Министерство образования и науки РФ

Новосибирский государственный технический университет

Кафедра прикладной математики

Лабораторная работа №3

по методам оптимизации

Факультет: ПМИ

Группа: ПМ-01

Студент: Конев А.М., Ряховский М.И.

Вариант: 1

Преподаватель: Черникова О.С., Чимитова Е.В.

Новосибирск

2013

# Цель работы

Ознакомится с методами штрафных функций при решении задач нелинейного программирования. Изучить типы штрафных и барьерных функций, их особенности, способы и области применения, влияние штрафных функций на сходимость алгоритмов, зависимость точности решения задачи нелинейного программирования от величины коэффициента штрафа.

# Задания варианта

Решить задачу нелинейного программирования при ограничениях: .

Переформулируем в привычных обозначениях:

Тогда преобразованная функция будет иметь вид:



# Метод штрафных функций

## Простейший тест, для проверки работы программы

В данном тесте будут использоваться другие ограничения, он нужен для проверки работы программы.

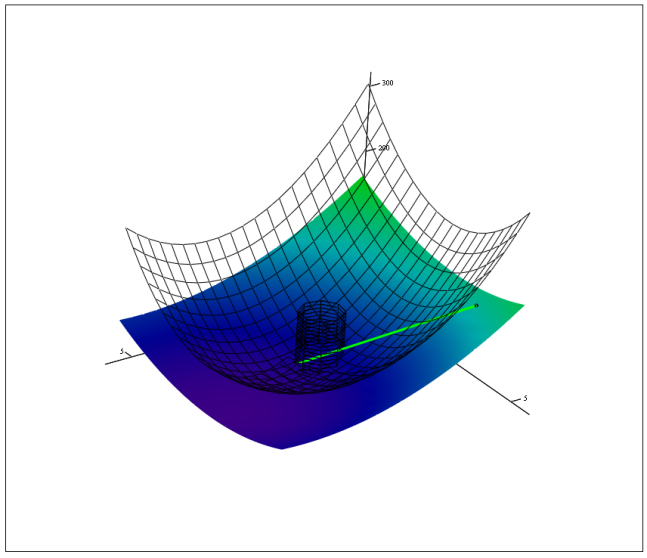


, , 



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  |
| -1 | -5 | 4 | 125 | 0.00E+00 |
| 0 | 0.8333333819 | -0.5527707265 | 17.5611247299 | 3.07E+02 |
| 1 | 0.8333333752 | -0.5527707331 | 17.5611247793 | 4.26E-08 |

Цветная поверхность – исходная функция, сетка – преобразованная и ограничение, линия - траектория.



## Все функции штрафа линейно зависят от ограничения







**Тест 1**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 16 | 17 | 765 | 0.00E+00 | 0 |
| 0 | -4.499999805 | -0.999999977 | 0.250000195 | 7.72E+02 | 138 |
| 1 | -3.99999998 | -0.999999984 | 1.000000041 | 4.25E+00 | 214 |
| 2 | -2.999999958 | -0.999999991 | 4.000000169 | 7.00E+00 | 293 |
| 3 | -0.999999977 | -0.999999997 | 16.00000019 | 8.00E+00 | 374 |
| 4 | -1.46727E-09 | -1.000000004 | 24.99999999 | 1.00E+00 | 450 |
| 5 | -8.11916E-09 | -1.000000011 | 24.99999992 | 2.09E-07 | 490 |

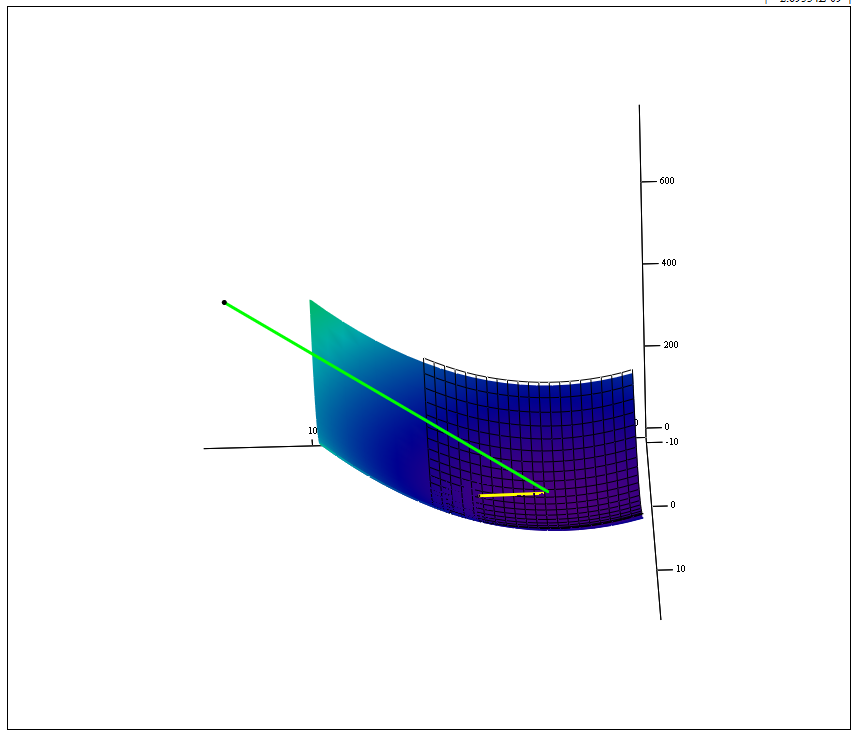
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 0 | 1 | 29 | 0.00E+00 | 0 |
| 0 | -4.499999961 | -0.999999981 | 0.250000039 | 2.43E+01 | 125 |
| 1 | -3.999999981 | -0.999999988 | 1.000000038 | 4.25E+00 | 201 |
| 2 | -2.999999959 | -0.999999994 | 4.000000164 | 7.00E+00 | 280 |
| 3 | -0.999999978 | -1.000000001 | 16.00000018 | 8.00E+00 | 361 |
| 4 | -2.69534E-09 | -1.000000008 | 24.99999997 | 1.00E+00 | 437 |
| 5 | -9.34722E-09 | -1.000000014 | 24.99999991 | 2.68E-07 | 477 |

**Тест 2**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 16 | 17 | 765 | 0.00E+00 | 0 |
| 0 | -2.999999819 | -0.999999929 | 4.000000725 | 7.97E+02 | 138 |
| 1 | -0.999999984 | -0.999999958 | 16.00000013 | 8.00E+00 | 219 |
| 2 | -2.32798E-09 | -0.999999964 | 24.99999998 | 1.00E+00 | 295 |
| 3 | -8.97986E-09 | -0.999999971 | 24.99999991 | 1.84E-07 | 335 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 0 | 1 | 29 | 0.00E+00 | 0 |
| 0 | -2.999999958 | -0.999999987 | 4.000000167 | 1.30E+01 | 122 |
| 1 | -0.999999977 | -0.999999993 | 16.00000018 | 8.00E+00 | 203 |
| 2 | -1.98432E-09 | -1 | 24.99999998 | 1.00E+00 | 279 |
| 3 | -8.63621E-09 | -1.000000007 | 24.99999991 | 1.78E-07 | 319 |

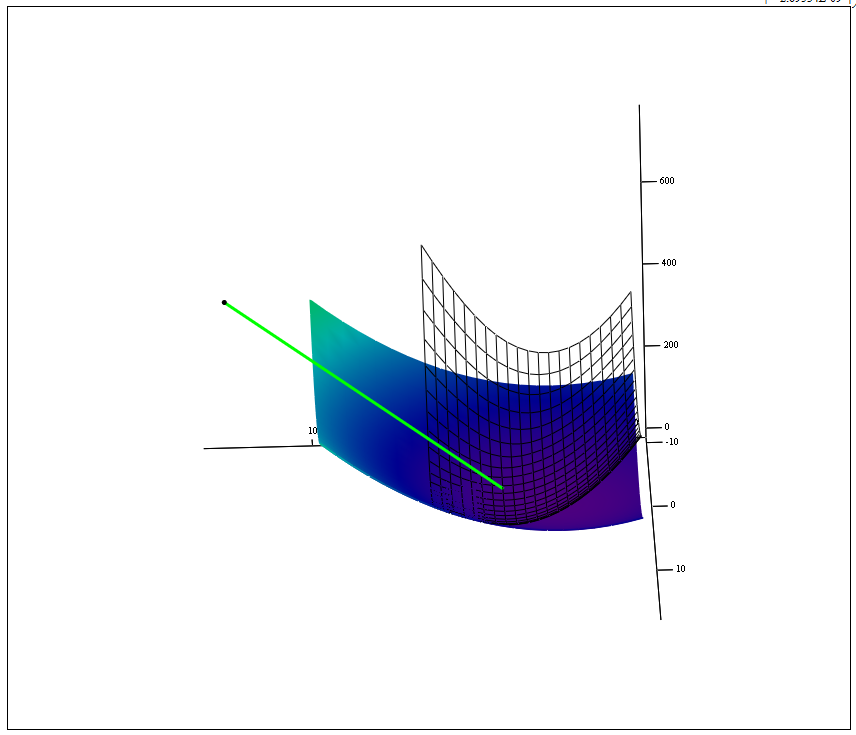


## Все функции штрафа квадратично зависят от ограничений



При всех начальных приближениях получили, что итераций очень много, но последние итерации почти «стоят на месте», поэтому только первые из них:

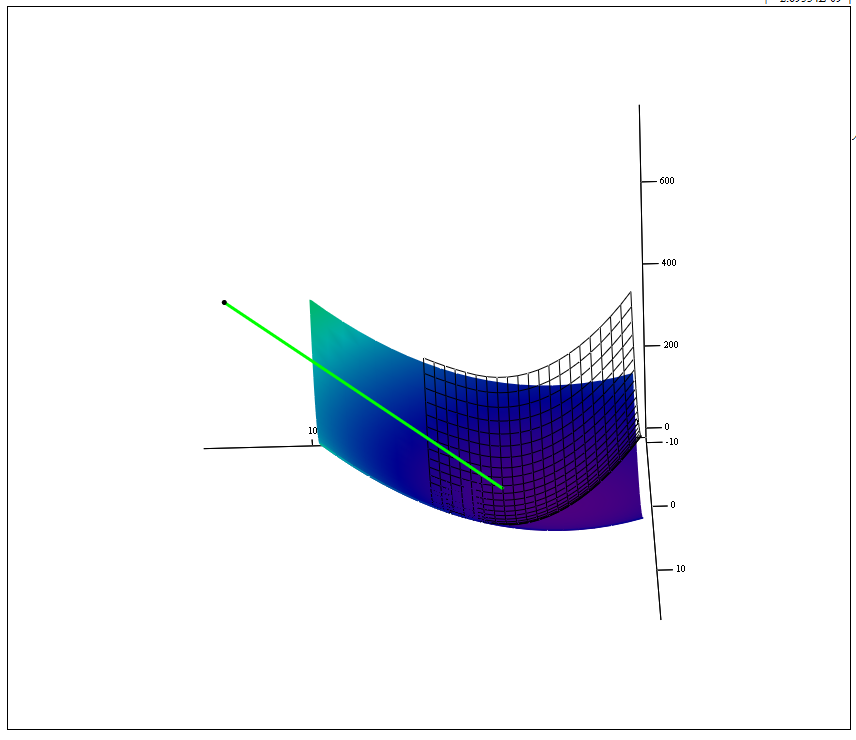
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 16 | 17 | 765 | 0.00E+00 | 0 |
| 0 | -1.666666542 | -0.999999949 | 11.11111194 | 1.04E+03 | 138 |
| 1 | -0.999999984 | -0.999999956 | 16.00000013 | 3.33E+00 | 214 |
| 2 | -0.555555536 | -0.999999962 | 19.75308659 | 2.22E+00 | 288 |
| 3 | -0.294117651 | -0.999999969 | 22.14532869 | 1.31E+00 | 362 |
| 4 | -0.151515139 | -0.999999976 | 23.50780545 | 7.13E-01 | 434 |
| 5 | -0.076923081 | -0.999999982 | 24.23668635 | 3.73E-01 | 503 |
| 6 | -0.038759694 | -0.999999989 | 24.61390538 | 1.91E-01 | 570 |
| 7 | -0.019455255 | -0.999999996 | 24.80582596 | 9.65E-02 | 634 |
| 8 | -0.00974659 | -1.000000002 | 24.9026291 | 4.85E-02 | 696 |
| 9 | -0.004878049 | -1.000000009 | 24.9512433 | 2.43E-02 | 755 |
| 10 | -0.002440217 | -1.000000016 | 24.97560378 | 1.22E-02 | 812 |
| 11 | -0.001220399 | -1.000000022 | 24.98779749 | 6.10E-03 | 866 |
| 12 | -0.000610273 | -1.000000029 | 24.99389764 | 3.05E-03 | 918 |
| 13 | -0.000305156 | -1.000000035 | 24.99694854 | 1.53E-03 | 968 |
| 14 | -0.000152586 | -1.000000042 | 24.99847417 | 7.63E-04 | 1015 |
| 15 | -7.62955E-05 | -1.000000049 | 24.99923705 | 3.82E-04 | 1060 |
| 16 | -3.81455E-05 | -1.000000055 | 24.99961855 | 1.91E-04 | 1102 |
| 17 | -1.90746E-05 | -1.000000062 | 24.99980925 | 9.54E-05 | 1142 |
| 18 | -9.53797E-06 | -1.000000069 | 24.99990462 | 4.77E-05 | 1182 |
| 19 | -9.54462E-06 | -1.000000075 | 24.99990455 | 4.78E-05 | 1222 |
| 20 | -2.38497E-06 | -1.000000082 | 24.99997615 | 1.20E-05 | 1262 |



## Различная зависимость функций штрафа от ограничений

, 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 16 | 17 | 765 | 0.00E+00 | 0 |
| 0 | -1.666666542 | -0.999999949 | 11.11111194 | 7.60E+02 | 138 |
| 1 | -0.999999984 | -0.999999956 | 16.00000013 | 3.33E+00 | 214 |
| 2 | -0.555555536 | -0.999999962 | 19.75308659 | 2.22E+00 | 288 |
| 3 | -0.294117651 | -0.999999969 | 22.14532869 | 1.31E+00 | 362 |
| 4 | -0.151515139 | -0.999999976 | 23.50780545 | 7.13E-01 | 434 |
| 5 | -0.076923081 | -0.999999982 | 24.23668635 | 3.73E-01 | 503 |
| 6 | -0.038759694 | -0.999999989 | 24.61390538 | 1.91E-01 | 570 |
| 7 | -0.019455255 | -0.999999996 | 24.80582596 | 9.65E-02 | 634 |
| 8 | -0.00974659 | -1.000000002 | 24.9026291 | 4.85E-02 | 696 |
| 9 | -0.004878049 | -1.000000009 | 24.9512433 | 2.43E-02 | 755 |
| 10 | -0.002440217 | -1.000000016 | 24.97560378 | 1.22E-02 | 812 |
| 11 | -0.001220399 | -1.000000022 | 24.98779749 | 6.10E-03 | 866 |
| 12 | -0.000610273 | -1.000000029 | 24.99389764 | 3.05E-03 | 918 |
| 13 | -0.000305156 | -1.000000035 | 24.99694854 | 1.53E-03 | 968 |
| 14 | -0.000152586 | -1.000000042 | 24.99847417 | 7.63E-04 | 1015 |
| 15 | -7.62955E-05 | -1.000000049 | 24.99923705 | 3.82E-04 | 1060 |
| 16 | -3.81455E-05 | -1.000000055 | 24.99961855 | 1.91E-04 | 1102 |
| 17 | -1.90746E-05 | -1.000000062 | 24.99980925 | 9.54E-05 | 1142 |
| 18 | -9.53797E-06 | -1.000000069 | 24.99990462 | 4.77E-05 | 1182 |
| 19 | -1.95462E-05 | -1.000000075 | 24.99980454 | 2.53E-04 | 1224 |
| 20 | -2.38211E-06 | -1.000000082 | 24.99997618 | 2.17E-04 | 1264 |
| 21 | -1.23903E-05 | -1.000000089 | 24.9998761 | 5.32E-04 | 1306 |
| 22 | -5.92105E-07 | -1.000000095 | 24.99999408 | 5.23E-04 | 1346 |
| 23 | -1.06003E-05 | -1.000000102 | 24.999894 | 1.78E-03 | 1388 |
| 24 | -1.45781E-07 | -1.000000109 | 24.99999854 | 1.78E-03 | 1428 |
| 25 | -1.0154E-05 | -1.000000115 | 24.99989846 | 6.82E-03 | 1470 |
| 26 | -1.20743E-07 | -1.000000122 | 24.99999879 | 7.43E-02 | 1510 |
| 27 | -1.0129E-05 | -1.000000129 | 24.99989871 | 5.37E-02 | 1552 |
| 28 | -1.26747E-07 | -0.999999962 | 24.99999873 | 2.74E-02 | 1592 |
| 29 | -1.0135E-05 | -0.999999969 | 24.99989865 | 1.10E-01 | 1634 |
| 30 | -4.14786E-09 | -0.999999976 | 24.99999996 | 1.10E-01 | 1674 |
| 31 | -1.00124E-05 | -0.999999982 | 24.99989988 | 4.31E-01 | 1716 |
| 32 | 3.15195E-09 | -0.999999989 | 25.00000003 | 4.31E-01 | 1756 |



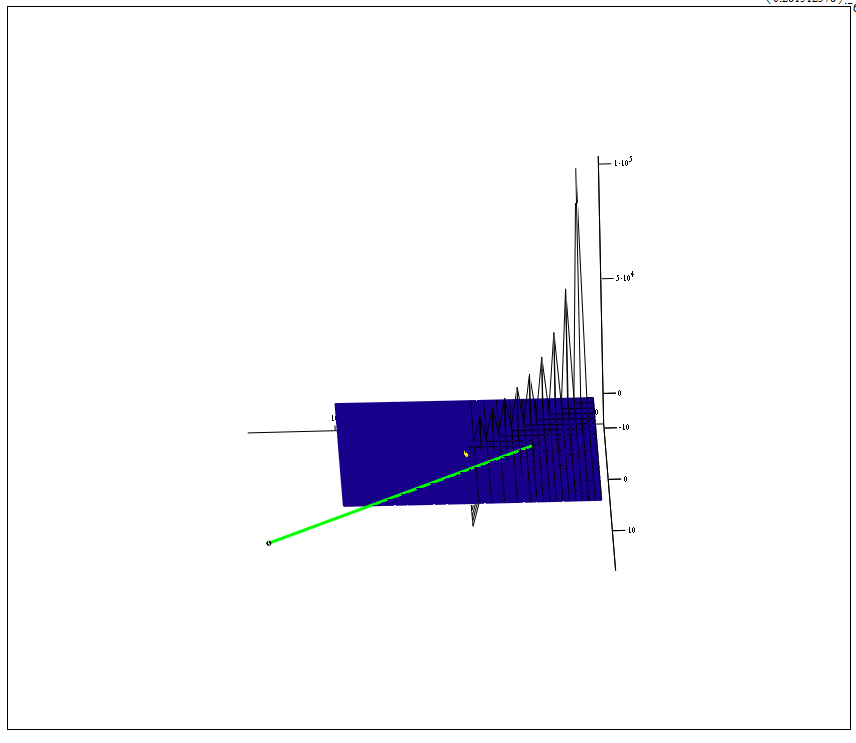
# Метод барьерных функций

## Барьерные функции обратны ограничениям



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 16 | 17 | 765 | 0.00E+00 | 0 |
| 0 | -4.987032704 | -0.9800557 | 0.000565926 | 7.65E+02 | 138 |
| 1 | -5.003005198 | -0.990051616 | 0.000108002 | 5.56E-02 | 224 |
| 2 | -5.001522872 | -0.995012994 | 2.71894E-05 | 2.77E-02 | 297 |
| 3 | -5.000766496 | -0.997503264 | 6.82121E-06 | 1.39E-02 | 366 |
| 4 | -5.000384523 | -0.998750811 | 1.70833E-06 | 6.94E-03 | 430 |
| 5 | -5.000192584 | -0.9993752 | 4.27464E-07 | 3.47E-03 | 489 |
| 6 | -5.000096374 | -0.999687553 | 1.06911E-07 | 1.74E-03 | 544 |
| 7 | -5.00004821 | -0.999843767 | 2.6733E-08 | 8.68E-04 | 593 |
| 8 | -5.000024109 | -0.999921878 | 6.68421E-09 | 4.34E-04 | 640 |
| 9 | -5.000012053 | -0.999960933 | 1.67152E-09 | 2.17E-04 | 682 |
| 10 | -5.000006025 | -0.999980468 | 4.17797E-10 | 1.09E-04 | 724 |
| 11 | -5.000006031 | -0.999990235 | 1.31728E-10 | 5.43E-05 | 764 |
| 12 | -5.000006038 | -0.999995115 | 6.0315E-11 | 2.71E-05 | 804 |
| 13 | -5.00000075 | -0.999997557 | 6.532E-12 | 1.36E-05 | 844 |
| 14 | -5.000000756 | -0.999998778 | 2.064E-12 | 6.78E-06 | 884 |
| 15 | -5.000000763 | -0.999999388 | 9.56E-13 | 3.39E-06 | 924 |
| 16 | -5.00000077 | -0.999999696 | 6.84E-13 | 1.70E-06 | 964 |
| 17 | -5.000000776 | -0.999999845 | 6.27E-13 | 8.48E-07 | 1004 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 0 | 1 | 29 | 0.00E+00 | 0 |
| 0 | 0.301209908 | 0.006151743 | 29.11516782 | 1.#IOe+000 | 110 |
| 1 | 0.282327225 | -0.007794097 | 28.88745347 | 1.69E-02 | 196 |
| 2 | 0.281545414 | -0.008369873 | 28.87805227 | 3.09E-05 | 260 |
| 3 | 0.281512378 | -0.008394188 | 28.87765509 | 5.54E-08 | 304 |



## Барьерные функции связанные логарифмически с ограничениями



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | 0 | 1 | 29 | 0.00E+00 | 0 |
| 0 | 0.091222769 | -0.245812258 | 26.48934844 | 1.#IOe+000 | 108 |
| 1 | 0.083926653 | -0.249685821 | 26.40928159 | 3.72E-03 | 186 |
| 2 | 0.083867402 | -0.249717179 | 26.40863207 | 2.59E-07 | 233 |

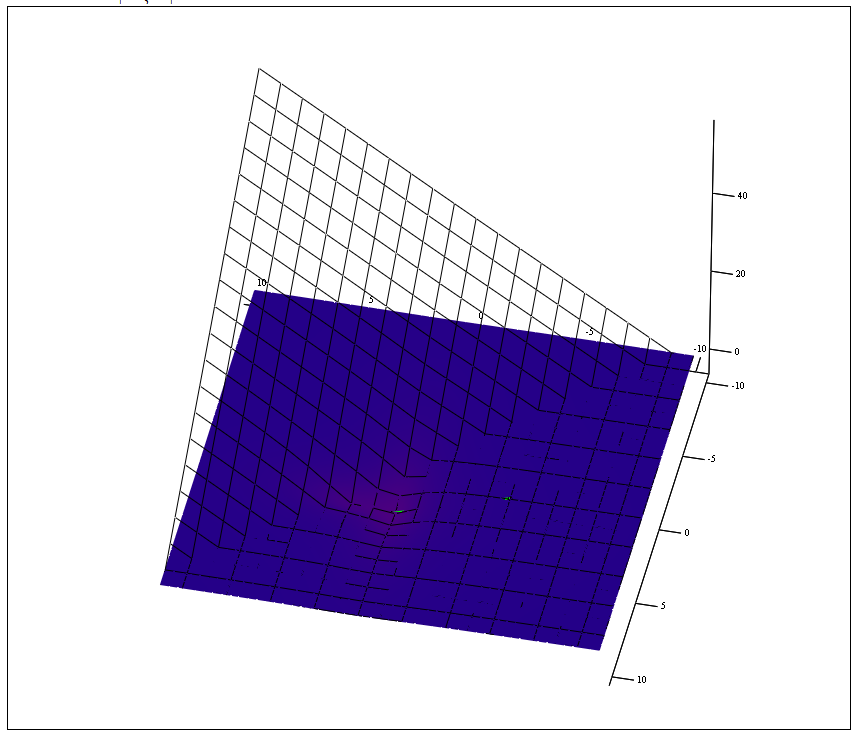
Из другого начального приближения была ошибка, поскольку одно из ограничений было изначально выполнено не удалось посчитать логарифм от отрицательного числа.

# Тестирование на функции из варианта

## Штрафные функции



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | -4.00000000 | 1.00000000 | -0.32541993 | 0.00E+00 | 0 |
| 0 | 1.06994005 | 2.00000002 | -4.53876823 | 8.21E+00 | 123 |
| 1 | 1.13883243 | 2.00000002 | -4.55099905 | 1.22E-02 | 192 |
| 2 | 1.13883242 | 2.00000001 | -4.55099905 | 0.00E+00 | 232 |



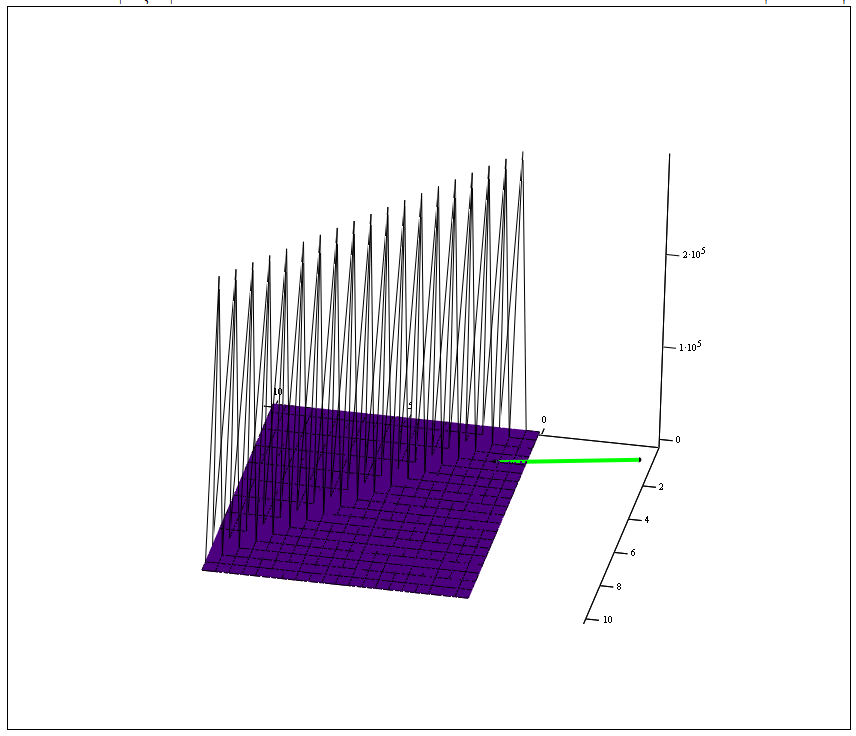


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | -4.00000000 | 1.00000000 | -0.32541993 | 0.00E+00 | 0 |
| 0 | 1.06994005 | 2.00000002 | -4.53876823 | 3.62E+01 | 123 |
| 1 | 1.13883243 | 2.00000002 | -4.55099905 | 1.22E-02 | 192 |
| 2 | 1.13883242 | 2.00000001 | -4.55099905 | 0.00E+00 | 232 |

## Барьерные функции



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер итерации |  |  |  |  | Кол-во вычислений функции |
| -1 | -4.00000000 | 1.00000000 | -0.32541993 | 0.00E+00 | 0 |
| 0 | 4.00000000 | 3.00000000 | -1.63003663 | 5.49E+08 | 125 |
| 1 | 3.99997001 | 4.12084308 | -0.80764957 | 5.49E+08 | 204 |
| 2 | 1.13909695 | 2.05759436 | -4.54708144 | 1.67E+04 | 326 |
| 3 | 1.17182599 | 2.05846446 | -4.54423816 | 2.99E-03 | 405 |
| 4 | 1.17183731 | 2.05846445 | -4.54423634 | 3.49E-10 | 445 |



# Выводы

В зависимости от типа ограничений и исходной функции, а так же метода поиска минимума надо выбирать метод (барьерных функций или штрафных), а так же вид этих функций. Так же можно подстраивать стратегию уменьшения (увеличения) коэффициентов штрафа. В методе барьерных функций, надо аккуратно выбирать критерий выхода из итерационного процента, поскольку при большом количестве итераций барьер может быть резко возрастающим и принимать большие значения.

# Код программы

## Файл «extra.h»

#pragma once

#include <math.h>

#define N\_dim 2 //размерность пространства

//Двухмерный вектор и матрица

struct vect\_2d{

double x, y;

vect\_2d(){

}

vect\_2d(double s\_x, double s\_y){

x = s\_x; y = s\_y;

}

vect\_2d operator+ (vect\_2d op2){

return vect\_2d(x+op2.x, y+op2.y);

}

vect\_2d operator- (vect\_2d op2){

return vect\_2d(x-op2.x, y-op2.y);

}

double operator\* (vect\_2d op2){

return x\*op2.x + y\*op2.y;

}

vect\_2d operator\* (double op2){

return vect\_2d(op2\*x, op2\*y);

}

double norm(){

return sqrt(x\*x + y\*y);

}

double& operator[] (int i){

if(i==0) return x;

if(i==1) return y;

}

};

struct matrix\_2d{

vect\_2d col1, col2;

matrix\_2d(){

};

matrix\_2d(vect\_2d s\_c1, vect\_2d s\_c2){

col1 = s\_c1; col2 = s\_c2;

}

matrix\_2d(double a11, double a12, double a21, double a22){

col1 = vect\_2d(a11, a21); col2 = vect\_2d(a12, a22);

}

vect\_2d operator \* (vect\_2d op2){

return vect\_2d(col1.x\*op2.x + col2.x\*op2.y, col1.y\*op2.x + col2.y\*op2.y);

}

matrix\_2d operator\* (matrix\_2d op2){

vect\_2d n\_col1 = vect\_2d(col1.x \* op2.col1.x + col2.x\*op2.col1.y, col1.x \* op2.col2.x + col2.x\*op2.col2.y);

vect\_2d n\_col2 = vect\_2d(col1.y \* op2.col1.x + col2.y\*op2.col1.y, col1.y \* op2.col2.x + col2.y\*op2.col2.y);

return matrix\_2d(n\_col1, n\_col2);

}

matrix\_2d operator\* (double op2){

return matrix\_2d(col1\*op2, col2\*op2);

}

matrix\_2d operator+(matrix\_2d op2){

return matrix\_2d(col1+op2.col1, col2+op2.col2);

}

};

## Файл «functor.h»

#pragma once

#include "extra.h"

#include <vector>

using namespace std;

typedef double(\*func\_2d)(vect\_2d); //функция двух переменных

typedef double(\*penalty\_func)(func\_2d, vect\_2d); //функция штрафа

//Класс функтора, использующийся для формирование функции с ограничениями

class functor{

public:

//Конструктор по-умолчанию

functor(){

}

//Конструкутор с параметрами

functor(func\_2d set\_mf, double set\_r0, vector<func\_2d> set\_bounds, vector<penalty\_func> set\_pen, vector<double> set\_pen\_coef){

init(set\_mf, set\_r0, set\_bounds, set\_pen, set\_pen\_coef);

}

//Инициализация

void init(func\_2d set\_mf, double set\_r0, vector<func\_2d> set\_bounds, vector<penalty\_func> set\_pen, vector<double> set\_pen\_coef){

main\_f = set\_mf;

penal\_r0 = set\_r0;

bounds = set\_bounds;

penaltys = set\_pen;

penaltys\_coef = set\_pen\_coef;

}

//Собственно вычисление

double operator()(vect\_2d arg){

double rez; //резульатт вычислений

double penalty = 0; //штраф

for(int i = 0; i < penaltys.size(); i++){

penalty += penaltys\_coef[i] \* penaltys[i](bounds[i], arg); //штраф от определённой компоненты

}

rez = main\_f(arg) + penal\_r0 \* penalty;

return rez;

}

//Вычисление значения основной функции

double main\_val(vect\_2d arg){

return main\_f(arg);

}

//Увеличить штраф

void panalty\_grow(){

for(int i = 0; i < penaltys\_coef.size(); i++){

penaltys\_coef[i] \*= 2;

}

}

//Уменьшить штраф

void panalty\_low(){

for(int i = 0; i < penaltys\_coef.size(); i++){

penaltys\_coef[i] /= 2;

}

}

bool check\_penalty(vect\_2d arg){

for(int i = 0; i < bounds.size(); i++){

if(bounds[i](arg) > 0)

return false;

}

return true;

}

private:

func\_2d main\_f; //основная функция

double penal\_r0; //основной коэффициент штрафов

vector<func\_2d> bounds; //ограничения

vector<penalty\_func> penaltys; //штрафные функции

vector<double> penaltys\_coef; //коэффциенты штрафа

};

## Файл «Gauss.h»

#pragma once

#include <stdio.h>

#include "functor.h"

//Метод Гаусса, для оптимизции функции двух аргументов с ограничениями, в качестве одномерного поиска используется метод Фиббоначи

class Gauss{

public:

Gauss(){} //Конструктор по-умолчанию

void set\_func(functor set\_min\_f); //Установка минимизируемой функции

void set\_start(vect\_2d set\_x0); //Установка начального приближения

void set\_eps(double set\_eps\_G); //Установка требуемой точности

vect\_2d find\_min(); //Поиск минимума

private:

functor min\_f; //Минимизируемая функция

vect\_2d x\_k; //Приближение

double eps\_G; //точность

int func\_calc; //количество вычислений функции

//x\_i - это номер компоненты, которая сейчас минимизируется

double Fib(double a, double b, int x\_i); // Используемый одномерный метод поиска - метод Фибоначчи

void find\_area(double x0, double& a, double& b, int x\_i); // Используемый метод для определения области одномреного минимума

double one\_min\_f(double x0, int x\_i); // Функция для одномерной минимизации

};

## Файл «Gauss.cpp»

#include "Gauss.h"

void Gauss::set\_func(functor set\_min\_f){

min\_f = set\_min\_f;

}

void Gauss::set\_start(vect\_2d set\_x0){

x\_k = set\_x0;

}

void Gauss::set\_eps(double set\_eps\_G){

eps\_G = set\_eps\_G;

}

vect\_2d Gauss::find\_min(){

func\_calc = 0;

vect\_2d arg\_min; //точка минимума

double func\_prev; //значение функции в точке с предыдущей итерацией

double func\_new; //значение функции на текущей итерации

double diff; //Разность между значениями функции на соседнийх итерациях

double a\_min, b\_min; //отрезок для одномерной минимизациия

int iter = 0; //счётчик итераций

FILE\* out\_tr = fopen("trajectory.txt", "w"); //Данные для вывода траектории(для графика)

FILE\* out\_f = fopen("iters\_info.txt", "w"); //Данные для вывода итераций(для табличек)

func\_prev = min\_f(x\_k);

fprintf(out\_tr, "%.15lf\t%.15lf\t%.15lf\n", x\_k.x, x\_k.y, min\_f.main\_val(x\_k));

fprintf(out\_f, "%d\t%.15lf\t%.15lf\t%.15lf\t%.3e\t%d\n", -1, x\_k.x, x\_k.y, min\_f.main\_val(x\_k), 0.0, func\_calc);

do{

for(int i = 0; i < N\_dim; i++){

find\_area(x\_k[i], a\_min, b\_min, i);

x\_k[i] = Fib(a\_min, b\_min, i);

}

func\_new = min\_f(x\_k);

diff = fabs(func\_new - func\_prev);

func\_prev = func\_new;

fprintf(out\_tr, "%.15lf\t%.15lf\t%.15lf\n", x\_k.x, x\_k.y, min\_f.main\_val(x\_k));

fprintf(out\_f, "%d\t%.15lf\t%.15lf\t%.15lf\t%.3e\t%d\n", iter, x\_k.x, x\_k.y, min\_f.main\_val(x\_k), diff, func\_calc);

printf("%5d)%.15lf\t%.15lf\t%.3e\r",iter, x\_k.x, x\_k.y, diff);

iter++;

if(!min\_f.check\_penalty(x\_k))

min\_f.panalty\_low();

}while(diff > eps\_G);

printf("\n");

arg\_min = x\_k;

fclose(out\_tr);

fclose(out\_f);

return arg\_min;

}

double Gauss::one\_min\_f(double x0, int x\_i){

vect\_2d arg1; //аргумент для минимизациия

for(int i = 0; i < N\_dim; i++){

if(i == x\_i)

arg1[i] = x0;

else

arg1[i] = x\_k[i];

}

return min\_f(arg1);

}

double Gauss::Fib(double a, double b, int x\_i){

double eps = 1E-8;

double x1, x2, f1, f2;

double fib\_max = (b-a)/eps;

long long int add\_fib;

int n = 2;

int point\_num;

vector<long long int> fib\_numbers;

fib\_numbers.push\_back(1);

fib\_numbers.push\_back(1);

do{

add\_fib = fib\_numbers[n-1] + fib\_numbers[n-2];

fib\_numbers.push\_back(add\_fib);

n++;

}while(fib\_max > add\_fib);

n = fib\_numbers.size() - 3;

x1 = a + fib\_numbers[n]\*(b-a)/fib\_numbers[n+2];

x2 = a + b - x1;

f1 = one\_min\_f(x1, x\_i);

f2 = one\_min\_f(x2, x\_i);

func\_calc += 2;

if(f1 < f2){

b = x2;

x2 = x1;

f2 = f1;

point\_num = 1;

}

else{

a = x1;

x1 = x2;

f1 = f2;

point\_num = 2;

}

const int true\_iters = n;

for(int k = 1; k < true\_iters; k++){

switch(point\_num){

case 1:{

x1 = a + fib\_numbers[n-k+1]\*(b-a)/fib\_numbers[n-k+3];

f1 = one\_min\_f(x1, x\_i);

}break;

case 2:{

x2 = a + fib\_numbers[n-k+2]\*(b-a)/fib\_numbers[n-k+3];

f2 = one\_min\_f(x2, x\_i);

}break;

};

func\_calc++;

if(f1 < f2){

b = x2;

x2 = x1;

f2 = f1;

point\_num = 1;

}

else{

a = x1;

x1 = x2;

f1 = f2;

point\_num = 2;

}

}

return (a+b)/2 ;

}

void Gauss::find\_area(double x0, double &a, double &b, int x\_i){

double delta = 1E-5;

double f0 = one\_min\_f(x0, x\_i);

double x1 = x0+delta;

double f1 = one\_min\_f(x1, x\_i);

double h = delta;

int k = 1;

func\_calc += 2;

if(f0 < f1){

x1 = x0 - delta;

h \*= -1;

}

bool end\_cycle = false;

while(!end\_cycle){

h \*= 2;

x0 = x1 + h;

f0 = one\_min\_f(x0, x\_i);

func\_calc++;

k++;

if(f1 > f0){

x1 = x0;

f1 = f0;

}

else{

end\_cycle = true;

x1 = x0;

x0 -= h + h/2;

}

};

if(x1>x0){

a = x0;

b = x1;

}

else{

a = x1;

b = x0;

}

}

## Файл «launch.cpp»

#include <windows.h>

#include "Gauss.h"

//Штрафные функции

double pen1(func\_2d f, vect\_2d arg){

return (f(arg) + fabs(f(arg))) / 2.0;

}

double pen2(func\_2d f, vect\_2d arg){

return pow((f(arg) + fabs(f(arg))), 2) / 2.0;

}

double pen3(func\_2d f, vect\_2d arg){

return pow((f(arg) + fabs(f(arg))), 4) / 2.0;

}

//Барьерные функции

double pen4(func\_2d f, vect\_2d arg){

return -1/f(arg);

}

double pen5(func\_2d f, vect\_2d arg){

return -log(-f(arg));

}

double bound1(vect\_2d arg){

return arg.x - arg.y - 1;

}

double bound2(vect\_2d arg){

return -arg.x;

}

double bound3(vect\_2d arg){

return arg.x - 4;

}

double func(vect\_2d arg){

return (arg.x + 5) \* (arg.x + 5) + (arg.y + 1) \* (arg.y + 1);

}

int main(){

//Данные для функтора

vector<func\_2d> bound\_v;

vector<penalty\_func> pen\_v;

vector<double> coef\_v;

double r0 = 1.0;

//Установка ограничений

bound\_v.push\_back(bound1);

bound\_v.push\_back(bound2);

bound\_v.push\_back(bound3);

//Установка функций штрафа

pen\_v.push\_back(pen5);

pen\_v.push\_back(pen5);

pen\_v.push\_back(pen5);

//Установка коэффициентов штрафа

coef\_v.push\_back(1.0);

coef\_v.push\_back(1.0);

coef\_v.push\_back(1.0);

functor min\_funct(func, r0, bound\_v, pen\_v, coef\_v);

//Данные для метода

double eps = 1E-6;

vect\_2d start(16, 17);

vect\_2d arg\_min;

Gauss our\_meth;

our\_meth.set\_func(min\_funct);

our\_meth.set\_start(start);

our\_meth.set\_eps(eps);

arg\_min = our\_meth.find\_min();

printf("%.15lf\t%.15lf\n", arg\_min.x, arg\_min.y);

system("pause");

return 0;

}