**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 «Информатика и системы управления»

Кафедра «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 8

по курсу « Методы оптимизации »

**BFGS, L-BFGS**

Студент: Яровикова А. С.

Группа: ИУ9-81Б

Преподаватель: Посевин Д. П.

Москва, 2024

СОДЕРЖАНИЕ

[ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ 3](#_Toc158799284)

[ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ 3](#_Toc158799285)

[РЕЗУЛЬТАТЫ 7](#_Toc158799286)

[ВЫВОДЫ 8](#_Toc158799287)

# **ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ**

**Цель работы:**

Реализовать поиск экстремума функции следующими методами: BFGS, L-BFGS.

**Постановка задачи:**

1. Исследовать алгоритмы оптимизации, включая метод BFGS, L-BFGS с целью нахождения локального минимума функции многих переменных.
2. Реализовать численные методы оптимизации на языке программирования Julia, включая функции для вычисления градиента и гессиана, а также методы поиска оптимального шага.
3. Применить алгоритмы оптимизации к заданной функции многих переменных с целью минимизации значения функционала, используя различные начальные условия и параметры методов по вариантам

# **ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ**

Вспомогательные функции:

**import** Pkg

Pkg.add("Plots")

**using** Plots

Pkg.add("LinearAlgebra")

**using** LinearAlgebra

**function df**(f, x::Vector{**Float64**}, i, h=**1e-5**)

x\_plus = copy(x)

x\_minus = copy(x)

x\_plus[i] += h

x\_minus[i] -= h

df\_dx = (f(x\_plus) - f(x\_minus)) / (**2**h)

**return** df\_dx

**end**

Метод Свенна для локализации минимума:

**using** LinearAlgebra

**function swann\_method**(f, x0, h=**0.1**)

first = x0

second = x0 + h

# если функция растет, меняем направление движения

**if** f(second) > f(first)

h = -h

first, second = second, second + h

**end**

last = second + h

# увеличиваем шаг движения, если функция уменьшается

**while** f(last) < f(second)

h \*= **2**

first, second, last = second, last, last + h

**end**

# перепрыгнули далеко

**if** second > last

first, second, last = last, second, first

**end**

**return** first, last

**end**

Метод Золотого сечения для поиска минимума:

**using** LinearAlgebra

**function golden\_section\_search**(f, a, b, eps=**1e-5**)

phi = (sqrt(**5**) - **1**) / **2**

x1 = b - phi \* (b - a)

x2 = a + phi \* (b - a)

**while** abs(b - a) > eps

**if** f(x1) <= f(x2)

b = x2

**else**

a = x1

**end**

x1 = b - phi \* (b - a)

x2 = a + phi \* (b - a)

**end**

**return** (a + b) / **2**

**end**

Метод BFGS:

max\_iter = **100**

d(x) = (**1** - x[**1**])^**2** + **100** \* (x[**2**] - x[**1**]^**2**)^**2**

start = [-**1.2**, **2.0**]

epsilon = **1e-4**

**function bfgs**(f, start, epsilon)

x = copy(start)

points = [copy(x)]

n = length(x)

B = I(n)

gradient = [df(f, x, j) **for** j **in** **1**:n]

i = **0**

**while** norm(gradient) > epsilon && i < max\_iter

s = -B \* gradient

g(alpha) = f(x + alpha \* s)

a, b = swann\_method(g, **0.0**)

alpha = golden\_section\_search(g, a, b)

x\_new = x + alpha \* s

gradient\_new = [df(f, x\_new, j) **for** j **in** **1**:n]

y = gradient\_new - gradient

s = alpha \* s

rho = **1** / dot(y, s)

B = (I - rho \* s \* y') \* B \* (I - rho \* y \* s') + rho \* s \* s'

x = x\_new

gradient = gradient\_new

push!(points, copy(x))

i += **1**

**end**

**return** points, i

**end**

points, i = bfgs(d, start, epsilon)

x\_min = points[**end**]

x = -**20**:**0.1**:**10**

y = -**20**:**0.1**:**20**

x\_coords = [point[**1**] **for** point **in** points]

y\_coords = [point[**2**] **for** point **in** points]

contour(x, y, (x, y) -> d([x, y]), levels = **10**, xlabel="x", ylabel="y", colorbar=false, size=(**500**, **500**))

scatter!(x\_coords, y\_coords, markersize=**2**, markershape=:circle, markercolor=:red, label = "")

p = plot!(x\_coords, y\_coords, label="Движение", line=:red)

println("BFGS:**\n** точка минимума: $x\_min**\n** кол-во итераций = $i")

p

Метод L-BFGS:

**function approx\_inv\_hessian\_lbfgs**(s\_hist, y\_hist, rho\_hist, gradient, m)

q = copy(gradient)

alpha = Vector{**Float64**}(undef, length(s\_hist))

beta = Vector{**Float64**}(undef, length(s\_hist))

**for** i **in** length(s\_hist):-**1**:**1**

alpha[i] = rho\_hist[i] \* dot(s\_hist[i], q)

q -= alpha[i] \* y\_hist[i]

**end**

z = m \* q

**for** i **in** **1**:length(s\_hist)

beta[i] = rho\_hist[i] \* dot(y\_hist[i], z)

z += (alpha[i] - beta[i]) \* s\_hist[i]

**end**

**return** z

**end**

**function lbfgs**(f, start, epsilon, m=**10**)

x = copy(start)

points = [copy(x)]

s\_hist = []

y\_hist = []

rho\_hist = []

gradient = [df(f, x, i) **for** i **in** **1**:length(x)]

i = **0**

**while** norm(gradient) > epsilon && i < max\_iter

s = -approx\_inv\_hessian\_lbfgs(s\_hist, y\_hist, rho\_hist, gradient, m)

g(alpha) = f(x + alpha \* s)

a, b = swann\_method(g, **0.0**)

alpha = golden\_section\_search(g, a, b)

x\_new = x + alpha \* s

gradient\_new = [df(f, x\_new, j) **for** j **in** **1**:length(x)]

y = gradient\_new - gradient

rho = **1** / dot(y, s)

**if** length(s\_hist) == m

popfirst!(s\_hist)

popfirst!(y\_hist)

popfirst!(rho\_hist)

**end**

push!(s\_hist, s)

push!(y\_hist, y)

push!(rho\_hist, rho)

x = x\_new

gradient = gradient\_new

push!(points, copy(x))

i += **1**

**end**

**return** points, i

**end**

points, i = lbfgs(d, start, epsilon)

x\_min = points[**end**]

x = -**20**:**0.1**:**10**

y = -**20**:**0.1**:**20**

x\_coords = [point[**1**] **for** point **in** points]

y\_coords = [point[**2**] **for** point **in** points]

contour(x, y, (x, y) -> d([x, y]), levels = **10**, xlabel="x", ylabel="y", colorbar=false, size=(**500**, **500**))

scatter!(x\_coords, y\_coords, markersize=**2**, markershape=:circle, markercolor=:red, label = "")

p = plot!(x\_coords, y\_coords, label="Движение", line=:red)

println("L-BFGS:**\n** точка минимума: $x\_min**\n** кол-во итераций = $i")

p

# **РЕЗУЛЬТАТЫ**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Точка экстремума** | **Количество итераций** |
| Метод BFGS | (0.9999996870967209, 0.9999992523068931) | 12 |
| Метод L-BFGS | (1.0000000782399192, 1.0000001488451917) | 12 |

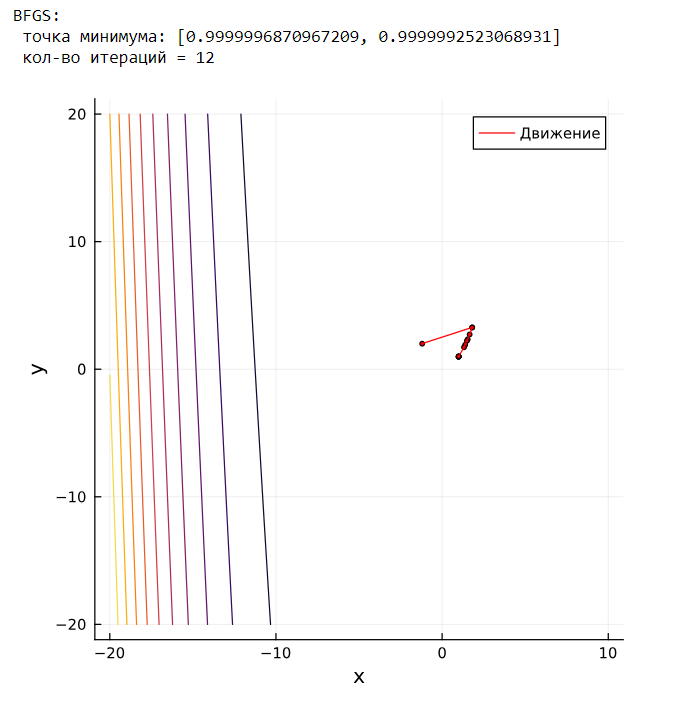


Рисунок 1 - BFGS

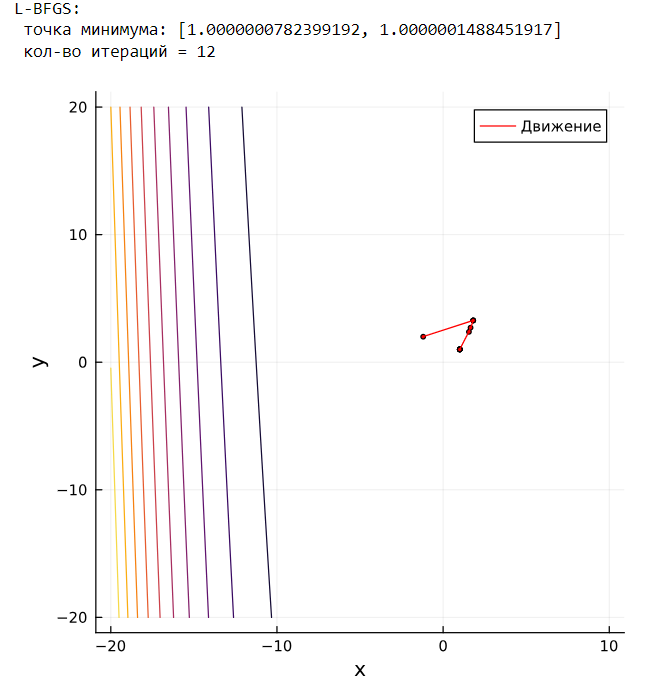


Рисунок 2 – L-BFGS

# **ВЫВОДЫ**

В ходе выполнения лабораторной работы были реализованы и сравнены методы оптимизации многомерных функций. Поиск экстремума был реализован с помощью следующих методов: BFGS, L-BFGS. Оба метода сошлись за 12 итераций, но с разной точностью. L-BFGS отличается лучшей точностью.