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 №5

“Градиентный спуск”

ПО КУРСУ:

«Методы оптимизация»

Студент ИУ9-82Б Потребина В. В.

Преподаватель Посевин Д. П.

Москва, 2024 г.

ОГЛАВЛЕНИЕ

1. Цели

Цель данной лабораторной работы — изучение и реализация метода градиентного спуска, предназначенного для поиска минимума многомерных функций. В ходе эксперимента проверяется эффективность метода на различных тестовых функциях, а также анализируется сходимость алгоритма.

2. Практическая реализация

```
function norm(p)
    return sqrt(sum(p .* p))
end


function numerical_gradient(f, x, h=1e-5)
    grad = zeros(length(x))  # Создаём вектор градиента
    for i in eachindex(x)
        x_step = copy(x)  # Делаем копию точки
        x_step[i] += h     # Смещаем только i-ю координату
        grad[i] = (f(x_step) - f(x)) / h  # Производная по i-му направлению
    end
    return grad
end


function gradient_descent(f, x0; α=0.1, tol_x=1e-5, tol_f=1e-5, maxiter=1000)
    x = x0  # Текущая точка
    history = [x]  # История точек
    α_init = α  # Сохраняем начальный шаг

    for k in 1:maxiter
        g = numerical_gradient(f, x)  # Вычисляем градиент вручную

        if norm(g) < tol_f  # Проверка ||∇f(x)|| ≤ ε3
            println("Метод сошелся по норме градиента на $k итерации")
            break
        end

        x_new = x - α * g  # Градиентный шаг

        # Откат шага, если f(x_new) >= f(x)
        while f(x_new) >= f(x)
            α /= 2  # Уменьшаем шаг
            x_new = x - α * g
        end
```

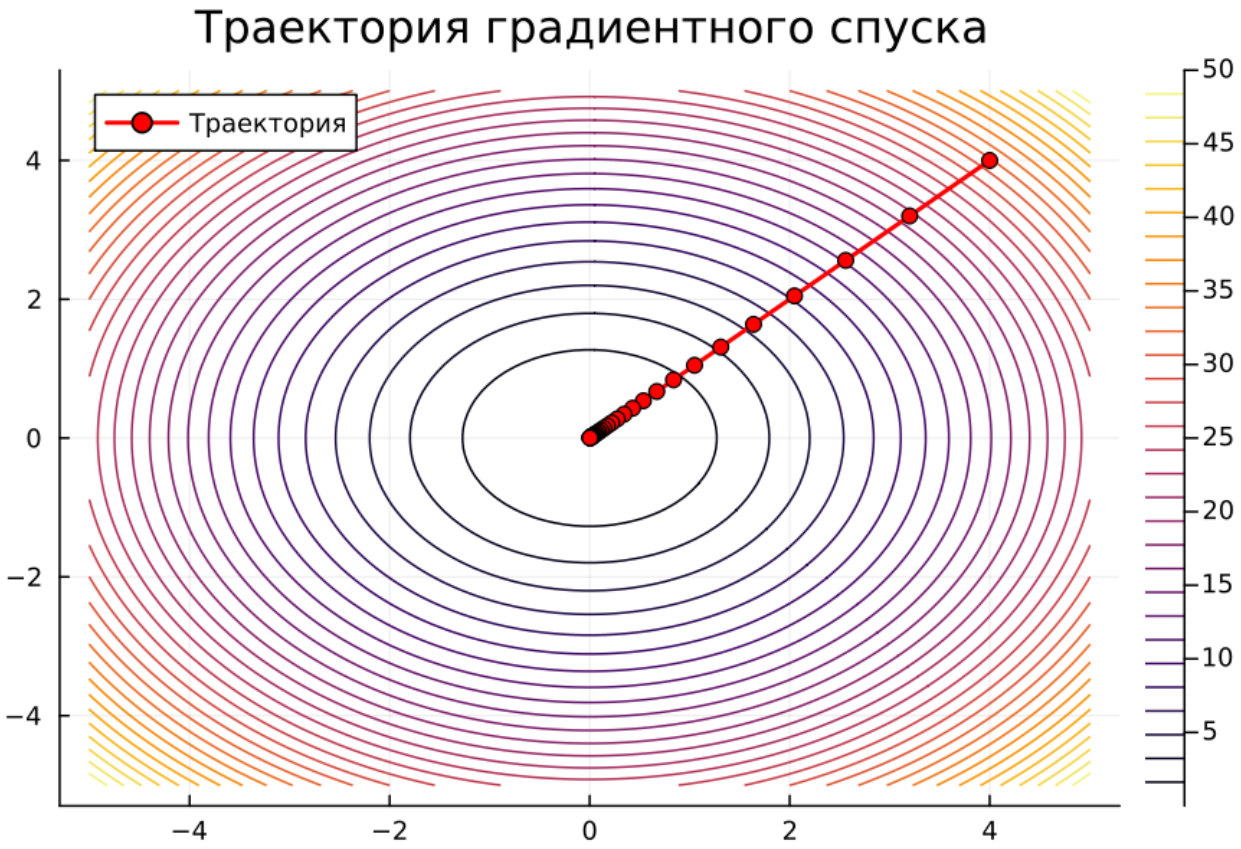
```
        # Условие остановки по изменениям
        if norm(x_new - x) < tol_x || abs(f(x_new) - f(x)) < tol_f
            println("Метод сошелся по критериям остановки на $k итерации")
            break
        end

        x = x_new
        α = α_init  # Сбрасываем α к начальному значению
        push!(history, x)
    end

    return x, history
end


function target_ravine(p::Vector{Float64})
    return sum(p .* p)
end

function target_rastrygin(p)
    A = 10
    result = A*length(p)
    for idx in 1:length(p)
        result += p[idx]^2 - A*cos(2*pi*p[idx])
    end
    return result
end

function target_schefill(p)
    A = 418.9829
    result = A*length(p)
    for idx in 1:length(p)
        result -= p[idx]*sin(sqrt(abs(p[idx])))
    end
    return result
end



x0 = [4.0, 4.0]  # Начальная точка

x_min, history = gradient_descent(target_ravine, x0)

println("Найденный минимум:", x_min)

using Plots

function plot_descent(f, history; xmin=-5, xmax=5, ymin=-5, ymax=5)
    x = range(xmin, xmax, length=100)
```

```
    y = range(ymin, ymax, length=100)
    Z = [f([xi, yi]) for xi in x, yi in y]

    plt = contour(x, y, Z, levels=30, title="Траектория градиентного спуска")
    traj_x = [p[1] for p in history]
    traj_y = [p[2] for p in history]

    plot!(plt, traj_x, traj_y, marker=:circle, color=:red, linewidth=2,
label="Траектория")
    display(plt)
end

plot_descent(target_ravine, history)  # Для первой функции
readline()
```

3. Результаты

Метод сошелся по критериям остановки на 33 итерации

Найденный минимум:[0.0031641304619759642, 0.0031641304619759642]



4. Выводы

Реализованный метод градиентного спуска успешно находит минимум различных функций. Численный расчёт градиента позволяет применять метод к любой функции, даже если её производные неизвестны.