Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа №4**

**по дисциплине «Методы оптимизации»**

Вариант: **10**

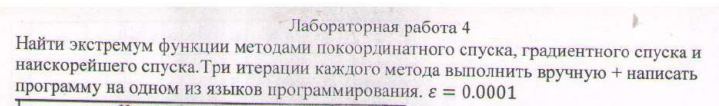
**Преподаватель:**   
Селина Елена Георгиевна

**Выполнил:**

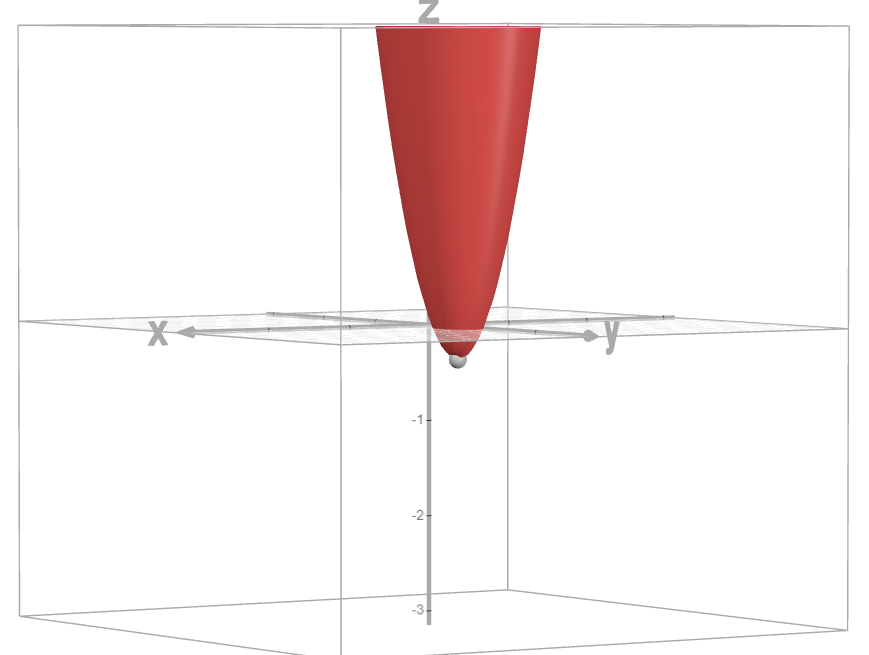
Поленов Кирилл Александрович

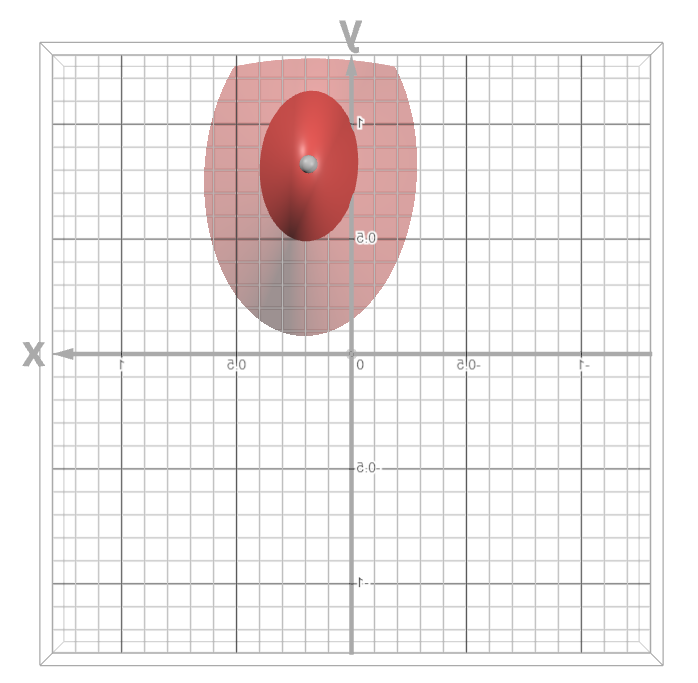
**Группа:** Р3213

Санкт-Петербург, 2025









# 1. Решение вручную

Исходная функция:

1. **Вычисление по методу Поокординатного спуска:**

**Шаг 0:**

x0 = 0.2857

Текущая точка минимума M (0.2857, -2.0000)

y0 = 0.8095

Текущая точка минимума M (0.2857, 0.8095)

**Шаг 1:**

x1 = 0.1854

Текущая точка минимума M (0.1854, 0.8095)

|f\_x(M1) - f\_x(M0)| = -0.0705

y1 = 0.8179

Текущая точка минимума M (0.1854, 0.8179)

|f\_y(M1) - f\_y(M0)| = -0.0002

**Шаг 2:**

x2 = 0.1851

Текущая точка минимума M (0.1851, 0.8179)

|f\_x(M2) - f\_x(M1)| = -0.00000062

-0.00000062<10-4 → |f (M)| <= ε.

Минимум с заданной погрешностью ε = 10-4 найден.

**Минимум достигается в точке** M (0.1851, 0.8179).

**Значение в минимуме** ym = f (M) = -0.3224.

1. **Вычисление по методу Градиентного спуска:**

**Шаг 0:**

M0\_x Компонента Градиента = -0.4000

M0\_y Компонента Градиента = -0.4000

Текущая точка минимума M (-0.4000, -0.4000)

**Шаг 1:**

M1\_x Компонента Градиента = 0.4800

M1\_y Компонента Градиента = 0.3600

|f(M1) - f(M0)| = -6.03200000

Текущая точка минимума M (0.4800, 0.3600)

**Шаг 2:**

M2\_x Компонента Градиента = 0.0900

M2\_y Компонента Градиента = 0.6200

|f(M2) - f(M1)| = -0.98020000

Текущая точка минимума M (0.0900, 0.6200)

**Рассмотрено 3 шага.**

|f(M2) - f(M1)| = -0.98020000 > 10-4 , значит **минимума с заданной точностью** ε = 10-4 **найти за 3 шага не удалось**.

Текущий минимум: M (0.0900, 0.6200).

**Значение в минимуме** ym = f (M) = -0.1322.

1. **Вычисление по методу Наискорейшего спуска:**

**Шаг 0:**

Градиент grad = (24.000000\*i, -16.000000\*j)

Модуль Градиента |grad| = 28.844410

Подстановка в исходную функцию 4608\*h^2 + -832\*h + 44.

Вторая производная от функции шага: 9216. \*h + -832.

Шаг h = 0.0903

Текущая точка минимума M (-0.1667, -0.5556)

**Шаг 1:**

Градиент grad = (-5.611111\*i, -8.416667\*j)

Модуль Градиента |grad| = 10.115574

Подстановка в исходную функцию 456.5262345679\*h^2 + -102.3248456790\*h + 6.4444444444

Вторая производная от функции шага: 913.0524691358\*h + -102.3248456790

Шаг h = 0.1121

Текущая точка минимума M (0.4622, 0.3877)

**Шаг 2:**

Градиент grad = (3.664152\*i, -2.442768\*j)

Модуль Градиента |grad| = 4.403763

Подстановка в исходную функцию 107.4080965534\*h^2 + -19.3931285444\*h + 0.7107246435

Вторая производная от функции шага: 214.8161931068\*h + -19.3931285444

Шаг h = 0.0903

Текущая точка минимума M (0.1314, 0.6082)

**Рассмотрено 3 шага.**

4.403763> 10-4, значит **минимума с заданной точностью** ε = 10-4 **найти за 3 шага не удалось**.

Текущий минимум: M (0.1314, 0.6082).

**Значение в минимуме** ym = f (M) = -0.16465782.

# 2. Программное решение

main.go

|  |
| --- |
| *package* main  *import* (  "fmt"  "math" )  *// Функция // 7x^2+3y^2+0.5xy-3x-5y+2 func* f(x, y float64) float64 {  *return* 7\*x\*x + 3\*y\*y + 0.5\*x\*y - 3\*x - 5\*y + 2 }  *// Метод покоординатного спуска // (x0, y0) - начальное приближение, tolerance - точность, maxIter - чтобы без бредика func* coordinateDescent(x0, y0, tolerance float64, maxIter int) (float64, float64) {  x, y := x0, y0  *for* iter := 0; iter < maxIter; iter++ {  fmt.Println("Итерация ", iter, ":")  *// Минимизация по x* newX := findX(y)   fmt.Printf("x%d = %.4f\n", iter, newX)  fmt.Printf("Текущая точка минимума M(%.4f, %.4f)\n", newX, y)   *if* iter >= 1 {  fmt.Printf("|f\_x(M%d) - f\_x(M%d)| = %.4f\n", iter, iter-1, f(newX, y)-f(x, y))  }   *if* math.Abs(f(newX, y)-f(x, y)) < tolerance {  x = newX  *break* }  x = newX   *// Минимизация по y* newY := findY(x)   fmt.Printf("y%d = %.4f\n", iter, newY)  fmt.Printf("Текущая точка минимума M(%.4f, %.4f)\n", x, newY)   *if* iter >= 1 {  fmt.Printf("|f\_y(M%d) - f\_y(M%d)| = %.4f\n", iter, iter-1, f(x, newY)-f(x, y))  }   *if* math.Abs(f(x, newY)-f(x, y)) < tolerance {  y = newY  *break* }  y = newY  fmt.Println("=================================================================================================")  }  fmt.Println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")  *return* x, y }  *func* findX(y float64) float64 {  *// выражаем x из ч.п. по y  return* (-0.5\*y + 3) / 14 }  *func* findY(x float64) float64 {  *// выражаем y из ч.п. по x  return* (-0.5\*x + 5) / 6 }  *// Метод градиентного спуска func* gradDescent(x0, y0, alpha, tolerance float64, maxIter int) (float64, float64) {  x, y := x0, y0  *for* iter := 0; iter < maxIter; iter++ {  fmt.Println("Итерация ", iter, ":")   *// Шаг против градиента по x* newX := x - alpha\*dfdx(x, y)  *// Шаг против градиента по y* newY := y - alpha\*dfdy(x, y)   fmt.Printf("M%d\_x Компонента Градиента = %.4f\n", iter, newX)  fmt.Printf("M%d\_y Компонента Градиента = %.4f\n", iter, newY)   *if* iter >= 1 {  fmt.Printf("|f(M%d) - f(M%d)| = %.4f\n", iter, iter-1, f(newX, newY)-f(x, y))  }   *if* math.Abs(f(x, y)-f(newX, newY)) < tolerance {  *break* }  x = newX  y = newY   fmt.Printf("Текущая точка минимума M(%.4f, %.4f)\n", x, y)  fmt.Println("=================================================================================================")  }  fmt.Println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")  *return* x, y }  *// Частная производная по x func* dfdx(x, y float64) float64 {  *return* 14\*x + 0.5\*y - 3 }  *// Частная производная по y func* dfdy(x, y float64) float64 {  *return* 6\*y + 0.5\*x - 5 }  *func* calcRatios(x, y, dx, dy float64) []float64 {  res := make([]float64, 0)  *// кэф перед x^2* res = append(res, 7\*dx\*dx+3\*dy\*dy+0.5\*dx\*dy)  *// кэф перед x* res = append(res, 7\*(x\*dx\*2)+3\*(y\*dy\*2)+0.5\*(dx\*y+x\*dy)-3\*dx-5\*dy)  *// свободный кэф* res = append(res, 7\*x\*x+3\*y\*y+0.5\*(x\*y)-3\*x-5\*y+2)   *return* res }  *func* findStep(ratios []float64) float64 {  stepEqRatios := make([]float64, 0)  *// производная функции от шага* stepEqRatios = append(stepEqRatios, 2\*ratios[0])  stepEqRatios = append(stepEqRatios, ratios[1])  fmt.Printf("Вторая производная от функции шага: %.10f\*h + %.10f\n", stepEqRatios[0], stepEqRatios[1])   *// приравняли к нулю и выразили шаг* step := -stepEqRatios[1] / stepEqRatios[0]  *return* step }  *func* fastestDescent(x0, y0, tolerance float64, maxIter int) (float64, float64) {  x, y := x0, y0  ratios := make([]float64, 3)  step := 0.0  *for* iter := 0; iter < maxIter; iter++ {  fmt.Println("Итерация ", iter, ":")  *// ч. п.* derivX := dfdx(x, y)  derivY := dfdy(x, y)  fmt.Printf("Градиент grad = (%.6f\*i, %.6f\*j)\n", derivX, derivY)  fmt.Printf("Модуль Градиента |grad| = %.6f\n", math.Abs(math.Sqrt(derivX\*derivX+derivY\*derivY)))   *if* math.Abs(math.Sqrt(derivX\*derivX+derivY\*derivY)) < tolerance {  *break* }  *// подставляем в исходную функцию уравнения для шага  // x\_new = x\_prev - step \* df/dx  // y\_new = y\_prev - step \* df/dy* ratios = calcRatios(x, y, -derivX, -derivY)  fmt.Printf("Подстановка у исходную функцию %.10f\*h^2 + %.10f\*h + %.10f\n", ratios[0], ratios[1], ratios[2])  *// считаем шаг через вторую производную + приравнивание к нулю* step = findStep(ratios)  fmt.Printf("Шаг h = %.4f\n", step)   *// новое приближение* newX := x - step\*derivX  newY := y - step\*derivY   x = newX  y = newY   fmt.Printf("Текущая точка минимума M(%.4f, %.4f)\n", x, y)  fmt.Println("=================================================================================================")  }  fmt.Println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")  *return* x, y }  *func* main() {  x0, y0 := 2.0, -2.0 *// Начальные приближения* tolerance := 0.0001 *// Точность* maxIter := 1000 *// Максимальное число итераций* fmt.Println("Метод Покоординатного Спуска")  fmt.Println("-------------------------------------------------------------------------------------------------")  xMin, yMin := coordinateDescent(x0, y0, tolerance, maxIter)  fmt.Printf("Минимум функции достигается в точке (x = %.4f, y = %.4f) со значением f(x,y) = %.4f\n", xMin, yMin, f(xMin, yMin))  fmt.Println("=================================================================================================\n")   fmt.Println("Метод Градиентного Спуска")  fmt.Println("--------------------------------------------------------------------------------------------------")  xMin, yMin = gradDescent(x0, y0, 0.1, tolerance, maxIter)  fmt.Printf("Минимум функции достигается в точке (x = %.4f, y = %.4f) со значением f(x,y) = %.4f\n", xMin, yMin, f(xMin, yMin))  fmt.Println("=================================================================================================\n")   fmt.Println("Метод Градиентного Спуска")  fmt.Println("--------------------------------------------------------------------------------------------------")  xMin, yMin = fastestDescent(x0, y0, tolerance, maxIter)  fmt.Printf("Минимум функции достигается в точке (x = %.4f, y = %.4f) со значением f(x,y) = %.4f\n", xMin, yMin, f(xMin, yMin))  fmt.Println("=================================================================================================\n") } |

# Вывод

В ходе выполнения лабораторной работы я научился находить минимум функции нескольких переменных методами Покоординатного, Градиентного и Наискорейшего спусков. Реализовал все методы на языке GoLang. В результате работы были найдены минимум уравнения на отрезке с определенной точностью.