**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 № 4

по курсу « Методы оптимизации »

**СРАВНЕНИЕ МЕТОДОВ ОПТИМИЗАЦИИ МНОГОМЕРНЫХ ФУНКЦИЙ – МЕТОДЫ НЬЮТОНА И ПАУЭЛЛА**

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 9](#_Toc158799286)

[ВЫВОДЫ 11](#_Toc158799287)

# **ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ**

**Цель работы:**

Реализовать поиск экстремума функции следующими методами: Ньютона, Ньютона с одномерной оптимизацией, Ньютона с фиксированной обратной матрицей Гессе, метод Пауэлла.

**Постановка задачи:**

1. Исследовать алгоритмы оптимизации, включая метод Ньютона, метод Ньютона с одномерной оптимизацией, метод Ньютона с фиксированной обратной матрицей Гессе и метод Пауэлла, 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

Вариант 1. .

# **ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ**

Вспомогательные функции:

**import** Pkg

Pkg.add("Plots")

**using** Plots

Pkg.add("LinearAlgebra")

**using** LinearAlgebra

**function f**(x)

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

**end**

x0 = [**6.0**,-**1.0**]

eps = **0.18**

**function df\_dx**(func, x, y)

i = **0.001**

**return** (func([x + i, y]) - func([x - i, y])) / (**2**i)

**end**

**function df\_dy**(func,x, y)

i = **0.001**

**return** (func([x, y + i]) - func([x, y - i])) / (**2**i)

**end**

**function d2f\_dx2**(func, x, y)

i = **0.001**

**return** (func([x + i, y]) - **2**\*func([x, y]) + func([x - i, y])) / (i^**2**)

**end**

**function d2f\_dxdy**(func, x, y)

i = **0.001**

**return** (func([x + i, y+i]) - func([x+i, y-i]) - func([x - i, y+i]) + func([x-i, y-i])) / (**4**\*i^**2**)

**end**

**function d2f\_dy2**(func, x, y)

i = **0.001**

**return** (func([x, y + i]) - **2**\*func([x, y]) + func([x, y - i])) / (i^**2**)

**end**

**function anti\_gradient**(func, x)

**return** [(-**1**)\*df\_dx(func, x[**1**],x[**2**]), (-**1**)\*df\_dy(func, x[**1**],x[**2**])]

**end**

**function inv\_hessian**(func, x)

hess = [d2f\_dx2(func,x[**1**],x[**2**]) d2f\_dxdy(func,x[**1**],x[**2**]); d2f\_dxdy(func,x[**1**],x[**2**]) d2f\_dy2(func,x[**1**],x[**2**])]

**return** inv(hess)

**end**

xs = []

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 Ньютона:

**function newton**()

x = x0

iters = **1**

push!(xs, x)

**while** true

g = anti\_gradient(f,x)

h = inv\_hessian(f,x)

x\_new = x + h \* g

push!(xs, x\_new)

**if** norm(g) < eps

**return** x\_new, iters

**end**

x = x\_new

iters += **1**

**end**

**end**

res, its = newton()

println("Экстремум: ", res)

println("Количество итераций: ", its)

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

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

x = -**2.5**:**0.1**:**6**

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

contour(x, y, (x, y) -> f([x, y]), levels=**20**, c=:reds, xlabel="x", ylabel="y", colorbar=false, size=(**600**, **500**))

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

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

display(p)

Метод Ньютона-Рафсона (с одномерной оптимизацией:

**function newton\_rafson**()

x = x0

iters = **1**

alpha = **0.0**

push!(xs, x)

**while** true

g = anti\_gradient(f,x)

h = inv\_hessian(f,x)

gfun(alp) = f(x + alpha\*h \* g)

a, b = swann(gfun, **1**, **0.0001**)

alpha = golden\_section(gfun, a, b, **0.0001**)

x\_new = x + alpha\*h \* g

push!(xs, x\_new)

**if** norm(g) < eps

**return** x\_new, iters

**end**

x = x\_new

iters += **1**

**end**

**end**

res, its = newton\_rafson()

println("Экстремум: ", res)

println("Количество итераций: ", its)

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

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

x = -**2.5**:**0.1**:**6**

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

contour(x, y, (x, y) -> f([x, y]), levels = **20**, xlabel="x", ylabel="y", c=:reds, colorbar=false, size=(**600**, **500**))

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

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

display(p)

Метод Ньютона с фиксированной обратной матрицей Гессе:

**function newton\_with\_fixH**()

x = x0

iters = **1**

alpha = **0.0**

push!(xs, x)

h = inv\_hessian(f,x)

**while** true

g = anti\_gradient(f,x)

gfun(alp) = f(x + alpha\*h \* g)

a, b = swann(gfun, **1**, **0.0001**)

alpha = golden\_section(gfun, a, b, **0.0001**)

x\_new = x + alpha\*h \* g

push!(xs, x\_new)

**if** norm(g) < eps

**return** x\_new, iters

**end**

x = x\_new

iters += **1**

**end**

**end**

res, its = newton\_constH()

println("Экстремум: ", res)

println("Количество итераций: ", its)

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

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

x = -**2.5**:**0.1**:**6**

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

contour(x, y, (x, y) -> f([x, y]), levels = **20**, xlabel="x", ylabel="y", c=:reds, colorbar=false, size=(**600**, **500**))

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

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

display(p)

Метод Пауэлла:

**function square\_approximation**(x0, x1, x2, y0, y1, y2)

a0 = y0

a1 = (y1-y0) / (x1-x0)

a2 = (**1.0** / (x2-x1)) \* ((y2-y0) / (x2-x0) - (y1-y0) / (x1-x0))

x\_star = (x1+x0) / **2.0** - a1 / (**2.0**\*a2)

**return** x\_star

**end**

**function pauell\_method**(delta\_x, delta\_y)

x\_best = copy(x0)

iters = **0**

xs = [x0]

**while** true

# по х

x0 = copy(x\_best)

# x1 x2 - точки по обе стороны от "наилучшей"

x1 = [x0[**1**] + delta\_x, x0[**2**]]

x2 = []

y0 = f(x0)

y1 = f(x1)

**if**(y0 > y1) # т.е. функция минимизируется

x2 = x0 + [**2.0** \* delta\_x, **0.0**]

**else** # т.е. функция наоборот растет (неубывает)

x2 = x0 - [delta\_x, **0.0**]

**end**

y2 = f(x2)

# вычисляем х\* с помощью формулы кв. аппроксимации

x\_star = [square\_approximation(x0[**1**], x1[**1**], x2[**1**], y0, y1, y2), x0[**2**]]

# находим у\_мин и соответствующий ему х\_мин

y\_min = min(y0, y1, y2)

**if** (y\_min == y0)

x\_min = copy(x0)

**elseif**(y\_min == y1)

x\_min = copy(x1)

**else**

x\_min = copy(x2)

**end**

# выбираем наилучшую точку между х\* и х\_\_min

x\_best = []

**if**(y\_min > f(x\_star))

x\_best = copy(x\_star)

**else**

x\_best = copy(x\_min)

**end**

# аналогично по y, тк у нас не одномерный случай

x0 = copy(x\_best)

x1 = x0 + [**0.0**, delta\_y]

x2 = []

y0 = f(x0)

y1 = f(x1)

**if**(y0 > y1)

x2 = x0 + [**0.0**, **2.0** \* delta\_y]

**else**

x2 = x0 - [**0.0**, delta\_y]

**end**

y2 = f(x2)

x\_star = [x0[**1**], square\_approximation(x0[**2**], x1[**2**], x2[**2**], y0, y1, y2)]

y\_min = min(y0, y1, y2)

**if** (y\_min == y0)

x\_min = copy(x0)

**elseif**(y\_min == y1)

x\_min = copy(x1)

**else**

x\_min = copy(x2)

**end**

**if**(y\_min > f(x\_star))

x\_best = copy(x\_star)

**else**

x\_best = copy(x\_min)

**end**

iters += **1**

push!(xs, x\_best)

# проверяем условие

**if**(norm(x\_min - x\_star) <= eps)

**return** x\_best, iters

**end**

**end**

**end**

res, its = pauell\_method(**0.01**, **0.01**)

println("Экстремум: ", res)

println("Количество итераций: ", its)

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

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

x = -**2.5**:**0.1**:**6**

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

contour(x, y, (x, y) -> f([x, y]), levels = **20**, xlabel="x", ylabel="y", c=:reds, colorbar=false, size=(**600**, **500**))

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

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

display(p)

# **РЕЗУЛЬТАТЫ**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Точка экстремума** | **Количество итераций** |
| Метод Ньютона | (2, -2) | 2 |
| Метод Ньютона-Рафсона с одномерной оптимизацией | (2.0000000058363074, -1.9999999985412964) | 2 |
| Метод Ньютона с фиксированной обратной матрицей Гессе | (2.000000005836693, -1.9999999985415733) | 2 |
| Метод Пауэла | (1.7499999999971045, -1.8749999999985534) | 3 |

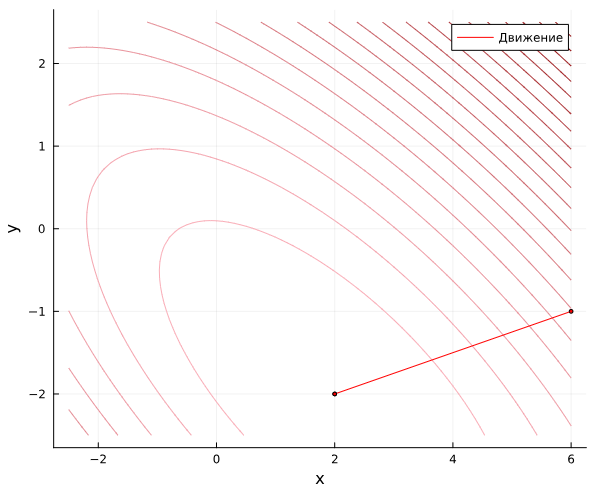


Рисунок 1 - Метод Ньютона

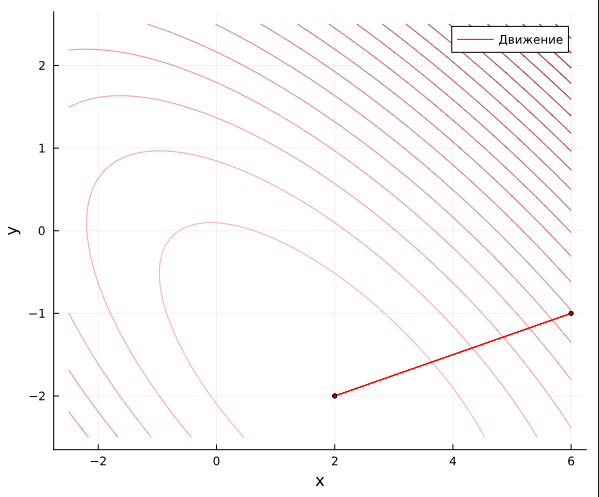


Рисунок 2 - Метод ньютона с одномерной оптимизацией

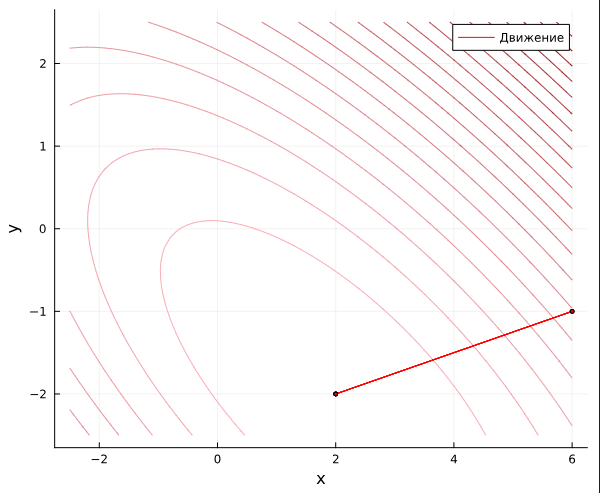


Рисунок 3 - Метод Ньютона с фиксированной матрицей Гессе

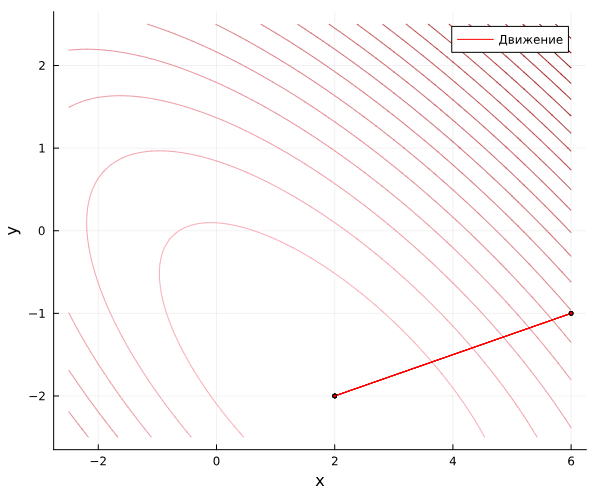


Рисунок 4 - метод Пауэлла

# **ВЫВОДЫ**

В ходе выполнения лабораторной работы были реализованы и сравнены методы оптимизации многомерных функций. Поиск экстремума был реализован с помощью следующих методов: метод Ньютона, метод Ньютона-Рафсона с одномерной оптимизацией, метод Ньютона с фиксированной обратной матрицей Гессе, метод Пауэлла. Наилучший результат показали все методы Ньютона.