|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Калужский филиал федерального государственного автономного образовательного учреждения высшего образования**  **«Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»**  **(КФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК4 Программное обеспечение ЭВМ, информационные технологии

**ЛАБОРАТОРНАЯ РАБОТА**

**«Минимизация функций»**

**по дисциплине: «*Методы принятия решений в программной инженерии*»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил: студент группы ИУК4-72Б | |  |  | Губин Е.В. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Проверил: | |  |  | Никитенко У.В. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | | | |
|  | - Балльная оценка:  - Оценка: | | | |

Калуга, 2025

**Цель:** ознакомиться с методами одномерного поиска, используемыми

в методах минимизации функций. Сравнить различные алгоритмы по эффективности на тестовых примерах.

**Задание (вариант №23):**

Найти минимум и максимум унимодальной на отрезке [a, b] функции f(x)

c точностью E.



**Результаты выполнения работы:**

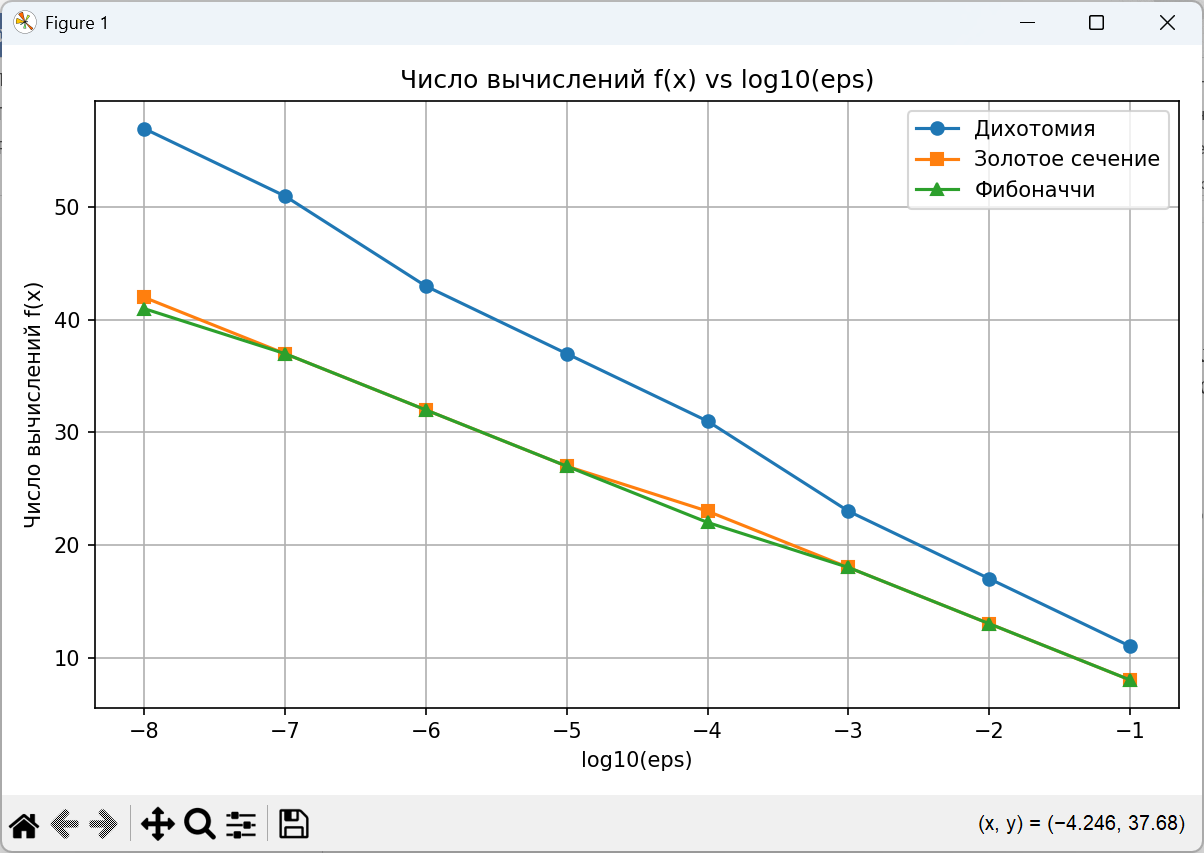


Рисунок 1 График зависимости количества вычислений целевой функции от логарифма задаваемой точности E

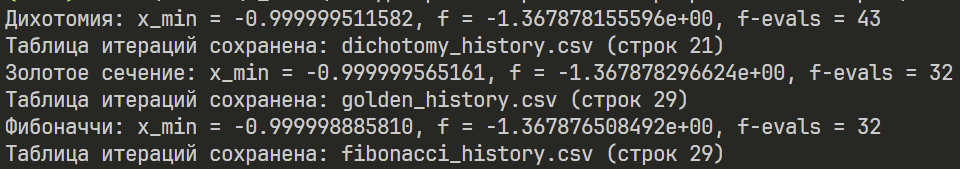


Рисунок 2 Результаты выполнения программы (поиск минимума функции различными методами)

Таблица 1 Результаты исследования метода дихотомии

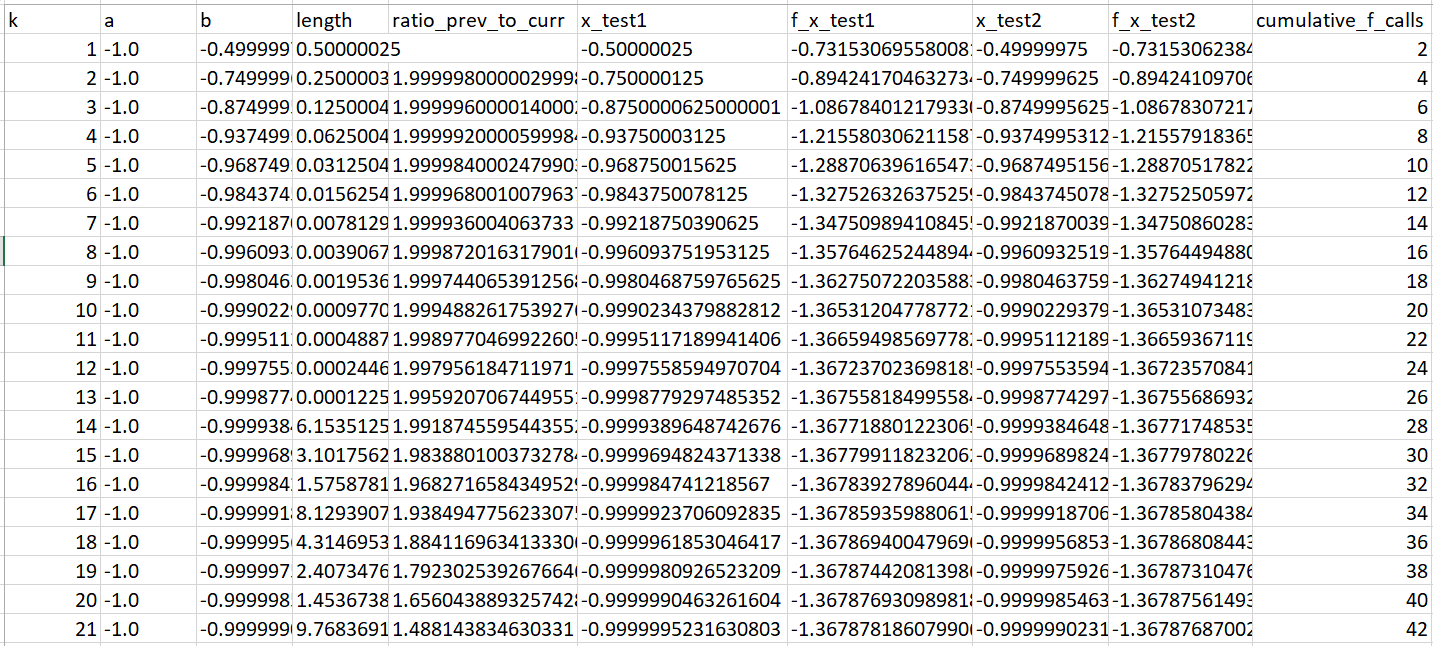


Таблица 2 Результаты исследования метода золотого сечения

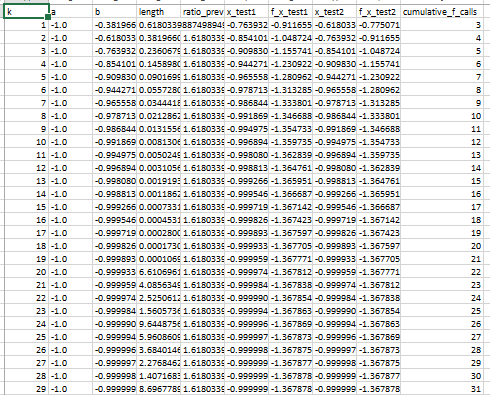
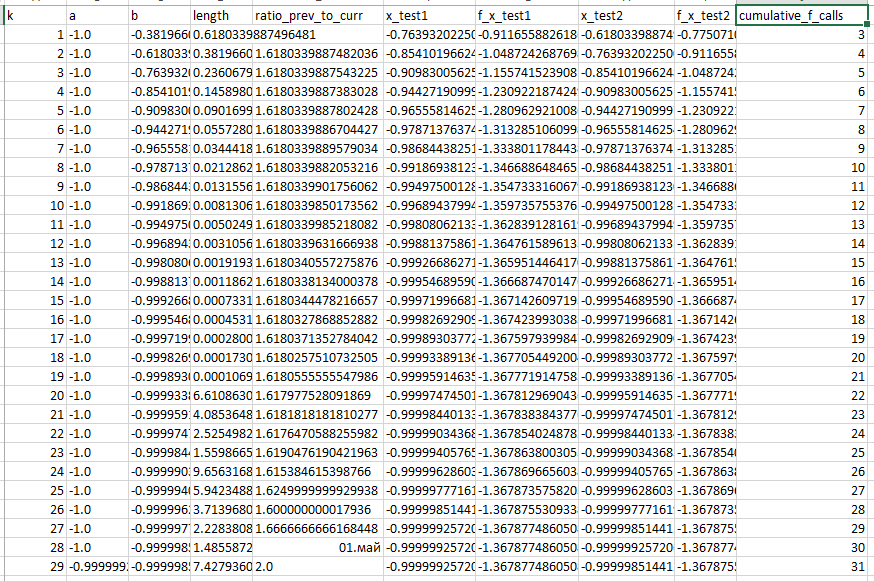


Таблица 3 Результаты исследования метода Фибоначчи



**Листинг программы:**

import math

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

class FuncCounter:

def \_\_init\_\_(self, f):

self.\_f = f

self.count = 0

def \_\_call\_\_(self, x):

self.count += 1

return self.\_f(x)

def reset(self):

self.count = 0

def f\_raw(x):

return x\*\*3 - math.exp(x)

def history\_to\_df(history):

df = pd.DataFrame(history)

lengths = df['length'].values

ratios = [np.nan]

for i in range(1, len(lengths)):

prev = lengths[i-1]

curr = lengths[i]

ratios.append(prev / curr if curr != 0 else np.nan)

df['ratio\_prev\_to\_curr'] = ratios

cols = ['k', 'a', 'b', 'length', 'ratio\_prev\_to\_curr',

'x\_test1', 'f\_x\_test1', 'x\_test2', 'f\_x\_test2', 'cumulative\_f\_calls']

cols\_present = [c for c in cols if c in df.columns]

return df[cols\_present]

def dichotomy\_with\_history(a, b, eps, f, delta=None):

if delta is None:

delta = eps / 4.0

history = []

while (b - a) > eps:

xm = (a + b) / 2.0

x1 = xm - delta

x2 = xm + delta

f1 = f(x1)

f2 = f(x2)

if f1 < f2:

b = x2

else:

a = x1

length = b - a

history.append({

'k': len(history) + 1,

'a': a, 'b': b, 'length': length,

'x\_test1': x1, 'f\_x\_test1': f1,

'x\_test2': x2, 'f\_x\_test2': f2,

'cumulative\_f\_calls': f.count

})

x\_min = (a + b) / 2.0

return x\_min, f(x\_min), history

def golden\_with\_history(a, b, eps, f):

phi = (math.sqrt(5) - 1) / 2.0

x1 = b - phi \* (b - a)

x2 = a + phi \* (b - a)

f1 = f(x1)

f2 = f(x2)

history = []

while (b - a) > eps:

if f1 < f2:

b = x2

x2 = x1

f2 = f1

x1 = b - phi \* (b - a)

f1 = f(x1)

else:

a = x1

x1 = x2

f1 = f2

x2 = a + phi \* (b - a)

f2 = f(x2)

length = b - a

history.append({

'k': len(history) + 1,

'a': a, 'b': b, 'length': length,

'x\_test1': x1, 'f\_x\_test1': f1,

'x\_test2': x2, 'f\_x\_test2': f2,

'cumulative\_f\_calls': f.count

})

x\_min = (a + b) / 2.0

return x\_min, f(x\_min), history

def fibonacci\_with\_history(a, b, eps, f):

L0 = b - a

F = [1, 1]

while F[-1] < L0 / eps:

F.append(F[-1] + F[-2])

n = len(F)

history = []

if n < 3:

x\_mid = (a + b) / 2.0

val = f(x\_mid)

history.append({

'k': 1, 'a': a, 'b': b, 'length': b-a,

'x\_test1': x\_mid, 'f\_x\_test1': val,

'x\_test2': np.nan, 'f\_x\_test2': np.nan,

'cumulative\_f\_calls': f.count

})

return x\_mid, val, history

x1 = a + (F[-3] / F[-1]) \* (b - a)

x2 = a + (F[-2] / F[-1]) \* (b - a)

f1 = f(x1)

f2 = f(x2)

k = 1

while k <= n - 2 and (b - a) > eps:

if f1 < f2:

b = x2

x2 = x1

f2 = f1

idx = n - k - 3

if idx >= 0:

x1 = a + (F[idx] / F[n - k - 1]) \* (b - a)

f1 = f(x1)

else:

x1 = (a + b) / 2.0

f1 = f(x1)

else:

a = x1

x1 = x2

f1 = f2

idx = n - k - 2

if idx >= 0:

x2 = a + (F[idx] / F[n - k - 1]) \* (b - a)

f2 = f(x2)

else:

x2 = (a + b) / 2.0

f2 = f(x2)

length = b - a

history.append({

'k': k,

'a': a, 'b': b, 'length': length,

'x\_test1': x1, 'f\_x\_test1': f1,

'x\_test2': x2, 'f\_x\_test2': f2,

'cumulative\_f\_calls': f.count

})

k += 1

x\_min = (a + b) / 2.0

return x\_min, f(x\_min), history

def main():

a0 = -1.0

b0 = 0.0

eps0 = 1e-6

fc = FuncCounter(f\_raw)

x\_d, fx\_d, hist\_d = dichotomy\_with\_history(a0, b0, eps0, fc)

df\_d = history\_to\_df(hist\_d)

df\_d.to\_csv("dichotomy\_history.csv", sep=";", index=False)

print(f"Дихотомия: x\_min = {x\_d:.12f}, f = {fx\_d:.12e}, f-evals = {fc.count}")

print(f"Таблица итераций сохранена: dichotomy\_history.csv (строк {len(df\_d)})")

fc = FuncCounter(f\_raw)

x\_g, fx\_g, hist\_g = golden\_with\_history(a0, b0, eps0, fc)

df\_g = history\_to\_df(hist\_g)

df\_g.to\_csv("golden\_history.csv", sep=";", index=False)

print(f"Золотое сечение: x\_min = {x\_g:.12f}, f = {fx\_g:.12e}, f-evals = {fc.count}")

print(f"Таблица итераций сохранена: golden\_history.csv (строк {len(df\_g)})")

fc = FuncCounter(f\_raw)

x\_f, fx\_f, hist\_f = fibonacci\_with\_history(a0, b0, eps0, fc)

df\_f = history\_to\_df(hist\_f)

df\_f.to\_csv("fibonacci\_history.csv", sep=";", index=False)

print(f"Фибоначчи: x\_min = {x\_f:.12f}, f = {fx\_f:.12e}, f-evals = {fc.count}")

print(f"Таблица итераций сохранена: fibonacci\_history.csv (строк {len(df\_f)})")

print("\nПервые 6 итераций (Дихотомия):")

print(df\_d.head(6).to\_string(index=False))

print("\nПервые 6 итераций (Золотое сечение):")

print(df\_g.head(6).to\_string(index=False))

print("\nПервые 6 итераций (Фибоначчи):")

print(df\_f.head(6).to\_string(index=False))

plt.figure(figsize=(8,5))

plt.plot(df\_d['k'], df\_d['length'], marker='o', label='Дихотомия')

plt.plot(df\_g['k'], df\_g['length'], marker='s', label='Золотое сечение')

plt.plot(df\_f['k'], df\_f['length'], marker='^', label='Фибоначчи')

plt.yscale('log')

plt.xlabel('k (итерация)')

plt.ylabel('Длина интервала (log scale)')

plt.title('Длина интервала по итерациям (eps = {})'.format(eps0))

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.savefig("interval\_lengths\_by\_iteration.png")

print("\nГрафик длины интервала сохранён: interval\_lengths\_by\_iteration.png")

if \_\_name\_\_ == "\_\_main\_\_":

main

**Выводы:** в ходе лабораторной работы я реализовал на языке Python три метода одномерной минимизации функции: метод дихотомии, метод золотого сечения и метод Фибоначчи. На основе построенных таблиц итераций и графиков были проанализированы границы интервалов неопределённости на каждом шаге, длины интервалов и их уменьшение, а также количество вычислений функции.