Многомерная минимизация с ограничениями. Метод наискорейшего спуска, метод внешних штрафов.

А. Климовский

13 Мая 2001

1 Постановка задачи

Рассмотрим следующую задачу на условный минимум (здесь и далее $f(x)$, $h_i(x) : \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \ldots, m$):

$$\min_{\substack{h_i(x) \le 0, \\ i=1,\ldots,m}} f(x) = \, ? \tag{1}$$

Тестовая задача:

$$\min_{\substack{h_i(x) \le 0, \\ i=1,\ldots,3}} x^2 + 5y^2 = \, ?, \tag{2}$$

$$h_1(x) := (x-2)^2 + (y-3)^2 - 16, \tag{3}$$

$$h_2(x) := (x-2)^2 + 2 - y, \tag{4}$$

$$h_3(x) := y - 4. \tag{5}$$

2 Краткое изложение метода и вычислительная схема решения

2.1 Метод внешних штрафов

Введем в рассмотрение функцию $W(t,x) : [0; +\infty) \times \mathbb{R}^n \mapsto [0; +\infty)$ (*внешний штраф*) такую, что (здесь и далее $D := \{x \in \mathbb{R}^n \; |h_i(x) \le 0, \; i = 1, \ldots, m\}$):

1. $W(t,x)$ - выпукла, $W(t,x) \in C^1([0; +\infty) \times \mathbb{R}^n)$;
2. $\forall x \in D, \; \forall t \ge 0 : W(t,x) = 0$;
3. $\forall x \in \mathbb{R}^n \setminus D : \lim_{t \to +\infty} W(t,x) = +\infty$.

Далее, рассмотрим функцию $\tilde{f}(x) := f(x) + W(t,x)$. Тогда, при некоторых условиях на функции $f(x), h_i(x), \; i = 1, \ldots, m$ решение x_t^0 задачи на безусловный минимум

$$\min_{x \in \mathbb{R}^n} \tilde{f}(x) = \, ? \tag{6}$$

при $t \to +\infty$ стремится к решению задачи (1).

2.2 Метод наискорейшего спуска

Рассмотрим метод решения уравнения (6). Для этого выберем начальное приближение x_0. А дальнейшие приближения $\{x_i\}_{i=1}^{n}$ к решению будем получать следующим образом. Положим

$$g_i(\alpha) := f(x_i - \alpha \nabla f(x_0)),\ \alpha \geq 0 \tag{7}$$

Далее, найдем решение одномерной задачи

$$\min_{\alpha \in [0;+\infty)} g_i(\alpha) \tag{8}$$

и обозначим его α_i^0, тогда

$$x_{i+1} := x_i - \alpha_i^0 \nabla f(x_0). \tag{9}$$

В данной работе $W(t,x)$ выбиралось равным $t\sum_{i=1}^{3}(h_+^i)^2(x)$ (здесь $h_+^i(x) := \frac{h_i(x)+|h_i(x)|}{2}$), производные в (7) вычислялись с точностью до $O(\epsilon^2)$, где ϵ — заданная точность, а решение задачи (8) вычислялось методом деления пополам.

3 Результаты счета

Ниже приведен вывод генерируемый программой:

```
5 iterations done...
x: 1.86645, y: 2.01781, f(x, y): 23.8414, penalty(x, y): 0.000642477
gradX: -2.69774 gradY: -58.3814
```

4 Приложение: исходные тексты программы

```
//-------------------------------------------------
// Multidimesional constrained minimization.
//
// Steepest descent method with outer penalties.
//
// Author: Anton Klimovsky, root@ludus.kharkiv.com
//-------------------------------------------------
#include <iostream.h>
#include <math.h>

const double eps = 1e-4;
const double tBoundary = 20;
const double T = 1.0e+6;
const double maxIterations = 1000;

inline double sqr(double x)
{
    return x*x;
}
```

```
double h1(double x, double y)
{
    return sqr(sqr(x-2))+sqr(sqr(y-3))-16;
}

double h2(double x, double y)
{
    return sqr(x-2)+2-y;
}

double h3(double x, double y)
{
    return y-4;
}

double f(double x, double y)
{
    return sqr(x)+5*sqr(y);
}

inline double truncate(double x)
{
    return (x > 0)? sqr(x) : 0;
}

double penalty(double x, double y, double T)
{
    return T*(truncate(h1(x, y))+
              truncate(h2(x, y))+
              truncate(h3(x, y)));
}

double f0(double x, double y)
{
    return f(x, y)+penalty(x, y, T);
}

class G {
    double x;
    double y;
    double dirX;
    double dirY;

public:
    G(double x0, double y0, double dirX0, double dirY0):
        x(x0),
        y(y0),
        dirX(dirX0),
        dirY(dirY0)
    {}
    double operator() (double t)
    {
        return f0(x-dirX*t, y-dirY*t);
    }
```

```
};

template<class F> void dihotomy(
    double a,
    double b,
    F f,
    double eps,
    double* x0
)
{
    double c;

    while ((b-a) > eps) {
        c = (a+b)/2;
        if (f((a+c)/2) < f((c+b)/2))
            b = c;
        else
            a = c;
    }
    *x0 = c;
}

void calc()
{
    double oldX;
    double oldY;
    double newX;
    double newY;
    double t0;
    double gradX;
    double gradY;
    double norm;
    long int i;

    newX = 2;
    newY = 3;
    i = 0;
    do {
        i++;

        oldX = newX;
        oldY = newY;

        gradX = (f0(oldX+eps, oldY)-f0(oldX-eps, oldY))/(2*eps);
        gradY = (f0(oldX, oldY+eps)-f0(oldX, oldY-eps))/(2*eps);

        norm = sqrt(sqr(gradX)+sqr(gradY));

        gradX /= norm;
        gradY /= norm;

        dihotomy(0, tBoundary, G(oldX, oldY, gradX, gradY), eps, &t0);

        newX = oldX-t0*gradX;
```

```
        newY = oldY-t0*gradY;
    }
    while (fabs(newX-oldX)+fabs(newY-oldY) > eps && i < maxIterations);

    cout << i << " iterations done..." << endl;

    cout << "x: " << newX << ", y: " << newY <<
         ", f(x, y): " << f(newX, newY) <<
         ", penalty(x, y): " << penalty(newX, newY, T) << endl;

    gradX = (f0(newX+eps, oldY)-f0(newX-eps, newY))/(2*eps);
    gradY = (f0(newX, newY+eps)-f0(newX, newY-eps))/(2*eps);
    cout << "gradX: " << gradX << " gradY: " << gradY << endl;
}

void main()
{
    calc();
}
```