Содержание

1. Симплекс метод

Описание метода

Пусть дана линейная функция $f(x) = a_1x_1 + a_2x_1 + \cdots + a_nx_n$ с линейными ограничениями вида $g^i = b^i_1x_1 + b^i_2x_2 + \cdots + b^i_nx_n \leqslant 0$, где $x_i > 0$ и $i = 1..k$, требуется минимизировать эту функцию. Для решения этой задачи можно просто перебрать все вершины получающегося n-мерного многогранника, но для больших размеров систем это слишком трудоемкая задача, так как количество всех вершин равно числу сочетаний $C_n + k^k$ [1], но там не все вершины являются допустимыми, т. е. не все удовлетворяют ограничениям g^i. Для того чтобы не перебирать все вершины, двигаются от вершины к вершине в направлении уменьшения функции, на этом основан симплекс метод.

Ограничения вида неравенств $g \leqslant 0$ и $\hat{g} \geqslant 0$ сводятся к ограничениям типа равенств добавлением исскуственых переменных $g + s_i = 0$ и $\hat{g} - s_i = 0$, где $s_i \geqslant 0$. А если нету ограничений на переменные x_i, т.е. $x_i \in (-\infty, \infty)$, то их разбивают на две положительные переменные $x^i = p_i - n_i$, $p_i \geqslant 0$, $n_i \geqslant 0$. Если надо функцию $f(x)$ максимизировать, то можно минимизировать функцию $-f(x)$. Таким образом все задачи линейного программирования сводятся к виду

$$\begin{cases} f(x) \to \min; \\ g^j = 0, \quad j = 1\,..\,k; \\ x_i \geqslant 0, \quad i = 1\,..\,n. \end{cases}$$

Алгоритм симплекс метода имеет наглядную геометрическую интерпритацию. Все переменные делят на два типа: базисные и небазисные. Небазисные переменные приравниваются к нулю. Что бы определить на рисунке вершину в кооторой мы находимся, надо пересеч прямые перпендикулярные небазисным переменным, они являются нулевым уровнем и одновременно гранью многогранника, если это действительно вершина, а не просто пересечение линий, т.е. если решение является допустимым и удовлетворяет ограничениям g_i. Симплексное отношение — это координата которая отсекается вводимой переменой от исключаемой.

Алгоритм симплекс метода при минимизации:

0 Предполагается что мы находимся в вершине, т. е. решение базисное и удовлетворяет ограничениям.

1) Определяется вводимая переменная по наибольшему положительному коэфициенту в целевой функции $f(x)$ в симлекс таблице.

2) Определяется исключаемая переменная по наименьшему симплексному соотношению

3) Производится переход от одной вершины к другой, т. е. в базисных переменных исключаемая заменяется на вводимую.

4) Переход на первый шаг

Алгоритм заканчивается тогда, когда положительных коэфициентов не осталось в целевой функции.

Решение заданой задачи

Требуется определить минимум заданой фукции с ограничениями

$$
\begin{gathered}
f(x) = -3x_1 + x_3 - 2x_4 \to \min,\\
\begin{cases}
15x_1 + 2x_2 - 3x_3 - 7x_4 + x_5 = 4,\\
2x_1 + x_2 + x_3 - 2x_4 = 3,\\
x_3 + 5x_4 + 2x_5 = 7,\\
x_i \geqslant 0, \quad i = 1\,..\,5.
\end{cases}
\end{gathered}
$$

Сначало нужно найти начальную вершину, обычно для этого применяется двухэтапный симлекс метод, но мы поступим иначе. Вначале возмем случайным образом переменные которые мы введем в базис. Таким способом мы можем не получить допустимое решение, т.к. может не выполнятся соотношение $x_i \geqslant 0$. Далее поочереди будем исключать из базиса те переменные которые являются недопустимыми, и если система совместна то мы получим начальное базисное решение.

Перепишем в виде, удобном для составлении симплекс таблицы

$$
\begin{cases}
15x_1 + 2x_2 - 3x_3 - 7x_4 + x_5 = 4,\\
2x_1 + x_2 + x_3 - 2x_4 = 3,\\
x_3 + 5x_4 + 2x_5 = 7,\\
f(x) + 3x_1 - x_3 + 2x_4 = 0.
\end{cases}
$$

Составим таблицу и методом Жордана гаусса сделаем первые три переменные базисными:

$$
\left(\begin{array}{c|cccccc|c}
0 & 15 & 2 & -3 & -7 & 1 & & 4\\
0 & 2 & 1 & 1 & -2 & 0 & & 3\\
0 & 0 & 0 & 1 & 5 & 2 & & 7\\
1 & 3 & 0 & -1 & 2 & 0 & & 0
\end{array}\right)
\begin{array}{l}
{}^{-2}\ {}^{-1}\\
15 \leftarrow +\\
\\
5 \leftarrow +
\end{array}
\sim
$$

$$
\sim
\left(\begin{array}{c|ccccc|c}
0 & 15 & 2 & -3 & -7 & 1 & 4\\
0 & 0 & 11 & 21 & -16 & -2 & 37\\
0 & 0 & 0 & 1 & 5 & 2 & 7\\
5 & 0 & -2 & -2 & 17 & -1 & -4
\end{array}\right)
\begin{array}{l}
\\
\leftarrow + \ \mid\ : 11\\
{}_{1}\\
\\
\end{array}
\sim
$$

$$
\sim \left(\begin{array}{c|ccccc|c}
0 & 15 & 2 & -3 & -7 & 1 & 4 \\
0 & 0 & 1 & 2 & -1 & 0 & 4 \\
0 & 0 & 0 & 1 & 5 & 2 & 7 \\
5 & 0 & -2 & -2 & 17 & -1 & -4
\end{array}\right) \sim
$$

$$
\sim \left(\begin{array}{c|ccccc|c}
0 & 15 & 0 & -7 & -5 & 1 & -4 \\
0 & 0 & 1 & 2 & -1 & 0 & 4 \\
0 & 0 & 0 & 1 & 5 & 2 & 7 \\
5 & 0 & 0 & 2 & 15 & -1 & 4
\end{array}\right) \sim
$$

$$
\sim \left(\begin{array}{c|ccccc|c}
0 & 15 & 0 & 0 & 30 & 15 & 45 \\
0 & 0 & 1 & 0 & -11 & -4 & -10 \\
0 & 0 & 0 & 1 & 5 & 2 & 7 \\
5 & 0 & 0 & 0 & 5 & -5 & -10
\end{array}\right)
\begin{array}{l}
\mid : 15 \\
\\
\\
\mid : 5
\end{array} \sim
$$

$$
\sim \left(\begin{array}{c|ccccc|c}
0 & 1 & 0 & 0 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & -11 & -4 & -10 \\
0 & 0 & 0 & 1 & 5 & 2 & 7 \\
1 & 0 & 0 & 0 & 1 & -1 & -2
\end{array}\right).
$$

Получили базисное решение. Будет ли оно допустимым? Нет, т. к. приравняв к нулю свободные переменные x_4 и x_5, получим решение $x = (3, -10, 7, 0, 0)$, в котором координаты не являются положительными. Исключим отрицательную переменную x_2 из базиса. Вводимой переменной выберем ту, которая имеет наибольший коэфициент при целевой функции, т. е. ведущим элементом будет -11.

$$
\left(\begin{array}{c|ccccc|c}
0 & 1 & 0 & 0 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & -11 & -4 & -10 \\
0 & 0 & 0 & 1 & 5 & 2 & 7 \\
1 & 0 & 0 & 0 & 1 & -1 & -2
\end{array}\right)
\begin{array}{l}
11 \\
\\
11 \\
11
\end{array} \sim
$$

$$
\sim \left(\begin{array}{c|ccccc|c}
0 & 11 & 2 & 0 & 0 & 3 & 11 \\
0 & 0 & -1 & 0 & 11 & 4 & 11 \\
0 & 0 & 5 & 11 & 0 & 2 & 22 \\
11 & 0 & 0 & 0 & 0 & -15 & -33
\end{array}\right)
$$

Так как все коэфициенты, не считая боковые, в последней строке отрицательны, то мы достигли минимума. Приравняв свободные переменные x_2 и x_5, получим решение $x = (1, 0, 2, 1, 0)$.

Проверка:

$$
f = -3 \cdot 1 + 1 \cdot 2 - 2 \cdot 1 = -3,
$$

в тоже время

$$11 \cdot f = 15 \cdot x_5 - 33 \implies f = -3.$$

Сходится, значит матричные вычисления верны.

2. Условная нелинейная минимизация

Требуется минимизировать функцию с ограничениями

$$f(x) = x_1^2 + x_2^2 \to \min,$$
$$\begin{cases} x_1^2 + 2(x_2 - 2) \leqslant 8, \\ x_1^2 + (x_2 - 2)^2 \geqslant 1. \end{cases}$$

Функция f(x) является параболойдом вращения, значит она выпукла. Применяя теорему Куна-Такера, составим функцию лагранжа и условия дополняющей нежесткости.

$$L = \lambda_0(x_1^2 + x_2^2) + \mu_1(x_1^2 + 2(x_2 - 2) - 8) + \mu_2(1 - x_1^2 - (x_2 - 2)^2)$$
$$\mu_1(x_1^2 + 2(x_2 - 2) - 8) = 0$$
$$\mu_2(1 - x_1^2 - (x_2 - 2)^2) = 0$$

Продифференцировав L, получим систему уравнений

$$\frac{\partial f}{\partial x_1} = 2\lambda_0 x_1 + 2\mu_1 x_1 - 2\mu_2 x_1 = 0 \tag{1a}$$
$$\frac{\partial f}{\partial x_2} = 2\lambda_0 x_2 + 2\mu_1 x_2 - 2\mu_2 x_2 = 0 \tag{1b}$$
$$\mu_1(x_1^2 + 2(x_2 - 2) - 8) = 0 \tag{1c}$$
$$\mu_2(1 - x_1^2 - (x_2 - 2)^2) = 0 \tag{1d}$$

Рассмотрим случай когда x^* не находится на границе, т.е. $x_1^2 + 2(x_2 - 2) - 8 \neq 0$ и $1 - x_1^2 - (x_2 - 2)^2 \neq 0$, тогда из уравнений (1c) и (1d) получим, что $\mu_1 = 0$ и $\mu_2 = 0$. В силу не тривиальности функции Лагранжа $\lambda_0 \neq 0$, тогда из (1a) и (1b) следует, что $x_1 = 0$ и $x_2 = 0$ являетсся локальным минимумом. В силу выпуклости целевой функции он будет глобальным.

3. Метод Давидона-Флетчера-Пауэлла

3.1. О методе

Многие численные методы многомерной минимизации, использующие градиент, последовательно приближаются к минимуму по такой схеме

$$x^{(k+1)} = x^{(k)} - \gamma_k M(f, x_k) f'(x^{(k)}),$$

где $f(x)$ — целевая функция, минимум котрой мы ищем, а $f'(x)$ — ее градиент. К примеру в методе наискорейшего спуска матрицу M заменяют на еденичную матрицу E, а в методе Ньютона M равна обратной матрице Гессе H^{-1}.

Матрицу M в методе Давидона-Флетчера-Пауэлла

$$D_{k+1} = D_k + A_k + B_k,$$

которые определяються из следующих соотношений

$$A_k = \frac{u_k\, u_k^{\mathrm{T}}}{u_k^{\mathrm{T}}\, v_k},$$
$$B_k = -\frac{D_k\, v_k\, v_k^{\mathrm{T}}\, D_k}{u_k^{\mathrm{T}}\, v_k},$$

где

$$u_k = x_{k+1} - x_k,$$
$$v_k = f'(x_{k+1}) - f'(x_k).$$

Причем $D_0 = E$, а предел последовательности матриц

$$\lim_{k\to\infty} D_k = H^{-1}$$

сходится к матрице Гессе, т.е. в начале алгоритм работает как метод наискорейшего спуска, а в при большом k как метод Ньютона.

3.2. Метод золотого сечения

Одним из самых эффективных методов одномерной минизации на унимодальных функциях является метод золотого сечения. Для того чтобы найти следующий подотрезок в котором находится минимум, вычисляется значение функции всего в одной точке. Опишем подробнее процесс нахождения минимума.

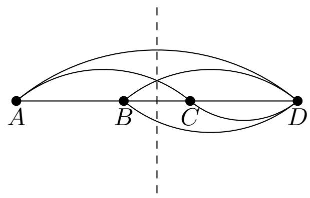

Рис. 1. Разбиение на отрезки методом золотого сечения

Пусть фукция $f(x)$ унимодальна на отрезке $[A, D]$. Вычислим значения функции в точках B и C. В силу унимодальности, если $f(B) < f(C)$, то минимум находится в отрезке $[A, C]$, если $f(B) > f(C)$, то в $[B, D]$. Из исходного отрезка мы получили подотрезок меньшей длинны, повторяя эту процедуру для полученного подотрезка рекурсивно, можно найти минимум с заданой точностью.

Точки B и C выбераются симметрично относительно центра так, чтобы выполнялись соотношения

$$\frac{AD}{AC} = \varphi, \qquad \frac{AD}{BD} = \varphi,$$

где φ золотое сечение

$$\varphi = \frac{1+\sqrt{5}}{2} = \frac{1}{-1 + \frac{1}{-1+\frac{1}{-1+\cdots}}}. \tag{*}$$

Покажем, что значение ункции на каждом шаге кроме первого надо вычислять всего лишь одной точке. Для этого достаточно доказать, что BD и CD в золотой пропорции.

$$\frac{BD}{CD} = \frac{l}{\varphi} : \left(l - \frac{l}{\varphi}\right) = \frac{1}{\varphi - 1}$$

Но из цепной дроби (*) видно, что $\dfrac{1}{\varphi-1} = \varphi$.

3.3. Работа программы на тестовых функциях

1) Функция Розенброка, $f(x_1, x_2) = (x_1 - x_2^2)^2 + (x_1 - 1)^2$ имеет минимум в $x^* = (1, 1)$, начальная точка $x^{(0)} = (-1.2, 1)$

2) Функция Розенброка, $x^{(0)} = (-2, 10)$

3) Функция Химмельблау, $f(x_1, x_2) = (x_1^2 + x_2 - 11) + (x_1 + x_2^2 - 7)$ имеет 4 глобальных минимума, где $f(x^*) = 0$:
   к $x^*_{(1)} = (\ \ 3.000000, \ \ 2.000000)$ сходится из $x^{(0)}_{(1)} = (-0.4, 1.0)$,
   к $x^*_{(2)} = (-2.805118, \ \ 3.131312)$ сходится из $x^{(0)}_{(2)} = (-0.2, 1.0)$,
   к $x^*_{(3)} = (-3.779310, -3.283186)$ сходится из $x^{(0)}_{(3)} = (-0.2, 0.8)$,
   к $x^*_{(4)} = (\ \ 3.584428, -1.848126)$ сходится из $x^{(0)}_{(4)} = (-0.4, 0.8)$.
   И есть один локальный максимум в $\hat{x} = (-0.270844, -0.923038)$, $f(\hat{x}) = 181.616$

4) Тест на одномерную минимизацию, функция имеющая большой овраг в форме $x_2 = 2 * x_1^3 - 2 * x_1$ и точку минимума $x^* = (0, 0)$, $x^{(0)} = (-1.2, 1)$

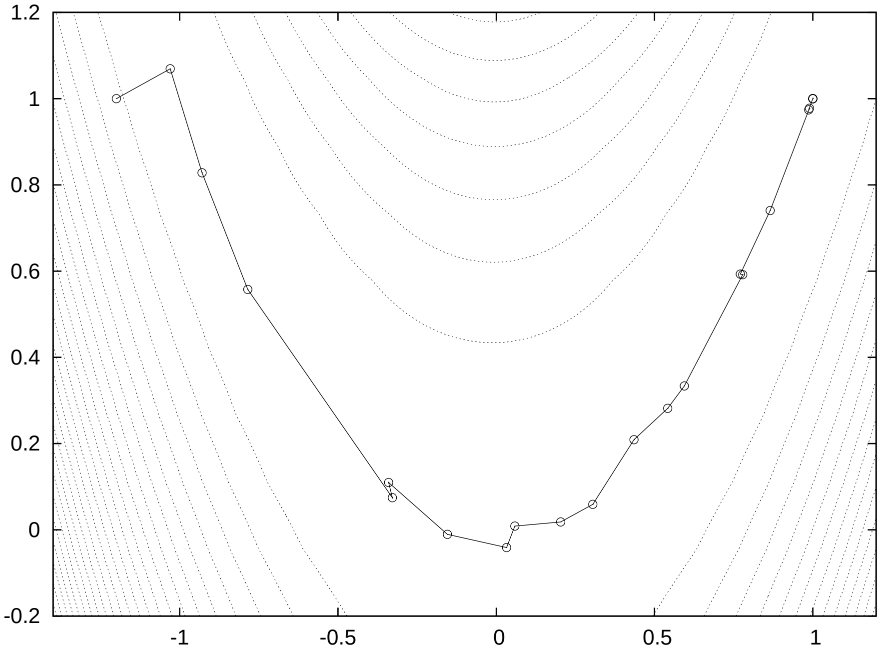

Рис. 2. Функция Розенброка

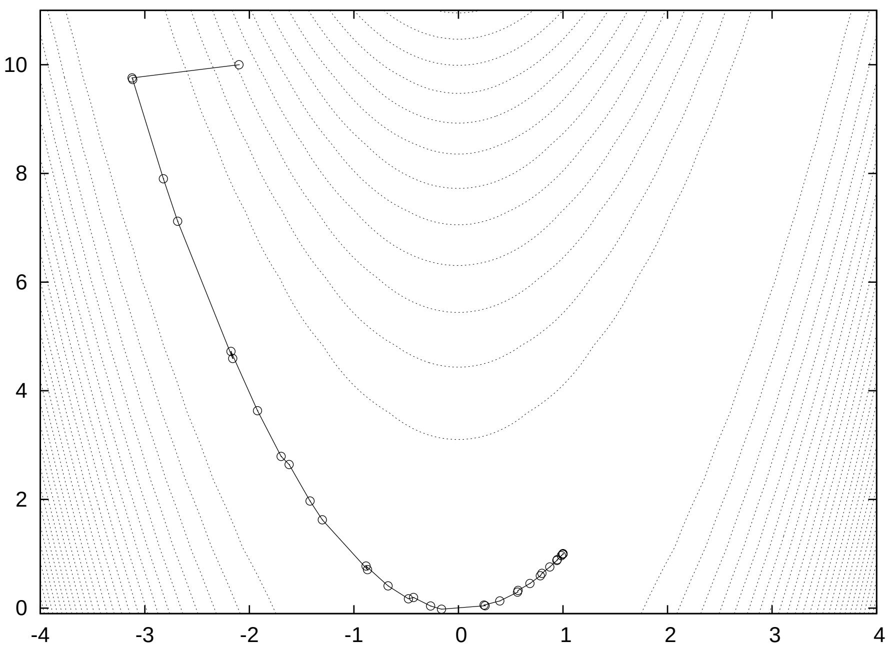

Рис. 3. Функция Розенброка

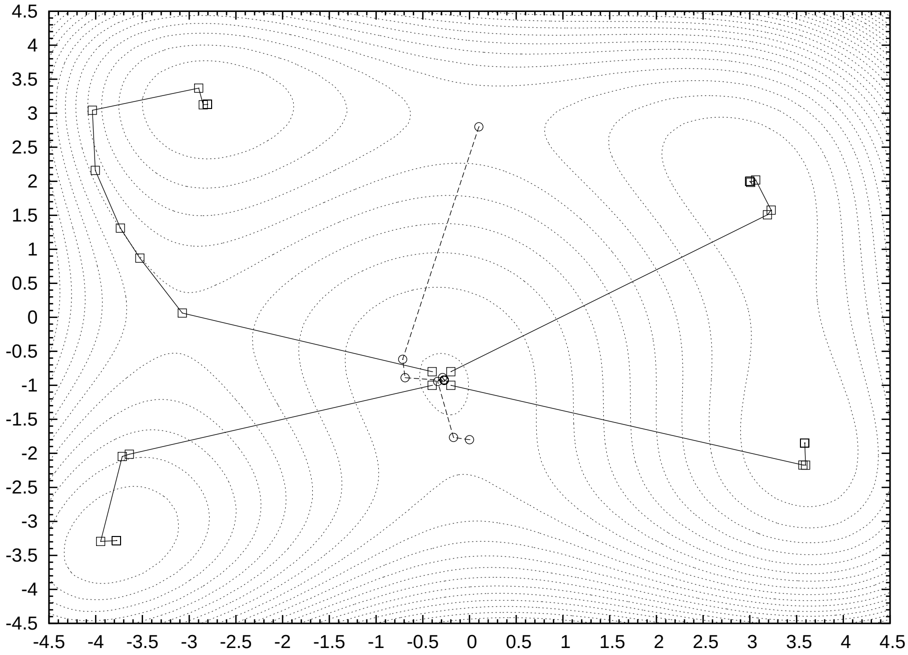

Рис. 4. Функция Химмельблау

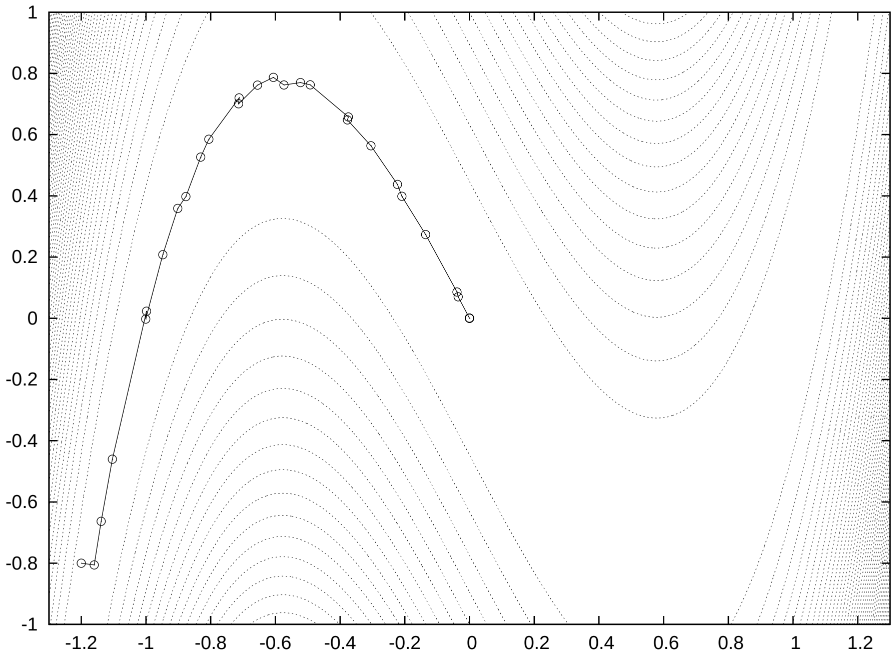

Рис. 5. Тест на одномерную минимизацию

3.4. Исследование заданной функции

Требуется исследовать заданую функцию на минимум

$$f(x_1, x_2) = \sqrt{1 + 2x_1 + x_2^2} + e^{x_1^2 + 2x_2^2} - x_1 - x_2 \tag{2}$$

Так как корня из отрицательного числа не существует, то область определения функции определяется из неравенства $1 + 2x_1 + x_2^2 \geqslant 0$, а границей области определения является парабола $x_1 = -\frac{1}{2} - \frac{x_2^2}{2}$. Исследуем функцию на выпуклость из условия положительной определенности матрицы Гессе.

$$H = \begin{pmatrix} f''_{x_1x_1} & f''_{x_1x_2} \\ f''_{x_2x_1} & f''_{x_2x_2} \end{pmatrix} \succ 0$$

Для этого все угловые миноры должны быть положительны, т.е. $h_{11} > 0$ и $h_{11}h_{22} - h_{12}h_{21} > 0$. Найдем частные производные

$$\frac{\partial f}{\partial x_1} = x_1(1 + 2x_1 + x_2^2)^{-\frac{1}{2}} + e^{x_1^2 + 2x_2^2} - 1,$$
$$\frac{\partial f}{\partial x_2} = x_2(1 + 2x_1 + x_2^2)^{-\frac{1}{2}} + 4x_2 e^{x_1^2 + 2x_2^2} - 1,$$
$$\frac{\partial^2 f}{\partial x_1^2} = -(1 + 2x_1 + x_2^2)^{-\frac{3}{2}} + (4x_1^2 + 2)e^{x_1^2 + 2x_2^2},$$
$$\frac{\partial^2 f}{\partial x_2^2} = (1 + 2x_1 + x_2^2)^{-\frac{1}{2}} - x_2^2(1 + 2x_1 + x_2^2)^{-\frac{3}{2}} + (16x_2^2 + 4)e^{x_1^2 + 2x_2^2},$$
$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_1} = -x_2(1 + 2x_1 + x_2^2)^{-\frac{3}{2}} + 8x_1x_2e^{x_1^2 + 2x_2^2}.$$

Кривые $h_{11} = 0$ и $h_{11}h_{22} - h_{12}h_{21} = 0$, определяющие область выпуклости, являются трансцендетными, их графики приводятся на рис. 3.4 , а на рис. 3.4 изображена заданая функция. Из рисунка видно, что глобальный минимум находится на границе области, там где функция не выпукла. Чтобы алгоритм сходился к этой точке, в функции (2) отбросим корень и с помощью штрафных функций сделаем условную минимизацию с неравенством $1 + 2x_1 + x_2^2 \leqslant 0$, т.е. будем минимизировать функцию

$$\hat{f}(x_1, x_2) = e^{x_1^2 + 2x_2^2} - x_1 - x_2 + 10000(max(0, 1 + 2x_1 + x_2^2))^2$$

ее график приведен на рис. 3.4.

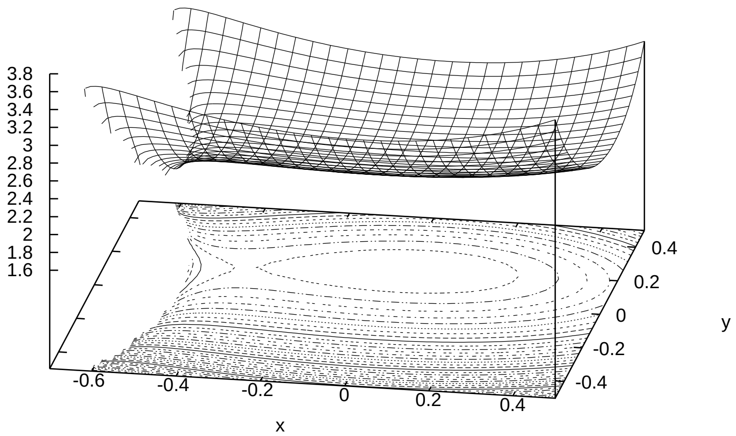

Рис. 6. Функция $f(x_1, x_2) = \sqrt{1 + 2x_1 + x_2^2} + e^{x_1^2 + 2x_2^2} - x_1 - x_2$

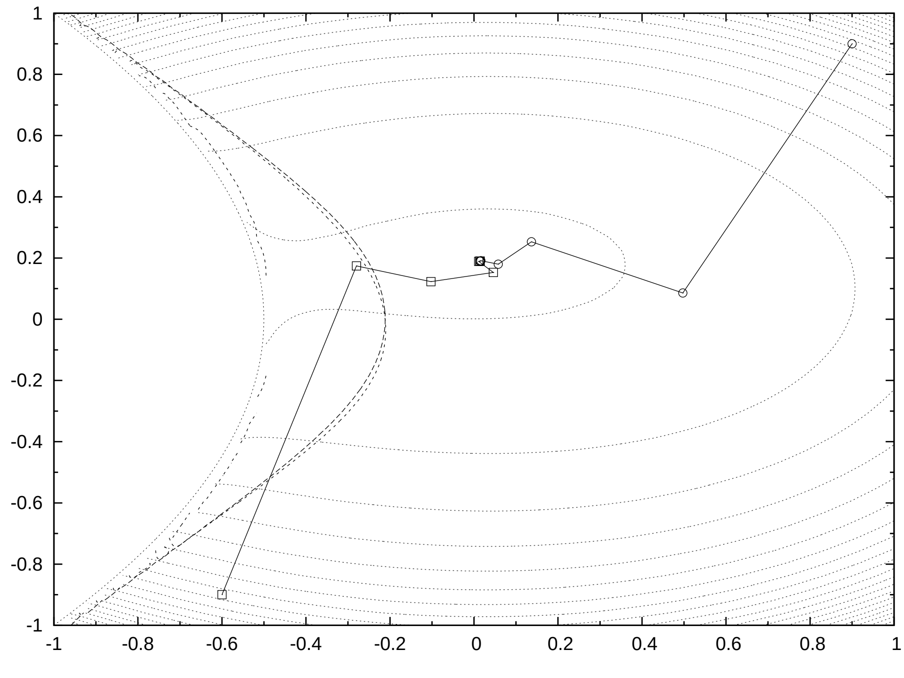

Рис. 7. Нахождение локального минимума в области выпуклости

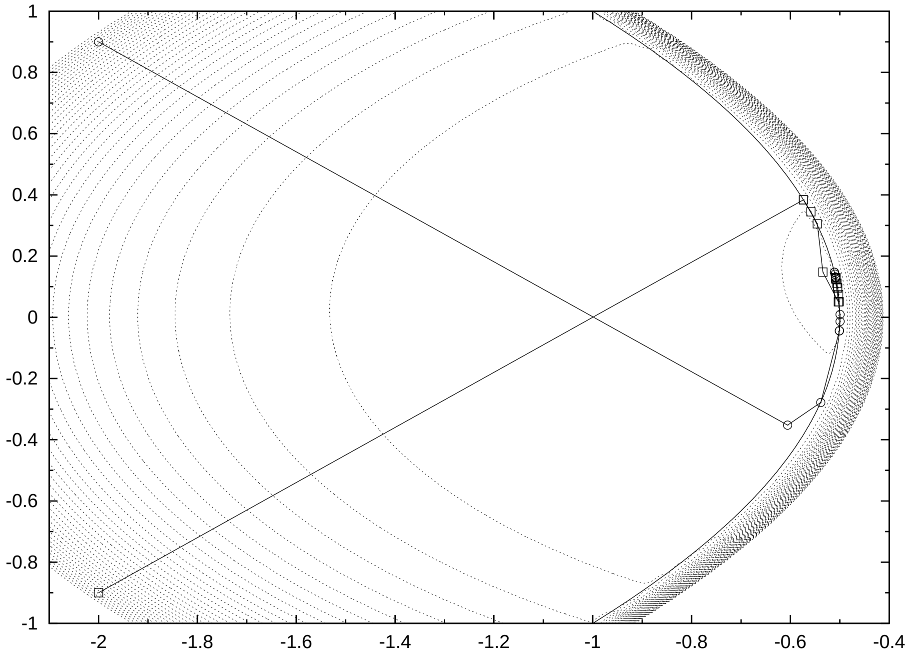

Рис. 8. Нахождение глобального минимума

4. Штрафные функции

Для численного решения нелинейной оптимизации применяют метод штрафных функций. Суть метода заключается в том, что функцию в области не удовлетворяющей условиям ее делуют сингулярной. К примеру есть функция с ограничениями

$$f(x) \to \min,$$
$$g_i(x) \leqslant 0, i = 1 \,..\, n$$

Для нее можно построить штрафную функцию

$$p(x) = h \sum_{i=1}^{n} (max(0, g_i(x)))$$

которая будет сохранять непрерывную производную и будет большой, при большом параметре штрафа h.

Приложение A

Исходный текст программы главного фаила main.cpp

```
#include <iostream>
#include <cmath>
#include "matrix.h"
#include "functionals.h"

/// оптимальные значения констант:
/// "const" >= 1e-9, иначе 1 + const = 1
/// EPSILON >= 2e-9
/// DIFF_STEP >= 1e-6, иначе погрешность диффер-ия сильно возрастает
/// MATRIX_EPS ~~ 1e-1, иначе интервал унимодальности находиться не верно,
///     min_s -> infiniry и программа может думать очень долго
///
/// оптимальные константы, если включены штрафы:
/// 1/TAX ~~ EPSILON, так как 1/TAX ~~ точность решения, если 1/TAX > EPSILON;
///     однако когда алгоритму надо идти вдоль границы области 1/TAX >> EPSILON
/// если 1/TAX > 2*DIFF_STEP, то будут ошибки определения производной на границе
#define GOLD_EPS 1e-9          ///точность одномерной минимизации
#define DIFF_STEP 1e-5         ///шаг дифференцирования
#define EPSILON 5e-9           ///правило останова
#define NORM_EPS 1e-4          ///для определения нулевой нормы[в s_unimodal]
#define UNIM_STEP 0.1          ///шаг для определения интервала унимодальности
#define TAX 1e4                ///параметр штрафа

#define _X_0_ 0.1, 2.8        ///начальная точна[обязательно double, double]
#define DEBUG (0)              ///отладочный\итоговый вывод программы

/// глобальные переменные
vector x_k(2);
```

```
matrix D(2, 2);                 ///приближенная матрица Гессе

/// штрафная функция (g[i] <= 0)
double penalty(vector x)
{
    double pnt = 0;
    //pnt += pow(max(0, x[0]*x[0] + 2*(x[1]-2) - 8), 2);        //УНЛО
    //pnt += pow(max(0, 1 - x[0]*x[0] - (x[1]-2)*(x[1]-2)), 2);

    //pnt += pow(max(0, -x[0]), 2);                              //проверка штрафов
    //pnt += pow(max(0, -x[1]), 2);
    //pnt += pow(max(0, x[0]*x[0] + x[1]*x[1] - 1), 2);

    pnt += pow(max(0, 1 + 2*x[0] + x[1]*x[1]), 2);        //исправленое задание
    return TAX * pnt;
}

/// целевая функция
double f(vector x)
{
    //return x[0]*x[0] + x[1]*x[1] + penalty(x);                //УНЛО
    //return pow(x[0] + 1, 2) + pow(x[1] - 1, 2) + penalty(x);  //проверка штрафов
    //return 100 * pow(x[1] - x[0]*x[0], 2) + pow(1 - x[0], 2); //тест Розенброка
    //return 100 * pow(x[1] - 2*pow(x[0], 3) + 2*x[0], 2) +  x[0]*x[0]; //мой тест
    return - (pow(x[0]*x[0]+x[1]-11, 2) + pow(x[0]+x[1]*x[1]-7, 2));      //Химмельблау
    //return exp(x[0]*x[0] +  2*x[1]*x[1]) - x[0] - x[1] + penalty(x);  //исп задание
    //return sqrt((1 + 2*x[0] + x[1]*x[1])) + exp(x[0]*x[0] + \
    //                           2*x[1]*x[1]) - x[0] - x[1];           //задание
}

/// градиент целевой функции
/// дифференцирование проиводиться с точностью до 4-го порядка малости от h
vector grad_f(vector x)
{
    double h = DIFF_STEP;
    vector g(2);
    vector dx0(2);
    vector dx1(2);
    init(dx0, h, 0.0);
    init(dx1, 0.0, h);
    g[0] = (f(x - 2*dx0) - 8*f(x - dx0) + 8*f(x + dx0) - f(x + 2*dx0)) / 12 / h;
    g[1] = (f(x - 2*dx1) - 8*f(x - dx1) + 8*f(x + dx1) - f(x + 2*dx1)) / 12 / h;
    return g;
}

/// функция возвращает f(x<k+1>)[нужна для золотого сечения]
inline double f_x_kk(double s)
{
    extern vector x_k;
    extern matrix D;
    return f(x_k - s*D*grad_f(x_k));
}

/// опр-ет интервал унимодальности, для применения в нем метода золотого сечения
```

```
/// для этого увеличивает s до тех пор, пока f(x) не начнет возрастать
void s_unimodal(double& s_min, double& s_max, double unim_step)
{
    extern vector x_k;
    extern matrix D;
    double s0 = 0;
    double s1 = unim_step;
    double s2 = 2*unim_step;
    int n = 0;
    while ((f_x_kk(s1) > f_x_kk(s2)) && norm(D*grad_f(x_k)) > NORM_EPS && (++n < 1e4))
    {
        s0 = s1;
        s1 = s2;
        s2 += unim_step;
    }
    s_min = s0;
    s_max = s2;
}

int main()
{
/// 0-й шаг
    extern vector x_k;                  ///x<k>
    extern matrix D;                    ///приближенная матрица Гессе (D<o> = E)
    int k;
    matrix A(2, 2);                     ///A = u * (u)^t / ((u)^t * v)
    matrix B(2, 2);                     ///B = -(D * v * (v)^t * D) / ((v)^t*D*v)
    vector x_kk(2);                     ///x<k+1>
    vector u(2);                        ///u<k> = x<k+1> - x<k>
    vector v(2);                        ///v<k> = grad_f(x<k+1>) - grad_f(x<k>)
    double s;                           ///параметр одномерной минимизации
    double s_min, s_max;                ///границы унимодальности

    k = 1;
    x_k = vector(2);
    init(x_k, _X_0_);
    D = identity(2);

/// 1 шаг
    s_unimodal(s_min, s_max, UNIM_STEP / norm(grad_f(x_k)));
    s = golden_section(f_x_kk, s_min, s_max, GOLD_EPS); ///метод золотого сечения
x_kk = x_k - s*D*grad_f(x_k);
printf("#       X               Y               Z        n\n");
    printf("%15.12lf %15.12lf %15.12lf  %d\n", x_k[0], x_k[1], f(x_k), k-1);
    printf("%15.12lf %15.12lf %15.12lf  %d\n", x_kk[0], x_kk[1], f(x_kk), k);

/// 2..n шаг
    do
    {
        u = x_kk - x_k;                                                 ///u<k>
        v = grad_f(x_kk) - grad_f(x_k);                                 ///v<k>
        A = u * transpose(u) / (transpose(u) * v);                      ///A<k>
        B = (D * v * transpose(v) * D) / (- (transpose(v) * D * v));    ///B<k>
        D += A + B;                                                     ///D<k+1>
```

```
        k++;
        x_k = x_kk;                                                    ///x<k+1>
        s_unimodal(s_min, s_max, UNIM_STEP);
        s = golden_section(f_x_kk, s_min, s_max, GOLD_EPS);
        x_kk = x_k - s*D*grad_f(x_k);
        printf("%15.12lf %15.12lf %15.12lf  %d\n", x_kk[0], x_kk[1], f(x_kk), k);
    } while (norm(x_kk - x_k) > EPSILON);

    putchar('\n');
    return 0;
}
```

Список литературы

[1] *Х. А. Таха* Введение в исследование операций, Пер. с англ., — М.:Вильямс, 2005. — 912 с.

[2] *А.В. Аттетков, С.В. Галкин* Методы опримизации. —М.:МГТУ, 2001. —440 с.

[3] *Д.М. Химмельблау* Методы опримизации. —М.:Мир, 1975. —531 с.