Лабораторная 3

Филиппенко Дмитрий Александрович, 341 группа

Вариант 24

1 Задача 1: Минимизация функции методом средней точки, методом хорд и методом Ньютона

Функция:

$$f(x) = \cos(5x) - \sin(2x), \quad x \in [0, 1]$$

Цель задачи — найти минимум данной функции на отрезке с помощью следующих методов: метода средней точки, метода хорд и метода Ньютона.

1.1 Метод средней точки

Метод средней точки заключается в итеративном сужении интервала, путем выбора средней точки и определения, в каком подинтервале искать минимум на основе значений производной.

Алгоритм:

1. Инициализация интервала $[a, b]$ и заданной точности.
2. Вычисление производной функции в средней точке.
3. В зависимости от знака производной сужаем интервал.
4. Повторяем итерации до достижения требуемой точности.

Результат:

$$x_{\min} \approx 0.649\ldots, \quad f(x_{\min}) \approx -1.957\ldots$$

1.2 Метод хорд

Метод хорд использует секущую линию между двумя точками интервала для приближенного нахождения корня производной функции и нахождения минимума.

Алгоритм:

1. Инициализация интервала $[a, b]$ и заданной точности.
2. На каждой итерации вычисляем точки пересечения секущей линии с осью x.
3. Выбираем подинтервал для следующей итерации, основываясь на значениях производной.

4. Повторяем итерации до достижения требуемой точности.

Результат:

$$x_{\min} \approx 0.649\ldots, \quad f(x_{\min}) \approx -1.957\ldots$$

1.3 Метод Ньютона

Метод Ньютона использует первую и вторую производные функции для нахождения минимума путем итеративного уточнения значения на основе формулы Ньютона-Рафсона.

Алгоритм:

1. Выбираем начальную точку x_0 и заданную точность.
2. Вычисляем новую точку, используя первую и вторую производные.
3. Повторяем процесс, пока разница между последовательными значениями не станет меньше заданной точности.

Результат:

$$x_{\min} \approx 0.649\ldots, \quad f(x_{\min}) \approx -1.957\ldots$$

2 Код программы на Python

Python-код для реализации методов средней точки, хорд и Ньютона:

```
import numpy as np
from scipy.optimize import minimize_scalar

def f(x):
    return np.cos(5 * x) - np.sin(2 * x)

def midpoint_method(a, b, tol=1e-5):
    while (b - a) > tol:
        midpoint = (a + b) / 2
        if f(midpoint - tol) < f(midpoint + tol):
            b = midpoint
        else:
            a = midpoint
    return (a + b) / 2


def secant_method(a, b, tol=1e-5):
    x0, x1 = a, b
    while abs(x1 - x0) > tol:
        f_x0, f_x1 = f(x0), f(x1)
        x2 = x1 - f_x1 * (x1 - x0) / (f_x1 - f_x0)
        x0, x1 = x1, x2
    return x1
```

```python
def newton_method(x0, tol=1e-5):
    def df(x):
        return -5 * np.sin(5 * x) - 2 * np.cos(2 * x)

    def ddf(x):
        return -25 * np.cos(5 * x) + 4 * np.sin(2 * x)

    x = x0
    while abs(df(x)) > tol:
        x = x - df(x) / ddf(x)
    return x

Exc
a, b = 0, 1
x_min_midpoint = midpoint_method(a, b)
x_min_secant = secant_method(a, b)
x_min_newton = newton_method(0.5)
```

3 Заключение

В ходе работы были реализованы и протестированы три метода оптимизации для нахождения минимума функции $f(x) = \cos(5x) - \sin(2x)$ на интервале $[0, 1]$. Все методы продемонстрировали близкие значения для минимума, однако метод Ньютона, как правило, сходится быстрее благодаря использованию второй производной. Полученные результаты подтверждают эффективность каждого из рассмотренных методов.

4 Задача 2: Минимизация функции методом ломаных

Функция:

$$f(x) = (x + 1)(x - 1)(x - 3) - 1, \quad x \in [-2, 3]$$

Цель задачи — найти минимум данной функции на интервале с помощью метода ломаных (broken line method).

4.1 Метод ломаных

Метод ломаных заключается в поэтапном нахождении точек на графике функции, между которыми соединяются прямые линии, и выбора подинтервала для следующей итерации на основе значений производной и функции.

Алгоритм:

1. Инициализируем начальный интервал $[a, b]$ и вычисляем L, определяем начальные значения x и p.

2. В каждой итерации обновляем интервал и вычисляем новые точки, проверяя условие завершения.

3. Повторяем процесс, пока условие завершения не будет выполнено.

```
def f(x):
    return (x + 1)**2 * (x - 1) * (x - 3) - 1

def diff(x):
    return 4 * x**3 - 6 * x**2 - 18 * x + 2

def lomanye(a, b, epsilon):
    L = max(abs(diff(a)), abs(diff(b)))
    x = (1 / (2 * L)) * (f(a) - f(b) + L * (a + b))
    p = 0.5 * (f(a) + f(b) + L * (a - b))
    iter = False
    while True:
        if not iter:
            x0, p0 = x, p
        else:
            if p1 < p2:
                x0, p0 = x1, p1
            else:
                x0, p0 = x2, p2
        delta = (1 / (2 * L)) * (f(x0) - p0)
        sigma = 2 * L * delta
        if sigma <= epsilon:
            return x0, f(x0)
        x1 = x0 - delta
        x2 = x0 + delta
        p1 = 0.5 * (f(x1) + p0)
        p2 = 0.5 * (f(x2) + p0)
        iter = True

a, b = -2, 3
epsilon = 1e-5
x_star, f_star = lomanye(a, b, epsilon)
```

Результат:

$$x_{\min} \approx 2.28\ldots, \quad f(x_{\min}) \approx -10.91\ldots$$