**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 № 6

по курсу «Методы оптимизации»

**МЕТОД НЕЛЬДЕРА-МИДА ДЕФОРМИРУЕМЫХ МНОГОГРАННИКОВ**

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 6](#_Toc158799286)

[ВЫВОДЫ 7](#_Toc158799287)

# **ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ**

**Цель работы:**

Реализовать поиск экстремума функции методом Нельдера-Мида деформируемых многогранников.

**Постановка задачи:**

1. Исследовать алгоритм оптимизации метод Нельдера-Мида, с целью нахождения локального минимума функции многих переменных.
2. Реализовать численный метод оптимизации на языке программирования Julia.
3. Применить алгоритмы оптимизации к заданной функции многих переменных с целью минимизации значения функционала, используя различные начальные условия и параметры методов по вариантам

Вариант 1. .

# **ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ**

Симплексный алгоритм:

**using** Plots

**using** LinearAlgebra

**function f**(x)

**return** (x[**1**]+x[**2**])^**2** + (x[**2**]+**2**)^**2**

**end**

xs = []

println("Точка экстремума")

**function simplex\_method**()

x1 = [**0**,**0**]

x2 = [(sqrt(**3**)+**1**)/(**2**\*sqrt(**2**)), (sqrt(**3**)-**1**)/(**2**\*sqrt(**2**))]

x3 = [(sqrt(**3**)-**1**)/(**2**\*sqrt(**2**)), (sqrt(**3**)+**1**)/(**2**\*sqrt(**2**))]

points = [x1,x2,x3]

points[**1**] = x1

points[**2**] = x2

points[**3**] = x3

push!(xs, x1)

push!(xs, x2)

push!(xs, x3)

**while** true

**if**(norm(points[**1**]- points[**2**])) < **0.001**

**return** points[**1**]

**end**

**if**(f(points[**2**]) >= f(points[**1**]) && f(points[**2**]) >= f(points[**3**]))

temp = points[**1**]

points[**1**] = points[**2**]

points[**2**] = temp

**elseif** (f(points[**3**]) >= f(points[**1**]) && f(points[**3**]) >= f(points[**2**]))

temp = points[**1**]

points[**1**] = points[**3**]

points[**3**] = temp

**end**

x4 = points[**2**] + points[**3**] - points[**1**]

**if**(f(x4) >= f(points[**2**]) && f(x4) >= f(points[**3**]))

points[**1**] = x4

points[**2**] = x4+(points[**2**] - x4)/**2**

points[**3**] = x4+(points[**3**] - x4)/**2**

push!(xs, x4)

push!(xs, points[**2**])

push!(xs, points[**3**])

**else**

points[**1**] = x4

push!(xs, x4)

push!(xs, points[**2**])

push!(xs, points[**3**])

**end**

**end**

**end**

r = simplex\_method()

println("Симплексный алгоритм: ", r)

x\_coords\_simplex = [x[**1**] **for** x **in** xs]

y\_coords\_simplex = [y[**2**] **for** y **in** xs]

Метод Нельдера-Мида:

**using** LinearAlgebra

**function nelder\_meed**()

x1 = [**0.0**,**0.0**]

x2 = [(sqrt(**3**)+**1**)/(**2**\*sqrt(**2**)), (sqrt(**3**)-**1**)/(**2**\*sqrt(**2**))]

x3 = [(sqrt(**3**)-**1**)/(**2**\*sqrt(**2**)), (sqrt(**3**)+**1**)/(**2**\*sqrt(**2**))]

points = [x1,x2,x3]

points[**1**] = x1

points[**2**] = x2

points[**3**] = x3

xs = []

center = [**0.0**,**0.0**]

beta = **2.0**

push!(xs, x1)

push!(xs, x2)

push!(xs, x3)

**while** true

points = sort(points, by=x -> f(x), rev=true)

push!(xs, points[**3**])

push!(xs, points[**1**])

push!(xs, points[**2**])

push!(xs, points[**3**])

center = (points[**2**]+points[**3**])/**2.0**

**if** (sqrt(((f(points[**1**]) - f(center))^**2** +((f(points[**2**]) - f(center))^**2**) + ((f(points[**2**]) - f(center))^**2**))/(**3.0**)) < **0.001**)

**return** points[**3**],xs

**end**

x4 = points[**2**]+points[**3**]-points[**1**]

beta = **2.0**

y\_min = f(points[**3**])

**if**(f(x4) < y\_min)

beta = **2.0**

x5 = beta\*x4 + (**1**-beta)\*center

**if**(f(x5)< f(x4) && f(x5)< f(points[**3**]) && f(x5)< f(points[**2**]))

points[**1**] = x5

**else**

**if**(f(x5) > f(x4))

points[**1**] = x4

**end**

**end**

**else**

**if** f(points[**3**]) < f(x4) < f(points[**2**])

points[**1**] = x4

**else**

**if** f(points[**2**]) < f(x4) < f(points[**1**])

points[**1**] = x4

**end**

points = sort(points, by=x -> f(x), rev=true)

beta = **0.5**

x5 = beta \* points[**1**] + (**1** - beta) \* center

**if** f(x5) < f(points[**1**])

points[**1**] = x5

**else**

#c

points[**1**] = points[**3**] + **0.5** \* (points[**1**] - points[**3**])

points[**2**] = points[**3**] + **0.5** \* (points[**2**] - points[**3**])

**end**

**end**

**end**

**end**

**end**

res,xs = nelder\_meed()

println("Нельдер-Мид: ", res)

x\_coords\_nm = [x[**1**] **for** x **in** xs]

y\_coords\_nm = [y[**2**] **for** y **in** xs]

Вывод графиков и результатов:

x = -**2.5**:**0.1**:**2.5**

y = -**2.5**:**0.1**:**2.5**

levels = []

**for** i **in** **1**:**10**

push!(levels, i^**4**\***0.01**)

**end**

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

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

plot!(x\_coords\_nm, y\_coords\_nm, label="нельдер-мид", line=:blue)

# for (i, (x, y)) in enumerate(xs)

# annotate!(x, y, text(string(i), :black, :left, 10))

# end

display(p)

# **РЕЗУЛЬТАТЫ**

|  |  |
| --- | --- |
| **Метод** | **Точка экстремума** |
| Симплексный метод | (1.9978997916637335, -2.0002918714906706) |
| Метод Нельдера-Мида | (1.9803776258664683, -1.9986732989178047) |

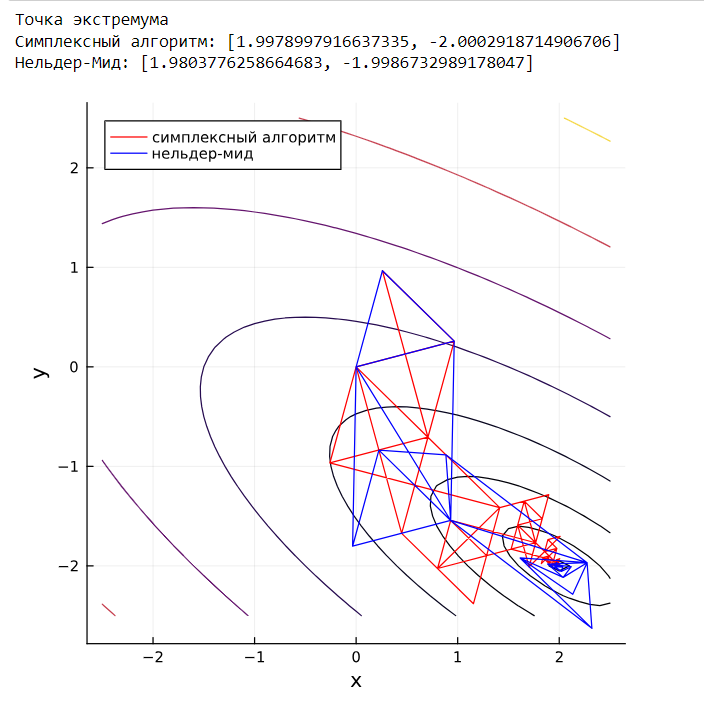


Рисунок 1 - Результат

# **ВЫВОДЫ**

В ходе выполнения лабораторной работы были реализованы и сравнены методы оптимизации многомерных функций. Поиск экстремума был реализован с помощью следующих методов: метод Нельдера-Мида, симплексный алгоритм. Наилучший результат показал метод Нельдера-Мида, который использует растяжение и редукцию симплексов.