швидкість обчислень із гнучкістю візуалізації та має розширення Optimization Toolbox, що підтримує широкий спектр алгоритмів — від класичних градієнтних методів (метод Ньютона, квазі-Ньютона, спряжених градієнтів) до еволюційних і глобальних оптимізаторів (генетичні алгоритми, симульоване відпалу, рої частинок). Завдяки універсальності MATLAB став фактичним стандартом у науково-інженерній освіті. Він широко застосовується для навчання методам оптимізації у курсах з обчислювальної математики,

аналізу даних та машинного навчання, зокрема у вищих навчальних закладах різних країн.

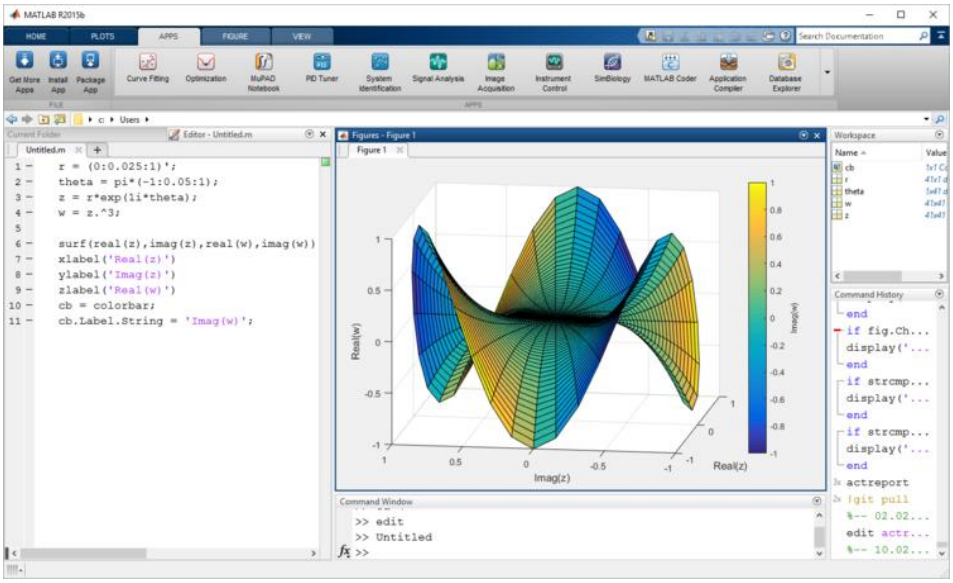

Рисунок 2.1. Інтерфейс MATLAB

Іншим поширеним середовищем є Wolfram Mathematica, яке відрізняється потужною символьної математикою та можливістю аналітичного аналізу функцій перед проведенням чисельної оптимізації. Інтеграція символьного та чисельного рівнів дозволяє користувачу спочатку дослідити властивості функції, а потім застосувати чисельні методи з високим рівнем контролю параметрів. Mathematica реалізує методи локальної й глобальної оптимізації, має функції FindMinimum та NMinimize, які підтримують автоматичне вибирання алгоритму залежно від типу задачі. Вона часто використовується у дослідницьких лабораторіях для експериментального аналізу алгоритмів оптимізації.

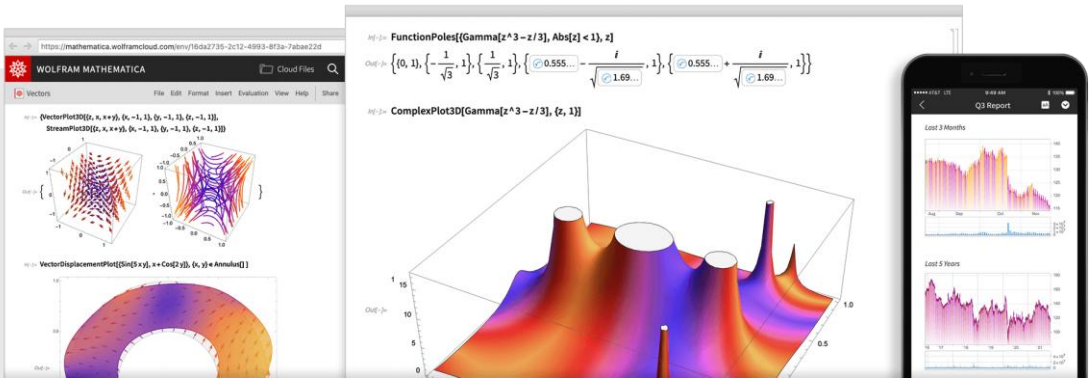

Рисунок 2.2. Інтерейс Wolfram Mathematica

До безкоштовних аналогів таких середовищ належить Scilab - відкрите програмне середовище, створене як альтернатива MATLAB. Scilab містить численні функції для оптимізації, у тому числі методи градієнтного спуску, Ньютона, квазі-Ньютона, а також стохастичні підходи. Його модуль SciOptimization використовується в освітніх програмах для вивчення чисельних методів, оскільки забезпечує відкритий вихідний код, графічне середовище та кросплатформну сумісність.

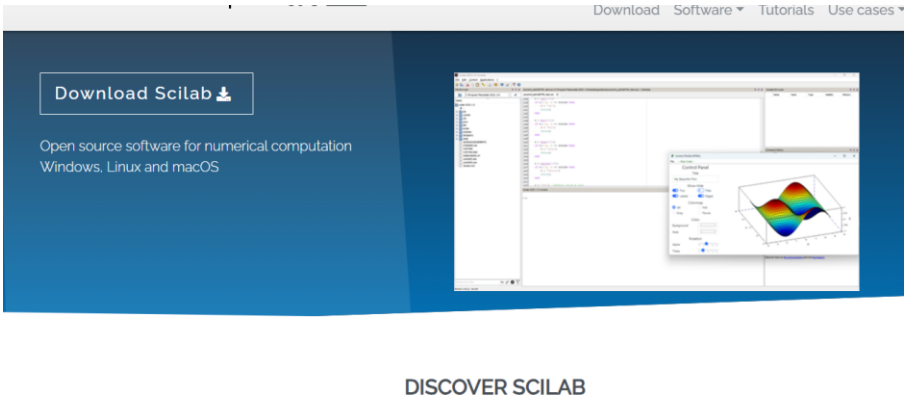

Рисунок 2.3 Сайт Scilab

Ще одним навчально-дослідним середовищем є GNU Octave, що підтримує повну сумісність із MATLAB-синтаксисом і широко застосовується у викладанні дисциплін, пов’язаних з обчислювальною математикою. Його оптимізаційний пакет optim реалізує методи дихотомії, золотого перетину, Бройдена-Флетчера-Ґольдфарба–Шанно (BFGS), методи обмеженої оптимізації

(L-BFGS) та алгоритми для нелінійних систем.

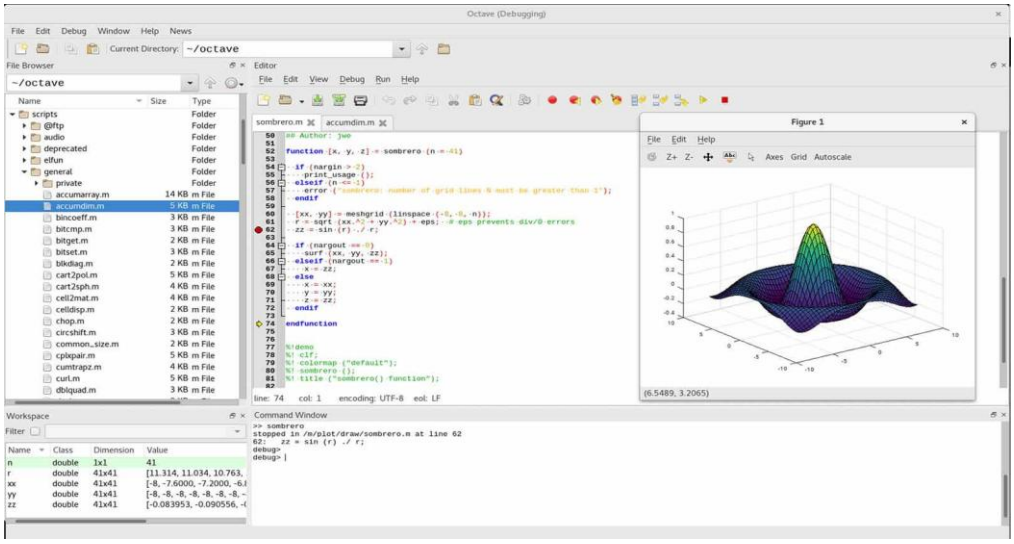

Рисунок 2.4. Інтерфейс GNU Octave

Сучасна тенденція у розробці програмних комплексів полягає у переході від монолітних середовищ до використання модульних бібліотек, що забезпечують гнучкість інтеграції в різні програмні продукти. Особливу роль тут відіграють бібліотеки для мов Python та C++, які активно використовуються як у науці, так і в промисловості.

Для Python найбільш відомими є бібліотеки SciPy та NumPy, які містять потужний набір алгоритмів оптимізації. Модуль scipy.optimize реалізує широкий спектр методів, від класичних (метод Ньютона-Квайда, Брента, Левенберга–Марквардта) до глобальних оптимізаторів (диференціальна еволюція, метод рою частинок). Ці бібліотеки відзначаються відкритістю, зручністю виклику функцій і можливістю комбінування чисельних методів з елементами машинного навчання.

Іншим сучасним інструментом є бібліотека Pyomo (Python Optimization Modeling Objects), яка поєднує математичне моделювання з вирішенням задач лінійної, цілочислової та нелінійної оптимізації. Pyomo дозволяє описувати задачі високого рівня абстракції у вигляді моделей, а для обчислень використовує зовнішні оптимізатори, такі як IPOPT, GLPK, CBC або Gurobi.

Завдяки відкритій структурі Pyomo активно застосовується у навчальних курсах з дослідження операцій та інженерного моделювання.

У середовищі C++ популярними є бібліотеки NLopt та COIN-OR (Computational Infrastructure for Operations Research). NLopt надає уніфікований інтерфейс до понад 25 алгоритмів оптимізації, зокрема градієнтних, еволюційних та стохастичних. Вона підтримує як детерміновані, так і випадкові методи, серед яких COBYLA, Nelder–Mead, CRS та StoGO. Бібліотека COIN-OR, у свою чергу, об’єднує набір проєктів із відкритим кодом, які реалізують лінійні, цілочислові й комбінаторні методи, включаючи оптимізатор CLP та Cbc. Обидві бібліотеки активно використовуються у наукових дослідженнях та освітніх проєктах, спрямованих на моделювання задач великих розмірів.

З поширенням технологій машинного навчання та штучного інтелекту чисельна оптимізація стала базовим елементом алгоритмів навчання. У цьому контексті ключову роль відіграють TensorFlow, PyTorch, JAX та Keras, які містять інтегровані оптимізаційні механізми для мінімізації функцій втрат у нейронних мережах.

TensorFlow Optimizer API включає реалізації стохастичного градієнтного спуску (SGD), Adam, RMSprop, Adagrad та інших методів, що дозволяють автоматично керувати швидкістю навчання, моментом інерції й адаптацією градієнтів. Аналогічно, у PyTorch реалізовано модуль torch.optim, який підтримує як базові оптимізатори, так і користувацькі варіації, що забезпечують гнучкість для дослідників і викладачів.

Іншим перспективним напрямом є JAX - бібліотека від Google, що забезпечує автоматичне диференціювання й високу продуктивність на GPU. Вона дозволяє експериментувати з власними оптимізаторами, що робить її зручним інструментом для навчання сучасних методів оптимізації в контексті глибинного навчання.

Розвиток хмарних обчислень сприяв появі платформ, які дозволяють проводити оптимізаційні обчислення безпосередньо у браузері. Серед таких

рішень варто відзначити MATLAB Online, Wolfram Cloud, Google Colab та Microsoft Azure Notebooks, які забезпечують віддалене виконання коду, візуалізацію процесу оптимізації та спільну роботу студентів. Такі сервіси інтегруються з хмарними сховищами, дозволяючи зберігати результати розрахунків та створювати інтерактивні навчальні середовища.

Серед спеціалізованих хмарних платформ для дослідження методів оптимізації вирізняється NEOS Server for Optimization, який надає веб-інтерфейс до понад 60 оптимізаційних алгоритмів і доступ до високопродуктивних обчислювальних ресурсів (Czyzyk et al., 1998). Користувач може формулювати задачу у форматі AMPL або GAMS, надсилати її через веб-інтерфейс і отримувати результати без необхідності встановлення програмного забезпечення. NEOS використовується в університетах для проведення лабораторних занять із дослідження операцій.

У сфері навчання важливе місце посідають інтерактивні візуальні середовища, орієнтовані на демонстрацію роботи алгоритмів [10, 12]. Поширеними прикладами є GeoGebra, що має вбудовані інструменти для побудови графіків функцій і пошуку мінімумів, та Desmos Graphing Calculator, який дозволяє візуалізувати криві та експериментально досліджувати властивості функцій.

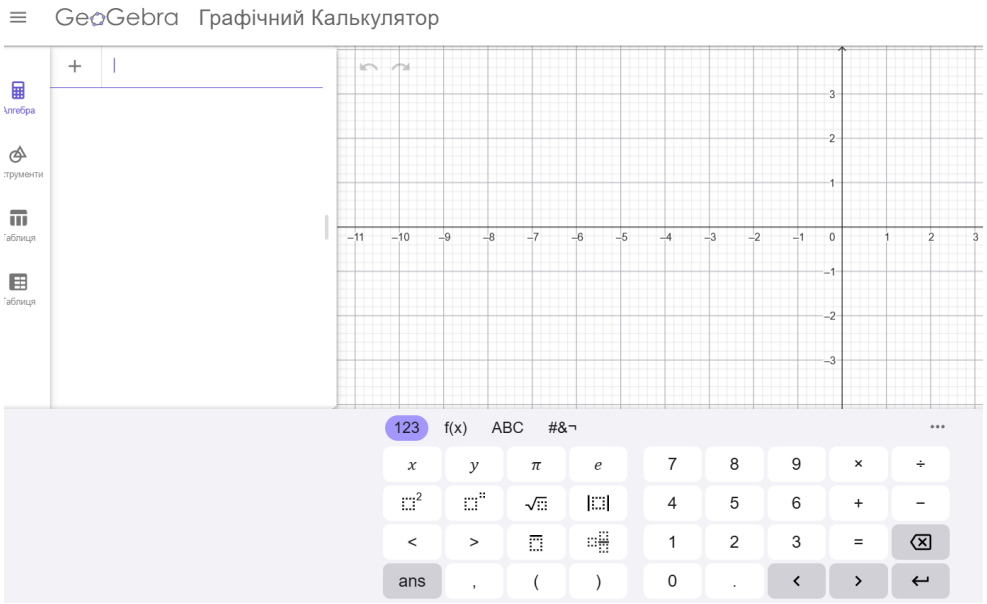

Рисунок 2.5. Інтерфейс GeoGebra

У вищих навчальних закладах активно застосовуються також симуляційні середовища, такі як Simulink, де оптимізаційні блоки використовуються для налаштування систем керування, а також платформи типу OptiY та Altair HyperStudy, що дозволяють проводити параметричні дослідження та оптимізацію інженерних конструкцій.

У навчальних курсах з комп'ютерних наук і прикладної математики особливої популярності набувають Jupyter Notebook та Google Colab, які дозволяють поєднувати програмний код, пояснення, формули та графіки у єдиному документі. Такий підхід сприяє активному навчанню, коли студент може одразу спостерігати, як зміна параметрів функції впливає на результат оптимізації.

Сучасні системи чисельної оптимізації представлені широким спектром інструментів, від академічних середовищ до промислових платформ і хмарних сервісів. Їх спільною рисою є прагнення забезпечити не лише обчислювальну ефективність, але й інтуїтивну взаємодію користувача з алгоритмом, можливість візуалізації та адаптацію під навчальні потреби. Класичні середовища на кшталт MATLAB, Mathematica та Scilab залишаються незамінними для фундаментальної підготовки, тоді як Python-орієнтовані бібліотеки, TensorFlow і PyTorch формують новий етап еволюції, інтеграцію оптимізаційних методів у системи штучного інтелекту. Розвиток хмарних сервісів і візуальних веб-платформ робить ці технології ще доступнішими для навчального процесу, відкриваючи перспективи створення інтерактивних освітніх комплексів нового покоління.

Попри значну кількість уже наявних програмних рішень, більшість із них мають низку обмежень, що ускладнюють їх повноцінне використання у навчальному процесі. Багато з них потребують підключення до Інтернету, мають високу вартість ліцензій або надмірні системні вимоги, що не завжди

відповідає можливостям студентських комп’ютерних лабораторій. Окрім того, складність інтерфейсу та перенасиченість функціоналом часто створюють бар’єр для початківців, які лише розпочинають ознайомлення з методами оптимізації.

У межах цього дослідження планується розроблення незалежного десктопного програмного комплексу, який не потребує підключення до мережі, має невеликий обсяг, мінімальні системні вимоги та простий інтерфейс, орієнтований на навчальне використання. Такий підхід дозволяє забезпечити автономність роботи, високу швидкодію та стабільність навіть на комп’ютерах із обмеженими ресурсами.

Архітектура майбутнього застосунку базуватиметься на модульному принципі, де основні компоненти: обробка даних, реалізація алгоритмів оптимізації та збереження результатів, відокремлені один від одного. Для розроблення передбачається використання мови програмування C# та середовища Microsoft Visual Studio, що забезпечують зручну інтеграцію обчислювальних і графічних компонентів. Зберігання даних у форматі XML гарантує прозорість, сумісність і легкість у модифікації навчальних прикладів.

Серед основних технічних переваг запропонованого рішення - швидке завантаження даних, локальна обробка без зовнішніх залежностей, повна відсутність необхідності в API-ключах або авторизації, а також можливість використання у дистанційних курсах. Завдяки автономній структурі додаток може бути інтегрований у систему електронного навчання (наприклад, Moodle або LMS університету) як допоміжний інструмент для проведення лабораторних робіт. Важливою перевагою є відкритість і розширюваність архітектури, яка дозволяє додавати нові методи оптимізації без зміни основного коду. Початково реалізований метод дихотомії стане базовим прикладом чисельного пошуку мінімуму функції, а в подальшому система може бути доповнена іншими методами, наприклад, золотого перетину, Фібоначчі, Ньютона чи градієнтного спуску. Таким чином, створення автономного

навчального програмного комплексу для дослідження методів чисельної оптимізації дозволить усунути обмеження існуючих рішень, забезпечити доступність інструменту для студентів будь-якого рівня підготовки та зробить процес навчання більш інтерактивним, наочним і технологічно гнучким.

2.2. Порівняльний аналіз програмних засобів для моделювання оптимізаційних процесів

Моделювання оптимізаційних процесів посідає ключове місце у сучасних наукових дослідженнях, інженерних розрахунках, фінансовому аналізі та освітній діяльності. Розвиток комп'ютерних технологій сприяв появі значної кількості програмних засобів, які реалізують методи чисельної оптимізації різного рівня складності - від елементарних алгоритмів пошуку екстремуму до комплексних систем моделювання нелінійних, стохастичних та багатокритеріальних задач.

Сучасні інструменти мають різні підходи до реалізації процесу оптимізації, однак об'єднуються спільною метою - забезпечення точного, ефективного й наочного пошуку найкращого рішення в межах заданих параметрів. Їх аналіз є важливим етапом при виборі програмного середовища для навчання та досліджень, оскільки дозволяє виявити переваги, обмеження та потенціал розвитку кожного підходу.

Серед найвідоміших і найпоширеніших інструментів для чисельної оптимізації у світі варто виокремити такі системи, як MATLAB, Wolfram Mathematica, Scilab, GNU Octave, Pyomo, TensorFlow, PyTorch, COIN-OR та NEOS Server for Optimization. Кожна з них має власну архітектуру, набір методів та рівень орієнтованості на освітні, наукові чи інженерні завдання.

Серед традиційних програмних систем для моделювання оптимізаційних процесів домінує MATLAB, розроблений компанією MathWorks. MATLAB є

універсальним середовищем, що поєднує чисельні обчислення, графічну візуалізацію та моделювання. Його компонент Optimization Toolbox включає широкий набір методів, серед яких методи Ньютона, спряжених градієнтів, квазі-Ньютона, симульованого відпалу, генетичні алгоритми та метод рою частинок. MATLAB підтримує оптимізацію як без обмежень, так і з обмеженнями, у тому числі лінійними та нелінійними. Для навчальних цілей MATLAB зручний завдяки інтуїтивному синтаксису, модульності та широким можливостям візуалізації. Крім того, MATLAB інтегрується із середовищем Simulink, що дозволяє моделювати складні динамічні системи з елементами оптимізації параметрів. Недоліком є комерційна ліцензія та висока вартість, що обмежує його використання у навчальних закладах з невеликим фінансуванням.

Wolfram Mathematica пропонує інший підхід - комбінацію символьних і чисельних методів. Система дозволяє проводити аналітичне дослідження функцій, знаходити критичні точки та здійснювати чисельне уточнення результатів. Функції NMinimize і FindMinimum реалізують автоматичний вибір алгоритму залежно від властивостей функції. Перевагою Mathematica є гнучкість у роботі з аналітичними виразами, можливість використання методу множників Лагранжа, комбінування градієнтних і безградієнтних методів у єдиному середовищі. Як і MATLAB, Mathematica відзначається високою точністю обчислень, проте її продуктивність у задачах великої розмірності є нижчою через універсальність символьного ядра.

Відкриті аналоги MATLAB - Scilab та GNU Octave - займають особливе місце в освітньому середовищі. Scilab - це безкоштовне середовище для технічних і наукових обчислень, яке має модулі Optimization та Global Optimization для розв’язання задач із використанням методів градієнтного спуску, Ньютона, квазі-Ньютона та стохастичних алгоритмів. Scilab підтримує інтерактивну візуалізацію, що дозволяє демонструвати процес збіжності алгоритмів. Його відкритий код робить систему придатною для навчальних лабораторних робіт, а простота інтерфейсу забезпечує швидке опанування

студентами базових методів оптимізації.

GNU Octave також сумісний із MATLAB-синтаксисом і містить набір функцій для знаходження мінімумів функцій та розв’язання задач оптимізації з обмеженнями. Він підтримує методи Брента, Бройдена–Флетчера–Ґольдфарба–Шанно (BFGS) та стохастичні методи. Octave застосовується переважно в університетських курсах як безкоштовна альтернатива MATLAB, проте має обмежені можливості графічного інтерфейсу та нижчу швидкодію при розв’язанні великих систем.

Завдяки поширенню мови Python чисельна оптимізація отримала новий імпульс розвитку. Найвідомішими бібліотеками є SciPy, Pyomo та CVXPY. Модуль scipy.optimize реалізує широкий спектр класичних алгоритмів, включаючи метод Ньютона, Левенберга–Марквардта, диференціальну еволюцію та метод рою частинок. Ці функції забезпечують високу точність при мінімальних вимогах до обчислювальних ресурсів.

Pyomo (Python Optimization Modeling Objects) використовується для математичного моделювання задач лінійної, нелінійної, цілочислової та стохастичної оптимізації. Його архітектура дозволяє відокремити формулювання задачі від вибору алгоритму, що є перевагою в навчальних і дослідницьких проєктах. Pyomo підтримує інтеграцію з відкритими та комерційними оптимізаторами (GLPK, CBC, IPOPT, Gurobi) та активно використовується в освітніх курсах зі спеціальностей «Дослідження операцій» і «Інтелектуальні системи керування».

CVXPY орієнтована на задачі опуклої оптимізації. Її висока продуктивність досягається завдяки внутрішнім оптимізаторам ECOS і SCS. Перевагою бібліотек Python є відкритість, кросплатформність і можливість інтеграції з машинним навчанням, однак недоліком залишається відсутність єдиного графічного інтерфейсу, що ускладнює використання в навчальних демонстраціях.

Оптимізація лежить в основі алгоритмів навчання глибинних нейронних

мереж, тому серед сучасних програмних інструментів особливе місце посідають TensorFlow, PyTorch і JAX. Вони реалізують адаптивні стохастичні методи, зокрема градієнтний спуск, Adam, RMSprop, Adagrad, які забезпечують ефективну мінімізацію функції втрат. TensorFlow має модуль tf.keras.optimizers, який містить високорівневі класи для керування параметрами навчання, тоді як PyTorch надає більш гнучкий інтерфейс для експериментів і наукових досліджень. JAX, розроблений Google, поєднує автоматичне диференціювання з оптимізованими обчисленнями на GPU, що робить його одним із найперспективніших інструментів для симуляцій оптимізаційних процесів.

AI-орієнтовані системи не лише виконують чисельні обчислення, але й моделюють складні нелінійні взаємозв’язки, що надає змогу вивчати властивості алгоритмів у контексті реальних процесів навчання. Проте вони потребують значних обчислювальних ресурсів, що обмежує їх використання у навчальному процесі в автономному режимі.

Поряд із локальними рішеннями активно розвиваються хмарні платформи. Найвідомішою серед них є NEOS Server for Optimization, створена в університеті Вісконсин-Медісон (США). NEOS надає веб-інтерфейс для розв’язання задач лінійної, нелінійної, комбінаторної оптимізації за допомогою понад 60 алгоритмів. Користувач може завантажити модель у форматі AMPL або GAMS, обрати оптимізатор і отримати результати через електронну пошту. NEOS активно використовується у дистанційних курсах і наукових лабораторіях, проте для його роботи необхідне стабільне підключення до Інтернету.

Іншим прикладом є платформа Google Colab, яка дозволяє запускати Python-бібліотеки для оптимізації без локальної інсталяції. Colab підтримує GPU, має доступ до бібліотек TensorFlow, PyTorch, SciPy та CVXPY. Це робить її зручною для навчання та лабораторних демонстрацій, хоча залежність від хмарної інфраструктури може бути недоліком для курсів, що потребують офлайн-доступу.

Порівняльний аналіз показує, що сучасні програмні засоби для моделювання оптимізаційних процесів значно відрізняються за призначенням, складністю, вартістю та вимогами до середовища виконання. Комерційні системи, такі як MATLAB і Mathematica, відзначаються високою точністю, широким спектром методів і можливостями візуалізації, однак їх використання обмежене ліцензійною політикою. Відкриті середовища Scilab і Octave забезпечують доступність і простоту, проте мають меншу швидкодію та вужчий набір алгоритмів. Python-бібліотеки забезпечують гнучкість і розширюваність, але не мають зручного графічного інтерфейсу для навчальних демонстрацій. Хмарні сервіси, такі як NEOS або Colab, відкривають широкі можливості для колективної роботи, проте залежать від мережевого доступу.

З огляду на це, доцільним є створення автономного програмного комплексу, який поєднує простоту використання, відсутність зовнішніх залежностей і доступність для навчального середовища. Розробка власного десктопного застосунку на мові C#, що реалізує методи чисельної оптимізації, зокрема метод дихотомії, забезпечить швидкодію, мінімальні системні вимоги та можливість роботи без Інтернету. Такий підхід сприятиме інтеграції програмного продукту у дистанційні курси з дисциплін «Методи оптимізації» та «Чисельні методи» й дозволить студентам виконувати практичні завдання автономно, використовуючи інтуїтивно зрозумілий інтерфейс.

2.3. Аналіз інтерфейсів та дидактичних можливостей подібних систем

Розвиток комп’ютерних технологій спричинив глибоку трансформацію підходів до навчання математичних дисциплін, зокрема до викладання методів чисельної оптимізації. Програмні системи, що реалізують алгоритми оптимізації, дедалі частіше інтегруються в навчальний процес як дидактичні

інструменти, що поєднують аналітичне моделювання з візуальним відображенням результатів. Одним із визначальних чинників їх ефективності є якість інтерфейсу користувача, адже саме він визначає рівень доступності, зручності та інтуїтивності взаємодії з програмою.

Інтерфейси сучасних систем оптимізації можна поділити на три основні типи: командно-орієнтовані, графічні (GUI) та веб-інтерфейси. Командно-орієнтовані середовища, такі як MATLAB, GNU Octave чи Python-бібліотеки SciPy та Pyomo, надають користувачеві потужні можливості контролю процесу, проте потребують певної підготовки. Їх основна перевага полягає у гнучкості, тобто користувач може формулювати власні моделі, змінювати параметри алгоритмів, комбінувати методи та виконувати серію експериментів. Водночас недоліком є складність сприйняття для початківців, що знижує дидактичну цінність у базовому курсі, де важливо не лише отримати результат, а й зрозуміти логіку роботи методу.

Графічні інтерфейси користувача (GUI) у системах типу MATLAB App Designer, Scilab Xcos, Wolfram Mathematica або Maple спрямовані на інтерактивність і наочність. Вони дозволяють студентам спостерігати за динамікою збіжності алгоритмів, зміною графіка функції при кожній ітерації, положенням точки мінімуму тощо. Такі інтерфейси підвищують рівень мотивації студентів, оскільки забезпечують миттєвий зворотний зв’язок між параметрами моделі та її результатом. Дослідження у сфері дидактики обчислювальної математики показують, що візуалізація процесів оптимізації сприяє формуванню глибшого розуміння принципів збіжності та поведінки алгоритмів у різних умовах [7].

Окрему групу становлять веб-інтерфейси для моделювання оптимізаційних процесів. Платформи на кшталт NEOS Server for Optimization, Google Colab або MATLAB Online забезпечують дистанційний доступ до обчислювальних ресурсів і готових оптимізаторів через браузер. Їх перевага полягає у відсутності необхідності встановлення програмного забезпечення, що

зручно для дистанційного навчання. Проте залежність від стабільного інтернет-з'єднання та відсутність локальної автономності створюють обмеження у використанні таких систем в умовах офлайн-навчання або при обмежених технічних можливостях.

У контексті використання в навчальному процесі, важливими характеристиками навчальних систем оптимізації є наочність, інтерактивність, адаптивність та доступність. Наочність досягається завдяки візуалізації геометричного сенсу алгоритму, а саме відображенню еволюції точок пошуку, ліній рівня, контурів функції та траєкторії збіжності. Інтерактивність полягає у можливості змінювати початкові параметри задачі, переглядати проміжні результати, порівнювати швидкість збіжності різних методів. Адаптивність проявляється у тому, що система може підлаштовувати рівень складності задач або деталізацію виведення під підготовку користувача. Нарешті, доступність визначається простотою інтерфейсу, відсутністю потреби в спеціальних технічних знаннях і можливістю працювати без додаткових бібліотек чи підключення до серверів.

Аналіз сучасних навчальних платформ показує, що найефективнішими з точки зору дидактики є системи, які поєднують інтерактивну візуалізацію з прямою маніпуляцією параметрами алгоритмів. Наприклад, у MATLAB існують навчальні демонстраційні модулі, де студент може пересувати повзунок, що визначає крок пошуку, і в реальному часі спостерігати, як змінюється швидкість збіжності методу градієнтного спуску. Подібні рішення реалізовані у Scilab Xcos, який дозволяє будувати блок-схеми оптимізаційних процесів. У Mathematica користувач може створювати інтерактивні панелі керування з використанням функцій Manipulate і Dynamic, що підвищує глибину експериментального дослідження.

Веб-платформи, такі як GeoGebra або Desmos, попри своє спочатку математично-графічне призначення, також використовуються для візуалізації процесів оптимізації, наприклад, демонстрації збіжності функцій до мінімуму

або ілюстрації дії методу дихотомії. Їх перевагою є легкість використання і доступність у браузері, однак обмеженням залишається неможливість моделювання складних багатовимірних задач або підключення до зовнішніх алгоритмів.

Важливим дидактичним аспектом сучасних систем є відкритість програмної архітектури. У середовищах із відкритим кодом (Scilab, GNU Octave, Python) студенти можуть не лише користуватися готовими функціями, а й модифікувати їх, спостерігаючи вплив змін на результат. Це сприяє розвитку аналітичного мислення та навичок програмування. На відміну від цього, комерційні системи часто є «чорними скриньками», у яких внутрішня логіка алгоритму недоступна, що знижує дидактичний потенціал у контексті навчання принципам реалізації методів оптимізації.

Разом із тим більшість розглянутих систем мають певні організаційні та технічні обмеження при використанні у навчальних закладах. Не всі з них забезпечують повну автономність, працюють офлайн або мають україномовний інтерфейс. Для проведення лабораторних занять у закритому середовищі без підключення до Інтернету виникає потреба у незалежних застосунках, які могли б забезпечувати базові обчислення, візуалізацію функцій і демонстрацію роботи алгоритмів без залучення зовнішніх серверів або хмарних сервісів.

У цьому контексті особливої актуальності набуває розроблення автономного навчального застосунку, який поєднує простоту використання, наочність і стабільну роботу без підключення до мережі. На відміну від хмарних або скриптових рішень, такий інструмент може бути використаний у комп’ютерних лабораторіях і під час дистанційного навчання у системах Moodle чи el.puet.edu.ua як допоміжний програмний засіб.

Розроблюваний у межах даного дослідження програмний комплекс передбачає саме таку модель взаємодії. Його інтерфейс орієнтований на інтуїтивну зрозумілість: користувач має змогу обирати метод оптимізації, вводити параметри задачі, спостерігати процес пошуку мінімуму у графічній

формі та зберігати результати у форматі XML для подальшого аналізу. Завдяки відсутності потреби у зовнішніх бібліотеках і використанню мови програмування C# забезпечується стабільність, швидкість і сумісність із будь-якою навчальною платформою.

Таким чином, аналіз інтерфейсних і дидактичних особливостей наявних систем показує, що, незважаючи на широкий вибір програмних засобів для оптимізації, більшість з них залишаються або надто складними для початкового рівня, або вимагають зовнішньої інфраструктури. Розробка автономного десктопного інструменту дозволяє поєднати найкращі риси сучасних підходів, а саме: візуалізацію, простоту, гнучкість і доступність, створюючи при цьому оптимальне середовище для навчання методам чисельної оптимізації у закладах вищої освіти.

2.4. Визначення недоліків існуючих рішень і шляхів їх усунення

Проведений аналіз сучасних програмних засобів для моделювання оптимізаційних процесів дозволяє зробити висновок, що, попри велику кількість існуючих систем, їх застосування у навчальному процесі має низку обмежень як технічного, так і методичного характеру. Більшість популярних рішень орієнтовані на професійне або промислове використання, а не на дидактичні потреби. Це призводить до того, що студенти, які тільки починають знайомство з методами чисельної оптимізації, стикаються з надмірною складністю інтерфейсу, високими системними вимогами або необхідністю спеціальних знань програмування. Такі фактори знижують ефективність навчання, оскільки увага зміщується з осмислення алгоритмічної сутності методів на технічні аспекти роботи із середовищем.

Одним із найбільш поширених недоліків сучасних програмних систем є

надмірна універсальність і складність структури. Системи MATLAB, Wolfram Mathematica чи Maple містять тисячі вбудованих функцій, які охоплюють усі напрями прикладної математики. З одного боку, це забезпечує широкі можливості для досліджень, але з іншого - ускладнює навчальний процес на початковому етапі. Користувачеві доводиться опановувати значний обсяг синтаксичних конструкцій, налаштувань і команд, перш ніж він зможе виконати навіть базову задачу пошуку мінімуму функції. Крім того, комерційна ліцензія таких середовищ є суттєвим обмеженням для навчальних закладів, оскільки потребує додаткових фінансових витрат.

Іншим важливим недоліком є відсутність повної автономності. Більшість сучасних рішень орієнтовані на хмарну архітектуру або потребують постійного доступу до мережі Інтернет. Хмарні сервіси, такі як NEOS Server for Optimization чи Google Colab, безумовно, зручні для колективної роботи та дистанційного навчання, однак вони залежать від зовнішньої інфраструктури, мають обмеження на розмір завдань, потребують авторизації та стабільного з’єднання. В умовах університетських лабораторій або локальних навчальних мереж така залежність може стати критичною. Автономність - необхідна умова для використання програмного засобу у комп’ютерних класах, де доступ до Інтернету обмежений або контролюється адміністративно.

Суттєвою проблемою залишається високий поріг входження при використанні програм із командним інтерфейсом. У середовищах MATLAB, Python, GNU Octave або Scilab студент повинен самостійно формулювати рівняння, вказувати межі пошуку, ініціалізувати параметри та контролювати збіжність алгоритму через консольні команди. Це корисно для досвідчених користувачів, але створює психологічний бар’єр для студентів, які лише починають знайомство з чисельними методами. У результаті навчальний процес перетворюється на тренування синтаксису, а не на дослідження суті алгоритму.

Не менш важливим недоліком є недостатня візуалізація оптимізаційних

процесів. У більшості систем процес збіжності подається у вигляді числових таблиць або векторів значень. Графічні відображення часто є додатковими, а не базовими елементами, тому користувач не має змоги спостерігати динаміку руху точки пошуку до екстремуму в реальному часі. У навчальному контексті така візуалізація має величезне значення, оскільки саме вона дозволяє зрозуміти логіку методу та геометричну інтерпретацію його кроків. Навіть у системах з інтерактивними елементами, як-от Mathematica чи MATLAB Live Editor, користувачеві необхідно попередньо створювати графічні об’єкти або налаштовувати панелі керування, що знову ж таки ускладнює процес підготовки.

До технічних недоліків слід віднести високі вимоги до ресурсів системи. Сучасні універсальні середовища часто потребують значного обсягу оперативної пам’яті та потужного процесора, що ускладнює їх роботу на застарілих або малопотужних комп’ютерах. Це особливо відчутно під час проведення лабораторних занять у навчальних закладах, де технічне забезпечення може бути обмеженим. Використання інтерпретованих мов (як-от Python або MATLAB) додатково знижує швидкодію, особливо при багаторазових ітераційних обчисленнях.

Додатковим аспектом, який обмежує дидактичний потенціал існуючих систем, є закритість архітектури або складність модифікації коду. У комерційних середовищах більшість алгоритмів реалізовані у вигляді вбудованих функцій, до яких користувач не має доступу. Це унеможливлює глибоке вивчення внутрішньої логіки реалізації методу та перешкоджає розширенню функціоналу системи. Для навчальних цілей така ситуація є неприйнятною, адже студент повинен мати змогу не лише використовувати готові інструменти, а й аналізувати їхню структуру та змінювати параметри реалізації.

Із дидактичного погляду, у більшості програмних систем бракує адаптивності навчального середовища. Сучасні платформи рідко враховують

різний рівень підготовки користувачів, не мають інтерактивних підказок або коментарів, що пояснюють дії алгоритму. У результаті студент отримує результат без розуміння етапів його досягнення, що суперечить самим принципам компетентнісного підходу до навчання.

Ще одним чинником, що обмежує використання сучасних програмних рішень у навчанні, є мовний бар'єр. Більшість середовищ має англомовні інтерфейси та довідкові матеріали, що створює труднощі для студентів, які не володіють англійською на достатньому рівні. Відсутність україномовних інтерфейсів або адаптованих методичних матеріалів суттєво знижує ефективність використання таких програм у національному освітньому контексті.

Визначені недоліки дозволяють сформулювати основні напрями їх усунення, які мають бути враховані під час розроблення нового інтерактивного програмного комплексу для навчання методам чисельної оптимізації. Перш за все, необхідно забезпечити простоту інтерфейсу, тобто мінімальну кількість керуючих елементів, чітку логіку дій та наочне відображення результатів. Користувач повинен мати можливість легко завантажити дані, вибрати метод, задати параметри та відразу побачити процес пошуку екстремуму на графіку. Така структура не лише зменшує когнітивне навантаження, а й підвищує мотивацію до самостійної роботи.

Другим напрямом є забезпечення автономності системи. Розроблюваний комплекс має функціонувати без підключення до Інтернету, використовувати локальні бібліотеки та зберігати результати у стандартних форматах, придатних для подальшої обробки (наприклад, XML). Такий підхід гарантує стабільність, незалежність від зовнішніх серверів і можливість інтеграції у навчальні курси дистанційного або змішаного формату (наприклад, у середовище Moodle).

Третім аспектом є оптимізація продуктивності. Реалізація алгоритмів у вигляді ефективних модулів мовою C# дозволяє досягти високої швидкодії та стабільності, що є важливим для багаторазових ітераційних процесів. Мова C#

також забезпечує широкий вибір інструментів для створення графічного інтерфейсу користувача (Windows Forms або WPF), що робить можливим поєднання обчислювальних і візуальних компонентів у єдиній системі.

Четвертим напрямом удосконалення є візуалізація процесів оптимізації. У майбутньому застосунку доцільно реалізувати покрокове відображення ходу алгоритму: показ позиції точки пошуку, меж інтервалу, ліній рівня або графіка функції. Це дозволить не лише демонструвати роботу методу, а й проводити порівняння між різними алгоритмами за швидкістю збіжності чи точністю результату.

П’ятий напрям стосується відкритості архітектури. Програмна структура має бути побудована за модульним принципом, щоб студенти могли легко змінювати або доповнювати реалізацію методів. Наприклад, у межах однієї системи можна реалізувати метод дихотомії як базовий, а потім додати метод золотого перетину, Фібоначчі, Ньютона чи градієнтного спуску, не змінюючи основної логіки роботи програми.

Шостим напрямом удосконалення є підвищення дидактичного потенціалу шляхом створення навчально-орієнтованого інтерфейсу. Це можуть бути вбудовані підказки, пояснення до етапів алгоритму, повідомлення про збіжність або попередження про можливі похибки. Такий підхід дозволить перетворити програму не лише на інструмент для обчислень, а й на інтерактивного «викладача», який супроводжує студента під час виконання лабораторних завдань.

Важливим шляхом усунення недоліків також є зменшення апаратних вимог. За рахунок оптимізації коду та використання лише необхідних компонентів програмний комплекс має залишатися легким, швидким у запуску та придатним для використання навіть на застарілих комп’ютерах. Це особливо важливо у контексті масового впровадження програмного забезпечення у навчальних аудиторіях, де технічне оновлення відбувається нечасто.

Останнім, але не менш важливим напрямом є мовна локалізація.

Наявність україномовного інтерфейсу, підказок і довідкових матеріалів забезпечить зручність використання програми у національному освітньому середовищі. Це також сприятиме популяризації української наукової термінології у сфері прикладної математики та інформаційних технологій.

Підсумовуючи вищенаведене, можна зазначити, що сучасні програмні системи для моделювання оптимізаційних процесів мають значні функціональні можливості, проте не завжди відповідають потребам навчального процесу. Їхні основні недоліки - надмірна складність, відсутність автономності, висока вартість, закритість архітектури та недооцінка дидактичних аспектів - можуть бути усунуті шляхом створення незалежного навчального комплексу. Такий комплекс має поєднувати простоту використання, наочність, автономність, відкриту структуру та адаптивний навчальний потенціал. Реалізація цих принципів у межах майбутнього програмного продукту дозволить сформувати сучасний інструмент для підготовки фахівців з чисельних методів та оптимізації, який відповідатиме як освітнім, так і технологічним вимогам часу.

2.5. Обґрунтування необхідності розробки інтерактивного навчального комплексу

Аналіз сучасних тенденцій розвитку інформаційних технологій у сфері освіти показує, що інтерактивні навчальні програмні комплекси стають невід’ємним елементом підготовки фахівців технічних спеціальностей. Зокрема, у галузі чисельних методів та оптимізації такі системи забезпечують можливість практичного засвоєння теоретичних знань шляхом моделювання реальних процесів, що відбуваються під час пошуку екстремумів функцій. Однак, як показали результати попереднього аналізу, існуючі рішення не

повною мірою відповідають вимогам сучасного освітнього процесу, особливо у контексті доступності, автономності та дидактичної орієнтації.

Основною проблемою залишається розрив між складністю промислових програмних пакетів та потребами навчального процесу. Такі системи, як MATLAB, Mathematica, Scilab або Python-бібліотеки, хоча й мають потужний інструментарій, не завжди придатні для навчання студентів початкових курсів через складність інтерфейсу, потребу у програмуванні або відсутність інтерактивної візуалізації. Більшість із них вимагають значних обчислювальних ресурсів або постійного підключення до Інтернету, що обмежує їх застосування в умовах автономної роботи. Навчальні заклади, особливо регіональні, часто не мають змоги підтримувати такі середовища, тому виникає потреба у створенні компактних, простих і незалежних навчальних інструментів.

Інтерактивний навчальний комплекс для вивчення методів чисельної оптимізації повинен поєднувати у собі три ключові складові: інтуїтивний графічний інтерфейс, модульну архітектуру для реалізації алгоритмів і наочну візуалізацію процесу. Головна мета такого комплексу полягає не лише у виконанні обчислень, а й у створенні умов для осмисленого навчання, коли студент спостерігає, як саме алгоритм звужує інтервал пошуку, як змінюється положення точки мінімуму, і як параметри задачі впливають на швидкість збіжності. Таким чином, програмний засіб має стати не просто симулятором, а інтерактивним середовищем пізнання, яке поєднує навчання, експеримент і аналіз.

З технічного погляду, розробка автономного десктопного комплексу на мові програмування C# є оптимальним рішенням, оскільки забезпечує стабільність, швидкодію, сумісність із Windows-платформами та можливість створення зручного графічного інтерфейсу. Збереження даних у форматі XML гарантує прозорість і доступність навчальних прикладів, що може використовуватися як для індивідуальної, так і для групової роботи студентів. Використання відкритої архітектури дозволяє розширювати функціонал —

додавати нові методи оптимізації, змінювати алгоритми та параметри без суттєвої модифікації системи. Це робить комплекс гнучким інструментом, придатним для використання у різних навчальних дисциплінах.

Інтерактивність майбутнього програмного засобу має особливе дидактичне значення. Завдяки покроковій візуалізації процесу обчислення студент може усвідомити логіку методу, зрозуміти роль точності, вибору початкових умов і кількості ітерацій. Такий підхід відповідає сучасним тенденціям STEM-освіти, де акцент робиться на дослідницькому підході, самостійності мислення і здатності застосовувати знання для розв’язання практичних завдань. Крім того, автономна робота комплексу дозволяє використовувати його у середовищах дистанційного навчання, інтегруючи у платформи Moodle або el.puet.edu.ua як окремий навчальний модуль.

Створення власного інтерактивного навчального комплексу з методів чисельної оптимізації має не лише практичну, а й педагогічну цінність. Такий інструмент дозволить стандартизувати проведення лабораторних робіт, забезпечити студентів єдиним середовищем для експериментів і підвищити ефективність викладання завдяки наочності та інтерактивності. Він сприятиме формуванню компетентностей у галузі аналітичного мислення, моделювання, інтерпретації результатів та алгоритмічного підходу до розв’язання завдань.

Таким чином, розроблення інтерактивного навчального комплексу є необхідним кроком у напрямі модернізації процесу викладання чисельних методів і оптимізації. Новий програмний продукт забезпечить поєднання простоти використання, автономності, наочності та адаптивності, що дозволить ефективно реалізувати принципи компетентнісного підходу в освіті та створити сучасне цифрове навчальне середовище, орієнтоване на активне пізнання й експериментальне навчання.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Класифікація методів чисельної оптимізації

Методи чисельної оптимізації становлять сукупність алгоритмічних підходів, призначених для знаходження екстремальних значень цільової функції - мінімумів або максимумів - у випадках, коли аналітичне рішення є неможливим або практично недосяжним. Розгалужена система таких методів класифікується за кількістю змінних (одновимірні та багатовимірні), за використанням або відсутністю похідних (градієнтні та безградієнтні), за наявністю обмежень (умовні та безумовні) і за характером пошуку (детерміновані або стохастичні) [8]. Класифікація відіграє важливу роль у навчальному процесі, оскільки дозволяє продемонструвати студентам логіку розвитку методів від найпростіших до складних.

Одним із найпростіших і наочних є метод дихотомії, який базується на послідовному поділі інтервалу [a,b] на дві частини та відсіченні тієї, що не містить точки мінімуму. На кожному кроці інтервал звужується, наближаючи оцінку до істинного значення екстремуму. Простота реалізації робить метод особливо корисним для навчальних цілей, адже він дозволяє візуально простежити процес збіжності - від початкового інтервалу до точки мінімуму (див. рисунок 3.1). Його недоліками є потреба у великій кількості обчислень для досягнення високої точності та неможливість прямого застосування в багатовимірних задачах.

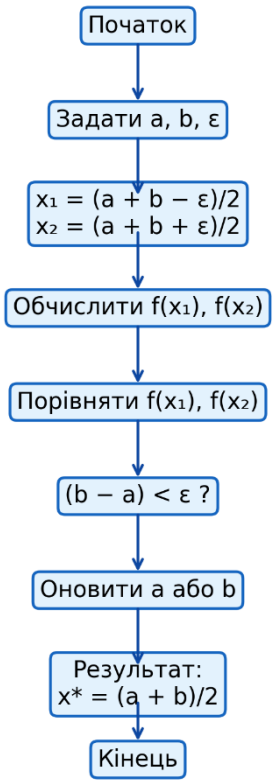

Рисунок 3.1. Блок схема методу дихотомії.

Близьким за логікою, але більш ефективним є метод золотого перетину, у якому розподіл інтервалу виконується з урахуванням коефіцієнта $\varphi = (1 + \sqrt{5})/2$. На відміну від дихотомії, цей метод повторно використовує одну з точок на кожній ітерації, що зменшує кількість обчислень функції. Завдяки цьому золотий перетин застосовується як базовий інструмент для демонстрації принципу оптимального використання обчислювальних ресурсів (див. рисунок 3.2). У навчальних курсах він показує, як невелика модифікація алгоритму може суттєво підвищити його ефективність.

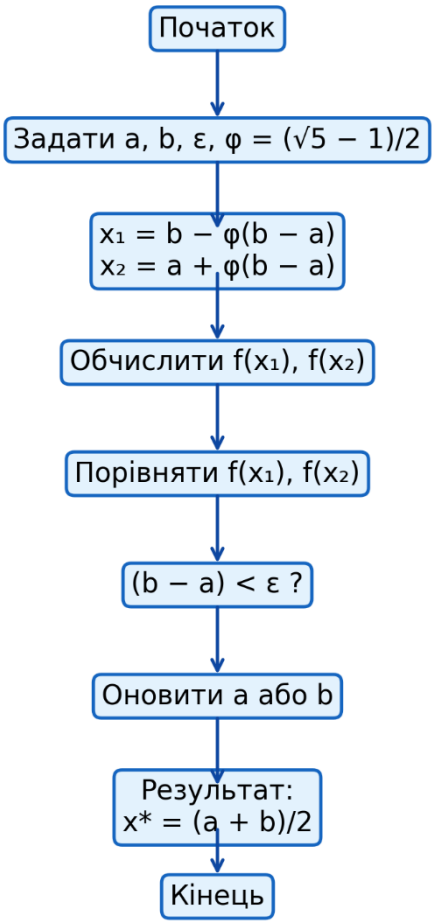

Рисунок 3.2. Блок схема «золотого перетину»

У багатовимірних задачах особливе місце посідають градієнтні методи, що ґрунтуються на використанні напрямку найшвидшого зменшення функції - градієнта ∇f(x). Класичний метод найшвидшого спуску виконує ітерації за формулою $x_{k+1} = x_k - \eta \nabla f(x_k)$, де η - довжина кроку [6]. У навчальному контексті ці методи дозволяють показати взаємозв’язок між аналітичною похідною та напрямом пошуку мінімуму (див. рисунок 3.3). Перевагою є універсальність, а недоліком — залежність від вибору початкової точки та швидкість збіжності, що сповільнюється у вузьких мінімальних долинах [4].

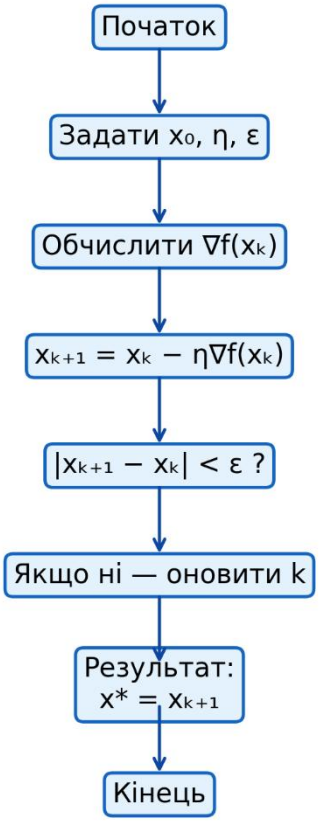

Рисунок 3.3. Блок схема градієнтного методу

До групи методів, які враховують кривизну поверхні цільової функції, належать методи Ньютона та квазі-Ньютона (BFGS, DFP). Метод Ньютона використовує обернену матрицю Гессіана $\nabla^2 f(x_k)$ та забезпечує квадратичну швидкість збіжності для гладких функцій [5]. Його практичне значення полягає в тому, що він демонструє переваги використання другої похідної для прискорення пошуку екстремуму, але одночасно вимагає значних обчислень і стабільного чисельного апарату (див. рисунок 3.4).

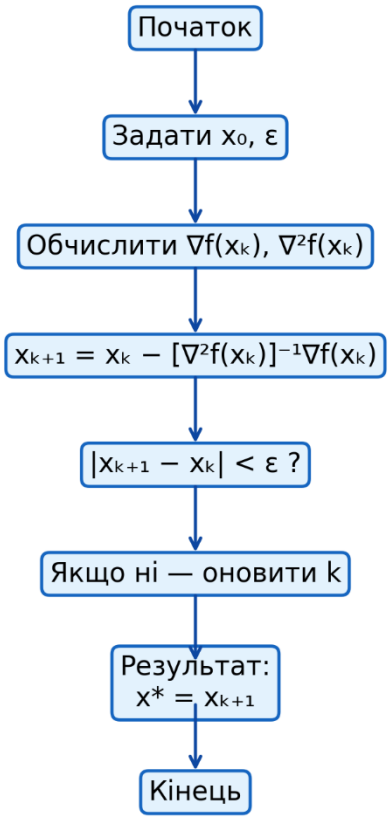

Рисунок 3.4. Блок схема методу Ньютона

Для задач, де аналітичні похідні недоступні або функція має стохастичний характер, використовуються еволюційні методи - генетичні алгоритми, метод рою частинок, імітаційне відпалу та інші. Вони ґрунтуються на принципах біологічної еволюції: популяція потенційних рішень піддається селекції, схрещенню та мутації, після чого обираються найкращі особини. Хоча такі підходи є менш передбачуваними, вони дозволяють долати обмеження локальної оптимізації. У навчальному процесі ці методи особливо корисні для демонстрації поняття глобального мінімуму, випадковості пошуку та впливу параметрів алгоритму на результат.

Окрему групу становлять безградієнтні методи прямого пошуку, зокрема метод Нелдера–Міда, що оперує симплексом точок і поступово трансформує його у напрямі покращення значення функції. Такі методи не потребують похідних і зручні для навчального використання, оскільки їх можна

візуалізувати у вигляді послідовності геометричних перетворень на площині. Вони демонструють студентам альтернативний підхід до пошуку екстремумів, коли інформація про похідні недоступна або обчислювально складна.

Класифікація методів чисельної оптимізації має і дидактичну цінність. Вона дозволяє формувати навчальні траєкторії, які починаються з інтервальних методів (дихотомічного та золотого перетину), переходять до градієнтних та другопохідних (методів Ньютона та BFGS), а завершуються стохастичними й еволюційними (генетичними, ройовими). Така послідовність забезпечує поступове ускладнення матеріалу й формує розуміння того, як алгоритми адаптуються до структури задачі, гладкості функції та наявності шуму.

Для майбутнього інтерактивного навчального комплексу така класифікація є методологічною основою структури системи. У базовій версії реалізовано дихотомічний метод, який поєднує простоту реалізації з високою наочністю. Надалі система може бути розширена реалізацією градієнтних і стохастичних підходів, що дозволить студентам проводити порівняльний аналіз точності та швидкості збіжності різних класів методів.

3.2. Математичне обгрунтування вираних методів оптимізації

Вибір конкретних методів чисельної оптимізації для реалізації в навчальному програмному комплексі ґрунтується на їх різній математичній сутності, можливості ілюструвати ключові аспекти пошуку екстремуму, а також на порівняльних властивостях збіжності, обчислювальної складності та наочності. У межах даної роботи акцент буде зроблено на інтервальних методах (зокрема, методу дихотомії), методах градієнтного типу, а також методі Ньютона. Для кожного з них нижче подано коротке формалізоване обґрунтування.

Метод дихотомії розглядає задачу мінімізації функції f(x) на інтервалі [a,b] за таких припущень: функція f неперервна і має єдиний локальний мінімум на цьому інтервалі. Алгоритм полягає у виборі двох точок $x_1 = (a + b - \varepsilon)/2$, $x_2 = (a + b + \varepsilon)/2$, і порівнянні значень $f(x_1)$ і $f(x_2)$. Якщо $f(x_1) < f(x_2)$, то область пошуку звужується до $[a, x_2]$, інакше до $[x_1, b]$. Таким чином, на кожній ітерації інтервал зменшується приблизно удвічі, і після k ітерацій довжина інтервалу становить (b − a)/2^k. Критерій зупинки використовує |b − a| < ε, що забезпечує наближення до оптимальної точки з бажаною точністю. Цей підхід є дуже наочним у навчальному контексті, оскільки демонструє звуження області пошуку, обмеженість ітераційного процесу та гарантію збіжності.

Метод градієнтного спуску базується на тому, що якщо $f : R^n \to R$ є диференційовною функцією і x_k - поточна ітерація, то можна обирати напрямок спуску як $-\nabla f(x_k)$. Ітерація має вигляд: $x_{k+1} = x_k - \eta_k \nabla f(x_k)$, де $\eta_k > 0$ - довжина кроку. При умові, що f має Lipschitz-неперервний градієнт із константою L, тобто $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, і вибрано $\eta_k \leq 1/L$, забезпечується монотонне зменшення значення функції $f(x_k)$. Зокрема, можна показати, що $f(x_{k+1}) \leq f(x_k) - (1/(2L))\|\nabla f(x_k)\|^2$. Це дає оцінку кількості ітерацій до досягнення критерію $\|\nabla f(x_k)\| \leq \varepsilon$ пропорційний $O(L/\varepsilon^2)$. У навчальному застосуванні метод градієнтного спуску дозволяє студентам експериментально впливати на параметр η_k, спостерігаючи швидкість збіжності та вплив початкових умов.

Метод Ньютона використовує інформацію про другу похідну (матрицю Гессіана) $H(x)=\nabla^2 f(x)$. Припускаючи, що f є двічі диференційовною і $H(x_k)$ невироджена, крок ітерації має форму $x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$. Для сильно опуклих функцій метод забезпечує квадратичну швидкість збіжності у локальній околі рішення: відстань до оптимуму зменшується приблизно як квадрат від попередньої. Проте обчислювальні витрати на кожну ітерацію більші - необхідно обчислити Гессіан та вирішити лінійну систему. У контексті

навчального комплексу реалізація методу Ньютона є доцільною як ілюстрація використання кривизни функції.

У контексті даного дослідження вибір саме цих трьох класів методів - інтервального (дихотомія), градієнтного та другого порядку (Ньютона) - обґрунтований поєднанням трьох критеріїв: дідактичної наочності, різного рівня складності реалізації та можливості побудови порівняльного аналізу. Інтервальні методи демонструють принцип звуження області пошуку, градієнтний - використання інформації першої похідної, а метод Ньютона - кривизни поверхні.

3.3. Алгоритмічні схеми методів та їх порівняльний аналіз за критеріями точності та швидкодії

Алгоритмічні схеми основних методів чисельної оптимізації - дихотомічного, градієнтного та методу Ньютона були детально наведені у попередньому пункті. На основі цих алгоритмів доцільно провести їх узагальнений порівняльний аналіз за двома ключовими критеріями: точністю (ступенем наближення до істинного екстремуму функції) та швидкодією (кількістю ітерацій і обчислювальною складністю процесу). Такий аналіз має не лише теоретичне, а й практичне значення для вибору оптимального методу при розв’язанні конкретних завдань оптимізації, а також для формування навчальних цілей при розробці програмного комплексу.

Метод дихотомії є базовим прикладом інтервального пошуку та має високу стабільність і передбачуваність. Його основною перевагою є простота реалізації й відсутність необхідності в обчисленні похідних, що робить його придатним для широкого кола неперервних, але не обов’язково диференційовних функцій. З іншого боку, метод характеризується низькою

швидкодією через велику кількість ітерацій, необхідних для досягнення заданої точності. Саме тому він найбільш підходить для навчального моделювання процесу звуження області пошуку, коли наочність має більшу вагу, ніж обчислювальна ефективність.

Градієнтний метод забезпечує істотно вищу швидкодію завдяки використанню інформації про напрямок зміни функції. У теоретичному плані він демонструє лінійну збіжність, однак якість результатів сильно залежить від вибору початкової точки та параметра кроку. При занадто великому кроці алгоритм може втратити стійкість, а при надто малому - збіжність стає повільною. Незважаючи на ці обмеження, метод градієнтного спуску залишається одним із найпоширеніших у прикладних задачах машинного навчання та аналізу даних, оскільки дозволяє швидко отримати наближене рішення для багатовимірних задач.

Метод Ньютона, у свою чергу, характеризується високою точністю та квадратичною швидкістю збіжності при наближенні до мінімуму. Завдяки використанню інформації про кривизну функції цей метод потребує значно менше ітерацій, проте кожна з них є обчислювально затратною. Крім того, для його стабільної роботи необхідно, щоб функція мала безперервні похідні другого порядку. Незважаючи на складність реалізації, метод Ньютона демонструє найвищу ефективність серед розглянутих підходів і тому є важливим орієнтиром при оцінці точності інших методів.

У таблиці 3.1 наведено узагальнені результати порівняння трьох розглянутих методів за критеріями точності, швидкодії, вимог до математичних властивостей функції, а також їхніх дидактичних переваг і недоліків.

Таблиця 3.1.

Порівняльний аналіз методів чисельної оптимізації

Метод	Необхідність похідних	Швидкодія	Точність	Складність реалізації	Дидактична цінність	Основні переваги	Основні недоліки
Дихотомії	Не	Низька	Висока для	Низька	Висока	Простота,	Велика

	потребує		одновимірних функцій			стабільність, універсальність для неперервних функцій	кількість ітерацій, непридатність до багатовимірних задач
Градієнтний	Перша похідна	Середня	Помірна, залежить від вибору кроку	Середня	Середня	Швидкість для опуклих функцій, інтуїтивна інтерпретація	Чутливість до початкових умов, ризик потрапляння в локальні мінімуми
Ньютона	Перша і друга похідні	Висока	Дуже висока при гладких функціях	Висока	Середня	Висока точність, мала кількість ітерацій	Високі обчислювальні витрати, складність реалізації

Як показує аналіз, метод дихотомії, реалізований у межах даної роботи, є найпридатнішим для освітніх цілей, оскільки демонструє базові принципи звуження області пошуку, не потребує складного математичного апарату та може бути інтегрований у навчальні курси з дисциплін «Методи оптимізації та дослідження оепрацій» або «Чисельні методи». Він також не потребує інтернет-з’єднання, має мінімальні вимоги до апаратних ресурсів і може бути використаний у складі дистанційних навчальних курсів як локальний інструмент для демонстрації процесів пошуку екстремумів.

У майбутніх версіях програмного комплексу доцільно передбачити додавання модулів реалізації градієнтного методу та методу Ньютона для розширення можливостей порівняння. Це дозволить студентам проводити експериментальні дослідження, оцінювати вплив параметрів алгоритму на швидкість і точність збіжності та глибше зрозуміти принципи роботи чисельних методів оптимізації.

3.4. Проєктування архітектури інтерактивного навчального комплексу та обґрунтування обраних іструментів розробки

Архітектура інтерактивного навчального комплексу для вивчення методів чисельної оптимізації була спроєктована відповідно до вимог наочності, простоти взаємодії користувача та модульності програмної реалізації. Метою архітектурного проєктування стало створення системи, яка забезпечує зручну взаємодію студента з алгоритмічними процесами пошуку екстремумів функцій, візуалізацію результатів і можливість подальшого збереження даних для повторного аналізу.

У розробленому застосунку було обрано середовище програмування C# з використанням технології Windows Presentation Foundation (WPF), що є частиною платформи .NET Framework. WPF надає розробнику можливість створювати інтерфейси з адаптивною роздільною здатністю, підтримкою векторної графіки, дво- і тривимірної візуалізації, а також широким набором елементів керування, стилів та шаблонів. Такий вибір обґрунтований необхідністю створення навчального інструменту, який би поєднував інтерактивність, візуальну привабливість та точність обчислень [3].

Архітектура програмного комплексу побудована за принципом модульного поділу функціональності. Основними структурними компонентами є:

1. Модуль обчислень, який реалізує чисельний алгоритм пошуку мінімуму методом дихотомії. Цей модуль відповідає за виконання розрахунків, обробку введених користувачем даних і формування результатів. Його центральним елементом є клас Dich, який містить метод FindMinimum(Start, End, Precision). У цьому класі реалізовано логіку звуження інтервалу, перевірку умов збіжності та збереження проміжних результатів.

Приклад фрагменту коду методу:

```
static Point FindMinimum(Container data)
{
   Dich dich = new Dich(data.poly, data.points);
   return dich.FindMinimum(data.Start, data.End, 0.0001);
}
```

Цей підхід дозволяє легко модифікувати алгоритм або додати інші методи оптимізації без зміни основної структури програми.

2. Модуль керування даними, який відповідає за імпорт, експорт і серіалізацію результатів роботи. Всі вхідні дані (поліном, список точок, діапазон пошуку) та результати мінімізації зберігаються у форматі XML, що забезпечує універсальність і зручність у використанні. Збереження результатів здійснюється за допомогою класу XmlSerializer, який створює файл із результатами пошуку, а також зберігає знімок поточного стану проєкту:

```
private static void SaveResultToFile(Point min, string path)
{
   XmlSerializer serializer = new XmlSerializer(typeof(Point));
   StreamWriter writer = new StreamWriter(path);
   serializer.Serialize(writer, min);
   writer.Close();
}
```

Крім того, система підтримує зворотне завантаження даних, користувач може імпортувати раніше збережений XML-файл для повторного аналізу чи продовження розрахунків.

3. Модуль користувацького інтерфейсу, реалізований у середовищі WPF з використанням мови розмітки XAML, надає студенту інтуїтивно зрозумілу взаємодію. Основне вікно програми містить поля для введення функції (полінома), набору точок і діапазону інтервалу.

Користувач послідовно вводить:

– формулу функції, наприклад $f(x)=x^2-4x+3$;

- список точок для побудови кривої;
- межі інтервалу пошуку.

Після цього натискається команда запуску розрахунку, і програма виконує ітераційний процес пошуку мінімуму.

Візуалізація результатів здійснюється у вигляді графічного відображення функції на інтервалі, що забезпечує наочне розуміння принципу збіжності методу.

Після запуску застосунку основний клас Program ініціалізує перевірку наявності робочого каталогу та файлів у ньому. Якщо користувач завантажує раніше створений проект, дані зчитуються у вигляді об’єкта класу Container, що містить усі необхідні параметри: формулу функції, точки та межі інтервалу. У разі введення нових даних користувач має змогу одразу зберегти їх для подальшого використання.

Логічна структура проєкту передбачає незалежність модулів, що дозволяє розширювати функціональність без порушення цілісності системи. Наприклад, у майбутніх версіях можна додати класи для реалізації градієнтного методу або методу Ньютона, які використовуватимуть ті самі інтерфейси даних, що й існуючий алгоритм дихотомії.

Важливим елементом архітектури є модуль візуалізації результатів, який реалізує побудову графіків у середовищі WPF. Це дозволяє студенту в реальному часі спостерігати процес звуження області пошуку та наближення до мінімуму функції. Після завершення обчислень користувач отримує можливість зберегти результати у форматах XML або HTML. HTML-звіт містить відформатовані дані обчислень, що може бути використано як складова лабораторної роботи або звіту про виконання практичного завдання.

Проєктована архітектура має низку важливих переваг:

- Модульність - кожен компонент системи (обчислення, введення/виведення, візуалізація) функціонує незалежно, що спрощує супровід і розширення.
- Простота навчального використання - інтерфейс є мінімалістичним і зрозумілим навіть для студентів початкових курсів.
- Автономність - система не потребує підключення до інтернету, що забезпечує можливість використання у локальному навчальному середовищі або на персональних пристроях без мережевої підтримки.
- Збереження даних - підтримка XML-формату дає змогу зберігати як результати, так і введені параметри для повторного аналізу.
- Можливість масштабування - наявна структура дозволяє у майбутньому реалізувати інші методи оптимізації (золотого перетину, градієнтні, Ньютона тощо) без суттєвої зміни архітектури.

Вибір мови програмування та середовища реалізації є одним із ключових етапів проєктування програмного забезпечення, оскільки саме ці чинники визначають продуктивність системи, можливості візуалізації та подальшу масштабованість. Для розробки інтерактивного навчального комплексу з чисельної оптимізації було обрано мову програмування C# та технологію Windows Presentation Foundation (WPF) у складі .NET Framework 4.8.

Мова C# належить до сучасних об’єктно-орієнтованих мов програмування, що забезпечують високий рівень абстракції, типобезпечність і структурованість коду. Вона поєднує можливості низькорівневого управління пам’яттю з простотою високорівневого синтаксису, що робить її придатною як для наукових обчислень, так і для створення складних графічних застосунків. Завдяки потужним інструментам стандартної бібліотеки .NET, C# дозволяє ефективно реалізовувати математичні алгоритми, забезпечуючи точність і стабільність обчислень. Крім того, C# має тісну інтеграцію з Windows API, що спрощує розробку десктопних застосунків з розвиненою графічною складовою.

Використання фреймворку WPF (Windows Presentation Foundation) обумовлено його перевагами у сфері побудови графічних інтерфейсів і візуалізації даних. WPF підтримує векторну графіку, що гарантує незалежність елементів інтерфейсу від роздільної здатності екрану та забезпечує чітке відображення на будь-яких пристроях. Його архітектура базується на мові розмітки XAML, яка дозволяє відокремити графічне представлення від логіки програми, що є ключовим принципом при побудові масштабованих систем за моделлю MVVM (Model–View–ViewModel) [3].

Серед основних технічних переваг WPF слід відзначити:

– можливість інтеграції з 2D- і 3D-графікою, що є важливим при візуалізації процесів оптимізації;

– підтримку анімації та шаблонів стилів, які сприяють створенню наочного та інтуїтивного інтерфейсу;

– механізм Data Binding, який дозволяє динамічно оновлювати візуальні елементи при зміні даних без необхідності ручного керування оновленнями;

– вбудовану підтримку мультимедіа, тексту, типографіки та документів, що розширює дидактичні можливості застосунку.

Окрім того, використання Visual Studio як середовища розробки забезпечило зручні засоби налагодження, аналізу продуктивності та тестування. Для зберігання й обміну даними між модулями застосовано серіалізацію у форматі XML, яка реалізована засобами стандартної бібліотеки .NET.

Таким чином, поєднання мови C# та технології WPF дозволило створити компактний, стабільний і візуально наочний навчальний комплекс, який об’єднує обчислювальний модуль, модуль даних та інтерфейс в єдину систему.

3.5. Побудова блок-схем, діаграм послідовності, компонентів і класів

Проєктування архітектури інтерактивного навчального комплексу передбачає не лише розробку програмного коду, а й побудову системи графічних моделей, які відображають логіку роботи алгоритмів, структуру програмних модулів і взаємодію між ними.

Першочергово були побудовані блок-схеми, які відображають покрокову логіку реалізації методу дихотомії та загальний цикл роботи програми. Алгоритмічна блок-схема методу дихотомії відтворює послідовність операцій, реалізовану в методі FindMinimum класу Dich: ініціалізація початкового інтервалу та параметра точності, обчислення двох внутрішніх точок, порівняння значень функції в цих точках, вибір підінтервалу, який містить мінімум, та повторення циклу до досягнення заданої точності. Ця блок-схема безпосередньо відповідає фрагменту коду, у якому створюється об’єкт Dich і викликається метод FindMinimum(data.Start, data.End, 0.0001) на основі даних, зчитаних із контейнера Container. Аналогічно, окрема блок-схема описує логіку роботи всього застосунку: запуск програми, перевірка наявності каталогу storage, зчитування XML-файлів, виконання пошуку мінімуму для кожного набору даних, виведення результатів у консоль та збереження результату у файл report.xml. Така візуалізація відображає реальний ланцюжок викликів методів OpenFile, FindMinimum і SaveResultToFile у класі Program і забезпечує наочний зв’язок між алгоритмічним описом і реалізацією в коді.

Наступним кроком стала розробка діаграм послідовності, які моделюють динаміку взаємодії між основними об’єктами системи в часі. На діаграмі послідовності сценарію «Пошук мінімуму» виділено такі учасники: користувач, головне вікно застосунку (інтерфейс WPF або консольний інтерфейс), модуль обробки даних, об’єкт Dich як реалізація алгоритму, а також підсистема збереження результатів. Згідно з описом у документі, користувач послідовно заповнює формулу полінома, список точок і діапазон інтервалу, після чого

ініціює пошук мінімуму, а система виводить результати та будує графік функції.

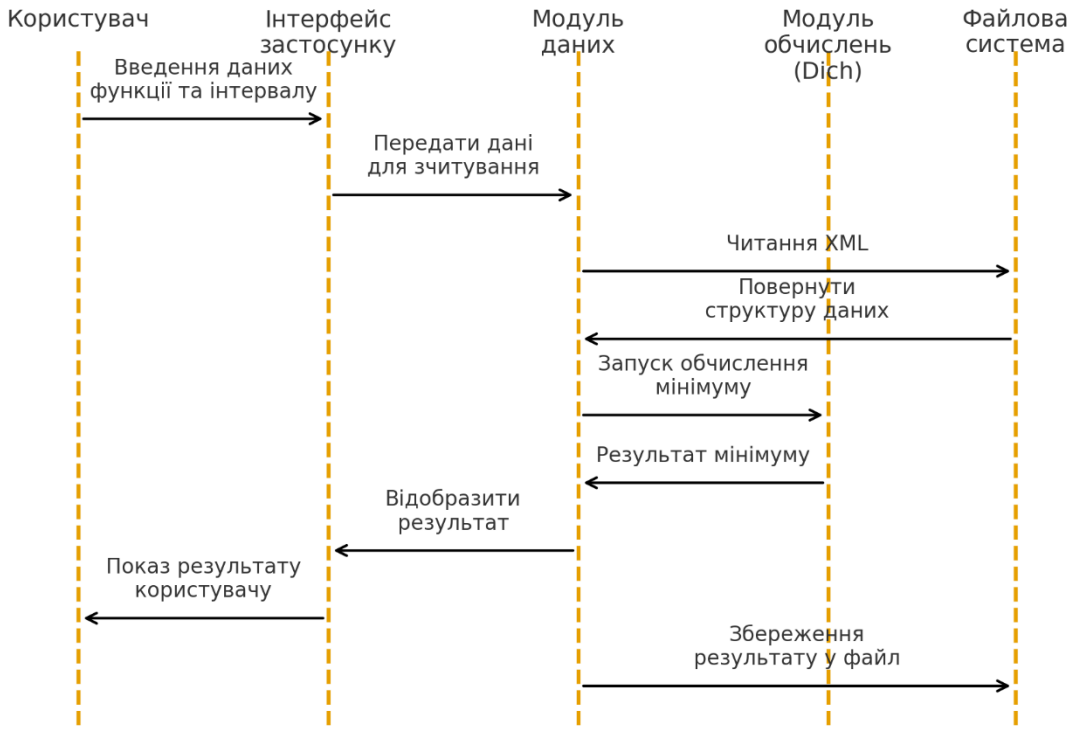

Рисунок 3.5. Діаграма послідовностей сценарію «Пошук мінімуму»

Цей сценарій відображено у вигляді послідовних повідомлень: від події введення даних у форму - до виклику методів читання, обчислення мінімуму та візуалізації результатів. Окрема діаграма послідовності моделює сценарій збереження результатів, де після завершення обчислень користувач через інтерфейс ініціює збереження звіту в HTML-форматі та «знімка» роботи програми у форматі XML, що також відображено в документі як один з етапів взаємодії з системою.

Таким чином, діаграми послідовності дозволяють формально описати часову структуру обміну повідомленнями між компонентами програми.

Важливу роль у відображенні структурної організації системи відіграє діаграма компонентів.

Рисунок 3.6. Діаграма компонентів

На ній комплекс представлено як набір логічно відокремлених компонентів: користувацький інтерфейс (WPF-застосунок або консольне вікно), бібліотека обчислень (модуль Library.dll, що містить класи Dich і Data), модуль керування даними (робота з XML та HTML через XmlSerializer та засоби введення/виведення), а також зовнішні ресурси — файлову систему, у якій зберігаються вхідні XML-файли та результати обчислень. Зв’язки між компонентами позначаються залежностями: інтерфейс взаємодіє з модулем обчислень через публічні методи бібліотеки, модуль обчислень очікує дані у вигляді структур Container і Point, а модуль доступу до даних забезпечує завантаження та збереження цих структур. Подібне подання відповідає описаній у документі системі класів і форм, де окремо розглядаються механізми розрахунку, введення даних і візуалізації результатів

Діаграма компонентів підкреслює модульність архітектури, а також можливість розширення - наприклад, через підключення додаткових компонентів для інших методів оптимізації.

На основі реальної реалізації було побудовано й діаграму класів, що є ключовою для розуміння внутрішньої структури програмного комплексу.

UML-діаграма бібліотеки Library, яка включає класи:

- AFunction<T> - абстрактний узагальнений клас із полем _container і методами Add, At, F, GetAll, RemoveAt тощо.
- Lagr - наслідує AFunction<Point>.
- Poly - наслідує AFunction<double>.
- Point - має властивості X, Y і методи ToString, конструктор.
- Dich - має поля _isWork, _lagrG, _polyF і методи FindMinimum, Stop, F.
- Data - з вкладеним класом Container, методами Read, Report, Write.

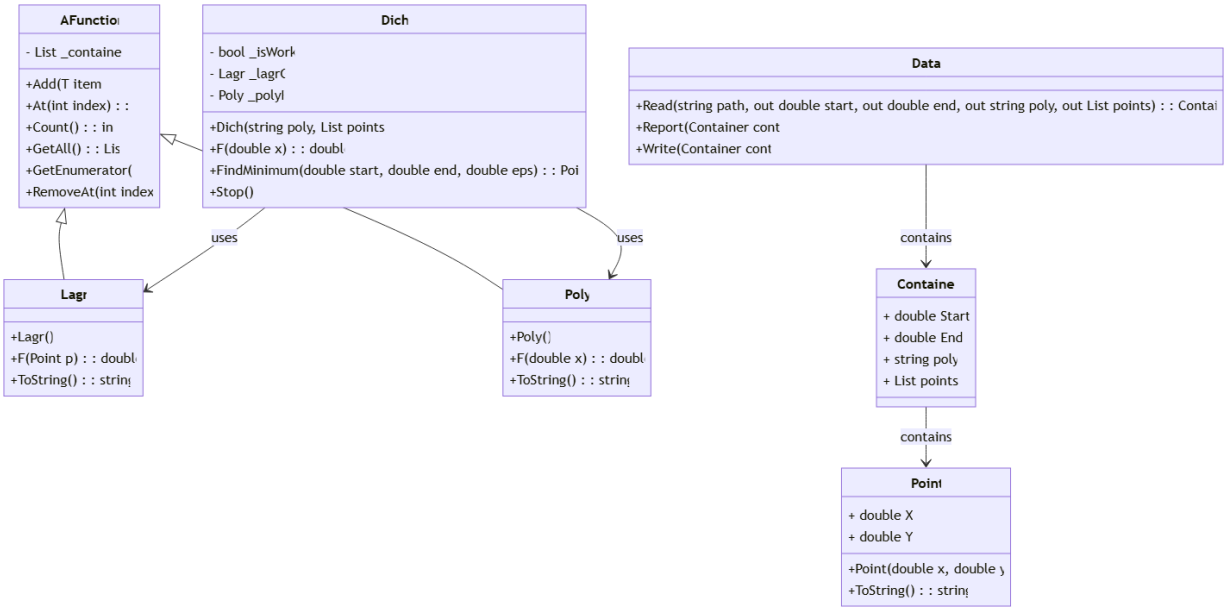

Рисунок 3.7. Діаграма класів розробленого застосунку

Центральне місце на діаграмі посідає клас Dich, який інкапсулює алгоритм методу дихотомії і має поля для зберігання інформації про поліном та список вузлових точок, а також метод FindMinimum, який реалізує ітераційний пошук мінімуму на заданому інтервалі. Клас Container використовується для збереження сукупності вхідних даних: меж інтервалу, полінома та списку точок, що відповідає його використанню в методі OpenFile, де через статичний клас Data здійснюється читання з XML-файлу та заповнення відповідних полів. Клас Point репрезентує окрему точку на графіку функції, що важливо як для

обчислювальної частини, так і для модулів візуалізації. Клас Program (у консольному варіанті) або головний клас вікна (у WPF-реалізації) виконує роль «керуючого» об’єкта, який ініціює зчитування даних, створює екземпляри Dich, запускає процес обчислення мінімуму та викликає методи збереження результатів. У діаграмі класів відображено асоціації між цими класами, зокрема зв’язок між Program і Container, між Container і Point, а також композиційні відносини, коли об’єкт Container містить колекцію точок. Така діаграма узгоджується з описом «системи класів», наведеної в документі, де зазначається, що задля проведення основних розрахунків було спроєктовано сукупність взаємопов’язаних класів, а на діаграмі класів показано форми та їх функціональні особливості.

На наступному рисунку показана UML-діаграма класів, що описує графічну частину (WPF) програми.

Вона містить три основні класи:

- App — наслідує Application, відповідає за запуск застосунку.
- MainWindow — головна форма (вікно програми). Містить:
  - Поля: _f, _g (імовірно, функції чи дані для побудови графіків).
  - Властивість: PlotModel — модель даних для побудови графіка.
  - Методи: Button_ClearFunctionValues_Click, Create_Click, FindSolution_Click, Open_Click, SaveAs_Click, MenuItem_AboutUs_Click, MenuItem_ExitClick — події та обробники інтерфейсу.
  - Вкладений клас: PolyItem (для опису поліномів або елементів списку).
- AboutUsWindow — додаткове вікно (форма “Про програму”) із методами AboutUsWindow() (конструктор) та Button_OK_Click() (закриває вікно).

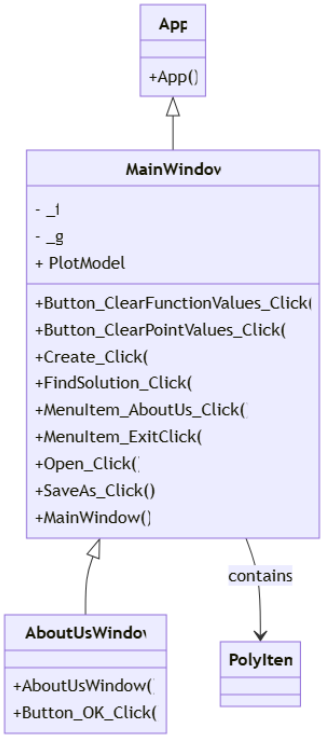

Рисунок 3.8. Форми та функціональні особливості класів

Комплексне використання блок-схем, діаграм послідовності, компонентів і класів має суттєвий дидактичний ефект. Блок-схеми дозволяють студентам послідовно простежити логіку алгоритму та зіставити її з програмним кодом. Діаграми послідовності демонструють, як саме користувач взаємодіє із системою і які об’єкти залучені на кожному етапі роботи. Діаграма компонентів формує уявлення про архітектурну структуру програмного комплексу, розподіл відповідальностей між модулями та можливості подальшої модифікації. Діаграма класів, своєю чергою, забезпечує глибше розуміння об’єктно-орієнтованої моделі програми, що має важливе значення для студентів, які вивчають не лише чисельні методи, а й принципи програмної інженерії.

У підсумку побудова зазначених діаграм на основі реального коду застосунку та його функціонального опису дозволила формалізувати архітектуру інтерактивного навчального комплексу, зробити її прозорою для аналізу та пояснення, а також підготувати ґрунт для подальшого розширення системи шляхом додавання нових методів оптимізації та додаткових навчальних сценаріїв.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Розробка структури програмного комплексу та модулів взаємодії

Розробка структури програмного комплексу для чисельного знаходження мінімуму методом дихотомії ґрунтувалася на принципах модульності, інкапсуляції та гнучкого розподілу функцій між логічними рівнями системи. Архітектура побудована за моделлю поділу на три основні частини: інтерфейс користувача, обчислювальний модуль і модуль зберігання та обробки даних. Такий підхід забезпечує можливість розширення функціональності програми, адаптацію під різні навчальні сценарії та повторне використання обчислювальної бібліотеки в інших проєктах.

Програмний комплекс реалізовано мовою C# із використанням технології Windows Presentation Foundation (WPF) - сучасного графічного фреймворку .NET, що забезпечує незалежність від роздільної здатності екрана, підтримує апаратне прискорення відтворення графіки, дво- та тривимірну візуалізацію, стилізацію елементів інтерфейсу та роботу з мультимедійними об’єктами.

Саме вибір WPF дозволив поєднати інтуїтивний графічний інтерфейс із високопродуктивними алгоритмами чисельних розрахунків.

В основі логіки функціонування лежить клас Dich, який реалізує метод дихотомії для знаходження мінімуму функції. У цьому класі визначено метод FindMinimum(double a, double b, double epsilon), що послідовно обчислює внутрішні точки інтервалу, порівнює значення функції у цих точках і зменшує інтервал до досягнення заданої точності. Алгоритм реалізовано у вигляді циклу з перевіркою умови збіжності, а результати обчислень повертаються у вигляді числового значення мінімуму. Цей фрагмент коду становить ядро обчислювального модуля системи й демонструє застосування структурного підходу до опису алгоритму:

```
public double FindMinimum(double a, double b, double epsilon)
```

```
{
    double x1, x2;
    while ((b - a) / 2 > epsilon)
    {
        x1 = (a + b - epsilon) / 2;
        x2 = (a + b + epsilon) / 2;
        if (f(x1) < f(x2)) b = x2;
        else a = x1;
    }
    return (a + b) / 2;
}
```

Для забезпечення роботи із вхідними даними створено клас Container, який реалізує структуру зберігання інформації про функцію, межі інтервалу та набір контрольних точок. Дані зчитуються із файлів XML за допомогою методу OpenFile() і серіалізуються через об’єкт XmlSerializer. Такий підхід гарантує зручність при збереженні результатів роботи користувача, а також підтримує можливість повторного відкриття файлів для подальших розрахунків. У коді реалізовано перевірку наявності каталогу storage, з якого програма автоматично зчитує всі файли для опрацювання:

```
string directory = "storage";
if (!Directory.Exists(directory))
{
    Console.WriteLine("Каталог storage не знайдено.");
    return;
}
var files = Directory.GetFiles(directory, "*.xml");
foreach (var file in files)
{
    Container data = OpenFile(file);
```

```
    double result = dich.FindMinimum(data.Start, data.End, 0.0001);
    SaveResultToFile(file, result);
}
```

Інтерфейс користувача побудований на основі WPF та надає змогу студенту взаємодіяти із системою через інтуїтивно зрозумілі елементи керування. Користувач вводить формулу полінома, межі інтервалу пошуку, список точок для обчислення та ініціює запуск алгоритму. Після завершення обчислень система відображає результат у графічному вікні — як числове значення мінімуму та графік функції, побудований у реальному часі. Це забезпечує наочність навчального процесу та сприяє кращому розумінню студентами принципів дії методу дихотомії.

Ключовим елементом архітектури є модуль збереження та відновлення даних, що реалізує функції експорту результатів у формат HTML та збереження «знімка» роботи у форматі XML. Це дозволяє не лише документувати результати обчислень, а й продовжити роботу над тими самими даними у майбутньому, що важливо в умовах навчального використання. Збереження результатів здійснюється через метод SaveResultToFile(), який генерує структурований звіт про знайдений мінімум і додає метадані про параметри обчислення.

У структурі програмного комплексу передбачено також взаємодію між модулями через об’єктну модель даних. Головна програма (Program.cs) виконує роль керуючого модуля, який координує усі етапи — від ініціалізації даних до завершення процесу обчислення. Вона створює екземпляри класів, викликає методи обробки, контролює процес візуалізації та організовує діалог із користувачем через графічний інтерфейс.

Отже, структура програмного комплексу має чітку модульну організацію, що забезпечує незалежність компонентів, простоту тестування та можливість розширення. Така архітектура дозволяє у майбутньому інтегрувати нові методи оптимізації або реалізувати багатопоточність для підвищення швидкодії. Крім

того, використання відкритих форматів збереження даних (XML, HTML) робить систему сумісною з іншими навчальними програмами та зручною для включення у дистанційні освітні курси.

4.2. Реалізація навчального модуля з візуалізацією процесу оптимізації та модуля методу дихотомії

Реалізація навчального модуля з візуалізацією процесу оптимізації передбачала поєднання теоретичних принципів чисельних методів із наочним поданням процесу пошуку мінімуму функції. Це дало змогу забезпечити інтерактивну взаємодію між користувачем і програмним комплексом, а також підвищити ефективність засвоєння студентами принципів дії алгоритмів оптимізації.

Основна частина функціоналу реалізована у вигляді двох взаємопов’язаних компонентів: модуля візуалізації та модуля методу дихотомії. Перший відповідає за побудову графічного представлення функції, інтервалів і поточного положення мінімуму, тоді як другий виконує безпосередні чисельні обчислення, забезпечуючи обробку математичних виразів і контроль точності.

Модуль методу дихотомії.

Модуль реалізовано у вигляді окремого класу Dich, який інкапсулює логіку пошуку мінімуму неперервної одновимірної функції. Використання методу дихотомії дає можливість послідовно звужувати інтервал пошуку, зберігаючи гарантовану збіжність алгоритму за умов монотонності функції.

Алгоритм реалізовано за класичною схемою, що включає такі кроки:

1. Визначення початкового інтервалу [a, b];
2. Вибір точності ε;
3. Обчислення двох внутрішніх точок x_1 і x_2 ;

4. Порівняння значень f(x_1) та f(x_2);
5. Звуження інтервалу до нових меж;
6. Повторення циклу до досягнення умов зупинки.

Програмна реалізація методу наведена у коді нижче:

```
public class Dich
{
   private Func<double, double> f;

   public Dich(Func<double, double> func)
   {
      f = func;
   }

   public double FindMinimum(double a, double b, double epsilon)
   {
      double x1, x2;
      while ((b - a) / 2 > epsilon)
      {
         x1 = (a + b - epsilon) / 2;
         x2 = (a + b + epsilon) / 2;
         if (f(x1) < f(x2)) b = x2;
         else a = x1;
      }
      return (a + b) / 2;
   }
}
```

Така реалізація забезпечує універсальність - користувач може передавати будь-яку функцію як параметр делегата Func<double, double>, що дозволяє застосовувати метод дихотомії до різних типів задач. Алгоритм є стабільним,

має лінійну складність за кількістю ітерацій і придатний для інтерактивного відображення проміжних результатів у модулі візуалізації.

Модуль візуалізації процесу оптимізації.

Для побудови графічної частини використано технологію Windows Presentation Foundation (WPF), яка забезпечує апаратне прискорення рендерингу та підтримку двовимірної графіки. Візуалізація дозволяє у реальному часі відстежувати зміну інтервалу пошуку, порівняння значень функції в обраних точках та поступове зближення до мінімуму.

У вікні програми користувач бачить координатну сітку з графіком функції та позначеними поточними точками x_1 і x_2. На кожній ітерації виконуються оновлення позицій точок і значень функції, що забезпечує розуміння принципів роботи алгоритму не лише формально, а й на інтуїтивному рівні. Візуалізація реалізована за допомогою елементів Canvas та Polyline, які дозволяють динамічно малювати криві функцій.

Фрагмент коду, що відповідає за відображення функції, наведено нижче:

```
Polyline graph = new Polyline();
graph.Stroke = Brushes.Blue;
for (double x = a; x <= b; x += 0.01)
{
   double y = f(x);
   graph.Points.Add(new Point(x * scaleX, y * scaleY));
}
canvas.Children.Add(graph);
```

У ході роботи алгоритму дані, отримані з обчислювального модуля, передаються у візуальний компонент для оновлення графіка. Цей механізм реалізує принцип Model-View-Controller (MVC), де модель відповідає за обчислення, представлення — за відображення, а контролер — за координацію між ними.

Взаємодія модулів.

Модулі взаємодіють через чітко визначені інтерфейси, що забезпечує гнучкість і можливість подальшої інтеграції нових алгоритмів. Головна програма створює об’єкт класу Dich, ініціалізує параметри функції, межі інтервалу та передає їх у компонент візуалізації. Після кожної ітерації обчислень результати оновлюються на графіку, що дозволяє студенту спостерігати динаміку наближення до мінімуму.

Архітектурно взаємодія модулів може бути подана у вигляді послідовності: Користувач → Інтерфейс (WPF) → Модуль дихотомії (Dich) → Модуль візуалізації → Вивід результатів.

Такий підхід дозволяє розглядати програму не лише як навчальний засіб, але й як демонстраційний інструмент для курсу «Методи оптимізації» або «Обчислювальна математика».

Освітній ефект і технічні особливості

Завдяки використанню інтерактивної візуалізації, студенти можуть у реальному часі спостерігати зміну параметрів функції, інтервалів і точок, що значно покращує розуміння дії алгоритму. Такий підхід відповідає принципам дидактичної наочності та інтерактивності навчання.

Крім того, реалізована архітектура дозволяє:

- працювати у офлайн-режимі без підключення до Інтернету;
- зберігати результати обчислень у форматі XML або HTML;
- легко адаптувати код для інших методів оптимізації (золотого перетину, Ньютона, генетичного алгоритму тощо).

Таким чином, розроблений модуль поєднує аналітичну точність чисельних методів із сучасними засобами комп’ютерної графіки, створюючи ефективний інструмент для навчання методам оптимізації в інтерактивному форматі.

4.3. Побудова інтерактивного інтерфейсу користувача

Інтерфейс користувача є ключовим компонентом інтерактивного навчального комплексу, оскільки саме він забезпечує комунікацію між користувачем і програмною системою, а також визначає зручність і ефективність роботи із застосунком. Під час розроблення інтерфейсу головним завданням було створення інтуїтивно зрозумілого, логічно структурованого й адаптивного середовища, що дозволяє користувачам будь-якого рівня підготовки взаємодіяти з програмою без потреби в спеціальній технічній підготовці.

Для реалізації графічного інтерфейсу використано технологію Windows Presentation Foundation (WPF), що входить до складу платформи .NET Framework. Вона забезпечує можливість побудови сучасних графічних інтерфейсів із використанням апаратного прискорення, підтримкою анімації, шаблонів даних, векторної графіки та інтеграції із зовнішніми бібліотеками. Вибір WPF був зумовлений також можливістю створення MVVM-архітектури (Model–View–ViewModel), яка дозволяє розділити логіку обчислень, модель даних і візуальне представлення, забезпечуючи гнучкість та масштабованість програмного коду.

Структура інтерфейсу.

Головне вікно програми розділено на три функціональні області:

1. Область введення даних, де користувач задає функцію, межі інтервалу та параметри точності обчислень;
2. Область відображення результатів, у якій візуалізується графік функції, поточні точки пошуку мінімуму та результат обчислення;
3. Область керування, що містить кнопки запуску алгоритму, очищення

вікна, збереження та експорту результатів.

Розмітку вікна створено у файлі MainWindow.xaml із використанням елементів Grid, StackPanel та Canvas. Нижче наведено приклад базової структури графічного інтерфейсу у форматі XAML:

```
<Window x:Class="MyConsoleApp.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Метод дихотомії" Height="600" Width="900">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <StackPanel Orientation="Horizontal" Grid.Row="0" Margin="10">
      <Label Content="Функція:"/>
      <TextBox x:Name="FunctionInput" Width="200"/>
      <Label Content="Інтервал:"/>
      <TextBox x:Name="StartInput" Width="50"/>
      <TextBox x:Name="EndInput" Width="50"/>
      <Label Content="Точність ε:"/>
      <TextBox x:Name="EpsilonInput" Width="60"/>
      <Button x:Name="StartButton" Content="Запустити" Width="100" Margin="10,0,0,0"/>
    </StackPanel>

    <Canvas x:Name="GraphCanvas" Grid.Row="1" Background="WhiteSmoke" Margin="10"/>
```

```
        <TextBlock x:Name="ResultOutput" Grid.Row="2" FontSize="16" FontWeight="Bold"
                   HorizontalAlignment="Center" Margin="10"/>
    </Grid>
</Window>
```

Така структура дозволяє швидко ініціалізувати дані для обчислень і відображати проміжні та кінцеві результати. Інтерфейс забезпечує повний цикл взаємодії користувача з програмою — від введення вхідних параметрів до аналізу графічних результатів.

Взаємодію між елементами інтерфейсу та обчислювальними модулями реалізовано у файлі MainWindow.xaml.cs. Під час натискання кнопки «Запустити» виконується обробник події, який зчитує введені дані, створює об’єкт класу Dich, ініціалізує функцію для обчислення та запускає алгоритм методу дихотомії. Після завершення розрахунків результати передаються у візуальний компонент для побудови графіка.

```
private void StartButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        double a = double.Parse(StartInput.Text);
        double b = double.Parse(EndInput.Text);
        double epsilon = double.Parse(EpsilonInput.Text);
        string expression = FunctionInput.Text;

        Func<double, double> f = x => EvaluateFunction(expression, x);
        Dich dich = new Dich(f);

        double min = dich.FindMinimum(a, b, epsilon);
```

```
            ResultOutput.Text = $"Мінімум функції: x = {min:F4}";

            DrawGraph(f, a, b);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка введення даних: {ex.Message}");
        }
    }
```

Реалізація передбачає обробку винятків (try-catch), що гарантує стабільність роботи програми навіть при введенні некоректних даних. Завдяки цьому система може бути безпечно використана студентами без загрози збоїв або втрати результатів.

Після завершення обчислень на графіку зображається функція та точка мінімуму, позначена маркером червоного кольору. Це дає змогу користувачу не лише побачити числовий результат, а й оцінити процес звуження інтервалу пошуку. Функція візуалізації реалізована наступним чином:

```
    private void DrawGraph(Func<double, double> f, double a, double b)
    {
        GraphCanvas.Children.Clear();
        Polyline graph = new Polyline();
        graph.Stroke = Brushes.Blue;
        graph.StrokeThickness = 2;

        for (double x = a; x <= b; x += 0.01)
        {
            double y = f(x);
            graph.Points.Add(new Point((x - a) * 50, 250 - y * 50));
        }
```

```
    GraphCanvas.Children.Add(graph);
}
```

Використання графічного елемента Polyline дозволяє малювати плавні криві функцій у двовимірній системі координат. Кожна точка обчислюється у циклі з кроком 0.01, що забезпечує достатню точність відображення навіть для функцій із різкими змінами похідних.

Інтерфейс програми реалізує низку інтерактивних можливостей: масштабування графіка, повторне обчислення функції при зміні параметрів, збереження результатів у файл, а також автоматичне оновлення при повторному запуску. Така поведінка забезпечує гнучкість навчального процесу та дозволяє викладачам демонструвати принципи роботи методу дихотомії у динамічній формі.

Крім того, система передбачає можливість подальшого розширення інтерфейсу: додавання нових вкладок для інших методів оптимізації, інтеграцію елементів довідки, реалізацію навчальних підказок і покрокових пояснень виконання алгоритму.

Таким чином, розроблений інтерактивний інтерфейс поєднує простоту використання з широкими можливостями візуалізації. Його архітектура побудована на основі WPF і реалізує класичний принцип розділення відповідальності між модулями. Це забезпечує ефективну взаємодію користувача з програмним комплексом, створюючи комфортні умови для навчання методам чисельної оптимізації.

4.4. Інструкція з використання програмного комплексу

Розроблений інтерактивний програмний комплекс призначений для

навчання методам чисельної оптимізації та демонстрації алгоритму пошуку мінімуму функції методом дихотомії. Програма орієнтована на використання у навчальному процесі при викладанні дисциплін що вивчають методи оптимізації та знаходження екстремумів функцій, а також для самостійної практичної роботи студентів.

Програмний комплекс розроблено під .NET Framework 4.8 і працює на операційних системах Windows 10 або новіших. Для його функціонування достатньо стандартного пакета бібліотек .NET — додаткових зовнішніх компонентів не потрібно.

Рекомендована структура проєкту має вигляд:

```
Console/
├── bin/
│   └── Debug/
│       ├── Console.exe
│       └── Library.dll
├── storage/
│   ├── example1.xml
│   ├── example2.xml
└── report.xml
```

Каталог storage використовується для зберігання вхідних файлів формату XML, які містять опис функції, межі інтервалу та допоміжні дані для обчислення. Програма автоматично зчитує всі доступні файли при запуску.

Для запуску необхідно відкрити Console.exe. Після ініціалізації відображається головне вікно програми (рисунок 4.1), яке поділене на три основні частини:

- панель введення параметрів (введення функції, меж інтервалу та точності ε);
- графічну область для побудови функції;
- панель результатів із текстовим виводом обчислень.

Користувач вводить функцію і визначає інтервал [a; b]. Після натискання кнопки «Запустити» виконується обчислення мінімуму за методом дихотомії. У нижній частині вікна з’являється результат:

Мінімум функції: x = 2.0000, а на графіку позначається точка мінімуму червоним кольором.

Вхідні дані зберігаються у форматі XML:

```
<Container>
   <Start>0</Start>
   <End>5</End>
   <Poly>x^2 - 4x + 3</Poly>
   <Points>...</Points>
</Container>
```

Під час запуску програма перевіряє наявність каталогу storage. Якщо він відсутній, на екран виводиться повідомлення: Directory not found

Метод дихотомії поступово зменшує інтервал [a, b], поки довжина не стане меншою за ε. Графічна візуалізація відображає зміщення точок x_1 , x_2 і середини інтервалу в реальному часі. Таким чином користувач спостерігає весь процес пошуку мінімуму, що значно підвищує розуміння принципів дії методу.

Результати автоматично серіалізуються у файл report.xml із використанням класу XmlSerializer. Запис містить мінімум функції, межі інтервалу та час виконання обчислень. Приклад методу:

```
private static void SaveResultToFile(Point min, string path)
{
   XmlSerializer serializer = new XmlSerializer(typeof(Point));
   StreamWriter writer = new StreamWriter(path);
   serializer.Serialize(writer, min);
   writer.Close();
}
```

Дані можуть бути використані для подальшого аналізу або експорту у

навчальні системи.

У разі неправильного введення даних програма виводить повідомлення:

- *«Помилка: неправильний формат введених даних»* – якщо замість числа введено текст;
- *«Directory not found»* – якщо відсутня папка для зчитування файлів;
- *«XML serialization error»* – при порушенні структури вхідного файлу.

Система реалізує обробку винятків, тому помилки не призводять до аварійного завершення роботи.

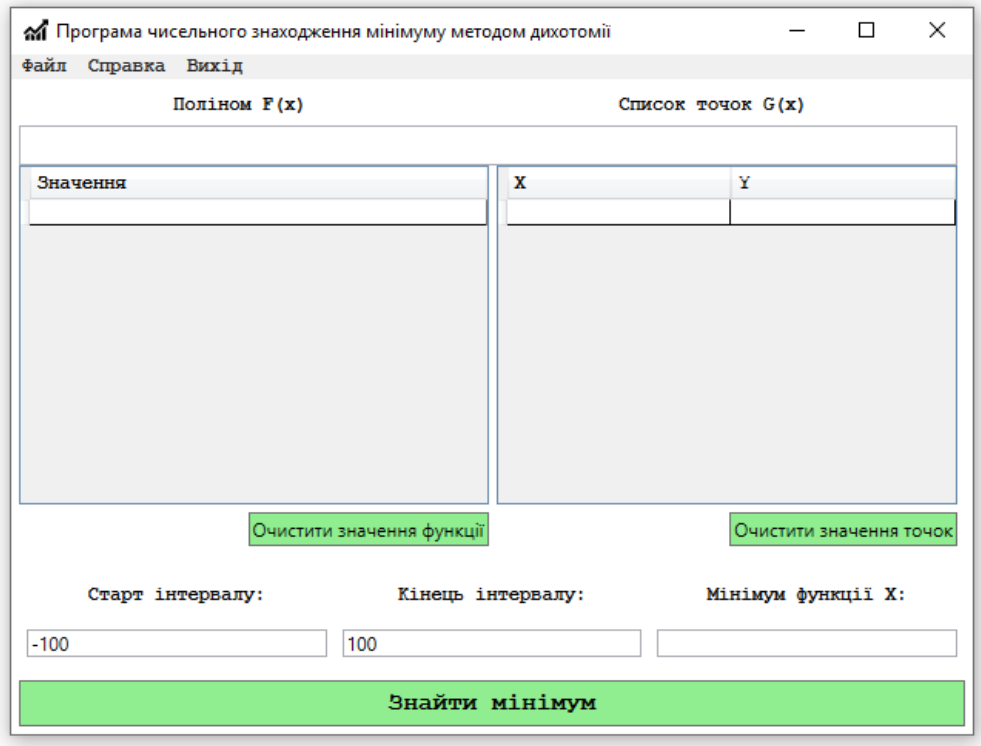

Рисунок 4. 1 Головне вікно програми

Після запуску основного додатка та ознайомлення з простим та зручним інтерфейсом, користувачеві необхідно заповнити дані такі поля:

Заповніть формулу многочлена:

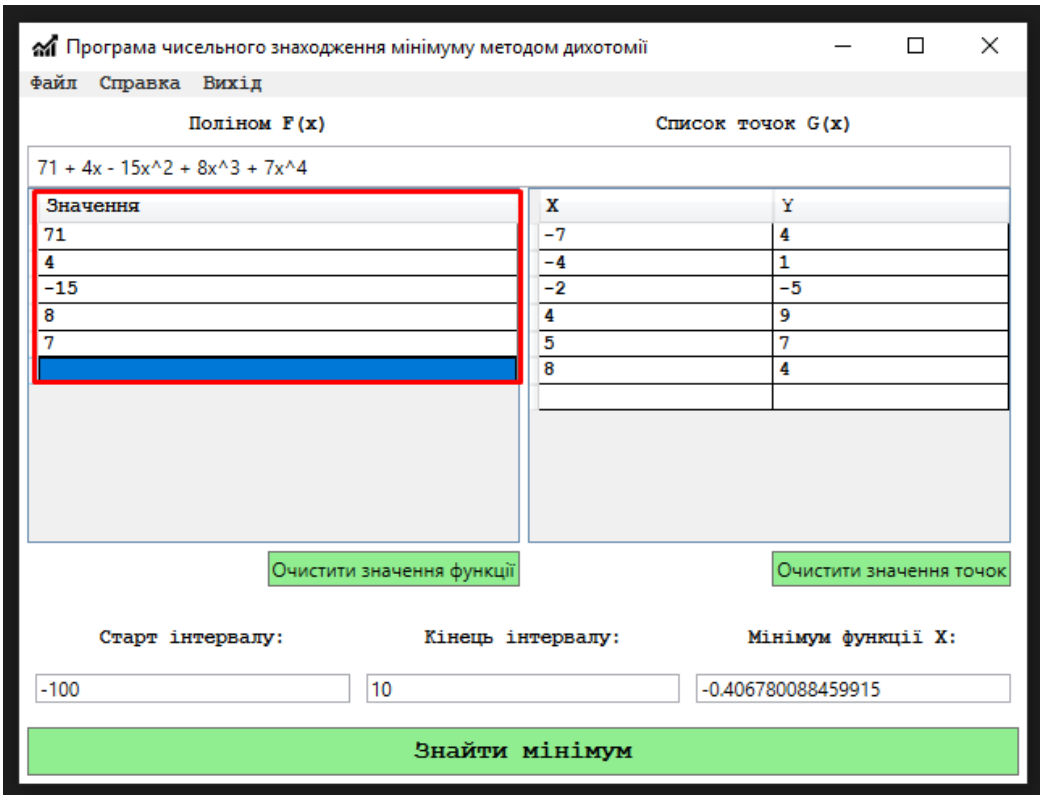

Рисунок 4.2. Заповнення формули многочлена

Далі заповнюємо список точок для розрахунку.

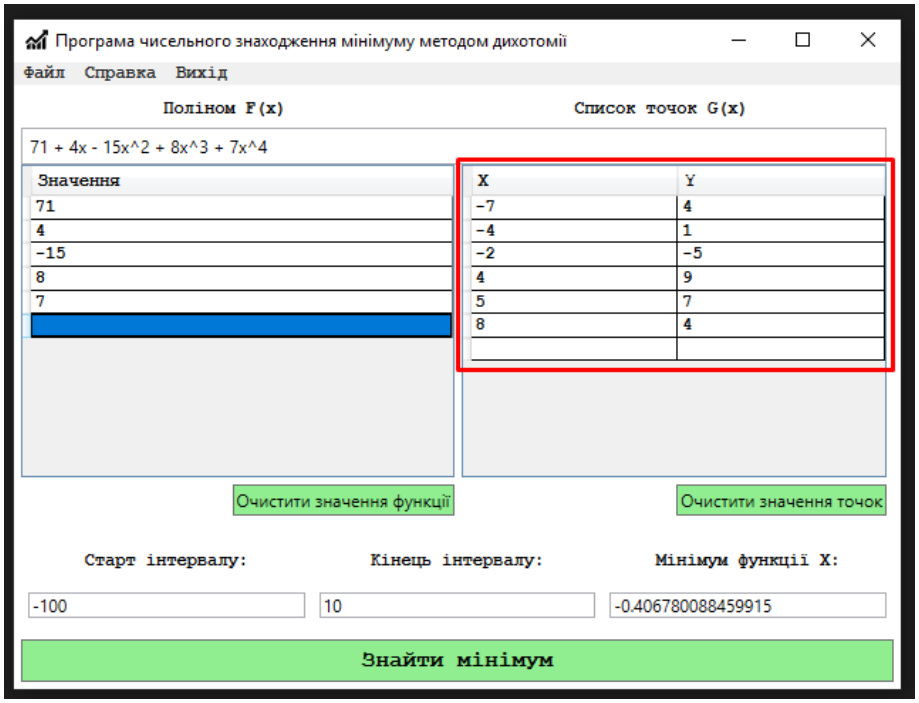

Рисунок 4.3. Заповнення точок для розрахунку

Вказуємо діапазон інтервалу, на якому будемо проводити розрахунки.

Старт інтервалу:	Кінець інтервалу:	Мінімум функції X:
-100	10	-0.406780088459915

Після введення даних починаємо шукати мінімум. Результати розрахунків будуть представлені та візуалізовані у графічному вікні.

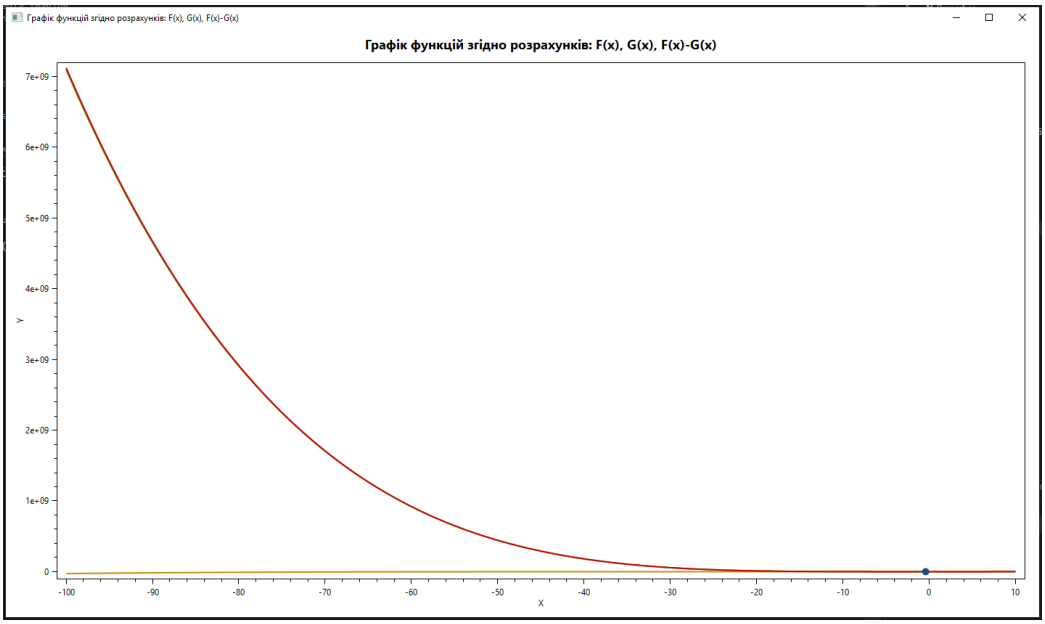

Рисунок 4.4. Результати побудови графіка

Після ознайомлення система надасть можливість збереження результатів в форматі html: відкриється діалогове вікно і користувач зможе зберегти результат пошуку в зручному для читання універсальному форматі:

Звіт

У результаті знаходження мінімуму методом дихотомії, з наступними вхідними даними:

Функція F:

71 + 4x - 15x^2 + 8x^3 + 7x^4

Функція G:

#	Значення X	Значення Y
1	-7	4
2	-4	1
3	-2	-5
4	4	9
5	5	7
6	8	4

Знайдене рішення:

X: -0,406780088459915

Рисунок 4.5. Звіт про виконані обчислення

Для зручності користувача, для збереження прогресу і введених даних, є можливість зберегти знімок програми в форматі xml:

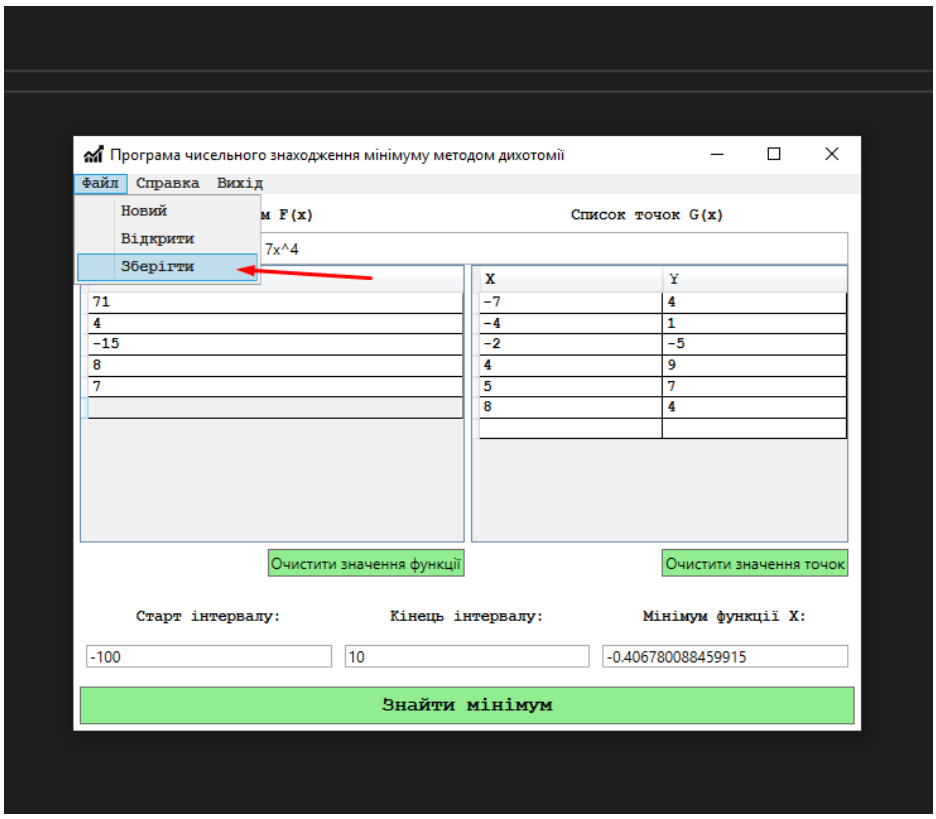

Рисунок 4.6. Процес збереження даних для проєкту

Можливий і зворотний процес: використовуючи збережений знімок з вихідними даними, є можливість завантажити його в систему і працювати з розглянутим прикладом

Отже, розроблений програмний комплекс реалізує сучасні принципи побудови навчальних програм: інтерактивність, візуалізацію, модульність і простоту використання. Використання C# та WPF забезпечило поєднання математичної точності, високої продуктивності та зручного інтерфейсу. Завдяки цьому програма може бути інтегрована у навчальні курси з програмування, обчислювальної математики або оптимізаційних методів як ефективний засіб демонстрації та практичного засвоєння матеріалу.

ВИСНОВКИ

У межах магістерської роботи було розроблено інтерактивний програмний комплекс для навчання методам чисельної оптимізації, який поєднує модульну архітектуру, чисельні методи пошуку екстремумів і засоби візуалізації процесів оптимізації в середовищі Windows Presentation Foundation (WPF). Запропоноване рішення орієнтоване на підвищення ефективності навчального процесу з дисциплін обчислювального та прикладного спрямування.

У ході виконання роботи:

– проведено аналіз існуючих навчальних програмних засобів для вивчення методів оптимізації та визначено їх переваги й обмеження;

– обґрунтовано вибір мови програмування C# і технології WPF, яка забезпечує векторну графіку, анімацію, макети, шаблони, мультимедіа та незалежність від роздільної здатності екрана;

– розроблено архітектуру програмного комплексу, що складається з обчислювального модуля, модуля даних і модуля візуалізації;

– реалізовано метод дихотомії як основний алгоритм пошуку мінімуму функції, адаптований до інтерактивної візуалізації;

– створено навчальний модуль візуалізації, який у реальному часі відображає графік функції, точки пошуку та поточні інтервали;

– спроєктовано та реалізовано інтерфейс користувача з інтуїтивною структурою, що забезпечує введення вхідних даних, управління параметрами обчислення та відображення результатів;

– проведено тестування програми, що підтвердило коректність обчислень, стабільність роботи та придатність для використання у навчальному процесі.

Розроблений програмний комплекс дозволяє користувачу:

– вводити аналітичні вирази функцій і параметри обчислення (інтервал, точність ε);

–

– отримувати числові результати з автоматичним збереженням у XML-звіт;

– аналізувати поведінку функцій різного типу в інтерактивному режимі без потреби підключення до мережі.

Практична цінність розробки полягає у поєднанні точності чисельних методів і дидактичної наочності. Система може бути використана як самостійний навчальний інструмент або інтегрована у дистанційні курси. Завдяки автономності, невеликому обсягу та простоті налаштування комплекс придатний для роботи у комп’ютерних класах, на лабораторних заняттях або при самостійному вивченні.

Отримані результати створюють передумови для подальшого розвитку програмного продукту, зокрема розширення набору реалізованих методів (градієнтний, Ньютона, золотого перетину, генетичний тощо); реалізації можливості побудови тривимірних графіків і анімації еволюції процесу оптимізації; інтеграції з навчальними системами та веб-інтерфейсами для дистанційного доступу; розробки тестового модуля для автоматичної перевірки знань студентів.

Таким чином, поставлену мету дослідження досягнуто, розроблений програмний комплекс підтвердив ефективність використання WPF і мови C# для побудови сучасних інтерактивних навчальних систем, що поєднують точність математичних алгоритмів із наочністю графічного подання.

По результатам роботи булj підготовлено статтю у збірник праць науково-практичного семінару «Комп’ютерні науки та інформаційні технології».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alridha A. H. et al. A Review of Optimization Techniques: Applications and Comparative Analysis. Iraqi Journal of Computer Science and Mathematics, Vol. 5 No. 2, 2024.

2. Beiranvand V. Best Practices for Comparing Optimization Algorithms. arXiv:1709.08242, 2017.

3. Microsoft. Windows Presentation Foundation (WPF) Overview. — URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/

4. Nielsen M. A. Neural Networks and Deep Learning. – Determination Press, 2015.

5. Nocedal J., Wright S. J. Numerical Optimization. – Springer, 2006.

6. Ruder S. An overview of gradient descent optimization algorithms. – arXiv:1609.04747, 2016.

7. Глушков В.М., Єременко А.В. Інформаційні технології у навчанні математичного моделювання та оптимізації. – Київ: Наукова думка, 2020. – 186 с.

8. Кисельов О. М. Чисельні методи оптимізації. – Київ: КНУ, 2020.

9. КОМП'ЮТЕРНІ НАУКИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ (КНІТ-2025): матеріали науково-практичного семінару. Випуск 4 / за ред. О.В. Ольховської – Полтава: Кафедра КНІТ ПУЕТ, 2025. – 56 с. URL: http://www.matmodel.puet.edu.ua/files/knit-zb2025-4.pdf

10. Кошова О.П., Ольховська О.В., Чілікіна Т.В., Шуляр С. Особливості розробки програмного забезпечення для моделювання та дослідження бізнес-процесів з допомогою кореляційно-регресійного аналізу Вісник Кременчуцького національного університету імені Михайла Остроградського. Кременчук: КрНУ, 2024, № 3 С.86-92. DOI <https://doi.org/10.32782/1995-0519.2024.3.12>

11. Черненко О. О. Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня магістра / С. В. Гаркуша, О. В. Ольховська, О. О. Черненко. – Полтава : ПУЕТ, 2023. – 68 с. Режим доступу: URL: http://elib.puet.edu.ua/action.php?kt_path_info=ktcore.SecViewPlugin.actions.document&fDocumentId=824868 - Назва з екрану

12. Чілікіна Т.В. Особливості розробки web-застосунків для системи дистанційного навчання з допомогою бібліотеки React / О.П. Кошова, О.О. Черненко, Т.В. Чілікіна, І.І. Комар // Системи та технології, 2023. Вип. 65(1). С. 20-31. https://st.umsf.in.ua/index.php/journal/article/view/101

ДОДАТОК А
Код програми

```
</Project>
using System;
using System.Collections.Generic;
using System.IO;
using System.Reflection;
using System.Xml.Serialization;
using Library;
using static Library.Data;

namespace MyConsoleApp
{
   static class Program
   {
      static void Main()
      {
         string pathToDirectoryDb =
Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location) + "\\storage";
         //System.Console.WriteLine($"Загружаем данные из папки? {pathToDirectoryDb}");

         if (!Directory.Exists(pathToDirectoryDb))
         {
            System.Console.WriteLine("Directory not found");
            return;
         }
         string[] allPath = Directory.GetFileSystemEntries(pathToDirectoryDb);
         foreach (string path in allPath)
         {
            if (Path.GetExtension(path) == ".xml")
            {
```

```
            Container loadedData = null;
            loadedData = OpenFile(path, out loadedData);
            System.Console.WriteLine();
            System.Console.ForegroundColor = ConsoleColor.Green;
            System.Console.WriteLine("Считываем данные  для анализа из файла: ");
            Console.ResetColor();
            System.Console.WriteLine($"#Поліном: {loadedData.poly}");
            System.Console.WriteLine($"#Список точок:\n{loadedData.points}");
            System.Console.WriteLine($"#Діапазон пошуку: [{loadedData.Start}; {loadedData.End}]");

            Point min = FindMinimum(loadedData);
            System.Console.WriteLine($"Мінімум: {min}");

            SaveResultToFile(min, "report.xml");
            /*Custom add ----------------------start*/
        }
    }
}



private static void SaveResultToFile(Point min, string path)
{
    XmlSerializer serializer = new XmlSerializer(typeof(Point));
    StreamWriter writer = new StreamWriter(path);
    serializer.Serialize(writer, min);
    writer.Close();
}

static Point FindMinimum(Container data)
{
    Dich dich = new Dich(data.poly, data.points);
```

```
            return dich.FindMinimum(data.Start, data.End, 0.0001);
        }

        static Container OpenFile(string path, out Container cont)
        {
            cont = null;
            cont = new Container();
            return Data.Read(path, out cont.Start, out cont.End, out cont.poly, out cont.points);
        }

        static bool InDouble(out double value) => double.TryParse(System.Console.ReadLine(), out value);

        static void Error(string text)
        {
            System.Console.WriteLine($"Помилка: {text}");
            System.Console.ReadKey();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props')"/>
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == " ">Debug</Configuration>
```

```
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{1D612170-6DD7-49BF-AD0B-57C347ADD961}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>Console</RootNamespace>
    <AssemblyName>Console</AssemblyName>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Data" />
```

```
    <Reference Include="System.Xml"/>
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Program.cs"/>
    <Compile Include="Properties\AssemblyInfo.cs"/>
  </ItemGroup>
  <ItemGroup>
   <ProjectReference Include="..\Library\Library.csproj">
    <Project>{6ef3b006-902b-4472-a17c-f001bd080fb1}</Project>
    <Name>Library</Name>
   </ProjectReference>
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets"/>
  <!-- To modify your build process, add your task inside one of the targets below and uncomment
it.
      Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
```