Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 №5

по курсу «Численные методы»

«Метод наискорейшего спуска поиска минимума функции многих переменных»

Студент:

Группа: ИУ9-61Б

Преподаватель: Домрачева А.Б.

Москва 2025

1 Постановка задачи

Дано: Функция двух переменных:

$$f(x_1, x_2) = \exp(x_1) + (x_1 + x_2)^2.$$

Начальное приближение: $x^0 = (1, 1)$.
Найти:

1. Найти минимум функции с точностью $\varepsilon = 0.001$ методом наискорейшего спуска.
2. Найти минимум аналитически.
3. Сравнить полученные результаты с аналитическим решением.

2 Основные теоретические сведения

2.1 Метод наискорейшего спуска

Метод наискорейшего спуска — это итерационный метод, используемый для нахождения минимума функции. Для функции $f(x_1, x_2)$ процесс выглядит следующим образом:

1. На k-м шаге вычисляется градиент функции в точке x^k:

$$\nabla f(x^k) = \left(\frac{\partial f}{\partial x_1}(x^k), \frac{\partial f}{\partial x_2}(x^k) \right).$$

2. Проверяется условие остановки:

$$\|\nabla f(x^k)\| = \max_{1 \le i \le n} \left| \frac{\partial f}{\partial x_i}(x^k) \right| < \varepsilon.$$

3. Если условие остановки не выполнено, определяется направление спуска $d^k = -\nabla f(x^k)$.

4. Рассматривается функция одной переменной:

$$\varphi_k(t) = f(x^k + t \cdot d^k) = f(x^k - t\nabla f(x^k)).$$

5. Находится t^*, минимизирующее $\varphi_k(t)$, с помощью метода одномерной оптимизации.

6. Обновляется точка:

$$x^{k+1} = x^k + t^* \cdot d^k.$$

В данном случае для поиска t^* используется метод парабол, который аппроксимирует $\varphi_k(t)$ квадратичной функцией и находит её минимум.

3 Аналитическое решение

Найдём минимум функции $f(x_1, x_2) = \exp(x_1) + (x_1 + x_2)^2$ аналитически.

Градиент:

$$\frac{\partial f}{\partial x_1} = \exp(x_1) + 2(x_1 + x_2), \quad \frac{\partial f}{\partial x_2} = 2(x_1 + x_2).$$

Приравниваем градиент к нулю для поиска стационарных точек:

$$\begin{cases} \exp(x_1) + 2(x_1 + x_2) = 0, \\ 2(x_1 + x_2) = 0. \end{cases}$$

Из второго уравнения: $x_1 + x_2 = 0 \implies x_2 = -x_1$.

Подставляем $x_2 = -x_1$ в первое уравнение:

$$\exp(x_1) + 2(x_1 + (-x_1)) = \exp(x_1) = 0.$$

Так как $\exp(x_1) > 0$ для всех x_1, стационарных точек в конечной области нет. Подставим $x_2 = -x_1$ в функцию:

$$f(x_1, -x_1) = \exp(x_1) + (x_1 + (-x_1))^2 = \exp(x_1).$$

Производная $g(x_1) = \exp(x_1)$:

$$\frac{d}{dx_1}\exp(x_1) = \exp(x_1),$$

которая никогда не равна нулю. Однако $\exp(x_1)$ монотонно убывает при $x_1 \to -\infty$, и $f(x_1, -x_1) \to 0$ при $x_1 \to -\infty$, $x_2 = -x_1 \to +\infty$.

Вывод: Функция не имеет конечного минимума. Минимальное значение $f \to 0$ достигается при $x_1 \to -\infty$, $x_2 = -x_1$.

4 Реализация

```
package main

import (
	"fmt"
	"math"
)

type Vector []float64

func Function3(x Vector) float64 {
	return math.Exp(x[0]) + math.Pow(x[0]+x[1], 2)
}

func Function(x Vector) float64 {
	return 7*math.Pow(x[0], 2) + 2*x[0]*x[1] + 5*math.Pow(x[1], 2) + x[0] - 10*x[1]
}

func Gradient(x Vector) Vector {
	dfdx0 := 14*x[0] + 2*x[1] + 1
	dfdx1 := 2*x[0] + 10*x[1] - 10
	return Vector{dfdx0, dfdx1}
}

func Gradient3(x Vector) Vector {
	dfdx0 := math.Exp(x[0]) + 2*(x[0]+x[1])
	dfdx1 := 2 * (x[0] + x[1])
	return Vector{dfdx0, dfdx1}
}

func GradNormInf(g Vector) float64 {
	return math.Max(math.Abs(g[0]), math.Abs(g[1]))
}
```

```go

func parabolaMethod(f func(float64) float64, a, b, eps float64) float64 {
  const maxIter = 100
  x1, x2, x3 := a, (a+b)/2, b
  f1, f2, f3 := f(x1), f(x2), f(x3)

  for i := 0; i < maxIter; i++ {
    // строим параболу через 3 точки
    A := (f3 - (x3*(f2-f1)+x2*f1-x1*f2)/(x2-x1)) / (x3*x3 - x3*(x1+x2) + x1*x2)
    B := (f2 - f1 - A*(x2*x2-x1*x1)) / (x2 - x1)
    // находим ее вершину
    xv := -B / (2 * A)

    // если вершина параболы близка к предыдущей точке x2, то выходим
    if math.Abs(x2-xv) < eps {
      return xv
    }

    if xv < x2 {
      // если xv слева от x2, то мы сдвигаем правый край
      x3, f3 = x2, f2
      x2, f2 = xv, f(xv)
    } else {
      // если xv справа от x2, то сдвигаем левый край
      x1, f1 = x2, f2
      x2, f2 = xv, f(xv)
    }
  }
  return (a + b) / 2
}

func SteepestDescent(x0 Vector, eps float64, maxIter int) (Vector, []float64,
    int) {
  x := make(Vector, len(x0))
  copy(x, x0)
  history := []float64{Function(x)}

  for k := 0; k < maxIter; k++ {
    grad := Gradient(x)

    // условие остановки
    if GradNormInf(grad) < eps {
      return x, history, k + 1
    }

    d := make(Vector, len(grad))
```

```
    for i := range d {
      // берём направление наискорейшего убывания
      d[i] = -grad[i]
    }

    // строим функцию phi(t) = f(x + t * d)
    phi := func(t float64) float64 {
      newX := make(Vector, len(x))
      for i := range newX {
        newX[i] = x[i] + t*d[i]
      }
      return Function(newX)
    }
    t := parabolaMethod(phi, 0, 1, 1e-5)

    for i := range x {
      x[i] += t * d[i]
    }
    history = append(history, Function(x))
  }
  return x, history, maxIter
}

func main() {
  x0 := Vector{1.0, 1.0}
  eps := 1e-3
  maxIter := 1000

  solution, _, iterations := SteepestDescent(x0, eps, maxIter)

  // Вывод результатов
  fmt.Printf("Начальная точка: x = [%.6f, %.6f] \n", x0[0], x0[1])
  fmt.Printf("Численное решение: x = [%.6f, %.6f]\n", solution[0], solution[1])
  fmt.Printf("Значение функции численное(): f(x) = %.6f\n", Function(solution))
  fmt.Printf("Количество итераций: %d\n", iterations)
}
```

Листинг 1: Метод наискорейшего спуска

5 Результаты

Таблица 1: Результаты метода наискорейшего спуска

Метод	Точка минимума	Значение $f(x)$	Итераций
Наискорейший спуск	$(-6.4446, 6.4441)$	0.001589	586
Аналитическое	$x_1 \to -\infty$, $x_2 = -x_1$	$\to 0$	—
Пример (аналитическое)	$(-10.0, 10.0)$	0.000045	—

6 Вывод

Метод наискорейшего спуска успешно нашёл приближение к минимуму функции с точностью $\varepsilon = 0.001$. Численное решение $x_1 \approx -6.4446, x_2 \approx 6.4441$, $f \approx 0.001589$ соответствует аналитическому поведению: $x_1 \to -\infty$, $x_2 = -x_1$, $f \to 0$. Для сравнения выбрана точка $x_1 = -10$, $x_2 = 10$, где $f \approx 0.000045$. Разница между численным и аналитическим значением функции составила 0.001544, что близко к заданной точности ε.

Метод наискорейшего спуска потребовал 586 итераций, что указывает на медленную сходимость. Использование метода парабол для поиска t^* обеспечило точное определение направления спуска. Графический анализ и аналитическое решение подтверждают отсутствие конечного минимума, но численный метод позволяет найти приближение, пригодное для практических целей.