МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Дипломна робота

РОЗРОБКА ОНЛАЙН КАЛЬКУЛЯТОРА ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ ГРАФІЧНИМ МЕТОДОМ

Виконала: студентка групи ПМі-43с
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Степанюк А.І.
(підпис) (прізвище та ініціали)

Керівник Олійник Р.М.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Дискретного аналізу та інтелектуальних систем
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри Притула М.М.

"31 "серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ У РОБОТУ СТУДЕНТУ

Степанюк Анні Ігорівній
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка онлайн калькулятора для розв'язання задач лінійного програмування графічним методом», керівник роботи

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від **"13" вересня 2022 року № 15**.

2. Строк подання студентом роботи **13.06.2023р.**

3. Вихідні дані до роботи документація по фреймворку React, JavaScript, IDE Visual Studio Code, Prettier, Typescript, UI бібліотека Material, інтернет ресурси

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) дослідити особливості розв'язання задач лінійного програмування графічним методом, розглянути можливі випадки, розробити вимоги до застосунку, створити сервіс з покроковим розв'язанням задачі

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Формалізація задачі для розробки онлайн калькулятора	18.10.2022	
2.	Огляд наукових джерел, книг, статей про лінійне програмування та графічний метод	03.11.2022	
3.	Аналіз попередніх робіт та програмних рішень, пов'язаних з калькуляторами лінійного програмування	20.12.2022	
4.	Розгляд наявних онлайн калькуляторів лінійного програмування, аналіз їхніх можливостей та обмежень	05.01.2023	
5.	Визначення функціональних вимог до калькулятора	10.01.2023	
6.	Розробка функціоналу калькулятора для введення задач, візуалізації графіку та отримання розв'язку	04.03.2023	
7.	Перевірка роботи калькулятора на тестових задачах лінійного програмування	28.05.2023	

Студент _____ Степанюк А.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Олійник Р.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дана бакалаврська дипломна робота полягає в розробці онлайн калькулятора для задач лінійного програмування графічним методом.

Для виконання завдання було обрано бібліотеку React для мови JavaScript, Typescript для строгої типізації, а також бібліотеку стилів Material UI.

Головною метою є створення зручного та ефективного інструменту, який дозволить користувачам визначати оптимальні розв'язки задач лінійного програмування шляхом використання графічного методу. Онлайн калькулятор повинен надавати можливість введення обмежень та цільової функції, відображати візуалізацію графіку цільової функції та обмежень, а також знаходити та відображати оптимальний розв'язок на графіці.

Результатом дипломної роботи є розроблений та реалізований функціональний онлайн калькулятор, який дозволяє користувачам вирішувати задачі лінійного програмування з використанням графічного методу. В ході виконання було розроблено інтуїтивно зрозумілий та ергономічний інтерфейс калькулятора, який відображає графіки цільової функції та обмежень на візуалізаційному полі, що дозволяє користувачу наочно спостерігати за залежностями та взаємодіями між обмеженнями та цільовою функцією.

ЗМІСТ

ВСТУП

У сучасному світі лінійне програмування є потужним інструментом для вирішення оптимізаційних задач в різних галузях, включаючи економіку, фінанси, транспорт, логістику та багато інших. Задача розподілу ресурсів та встановлення цінової політики є однією з основних задач, що можуть бути вирішені за допомогою лінійного програмування. Задачі лінійного програмування полягають у максимізації або мінімізації лінійної функції (цільової функції) за умови заданого набору лінійних обмежень.

З метою полегшити процес розв'язання задач лінійного програмування та забезпечити швидкі та ефективні рішення, розробка онлайн калькулятора стає актуальною та значущою задачею. Онлайн калькулятор забезпечує зручний та доступний інструмент для користувачів, щоб вони могли швидко та легко вирішувати задачі лінійного програмування. Графічний метод є одним з найпоширеніших методів розв'язку задач лінійного програмування, оскільки він надає можливість візуалізувати обмеження та цільову функцію на графіку та шукати оптимальний розв'язок шляхом ітеративного покрокового перетину лінійних функцій.

Також розроблений онлайн калькулятор для задач лінійного програмування з графічним методом надає користувачам можливість переглядати покроковий результат розв'язку, дозволяє краще розуміти процес розв'язування задачі.

Крім того, в калькуляторі передбачена можливість обрати кількість обмежень, що дозволяє користувачу налаштувати задачу відповідно до своїх потреб. Ця функція дає гнучкість та зручність у вирішенні різноманітних задач лінійного програмування з різними обмеженнями.

Цей підхід робить онлайн калькулятор більш інтерактивним та зручним для користувачів, дозволяючи їм бачити не лише кінцевий

результат, а й кожен етап процесу розв'язку, що сприяє кращому розумінню та аналізу задачі.

Тож, розроблений онлайн калькулятор для задач лінійного програмування графічним методом буде корисним та ефективним інструментом для студентів, дослідників та фахівців, які займаються оптимізаційними задачами.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 ПОНЯТТЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ ТА ЇХ ЗАСТОСУВАННЯ

Задачі лінійного програмування - це математичні задачі оптимізації, в яких метою є максимізація або мінімізація лінійної функції від змінних, що підлягають певним обмеженням. Лінійне програмування зазвичай використовується для розв'язання проблем, де потрібно приймати оптимальні рішення з обмеженими ресурсами, а це цілий спектр різноманітних галузей:

- Управління виробництвом. Наприклад, планування виробничих потоків, розподіл роботи або оптимальне використання обладнання.
- Логістика та транспортування. Це можуть бути проблеми маршрутизації транспорту, планування доставки, ефективного розміщення складів або розкладу рейсів.
- Фінанси та інвестиції. Прикладом застосування задач лінійного програмування тут може бути оптимізований розподіл фінансових ресурсів, планування інвестиційного портфеля, або розрахунок прибутковості проектів.
- Маркетинг та реклама. Це вирішення проблем пов’язаних з маркетинговими стратегіями, вигідним розподілом рекламного бюджету, плануванням маркетингових кампаній та асортиментним плануванням.

Це лише кілька прикладів застосування лінійного програмування. У реальному світі лінійне програмування використовується в багатьох інших галузях, де виникають задачі оптимізації з обмеженими ресурсами.

1.2 ПЕРША ЗГАДКА ПРО ЗАДАЧУ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Взагалі, задачу лінійного програмування вперше сформулював

Джордж Б. Данціг приблизно в 1947 році, коли він працював консультантом з математики в бухгалтерії повітряних сил Армії США над створенням механізованого планувального інструменту для програми часового стадійного розгортання, навчання та логістичного забезпечення. Хоча радянський математик і економіст Л. В. Канторович сформулював і вирішив проблему цього типу, пов'язану з організацією та плануванням, в 1939 році, його робота залишалася невідомою до 1959 року. Тому зазвичай проблему загального класу задач лінійного програмування приписують саме Данцігу. Оскільки повітряні сили називають свої різні плани та розклади "програмами", перша опублікована стаття Данціга відносилася до цієї проблеми як "Програмування в лінійній структурі". Термін "лінійне програмування" був створений економістом і математиком Т. К. Купмансом влітку 1948 року, коли він і Данціг прогулювалися біля пляжу Санта-Моніка в Каліфорнії.

Цікаво, що термін "лінійне програмування" виник як результат неточного перекладу англійського "linear programming". Одне із значень слова "programming" – складання планів, планування. Отже, правильним перекладом англійського "linear programming" було б не "лінійне програмування", а "лінійне планування", що більш точно відображає сутність вирішуваних задач. Однак, терміни ЛП, математичне програмування і так далі в наших інформаційних джерелах стали загальноприйнятими.

1.3 ФОРМАЛІЗОВАНИЙ ОПИС ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Отож, формулювати задачі лінійного програмування можна наступним чином:

1. Є об'єктивна функція, яку потрібно максимізувати або мінімізувати. Це лінійна функція, яка залежить від змінних.

2. Існують обмеження, які обмежують значення змінних. Ці обмеження також є лінійними нерівностями або рівностями.

Розв’язати задачу лінійного програмування – означає знайти оптимальне значення цільової функції при дотриманні обмежень задачі, основна мета полягає в знаходженні набору значень змінних, які максимізують або мінімізують цільову функцію, з урахуванням усіх обмежень.

Формалізований опис задачі лінійного програмування складається з:

- Цільової функції
- Обмежень
- Умови невід’ємності змінних

Формалізований опис задачі включає точне і математично коректне визначення всіх її елементів та уточнює всі необхідні деталі, щоб чітко сформулювати задачу. Він є важливим, оскільки дозволяє точно визначити проблему та зрозуміти її структуру. Крім того, формалізований опис є необхідним для застосування різних розв’язувальних методів та алгоритмів для знаходження оптимального розв’язку.

Загальна форма задачі лінійного програмування має наступний вигляд:

1. Цільова функція має загальний вигляд:

$$\mathrm{F(x)} = \sum_{i=1}^{n} (c_i x_i) \rightarrow \max(min) \quad ,$$

де c_i – коефіцієнти цільової функції,

x_i – змінні цільової функції

В ході даної роботи ми будемо розглядати цільову функцію з двома змінними, її вигляд:

$$\mathrm{F}(x_1, x_2) = \ c_1 x_1 + c_2 x_2 \ \rightarrow \max(min) \, ,$$

2. Обмеження, що представляються системою рівнянь або нерівностей (їх ще називають функціональні обмеження):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \{\leq, =, \geq\} b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \{\leq, =, \geq\} b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}x_m + a_{m2}x_m + \cdots + a_{mn}x_n \{\leq, =, \geq\} b_m \end{cases}$$

Умова невід'ємності змінних цільової функції (інша назва – прямі обмеження):

$$x_1 \geq 0, x_2 \geq 0 \ldots x_n \geq 0$$

1.4 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Задачі лінійного програмування можуть включати різноманітні умови та обмеження, які використовуються для моделювання різних реальних ситуацій.

Як зазначалось на початку, практичне застосування задач лінійного програмування поширене у повсякденному житті. Для більшої наочності наведу декілька прикладів типових задач та їх математичного представлення.

1.4.1 ЗАДАЧА ПРО ОПТИМАЛЬНИЙ ПЛАН ВИПУСКУ ПРОДУКЦІЇ

Задача про оптимальний план випуску продукції полягає в пошуку оптимального розподілу виробничих ресурсів для виготовлення товарів або послуг з метою максимізації прибутку або мінімізації витрат при заданих обмеженнях.

Формально, задачу можна сформулювати наступним чином:

Маємо набір N виробничих факторів, таких як праця, матеріали, обладнання тощо, і M видів продукції, які можна виготовити, Кожен вид продукції має свою вартість продажу та вимоги щодо використання

виробничих факторів (прибуток від виробництва і реалізації одиниці *j*-го виду продукції становить c_j, запаси ресурсів обмежені і становлять відповідно b_i одиниць, x_j – планований обсяг випуску продукції *j*-го виду, витрата *i*-го виду ресурсів на одиницю *j*-го виду продукції становить a_{ij} одиниць).

Завдання: встановити такий план $X=(x_1, x_2, \ldots, x_n)$ випуску продукції, який при фіксованих обсягах ресурсів максимізує загальний прибуток підприємства.

$$\max f(x) = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1,...,m, \quad i=1,\ldots,m,$$

$$x_j \ge 0 \quad j = 1,...,n.$$

1.4.2 ЗАДАЧА ПРО СУМІШІ

У загальному вигляді, задача про суміші передбачає наявність деякого набору інгредієнтів з відомими властивостями (наприклад, вартість, хімічний склад, поживна цінність тощо) і необхідність створити суміші або продукти з цих інгредієнтів з максимальною вигодою або заданими обмеженнями.

Основні елементи задачі про суміші включають:

- Інгредієнти: Набір різних інгредієнтів, що доступні для створення сумішей.
- Властивості інгредієнтів: Кожен інгредієнт має свої властивості, такі як вартість, розмір, хімічний склад, обмеження тощо.
- Цільова функція: Визначення мети задачі, яку можна максимізувати або мінімізувати, таку як вигода, вартість, якість тощо.
- Обмеження: Набір обмежень, які обмежують складання сумішей, такі як обсяги інгредієнтів, поживність, хімічні обмеження, виробничі обмеження тощо.

Задача про суміші може мати різні варіації, залежно від конкретного контексту і обмежень. Це широко використовувана задача в різних галузях, таких як виробництво, тваринництво (складання кормового раціону), логістика, хімічна промисловість, харчова промисловість (задача вибору дієти), фармацевтика та інші.

1.4.3 ЗАДАЧА ПРО РОЗКРІЙ МАТЕРІАЛІВ

Основна мета задачі про розкрій матеріалів полягає у визначенні оптимального способу розкрою аркушів матеріалу з мінімальним відходом, задовольняючи при цьому вимоги щодо розмірів і кількості шматків, які необхідно отримати.

У задачі про розкрій матеріалів зазвичай враховуються такі фактори:

- Розміри аркушів матеріалу: Це включає довжину, ширину та можливі обмеження щодо форми аркушів.
- Розміри шматків: Необхідно визначити оптимальний спосіб розрізання аркушів таким чином, щоб отримати необхідну кількість і розміри шматків.
- Обмеження: Можуть існувати обмеження на кількість аркушів, доступні матеріали, обробка часу, обсяги замовлень тощо.
- Цільова функція: Метою є мінімізація відходу матеріалу або мінімізація затрат, пов'язаних з розкроєм матеріалу.

Застосування задачі про розкрій матеріалів дозволяє економити матеріали, підвищувати продуктивність та покращувати загальну ефективність виробничих процесів.

1.5 АЛЬТЕРНАТИВНІ МЕТОДИ РОЗВ’ЯЗУВАННЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Тепер, коли ми знаємо з чим маємо справу, розглянемо основні методи розв’язування задач лінійного програмування:

- Симплексний метод

- Метод штучного базису
- Графічний метод

1.5.1 СИМПЛЕКСНИЙ МЕТОД

Суть симплексного методу полягає у пошуку оптимального розв'язку задачі шляхом переходу від одного базисного розв'язку до іншого. Так як маючи обмежену область допустимих значень можна почати обчислювати кожне зі значень цільової функції у всіх вершинах області і знайти екстремум простим перебором, що є дуже трудомістко методом, то симплексний метод пропонує цілеспрямовано рухатись по суміжним вершинам області допустимих значень. Цей метод ефективний у випадках, коли кількість змінних та обмежень є помірними. Однак, він може стати менш ефективним при великих розмірах задачі, коли кількість змінних та обмежень дуже велика.

Загалом, щоб розв'язати задачу лінійного програмування симплексним методом, потрібно виконати такі кроки:

1. Привести задачу до канонічної форми (канонічна форма виглядає так: $\min\ \mathrm{f}(x) = \sum_{j=1}^{n} c_j x_j$

   і має обмеження: $\sum_{j=1}^{n} a_{ij} x_j = b$, $i = 1,\ldots,\mathrm{m}$, $x_\mathrm{j} \geq 0$, $j = 1,\ldots,\mathrm{n}$.
2. Обрати опорний план – обрахувати початкове допустиме базисне рішення.
3. Виконати оцінку оптимальності. Якщо вона вже є оптимальною, алгоритм закінчується і повертається оптимальний розв'язок.
4. Якщо розв'язок не оптимальний – повернутись до кроку 3.

Отож, симплексний метод є універсальним і може застосовуватися для розв'язання широкого спектру задач лінійного програмування, легким для розуміння, а також в більшості випадків доволі ефективним.

Недоліками даного методу є низька швидкість виконання, особливо

при великій кількості змінних і обмежень метод може бути трудомістким та займати багато часу, а також вразливим до початкового базису.

1.5.2 МЕТОД ШТУЧНОГО БАЗИСУ

Наступним є метод штучного базису, що є альтернативою для розв’язання задач, коли базисні змінні існують не в усіх обмеженнях задачі. Суть методу – ввести штучні змінні у всі обмеження, де відсутні базисні змінні. Вони використовуються для побудови початкового базисного рішення і не мають жодного впливу на цільову функцію. Після введення змінних задача стає допустимою і може бути розв’язаною за допомогою симплексного методу.

Щоб розв’язати задачу лінійного програмування методом штучного базису, потрібно:

1. Ввести штучні змінні в обмеження, які не містять базисних змінних.
2. Розв’язати задачу симплексним методом.
3. Перевірити, чи розв’язок є оптимальним.
4. Якщо штучні змінні залишаються у базисі, то задача недопустима.
5. Якщо оптимальний розв’язок досягнутий, виконується процедура вилучення штучного базису.

Метод штучного базису легко реалізувати та зрозуміти, а також є доволі гнучким, адже може бути застосований як до задач з невеликим числом змінних та обмежень, так і для складних задач. Проте головною перевагою цього методу є звичайно ж те, що його можна використовувати навіть якщо початковий базис є недопустимим. Це дозволяє знаходити допустимі розв'язки, які потім можуть бути піддані оптимізації.

Проте у випадках застосування до складних задач з великою кількістю обмежень і змінних, використання методу штучного базису може означати виконання ітерацій значну кількість раз для знаходження оптимального розв’язку, що може вплинути на час його виконання. Отож,

потрібно враховувати поліноміальну складність цього алгоритму та чутливість до вибору початкового базису - невдалий вибір може призвести до багатократних ітерацій і затримок в знаходженні оптимального розв'язку.

1.6 ГРАФІЧНИЙ МЕТОД

Графічний метод розглянемо детальніше, окремо виділимо різні випадки, залежно від властивостей задачі та її обмежень. Отож, ми будемо розглядати задачу лінійного програмування у двовимірному просторі такого вигляду:

$$\mathrm{F}(x_1, x_2) = c_1 x_1 + c_2 x_2 \rightarrow \max(min),$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \leq b_1 \\ a_{21}x_1 + a_{22}x_2 \leq b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 \leq b_i \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 \leq b_m \\ x_1 \geq 0;\ x_2 \geq 0 \end{cases}$$

Цей приклад буде дуже корисним для наочного зображення структури та пояснення задач лінійного програмування, адже реальні математичні моделі містять набагато більше змінних, тому випадок двох змінних не має особливого прикладного значення.

Основна ідея графічного методу полягає в тому, що всі обмеження на площині визначають напівплощини з межовими прямими, отже ми можемо представити кожне обмеження у вигляді лінії на координатній площині. Наприклад, обмеження $a_{11}x_1 + a_{12}x_2 \leq b_1$ буде прямою на площині, аналогічно для інших обмежень.

1.6.1 ЗАГАЛЬНІ ВІДОМОСТІ

Область допустимих розв'язків - це перетин усіх напівплощин – опукла множина точок на площині X_1OX_2 (також називають областю допустимих планів задачі або многокутником розв’язків задачі лінійного

програмування), тобто область на площині, де всі обмеження виконуються. Ця область є фізично можливими значеннями для змінних x_1 і x_2. Цільова функція $F(x_1, x_2)$, яка максимізується, також може бути представлена у вигляді прямої на координатній площині. Оптимальний розв'язок задачі полягає в знаходженні точки перетину цільової функції і області допустимих розв'язків. Ця точка вказує значення x_1 і x_2, при яких досягається максимальне значення функції $F(x_1, x_2) = c_1 x_1 + c_2 x_2 \rightarrow max$. Таким чином, графічний метод надає геометричне розуміння та візуалізацію задачі лінійного програмування - він дозволяє шукати оптимальні розв'язки на основі аналізу графіків обмежень та цільової функції.

В двовимірній моделі цільової функції $F(x_1, x_2)$, лінії рівня та градієнт можуть бути використані для візуалізації та аналізу поведінки функції. Лінії рівня - це множина точок на площині, де значення функції є постійним значенням. Іншими словами, це лінії, на яких функція має однакові значення. У двовимірному випадку, лінії рівня для лінійної функції - це сімейство паралельних прямих на площині X_1OX_2, також можуть бути площиною або гіперплощиною (кількість змінних у даному випадку більша або рівна 4). Екстремальні точки допустимих рішень задачі лінійного програмування є кутовими точками цієї області.

Градієнт функції - це вектор, який вказує напрямок найшвидшого зростання функції (антиградієнт функції в цій точці вказує напрямок найшвидшого спадання функції). В двовимірному випадку, градієнт є перпендикулярним до лінії рівня і визначається:

$$\vec{n} = \nabla F(x_1, x_2) = \left(\frac{\partial F}{\partial x_1}; \frac{\partial F}{\partial x_2} \right) = (c_1, c_2).$$

Щоб знайти кутові точки, потрібно побудувати лінію рівня цільової функції, потім паралельно зміщаючи її паралельно до самої себе в

напрямку області допустимих планів, знайти екстремуми. Максимальному значенню функції буде відповідати та кутова точка, що була знайдена за допомогою паралельного пересування в напрямку градієнта.

Отож, в геометричному плані, задача звелась до пошуку такої точки, що:

1. Належить області допустимих значень.
2. Через неї проходить лінія рівня, що відповідає максимальному (мінімальному) значенню функції *F*.

1.6.2 АЛГОРИТМ РОЗВ'ЯЗКУ

Алгоритм розв'язку задачі лінійного програмування графічним методом:

1. Для кожного з обмежень побудуйте графіки граничних прямих на площині.
2. Визначте область допустимих значень (це область перетину усіх обмежень).
3. Побудуйте градієнт цільової функції.
4. Побудуйте лінію рівня цільової функції.
5. Зміщуйте лінію рівня цільової функції паралельно самій собі в напрямку градієнта.
6. Визначте екстремальні точки.
7. Обчисліть значення цільової функції в отриманих точках.

Для перевірки розв'язку, підставте його значення в усі обмеження та цільову функцію та переконайтесь, що він задовольняє усі обмеження та максимізує (або мінімізує) значення цільової функції.

1.6.3 КІЛЬКІСТЬ РОЗВ'ЯЗКІВ ЗАЛЕЖНО ВІД ВЛАСТИВОСТЕЙ ЗАДАЧІ ТА ОБМЕЖЕНЬ

При розв'язанні задачі лінійного програмування графічним методом можуть виникати різні випадки, залежно від властивостей задачі та її

обмежень. Ось деякі з них:

1. Єдиний оптимальний розв’язок. У цьому випадку, після побудови графіку та визначення області допустимих розв’язків, знайдеться одна точка перетину лінії рівня з областю, що максимізує (або мінімізує) цільову функцію. Ця точка буде оптимальним розв’язком.

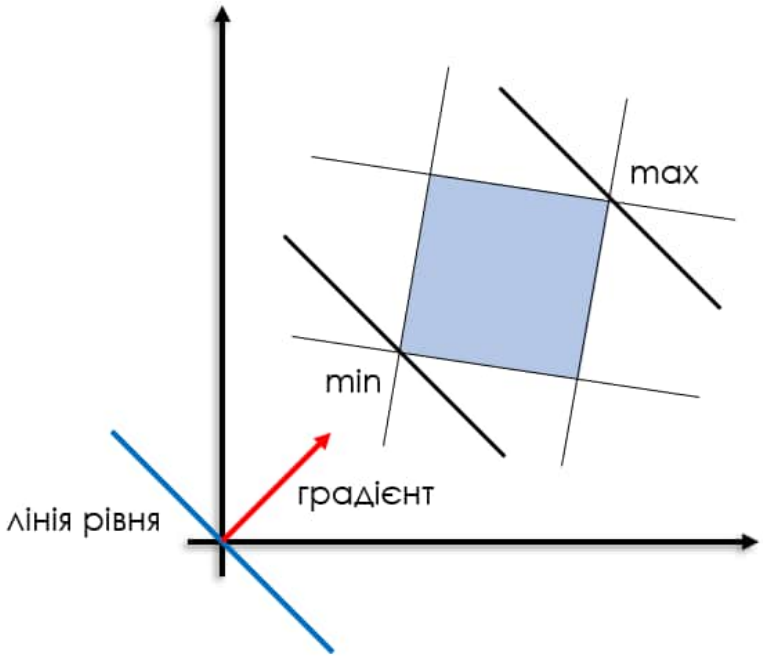


Рис. 1.6.3.1 – графічне представлення єдиного оптимального розв’язку

2. Множина оптимальних розв’язків. В деяких випадках може виникнути ситуація, коли лінія рівня проходить через декілька точок перетину з областю допустимих розв'язків. У цьому випадку, всі ці точки будуть оптимальними розв'язками, і задача матиме множину оптимальних розв'язків.

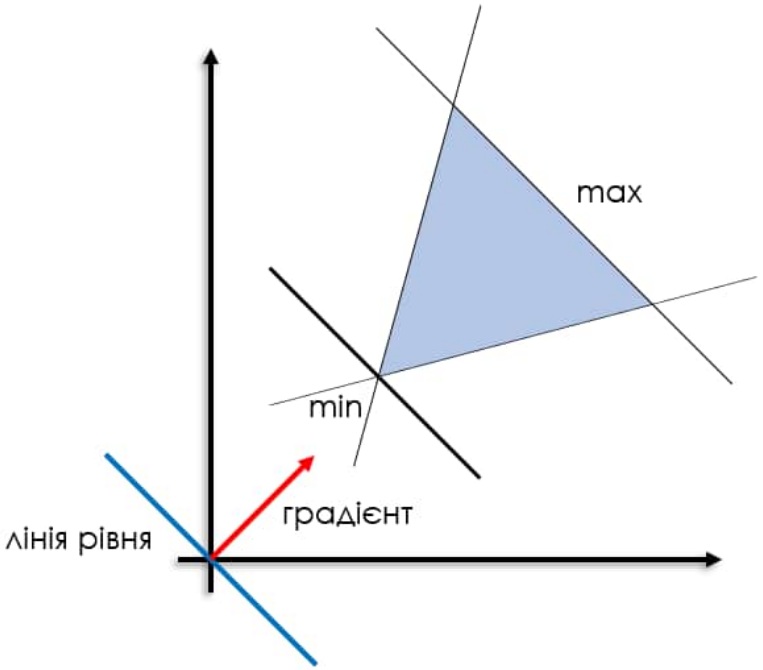


Рис. 1.6.3.2 – графічне представлення множини оптимальних максимумів

3. Необмежений екстремум. У задачах максимізації, необмежений розв'язок означає, що значення функції цілі може збільшуватись без обмежень, протягом нескінченності. Геометрично це відображається на графіку функції цілі, де немає верхньої межі.

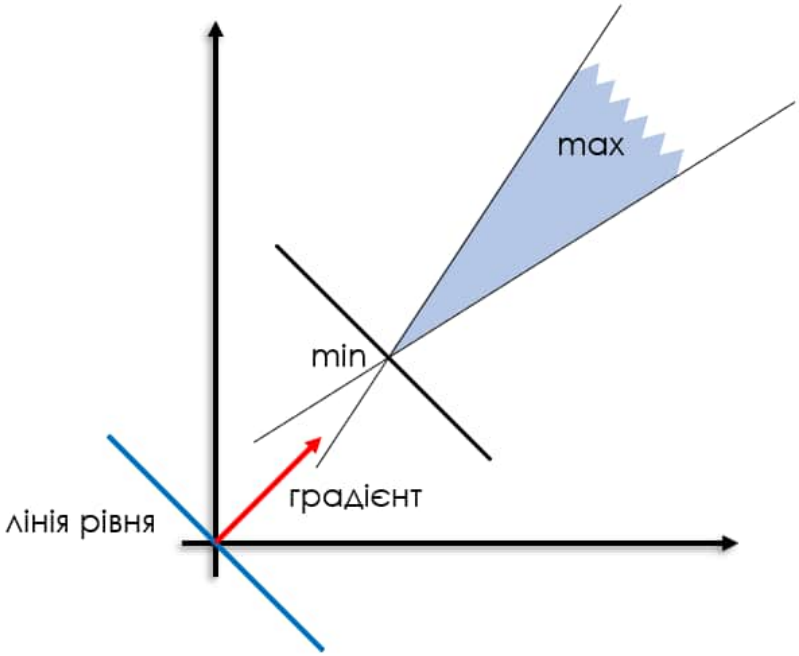


Рис. 1.6.3.3 – графічне представлення необмеженого максимума

1.6.4 ПРИКЛАД РОЗВ'ЯЗАННЯ ЗАДАЧІ ГРАФІЧНИМ МЕТОДОМ

Розглянемо приклад розв'язання задачі графічним методом.

Знайти максимальне значення функції

$$\mathrm{F}(x_1, x_2) = 3x_1 + 2x_2 \rightarrow max\ ,$$

При заданих обмеженнях:

$$\begin{cases} 2x_1 + x_2 \leq 10 \\ 2x_1 - x_2 \geq 0 \\ x_1 + x_2 \leq 6 \end{cases}$$

$$x_1 \geq 0;\ x_2 \geq 0$$

Хід розв'язання:

Спочатку побудуємо область допустимих значень задачі. Для побудови графіку кожному з обмежень ставимо знак рівності, обчислюємо дві точки та проводимо пряму. Таким чином отримуємо:

	x_1	x_2
$2x_1 + x_2 = 10$	3	4
	5	0
$2x_1 - x_2 = 0$	0	0
	2	4
$x_1 + x_2 = 6$	0	6
	6	0

Щоб з'ясувати яка з напівплощин буде розглядатись, в кожну нерівність підставимо точку (0;0) або (1;1) для другої нерівності:

1. (0;0) => $0 \leq 10$ – вірно, отже точка (0;0) входить до шуканої півплощини.
2. (1;1) => $1 \geq 0$ – вірно, отже точка (1;1) входить до шуканої півплощини.
3. (0;0) => $0 \leq 6$ – вірно, отже точка (0;0) входить до шуканої півплощини.

Отримуємо область допустимих значень, що знаходиться на перетині всіх напівплощин:

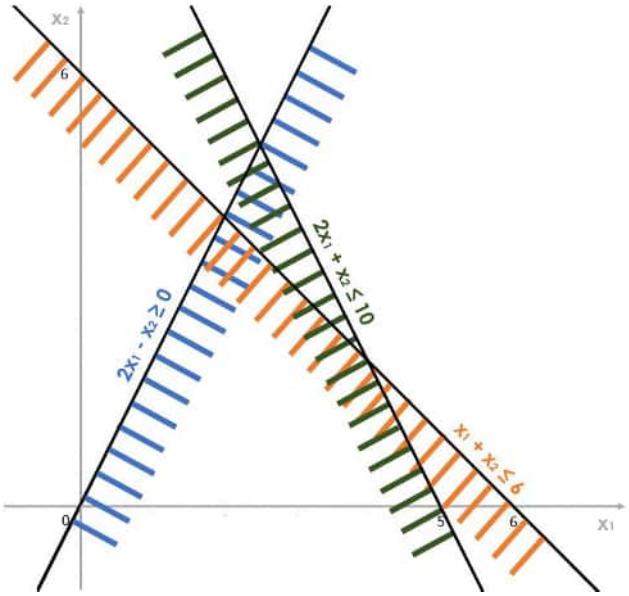


Рис. 1.6.4.1 – область допустимих значень

Наступним кроком є побудова лінії рівня та градієнта цільової функції.

Знаходимо координати вектору градієнта. Для цього потрібно знайти часткові похідні, для лінійної функції це фактично коефіцієнти при невідомих.

$$\mathrm{F}(x_1, x_2) = 3x_1 + 2x_2 \rightarrow max$$

$$grad\ F = \left(\frac{\partial F}{\partial x_1}; \frac{\partial F}{\partial x_2}\right) = (3; 2)$$

Будуємо вектор з точки $(0: 0)$ в точку $(3; 2)$ та перпендикулярну йому лінію рівня. Далі зміщаємо лінію рівня паралельно самій собі в напрямку градієнта, знаходимо останню точку дотику паралельної прямої з областю допустимих значень – це і буде наш розв'язок.

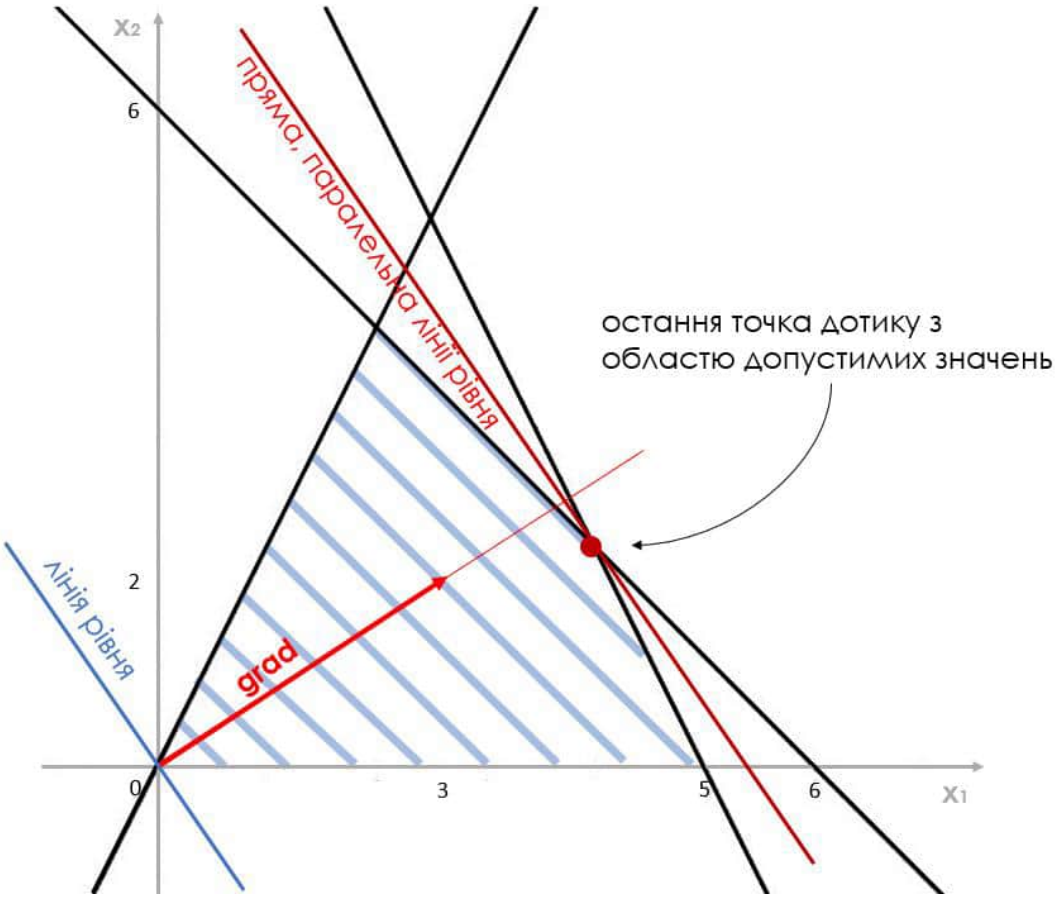


Рис. 1.6.4.2– графічний розв'язок

В даному випадку існує лише один оптимальний розв'язок задачі, він лежить на перетині двох прямих, тому координати можемо знайти

розв’язавши систему рівнянь, що відповідають цим прямим

$$\begin{cases} 2x_1 + x_2 = 10 \\ x_1 + x_2 = 6 \end{cases}$$

Отримаємо розв’язок – точку з координатами (4; 2). Значення цільової функції при цьому F* = 16.

2 ТЕХНІЧНИЙ СТЕК ТА СЕРЕДОВИЩЕ РОЗРОБКИ

Для того, щоб реалізувати онлайн калькулятор для розв’язку задач лінійного програмування, я використовую React – бібліотеку JavaScript, а також Typescript для визначення типів, бібліотеку стилів Material UI, ESLint – інструмент статичного аналізу коду для виявлення проблемних шаблонів та стилістичної витриманості коду, Prettier для автоматичного форматування коду, а як середовище розробки я обрала Visual Studio Code.

2.1 REACT

React - це JavaScript-бібліотека, яка дозволяє створювати інтерфейси користувача шляхом компонування окремих частин коду, відомих як компоненти. Вона використовує декларативний підхід, що спрощує розробку та підтримку інтерфейсів. React також пропонує ефективний спосіб роботи з інтерфейсом, що дозволяє оптимізувати продуктивність додатків. Більш того, React має гнучку архітектуру, що дозволяє легко розширювати та перевикористовувати компоненти для створення складних інтерфейсів.

React використовує концепцію, що логіка відображення пов'язана з іншою логікою інтерфейсу користувача, такою як обробка подій, зміна стану з часом і підготовка даних для рендерингу.

Замість того, щоб штучно розділяти технології, розташовуючи розмітку і логіку у окремих файлах, React розділяє відповідальність між вільно зв'язаними одиницями, відомими як "компоненти". Ці компоненти містять як розмітку, так і логіку, що дозволяє їм працювати разом. Використання компонентів дозволяє зберігати код більш організованим і

зрозумілим.

Хоча React не вимагає використання JSX (розширеної синтаксису JavaScript), багато людей цінують його за те, що він надає візуальну допомогу при роботі з інтерфейсом користувача в коді JavaScript. JSX також допомагає React виводити зрозумілі повідомлення про помилки та попередження, що полегшує пошук і виправлення проблем.

Компоненти в React дозволяють розділити інтерфейс користувача на незалежні, повторно використовувані частини, які функціонують незалежно одна від одної. Вони подібні до функцій JavaScript у своєму концептуальному підході. Компоненти приймають вхідні дані, відомі як "пропси", і повертають React-елементи, які описують, що має з'явитися на екрані.

Існують функціональні та класові компоненти у React. Розробляючи свій web-додаток, я використовувала саме функціональні компоненти, оскільки вони підтримують хуки, за рахунок чого можуть бути оптимізованими, а також вони є прості для читання і легкі для тестування.

Для взаємодії зі станом компонента, я використовую хуки. Хуки - це нововведення в React 16.8, яке дозволяє використовувати функціональні компоненти з можливостями React без необхідності використовувати класові компоненти.

За допомогою хуків можна винести логіку стану з компонента, що дозволяє легко тестувати і повторно його використовувати. Хуки дозволяють використовувати логіку стану без зміни структури компонентів, що сприяє перевикористанню логіки. Також використовуючи хуки можна легко розділити один компонент на більш маленькі функції залежно від їхнього призначення.

Безпосередньо у своєму проєкті я використовую useState - один із найпоширеніших хуків в React. Він дозволяє додати стан до функціонального компонента, використовується для ініціалізації одного

або кількох станових змінних. Він повертає масив з двома елементами: поточним значенням станової змінної та функцією, яку можна використовувати для її оновлення.

Також широко застосованим є хук useEffect, що дозволяє виконувати певні дії після кожного рендерингу компонента або при зміні певних залежностей. У хук useEffect передається два аргументи: функція, яка представляє ефект, і масив залежностей. Функція ефекту виконується після кожного рендерингу компонента або при зміні певних залежностей. У випадку, якщо масив залежностей передається як пустий ([]), ефект виконується тільки після першого рендерингу компонента. Функція, що передається як ефект, може повертати іншу функцію, яка використовується для очищення ефекту перед наступним викликом або при знищенні компонента. Хук useEffect дозволяє контролювати життєвий цикл компонента, виконувати необхідні дії після рендерингу і забезпечує більшу гнучкість у роботі з побічними ефектами у функціональних компонентах React.

2.2 CANVAS

Важливо окремо згадати, що для графічного представлення я використовую HTML-елемент <canvas>. Під час розробки застосунку я спробувала декілька бібліотек, що надають можливість створювати графіки (наприклад, React Chart.js, Victory, Recharts), проте більшість з них не задовольняли потреб для розробки онлайн калькулятора задач лінійного програмування графічним методом, тому я прийняла рішення використовувати HTML елемент <canvas> для відображення графіків функцій. Він надає покращені можливості контролю над зображеннями та дозволяє реалізовувати складні візуальні ефекти та анімацію. Також його перевагами є:

- Растровий контекст: <canvas> працює на основі растрової графіки (можливість малювати пікселі безпосередньо на екрані).
- JavaScript API: вбудовані різноманітні методи та функції для малювання фігур, ліній, тексту, зображень і багато іншого.
- Координатна система: <canvas> має координатну систему, де точка (0, 0) знаходиться в лівому верхньому куті. Я використовую цю систему координат для позиціонування та малювання об'єктів.
- Анімація і взаємодія: Важливо згадати, що з використанням JavaScript із елементом <canvas> можна створювати анімації, що реагують на події миші або клавіатури та створювати інтерактивні веб-додатки, хоча мій додаток і не містить анімацій.

2.3 TYPESCRIPT

Також я використовувала TypeScript для строгої типізації та розширення можливостей мови JavaScript. Основними перевагами TypeScript є:

- Статична типізація: TypeScript є статично типізованою мовою програмування, а це означає, що типи перевіряються на етапі компіляції, тобто помилки виявляються під час розробки. Статична типізація також полегшує рефакторинг коду і дозволяє зрозуміти структуру даних іншим розробникам.
- Визначення типів: TypeScript дозволяє визначати типи для змінних, параметрів функцій, властивостей об'єктів та інших елементів коду. Це допомагає забезпечити краще розуміння коду, а також покращує роботу з інтегрованими середовищами розробки, які надають підказки та автодоповнення на основі визначених типів.

- Розширені функції: TypeScript надає розширені можливості, такі як наслідування класів, інтерфейси, перечислення, узагальнені типи та багато іншого. Це дозволяє розробникам використовувати більш потужні конструкції мови для побудови складних програмних рішень.
- TypeScript легко інтегрувати до вже існуючої кодової бази, адже це лише «надбудова» над JavaScript. Тому є можливим поступово впроваджувати TypeScript без необхідності переписувати весь код.
- Підтримка та оновлення: TypeScript активно розвивається і підтримується Microsoft та великою спільнотою розробників. Це означає, що ви можете очікувати постійну підтримку, оновлення та виправлення помилок. Крім того, TypeScript інтегрується з іншими інструментами, такими як редактори коду та фреймворки, що допомагають полегшити розробку.

Все це робить TypeScript потужним інструментом для будь-якого розробника, особливо для проектів великого масштабу.

2.4 ESLINT

Окрім цього, я використовую ESLint – інструмент для аналізу коду, що допомагає виявляти та виправляти потенційні проблеми в коді та забезпечує вищу якість програмного забезпечення.

ESLint перевіряє код на основі набору правил, які можна налаштувати відповідно до конкретних вимог. Ці правила охоплюють різні аспекти програмування, такі як правильний синтаксис, норми форматування, уникнення потенційно небезпечних практик та інші. Наприклад, у моєму проєкті заборонено оголошувати змінні, що не використовуються, заборонено створювати пусті блоки коду, також описані правила форматування та витримки стилю. При виявленні проблем, ESLint виділяє їх як помилки або попередження, що допомагає виправити їх перед

релізом програми.

Однією з переваг ESLint є те, що він підтримує розширення, які дозволяють налаштувати його для використання з різними стандартами кодування або специфічними вимогами проекту. Це дозволяє використовувати ESLint в різних командних проектах з різними стилями написання коду.

ESLint можна легко інтегрувати з більшістю редакторів коду і інтегрованих середовищ розробки (IDE). Він надає розширену підказку, автодоповнення та відмічення проблем прямо в редакторі, що полегшує процес розробки та допомагає підтримувати чистоту та якість коду.

2.5 PRETTIER

Щоб полегшити роботу з ESLint, я використовую такий інструмент як Prettier. Він автоматично аналізує код і переписує його згідно з визначеними правилами форматування.

Основна мета Prettier полягає в тому, щоб усі учасники проекту використовували однаковий стиль коду без необхідності вручну вирівнювати та форматувати його. У великих проєктах це допомагає знизити рівень суперечок та дискусій з приводу стилю форматування, оскільки Prettier самостійно вирішує, як форматувати код.

Prettier підтримує багато мов програмування, включаючи JavaScript, TypeScript, CSS, HTML, JSON і багато інших. Також він надає можливість налаштувати правила форматування відповідно до своїх потреб, вказавши якими повинні бути відступи, вирівнювання, розташування дужок і т.д.

Одна з великих переваг Prettier - його простота використання. Його можна інтегрувати його зі своїм редактором коду або використовувати через термінал. Після запуску Prettier автоматично форматує весь код у проєкті згідно з встановленими правилами.

Використання Prettier разом з іншими інструментами, такими як ESLint,

допомагає забезпечити якість та чистоту коду.

2.6 VISUAL STUDIO CODE

В якості середовища розробки я використовую Visual Studio Code, що є безкоштовним редактором коду, розробленим компанією Microsoft. Він став популярним серед розробників завдяки своїй легкості використання, широкому спектру функціональних можливостей та розширюваності.

Це зручний та потужний редактор коду з підсвічуванням синтаксису, автодоповненням, переходами до визначення, розгортанням коду, інтегрованим терміналом та багатьма іншими корисними функціями. Його можна налаштувати враховуючи свої вподобання та потреби і використовувати різні режими редактора для більшої продуктивності.

Також він зручний тим, що підтримує багато мов програмування, включаючи JavaScript, TypeScript, Python, C#, Java, PHP та багато інших. Окрім цього, VS Code має потужну систему розширень, які дозволяють підключити різноманітні інструменти розробки, теми оформлення, плагіни для роботи з різними фреймворками та бібліотеками.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Отож, я реалізувала онлайн калькулятор задач лінійного програмування графічним методом. На початку роботи з сервісом користувач має можливість задати цільову функцію та систему нерівностей. Для того, щоб онлайн калькулятор був універсальним, я додала користувачу можливість задавати кількість рівнянь в системі за допомогою кнопок «+» та «-». Таким чином, користувач має змогу ввести від 2 до 4 нерівностей.

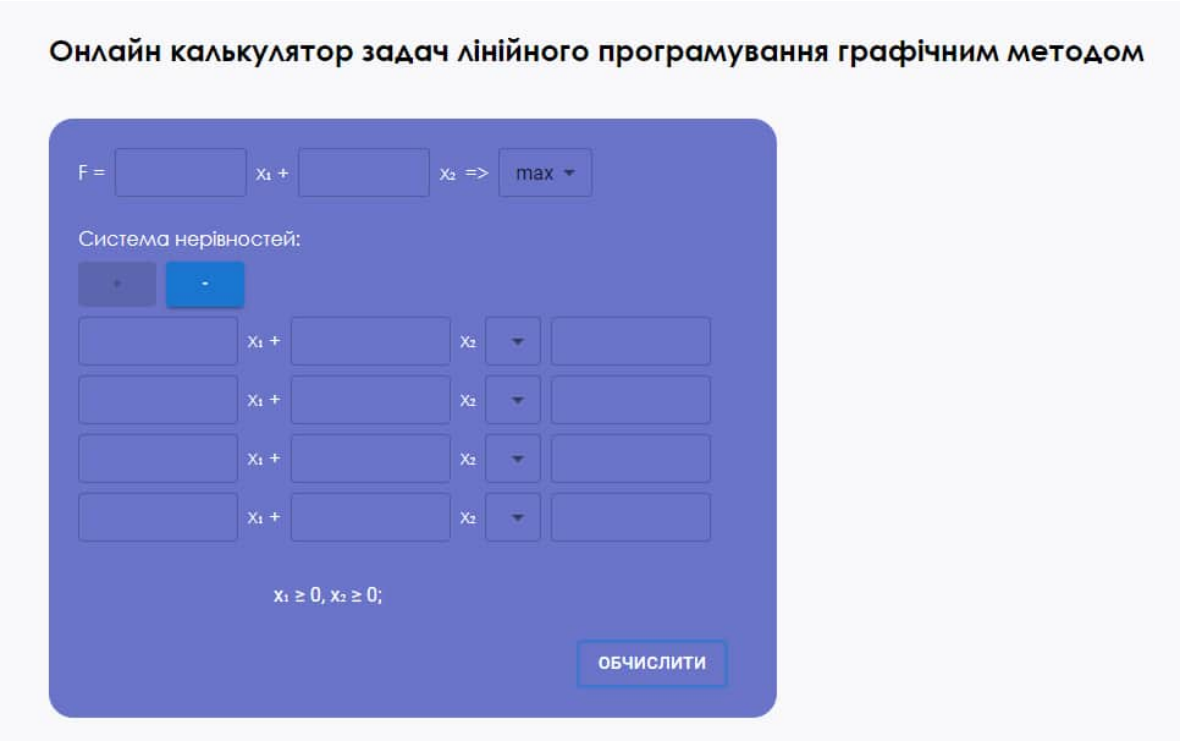


Рис. 3.1 – початкова форма

Розглянемо роботу онлайн калькулятора на прикладі задачі:

$$F(x_1, x_2) = 9x_1 + 6x_2 \rightarrow max\ ,$$

При заданих обмеженнях:

$$\begin{cases} 9x_1 + 3x_2 \leq 54 \\ -3x_1 + x_2 \leq 3 \end{cases}$$

$$x_1 \geq 0;\ x_2 \geq 0$$

Перенесемо умову до користувацької форми та отримаємо покрокове вирішення задачі.

Загальний вигляд додатку:

Онлайн калькулятор задач лінійного програмування графічним методом

F = 9 x_1 + 6 x_2 => max

Система нерівностей:

9 x_1 + 3 x_2 ≤ 54

-3 x_1 + 1 x_2 ≤ 3

$x_1 \geq 0, x_2 \geq 0;$

обчислити

Крок №1

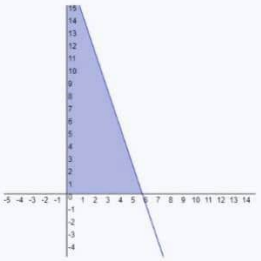
Розглянемо першу нерівність системи: $9 x_1 + 3 x_2 \leq 54$

Побудуємо пряму: $9 x_1 + 3 x_2 = 54$
Нехай $x_1 = 0$, тоді $3 x_2 = 54 \Rightarrow x_2 = 18$
Нехай $x_2 = 0$, тоді $9 x_1 = 54 \Rightarrow x_1 = 6$

Знайдено дві точки: (0, 18) та (6 ,0), через які проводимо пряму.

Щоб знайти напівплощину, де знаходяться допустимі розв'язки, повернемось до нерівності $9 x_1 + 3 x_2 \leq 54$
Підставимо в неї точку (0;0) та перевіримо чи виконується рівність:
$0 \leq 54$

Нерівність виконується, отже точка (0;0) входить до шуканої напівплощини, замалюємо її на графіку



Крок №2

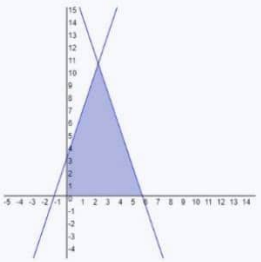
Розглянемо другу нерівність системи: $-3 x_1 + x_2 \leq 3$

Побудуємо пряму: $-3 x_1 + x_2 = 3$
Нехай $x_1 = 0$, тоді $x_2 = 3$
Нехай $x_2 = 0$, тоді $-3 x_1 = 3 \Rightarrow x_1 = -1$

Знайдено дві точки: (0, 3)та (-1 ,0), через які проводимо пряму.

Щоб знайти напівплощину, де знаходяться допустимі розв'язки, повернемось до нерівності $-3 x_1 + x_2 \leq 3$
Підставимо в неї точку (0;0) та перевіримо чи виконується рівність:
$0 \leq 3$

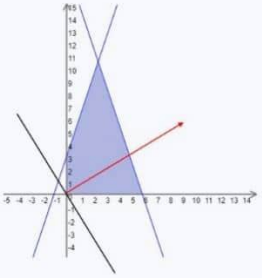
Нерівність виконується, отже точка (0;0) входить до шуканої напівплощини, замалюємо її на графіку



Крок №3

Побудуємо вектор-градієнт функції $\nabla F = (9; 6)$, координатами якого є коефецієнти цільової функції.

Також побудуємо перпендикулярну градієнту лінію рівня.

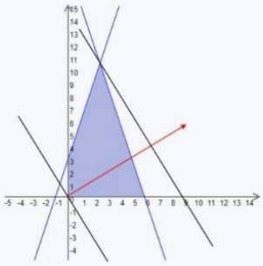


Крок №4

Переміщуємо лінію градієнта паралельно самій собі в напрямку вектора градієнта та знаходимо останню точку дотику паралельної прямої з областю допустимих значень - це буде максимум функції F.

Як бачимо, остання точка дотику знаходиться на перетині прямих, тож для знаходження розв'язку достатньо буде розв'язати систему рівнянь:
$9 x_1 + 3 x_2 = 54$
$-3 x_1 + x_2 = 3$

Отримуємо значення $x_1 = 2.5$, $x_2 = 10.5$. Обчислимо екстремум функції F:
$F_{max} = 9 * 2.5 + 6 * 10.5 = 85.5$



Відповідь:

$x_1 = 2.5$
$x_2 = 10.5$
$F_{max} = 85.5$

Рис. 3.2 – загальний вигляд додатку

Як бачимо, застосунок показує покрокове розв’язання задачі, розглянемо кожен з етапів детальніше.

Перший крок – намалювати пряму, що відповідає першій нерівності та визначити її область допустимих значень.

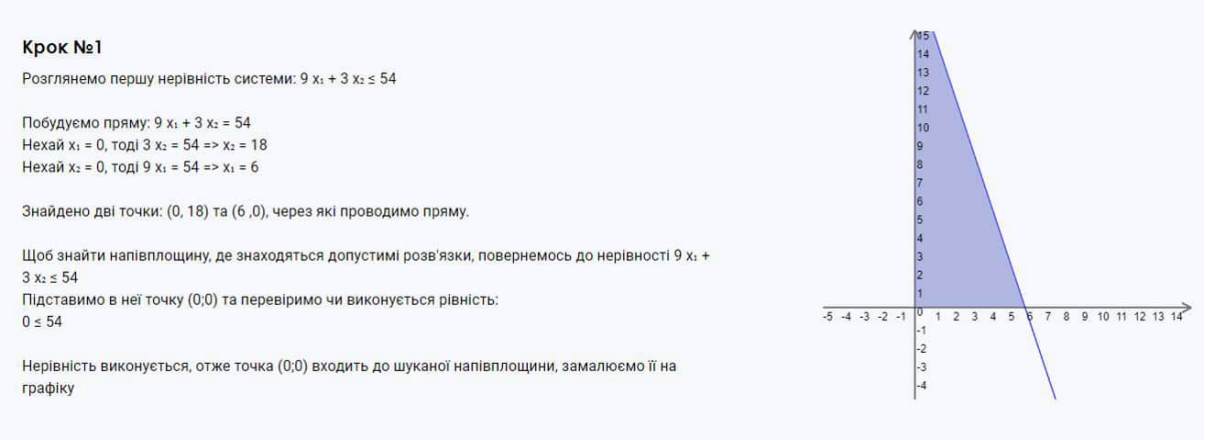


Рис. 3.3 – перший крок розв’язування задачі

Наступним кроком є зображення другої прямої та пошук області допустимих значень, що задовольняла б обидва обмеження:

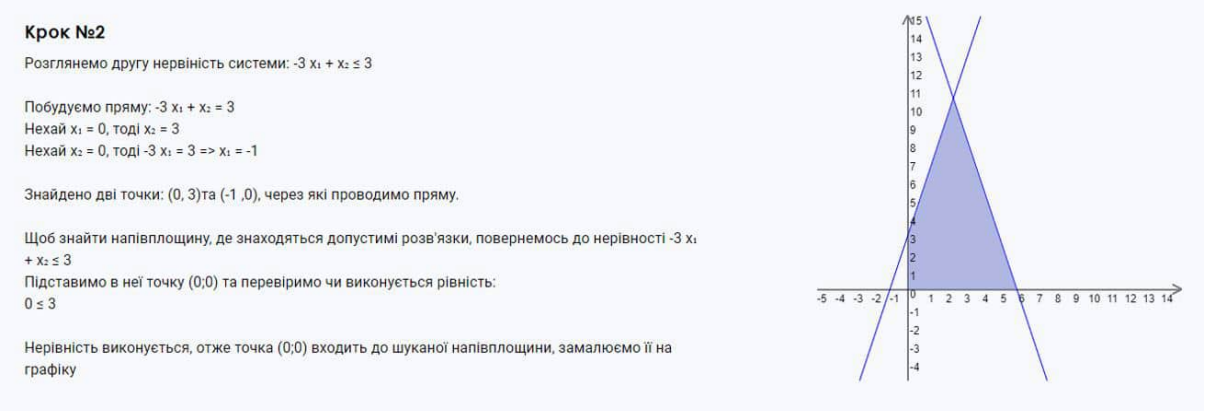


Рис. 3.4 – другий крок розв’язування задачі

Після визначення області допустимих значень, будуємо градієнт цільової функції та її лінію рівня:

Крок №3

Побудуємо вектор-градієнт функції ∇F = (9; 6), координатами якого є коефецієнти цільової функції.

Також побудуємо перпендикулярну градієнту лінію рівня.

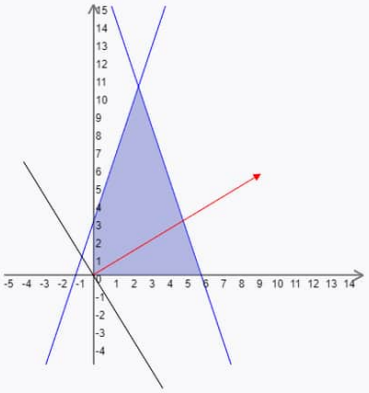


Рис. 3.5 – третій крок розв’язування задачі

Зміщуємо лінію рівня паралельно самій собі та отримуємо точку екстремуму цільової функції.

Крок №4

Переміщуємо лінію градієнта паралельно самій собі в напрямку вектора градієнта та знаходимо останню точку дотику паралельної прямої з областю допустимих значень - це буде максимум функції F.

Як бачимо, остання точка дотику знаходиться на перетині прямих, тож для знаходження розв'язку достатньо буде розв'язати систему рівнянь:

$9\ x_1 + 3\ x_2 = 54$

$-3\ x_1 + x_2 = 3$

Отримуємо значення $x_1 = 2.5$, $x_2 = 10.5$. Обчислимо екстремум функції F:

$F_{max} = 9 * 2.5 + 6 * 10.5 = 85.5$

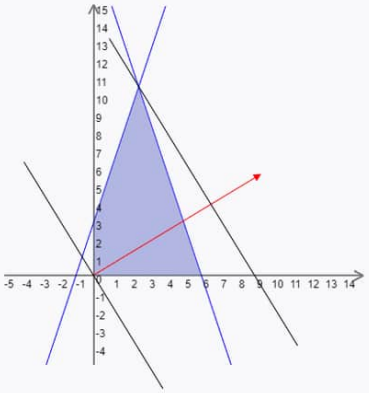


Рис. 3.6 – останній крок розв’язування задачі, представлення результату

ВИСНОВКИ

У даній дипломній роботі було успішно розроблено онлайн калькулятор для задач лінійного програмування з використанням графічного методу.

Результатом роботи є функціональний онлайн калькулятор, який дозволяє користувачам вводити свої задачі лінійного програмування та отримувати точні розв'язки. Калькулятор надає можливість візуального представлення графічного методу, що спрощує розуміння процесу оптимізації та допомагає зробити обґрунтовані рішення.

Покроковий результат розв'язку задачі, представлений калькулятором, дозволяє користувачам бачити послідовність операцій та зміну геометричних елементів на графіці, що сприяє глибшому розумінню процесу розв'язку задачі.

Онлайн калькулятор є зручним інструментом для студентів, викладачів, дослідників та фахівців з лінійного програмування. Він дозволяє ефективно вирішувати задачі та економити час, спрощуючи процес оптимізації та розв'язання лінійних задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vanderbei, R. J. (2008). Linear Programming: Foundations and Extensions.
2. Mokhtar S. Bazaraa, John J. Jarvis, HanifD.Sherali. Linear Programming and Network Flows, Fourth Edition.
3. Юдін Д. Б.,Гольштейн Е. Р. Лінійне програмування.Теорія, методи та додатки.
4. Документація по React - Режим доступу: https://uk.legacy.reactjs.org/
5. Веб-сайт Material UI - Режим доступу: https://mui.com/material-ui/
6. Документація по Typescript - Режим доступу: https://www.typescriptlang.org/