МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни «Математичні методи оптимізації»
для студентів спеціальності
G7 «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 19.02.2026 р.

Харків
НТУ «ХПІ»
2026

Методичні вказівки до виконання лабораторних робіт з дисципліни «Математичні методи оптимізації» для студентів спеціальності G7 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» / уклад. І.Л. Красніков, А.К. Бабіченко, Я.О. Кравченко – Харків : НТУ «ХПІ». – 40 с.

Укладач І.Л. Красніков, А.К. Бабіченко, Я.О. Кравченко

Рецензент А.О. Зуєв

Кафедра автоматизації технологічних систем та екологічного моніторингу

ЛАБОРАТОРНА РОБОТА № 1

ГЕОМЕТРИЧНЕ ПОДАННЯ ЗАДАЧ ОПТИМІЗАЦІЇ

Мета роботи: ознайомитися з геометричними поданнями одновимірних і багатовимірних задач оптимізації. Навчитися будувати та аналізувати лінії рівня цільових функцій за допомогою *MATLAB*

1.1. Загальні положення

Математичне формулювання задач оптимізації зводиться до знаходження екстремуму функції однієї або декількох змінних:

$$R = R(U_1, U_2, ..., U_N) \rightarrow extr. \qquad (1.1)$$

На незалежні змінні функції R можуть накладатися обмеження двох типів. Обмеження типу рівність задають у вигляді

$$h_i(U) = 0 \quad i = 1,2...m, \qquad m < n$$

а обмеження типу нерівність задають у вигляді

$$g_j(U) >= 0, \qquad j = 1,2...k.$$

Функцію (1.1) називають цільовою функцією.

Для зручності аналізу задач оптимізації доцільно використовувати їх геометричне подання. У разі одновимірної задачі це графік функції $R(U)$. Для задач із двома та більшою кількістю змінних застосовують геометричні уявлення у вигляді поверхні $R(U)$ та ліній рівня, тобто множин точок, для яких $R(U) = C$, де C є сталою.

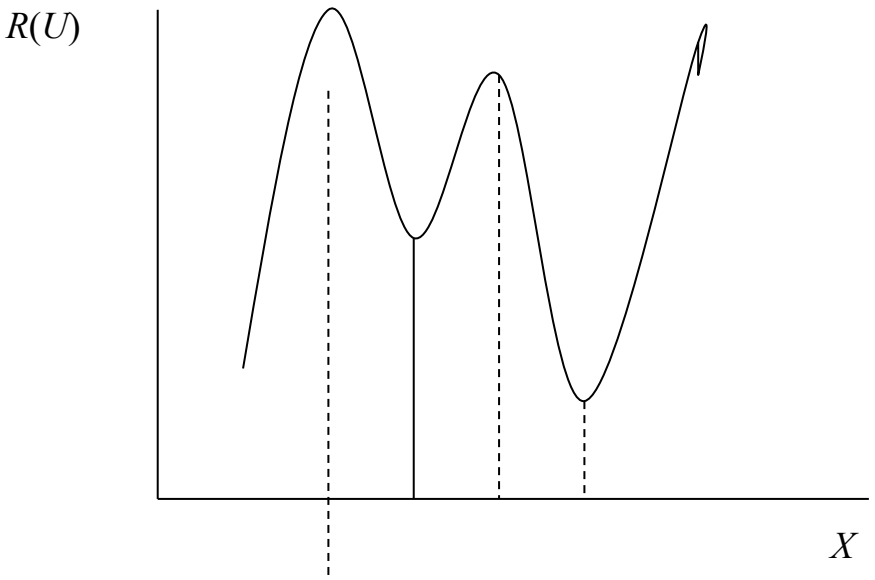


Рис. 1.1. Геометричне подання одномірної функції

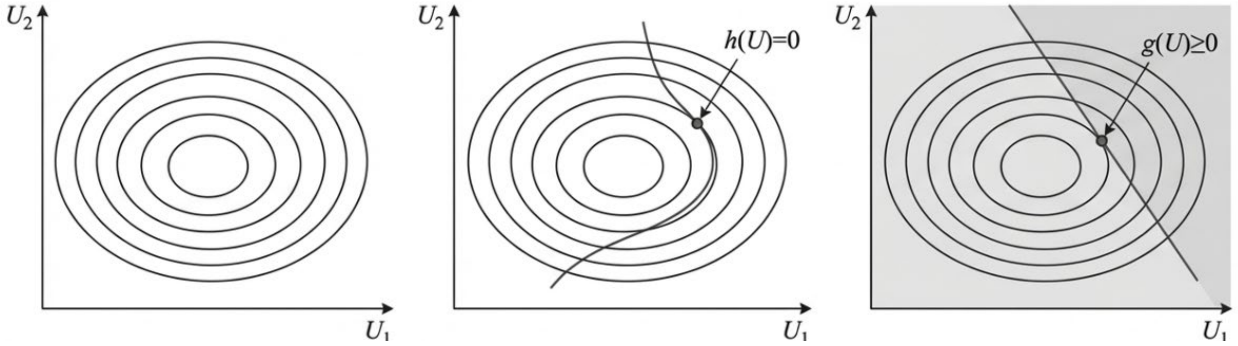


Рис. 1.2. Геометричне подання двомірної функції

Цільова функція може мати кілька екстремумів одного типу. У такому разі як оптимальне значення обирають глобальний екстремум потрібного типу. Під час пошуку максимуму це найбільше зі значень у точках максимуму, а під час пошуку мінімуму це найменше зі значень у точках мінімуму. Відповідно, такі екстремуми називають глобальними, а решту екстремумів того самого типу локальними.

1.2. Особливі точки і лінії цільової функції

Пошук оптимуму істотно ускладнюється, якщо рельєф цільової функції має сідлові точки та ярові ділянки. У сідловій точці в напрямі однієї

змінної може спостерігатися максимум, тоді як у напрямі іншої змінної спостерігається мінімум (див.рис.1.3).

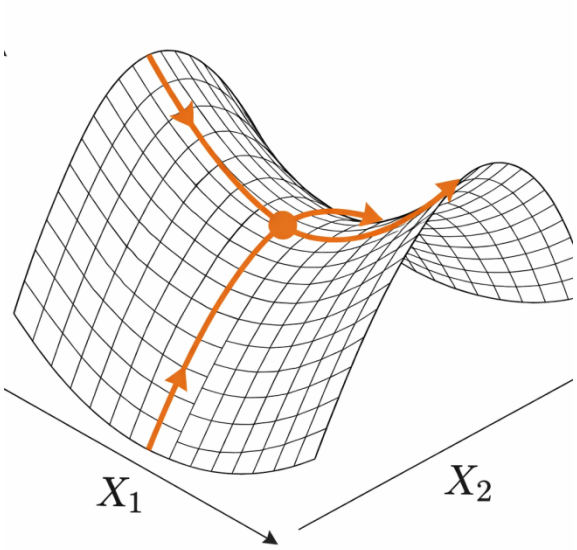


Рис. 1.3. Сідлова функція

Іншим типом особливостей є яри. За наявності ярової структури вздовж певного напряму значення цільової функції змінюється дуже слабо, тоді як у поперечному напрямі зміна є значно інтенсивнішою (див.рис.1.4). На лініях рівня це зазвичай відповідає витягнутим замкненим контурам, а на поверхні спостерігається вузька долина, що ускладнює локалізацію екстремуму та може зменшувати швидкість збіжності чисельних методів.

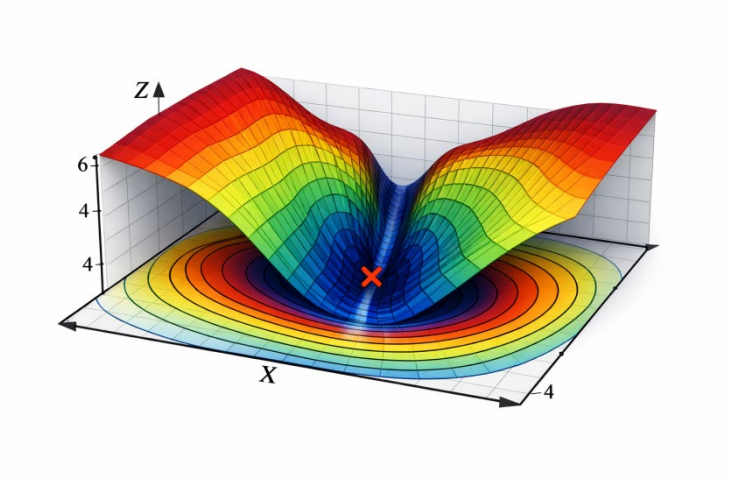


Рис. 1.4. Ярова функція

1.3. Порядок виконання роботи

Побудувати у *MATLAB* лінії рівня для функцій:

1. R1(x, y) = (x - 2)^2 + (y - 1)^2 - 4.
2. R2(x, y) = 4x^2 + y^2 - 2x - 2y + 13.
3. R3(x, y) = (x^2 + y - 11)^2 + (y + x - 7)^2 (функція Хіммельблау).
4. R4(x, y) = (1.5 - x(1 - y))^2 + (2.25 - x(1 - y^2))^2 + (2.625 - x(1 - y^3))^2 (функція Біла).
5. R5(x, y) = 3(1 - x)^2·exp(-x^2 - (y + 1)^2) - 10(x/5 - x^3 - y^5)·exp(-x^2 - y^2) - (1/3)·exp(-(x + 1)^2 - y^2).
6. R6(x, y) = (1 - x)^2 + 100(y - x^2)^2 (функція Розенброка).

1.4. Приклад реалізації в *MATLAB*

Для виконання лабораторної роботи студент створює файл програми *L1.m* та формує його шляхом копіювання наведених нижче фрагментів коду. Фрагменти необхідно вставити у вказаному порядку без пропусків. Після збереження файла доцільно виконати оновлення інформації про функції та очистити кеш, щоб уникнути запуску застарілої версії з іншого каталогу:

```
clear functions
rehash
```

Під час виконання програма формує для кожної тестової функції окреме графічне вікно, у якому відображаються чотири подання: просторова сітка поверхні (`mesh`), просторова поверхня з освітленням (`surfl`), лінії рівня (contour) та заповнені лінії рівня (`contourf`). Такий набір графіків дозволяє порівнювати рельєф функції та структуру ліній рівня для подальшої якісної інтерпретації.

```
function Lab1(MODE, k)
% Lab 1: Geometric representation of optimization problems
% MATLAB R2014a compatible.
```

```
%
% Usage:
%   Lab1         % MODE=2 (all 6 functions)
%   Lab1(1, 5)      % MODE=1, k=5
%
% MODE = 1 -> plot one function (k)
% MODE = 2 -> plot all functions 1..6

if nargin < 1
    MODE = 2;
end
if nargin < 2
    k = 5;
end

close all; clc;

if MODE == 1
    plotOne(k);
else
    for kk = 1:6
        plotOne(kk);
    end
end

    function plotOne(kLocal)
        switch kLocal
            case 1
                x = linspace(-5, 5, 200);
                y = linspace(-5, 5, 200);
                [X, Y] = meshgrid(x, y);
                Z = (X - 2).^2 + (Y - 1).^2 - 4;
                cmap = autumn;
                figName = 'F1: (x-2)^2 + (y-1)^2 - 4';

            case 2
                x = linspace(-5, 5, 200);
                y = linspace(-5, 5, 200);
                [X, Y] = meshgrid(x, y);
                Z = 4*X.^2 + Y.^2 - 2*X - 2*Y + 13;
                cmap = winter;
                figName = 'F2: 4x^2 + y^2 - 2x - 2y + 13';

            case 3
                x = linspace(-6, 6, 200);
                y = linspace(-6, 6, 200);
                [X, Y] = meshgrid(x, y);
                Z = (X.^2 + Y - 11).^2 + (Y + X - 7).^2;
                cmap = spring;
                figName = 'F3: (x^2+y-11)^2 + (y+x-7)^2';
```

```
            case 4
                x = linspace(-4.5, 4.5, 200);
                y = linspace(-4.5, 4.5, 200);
                [X, Y] = meshgrid(x, y);
                Z = (1.5 - X.*(1 - Y)).^2 + (2.25 - X.*(1 -
Y.^2)).^2 + ...
                    (2.625 - X.*(1 - Y.^3)).^2;
                cmap = summer;
                figName = 'F4: Beale';

            case 5
                x = linspace(-3, 3, 200);
                y = linspace(-3, 3, 200);
                [X, Y] = meshgrid(x, y);
                Z = 3*(1 - X).^2 .* exp(-X.^2 - (Y + 1).^2) -
...
                    10*(X/5 - X.^3 - Y.^5) .* exp(-X.^2 - Y.^2)
- ...
                    (1/3)*exp(-(X + 1).^2 - Y.^2);
                cmap = hot;
                figName = 'F5: complex surface';

            case 6
                x = linspace(-2, 2, 200);
                y = linspace(-1, 3, 200);
                [X, Y] = meshgrid(x, y);
                Z = (1 - X).^2 + 100*(Y - X.^2).^2;
                cmap = copper;
                figName = 'F6: Rosenbrock';

            otherwise
                error('k must be 1..6');
        end

        figure('Name', figName, 'NumberTitle', 'off');

        subplot(2,2,1);
        mesh(X, Y, Z);
        colormap(cmap);
        xlabel('x'); ylabel('y'); zlabel('R(x,y)');
        title('3D (mesh)'); grid on;

        subplot(2,2,2);
        surfl(X, Y, Z);
        colormap(cmap);
        shading interp;
        alpha(0.8);
        xlabel('x'); ylabel('y'); zlabel('R(x,y)');
        title('3D (surfl)'); grid on;

        subplot(2,2,3);
```

```
        contour(X, Y, Z, 20);
        colorbar;
        xlabel('x'); ylabel('y');
        title('contour'); grid on;

        subplot(2,2,4);
        contourf(X, Y, Z, 20);
        colorbar;
        xlabel('x'); ylabel('y');
        title('contourf'); grid on;
    end

end
```

Після створення та збереження файла Lab1.m запуск програми виконують у двох режимах. Для побудови однієї тестової функції використовують виклик L1(1, k), де k є номером функції відповідно до таблиці варіантів (k = 1…6).

Для побудови всіх шести тестових функцій послідовно використовують виклик L1 без параметрів.

У межах лабораторної роботи студент змінює лише параметри MODE та k під час запуску. За потреби допускається коригування діапазонів x і y у блоці відповідної функції для отримання більш інформативного графічного подання.

1.5. Аналіз отриманих результатів

За результатами виконання програми студент отримує графічні подання цільової функції для заданого варіанта. У звіті необхідно навести побудовані графіки поверхні та ліній рівня.

Під час аналізу результатів слід дати якісну характеристику рельєфу цільової функції, зокрема визначити кількість і розташування мінімумів, наявність локальних екстремумів та можливих сідлових точок. Окрему увагу необхідно приділити формі ліній рівня та їх взаємному розташуванню.

У разі наявності ярової структури потрібно зазначити її характерні ознаки, що проявляються у вигляді витягнутих замкнених контурів на

площині ліній рівня та вузьких долин на поверхні функції. За результатами аналізу слід зробити висновок щодо складності пошуку екстремуму чисельними методами для даної цільової функції.

Контрольні питання

1. Дати визначення критерію оптимальності
2. Визначити, що таке цільова функція і які її властивості
3. Геометричне представлення одновимірних задач оптимізації
4. Геометричне представлення багатовимірних задач оптимізації
5. Пояснити обмеження в задачах оптимізації. Геометричне представлення багатовимірних завдань умовної оптимізації з обмеженнями
6. Що таке особливі точки і лінії цільової функції

ЛАБОРАТОРНА РОБОТА № 2
БЕЗГРАДІЄНТНІ МЕТОДИ ПОШУКУ ЕКСТРЕМУМУ ФУНКЦІЇ ОДНІЄЇ ЗМІННОЇ

Мета роботи: вивчити алгоритми методів виключення інтервалів і поліноміальної апроксимації та набути навичок практичної роботи з ними у *MATLAB*

2.1. Загальні положення

Безградієнтні методи пошуку екстремуму функції однієї змінної застосовують для задач мінімізації або максимізації на відрізку, коли похідна функції відсутня, важко обчислюється або її використання не передбачене умовами задачі. Типова постановка має вигляд $\min f(x)$ за умови $x \in [A, B]$, де A і B є межами допустимих значень змінної. Найпоширенішу групу безградієнтних методів для одномірної оптимізації становлять методи виключення інтервалів, у яких на поточному інтервалі обирають кілька пробних точок, обчислюють значення $f(x)$ та, за результатами порівняння, відкидають ту частину інтервалу, в якій екстремум знаходитися не може. Процес повторюють доти, доки довжина інтервалу локалізації не стане меншою за задану точність Δ. Для коректного застосування таких методів зазвичай вимагають унімодальності функції на $[A, B]$, тобто наявності лише одного мінімуму в межах цього інтервалу. Окремий клас підходів пов’язаний з апроксимацією функції на поточному інтервалі простішою моделлю. У випадку одномірної задачі найчастіше застосовують параболічну, тобто квадратичну, інтерполяцію за трьома точками, після чого мінімум наближувальної параболи знаходять аналітично.

2.2. Опис методів розрахунку

В одномірних задачах оптимізації безградієнтні методи реалізують пошук мінімуму на відрізку [*A*, *B*] шляхом послідовного звуження інтервалу локалізації та уточнення положення мінімуму за значеннями цільової функції *f(x)* у пробних точках. Далі розглядаються методи виключення інтервалів, зокрема метод п’яти точок і метод «золотого перетину», а також метод параболічної, тобто квадратичної, інтерполяції. Зазначені підходи покладено в основу алгоритму функції fminbnd середовища *MATLAB*.

2.2.1. Метод п'яти точок

Зону пошуку, тобто відрізок [*A*, *B*], розбивають на чотири рівні частини і обчислюють значення цільової функції в точках розбиття. У результаті одержують п’ять пробних точок X_1, X_2, X_3, X_4, X_5, як показано на рис. 2.1.

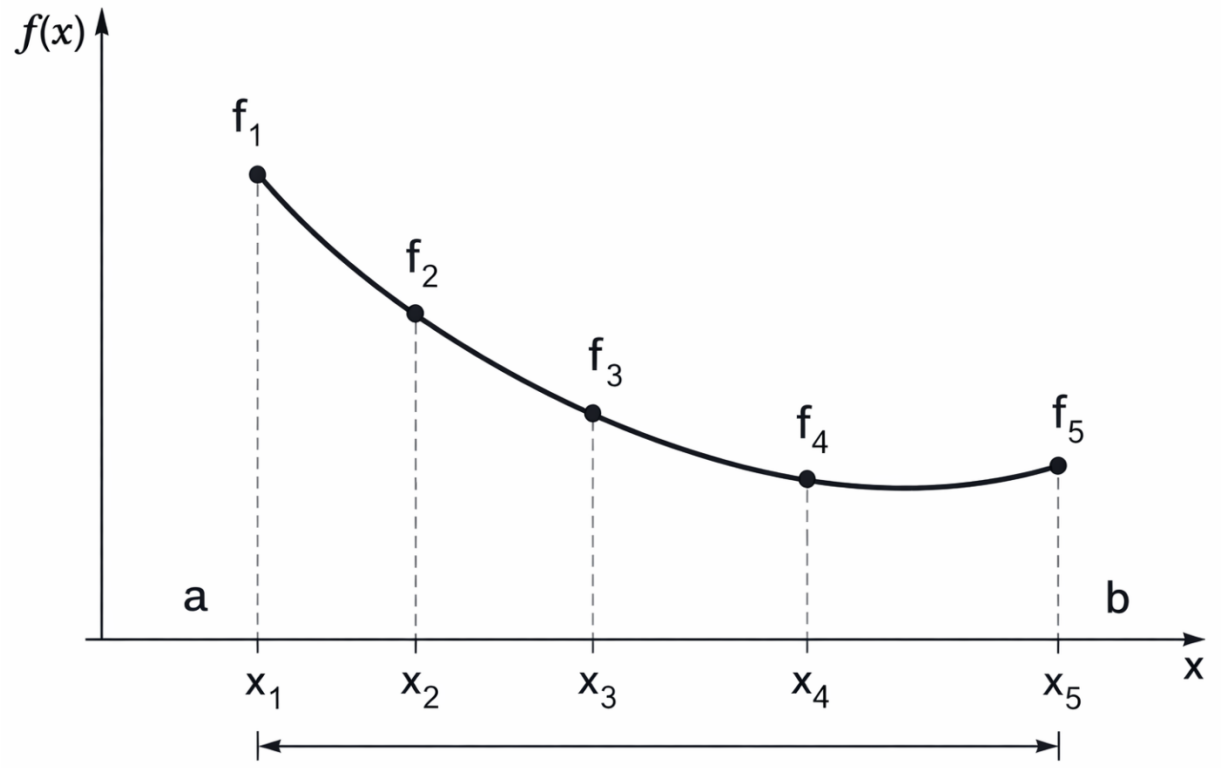


Рис. 2.1. Метод п’яти точок

Серед п’яти значень $f(X_1), f(X_2), f(X_3), f(X_4), f(X_5)$ визначають найменше. Якщо найменше значення досягається в точці $X3$, то нову зону пошуку вибирають як $[X_2, X_4]$. Якщо найменше значення досягається в точках X_1 або X_2, то нова зона пошуку $[X_1, X_3]$. Якщо найменше значення досягається в точках X_4 або X_5, то нова зона пошуку $[X_3, X_5]$. Далі процедуру звуження інтервалу повторюють доти, доки довжина інтервалу локалізації не стане меншою або рівною заданому значенню.

Алгорит м мет оду п'ят и т очок

Вихідні дані: A, ліва межа зони пошуку; B, права межа зони пошуку; Δ, точність локалізації мінімуму.

Крок 1. Задати A, B, Δ.

Крок 2. Обчислити C = (B − A)/4 та задати точки X_1 = A, X_2 = A + C, X_3 = A + 2C, X4 = A + 3C, X5 = B.

Крок 3. Обчислити $f(X_1), f(X_2), f(X_3), f(X_4), f(X_5)$.

Крок 4. Визначити точку Xmin, у якій значення f(Xmin) є найменшим серед $f(X_1), f(X_2), f(X_3), f(X_4), f(X_5)$. Якщо мінімальне значення досягається у двох або більше пробних точках, як *Xmin* прийняти ту з них, що має координату, найближчу до середини інтервалу, тобто до точки X_3.

Крок 5. Виконати звуження інтервалу: якщо Xmin = X_3, то [A, B] := $[X_2, X_4]$; якщо *Xmin* = X_1 або *Xmin* = X_2, то [A, B] := $[X_1, X_3]$; якщо Xmin = X_4 або *Xmin* = X_5, то $[A, B] := [X_3, X_5]$.

Крок 6. Якщо $(B - A) \le 2\Delta$, завершити пошук і прийняти $x^* = (A + B)/2$. Якщо $(B - A) > 2\Delta$, перейти до кроку 2.

Абсолютну похибку визначення положення мінімуму в момент завершення оцінюють як

$$\Delta abs \le (B - A)/2. \qquad (2.1)$$

2.2.2. Метод "золотого перетину"

Метод «золотого перетину» організовує звуження інтервалу [*A, B*] так, щоб після переходу до нового інтервалу одна з внутрішніх точок залишалася спільною для двох послідовних кроків. Це дає змогу повторно використовувати одне з уже обчислених значень *f(x)* і зменшувати кількість нових обчислень на кожній ітерації.

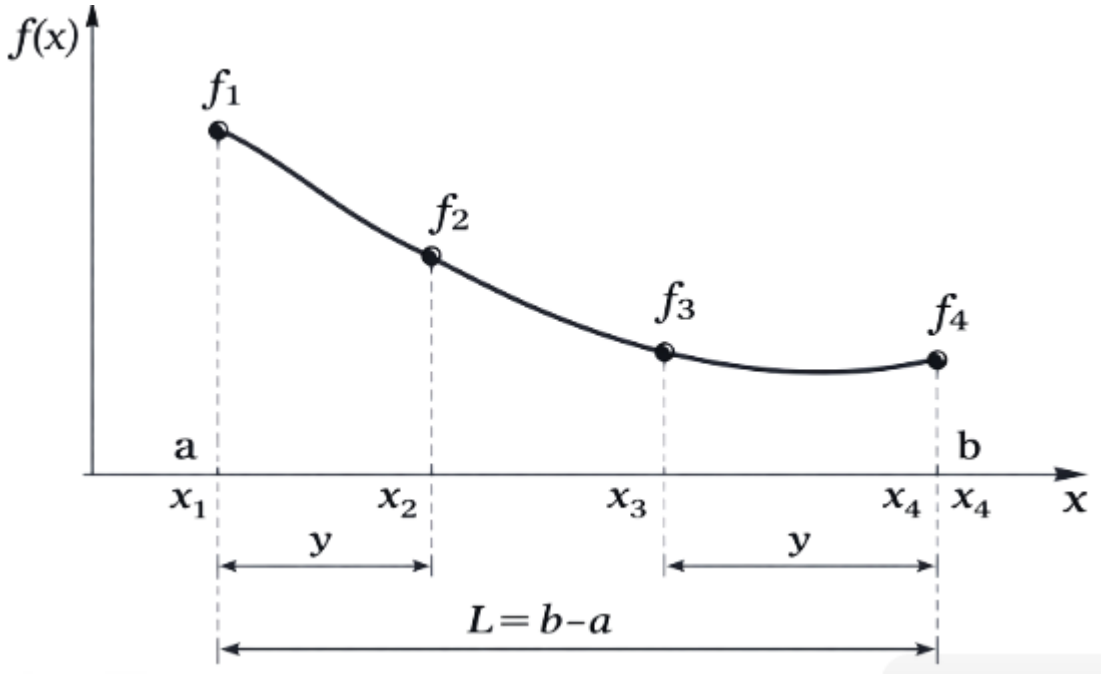


Рис. 2.2. Метод «золотого перетину»

Позначимо

$$Y = X_2 - X_1 = X_4 - X_3;\ L = B - A.$$

Тоді відрізок [X_1, X_4] розбивається на три частини: *Y, (L − 2Y), Y*.

Значення Y вибирається з умови, що після звуження інтервалу поділ нового інтервалу зберігає те саме співвідношення, що й поділ початкового. Цю умову записують у вигляді

$$\frac{Y}{L} = \frac{L - 2Y}{L - Y}\ . \qquad (2.2)$$

Значення *Y* визначається як рішення наступного рівняння

$$Y^2 - 3LY + L^2 = 0, \qquad (2.3)$$

$$\frac{Y}{L} = \frac{3 \pm \sqrt{5}}{2}.$$

З двох коренів рівняння залишаємо

$$\frac{Y}{L} = \frac{3 - \sqrt{5}}{2} \approx 0.38.$$

оскільки другий корінь більше 1, що не відповідає постановці задачі. Знайдене ірраціональне співвідношення називають «золотим перетином».

Вводять також позначення z = (3 − √5)/2.

Алгоритм методу «золотого перетину»

Вихідні дані: A – ліва межа зони пошуку; *B* – права межа зони пошуку; Δ - точність локалізації екстремуму; функція f(x) унімодальна на [A, B].

Крок 1. Присвоїти $L = B - A$; $X_1 = A$; $X_4 = B$.

Крок 2. Присвоїти

$$z = \frac{3 - \sqrt{5}}{2}; \qquad X_2 = X_1 + zL; \qquad X_3 = X_4 - zL.$$

Обчислити $f(X_2)$ і $f(X_3)$.

Крок 3. Вибрати підінтервал, в якому локалізований мінімум.

Якщо $f(X_2) \le f(X_3)$ то $X_4 = X_3$, $X_3 = X_2$, $X_2 = X_1 + (X_4 - X_3)$ і обчислити $f(X_2)$

Якщо $f(X_2) > f(X_3)$ то $X_1 = X_2$; $X_2 = X_3$; $X_3 = X_1 + (X_4 - X_2)$ і обчислити $f(X_3)$.

Крок 4. Обчислити довжину поточного інтервалу L = X4 − X1. Якщо L ≤ 2Δ, завершити пошук і прийняти x* = (X1 + X4)/2. Якщо L > 2Δ, повернутися до кроку 3.

Точність визначення екстремуму методом «золотого перетину» при заданому числі обчислень значень функції f(x) можна оцінити через довжину інтервалу локалізації. Після S обчислень функції абсолютна похибка визначення положення екстремуму становить

$$\Delta = \frac{B - A}{2}\left(\frac{\sqrt{5}-1}{2}\right)^{S-3}, \qquad (2.4)$$

де Δ – абсолютна похибка у визначенні положення екстремуму після S обчислень функції.

2.2.3. Метод квадратичної апроксимації

Метод квадратичної інтерполяції полягає в тому, що на поточному інтервалі функцію f(x) замінюють параболою, побудованою за трьома різними точками, після чого мінімум наближувальної параболи знаходять аналітично за координатою її вершини.

Нехай задані три точки α, β, γ та значення функції в них f_α, f_β, f_γ. Тоді f(x) апроксимують поліномом другого степеня

$$\varphi(x) = Ax^2 + Bx + C, \qquad (2.5)$$

Коефіцієнти A, B, C визначаються з системи рівнянь

$$\begin{aligned} A\alpha^2 + B\alpha + C &= f_\alpha, \\ A\beta^2 + B\beta + C &= f_\beta, \\ A\gamma^2 + B\gamma + C &= f_\gamma. \end{aligned} \qquad (2.6)$$

Після перетворень цієї системи отримують формули для розрахунку коефіцієнтів параболи.

$$A = \left[\frac{(\gamma-\beta) f_\alpha + (\alpha-\gamma) f_\beta + (\beta-\alpha) f_\gamma}{\Delta} \right]$$

$$B = \left[\frac{(\beta^2-\gamma^2) f_\alpha + (\gamma^2-\alpha^2) f_\beta + (\alpha^2-\beta^2) f_\gamma}{\Delta} \right] \quad , \qquad (2.7)$$

$$C = \left[\frac{\beta\gamma(\gamma-\beta) f_\alpha + \gamma\alpha(\alpha-\gamma) f_\beta + \alpha\beta(\beta-\alpha) f_\gamma}{\Delta} \right]$$

де $\Delta = (\alpha-\beta)(\beta-\gamma)(\gamma-\alpha)$

Функція $\varphi(x)$ матиме мінімум в точці $x = -B/2A$, якщо $A > 0$. Отже, мінімум полінома знаходиться в точці

$$\delta = \frac{1}{2}\left[\frac{(\beta^2-\gamma^2) f_\alpha + (\gamma^2-\alpha^2) f_\beta + (\alpha^2-\beta^2) f}{(\beta-\gamma) f_\alpha + (\gamma-\alpha) f_\beta + (\alpha-\beta) f_\gamma} \right]. \qquad (2.8)$$

Алгоритм методу квадратичної апроксимації

Нехай задані унімодальна функція однієї змінної f(x), початкова апроксимація положення мінімуму A та довжина кроку H.

Крок 1. Обчислити $f(A)$ и $f(A + H)$

Крок 2. Якщо $f(A) < f(A + H)$, то взяти третьою точкою $A - H$ і обчислити $f(A - H)$. Інакше взяти третьою точкою $A + 2H$ і знайти $f(A + 2H)$.

Крок 3. Використовуючи три точки, обчислити δ з формулою (2.8) та знайти $f(\delta)$.

Крок 4. Якщо різниця між найменшим значенням функції та наступним найменшим значенням функції є меншою за задану точність, процедуру завершити.

Крок 5. Якщо процедура не завершена, відкинути точку з найбільшим значенням функції та повернутися до кроку 3.

2.3 Безградієнтні методи пошуку екстремуму функції однієї змінної у *MATLAB*

Для мінімізації безперервної функції однієї змінної на заданому інтервалі в *MATLAB* застосовується функція fminbnd. Задача мінімізації має вигляд min f(x) за умови a < x < b. Алгоритм fminbnd поєднує кроки методу «золотого перетину» та кроки параболічної, тобто квадратичної інтерполяції.

Синтаксис функції fminbnd:

x = fminbnd(myfun, a, b)

x = fminbnd(myfun, a, b, options)

[*x, fval*] = *fminbnd(...)*

[*x, fval, exitflag*] = *fminbnd(...)*

[*x, fval, exitflag, output*] = *fminbnd(...)*

У наведених записах прийняті такі позначення:

x – значення аргументу, що відповідає знайденому локальному мінімуму функції f(x);

fval – значення функції f(x) у точці x;

exitflag – ознака завершення алгоритму;

output – структура, що містить інформацію про процес (*iterations, funcCount, algorithm, message*);

myfun – цільова функція;

a, b – ліва та права межі інтервалу;

options – параметри алгоритму, що задаються через *optimset*.

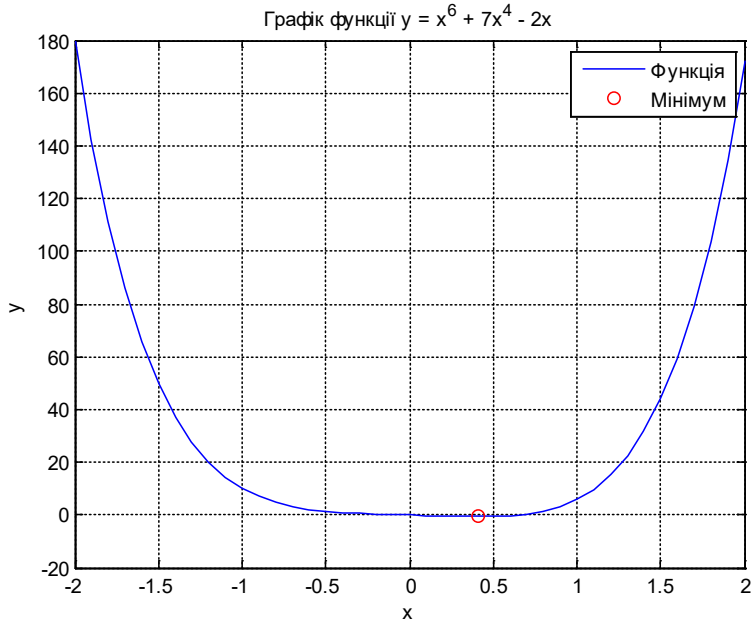
Приклад мінімізації у MATLAB з побудовою графіка.

Наведений приклад демонструє застосування fminbnd, виведення підсумкових характеристик та відображення точки мінімуму на графіку.

$$y = x^6 + 7x^4 - 2x, \quad x \in [-2, 2]$$

```
clear; clc; close all;
% Цільова функція
myfun = @(x) x.^6 + 7*x.^4 - 2*x;
```

```
% Межі пошуку
a = -2;
b =  2;
% Опції (виведення ітерацій та точність за аргументом)
options = optimset('Display','iter','TolX',1e-4);
% Пошук мінімуму
[x_min, fval, exitflag, output] = fminbnd(myfun, a, b,
options);
% Виведення результатів
fprintf('x_min = %.6f\n', x_min);
fprintf('f(x_min) = %.6f\n', fval);
fprintf('exitflag = %d\n', exitflag);
fprintf('iterations = %d\n', output.iterations);
fprintf('funcCount = %d\n', output.funcCount);
fprintf('algorithm = %s\n', output.algorithm);
% Побудова графіка і позначення мінімуму
x = linspace(a, b, 1000);
y = myfun(x);
figure;
plot(x, y);
grid on;
hold on;
plot(x_min, fval, 'ro');
title('Графік f(x) та точка мінімуму');
xlabel('x');
ylabel('f(x)');
legend('f(x)', 'мінімум');
hold off;
```




```
>> LAB2OPT
```

```
 Func-count      x          f(x)          Procedure
     1      -0.472136      1.30318        initial
     2       0.472136    -0.585366        golden
     3        1.05573      7.96884        golden
     4      0.0917203    -0.182945        parabolic
     5       0.598476    -0.252985        parabolic
     6       0.354592     -0.59653        parabolic
     7       0.408797    -0.617435        parabolic
     8       0.407106    -0.617382        parabolic
     9       0.410262    -0.617446        parabolic
    10       0.409929    -0.617447        parabolic
    11       0.409595    -0.617445        parabolic

Optimization terminated:
 the current x satisfies the termination criteria
using OPTIONS.TolX of 1.000000e-04

x_min = 0.410049
f(x_min) = -0.617447
exitflag = 1
iterations = 11
funcCount = 12
algorithm = golden section search, parabolic
interpolation
```

2.4. Порядок виконання роботи

2.4.1 Записати цільову функцію *f(x)* та межі інтервалу [*A, B*] відповідно до варіанта з таблиці 2.1.

2.4.2 Обрати значення точності Δ та зазначити його у звіті.

2.4.3 Виконати два цикли звуження інтервалу методом п’яти точок за алгоритмом п. 2.2.1.

2.4.4 Для кожного з двох циклів записати пробні точки X_1–X_5 та відповідні значення $f(X_1)$–$f(X_5)$.

2.4.5 Виконати мінімізацію заданої функції у MATLAB за допомогою fminbnd на інтервалі [A, B] з обраною точністю Δ.

2.4.6 Зафіксувати отримані значення *x_min* і *f(x_min)*, а також значення iterations, funcCount і algorithm зі структури output.

2.4.7 Реалізувати метод п’яти точок у *MATLAB* у вигляді окремого файла *five_points_method.m* та виконати пошук мінімуму на інтервалі [*A, B*] з точністю Δ.

2.4.8 Зафіксувати для методу п’яти точок значення *x_min* і *f(x_min)*, а також кількість обчислень цільової функції *funcCount*, отриману зі скрипта.

2.4.9 Побудувати графік *f(x)* на інтервалі [*A, B*] та позначити на ньому знайдену точку мінімуму.

2.4.10 Оформити звіт відповідно до вимог розділу 2.5, навівши результати для fminbnd і для реалізації методу п’яти точок, а у висновку коротко порівняти значення *x_min* і *f(x_min)* та кількість обчислень цільової функції funcCount для обох підходів за однакового значення точності Δ.

Скрипт методу п’яти точок для реалізації в *MATLAB*

```
% ЛР2. Виклик five_points_method
clear; clc; close all;
 % Цільова функція
myfun = @(x) x.^6 + 7*x.^4 - 2*x;
% Межі та точність
A = -2;
B =  2;
Delta = 1e-4;
% Виклик методу п'яти точок
[xmin5, fmin5, S5] = five_points_method(myfun, A, B,
Delta);
% Виведення результатів
fprintf('Метод п''яти точок:\n');
fprintf('xmin = %.6f\n', xmin5);
fprintf('f(xmin) = %.6f\n', fmin5);
fprintf('Число обчислень f(x): %d\n', S5);
% Графік
x = linspace(A, B, 1000);
y = myfun(x);
figure;
plot(x, y);
grid on;
hold on;
```

```
plot(xmin5, fmin5, 'ro');
title('Графік f(x) та точка мінімуму (метод п''яти
точок)');
xlabel('x');
ylabel('f(x)');
legend('f(x)', 'мінімум');
hold off;

function [xmin, fmin, funcCount] = five_points_method(f, A,
B, Delta)
% Метод п'яти точок
funcCount = 0;

while (B - A) > 2*Delta
    C  = (B - A)/4;
    X1 = A;
    X2 = A + C;
    X3 = A + 2*C;
    X4 = A + 3*C;
    X5 = B;

    F1 = f(X1); F2 = f(X2); F3 = f(X3); F4 = f(X4); F5 =
f(X5);
    funcCount = funcCount + 5;

    [~, idx] = min([F1 F2 F3 F4 F5]);

    if idx == 3
        A = X2; B = X4;
    elseif (idx == 1) || (idx == 2)
        A = X1; B = X3;
    else
        A = X3; B = X5;
    end
end

xmin = (A + B)/2;
fmin = f(xmin);
funcCount = funcCount + 1;
end
```

2.5. Вимоги до оформлення звіту

Звіт подається у друкованому або електронному вигляді та має містити назву роботи, мету, вихідні дані варіанта, стислий опис застосованих методів і результати розрахунків. У звіті наводять задану цільову функцію *f(x)*, межі

інтервалу [*A, B*], вибране значення точності Δ, а також отримані значення *x_min* і *f(x_min)* для обох способів розв'язання, за допомогою fminbnd і за допомогою реалізації методу п'яти точок. Для fminbnd додатково зазначають значення *exitflag* і характеристики процесу пошуку зі структури output у вигляді, в якому їх подає *MATLAB*, а саме *iterations*, *funcCount*, *algorithm* і, за потреби, *message*. Для методу п'яти точок у звіті наводять значення *x_min* і *f(x_min)*, а також кількість обчислень цільової функції *funcCount*, отриману зі скрипта. У звіт включають графік *f(x)* на інтервалі пошуку з позначенням знайденої точки мінімуму. За наявності відмінностей у значеннях *x_min* у тексті звіту зазначають досягнуту точність локалізації через задане значення Δ або через довжину фінального інтервалу локалізації.

Контрольні питання

1. За яких умов методи виключення інтервалів можна застосовувати для пошуку мінімуму функції на відрізку [*A, B*].
2. У чому полягає принцип звуження інтервалу в методі п'яти точок.
3. Яка особливість вибору внутрішніх точок у методі «золотого перетину» та як вона впливає на обчислення *f(x)* на наступних кроках.
4. У чому полягає ідея параболічної інтерполяції в одномірній оптимізації.
5. Які дані повертає функція fminbnd і яке призначення змінних *x, fval, exitflag* та структури *output*.
6. Які характеристики процесу пошуку доцільно використовувати для аналізу виконання алгоритму в *MATLAB* (*iterations, funcCount*) і як їх отримати.

Таблиця 2.1. Функції, що досліджуються.

	цільова функція	обмеження
1	$3x^4 + (x - 1)^2$	$0 \le x \le 4$
2	$3x^2 + (12/x^3) - 5$	$0,5 \le x \le 2,5$
3	$(10x^3 + 3x^2 + x + 5)^2$	$-2 \le x \le 1$
4	$x^3 + x$	$-2 \le x \le 2$
5	$x^4 + x^2$	$-1 \le x \le 1$
6	$4x^4 - x^2 + 5$	$0 \le x \le 2$
7	$(3x - 2)^2 (2x - 3)^2$	$-1 \le x \le 1$
8	$6x^5 - 4x^3 + 10$	$0 \le x \le 1$
9	$-0,1x^3 + 2x^2 - 10x$	$2 \le x \le 5$

ЛАБОРАТОРНА РОБОТА № 3

МЕТОД НЕЛДЕРА-МІДА ДЛЯ МІНІМІЗАЦІЇ ФУНКЦІЇ БАГАТЬОХ ЗМІННИХ

Мета роботи: ознайомитися з методом Нелдера–Міда та набути навичок його застосування в *MATLAB* для мінімізації функцій багатьох змінних.

3.1. Загальні положення

Метод Нелдера–Міда належить до безградієнтних методів прямого пошуку і призначений для мінімізації функції кількох змінних без обчислення похідних. Пошук здійснюється шляхом послідовної перебудови симплекса, тобто множини пробних точок. У двовимірному випадку симплексом є трикутник, вершини якого задають три пробні точки. На кожній ітерації вершини впорядковують за значенням цільової функції, після чого найгіршу вершину замінюють новою точкою, сформованою однією з операцій алгоритму. Процес триває до виконання умови зупинки, що пов’язана зі зменшенням розмірів симплекса та стабілізацією значень функції. Оскільки метод не використовує похідних, результат є локальним і залежить від стартових даних, зокрема від масштабування змінних та вибору початкового симплекса. Це слід враховувати під час інтерпретації отриманого мінімуму і під час порівняння результатів для різних початкових точок та різних розмірів симплекса.

3.2. Опис методу розрахунку

Нехай задано цільову функцію *f(u)*, де *u* = [*x, y*]. Початковий симплекс у двовимірному випадку задається трьома вершинами *u1, u2, u3*. На кожній

ітерації вершини впорядковують так, щоб $f(u1) \le f(u2) \le f(u3)$, де $u1$ є найкращою вершиною, $u3$ є найгіршою.

Позначимо центр ваги двох найкращих вершин як

$$u0 = (u1 + u2)/2.$$

Далі виконують одну з типових операцій: відбиття, розтягнення, стиснення або редукцію (зменшення) симплекса.

Алгоритм методу Нелдера–Міда

Вихідні дані: $f(u)$, початковий симплекс $u1, u2, u3$; коефіцієнти α, γ, ρ, σ; точність Δ; максимальна кількість ітерацій $Nmax$.

3.2.1.1 Впорядкувати вершини симплекса так, щоб $f(u1) \le f(u2) \le f(u3)$.

3.2.1.2 Обчислити

$$u0 = (u1 + u2)/2.$$

3.2.1.3 Виконати відбиття найгіршої вершини:

$$ur = u0 + \alpha(u0 - u3).$$

3.2.1.4 Якщо f(ur) < f(u1), виконати розтягнення:

$$ue = u0 + \gamma(ur - u0).$$

Прийняти як нову вершину $u3$ точку ue, якщо $f(ue) < f(ur)$, інакше прийняти ur.

3.2.1.5 Якщо $f(u1) \le f(ur) < f(u2)$, прийняти ur замість $u3$.

3.2.1.6 Якщо $f(u2) \le f(ur) < f(u3)$, виконати зовнішнє стиснення:

$$uc = u0 + \rho(ur - u0).$$

Якщо f(uc) ≤ f(ur), прийняти uc замість u3, інакше перейти до редукції.

3.2.1.7 Якщо f(ur) ≥ f(u3), виконати внутрішнє стиснення:

$$uc = u0 - \rho(u0 - u3).$$

Якщо $f(uc) < f(u3)$, прийняти uc замість u3, інакше перейти до редукції.

3.2.1.8 Редукція: $u2 = u1 + \sigma(u2 - u1)$, $u3 = u1 + \sigma(u3 - u1)$.

3.2.1.9 Перевірити умову зупинки. Якщо максимальна довжина ребра симплекса $dV \leq \Delta$ або кількість ітерацій досягла *Nmax*, завершити пошук. Інакше повернутися до п. 3.2.1.1.

Рекомендовані значення коефіцієнтів для двовимірного випадку: α = 1 (відбиття), γ = 2 (розтягнення), ρ = 0,5 (стиснення), σ = 0,5 (редукція).

3.3. Реалізація методу в *MATLAB*

Для мінімізації функцій багатьох змінних без використання похідних у MATLAB застосовується функція fminsearch, яка реалізує метод Нелдера–Міда. Функція виконує пошук локального мінімуму, що залежить від початкового наближення x0 та параметрів у options. Цільова функція задається як function handle fun, який приймає вектор змінних x.

Синтаксис *fminsearch*:

```
x = fminsearch(fun, x0)
x = fminsearch(fun, x0, options)
[x, fval] = fminsearch(...)
[x, fval, exitflag] = fminsearch(...)
[x, fval, exitflag, output] = fminsearch(...)
```

У наведених записах прийняті такі позначення:

x – вектор змінних, що відповідає знайденому локальному мінімуму;

fval – значення цільової функції у точці *x*;

exitflag – ознака завершення алгоритму;

output – структура з інформацією про процес (*iterations, funcCount, algorithm, message*);

fun – цільова функція, задана як *function handle*;

x0 – початкове наближення;

options – параметри алгоритму, що задаються через optimset.

Параметри *options* задаються за допомогою *optimset*.

Для контролю процесу пошуку використовують такі параметри:

Display для виведення інформації у командне вікно;

TolX для точності за аргументом;

TolFun для точності за значенням функції;

MaxIter для обмеження числа ітерацій;

MaxFunEvals для обмеження числа обчислень цільової функції. Наприклад, команда

options = optimset('Display','iter','TolX',1e-6,'TolFun',1e-6)

забезпечує виведення результатів по ітераціях та задає точність завершення.

Цільову функцію fun задають у вигляді анонімної функції або окремої *m*-функції. Для двовимірного випадку зручно задавати *fun* як функцію від вектора *u*, наприклад

fun = @(u) (1-u(1))^2 + 100(u(2)-u(1)^2)^2.*

Приклад застосування *fminsearch* для функції Розенброка:

```
clear; clc;
fun = @(u) (1 - u(1))^2 + 100*(u(2) - u(1)^2)^2;
x0  = [-1.2; 1.0];
options = optimset('Display','iter','TolX',1e-6,'TolFun',1e-6);
[x, fval, exitflag, output] = fminsearch(fun, x0, options);
fprintf('x* = %.6f  %.6f\n', x(1), x(2));
fprintf('fval = %.12e\n', fval);
fprintf('exitflag = %d\n', exitflag);
```

```
fprintf('iterations = %d\n', output.iterations);
fprintf('funcCount = %d\n', output.funcCount);
fprintf('algorithm = %s\n', output.algorithm);
fprintf('message: %s\n', output.message);
```

В додатку наведено варіант застосування *fminsearch* з візуалізацією переміщення симплекса на полі рівнів функції та спостерігати траєкторію пошуку.

3.4. Порядок виконання роботи

3.4.1 Обрати варіант цільової функції відповідно до п. 3.5 та задати межі області побудови графіка для ліній рівня.

3.4.2 Задати початкову точку u0 = [x0; y0] для запуску алгоритму.

3.4.3 Виконати мінімізацію обраної функції стандартними засобами MATLAB за допомогою fminsearch, задавши options та зафіксувавши отримані результати.

3.4.4 Запустити демонстраційний скрипт з візуалізацією переміщення симплекса на полі рівнів функції та спостерігати траєкторію пошуку. У звіті навести отриманий графік та коротко описати, як змінювався симплекс у процесі наближення до мінімуму.

3.4.5 Для кожного запуску зафіксувати значення x, fval, exitflag, а також значення iterations, funcCount, algorithm і, за потреби, message зі структури output. Інтерпретацію причини зупинки подати за текстом output.message.

3.4.6 Повторити розрахунок для іншої початкової точки u0 та інших розмірів початкового симплекса, якщо це передбачено демонстраційним скриптом, і зафіксувати ті самі показники.

3.4.7 Порівняти результати двох запусків за значенням fval, кількістю ітерацій iterations та кількістю обчислень funcCount і зробити висновок про вплив стартових умов на збіжність і на отриманий локальний мінімум.

3.5 Варіанти функцій для дослідження

Варіант 1: квадратична функція f(x,y) = 4x^2 + y^2 − 4x + 2y, область побудови x ∈ [−5; 5], y ∈ [−5; 5]. Точний мінімум у точці (0,5; −1).

Варіант 2: квадратична функція f(x,y) = x^2 + 2y^2 + xy, область побудови x ∈ [−5; 5], y ∈ [−5; 5]. Точний мінімум у точці (0; 0).

Варіант 3: квадратична функція f(x,y) = 3x^2 + 2xy + 4y^2 − 2x − 8y, область побудови x ∈ [−4; 4], y ∈ [−4; 4]. Точний мінімум у точці (0; 1).

Варіант 4: квадратична функція f(x,y) = 2x^2 + 3y^2 + 2xy − 6x + 4y, область побудови x ∈ [−6; 6], y ∈ [−6; 6]. Точний мінімум у точці (11/5; −7/5), тобто (2,2; −1,4).

Варіант 5: квадратична функція f(x,y) = 5x^2 + 2y^2 − 2xy + 8x − 6y, область побудови x ∈ [−6; 6], y ∈ [−6; 6]. Точний мінімум у точці (−5/9; 11/9), тобто приблизно (−0,5556; 1,2222).

Варіант 6: функція Бута f(x,y) = (x + 2y − 7)^2 + (2x + y − 5)^2, область побудови x ∈ [−10; 10], y ∈ [−10; 10]. Точний мінімум у точці (1; 3).

Варіант 7: функція Розенброка f(x,y) = (1 − x)^2 + 100(y − x^2)^2, область побудови x ∈ [−2; 2], y ∈ [−1; 3]. Точний мінімум у точці (1; 1). Для цього рельєфу збільшення числа ітерацій часто спостерігається при старті в точках із від’ємним x та відносно великим y, наприклад x0 = (−2; 3).

Варіант 8: функція Біля f(x,y) = (1,5 − x + xy)^2 + (2,25 − x + xy^2)^2 + (2,625 − x + xy^3)^2, область побудови x ∈ [−4,5; 4,5], y ∈ [−4,5; 4,5]. Точний мінімум у точці (3; 0,5). Зростання числа ітерацій часто спостерігається при старті поблизу кутів області, наприклад x0 = (−4,5; 4,5).

Варіант 9: функція Гіммельблау f(x,y) = (x^2 + y − 11)^2 + (x + y^2 − 7)^2, область побудови x ∈ [−6; 6], y ∈ [−6; 6]. Функція має кілька глобальних мінімумів у точках (3; 2), (−2,805118; 3,131312), (−3,779310; −3,283186), (3,584428; −1,848126). Старт у центральній зоні, наприклад x0 = (0; 0), може приводити до різних мінімумів залежно від траєкторії пошуку.

3.6. Вимоги до оформлення звіту

Звіт подається у друкованому або електронному вигляді та має містити назву роботи, мету, вихідні дані обраного варіанта з п. 3.5, а також стислий опис виконаних дій у MATLAB. У звіті обов'язково наводять запис цільової функції у вигляді function handle fun, початкове наближення x0 у вигляді вектора, задані параметри options, а також межі області побудови графіка для ліній рівня, якщо використовувалась візуалізація. Для кожного запуску необхідно зафіксувати результати оптимізації: x, fval, exitflag і поля структури output у тому вигляді, як їх повертає MATLAB, а саме iterations, funcCount, algorithm, message. Якщо виконувалась візуалізація, до звіту включають графік ліній рівня з траєкторією переміщення симплекса та короткий опис характеру збіжності. У висновку слід порівняти результати двох запусків за значенням fval, кількістю ітерацій iterations та кількістю обчислень funcCount і зробити висновок щодо впливу стартових умов на збіжність і на отриманий локальний мінімум.

Контрольні питання

1. За яких умов доцільно застосовувати метод Нелдера–Міда і чому він належить до безградієнтних методів.
2. Що називають симплексом у двовимірній задачі та як вибір початкового симплекса впливає на результат.
3. Яке призначення операцій відбиття, розтягнення, стиснення та редукції в алгоритмі Нелдера–Міда.
4. Чому результат методу Нелдера–Міда є локальним і як це проявляється при зміні початкової точки x0.
5. Які дані повертає функція fminsearch і що означають змінні x, fval, exitflag та структура output.
6. Які поля output доцільно використовувати для аналізу збіжності та як за output.message сформулювати причину зупинки алгоритму.
7. Як впливають параметри TolX, TolFun, MaxIter, MaxFunEvals на завершення пошуку та на відтворюваність результатів.

ЛАБОРАТОРНА РОБОТА № 4
ГРАДІЄНТНІ МЕТОДИ БЕЗУМОВНОЇ ОПТИМІЗАЦІЇ В MATLAB

Мета роботи: набути практичних навичок застосування градієнтних методів безумовної оптимізації для функцій багатьох змінних у середовищі *MATLAB* з використанням функції *fminunc*.

4.1. Загальні положення

У лабораторній роботі розглядаються градієнтні методи безумовної оптимізації, реалізовані у функції fminunc. Спільною рисою цих методів є використання інформації про градієнт цільової функції для визначення напрямку пошуку мінімуму. Відмінності між методами полягають у способі формування напрямку спуску та використанні інформації про кривизну функції.

Метод найшвидшого спуску є базовим градієнтним методом. На кожній ітерації напрямок пошуку вибирається протилежним до напрямку градієнта цільової функції в поточній точці. Довжина кроку визначається окремою процедурою одномірної мінімізації. Метод простий за реалізацією, але чутливий до форми рельєфу функції. У задачах з витягнутими рівнями або яровою структурою може збігатися повільно.

Метод DFP (Девідона – Флетчера – Пауелла) належить до квазіньютонівських методів. Його ідея полягає в побудові наближення до оберненої матриці гесіана на основі змін градієнта між ітераціями. Це дозволяє враховувати локальну кривизну цільової функції без явного обчислення гесіана. У порівнянні з методом найшвидшого спуску DFP зазвичай забезпечує швидшу збіжність, проте може бути менш стійким, ніж сучасні квазіньютонівські схеми.

*Метод BFGS (Бройдена – Флетчера – Голдфарба – Шенно)*також є квазіньютонівським і вважається одним із найбільш ефективних практичних методів безумовної оптимізації. Як і DFP, він використовує інформацію про

зміну градієнта, проте застосовує іншу формулу оновлення наближення гесіана, що забезпечує кращу чисельну стійкість. Саме тому схема BFGS використовується у *fminunc* як метод за замовчуванням.

4.2. Функція *fminunc* та її призначення

Наведені в п. 4.1 методи реалізуються у MATLAB засобами функції *fminunc*, яка призначена для безумовної мінімізації диференційовних функцій багатьох змінних і виконує пошук локального мінімуму. Практичний результат мінімізації визначається вибором початкового наближення $x0$ та налаштуваннями алгоритму.

У *fminunc* передбачено два режими роботи: *quasi-newton* (квазіньютонівський метод) та *trust-region* (метод довірчої області). У режимі *quasi-newton* (квазіньютонівський метод) *fminunc* використовує наближення гесіана; типовою схемою є *BFGS* (метод Бройдена – Флетчера – Голдфарба – Шенно). За потреби можна застосувати іншу схему оновлення, зокрема *DFP* (метод Девідона – Флетчера – Пауелла). Окремо у межах лабораторної роботи розглядається метод найшвидшого спуску як базовий градієнтний підхід для порівняння з квазіньютонівськими схемами.

Функція *fminunc* підтримує такі варіанти виклику:

```
x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
[x,fval] = fminunc(...)
[x,fval,exitflag,output] = fminunc(...)
[x,fval,exitflag,output,grad,hessian] = fminunc(...)
```

де `x` і `fval` є відповідно вектором знайдених значень змінних та значенням цільової функції у точці x.

Параметр `exitflag` є числовим кодом причини завершення алгоритму; для коректної інтерпретації результату у звіті необхідно наводити

exitflag разом із текстом output.message, оскільки саме повідомлення output.message однозначно пояснює, який критерій зупинки спрацював.

Структура output містить службову інформацію про перебіг мінімізації, зокрема:

iterations (кількість ітерацій),

funcCount (кількість обчислень цільової функції),

firstorderopt (показник оптимальності першого порядку),

algorithm (назва використаного алгоритму),

stepsize (кінцеве зміщення за змінними x)

message (повідомлення про завершення).

Залежно від обраного алгоритму можуть бути доступні додаткові поля, наприклад cgiterations.

Вихідні аргументи grad і hessian відповідають градієнту та гесіану цільової функції в точці знайденого розв'язку.

Вхідні аргументи задаються так:

fun є цільовою функцією і може повертати лише її значення або, за потреби, також градієнт і гесіан;

x0 є початковою точкою, що задається користувачем; options містить налаштування параметрів алгоритму, точності зупинки та обмежень на кількість ітерацій і обчислень.

Для вибору конкретного градієнтного методу у функції fminunc використовується задання параметрів алгоритму через структуру options. Залежно від обраної схеми оновлення наближення гесіана або способу формування напрямку пошуку можуть застосовуватися такі варіанти задання методу:

Квазіньютонівський метод *BFGS* (Бройдена – Флетчера – Голдфарба – Шенно), який є типовим варіантом і забезпечує добру збіжність для широкого класу задач.

```
opt_bfgs = optimset(opt_common,'HessUpdate','bfgs');
```

Квазіньютонівський метод *DFP (*Девідона – Флетчера – Пауелла), що використовує альтернативну формулу оновлення наближення гесіана.

```
opt_dfp = optimset(opt_common,'HessUpdate','dfp');
```

Метод найшвидшого спуску, у якому напрямок пошуку визначається лише за градієнтом цільової функції без урахування інформації про її кривизну.

```
 opt_sd =optimset(opt_common,'HessUpdate','steepdesc');
```

4.3. Порядок виконання лабораторної роботи

4.3.1 Обрати цільову функцію f(x) відповідно до свого варіанта з таблиці та задати початкову точку x0.

4.3.2 Виконати мінімізацію функції за допомогою fminunc у режимі quasi-newton з типовою схемою BFGS.

4.3.3 Повторити мінімізацію для тієї самої функції та того самого x0 зі схемою DFP.

4.4.4 Повторити мінімізацію для тієї самої функції та того самого x0 методом найшвидшого спуску.

4.4.5 Для кожного запуску зчитати x, fval, exitflag і output.message, а також iterations, funcCount і firstorderopt зі структури output.

4.4.6 Порівняти результати трьох варіантів за значенням fval, кількістю ітерацій та кількістю обчислень функції, зробити висновок щодо відмінностей збіжності.

4.4. Приклад виконання лабораторної роботи

Розглянемо задачу мінімізації функції Розенброка

$$f(x) = (1 - x1)^2 + 100(x2 - x1^2)^2,$$

яка має глобальний мінімум у точці (1; 1) і характеризується яровою структурою. У цьому прикладі мінімізація виконується в режимі quasi-newton з line search. Для аналізу результатів у звіті необхідно навести знайдені x і fval, а також exitflag і поля структури output, зокрема iterations, funcCount, firstorderopt, algorithm, message. Причину зупинки слід формулювати за текстом output.message, оскільки саме він однозначно пояснює, який критерій зупинки спрацював.

Скрипт запуску Lab4_rosenbrock_fminunc.m

Для виконання прикладу використовуються чотири окремі MATLAB-файли, що мають знаходитися в одному робочому каталозі.

Файл rozen.m (цільова функція)

```
function f = rozen(x)
f = 100*(x(2) - x(1)^2)^2 + (x(1) - 1)^2;
end
```

Файл rozen_g.m (градієнт цільової функції)

```
function g = rozen_g(x)
g = [ -400*x(1)*(x(2)-x(1)^2) + 2*(x(1)-1);
       200*(x(2)-x(1)^2) ];
end
```

Файл outfun_traj.m (збереження траєкторії ітерацій)

```
function stop = outfun_traj(x,optimValues,state)
stop = false;
global x_hist
if isequal(state,'iter')
    x_hist = [x_hist; x'];
end
end
```

Файл *run_fminunc.m* (головний файл запуску)

```
clear, clc, close all
global x_hist
x_hist = [];
x0 = [-3; 10];   % початкова точка
opt_common = optimset( ...
    'GradObj','on', ...
    'LargeScale','off', ...
    'OutputFcn',@outfun_traj);
opt = optimset(opt_common,'HessUpdate','bfgs');
% [xmin,fmin] = fminunc(@(x) deal(rozen(x),rozen_g(x)), x0,
opt);
[Xopt,fopt,exitflag,output] = fminunc(@(x)
deal(rozen(x),rozen_g(x)), x0, opt);
[x1,x2] = meshgrid(-2:0.05:2,-1:0.05:3);
z = 100*(x2 - x1.^2).^2 + (x1 - 1).^2;
figure
contour(x1,x2,z,30), grid on, hold on
plot(x_hist(:,1),x_hist(:,2),'r-
o','MarkerSize',9,'LineWidth',1.5)
% plot(1,1,'bo','MarkerSize',8,'LineWidth',1)
xlabel('x_1'), ylabel('x_2')
title('Лінії рівня та траєкторія збіжності')
hold off
history.x = x_hist;
history.fval = [];
disp('Xopt =')
disp(Xopt.')
disp('fopt =')
disp(fopt)
disp(['iterations: ', num2str(output.iterations)])
disp(['funcCount: ', num2str(output.funcCount)])
disp(['firstorderopt: ', num2str(output.firstorderopt)])
disp(['algorithm: ''', output.algorithm, ''''])
disp(['message: ''', output.message, ''''])
disp('history =')
disp(['   x: [', num2str(size(history.x,1)), 'x',
num2str(size(history.x,2)), ' double]'])
disp('   fval: []')
```

В результаті розрахунку отримуемо такі дані та графік

```
Local minimum found.

Optimization completed because the size of the gradient is less
than the default value of the function tolerance.

<stopping criteria details>

Xopt =     1.0000    0.9999

fopt =    1.4943e-09
```

```
iterations: 54
funcCount: 69
firstorderopt: 0.0011675
algorithm: 'medium-scale: Quasi-Newton line search'
message: 'Local minimum found.

Optimization completed because the size of the gradient is less
than
the default value of the function tolerance.

Stopping criteria details:

Optimization completed: The first-order optimality measure,
9.786058e-07, is less than options.

TolFun = 1.000000e-06.

Optimization Metric
Options
relative norm(gradient) =   9.79e-07
TolFun =   1e-06 (default)'
history =
    x: [55x2 double]
    fval: []
```

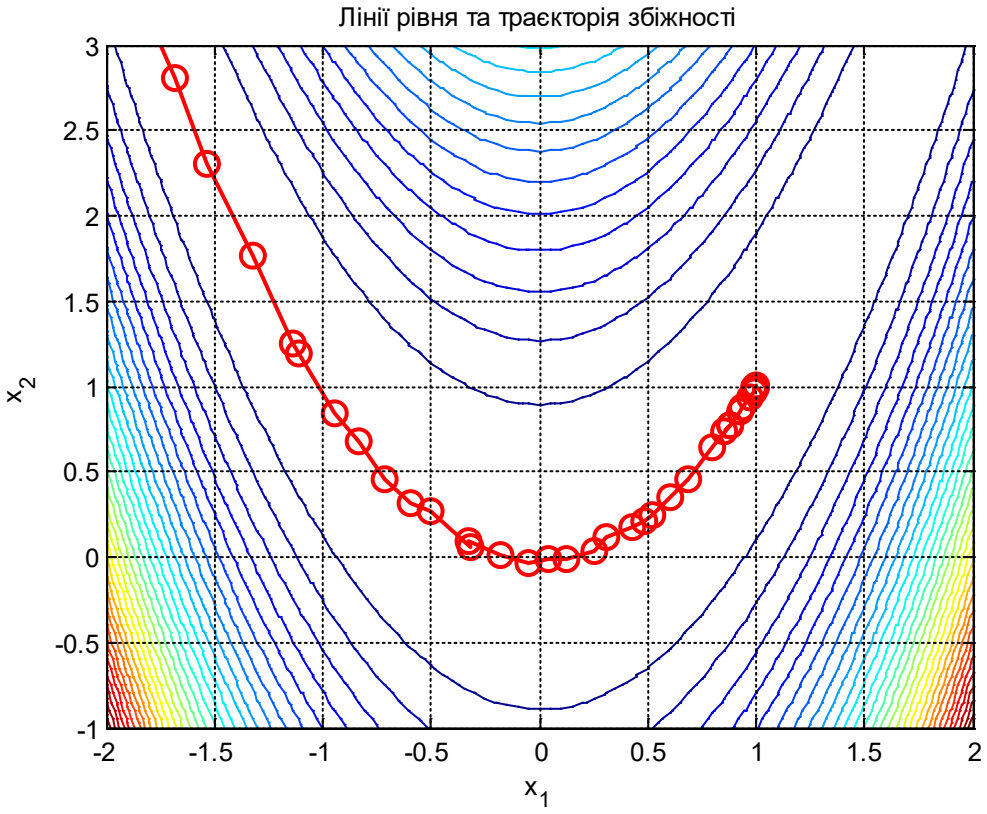


Рис. 4.1. Метод BFGS

4.5. Вимоги до оформлення звіту

Звіт подається у друкованому або електронному вигляді та має містити назву роботи, мету, вихідні дані задачі й результати розрахунків. У вихідних даних необхідно навести цільову функцію f(x) для свого варіанта, початкову точку x0 та вказати, чи задавався градієнт (параметр GradObj: on або off). Для кожного з трьох запусків (BFGS, DFP, найшвидший спуск) у звіті обов’язково наводять знайдені значення x та fval, значення exitflag, а також поля структури output: iterations, funcCount, firstorderopt, algorithm, message.

Причину зупинки алгоритму слід пояснювати за текстом output.message. У підсумковому аналізі потрібно порівняти три варіанти за fval, iterations і funcCount (за однакових налаштувань точності), а також зробити висновок щодо відмінностей збіжності для квазіньютонівських схем і методу найшвидшого спуску.

Приклад оформлення фрагмента звіту (для підстановки своїх значень)

Вихідні дані: f(x) = ...
Початкова точка: x0 = [.....; .....]
Налаштування: GradObj = on/off; LargeScale = off; TolX = ...; TolFun = ...

Метод: BFGS (HessUpdate = 'bfgs')
x = [..... .....]
fval = ...
exitflag = ...
iterations = ...
funcCount = ...
firstorderopt = ...
algorithm = '.....'
message: '.....'

Метод: DFP (HessUpdate = 'dfp')
x = [..... .....]
fval = ...
exitflag = ...
iterations = ...
funcCount = ...

```
firstorderopt = ...
algorithm = '.....'
message: '.....'

Метод: найшвидшого спуску (HessUpdate = 'steepdesc')
x = [.....  .....]
fval = .....
exitflag = ...
iterations = ...
funcCount = ...
firstorderopt = ...
algorithm = '.....'
message: '.....'
```

Контрольні питання

1. Яке призначення функції fminunc і за яких умов її коректно застосовувати.
2. У чому відмінність режимів quasi-newton і trust-region у fminunc та як це відображається у полі output.algorithm.
3. Який зміст показника firstorderopt і чому його доцільно наводити у звіті разом із fval.
4. Як у параметрах options обираються схеми BFGS і DFP, та яка роль параметра HessUpdate.
5. У чому полягає метод найшвидшого спуску та чому він часто поступається BFGS і DFP на функціях із «яружною» структурою.
6. Як інтерпретувати exitflag та message, і чому саме output.message використовують для пояснення причини зупинки.
7. Як впливає вибір початкової точки x0 на результат мінімізації та на кількість ітерацій.
8. Чим відрізняється запуск із GradObj = on від запуску з GradObj = off з погляду кількості обчислень funcCount і відтворюваності результатів.

Навчальне видання

Методичні вказівки
до виконання лабораторних робіт
з дисципліни
«Математичні методи оптимізації»
для студентів спеціальності
G7 «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка»

Укладачі:

КРАСНІКОВ Ігор Леонідович
БАБІЧЕНКО Анатолій Костянтинович
КРАВЧЕНКО Яна Олегівна

Відповідальний за випуск проф. Красніков І.Л.
Роботу рекомендувала до друку доц. Крилова В.А.

В авторській редакції

План 2026 р., поз.154

Гарнітура Times New Roman.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Самостійне електронне видання