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 № 2

Тема: «Численные методы поиска условного экстремума. Метод штрафов»

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 условного экстремума действительной функции.

Задачи:

1. Ознакомиться с постановкой задачи, определяемой 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 методом штрафов, с заданной точностью $\varepsilon = 0.01$.
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 внешних штрафов.

1. Задать значения:

   n – размерность вектора x;

   m – число ограничений-равенств;

   l – число всех ограничений;

   ε_1 – точность решения задачи;

   ε_2 – точность решения задачи безусловной минимизации;

   $k = 0$ – номер итерации;

   x^0 – начальное приближение для x^*, задается вне множества допустимых решений;

   $r^0 > 0$ – начальное значение параметра штрафа, обычно выбирают $r^0 = 0.01; 0.1; 1;$

   $C > 0$ – число для увеличения параметра штрафа, обычно выбирают $C \in [4,10]$;

2. Задаем вспомогательную функцию

$$P(x^k, r^k) = f(x^k) + \Phi(r^k, h(h^k), g(x^k))$$

где штрафная функция

$$\Phi\left(r^k, h(h^k), g(x^k)\right) = \frac{r^k}{2}\left\{\sum_{i=1}^{m}\left(h_i(x^k)\right)^2 + \sum_{j=m+1}^{l}\left(g_j^+(x^k)\right)^2\right\}$$

$g_j^+(x)$ – срезка функции

$$g_j^+(x) = \begin{cases} g_j(x), & g_j(x) > 0, \\ 0, & g_j(x) \le 0. \end{cases}$$

3. Найдем значение $\tilde{x}$, доставляющее минимум функции $P(\tilde{x}, r^k)$ по x при фиксированном r^k с помощью одного из методов безусловной минимизации (лабораторные работы 1, 2). В качестве начальной точки используется x^*, в качестве параметра окончания – константа ε_2.
4. Вычислить $\Phi(\tilde{x}, r^k)$. Если $\left|\Phi(\tilde{x}, r^k)\right| \le \varepsilon_1$, то решение задачи найдено $x^* = \tilde{x}$, в противном случае переходим к шагу 5.
5. Изменяем значения $r^{k+1} = Cr^k;\ x^{k+1} = \tilde{x}; k = k + 1$. Возвращаемся к шагу 2.

Индивидуальное задание (2 вариант).

Целевая функция имеет вид: $f(x_1, x_2, x_3, x_4) = x_1^2 + x_2^2 + x_3^2 + x_4^2$.

Область допустимых значений: $x_1 + 2x_2 + 3x_3 + 5x_4 = 10,\ x_1 + 2x_2 + 5x_3 + 6x_4 = 15$

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

Функция Лагранжа.

$$L(x, \lambda) = \lambda_0(x_1^2 + x_2^2 + x_3^2 + x_4^2) + \lambda_1(x_1 + 2x_2 + 3x_3 + 5x_4 - 10) + \lambda_2(\ x_1 + 2x_2 + 5x_3 + 6x_4 - 15)$$

Найдем частные производные $L(x, \lambda)$ и приравниваем к нулю:

$$\frac{\partial L}{\partial x_1} = 2\lambda_0 + \lambda_1 + \lambda_2$$

$$\frac{\partial L}{\partial x_2} = 2\lambda_0 + 2\lambda_1 + 2\lambda_2$$

$$\frac{\partial L}{\partial x_3} = 2\lambda_0 + 3\lambda_1 + 5\lambda_2$$

$$\frac{\partial L}{\partial x_4} = 2\lambda_0 + 5\lambda_1 + 6\lambda_2$$

Система выше дополняется:

$$\frac{\partial L}{\partial \lambda_1} = x_1 + 2x_2 + 3x_3 + 5x_4 - 10 = 0$$

$$\frac{\partial L}{\partial \lambda_2} = \ x_1 + 2x_2 + 5x_3 + 6x_4 - 15 = 0$$

Из системы вытекают три решения:

1) $\lambda_0 = 1, \lambda_1 = 0, \lambda_2 = -\frac{5}{8}, x_1 = (\frac{5}{16}, \frac{5}{8}, \frac{25}{16}, \frac{15}{8})$.
2) $\lambda_0 = 1, \lambda_1 = -\frac{10}{11}, \lambda_2 = 0, x_2 = (\frac{5}{16}, \frac{5}{8}, \frac{25}{16}, \frac{15}{8})$.
3) $\lambda_0 = 1, \lambda_1 = \frac{15}{4}, \lambda_2 = \frac{15}{16}, x_3 = (-\frac{5}{32}, -\frac{5}{16}, \frac{65}{32}, \frac{15}{16})$.

Матрица Гессе

$$H = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$\Delta_1 = 4 > 0, \Delta_2 = 4 > 0, \Delta_3 = 8 > 0, \Delta_4 = 16 > 0 =>$

H определана положительно.

Т.к. матрица Гессе положительно определена, то в любой стационарной точке функцией достигается локальный минимум.

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

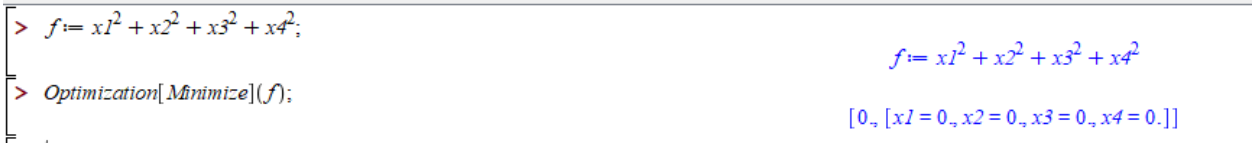

Рисунок 1. Решение задачи в maple.

- Найти приближенное решение задачи согласно варианту методом штрафов, с заданной точностью $\varepsilon = 0.01$.

  Метод внешних штрафов.

Листинг программы содержится в Приложении.

Пример выполнения программы.

```
step = 5        Phi = 0.00516264531
(0.00086617489 , 0.00173234978 , 0.00363856764 , 0.00485089593)
f_min = 4.05216604e-05

D:\GoogleDrive\!Study\!Lab_Work\Методы Оптимизации\МО_лаб2\OptMet
шил работу с кодом 0.
```

Рисунок 2. Пример выполнения программы.

Вывод

Таким образом, в ходе выполнения лабораторной работы приобрели навыки численного решения задач поиска условного экстремума действительной функции, соответствующий метод был программно реализован.

Ответы на контрольные вопросы

1. Задача условной оптимизации. Задача поиска минимума (максимума) функции $f(x^*) = min\, f(x)$ $(f(x^*) = max\, f(x)), x \in X$ когда множество допустимых решений задается равенством и неравенствами:

$$X = \begin{cases} x \in R^n q_j(x) = 0, j = 1, .., m, m < n, \\ g_{j(x)} \leq 0, j = m+1, .., p \end{cases}$$

функции $f(x)$, $g\,j$ $(x), j$ = 1. . p дважды непрерывно дифференцируемые функции.

Пример практического применения: поиск параллелепипеда максимального объема, вписанный в эллипсоид

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1.$$

$$f = V_n = abc \to max$$

Критерий: $\sqrt[3]{abc} \leq \frac{1}{3}(a+b+c)$.

2. Функция

$$L(x, \lambda) = \lambda_0 f(x) + \sum_{j=1}^{p} \lambda_j g_j(x), \lambda = (\lambda_0, .., \lambda_p),$$

называется обобщенной функцией Лагранжа. Числа $\lambda\,0$, . . $\lambda\,p$ называются множителями Лагранжа.

3. Необходимые условия условного минимума (максимума) первого порядка:

Пусть x^* - точка локального минимума (максимума), тогда найдутся такие числа $\lambda_0, .. \lambda_p$, что выполняются следующие условия: условие нетривиальности $\sum \lambda_j^2 > 0$;

стационарности функции Лагранжа по x: $\frac{\partial L(x^*, \lambda_0^*, \lambda^*)}{\partial_{x_j}} = 0, j = 1, .. n$; не отрицательности $\lambda_0^*, \geq 0, \lambda_j^* \geq 0, j = m+1, .. p$; дополняющей не жёсткости $\lambda_j^* g_j(x^*) = 0, j = m+1, .. p$.

Достаточные условия условного минимума (максимума) первого порядка:

Пусть точка (x^*, λ^*) удовлетворяет необходимым условиям первого порядка, $\lambda_0^* \neq 0$, суммарное число активных в точке x^* ограниченийнеравенств и ограничений равенств совпадает с числом переменных n. Тогда если $\lambda_j^* > 0$ $(\lambda_j^* < 0)$ для всех j, то точка x^* является точкой условного локального минимума (максимума).

Необходимые условия условного минимума (максимума) второго порядка:

Пусть x^* - точка локального минимума (максимума), точка (x^*, λ^*) – решение. Тогда второй дифференциал классической функции Лагранжа, вычисленный в точке (x^*, λ^*), неотрицателен (не положителен):

$$d^2L(x^*, \lambda^*) \geq 0 (d^2L(x^*, \lambda^*) \leq 0$$

для всех dx, таких что

$$dg_j(x^*) = 0, \lambda_j^* > 0 (\lambda_j^* < 0)$$

$$dg_j(x^*) \leq 0, \quad \lambda_j^* = 0.$$

Достаточные условия условного минимума (максимума) второго порядка:

Пусть (x^*, λ^*) удовлетворяет необходимым условиям при $\lambda_0^* \neq 0$. Если в этой точке

$$d^2L(x^*, \lambda^*) > 0 (d^2L(x^*, \lambda^*) < 0)$$

для всех ненулевых dx, таких что

$$dg_j(x^*) = 0, \lambda_j^* > 0 (\lambda_j^* < 0)$$

$$dg_j(x^*) \leq 0, \quad \lambda_j^* = 0.$$

то точка x^*является точкой локального минимума (максимума).

4. Численные методы решения условной оптимизации: метод штрафных функций.

5. Численные методы решения безусловной оптимизации: метод наискорейшего спуска, метод Ньютона, метод сопряженных градиентов.

6. Для ограничений-равенств: метод внешних штрафов. Штрафная функция

$$F\left(r^k, h(x^k), g(x^k)\right) = \frac{r^k}{2}\left\{\sum_{i=1}^{m}\left(h_i(x^k)\right)^2 + \sum_{j=m+1}^{t}\left(g_j^+(x^k)\right)^2\right\},$$

Для ограничений-неравенств: метод внутренних штрафов. Штрафная функция

$$F\left(r^k, g(x^k)\right) = r^k \sum_{j=1}^{t} \frac{1}{g_j(x^k)}$$

или

$$F\left(r^k, g(x^k)\right) = r^k \sum_{j=1}^{t} \ln(-g_j(x^k)).$$

7. Начальная точка для штрафных методов решения задачи условной оптимизации задается вне множества допустимых решений D.

Приложение.

```
#include <iostream>
#include <vector>
using namespace std;

const double eps1 = 0.01;
const double eps2 = 0.01;

double f(double x1, double x2, double x3, double x4)
{
    return x1 * x1 + x2 * x2 + x3 * x3 + x4 * x4;
}

double df1(double x1)
{
    return 2 * x1;
}

double df2(double x2)
{
    return 2 * x2;
}

double df3(double x3)
{
    return 2 * x3;
}

double df4(double x4)
{
    return 2 * x4;
}

double h1(double x1, double x2, double x3, double x4)
{
    return x1 + 2 * x2 + 3 * x3 + 5 * x4 - 10;
}

double h2(double x1, double x2, double x3, double x4)
{
    return x1 + 2 * x2 + 5 * x3 + 6 * x4 - 15;
}

double P(double x1, double x2, double x3, double x4, double rk)
{
    double h_12 = h1(x1, x2, x3, x4);
```

```
    double h_22 = h2(x1, x2, x3, x4);
    return f(x1, x2, x3, x4) + rk * (h_12 * h_12 + h_22 * h_22)
/ 2.;
}

double Phi(double x1, double x2, double x3, double x4, double
rk)
{
    double h_12 = h1(x1, x2, x3, x4);
    double h_22 = h2(x1, x2, x3, x4);
    return rk * (h_12 * h_12 + h_22 * h_22) / 2.;
}

double dPdx1(double x1, double x2, double x3, double x4, double
rk) {
    double h_12 = h1(x1, x2, x3, x4);
    double h_22 = h2(x1, x2, x3, x4);
    return 2. * x1 + rk * (h_12 + h_22);
}

double dPdx2(double x1, double x2, double x3, double x4, double
rk) {
    double h_12 = h1(x1, x2, x3, x4);
    double h_22 = h2(x1, x2, x3, x4);
    return 2. * x2 + 2. * rk * (h_12 + h_22);
}

double dPdx3(double x1, double x2, double x3, double x4, double
rk) {
    double h_12 = h1(x1, x2, x3, x4);
    double h_22 = h2(x1, x2, x3, x4);
    return 2. * x3 + rk * (3. * h_12 + 5. * h_22);
}

double dPdx4(double x1, double x2, double x3, double x4, double
rk) {
    double h_12 = h1(x1, x2, x3, x4);
    double h_22 = h2(x1, x2, x3, x4);
    return 2. * x4 + rk * (5. * h_12 + 6. * h_22);
}
#pragma region d^2P/dx_i*x_j
double dPdx1x1(double rk) {
    return 2. + 2. * rk;
}
double dPdx1x2(double rk) {
    return 4. * rk;
```

```
}
double dPdx1x3(double rk) {
    return 8. * rk;
}
double dPdx1x4(double rk) {
    return 11. * rk;
}
double dPdx2x2(double rk) {
    return 2. + 8. * rk;
}
double dPdx2x3(double rk) {
    return 16. * rk;
}
double dPdx2x4(double rk) {
    return 22. * rk;
}
double dPdx3x3(double rk) {
    return 2. + 34. * rk;
}
double dPdx3x4(double rk) {
    return 45. * rk;
}
double dPdx4x4(double rk) {
    return 2. + 61 * rk;
}
#pragma endregion



vector<double> Markvardt(double x1, double x2, double x3, double
x4, double rk, int step)
{
    double h_inv[4][4];
    cout.precision(9);
    int k = 0;

    double l = 10000.;

    #pragma region Start value
        double pr_x1 = x1;
        double pr_x2 = x2;
        double pr_x3 = x3;
        double pr_x4 = x4;
    #pragma endregion
    #pragma region End value
        double fut_x1 = 0.;
```

```
        double fut_x2 = 0.;
        double fut_x3 = 0.;
        double fut_x4 = 0.;
    #pragma endregion

    double d1, d2, d3, d4;
    //
    while (true)
    {
        double P1 = dPdx1(pr_x1, pr_x2, pr_x3, pr_x4, rk);
        double P2 = dPdx2(pr_x1, pr_x2, pr_x3, pr_x4, rk);
        double P3 = dPdx3(pr_x1, pr_x2, pr_x3, pr_x4, rk);
        double P4 = dPdx4(pr_x1, pr_x2, pr_x3, pr_x4, rk);

        if (pow(P1 * P1 + P2 * P2 + P3 * P3 + P4 * P4, 0.5) <
eps2) { //if( ||grad(P)|| < eps2 )

            //cout << "Markvardt :: step = " << step << endl;
            //cout << "(" << pr_x1 << " , " << pr_x2 << " , " <<
pr_x3 << " , " << pr_x4 << "),\tP_min = " << P(pr_x1, pr_x2,
pr_x4, pr_x4, rk) << endl;
            //cout << "k = " << k << endl;
            vector<double> res = { pr_x1, pr_x2, pr_x3, pr_x4 };
            return res;
        }
        else {
            //
            double det = (l * l * l + 105. * l * l * rk + 74. *
l * rk * rk + 6. * l * l + 420. * l * rk + 148. * rk * rk + 12.
* l + 420. * rk + 8.);
            #pragma region inversed Hesse
            h_inv[0][0] = (l * l + 103 * l * rk + 69 * rk * rk +
4 * l + 206 * rk + 4) / det;
            h_inv[0][1] = h_inv[1][0] = -(2 * (2 * l + 5 * rk +
4)) * rk / det;
            h_inv[0][2] = h_inv[2][0] = -(8 * l - 7 * rk + 16) *
rk / det;
            h_inv[0][3] = h_inv[3][0] = -(11 * l + 14 * rk + 22)
* rk / det;
            h_inv[1][1] = (l * l + 97 * l * rk + 54 * rk * rk +
4 * l + 194 * rk + 4) / det;
            h_inv[1][2] = h_inv[2][1] = -(2 * (8 * l - 7 * rk +
16)) * rk / det;
            h_inv[1][3] = h_inv[3][1] = -(2 * (11 * l + 14 * rk
+ 22)) * rk / det;
```

```
            h_inv[2][2] = (l * l + 71 * l * rk + 5 * rk * rk + 4
* l + 142 * rk + 4) / det;
            h_inv[2][3] = h_inv[3][2] = -(5 * (9 * l + 2 * rk +
18)) * rk / det;
            h_inv[3][3] = (l * l + 44 * l * rk + 20 * rk * rk +
4 * l + 88 * rk + 4) / det;
            #pragma endregion
            #pragma region vector d_k
            d1  =  -(h_inv[0][0]  *  P1  +  h_inv[0][1]  *  P2  +
h_inv[0][2] * P3 + h_inv[0][3] * P4);
            d2  =  -(h_inv[1][0]  *  P1  +  h_inv[1][1]  *  P2  +
h_inv[1][2] * P3 + h_inv[1][3] * P4);
            d3  =  -(h_inv[2][0]  *  P1  +  h_inv[2][1]  *  P2  +
h_inv[2][2] * P3 + h_inv[2][3] * P4);
            d4  =  -(h_inv[3][0]  *  P1  +  h_inv[3][1]  *  P2  +
h_inv[3][2] * P3 + h_inv[3][3] * P4);
            #pragma endregion

            fut_x1 = pr_x1 + d1;
            fut_x2 = pr_x2 + d2;
            fut_x3 = pr_x3 + d3;
            fut_x4 = pr_x4 + d4;

            if (P(fut_x1, fut_x2, fut_x3, fut_x4, rk) < P(pr_x1,
pr_x2, pr_x3, pr_x4, rk)) {
                l /= 2;
                k++;
                pr_x1 = fut_x1;
                pr_x2 = fut_x2;
                pr_x3 = fut_x3;
                pr_x4 = fut_x4;
                continue;
            }
            else
            {
                l *= 2;
                continue;
            }

        }
    }
}

void Shtraf() {
    cout.precision(9);
    double x1, x2, x3,x4;
```

```
    double rk = 0.1;
    double C = 5.;
    int k = 0;
#pragma region start value
    x1 = x2 = x3 = x4 = 5.;
#pragma endregion

    vector<double> x;
M:  x = Markvardt(x1, x2, x3, x4, rk, k);
    double phi = Phi(x[0], x[1], x[2], x[3], rk);

    if (abs(phi) <= eps1)
    {
        cout << "step = " << k << "\tPhi = " << phi << endl;
        x1 = x[0];
        x2 = x[1];
        x3 = x[2];
        x4 = x[3];
        cout << "(" << x1 << " , " << x2 << " , " << x3 << " , "
<< x4 << ")\nf_min = " << f(x1, x2, x3, x4) << endl;;
    }
    else
    {
        rk /= C;
        x1 = x[0];
        x2 = x[1];
        x3 = x[2];
        x4 = x[3];
        k++;
        goto M;
    }

}

int main()
{
    Shtraf();
    return 0;
}
```