

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе «Решение системы линейных алгебраических уравнений СЛАУ»
по дисциплине «**Вычислительная математика**»

Автор: Билошицкий Михаил Владимирович

Факультет: ПИиКТ

Группа: Р3126

Преподаватель: Малышева Татьяна Алексеевна



Санкт-Петербург, 2023

Содержание

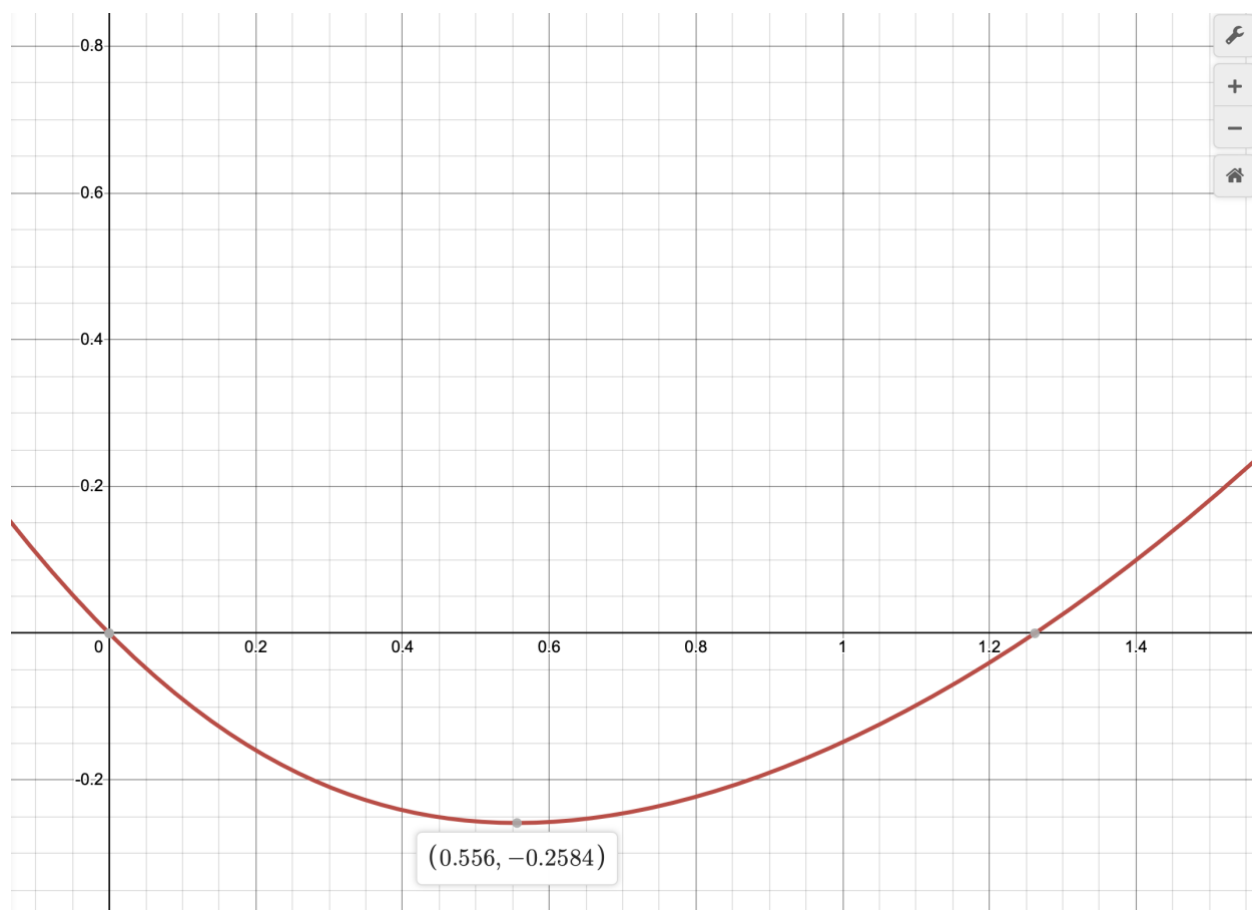
Содержание	2
График	3
Листинг программы.....	3
Примеры и результаты работы программы	7

Задание

Решить задачу тремя методами: методом половинного деления, методом золотого сечения, и методом Ньютона. Написать программу на языке Python, которая выполняет 25 шагов каждого метода.

$$2, f(x) = \ln(1+x^2) - \sin x, [a, b] = \left[0, \frac{\pi}{4}\right], \epsilon = 0.03;$$

График



Листинг программы

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

```
from equation_result import Result
```

```
class Equation:
```

```
    def __init__(self, f, a, b, eps, max_iterations=5000) -> None:
```

```
        self.f = f
```

```
        self.a = a
```

```
        self.b = b
```

```
        self.eps = eps
```

```
        self.max_iterations = max_iterations
```

```
    # Метод библиотеки scipy
```

```
    def python_method(self) -> Result:
```

```
        method_name = "Библиотека Python"
```

```
        try:
```

```
            result = root_scalar(self.first_derivative, bracket=[self.a, self.b],  
xtol=self.eps)
```

```
            if result.converged: return Result(result.root, result.iterations,  
method_name, self)
```

```
            else: return None
```

```
        except ValueError as e:
```

```
            return Result(success=False, message=str(e), method_name=method_name,  
equation=self)
```

```
    # Метод половинного деления
```

```
    def bisection_method(self) -> Result:
```

```
        method_name = "Половинного деления"
```

```
        table = [["#", "a", "b"]]
```

```
        a, b = self.a, self.b
```

```
        f = self.first_derivative
```

```
        i = 0
```

```
        if f(a) * f(b) >= 0:
```

```
            return Result(
```

```
                success=False,
```

```
                message="Метод не применим. f(a) и f(b) Должны иметь разные  
знаки.",
```

```
                method_name=method_name,
```

```
                equation=self)
```

```

while (b - a) / 2.0 > self.eps and i < self.max_iterations:
    i += 1

    x0 = (a + b) / 2.0
    if f(x0) == 0:
        return x0
    elif f(a) * f(x0) < 0:
        b = x0
    else:
        a = x0

    table.append([i, a, b])

return Result((a + b) / 2.0, i, method_name, self, table)

```

Метод Ньютона

```

def newton_method(self) -> Result:
    method_name = "Ньютона"
    table = [["#", "f(x)", "f'(x)"]]

```

```

a, b = self.a, self.b
f = self.first_derivative
i = 0

```

```

if self.first_derivative(a) * self.first_derivative(b) >= 0:
    return Result(
        success=False,
        message="Метод не применим. f'(a) и f'(b) Должны иметь разные
знаки.",
        method_name=method_name,
        equation=self)

```

```

x0 = None

```

```

while abs(f(a)) > self.eps and i < self.max_iterations:
    i += 1

    x0 = a - f(a) / (f(b) - f(a)) * (b - a)
    prev_x = x0

```

```

        if self.second_derivative(x0) == 0:
            return Result(
                success=False,
                message="Метод не применим.  $f'(x_0) = 0$ .",
                method_name=method_name,
                equation=self)

        if f(x0) == 0:
            break
        elif f(a) * f(x0) < 0:
            b = x0
        else:
            a = x0

        table.append([i, self.f(x0), self.first_derivative(x0)])

    return Result(x0, i, method_name, self, table)

```

Метод золотого сечения

```

def golden_section_method(self) -> Result:
    method_name = "Золотого сечения"
    table = [["#", "a", "b"]]

    a, b = self.a, self.b
    f = self.f
    i = 0

    fc, fd = 0, 0
    golden_ratio = (1 + 5 ** 0.5) / 2
    c = b - (b - a) / golden_ratio
    d = a + (b - a) / golden_ratio

    while abs(c - d) > self.eps and i < self.max_iterations:
        i += 1

        if fc == 0: fc = f(c)
        if fd == 0: fd = f(d)

        if fc < fd:
            b = d

```

```

        fc, fd = 0, fc
    else:
        a = c
        fc, fd = fd, 0

    table.append([i, a, b])

    c = b - (b - a) / golden_ratio
    d = a + (b - a) / golden_ratio

    return Result((a + b) / 2, i, method_name, self, table)

def first_derivative(self, x, h=1e-6):
    f = self.f
    return (f(x + h) - f(x)) / h

def second_derivative(self, x, h=1e-6):
    f = self.f
    return (f(x + 2*h) - 2*f(x + h) + f(x)) / (h**2)

def show(self, left=-10, right=10):
    x = np.linspace(left, right, 100)
    y = [self.f(x_i) for x_i in x]
    plt.plot(x, y)
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.grid(True)
    plt.axhline(0, color='black', lw=0.5)
    plt.axvline(0, color='black', lw=0.5)
    plt.show()

```

Примеры и результаты работы программы

```

def f(x): return log(1 + x ** 2) - sin(x)
a, b = 0, pi / 4

```

eps = 0.03

Вывод:

Метод: Библиотека Python

Корень: 0.5515037302450237

Количество итераций: 4

Границы и точность: a=0, b=0.7853981634, eps=0.03

Уравнение: def f(x): return log(1 + x ** 2) - sin(x)

Таблица:

Метод: Половинного деления

Корень: 0.5645049299419159

Количество итераций: 4

Границы и точность: a=0, b=0.7853981634, eps=0.03

Уравнение: def f(x): return log(1 + x ** 2) - sin(x)

Таблица:

+-----+-----+-----+			
#	a	b	
+=====+=====+=====+			
1	0.392699	0.785398	
+-----+-----+-----+			
2	0.392699	0.589049	
+-----+-----+-----+			
3	0.490874	0.589049	
+-----+-----+-----+			
4	0.539961	0.589049	
+-----+-----+-----+			

Метод: Ньютона

Корень: 0.555967930704798

Количество итераций: 18

Границы и точность: $a=0$, $b=0.7853981634$, $\text{eps}=0.03$

Уравнение: $\text{def } f(x): \text{return } \log(1 + x^{**2}) - \sin(x)$

Таблица:

#	$f(x)$	$f'(x)$
1	-0.255673	0.0832323
2	-0.258224	0.0230205
3	-0.258412	0.00606218
4	-0.258425	0.00157389
5	-0.258425	0.000407075
6	-0.258426	0.000105184
7	-0.258426	2.71715e-05
8	-0.258426	7.01827e-06
9	-0.258426	1.81272e-06
10	-0.258426	4.68348e-07
11	-0.258426	1.20959e-07
12	-0.258426	3.13083e-08
13	-0.258426	8.04912e-09
14	-0.258426	1.9984e-09
15	-0.258426	6.66134e-10

16	-0.258426	1.11022e-10
17	-0.258426	5.55112e-11

Метод: Золотого сечения

Корень: 0.542696783556765

Количество итераций: 4

Границы и точность: a=0, b=0.7853981634, eps=0.03

Уравнение: def f(x): return log(1 + x ** 2) - sin(x)

Таблица:

#	a	b
1	0.299995	0.785398
2	0.485403	0.785398
3	0.485403	0.67081
4	0.485403	0.599991