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

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

```
Функция f унимодальна на интервале [0.000000, 4.000000] по первой производной
Функция f унимодальна на интервале [0.000000, 4.000000] по второй производной
Функция g не унимодальна на интервале [-2.000000, 2.000000] по первой производной
Функция g не унимодальна на интервале [-2.000000, 2.000000] по второй производной
Метод перебора:
Приближенное значение x*: 2.0
Количество итераций: 401
Метод бисекции:
Приближенное значение x*: 2.0000038146972656
Количество итераций: 19
Метод золотого сечения:
Приближенное значение x*: 2.0
Метод Фибоначчи:
Приближенное значение x*: 1.9636363636363638
Количество итераций: 7
```