Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет ПИ и КТ

OptMethods edition

Лабораторная работа №2

по дисциплине: «Методы оптимизации»

«Одномерная оптимизация»

Вариант 1

Выполнил:

Болорболд Аригуун,

группа P3211

Преподаватель:

Селина Елена Георгиевна

Санкт-Петербург

2024

Задание:

Решить задачу четырьмя методами: методом половинного деления, методом золотого сечения, методом хорд и методом Ньютона. По 5 шагов каждого метода выполнить вручную + написать программу по каждому методу на одном из языков программирования.

1. $f(x) = x^2 - 3x + x\ln x\,, [a, b] = [1, 2], \varepsilon = 0{,}05$

Решение:

1. Метод половинного деления:

№	a	b	x_1	x_2	y_1	y_2	|b – a|
1	1	2	1,475	1,525	-1,6761	-1,6058	1
2	1	1,525	1,2375	1,2875	-1,9174	-1,8795	0,525
3	1	1,2875	1,11875	1,16875	-1,9791	-1,9580	0,2875
4	1	1,16875	1,059375	1,109375	-1,9947	-1,9822	0,16875
5	1	1,109375	1,0296875	1,0796875	-1,9987	-1,9905	0,109375

2. Метод золотого сечения:

№	a	b	x_1	x_2	y_1	y_2	|b – a|
1	1	2	1,382	1,618	-1,7889	-1,4575	1
2	1	1,618	1,236	1,382	-1,9184	-1,7889	0,618
3	1	1,382	1,146	1,236	-1,9685	-1,9184	0,382
4	1	1,236	1,09	1,146	-1,9879	-1,9685	0,23608
5	1	1,146	1,056	1,09	-1,9953	-1,9879	0,146

3. Метод хорд:

№	a	b	f’(a)	f’(b)	~x	|f’(~x)|
1	1	2	-1	1,69314	1,371313	0,058394
2	1	1,371313	-1	0,058394	1,350826	0,0023698

4. Метод Ньютона:

№	x	f’(x)	f’’(x)	|f’(x)|
1	1,5	1,40547	2,666665	1,40547
2	0,97295	-0,08152	3,0278014	0,08152
3	0,9998746	-0,000376	3,0001253	0,0003759

Исходный код решения:

```
from math import log
def derivative(x: float) -> float:
    return 2*x - 2 + log(x)
def derivative2(x: float) -> float:
    return 2 + 1 / x
def bisection_method(function, a, b, epsilon):
    i = 2
    while abs(b - a) > 2 * epsilon:
        i+=1
        x1, x2 = round((a + b - epsilon) / 2, ndigits=i), round((a + b +
epsilon) / 2, ndigits=i)
        y1, y2 = function(x1), function(x2)
        if y1 > y2:
            a = float(x1)
        else:
            b = float(x2)
        # print(f"Итерация №{i}: a = {a}, b = {b}, x1 = {x1}, x2 = {x2}, y1
= {y1}, y2 = {y2}, |b - a| = {round(abs(b - a), ndigits=i+3)}")
    xm = round((a + b) / 2, ndigits=i)
    ym = function(xm)
    return xm, ym
def golden_ratio_method(function, a, b, epsilon):
    i = 2
    x1, x2 = round(b - round((b - a) / 1.618, ndigits=3), ndigits=3),
round(a + round((b - a) / 1.618, ndigits=3), ndigits=3)
    y1, y2 = function(x1), function(x2)
    # print(f"Итерация №{1}: a = {a}, b = {b}, x1 = {x1}, x2 = {x2}, y1 =
{y1}, y2 = {y2}, |b - a| = {round(abs(b - a), ndigits=i+3)}")
    while abs(b - a) > epsilon:
        i += 1
        if y1 < y2:
            b = x2
            x2 = x1
            y2 = y1
            x1 = round(b - round((b - a) / 1.618, ndigits=3), ndigits=3)
            y1 = function(x1)
        else:
            a = x1
            x1 = x2
            y1 = y2
            x2 = round(a + round((b - a) / 1.618, ndigits=3), ndigits=3)
            y2 = function(x2)
        # print(f"Итерация №{i-1}: a = {a}, b = {b}, x1 = {x1}, x2 = {x2},
y1 = {y1}, y2 = {y2}, |b - a| = {round(abs(b - a), ndigits=i+3)}")
    xm = (a + b) / 2
    ym = function(xm)
    return xm, ym

def chord_method(function, a, b, epsilon):
    i = 2
    d_a = derivative(a)
    d_b = derivative(b)
    _x = a - (d_a / (d_a - d_b)) * (a - b)
    while abs(derivative(_x)) > epsilon:
        i += 1
        # print(f"Итерация №{i - 2}: a = {a}, b = {b}, f'(a) = {d_a}, f'(b)
= {d_b}, _x={_x}, |f'(_x)| = {derivative(_x)}")
        if derivative(_x) > 0:
            b = _x
```

```
        else:
            a = _x
        _x = a - (d_a / (d_a - d_b)) * (a - b)
    # print(f"Итерация №{i - 1}: a = {a}, b = {b}, f'(a) = {d_a}, f'(b) =
{d_b}, _x={_x}, |f'(_x)| = {derivative(_x)}")
    return _x, function(_x)

def newton_method(function, a, b, epsilon):
    i = 2
    x = round((a + b) / 2, ndigits=i)
    # print(f"Итерация №{1}: x = {x}, f'(x) = {derivative(x)}, f\"(x) =
{derivative2(x)}, |f'(x)| = {abs(derivative(x))}")
    while abs(derivative(x)) > epsilon:
        i += 1
        x = x - (derivative(x) / derivative2(x))
        # print(f"Итерация №{i - 1}: x = {x}, f'(x) = {derivative(x)},
f\"(x) = {derivative2(x)}, |f'(x)| = {abs(derivative(x))}")
    return x, function(x)
def main():
    function = lambda x: x**2 - 3 * x + x * log(x)
    a, b, epsilon = 1, 2, 20**-1
    golden_ratio_method(function, a, b, epsilon)
    result_list = [bisection_method(function, a, b, epsilon),
golden_ratio_method(function, a, b, epsilon), chord_method(function, a, b,
epsilon), newton_method(function, a, b, epsilon)]
    print("\033[2;31m" + "Лабораторная работа №2: одномерная оптимизация" +
"\033[0m")
    print("\033[1;35m" + "1. Метод половинного деления", "2. Метод золотого
сечения", "3. Метод хорд", "4. Метод Ньютона" + "\033[0m", sep="\n")
    match input("\033[1;32m" + "Введите номер метода (или что-то другое для
получения значений всех методов):\n" + "\033[0m"):
        case '1':
            xm, ym = result_list[0]
            print("\033[3;36m" + f"x_m = {xm}, y_m = {ym}" + "\033[0m")
        case '2':
            xm, ym = result_list[1]
            print("\033[3;36m" + f"x_m = {xm}, y_m = {ym}" + "\033[0m")
        case '3':
            xm, ym = result_list[2]
            print("\033[3;36m" + f"x_m = {xm}, y_m = {ym}" + "\033[0m")
        case '4':
            xm, ym = result_list[3]
            print("\033[3;36m" + f"x_m = {xm}, y_m = {ym}" + "\033[0m")
        case _:
            for i in range(len(result_list)):
                xm, ym = result_list[i]
                print("\033[3;36m" + f"{i + 1}. x_m = {xm}, y_m = {ym}" +
"\033[0m")

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