МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук, фізики та математики

Кафедра комп'ютерних наук та програмної інженерії

Програмне середовище навчального призначення «Предмет та основні задачі обчислювальної геометрії» Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «магістр»

Виконав: студент 2 курсу 231М групи
Спеціальності: 122 «Комп'ютерні науки»
Освітньо-професійної (наукової) програми: «Комп'ютерні науки»
Гаран Олег Віталійович

Керівник: Львов М.С.
наук. керівник доктор фізико-математичних наук, професор
Рецензент: Івашина Ю.К.
кандидат фізико-математичних наук, доцент

Херсон – Івано-Франківськ – 2023

ЗМІСТ

ВСТУП

ПРЕДМЕТ ОБЧИСЛЮВАЛЬНОЇ ГЕОМЕТРІЇ. ОГЛЯД ОСНОВНИХ ЗАДАЧ ОБЧИСЛЮВАЛЬНОЇ ГЕОМЕТРІЇ

Обчислювальна геометрія є галуззю математики та інформатики, яка займається розв'язуванням геометричних задач з використанням алгоритмів та комп'ютерів. Основні задачі обчислювальної геометрії включають:

Перетин прямих та площин: обчислювання точок перетину прямих та площин у тривимірному просторі.

Обчислення відстаней: обчислення відстані між точками, відстані від точки до прямої або площини.

Побудова опуклої оболонки: знаходження найменшого опуклого многокутника, що охоплює задану множину точок.

Побудова діаграми Вороного: побудова діаграми Вороного для заданої множини точок, що розділяє простір на області, кожна з яких належить лише одній точці.

Обчислення перетину многокутників: обчислення області перетину двох многокутників.

Обчислення інтервалів перетину: обчислення інтервалів, в яких знаходяться точки перетину двох променів або відрізків.

Обчислення найближчих сусідів: знаходження найближчої точки або множини точок до даної точки.

Ці задачі мають велике застосування в різних областях, таких як комп'ютерна графіка, геоінформатика, робототехніка, медична інженерія та інші.

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд основних задач обчислювальної геометрії

Хоча чисельна геометрія нараховує не менше тисячі років, але велике практичне значення вона набула під час другої світової війни, коли потреби промисловості, особливо літакобудування, стимулювали розробку нових методів проектування. Звичайно під обчислювальною геометрію, розуміють науку про способи представлення, аналіз та синтез інформації про геометричні об'єкти. Після обчислення математичного представлення форми об'єкта, цілком природно зберігати його в пам'яті ЕОМ, що дає ряд переваг. При цьому форма об'єкта знаходиться в пам'яті машини у вигляді чисто цифрових даних. За допомогою ЕОМ можна легко обчислити такі геометричні характеристики об'єкта, як об'єм його частин, контури або площі поперечних перетинів. Також легко провести візуалізацію інформації відносно форми об'єкта, що є суттєвим при його проектуванні.

Задачі, що розв'язуються методами обчислювальної геометрії досить умовно можна поділити на дві групи. До першої відносяться задачі, що пов'язані з проектуванням кривих і поверхонь з потрібними властивостями. До другої групи належать задачі, які виникають у різноманітних інженерних та економічних застосування коли об'єкти цих задач допускають геометричне трактування. Це задачі регіонального пошуку, задачі про покриття і проміжки , задачі геометрії прямокутників, визначення певних характеристик множини точок на площині і в просторі.[1]

Метод обчислювальної геометрії мають в своїй основі досягнення цілого ряду областей математики . Частина з них , такі як аналітична геометрія і чисельні методи, викладаються в інженерному курсі вищої математики, а диференціальна геометрія, яка є базовою для обчислювальній геометрії, входить тільки в університетські курси. Тому в даний посібник включені основні поняття диференціальна геометрія кривих і поверхонь в обсязі необхідному для розуміння характеристик геометричних об'єктів і основних операцій з ними.

Обчислювальна геометрія – це наука (розділ інформатики), в якій розглядаються алгоритми розв'язування геометричних задач для роботи з графічними зображеннями, наприклад задача тріангуляції полігонів, побудови згладжуючих кривих, геометричного пошуку побудови перетину та об'єднання об'єктів, здійснення афінних та проєктивних перетворень тощо. Класична геометрія часто не дає можливості спроєктувати ефективні алгоритми для розв'язування задач на ЕОМ. Тому необхідно було запропонувати нові підходи в геометрії, які сприятимуть ефективності обчислень на комп'ютері. Обчислювальна геометрія надала класичній геометрії обчислювальної форми. Обчислювальна геометрія має свою специфіку і відрізняється від класичної геометрії, на яку вона опирається. Знання з обчислювальної геометрії важливі для ефективного розв'язування геометричних задач у різних галузях людської діяльності. Основу обчислювальної геометрії заклали геометричні задачі, для яких необхідно було розробити оптимальні алгоритми їх розв'язування, хоча геометричні обчислення мають давню історію. Як самостійна дисципліна вона зародилася в кінці 1970-х років. Термін "обчислювальна геометрія" запропонований у 1975 р. в книзі Препати і Шеймоса [46]. Її успіх як окремої галузі наукових досліджень можна пояснити широтою застосування – комп'ютерна графіка, геоінформаційні системи (ГІС), робототехніка, конструкторська справа та інші галузі, в яких геометричні алгоритми відіграють значну роль. Знання обчислювальної геометрії сприяють ефективному розв'язуванню геометричних задач у різноманітних галузях. Багато задач обчислювальної геометрії виникає при візуалізації геометричних об'єктів. Зокрема, статичні задачі перетину геометричних об'єктів, тріангуляції полігонів та набору точок, задачі геометричного пошуку, динамічні задачі в яких поступово змінюються вхідні дані.[2]

1.2 Огляд та аналіз аналогів програмного середовища навчального призначення

Аналіз доцільності використання програмних засобів (реалізація на персональних електронних обчислювальних машинах) для навчання є предметом вивчення та дослідження багатьма фахівцями в галузі інформатизації освіти. Цей аналіз дозволяє створити обґрунтовані підходи до класифікації програмних засобів, які використовуються у навчальних цілях, і встановити вимоги до їх якості на основі дидактичної доцільності.

На сьогоднішній день під час навчання курсу інформатики та інших загальноосвітніх предметів часто використовуються різні види програмного забезпечення. Це може бути як математичне програмне забезпечення для розв'язування певних завдань на персональних електронних обчислювальних машинах, так і програмне забезпечення, яке відповідає навчальним цілям та завданням навчального процесу. Крім того, розробляються спеціальні програми і системи для навчання та виховання.

Цей аналіз допомагає визначити, наскільки програмні засоби відповідають потребам навчання і допомагають досягти бажаних педагогічних результатів. Такий підхід допомагає покращити якість освіти та сприяє ефективному використанню програмного забезпечення в навчальному процесі. [3]

Розробка програмного забезпечення (ПЗ) для навчальних цілей в основному поділяється на два напрямки:

а) Ініціативна розробка авторськими колективами або окремими авторами: В цьому випадку, автори створюють програмне забезпечення на власний розсуд, враховуючи тематику навчальних курсів або незалежно від них. Такий підхід дозволяє авторам виявити творчий підхід і реалізувати власні ідеї в програмному забезпеченні для навчання.

б) Розробка за координаційними планами відомств, наукових організацій, фірм: У цьому випадку розробка програмного забезпечення проводиться відповідно до планів та ініціатив відомств, наукових організацій або компаній. Метою є насичення сфери освіти програмними засобами, які можуть бути використані у процесі навчання.

Програмне забезпечення навчального призначення використовується в різних форматах:

Окремі програми: Це програми, які забезпечують частину занять або весь урок, декілька уроків і призначені для навчання конкретного матеріалу.

Пакети програм: Це набір програм, які забезпечують навчальний предмет, розділ або курс інформативною підтримкою, включаючи різноманітні інтерактивні вправи, довідковий матеріал і т.д.

Практикуми: Це програмне забезпечення, яке дозволяє проводити практичні заняття та вправи для навчального курсу.

Компоненти комп'ютерного курсу або мультимедійного курсу: Це програми і матеріали, які становлять частину великого навчального курсу.

Використання програмного забезпечення для навчання часто супроводжується навчально-методичною літературою, яка допомагає структурувати процес використання ПЗ в навчанні.

У більшості випадків доцільність використання програмного забезпечення обгрунтовується авторами через необхідність автоматизації контролю і тестування знань. Проте, сучасні ПЕОМ можуть забезпечити більш ефективний контроль, діагностику помилок та інтерактивний підхід до навчання. Переважна більшість програм для навчання обмежуються пропозицією вибрати вірну відповідь з обмеженого списку або констатацією правильності відповіді, але цей підхід не завжди є методологічно виправданим.

Програмні рішення, що реалізують підходи програмованого навчання за допомогою комп'ютера, широко представлені як у вітчизняних, так і у зарубіжних розробках. Ці програми в основному представляють собою послідовні або розгалужені програми, які виконуються на комп'ютері. Вони

зазвичай не пропонують чогось нового у відношенні до дидактичних можливостей.

Деякі програми підтримки навчання, наприклад, курси з інформатики, математики та фізики, спрямовані на створення спеціалізованого програмного середовища для взаємодії з користувачем. Ці програми більш перспективні для створення контролюючих програм, тренінгових програм або програм, які використовують ідеї програмованого навчання. Вони ширше використовують можливості сучасних ПЕОМ, зокрема графіку та обчислення, та надають учням інструмент для дослідження предметної області та втілення ідей діяльнісного підходу до навчання.[4]

Інструментальні програмні засоби призначені для створення програмних засобів для навчання, включаючи графічні інтерфейси та сервісні додатки, підготовку навчальних та організаційних матеріалів:

1. Забезпечення взаємодії користувача з програмою через діалогові засоби підтримки.
2. Контроль над процесом засвоєння навчального матеріалу та реакція програми на результати контролю, включаючи самоконтроль.
3. Включення в програму можливостей для проведення обчислювальних операцій.
4. Додавання в програму засобів, які поліпшують її дизайн та оформлення.
5. Автоматизоване генерування та розсилання навчальних матеріалів на робочі місця учнів.
6. Можливість взаємодії програми з загальнопризначеними програмними засобами, такими як редактор тексту.
7. Створення декількох робочих областей на екрані для створення активних зон, які реагують на дії користувача.

Також важливим є розробка сервісних програмних засобів для викладачів, які автоматизують контроль засвоєння матеріалу та просування у

навчанні, генерують завдання, керують процесом навчання та дозволяють передавати програмне забезпечення через мережу.

Сервісні програмні засоби є надзвичайно важливими для викладачів, оскільки вони допомагають управляти навчальним процесом та полегшують роботу користувача. У різних програмних системах часто зустрічаються програмні засоби, які моделюють об'єкти, процеси, експерименти або явища для демонстрації та контролю. Вони можуть бути як програмами-тренажерами, які об'єднують демонстрацію та контроль експерименту, так і програмами, які створюють модель об'єкта або процесу для навчання та надають зворотний зв'язок.

Використання гри в програмах навчального призначення дозволяє імітувати навчальну ситуацію та надавати завдання на фоні ігрового сюжету. Проте такі програми часто більше спрямовані на розвагу, а не на навчання, і надто часто мають невеликий навчальний вплив. Рідше вони допомагають розвивати логічне мислення.

Використання ПЗ навчального призначення різних типів може бути доцільним в навчальному процесі, особливо якщо орієнтуватися на такі аспекти:

Застосування програмних засобів підвищує мотивацію навчання, оскільки дозволяє учням вибирати режим роботи з ПЗ, виконувати різноманітні завдання самостійно, візуалізувати матеріал і навіть користуватися ігровими сценаріями.

Використання сучасної комп'ютерної графіки та інших засобів наочності сприяє розвитку наочно-образного мислення.

Організація дослідницької роботи з використанням ПЗ сприяє розвитку дослідницьких навичок та навчає самостійно здобувати знання.

Ідеї алгоритмізації навчання в ПЗ можуть розвивати алгоритмічний та логічний стиль мислення.

Використання навчальних баз даних, електронних таблиць та інших інструментів допомагає формувати інформаційну культуру учнів.

Таким чином, використання ПЗ в навчанні може бути дієвим засобом для покращення процесу навчання та розвитку учнів.

Отже, слід підкреслити важливість виокремлення ключових психолого-дидактичних аспектів при розробці перспективних ПЗ для навчання та розвитку інтелектуального потенціалу особистості. Це включає в себе аналіз умов використання сучасних інформаційних технологій, особливо у відповідність з методичними потребами.[5]

Порушимо тепер питання типології програмних засобів, що використовуються в навчальних цілях з різних функціональних та методичних аспектів. Розподіл програмних засобів за функціональним призначенням може включати такі категорії. Перша з них - це прикладні програми, спрямовані на створення та підтримку навчального діалогу користувача з комп'ютером. Зазвичай їх визначають як педагогічні програмні засоби (ППЗ). Головна мета ППЗ полягає в наданні навчальної інформації та у керуванні навчанням, зокрема враховуючи індивідуальні можливості та вподобання учнів. Зазвичай ППЗ надають зворотний зв'язок користувача з програмою, щоб забезпечити ефективне засвоєння нової інформації.

Системи та програмні засоби для оцінки навченості учнів, їх знань, навичок та інтелектуального розвитку стають все більш важливими. Інструментальні програмні засоби (ІПЗ) призначені для використання у практиці навчання можна поділити на наступні категорії:

- Інструментальні системи, спрямовані на розробку автоматизованих засобів або систем для контролю, консультування та тренування. Вони дозволяють замінити традиційні "паперові" методи подачі навчального матеріалу на електронний формат.

- Авторські програмні системи, призначені для створення програмного забезпечення для навчання.
- Системи комп'ютерного моделювання, які можуть демонструвати або імітувати різні процеси та явища.

- Програмні засоби, які містять як предметні середовища, так і елементи педагогічної технології для навчання.
- Інструментальні програмні засоби, які допомагають систематизувати навчальну інформацію на основі обробки даних, такі як інформаційно-пошукові системи та навчальні бази даних.
- Експертні системи для навчання, які допомагають користувачам отримувати знання та розв'язувати навчальні завдання, подаючи інформацію у формі, зрозумілій для учнів.

Предметно-орієнтовані програмні середовища надають можливість створювати моделі досліджуваних об'єктів або їхніх відношень в рамках конкретної предметної галузі, не обов'язково точно відображаючи об'єктивну реальність. Зазвичай через них організовується навчальна діяльність, використовуючи моделі, які відображають об'єкти та закономірності певної предметної області.

Програмні засоби, спрямовані на формування навчальної та інформаційної культури, використовують системи підготовки текстів, електронні таблиці, графічні та музичні редактори або інтегровані системи для комплексного використання цих інструментів. Такі програмні засоби автоматизують обробку результатів навчального експерименту, включаючи вимірювання і контроль за процесами, що відбуваються, і реєстрацію цієї інформації.

Керуючі програмні засоби призначені для управління діями реальних об'єктів.[6]

Навчальні середовища для програмування створені для початкового навчання навичкам програмування та розвитку основних компонентів алгоритмічного та програмістського стилю мислення.

Також існують програмні засоби, які допомагають викладачам виконувати певні функції.

Ці програмні засоби зазвичай генерують команди, які відносяться до взаємодії з комп'ютером, вказівки щодо припинення роботи, перевірки,

необхідних модифікацій даних, отримання додаткових відомостей та обговорення ходу роботи.

Сервісні програмні засоби спрямовані на полегшення роботи користувача, автоматизацію контролю за результатами навчання, генерацію та розсилання організаційно-методичних матеріалів, завантаження та передачу програмного забезпечення через мережу, а також управління навчальним процесом.

Ігрові програмні засоби використовуються для створення ігрових та навчально-ігрових активностей.[7]

Крім цієї типології програмних засобів за їхнім функціональним призначенням, існує також і доцільна типологія за методичним призначенням. Методичне призначення кожного типу програмних засобів відображає методичні цілі їх використання в навчальному процесі та можливості, які сприяють підвищенню якості навчання і переводять його на більш високий рівень.

Типологія програмних засобів з методичним призначенням може бути подана наступним чином:

- Навчальні програми - їх головною метою є передача знань та розвиток вмінь та навичок, забезпечуючи необхідний рівень засвоєння через взаємодію з програмою.
- Тренажери - призначені для відпрацювання навчальних вмінь та навичок, зазвичай використовуються для закріплення раніше вивченого матеріалу.
- Контролюючі програми - призначені для контролю та самоконтролю рівня оволодіння навчальним матеріалом.
- Інформаційно-пошукові програми - надають можливість користувачу вибирати та виводити необхідну інформацію, сприяючи розвитку навичок систематизації даних.

- Імітаційні програми - моделюють аспекти реальності для вивчення їх структурних або функціональних характеристик з обмеженим числом параметрів.
- Моделюючі програми - надають основні елементи та функції для створення моделі реальних або віртуальних об'єктів для вивчення та дослідження.
- Демонстраційні програми - надають візуалізацію навчального матеріалу та демонструють явища та процеси між об'єктами.
- Навчально-ігрові програми - використовуються для створення навчальних ситуацій.
- Дозвільні програми - організовують навчально-ігрову та просто ігрову діяльність для розвитку уваги, реакції, пам'яті та інших навичок учнів.

Аналіз досвіду створення та використання програмного забезпечення для навчання свідчить, що вони часто мають комбінований методичний спрямованість. Отже, розумна ініціатива включає в себе розробку програмного забезпечення для навчання, яке може реалізовувати комплекс методичних завдань.

1.3 Постановка задачі і вимоги до ПСНП

Вимоги до змісту ПСНП.

Основні теми та питання:

- Постановка задачі побудови опуклої лінійної оболонки точкової множини на площині
- Прості алгоритми побудови опуклої лінійної оболонки точкової множини на площині.
- Ефективні алгоритми побудови опуклої лінійної оболонки точкової множини на площині.

- Постановка задачі розбиття площини за Г.Вороним та задачі оптимальної триангуляції точкової множини за Делоне. Зв'язок між цими задачами
- Ефективний алгоритм задачі розбиття площини за Г.Вороним
- Задача пошуку мінімальної відстані між парами точок на площині та єфективний лгоритм її розв'язання
- Задача пошуку діаметра точкової множини на площині та єфективний лгоритм її розв'язання
- Задача розв'язання систем M лінійних нерівностей з N невідомими
- Алгоритм Фур'є-Моцкіна розв'язання систем лінійних нерівностей
- Метод трапецоідів розв'язання систем лінійних нерівностей
- Бібліотека класів реалізації задач обчислювальної геометрії (об'єктно-орікєнтований підхід)

Вимоги до складу ПСНП

- Лекції за темами ПСНП – повні тексти лецій оформлені у ПДФ у вигляді посібника зі Змістом та перехрестними посиланнями
- Бібліотека класів (C++) або бібліотека Python

Вимоги до реалізації ПСНП

- Локальна програмна система
- Зручний інтерфейс користувача
- Можливість авторизованого редагування змісту

РОЗДІЛ 2 ТЕОРЕТИЧНІ АСПЕКТИ ОБЧИСЛЮВАЛЬНОЇ ГЕОМЕТРІЇ

2.1. Задачі побудови опуклої лінійної оболонки

Побудова опуклої лінійної оболонки є важливою задачею в обчислювальній геометрії та має багато застосувань у різних галузях, включаючи комп'ютерну графіку, геодезію, маршрутизацію, проектування логістичних маршрутів та багато інших. Задача полягає в побудові найменшого опуклого многокутника (опуклої лінійної оболонки), який містить усі задані точки. Опуклий многокутник - це многокутник, всі внутрішні кути якого менше 180 градусів.[8]

Основні задачі та варіації побудови опуклої лінійної оболонки включають наступне:

- Постановка задачі: Задача полягає в знаходженні опуклої лінійної оболонки для даного набору точок на площині.
- Алгоритми побудови:
    - Алгоритм "Замітки Грема": Вибір точки з найнижчою y-координатою та сортування інших точок за їхніми кутами відносно цієї точки.
    - Алгоритм "Джарвіса" (або "Gift Wrapping"): Вибір будь-якої точки та обертання лінії до найближчої точки, вибору точки з найбільшим кутом, і так далі, поки не повернеться до початкової точки.
    - Алгоритм Керка (Quickhull): Використовує поділ і завоювання для пошуку вершин опуклої лінійної оболонки.
- Робота в реальному часі: Розв'язки для великих масивів точок, де важлива швидкість виконання.
- Задача зі зменшеною розмірністю: Робота з точками у високорозмірних просторах та побудова опуклої лінійної оболонки у зменшеній розмірності.

- Паралельні алгоритми: Розв'язки для паралельного обчислення опуклої лінійної оболонки на багатьох процесорах чи ядрах.
- Подальший аналіз оболонки: Визначення властивостей опуклої лінійної оболонки, таких як довжина, площа, периметр і інші.
- Застосування в геоінформатиці: Визначення опуклої лінійної оболонки для геоданих та географічних інформаційних систем.

Задача побудови опуклої лінійної оболонки має практичне значення в багатьох сферах, і для її вирішення розроблені різні алгоритми, які вибираються в залежності від конкретних умов і обмежень завдання.

2.1.1. Постановка задачі побудови опуклої лінійної оболонки точкової множини на площині

Постановка задачі побудови опуклої лінійної оболонки точкової множини на площині може бути сформульована наступним чином [9]:

Вхід: Маємо задану точкову множину P, яка складається з N точок на площині. Кожна точка в множині P представлена координатами (x, y).

Вихід: Потрібно знайти опуклу лінійну оболонку точкової множини P, яка є найменшим опуклим многокутником, який містить усі точки з множини P.

Задача полягає в знаходженні границь цього опуклого многокутника, який буде обмежувати точки з множини P. Опуклий многокутник - це многокутник, в якому всі внутрішні кути менше 180 градусів, тобто всі його сторони не перетинаються.

Існують різні алгоритми для вирішення цієї задачі, і вони можуть бути вибрані в залежності від конкретних вимог та умов задачі. Задача побудови опуклої лінійної оболонки має широкі застосування в різних галузях, включаючи комп'ютерну графіку, геодезію, робототехніку, транспортну логістику та багато інших.

2.1.2. Прості алгоритми побудови опуклої лінійної оболонки точкової множини на площині.

Існують декілька простих алгоритмів для побудови опуклої лінійної оболонки точкової множини на площині. Розглянемо два з найпоширеніших і найпростіших алгоритмів:

- Алгоритм "Замітки Грема" (Graham's Scan)
- Алгоритм "Джарвіса" (Jarvis March або Gift Wrapping)

Алгоритм "Замітки Грема" (Graham's Scan) - це алгоритм для побудови опуклої оболонки точкової множини на площині. Опукла оболонка - це найменша опукла фігура, яка включає всі точки даної множини[10].

Основні кроки алгоритму "Замітки Грема":

- Знаходження точки з найменшою y-координатою: З точок множини обирається точка P, яка має найменшу y-координату (у випадку рівних y-координат - точка з найменшою x-координатою). Ця точка стає початковою точкою опуклої оболонки.
- Сортування за кутом відносно точки P: Усі інші точки сортуються за кутом, який вони утворюють відносно точки P. Для цього можна використовувати тангенс кута між відмітною точкою і точкою P. Точки сортуються за збільшенням кута.
- Побудова опуклої оболонки за допомогою стеку: Створюється порожній стек, в який будуть додаватися точки опуклої оболонки в порядку обходу. Перші дві точки з відсортованого списку точок додаються в стек.
- Пошук опуклої оболонки: Починаючи з третьої точки відсортованого списку, виконується ітерація по точкам. Для кожної точки, доки вона утворює неопуклу фігуру разом з верхніми двома точками в стеці, відстранюються верхні точки зі стеку. Потім поточна точка додається в стек.
- Завершення побудови опуклої оболонки: По закінченні обходу всіх точок множини опукла оболонка знаходиться у стеці.

Результатом виконання алгоритму є опукла оболонка, яка може бути подана у вигляді послідовності вершин, які визначають межі оболонки. Алгоритм "Замітки Грема" досить ефективний і широко використовується в комп'ютерній графіці, обробці зображень, геоінформатиці та інших галузях для обчислення опуклих оболонок.

Алгоритм Джарвіса (іноді відомий як алгоритм обгортання Джарвіса або алгоритм обгортання ближньої точки) - це інший числовий метод для побудови опуклої оболонки точкової множини на площині. Цей алгоритм досить простий для реалізації і легко зрозуміти [11].

Основні кроки алгоритму Джарвіса:

- Знаходження точки з найменшою у-координатою (і найменшою х-координатою в разі рівних у-координат): З точок множини обирається початкова точка P0, яка має найменшу у-координату (у випадку рівних у-координат - точка з найменшою х-координатою). Ця точка стає початковою точкою опуклої оболонки.
- Пошук наступної точки оболонки: Починаючи з початкової точки P0, обирається наступна точка P1. Для цього здійснюється обхід всіх інших точок множини, і обирається точка, яка утворює найменший кут з поточною точкою та точкою P0. Ця точка стає наступною точкою оболонки.
- Повторення: Процес пошуку наступної точки повторюється до тих пір, поки не буде досягнуто початкової точки P0, утворюючи таким чином замкнутий оболонку.
- Завершення побудови опуклої оболонки: По закінченні обходу всіх точок множини отримується опукла оболонка у вигляді послідовності вершин, які визначають межі оболонки.

Алгоритм Джарвіса є простим і легко зрозумілим методом для побудови опуклої оболонки, але в порівнянні з іншими алгоритмами, такими як алгоритм "Замітки Грема", він може бути менш ефективним для великих множин точок через свою квадратичну складність за кількістю точок. Але він все ж

залишається корисним для невеликих множин точок та для навчання цих алгоритмів [12].

Ці алгоритми є досить простими для реалізації і працюють для опуклих оболонок, які не мають перетинів. Однак вони можуть бути неефективними для великих множин точок, оскільки мають часову складність O(N^2), де N - кількість точок у вихідній множині. Для великих множин точок рекомендується використовувати більш ефективні алгоритми, такі як алгоритм Керка (Quickhull) або інші.

2.1.3. Ефективні алгоритми побудови опуклої лінійної оболонки точкової множини на площині.

Існують більш ефективні алгоритми для побудови опуклої лінійної оболонки точкової множини на площині, які працюють швидше, ніж прості алгоритми, особливо для великих множин точок. Наведемо кілька з них:

- Алгоритм Керка (Quickhull)
- Алгоритм сканування від нижнього до верхнього обгортання (Lower Envelope Scan)
- Алгоритм Гіфта (Gift Wrapping або Jarvis March):

Алгоритм Керка (Quickhull) - це ефективний числовий метод для побудови опуклої оболонки точкової множини на площині. Цей алгоритм базується на розділенні простору за допомогою опорних точок і дозволяє знаходити опуклу оболонку в середньому за лінійний час (O(n * log(n))) для випадкових точок.

Основні кроки алгоритму Керка [13]:

• Знаходження екстремальних точок: З множини точок обираються точки з найбільшими і найменшими значеннями x-координати, які будуть розглядатися як крайні точки опуклої оболонки.

• Побудова опуклої оболонки за допомогою рекурсії: Алгоритм починається зі створення опуклої оболонки для двох крайніх точок. Спочатку

будується відрізок, який з'єднує ці дві точки. Потім знаходяться всі точки, які лежать ззовні цього відрізка і розділяються на два підсписки: ті, що лежать зліва від відрізка і ті, що лежать справа від відрізка. Для кожного з цих підсписків рекурсивно будується опукла оболонка.

- Завершення побудови опуклої оболонки: Після закінчення всіх рекурсивних викликів опукла оболонка будується шляхом об'єднання опуклих оболонок, які були побудовані для лівої і правої частини від відрізка між крайніми точками.

Алгоритм Керка є ефективним методом для побудови опуклої оболонки, і він часто використовується в комп'ютерній графіці, обробці зображень, геоінформатиці та інших галузях, де важлива оптимізація часу виконання. Він також відомий своєю ефективністю навіть для великих множин точок [14].

Алгоритм сканування від нижнього до верхнього обгортання (Bottom-Up Convex Hull) - це числовий метод для побудови опуклої оболонки точкової множини на площині. Цей алгоритм був розроблений для обчислення опуклої оболонки за допомогою сканування точок знизу вгору.

Основні кроки алгоритму сканування від нижнього до верхнього обгортання [15]:

- Сортування точок за y-координатою: Спочатку всі точки множини сортуються за зростанням y-координати (у випадку рівних y-координат - за зростанням x-координати). Це допомагає забезпечити, що точки внизу множини будуть першими в обробці.
- Початкове наближення: Два перших точки з відсортованого списку стають першими двома точками опуклої оболонки.
- Сканування від нижнього до верхнього: Виконується сканування точок з третьої точки до останньої. Для кожної точки визначається, чи вона лежить в опуклій оболонці, використовуючи властивість опуклої оболонки, яка полягає в тому, що для кожної точки опуклої оболонки всі інші точки оболонки лежать справа від відрізка, який

об'єднує поточну точку і точку зверху стеку (початкову точку опуклої оболонки).

- Оновлення опуклої оболонки: Якщо поточна точка лежить в опуклій оболонці, вона додається до оболонки, і ця точка стає новою точкою зверху стеку. Якщо точка не лежить в опуклій оболонці, вона відкидається.
- Завершення побудови опуклої оболонки: По закінченні сканування отримується опукла оболонка у вигляді послідовності вершин, які визначають межі оболонки.

Алгоритм сканування від нижнього до верхнього обгортання - це один із способів побудови опуклої оболонки, який зазвичай є швидким і простим для реалізації. Він добре працює для великих множин точок та залишається популярним методом в комп'ютерній графіці, обробці зображень, геоінформатиці та інших застосуваннях.

Алгоритм Гіфта (Gift Wrapping), також відомий як алгоритм обгортання Джарвіса (Jarvis March), є числовим методом для побудови опуклої оболонки точкової множини на площині. Цей алгоритм отримав свою назву через аналогію з процесом запаковування подарунка [16].

Основні кроки алгоритму Гіфта:

- Знаходження точки з найменшою у-координатою (і найменшою х-координатою в разі рівних у-координат): З множини точок обирається початкова точка P0, яка має найменшу у-координату (у випадку рівних у-координат - точка з найменшою х-координатою). Ця точка стає початковою точкою опуклої оболонки.
- Пошук наступної точки оболонки: Починаючи з початкової точки P0, знаходиться наступна точка P1, яка утворює найменший кут з поточною точкою відносно всіх інших точок. Це робиться шляхом обходу всіх інших точок множини та обирання тієї, яка утворює мінімальний кут.

- Оновлення опуклої оболонки: Точка P1 додається до опуклої оболонки, і вона стає новою поточною точкою. Процес пошуку наступної точки повторюється до тих пір, поки не буде досягнуто початкової точки P0, утворюючи таким чином замкнутий оболонку.
- Завершення побудови опуклої оболонки: По закінченні обходу отримується опукла оболонка у вигляді послідовності вершин, які визначають межі оболонки.

Алгоритм Гіфта є простим для реалізації і добре працює для невеликих множин точок. Однак він може бути менш ефективним для великих множин точок порівняно з іншими алгоритмами, такими як алгоритм Керка (Quickhull). Але він все ж залишається корисним для навчання та для обробки невеликих наборів точок [17].

Ці алгоритми мають кращу часову складність і ефективно працюють для великих множин точок. Вибір конкретного алгоритму залежить від ваших вимог та обмежень, а також від вже наявних даних.

2.2 Задачі розбиття площини за Г.Вороним

Задачі розбиття площини за алгоритмом Г. Вороного включають в себе розподіл площини на клітини (так звані Вороного клітини або регіони Вороного), де кожна клітина відповідає конкретній точці у просторі, відомій як точка Вороного. Кожна Вороного клітина містить всі точки, які найближчі до відповідної точки Вороного в порівнянні з іншими точками [18].

Наведемо деякі задачі, пов'язані з розбиттям площини за алгоритмом Г. Вороного:

- Розташування найближчого сусіда: Задача полягає в знаходженні найближчого сусіда для кожної точки в множині точок. Кожна Вороного клітина буде містити точки, які найближчі до відповідної точки Вороного.

- Геометричні оболонки: Вороного клітини можуть бути використані для знаходження геометричних оболонок (наприклад, мінімального опуклого багатокутника), що містить всі точки в множині.
- Планування мережі: Вороного діаграми можуть бути використані для планування мережі транспорту, включаючи розташування розвилок доріг або інфраструктури.
- Розташування репрезентаційних об'єктів: Вороного діаграми можуть використовуватися для розташування репрезентаційних об'єктів, які обслуговують певну область (наприклад, лікарень, поліцейських дільниць тощо).
- Сегментація зображень: Вороного діаграми можуть використовуватися для сегментації зображень, де кожен піксель належить до Вороного клітини, яка представляє певний об'єкт або регіон на зображенні.
- Віддаленість до ближніх сусідів: Вороного діаграми дозволяють визначити середню відстань від кожної точки до її найближчого сусіда.
- Обробка геоданих: Вороного діаграми допомагають при аналізі геоданих, включаючи розподіл даних на регіони або клітини.

Це лише деякі з прикладів задач, які можна вирішити за допомогою алгоритму Вороного. Цей підхід має широкий спектр застосувань у різних галузях, де необхідно аналізувати просторові дані та робити рішення на основі розташування точок.

2.2.1 Постановка задачі розбиття площини за Г.Вороним та задачі оптимальної триангуляції точкової множини за Делоне. Зв’язок між цими задачами

Задачі розбиття площини за Г. Вороним та задачі оптимальної триангуляції точкової множини за Делоне є двома різними задачами, але вони пов'язані між собою через поняття Вороного діаграми та триангуляції Делоне. Розглянемо постановку задач та їх відмінності.

Задача розбиття площини за Г. Вороним (Voronoi Diagram) - це математична концепція і метод, який використовується для поділу площини на області, такі, що кожна область має визначений центр (точку) і включає всі точки площини, які найближчі до центру цієї області. Задачу також називають діаграмою Вороного або Вороного триангуляцією [19].

Основна ідея полягає в розбитті площини на області таким чином, щоб кожна точка у площині належала лише одній з цих областей, і ця область об'єднувала всі точки, які найближчі до свого центру. Вороного діаграма широко використовується в різних галузях, включаючи геометрію, науку про матеріали, графіку, обробку зображень, геоінформатику та інші області.

Задачу Вороного можна вирішувати алгоритмами, які будують діаграму Вороного з використанням властивостей точок, розташованих на площині. Основна ідея полягає в обчисленні відстаней від кожної точки до потенційних центрів Вороного-областей і визначенні, які точки належать до якої області. Вороного діаграми мають багато застосувань, включаючи створення алгоритмів обробки даних, аналізу геопросторових даних, анімації та багато інших областей [20].

Отже задача розбиття площини за Г. Вороним (Voronoi Diagram) складається з наступних даних:

Вхід: Маємо точкову множину P на площині.

Вихід: Потрібно розбити площину на області (регіони Вороного), такі, що кожна область включає всі точки, які найближчі до конкретної точки Вороного.

Вороного діаграма розбиває площину на клітини, де кожна клітина належить до одного з регіонів Вороного, і містить всі точки, які найближчі до відповідної точки Вороного. Вороного діаграма - це декомпозиція площини на клітини, де кожна клітина призначена певній точці вихідної множини.

Задача оптимальної триангуляції точкової множини за Делоне (Delaunay Triangulation) - це математична задача триангуляції площини, що визначає оптимальний спосіб розбиття точкової множини на трикутники так, щоб кожний трикутник задовольняв "критерію Делоне". Цей критерій вимагає, щоб жодній точці не було б ближче двох інших точок, які належать до інших трикутників [21].

Така триангуляція має безліч застосувань у геометрії, обчислювальній геометрії, графіці, геоінформатиці, комп'ютерному зору та інших областях. Оптимальна триангуляція за Делоне може бути важливою, наприклад, для обчислення найближчих сусідів для кожної точки в множині, для створення триангуляційних мереж для сіток скінченних елементів в чисельних методах, для обробки зображень та відео, для анімації та багатьох інших завдань.

Для побудови оптимальної триангуляції за Делоне, існують різні алгоритми, включаючи ітеративні методи, рандомізовані алгоритми, алгоритми на основі призм та інші. Задача полягає в знаходженні такого набору трикутників, що задовольняють критерію Делоне для вказаних точок, і вона часто вирішується з використанням комп'ютерних програм та бібліотек для геометричних обчислень [22].

Отже задача оптимальної триангуляції точкової множини за Делоне (Delaunay Triangulation) складається з наступних даних:

Вхід: Маємо точкову множину P на площині.

Вихід: Потрібно побудувати трикутну мережу, таку, що жодні точки з множини P не лежать всередині об'ємлючого кола жодного трикутника.

Триангуляція Делоне - це така трикутна мережа, в якій кожен трикутник виконує умову Делоне, яка визначається тим, що жодні точки з множини P не можуть лежати всередині об'ємлючого кола жодного трикутника.

Зв'язок між цими задачами:

Вороного діаграма і тріангуляція Делоне взаємно пов'язані. Це означає, що інформація, отримана при побудові однієї з цих структур, дозволяє побудувати іншу. Зокрема, для будь-якої Вороного діаграми існує тріангуляція Делоне, де центри об'ємлюючих кол можуть бути з'єднані ребрами трикутників. Зворотно, для будь-якої тріангуляції Делоне існує Вороного діаграма, в якій границі областей Вороного визначаються ребрами трикутників [23].

Таким чином, Вороного діаграма і тріангуляція Делоне є двома взаємно-двозначними структурами, які використовуються для аналізу та обробки точкових даних на площині, і вони доповнюють один одного при вирішенні різних завдань у галузях, таких як комп'ютерна графіка, обробка зображень, геоінформатика, геодезія, транспортна логістика і багато інших.

2.2.2. Ефективний алгоритм задачі розбиття площини за Г.Вороним

Ефективний алгоритм для задачі розбиття площини за Г. Вороним, відомий як "Алгоритм Форчуна" (Fortune's Algorithm), базується на поділі і завоюванні та використовує структуру даних, відому як "список подій" (event queue). Цей алгоритм дозволяє побудувати Вороного діаграму з часовою складністю O(n log n), де n - кількість точок у вихідній множині [24].

Основні кроки алгоритму Форчуна для задачі розбиття площини за Г. Вороним такі:

- Ініціалізація: Всі точки відсортовуються за їхніми координатами x, і список подій створюється на основі цих точок.
- Обробка подій від лівої до правої: Алгоритм обходить список подій від лівої до правої, розглядаючи кожну подію. Події можуть бути точками, ребрами або перетинами ребер.
- Структура даних "Параболічні лінії" (Beach Line): Алгоритм використовує структуру даних, відому як "Параболічні лінії" або

"Beach Line", для представлення границь Вороного клітин. Ця структура допомагає визначати, які ребра Вороного мають бути додані на кожному кроці обробки подій.

- Завоювання та поділ: Під час обробки кожної події алгоритм визначає, які клітини Вороного створюються чи змінюються. Це включає в себе обробку нових точок, видалення "завоюваних" ребер та додавання нових ребер.
- Побудова границь клітин: Вороного діаграма побудована під час обробки всіх подій, і границі клітин Вороного визначаються на основі створеної структури "Параболічних ліній" та ребер, які були додані.

Алгоритм Форчуна є одним з найшвидших та найефективніших алгоритмів для задачі розбиття площини за Г. Вороним, і він знаходить застосування в різних областях, включаючи комп'ютерну графіку, обробку зображень, геоінформатику, логістику та багато інших.

2.2.3. Задача пошуку мінімальної відстані між парами точок на площині та єфективний лгоритм її розв’язання

Задача пошуку мінімальної відстані між парами точок на площині відома як "задача найближчого сусіда" (Closest Pair Problem). Основна мета полягає в тому, щоб знайти пару точок з набору, які мають мінімальну відстань між собою.

Один із ефективних алгоритмів для розв'язання цієї задачі - це "алгоритм розділення та завоювання". Наведемо кроки цього алгоритму [25]:

- Сортування за координатою x: Спочатку всі точки сортуються за їхніми координатами x.
- Розділення плоскості: Плоскість розділяється на дві рівні частини, і центральна вертикальна лінія розділення ділить точки на ліву та праву підмножини.

- Рекурсивний пошук: Рекурсивно обчислюється мінімальна відстань в лівій та правій підмножинах.
- Мінімальна відстань на границі: Обчислюється мінімальна відстань між парами точок, де одна точка лежить в лівій підмножині, а інша - в правій підмножині.
- Знаходження найменшої відстані: Мінімальна відстань серед мінімальних відстаней з кроку 3 і кроку 4 є відповіддю на задачу.

Цей алгоритм має часову складність O(n log n), де n - кількість точок на площині. Він є дуже ефективним і використовується в різних застосуваннях, включаючи комп'ютерну графіку, геоінформатику, обробку зображень та багато інших областей, де важливо знаходити найближчі пари точок.

2.2.4.Задача пошуку діаметра точкової множини на площині та єфективний лгоритм її розв’язання

Задача пошуку діаметра точкової множини на площині полягає в знаходженні найбільшої можливої відстані між будь-якою парою точок з даної множини. Діаметр визначається як максимальна відстань між будь-якою парою точок [26].

Ефективний алгоритм для розв'язання цієї задачі використовує підхід поділу та завоювання (divide and conquer). Ось кроки цього алгоритму:

- Сортування точок за координатами x: Спочатку всі точки сортуються за їхніми координатами x. Це допомагає при розділенні множини на дві підмножини.
- Рекурсивне розділення: Множина точок розділяється на дві рівні підмножини, де ліва множина містить точки з лівої половини відсортованого списку, а права множина - точки з правої половини.
- Знаходження діаметра лівої та правої підмножин: Рекурсивно знаходяться діаметри лівої та правої підмножин, а також найменший x та найбільший x координати точок в правій та лівій підмножині відповідно.

● Знаходження "середнього" діаметра: Діаметр між точками лівої підмножини та точками правої підмножини має бути такий, що точка з лівої підмножини має найбільшу x координату в правій підмножині, і точка з правої підмножини має найменшу x координату в лівій підмножині. Цей діаметр знаходиться шляхом обчислення відстаней між цими "середніми" точками.

● Знаходження максимального діаметра: Максимальний діаметр обчислюється шляхом порівняння діаметрів, знайдених на кроках 3 та 4.

Цей алгоритм має часову складність O(n log n), де n - кількість точок у вихідній множині. Це ефективний спосіб знаходження діаметра точкової множини на площині і знаходить застосування в різних областях, включаючи комп'ютерну графіку, обробку зображень, геоінформатику та інші галузі, де важливо визначити максимальну відстань між точками.

2.3. Розв’язання систем M лінійних нерівностей

Для розв'язання системи M лінійних нерівностей, використовуються різні методи, включаючи метод симплексу, метод ітераційного покращення, метод перебору та інші. Основна мета полягає в знаходженні такого вектора змінних, який задовольняє всі нерівності системи. Тут я надам загальний опис методу симплексу, який є одним із найпоширеніших методів для розв'язання цих систем.

Метод симплексу:

● Початкове наближення: Спочатку потрібно мати початкове наближення вектора змінних, яке задовольняє систему лінійних нерівностей. Зазвичай, початкове наближення встановлюється як будь-який допустимий вектор.

● Відбір базису: Вибирається початковий базис (підмножина індексів змінних), інші змінні вважаються неосновними. Початковий базис обирається так, щоб відповідні основні змінні були рівні відомим значенням.

- Знаходження опорного рішення: Використовуючи базис та нерівності, знаходиться опорне рішення задачі лінійного програмування для заданої системи.
- Перевірка на оптимальність: Перевіряється, чи є поточне опорне рішення оптимальним. Якщо так, то рішення знайдено.
- Покращення: Якщо поточне опорне рішення не є оптимальним, то відбувається пошук напрямку покращення шляхом зміни неосновних змінних.
- Аналіз на нескінченність або недопустимість: Проводиться аналіз, чи є задача безкінечною або недопустимою. Якщо так, то розв'язок відсутній або недопустимий.
- Оновлення базису та повернення до кроку 3: Якщо поточне рішення ще не є оптимальним, базис оновлюється, і процес повертається до кроку 3 для пошуку нового опорного рішення.

Цей процес повторюється до досягнення оптимального розв'язку або визначення недопустимості або безкінечності задачі. Метод симплексу є дуже ефективним для багатьох лінійних задач і використовується в оптимізації та оптимальному управлінні в різних областях.

2.3.1 Задача розв’язання систем M лінійних нерівностей з N невідомими

Задача розв'язання системи M лінійних нерівностей з N невідомими полягає в знаходженні такого вектора змінних x = [x_1, x_2, ..., x_N], який задовольняє системі нерівностей. Система має вигляд:

$$A * x \leq b$$

де:

A - матриця розміру MxN, що визначає коефіцієнти лівої частини нерівностей.

x - вектор невідомих змінних розміру Nx1.

b - вектор правої частини нерівностей розміру Mx1.

Для розв'язання цієї задачі використовуються різні методи. Один із найпоширеніших способів - це метод симплексу, який був вже згаданий в попередньому відповіді. Однак існують інші методи та бібліотеки для розв'язання цих систем. Ось загальний опис алгоритму для розв'язання системи лінійних нерівностей:

Початкове наближення: Зазвичай, для вирішення цієї задачі потрібно мати початкове наближення вектора x, яке задовольняє системі нерівностей.

Обирання методу: Вибирається метод розв'язання системи. Один з найпоширеніших методів - це метод симплексу. Інші методи включають методи ітераційного покращення та методи внутрішньої точки.

Знаходження розв'язку: Використовуючи обраний метод, знаходиться розв'язок системи. Це може включати в себе ітерації та оптимізацію.

Перевірка оптимальності та допустимості: Після знаходження розв'язку проводиться перевірка, чи є розв'язок оптимальним та допустимим. Якщо так, то рішення знайдено.

Аналіз результату: Якщо розв'язок не є оптимальним або допустимим, виконуються подальші ітерації та корекції, щоб знайти оптимальний розв'язок або визначити недопустимість задачі.

Важливо відзначити, що в реальних задачах роз'язування систем лінійних нерівностей часто використовуються оптимізовані бібліотеки та програми для лінійного програмування, такі як CPLEX, Gurobi, SciPy, та інші, які включають в себе різні методи та оптимізації для ефективного розв'язання таких задач.

2.3.2 Алгоритм Фур’є-Моцкіна розв’язання систем лінійних нерівностей

Алгоритм Фур'є-Моцкіна (також відомий як "метод Фур'є-Моцкіна") - це числовий метод для розв'язання системи лінійних нерівностей з обмеженнями. Цей метод може використовуватися для знаходження найбільшого можливого

значення цільової функції (яка представлена лінійною комбінацією змінних) за умовами задачі лінійного програмування з обмеженнями.

Основні кроки алгоритму Фур'є-Моцкіна такі:

- Початкове наближення: Початкове наближення до розв'язку задачі задається у вигляді допустимої точки в гіперкубі (багатовимірному кубі), який обмежує область значень змінних рішення.
- Центральний путь: Алгоритм прослідковує центральний путь в гіперкубі, намагаючись знайти найбільше можливе значення цільової функції за умовами обмежень. Центральний путь - це послідовність точок, які рухаються в напрямку оптимального розв'язку задачі.
- Зменшення кроку: Зазвичай алгоритм використовує правило зменшення кроку для наближення до оптимального розв'язку. Крок зменшується з кожною ітерацією, і алгоритм намагається збільшити точність розв'язку.
- Зупинка за умови: Алгоритм продовжує виконуватися до тих пір, поки не буде досягнута певна умова зупинки. Це може бути, наприклад, досягнення заданої точності розв'язку або досягнення заданої кількості ітерацій.
- Результати: Після закінчення алгоритму отримується оптимальний розв'язок задачі лінійного програмування з обмеженнями.

Алгоритм Фур'є-Моцкіна є досить потужним та загальним методом для розв'язання задач лінійного програмування з обмеженнями. Він може застосовуватися до широкого спектра задач оптимізації та використовується в багатьох галузях, включаючи економіку, інженерію, логістику та інші.

2.3.3 Метод трапецоідів розв’язання систем лінійних нерівностей

Метод трапецоїдів (також відомий як метод змішаного цілочисельного програмування) є числовим методом для розв'язання систем лінійних нерівностей, який враховує цілісність змінних. Цей метод застосовується до

задач оптимізації з цілочисельними обмеженнями і може бути корисним в таких випадках, коли потрібно знайти цілочисельний розв'язок для системи лінійних нерівностей.

Основні кроки методу трапецоїдів на високому рівні такі:

- Початкове наближення: Починається з початкового наближення до розв'язку задачі.
- Розв'язання лінійної задачі оптимізації: Спочатку розв'язується лінійна задача оптимізації, де змінні необов'язково цілісні. Ця задача допомагає встановити нижню та верхню межі для цілочисельних змінних.
- Змішана цілочисельна програмація: Після розв'язання лінійної задачі оптимізації використовується метод цілочисельного програмування для знаходження найкращого цілочисельного розв'язку, враховуючи нижні та верхні межі для змінних.
- Аналіз розв'язку: Після знаходження цілочисельного розв'язку аналізується, чи він відповідає умовам системи лінійних нерівностей.
- Покращення наближення і повторення: Якщо розв'язок не відповідає умовам системи лінійних нерівностей, то наближення покращується, і процес повторюється.

Метод трапецоїдів допомагає знайти оптимальний цілочисельний розв'язок для системи лінійних нерівностей, що може бути важливим в багатьох практичних задачах. Однак важливо враховувати, що цей метод може бути обчислювально витратним, і час виконання може зростати з розміром задачі.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Вибір засобів і середовища розробки

Python і C++ - дві мови програмування зі своїми перевагами та недоліками. В цьому дописі ми розглянемо порівняння між Python та C++.

C++ - це компільована мова програмування, яка оптимізується компілятором і виконується безпосередньо машинним кодом. Це дозволяє їй працювати значно швидше, ніж Python. Python, натомість, є інтерпретованою мовою, яка виконується в режимі реального часу і вимагає більше часу для виконання.

Python - проста та легка в використанні мова програмування, яку можна використовувати для різних завдань, включаючи наукові обчислення, розробку веб-додатків та інше. C++, натомість, є більш складною мовою програмування, яку можна використовувати для розробки великих та складних програм.

C++ - це мова програмування, яка була створена для створення надійних та стабільних додатків. Вона забезпечує більш високу продуктивність та безпеку, ніж Python. Python також може бути надійною мовою програмування, але вона не має такого ступеня контролю за пам'яттю та продуктивністю.

C++ - це дуже розширювана мова програмування, яку можна використовувати для створення бібліотек і модулів, які можуть бути використані в інших додатках. Python також є розширюваною мовою програмування, але вона не має такого ж рівня гнучкості та контролю.

Загалом, Python і C++ - це дві різні мови програмування, кожна з яких має свої переваги та недоліки.

Оскільки додаток має бути легким та простим для написанням з використанням функцій PDF файлу ми будемо використовувати мову програмування Python. За допомогою даної мови програмування ми можемо додати нові функції в нашу програму.

3.2 Розробка бібліотек

Маючи визначеність з мовою програмування опишемо деякі бібліотеки які будуть використані при розробці програмного продукту а також опишемо створені бібліотеки які власне будуть виконувати функції по роботі з pdf файлом з предмету обчислювальної геометрії.

Бібліотека os, надає функції для взаємодії з операційною системою. Після імпорту os, можна використовувати ці функції для виконання різних операцій на рівні операційної системи, таких як робота з файлами та каталогами, керування процесами, отримання інформації про операційну систему тощо.

Ось декілька прикладів того, як використовується бібліотека os в проекті:

Отримання поточного робочого каталогу:

```
current_directory = os.getcwd()
print(current_directory)
```

Перевірка наявності файлу чи каталогу:

```
file_path = "шлях_до_файлу_або_каталогу"
exists = os.path.exists(file_path)
if exists:
    print(f"{file_path} існує.")
else:
    print(f"{file_path} не існує.")
```

Робота з каталогами:

```
# Створення нового каталогу
os.mkdir("новий_каталог")
# Перейменування каталогу
os.rename("старий_каталог", "новий_каталог")
# Видалення каталогу
os.rmdir("каталог_для_видалення")
```

Робота з файлами:

```
# Створення файлу і запис даних у нього
```

```
with open("новий_файл.txt", "w") as file:
    file.write("Це текст, який записаний у файл.")
# Перевірка наявності файлу
file_path = "шлях_до_файлу.txt"
if os.path.isfile(file_path):
    print(f"{file_path} - це файл.")
```

Отримання інформації про операційну систему:

```
os_name = os.name
print(f"Операційна система: {os_name}")
```

Це лише декілька прикладів можливого використання бібліотеки os, в проекті дана бібліотека використовується для сторінок «Про автора» а також опису основних функцій.

Бібліотека re, надає функціональність для роботи з регулярними виразами. Регулярні вирази - це потужний інструмент для пошуку та обробки текстових даних за допомогою паттернів. Вони дозволяють виконувати складний пошук, заміну та аналіз тексту.

Після імпорту re, ви можете використовувати різні функції для роботи з регулярними виразами. Ось декілька прикладів:

Пошук тексту, який відповідає певному регулярному виразу:

```
import re

text = "Це приклад тексту для пошуку."
pattern = r"пошуку"
match = re.search(pattern, text)
if match:
    print("Знайдено відповідну частину тексту.")
else:
    print("Не знайдено відповідності.")
```

Заміна тексту за допомогою регулярних виразів:

```
import re

text = "Це приклад тексту з числами 123 і 456."
pattern = r"\d+"
replacement = "999"
new_text = re.sub(pattern, replacement, text)
print(new_text)
```

Розбиття тексту на частини за допомогою регулярних виразів:

```
import re
text = "Це речення розділиться на слова."
words = re.split(r"\s", text)
print(words)
```

Бібліотека re дозволяє виконувати багато інших операцій з регулярними виразами, такі як пошук всіх входжень, знаходження груп відповідності, валідація тексту тощо. Вона корисна для обробки текстових даних та виконання різних завдань, пов'язаних з пошуком та аналізом тексту.

import tkinter - це спосіб імпортувати бібліотеку tkinter в Python. tkinter - це стандартна бібліотека інтерфейсу користувача (UI) для створення графічних програм та віконних додатків в Python.

Після імпортування tkinter, ви можете використовувати її класи та функції для створення графічних інтерфейсів, вікон, кнопок, текстових полів, міток і багатьох інших елементів для взаємодії з користувачем.

Ось простий приклад використання tkinter для створення вікна нашого проекту:

```
import tkinter as tk

# Створити вікно
root = tk.Tk()
root.title("Приклад tkinter")
```

```
# Додати мітку
label = tk.Label(root, text="Привіт, tkinter!")
label.pack()

# Запустити цикл обробки подій
root.mainloop()
```

У цьому прикладі ми імпортуємо tkinter під ім'ям tk, створюємо вікно (tk.Tk()), додаємо мітку (tk.Label()) та викликаємо метод mainloop() для початку циклу обробки подій.

Опишимо також наступний фрагмент коду:

```
from tkinter import PhotoImage
from tkinter import filedialog
from tkinter import messagebox
```

Цей фрагмент коду імпортує певні елементи з бібліотеки tkinter для роботи з графічним інтерфейсом користувача в Python.

- from tkinter import PhotoImage: Це імпорт класу PhotoImage з бібліотеки tkinter. PhotoImage використовується для завантаження та відображення растрових зображень, таких як GIF-зображення, у ваших програмах.
- from tkinter import filedialog: Це імпорт підмодуля filedialog з бібліотеки tkinter. filedialog надає функціонал для вибору файлів та директорій, відкриття та збереження файлів, що є корисним у графічних програмах.
- from tkinter import messagebox: Це імпорт підмодуля messagebox з бібліотеки tkinter. messagebox дозволяє створювати вікна з повідомленнями, попередженнями та підтвердженнями для спілкування з користувачем.

Ці імпорти дозволяють використовувати функції та класи, які надаються цими підмодулями, для створення більш складних графічних інтерфейсів та взаємодії з користувачем в застосунках, що використовують tkinter.

import fitz - це імпорт бібліотеки PyMuPDF (також відомої як Fitz) у вашому Python-коді. PyMuPDF - це бібліотека, яка надає можливість обробки та редагування PDF-документів в Python.

Зазвичай ви можете використовувати PyMuPDF для виконання різних завдань, пов'язаних з PDF-документами, таких як витягування тексту, зображень, маніпулювання сторінками, створення нових PDF-файлів і багато іншого.

Після імпорту fitz ви можете використовувати його функції та методи для роботи з PDF-документами у вашому коді.

import datetime - це імпорт модуля datetime у Python. Модуль datetime надає функціональність для роботи з датою і часом в Python.

За допомогою цього модуля ви можете виконувати операції, такі як отримання поточної дати та часу, створення дати та часу, виконання арифметичних операцій з датами, форматування дати і часу для виведення на екран тощо.

Наприклад, datetime.datetime.now() дає поточну дату та час, і ви можете використовувати різні методи цього об'єкта для отримання інших параметрів, таких як день, місяць, година, хвилина і т. д.

3.3 Розробка інтерфейсу користувача

При розробці інтерфейсу головне це те щоб інтерфейс був як простий для використання так і в той же час багатофункціональний. Оскільки в нас програмний продукт орієнтується на ПСНП та використання його як студентами так і викладачем то ми маємо в першу чергу познайомити користувача з інтерфейсом програми та вказати на гарячі клавіші які він може використовувати під час роботи з програмним продуктом.

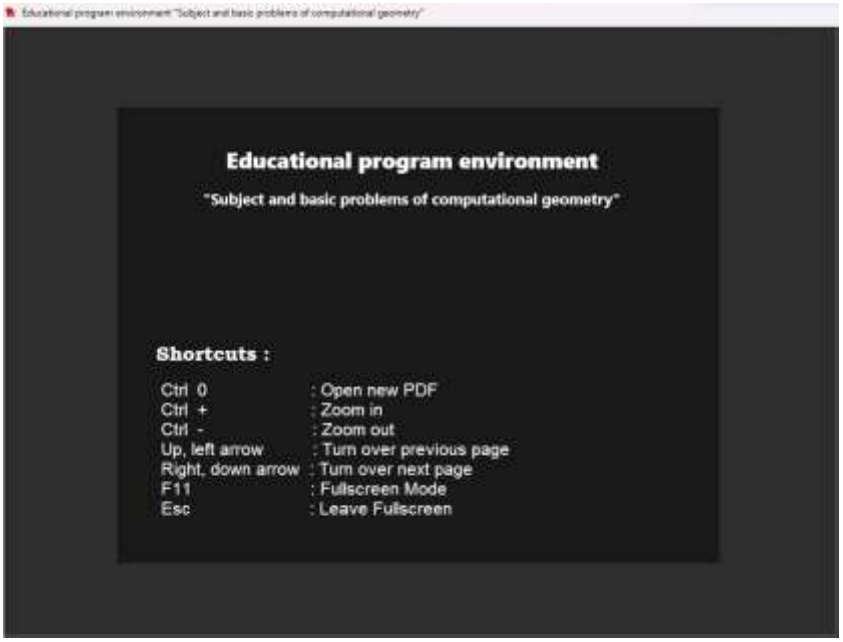

Рис.3.1. – Головне вікно. Опис кнопок що користувач може використовувати під час роботи з програмою

В першу чергу користувач має мати доступ до першочергових кнопок тому для початку в програмі надається список кнопок за допомогою яких можна відкрити файл та переглянути його.

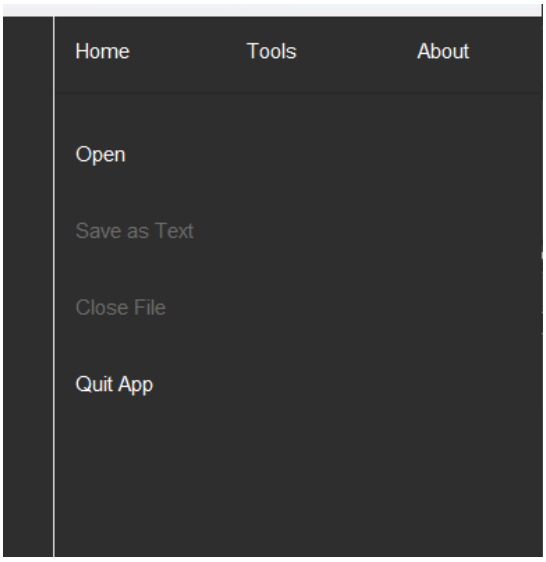

Рис.3.2. – Пункт меню «Home»

Після того як файл був відкритий користувач може перейти в пункт меню «Tools» де власне може використовувати основний функціонал програми.

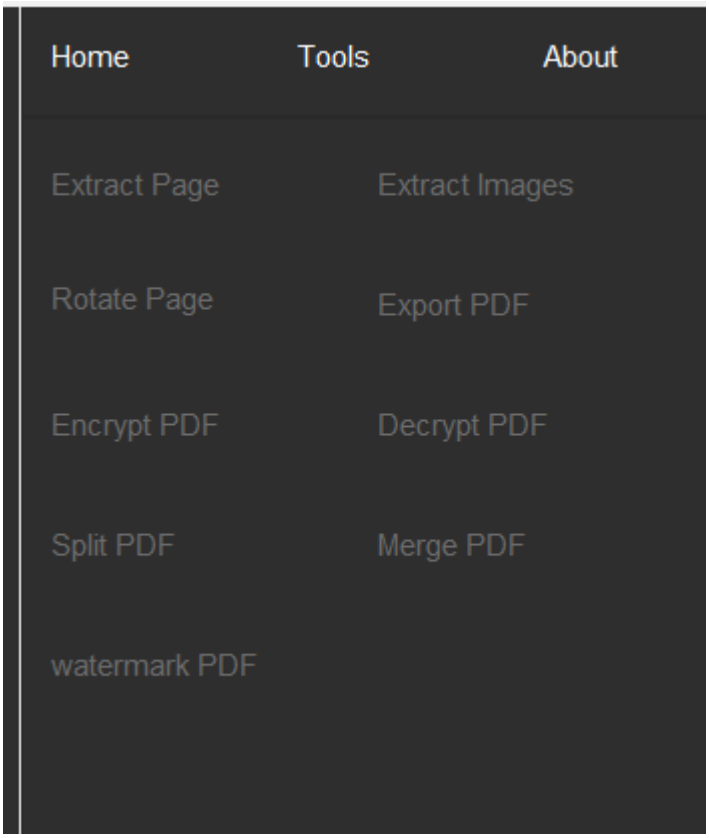

Рис.3.3. – Пункт меню «Tools»

Пункт меню «About» розповідає про автора що створював проект та коли він був створений.

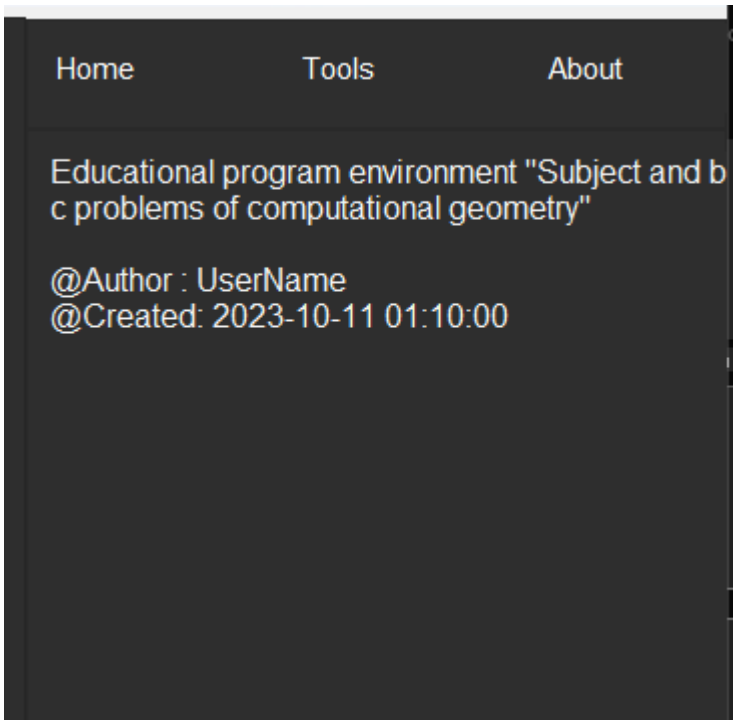

Рис.3.4. – Пункти меню в вкладці «Home»

Звісно потрібно також мати можливість бачити файли якими користувач вже користувався. Це дозволе зекономити час на їх відкриття так як в даному пункті вказано час останнього відкриття файлів.

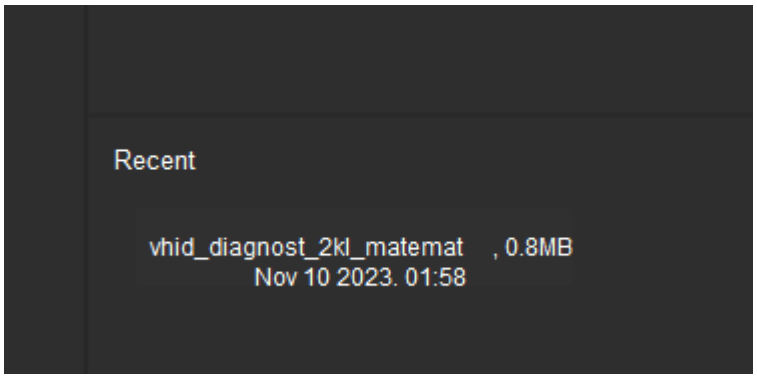

Рис.3.5. – Перегляд останнього відкриття файлу

Звісно що в програмі користувач має змогу переходу по сторінок та використання zoom для збільшення тексту та зручності його читання.

Рис.3.6. – Загальний вигляд відкритого PDF файлу з завданнями з обчислювальної геометрії

Отже в результаті отримаємо простий та зрозумілий користувачеві інтерфейс програмного продукту що дозволяє швидко орієнтуватися та робити маніпуляції з pdf файлом а також ознайомлюватися з задачами обчислювальної геометрії та ставити свої примітки що до виконання завдань.

3.4 Реалізація ПСНП

Оскільки бібліотеки які ми використовуємо для розробки ПСНП були описані в розділі вище опишемо основні функції що можна використовувати в програмному продукті та наведемо код що використаний в проекті.

Звісно що основною функцією є зчитування файлу для відображення. В розробленому додатку для цього використовується наступний код:

```
def read_pdf(self):
        metadata = self.pdf.metadata
        numPages = self.pdf.page_count
        toc = self.pdf.get_toc()

        page = self.pdf.load_page(0)
        pagesize = page.mediabox_size

        return metadata, numPages, toc, tuple(pagesize)
```

Оскільки клієнта може цікавити лише окрема сторінка або викладачеві потрібно відправити студентам на опрацювання конкретні сторінки то ми опишемо функцію яка зможе вирізати потрібну сторінку з файлу. Для цього створимо наступну функцію:

```
def extract_pdf_page(self, page_num, outfile):
        pdf2 = fitz.open()
        pdf2.insertPDF(self.pdf, from_page=page_num, to_page=page_num)
        pdf2.save(outfile)

        return 'done'
```

Оскільки нажаль деякі конспекти або книги є відсканені та поміщені в pdf файл потрібно надати функцію з орієнтацією сторінки., тобто щоб клієнт міг повернути заданий текст для зручного зчитування. Створемо відповідну функцію для цього:

```
def rotate_pdf_page(self, from_, to_, outfile, angle):
    pdf2 = fitz.open()
    pdf2.insertPDF(self.pdf, from_page=from_, to_page=to_, rotate=angle)
    pdf2.save(outfile)

    return 'done'
```

Не менш фажлива є можливість користувача використовувати зображення в заданому файлі тобто експортувати його з файлу. Функція що допомагає це зробити:

```
def extract_page_images(self, page_num):
    if not os.path.exists('images/'):
        os.mkdir('images/')

    filename = os.path.basename(self.filepath)
    for image in self.pdf.getPageImageList(page_num):
        xref = image[0]
        pix = fitz.Pixmap(self.pdf, xref)
        if pix.n < 5:
            pix.writePNG(f"images/{filename}-page{page_num}-
{xref}.png")
        else:
            pix1 = fitz.Pixmap(fitz.csRGB, pix)
            pix1.writePNG(f"images/{filename}-page{page_num}-
{xref}.png")
            pix1 = None
        pix = None
    return len(self.pdf.getPageImageList(page_num))
```

Також за допомогою даного програмного комплексу користувач може об'єднати файл pdf. Це зручно коли потрібно зєднати файл з лекцією та практикою. Функція що дозволяє це зробити має наступний вигляд:

```
def merge_pdf_file(self, infile,  from_, to_, outfile):
        pdf2 = fitz.open(infile)
        pdf2.insertPDF(self.pdf, from_page=from_, to_page=to_)
        pdf2.save(outfile)

        return 'done'
```

В результаті отримаємо готовий програмний продукт що може бути використаний для різних цілей.

3.5 Тестування ПСНП

Для тестування зчитування PDF файлів використовуючи бібліотеку PyPDF2 і фреймворк unittest, спершу потрібно створити тестовий клас, який буде містити тести. Отже опишемо функцію яка буде займатися тестуванням зчитування інформації з pdf файлу:

```
import unittest
import PyPDF2

class TestPDFReading(unittest.TestCase):
   def test_pdf_reading(self):
      # Відкриваємо PDF файл для зчитування
      with open('sample.pdf', 'rb') as pdf_file:
         pdf_reader = PyPDF2.PdfFileReader(pdf_file)

         # Перевіряємо, чи PDF має принаймні одну сторінку
         self.assertGreater(pdf_reader.numPages, 0)

         # Перевіряємо, чи можемо зчитати текст з першої сторінки
         page = pdf_reader.getPage(0)
         text = page.extractText()
```

```
            self.assertTrue(text)

    if __name__ == '__main__':
        unittest.main()
```

Ми створили тестовий клас TestPDFReading, який успадковує від unittest.TestCase. У методі test_pdf_reading, ми відкриваємо PDF файл, перевіряємо, чи є у ньому принаймні одна сторінка та чи можемо зчитати текст з першої сторінки. Ми використали методи assertGreater та assertTrue для визначення того, чи тести пройшли успішно.

Для тестування збереження PDF-файлу використовуючи бібліотеку PyPDF2 та фреймворк unittest, потрібно створити тестовий клас, який буде містити тести для збереження PDF-файлу. Ось приклад того, як це було зроблено в нашій роботі:

```
    import unittest
    import PyPDF2

    class TestPDFSaving(unittest.TestCase):
        def test_pdf_saving(self):
            # Спершу створіть новий PDF файл
            new_pdf = PyPDF2.PdfFileWriter()

            # Додайте пусту сторінку до нового PDF файлу
            new_page = PyPDF2.PdfFileWriter().addBlankPage(width=612, height=792)  # Розмір сторінки в точках (8.5 x 11 дюймів)
            new_pdf.addPage(new_page)

            # Збережіть новий PDF файл
            with open('output.pdf', 'wb') as output_file:
                new_pdf.write(output_file)
```

```
        # Перевірте, чи файл був створений успішно
        self.assertTrue('output.pdf')

if __name__ == '__main__':
    unittest.main()
```

У цьому прикладі ми створили тестовий клас TestPDFSaving, який успадковує від unittest.TestCase. У методі test_pdf_saving, ми спершу створюємо новий PDF файл за допомогою PyPDF2, додаємо до нього пусту сторінку та зберігаємо новий PDF файл під ім'ям 'output.pdf'. Потім ми перевіряємо, чи був файл створений успішно за допомогою методу assertTrue.

Для тестування збереження текстового вмісту з PDF файлу в TXT файл використовуючи бібліотеку PyPDF2 та фреймворк unittest, спершу потрібно створити тестовий клас, який буде містити тести для цієї операції. Заданий тест має наступний вигляд:

```
import unittest
import PyPDF2

class TestPDFtoTXT(unittest.TestCase):
    def test_pdf_to_txt(self):
        # Відкриваємо PDF файл для зчитування
        with open('input.pdf', 'rb') as pdf_file:
            pdf_reader = PyPDF2.PdfFileReader(pdf_file)

            # Витягаємо текст з усіх сторінок
            text = ''
            for page_num in range(pdf_reader.numPages):
                page = pdf_reader.getPage(page_num)
                text += page.extractText()
```

```
        # Зберігаємо текст у файл формату TXT
        with open('output.txt', 'w') as txt_file:
            txt_file.write(text)

        # Перевіряємо, чи файл був створений успішно
        self.assertTrue('output.txt')

if __name__ == '__main__':
    unittest.main()
```

У цьому прикладі ми створили тестовий клас TestPDFtoTXT, який успадковує від unittest.TestCase. У методі test_pdf_to_txt, ми відкриваємо PDF файл для зчитування, витягаємо текст з усіх сторінок і зберігаємо його у файл формату TXT під ім'ям 'output.txt'. Потім ми перевіряємо, чи був файл створений успішно за допомогою методу assertTrue.

Для запуску цих тестів використайте команду python -m unittest ім'я_файлу.py, де ім'я_файлу.py - назва файлу з тестами. PDF файл під назвою 'sample.pdf' у тій самій текі, де знаходиться ваш тестовий файл.

ВИСНОВКИ

В даній роботі були досліджені та вивчені основні аспекти обчислювальної геометрії. Починаючи з визначення основних задач, таких як перетин прямих та площин, обчислення відстаней та побудова опуклої оболонки, автор детально розглянув теоретичні аспекти та методи їх вирішення. Робота включала в себе аналіз аналогів програмного середовища навчального призначення, а також постановку задачі та вимоги до програмно-системного навчального продукту (ПСНП).

У другому розділі були розглянуті задачі побудови опуклої лінійної оболонки, розбиття площини за Г. Вороним, розв'язання систем М лінійних нерівностей. Детально розглянуті постановки цих задач та вивчені різні алгоритми для їх вирішення.

У третьому розділі роботи була проведена практична реалізація з використанням вибраних інструментів та середовищ розробки. Були створені бібліотеки, розроблений інтерфейс користувача та реалізовано програмно-системний навчальний продукт (ПСНП). Всі етапи роботи були піддані тестуванню для перевірки функціональності та відповідності вимогам.

В цілому, дана робота розкрила важливі аспекти обчислювальної геометрії, включаючи теоретичний аналіз та практичну реалізацію алгоритмів. Результати цієї роботи можуть бути корисні для різних сфер, де вимагаються точні розрахунки та аналіз геометричних об'єктів.

СПИСОК ВИКОРАСТАННИХ ДЖЕРЕЛ

1. Ивановский С.А., Преображенский А.С., Симончик С.К. Алгоритмы вычислительной геометрии. Выпуклые оболочки: простые алгоритмы // Компьютерные инструменты в образовании, 2007, №1. С. 4-19.Статья №1.

2. Graph Theory: An Introductory Course by Bollobas.B Springer 1979

3. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение, М.: Мир, 480 с., 1989.

4. D. R. Chand and S. S. Kapur, An algorithm for convex polytopes, J. ACM, 17:78-86, 1970.

5. F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. Commun. ACM, 20:87-93, 1977.

6. H. Edelsbrunner. Algorithms in Combinatorial Geometry. Springer-Verlag, Berlin, 1987.

A. M. Day. The implementation of an algorithm to find the convex hull of a set of three-dimensional points. ACM Trans. on Graphics, 9:105-132, 1990.

7. Ньюмен, Спрулл, Основы интерактивной машинной графики, М. Мир, 1976.

8. Энджел Й. Практическое введение в машинную графику, Радио и Связь, 1984.

9. А. Вэн-Дэм, Дж. Фоли, Основы интерактивной машинной графики, т.1-2, М. Мир, 1985.

10. Е.В. Жикин, А.В.Боресков, Компьютерная графика. Динамика, реалистические ихображения, М., Диалог-МИФИ, 1995, 1997.

11. Компьютер обретает разум. Пер. с англ. Под ред. В.Л.Стефанюка, М. Мир, 1990.

12. Роджерс, алгоритмические основы машинной графики. М. Мир, 1989.

13. Грайс, Графические средства персональных компьютеров, М., Мир, 1980.

14. Роджерс, Адамс, Математические основы машинной графики, М. Машиностроение, 1985.

15. Гилой, Интерактивная машинная графика, М., Мир, 1981.

16. Ф. Препарата, М. Шеймос, Вычислительная геометрия: Введение, М. Мир, 1989.

17. А.Фокс, М. Пратт, Вычислительная геометрия, М., Мир, 1982.

18. А.Б.Бояресков, Е.В.Шикина, Г.Е.Шикина, Компьютерная графика:первое знакомство, Под ред. Е.В.Шикина, М., Финансы и статистика, 1996.

19. А.В.Фролов, Г.В.Фролов, Графический интерфейс GDI вMS WINDOWS, Москва, Изд-во Диалог-МИФИ, 1994.

20. А.Хонич, Как самому создать трехмерную игру. М.:МИКРОАРТ, 1996.

21. Роджерс Д., Адамс А. Математические основы машинной графики. — М. :

22. Мир, 2001. — ISBN 5030021434.

23. Голованов Н. Н. Геометрическое моделирование. — М. : Физматлит, 2002. — ISBN 5940520480.

24. Дубровин Б. А., Новиков С. П., Фоменко А. Т. Современная геометрия: Методы и приложения. Т. 1. — 6-е изд. — М. : УРСС: Книжный дом «ЛИБРОКОМ», 2013. — ISBN 978-5-453-00047-0.

25. Фиников С. П. Курс дифференциальной геометрии. — М. : URSS, 2017.

26. Python home site. — 2018. — URL: https://www.python.org/.

27. Плас Д. В. Python для сложных задач. Наука о данных и машинное обучение. М. : Питер, 2018. — ISBN 978-5-496-03068-7.