Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №2

з дисципліни «Методи Оптимізації та Дослідження Операцій» на тему:

«Одновимірний пошук оптимуму, Методи оптимізації з виключенням інтервалів»

Виконав

Студент О. А. Онищенко

Прийняли

Викладач Л. Ю. Дейнега

2024

ОДНОВИМІРНИЙ ПОШУК ОПТИМУМУ, МЕТОДИ ОПТИМІЗАЦІЇ З ВИКЛЮЧЕННЯМ ІНТЕРВАЛІВ

Мета роботи

Вивчити одновимірні методи оптимізації з виключенням інтервалів; навчитися застосовувати методи оптимізації для аналізу й обробки інформації.

Постановка задачі

Побудувати графік заданої функції на заданому інтервалі за допомогою пакету matplotlib. Розробити програмну реалізацію процедури зменшення інтервалу пошуку з використанням обох вивчених методів одновимірного пошуку.

Функція:

$$(x-2)^2$$

Інтервал: $[-1, 3]$

Результати виконання

Під час розробки цієї програми було використано два методи одновимірного пошуку оптимуму – метод золотого перетину та метод поділу інтервалу навпіл.

Метод золотого перетину потроху з кожним кроком підходить все ближче до оптимуму функції за рахунок відкидання інтервалів, де значення функції є більшим. В нашому випадку саме це і треба, бо функція унімодальна і має один екстремум, який є її мінімумом на відрізку

[-1, 3]. Цей метод на кожному кроці створює інтервал $[c, d]$, який розташовується між двома границями першопочаткового інтервалу $[a, b]$, а позиції точок c та d визначаються за рахунок використання значення золотого перетину $\frac{\sqrt{5}-1}{2}$. Тоді метод обраховує значення функції в точках c та d, і порівнює їх. В залежності від того, яке зі значень є меншим, метод відкидає або інтервал $[a, c]$ і встановлює значення a на місце c, або інтервал $[d, b]$ та встановлює значення b на місце d.

Метод поділу інтервалу навпіл працює дещо схоже на алгоритм бінарного пошуку. На кожному кроці метод розраховує середню точку у інтервалі $[a, b]$. Після цього перевіряє чи знаходиться оптимум у лівій частині інтервалу, себто зліва від середньої точки c, чи у правій частині інтервалу, або справа від точки c. В залежності від цього порівняння метод відкидає або ліву, або праву частину інтервалу.

Обидва методи завершують свою роботу коли похибка стає достатньо малою для того, аби припустити що знайдене значення і є нашим результатом – оптимумом зазначеної функції $(x - 2)^2$.

Для побудови всіх графіків було використано бібліотеку matplotlib. Серед графіків - графік первісної функції на відрізку [-1, 3] з умови задачі, графік візуалізації методу золотого перетину та графік візуалізації методу поділу інтервалу навпіл.

При отриманні результатів двома вищезгаданими методами ми також використали метод scipy.optimize, який називається optimize.minimize_scalar. Цей метод використовує алгоритм обмеженої оптимізації для мінімізації одновимірних функцій в заданих межах [-1, 3], як передбачено в умові задачі.

Додатковий метод від Sciру було використано для гарантування точності результатів, отриманих з наших двох ручних функцій, і для знаходження правильної відповіді.

Програма виводить на екран всі значення, отримані за допомогою методів, використаних для отримання оптимуму функції, і порівнює їх. Якщо результати збігаються, виводиться повідомлення про успіх. В іншому випадку виводиться повідомлення про помилку.

Код програми

```
from rich.console import Console
from rich.traceback import install

import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt

install()
console = Console()


def f(x: float) -> float:
    return (x - 2) ** 2


def plotFunction(f, a: float, b: float, title: str) -> None:
    x = np.linspace(a, b, 1000)
    plt.figure(figsize=(10, 6))
    plt.plot(x, f(x), label="f(x)")
    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.legend()
    plt.plot([a, b], [f(a), f(b)], "r--", label="Initial Interval")


def goldenSearch(
    f: callable = f, a: float = -1, b: float = 3, tol: float = 1e-5
) -> float:
    ratio = (5**0.5 - 1) / 2
    c = b - ratio * (b - a)
    d = a + ratio * (b - a)

    plotFunction(f, a, b, "Golden Search Method Visualization")

    while abs(c - d) > tol:
        if f(c) < f(d):
```

```
            b = d
        else:
            a = c
        c = b - ratio * (b - a)
        d = a + ratio * (b - a)

        plt.plot([a, b], [f(a), f(b)], "g--", label="Current Interval")
        plt.pause(0.01)

    plt.plot([a, b], [f(a), f(b)], "b--", label="Final Interval")
    plt.show()
    return (a + b) / 2


def bisectionSearch(
    f: callable = f, a: float = -1, b: float = 3, tol: float = 1e-5
) -> float:
    plotFunction(f, a, b, "Bisection Search Method Visualization")
    L = b - a

    while L > tol:
        x1 = a + L / 4
        xm = (a + b) / 2
        x2 = b - L / 4

        if f(x1) > f(xm):
            if f(xm) < f(x2):
                a = x1
                b = x2
            else:
                a = xm
        else:
            b = xm
        L = b - a

        plt.plot([a, b], [f(a), f(b)], "g--", label="Current Interval")
        plt.pause(0.01)

    plt.plot([a, b], [f(a), f(b)], "b--", label="Final Interval")
    plt.show()

    return (b + a) / 2


def main() -> None:
    with console.status("Optimizing...", spinner="point"):
        resGolden: float = f"{goldenSearch():.2f}"
        resBisection: float = f"{bisectionSearch():.2f}"
```

```
        scalarBounded: float = f"{optimize.minimize_scalar(f, bounds=(-1,
3)).x:.2f}"

    console.print(f"Golden Section Method: {resGolden}")
    console.print(f"Bisection Method: {resBisection}")
    console.print(f"Bounded Scalar (from scipy): {scalarBounded}")
    console.print()
    console.print(
        "[green bold]Correct answer found![/green bold]"
        if resGolden == resBisection == scalarBounded
        else "[red bold]Doesn't match![/red bold]"
    )

    plt.show()


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

Результати роботи програми

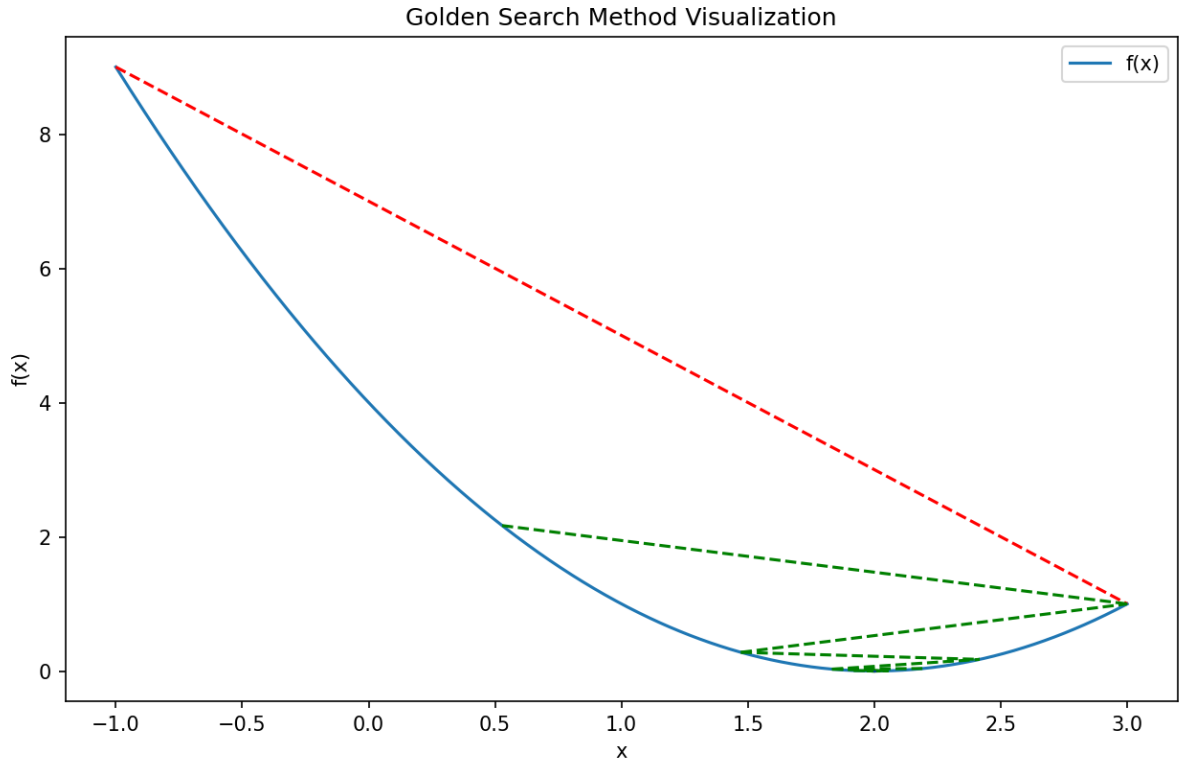

Рисунок 1.1 – Візуалізації роботи методу золотого перетину

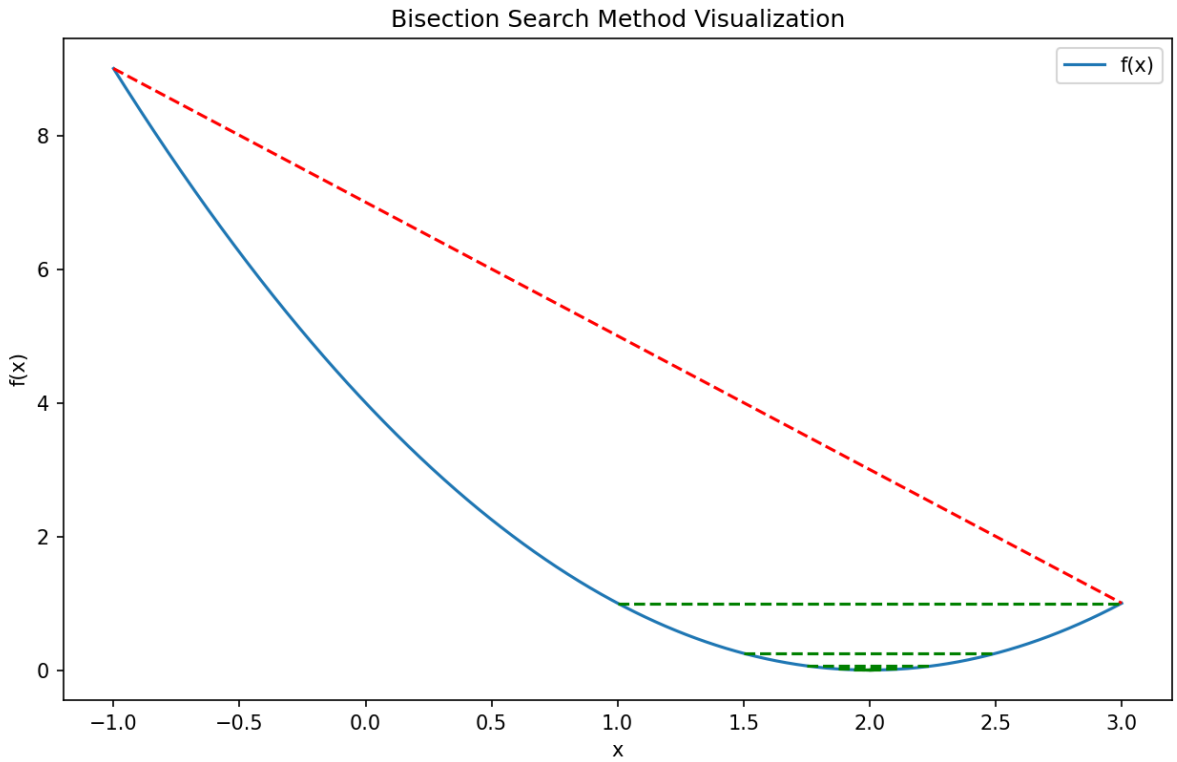

Рисунок 1.2 – Візуалізація роботи методу поділу інтервалу навпіл

```
Golden Section Method: 2.00
Bisection Method: 2.00
Bounded Scalar (from scipy): 2.00

Correct answer found!
```

Рисунок 1.3 – Результати розрахунку оптимумів по завершенню роботи програми

Висновки

Таким чином, ми вивчили одновимірні методи оптимізації з виключенням інтервалів, а також навчилися застосовувати методи оптимізації для аналізу й обробки інформації.

Контрольні питання

У чому полягають питання аналізу "у статиці" і "в динаміці", що виникають при аналізі оптимізаційних завдань?

Аналіз оптимізаційних задач як у статиці, так і в динаміці пов'язаний з кількома аспектами:

Складність: Складність проблеми зростає зі збільшенням кількості цілей, обмежень та змінних. Ця складність може спричинити труднощі в пошуку оптимальних рішень.

Невизначеність: Реальні програми часто пов'язані з невизначеностями, які можуть впливати на процес оптимізації. Ці невизначеності можуть виникати з різних чинників, таких як похибки вимірювань, апроксимації моделей і непередбачувані фактори навколишнього середовища.

Обчислювальне навантаження: Обчислювальне навантаження може бути значним, особливо для масштабних задач або таких, що вимагають високого ступеня точності.

Формулювання проблеми: Правильне формулювання проблеми і розробка відповідної системи прийняття рішень може бути складним завданням. Це включає належне визначення цільової функції, обмежень та змінних рішення.

У чому полягають необхідні умови того, що дана точка є точкою локального мінімуму (максимуму)?

Точка є локальним мінімумом (максимумом), якщо вона задовольняє наступним умовам:

Перша похідна функції в цій точці дорівнює нулю (тобто точка є критичною).

Друга похідна функції в цій точці додатна (від'ємна) для локального мінімуму (максимуму).

Якщо функція визначена на проміжку, то точка повинна знаходитись в межах цього проміжку.

Сформулюйте достатні умови оптимальності.

Умови Каруша-Куна-Таккера (ККТ) забезпечують достатні умови для того, щоб розв'язок був оптимальним в задачі нелінійного програмування. Ці умови є наступними:

Первинна здійсненність: Розв'язок повинен задовольняти всім обмеженням задачі.

Подвійна здійсненність: Множники Лагранжа, пов'язані з обмеженнями нерівностей, повинні бути невід'ємними.

Комплементарна слабкість: Добуток кожної нерівності на відповідний множник Лагранжа повинен дорівнювати нулю.

Стаціонарність: Градієнт функції Лагранжа, як по відношенню до змінних рішення, так і по відношенню до множників Лагранжа, повинен дорівнювати нулю.

Ці умови узагальнюють метод множників Лагранжа, дозволяючи враховувати як рівність, так і нерівність обмежень. Якщо ці умови виконуються, розв'язок вважається оптимальним.