Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики

Лабораторна робота №1
з курсу "Проблеми багатозначного аналізу"

Виконала студентка групи ОМ-2
Живолович Олександра Євгенівна

Київ, 2021

Зміст

1 Постановка задачі

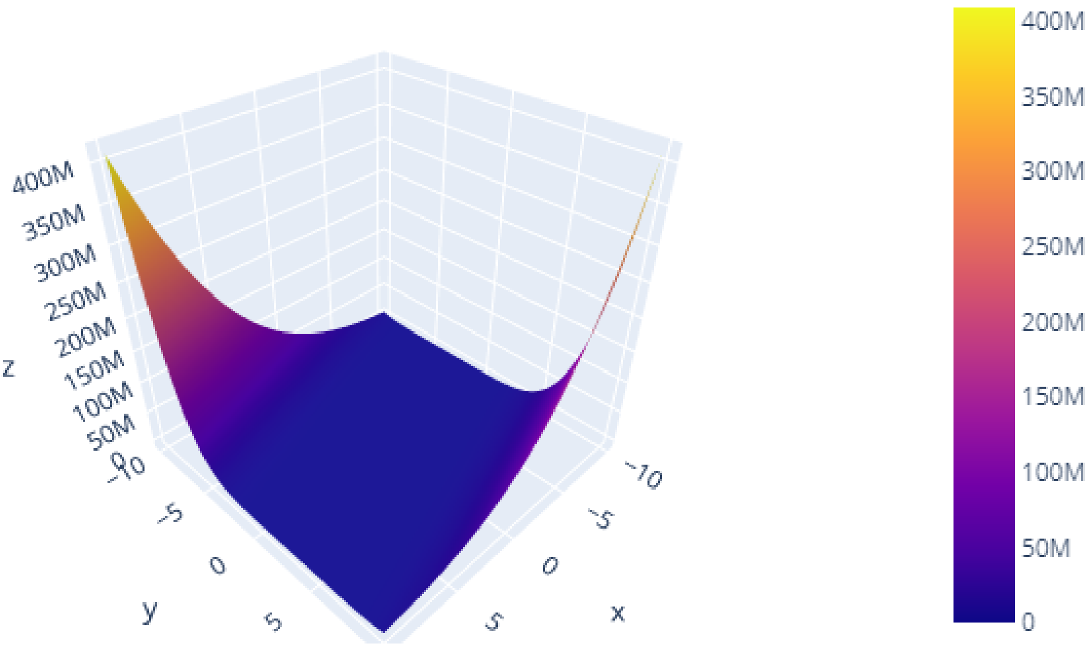
Побудувати графік субдиференціалу функції

$$f(x_1, x_2) = |-4x_1 + 2x_2 + 4| + |2x_1 + 3x_2 - 2| + 0.1(x_1 - 3x_2)^6, \quad (x_1, x_2) \in \mathbb{R}^2. \tag{1}$$

Розв'язати задачу

$$f(x_1, x_2) \to \min \tag{2}$$

Рис. 1: Графік f



2 Субдиференціал

2.1 Теорія

Для того, щоб знайти субдиференціал функції f, скористаємося теоремою Моро-Рокафеллера, тобто

$$\partial f(x) = \sum_{i=1}^{3} \partial f_i(x)$$

де

$$
\begin{aligned}
f_1(x_1, x_2) &= | - 4x_1 + 2x_2 + 4| \\
f_2(x_1, x_2) &= |2x_1 + 3x_2 - 2| \\
f_3(x_1, x_2) &= 0.1(x_1 - 3x_2)^6
\end{aligned}
$$

Розглянемо $f_1(x_1, x_2)$. Для того, щоб знайти субдиференціал у точці, де функція не є диференційовною скористаємося теоремою Дубовітського - Мілютіна, тобто

$$
\partial f_1(x_1, x_2) = \partial \max\{-4x_1 + 2x_2 + 4, 4x_1 - 2x_2 - 4\} = \partial \max\{f_{11}(x_1, x_2), f_{12}(x_1, x_2)\} = \\
= co\left(\bigcup_{i=1}^{n} \partial f_{1i}(x_1, x_2)\right)
$$

Тоді

$$
\partial f_1(x_1, x_2) = \begin{cases} \{(-4, 2)\}, 2x_2 + 4 > 4x_1 \\ \{(4, -2)\}, 2x_2 + 4 < 4x_1 \\ [(-4, 2), (4, -2)],\ 2x_2 + 4 = 4x_1 \end{cases}
$$

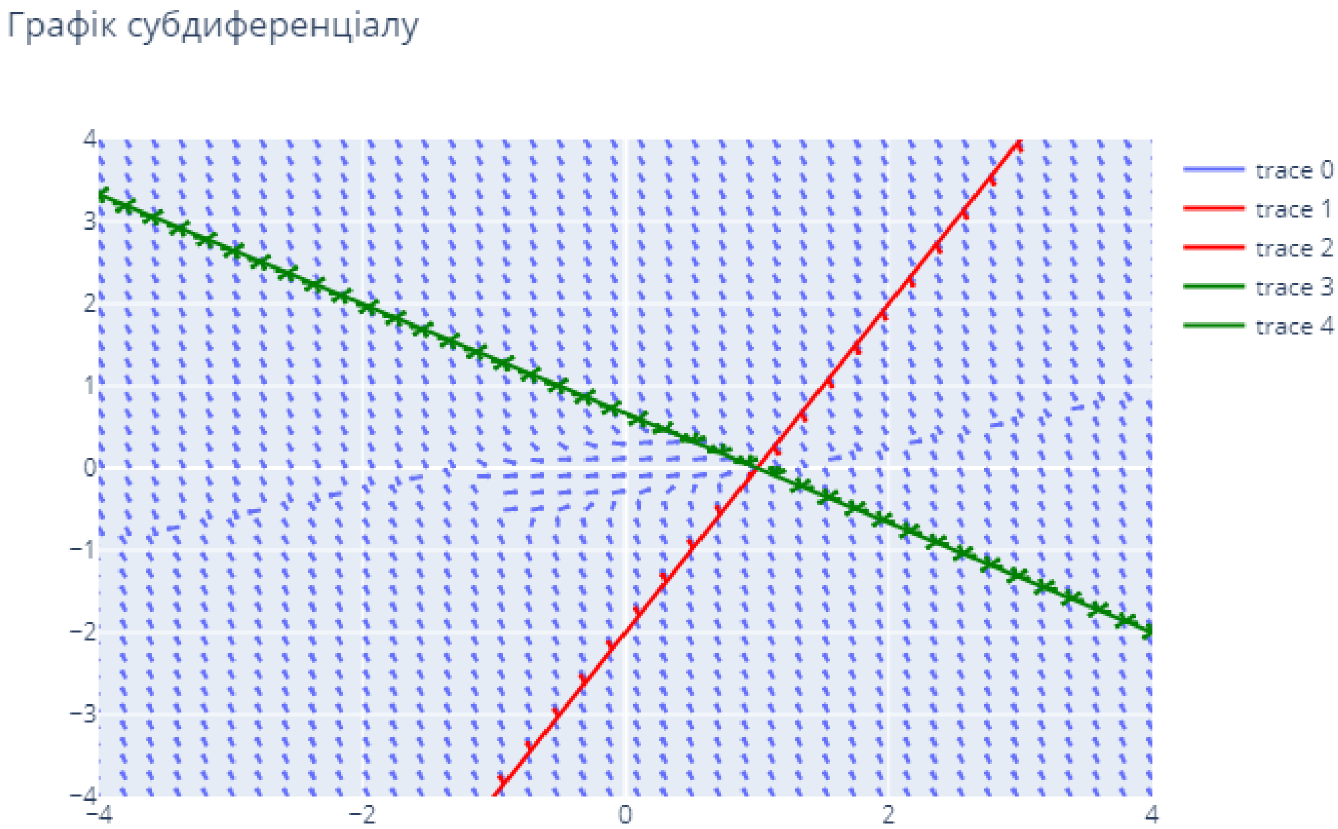
Аналогічно для $f_2(x_1, x_2)$:

$$
\partial f_2(x_1, x_2) = \begin{cases} \{(2, 3)\}, 2x_1 + 3x_2 - 2 > 2 \\ \{(-2, -3)\}, 2x_1 + 3x_2 - 2 < 2 \\ [(-2, 3), (2, -3)],\ 2x_1 + 3x_2 - 2 = 2 \end{cases}
$$

Оскільки $f_3(x_1, x_2)$ диференційовна $\forall x \in \mathbb{R}^2$, то її субдиференціал співпадає з градієнтом в усіх точках

$$
\partial f_3(x_1, x_2) = \{(0.6(x_1 - 3x_2)^5, -1.8(x_1 - 3x_2)^5)\}
$$

Рис. 2: Субдиференціал



3 Алгоритм

Нехай $f : \mathbb{R}^n \to \mathbb{R}$ опукла. Для мініміщації f, використовуємо субградієнтний метод

$$x_{k+1} = x_k - \alpha_k g(x_k) \tag{3}$$

де x_k є k-тою точкою ітерації, $g(x_k)$ - будь-який субградієнт функції f у точці x_k, α_k - розмір k-того кроку.

Отже, на кожному році методу, ми робимо крок у напрямку від'ємного субградієнту функції f. У точках, де f диференційовна, субградієнт співпадає з градієнтом функції, а отже у цих точках метод зводиться до градієнтного методу.

Оскільки метод не є методом спуску, то гарним підходом є збереження найкращого поточного результату

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f^{(k)}\} \implies f_{\text{best}}^{(k)} = \min\{f(x_1), \dots, f(x_k)\}.$$

Важливою частною методу є спосіб вибору α_k - розміру кроку. Існують наступні широковживанні методи вибору α_k:

- Метод постійного розміру кроку

$$\alpha_k = h$$

- Метод кроку постійної довжини

$$\alpha_k = \frac{h}{||g(x_k)||}$$

- Метод сумовних з квадратом кроків

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

Наприклад, $\alpha_k = a/(b+k)$, $a > 0$, $b \geq 0$.

- Метод несумовних кроків, що зменшуються

$$\lim_{k \to \infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

Наприклад, $\alpha_k = a/\sqrt{k}$.

4 Код

Було обрано мову програмування *Python3*, оскільки вона є дуже зручною для роботи з науковими обчисленнями, а також існують багато зручних пакетів для візуалізації даних. Код можна знайти за посиланням.

Функціїї `plotObjective()` та `plotSubDiff()` будують графіки функції та субдиференціалу відповідно, за допомогою програмного пакету `plotly`.

Усі обчисленнями виконується у функції `optimize`, що вирішує задачу негладкої оптимізації за допомогою субградієнтного методу. Функція має багато параметрів, тобто є гнучкою у своїі роботі, з багатьма значеннями, виставленими за замовчуванням.

Графіки для візуалізації результатів роботи алгоритму будуються за домопомогою пакету `matplotlib`.

```
    def optimize(f,g,initial_point, strategy=Strategy.consant_size, h=0.1, a=1.0,
    b=1.0, max_iterations=10000, tolerance=1e-6,
):
    points = []
    vals = []
    best_point = initial_point
    best_value = f(best_point)
    points.append(best_point)
    vals.append(best_value)

    step_size = 0.0
    for iter in range(1, max_iterations):
        if strategy == Strategy.consant_size:
            step_size = h
        elif strategy == Strategy.constant_length:
            step_size = h / np.linalg.norm(g(points[-1]))
        elif strategy == Strategy.square_summable:
            step_size = a / (iter + b)
        elif strategy == Strategy.diminishing:
            step_size = a / np.sqrt(iter)

        new_point = points[-1] - step_size * g(points[-1])
        new_value = f(new_point)
        points.append(new_point)
        vals.append(new_value)

        if new_value < best_value:
            best_point = new_point
            best_value = new_value
        if (
            abs(vals[-1] - vals[-2]) < tolerance
            and np.linalg.norm(points[-1] - points[-2]) < tolerance
        ):
            break

    return points, vals, best_point, best_value
```

5 Обчислювальні експерименти

Сторонні математичні системи, такі як *WolframAlpha*, знаходять мінімум функції у точці $(1.25665, 0.0133085)$, зі значенням $f(x^*) = 0.877695$. Розглянемо застосування субградієнтного методу з різними способами вибору кроку

5.1 Постійний розмір

1. $h = 0.01$: $x_{best} = (1.2511, 0.0065)$, $f(x_{best}) = 0.8792$, $\Delta = 0.001$
2. $h = 0.001$: $x_{best} = (1.25490.0100)$, $f(x_{best}) = 0.8779$, $\Delta = 0.0002$
3. $h = 0.0001$: $x_{best} = (0.59930.0999)$ $f(x_{best}) = 3.3038$, $\Delta = 2.4261$

5.2 Постійна довжина

1. $h = 0.01$: $x_{best} = (1.2536, 0.0076)$, $f(x_{best}) = 0.8784$, $\Delta = 0.0007$
2. $h = 0.001$: $x_{best} = (0.9211, 0.0533)$, $f(x_{best}) = 1.4438$, $\Delta = 0.5661$
3. $h = 0.0001$: $x_{best} = (0.0985, 0.0164)$, $f(x_{best}) = 6.3923$, $\Delta = 5.5146$

5.3 Сумовні з квадратом

1. $a = 0.1, b = 1.0$: $x_{best}) = (1.2098, -0.0072)$, $f(x_{best}) = 0.8930$, $\Delta = 0.01535$
2. $a = 0.05, b = 1.0$: $x_{best}) = (1.0966, -0.0449)$, $f(x_{best}) = 0.9307$, $\Delta = 0.05306$
3. $a = 0.05, b = 10.0$: $x_{best}) = (1.0573, -0.0380)$, $f(x_{best}) = 0.9536$, $\Delta = 0.07592$

5.4 Несумовні, що зменшуються

1. $a = 0.1$: $x_{best} = (1.2537, 0.00752)$, $f(x_{best}) = 0.8784$, $\Delta = 0.0007$
2. $a = 0.05$: $x_{best} = (1.2544, 0.00895)$, $f(x_{best}) = 0.8781$, $\Delta = 0.0004$
3. $a = 0.01$: $x_{best} = (1.1841, -0.01580)$, $f(x_{best}) = 0.9016$, $\Delta = 0.0239$

Візуалізації цих експериментів наведено у додатках.

6 Висновки

Була вирішена задача негладкої оптимізації за допомогою субградієнтного методу. Були зроблені такі спостереження:

1. Оскільки функція дуже швидко зростає, то потрібно відразу брати малі кроки для того щоб метод був збіжний
2. Надто малий розмір року призводить до вкрай повільної збіжності
3. Методи постійного розміру кроку та постійної довжини кроку показують гарні результати, але потребують гарного підбору параметру.
4. Метод сумовних з квадратом кроків показав гарний результат, навіть для не дуже вдалих параметрів.
5. Метод несумовних кроків, що зменшуються показав дуже гарний результат, але треба розуміти що зменшувати розмір параметру потрібно обережно, адже можно швидко прийти до дуже повільної збіжності.

Література

[1] *Subgradient Methods*, Stephen Boyd, Lin Xiao, and Almir Mutapcic Notes for EE392o, Stanford University, Autumn, 2003

7 Додатки

Рис. 3: Постійний розмір кроку

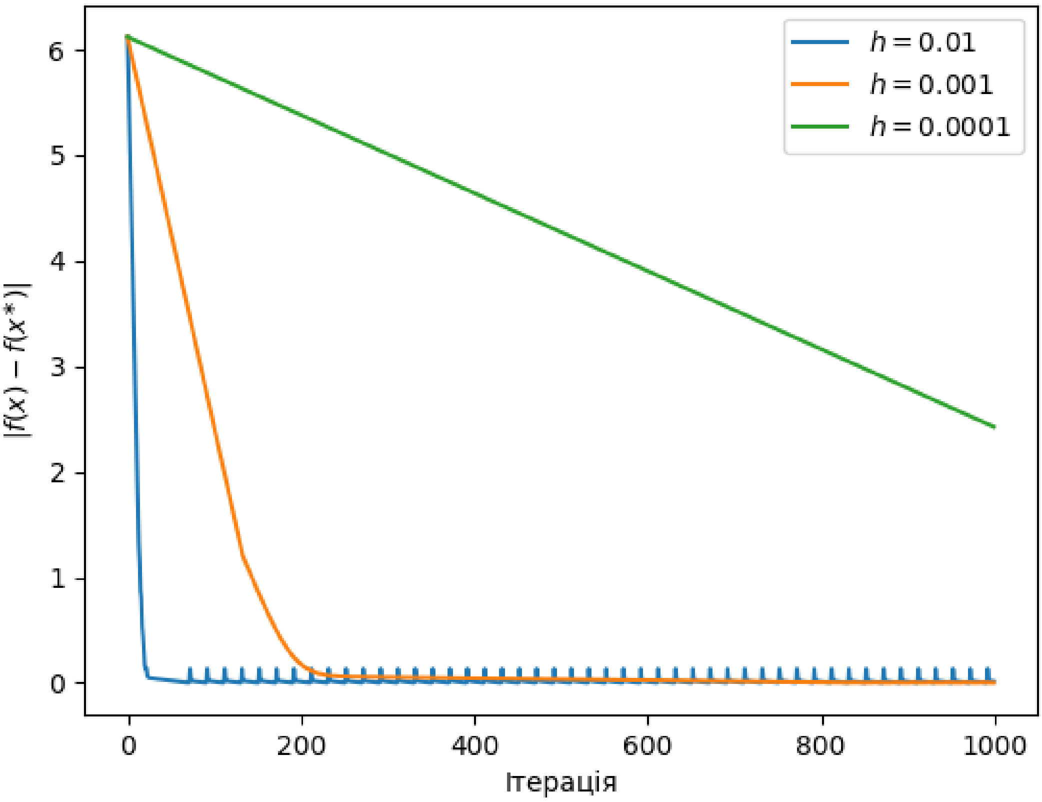


Рис. 4: Постійна довжина кроку

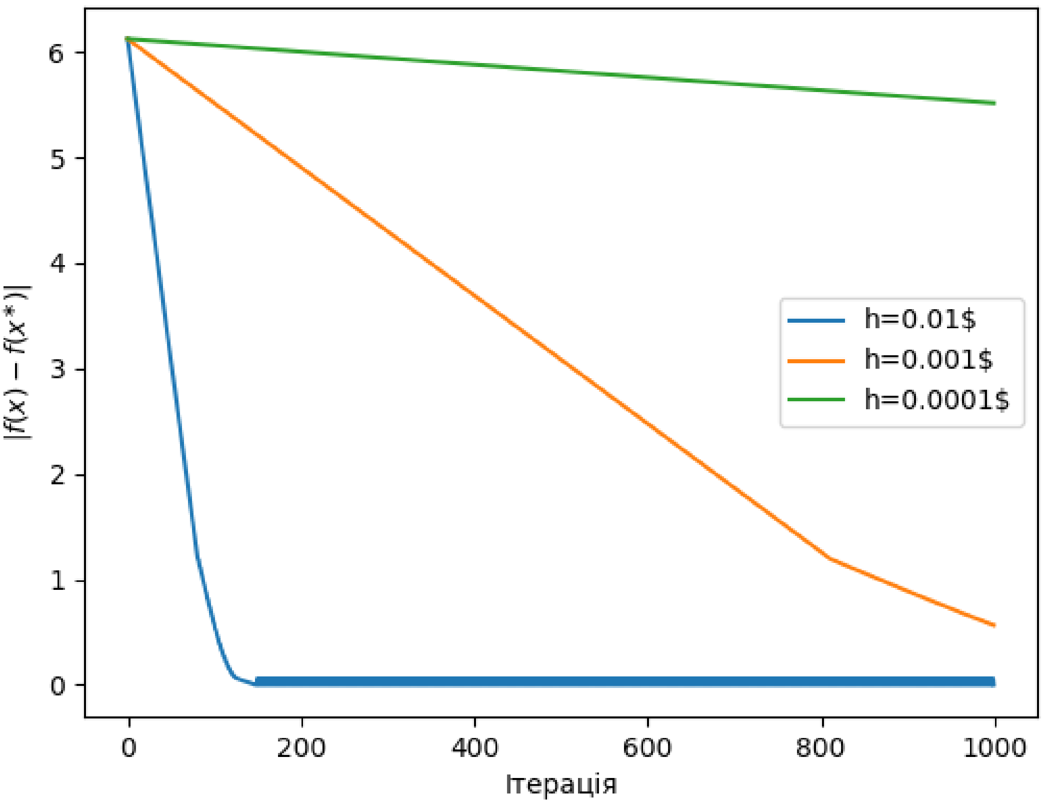


Рис. 5: Сумовні з квадратом кроки

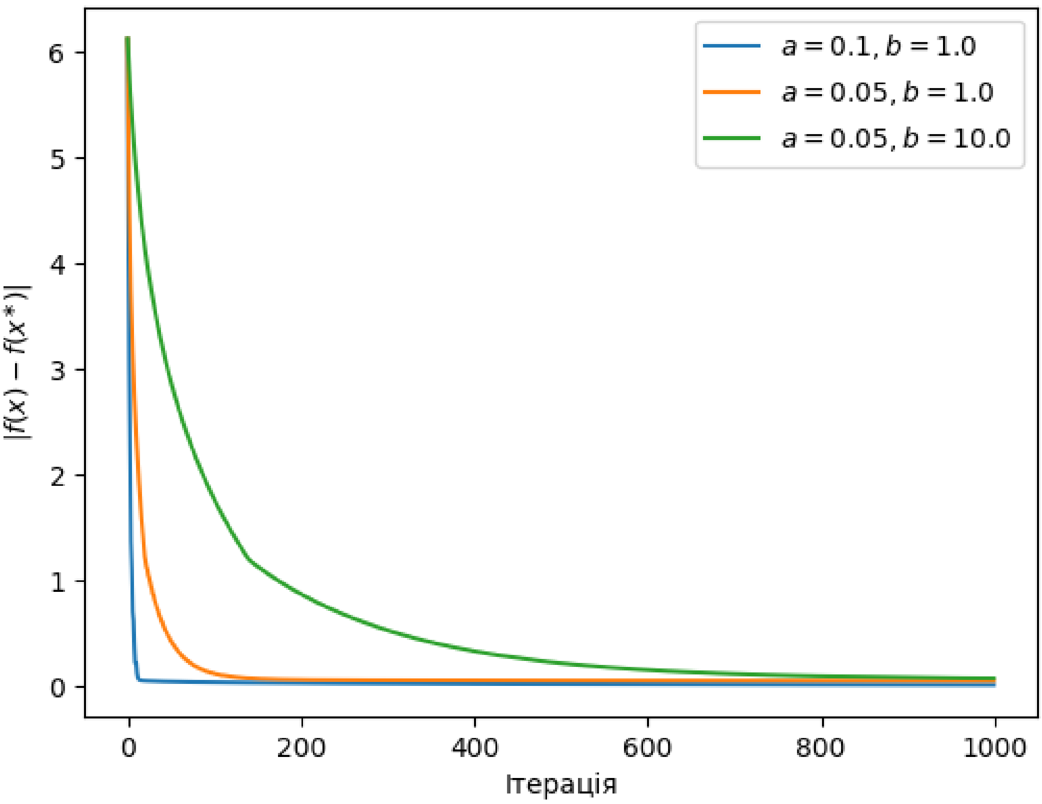


Рис. 6: Кроки, що зменшуються