Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара
Факультет прикладної математики та інформаційних технологій
Кафедра комп’ютерних технологій

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 3

з курсу «Методи оптимізації»

на тему «Числові методи безумовної оптимізації»

Варіант №14

Виконав:
студент гр. ПА-22-2
Овдієнко Андрій

Дніпро
2025

Зміст

1. Постановка задачі

1.1 Мета

Познайомитись практично з ітераційними методами розв’язання задач безумовної оптимізації.

1.2 Основна постановка задачі

Розв’язати задачу безумовної оптимізації:

$$f(x) \to \min,$$

$$x \in E^n$$

Цільову функції мають вигляд:

1) $f(x) = 2 \cdot {x_1}^2 + 2 \cdot x_1 \cdot x_2 + 1 \cdot {x_2}^2 - 2 \cdot x_1 - 3 \cdot x_2$,

2) функція Bean Function: $g(x) = 0.5 \cdot (x_2 - {x_1}^2)^2 + (1 - x_1)^2 + (1 - x_2)^2$.

1. Знайти точку мінімуму функції f(x) класичним методом вручну.

2. Зробити кілька кроків (не менше двох) методом найшвидшого спуску для функції f(x) вручну.

3. Знайти точку мінімуму функції f(x) методом Ньютона вручну.

4. Розробити програму знаходження оптимального розв’язку задачі безумовної оптимізації градієнтним методом з дробленням кроку. Застосувати її для знаходження оптимального розв’язку для функцій f(x) та g(x) із заданою точністю ε .

5. Розробити програму знаходження оптимального розв’язку задачі безумовної оптимізації методом Ньютона. Застосувати її для знаходження оптимального розв’язку для функцій f(x) та g(x) із заданою точністю ε .

6. Виконати геометричну інтерпретацію отриманих результатів за трьома методами для цільової функції f(x) . Для цього побудувати на площині лінії рівня, траекторію наближення до точки мінімуму, зобразивши напрямки спуску різними кольорами.

7. Скласти звіт.

2. Опис розв'язку

2.1 Знаходження точки мінімума функції f(x) класичним методом вручну

$$f(x) = 2 \cdot {x_1}^2 + 2 \cdot x_1 \cdot x_2 + 1 \cdot {x_2}^2 - 2 \cdot x_1 - 3 \cdot x_2.$$

Знайдемо точку мінімуму функції f (x) класичним методом. Градієнт цієї функції має вигляд:

$$f'(x) = \begin{pmatrix} 4 \cdot x_1 + 2 \cdot x_2 - 2 \\ 2 \cdot x_1 + 2 \cdot x_2 - 3 \end{pmatrix}$$

За необхідної умови мінімуму маємо:

$$\begin{cases} 4 \cdot x_1 + 2 \cdot x_2 - 2 = 0 \\ 2 \cdot x_1 + 2 \cdot x_2 - 3 = 0 \end{cases}$$

$$\begin{cases} 4 \cdot \dfrac{-2 \cdot x_2 + 3}{2} + 2 \cdot x_2 - 2 = 0 \\ x_1 = \dfrac{-2 \cdot x_2 + 3}{2} \end{cases}$$

$$\begin{cases} -4 \cdot x_2 + 6 + 2 \cdot x_2 - 2 = 0 \\ x_1 = \dfrac{-2 \cdot x_2 + 3}{2} \end{cases}$$

$$\begin{cases} -2 \cdot x_2 + 4 = 0 \\ x_1 = \dfrac{-2 \cdot x_2 + 3}{2} \end{cases}$$

$$\begin{cases} 2 \cdot x_2 = 4 \\ x_1 = \dfrac{-2 \cdot x_2 + 3}{2} \end{cases}$$

$$\begin{cases} x_2 = 2 \\ x_1 = \dfrac{-2 \cdot 2 + 3}{2} \end{cases}$$

$$\begin{cases} x_2 = 2 \\ x_1 = -0.5 \end{cases}$$

Будуємо матрицю других похідних (Гесіан):

$$f''(x) = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix}$$

Кутовімінори $M_1 = 4 > 0$, $M_2 = \begin{vmatrix} 4 & 2 \\ 2 & 2 \end{vmatrix} = 4 \cdot 2 - 2 \cdot 2 = 8 - 4 = 4 > 0$. Отже, Гесіан є додатно визначений, тобто точка $x^* = (-0.5;\ 2)$ є точкою мінімума $f(x^*) = 2 \cdot (-0.5)^2 + 2 \cdot (-0.5) \cdot 2 + 1 \cdot 2^2 - 2 \cdot (-0.5) - 3 \cdot 2 = \frac{1}{2} - 2 + 4 + 1 - 6 = \frac{1}{2} - 3 = -2.5$.

2.1 Метод найшвидшого спуска для функції f(x), вручну

Знайдемо два перші наближення до точки мінімуму функції f (x) за методом найшвидшого градієнтного спуску.

$$f(x) = 2 \cdot x_1^2 + 2 \cdot x_1 \cdot x_2 + 1 \cdot x_2^2 - 2 \cdot x_1 - 3 \cdot x_2.$$

Нехай $x^{(0)} = (0; 0)$ - початкова точка, $f(x^{(0)}) = 0$. Як початкову можна брати будь-яку точку простору E^2.

Для визначення крокового множника a_k побудуємо функцію $g(a) = f(x^{(k)} - a \cdot f'(x^{(k)}))$, мінімум якої при $a \geq 0$ необхідно визначити.

Перша ітерація. Обчислимо

$$x^{(1)} = x^{(0)} - a_0 \cdot f'(x^{(0)}),$$

де a_0 – шуканий кроковий множник,

$$f'(x) = \begin{pmatrix} 4 \cdot x_1 + 2 \cdot x_2 - 2 \\ 2 \cdot x_1 + 2 \cdot x_2 - 3 \end{pmatrix},$$

$$f'(x^{(0)}) = \begin{pmatrix} -2 \\ -3 \end{pmatrix}.$$

Тоді отримаємо

$$x^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - a_0 \cdot \begin{pmatrix} -2 \\ -3 \end{pmatrix} = \begin{pmatrix} 2 \cdot a_0 \\ 3 \cdot a_0 \end{pmatrix}.$$

Длявизначення a_0 побудуємо функцію

$$g(a_0) = f(x^{(0)} - a_0 \cdot f'(x^{(0)}))$$

і знайдемо мінімум цієї функції, тобто

$$g(a_0) = 2 \cdot (2a_0)^2 + 2 \cdot (2a_0) \cdot (3a_0) + 1 \cdot (3a_0)^2 - 2 \cdot (2a_0) - 3 \cdot (3a_0)$$
$$= 8 \cdot a_0^2 + 12 \cdot a_0^2 + 9 \cdot a_0^2 - 4 \cdot a_0 - 9 \cdot a_0 = 29 \cdot a_0^2 - 13 \cdot a_0$$
$$\rightarrow min, a_0 \geq 0.$$

$$g'(a_0) = 0;$$

$$58 \cdot a_0 - 13 = 0;$$

$$a_0 = \frac{13}{58}.$$

Таким чином,

$$x^{(1)} = \begin{pmatrix} 2 \cdot a_0 \\ 3 \cdot a_0 \end{pmatrix} = \begin{pmatrix} 2 \cdot (\frac{13}{58}) \\ 3 \cdot (\frac{13}{58}) \end{pmatrix} = \begin{pmatrix} \frac{13}{29} \\ \frac{39}{58} \end{pmatrix}$$

Скористаємося кодом у WolframCloud:

“

f=.

f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

a0 = -13/58;

x1=-2*a0;

x2=-3*a0;

f[x1,x2]

”

$$f(x^{(1)}) = 2 \cdot (\frac{13}{29})^2 + 2 \cdot \frac{13}{29} \cdot \frac{39}{58} + 1 \cdot (\frac{39}{58})^2 - 2 \cdot (\frac{13}{29}) - 3 \cdot \frac{39}{58} = -\frac{169}{116}.$$

Умова монотонності виконується, так як $f(x^{(1)}) = -\frac{169}{116} < 0 = f(x^{(0)})$.

Перевіримо критерій завершення ітераційного процесу, тобто умову $||x^{(k+1)} - x^{(k)}|| < \varepsilon$, де ε - задана точність. Нехай $\varepsilon = 10^{-3}$.

Використаємо код WolframCloud:

“

x01=0;

x02=0;

a0 = -13/58;

x11=-2*a0;

x12=-3*a0;

Sqrt[(x11-x01)^2+(x12-x02)^2]

N[Sqrt[(x11-x01)^2+(x12-x02)^2]]

”

$$||x^{(1)} - x^{(0)}|| = \sqrt{(x_1{}^{(1)} - x_1{}^{(0)})^2 + (x_2{}^{(1)} - x_2{}^{(0)})^2} = \sqrt{(\frac{13}{29} - 0)^2 + (\frac{39}{58} - 0)^2}$$

$$= \frac{13\sqrt{13}}{58} = 0.808141 > \varepsilon.$$

Друга ітерація. Обчислимо

$$x^{(2)} = x^{(1)} - a_1 \cdot f'\left(x^{(1)}\right),$$

де a_1 – шуканий кроковий множник.

$$f'(x) = \begin{pmatrix} 4 \cdot \mathrm{x}_1 + 2 \cdot \mathrm{x}_2 - 2 \\ 2 \cdot \mathrm{x}_1 + 2 \cdot \mathrm{x}_2 - 3 \end{pmatrix},$$

$$x^{(1)} = \begin{pmatrix} \frac{13}{29} \\ \frac{39}{58} \end{pmatrix},$$

Використаємо код у WolframCloud

“

a0 = -13/58;

x1=-2*a0;

x2=-3*a0;

4*x1+2*x2-2

2*x1+2*x2-3

”

$$f'\left(x^{(1)}\right) = \begin{pmatrix} \frac{33}{29} \\ -\frac{22}{29} \end{pmatrix}.$$

$$x^{(2)} = \begin{pmatrix} \frac{13}{29} \\ \frac{39}{58} \end{pmatrix} - a_1 \cdot \begin{pmatrix} \frac{33}{29} \\ -\frac{22}{29} \end{pmatrix} = \begin{pmatrix} \frac{13}{29} - \frac{33}{29} \cdot a_1 \\ \frac{39}{58} - \frac{22}{29} \cdot a_1 \end{pmatrix},$$

Використаємо код WolframCloud

“

```
f=.
f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

a1 = .;
x1=13/29 - 33/29 * a1;
x2=39/58 - 22/29 * a1;

g=Factor[FullSimplify[f[x1,x2]]]
```

”

$$g(a_1) = 2 \cdot (\frac{13}{29} - \frac{33}{29} \cdot a_1)^2 + 2 \cdot (\frac{13}{29} - \frac{33}{29} \cdot a_1) \cdot (\frac{39}{58} - \frac{22}{29} \cdot a_1) + 1$$
$$\cdot (\frac{39}{58} - \frac{22}{29} \cdot a_1)^2 - 2 \cdot (\frac{13}{29} - \frac{33}{29} \cdot a_1) - 3 \cdot (\frac{39}{58} - \frac{22}{29} \cdot a_1)$$
$$= \frac{-4901 - 2401 \cdot a_1 + 16456 \cdot {a_1}^2}{3364} \to min, a_1 \geq 0.$$

Використаємо код WolframCloud

“

```
f=.
f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

a1 = .;
x1=13/29 - 33/29 * a1;
x2=39/58 - 22/29 * a1;

g=Factor[FullSimplify[f[x1,x2]]];

ga1 =D[g,a1]
```

”

$$g'(a_1) = \frac{-2401 + 32912 \cdot a_1}{3364};$$

$$g'(a_1) = 0;$$

$$\frac{-2401 + 32912 \cdot a_1}{3364} = 0;$$

Використаємо код у WolframeCloud:

“

```
f=.
f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

a1 = .;
x1=13/29 - 33/29 * a1;
x2=39/58 - 22/29 * a1;

g=Factor[FullSimplify[f[x1,x2]]];

ga1 =D[g,a1] ;

Solve[{ga1==0,a1>=0},a1]
```

”

отримали $a_1 = \frac{5}{68}$.

Використаємо код у wolframCloud:

“

```
f=.
f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

x1 = 13/29-33/29*5/68
x2 = 39/58 -22/29*5/68

result=f[x1,x2]
```

”

$$x^{(2)} = \begin{pmatrix} \frac{13}{29} - \frac{33}{29} \cdot a_1 \\ \frac{39}{58} - \frac{22}{29} \cdot a_1 \end{pmatrix} = \begin{pmatrix} \frac{13}{29} - \frac{33}{29} \cdot \frac{5}{68} \\ \frac{39}{58} - \frac{22}{29} \cdot \frac{5}{68} \end{pmatrix} = \begin{pmatrix} \frac{719}{1972} \\ \frac{304}{493} \end{pmatrix}$$

$$f(x^{(2)}) = -\frac{169659}{114376}$$

Використаємо код у WolframCloud;

“

```
f=.
f[x1_,x2_]:=2*x1^2+2*x1*x2+1*x2^2-2*x1-3*x2;

x1 = 13/29-33/29*5/68;
x2 = 39/58 -22/29*5/68;

result=f[x1,x2]

fx2 =N[result]

result=f[13/29,39/58]

fx1 =N[result]
```

”

Умова монотонності виконується, так як $f(x^{(2)}) = -\frac{169659}{114376} = -1.48334 < -1.4569 = -\frac{169}{116} = f(x^{(1)})$.

Перевіримо критерій завершення ітераційного процесу, тобто умову $||x^{(k+1)} - x^{(k)}|| < \varepsilon$, де ε - задана точність. Нехай $\varepsilon = 10^{-3}$.

Використаємо код WolframCloud:

“

```
x11=13/29;
x12=39/58;
```

x21=719/1972;

x22=304/493;

result = Sqrt[(x11-x21)^2+(x12-x22)^2]

N[result]

”

$$||x^{(2)} - x^{(1)}|| = \sqrt{(x_1{}^{(2)} - x_1{}^{(1)})^2 + (x_2{}^{(2)} - x_2{}^{(1)})^2}$$

$$= \sqrt{(\frac{719}{1972} - \frac{13}{29})^2 + (\frac{304}{493} - \frac{39}{58})^2} = \frac{55\sqrt{13}}{1972} = 0.100561 > \varepsilon.$$

Наближене значення мінімуму за дві ітерації $x^* = x^{(2)} = \begin{pmatrix} \frac{719}{1972} \\ \frac{304}{493} \end{pmatrix} = \begin{pmatrix} 0.364604 \\ 0.616633 \end{pmatrix}$, $f(x^*) = -\frac{169659}{114376} = -1.48334.$

2.3 Метод Ньютона, вручну f(x)

Розв’яжемо задачу методом Ньютона. Матриця других похідних для квадратичної функції є постійною і для заданої функції має вигляд:

$$\mathrm{f}''(\mathrm{x}) = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix}$$

Перевіримо визначник матриці Гесіан.

$$\begin{vmatrix} 4 & 2 \\ 2 & 2 \end{vmatrix} = 4 \cdot 2 - 2 \cdot 2 = 8 - 4 = 4 \neq 0.$$

Головна умова існування оберненої матриці виконана.

$$[\mathrm{f}''(\mathrm{x})]^{-1} = \frac{1}{4}\begin{pmatrix} 2 & -2 \\ -2 & 4 \end{pmatrix}$$

Нехай $\mathrm{x}^{(0)} = (0; 0)$ - початкова точка, $\mathrm{f}(\mathrm{x}^{(0)}) = 0$.

За ітераційною формулою для першого наближення маємо:

$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} - [\mathrm{f}''(\mathrm{x}^{(0)})]^{-1} \cdot \mathrm{f}'(\mathrm{x}^{(0)}),$$

$$\mathrm{f}'(\mathrm{x}) = \begin{pmatrix} 4 \cdot \mathrm{x}_1 + 2 \cdot \mathrm{x}_2 - 2 \\ 2 \cdot \mathrm{x}_1 + 2 \cdot \mathrm{x}_2 - 3 \end{pmatrix}$$

$$f'(x^{(0)}) = \begin{pmatrix} -2 \\ -3 \end{pmatrix},$$

$$x^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 2 & -2 \\ -2 & 4 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ -3 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -1 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ -3 \end{pmatrix} =$$

$$= \begin{pmatrix} (-\frac{1}{2}) \cdot (-2) + \frac{1}{2} \cdot (-3) \\ \frac{1}{2} \cdot (-2) + (-1) \cdot (-3) \end{pmatrix} = \begin{pmatrix} 1 - \frac{3}{2} \\ -1 + 3 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ 2 \end{pmatrix} = \begin{pmatrix} -0.5 \\ 2 \end{pmatrix}.$$

Як бачимо, $x^* = x^{(1)} = \begin{pmatrix} -0.5 \\ 2 \end{pmatrix}$ – точка мінімуму $f(x^*) = -2.5$. Тобто, для квадратичної функції, матриця других похідних $f''(x)$ якої є додатно визначеною, метод Ньютона збігається за одну ітерацію до точного значення мінімуму.

2.5 Градієнтний метод з дробленням кроку, запрограмовування f(x) і g(x)

Градієнтний метод відбувається таким чином, що програма сама вибирає початкові значення $\beta = \varepsilon;\ \alpha = \varepsilon;\ \lambda = \varepsilon$. А далі перевіряє, чи вонується умова для к={0,1,2,3,4,5,…,n}:

$$f\left(x^{(k)} - \alpha f'\left(x^{(k)}\right)\right) < f\left(x^{(k)}\right)$$

Рисунок 1 – Умова вибору а в градієнтному методі дроблення кроку.

Якщо умова не виконується – то робимо робимо дроблення $\alpha = \alpha * \lambda$.

Якщо умова виконується – то робимо наступну ітерацію градієнтного метода.

$$x^{k+1} = x^k + a * f'(x^k), a > 0, k = \{0,1,2,3,4,5,\dots,n\}.$$

Критерій зупинки є всі 3 наступні умови нижче одночасно:

$$\left\| x^{(k+1)} - x^{(k)} \right\| \leq \varepsilon_1,$$

$$\left| f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right) \right| \leq \varepsilon_2,$$

$$\left\| f'\left(x^{(k+1)}\right) \right\| \leq \varepsilon_3.$$

Рисунок 2 – Критерії збіжності, які були реалізовані.

Оптимальне значення $\varepsilon = 1e - 4 = 0.0001$. Початкова точка $x^0 = (13; 13)$ Були отримані приблизні результати.

```
Градієнтний метод з дробленням кроку
    f(-0.49993118092012306; 1.9998886483896818) = -2.499999993454918;
    g(1.0000223575074811; 1.0000447158478738) = 2.499365191897536e-09.
```

Рисунок 3 – Результати градієнтного метода з дробленням кроку.

Як бачимо – для функції f(x) маємо дійсно наближенний резльтат.

2.6 Метод Ньютона, запрограмовування f(x) і g(x)

Для метода Ньютона слід було знайти похідні другого порядку і перевірити, щоб визначник похідних другого порядку не був нульовим. Тоді можна використовувати ітераційну формулу:

$$x^{k+1} = x^k - [f''(x^k)]^{-1} \cdot f'(x^k).$$

Критерій зупинки є всі 3 наступні умови нижче одночасно:

$$\left\| x^{(k+1)} - x^{(k)} \right\| \leq \varepsilon_1,$$

$$\left| f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right) \right| \leq \varepsilon_2,$$

$$\left\| f'\left(x^{(k+1)}\right) \right\| \leq \varepsilon_3.$$

Рисунок 4 – Критерії збіжності, які були реалізовані.

Обрано значення $\varepsilon = 1e - 4 = 0.0001$, щоб точність співпадала з градієнтним методом дроблення кроку, хочу спокійно можна встановити точність $\varepsilon = 1e - 14$. Початкова точка $x^0 = (13; 13)$ Були отримані приблизні результати.

```
Метод Ньютона
    f(-0.5; 2.0) = -2.5;
    g(1.0000210159814642; 1.0000133311728103) = 1.0312719985180364e-09.
```

Рисунок 5 – Результати метода Ньютона.

3. Код програми на Python

```
import math

class Point:
    def __init__(self, x=0, y=0):
        self.__x = float(str(x).replace(' ','').replace(',','.'))
        self.__y = float(str(y).replace(' ','').replace(',','.'))

    def get_x(self):
        return self.__x
    def get_y(self):
        return self.__y
    def set_x(self, x):
        self.__x = x
    def set_y(self, y):
        self.__y = y

    def __str__(self):
        return '(' +str(self.get_x()) + '; ' + str(self.get_y()) + ')'

    def f(self):
        return 2 * self.get_x() ** 2 + 2* self.get_y() * self.get_x() + self.get_y() ** 2 -
2*self.get_x() - 3* self.get_y()

    def fx(self):
        return 4 * self.get_x() + 2 * self.get_y() - 2

    def fy(self):
        return 2 * self.get_x() + 2 * self.get_y() - 3

    def fxx(self):
        return 4

    def fxy(self):
```

```
        return 2

    def fyy(self):
        return 2

    def fyx(self):
        return 2

    def g(self):
        return 0.5 * (self.get_y() - self.get_x() ** 2) ** 2 + (1 - self.get_x()) ** 2 + (1 - self.get_y()) ** 2

    def gx(self):
        return -2*(1-self.get_x()) - 2 * self.get_x() * (self.get_y() - self.get_x() ** 2)

    def gy(self):
        return -2*(1-self.get_y()) +  (self.get_y() - self.get_x() ** 2)

    def gxx(self):
        return 2 + 4 * self.get_x() ** 2 - 2 * (self.get_y() - self.get_x() ** 2)

    def gxy(self):
        return (-2) * self.get_x()

    def gyx(self):
        return (-2) * self.get_x()

    def gyy(self):
        return 3


    def diff(self, another):
        return math.sqrt(  (self.get_x() - another.get_x()) ** 2 +  (self.get_y() - another.get_y()) ** 2 )
```

```
def gradient_method_with_step_separation(function = 'f',point = Point(0,0),eps = 1e-5):


    last_point = Point(point.get_x(), point.get_y())
    lam = eps
    a =  lam


    while True:

        while True:

            was = True

            match (function):
                case 'f':
                    temp_point = Point(last_point.get_x() - a * last_point.fx(), last_point.get_y() - a
* last_point.fy())
                    if temp_point.f() < last_point.f():
                        was = False
                case 'g':
                    temp_point = Point(last_point.get_x() - a * last_point.gx(), last_point.get_y() - a
* last_point.gy())
                    if temp_point.g() < last_point.g():
                        was = False

            if not was:
                break

            a *= lam

        match (function):
            case 'f':
                point.set_x(last_point.get_x() - a * last_point.fx())
                point.set_y(last_point.get_y() - a * last_point.fy())
```

```
            case 'g':
                point.set_x(last_point.get_x() - a * last_point.gx())
                point.set_y(last_point.get_y() - a * last_point.gy())



        was = True
        if point.diff(last_point) < eps:
            match (function):
                case 'f':
                    if math.fabs(point.f() - last_point.f()) < eps:
                        norm_of_gradient = math.sqrt((point.fx()) ** 2 + (point.fy()) ** 2)
                        if norm_of_gradient < eps:
                            was = False

                case 'g':
                    if math.fabs(point.g() - last_point.g()) < eps:
                        norm_of_gradient = math.sqrt((point.gx()) ** 2 + (point.gy()) ** 2)
                        if norm_of_gradient < eps:
                            was = False



        if not was:
            break

        last_point = Point(point.get_x(), point.get_y())

    def newton_method(function = 'f',point = Point(0,0),eps = 1e-5):
        last_point = Point(point.get_x(), point.get_y())

        match (function):
            case 'f':
                temp = last_point.fxx() * last_point.fyy() - last_point.fxy() * last_point.fyx()
                if temp == 0:
```

```
    raise ValueError("Визначник = 0.")

det = 1 / temp

while True:

    point.set_x(
        last_point.get_x() - det * (
        (last_point.fyy()*last_point.fx())
        -
        (last_point.fyx()*last_point.fy())
        )
    )

    point.set_y(
        last_point.get_y() - det * (
        (-1) * (last_point.fxy()*last_point.fx())
        +
        last_point.fxx() * last_point.fy()
        )
    )

    was = True
    if point.diff(last_point) < eps:

        if math.fabs(point.f() - last_point.f()) < eps:
            norm_of_gradient = math.sqrt((point.fx()) ** 2 + (point.fy()) ** 2)
            if norm_of_gradient < eps:
                was = False


    if not was:
        break

    last_point = Point(point.get_x(), point.get_y())
```

```
case 'g':
    temp = last_point.gxx() * last_point.gyy() - last_point.gxy() * last_point.gyx()
    if temp == 0:
        raise ValueError("Визначник = 0.")

    det = 1 / temp

    while True:

        point.set_x(
            last_point.get_x() - det * (
                (last_point.gyy() * last_point.gx())
                -
                (last_point.gyx() * last_point.gy())
            )
        )

        point.set_y(
            last_point.get_y() - det * (
                (-1) * (last_point.gxy() * last_point.gx())
                +
                last_point.gxx() * last_point.gy()
            )
        )

        was = True
        if point.diff(last_point) < eps:

            if math.fabs(point.g() - last_point.g()) < eps:
                norm_of_gradient = math.sqrt((point.gx()) ** 2 + (point.gy()) ** 2)
                if norm_of_gradient < eps:
                    was = False

        if not was:
```

```
                break

            last_point = Point(point.get_x(), point.get_y())

def main():
    eps = 1e-4

    x0 = 13
    y0 = 13

    print("Градієнтний метод з дробленням кроку")

    point = Point(x0, y0)
    gradient_method_with_step_separation(function='f', point=point, eps=eps)
    print(f"\tf{point} = {point.f()};")

    point = Point(x0,y0)
    gradient_method_with_step_separation(function='g',point=point,eps=eps)
    print(f"\tg{point} = {point.g()}.")

    print("\n\nМетод Ньютона")

    point = Point(x0, y0)
    try:
        newton_method(function='f', point=point, eps=eps)
        print(f"\tf{point} = {point.f()};")
    except ValueError as e:
        print(e)

    point = Point(x0,y0)
    try:
        newton_method(function='g', point=point, eps=eps)
        print(f"\tg{point} = {point.g()}.")
    except ValueError as e:
        print(e)
```

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

4. Геометрична інтерпритація

Для геометричної ітерпритації функції f(x) використаємо Desmos Геометрія. Відобразимо початкову точку (0,0) з якої починається наближення і саму точку мінімума (-0.5; 2).

Прирівняємо праву частину f(x) певній константі і отримаємо еліпс – це є лінія рівня. Побудуємо кілька таких ліній рівня.

Також відобразимо шляхи, які були пройдені при кожному з трьох методів: Ньютона, Градієнтний дроблення кроку, Градієнтний найшвидшого спуска (тут лише 2 ітерації).

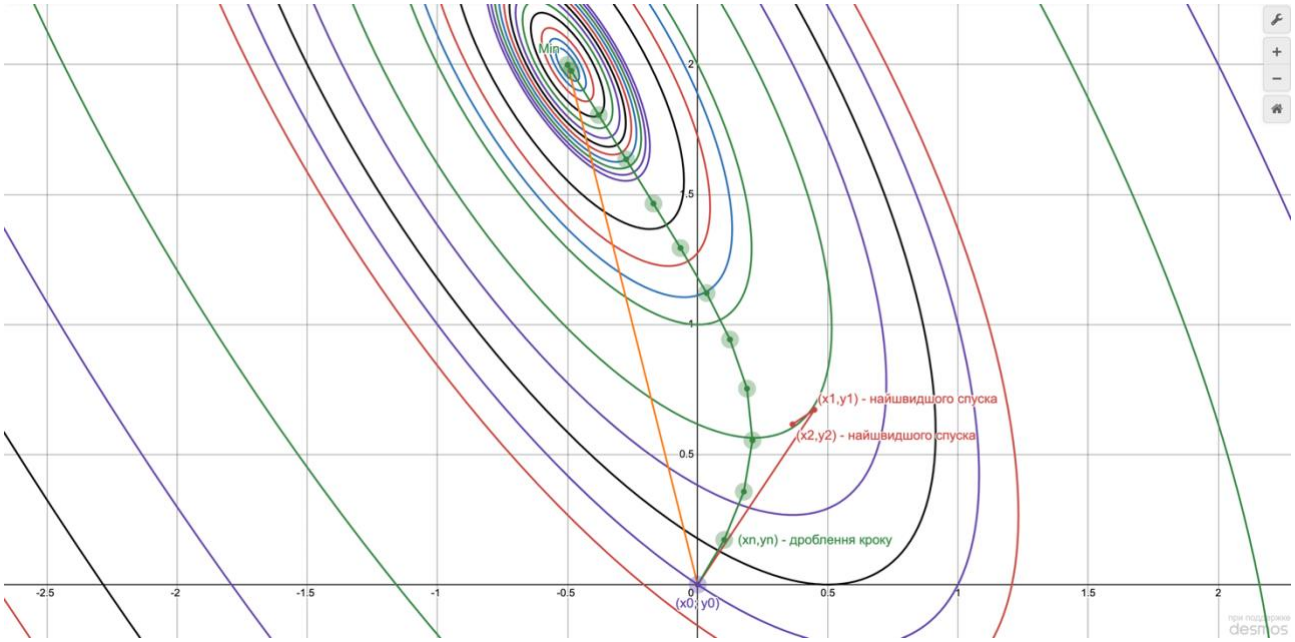

Рисунок 6 – Геометрична інтерпритація наближення трьох методів: Ньютона (поморанчева лінія), Найшвидшого спуска (червона лінія), Дроблення кроку (зелена лінія) - та лінії рівня.

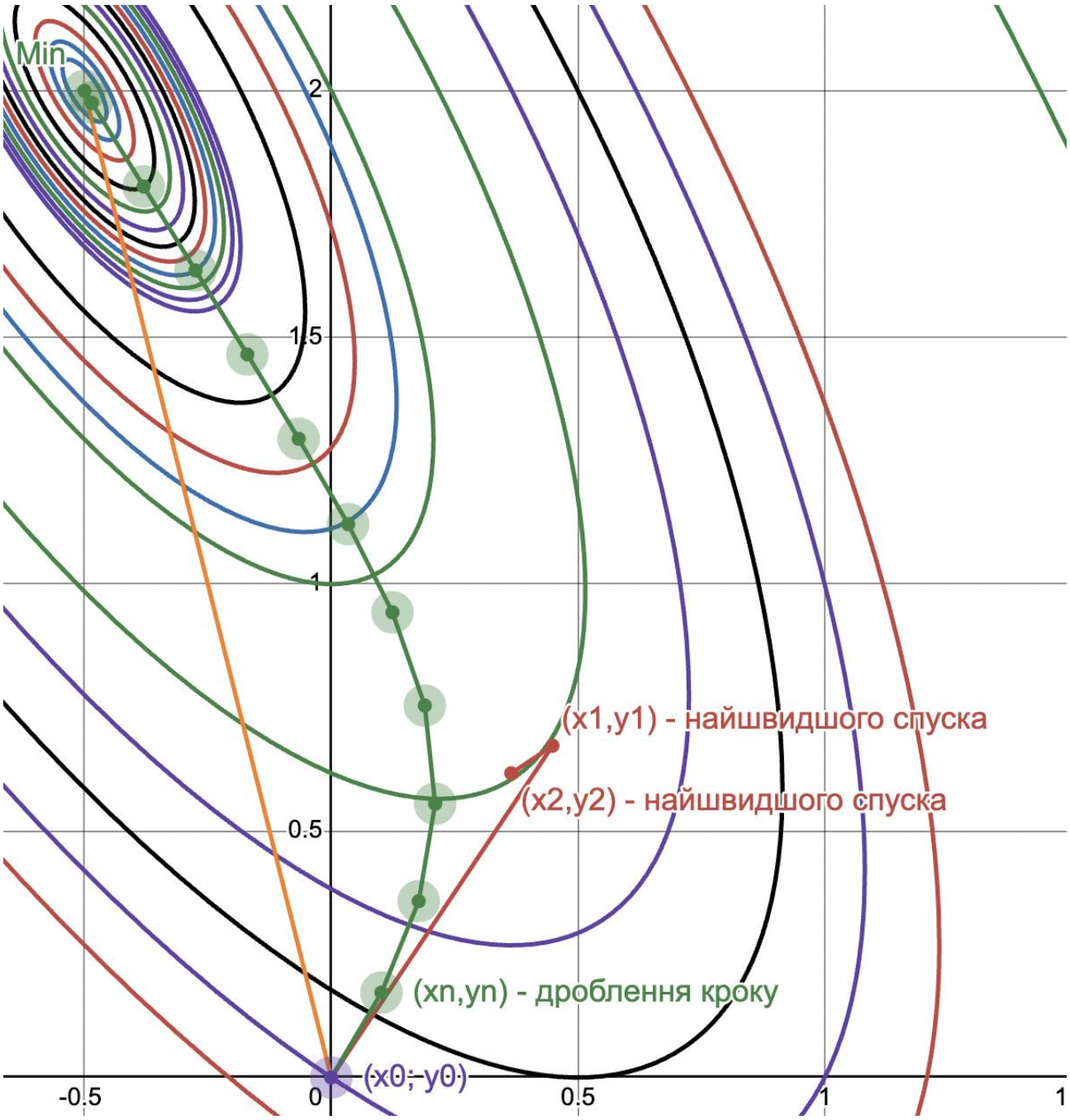

Рисунок 7 – Геометрична інтерпритація наближення трьох методів: Ньютона (поморанчева лінія), Найшвидшого спуска (червона лінія), Дроблення кроку (зелена лінія) - та лініі рівня.

Бачимо, що найшвидше надає точний результат метод Ньютона – за одну ітерацію – одразу по прямій (квадратична збіжність). Наступним йде Дроблення кроку, звісно тут точок наближення дуже багато, тому продемостровані лише деякі, але можно побачити, що збіжність відбувається подібно до лініі рівня. І той, що було зроблено лише руками перші два кроки – найшвидшого спуска.

5. Аналіз результатів

Для f(x) маємо точку мінімума (-0.5; 2). Знайдемо власноруч для g(x) точку мінімума (точну).

$$g(x) = 0.5 \cdot (x_2 - {x_1}^2)^2 + (1 - x_1)^2 + (1 - x_2)^2$$

Знайдемо часткові похідні першого порядку.

$$g'(x) = \begin{pmatrix} -2 \cdot (1 - x_1) - 2 \cdot x_1 \cdot (-{x_1}^2 + x_2) \\ -2 \cdot (1 - x_2) - {x_1}^2 + x_2 \end{pmatrix}$$

За необхідної умови мінімуму маємо:

$$\begin{cases} -2 \cdot (1 - x_1) - 2 \cdot x_1 \cdot (-{x_1}^2 + x_2) = 0 \\ -2 \cdot (1 - x_2) - {x_1}^2 + x_2 = 0 \end{cases};$$

Використаємо код у Wolfram Cloud

“

```
g[x_,y_]:=0.5*(y-x^2)^2 + (1-x)^2+(1-y)^2


x=.
y=.
R = g[x,y];


Rx = D[R,x];
Ry = D[R,y];


Solve[{Rx==0,Ry==0},{x,y}]
```

”

Отримали рішення (1;1) – точка мінімума.

$$g(1; 1) = 0.5 \cdot (1 - 1^2)^2 + (1 - 1)^2 + (1 - 1)^2 = 0.$$

Тоді можна побудувати таблицю порівнянь точності методів при $\varepsilon = 1e - 4$.

	Точне значення	Метод Ньютона	Похибка
f(x)	-2.5	-2.5	0
g(x)	0	0.00000000103127199851803 64	0.00000000103127199851803 64

	Точне значення	Градієнтний метод з дробленням кроку	Похибка
f(x)	-2.5	-2.499999993454918	0.000000006545082
g(x)	0	0.00000000249936519189753 6	0.00000000249936519189753 6

Проаналізувавши отримані дані можна сказати, що метод Ньютона є певною мірою більш точним, але лише трохи, хоча в свою чергу є ефективнішим, бо вимагає меншої кількості ітерацій.

6. Висновки

Познайомилися практично з ітераційними методами розв'язання задач безумовної оптимізації.

Розв'язали задачі безумовної оптимізації: $f(x) \to \min,\ x \in E^n$ - цільові функції яких мають вигляд:

$$f(x) = 2 \cdot {x_1}^2 + 2 \cdot x_1 \cdot x_2 + 1 \cdot {x_2}^2 - 2 \cdot x_1 - 3 \cdot x_2,$$

$$g(x) = 0.5 \cdot ({x_2} - {x_1}^2)^2 + (1 - x_1)^2 + (1 - x_2)^2.$$

Знайти точку мінімуму функції f(x) та g(x) класичним методом вручну.отримали результати:

$$f_{min}(-0.5; 2) = -2.5;$$

$$g_{min}(1; 1) = 0.$$

Зробити два кроки методом найшвидшого спуску для функції f(x) вручну. Отримали результат: $f(0.364604; 0.616633) = -1.48334$.

Знайти точку мінімуму функції f(x) методом Ньютона вручну. Отримали результат, що збігається з класичним методом на першій ітерації.

Розробили програму знаходження оптимального розв'язку задачі безумовної оптимізації градієнтним методом з дробленням кроку. Застосували її для знаходження оптимального розв'язку для функцій f(x) та g(x) із заданою точністю $\varepsilon = 1e - 4$. За початкову точку обирали як (0; 0), так і (13; 13). Отримали результати:

$$f_{min}(0.49993118092012306; 1.9998886483896818) = -2.499999993454918;$$

$$g_{min}(1.0000223575074811;\ 1.0000447158478738) = 0.000000002499365191897536.$$

Розробили програму знаходження оптимального розв'язку задачі безумовної оптимізації методом Ньютона. Застосували її для знаходження оптимального розв'язку для функцій f(x) та g(x) із заданою точністю $\varepsilon = 1e - 4$. За початкову точку обирали як (0; 0), так і (13; 13). Лише для g(x) не справлялася початкова точка (0; 0) – застрягла в локальному мінімумі. Отримали результати:

$$f_{min}(-0.5; 2) = -2.5;$$

$$g_{min}(1.000021015981416 42;\ 1.0000133311728103) = 0.00000000103127199851803 64.$$

Виконали геометричну інтерпретацію отриманих результатів за трьома методами для цільової функції f(x) . Для цього побудували на площині лінії рівня, траекторію наближення до точки мінімуму, зобразивши напрямки спуску різними кольорами.

Виявили, що метод Ньютона є більш оптимальним – бо він має квадратичну збіжність, тобто за меншу кількість ітерацій дістався до глобального мінімума, чи досить близького значення до нього.

Похибка в обох повністю реалізованих методах не перевищує 1e-8.