**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: P3126

Преподаватель: Малышева Татьяна Алексеевна



Санкт-Петербург, 2023

# Содержание

[Содержание 2](#_Toc162348435)

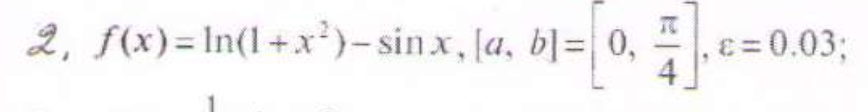
[График 3](#_Toc162348436)

[Листинг программы 3](#_Toc162348437)

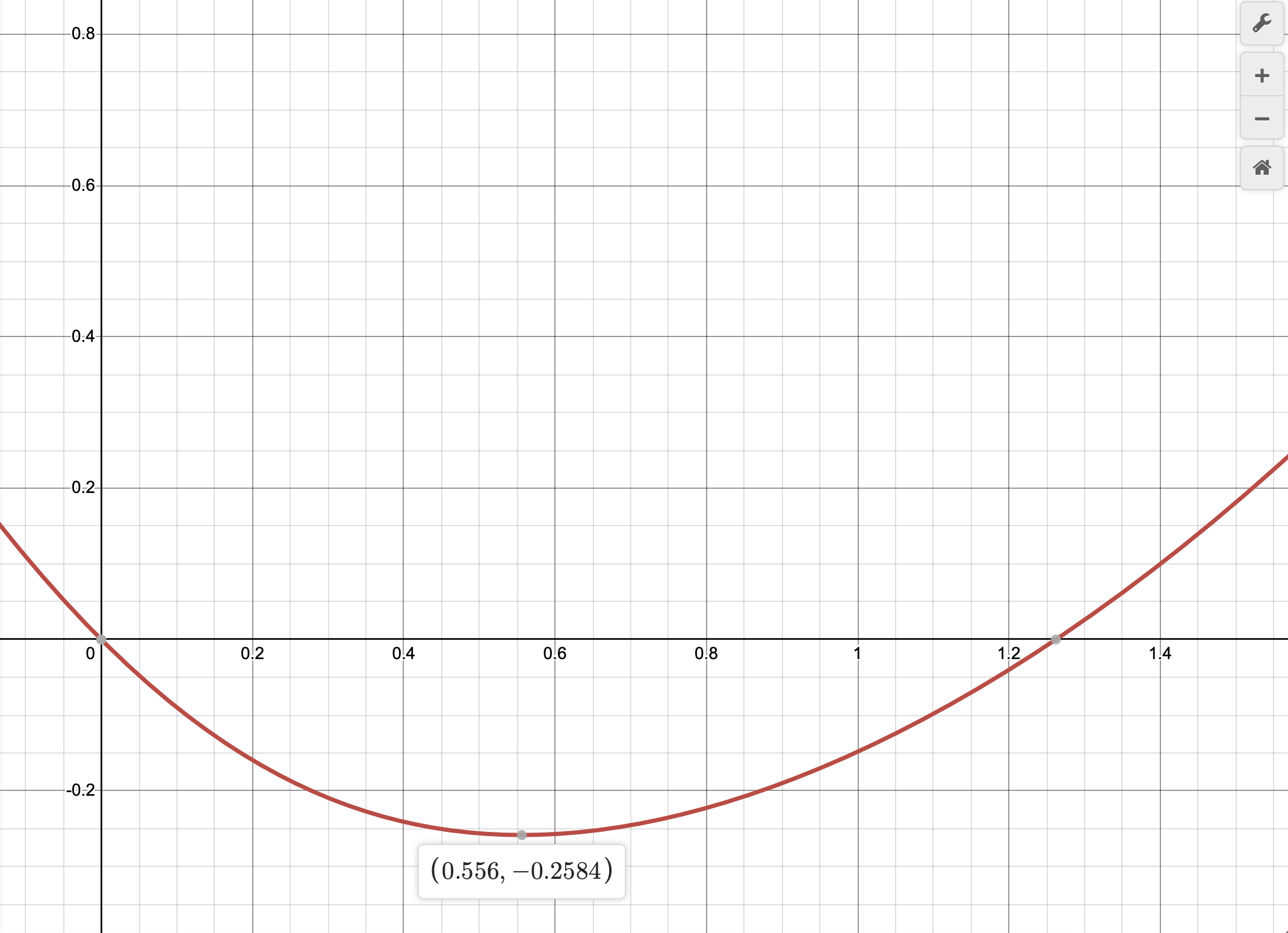
[Примеры и результаты работы программы 5](#_Toc162348438)

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.



# 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 = [["#", "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 = a - f(a) / (f(b) - f(a)) \* (b - a)

while abs(f(a)) > self.eps and i < self.max\_iterations:

i += 1

x0 = x0 - self.first\_derivative(x0) / self.second\_derivative(x0)

if self.second\_derivative(x0) == 0:

return Result(

success=False,

message="Метод не применим. f'(x0) = 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, x0, f"{self.first\_derivative(x0):.2e}" + "."])

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=0.785398163397448, 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=0.785398163397448, eps=0.03

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

Таблица:

+-----+-------------------+-------------------+

| # | a | b |

+=====+===================+===================+

| 1 | 0.392699081698724 | 0.785398163397448 |

+-----+-------------------+-------------------+

| 2 | 0.392699081698724 | 0.589048622548086 |

+-----+-------------------+-------------------+

| 3 | 0.490873852123405 | 0.589048622548086 |

+-----+-------------------+-------------------+

| 4 | 0.539961237335746 | 0.589048622548086 |

+-----+-------------------+-------------------+

Метод: Ньютона

Корень: 0.5530226388201814

Количество итераций: 1

Границы и точность: a=0, b=0.785398163397448, eps=0.03

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

Таблица:

+-----+-------------------+------------+

| # | x | f'(x) |

+=====+===================+============+

| 1 | 0.553022638820181 | -3.94e-03. |

+-----+-------------------+------------+

Метод: Золотого сечения

Корень: 0.542696783556765

Количество итераций: 4

Границы и точность: a=0, b=0.785398163397448, eps=0.03

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

Таблица:

+-----+-------------------+-------------------+

| # | a | b |

+=====+===================+===================+

| 1 | 0.299995403716082 | 0.785398163397448 |

+-----+-------------------+-------------------+

| 2 | 0.485402759681367 | 0.785398163397448 |

+-----+-------------------+-------------------+

| 3 | 0.485402759681367 | 0.670810115646652 |

+-----+-------------------+-------------------+

| 4 | 0.485402759681367 | 0.599990807432163 |

+-----+-------------------+-------------------+