МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРОНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

«Методы оптимизации»

Вариант № 13

Выполнил работу:

Студент группы P3218

Рамеев Тимур Ильгизович

Преподаватель:

Кудашов Вячеслав Николаевич

Санкт-Петербург 2024

Содержание

Содержание 2
Задание 3
Исходная функция 3
Интервал 3
График 3
Исходный код реализованных методов 4
Метод половинного деления 4
Метод Ньютона 4
Метод золотого сечения 5
Вывод программы 6

Задание

Найти минимум функции на заданном интервале. Решить задачу методом половинного деления, методом золотого сечения и методом Ньютона; написать программу на языке Python. Параметры ε и δ принять равными 10^{-10}. Программа должна выполнить 25 итераций или остановиться при достижении критерия остановки итерационного процесса. Осуществлять вывод вычисленных значений на каждой итерации с точностью до 5 знаков после запятой.

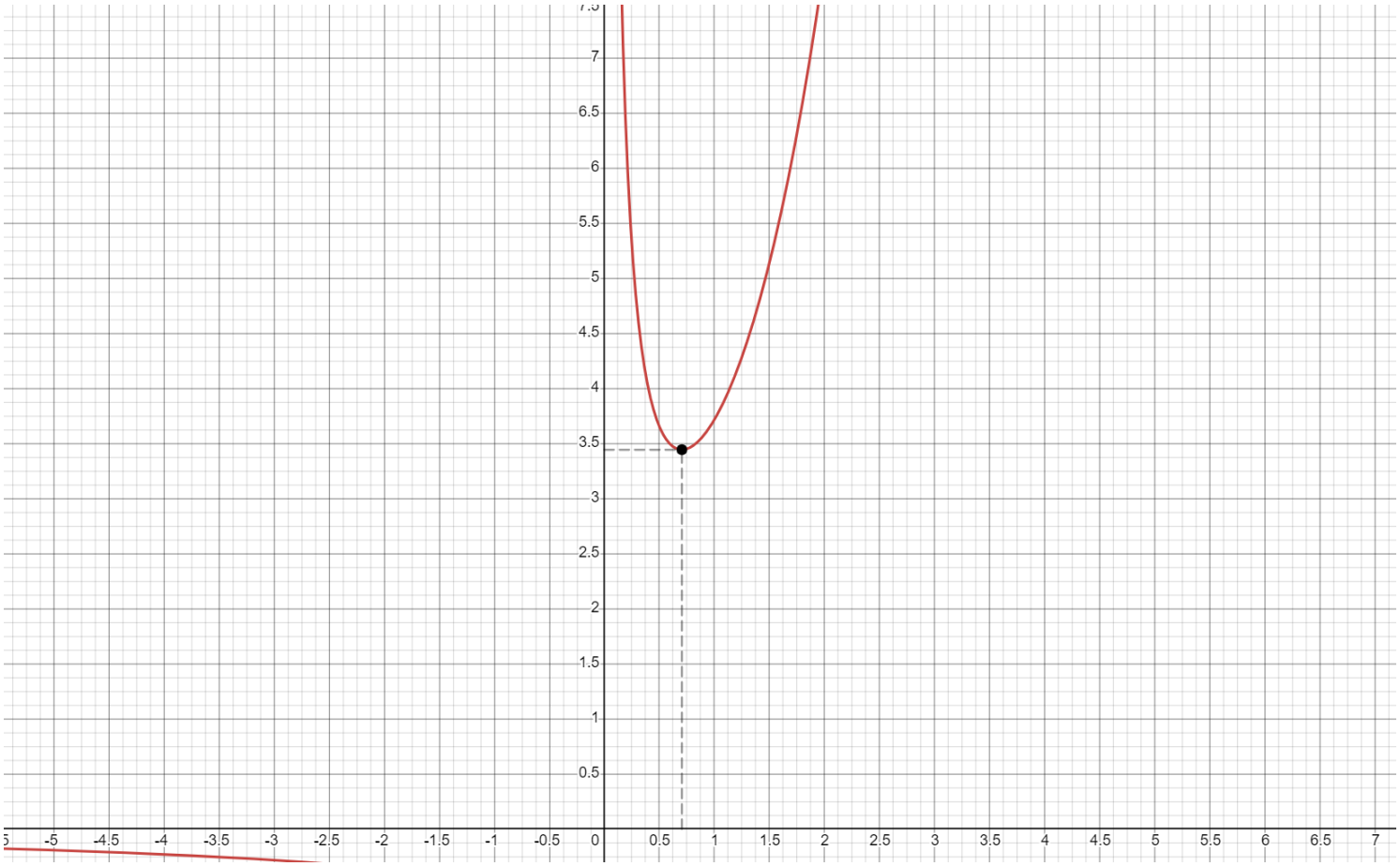
Исходная функция

$$f(x) = \frac{1}{x} + e^x$$

Интервал

$$[a, b] = [0.5, 1.5]$$

График



Исходный код реализованных методов

Метод половинного деления

```
def dividing_segment_in_half(rounding):
    print("Метод половинного деления:")
    current_a = a
    current_b = b
    counter = 0
    while current_b - current_a > 2 * eps and counter < 25:
        counter += 1
        x1 = (current_a + current_b - eps) / 2
        x2 = (current_a + current_b + eps) / 2
        y1 = function_value(x1)
        y2 = function_value(x2)
        if y1 > y2:
            current_a = x1
        else:
            current_b = x2
            curr_val = (current_a + current_b) / 2
        print(f'{counter}: x={round(curr_val, rounding)}\tf(x)={round(function_value(curr_val),
rounding)}')
    ret_val = (current_a + current_b) / 2
    print_result(ret_val, function_value(ret_val))
```

Метод Ньютона

```
def newton_method(rounding):
    print("Метод Ньютона:")
    current_approximation = (a + b) / 2
    counter = 0
    while abs(first_derivative_value(current_approximation)) > eps and counter < 25:
        counter += 1
        current_approximation = current_approximation -
first_derivative_value(current_approximation) / second_derivative_value(current_approximation)
        print(f'{counter}: x={round(current_approximation,
rounding)}\tf(x)={round(function_value(current_approximation), rounding)}')
    print_result(current_approximation, function_value(current_approximation))
```

Метод золотого сечения

```
def golden_ratio(rounding):
    print("Метод золотого сечения:")
    golden = (1 + 5 ** 0.5) / 2
    current_a = a
    current_b = b
    y1 = function_value(a + (b - a) * (1 - 1 / golden))
    y2 = function_value(a + (b - a) * (1 / golden))
    counter = 0
    while  current_b - current_a > 2 * eps and counter < 25:
        counter += 1
        if y1 < y2:
            current_b = current_a + (current_b - current_a) * (1 / golden)
            y2 = y1
            y1 = function_value(current_a + (current_b - current_a) * (1 - 1 / golden))
        else:
            current_a = current_a + (current_b - current_a) * (1 - 1 / golden)
            y1 = y2
            y2 = function_value(current_a + (current_b - current_a) * (1 / golden))
        curr_val = (current_a + current_b) / 2
        print(f'{counter}: x={round(curr_val, rounding)}\tf(x)={round(function_value(curr_val),
rounding)}')
    ret_val = (current_a + current_b) / 2
    print_result(ret_val, function_value(ret_val))
```

Вывод программы

	Метод половинного деления		Метод золотого сечения		Метод Ньютона	
№ итерации	x	f(x)	x	f(x)	x	f(x)
1	0.75	3.45033	0.80902	3.48177	0.53788	3.57152
2	0.625	3.46825	0.69098	3.44289	0.75218	3.45109
3	0.625	3.46825	0.76393	3.45572	0.67298	3.446
4	0.625	3.46825	0.71885	3.44318	0.72028	3.44336
5	0.70313	3.44228	0.69098	3.44289	0.69333	3.44268
6	0.70313	3.44228	0.7082	3.44236	0.70931	3.44241
7	0.70703	3.44233	0.69756	3.44241	0.7	3.44232
8	0.70508	3.44229	0.70414	3.44228	0.70549	3.44229
9	0.7041	3.44228	0.70007	3.44232	0.70228	3.44228
10	0.70361	3.44228	0.70259	3.44228	0.70416	3.44228
11	0.70337	3.44228	0.70414	3.44228	0.70306	3.44228
12	0.70337	3.44228	0.70318	3.44228	0.70371	3.44228
13	0.70343	3.44228	0.70377	3.44228	0.70333	3.44228
14	0.70343	3.44228	0.70341	3.44228	0.70355	3.44228
15	0.70343	3.44228	0.70363	3.44228	0.70342	3.44228
16	0.70347	3.44228	0.70349	3.44228	0.7035	3.44228
17	0.70346	3.44228	0.70341	3.44228	0.70345	3.44228
18	0.70346	3.44228	0.70346	3.44228	0.70348	3.44228
19	0.70347	3.44228	0.70349	3.44228	0.70346	3.44228
20	0.70347	3.44228	0.70347	3.44228	0.70347	3.44228
21	0.70347	3.44228	0.70346	3.44228	0.70347	3.44228
22	0.70347	3.44228	0.70347	3.44228	0.70347	3.44228
23	0.70347	3.44228	0.70347	3.44228	0.70347	3.44228
24	0.70347	3.44228	0.70347	3.44228	0.70347	3.44228
25	0.70347	3.44228	0.70347	3.44228	0.70347	3.44228
$x_к$	0.7034664005337656		0.7034667702091327		0.7034671860632785	
$F(x_к)$	3.4422772944990294		3.4422772944990294		3.4422772944951916	