Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №4

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

«Методи оптимізації з використанням похідних»

Виконав

Студент групи КНТ-122 О. А. Онищенко

Прийняли

Викладач Л. Ю. Дейнега

2024

МЕТОДИ ОПТИМІЗАЦІЇ З ВИКОРИСТАННЯМ ПОХІДНИХ

Мета роботи

Вивчити одновимірні методи оптимізації з використанням похідних

Постановка задачі

Розробити програмну реалізацію одновимірних методів оптимізації з використанням похідних

Функція:

$$(x-2)^2$$

Інтервал: $[-1,3]$

Результати виконання

Код програми

```
from rich.console import Console
from scipy.misc import derivative
from rich.traceback import install
from scipy.optimize import minimize_scalar

install()
console = Console()


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


def derivative(f=f, x=0, step=0.0001):
    return (f(x + step) - f(x)) / step
```

```
def newton(f=f, start=-1, end=3, step=0.1, epsilon=1e-6):
    X_N = start
    while 1:
        F_Prime = derivative(f, X_N)
        F_Double_Prime = derivative(lambda x: derivative(f, x), X_N)

        if abs(F_Prime) < epsilon:
            break
        if F_Double_Prime == 0:
            break

        X_N1 = X_N - F_Prime / F_Double_Prime
        if abs(X_N1 - X_N) < epsilon:
            break
        X_N = X_N1
    return X_N


def bolzani(f=f, start=-1, end=3, step=0.1, epsilon=1e-6):
    Midpoint = (start + end) / 2
    while (end - start) > epsilon:
        Midpoint = (start + end) / 2
        Left_Derivative = derivative(f, start)
        Right_Derivative = derivative(f, end)
        Mid_Derivative = derivative(f, Midpoint)

        if abs(Mid_Derivative) < epsilon:
            return Midpoint

        if Left_Derivative > 0 and Mid_Derivative < 0:
            end = Midpoint
        elif Mid_Derivative > 0 and Right_Derivative < 0:
            start = Midpoint
        else:
            if abs(Left_Derivative) < abs(Right_Derivative):
                end = Midpoint
            else:
                start = Midpoint
    return Midpoint


def chord(f=f, start=-1, end=3, step=0.1, epsilon=1e-6):
    x_n = (start + end) / 2
    while abs(f(x_n)) > epsilon:
        derivative_value = derivative(f, x_n)
        if derivative_value == 0:
            break
        x_n = x_n - f(x_n) / derivative_value
    return x_n
```

```
def cube(f=f, start=-1, end=3, step=0.1, epsilon=1e-6):
    import numpy as np

    x_values = np.array([start, end])
    y_values = np.array([f(start), f(end)])
    derivative_at_start = derivative(f, start)

    A = np.array([[start**2, start, 1], [end**2, end, 1], [2 * start, 1,
0]])
    b = np.array([f(start), f(end), derivative_at_start])

    coeffs = np.linalg.solve(A, b)

    a, b, _ = coeffs
    vertex_x = -b / (2 * a)

    if start <= vertex_x <= end:
        x_star = vertex_x
    else:
        x_star = start if f(start) < f(end) else end

    return x_star


def getRes(f=f, start=-1, end=3, step=0.1, epsilon=1e-6):
    methods = {
        "Ньютона-Рафсона": {"function": newton, "result": None},
        "Больцано": {"function": bolzani, "result": None},
        "Січних": {"function": chord, "result": None},
        "Кубічної апроксимації": {"function": cube, "result": None},
    }

    for name, method in methods.items():
        method["result"] = method["function"](f, start, end, step,
epsilon)
        console.print(f"[bold]{name}[/]: {method['result']:.2f}")

    return methods


def checkCorrect(results):
    correct = minimize_scalar(f, bounds=(-1, 3)).x

    for name, method in results.items():
        if f"{method['result']:.2f}" == f"{correct:.2f}":
            console.print(
```

```
                f"[green bold]Знайдено правильну відповідь![/green bold]
{name}"
            )
        else:
            console.print(f"[red bold]Не збігається![/red bold] {name}")


def main() -> None:
    console.print()
    res = getRes()
    console.print()
    checkCorrect(res)
    console.print()


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

Результати роботи програми

```
Ньютона-Рафсона: 2.00
Больцано: 2.00
Січних: 2.00
Кубічної апроксимації: 2.00

Знайдено правильну відповідь! Ньютона-Рафсона
Знайдено правильну відповідь! Больцано
Знайдено правильну відповідь! Січних
Знайдено правильну відповідь! Кубічної апроксимації
```

Рисунок 1.1 – Результати розрахунку мінімумів по завершенню роботи програми

Висновки

Таким чином, ми вивчили одновимірні методи оптимізації з використанням похідних

Контрольні питання

Розглянуті раніше методи оптимізації засновані на припущенні про унімодальність й у ряді випадків про безперервність досліджуваної цільової функції. Виконання якої додаткової умови необхідно для реалізації методів оптимізації з використанням похідних?

Умовою, необхідною для реалізації методів оптимізації з використанням похідних, є припущення про диференційованість. Похідна цільової функції повинна існувати і бути неперервною в точці, що нас цікавить, аби ці методи працювали коректно.

Опишіть пошукову процедуру, що реалізує метод Ньютона-Рафсона

Метод Ньютона-Рафсона - це процедура пошуку, яка реалізує використання числових похідних для знаходження коренів функції.

Процедура виглядає наступним чином:

1. Вибираємо початкове значення кореня x_n.

2. Обчислити похідну функції в точці $x_n, f'(x_n)$.

3. Якщо $f'(x_n)$ дорівнює нулю, завершити цикл.

4. Обчислити наступне наближення для кореня, $x_{n+1} = x_n - f(x_n)/f'(x_n)$.

5. Якщо зміна x від попередньої ітерації мала, то завершити цикл.

6. Встановити x_n у значення x_{n+1} і перейти до кроку 2.

Опишіть схему пошуку з використанням методу середньої точки (пошуку Больцано)

Метод середньої точки, також відомий як пошук Больцано, - це алгоритм пошуку, який передбачає багаторазове розбиття поточного інтервалу на частини, поки не буде знайдено корінь.

Процедура виглядає наступним чином:

1. Вибрати початковий інтервал $[a, b]$, який містить корінь.
2. Обчислити медіану інтервалу, $m = (a + b)/2$.
3. Якщо $f(m) = 0$, то завершити цикл.
4. Якщо знак $f(a)$ відрізняється від знаку $f(m)$, то встановити $b = m$. Інакше встановити $a = m$.
5. Якщо інтервал скоротився до певного допустимого значення, завершити роботу циклу.
6. Встановити нову медіану як m і перейти до кроку 2.