Федеральное государственное бюджетное образовательное учреждение высшего образования
"Уфимский государственный авиационный технический университет"

Кафедра Высокопроизводительных вычислительных технологий и систем

Дисциплина: Методы оптимизации

Отчет по лабораторной работе № 1

Тема: «Методы многомерной минимизации (безусловный экстремум)»

Группа ПМ-453	Фамилия И.О.	Подпись	Дата	Оценка
Студент	Шамаев И.Р.			
Принял	Казакова Т.Г.			

Уфа 2022

Цель работы. Приобретение навыков численного решения задач поиска безусловного экстремума действительной функции от n переменных.

Задачи:

1. Ознакомиться с постановкой задач, определяемых вариантом задания к лабораторной работе.
2. Найти решение поставленной задачи условной оптимизации, используя теоремы о необходимых и достаточных условиях.
3. Найти решение задачи безусловной оптимизации для заданной целевой функции в пакете Maple (Optimization).
4. Найти приближенное решение задачи согласно варианту, с заданной точностью ε= 0.01.
5. Провести анализ найденного приближенного решения (является ли стационарная точка точкой экстремума).
6. Ответьте на вопросы, указанные в задании.
7. По результатам выполненной лабораторной работы составьте отчет.

Теоретическая часть.

Метод сопряженных градиентов

1. Задать значения $k = 0, x^k \in D, \varepsilon > 0, M$ – допустимое число итераций.
2. Вычислить $d_k = -\nabla f(x^k)$. Найти λ_k, что $f(x^k + \lambda_k d_k) = \min_{\lambda \geq 0} f(x^k + \lambda d_k)$.
3. Вычислить $x^{k+1} = x^k + \lambda_k d_k$.Если $||d_k|| < \varepsilon$, то поиск решения x^* завершен. $x*$=$x\ k$+1. В противном случае увеличить номер шага k =k +1 и перейти к пункту 4.
4. Новое направление поиска $d_k = -\nabla f(x^k) + \omega_k d_{k-1}, \omega = \frac{\left|\left|\nabla f(x^k)\right|\right|^2}{\left|\left|\nabla f(x^{k-1})\right|\right|^2}$.

   Вычислить $x^{k+1} = x^k + \lambda_k d_k$.
5. Если $||d_k|| < \varepsilon$ или k +1>M , то поиск решения x^* завершен. $x^{k+1} = x^*$. В противном случае увеличить номер шага k =k +1 и перейти к пункту 2.

Метод Марквардта

1. Зададим следующие значения: $k = 0, x^k \in D; \lambda_0$ - некоторое большое значение, например, $\lambda_0 = 10000$; $\varepsilon > 0$ требуемая точность решения; М - допустимое число итераций.
2. Если $\left|\left|\nabla \mathrm{f}(\mathrm{x}^k)\right|\right| < \varepsilon$, то поиск решения x^* завершен. $x^* = x^k$. В противном случае переходим к пункту 3.
3. Вычислим $d_k = -[H(x^k) + \lambda_k E]^{-1} \nabla f(x^k)$
4. Вычислим $x^{k+1} = x^k + d_k$
5. Проверим выполнение неравенства $f(x^{k+1}) < f(x^k)$. Если выполняется, то следующий шаг 6, если нет - шаг 7.
6. $\lambda_{k+1} = \frac{1}{2}\lambda_k, k = k + 1$. Перейти к шагу 2.
7. $\lambda_k = 2\lambda_k$. Вернуться к шагу 3.

Индивидуальное задание (2 вариант).

Целевая функция имеет вид: $f(x_1, x_2) = (x_1 + 4x_2)^2 + (x_1 - 3)^2$.

- Найдем решение задачи безусловной оптимизации для заданной целевой функции, используя теоремы о необходимых и достаточных условиях экстремума.

$$\frac{\partial f}{\partial x_1} = 4x_1 + 8x_2 - 6 = 0$$

$$\frac{\partial f}{\partial x_2} = 8(x_1 + 4x_2) = 0$$

Отсюда следует, что $x_1 = 3, x_2 = -\frac{3}{4}$.

Матрица Гессе $H = \begin{pmatrix} 4 & 8 \\ 8 & 32 \end{pmatrix}$.

$\Delta_1 = 4 > 0, \Delta_2 = 64 > 0 => H$ определана положительно

Т.к. матрица Гессе положительно определена, то в точке $(3, -\frac{3}{4})$ функция $(x_1 + 4x_2)^2 + (x_1 - 3)^2 \rightarrow min, f_{min} = 0$.

Так как H (x)>0, то функция f (x) является строго выпуклой

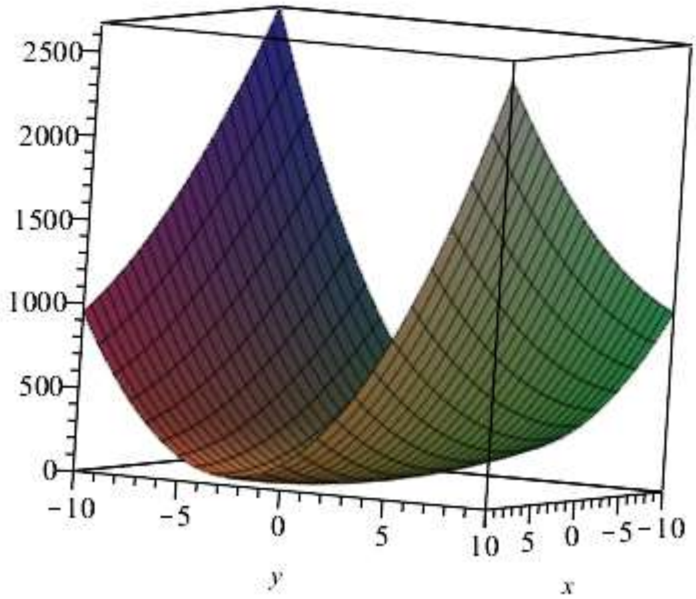

Рисунок 1 График целевой функции.

Из рисунка выше можно убедится, что точке $(3, -\frac{3}{4})$ локального минимума является и точкой глобального минимума.

- Найдем решение задачи безусловной оптимизации для заданной целевой функции в пакете Maple (Optimization).

```
> f := (x + 4·y)^2 + (x − 3)^2;
                    f := (x + 4 y)^2 + (x − 3)^2
>
> Optimization[Minimize](f);
     [0., [x = 3.00000000000000, y = −0.750000000000000]]
>
```

Рисунок 2. Решение задач в maple.

- Найдем приближенное решение задачи безусловной минимизации численными методами поиска безусловного экстремума согласно варианту, с заданной точностью $\varepsilon = 0.01$.

1. Метод сопряженных градиентов.

Листинг программы содержится в Приложении, Пункт 1.

Пример выполнения программы.

```
2.997900679;-0.749440263
f(x1,x2) = 0.000004427
k = 13
```

Рисунок 3. Пример выполнения программы.

2. Метод Марквардта.

Листинг программы содержится в Приложении, Пункт 2.

Пример выполнения программы.

```
(3.00309496 , -0.75082191),     F_min = 9.6158918e-06
k = 10

D:\GoogleDrive\!Study\!Lab_Work\Методы Оптимизации\МО_лаб
л работу с кодом 0.
```

Рисунок 4. Пример выполнения программы.

Вывод

Таким образом, в ходе выполнения лабораторной работы приобрели навыки численного решения задач поиска безусловного экстремума действительной функции от n переменных, соответствующие методы были программно реализованы.

Ответы на контрольные вопросы

1. **Необходимые и достаточные условия локального экстремума функции $f: D \to \mathbb{R}, D \subset \mathbb{R}^n$.**

Необходимое условие минимума(максимума) первого порядка:

Пусть x^* есть точка локального экстремума функции $f(x)$ и $f(x)$ дифференцируема в точке x^*. Тогда градиент функции $f(x)$ в точке x^* равен нулю, т.е. $\nabla f(x^*) = 0$.

Необходимое условие минимума(максимума) второго порядка:

Пусть x^* есть точка локального экстремума функции $f(x)$ и $f(x)$ дважды дифференцируема в этой точке. Тогда матрица Гессе $H(x^*)$ неотрицательно (неположительно) определена, т.е.

$$H(x^*) \geq 0,$$

$$H(x^*) \leq 0.$$

Достаточное условие:

Пусть $f(x)$ в точке x^* дважды дифференцируема, её градиент равен нулю, а матрица Гессе является положительно (отрицательно) определенной, т.е.

$$\nabla f(x^*) = 0 \,, H(x^*) > 0,$$

$$(H(x^*) < 0).$$

тогда точка x^* является точкой локального минимума (максимума) функции $f(x)$.

2. **Как определяется порядок численного метода решения задачи оптимизации? Приведите примеры численных методов нулевого, первого и второго порядка.** Определяется исходя из порядка

производной, применяемой в конкретном случае.

Методы прямого поиска или методы нулевого порядка: основаны на использовании информации только о целевой функции.

Методы первого порядка: используется вычисление целевой функции и ее производных до первого порядка включительно. Это различные градиентные методы.

Методы второго порядка: используется вычисление целевой функции и ее производных до второго порядка включительно. Это метод Ньютона.

3. **Какие методы одномерной минимизации Вы знаете? В каких численных методах первого и второго порядка они используются?**

К основным численным методам одномерной минимизации относят:

- метод равномерного поиска;

- метод деления отрезка пополам;
- метод дихотомии;
- метод золотого сечения;
- метод Фибоначчи;
- метод квадратичной интерполяции и др.

Метод Ньютона 2-го порядка.

4. **Комбинацией каких методов оптимизации является метод Марквардта?**

Является альтернативой методу Ньютона. Может рассматриваться как комбинация последнего с методом градиентного спуска.

5. **Для каких функций эффективно применение рассмотренных методов первого и второго порядка?**

Для непрерывно дифференцируемых функций.

Приложение.

Пункт 1. (язык программирования Python)

```
import numpy as np
from scipy.optimize import minimize_scalar

def f(x1, x2):
    return (x1 + 4 * x2) ** 2 + (x1 - 3) ** 2

def df1(x1, x2):
    return 2 * (x1 + 4 * x2) + 2 * (x1 - 3)

def df2(x1,x2):
    return 8 * (x1 + 4 * x2)

def CGM():
    k = 0
    eps = 0.01
    pr_x1 = 10
    pr_x2 = 10
    M = 1000


    while True:
        # STEP 2
        d1 = -df1(pr_x1, pr_x2)
        d2 = -df2(pr_x1, pr_x2)

        lmd = float(minimize_scalar(lambda l: f(pr_x1 + l * d1, 
pr_x2 + l * d2)).x) ## STEP 2
        #STEP 3
        fut_x1 = pr_x1  + lmd * d1
        fut_x2 = pr_x2 + lmd * d2

        if np.sqrt(d1**2+d2**2) < eps:
            print('%.9f'%fut_x1,'%.9f'%fut_x2,sep=";")
            print("f(x1,x2)", '%.9f'%f(fut_x1, fut_x2), sep=" = 
")
            print("k", k,sep=" = ")
            break
        else: ## STEP 4
            k+=1

            w = (df1(fut_x1, fut_x2)**2 + df2(fut_x1, 
fut_x2)**2)/(df1(pr_x1, pr_x2)**2 + df2(pr_x1, pr_x2)**2)
            pr_x1 = fut_x1
            pr_x2 = fut_x2
            d1 = -df1(pr_x1, pr_x2) + w*d1
            d2 = -df2(pr_x1, pr_x2) + w*d2

            fut_x1 = pr_x1 + lmd * d1
            fut_x2 = pr_x2 + lmd * d2
```

```
            if(np.sqrt(d1**2+d2**2) < eps or k+1 > M):
                print('%.9f'%fut_x1, '%.9f'%fut_x2, sep=";")
                print("f(x1,x2)", '%.9f'%f(fut_x1,fut_x2),sep="
= ")

                print("k", k, sep=" = ")
                break
            else:
                k+=1
                pr_x1 = fut_x1
                pr_x2 = fut_x2
                continue


    return 0

CGM()
```

Пункт 2. (язык программирования C++)

```
#include <iostream>
#include <vector>

using namespace std;

double f(double x1, double x2)
{
    return (x1 + 4 * x2) * (x1 + 4 * x2) + (x1 - 3) * (x1 - 3);
}

double df1(double x1, double x2) {
    return 4 * x1 + 8 * x2 - 6;
}

double df2(double x1, double x2) {
    return 8 * x1 + 32 * x2;
}

int main()
{
    cout.precision(9);
    int k = 0;
    double eps = 0.01;
    double lambda = 100.;

    double pr_x1 = 10.;
    double pr_x2 = 10.;
    double fut_x1 = 0.;
    double fut_x2 = 0.;
    double h[4];
```

```
    double d1, d2;


    while (true) {
        double f1 = df1(pr_x1, pr_x2);
        double f2 = df2(pr_x1, pr_x2);
        if (pow(f1 * f1 + f2 * f2, 0.5) < eps) {
            cout << "(" << pr_x1 << " , " << pr_x2 << "),\tF_min
= " << f(pr_x1, pr_x2) << endl;
            cout << "k = " << k << endl;
            break;
        }
        else
        {
            double det_h = (4. + lambda) * (32. + lambda) -
64.;//lambda * lambda - 36. * lambda + 64.;///
            h[0] = (32. + lambda) / det_h;
            h[1] = h[2] = -8. / det_h;
            h[3] = (4. + lambda) / det_h;

                //{    {(32.-lambda)/det_h,-8./det_h},{-8./det_h,
(4. - lambda) / det_h} };
            d1 = -(h[0] * f1 + h[1] * f2);
            d2 = -(h[2] * f1 + h[3] * f2);

            fut_x1 = pr_x1 + d1;
            fut_x2 = pr_x2 + d2;

            if (f(fut_x1, fut_x2) < f(pr_x1, pr_x2)) {
                lambda /= 2;
                k++;
                pr_x1 = fut_x1;
                pr_x2 = fut_x2;
                continue;
            }
            else
            {
                lambda *= 2;
                continue;
            }
        }
    }

    return 0;
}
```