**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 ФУНКЦИЙ ДВУХ ПЕРЕМЕННЫХ**

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 11](#_Toc158799286)

[ВЫВОДЫ 21](#_Toc158799287)

# **ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ**

**Цель работы:**

Изучить и провести сравнительный анализ эффективности различных методов оптимизации функций двух переменных с использованием градиентного спуска.

**Постановка задачи:**

1. Изучить и реализовать методы наискорейшего спуска, сопряженных градиентов, Гаусса-Зейделя и покоординатный метод для оптимизации функций двух переменных для различных значений и с использованием градиентного спуска и сравнить их эффективность.
2. Зафиксировать начальные значения итераций для каждой комбинации параметров α и β. Провести серию экспериментов, изменяя α и β, и записать количество итераций, требуемых для сходимости к оптимальному решению.
3. Оптимизировать четырьмя перечисленными методами функцию по варианту

Вариант 1. .

# **ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ**

Вспомогательные функции:

**import** Pkg

Pkg.add("Plots")

**using** Plots

Pkg.add("LinearAlgebra")

**using** LinearAlgebra

alpha, beta = **1**, **1**

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

f(x) = func(x[**1**],x[**2**])

x0 = [**1.0**, **1.0**]

**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**

**function gradient**(f, x::Vector{**Float64**})

grad = []

**for** i **in** **1**:length(x)

push!(grad, df(f, x, i))

**end**

**return** grad

**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 наискорейшего спуска:

**function gradient\_descent**(f, x0)

eps1, eps2 = **1e-6**, **1e-10**

x = x0

trajectory = [copy(x)]

**while** true

grad = gradient(f, x)

prev\_grad = copy(grad)

l, r = swann\_method(alpha -> f(x - alpha \* grad), **1e-2**)

alpha = golden\_section\_search(alpha -> f(x - alpha \* grad), l, r)

x\_new = x - alpha \* grad

**if** norm(x - x\_new) < eps1 || norm(f(x) - f(x\_new)) < eps2

**break**

**end**

x = x\_new

push!(trajectory, copy(x))

**if** dot(prev\_grad, gradient(f, x)) > **1**

println("not ortogonal")

**end**

# println("(grad, prev\_grad) = ", dot(prev\_grad, gradient(f, x)))

**end**

**return** x, trajectory

**end**

x\_min, trajectory = gradient\_descent(f, x0)

print("Точка минимума: ", x\_min, "; Количество итераций: ", length(trajectory)-**1**)

alpha, beta = **10**, **10**

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

f(x) = func(x[**1**],x[**2**])

x0 = [**1.0**, **1.0**]

print()

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

levels = []

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

push!(levels, i^**2**\***0.1**)

**end**

contour(x, y, (x, y) -> f([x, y]), levels = levels, 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 alpha=$alpha, beta=$beta", line=:red)

**for** a **in** [**1**, **5**, **500**], b **in** [**1**, **5**, **500**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = gradient\_descent(f, x0)

println("α=$a, β=$b: Количество итераций= $(length(trajectory)-1)")

**end**

levels = []

**for** i **in** **1**:**15**

push!(levels, i^**2**\***0.1**)

**end**

**for** a **in** [**1**, **10**, **100**], b **in** [**1**, **10**, **100**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = gradient\_descent(f, x0)

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

display(p)

**end**

Метод сопряженных градиентов:

**function conjugate\_gradient**(f, x0)

eps1, eps2 = **1e-6**, **1e-10**

x = x0

prev\_x = copy(x)

grad = gradient(f, x)

d = -grad

trajectory = [x]

**while** true

prev\_grad=copy(grad)

l, r = swann\_method(alpha -> f(x + alpha \* d), **1e-7**)

alpha = golden\_section\_search(alpha -> f(x + alpha \* d), l, r)

x += alpha \* d

grad = gradient(f, x)

**if** norm(x - prev\_x) < eps1 || norm(f(x) - f(prev\_x)) < eps2

**break**

**end**

beta = dot(grad, grad) / dot(prev\_grad, prev\_grad)

d = -grad + beta \* d

prev\_x = copy(x)

push!(trajectory, x)

**end**

**return** x, trajectory

**end**

x\_min, trajectory = conjugate\_gradient(f, x0)

print("Точка минимума: ", x\_min, "; Количество итераций: ", length(trajectory)-**1**)

alpha, beta = **10**, **1**

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

f(x) = func(x[**1**],x[**2**])

x0 = [**1.0**, **1.0**]

print()

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

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

display(p)

**for** a **in** [**1**, **5**, **500**], b **in** [**1**, **5**, **500**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = conjugate\_gradient(f, x0)

println("α=$a, β=$b: Количество итераций= $(length(trajectory)-1)")

**end**

levels = []

**for** i **in** **1**:**15**

push!(levels, i^**2**\***0.1**)

**end**

**for** a **in** [**1**, **10**, **100**], b **in** [**1**, **10**, **100**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = conjugate\_gradient(f, x0)

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

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

display(p)

**end**

Покоординатный метод:

**function coordinate\_descent**(f, x0)

eps1, eps2 = **1e-6**, **1e-10**

x = x0

n = length(x)

trajectory = [x]

alpha = **0.3**

e = []

**for** i **in** **1**:n

ei = zeros(n)

ei[i] = **1**

push!(e, ei)

**end**

**while** true

x\_new = copy(x)

min\_f = f(x)

i1 = **1**

ok = false

**while** true

**for** i **in** **1**:n

**for** j **in** [-**1**, **1**]

xi = copy(x)

xi[i] += alpha\*j

**if** f(xi) <= min\_f

x\_new = xi

i1 = i

min\_f = f(xi)

ok = true

**end**

**end**

**end**

**if** ok

**break**

**end**

alpha /= **2**

**end**

**if** norm(x - x\_new) < eps1 || norm(f(x) - f(x\_new)) < eps2

**break**

**end**

x = copy(x\_new)

push!(trajectory, x)

**end**

**return** x, trajectory

**end**

x\_min, trajectory = coordinate\_descent(f, x0)

print("Точка минимума: ", x\_min, "; Количество итераций: ", length(trajectory)-**1**)

alpha, beta = **10**, **1**

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

f(x) = func(x[**1**],x[**2**])

x0 = [**1.0**, **1.0**]

print()

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

levels = []

**for** i **in** **1**:**20**

push!(levels, i^**2**\***0.1**)

**end**

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

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

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

display(p)

**for** a **in** [**1**, **5**, **500**], b **in** [**1**, **5**, **500**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = coordinate\_descent(f, x0)

println("α=$a, β=$b: Количество итераций = $(length(trajectory)-1)")

**end**

levels = []

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

push!(levels, i^**2**\***0.1**)

**end**

**for** a **in** [**1**, **10**, **100**], b **in** [**1**, **10**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = coordinate\_descent(f, x0)

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

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

display(p)

**end**

Метод Гаусса-Зейделя:

**function coordinate\_descent**(f, x0)

eps1, eps2 = **1e-6**, **1e-10**

x = x0

n = length(x)

trajectory = [x]

alpha = **0.3**

e = []

**for** i **in** **1**:n

ei = zeros(n)

ei[i] = **1**

push!(e, ei)

**end**

**while** true

x\_new = copy(x)

min\_f = f(x)

i1 = **1**

ok = false

**while** true

**for** i **in** **1**:n

**for** j **in** [-**1**, **1**]

xi = copy(x)

xi[i] += alpha\*j

**if** f(xi) <= min\_f

x\_new = xi

i1 = i

min\_f = f(xi)

ok = true

**end**

**end**

**end**

**if** ok

**break**

**end**

alpha /= **2**

**end**

**if** norm(x - x\_new) < eps1 || norm(f(x) - f(x\_new)) < eps2

**break**

**end**

x = copy(x\_new)

push!(trajectory, x)

**end**

**return** x, trajectory

**end**

x\_min, trajectory = coordinate\_descent(f, x0)

print("Точка минимума: ", x\_min, "; Количество итераций: ", length(trajectory)-**1**)

alpha, beta = **10**, **1**

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

f(x) = func(x[**1**],x[**2**])

x0 = [**1.0**, **1.0**]

print()

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

levels = []

**for** i **in** **1**:**20**

push!(levels, i^**2**\***0.1**)

**end**

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

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

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

display(p)

**for** a **in** [**1**, **5**, **500**], b **in** [**1**, **5**, **500**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = coordinate\_descent(f, x0)

println("α=$a, β=$b: Количество итераций = $(length(trajectory)-1)")

**end**

levels = []

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

push!(levels, i^**2**\***0.1**)

**end**

**for** a **in** [**1**, **10**, **100**], b **in** [**1**, **10**]

alpha, beta = a, b

func(x, y) = alpha \* x^**2** + beta \* y ^ **2**

x\_min, trajectory = coordinate\_descent(f, x0)

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

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

x = -**1**:**0.1**:**1**

y = -**1**:**0.1**:**1**

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

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

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

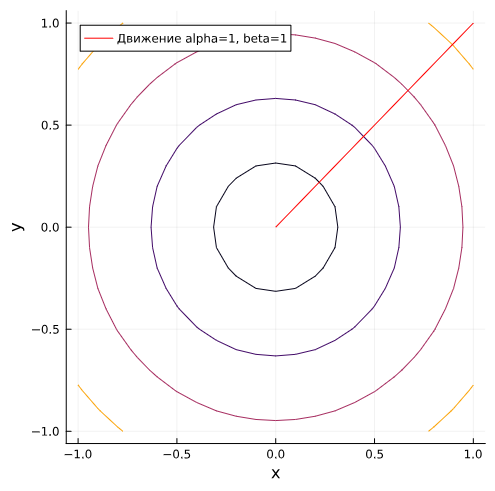
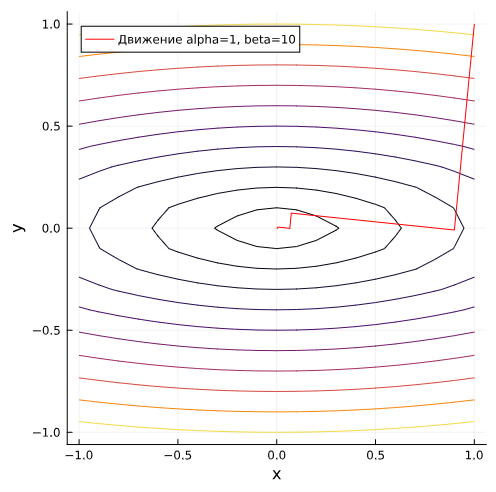
display(p)

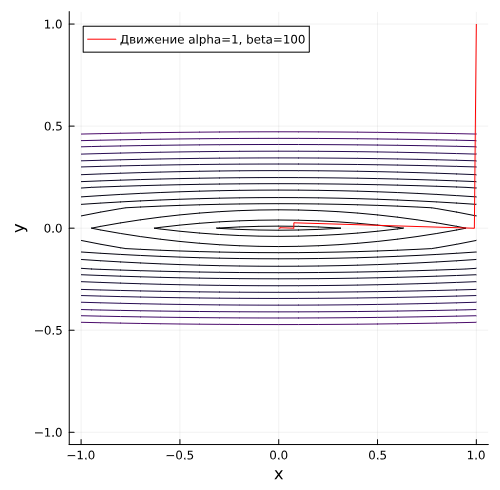
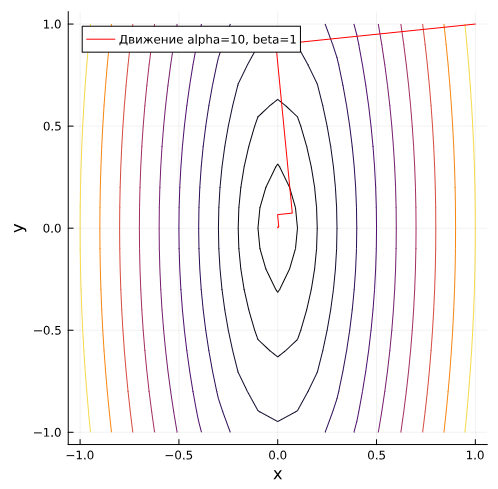
**end**

# **РЕЗУЛЬТАТЫ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
|  | |  |  | |  |
| Метод наискорейшего спуска | | 12 | 2131 | | 110 |
| Метод сопряженных градиентов | | 3 | 6 | | 6 |
| Покоординатный метод | | 35 | 35 | | 37 |
| Метод Гаусса-Зейделя | | 2 | 2 | | 2 |
|  | | | | | |
|  | |  |  |  | |
| Метод наискорейшего спуска | | 1 | 2 | 3 | |
| Метод сопряженных градиентов | | 1 | 1 | 2 | |
| Покоординатный метод | | 34 | 36 | 42 | |
| Метод Гаусса-Зейделя | | 2 | 2 | 2 | |
|  | | | | | |
|  |  | |  | |  |
| Метод наискорейшего спуска | 12 | | 2131 | | 110 |
| Метод сопряженных градиентов | 3 | | 6 | | 6 |
| Покоординатный метод | 35 | | 35 | | 37 |
| Метод Гаусса-Зейделя | 2 | | 2 | | 2 |

Для метода наискорейшего спуска:

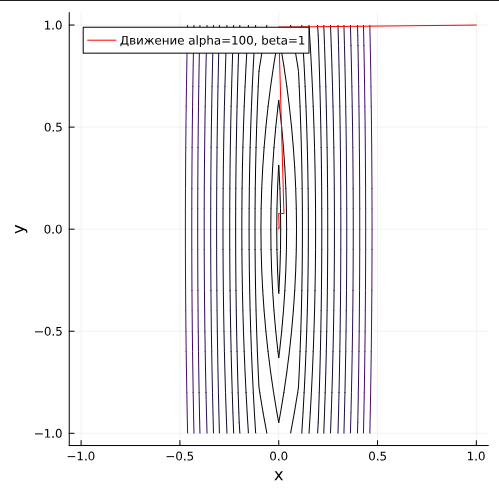
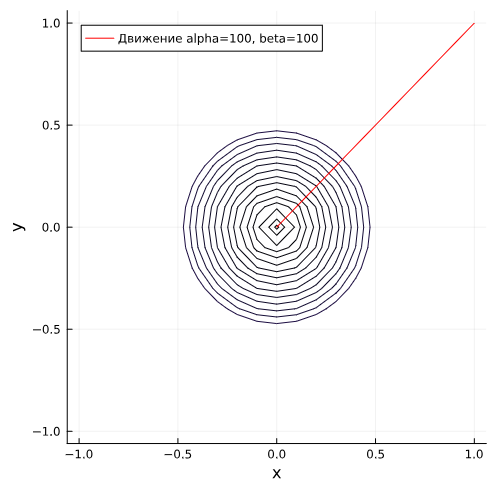
 

Рисунок 1. Метод наискорейшего спуска при различных значениях

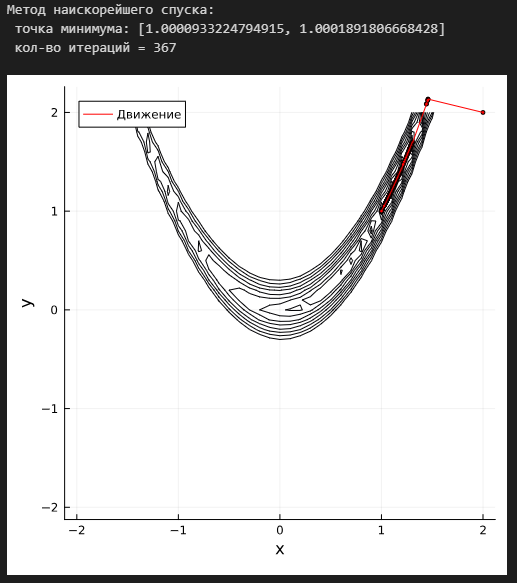


Рисунок 2. Метод наискорейшего спуска для функции Розенброка

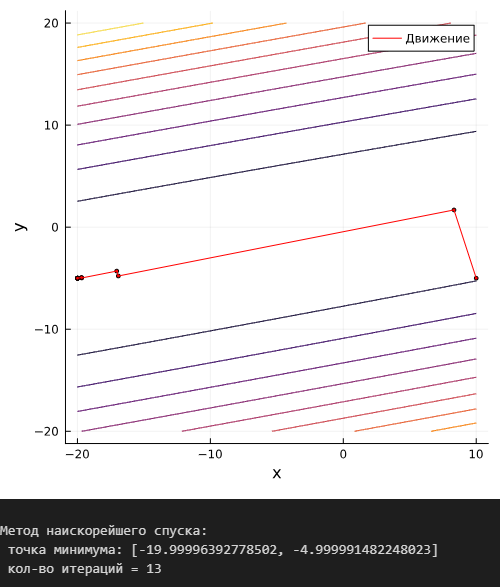
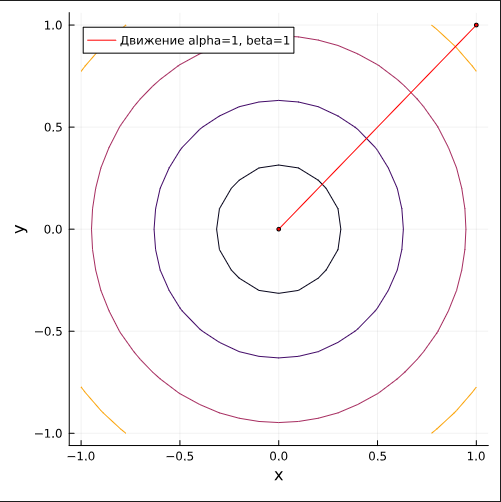
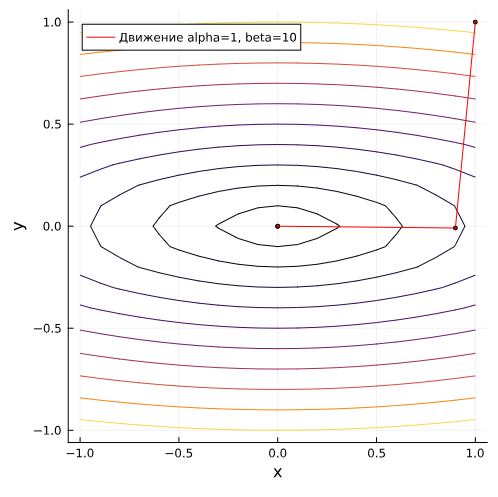
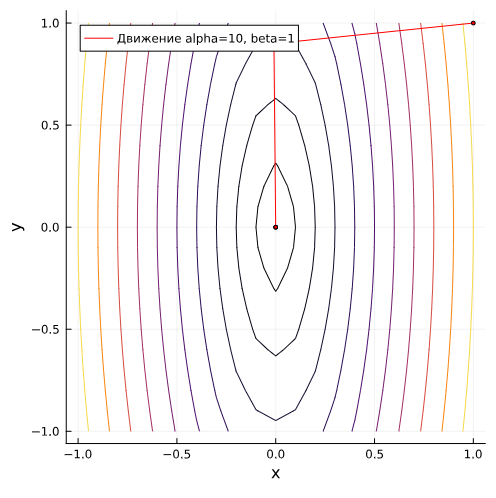
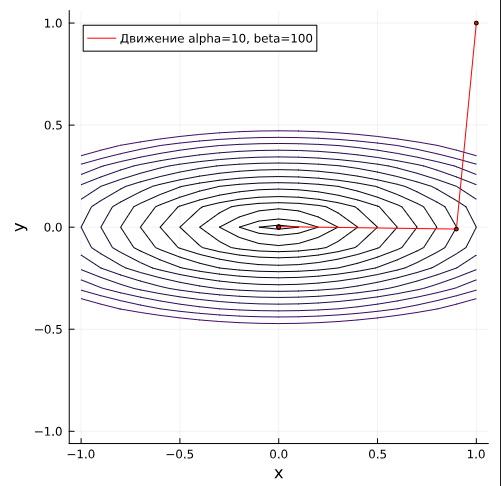


Рисунок 3. Метод наискорейшего спуска для функции по варианту

Для метода сопряженных градиентов:

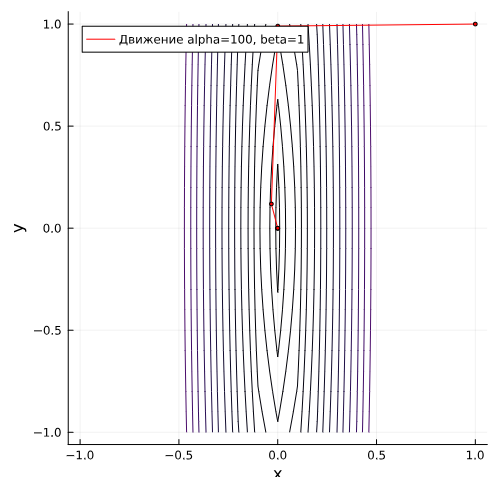
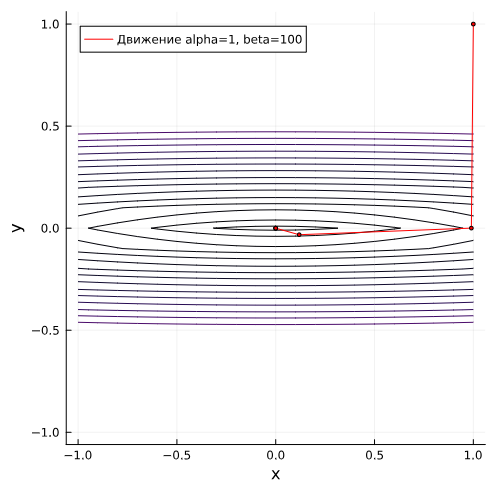
 

Рисунок 4. Метод сопряженных градиентов при различных значениях

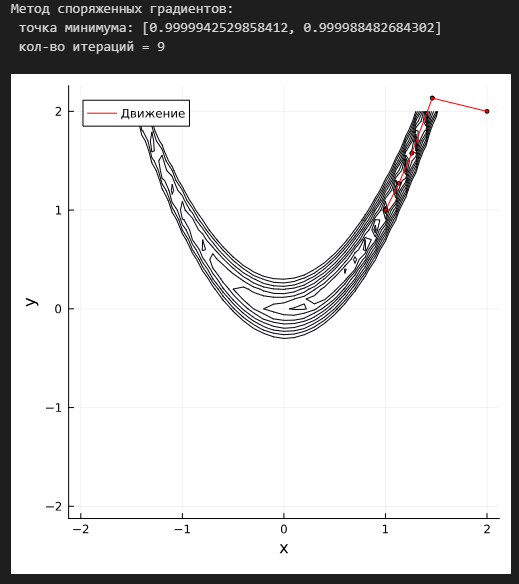


Рисунок 5. Метод сопряженных градиентов для функции Розенброка

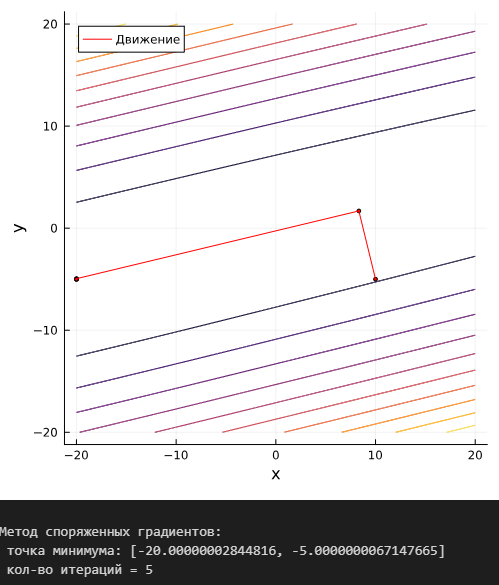
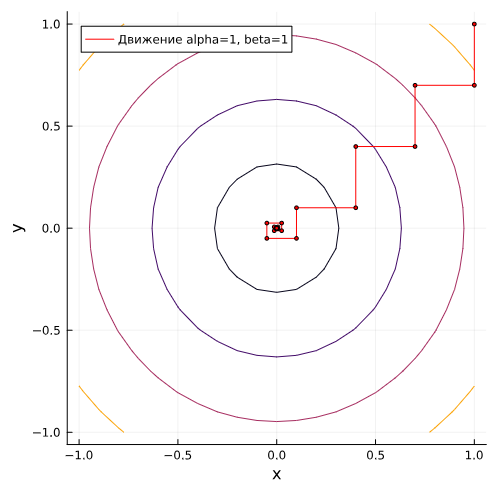
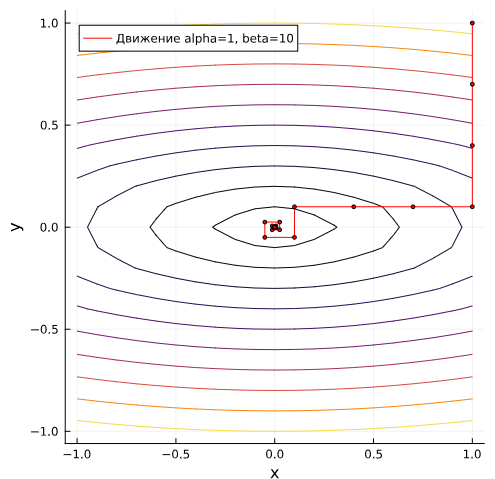
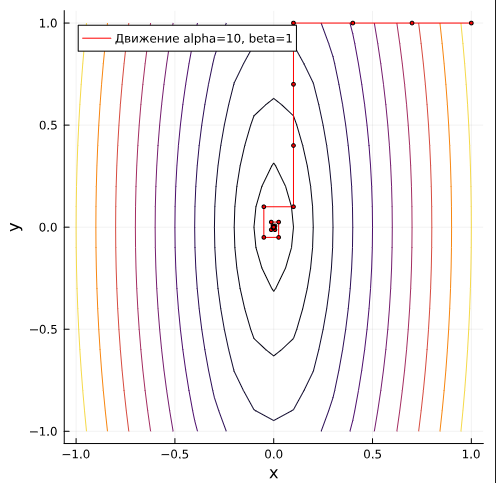
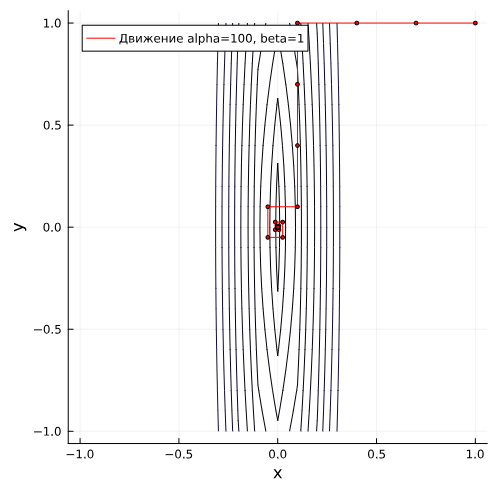


Рисунок 6. Метод сопряженных градиентов для функции по варианту

Для покоординатного метода:

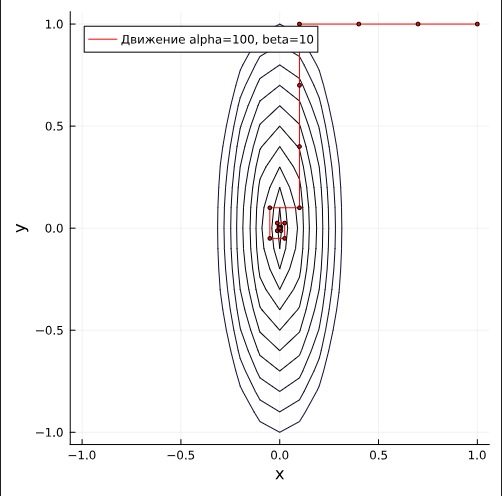
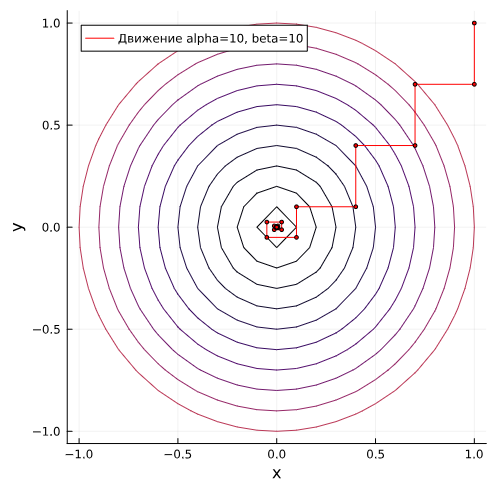
 

Рисунок 7. Покоординатный метод при различных значениях

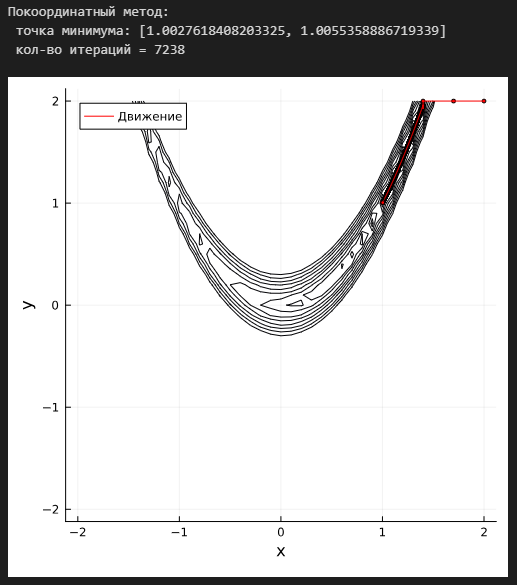


Рисунок 8. Покоординатный метод для функции Розенброка

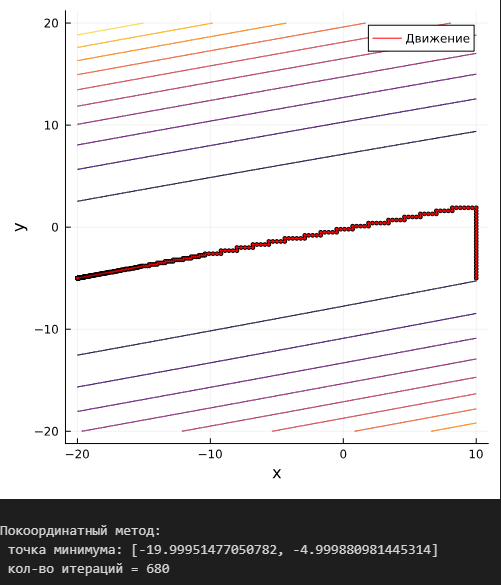
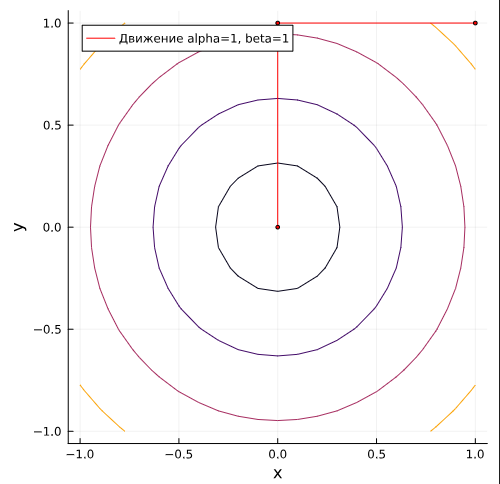
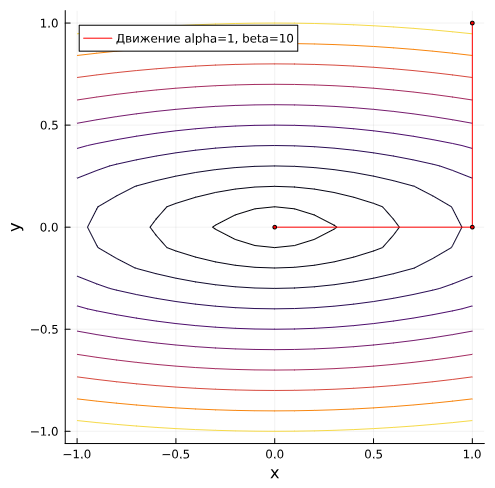
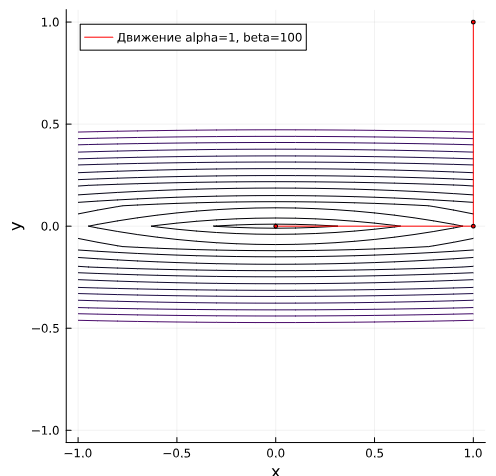
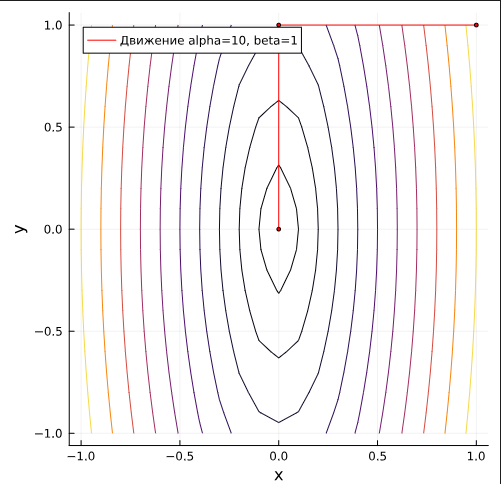


Рисунок 9. Покоординатный метод для функции по варианту

Для метода Гаусса-Зейделя:

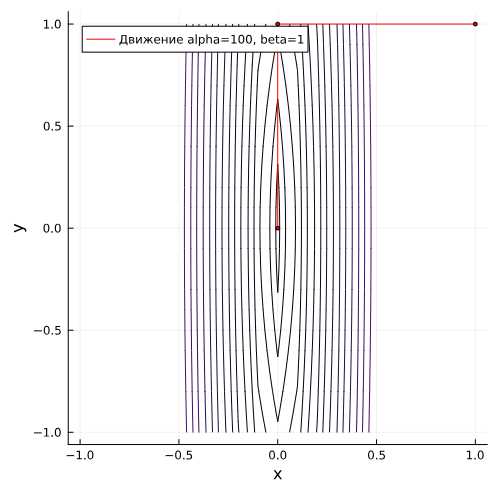
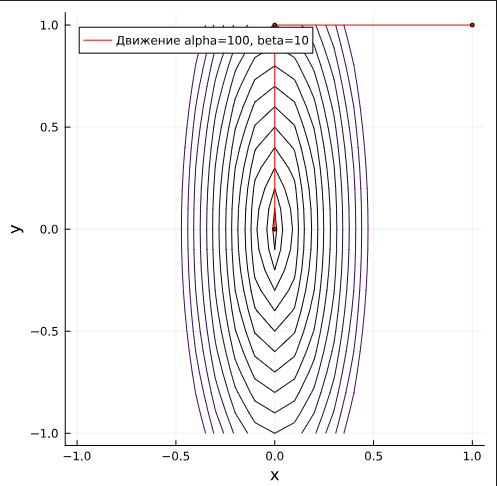
 

Рисунок 10. Метод Гаусса-Зейделя при различных значениях

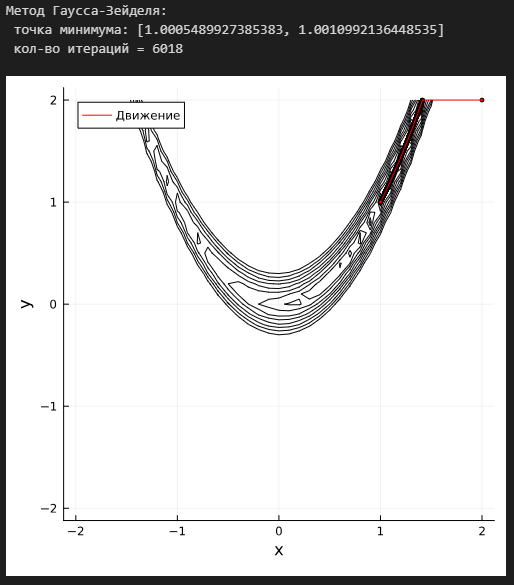


Рисунок 11. Метод Гаусса-Зейделя для функции Розенброка

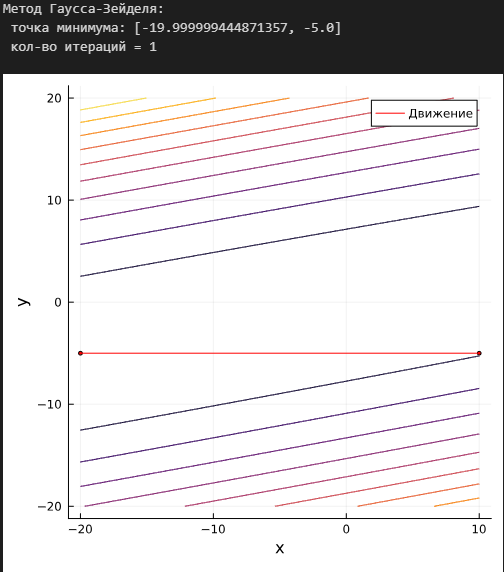


Рисунок 12. Метод Гаусса-Зейделя для функции по варианту

# **ВЫВОДЫ**

В ходе выполнения лабораторной работы были реализованы и сравнены методы оптимизации функции двух переменных: метод наискорейшего спуска, метод сопряженных градиентов, покоординатный метод, метод Гаусса-Зейделя.