|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа №2**

**“Реализация методов оптимизации. Перебор, бисекция, золотое сечение, Фибоначчи.”**

**ПО КУРСУ:**

***«Методы оптимизация»***

Студент ИУ9-82Б Потребина В. В.

Преподаватель Посевин Д. П.

*Москва, 2024 г.*

**ОГЛАВЛЕНИЕ**

[1. Постановка задачи 3](#_heading=h.30j0zll)

[2. Практическая реализация 7](#_heading=h.1fob9te)

3[. Результаты](#_heading=h.1fob9te) 10

# 1. Постановка задачи

**Цели:**

1. **Аппроксимация производных**: Разработать методы для численного вычисления первой и второй производных функции.
2. **Проверка унимодальности**: Определить, является ли функция унимодальной на заданном интервале, используя первую и вторую производные.
3. **Минимизация функции**: Реализовать и сравнить различные методы оптимизации для нахождения минимума функции на заданном интервале.

**Постановка задач:**

1. **Аппроксимация производных**:
   * Разработать функции для численного вычисления первой и второй производных функции с использованием центральных разностей.
   * Обеспечить точность вычислений с помощью подходящего шага h*h*.
2. **Проверка унимодальности**:
   * Реализовать методы для проверки унимодальности функции на заданном интервале:
     + По первой производной: функция считается унимодальной, если первая производная меняет знак только один раз.
     + По второй производной: функция считается унимодальной, если вторая производная не меняет знак.
   * Применить эти методы к примерным функциям f(x)=(x−2)2+3*f*(*x*)=(*x*−2)2+3 и g(x)=x4−4x2+3*g*(*x*)=*x*4−4*x*2+3 на заданных интервалах.
3. **Минимизация функции**:
   * Реализовать и сравнить следующие методы оптимизации:
     + **Метод перебора**: Простой метод, который перебирает значения функции с заданным шагом и находит минимальное значение.
     + **Метод бисекции**: Метод, который делит интервал пополам и выбирает подходящий подинтервал для дальнейшего поиска минимума.
     + **Метод золотого сечения**: Метод, который использует золотое сечение для выбора точек в интервале и сужает интервал до тех пор, пока не будет найден минимум.
     + **Метод Фибоначчи**: Метод, который использует числа Фибоначчи для выбора точек в интервале и сужает интервал до тех пор, пока не будет найден минимум.
   * Применить эти методы к функции f(x)=(x−2)2+3*f*(*x*)=(*x*−2)2+3 на интервале [0,4][0,4] и сравнить их эффективность по количеству итераций и точности найденного минимума.

# 2. Практическая реализация

using Printf

# Аппроксимация первой производной

function approximate\_first\_derivative(f, x, h=1e-5)

    return (f(x + h) - f(x - h)) / (2h)

end

# Аппроксимация второй производной

function approximate\_second\_derivative(f, x, h=1e-5)

    return (f(x + h) - 2f(x) + f(x - h)) / h^2

end

# Проверка унимодальности по первой производной

function is\_unimodal\_first\_derivative(f, a, b, h=1e-5)

    x = range(a, stop=b, length=1000)

    first\_derivative = [approximate\_first\_derivative(f, xi, h) for xi in x]

    # Проверяем знак первой производной

    sign\_changes = 0

    for i in 2:length(first\_derivative)

        if first\_derivative[i] \* first\_derivative[i-1] < 0

            sign\_changes += 1

        end

    end

    return sign\_changes == 1

end

# Проверка унимодальности по второй производной

function is\_unimodal\_second\_derivative(f, a, b, h=1e-5)

    x = range(a, stop=b, length=1000)

    second\_derivative = [approximate\_second\_derivative(f, xi, h) for xi in x]

    sign\_changes = 0

    for i in 2:length(second\_derivative)

        if second\_derivative[i] \* second\_derivative[i-1] < 0

            sign\_changes += 1

        end

    end

    return sign\_changes == 0

end

# Метод перебора

function perebor(f, a, b, step)

    x\_vals = a:step:b

    y\_vals = f.(x\_vals)

    min\_index = argmin(y\_vals)

    return x\_vals[min\_index], y\_vals[min\_index], length(x\_vals)

end

# Метод бисекции

function bisection(f, a, b, eps)

    a = Float64(a)

    b = Float64(b)

    intervals = [(a, b)]

    iters = 0

    while b - a > eps

        iters += 1

        m = (a + b) / 2

        if f(m - eps) < f(m + eps)

            b = m

        else

            a = m

        end

        push!(intervals, (a, b))

    end

    min = (a + b) / 2

    return min, f(min), iters

end

# Метод золотого сечения

function golden\_section(f, a, b, eps)

    k = (sqrt(5) - 1) / 2

    x1 = a + (1 - k) \* (b - a)

    x2 = a + k \* (b - a)

    a = Float64(a)

    b = Float64(b)

    intervals = [(a, b)]

    iters = 0

    while abs(x1 - x2) > eps

        iters += 1

        if f(x1) <= f(x2)

            b = x2

            x2 = x1

            x1 = a + b - x1

        else

            a = x1

            x1 = x2

            x2 = a + b - x2

        end

        push!(intervals, (x1, x2))

    end

    min = (a + b) / 2

    return min, f(min), iters

end

# Функция Фибоначчи

function fibonacci(n)

    if n == 1 || n == 2

        return 1

    end

    return fibonacci(n - 1) + fibonacci(n - 2)

end

# Метод Фибоначчи

function fibonacci\_search(f, a, b, eps)

    n = 10

    fib = [fibonacci(i) for i in 1:n]  # Генерация чисел Фибоначчи

    x1 = a + (fib[n-2] / fib[n]) \* (b - a)

    x2 = a + (fib[n-1] / fib[n]) \* (b - a)

    a = Float64(a)

    b = Float64(b)

    intervals = [(a, b)]

    iters = 0

    for k in 1:(n-3)

        iters += 1

        if f(x1) > f(x2)

            a = x1

            x1 = x2

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

        else

            b = x2

            x2 = x1

            x1 = a + (fib[n-k-2] / fib[n-k]) \* (b - a)

        end

        push!(intervals, (a, b))

    end

    min = (a + b) / 2

    return min, f(min), iters

end

# Пример функций

f(x) = (x - 2)^2 + 3

g(x) = x^4 - 4\*x^2 + 3

a, b = 0, 4

c, d = -2, 2

# Проверка на унимодальность для функции f

if is\_unimodal\_first\_derivative(f, a, b)

    @printf("Функция f унимодальна на интервале [%f, %f] по первой производной\n", a, b)

else

    @printf("Функция f не унимодальна на интервале [%f, %f] по первой производной\n", a, b)

end

if is\_unimodal\_second\_derivative(f, a, b)

    @printf("Функция f унимодальна на интервале [%f, %f] по второй производной\n", a, b)

else

    @printf("Функция f не унимодальна на интервале [%f, %f] по второй производной\n", a, b)

end

# Проверка на унимодальность для функции g

if is\_unimodal\_first\_derivative(g, c, d)

    @printf("Функция g унимодальна на интервале [%f, %f] по первой производной\n", c, d)

else

    @printf("Функция g не унимодальна на интервале [%f, %f] по первой производной\n", c, d)

end

if is\_unimodal\_second\_derivative(g, c, d)

    @printf("Функция g унимодальна на интервале [%f, %f] по второй производной\n", c, d)

else

    @printf("Функция g не унимодальна на интервале [%f, %f] по второй производной\n", c, d)

end

# Поиск минимума методом перебора

step = 0.01

x\_min\_perebor, f\_min\_perebor, iters\_perebor = perebor(f, a, b, step)

println("Метод перебора:")

println("Приближенное значение x\*: ", x\_min\_perebor)

# println("Минимальное значение функции: ", f\_min\_perebor)

println("Количество итераций: ", iters\_perebor)

# Поиск минимума методом бисекции

eps = 1e-5

x\_min\_bisection, f\_min\_bisection, iters\_bisection = bisection(f, a, b, eps)

println("Метод бисекции:")

println("Приближенное значение x\*: ", x\_min\_bisection)

# println("Минимальное значение функции: ", f\_min\_bisection)

println("Количество итераций: ", iters\_bisection)

# Поиск минимума методом золотого сечения

x\_min\_golden, f\_min\_golden, iters\_golden = golden\_section(f, a, b, eps)

println("Метод золотого сечения:")

println("Приближенное значение x\*: ", x\_min\_golden)

# println("Минимальное значение функции: ", f\_min\_golden)

println("Количество итераций: ", iters\_golden)

# Поиск минимума методом Фибоначчи

x\_min\_fibonacci, f\_min\_fibonacci, iters\_fibonacci = fibonacci\_search(f, a, b, eps)

println("Метод Фибоначчи:")

println("Приближенное значение x\*: ", x\_min\_fibonacci)

# println("Минимальное значение функции: ", f\_min\_fibonacci)

println("Количество итераций: ", iters\_fibonacci)

# 3. Результаты

