Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №2

по дисциплине «Методы оптимизации»

Вариант 13

Выполнила:

Павличенко Софья Алексеевна, Р3215

Преподаватель:

Селина Елена Георгиевна

Санкт-Петербург 2025г.

Оглавление

[Задание 2](#_Toc193567068)

[Решение 3](#_Toc193567069)

[Метод половинного деления 4](#_Toc193567070)

[Исходный код программы 5](#_Toc193567071)

[Результаты работы программы 6](#_Toc193567072)

[Метод золотого сечения 6](#_Toc193567073)

[Исходный код программы 8](#_Toc193567074)

[Результаты работы программы 8](#_Toc193567075)

[Метод хорд 8](#_Toc193567076)

[Исходный код программы 10](#_Toc193567077)

[Результаты работы программы 11](#_Toc193567078)

[Метод Ньютона 11](#_Toc193567079)

[Исходный код программы 13](#_Toc193567080)

[Результаты работы программы 13](#_Toc193567081)

[Вывод 13](#_Toc193567082)

# 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.

# Решение

## Метод половинного деления

**Шаг 1:**  
Рассматриваем отрезок

​⇒ Отсекаем конец отрезка: , от 0.5000 до 1.0005.

**Шаг 2:**  
  
Рассматриваем отрезок

Отсекаем конец отрезка: от 0.5000 до 0.7508.

**Шаг 3:**  
Рассматриваем отрезок [0.5000; 0.7508].

Отсекаем начало отрезка: , от 0.6249 до 0.7508.

**Шаг 4:**  
  
Рассматриваем отрезок

​ Отсекаем начало отрезка: , от 0.6873 до 0.7508.

**Шаг 5:**  
  
Рассматриваем отрезок

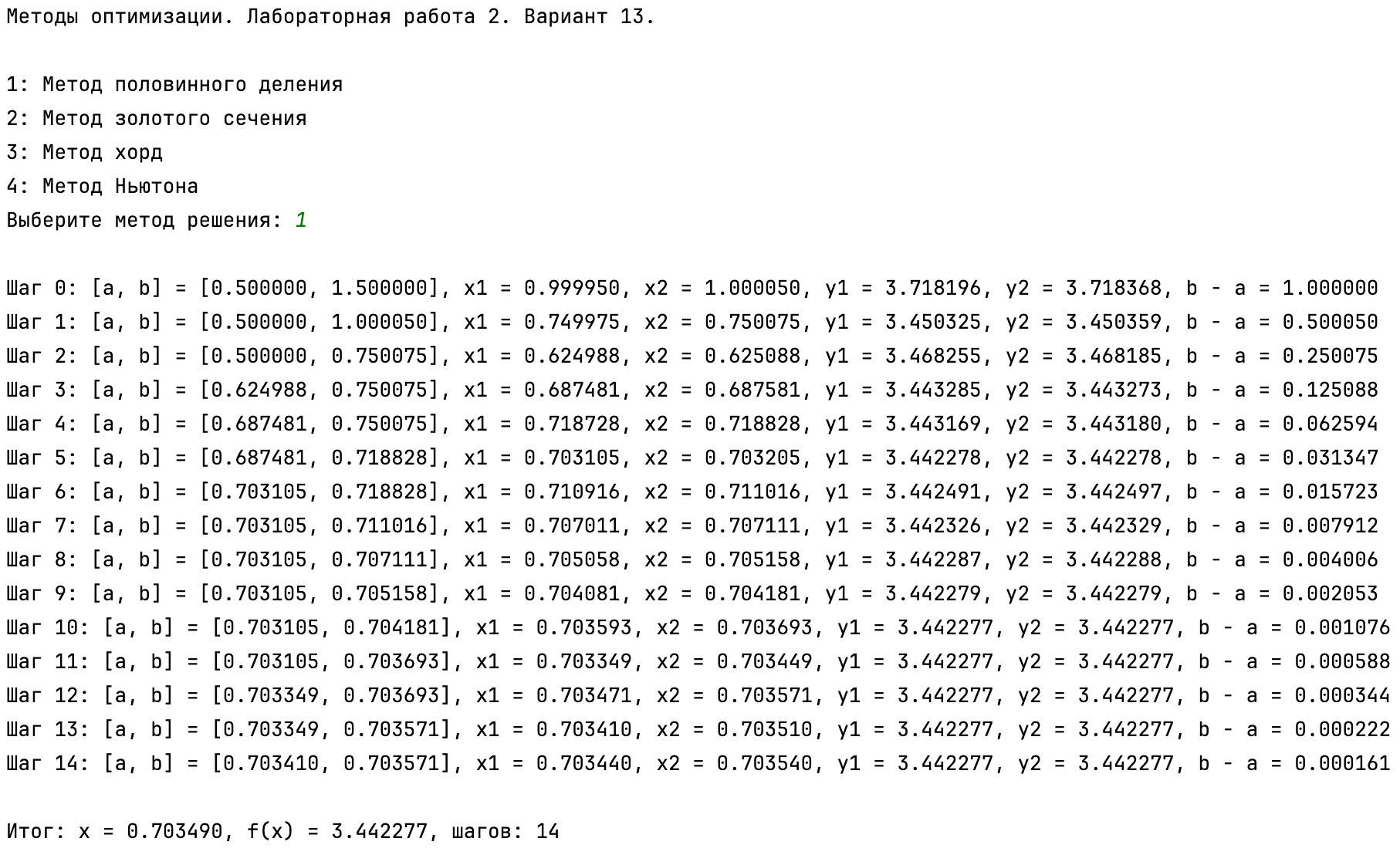
Отсекаем конец отрезка: , от 0.6873 до 0.7195.

…

### Исходный код программы

def half\_division\_method(a, b, eps, n=0):  
 *"""Метод половинного деления"""* if n > MAX\_ITERATIONS:  
 raise ValueError("Метод не сошёлся за максимальное число итераций.")  
 x1 = (a + b - eps) / 2  
 x2 = (a + b + eps) / 2  
 y1 = f(x1)  
 y2 = f(x2)  
 print(f"Шаг {n}: [a, b] = [{a:.6f}, {b:.6f}], x1 = {x1:.6f}, x2 = {x2:.6f}, y1 = {y1:.6f}, y2 = {y2:.6f}, b - a = {b - a:.6f}")  
 if b - a <= 2\*eps:  
 return (a + b) / 2, f((a + b) / 2), n  
 if y1 > y2:  
 return half\_division\_method(x1, b, eps, n + 1)  
 else:  
 return half\_division\_method(a, x2, eps, n + 1)

### Результаты работы программы



## Метод золотого сечения

**Шаг 1:**

Рассматриваем отрезок

Так как , отсекаем правую часть отрезка:

Новый отрезок:

**Шаг 2:**

Рассматриваем отрезок .

Так как , отсекаем правую часть отрезка:

Новый отрезок:

**Шаг 3:**

Рассматриваем отрезок .

Так как , отсекаем левую часть отрезка:

Новый отрезок: .

**Шаг 4:**

Рассматриваем отрезок .

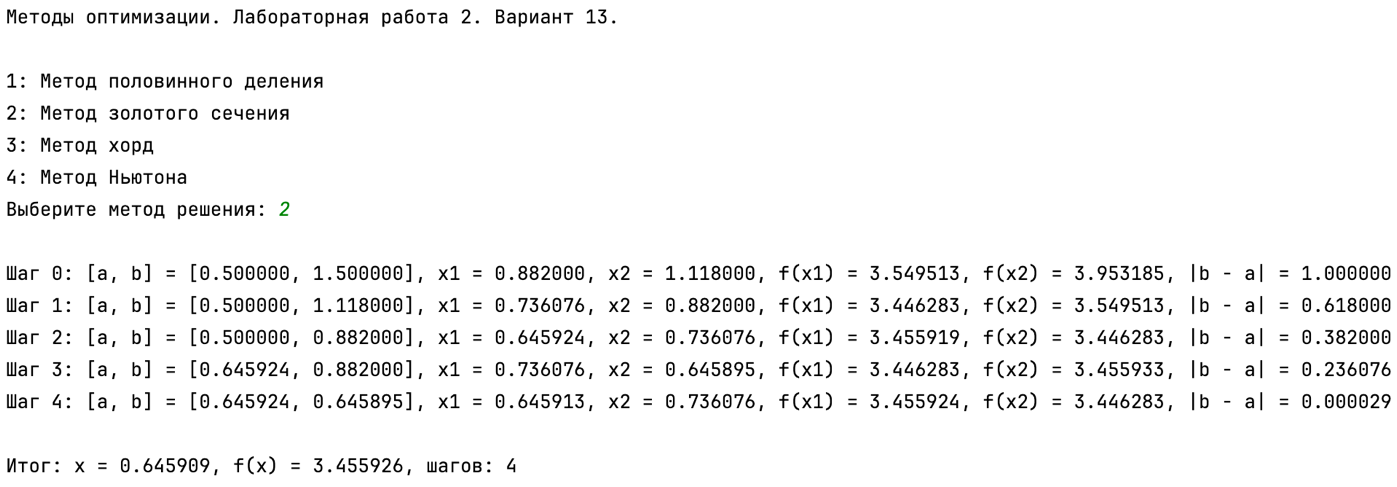
Так как , отсекаем правую часть отрезка:

Новый отрезок:

### Исходный код программы

def golden\_section\_method(a, b, x1, x2, eps, n=0):  
 *"""Метод золотого сечения"""* if n > MAX\_ITERATIONS:  
 raise ValueError("Метод не сошёлся за максимальное число итераций.")  
 print(f"Шаг {n}: [a, b] = [{a:.6f}, {b:.6f}], x1 = {x1:.6f}, x2 = {x2:.6f}, f(x1) = {f(x1):.6f}, "  
 f"f(x2) = {f(x2):.6f}, |b - a| = {abs(b - a):.6f}")  
 if abs(b - a) <= eps:  
 return (a + b) / 2, f((a + b) / 2), n  
 if f(x1) < f(x2):  
 return golden\_section\_method(a, x2, a + 0.382 \* (x2 - a), x1, eps, n + 1)  
 else:  
 return golden\_section\_method(x1, b, x2, eps, a + 0.618 \* (b - x1), n + 1)

### Результаты работы программы



## Метод хорд

**Шаг 1:**

Рассматриваем отрезок .

Так как , отсекаем левую часть отрезка:

Новый отрезок: .

**Шаг 2:**

Рассматриваем отрезок

Так как , отсекаем левую часть отрезка:

Новый отрезок: .

**Шаг 3:**

Рассматриваем отрезок .

Так как , отсекаем левую часть отрезка:

Новый отрезок:

**Шаг 4:**

Рассматриваем отрезок .

Так как , отсекаем левую часть отрезка:

Новый отрезок:

**Шаг 5:**

Рассматриваем отрезок .

Так как , отсекаем левую часть отрезка:

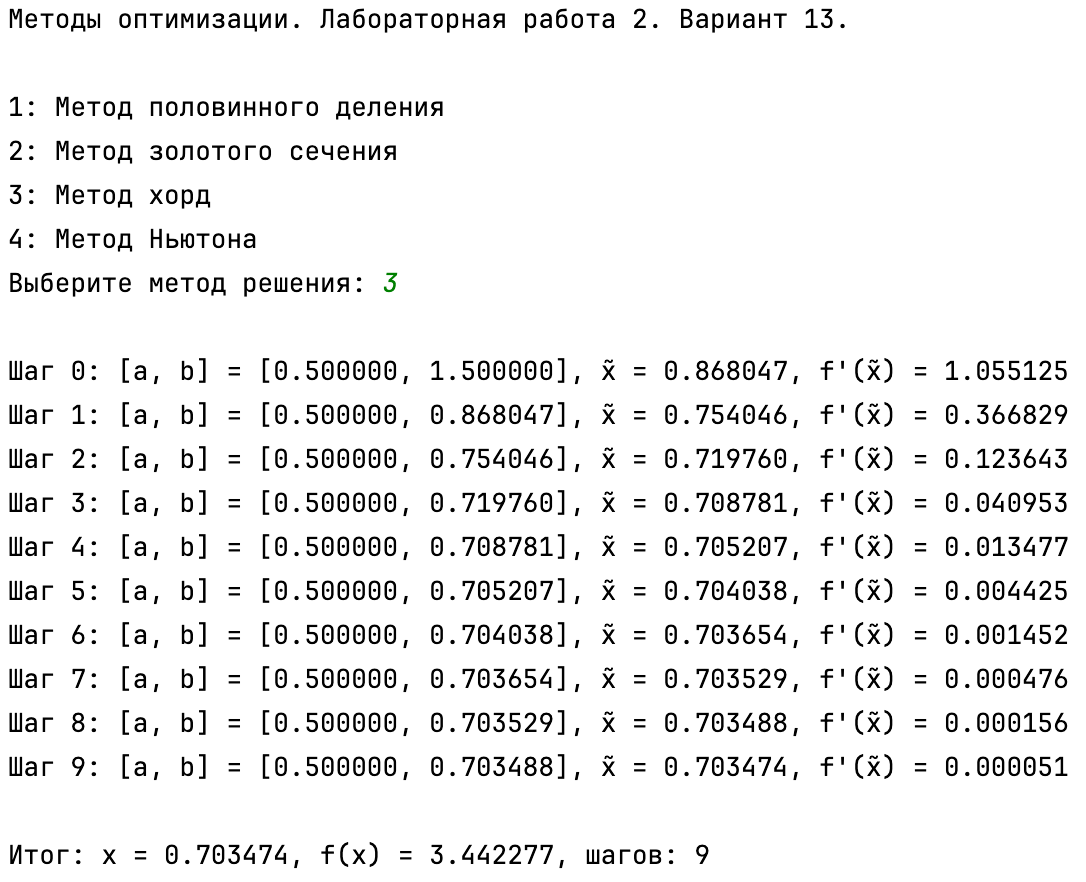
Новый отрезок:

…

### Исходный код программы

def chord\_method(a, b, eps, n=0):  
 *"""Метод хорд"""* if n > MAX\_ITERATIONS:  
 raise ValueError("Метод не сошёлся за максимальное число итераций.")  
 tx = a - df(a)/(df(a) - df(b)) \* (a - b)  
 print(f"Шаг {n}: [a, b] = [{a:.6f}, {b:.6f}], x̃ = {tx:.6f}, f'(x̃) = {df(tx):.6f}")  
 if abs(df(tx)) <= eps:  
 return tx, f(tx), n  
 if df(tx) > 0:  
 return chord\_method(a, tx, eps, n + 1)  
 else:  
 return chord\_method(tx, b, eps, n + 1)

### Результаты работы программы



## Метод Ньютона

Начнем с середины заданного отрезка .

**Шаг 1:**

Касательная к графику функции в точке пересекает ось Oy в точке:

В точке вычисляем значение производной:

Так как , продолжаем итерацию:

**Шаг 2:**

Касательная к графику функции в точке пересекает ось Oy в точке:

В точке вычисляем значение производной:

Так как , продолжаем итерацию:

**Шаг 3:**

Касательная к графику функции в точке пересекает ось Oy в точке:

В точке вычисляем значение производной:

Так как , продолжаем итерацию:

**Шаг 4:**

Касательная к графику функции в точке пересекает ось Oy в точке:

В точке вычисляем значение производной:

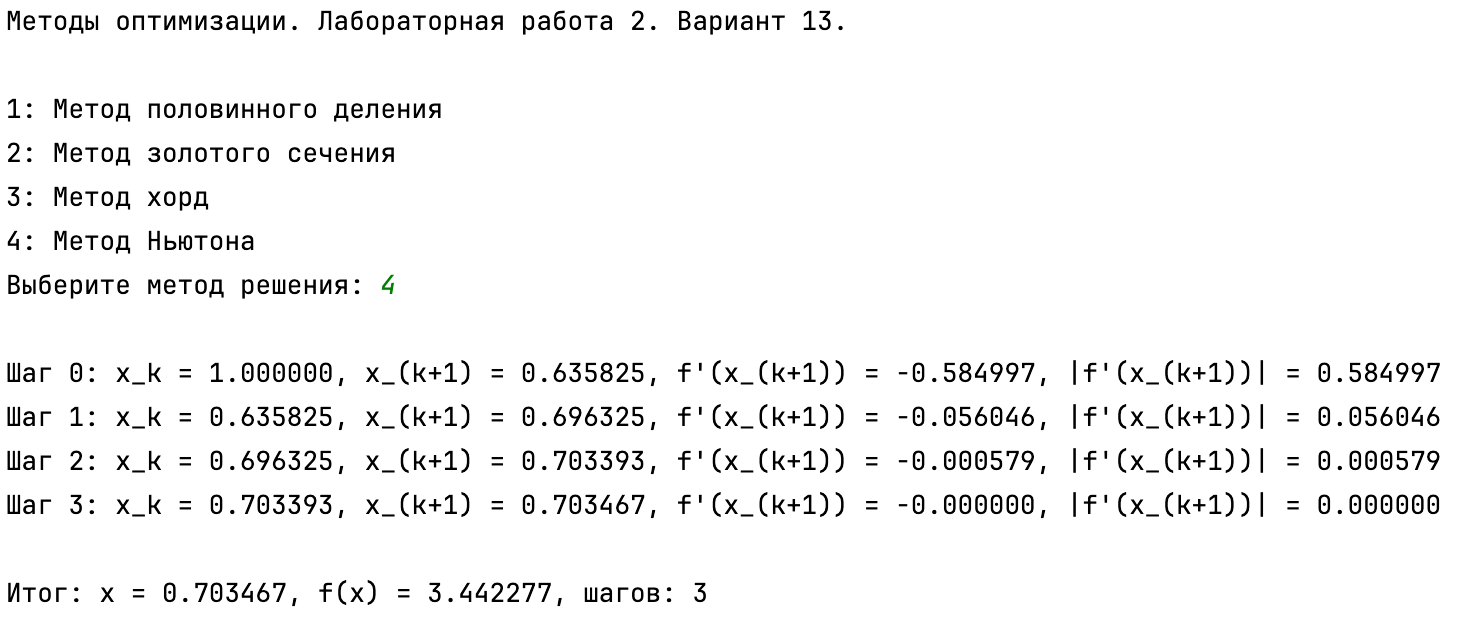
Так как , завершили итерацию:

Минимум найден в точке , значение функции в минимуме .

### Исходный код программы

def newtons\_method(x0, eps, n=0):  
 *"""Метод Ньютона"""* x1 = x0 - df(x0)/ddf(x0)  
 print(f"Шаг {n}: x\_k = {x0:.6f}, x\_(k+1) = {x1:.6f}, f'(x\_(k+1)) = {df(x1):.6f}, |f'(x\_(k+1))| = {abs(df(x1)):.6f}")  
 if abs(df(x1)) <= eps:  
 return x1, f(x1), n  
 return newtons\_method(x1, eps, n + 1)

### Результаты работы программы



# Вывод]

В ходе выполнения лабораторной работы я научилась находить минимум функции методами половинного деления, золотого сечения, хорд и Ньютона, реализовав методы на языке программирования Python. В результате работы были найдены минимумы уравнений на отрезке с определенной точностью.