Министерство образования и науки РФ

Новосибирский государственный технический университет

Кафедра прикладной математики

Лабораторная работа №2

по методам оптимизации

Факультет: ПМИ

Группа: ПМ-01

Студент: Конев А.М., Ряховский М.И.

Вариант: 12

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

Новосибирск

2013

# Цель работы

Ознакомится с методами поиска минимума функции  переменных в оптимизационных задачах без ограничений.

# Часть 1



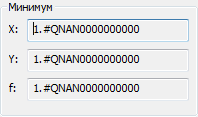
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Флетчера-Ривса | F:\OM\2\мсгФР 1.png  Кол-во вычислений функции: 250 | F:\OM\2\мсгФР 2.png  Кол-во вычислений функции: 129 | F:\OM\2\мсгФР 3.png  Кол-во вычислений функции: 747 |
| Бройдена | F:\OM\2\Бройдена 1.png  Кол-во вычислений функции: 124 | F:\OM\2\Бройдена 2.png  Кол-во вычислений функции: 10 | F:\OM\2\Бройдена 3.png  Кол-во вычислений функции: 128 |
| Ньютона | F:\OM\2\Ньютона 1.png  Кол-во вычислений функции: 1132 | F:\OM\2\Ньютона 2.png  Кол-во вычислений функции: 758 | F:\OM\2\Ньютона 3.png  Кол-во вычислений функции: 16773 |
| Гаусса | F:\OM\2\Гаусса 1.png  Кол-во вычислений функции: 458 | F:\OM\2\Гаусса 2.png  Кол-во вычислений функции: 356 | F:\OM\2\Гаусса 3.png  Кол-во вычислений функции: 299 |



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Флетчера-Ривса | F:\OM\2\-6, -6, -9\мсгФР 1.png  Кол-во вычислений функции: 250 | F:\OM\2\-6, -6, -9\мсгФР 2.png  Кол-во вычислений функции: 129 | F:\OM\2\-6, -6, -9\мсгФР 3.png  Кол-во вычислений функции: 747 |
| Бройдена | F:\OM\2\-6, -6, -9\Бройдена 1.png  Кол-во вычислений функции: 124 | F:\OM\2\-6, -6, -9\Бройдена 2.png  Кол-во вычислений функции: 10 | F:\OM\2\-6, -6, -9\Бройдена 3.png  Кол-во вычислений функции: 128 |
| Ньютона | F:\OM\2\-6, -6, -9\Ньютон 1.png  Кол-во вычислений функции: 1444 | F:\OM\2\-6, -6, -9\Ньютон 2.png  Кол-во вычислений функции: 1071 | F:\OM\2\-6, -6, -9\Ньютон 3.png  Кол-во вычислений функции: 18705 |
| Гаусса | F:\OM\2\-6, -6, -9\Гаусс 1.png  Кол-во вычислений функции: 458 | F:\OM\2\-6, -6, -9\Гаусс 2.png  Кол-во вычислений функции: 356 | F:\OM\2\-6, -6, -9\Гаусс 3.png  Кол-во вычислений функции: 299 |



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Флетчера-Ривса | F:\OM\2\-6, -9, -9\мсгФР 1.png  Кол-во вычислений функции: 250 | F:\OM\2\-6, -9, -9\мсгФР 2.png  Кол-во вычислений функции: 220 | F:\OM\2\-6, -9, -9\мсгФР 3.png  Кол-во вычислений функции: 751 |
| Бройдена | F:\OM\2\-6, -9, -9\Бройдена 1.png  Кол-во вычислений функции: 124 | F:\OM\2\-6, -9, -9\Бройдена 2.png  Кол-во вычислений функции: 10 | F:\OM\2\-6, -9, -9\Бройдена 3.png  Кол-во вычислений функции: 128 |
| Ньютона | F:\OM\2\-6, -9, -9\Ньютона 1(не работал).png  Кол-во вычислений функции: 46 | F:\OM\2\-6, -9, -9\Ньютона 2 (сделал пару итераций, но в итоге - ошибка).png  Кол-во вычислений функции: 373 | F:\OM\2\-6, -9, -9\Ньютона 3(не работал).png  Кол-во вычислений функции: 600 |
| Гаусса | F:\OM\2\-6, -9, -9\Гаусса 1.png  Кол-во вычислений функции: 458 | F:\OM\2\-6, -9, -9\Гаусса 2.png  Кол-во вычислений функции: 356 | F:\OM\2\-6, -9, -9\Гаусса 3.png  Кол-во вычислений функции: 299 |

В последнем случае, для любого из указанных приближений метод Ньютона выдавал следующий результат: 

# Вывод по части 1

Скорость сходимости и сходимость метода вообще зависит от начального приближения и поведения функции, некоторые методы(среди приведённых выше – метод Бройдена) на указанной функции могут очень плохо сходится(на данной функции), при разных начальным приближениях, другие(методы Гаусса и Флетчера-Ривса) сходятся почти всегда.

При изменении точности сильнее всего менялись вычислительные затраты у метода Ньютона, и в последнем случае он перестал получать адекватный результат. Остальные методы почти не меняли свои вычислительные затраты, при изменении точности.

* Метод Флетчера-Ривса: ломанная с небольшим числом изломов;
* Метод Бройдена: почти прямая;
* Метод Ньютона: ломаная с большим числом изломов, иногда осциллирующая;
* Метод Гаусса: набор ортогональных отрезков.

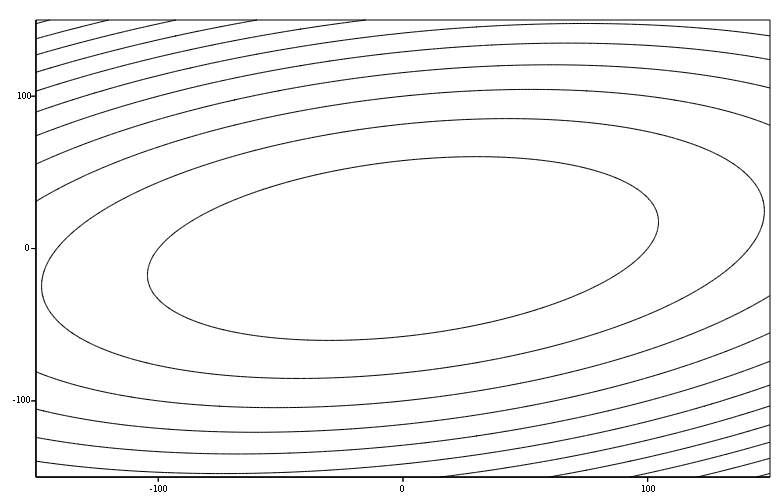
# Часть 2

## Тестирование на квадратичной функции

Квадратичная функция: 

Точный минимум: 

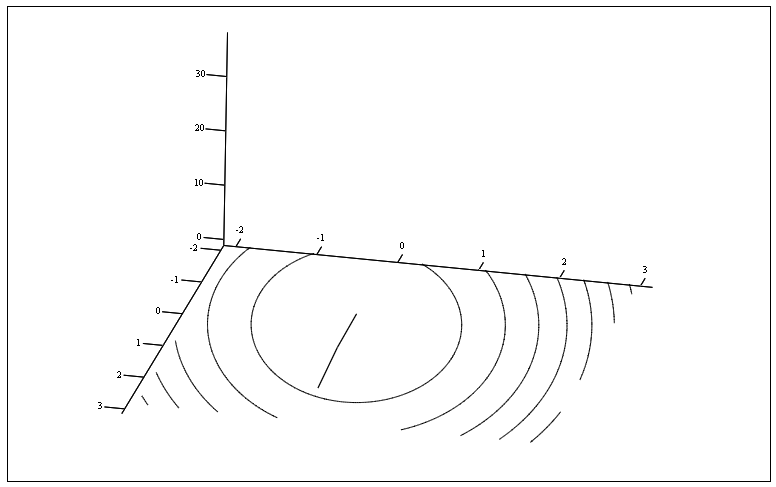
Линии уровня:



**Тест 1.**

Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 2.000000000000000 | 2.000000000000000 | 1.01980e+001 | 0 |
| 0 | 1.633802820220717 | 0.169014101103587 | 3.15996e+000 | 52 |
| 1 | -0.000000010783766 | 0.000000006119390 | 5.49802e-008 | 108 |



**Тест 2.**

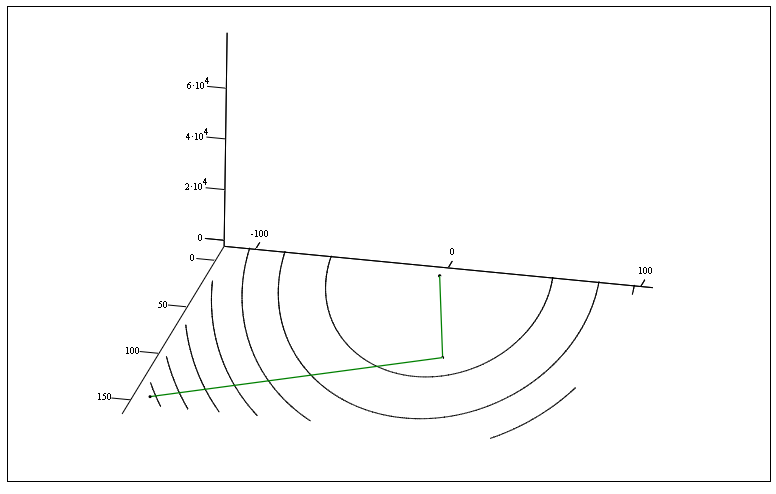
Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 2.000000000000000 | 2.000000000000000 | 1.01980e+001 | 0 |
| 0 | 1.633802820220717 | 0.169014101103587 | 3.15996e+000 | 52 |
| 1 | -0.000000010783766 | 0.000000006119390 | 5.49802e-008 | 108 |
| 2 | -0.000000000000000 | 0.000000000000000 | 4.11603e-017 | 167 |

**Тест 3.**

Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 151.000000000000000 | -106.000000000000000 | 8.86472e+002 | 0 |
| 0 | 82.656995839327422 | 25.828294790316960 | 1.57116e+002 | 52 |
| 1 | 0.000000849789558 | -0.000000761004021 | 5.94857e-006 | 108 |
| 2 | 0.000000000000001 | -0.000000000000001 | 4.45332e-015 | 167 |

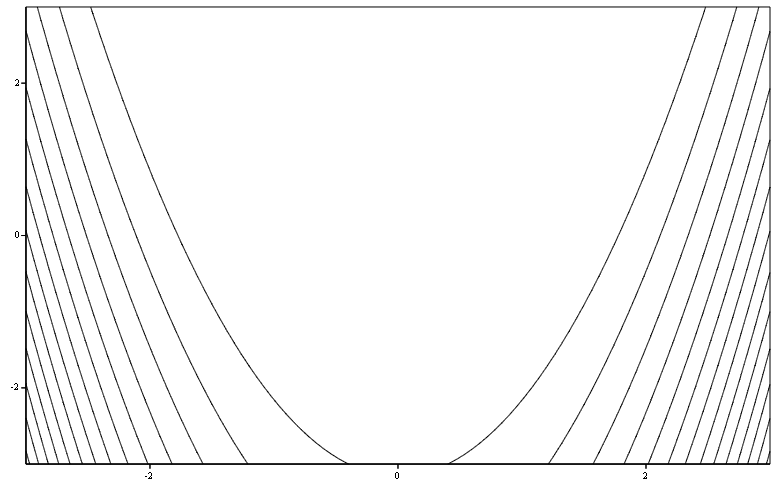
****

## Тестирование на функции Розенброка



Точный минимум: 

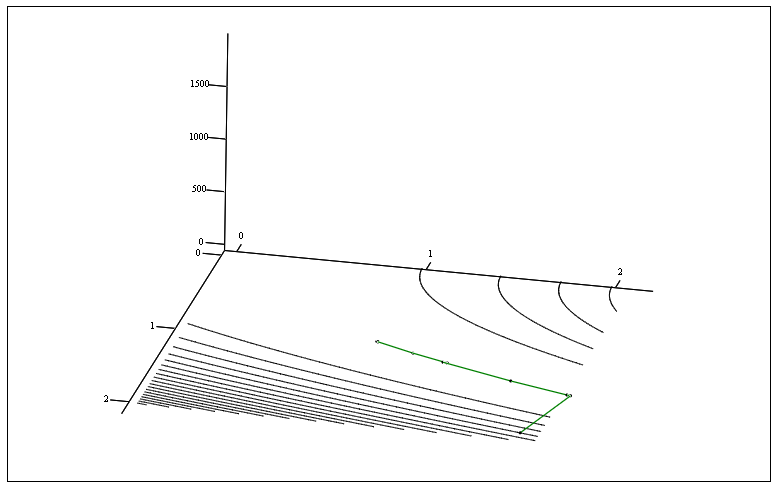
Линии уровня:



# Тест 1

Точность: 

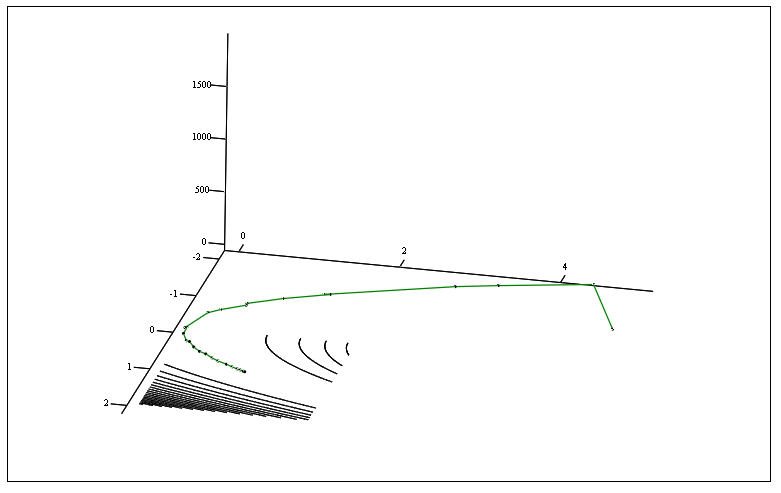
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 2.000000000000000 | 2.000000000000000 | 1.65118e+003 | 0 |
| 0 | 1.460564279128211 | 2.134690567009186 | 2.98961e-001 | 30 |
| 1 | 1.455391745272424 | 2.113991119959921 | 3.44344e+000 | 79 |
| 2 | 1.343861673896696 | 1.790082011983874 | 9.75666e+000 | 148 |
| 3 | 1.186036532930975 | 1.393476409176965 | 7.14353e+000 | 207 |
| 4 | 1.191092010205846 | 1.417921202942941 | 7.69257e-001 | 256 |
| 5 | 1.105794395619726 | 1.215990083686236 | 3.49054e+000 | 315 |
| 6 | 1.002289100603031 | 1.004051499383473 | 2.42433e-001 | 374 |
| 7 | 1.003429650858960 | 1.006782599757629 | 4.59125e-002 | 426 |
| 8 | 0.999994559721669 | 0.999986598971112 | 1.11748e-003 | 487 |
| 9 | 1.000000713177735 | 1.000001408825128 | 9.13810e-006 | 543 |



# Тест 2

Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | -1.000000000000000 | 5.000000000000000 | 1.78528e+003 | 0 |
| 0 | -2.106799486384682 | 4.445213290032741 | 1.47032e+000 | 32 |
| 1 | -2.103431005948571 | 4.438479201211190 | 6.28443e+000 | 71 |
| 2 | -1.841201705306446 | 3.335698799449812 | 4.69657e+001 | 142 |
| 3 | -1.708818310740536 | 2.839013793143975 | 6.29380e+001 | 198 |
| 4 | -1.185004837806624 | 1.356716634879432 | 2.85244e+001 | 267 |
| 5 | -1.199682218082431 | 1.430966086462338 | 8.53049e+000 | 316 |
| 6 | -0.977636774683807 | 0.906920237783377 | 2.50442e+001 | 377 |
| 7 | -0.765210443923247 | 0.522419029597842 | 2.61086e+001 | 431 |
| 8 | -0.717282491739029 | 0.525889702647572 | 2.28507e+000 | 490 |
| 9 | -0.541193453800124 | 0.257847917663638 | 1.27644e+001 | 549 |
| 10 | -0.430015150275601 | 0.132117530204404 | 1.59401e+001 | 605 |
| 11 | 0.010788710204867 | -0.025910241372859 | 5.52972e+000 | 678 |
| 12 | -0.000155059012900 | -0.005315556798703 | 2.26556e+000 | 727 |
| 13 | 0.166712520732912 | 0.000104132037058 | 5.54071e+000 | 788 |
| 14 | 0.345477497069932 | 0.087216095430069 | 7.15027e+000 | 844 |
| 15 | 0.358719345025485 | 0.131145971003404 | 1.70919e+000 | 905 |
| 16 | 0.487818746961405 | 0.220097524134014 | 4.34013e+000 | 959 |
| 17 | 0.587638990536363 | 0.320696937348473 | 6.99157e+000 | 1020 |
| 18 | 0.636953734176216 | 0.411996944624736 | 2.64575e+000 | 1081 |
| 19 | 0.725977787579105 | 0.516990517029173 | 3.10900e+000 | 1135 |
| 20 | 0.789365452421682 | 0.608137740801202 | 5.24043e+000 | 1194 |
| 21 | 0.852528829573156 | 0.732421323570349 | 2.47907e+000 | 1258 |
| 22 | 0.906971287866809 | 0.818577186104601 | 1.50498e+000 | 1312 |
| 23 | 0.947560388721998 | 0.891920985083487 | 2.45751e+000 | 1373 |
| 24 | 0.966784915938943 | 0.935647303558125 | 4.84120e-001 | 1432 |
| 25 | 0.989799376265569 | 0.978747988577736 | 4.05421e-001 | 1488 |
| 26 | 1.000378413873608 | 1.000706448821077 | 2.32805e-002 | 1549 |
| 27 | 1.000009765547939 | 1.000020235825453 | 2.97783e-004 | 1605 |



# Тест 3

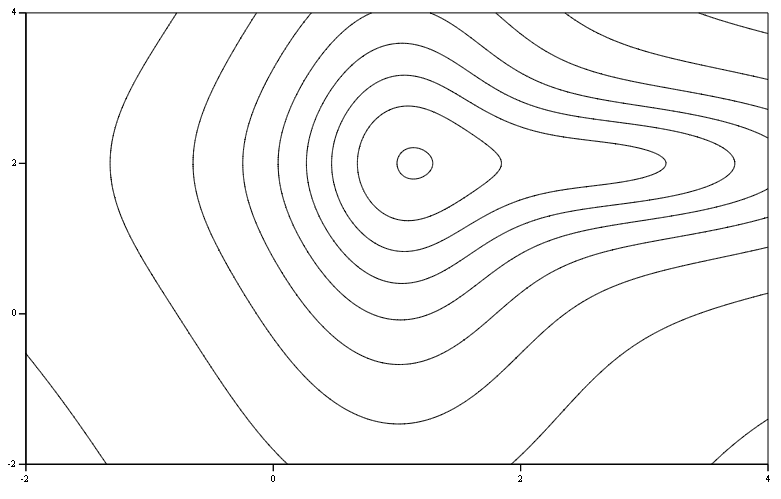
Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 2.000000000000000 | 2.000000000000000 | 1.65118e+003 | 0 |
| 0 | 1.460564279128211 | 2.134690567009186 | 2.98961e-001 | 30 |
| 1 | 1.455391745272424 | 2.113991119959921 | 3.44344e+000 | 79 |
| 2 | 1.343861673896696 | 1.790082011983874 | 9.75666e+000 | 148 |
| 3 | 1.186036532930975 | 1.393476409176965 | 7.14353e+000 | 207 |
| 4 | 1.191092010205846 | 1.417921202942941 | 7.69257e-001 | 256 |
| 5 | 1.105794395619726 | 1.215990083686236 | 3.49054e+000 | 315 |
| 6 | 1.002289100603031 | 1.004051499383473 | 2.42433e-001 | 374 |
| 7 | 1.003429650858960 | 1.006782599757629 | 4.59125e-002 | 426 |
| 8 | 0.999994559721669 | 0.999986598971112 | 1.11748e-003 | 487 |
| 9 | 1.000000713177735 | 1.000001408825128 | 9.13810e-006 | 543 |
| 10 | 1.000000000018315 | 1.000000000044090 | 3.30335e-009 | 602 |

## Тестирование на заданной функции



Линии уровня:



# Тест 1

Точность: 

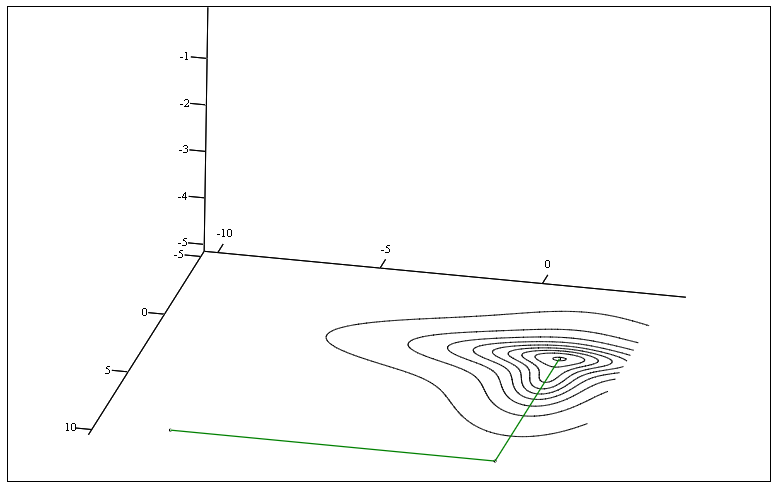
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 1.000000000000000 | 0.000000000000000 | 8.64639e+002 | 0 |
| 0 | 1.000192761654087 | 2.000031049125818 | 7.48917e-001 | 37 |
| 1 | 1.138832378363304 | 2.000137155010453 | 5.81874e-003 | 89 |
| 2 | 1.138825213428120 | 2.000000796216297 | 4.92452e-005 | 155 |

# F:\OM\2\Часть 2\Вариант 1,0 10-3.png

# Тест 2

Точность: 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 10.000000000000000 | -8.000000000000000 | 1.53907e+006 | 0 |
| 0 | 9.999999954200481 | 2.002232434059634 | 4.70367e+000 | 20 |
| 1 | 1.138832364317517 | 1.998240284619400 | 7.46554e-002 | 93 |
| 2 | 1.138832491020316 | 1.998240284691891 | 7.46554e-002 | 115 |
| 3 | 1.138884138404552 | 1.998243505435294 | 7.45154e-002 | 179 |
| 4 | 1.138832851016732 | 2.000000031477863 | 2.50675e-006 | 238 |



# Тест 3

Точность: 

Приведено только часть из итераций из начало и из конца, метод завершил поиск, поскольку , а этот вектор используется для нахождения скалярного произведения, которое расположено в знаменателе.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Число вычислений функции |
| -1 | 10.000000000000000 | -8.000000000000000 | 1.53907e+006 | 0 |
| 0 | 9.999999954200481 | 2.002232434059634 | 4.70367e+000 | 20 |
| 1 | 1.138832364317517 | 1.998240284619400 | 7.46554e-002 | 93 |
| 2 | 1.138832491020316 | 1.998240284691891 | 7.46554e-002 | 115 |
| 3 | 1.138884138404552 | 1.998243505435294 | 7.45154e-002 | 179 |
| 4 | 1.138832851016732 | 2.000000031477863 | 2.50675e-006 | 238 |
| 5 | 1.138832851004002 | 2.000000031476924 | 2.50667e-006 | 258 |
| 6 | 1.138832850991214 | 2.000000031475981 | 2.50660e-006 | 278 |
| 7 | 1.138832850978411 | 2.000000031475037 | 2.50652e-006 | 298 |
|  |  |  |  |  |
| 11495 | 1.138832758829904 | 2.000000024466007 | 1.96059e-006 | 233946 |
| 11496 | 1.138832758819946 | 2.000000024465279 | 1.96053e-006 | 233966 |
| 11497 | 1.138832758810250 | 2.000000024464570 | 1.96047e-006 | 233986 |
| 11498 | 1.138832758804045 | 2.000000024464117 | 1.96044e-006 | 234006 |
| 11499 | 1.138832758794905 | 2.000000024463449 | 1.96038e-006 | 234026 |
| 11500 | 1.138832758788600 | 2.000000024462988 | 1.96035e-006 | 234046 |
| 11501 | 1.138832758779382 | 2.000000024462314 | 1.96029e-006 | 234066 |
| 11502 | 1.138832758773178 | 2.000000024461861 | 1.96026e-006 | 234086 |

# Выводы по части 2

Для указанных функций метод Бройдена сходится хорошо, однако в некоторых случаях, где градиент мало меняется в окрестности экстремума, до нужной точности может сходится продолжительное время.

# Тест программы

## Файл “extra.h”

#pragma once

#include <math.h>

//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);

}

};

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 “Broyden.h”

#pragma once

#include <vector>

#include <stdio.h>

#include <math.h>

#include "extra.h"

using namespace std;

//Реализация метода Бройдена

typedef double(\*func\_2d)(vect\_2d);

typedef vect\_2d(\*grad\_f)(vect\_2d);

class Broyden{

public:

void init(func\_2d set\_f, grad\_f set\_g, double set\_eps, int set\_iter); //установка вычисляемой функции, её градиента и точности

void minimization(vect\_2d x0, vect\_2d &x\_m); //Минимизация функции, x0 - начальная точка, x\_m - точка минимума

private:

func\_2d min\_f; //минимизируеммая функция

grad\_f min\_f\_grad;

long int calc\_count; //количество вычислений функции

double meth\_eps; //точтность метода

int max\_iter; //максимальнео число итераций

vect\_2d x\_k; //приближение на преддущей операции

vect\_2d grad\_f\_k; //градиент на предыдущей итерации

matrix\_2d etta\_k; //приближение матрицы Гессе не предыдущей итерации

double Fib(double a, double b); // Используемый одномерный метод поиска - метод Фибоначчи

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

double one\_min\_f(double lambda); // Функция для одномерной минимизации

};

## Файл “Broyden.cpp”

#include "Broyden.h"

void Broyden::init(func\_2d set\_f, grad\_f set\_g, double set\_eps, int set\_iter){

min\_f = set\_f;

min\_f\_grad = set\_g;

meth\_eps = set\_eps;

max\_iter = set\_iter;

}

void Broyden::minimization(vect\_2d x0, vect\_2d &x\_m){

x\_k = x0;

bool end\_cycle = false;

etta\_k = matrix\_2d(1,0,0,1); //начальное приблежение - единичная матрица

calc\_count = 0;

grad\_f\_k = min\_f\_grad(x\_k);

int iter = 0;

FILE \*out\_f = fopen("bro.txt", "w");

FILE \*out\_fc = fopen("broc.txt", "w");

fprintf(out\_fc,"%.15lf\t%.15lf\t%.15lf\n", x\_k.x, x\_k.y, -5.0);

fprintf(out\_f,"%d\t%.15lf\t%.15lf\t%.5e\t%d\n", -1, x\_k.x, x\_k.y, grad\_f\_k.norm(), calc\_count);

while(!end\_cycle && iter < max\_iter){

if(iter == 11818)

iter = iter;

double a, b; //отрезок одномерной минимизации

find\_area(0, a, b); //назодим отрезок

double lambda\_k = Fib(a,b); //поличаем лямбду

vect\_2d x\_k1 = x\_k - (etta\_k\*grad\_f\_k)\*lambda\_k; //новое приближение

vect\_2d grad\_f\_k1 = min\_f\_grad(x\_k1); //новый градиент

vect\_2d dg = grad\_f\_k1 - grad\_f\_k;

vect\_2d dx = x\_k1 - x\_k;

vect\_2d add\_v = dx - etta\_k\*dg; //вспомогательный вектор

if(add\_v.norm() != 0){

double denom = 1.0/(add\_v \* dg);

//Комнотенты новой матрицы

double a11 = add\_v.x \* add\_v.x;

double a12 = add\_v.x \* add\_v.y;

double a22 = add\_v.y \* add\_v.y;

matrix\_2d d\_etta = matrix\_2d(a11,a12,a12,a22) \* denom;

etta\_k = etta\_k + d\_etta;

x\_k = x\_k1;

grad\_f\_k = grad\_f\_k1;

double g\_norm = grad\_f\_k.norm();

if(g\_norm < meth\_eps)

end\_cycle = true;

printf("%d\t%.15lf\t%.15lf\t%.5e\t%d\r", iter, x\_k.x, x\_k.y, g\_norm, calc\_count);

fprintf(out\_f,"%d\t%.15lf\t%.15lf\t%.5e\t%d\n", iter, x\_k.x, x\_k.y, g\_norm, calc\_count);

fprintf(out\_fc,"%.15lf\t%.15lf\t%.15lf\n", x\_k.x, x\_k.y, -5.0);

iter++;

//Обновление метода

if(iter%1000 == 0){

etta\_k = matrix\_2d(1,0,0,1); //начальное приблежение - единичная матрица

grad\_f\_k = min\_f\_grad(x\_k);

}

}

else{

end\_cycle = true;

printf("\nVector add\_v = 0");

}

};

printf("\n");

x\_m = x\_k;

}

double Broyden::one\_min\_f(double lambda){

calc\_count++; //увеличиваем счётчик числа вычислений функции

return min\_f(x\_k - (etta\_k\*grad\_f\_k)\*lambda);

}

double Broyden::Fib(double a, double b){

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);

f2 = one\_min\_f(x2);

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);

}break;

case 2:{

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

f2 = one\_min\_f(x2);

}break;

};

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 Broyden::find\_area(double x0, double& a, double& b){

double delta = 1E-5;

double f0 = one\_min\_f(x0);

double x1 = x0+delta;

double f1 = one\_min\_f(x1);

double h = delta;

int k = 1;

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);

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 "Broyden.h"

//Квадратичная функция

double func(vect\_2d arg){

return arg.x \* arg.x - arg.x \* arg.y + 3 \* arg.y \*arg.y;

}

vect\_2d grad\_func(vect\_2d arg){

return vect\_2d(2\*arg.x - arg.y, 6\*arg.y - arg.x);

}

//Функция Розенброка

double func1(vect\_2d arg){

return 100\*(arg.y - arg.x\*arg.x)\*(arg.y - arg.x\*arg.x) + (1-arg.x)\*(1-arg.x);

}

vect\_2d grad\_func1(vect\_2d arg){

return vect\_2d(400\*arg.x\*arg.x\*arg.x - 400\*arg.x\*arg.y + 2\*arg.x - 2, 200\*(arg.y - arg.x\*arg.x));

}

//Функция из варианта: -3/(1 + ((x-3)/2)^2 + (y-2)^2) - 3/(1 + (x-1)^2 + ((y-2)/3)^2)

double func2(vect\_2d arg){

return (-3.0/(1.0 + (arg.x-3)\*(arg.x-3)/4.0 + (arg.y-2)\*(arg.y-2)) - 3.0/(1.0 + (arg.x-1)\*(arg.x-1) + (arg.y-2)\*(arg.y-2)/9.0));

}

vect\_2d grad\_func2(vect\_2d arg){

double x = arg.x, y = arg.y;

double denom1 = (1.0 + (x-1)\*(x-1)+(y-2)\*(y-2)/9.0)\*(1.0 + (x-1)\*(x-1)+(y-2)\*(y-2)/9.0);

double denom2 = 2\*(1.0 + (x-3)\*(x-3)/4.0 + (y-2)\*(y-2))\*(1.0 + (x-3)\*(x-3)/4.0 + (y-2)\*(y-2));

double grad\_x = 6.0\*(x-1)/denom1 + 3.0\*(x-3)/denom2;

double grad\_y = 2.0\*(y-2)/3.0 \* (1/denom1 + 9\*denom2);

return vect\_2d(grad\_x, grad\_y);

}

int main(){

Broyden our\_meth;

our\_meth.init(func2, grad\_func2, 1E-6, 40000);

vect\_2d start(10, -8);

vect\_2d min;

our\_meth.minimization(start, min);

printf("%.15lf\t%.15lf\n%.15lf\n",min.x, min.y, func2(min));

system("pause");

}