Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №3**

По дисциплине: «Компьютерные системы моделирования»

Тема: «Аналитическое моделирование. Применение численных методов для решения задачи оптимизации»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Соболев Р.В. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь, 2025

**Оглавление**

[**Цель и постановка задачи для варианта.** 3](#_Toc191257819)

[**Описание алгоритмов.** 4](#_Toc191257820)

[**Отчет по реализации.** 7](#_Toc191257821)

[**Таблица зависимости количества вычислений от точности** 13](#_Toc191257822)

[**Ссылка на репозиторий** 14](#_Toc191257823)

# **Цель и постановка задачи для варианта.**

**Цель работы**: Научиться реализовывать на языке python методы поиска безусловного минимума или максимума функции и применять их к решению трансцендентных уравнений.

**Постановка задачи:**

Найти безусловный минимум функции одной переменной , представленной трансцендентным уравнением. Минимумом функции является такая точка , что . Значение минимальной точки вычисляется приближенно с заданной точностью . При вычислении применить прямые методы решения задачи, метод дихотомии или бисекции, метод «золотого сечения» или Фибоначчи, а также метод, использующий значение производных целевой функции – метод хорд или секущих и метод Ньютона (касательных).

**Функция:**

# **Описание алгоритмов.**

Алгоритмы основываются на возможность определения значений в заданных точках.

***Метод дихотомии (разделение пополам с максимальным уменьшением)***

В этом методе пробные точки и располагаются близко к середине очередного отрезка , т.е.

где – малое число ( ). Название метода объясняется тем, что отношение длин нового и исходного отрезков

близко по значению к . Для любых точек , величина , поэтому указанный выбор пробных точек объясняется стремлением обеспечить максимально возможное относительное уменьшение отрезка на каждой итерации поиска .

Результатом метода является приближенное значение середины последнего из найденных отрезков при том, что неравенство должно быть достигнуто.

*Алгоритм*

Шаг 1. Определить и по формуле (1) и вычислить и . Перейти к шагу 2.

Шаг 2. Сравнить и . Если , то перейти к отрезку , положив , иначе – к отрезку , положив . Перейти к шагу 3.

Шаг 3. Найти достигнутую точность , где .

Если , то перейти к следующей итерации, вернувшись к шагу 1. Если , то завершить поиск , перейдя к шагу 4.

Шаг 4. Положить .

***Метод хорд (секущих)***

В основе метода касательных Ньютона лежит уточнение корня за принятия в качестве каждого нового приближения точки пересечения с осью абсцисс касательной, проведенной к графику функции в точке предыдущего приближения. Для построения касательной необходимо вычислять производную функции в данной точке. Метод секущих является производным от метода касательных путем замены производной на приближенную разностную формулу.

В этом методе производная функции подсчитывается с помощью следующих соотношений.

– в точке используется формула , где – малая положительная величина.

– в точке используется формула

– в точках , используется формула .

Вычисленное значение определяет тангенс угла наклона секущей.

Изображение выглядит как линия, Параллельный, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2. Метод секущих для функции

Методика применения метода секущих совпадает с методом Ньютона, но используется формула

*Алгоритм метода хорд (без производных)*

Шаг 1. Пусть k = 0.

Шаг 2. Следующий номер итерации k = k +1. Очередное приближение определяется по формуле

Далее вычисляется .

Шаг 3. Если , то перейти к шагу 5 Если

.

Шаг 4. Если , то перейти к шагу 2.

Шаг 5. Вывести значение корня .

*Для реализации метода хорд можно использовать программу, текст которой представлен ниже.*

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

# **Отчет по реализации.**

**Программная реализация**

**Реализация численных методов поиска экстремумов**

**Метод дихотомии**

Метод дихотомии — это численный метод, который последовательно делит интервал пополам, чтобы найти экстремум.

class DichotomySearch:

    def \_\_init\_\_(self, func, start, end, eps):

        self.\_func = func

        self.\_start = start

        self.\_end = end

        self.\_eps = eps

def search(self, flag):

        a = self.start

        b = self.end

        delta = self.eps

        iteration = 0

        while True:

            iteration += 1

            x1 = (b + a - delta) / 2

            x2 = (b + a + delta) / 2

            f\_x1 = self.func(x1)

            f\_x2 = self.func(x2)

            if flag:  # true для максимума

                if f\_x1 >= f\_x2:

                    b = x2

                else:

                    a = x1

            else:  # false для минимума

                if f\_x1 <= f\_x2:

                    b = x2

                else:

                    a = x1

            epsilon\_n = (b - a) / 2

            if epsilon\_n < self.eps:

                break

        return (a + b) / 2, iteration

**Метод хорд**

class ChordSearch:

    def \_\_init\_\_(self, func, start, end, eps, max\_iter=1000):

        self.\_func = func

        self.\_start = start

        self.\_end = end

        self.\_eps = eps

        self.\_max\_iter = max\_iter

def \_derivative(self, x):

        # Вычисление производной для функции x^5 \* sin(5x)

        return 5 \* x\*\*4 \* np.sin(5 \* x) + 5 \* x\*\*5 \* np.cos(5 \* x)

def search(self):

        a = self.start

        b = self.end

        prev\_x = None

        current\_x = None

        iteration = 0

        while iteration < self.max\_iter:

            iteration += 1

            # Вычисляем производные на концах интервала

            d\_a = self.\_derivative(a)

            d\_b = self.\_derivative(b)

            # Проверка деления на ноль

            if abs(d\_b - d\_a) < 1e-12:

                raise ZeroDivisionError("Знаменатель формулы равен нулю")

            # Вычисляем новое приближение по методу хорд

            current\_x = a - (d\_a \* (b - a)) / (d\_b - d\_a)

            # Проверка, что current\_x находится внутри интервала [a, b]

            if current\_x < a or current\_x > b:

                # Если current\_x выходит за пределы интервала, ограничиваем его

                current\_x = np.clip(current\_x, a, b)

            # Проверка условия останова |x\_k - x\_{k-1}| < eps

            if prev\_x is not None and abs(current\_x - prev\_x) < self.eps:

                break

            # Вычисляем производную в новой точке

            d\_current = self.\_derivative(current\_x)

            # Проверка условия завершения (производная близка к нулю)

            if abs(d\_current) < self.eps:

                break

            # Обновление границ интервала согласно алгоритму

            if d\_current \* d\_b > 0:

                b = current\_x

            else:

                a = current\_x

            prev\_x = current\_x

        if iteration >= self.max\_iter:

            raise RuntimeError(f"Метод не сошелся за {self.max\_iter} итераций.")

        return current\_x, iteration

**Пример вывода программы**

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Для метода дихотомии**

Метод дихотомии:

Минимум: 4.753818678283691

Максимум: 2.8939868759155276

Кол-во итераций: 15

**Для метода хорд**

Метод хорд:

Экстремум: 2.2815489342778226

Кол-во итераций: 23

Метод хорд требует больше итераций, чем метод дихотомии, для достижения той же точности.

Разница становится более заметной при уменьшении ε.

**Анализ результатов**

Если точность ε высокая, то **метод дихотомии** предпочтительнее, так как он требует меньше итераций и вычислений функции.

Если производная функции легко вычисляется, и поведение функции известно, можно использовать **метод хорд**, но он может потребовать больше итераций для достижения той же точности.

# **Таблица зависимости количества вычислений от точности**

Зависимость количества вычислений функции от заданной точности ε

|  |  |  |
| --- | --- | --- |
| Точность (ε) | Количество итераций | |
| Метод хорд | Метод дихотомии |
| 10^(-2) | 10 | 9 |
| 10^(-3) | 17 | 12 |
| 10^(-4) | 23 | 15 |
| 10^(-5) | 29 | 19 |
| 10^(-6) | 35 | 22 |
| 10^(-7) | 41 | 25 |

# **Ссылка на репозиторий**

https://github.com/affinePassion/ComputerSystems