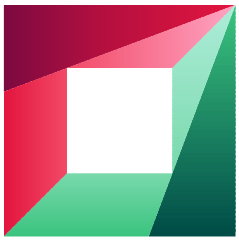
МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ  НЭТИ

Кафедра прикладной математики

Лабораторная работа №2
по дисциплине «Методы оптимизации»

Методы спуска

ФПМИ

Факультет:	ПМИ
Группа:	ПМ-71
Студенты:	Баштовой Павел, Востриков Вячеслав, Бурдуков Вадим
Бригада:	4
Преподаватели:	Чимитова Е.В.
Дата:	01.05.2020

Новосибирск

2020

1. Цель работы:

Ознакомиться с методами поиска минимума функции *n* переменных в оптимизационных задачах без ограничений.

2. Задание к лабораторной работе:

1) Реализовать два метода поиска экстремума функции. Включить в реализуемый алгоритм собственную процедуру, реализующую одномерный поиск по направлению.
2) С использованием разработанного программного обеспечения исследовать алгоритмы на квадратичной функции $f(\bar{x})=100(y-x)^2+(1-x)^2$, функции Розенброка $f(\bar{x})=100(y-x^2)^2+(1-x)^2$ и на заданной в соответствии с вариантом тестовой функции, осуществляя спуск из различных исходных точек.

Выбранные методы: метод Ньютона, метод Бройдена.

Функция варианта 4: $$f(x,y)=\frac{2}{1+(\frac{x-1}{2})^2+(y-2)^2}+\frac{1}{1+(\frac{x-3}{3})^2+(\frac{y-1}{3})^2}$$

3. Ход работы:

1) Метод Ньютона

- Для квадратичной функции: $f(\bar{x})=100(y-x)^2+(1-x)^2$

  Минимум функции достигается в точке (1,1): f(1,1)=0

  Из точки x_0 = (2,9)

i	(x_i,y_i)	$f(x_i, y_i)$	(s_1, s_2)	λ	$\lvert x_i - x_{i-1}\rvert$ $\lvert y_i - y_{i-1}\rvert$ $\lvert f_i - f_{i-1}\rvert$	Градиент, Матрица вторых производных
1	(1.000000,1.000000)	0.000000	(-1.000000,-8.000000)	1.000000	1.000000 8.000000 4901.000000	(-0.000000,0.000000) (-202.000000,200.000000, 200.000000, -200.000000)

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-4	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-5	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-6	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-7	1	7	(1.000000000000, 1.000000000000)	0.000000000000

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-4	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-5	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-6	1	7	(1.000000000000, 1.000000000000)	0.000000000000
1e-7	1	7	(1.000000000000, 1.000000000000)	0.000000000000

- Для функции Розенброка: $f(\bar{x})=100(y-x^2)^2+(1-x)^2$

  Минимум функции достигается в точке (1,1): f(1,1)=0

  Из точки $x_0 = (2,9)$

i	(x_i,y_i)	$f(x_i, y_i)$	(s_1, s_2)	λ	$\lvert x_i - x_i - 1\rvert$ $\lvert y_i - y_i - 1\rvert$ $\lvert f_i - f_i - 1\rvert$	Градиент, Матрица вторых производных
1	(2.001001,4.004004)	1.002003	(0.001001, -4.995996)	1.000000	0.001001 4.995996 2499.997997	(2.002804,-0.000200) (-3205.204406,800.400400, 800.400400,-200.000000)
2	(1.850881,3.403223)	0.774789	(-0.150120,-0.600781)	0.150000	0.150120 0.600781 0.227214	(18.386999,-4.507377) (-2751.622943,740.352374, 740.352374,-200.000000)
3	(1.663938,2.738474)	0.532108	(-0.186943,-0.664749)	1.210000	0.186943 0.664749 0.242682	(21.438203,-6.042992) (-2229.037392,665.575151, 665.575151,-200.000000)
4	(1.389614,1.913485)	0.182575	(-0.274324,-0.824989)	2.910000	0.274324 0.824989 0.349533	(10.530364,-3.508576) (-1553.839343,555.845696, 555.845696,-200.000000)
5	(1.209004,1.448193)	0.061903	(-0.180610,-0.465292)	2.090000	0.180610 0.465292 0.120672	(6.945767,-2.699642) (-1176.752388,483.601717, 483.601717,-200.000000)
6	(1.030486,1.059189)	0.001665	(-0.178518,-0.389004)	3.160000	0.178518 0.389004 0.060238	(1.179098,-0.542523) (-852.606284,412.194429, 412.194429,-200.000000)
7	(1.002422,1.005201)	0.000018	(-0.028065,-0.053988)	1.420000	0.028065 0.053988 0.001647	(-0.136170,0.070336) (-805.738460,400.968610, 400.968610,-200.000000)
8	(0.999947,0.999906)	0.000000	(-0.002474,-0.005295)	0.950000	0.002474 0.005295 0.000018	(-0.004690,0.002292) (-801.910666,399.978812, 399.978812,-200.000000)
9	(1.000000,1.000000)	0.000000	(0.000053,0.000095)	1.000000	0.000053 0.000095 0.000000	(0.000001,-0.000001) (-802.000196,400.000049, 400.000049,-200.000000)

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	9	1405	(1.000000121696, 1.000000240573)	0.000000000000
1e-4	9	1405	(1.000000121696, 1.000000240573)	0.000000000000
1e-5	9	1405	(1.000000121696, 1.000000240573)	0.000000000000
1e-6	10	1412	(1.000000000000, 1.000000000000)	0.000000000000
1e-7	10	1412	(1.000000000000, 1.000000000000)	0.000000000000

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	15	2817	(0.999999995947, 0.999999991853)	0.000000000000
1e-4	15	2817	(0.999999995947, 0.999999991853)	0.000000000000
1e-5	15	2817	(0.999999995947, 0.999999991853)	0.000000000000
1e-6	15	2817	(0.999999995947, 0.999999991853)	0.000000000000
1e-7	15	2817	(0.999999995947, 0.999999991853)	0.000000000000

- Для функции из варианта: $f(x,y) = \frac{2}{1+(\frac{x-1}{2})^2+(y-2)^2} + \frac{1}{1+(\frac{x-3}{3})^2+(\frac{y-1}{3})^2}$

Из точки $x_0 = (2,9)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	8	191	(1.1893595178027849, 1.9746257573697996)	0.37575261574909224
1e-4	9	212	(1.189880298598953, 1.9744474628146045)	0.37575258966566694
1e-5	9	212	(1.189880298598953, 1.9744474628146045)	0.37575258966566694
1e-6	9	212	(1.189880298598953, 1.9744474628146045)	0.37575258966566694
1e-7	12	275	(1.1898762955612774, 1.9744455651609445)	0.3757525896643627

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	10	233	(1.1898450560085228, 1.9744509689736078)	0.37575258974042181
1e-4	10	233	(1.1898450560085228, 1.9744509689736078)	0.37575258974042181
1e-5	10	233	(1.1898450560085228, 1.9744509689736078)	0.37575258974042181
1e-6	12	275	(1,1898787427834705, 1.9744454097641013)	0.37575258966439284
1e-7	13	296	(1.1898745090769687, 1.9744464739296783)	0.37575258966497166

Метод Бройдена:

- Для квадратичной функции: $f(\bar{x}) = 100(y-x)^2 + (1-x)^2$

  Минимум функции достигается в точке (1,1): f(1,1)=0

  Из точки $x_0 = (2,9)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	2	96	(1.000000009119198,1.000000026989493)	0.000000000000032
1e-4	2	96	(1.000000009119198,1.000000026989493)	0.000000000000032
1e-5	2	96	(1.000000009119198,1.000000026989493)	0.000000000000032
1e-6	2	155	(1.000000000000007,1.000000000000006)	0.000000000000000
1e-7	2	155	(1.000000000000007,1.000000000000006)	0.000000000000000

i	(x_i,y_i)	$f(x_i, y_i)$	(s_1, s_2)	λ	$\lvert x_i - x_i{-}1\rvert$ $\lvert y_i - y_i{-}1\rvert$ $\lvert f_i - f_i{-}1\rvert$	Градиент
1	(5.46882e+00,5.49128e+00)	2.00208e+01	(7.03304e-01, 7.10889e-01)	-1.00000e+00	-3.53118e+00 3.49128e+00 -4.94398e+03	(4.44485e+00, 4.49279e+00)
2	(1.00000e+00,1.00000e+00)	3.20179e-14	(-7.05296e-01, 7.08913e-01)	1.00251e+00	-4.46882e+00 -4.49128e+00 -2.00208e+01	(-3.55582e-06, 3.57406e-06)

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	2	98	(0.999999989459194 ,0.999999989383514)	0.000000000000000
1e-4	2	98	(0.999999989459194 ,0.999999989383514)	0.000000000000000
1e-5	2	98	(0.999999989459194 ,0.999999989383514)	0.000000000000000
1e-6	2	98	(0.999999989459194,0.999999989383514)	0.000000000000000
1e-7	2	98	(0.999999989459194,0.999999989383514)	0.000000000000000

- Для функции Розенброка: $f(\bar{x})=100(y-x^2)^2+(1-x)^2$

  Минимум функции достигается в точке (1,1): f(1,1)=0

  Из точки $x_0 = (2,9)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	12	735	(1.000009992393752, 1.000019840511584)	0.000000000101932
1e-4	12	735	(1.000009992393752, 1.000019840511584)	0.000000000101932
1e-5	13	794	(1.000000000344163, 1.000000000927149)	0.000000000000000
1e-6	13	794	(1.000000000344163, 1.000000000927149)	0.000000000000000
1e-7	12	853	(0.999999999999796, 0.999999999999595)	0.000000000000000

i	(x_i,y_i)	$f(x_i, y_i)$	(s_1, s_2)	λ	$\lvert x_i - x_i{-}1\rvert$ $\lvert y_i - y_i{-}1\rvert$ $\lvert f_i - f_i{-}1\rvert$	Градиент
1	(1.55285e+00,2.41371e+00)	3.06201e-01	(6.03756e-01, 7.97169e-00)	2.61840e-05	-7.44715e+00 4.13707e-01 -6.24164e+05	(-3.56608e-01, 4.70848e-01)
2	(1.55267e+00,2.41041e+00)	3.05458e-01	(9.98462e-01, -5.54463e-02)	7.32346e-03	-1.82866e-04 -3.29299e-03 -7.42652e-04	(1.33567e+00, -7.41721e-02)
3	(1.42490e+00,2.01369e+00)	2.08265e-01	(9.51853e-01, -3.06556e-01)	3.01478e+03	-1.27769e-01 -3.96722e-01 -9.71929e-02	(1.03400e+01, -3.33014e+00)
4	(1.30771e+00,1.68961e+00)	1.36730e-01	(9.40407e-01, -3.40050e-01)	1.45087e+00	-1.17188e-01 -3.24084e-01 -7.15357e-02	(1.13409e+01, -4.10086e+00)
5	(1.29019e+00,1.66685e+00)	8.47220e-02	(-7.92313e-01, 6.10115e-01)	-2.08985e+00	-1.75239e-02 -2.27570e-02 -5.20076e-02	(-5.88019e-01, 4.52801e-01)
6	(1.20088e+00,1.43288e+00)	4.88503e-02	(9.34245e-01, -3.56632e-01)	8.25476e-01	-8.93134e-02 -2.33969e-01 -3.58717e-02	(4.83023e+00, -1.84385e+00)
7	(1.12572e+00,1.25558e+00)	2.94221e-02	(9.20704e-01, -3.90262e-01)	5.28337e+00	-7.51520e-02 -1.77298e-01 -1.94283e-02	(5.50572e+00, -2.33373e+00)
8	(1.11058e+00,1.23471e+00)	1.24009e-02	(-8.09526e-01, 5.87084e-01)	-5.55901e-01	-1.51404e-02 -2.08770e-02 -1.70211e-02	(-3.62073e-01, 2.62582e-01)
9	(1.05602e+00, 1.11140e+00)	4.57136e-03	(9.14481e-01, -4.04629e-01)	1.08723e+00	-5.45601e-02 -1.23308e-01 -7.82956e-03	(1.71098e+00, -7.57054e-01)

10	(1.00530e+00, 1.00828e+00)	5.82758e-04	(8.97332e-01, -4.41356e-01)	-5.64398e+00	-5.07190e-02 -1.03118e-01 -3.98860e-03	(9.57626e-01, -4.71011e-01)
11	(1.00887e+00, 1.01768e+00)	8.07962e-05	(9.34879e-01, -3.54967e-01)	2.76342e-01	3.56887e-03 9.39935e-03 -5.01962e-04	(7.58738e-02, -2.88088e-02)
12	(9.99993e-01,9.99974e-01)	1.45782e-08	(8.93889e-01, -4.48289e-01)	1.91349e+00	-8.87972e-03 -1.77062e-02 -8.07816e-05	(4.80652e-03, -2.41049e-03)
13	(1.00001e+00, 1.00002e+00)	1.01932e-10	(9.37418e-01, -3.48207e-01)	9.64787e-01	1.72054e-05 4.63189e-05 -1.44762e-08	(7.77357e-05, -2.88752e-05)

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	2	725	(0.999979765749385, 0.999957637453574)	0.000000000768321
1e-4	2	781	(0.999999630679091,0.999999299328971)	0.000000000000281
1e-5	2	840	(0.999999999966891,0.999999999930543)	0.000000000000000
1e-6	2	840	(0.999999999966891,0.999999999930543)	0.000000000000000
1e-7	2	840	(0.999999999966891 ,0.999999999930543)	0.000000000000000

- Для функции из варианта: $f(x,y) = \frac{2}{1+(\frac{x-1}{2})^2+(y-2)^2} + \frac{1}{1+(\frac{x-3}{3})^2+(\frac{y-1}{3})^2}$

Из точки $x_0 = (2,9)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	4	318	(1.1911556773071319,1.9734521815603459)	0.37575296337604863
1e-4	5	396	(1.1898894674029714, 1.9744492268847906)	0.37575258967770692
1e-5	5	396	(1.1898894674029714, 1.9744492268847906)	0.37575258967770692
1e-6	6	468	(1.1898776527518449, 1.9744456177039862)	0.37575258966426434
1e-7	7	603	(1.1898776067991292, 1.9744457397754252)	0.37575258966425878

Из точки $x_0 = (4,4)$

Точность	Количество итераций	Количество вычислений	Найденная точка	Значение в точке
1e-3	4	321	(1.1900596132701449, 1.9735845679120714)	0.37575279896513375
1e-4	5	398	(1.1898853305930401, 1.974445980276655)	0.37575258966841507
1e-5	5	398	(1.1898853305930401, 1.974445980276655)	0.37575258966841507
1e-6	6	468	(1.1898774749605916, 1.9744461398187774)	0.37575258966430064
1e-7	6	468	(1.1898774749605916, 1.9744461398187774)	0.37575258966430064

4. ***Вывод:*** Метод Бройдена хорошо работает на данной к варианту функции, сходится за наименьшее количество итераций, в отличии от Ньютона, но требует много вычислений. Ньютон требует меньше вычислений, но и сходится медленней, а также чувствителен к начальному приближению: если точка лежит далеко от максимума, метод может не сойтись.

Текст программы (Ньютон):

```
#define _CRT_SECURE_NO_WARNINGS
#include <math.h>
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#define MAX_ITER 10000
using namespace std;
const double eps = 0.00001;
static int f_calc;

struct point
{
	double x, y;
	point operator+ (const point& X)
	{
		point res;
		res.x = this->x + X.x;
		res.y = this->y + X.y;
		return res;
	}
	point operator- (const point& X)
	{
		point res;
		res.x = this->x - X.x;
		res.y = this->y - X.y;
		return res;
	}
	point operator/ (int x)
	{
		point res;
		res.x = this->x / x;
		res.y = this->y / x;
		return res;
	}
	point operator* (int x)
	{
		point res;
		res.x = this->x * x;
		res.y = this->y * x;
		return res;
	}
	point operator-()
	{
		this->x = -this->x;
		this->y = -this->y;
		return *this;
	}
};

double func_1(point x)
```

```
{
	return 2/(1 + ((x.x - 1) / 2) * ((x.x - 1) / 2) + (x.y - 1) * (x.y - 1));
}
double func_2(point x)
{
	return 1/(1 + ((x.x - 3) / 3) * ((x.x - 3) / 3) + ((x.y - 1) / 3) * ((x.y - 1) / 3));
}
double Function(point x)
{
	f_calc++;
	return func_1(x)  + func_2(x);

}

double* Gradient(point x) // первые производные
{
	double* res = new double[2];
	//x
	res[0] = -((x.x - 1) / (((x.x - 1) * (x.x - 1)/4 + (x.y - 1) * (x.y - 1) + 1) * ((x.x - 1) *
(x.x - 1)/4 + (x.y - 1) * (x.y - 1) + 1))) - ((2 * (x.x - 3)) / (9*((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-
1)/9+1) * ((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-1)/9+1)));
	//y
	res[1] =-2*(x.y-1)/(9*((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-1)/9+1)*((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-
1)/9+1))-(4*(x.y-1)/(((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-1)+1)*((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-1)+1)));
	return res;
}

double** Gesse(point x) // вторые производные
{
	double** res = new double* [2];
	res[0] = new double[2];
	res[1] = new double[2];
	double t;
	//xx
	res[0][0] = (8*pow((x.x-3),2)/(81*pow(((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-1)/9+1),3)))-
(2/(9*pow(((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-1)/9+1),2)))-(1/(pow(((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-
1)+1),2)))+(pow((x.x-1),2)/(pow(((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-1)+1),3)));
	//xy
	res[0][1] = (8 *(x.x - 3) * (x.y - 1) / (81 * pow(((x.x - 3) * (x.x - 3) / 9 + (x.y - 1) * (x.y
- 1) / 9 + 1), 3))) + (4 * (x.x - 1) * (x.y - 1) / (pow(((x.x - 1) * (x.x - 1) / 4 + (x.y - 1) * (x.y -
1) + 1), 3)));
	res[1][0] =res[0][1];
	//yy
	res[1][1] = (8*pow((x.y-1),2)/81*pow(((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-1)/9+1),3))+(16*(x.y-
1)*(x.y-1)/pow(((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-1)+1),3))-(2/(9*pow(((x.x-3)*(x.x-3)/9+(x.y-1)*(x.y-
1)/9+1),2)))-(4/(pow(((x.x-1)*(x.x-1)/4+(x.y-1)*(x.y-1)+1),2)));
	return res;
}

double** matrix_H(point x)
{
	double** res = Gesse(x);
	double opred = 0.0;
	double t;
	t = res[0][0];
	res[0][0] = res[1][1];
	res[1][1] = t;
	t = res[0][1];
	res[0][1] = -res[1][0];
	res[1][0] = -t;
	opred += res[0][0] * res[1][1] - res[1][0] * res[0][1];
	for (int i = 0; i < 2; i++)
```

```
			for (int j = 0; j < 2; j++)
				res[i][j] /= opred;
	return res;
}

double** multipl_grad(double** H, point x)
{
	double** res = new  double* [2];
	res[0] = new double;
	res[1] = new double;

	double* grad = Gradient(x);
	res[0][0] = H[0][0] * grad[0] + H[0][1] * grad[1];
	res[1][0] = H[1][0] * grad[0] + H[1][1] * grad[1];
	return res;
}

double lymbda(double** H, point x)
{
	double h = 1;
	double delta = 0.01;
	point res1, res2, res3, m;
	res1.x = x.x - h * H[0][0];
	res1.y = x.y - h * H[1][0];

	res2.x = x.x - (delta + h) * H[0][0];
	res2.y = x.y - (h + delta) * H[1][0];

	res3.x = x.x - (h - delta) * H[0][0];
	res3.y = x.y - (h - delta) * H[1][0];

	if (Function(res1) < Function(res2) && Function(res3) > Function(res1))
		return h;
	else
	{
		if (Function(res2) > Function(res3))
		{
			delta = -delta;
			m = res1;
		}
		else
		{
			m = res1;
		}

	}
	h += delta;
	res1.x = x.x - h * H[0][0];
	res1.y = x.y - h * H[1][0];
	int i = 0;
	while (Function(res1) < Function(m)&& i<MAX_ITER)
	{
		i++;
		h += delta;
		m = res1;
		res1.x = x.x - h * H[0][0];
		res1.y = x.y - h * H[1][0];
	}
	return h;
}

void multipl_H(double alpha, double** H)
```

```
{
	for (int i = 0; i < 2; i++)
		for (int j = 0; j < 1; j++)
			H[i][j] *= alpha;
}

point NewPoint(point x, int iter, double& h)
{
	point res = x;
	double** H = matrix_H(x);
	double** grad_f;
	grad_f = multipl_grad(H, x);
	h = lymbda(grad_f, x);
	multipl_H(h, grad_f);
	res.x = x.x - grad_f[0][0];
	res.y = x.y - grad_f[1][0];
	return res;
}

double Norm(point X)//норма
{
	double res = 0.0;
	double* g = Gradient(X);
	res += sqrt(g[0] * g[0] + g[1] * g[1]);
	return res;
}

point Newton(point x0, double h, double eps)
{
	point x = x0;
	int iter = 0;
	do
	{
		point mem = x;
		iter++;
		x = NewPoint(x, iter, h);

		point s;
		s.x = x.x - mem.x;
		s.y = x.y - mem.y;
		double deltaF;
		deltaF = fabs(Function(x) - Function(mem));

		double* grad = Gradient(x);
		double** H = Gesse(x);

		printf("(%f,%f) %f (%f,%f) %f %f %f %f (%f,%f) \n",
			x.x, x.y, Function(x), s.x, s.y, h, fabs(s.x), fabs(s.y), deltaF, grad[0],
grad[1]);
		printf("(%f,%f,%f,%f)\n", -H[0][0], -H[0][1], -H[1][0], -H[1][1]);
		h = 0.5;
	} while (fabs(Norm(x)) > eps);
	printf("\niter = %d\n", iter);
	printf("count = %d\n", f_calc);
	return x;
}

void main()
{
	double h = 1;
	point X;
	point p0;
```

```
	p0.x = 2.0;
	p0.y = 9.0;
	X = Newton(p0, h, eps);

	printf("X is: %.12lf, %.12lf\n", X.x, X.y);
	printf("func(x) = %.12lf\n", Function(X));

	_getch();
}
```

Текст программы(Бройден):

Launch.cpp

```
#include <windows.h>
#include <iostream>
#include "Broyden.h"

//Квадратичная функция
double func(vect_2d arg){
	double x = arg.x;
	double y = arg.y;
	return 100*(y-x)*(y-x) + (1-x)*(1-x);
}

vect_2d grad_func(vect_2d arg){
	double x = arg.x;
	double y = arg.y;
	return vect_2d(202*x-200*y-2, -200*x+200*y);
}

//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));
}

int main(){
	auto f = func1;
	auto g_f = grad_func1;

	Broyden our_meth;
	our_meth.init(f, g_f, 1E-7, 5000);
	vect_2d start(4, 4);
	vect_2d min;

	our_meth.minimization(start, min);

	printf("%.15lf\t%.15lf\n%.15lf\n",min.x, min.y, f(min));
	std::cout.precision(10);
	std::cout << std::scientific;
	std::cout << min.x << " " << min.y << "\n";
	std::cout << f(min) << "\n";
	system("pause");
```

```
}
```
Broyden.cpp
```
#include "Broyden.h"
#include <iostream>

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 приближение - 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 %.5e\t %.5e\t s1 %.5e\t s2 %.5e\t g1 %.5e\t g2 %.5e\t lambda %.5e\t  %d\n",
		-1, x_k.x, x_k.y, grad_f_k.x / grad_f_k.norm(), grad_f_k.y / grad_f_k.norm(), grad_f_k.x,
grad_f_k.y, -1.0, 0);
	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;
			std::cout << "diff" << std::endl;
			std::cout << "ITER:" <<iter<< std::endl;
			std::cout.precision(5);
			std::cout << std::scientific;
			std::cout << dx.x << std::endl;
			std::cout << dx.y << std::endl;
			std::cout << min_f(x_k1) - min_f(x_k) << std::endl;
			std::cout << "f(x,y) " << min_f(x_k1) << std::endl;

			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 %.5e\t %.5e\t s1 %.5e\t s2 %.5e\t g1 %.5e\t g2 %.5e\t lambda
%.5e\t  %d\n",
                                iter+1, x_k.x, x_k.y, grad_f_k.x / grad_f_k.norm(), grad_f_k.y /
grad_f_k.norm(), grad_f_k.x, grad_f_k.y, lambda_k, 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 приближение - 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;
		}
}
```

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

};
```

Extpa.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);
      }
};
```