МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
(повна назва факультету/інституту)

_____КАФЕДРА КОМП'ЮТЕРНИХ НАУК_____
(повна назва випускової кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПО ДОСЛІДЖЕННЮ ЕФЕКТИВНОСТІ МЕТОДІВ ОПТИМІЗАЦІЇ

Рівень вищої освіти другий (магістерський)

Спеціальність _____122___Комп'ютерні науки__
(код і найменування спеціальності)

Освітня програма _____Комп'ютерні науки_____
(назва освітньої програми)

Виконав студент групи МгІТ-1-23

_____Йора М.І.
(прізвище та ініціали)

Науковий керівник ___к. т. н., доц. Корогод Г.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент__д. ф.-м. н. проф._ Краснитський С.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ
ТА ДИЗАЙНУ

Факультет Мехатроніки та комп'ютерних технологій
(повна назва факультету/інституту)

Кафедра Комп'ютерних наук
(повна назва кафедри)

Рівень вищої освіти другий (магістерський)
(перший (бакалаврський) / другий (магістерський))

Спеціальність 122 Комп'ютерні науки
(код і назва спеціальності)

Спеціалізація _____
(код і назва спеціальності)

Освітня програма Комп'ютерні науки
(назва освітньої програми)

ЗАТВЕРДЖУЮ

***** Завідувач кафедри КН

_____ Наталія Чупринка

(підпис)

«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

1. Тема роботи: Розроблення програмного забезпечення по дослідженню ефективності методів оптимізації

Науковий керівник роботи Корогод Ганна Олександрівна, к.т.н., доцент

затверджені наказом КНУТД від "_03" _вересня_2024року № 188-уч.

2. Строк подання студентом дипломної роботи 20.11.2024 р.

3. Вихідні дані до дипломної роботи (проекту) Розробки кафедри комп'ютерних наук, рекомендована література.

4. Зміст дипломної бакалаврської роботи: 1) Постановка задачі і аналіз предметної області, 2) Методи дослідження, 3) Обґрунтування програмних технологій і розробка програмного продукту.

5. Дата видачі завдання 10.08.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу кваліфікаційної роботи	Орієнтовний термін виконання	Примітка про виконання
1	Вступ	31.08.24 р.	
2	Розділ 1. Лінійні моделі	30..09.24 р.	
3	Розділ 2. Нелінійні моделі	21.10.24 р.	
4	Розділ 3. Аналіз способів дослідження ефективності методів оптимізації	10.11.24 р.	
5	Висновки	12.11.24 р.	
6	Оформлення (чистовий варіант)	14.11.24 р.	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку (за 14 днів до захисту)	15.11.24 р.	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	17.11.24 р.	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	25.11.24 р.	
10	Подання кваліфікаційної роботи завідувачу кафедри (за 7днів до захисту)		

З завданням ознайомлений:

Студент _____ Максим Йора

Науковий Керівник _____ Ганна Корогод

АНОТАЦІЯ

Йора М.І. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПО ДОСЛІДЖЕННЮ ЕФЕКТИВНОСТІ МЕТОДІВ ОПТИМІЗАЦІЇ

Дипломна кваліфікаційна робота за спеціальністю 122 - «Комп'ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2024 рік

В роботі виконано обзір основних способів дослідження ефективності методів оптимізації. Розглянуто детально такі способи як теоретичний аналіз, емпіричне тестування, статистичний аналіз, візуалізація процесу оптимізації, робастність методу, аналіз витрат, гібридні методи. Розроблено комп'ютерну програму для тестування методів градієнтного спуску.

Ключові слова: збіжність, помилка, обчислювальна складність, алгоритми спуску.

ABSTRACT

Yora M.I. DEVELOPMENT OF SOFTWARE FOR RESEARCHING THE EFFICIENCY OF OPTIMIZATION METHODS

Diploma qualification work in specialty 122 - "Computer Science". - Kyiv National University of Technologies and Design, Kyiv, 2024

The paper reviews the main methods of studying the effectiveness of optimization methods. Methods such as theoretical analysis, empirical testing, statistical analysis, visualization of the optimization process, robustness of the method, cost analysis, hybrid methods are considered in detail. A computer program for testing gradient descent methods is developed.

Keywords: convergence, error, computational complexity, descent algorithms.

Зміст

Вступ

У сучасному світі ефективність у прийнятті рішень є ключовим фактором розвитку в будь-якій сфері: від економіки до інформаційних технологій, оскільки дозволяють знаходити найкращі рішення серед множини можливих варіантів, підвищувати продуктивність та ефективність функціонування складних систем тощо. Крім того, багато задач є багатовимірними, нелінійними, з великою кількістю обмежень, що створює нагальну потребу в аналізі ефективності методів оптимізації, що застосовуються. В свою чергу, розвиток обчислювальних потужностей стимулюють вдосконалення існуючих методів оптимізації, включаючи алгоритми рою часток, гібридні підходи тощо.

Таким чином, актуальним слід вважати дослідження методів оптимізації при різних умовах, що дозволяють зробити висновки щодо вибору алгоритму для конкретної задачі, налаштування його параметрів, а також ідентифікувати області для вдосконалення.

Об’єкт дослідження – методи розв’язання задач оптимізації.

Предмет дослідження – ефективність градієнтних методів при вирішенні задач нелінійного програмування.

Метою роботи є розробка програмного забезпечення по дослідженню ефективності методів оптимізації, що матиме теоретичну та практичну цінність по їх подальшому розвитку та вдосконаленню.

Ефективність оцінюється за різними критеріями, такими як швидкість збіжності, точність результату, стійкість до шуму, обчислювальна складність тощо. Нижче наведено основні способи та підходи до аналізу ефективності методів оптимізації.

1. Теоретичний аналіз

Цей спосіб включає дослідження властивостей алгоритму за допомогою математичного моделювання.

Основні показники:

- Швидкість збіжності:
    - Лінійна, квадратична чи надлінійна збіжність.
    - Час досягнення мінімуму за кількість ітерацій.
- Гарантії оптимальності:
    - Умови, за яких алгоритм забезпечує знаходження глобального мінімуму.
    - Залежність від структури функції (гладкість, опуклість).
- Обчислювальна складність:
    - Залежність витрат обчислень від розмірності задачі.
    - Аналіз кількості викликів функції або градієнта.

2. Емпіричне тестування

2. Практичне тестування алгоритму на реальних чи штучно створених задачах.

Процедура:

1. Вибір тестових задач:
    - Реальні задачі: приклади з інженерії, економіки, машинного навчання тощо.
    - Штучні функції: функції Розенброка, Растрігіна, Акермана тощо, які мають добре відомі властивості.
2. Метрики оцінки:
    - Час збіжності: кількість ітерацій або обчислювальна складність.
    - Помилка: відстань до істинного мінімуму (якщо відомий).
    - Стійкість: здатність алгоритму працювати в умовах шуму чи неточних обчислень.
3. Порівняння з іншими методами:
    - Порівняння різних алгоритмів на однакових задачах.
    - Використання середніх результатів по множині задач.

3. Статистичний аналіз

Застосовується для узагальнення результатів емпіричних тестів.

Методи:

- Аналіз розподілу результатів:
    - Побудова гістограм або коробкових діаграм (box plots) для оцінки стабільності алгоритму.
- Статистичні тести:
    - Тест Манна-Уїтні, t-тест або ANOVA для перевірки значущості різниць між алгоритмами.
- Чутливість до параметрів:
    - Аналіз, як зміна параметрів алгоритму (наприклад, швидкості навчання) впливає на результат.

4. Вивчення стійкості

Стійкість алгоритму до різних факторів визначає його надійність у реальних умовах.

Аспекти:

- Шум у даних: перевірка алгоритму на функціях, значення яких містять випадкові збурення.
- Змінність початкових умов: як початкове наближення впливає на результат.
- Робота з обмеженнями: перевірка здатності алгоритму враховувати обмеження на змінні або функції.

5. Візуалізація процесу оптимізації

Візуалізація допомагає зрозуміти динаміку роботи алгоритму.

Методи:

- Траєкторія пошуку:
    - Графіки траєкторій у просторі рішень (для низьковимірних задач).
- Зміна значення цільової функції:
    - Логарифмічний графік значень цільової функції від номера ітерації.

- Графіки параметрів:
    - Динаміка зміни внутрішніх параметрів алгоритму (наприклад, навчальної швидкості).

6. Оцінка робастності

Робастність визначає, наскільки ефективно алгоритм працює за несприятливих умов.

Способи оцінки:

- Робота на задачах з багатьма локальними мінімумами.
- Перевірка на умовах великої розмірності.
- Зміна типу функції (гладка, негладка, з розривами тощо).

7. Аналіз витрат

Оцінка витрат обчислювальних ресурсів, таких як час і пам'ять:

- Час роботи: залежність від кількості змінних і складності функції.
- Використання пам’яті: важливо для методів, які зберігають велику кількість проміжних даних.

8. Гібридні методи тестування

Комбінування теоретичних та емпіричних підходів для більш повної оцінки. Теоретичний аналіз використовується для грубого розуміння властивостей алгоритму, а емпіричне тестування – для підтвердження результатів на практиці.

Практична цінність. Результати аналізу дозволяють зробити висновки щодо вибору алгоритму для конкретної задачі, налаштування його параметрів, а також ідентифікувати області для вдосконалення

Розділ 1
ТЕОРЕТИЧНІ АСПЕКТИ МЕТОДІВ ОПТИМІЗАЦІЇ

1.1 Метричні та лінійні простори і їх підмножини

Множина X має назву ***метричного простору*** [1] , якщо для кожної пари елементів (*x, y*) визначене дійсне число $\rho(x, y)$, що має назву відстань між *x* і *y*, причому виконуються наступні умови (аксіоми метрики) :

1. $\rho(x, y) \geq 0$; $(\rho(x, y) = 0) \Leftrightarrow (x = y)$;
2. $\rho(x, y) = \rho(y, x)$;
3. $\rho(x, y) \leq \rho(x, z) + \rho(z, y),\ \forall\, x, y, z \in$ X.

Перша з вказаних властивостей зветься аксіомою *невід'ємності*, друга — аксіомою *симетричності* і третя — *аксіомою* (*або нерівністю*) *трикутника* [1]

$X = R^1$. Покладемо $\rho(x, y) = |x - y|$. Властивості 1, 2 очевидні.

Доведемо третю аксіому метрики:

Оскільки $|\mathrm{a} + b| \leq |\mathrm{a}| + |b|,\ \forall\, \mathrm{a}, b \in R^1$, то

$$\rho(x, y) = |x - y| = |(x - z) + (z - y)| \leq |x - z| + |z - y| = \rho(x, \mathrm{z}) + \rho(\mathrm{z}, y)$$

Лінійні простори і підпростори

Множина L називається лінійним або векторним простором над полем K, а його елементи називаються векторами [2] , якщо на ньому визначені операції:

$\forall x, y \in L, \qquad \alpha \in K$

a) векторного додавання: $x + y \in L$;

b) множення вектора $x \in L$ на скаляр $\alpha \in K$: $\alpha x \in L$,

що задовольняють таким умовам:

1. $x + y = y + x$;
2. $(x + y) + z = x + (\, y + z\,)$;
3. $x + y = x + z \Rightarrow y = z$;
4. $\alpha x + \alpha y = \alpha(x + y)$;
5. $\alpha x + \beta x = (\alpha + \beta)x$;
6. $\alpha(\beta x) = (\alpha\beta)x$;
7. $1 \cdot x = x$;

За означенням, $(-1) \cdot x = -x; \quad x - y = x + (-y);$

За прийнятими положеннями в просторі L існує O^* — нульовий елемент, який володіє наступними властивостями:

$$x + O^* = x; x - x = O^*;$$

Дійсно, покладемо

$$O^* = 0 \cdot x. \tag{1.1}$$

Тоді $x + O^* = x + 0 \cdot x = (1 + 0)x = 1 \cdot x = x$.

Неважко бачити, що вказаний нульовий елемент єдиний: якщо існує інший елемент 0 ' з тими ж властивостями, то маємо

$$\begin{cases} x + y = x + O^* + y = (x + y) + O^* \\ x + y = x + y + O' = (x + y) + O' \end{cases} \Rightarrow \mathrm{O}' = \mathrm{O}^*.$$

Подалі в якості поля K за умовчуванням розглядається множина дійсних чисел R^1. Інший можливий варіант — поле комплексних чисел Z.

Були розглянуті наступні випадки:

1) R^N — лінійний простір. Для $x = \begin{pmatrix} x_1 \\ \cdots \\ x_N \end{pmatrix}$, $y = \begin{pmatrix} y_1 \\ \cdots \\ y_N \end{pmatrix}$, $\alpha \in R^1$, за означенням,

$$x + y = \begin{pmatrix} x_1 + y_1 \\ \dots \\ x_n + y_n \end{pmatrix}, \alpha x = \begin{pmatrix} \alpha x_1 \\ \dots \\ \alpha x_N \end{pmatrix}.$$

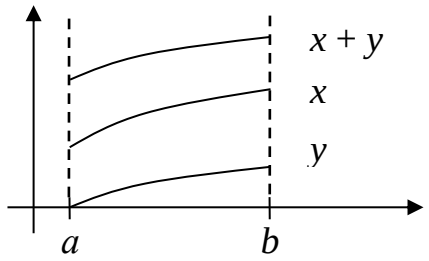
2) $C[a,b]$ — лінійний простір. Операції додавання та множення на скаляр визначаються звичайним чином як відповідні дії над функціями, тобто для

$$x = x(t), y = y(t), \alpha \in R^1$$

$$(x + y)(t) = x(t) + y(t), (\alpha x)(t) = \alpha \cdot x(t) \qquad (1.2)$$

(див. рис. 3.1). Відомі теореми математичного аналізу гарантують, що при $x, y \in C([a, b])$ функції $x + y, \alpha x$ також належать $C([a, b]$.

3) $L_2[a,b]$ — також лінійний простір. Дійсно, визначимо суму функцій з цього простору та добуток функції на скаляр рівністю (1.2).



Треба перевірити, що так визначені функції αx та $x + y$ також належать $L_2[a,b]$. Маємо

$\int_a^b (\alpha x)^2(t)dt = \alpha^2 \int_a^b x^2(t)dt < +\infty$, оскільки, за умовою, $\int_a^b x^2(t)dt < +\infty$.

Далі для $x \in L_2[a, b],\ y \in L_2[a, b] => (x + y) \in L_2[a, b]$, наприклад, тому, що має місце нерівність

$$(u + v)^2 \leq 2(u^2 + v^2),\ \forall\, u, v\ \in\ R^1. \qquad (1.3)$$

Завдяки (1.3) маємо

$$\int_a^b (x(t)+y(t))^2 dt \le \int_a^b 2[x^2(t)+y^2(t)]dt = 2\left[\int_a^b x^2(t)dt + \int_a^b y^2(\mathrm{t})\mathrm{dt}\right] < +\infty$$

Підмножина $M \subset L$ називається *підпростором* L, якщо вона також є лінійним простором.

Приклади підпросторів.

1)

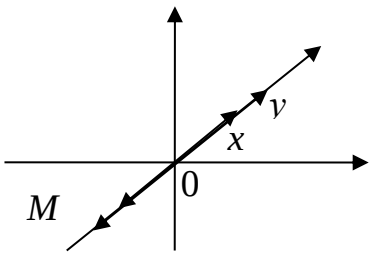


Рис.3.2

$L = R^2$; множина (двовимірних) векторів, що мають початок у початку координат і кінці — на фіксованій прямій, що проходить через початок координат, утворюють підпростір в L. (Пояснимо, що саму множину R^2 можна розглядати як точкову множину, а можна — як векторний простір. Саме в останньому сенсі R^2 і розглядається у даному прикладі).

2) Множина розв'язків однорідної системи лінійних рівнянь.

У загальному випадку система лінійних рівнянь має вигляд

$$\begin{matrix} a_{1,1}\,x_1 & + & a_{1,2}\,x_2 & + \ldots + & a_{1,n}\,x_n & = b_1 \\ \ldots & & \ldots & + \ldots + & \cdots \quad \cdots & \\ a_{m,1}\,x_1 & + & a_{m,2}\,x_2 & + \cdots + & a_{m,n}\,x_n & = b_m \end{matrix}, \qquad (1.4)$$

де $a_{ij}, b_j, i = 1, \ldots, n, j = 1, \ldots, m$ — відомі числа. Якщо ввести матрицю $A = A_{m\times n} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & \ldots & \ldots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}$ і вектори $b = \begin{pmatrix} b_1 \\ \ldots \\ b_m \end{pmatrix}, x = \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix}$,

то систему (1.4) можна подати у векторно-матричному вигляді:

$$Ax = b \tag{1.5}$$

Множина L розв'язків системи рівнянь (1.4) ((1.5)) є, очевидно, деякою підмножиною простору R^n. При $b = 0$ система (1.4) ((1.5)) називається однорідною. Вона має вигляд

$$Ax = 0, \tag{1.6}$$

де $0 = \begin{pmatrix} 0 \\ \dots \\ 0 \end{pmatrix}$.

Множина розв'язків системи (1.6) *утворює підпростір в просторі* R^n . Це випливає із співвідношень

$$A(x + y) = Ax + Ay, A(\alpha x) = \alpha A(x), \text{ де } x, y \in R^n, \alpha \in R^1,$$

завдяки яким якщо x та y задовольняють рівняння (1.5), то $x + y$ та αx також задовольняють (1.5).

Лінійні комбінації системи векторів

Нехай L — лінійний простір, $x^1, x^2, \dots x^k \in L$; $c_1, c_2, \dots, c_k$ — коефіцієнти; у виразі $x^1, x^2, \dots x^k$цифри 1,2,…k — верхні індекси (а не показники степенів). Така нумерація векторів вживається для виключення плутанини з позначеннями координат вектору $x \in R^n$, що використовуються у вищенаведених формулах. В подальшому, якщо невірне трактування позначень малоймовірне, вживається нумерація векторів з L і за допомогою нижніх індексів.

Лінійною комбінацією векторів $x^1, x^2, \dots x^k$ з коефіцієнтами $c_1, c_2, \dots, c_k$ називається вираз:

$$c_1 x^1 + c_2 x^2 + \dots + c_k x^k.$$

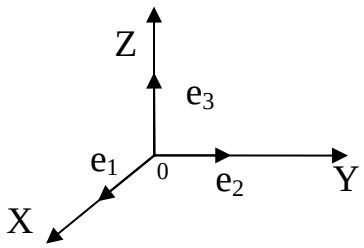
Лінійною оболонкою $\mathcal{L}(V)$ системи векторів $V \subset L$ називається сукупність всіляких лінійних комбінацій векторів цієї системи. Очевидно, лінійна оболонка системи векторів утворює підпростір в L.

Приклади лінійних оболонок систем векторів

1) Якщо V — система, що складається з єдиного вектору x, то $\mathcal{L}(V)$ є множиною $\{cx, c \in K\}$, де c — коефіцієнти. При $L = R^2$ маємо $\mathcal{L}(V) = M$, де

M — підпростір, що зображений на рис. 3.2.

2) Сам векторний простір R^2 (інакше, площина XOY) є лінійною оболонкою векторів $e_1 = (1,0), e_2 = (0,1)$.
3) Векторний простір R^3 є лінійною оболонкою векторів $e_1 = (1,0,0), e_2 = (0,1,0), e_3 = (0,0,1)$ — див. рис. 3.3.



4)

4) Розглянемо множину P_n всіх поліномів від $t \in R^1$ степеня не більше n (n — ціле невід'ємне), тобто функцій виду

$$a_n t^n + a_{n-1} t^{n-1} + \cdots + a_1 t + a_0;\ a_i \in R^1, i = 1,2, \ldots, n\ .$$

Легко бачити, що множина P_n утворює підпростір в множині $C(-\infty, \infty)$. З іншого боку, P_n є лінійною оболонкою функцій

$$\left.\begin{array}{c} x_n = x_n(t) = t^n \\ x_{n-1} = x_{n-1}(t) = t^{n-1} \\ \cdots \\ x_1 = x_1(t) = t \\ x_0 = 1 \end{array}\right\},$$

тобто $P_n = \mathcal{L}(x_0, x_1, \ldots, x_n)$. Тут вектори x_i простору P_n (що є функціями від t) пронумеровано нижніми індексами, на відміну від виразу (1.8).

Комбінація $c_1x^1 + c_2x^2 + \cdots + c_kx^k$ називається нетривіальною, якщо серед $c_1, c_2, \ldots, c_n$ є такі, що не дорівнюють нулю, в іншому випадку комбінація називається тривіальною.

Система векторів $x_1, x_2, \ldots, x_k$ називається лінійно залежною, якщо існує нетривіальна лінійна комбінація векторів, що дорівнює нульовому вектору. В іншому випадку система називається лінійно незалежною.

Якщо система $x_1, x_2, \ldots, x_k$ лінійно незалежна і має місце рівність

$$c_1x^1 + c_2x^2 + \cdots + c_kx^k = 0,$$

тоді

$$c_1 = c_2 = \ldots = c_k = 0$$

Приклади лінійно залежних і лінійно незалежних систем.

1) У просторі $L=R^3$ вектори $\mathrm{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathrm{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \mathrm{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ утворюють лінійно незалежну систему. Дійсно, лінійна комбінація цих векторів виражається через координати і коефіцієнти їх лінійної комбінації наступним чином:

$$c_1e_1 + c_2e_2 + c_3e_3 = \begin{pmatrix} c_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ c_2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ c_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}.$$

При цьому

$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = 0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow c_1 = c_2 = c_3 = 0$, а отже, e_1, e_2, e_3 - лінійно незалежні вектори.

2) Більш загальним чином, у просторі R^N вектори $e_1 = \begin{pmatrix} 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \end{pmatrix}, \dots, e_N = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \end{pmatrix}$ утворюють лінійно незалежну систему, що перевіряється цілком аналогічно.
3) У просторі P_n(лінійних оболонок) вектори x_0, x_1,…, x_n (функції від t) утворюють лінійно незалежну систему. Дійсно, оскільки поліном степеня n має щонайбільше n дійсних коренів, то рівність
$$c_nt^n + c_{n-1}t^{n-1} + \cdots + c_1t + c_0 \equiv 0$$
може мати місце лише у випадку $c_0 = c_1 = \cdots = c_n = 0$.
4) У просторі $C([-\infty, \infty])$ функції $x = x(t) = sin^2t, y = y(t) = cos^2t, z = z(t) \equiv 1$ утворюють лінійно залежну систему, що одразу випливає з відомого співвідношення $sin^2t + cos^2t = 1 \; \forall t \in R^1$.

Означення лінійної залежності і незалежності було надане для систем, що складаються із скінченної кількості векторів. Дамо відповідне означення для нескінченних систем.

Нескінченна система векторів називається лінійно незалежною, якщо будь-яка скінченна підсистема її є лінійно незалежною.

В просторі $C([a,b])$ система $x_n = x_n(t) = t^n$ є лінійно незалежною.

Поняття про базис лінійного простору

Лінійно незалежна система векторів $\{x_i\}$ лінійного простору незивається його алгебраїчним базисом (або базисом Гамеля), якщо цей простір є лінійною оболонкою системи $\{x_i\}$.

Іншими словами, система $\{x_i\}$ є алгебраїчним базисом простору L, якщо кожний елемент L є (скінченною) лінійною комбінацією тих чи інших векторів даної системи.

Як це доводиться у більш фундаментальних курсах, алгебраїчний базис існує у довільному лінійному просторі. Якщо простір має базис, що складається із скінченної кількості векторів, то такий простір називається *скінченновимірним*, у протилежному випадку — *нескінченноовимірним*. Кількість векторів базису простору називається його *розмірністю*. Розмірність простору L позначається $dim\ L$. Взагалі кажучи, лінійний простір має не єдиний базис. Можна довести [2] , що розмірність простору не залежить від конкретного вибору базису.

Приклади базисів у лінійних просторах

1) У просторі R^N базис утворюють вектори $e_1 = \begin{pmatrix} 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \end{pmatrix}, \dots, e_N = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \end{pmatrix}$ (перевірте це). Таким чином, розмірність простору R^N дорівнює N ($dim\ R^N = N$).

2) У просторі $P([a,b])$ всіх поліномів від $t \in [a,b]$ базис утворює система функцій $x_n = x_n(t) = t^n, n = 1,2,\ldots..$ Цей простір є нескінченновимірним ($dim\, P([a,b]) = \infty$).

1.2 Лінійне програмування (ЛП)

Знайти точку $x = (x_1, x_2, \ldots, x_n)$ *n- вимірного* простору R^n, яка мінімізує (максимізує) лінійну функцію

$$f(x) = \langle c, x \rangle = \sum_{i=1}^{n} c_i x_i, \qquad c = (c_1, \ldots, c_n) \tag{2.1}$$

при виконанні системи обмежень

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = 1, \ldots, r\,, \tag{2.2}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, i = r+1, \ldots, r+l\,, \tag{2.3}$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i, i = r+l+1, \ldots, r+l+m\,, \tag{2.4}$$

$$x_j \ge 0, j = 1, \ldots, k, k \le n. \tag{2.5}$$

Тут a_{ij}, b_i, c_i — константи. Множина точок з простору R^n, для яких виконуються співвідношення (2.2) — (2.5), називаються ще припустимою областю задачі ЛП [3, 33, 40] .

Транспортна задача

Запаси деякого однорідного продукту розподілені на кількох базах постачання, і цей продукт потрібно доставити до кількох пунктів призначення. Відома вартість перевезень продукту між базами постачання і

пунктами призначення. Задача: визначити, яку кількість продукту треба перевезти з кожної бази постачання до кожного пункту призначення, що забезпечити потреби кожного пункту, причому так, щоб сумарна вартість перевезень була мінімальною.

При розв'язанні даної задачі вважається відомими дані:

m — кількість баз постачання

$n -$ кількість пунктів призначення

a_i — кількість одиниць даного продукту на i-й базі ($1 \leq i \leq m$

b_j — потреба j-го пункту призначення в даному продукті

c_{ij} — вартість перевезення одиниці продукту з i-ї бази в j-й пункт призначення

Нехай x_{ij} — кількість одиниць продукту, що планується для перевезення $i \to j$. Тоді сумарна вартість перевезень дорівнюватиме

$$S(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}.$$

Треба визначити величини x_{ij} таким чином, щоб величина $S(x)$ виявилася мінімальною з можливих. При цьому на величини x_{ij} накладаються очевидні обмеження:

A) $x_{ij} \geq 0, 1 \leq i \leq m, 1 \leq j \leq n$;

B) $\sum_{i=1}^{m} x_{ij} = b_j\,, 1 \leq j \leq n$ (умова задоволення потреб кожного пункту);

C) $\sum_{j=1}^{n} x_{ij} \leq a_i, 1 \leq i \leq m$ (неперевищення розміру запасів на базах).

Задача про планування виробництва

Підприємство виготовляє кілька видів продукції, і для її виготовлення використовується кілька видів ресурсів. В результаті своєї діяльності підприємство одержує певний прибуток. Треба скласти такий план виробництва, який задовольняє обмеження на кількість виготовлених виробів і при цьому гарантує максимальну величину прибутку.

Дані, що вважаються відомими:

n — кількість видів продукції;

m — кількість видів ресурсу;

b_i — кількість ресурсу i-го типу ($1 \le i \le m$);

c_j — прибуток від реалізації j-го типу продукції;

a_{ij} — витрати і-го типу ресурсів на одиницю j-го типу продукції;

d_j, D_j — відповідно, найменша і найбільша кількість обсягів виробництва j-го типу продукції.

Скласти план $x = (x_1, \dots, x_n)$ виробництва, при якому задовольняються задані обмеження на випуск і забезпечується якомога більший прибуток.

Модель даної задачі у формі задачі ЛП наступна:

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

$$\sum_{j=1}^{m} a_{ij} x_j \le b_i, 1 \le i \le m;$$

$$d_j \le x_j \le D_j.$$

Деякі властивості припустимої області задачі ЛП

Зауважимо, що кожне з обмежень (2.2) — (2.4) може бути представленим у будь-якій іншій формі з числа даних обмежень. Наприклад,

для представлення обмеження (2.3) у формі (2.2) досить ввести додаткові невід'ємні змінні $w_{r+1}, \dots, w_{r+l}$ і замінити рівності (2.3) рівностями

$$\sum_{j=1}^{n} a_{ij}x_j + w_i = b_i, i = r+1, \dots, r+l\,, \tag{2.6}$$

причому у вираз функції мети f зазначені змінні w_i ввести з нульовими коефіцієнтами:

$$f(x) = \sum_{i=1}^{n} c_i x_i + 0 \cdot w_{r+1} + \cdots 0 \cdot w_{r+l}\,. \tag{2.7}$$

Неважко побачити, що задача ЛП у формі (2.7),(2.6) еквівалентна задачі ЛП у початковій формі (2.1),(2.3). Тобто якщо розв'язком задачі (2.1),(2.3) є вектор $x_0 = (x_1^0, \dots x_n^0)$,то в розв'язку задачі (2.7),(2.6) перші n компонентів будуть якраз $(x_1^0, \dots x_n^0)$. І навпаки, якщо є розв'язок задачі (2.7),(2.6), то його перші n компонентів дадуть розв'язок задачі (2.1),(2.3).

Множина $\{x = (x_1, x_2, \dots, x_n) \in R^n\}$, для елементів якої виконується нерівність

$$\sum_{i=1}^{n} a_i x_i \le b, \tag{2.8}$$

де a_i та b — константи, називається півпростором в R^n. Підмножина R^n, що задається рівністю

$$\sum_{i=1}^{n} a_i x_i = b, \tag{2.9}$$

називається гіперплощиною.

Зрозуміло, що множина, яка визначається нерівністю

$$\sum_{i=1}^{n} a_i x_i \geq b, \tag{2.10}$$

також є півпростором, оскільки (2.10) можна записати і вигляді

$$\sum_{j=1}^{n} (-a_{ij}) x_j \leq -b_i.$$

Кожна гіперплощина може бути представлена як перетин двох півпросторів. Наприклад гіперплощина (2.9) є перетином півпросторів (2.8) і (2.10). Зауважимо, що при $n = 2$ гіперплощина є прямою в декартовій площині $X_1 O X_2$ з рівнянням виду

$$a_1 x_1 + a_2 x_2 = b, \tag{2.11}$$

а півпростір $a_1 x_1 + a_2 x_2 \leq b$ — півплощина, що знаходиться по одну сторону від зазначеної прямої.

Зауважимо, що кожна гіперплощина (2.9), (зокрема, (2.11)) є лінією рівня b функції $f(x) = \sum_{i=1}^{n} a_i x_i$ (при $n = 2: f(x) = a_1 x_1 + a_2 x_2$).

Відрізком в просторі R^n, що з'єднує дві точки u, v, називається множина точок виду

$$\lambda u + (1 - \lambda) v, \tag{2.12}$$

де $0 \leq \lambda \leq 1$. Внутрішньою точкою відрізка (2.12) називається будь-яка точка цього відрізку, що відповідає значенню $\lambda \neq 0$ або 1.

Переписавши вираз (2.12) у вигляді

$$v + \lambda(u - v), \tag{2.12′}$$

неважко переконатися за допомогою рисунку для двовимірного простору (декартова площина), що множину (2.12) можна уявити як множину точок, котра утворюється з точки v додаванням до неї вектора $\lambda(u - v)$ при зміні коефіцієнта λ від 0 до 1 (при $\lambda = 1$ одержуємо точку $v + (u - v) = u$).

Множина називається опуклою, якщо з кожною парою своїх точок вона містить весь відрізок, що з'єднує ці точки.

Зауважено, що перетин двох і більше опуклих множин є опуклою множиною. Множина, що є перетином скінченної кількості півпросторів, називається багатогранною (або многогранною) множиною. Обмежена багатогранна множина називається багатогранником (многогранником).

Допустима множина задачі лінійного програмування є опуклою множиною.

Вершиною багатогранника X в R^n називається така точка $x \in X$, яка *не є внутрішньою точкою* ніякого відрізку, що з'єднує дві різні точки з X. Вершини багатогранників обмежень називаються ще опорними планами задач ЛП.

Наприклад, точка x_0 є внутрішньою точкою відрізка, що з'єднує точки x_1,x_2 даного многокутника. При цьому легко бачити, що для жодної із вершин A,B,C,D,E,O неможливо знайти відрізок з кінцями у даному многокутнику, такий, щоб дана вершина була внутрішньою точкою такого відрізка.

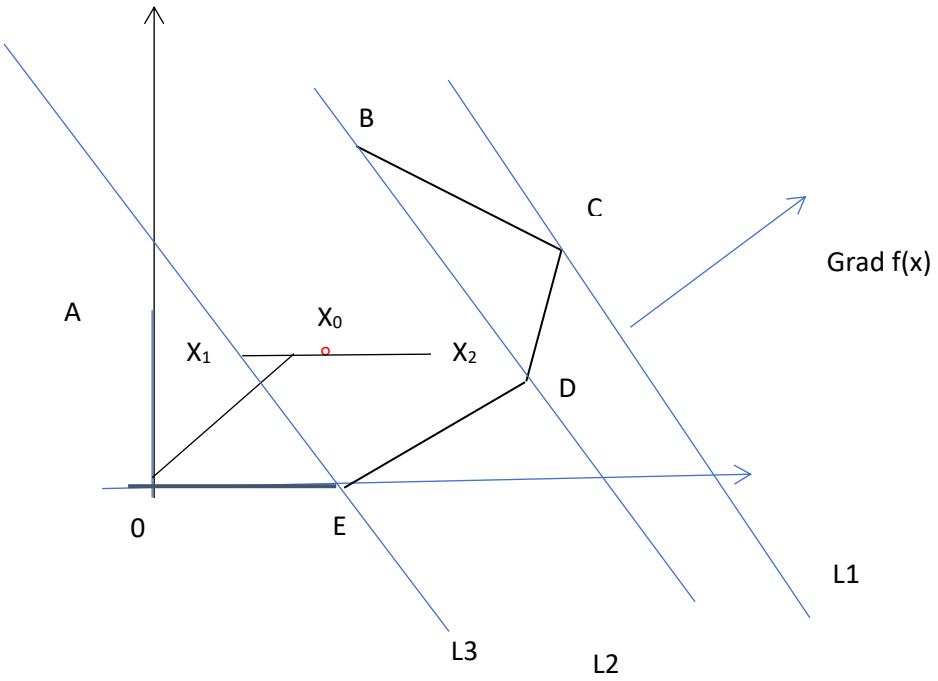


Рис. 3.4

Розглядаючи двовимірну задачу лінійного програмування, з геометричних міркувань неважко прийти до висновку, що серед точок розв'язку цієї задачі обов'язково буде якась вершина многогранника обмежень. Дійсно, нехай, наприклад, задача полягає в визначенні точки мінімуму з обмеженнями у вигляді многокутника з рисунку 1. Прямі L1, L2, L3 за умовою є лініями рівня функції $f(x_1, x_2) = a_1x_1 + a_2x_2$, де a_1, a_2 — додатні коефіцієнти. Останнє зумовлює напрямок градієнту цієї функції ($grad\ f(x_1, x_2) = (a_1, a_2)$). Оскільки градієнт вказує на напрямок зростання функції, то для знаходження точки мінімуму треба зміщатися по лініях рівня f в напрямку, протилежному напрямку градієнта. Найменше з можливих значень функції f при умові знаходження в області обмежень буде на прямій L3. Єдина її точка перетину з многокутником даної задачі є точка О. Ця точка (у даному разі О =(0,0)) і є розв'язком.

До вказаного висновку можна прийти строго алгебраїчним шляхом. Більше того, виявляється, що в задачі ЛП довільної розмірності глобальний (для многокутника обмежень) екстремум досягається в тій чи іншій вершині многокутника. Крім того, число вершин в довільній множині обмежень задачі ЛП обов'язково є скінченним. Доведення цих фактів можна знайти в літературі [3 – 7, 33].

Із сказаного випливає, що в принципі розв'язати довільну задачу ЛП можна, обчислюючи значення функції мети в вершинах многокутника обмежень і відбираючи такі вершини, в яких вказані значення є екстремальними.

Але при цьому підході треба знати координати згаданих вершин (опорних планів). До того ж, виявляється, що не обов'язково перебирати всі вершини. Зупинитись можна на такому опорному плані, в якому значення функції мети є «не гіршим», ніж у так званих сусідніх вершинах. Реалізує вказані дії широко відомий симплекс-метод розв'язку задачі ЛП.

Скористатися цим методом можна за допомогою відповідного програмного комп'ютерного забезпечення. Нижче ми розглянемо основні принципові положення алгоритму даного методу.

Перше питання: як серед допустимих точок многокутника обмежень виділити вершини (опорні плани).

Будемо вважати, що задача сформульована у формі (2.1), (2.2) (і (2.5)). Для більшої зручності запишемо знову вирази функції мети f і множини обмежень X типу (2.2), (2.5):

$$f(x) = \sum_{i=1}^{n} c_i x_i, \qquad c = (c_1, \dots, c_n) \tag{2.13}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = 1, \dots, r\ , \tag{2.14}$$

$$x_j \geq 0, j = 1, \dots, n. \tag{2.15}$$

Формулювання умов задачі ЛП називається стандартною формою цієї задачі. Введемо матрицю A обмежень (2.14); $A = \left(a_{ij}\right)_{m \times n}$. Введемо ще позначення: A_j — j-й стовпчик даної матриці і $b = \begin{pmatrix} b_1 \\ \dots \\ b_m \end{pmatrix}$ — вектор-стовпчик правих частин системи рівностей (2.14). Відмітимо, що у введених позначеннях система рівностей (2.14) приймає вид

$$\sum_{j=1}^{n} A_j x_j = b. \tag{2.15}$$

Допустимий розв'язок x (за означенням, це просто елемент множини обмеженьX) називається *базисним* якщо система векторів умов (тобто підсистема сукупності векторів $\{A_j\}$), що відповідає його додатним компонентам, є лінійно незалежною системою.

Наприклад, якщо $x = (2,5,0,0)$, то x буде базисним елементом (інакше, базисною точкою), якщо вектори A_1, A_2 є лінійно незалежними.

Нагадаємо, що система векторів $\{A_j, j = 1,..,n\}$ називається лінійно незалежною (або вектори $A_j, j = 1,..,n$ називаються лінійно незалежними), якщо рівність $a_1A_1 + \cdots + a_nA_n = 0$, в якій $a_1, \ldots, a_n$ — коефіцієнти, можлива лише, коли всі $a_1, \ldots, a_n$ дорівнюють 0.

З теореми відомо, що допустимий розв'язок задачі ЛП є вершиною многокутника обмежень тоді і тільки тоді, коли зазначений розв'язок є базисним.

Проілюструємо, як можна це використати для розв'язання задач ЛП на прикладі (пошук опорних планів). Знайти опорний план задачі ЛП, допустима множина якої задається наступними обмеженнями:

$$\begin{gathered} 3x_1 + x_2 + 2x_3 + x_4 - 2x_5 = 5, \\ 6x_1 + x_2 + 3x_3 + 2x_4 - 4x_5 = 9, \\ 10x_1 + x_2 + 6x_3 + 3x_4 - 7x_5 = 14, \\ x_i \geq 0, i = 1, \ldots,5. \end{gathered} \tag{2.16}$$

Матриця A. Візьмемо три довільні вектори-стовпці матриці A, наприклад, A_1, A_2, A_3 . Дослідимо ці вектори на лінійну незалежність. Це можна зробити, підрахувавши визначник

$$det\begin{pmatrix} 3 & 1 & 2 \\ 6 & 1 & 3 \\ 10 & 1 & 6 \end{pmatrix} = 18 + 30 + 12 - 20 - 9 - 36 = -5 \neq 0.$$

Оскільки визначник не дорівнює 0, вектори лінійно незалежні. Тепер треба з'ясувати, чи будуть розв'язки системи (2.16), що відповідають значенням

$$x_4 = x_5 = 0, \tag{2.17}$$

задовольняти умові $x_1 > 0, x_2 > 0, x_3 > 0$. За умови (2.17) система (2.16) приймає вид

$$3x_1 + x_2 + 2x_3 = 5,$$

$$6x_1 + x_2 + 3x_3 = 9,$$

$$10x_1 + x_2 + 6x_3 = 14.$$

Розв'язавши цю систему, наприклад, методом Гаусса, одержимо:

$$x_1 = \frac{7}{5}, x_2 = \frac{6}{5}, x_3 = -\frac{1}{5}.$$

Не виконана умова невід'ємності. Тобто знайдена точка

$(\frac{7}{5}, \frac{6}{5}, -\frac{1}{5}, 0,0)$ не є допустимою точкою задачі.

Візьмемо інші вектори-стовпці, наприклад, A_1, A_2, A_4. Аналогічним чином, обчислюючи

$$det\begin{pmatrix} 3 & 1 & 1 \\ 6 & 1 & 2 \\ 10 & 1 & 3 \end{pmatrix} = 1 \neq 0,$$

переконуємось в лінійній незалежності цих векторів. Підставляючи в систему (2.16) значення $x_3 = x_5 = 0$ і розв'язуючи відповідну систему, знаходимо її розв'язок:

$$x_1 = 1, x_2 = 1, x_3 = 1,$$

так що точка (1,1,0,1,0) виявилися (згідно з теоремою про допустимий розв'язок задачі ЛП) вершиною многокутника обмежень (опорним планом нашої задачі). В принципі, перебираючи всі можливі варіанти виконання вищеописаних дій, можна знайти всі опорні плани задачі і обчислити значення в них функції мети. Але такий шлях є надто громіздким, уникнути

зайвих викладок можна завдяки вже згаданому вище алгоритму симплекс-методу. Ідея цього методу якого полягає в тому, щоб спочатку знайти якусь вершину многогранника обмежень X, а потім від неї перейти до іншої вершини, в якій значення функції не більше для задачі мінімізації (не менше для задачі максимізації), ніж у попередній, і таким чином за скінченну кількість кроків буде знайдено розв’язок задачі лінійного програмування або з’ясовано, що вона не має розв’язків. Дослідженню задач лінійного програмування і методам їх розв’язування присвячена значна кількість робіт [27, 33].

Висновки до розділу 1

У даному розділі дається опис множин, які часто використовуються як множини обмежень в задачах оптимізації. Сформульована одна з поширених задач теорії оптимізації – задача лінійного програмування. У загальних рисах і на прикладі двовимірної задачі наведено основну ідею поширеного методу її розв’язання – симплекс методу.

Розділ 2
МАТЕМАТИЧНІ АСПЕКТИ МЕТОДІВ ОПТИМІЗАЦІЇ

2. 1. Задача на умовний екстремум

В математичному аналізі традиційно розглядається так звана класична задача на умовний екстремум: знайти екстремуми функції

$$y = f(x) = f(x_1, x_2, \dots, x_n)$$

при умові, що її аргументи зв'язані між собою співвідношеннями

$$g_i(x) = g_i(x_1, x_2, \dots, x_n) = 0,\ i = 1, \dots, , k, \qquad \text{(A)}$$

де функції $f(x)$, $g_i(x)$, i = 1,…k, визначені на деякій відкритій множині $G \subset R^n$. Співвідношення (2.1) називаються умовами з в'язку або рівняннями зв'язку (див., наприклад, [3,8 - 10]

Точка $x^{(0)} = \left(x_1^{(0)}, \dots, x_n^{(0)}\right)$ називається точкою локального умовного мінімуму (максимуму) функції $y = f(x)$ відносно рівнянь зв'язку (2.1), якщо існує такий окіл точки $x^{(0)}$, що для будь-якої точки x цього околу, координати якої задовольняють рівняння (А), виконується нерівність

$$f\left(x^{(0)}\right) \leq (\geq) f(x).$$

Класична необхідна умова екстремуму функції кількох змінних полягає, як відомо, в наступному.

Якщо функція

$$y = f(x) = f(x_1, x_2, \dots, x_n)$$

в точці $x^{(0)}$ має локальний екстремум і диференційована в ній, то має місце рівність

$$grad\ f\left(x^{(0)}\right) = 0, \qquad \text{(B)}$$

де $grad\, f(x^{(0)}) = \left(\frac{\partial f}{\partial x_1}(x^{(0)}), \dots, \frac{\partial f}{\partial x_n}(x^{(0)})\right)$.

Умова (В) є лише необхідною умовою екстремуму. Її застосування для знаходження екстремальних точок в задачах з обмеженнями (типу (А)) може входити лише в початкові етапи дослідження. Методи, що налаштовані на реальне розшукання зазначених точок, потребують значно більшої кількості інформації відносно поведінки функції f. До таких засобів розв'язання екстремальних задач відносяться, наприклад, описані нижче методи Лагранжа і Якобі.

2.2. Пошук умовного екстремуму диференційованих функцій методом Лагранжа

Нехай $f: R^N \to R$ — дійсна функція аргументу $x = (x_1, \dots, x_N) \in R^N$. Розглядається задача пошуку точок її екстремумів (локальних чи глобальних) за умови наявності обмежень на значення x виду деяких рівностей. Формальний запис задачі наступний:

$$f(x) \to \min(\max) \tag{1}$$

$$f_1(x) = 0,$$

$$\dots\dots\dots \tag{2}$$

$$f_m(x) = 0,$$

де $f_1, \dots, f_m$ — деякі дійсні функції. За умовою, всі функції $f, f_1, \dots, f_m$ мають достатню кількість похідних. Рівняння (2) мають ще назву рівнянь або рівностей зв'язку.

У даному підрозділі для розв'язання поставленого питання використовується так званий метод множників Лагранжа (інші назви — метод Лагранжа, метод невизначених коефіцієнтів).

Зазначений метод базується на наступному положенні.

Нехай x_0 — точка умовного екстремуму функції f при виконанні рівностей (2). Тоді в цій точці вектори (градієнти)

$$grad\, f(x_0), gradf_1(x_0), \dots, gradf_m(x_0)$$

лінійно залежні, тобто існують числа $\lambda_0, \lambda_1, \dots, \lambda_m$, не всі рівні 0 і такі, що

$$\lambda_0 grad\, f(x_0) + \lambda_1 gradf_1(x_0) + \cdots + \lambda_m gradf_m(x_0) = 0. \quad (3)$$

Якщо в точці умовного екстремуму градієнти $gradf_1(x_0), \dots, gradf_m(x_0)$ лінійно незалежні, то існують такі числа $\lambda_1, \dots, \lambda_m$, що в цій точці

$$grad\, f(x_0) + \sum_{j=1}^{m} \lambda_j gradf_j(x_0) = 0, \quad (4)$$

тобто в цій точці градієнт функції f є лінійною комбінацією градієнтів функцій $f_j, j = 1, \dots, m$.

В координатній формі запис рівності (4) має вигляд

$$\frac{\partial f}{\partial x_i}(x_0) + \sum_{j=1}^{m} \lambda_j \frac{\partial f_j}{\partial x_i}(x_0) = 0, i = 1, \dots, N. \quad (5)$$

Функція

$$L(x) \stackrel{\text{def}}{=} f(x) + \sum_{j=1}^{m} \lambda_j f_j(x) \quad (6)$$

називається функцією Лагранжа задачі (1),(2), а числа $\lambda_1, \dots, \lambda_m$ — коефіцієнтами або множниками Лагранжа.

Зауважимо, що в літературі функція Лагранжа часто записується із знаком (-) перед сумою в правій частині. Легко бачити, що на визначення точок екстремуму це не впливає.

Очевидно, точка x_0, котра задовольняє систему рівностей (5), є розв'язком системи рівнянь

$$\frac{\partial L}{\partial x_i}(x) = 0, i = 1, \dots, N. \quad (7)$$

Співвідношення (7) означає, що коли точка x_0 є точкою *умовного* екстремуму функції f, то вона є стаціонарною точкою функції Лагранжа $L(x)$ і може бути знайдена за правилами дослідження функцій на наявність з*вичайних* (*не умовних*) точок екстремумів.

Зауважимо, що точки умовного екстремуму функцій f і L співпадають, як би не були визначені коефіцієнти $\lambda_1, \dots, \lambda_m$. Ми вибираємо їх таким чином, щоб дана точка умовного екстремуму виявилася стаціонарною точкою функції *f*.

Реалізація методу Лагранжа.

Для розшуку точок умовного екстремуму слід розв'язати систему $N + m$ рівнянь (2), (7) відносно невідомих $x_1, \dots, x_N$, виключивши змінні $\lambda_1, \dots, \lambda_m$. *Всі точки умовного екстремуму функції f будуть міститися серед знайдених таким чином розв'язків.* Питання про те, які саме з них будуть фактично шуканими точками, можуть вимагати додаткових міркувань.

Наприклад, $N = 2$, $f(x, y) = xy$. Дослідити на умовний екстремум функцію f методом невизначених коефіцієнтів за умови $x - y = 0$.

Оскільки кількість обмежень дорівнює 1, то досліджування на лінійну незалежність градієнтів функцій обмежень непотрібне. Вводимо функцію Лагранжа $L(x,y)$:

$$L(x, y) = xy + \lambda(x - y)$$

і досліджуємо її на безумовний екстремум, додаючи до відповідних співвідношень рівняння зв'язку. Маємо систему рівнянь

$$\frac{\partial L}{\partial x} = y + \lambda = 0$$

$$\frac{\partial L}{\partial y} = x - \lambda = 0$$

$$x - y = 0.$$

З останнього рівняння маємо $x = y$, тому перші два дадуть

$$x + \lambda = 0$$

$$x - \lambda = 0,$$

звідки $x = 0$, тому і $y = 0$. Отже єдиною підозрілою на екстремум точкою є $x_0 = (0,0)$. Маємо $f(0,0) = 0 \cdot 0 = 0$, При цьому $f(x,0) = f(0,y) = 0$, але точки виду $(x,y), x \neq y$ не входять в область обмежень, а для точок виду (x,x) маємо $f(x,x) > 0$ при $x \neq 0$, отже знайдена точка $x_0 = (0,0)$ є точкою строгого мінімуму на всій множині обмежень.

Розглянемо випадок, коли на кривій $y = x^2, z = x^2$ в просторі $R^3 = R^3(x,y,z)$знайти точку $\boldsymbol{a}$ що є найближчою до точки **c** = (0,0,1).

Відстань $\|u - v\|$ між точками $u = (u_1, \dots, u_N)$ і $v = (v_1, \dots, v_N)$ в N-вимірному просторі знаходиться за формулою

$$\|u - v\| = \sqrt{(u_1 - v_1)^2 + \cdots + (u_N - v_N)^2}. \qquad (8)$$

При цьому якщо відстань $\|u - v\|$ між точками $u \in U, v \in V$ є мінімальною відстанню між точками множин U, V (тобто $\|u - v\| = min_{\alpha \in U, \beta \in V}\|\alpha - \beta\|$),то $\|u - v\|^2$ буде мінімальним *квадратом* відстаней між точками цих множин і навпаки.

Позначимо x, y, z координати шуканої точки $\boldsymbol{a}$. Будемо шукати $\boldsymbol{a}$ з умови мінімуму *квадрату* відстані до точки c (що, згідно з зауваженням 1, є еквівалентним умові задачі). Враховуючи сказане, застосовуючи формулу (8) при $N = 3$, одержуємо формальний запис умови задачі:

Знайти мінімальне значення функції

$$f \stackrel{\text{def}}{=} \|a - c\|^2 = x^2 + y^2 + (z - 1)^2$$

за умов

$$\varphi_1(x,y,z) \stackrel{\text{def}}{=} y - x^2 = 0,$$

$$\varphi_2(x,y,z) \stackrel{\text{def}}{=} z - x^2 = 0.$$

Будуємо функцію Лагранжа даної задачі:

$$L(x,y,z) = x^2 + y^2 + (z-1)^2 + \lambda_1(x^2 - y) + \lambda_2(x^2 - z).$$

Знаходимо стаціонарні точки цієї функції з умови $gradL = 0$:

$$\begin{cases} \frac{\partial L}{\partial x} = 2x + 2\lambda_1 x + 2\lambda_2 x = 0, \\ \frac{\partial L}{\partial y} = 2y - \lambda_1 = 0, \\ \frac{\partial L}{\partial z} = 2(z-1) - \lambda_2 = 0. \end{cases} \quad (9)$$

З останніх двох рівнянь маємо $\lambda_1 = 2y, \lambda_2 = 2(z-1)$. Підставивши ці співвідношення у перше рівняння системи (9) і врахувавши рівняння зв'язку, одержимо систему рівностей

$$\begin{cases} 2x + 4xy + 4(z-1)x = 0, \\ x^2 - y = 0, \\ x^2 - z = 0. \end{cases} \quad (10)$$

Очевидним коренем цієї системи є $x = 0$. Згідно з другим і третім рівняннями останньої системи одержуємо точку $a^1 = (0,0,0)$. В цій точці маємо значення функції *f*:

$$f(a^1) = f(0,0,0,) = 1.$$

Якщо $x \neq 0$, то з першого рівняння (10) маємо

$$1 + 2y + 2(z-1) = 0,$$

Звідки, враховуючи друге і третє рівняння системи (10), одержуємо

$$1 + 2x^2 + 2(x^2 - 1) = 0,$$

або $4x^2 = 1$, отже $x = \pm\frac{1}{2}$. Знову з другого і третього рівнянь системи (10), знаходимо $y = z = \frac{1}{4}$. Знайдено дві нові точки

$$a^2 = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right), a^3 = \left(-\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right).$$

В кожній з цих точок функція f приймає однакові значення:

$$f(a^2) = f(a^3) = \frac{1}{4} + \frac{1}{16} + \frac{9}{16} = \frac{7}{8} < f(a^1) = 1.$$

Отже, саме в точках a^2, a^3 (а не в точці a^1) досягається шуканий мінімум.

2.3. Пошук умовного екстремуму диференційованих функцій методом Якобі

Нижче у даному розділі для розв'язання задачі (1), (2) використовується метод зниження розмірності задачі за рахунок вираження частини змінних через інші змінні (метод Якобі).

Позначимо *E* множину точок з R^N, для яких виконуються всі рівності зв'язку (2).

Вважаємо, що m змінних з числа $x_1, \dots, x_N$ за допомогою зазначених рівнянь зв'язку можна виразити (хоча б локально) через інші змінні.

Без обмеження загальності будемо вважати, що мова йде про змінні $x_1, \dots, x_m$, так що у такому випадку маємо *m* рівностей

$$x_1 = \varphi_1(x_{m+1}, \dots, x_N),$$

$$x_2 = \varphi_2(x_{m+1}, \dots, x_N),$$

…….. 11)

$$x_m = \varphi_m(x_{m+1}, \dots, x_N),$$

де $\varphi_1, \dots \varphi_m$ — деякі функції від $N - m$ змінних.

Наприклад, якщо існує точка $x_0 = (x_1^0, \dots, x_m^0, x_{m+1}^0, \dots, x_N^0) \in E$, для якої детермінант матриці (матриці Якобі)

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_0) & \dots & \frac{\partial f_1}{\partial x_m}(x_0) \\ \dots & \dots & \dots \\ \frac{\partial f_m}{\partial x_1}(x_0) & \dots & \frac{\partial f_m}{\partial x_m}(x_0) \end{pmatrix}$$

відмінний від 0, то існують такі функції $\varphi_1, \dots \varphi_m$, для яких в деякому околі V_0точки x_0 мають місце рівності 11).

Позначимо ще $\dot{x_0}$ точку $(x_{m+1}^0, \dots, x_N^0) \in R^{N-m}$(ця точка представлена останніми $N - m$ координатами точки x_0). Зауважимо, що згідно з рівностями (11) координатне представлення точки x_0 можна подати у вигляді

$$x_0 = (\varphi_1(x_{m+1}^0, \dots, x_N^0), \dots, \varphi_m(x_{m+1}^0, \dots, x_N^0), x_{m+1}^0, \dots, x_N^0) \qquad (12)$$

Введемо функцію $g = g(x_{m+1}, \dots, x_N)$ за допомогою рівності $g(x_{m+1}, \dots, x_N) = f(\varphi_1(x_{m+1}, \dots, x_N), \dots, \varphi_m(x_{m+1}, \dots, x_N), x_{m+1}, \dots, x_N)$. Ця функція визначена у деякому $(N - m)$-вимірному околі точки $\dot{x_0}$. Має місце наступне твердження.

*Точка x_0 є точкою **умовного** екстремуму для функції f відносно рівнянь зв'язку* (2) *тоді і тільки тоді, коли точка $\dot{x_0}$ є точкою **звичайного** екстремуму для функції g.*

Зауважимо, що для знаходження координат точки x_0, коли є відомими координати точки $\dot{x_0}$, можна користатися рівністю (12). Наприклад, для *N* =2,

$$f(x_1, x_2) = x_1^2 + x_2^2, \quad (13)$$

$$x_1 + x_2 - 1 = 0. \quad (14)$$

Дослідити на екстремум функцію f за умови виконання рівняння зв'язку (14).

Маємо частковий випадок вищенаведеної схеми при $m = 1, f_1(x) = x_1 + x_2 - 1$. Використовуючи рівняння (14), виразимо змінну x_1 через x_2, після чого введемо функцію g:

$$x_1 = 1 - x_2, \quad (15)$$

$$g(x_2) = (1 - x_2)^2 + x_2^2 = 2x_2^2 - 2x_2 + 1.$$

Досліджуємо на екстремум функцію g:

$$g'(x_2) = 4x_2 - 2 = 0,$$

$$x_2 = \frac{1}{2}.$$

При цьому $g''\left(\frac{1}{2}\right) = 4 > 0$. Так що в точці $x_2 = \frac{1}{2}$ функція g має мінімум (строгий). З рівності (15) маємо $x_1 = 1 - \frac{1}{2} = \frac{1}{2}$. Тому згідно з рівністю 12) маємо точку мінімуму функції f : $x_0 = (\frac{1}{2}, \frac{1}{2})$. При цьому мінімальне значення f_{min} дорівнює $f\left(\frac{1}{2}, \frac{1}{2}\right) = \frac{1}{2}$..

Знайдена точка *умовного* екстремуму $x_0 = (\frac{1}{2}, \frac{1}{2})$ не є точкою *безумовного* екстремуму функції *f*. Дійсно, для визначення стаціонарних точок функції f маємо систему рівнянь

$$\begin{cases} \frac{\partial f}{\partial x_1} = 2x_1 = 0 \\ \frac{\partial f}{\partial x_2} = 2x_2 = 0 \end{cases}$$

з єдиним розв'язком (0,0) $\left(\neq (\frac{1}{2}, \frac{1}{2})\right)$.

Розглянемо випадок, коли *N* = 3. Для меншої громіздкості позначень замість x_1, x_2, x_3 будемо писати x, y, z відповідно.

Знайти найбільше значення добутку

$$f = xyz \quad (x \geq 0, y \geq 0, z \geq 0) \tag{16}$$

за умови, що

$$x + y + z = 3c, \tag{17}$$

де $c = const$, а позначення $3c$ (а не c) застосовується тільки для більшої зручності при написанні подальших співвідношень.

Застосуємо наведену вище загальну схему. Тут

$$m = 1, f_1 = f_1(x, y, z) = x + y + z - 3c.$$

Маємо з (17) та (16):

$$z = 3c - (x + y), \tag{18}$$

$$g = g(x, y) = xyz\,|_{z=3c-(x+y)} = xy(3c - (x + y)) = 3cxy - x^2y - xy^2 .$$

Оскільки всі змінні за умовою невід'ємні, то з рівності (17) маємо наступні обмеження на x, y при дослідженні на екстремум функції g:

$$0 \leq x \leq 3c, 0 \leq y \leq 3c, 0 \leq x + y \leq 3c.$$

Наступний рисунок відповідає зазначеним обмеженням.

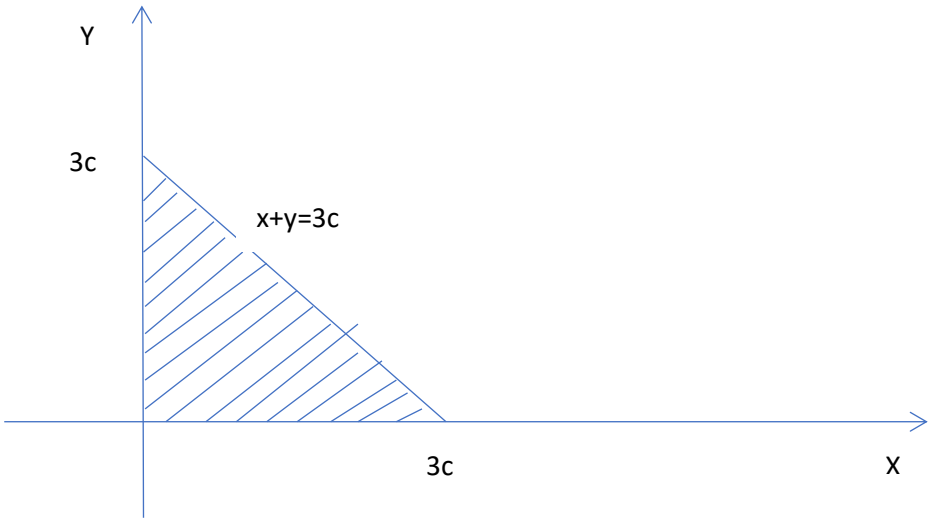


Рис. 2.1

Розшукуємо стаціонарні точки функції g.

$$\frac{\partial g}{\partial x} = \frac{\partial}{\partial x}(3cxy - x^2y - xy^2) = 3cy - 2xy - y^2 = y(3c - 2x - y) = 0;$$

$$\frac{\partial g}{\partial y} = \frac{\partial}{\partial y}(3cxy - x^2y - xy^2) = 3cx - x^2 - 2xy = x(3c - x - 2y) = 0.$$

Очевидно, будь-яка з стаціонарних точок, що має хоч одну нульову координату, не може бути розв'язком нашої задачі, оскільки наявність нульової координати веде до нульового значення функції f, а таке значення точно не є максимальним за нашими умовами. Отже, звернімося до інших розв'язків останньої системи рівнянь. Маємо

$$3c - 2x - y = 0$$

$$3c - x - 2y = 0$$

Розв'язуючи цю систему, одержуємо відповідь $x = y = c, g(c, c) = c^3$, При цьому на границі області обмежень функції g остання дорівнює 0 ($< c^3$), так що зазначена точка (c, c) є точкою максимуму функції g. Згідно з (18) маємо $z = 3c - 2c = c$. Таким чином, максимум функції f досягається в точці (c, c, c) і є рівним $c \cdot c \cdot c = c^3$.

2.4. Прямі методи розв'язання задач оптимізації

Прямі методи базуються на послідовному переборі або оцінці точок в області визначення, щоб знайти оптимальне рішення. Основні риси:

1. Не використовують похідні функції. Можуть працювати з функціями, які не є гладкими або мають розриви.
2. Застосовуються до широкого класу задач. Використовуються, коли функція цільова не має аналітичної форми або обчислення похідних складне.
3. Типові методи:

- Метод повного перебору.
- Метод випадкового пошуку.
- Метод симплексів (наприклад, метод Нелдера-Міда).
- Метод гілок і меж.

4. Переваги:
   - Можуть знаходити глобальні оптимуми.
   - Робочі для задач, де інші методи непридатні (наприклад, при розривних функціях).
5. Недоліки:
   - Повільна збіжність.
   - Можуть вимагати значних обчислювальних ресурсів.

Більшість застосовуваних на практиці прямих методів використовують алгоритми спуску для вирішення задач математичного програмування і вони є ітеративними.

Загальне правило побудови алгоритми спуска для вирішення задач математичного програмування послідовності $\bar{x}^0, \bar{x}^1, \cdots, \bar{x}^k, \bar{x}^{k+1}, \cdots$ може бути представленим у вигляді:

$$\bar{x}^{k+1} = \bar{x}^k + \alpha_k \bar{p}^k, \quad k = 0,1, \cdots$$

Тут $\bar{x}^0$ - початкова точка алгоритму, $\bar{p}^k$ - прийнятий на ітерації напрямок переходу з точки $\bar{x}^k$ в точку $\bar{x}^{k+1}$. $\bar{p}^k$ називається напрямком спуска, α^k - числовий множник, що визначає швидкість або величину кроку.

2.5. Алгоритмічні складові методів градієнтного спуску

Загальна схема методів градієнтного спуску включає наступні кроки Ініціалізація: обирається початкове значення параметрів моделі.

Обчислення градієнта: обчислюється градієнт функції в поточній точці. Градієнт показує напрямок найшвидшого зростання функції.

Оновлення параметрів: параметри моделі оновлюються в напрямку, протилежному градієнту. Це здійснюється за допомогою формули:

нові параметри = старі параметри − крок оновлення $*$ градієнт.

Перевірка умови зупинки: перевіряється умова зупинки, яка може бути досягненням заданої точності або досягненням певного числа ітерацій.

Якщо умова зупинки не виконана, повернення до кроку 2. Інакше, оптимізація завершується, і отримані параметри є розв'язком задачі оптимізації.

У методах градієнтного спуску важливими параметрами є крок оновлення (learning rate) і початкове значення параметрів. Вибір правильного кроку оновлення може бути складною задачею, оскільки значення, котре є занадто великим, може призводити до нестійкої збіжності або перепаду, а занадто малий крок може затримувати процес оптимізації. Отже, варто експериментувати з різними значеннями кроку, щоб знайти оптимальний крок оновлення.

Вибір початкової точки $\bar{x}^0$ виробляється, виходячи з фізичного змісту вирішуваної задачі та наявності попередньої інформації про положення точок екстремуму.

Вибір параметрів напрямків і величини кроку визначається стратегією і обраним методом рішення. Критерій перевірки завершення обчислень ітераційного процесу дає інформацію про те, що або рішення завдання має тривати, або знайдена точка, яка претендує на роль екстремуму, і процедуру пошуку слід завершити.

Один з основних критеріїв перевірки завершення обчислень градієнтного спуску - це досягнення певної точки зупинки або заданої точності. Існує кілька широко використовуваних критеріїв, включаючи наступні:

Зупинка за нормою градієнта: обчислюється норма вектора градієнта і порівнюється з певним заданим порогом. Якщо норма градієнта менша за цей

поріг, то вважається, що оптимізація завершена, оскільки знаходиться близько до локального екстремуму.

Зупинка за зміною функціоналу: обчислюється різниця між значеннями функціоналу (функції, яку мінімізуємо) на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена.

Зупинка за зміною параметрів: обчислюється різниця між значеннями параметрів на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена.

Зупинка за кількістю ітерацій: задається максимальна кількість ітерацій, яку необхідно виконати. Якщо ця максимальна кількість досягнута, то вважається, що оптимізація завершена.

Зупинка за зміною величини кроку: обчислюється різниця між величиною кроку (напрямок спуску) на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена.

Висновки до розділу 2

В даному розділі розглянуто деякі поширені методи розв’язання математичних задач теорії оптимізації. Розглянуто деякі непрямі методи – Лагранжа та Якобі. Дано означення та коротко охарактеризовано основні риси прямих методів. Останні проілюстровані загальним описанням методів спуску і більш детально – методом градієнтного спуску. В додатку роботи представлено програмний код алгоритму зазначеного методу.

Розділ 3
АНАЛІЗ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СПОСОБІВ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ОПТИМІЗАЦІЇ

Оптимізація є однією з ключових задач сучасної науки та інженерії, яка знаходить застосування у різних сферах — від машинного навчання до промислового проєктування [22-24]. Ефективність методів оптимізації оцінюється за низкою критеріїв, зокрема швидкістю збіжності, стійкістю до локальних мінімумів, обчислювальною складністю та адаптивністю до різних класів задач. Теоретичний аналіз цих методів дозволяє систематизувати знання про їхні властивості, виявити переваги та обмеження, а також обрати відповідний метод для конкретного застосування.

3.1. Класи методів оптимізації

Методи оптимізації можна розділити на два основні класи: детерміновані та стохастичні.

1. Детерміновані методи (наприклад, градієнтний спуск, метод Ньютона) базуються на строгих математичних моделях, передбачаючи наявність повної інформації про функцію. Їх ефективність часто визначається такими характеристиками, як швидкість збіжності (лінійна, квадратична) та чисельна стабільність. Теоретичний аналіз цих методів включає вивчення випуклості функції, ліпшицевості градієнта та поведінки на великих наборах даних.

2. Стохастичні методи (наприклад, стохастичний градієнтний спуск, еволюційні алгоритми, методи рою частинок) враховують випадковість у пошуку рішення. Вони особливо ефективні для задач, де обчислення градієнта або інших властивостей функції є складним або неможливим. Стохастичні методи характеризуються високою здатністю уникати локальних мінімумів, але часто страждають від

повільнішої збіжності. Теоретичний аналіз цих методів зосереджений на дослідженні ролі стохастичних факторів, таких як розмір вибірки чи швидкість навчання, а також умов збіжності.

Основні критерії оцінки ефективності

1. **Швидкість збіжності**: Один із ключових параметрів, що визначає, наскільки швидко метод досягає оптимального рішення. Детерміновані методи, такі як метод Ньютона, зазвичай демонструють високу швидкість збіжності (квадратична), тоді як стохастичні методи часто мають лише сублінійну збіжність [24-25].
2. **Обчислювальна складність**: Цей критерій враховує кількість обчислень, необхідних для кожної ітерації. Деякі методи, наприклад, класичний градієнтний спуск, мають низьку обчислювальну вартість, тоді як методи другого порядку, такі як метод Ньютона, потребують більших ресурсів через обчислення та інверсію матриці Гессе [26] .
3. **Стійкість відносно локальних мінімумів**. Для задач невипуклої оптимізації важливо, щоб метод не застрягав у локальних точках мінімуму. Стохастичні методи та еволюційні алгоритми зазвичай краще справляються з такими викликами завдяки випадковому характеру пошуку.
4. **Універсальність та адаптивність**. Універсальні методи можуть застосовуватися до широкого класу задач без значного доопрацювання. Наприклад, еволюційні алгоритми здатні працювати навіть у разі відсутності аналітичного виразу функції.

Теоретичний аналіз є важливим інструментом для розуміння меж застосовності кожного методу оптимізації. Він дозволяє виявити оптимальні умови використання, наприклад, для яких типів задач конкретний метод є найбільш ефективним. Також аналіз допомагає у розробці нових, більш

ефективних алгоритмів шляхом інтеграції переваг існуючих підходів. Таким чином, дослідження ефективності методів оптимізації є багатоаспектною задачею, яка вимагає як теоретичного, так і практичного підходу. Результати такого аналізу сприяють розвитку нових технологій і вдосконаленню процесів у різних сферах людської діяльності.

3.2. Емпіричне тестування ефективності методів оптимізації

Емпіричне тестування є невід'ємною частиною оцінки ефективності методів оптимізації [28-30]. Воно дозволяє перевірити теоретичні припущення та зрозуміти, як алгоритми поводяться на практиці в умовах реальних або симульованих задач. У ході емпіричного аналізу використовуються експерименти на різних наборах тестових функцій, порівнюються швидкість збіжності, точність результату, а також стійкість алгоритмів до варіацій параметрів та випадкових перешкод.

Основні етапи емпіричного тестування

1. Вибір тестових задач: Для оцінки ефективності алгоритмів зазвичай використовуються стандартні тестові функції, що мають добре відомі властивості. Набір тестових функцій може включати:
    - Унімодальні функції (наприклад, квадратична функція): перевіряється здатність алгоритму швидко збігатися до єдиного мінімуму.
    - Мультимодальні функції (наприклад, функція Розенброка або Растрігіна): оцінюється здатність алгоритму уникати локальних мінімумів.
    - Функції з обмеженнями: перевіряється ефективність роботи алгоритму в умовах, коли рішення обмежені певними областями.
2. Вибір метрик оцінки: Основні метрики включають:

- Кількість ітерацій до збіжності: використовується для оцінки швидкості роботи.
- Помилка відносно глобального мінімуму: дозволяє оцінити точність результату.
- Час виконання алгоритму: дає уявлення про обчислювальну ефективність.
- Робастність: аналізується варіація результатів при зміні початкових умов або параметрів.

3. Налаштування параметрів алгоритму: Більшість методів оптимізації мають параметри, які впливають на їхню продуктивність, наприклад, швидкість навчання в градієнтному спуску чи кількість агентів у методах рою частинок. Емпіричне тестування включає як пошук оптимальних параметрів, так і оцінку поведінки алгоритму за їхніх варіацій.
4. Порівняння алгоритмів: У тестуванні порівнюються кілька алгоритмів на одних і тих самих задачах. Наприклад, можна зіставити градієнтний спуск із методами другого порядку (такими як метод Ньютона) або зі стохастичними алгоритмами, наприклад, стохастичним градієнтним спуском чи генетичними алгоритмами.

Результати емпіричного тестування

1. **Детерміновані методи**: Емпіричне тестування показує, що методи градієнтного спуску ефективні на гладких, випуклих функціях із добре визначеним глобальним мінімумом [29-30] . Проте вони демонструють повільну збіжність на функціях із вузькими долинами або плато. Метод Ньютона зазвичай швидший завдяки квадратичній швидкості збіжності, але має високу обчислювальну вартість і погано працює на великих задачах або функціях без гладкого гессіана.

2. **Стохастичні методи**: Стохастичний градієнтний спуск (SGD) показує високу ефективність на великих наборах даних завдяки обробці випадкових підвибірок. Однак його результати часто мають більшу варіативність, ніж у детермінованих методів. Еволюційні алгоритми, такі як генетичні алгоритми, добре справляються з мультимодальними функціями, але потребують великої кількості ітерацій для досягнення прийнятного результату.
3. **Гібридні методи**: У ході емпіричних тестувань було виявлено, що комбіновані підходи (наприклад, поєднання детермінованих і стохастичних методів) забезпечують краще балансування між точністю та швидкістю. Наприклад, алгоритм Adam, який поєднує стохастичний градієнтний спуск із адаптивним регулюванням швидкості навчання, показує стабільну збіжність навіть для задач з нерівномірними ландшафтами функцій.

3.3. Статистичний аналіз ефективності методів оптимізації

Статистичний аналіз є важливим інструментом для об’єктивного оцінювання ефективності методів оптимізації. На відміну від теоретичного та емпіричного підходів, він дозволяє враховувати варіативність результатів, випадковість у стохастичних алгоритмах, а також вплив різних параметрів на продуктивність. Завдяки статистичним методам можна визначити, чи є відмінності між алгоритмами значущими, а також оцінити їхню стійкість до змін умов.

Основні цілі статистичного аналізу

1. Оцінка стабільності: Аналіз дозволяє визначити, наскільки результати методу варіюються при різних початкових умовах, параметрах або реалізаціях стохастичних процесів.
2. Порівняння алгоритмів: Використовуючи статистичні методи, можна перевірити, чи є один алгоритм значно кращим за інший за обраними метриками.

3. Аналіз впливу параметрів: Визначення того, як зміна параметрів, таких як швидкість навчання або розмір вибірки, впливає на продуктивність методу.
4. Інтерпретація розподілів результатів: Дослідження розподілу показників ефективності (наприклад, часу збіжності чи точності) для кращого розуміння поведінки методів.

Метрики для статистичного аналізу

Для оцінки ефективності методів оптимізації зазвичай використовуються наступні метрики:

- Середнє значення метрики: Наприклад, середня кількість ітерацій до збіжності або середнє значення помилки.
- Дисперсія та стандартне відхилення: Відображають розкидання результатів і допомагають оцінити стабільність методу.
- Медіана: Використовується для оцінки центральної тенденції в умовах, коли розподіл має значні відхилення або аномальні значення.
- Процентилі: Відображають діапазон результатів, наприклад, 90-й процентиль показує, наскільки ефективно метод працює у найкращих випадках.

Методи статистичного аналізу

1. Аналіз дисперсії (ANOVA): Використовується для порівняння кількох алгоритмів на основі середніх значень обраної метрики. ANOVA дозволяє визначити, чи є відмінності між алгоритмами статистично значущими. Якщо значущість виявлено, застосовуються пост-хок тести (наприклад, тест Тьюкі) для детального порівняння пар алгоритмів.

2. Тест Стьюдента (t-тест): Використовується для порівняння двох алгоритмів. Наприклад, можна оцінити, чи метод А збігається швидше за метод В на статистично значущому рівні.
3. Неметричні тести: Якщо результати не відповідають нормальному розподілу, застосовуються неметричні методи, такі як тест Манна-Уітні або тест Крускала-Уолліса.
4. Регресійний аналіз: Дозволяє досліджувати вплив параметрів алгоритму на його ефективність. Наприклад, регресія може виявити, як швидкість навчання впливає на час збіжності або точність методу.
5. Бутстреп-аналіз: Використовується для оцінки стабільності результатів за рахунок повторного вибіркового відбору з отриманих даних. Цей метод особливо корисний для алгоритмів із невеликими вибірками.

Практичний приклад аналізу

Розглянемо порівняння стохастичного градієнтного спуску (SGD) і методу Adam на мультимодальній функції Растрігіна. Статистичний аналіз проводиться за такими етапами:

1. Збираються результати для кожного методу: кількість ітерацій, точність, час виконання.
2. Розраховується середнє значення та стандартне відхилення для кожної метрики.
3. Виконується t-тест для перевірки гіпотези про рівність середніх значень. Якщо гіпотеза відхиляється, це свідчить про статистично значущу відмінність.
4. Проводиться аналіз дисперсії, щоб виявити, чи є результати методу стабільними.

У нашому прикладі результати можуть показати, що Adam демонструє меншу кількість ітерацій до збіжності ($p < 0.05$), але має більший розкид часу виконання, ніж SGD. Це свідчить про те, що Adam краще підходить для задач із вимогами до точності, тоді як SGD може бути перевагою у високочастотних ітераційних процесах.

Висновки щодо стохастичного аналізу ефективності

Статистичний аналіз ефективності методів оптимізації є ключовим етапом для обґрунтованого вибору алгоритму. Він дозволяє виявити закономірності в поведінці методів, підтвердити або спростувати гіпотези, а також розробити рекомендації щодо вибору алгоритму залежно від конкретних умов. Цей підхід, у поєднанні з теоретичним і емпіричним аналізом, забезпечує комплексну оцінку методів оптимізації.

3.4. Визначення стійкості методів оптимізації

Стійкість методів оптимізації є важливою характеристикою, яка визначає їх здатність стабільно забезпечувати прийнятні результати в умовах змінних параметрів, початкових умов, властивостей функції, а також за наявності шумів або похибок. Вивчення стійкості дозволяє оцінити, наскільки алгоритм є надійним у реальних сценаріях, де ідеальні умови рідко виконуються.

Аспекти поняття стійкості в контексті оптимізації

Стійкість методів оптимізації характеризується декількома аспектами:

1. Чутливість до початкових умов: Відображає, чи змінюється результат алгоритму залежно від вибору початкової точки. Наприклад, градієнтний спуск часто застряє в локальних мінімумах при невдалому початковому наближенні.

2. Робастність до шумів: Здатність алгоритму знаходити оптимальні рішення в умовах зашумлених або неточних градієнтів чи функцій цілі.
3. Варіативність результатів: Оцінюється наскільки сильно змінюються результати при багаторазовому запуску алгоритму за тих самих умов (важливо для стохастичних методів).
4. Стійкість до змін параметрів: Наприклад, швидкість навчання в градієнтному спуску або кількість агентів у рої частинок може суттєво впливати на продуктивність.
5. Методи дослідження стійкості

1. Аналіз чутливості: Оцінюється вплив зміни одного або декількох параметрів на ефективність алгоритму. Наприклад, у методах градієнтного спуску можна варіювати розмір кроку, щоб знайти діапазон значень, за якого алгоритм стабільно збігається до оптимуму.
2. Моделювання зашумлених умов: У задачах оптимізації нерідко виникає шум у градієнті (наприклад, через похибки вимірювань). Для тестування стійкості додають випадкові перешкоди до функції цілі або її похідних і перевіряють, чи здатен алгоритм досягти прийнятного результату.
3. Статистичний аналіз варіативності результатів: Виконується багаторазовий запуск алгоритму з різними початковими умовами. Оцінюються середнє значення, дисперсія та стандартне відхилення метрик, таких як час збіжності чи точність. Стохастичні методи, наприклад, стохастичний градієнтний спуск (SGD), часто демонструють більшу варіативність, ніж детерміновані.
4. Порівняльний аналіз різних сценаріїв: Алгоритми тестуються на функціях із різними властивостями: гладкість, випуклість, наявність локальних мінімумів, обмеження в області пошуку. Наприклад, методи еволюційного типу часто краще працюють на мультимодальних

функціях, тоді як детерміновані методи ефективніші на гладких випуклих функціях.

Приклади вивчення стійкості

1. Градієнтний спуск: У градієнтному спуску стійкість сильно залежить від вибору швидкості навчання. Занадто великий крок призводить до розходження, тоді як занадто малий робить алгоритм повільним. Додавання випадкових шумів до градієнта може викликати «коливання» поблизу мінімуму, але введення таких модифікацій, як моментум або алгоритм Adam, підвищує стійкість.
2. Методи рою частинок: Ці алгоритми часто демонструють високу стійкість до варіацій початкових умов завдяки колективному пошуку, але їх параметри, такі як інерційний коефіцієнт або коефіцієнт соціального впливу, суттєво впливають на результати. Наприклад, надто велика інерція може зробити алгоритм нестійким.
3. Еволюційні алгоритми: Ці методи демонструють хорошу стійкість до локальних мінімумів, однак їхня продуктивність залежить від налаштувань, таких як розмір популяції або ймовірність мутацій. Надмірна мутація може спричинити втрату збіжності, тоді як її недостатність робить алгоритм чутливим до локальних мінімумів.

Метрики для оцінки стійкості

1. Коефіцієнт варіації: Відображає відношення стандартного відхилення результатів до середнього значення, дозволяючи порівнювати варіативність між алгоритмами.
2. Частота збіжності до глобального мінімуму: Для мультимодальних функцій цей показник визначає, як часто алгоритм знаходить оптимальне рішення.

3. Середня кількість ітерацій до збіжності: Доповнює аналіз стійкості, особливо якщо вона сильно варіюється в залежності від параметрів.

3.5. Візуалізація процесу дослідження ефективності методів оптимізації

Візуалізація є потужним інструментом для розуміння роботи методів оптимізації, демонстрації їх ефективності та аналізу поведінки в процесі збіжності. Графічне представлення допомагає не лише проілюструвати, як алгоритм шукає оптимум, але й виявити проблемні області, наприклад, застрягання в локальних мінімумах чи повільну збіжність. Крім того, візуалізація дозволяє порівнювати методи між собою, оцінюючи їхню продуктивність у наочній формі.

Цілі візуалізації

1. Ілюстрація траєкторії пошуку: Візуалізація траєкторії пошуку допомагає зрозуміти, як алгоритм переміщується простором рішень і як швидко він наближається до оптимуму.
2. Дослідження ландшафту функції цілі: Графічне представлення функції цілі дозволяє побачити її особливості: мультимодальність, плато, вузькі долини тощо, що допомагає інтерпретувати результати роботи алгоритмів.
3. Порівняння продуктивності алгоритмів: Візуалізація збіжності, наприклад, графіки залежності значення функції цілі від кількості ітерацій, дозволяє оцінити ефективність методів.
4. Оцінка стабільності: Використання графічних методів для аналізу варіацій результатів допомагає зрозуміти, наскільки алгоритм є стійким до змін параметрів або початкових умов.

Методи візуалізації процесу оптимізації, що описуються в [29] мають наступні аспекти:

1. Траєкторії пошуку:
    - Для функцій у двовимірному просторі траєкторія пошуку представляється у вигляді кривої на поверхні функції цілі. Такі графіки показують шлях, яким алгоритм рухається до мінімуму.
    - Для більш складних задач із багатьма змінними використовуються проекції або спрощення, наприклад, візуалізація лише кількох ключових параметрів.
2. Ландшафт функції цілі:
    - Контурні графіки (ізолінії): Відображають рівні значень функції цілі. Алгоритм накладається на контурний графік, що показує, як метод знаходить шляхи до глобального мінімуму.
    - Тривимірні графіки: Застосовуються для більш детального аналізу функції та траєкторії алгоритму.
3. Графіки збіжності:
    - Побудова графіків залежності значення функції цілі від кількості ітерацій. Ці графіки дозволяють порівняти швидкість збіжності різних методів, оцінити стабільність і аналізувати поведінку на різних етапах пошуку.
    - Наприклад, градієнтний спуск може демонструвати експоненційну збіжність на випуклих функціях, тоді як методи рою частинок можуть мати складну нелінійну динаміку.
4. Аналіз розподілів результатів:
    - Для стохастичних алгоритмів важливо показати, як розподіляються значення функції цілі після багаторазових запусків. Для цього використовуються гістограми або діаграми розсіювання.

5. Візуалізація залежності параметрів:
    - Для дослідження впливу параметрів (наприклад, швидкості навчання) будуються тривимірні графіки або теплові карти, які показують, як змінюється продуктивність алгоритму в залежності від параметрів.

Практичні приклади

1. Градієнтний спуск:
    - На контурному графіку функції Розенброка траєкторія градієнтного спуску показує, як алгоритм рухається вузькою долиною до глобального мінімуму. Візуалізація дозволяє помітити повільну збіжність у напрямках із різними кривизнами.
2. Метод рою частинок (PSO):
    - Рух агентів можна візуалізувати у вигляді анімації, яка показує, як частинки досліджують простір і збираються навколо оптимального рішення. Такі графіки демонструють як глобальне, так і локальне співробітництво між частинками.
3. Метод еволюційних стратегій:
    - Розподіл особин у популяції на кожній ітерації можна візуалізувати для аналізу того, як алгоритм адаптується до мультимодальних функцій і долає локальні мінімуми.

Висновки щодо методів візуалізації

Візуалізація процесу розв’язання оптимізаційних задач дозволяє глибше зрозуміти особливості роботи методів, виявити їхні слабкі сторони та підвищити ефективність шляхом адаптації параметрів. Графічне представлення результатів робить аналіз інтуїтивно зрозумілим, а також

слугує важливим інструментом для пояснення результатів у наукових дослідженнях і практичних застосуваннях.

3.6. Оцінка робастності методів оптимізації

Робастність методів оптимізації означає їхню здатність досягати високої ефективності за умов змін зовнішніх факторів, параметрів алгоритму або властивостей функції цілі. Це ключовий аспект для алгоритмів, які застосовуються у практичних задачах, де часто існують похибки, шум, зміна середовища чи інші невизначеності. Робастні методи забезпечують стабільні результати, навіть якщо умови їхнього використання виходять за межі попередніх очікувань.

Ключові аспекти робастності

1. Стійкість до шумів: Методи оптимізації часто використовуються для задач, у яких оцінка функції цілі або її градієнта піддається шумам. Наприклад, у машинному навчанні похибки у вибірці можуть впливати на точність розрахунків. Робастний алгоритм повинен враховувати ці особливості та демонструвати прийнятну продуктивність навіть за умов значних похибок.
2. Адаптація до властивостей функції: У багатьох випадках властивості функції цілі (гладкість, випуклість, мультимодальність) можуть значно варіюватися. Робастні методи повинні показувати ефективність на різних класах функцій без значних модифікацій.
3. Чутливість до параметрів: Деякі алгоритми демонструють високу залежність від початкових параметрів (наприклад, швидкість навчання у градієнтному спуску або кількість частинок у PSO). Робастність передбачає, що ефективність алгоритму незначно змінюється у широкому діапазоні значень параметрів.

4. Масштабованість: Робастні алгоритми мають ефективно працювати при збільшенні розмірності задачі або кількості обмежень.

Методи оцінки робастності

Аналіз впливу шуму:

1. До функції цілі додається випадковий шум із заданими характеристиками (наприклад, нормальний розподіл із середнім значенням 0 і варіативністю 1). Оцінюється, наскільки значення функції, час збіжності або траєкторія пошуку змінюються за цих умов.
2. Наприклад, для стохастичного градієнтного спуску (SGD) тестують, чи може алгоритм зберігати стабільність, якщо оцінка градієнта є приблизною.

Випробування на різних функціях:

1.Робастність перевіряється на наборах тестових функцій із різними властивостями: випуклі (наприклад, квадратична функція), невипуклі (функція Розенброка), мультимодальні (функція Растрігіна).

2.Для кожної функції оцінюється кількість ітерацій до збіжності, точність, швидкість виконання тощо.

5. **Зміна параметрів алгоритму**:

1. Проводиться серія експериментів із варіюванням ключових параметрів. Наприклад, змінюється швидкість навчання у градієнтному спуску, інерційний коефіцієнт у PSO чи ймовірність мутації в еволюційних алгоритмах.
2. Графіки залежності ефективності від параметрів допомагають визначити, наскільки метод стійкий до змін.

6. **Стохастичні тести**:

Метод запускається багаторазово за різних початкових умов. Аналізуються середні значення, дисперсія та коефіцієнт варіації метрик ефективності.

7. **Робастність до високої розмірності**:

   1. Алгоритми тестуються на задачах із поступово зростаючою кількістю змінних. Аналізується, чи змінюється продуктивність пропорційно до збільшення розмірності, і чи зберігає метод здатність знаходити рішення.

Моментум у чисельній оптимізації

Моментумом називається концепція, яка використовується в чисельних методах і алгоритмах оптимізації для прискорення збіжності. У контексті перевірки стійкості алгоритмів, моментум може мати кілька значень залежно від конкретного підходу:

У градієнтних методах, таких як градієнтний спуск, моментум дозволяє згладжувати коливання під час руху до мінімуму функції. Це досягається за рахунок врахування попереднього оновлення параметрів.

Формула оновлення з моментумом виглядає так:

$$v_t = \beta v_{t-1} + (1-\beta)\,\nabla f(\vartheta_{t-1}), \vartheta_t = \vartheta_{t-1} - \eta\, v_t,$$

де v_t — швидкість оновлення, η – швидкість навчання, β — коефіцієнт моментуму (зазвичай β ∈[0.8,0.99]), $\nabla f(\vartheta_{t-1})$— градієнт на поточному кроці.

1. Градієнтний спуск із моментумом

На функції Розенброка досліджується вплив параметра моментуму. Результати показують, що за оптимальних значень моментум дозволяє

прискорити збіжність, але занадто велике значення може призводити до осциляцій.

2. **Рой частинок (PSO)**

Для тестування робастності додають випадкові перешкоди до значень функції цілі. Результати свідчать, що PSO з адаптивними параметрами демонструє високу стійкість навіть за наявності шуму, тоді як статичні параметри значно знижують ефективність.

3. **Еволюційні алгоритми**:

У задачах з обмеженнями змінюється розмір популяції. Експерименти показують, що алгоритм добре адаптується до змін, якщо мутації мають динамічну стратегію, але за надто малої популяції можливе застрягання в локальних мінімумах.

Метрики для оцінки робастності алгоритму оптимізації

1. **Середнє значення метрик ефективності**: Наприклад, середній час збіжності, точність або значення функції цілі в кінцевій точці.
2. **Стандартне відхилення**: Показує розкид результатів у різних сценаріях. Чим менше відхилення, тим метод робастніший.
3. **Рівень шумостійкості**: Визначається як максимальна інтенсивність шуму, за якої алгоритм ще досягає прийнятного результату.
4. **Частота знаходження глобального мінімуму**: Для мультимодальних функцій визначає, наскільки часто метод уникає локальних мінімумів.

Висновки щодо методів оцінки робастності

Оцінка робастності ефективності методів оптимізації дозволяє зробити висновки про їхню придатність для використання в умовах невизначеності.

Робастні алгоритми є більш універсальними, адже забезпечують стабільні результати в широкому діапазоні умов. Такий аналіз є важливим етапом у розробці нових методів оптимізації, а також у виборі алгоритму для конкретної задачі.

3.7. Аналіз витрат ефективності методів оптимізації як характеристика ефективності методів оптимізації

Аналіз витрат у контексті методів оптимізації передбачає оцінку ресурсів, необхідних для досягнення оптимального або прийнятного рішення. До таких ресурсів належать час, обчислювальна потужність, пам'ять, кількість ітерацій, а також обсяги даних. Вибір методу оптимізації часто визначається не тільки його теоретичною ефективністю, а й витратами, які він вимагає в практичному застосуванні.

Ключові аспекти витрат в процесах оптимізації

1. **Обчислювальна складність**:

Оцінюється, як змінюється кількість операцій, необхідних для виконання алгоритму, зі збільшенням розмірності задачі чи інших характеристик (наприклад, кількість змінних, обмежень). Наприклад, градієнтний спуск має відносно низьку обчислювальну складність, але метод Ньютона потребує обчислення та інверсії матриці Гессе, що значно збільшує витрати.

2. **Кількість ітерацій до збіжності**:

Деякі алгоритми досягають оптимуму за меншу кількість ітерацій, однак кожна ітерація може вимагати більше ресурсів. Наприклад, методи другого порядку зазвичай швидше збігаються, але кожен їхній крок є більш ресурсозатратним.

3. **Час виконання**:

Загальний час роботи алгоритму є критично важливим для задач із жорсткими часовими обмеженнями. Реальні витрати залежать не лише від теоретичної складності, але й від способу реалізації алгоритму, доступності апаратних ресурсів і паралельності обчислень.

4. **Витрати пам’яті**:

Методи, які зберігають великі обсяги проміжних результатів (наприклад, еволюційні алгоритми чи методи рою частинок), можуть бути непридатними для задач із великими розмірностями через обмеження пам'яті.

5. **Витрати на оцінку функції цілі**:

У деяких задачах, таких як оптимізація в реальному часі чи задачі машинного навчання, обчислення значення функції цілі або її градієнта є дорого вартісними. Методи, які зменшують кількість таких обчислень, часто мають перевагу.

Методи оцінки витрат

1. **Теоретичний аналіз складності**:

Для кожного методу визначають часову та просторову складність у термінах *O*-нотації. Наприклад:

1) Градієнтний спуск: $O(n)$ для обчислення градієнта при n змінних.
2) Метод Ньютона: $O(n^3)$ для інверсії матриці Гессе.
3) Генетичні алгоритми: $O(k \cdot n)$, де k — розмір популяції, n — кількість змінних.

2. **Емпіричні вимірювання**:

Час виконання алгоритму вимірюється на тестових задачах різної складності. Аналізуються середній час, максимальні та мінімальні витрати.

Наприклад, для задачі з 10 змінними градієнтний спуск може завершити оптимізацію за кілька секунд, тоді як метод рою частинок (PSO) може потребувати хвилини.

3. **Оцінка ефективності ітерацій**:

Кількість ітерацій до збіжності вимірюється для різних функцій і параметрів. Алгоритми з великою кількістю ітерацій часто не є економічно вигідними, навіть якщо окремі ітерації є швидкими.

4. **Порівняння ресурсної ефективності**:

Визначається співвідношення між точністю результату та витратами. Наприклад, метод, який вимагає більше часу, але забезпечує значно кращий результат, може бути більш ефективним у задачах із високими вимогами до точності.

Приклади оцінки витрат

1. **Градієнтний спуск**:

Простий у реалізації та має низькі витрати на кожну ітерацію, але його продуктивність падає на функціях із складним ландшафтом (наприклад, вузькими долинами). Якщо кроки підібрані неправильно, алгоритм може застрягнути, що збільшує загальні витрати.

2. **Методи другого порядку**:

Хоча ці методи швидко збігаються (наприклад, метод Ньютона), бчислення гессіана робить їх непридатними для задач із великою кількістю змінних. Витрати часу та пам'яті зростають експоненційно.

3. **Еволюційні алгоритми**:

Генетичні алгоритми та PSO є ефективними для мультимодальних функцій, але їхній високий обчислювальний час пов’язаний із великою кількістю ітерацій і необхідністю обчислення функції цілі для кожного члена популяції.

4. **Гібридні методи**:

Наприклад, поєднання глобального пошуку (для уникнення локальних мінімумів) із локальним уточненням результату (швидкий градієнтний спуск) дозволяє досягти оптимального співвідношення витрат і точності.

Метрики для аналізу витрат

1. Час досягнення оптимального рішення: відображає реальні часові витрати.
2. Кількість обчислень функції цілі: Особливо важлива для задач із дорогими оцінками.
3. Співвідношення «час–точність»: Порівняння витрат на пошук рішення із досягнутою точністю.
4. Ресурсна ефективність: вимірюється як відношення отриманого результату до витрат (часу, пам’яті, енергії).

Аналіз витрат дозволяє оцінити практичну ефективність методів оптимізації в різних сценаріях. У виборі алгоритму часто доводиться балансувати між обчислювальними витратами та точністю результату. Методи з низькими витратами є більш придатними для задач із великими розмірностями або

обмеженими ресурсами, тоді як більш затратні алгоритми можуть забезпечити високу точність у критичних задачах. Проведення такого аналізу є ключовим етапом у прийнятті рішень щодо застосування конкретного методу в реальних умовах.

3.8. Гібридні методи тестування ефективності методів оптимізації

Гібридні методи тестування ефективності методів оптимізації поєднують різні підходи для більш глибокої оцінки алгоритмів. Вони об’єднують сильні сторони окремих методів тестування, дозволяючи враховувати широкий спектр факторів: точність, стійкість, швидкість збіжності, витрати ресурсів та адаптивність до різних умов. Такий підхід є особливо цінним для задач, які характеризуються високою складністю, мультимодальністю чи непередбачуваністю.

Особливості гібридних методів тестування

1. **Комбінування теоретичного й емпіричного аналізу**:
    - Теоретичний аналіз забезпечує розуміння поведінки алгоритму в ідеальних умовах, тоді як емпіричне тестування виявляє його реальну продуктивність у практичних сценаріях.
    - Наприклад, для градієнтного спуску можна спочатку аналізувати збіжність на випуклих функціях аналітично, а потім перевірити його поведінку на шумових або невипуклих функціях емпірично.
2. **Поєднання детермінованих і стохастичних тестів**:
    - Детерміновані тести дозволяють оцінити поведінку алгоритму за фіксованих умов, тоді як стохастичні тести враховують вплив випадковості, наприклад, при варіюванні початкових точок або наявності шуму [33].

- Метод рою частинок (PSO) може бути протестований на функціях Розенброка із зафіксованими параметрами, а потім на задачах із випадковими початковими умовами для аналізу стійкості.

3. **Тестування на різних наборах функцій**:
   - Гібридний підхід передбачає використання як стандартних бенчмарків (наприклад, функцій Швефеля, Акермана), так і реальних практичних задач.
   - Це дозволяє перевірити адаптивність методів до різних класів проблем, від простих випуклих до складних багатояристих.
4. **Аналіз ефективності за кількома метриками**:
   - Використовуються різні критерії: швидкість збіжності, точність, витрати ресурсів, стійкість до змін параметрів. Важливо поєднувати кілька метрик для комплексної оцінки.
   - Наприклад, для порівняння методів еволюційної оптимізації та градієнтного спуску може бути враховано час збіжності, кількість ітерацій, стабільність результатів і обчислювальну складність.

Етапи гібридного тестування

1. **Підготовчий етап**:
   - Визначення набору тестових функцій із різними характеристиками (гладкість, розмірність, мультимодальність).
   - Вибір параметрів і умов тестування для кожного методу, зокрема діапазонів початкових значень, рівнів шуму, обмежень тощо.
2. **Проведення комбінованих експериментів**:
   - Спочатку виконуються теоретичні розрахунки або симуляції для визначення базової ефективності методів.

- Після цього проводяться емпіричні експерименти на реальних і штучно створених задачах, щоб підтвердити або уточнити отримані результати.

3. **Ітеративне покращення сценаріїв тестування**:
   - На основі проміжних результатів уточнюються параметри тестування. Наприклад, якщо алгоритм демонструє погані результати на функціях із високою розмірністю, додаються додаткові задачі для аналізу цього аспекту.
4. **Аналіз результатів із використанням візуалізації**:
   - Побудова графіків збіжності, гістограм розподілу значень функції цілі, тривимірних візуалізацій траєкторій пошуку тощо.
5. **Систематизація результатів**:
   - Узагальнення отриманих даних у вигляді порівняльних таблиць або інтегральних метрик (наприклад, зваженого середнього результатів для різних задач).

Приклади гібридного тестування

1. **Тестування градієнтних методів і PSO**:
   - Гібридний підхід дозволяє перевірити ефективність градієнтного спуску на випуклих функціях із високою розмірністю, а PSO — на мультимодальних функціях. Потім обидва методи тестуються на функціях зі штучним шумом для оцінки стійкості.
2. **Гібридизація глобального і локального пошуку**:
   - Алгоритми, які використовують глобальний пошук (наприклад, генетичні алгоритми), комбінуються із методами локального уточнення (градієнтний спуск). Ефективність тестується на задачах із різними рівнями складності.
3. **Випробування еволюційних стратегій із динамічними обмеженнями**:

- Умови тестування змінюються в процесі експерименту, що дозволяє перевірити адаптивність методів до динамічних середовищ.

Переваги гібридного підходу

1. **Комплексність**:
   - Дає можливість оцінити методи з різних перспектив, враховуючи всі аспекти їхньої роботи.
2. **Гнучкість**:
   - Дозволяє адаптувати сценарії тестування до специфіки задачі.
3. **Об’єктивність**:

Поєднання різних методів мінімізує упередженість і дозволяє отримати більш достовірні результати.

Фрагмент програмного коду

```
private void button10_Click(object sender, EventArgs e)
 {
     double xw, yw, st = 0.1, tv = 0.01;
     xw = xnn; yw = ynn;
     Pen mypen = new Pen(Color.Black, 0.01F);
     // Найшвидший  крок
     double xtemp, ytemp;
     // while (Math.Abs(dxa(x, y)) + Math.Abs(dya(x, y)) > tv)
     for (int i = 1; i < 10; i++)
     {
         xtemp = xw;
         ytemp = yw;
         vnapr(ref xw, ref yw);
         mg.DrawLine(mypen, Convert.ToSingle(xtemp), Convert.ToSingle(ytemp),
Convert.ToSingle(xw), Convert.ToSingle(yw));

     }
     xnn = xw; ynn = yw;
 }
 private void vnapr(ref double xr, ref double yr)
 {
     double x, y, st = 0.2, tv = 0.01;
     int i = 1;
     x = xr; y = yr; // початкова точка
    // double f1, f2;
    // f1=f(x, y);
     // f2 = f(x - dxa(x, y) * st, y - dya(x, y) * st);
     while (f(x, y) > f(x - dxa(x, y) * st, y - dya(x, y) * st))
```

```
    {
        st = st * 1.65;
    } // діапазон пошуку
    double x1, x2, x3, x4 = 1;
    x1 = 0;
    x2 = st;
    x3 = x1 + st * 0.618;

    Double temp;
    while (Math.Abs(x2 - x1) > tv)
    {
        if (i > 10) break;
        i++;
        x4 = x1 + (x2 - x3);
        if (x4 < x3)
        {
            temp = x4;
            x4 = x3;
            x3 = temp;
        }
        if (f(x - dxa(x, y) * x3, y - dya(x, y) * x3) > f(x - dxa(x, y) * x4, y -
dya(x, y) * x4))
        {
            x1 = x3;
            x3 = x4;
        }
        else { x2 = x4; }
    }
    x = xr; y = yr;
    xr = xr - dxa(x, y) * x3;
    yr = yr - dya(x, y) * x3;
    //xr = xr - dx(xr) * x3;       yr = yr - dy(yr) * x3;
 }
 private void vnaprgr(ref double xr, ref double yr)
 {
    double x, y, st = 0.2, tv = 0.01;
    x = xr; y = yr; //початкова точка
    while (ngr(x, y) < ngr(x - dxa(x, y) * st, y - dya(x, y) * st))
    {
        st = st * 1.65;
    } // діапазон пошуку
    double x1, x2, x3, x4 = 1;  // точки в діаназоні пошуку
    x1 = 0;
    x2 = st;
    x3 = x1 + st * 0.618;
    int i = 1;

    while (x2 - x1 > tv)
    {
        if (i > 29) break;
        i++;
        x4 = x1 + (x2 - x3);
        if (x4 < x3)
        {
            Double temp;
            temp = x4;
            x4 = x3;
            x3 = temp;
        }
        if (ngr(x - dxa(x, y) * x3, y - dya(x, y) * x3) > ngr(x - dxa(x, y) * x4,
y - dya(x, y) * x4))
        {
            x1 = x3;
```

```
                x3 = x4;
            }
            else { x2 = x4; }
        }
        x = xr; y = yr;
        xr = xr - dxa(x, y) * x3;
        yr = yr - dya(x, y) * x3;
    }
        }
}
```

3.9. Тестування методів градієнтного спуску

Програмний інтерфейс запропонованої комп'ютерної програми наведено на рис.1.

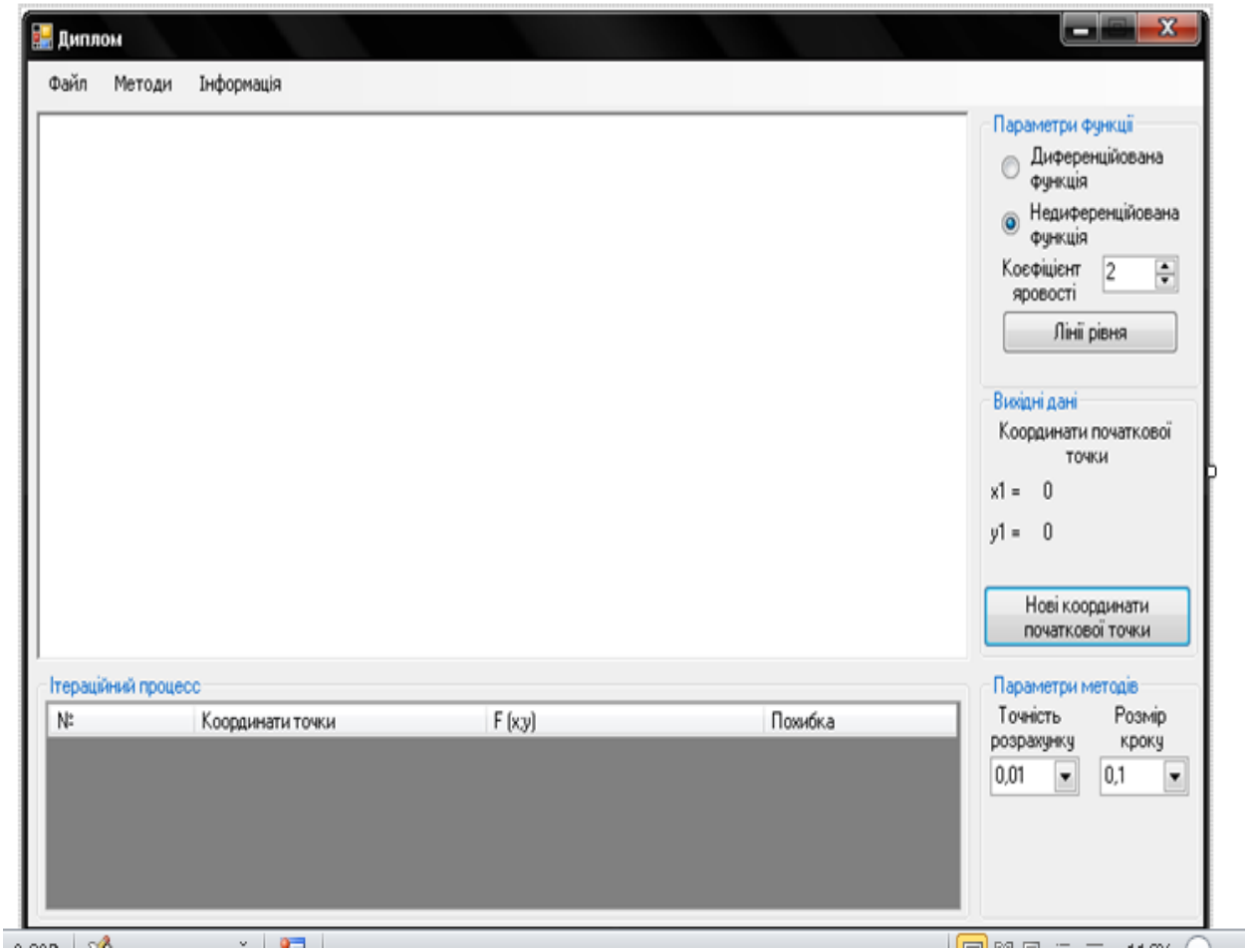


Рис. 1 Програмний інтерфейс

Основними функціями програмної системи є:

1.Програмування та відображення ліній рівня типових функцій, які можуть бути використані для демонстрації можливостей алгоритмів нульового

порядку.

2. Програмування найбільш поширених алгоритмів нульового порядку (опуклий симплекс метод).

3. Програмування та відображення ліній, що відображають траєкторію спуску алгоритмів.

Приклади результатів роботи програми по дослідженню симплекс-методу пошуку екстремальних значень наведено нижче у графічному вигляді.

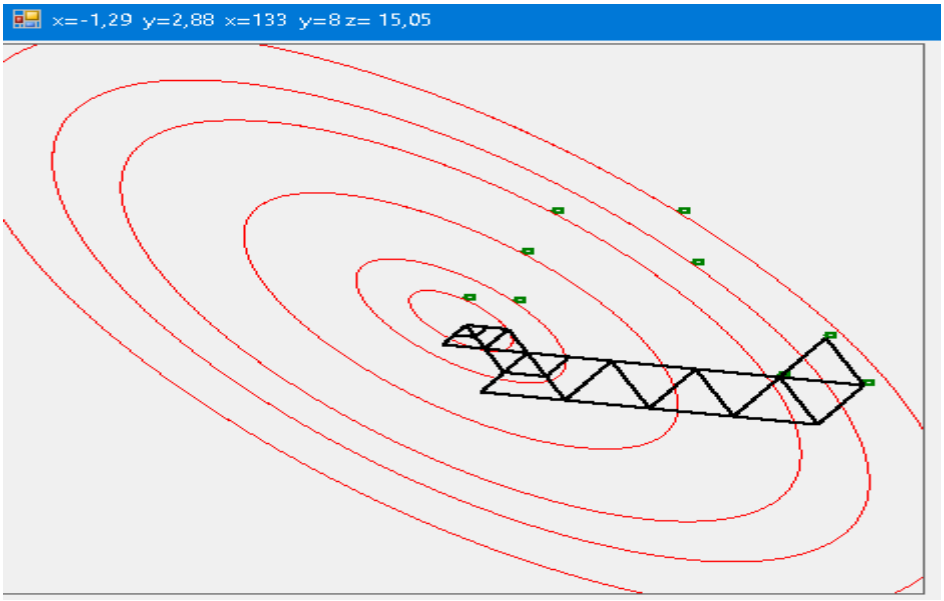


Рис. 2 Траєкторія симплекс -методу для еліптичної функції

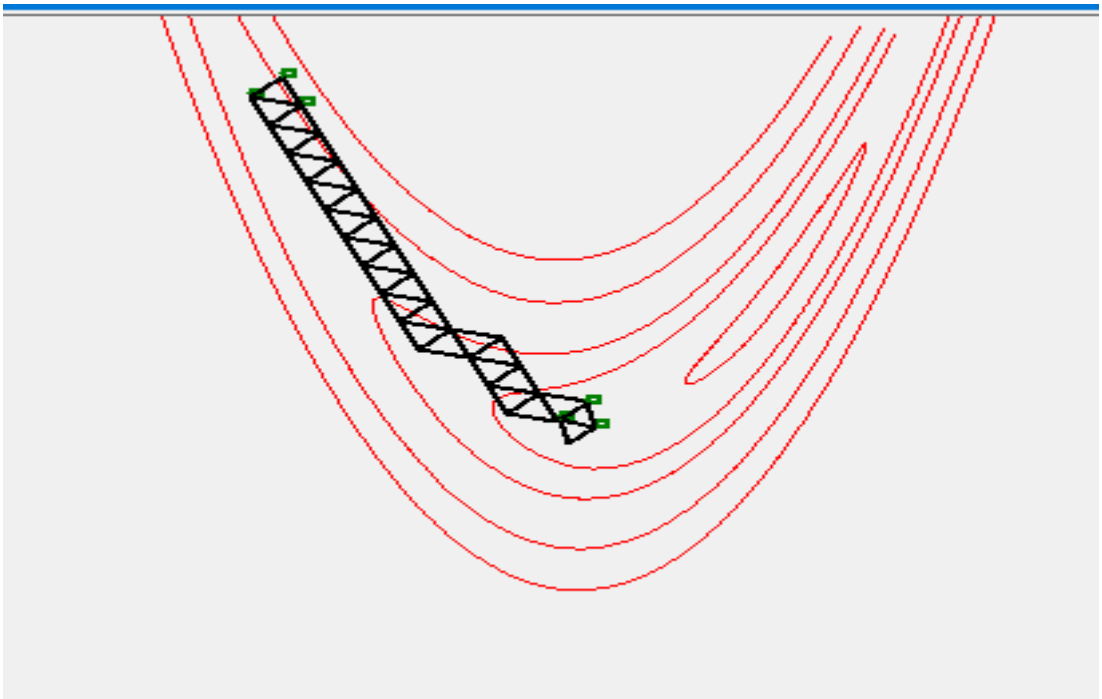


Рис. 3 Траєкторія симплекс-методу для функції Розенброка

З рисунків можна побачити деякі недоліки методу.

1. Певні труднощі, пов'язані з масштабуванням, оскільки всі координати вершин симплексу залежать від єдиного масштабного множника α. Щоб обійти такі труднощі, у практичних задачах треба про масштабувати всі змінні для того, щоб їх значення були приблизно однаковими за величиною.

2. Отримана на попередніх ітераціях інформація не використовується для прискорення пошуку, що призводить до зниження швидкості розрахунків.

3. При розширенні задачі потрібен перерахунок значень цільової функції у всіх точках зразка. Зокрема, якщо з якої-небудь причини цільова функція зменшується (наприклад, якщо зустрічається область із вузьким «яром» або «хребтом»), то пошук повинен тривати із зменшеною величиною кроку.

Висновки до розділу 3

У даному розділі наводиться класифікація способів дослідження ефективності методів розв’язування задач теорії оптимізації. сформульованих у вигляді пошуку екстремальних значень цільової функції за наявністю тих чи інших обмежень. Сформульовано основні критерії ефективності та способи їх реалізації. Розроблено комп’ютерну програму для тестування методів градієнтного спуску.

Висновки

В даній дипломній роботі розглядаються питання, що пов’язані з існуючими на даний час способами дослідження якостей методів розв’язання задач оптимізації. Розглянуто такі аспекти методів як швидкість збіжності, гарантії оптимальності та обчислювальна складність, визначення стійкості методів оптимізації (робастність).

В якості способів перевірки обрано теоретичний аналіз та емпіричне тестування ефективності методів оптимізації, статистичний аналіз, гібридне тестування, методи візуалізації, зміст яких розкривається в розділі 3.

Емпіричне тестування дає змогу глибше зрозуміти особливості поведінки методів оптимізації в реальних умовах. Воно показує, що жоден метод не є універсальним: ефективність алгоритму залежить від типу задачі, властивостей функції та обмежень. У процесі тестування важливо враховувати як якісні, так і кількісні показники, щоб забезпечити повноцінну оцінку методів оптимізації та сприяти їхньому подальшому вдосконаленню.

Статистичний аналіз ефективності методів оптимізації є ключовим етапом для обґрунтованого вибору алгоритму. Він дозволяє виявити закономірності в поведінці методів, підтвердити або спростувати гіпотези, а також розробити рекомендації щодо вибору алгоритму залежно від конкретних умов. Цей підхід, у поєднанні з теоретичним і емпіричним аналізом, забезпечує комплексну оцінку методів оптимізації.

Оцінка робастності ефективності методів оптимізації дозволяє зробити висновки про їхню придатність для використання в умовах невизначеності. Робастні алгоритми є більш універсальними, адже забезпечують стабільні результати в широкому діапазоні умов. Такий аналіз є важливим етапом у розробці нових методів оптимізації, а також у виборі алгоритму для конкретної задачі.

Гібридні методи тестування ефективності дозволяють отримати повну картину про можливості алгоритмів оптимізації, включаючи їхні сильні та слабкі сторони. Вони є особливо корисними в задачах із високою невизначеністю або складністю, забезпечуючи надійність результатів і допомагаючи обрати найкращий метод для конкретного застосування.

Дослідження ефективності є важливим інструментом для розуміння меж застосовності кожного методу оптимізації. Зазначене дослідження дозволяє виявити оптимальні умови використання, наприклад, для яких типів задач конкретний метод є найбільш ефективним. Також такий аналіз допомагає у розробці нових, більш ефективних алгоритмів шляхом врахування переваг існуючих підходів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Meise, R., Vogt.,D. Functional Analysis: An Introduction, Oxford University Press, 1997, 450 p.

2. Кривий С.Л. Дискретна математика, *Чернівці – Київ, Букрек*, 2017, 567 с.

3. Жалдак М.І., Триус Ю.В. Основи теорії і методів оптимізації, *Брама-Україна*, 2005. 607 с.

4. Щербань В.Ю., Краснитський С.М., Астістова Т.І., Яхно В.М. Методи представлення, збереження та аналізу даних інформаційних систем, *Київ, Фастбінд Україна*, 2023, 470 с.

5. Boyd,S., Vandenberghe,L., *Convex Optimization* ,Cambridge University Press, 2004, 716 p.

6. Kidd, J., *Managing with Operations Research (ed.)*, Heritage Publishers, New Delhi, 1986.

7. Knotts, U.S. (Jr) and E.W. Swift, *Management Science for Management Decisions*, Allyn and Bacon Inc., Massachussetts, 2019.

8. Kwak, N.K., *Mathematical Programming with Business Applications*, McGraw-Hill, New York, 2017.

9. Krajewski, L.J. and H.E. Thompson, *Management Science*, John Wiley & Sons, New York, 2019.

10. Luenberger,D. G., *Optimization by Vector Space Methods*. Wiley, 1997, 326 p.

11. Lapin, L., *Quantitative Methods for Business Decisions (6th Edn.)*, The Dryden Press, New York, 2014.

12. Levin, R.I., C.A. Kirkpatrick and D.S. Rubin, *Quantitative Approach to Management (5th Edn.)*, McGraw-Hill, Singapore, 2014.

13. Lee, S.M., L.J. Moore and B.W. Tayler, *Management Science*, Allyn and Bacon, M.A., 2019. Lee, S.M., *Goal Programming for Decision*

Analysis, Auerbach Publishers, Philadelphia, 2012.

14. Lee, S.M. and L.J. Moore, *Introduction to Decision Science*, Petroceli/Charter Inc., New York, 201975. Loomba, N.P., *Management – A Quantitative Perspective*, Macmillan Publishing Company, New York, 2018.

15. Mathur, K., and D. Solow, *Management Science*, Prentice-Hall, Inc., 2014.

16. Markland, R.E., *Topics in Management Science*, John Wiley & Sons, 201979.

17. Mitchell, G.H., *Operations Research Techniques and Examples*, The English Univ. Press Ltd., London, 2017.

18. Nar Singh Deo, *System Simulation with Digital Computer*, Prentice-Hall of India, New Delhi, 2017.

19. Phillips, D.T., A. Ravindra and J.J. Solberg, *Operations Research: Principles and Practice*, Wiley, New York, 2016.

20. Raifa, H., *Decision Analysis*, Reading Mass, Addition-Wesley Publishing Co., Philippines, 2018.

21. Saatty, T.L., *Mathematical Methods of Operations Research*, McGraw-Hill, New York, 2019.

22. Simmons, D.M., *Non-linear Programming for Operations Research*, Prentice-Hall, Englewood Cliffs, N.J., 2015.

23. Sharma, J.K., *Quantitative Techniques for Managerial Decisions*, Macmillan India Ltd., New Delhi, 2001.

24. Taha, H.A., *Operations Research – An Introduction*, Prentice-Hall Inc., New Jersey, 2017.

25. Trueman, R.E., *Quantitative Methods for Decision Making in Business*, Holt-Saunders, New York, 201981.

26. Turban, E. and J.R. Meredith, *Fundamentals of Management Science (3rd Edn.)*, Business Applications Inc. Taxes, 2015.

27. Vajda, S., *Theory of Linear and Non-linear Programming*, Longman,

London, 201974.

28. Vazsonyi, A. and H.F. Spiror, *Quantitative Analysis for Business*, Prentice-Hall of India, New Delhi,

29. 2017. Waye, E.L., *Quantitative Methods in Accounting*, D. Van Nostrand Company, New York, 2010.

30. Wanger, H.M., *Principles of Operations Research* with *Applications to Management Science*, Prentice-Hall of India, New Delhi, 201982.

31. Wisniewski, M., *Quantitative Methods for Decision Makers*, Macmillan India Ltd., New Delhi, 201996.

32. Wiest, J.D. and F.K. Levy, *A Management Guide to PERT/CPM*, Prentice-Hall Englewood Cliffs, N.J., 201969. Winston, W.I., *Operations Research: Applications and Algorithms*, Duxbury Press, California, 2014.

33. Zoints, S., *Linear and Integer Programming*, Prentice-Hall, Englewood Cliffs, N.J., 201974. Zountendijk, G., *Mathematical Programming Methods*, North-Holland, Amsterdam, 2016.

ДОДАТОК

Лінії рівня

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;

namespace LinRiv
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        /// <summary>
        ///
        /// </summary>

        /// <param name="z"></param>
        /// <param name="x"></param>
        /// <returns></returns>
        private double det(double z, double x)
        {
            return z - (x - 3) * (x - 3);
        }
        private double yplus(double detx)
        {

            return +Math.Sqrt((float)detx / 5);
        }
        private double yminus(double detx)
        {
            return -Math.Sqrt((float)detx / 5);
        }
        private double f(double x, double y)
        {
            return Math.Pow((x - 3), 2) + 5 * (y) * (y);
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        double xmi, xma, ymi, yma, stepxg, stepxu, z, dex, dey, me11,
me22;
        private void mymove(object sender, MouseEventArgs e)
```

```
        {
            double x, y, xg, yg;
            x = e.X;
            y = e.Y;

            xg = ((x - dex) / me11);
            yg = ((y - dey) / me22);


            this.Text = "x=" + Convert.ToString(xg) + "  y=" +
Convert.ToString(yg) + "  x=" + Convert.ToString(x) + "  y=" +
Convert.ToString(y) + "z=" + Convert.ToString(f(xg, yg));

        }
        private Single xpovorot(double x, double y)
        {
            float ugol = (float)(Math.PI / 4);
            return (float)(x * Math.Cos(ugol) - y * Math.Sin(ugol));
        }
        private Single ypovorot(double x, double y)
        {
            float ugol = (float)(Math.PI / 4);
            return (float)(x * Math.Sin(ugol) + y * Math.Cos(ugol));
        }
        private void drowmy(object sender, MouseEventArgs e)
        {
            this.Refresh();

            Graphics mg;
            mg = this.CreateGraphics();

            xmi = 1;
            xma = 5;
            stepxg = 0.001F;
            stepxu = 0.1F;
            ymi = 999999;
            yma = -9999999;

            double x, xr, ys;
            ys = 0;
            z = f(xmi, ys);
            for (x = xmi; x <= xma; x = x + stepxg)
            {
                if (det(z, x) >= 0)
                {
                    if (yminus(det(z, x)) < ymi)
                    {
                        ymi = yminus(det(z, x));
                    }
                    if (yplus(det(z, x)) > yma)
                    {
                        yma = yplus(det(z, x));
                    }

                }
            }
```

```
            xmi = 1;
            xma = 5;
            stepxg = 0.001F;
            stepxu = 0.1F;
            ymi = -2;
            yma = +2;

            float m11 = (float)((this.Size.Width) / (xma - xmi));
            float m12 = 0;
            float m21 = 0;
            float m22 = (float)(-(this.Size.Height) / (yma - ymi));
            float dx = (float)(-xmi * m11);
            float dy = (float)(this.Size.Height - m22 * ymi);
            ///xp=m11*xg + dx   yp=m22*yg+dy

            dex = dx;
            dey = dy;
            me11 = m11;
            me22 = m22;
            System.Drawing.Drawing2D.Matrix myMatrix = new
System.Drawing.Drawing2D.Matrix(m11, m12, m21, m22, dx, dy);
            //myMatrix.Rotate((float)(Math.PI /4));

            mg.Transform = myMatrix;
            Pen mypen = new Pen(Color.Red, 0.01F);
            double st;
            xr = 1; st = 2;
            z = f(xmi, ys);
            //mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yplus(det(z, xr))), Convert.ToSingle(xr + st),
Convert.ToSingle(yplus(det(z, xr ))));
            for (x = xmi; x <= xma; x = x + stepxu)
            {
                z = f(x, ys);
                for (xr = x; xr <= xma - (x - xmi); xr = xr + stepxg)
                {
                    if (det(z, xr) >= 0 & det(z, xr + stepxg) >= 0)
                    {
                        mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yplus(det(z, xr))), Convert.ToSingle(xr + stepxg),
Convert.ToSingle(yplus(det(z, xr + stepxg))));
                        mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yminus(det(z, xr))), Convert.ToSingle(xr + stepxg),
Convert.ToSingle(yminus(det(z, xr + stepxg))));

                    }
                }
            }
            return;
            //myMatrix = new Matrix();
            mypen = new Pen(Color.Green, 0.01F);
            myMatrix.Rotate((float)(4 * Math.PI));
            //myMatrix.Scale(1, 2, MatrixOrder.Append);
            //myMatrix.Translate(5, 0, MatrixOrder.Append);
            mg.Transform = myMatrix;
```

```
            xr = 1; st = 2;
            z = f(xmi, ys);
            mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yplus(det(z, xr))), Convert.ToSingle(xr + st),
Convert.ToSingle(yplus(det(z, xr))));

            for (x = xmi; x <= xma; x = x + stepxu)
            {
                z = f(x, ys);
                for (xr = x; xr <= xma - (x - xmi); xr = xr + stepxg)
                {
                    if (det(z, xr) >= 0 & det(z, xr + stepxg) >= 0)
                    {
                        mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yplus(det(z, xr))), Convert.ToSingle(xr + stepxg),
Convert.ToSingle(yplus(det(z, xr + stepxg))));
                        mg.DrawLine(mypen, Convert.ToSingle(xr),
Convert.ToSingle(yminus(det(z, xr))), Convert.ToSingle(xr + stepxg),
Convert.ToSingle(yminus(det(z, xr + stepxg))));

                    }
                }
            }

        }
```