

Министерство науки и высшего образования Российской Федерации
Новосибирский государственный технический университет

Методы оптимизации
Лабораторная работа №3

Факультет: ФПМИ
Группа: ПМ-63
Студенты: Кожекин М.В.
Утюганов Д.С.
Вариант: 5

Новосибирск
2019

1. Цель работы

Ознакомиться 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.

2. Задание

Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с использованием **метода штрафных функций**.

Исследовать сходимость **метода штрафных функций** в зависимости от

- выбора штрафных функций,
- начальной величины коэффициента штрафа,
- стратегии изменения коэффициента штрафа,
- начальной точки,
- задаваемой точности.

Сформулировать выводы.

Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с ограничением типа неравенства (**только пункт а**) с использованием **метода барьерных функций**.

Исследовать сходимость **метода барьерных функций** (**только пункт а**) в зависимости от

- выбора барьерных функций,
- начальной величины коэффициента штрафа,
- стратегии изменения коэффициента штрафа,
- начального приближения,
- задаваемой точности.

Сформулировать выводы.

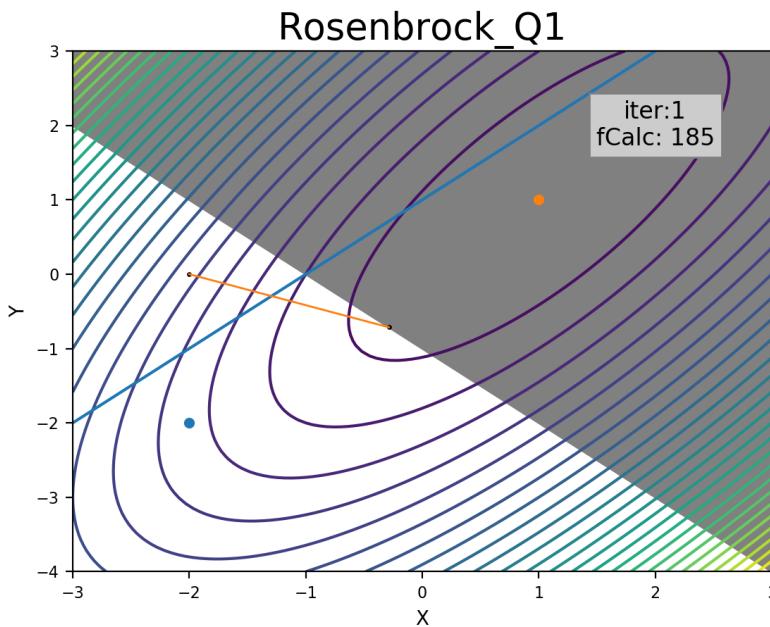
Вариант 5

При ограничении:

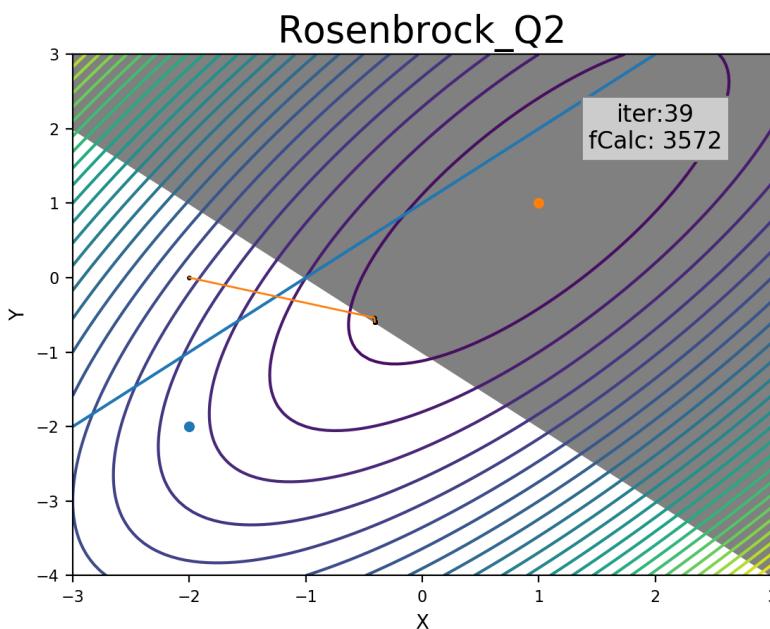
- a) $x + y \leq -1$
- б) $y = x + 1$

3. Исследование сходимости метода штрафных функций

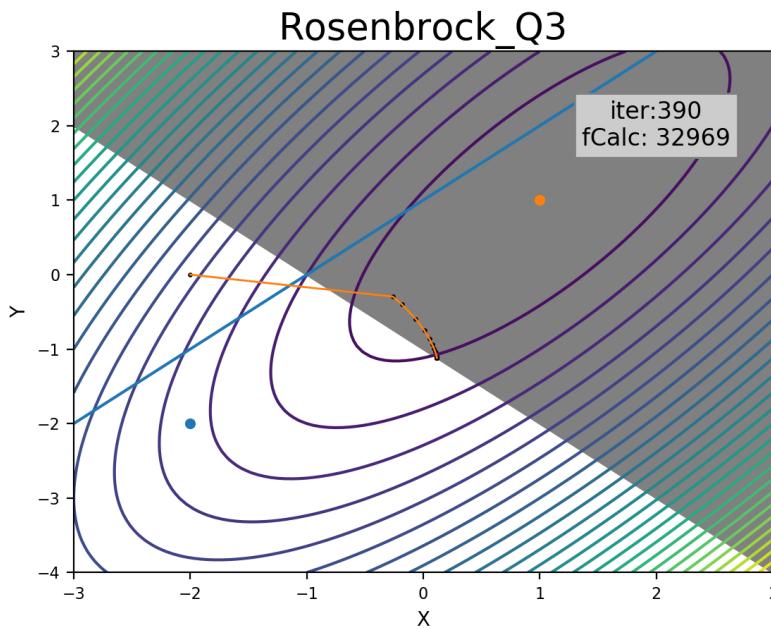
$$G_j[g_j(\bar{x})] = \frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \}$$



$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^2$$



$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^\alpha, \text{если } \alpha \text{ чётное}$$

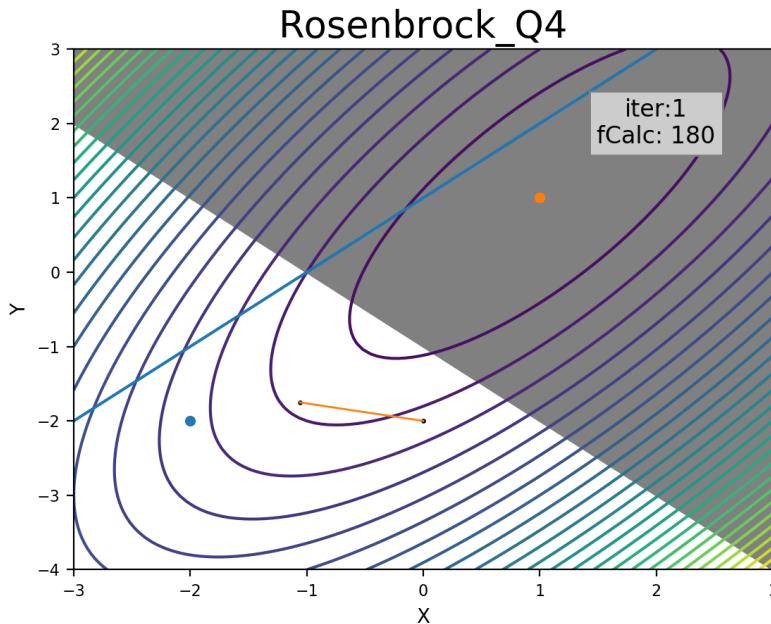


3.1. Вывод

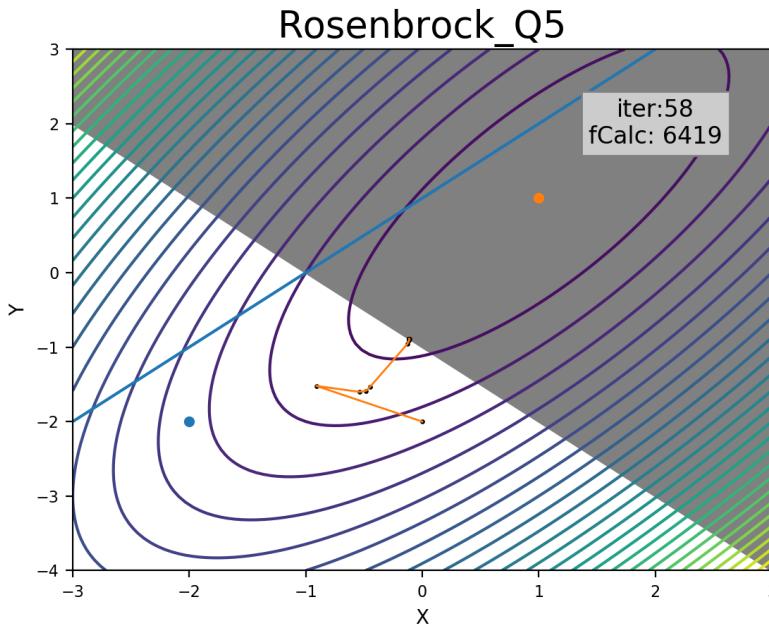
Повышение степени α в штрафной функции ведёт к повышению числа итераций и вычислений целевой функции.

4. Исследование сходимости метода барьерных функций

$$G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$$



$$G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$$



4.1. Вывод

Первая барьерная функция $G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$ находит неоптимальное решение, но всего за 1 итерацию.

Вторая барьерная функция $G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$ находит глобальный минимум за несколько десятков итераций.

5. Зависимость скорости сходимости метода от заданной точности

$$G_j[g_j(\bar{x})] = \frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \}$$

ε	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
$1 \cdot 10^{-3}$	1	99	20	$(-2.860490e - 01, -7.140560e - 01)^T$	5.69
$1 \cdot 10^{-4}$	1	108	20	$(-2.857464e - 01, -7.142962e - 01)^T$	5.69
$1 \cdot 10^{-5}$	1	118	20	$(-2.857203e - 01, -7.142849e - 01)^T$	5.69
$1 \cdot 10^{-6}$	1	127	20	$(-2.857148e - 01, -7.142853e - 01)^T$	5.69
$1 \cdot 10^{-7}$	1	137	20	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^2$$

ε	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
$1 \cdot 10^{-3}$	9	581	$5,120$	$(-2.328447e - 01, -7.663880e - 01)^T$	5.7
$1 \cdot 10^{-4}$	13	828	$81,920$	$(-2.288806e - 01, -7.710665e - 01)^T$	5.71
$1 \cdot 10^{-5}$	16	1,043	$6.55 \cdot 10^5$	$(-2.287137e - 01, -7.712795e - 01)^T$	5.71
$1 \cdot 10^{-6}$	19	1,259	$5.24 \cdot 10^6$	$(-2.286862e - 01, -7.713131e - 01)^T$	5.71
$1 \cdot 10^{-7}$	22	1,491	$4.19 \cdot 10^7$	$(-2.286815e - 01, -7.713184e - 01)^T$	5.71

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^\alpha, \text{если } \alpha \text{ чётное}$$

ε	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
$1 \cdot 10^{-3}$	45	2,508	$3.52 \cdot 10^{14}$	$(-6.784216e - 02, -9.231312e - 01)^T$	6.36
$1 \cdot 10^{-4}$	67	3,697	$1.48 \cdot 10^{21}$	$(-7.322553e - 02, -9.257491e - 01)^T$	6.36
$1 \cdot 10^{-5}$	91	5,181	$2.48 \cdot 10^{28}$	$(-7.499135e - 02, -9.249140e - 01)^T$	6.36
$1 \cdot 10^{-6}$	87	6,014	$1.55 \cdot 10^{27}$	$(-2.640605e - 01, -7.358102e - 01)^T$	5.68
$1 \cdot 10^{-7}$	84	6,623	$1.93 \cdot 10^{26}$	$(-2.637844e - 01, -7.360415e - 01)^T$	5.68

$$G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$$

ε	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
$1 \cdot 10^{-3}$	1	94	5	$(-1.061042e + 00, -1.751568e + 00)^T$	11.68
$1 \cdot 10^{-4}$	1	103	5	$(-1.060892e + 00, -1.750890e + 00)^T$	11.67
$1 \cdot 10^{-5}$	1	113	5	$(-1.060884e + 00, -1.750837e + 00)^T$	11.67
$1 \cdot 10^{-6}$	1	122	5	$(-1.060879e + 00, -1.750834e + 00)^T$	11.67
$1 \cdot 10^{-7}$	1	132	5	$(-1.060879e + 00, -1.750834e + 00)^T$	11.67

$$G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$$

ε	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
$1 \cdot 10^{-3}$	11	768	$4.88 \cdot 10^{-3}$	$(-6.919326e - 01, -6.994372e - 01)^T$	8.6
$1 \cdot 10^{-4}$	25	1,912	$2.98 \cdot 10^{-7}$	$(-2.617377e - 01, -7.386586e - 01)^T$	5.69
$1 \cdot 10^{-5}$	28	2,292	$3.73 \cdot 10^{-8}$	$(-2.603444e - 01, -7.397926e - 01)^T$	5.69
$1 \cdot 10^{-6}$	31	2,703	$4.66 \cdot 10^{-9}$	$(-2.597752e - 01, -7.402737e - 01)^T$	5.68
$1 \cdot 10^{-7}$	45	3,477	$2.84 \cdot 10^{-13}$	$(-2.596667e - 01, -7.403336e - 01)^T$	5.68

5.1. Вывод

Повышение точности ведёт как к повышению точности решения, так и росту количества вычислений целевой функции.

6. Зависимость скорости сходимости метода от коэффициента изменения штрафа

$$G_j[g_j(\bar{x})] = \frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \}$$

$rMult$	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	1	185	20	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
3	1	185	30	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
4	1	185	40	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
5	1	185	50	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
6	1	185	60	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
7	1	185	70	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
8	1	185	80	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
9	1	185	90	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69
10	1	185	100	$(-2.857143e - 01, -7.142857e - 01)^T$	5.69

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^2$$

$rMult$	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	39	3,572	$5.5 \cdot 10^{12}$	$(-2.286810e - 01, -7.713190e - 01)^T$	5.71
3	26	2,488	$2.54 \cdot 10^{13}$	$(-2.138349e - 01, -7.861651e - 01)^T$	5.73
4	21	2,045	$4.4 \cdot 10^{13}$	$(-2.288108e - 01, -7.711892e - 01)^T$	5.71
5	18	1,991	$3.81 \cdot 10^{13}$	$(-2.630373e - 01, -7.369627e - 01)^T$	5.68
6	12	1,573	$2.18 \cdot 10^{10}$	$(-2.631595e - 01, -7.368405e - 01)^T$	5.68
7	15	1,632	$4.75 \cdot 10^{13}$	$(-2.613602e - 01, -7.386398e - 01)^T$	5.68
8	14	1,555	$4.4 \cdot 10^{13}$	$(-2.612669e - 01, -7.387331e - 01)^T$	5.68
9	14	1,559	$2.29 \cdot 10^{14}$	$(-2.629791e - 01, -7.370209e - 01)^T$	5.68
10	13	1,247	$1 \cdot 10^{14}$	$(-4.068592e - 01, -5.931408e - 01)^T$	6.08

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^\alpha, \text{если } \alpha \text{ чётное}$$

$rMult$	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	252	21,777	$7.24 \cdot 10^{76}$	$(-2.640176e - 01, -7.359824e - 01)^T$	5.68
3	164	14,609	$1.77 \cdot 10^{79}$	$(-2.666497e - 01, -7.333503e - 01)^T$	5.68
4	131	11,477	$7.41 \cdot 10^{79}$	$(-9.382202e - 02, -9.061780e - 01)^T$	6.23
5	115	10,255	$2.41 \cdot 10^{81}$	$(-4.045736e - 02, -9.595426e - 01)^T$	6.63
6	103	9,297	$1.41 \cdot 10^{81}$	$(2.094200e - 02, -1.020942e + 00)^T$	7.22
7	96	8,697	$1.35 \cdot 10^{82}$	$(7.382735e - 02, -1.073827e + 00)^T$	7.84
8	90	8,200	$1.9 \cdot 10^{82}$	$(1.064588e - 01, -1.106459e + 00)^T$	8.28
9	85	7,781	$1.29 \cdot 10^{82}$	$(1.188203e - 01, -1.118820e + 00)^T$	8.46
10	81	7,450	$1 \cdot 10^{82}$	$(1.167081e - 01, -1.116708e + 00)^T$	8.43

$$G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$$

$rMult$	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	1	180	5	$(-1.060879e + 00, -1.750834e + 00)^T$	11.67
3	1	180	3.33	$(-1.060879e + 00, -1.750834e + 00)^T$	12.66
4	1	180	2.5	$(-1.060879e + 00, -1.750834e + 00)^T$	13.16
5	1	180	2	$(-1.060879e + 00, -1.750834e + 00)^T$	13.46
6	1	180	1.67	$(-1.060879e + 00, -1.750834e + 00)^T$	13.66
7	1	180	1.43	$(-1.060879e + 00, -1.750834e + 00)^T$	13.8
8	1	180	1.25	$(-1.060879e + 00, -1.750834e + 00)^T$	13.9
9	1	180	1.11	$(-1.060879e + 00, -1.750834e + 00)^T$	13.99
10	1	180	1	$(-1.060879e + 00, -1.750834e + 00)^T$	14.05

$$G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$$

$rMult$	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	58	6,419	$3.47 \cdot 10^{-17}$	$(-2.596589e - 01, -7.403411e - 01)^T$	$6.24 \cdot 10^{291}$
3	18	2,356	$2.58 \cdot 10^{-8}$	$(-6.540634e - 01, -6.540635e - 01)^T$	8.21
4	43	5,321	$1.29 \cdot 10^{-25}$	$(-2.403403e - 01, -7.596596e - 01)^T$	$2.32 \cdot 10^{283}$
5	36	4,271	$6.87 \cdot 10^{-25}$	$(-1.627271e - 01, -8.372728e - 01)^T$	$1.24 \cdot 10^{284}$
6	30	3,561	$4.52 \cdot 10^{-23}$	$(-1.415051e - 01, -8.584949e - 01)^T$	$8.13 \cdot 10^{285}$
7	27	3,196	$1.52 \cdot 10^{-22}$	$(-1.290716e - 01, -8.709284e - 01)^T$	$2.74 \cdot 10^{286}$
8	25	2,956	$2.65 \cdot 10^{-22}$	$(-1.216684e - 01, -8.783316e - 01)^T$	$4.76 \cdot 10^{286}$
9	24	2,802	$1.25 \cdot 10^{-22}$	$(-1.170250e - 01, -8.829749e - 01)^T$	$2.25 \cdot 10^{286}$
10	23	2,681	$1 \cdot 10^{-22}$	$(-1.139555e - 01, -8.860444e - 01)^T$	$1.8 \cdot 10^{286}$

6.1. Вывод

Рост коэффициента изменения штрафа ведёт к уменьшению числа итераций и вычислений целевой функции

7. Зависимость скорости сходимости метода от начального значения штрафа

$$G_j[g_j(\bar{x})] = \frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \}$$

r_0	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	3	474	16	$(-4.717373e - 02, -9.528263e - 01)^T$	6.57
4	2	361	16	$(-2.236842e - 01, -7.763158e - 01)^T$	5.71
8	1	185	16	$(-1.428572e - 01, -8.571428e - 01)^T$	5.96
16	1	185	32	$(-7.142857e - 01, -7.142857e - 01)^T$	8.82
32	1	185	64	$(-1.000000e + 00, -1.000000e + 00)^T$	12
64	1	183	128	$(-1.000000e + 00, -1.000000e + 00)^T$	12

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^2$$

r_0	Итераций	Вычислений f	r	\mathbf{x}	$f(\mathbf{x})$
2	41	3,807	$4.4 \cdot 10^{12}$	$(-2.684542e - 01, -7.315458e - 01)^T$	5.68
4	40	3,646	$4.4 \cdot 10^{12}$	$(-2.689962e - 01, -7.310038e - 01)^T$	5.68
8	39	3,647	$4.4 \cdot 10^{12}$	$(-1.919310e - 01, -8.080690e - 01)^T$	5.78
16	1	185	32	$(-5.652174e - 01, -5.652174e - 01)^T$	7.35
32	1	185	64	$(-7.435897e - 01, -7.435898e - 01)^T$	9.12
64	1	185	128	$(-8.591549e - 01, -8.591549e - 01)^T$	10.37

$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^\alpha, \text{если } \alpha \text{ чётное}$$

r_0	Итераций	Вычислений f	г	\mathbf{x}	$f(\mathbf{x})$
2	257	22,247	$4.63 \cdot 10^{77}$	$(8.733294e - 02, -1.087333e + 00)^T$	8.02
4	254	21,979	$1.16 \cdot 10^{77}$	$(3.775046e - 02, -1.037750e + 00)^T$	7.4
8	252	21,672	$5.79 \cdot 10^{76}$	$(-2.966638e - 02, -9.703336e - 01)^T$	6.72
16	253	21,552	$2.32 \cdot 10^{77}$	$(-2.704472e - 01, -7.295528e - 01)^T$	5.69
32	167	16,005	$5.99 \cdot 10^{51}$	$(-2.628375e - 01, -7.371625e - 01)^T$	5.68
64	166	16,155	$5.99 \cdot 10^{51}$	$(-2.578166e - 01, -7.421833e - 01)^T$	5.68

$$G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$$

r_0	Итераций	Вычислений f	г	\mathbf{x}	$f(\mathbf{x})$
2	28	3,647	$7.45 \cdot 10^{-9}$	$(-2.631686e - 01, -7.368314e - 01)^T$	$1.34 \cdot 10^{300}$
4	1	183	2	$(-8.672954e - 01, -1.296810e + 00)^T$	10.89
8	1	180	4	$(-1.000000e + 00, -1.618034e + 00)^T$	11.6
16	1	177	8	$(-1.227380e + 00, -2.090309e + 00)^T$	11.14
32	1	185	16	$(-1.595075e + 00, -2.780173e + 00)^T$	6.36
64	1	189	32	$(-2.160645e + 00, -3.779900e + 00)^T$	-10.66

$$G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$$

r_0	Итераций	Вычислений f	г	\mathbf{x}	$f(\mathbf{x})$
2	55	5,735	$5.55 \cdot 10^{-17}$	$(-2.765279e - 01, -7.234721e - 01)^T$	$9.98 \cdot 10^{291}$
4	56	5,902	$5.55 \cdot 10^{-17}$	$(-2.765339e - 01, -7.234661e - 01)^T$	$9.98 \cdot 10^{291}$
8	26	3,390	$1.19 \cdot 10^{-7}$	$(-6.597586e - 01, -6.597589e - 01)^T$	8.26
16	58	6,277	$5.55 \cdot 10^{-17}$	$(-2.461733e - 01, -7.538267e - 01)^T$	$9.98 \cdot 10^{291}$
32	57	6,017	$2.22 \cdot 10^{-16}$	$(-2.651979e - 01, -7.348021e - 01)^T$	5.68
64	58	6,079	$2.22 \cdot 10^{-16}$	$(-2.628074e - 01, -7.371927e - 01)^T$	5.68

7.1. Вывод

Для метода барьеров рост начального значения штрафа ведёт к тому, что метод в первую очередь стремится не найти локальный минимум, а в первую очередь стремится отойти как можно дальше от штрафной области.

8. Зависимость скорости сходимости метода от начального приближения

Параметры методов:

$$E = 1e^{-12}$$

$$r_0 = 10$$

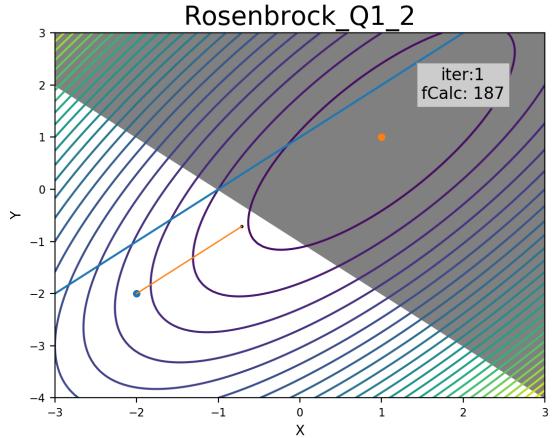
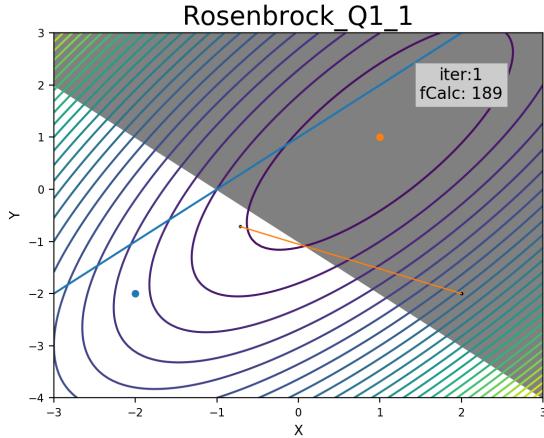
$$rMult = 2$$

$$\alpha = 8$$

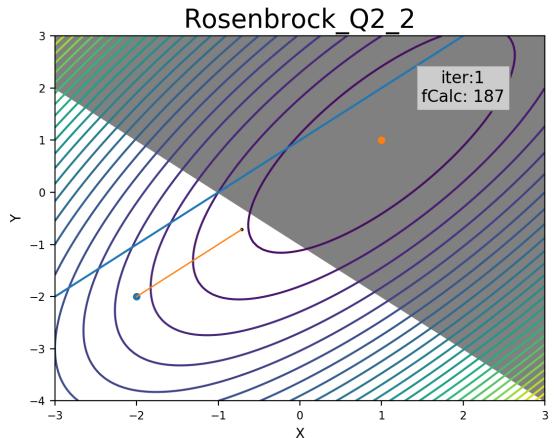
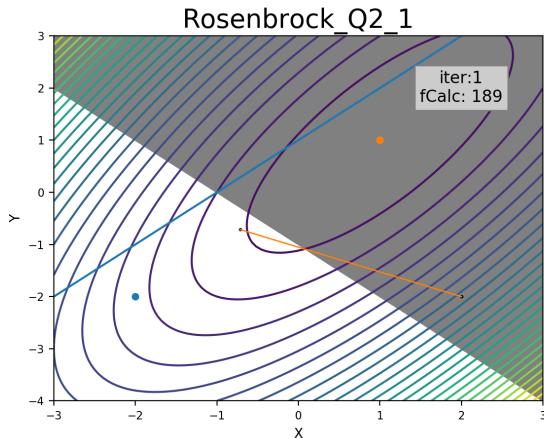
$$x_{0\text{штраф.}} = (2.5, -2)$$

$$x_{0\text{бап.}} = (0, -2)$$

$$G_j[g_j(\bar{x})] = \frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \}$$



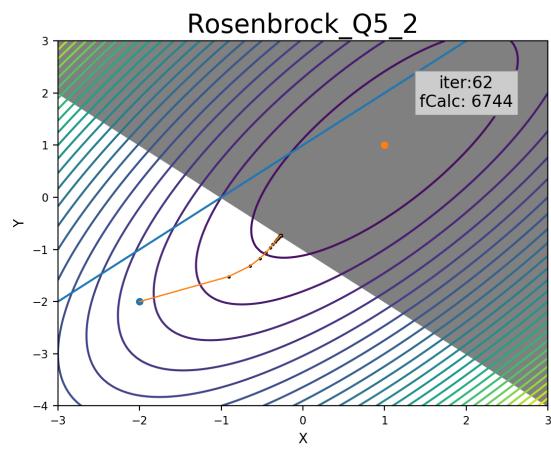
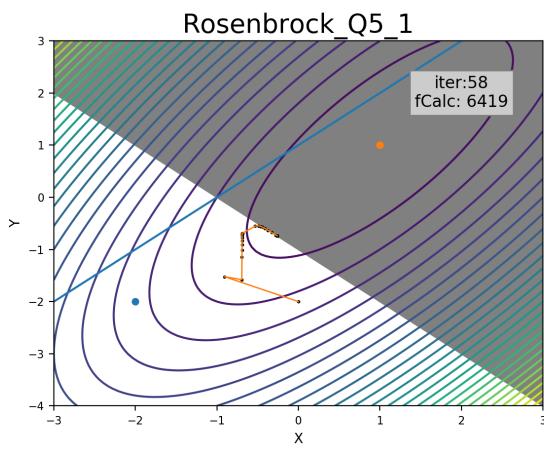
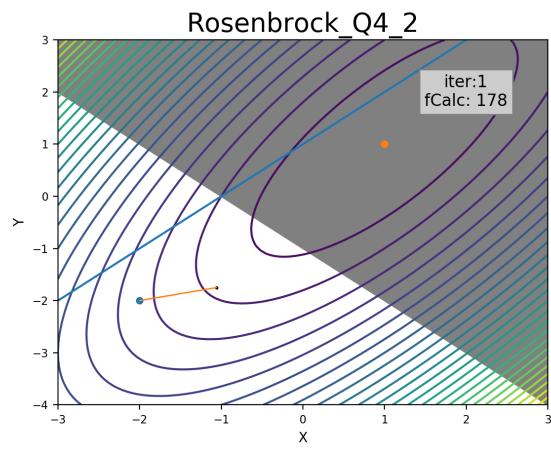
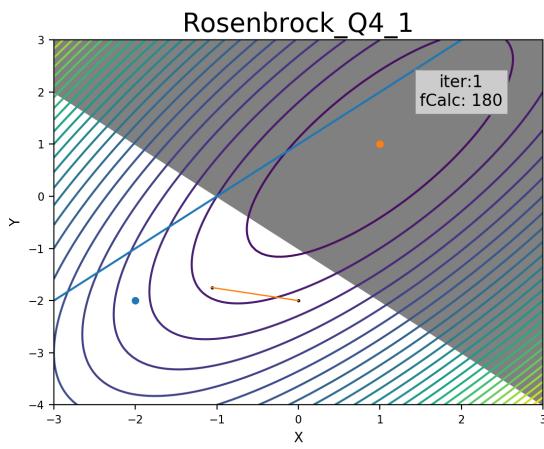
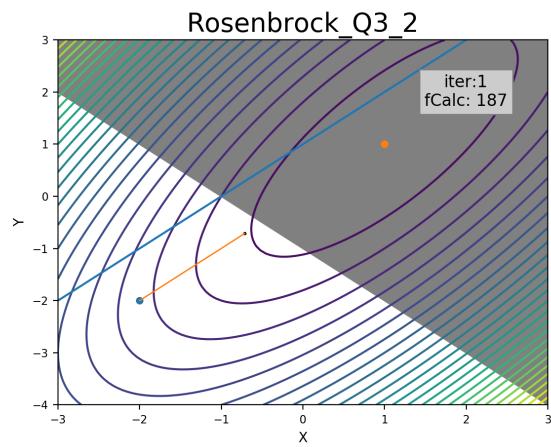
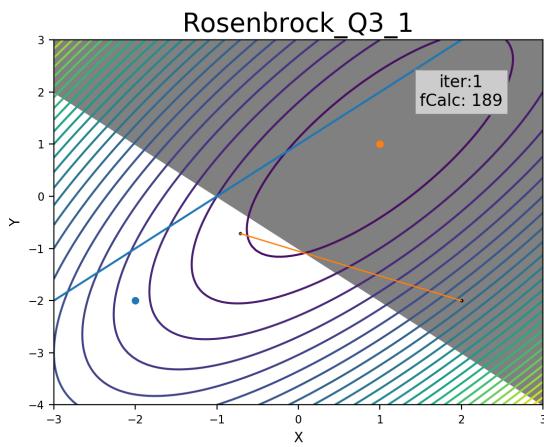
$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^2$$



$$G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{ g_j(\bar{x}) + |g_j(\bar{x})| \} \right]^\alpha, \text{если } \alpha \text{ чётное}$$

$$G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$$

$$G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})]$$



8.1. Вывод

Начальное приближение находящееся в области штрафа заставляет метод быстро покинуть его

9. Исходный код программы

`head.h`

```

1 #pragma once
2 #define _CRT_SECURE_NO_WARNINGS
3 #define UNICODE
4 #include <fstream>
5 #include <iostream>
```

```

6 #include <vector>
7 #include <string>
8 #include <iomanip>
9 #include <functional>
10 #include <cmath>
11 #include <math.h>
12
13 using namespace std;
14
15
16 // float || double
17 typedef double real;
18 typedef vector <real> vector1D;
19 typedef vector <vector <real>> matrix2D;
20
21
22
23 // Умножение на константу
24 inline bool operator==(const vector1D& a, const vector1D& b) {
25 #ifdef _DEBUG
26     if (a.size() != b.size())
27         throw std::exception();
28 #endif
29     for (int i = 0; i < a.size(); ++i)
30         if (a[i] != b[i])
31             return false;
32
33     return true;
34 }
35
36
37 // Сложение векторов
38 inline vector1D operator+(const vector1D& a, const vector1D& b) {
39 #ifdef _DEBUG
40     if (a.size() != b.size())
41         throw std::exception();
42 #endif
43     vector1D result = a;
44     for (int i = 0; i < b.size(); i++)
45         result[i] += b[i];
46     return result;
47 }
48
49
50 // Сложение матриц
51 inline matrix2D operator+(const matrix2D& a, const matrix2D& b) {
52 #ifdef _DEBUG
53     if (a.size() != b.size())
54         throw std::exception();
55 #endif
56     matrix2D result = a;
57     for (int i = 0; i < b.size(); i++)
58         for (int j = 0; j < b.size(); j++)
59             result[i][j] += b[i][j];
60     return result;
61 }
62
63
64 // Сложение матриц
65 inline matrix2D operator/(const matrix2D& a, const real& b) {

```

```

66     matrix2D result = a;
67     for (int i = 0; i < a.size(); i++)
68         for (int j = 0; j < a.size(); j++)
69             result[i][j] /= b;
70     return result;
71 }
72
73
74
75 // Вычитание векторов
76 inline vector1D operator-(const vector1D& a, const vector1D& b) {
77 #ifdef _DEBUG
78     if (a.size() != b.size())
79         throw std::exception();
80 #endif
81     vector1D result = a;
82     for (int i = 0; i < b.size(); i++)
83         result[i] -= b[i];
84     return result;
85 }
86
87
88 inline vector1D operator-(const vector1D& a) {
89     vector1D result = a;
90     for (int i = 0; i < a.size(); i++)
91         result[i] = -result[i];
92     return result;
93 }
94
95
96 // Умножение матрицы на вектор
97 inline vector1D operator*(const matrix2D& a, const vector1D& b) {
98     vector1D result = { 0.0, 0.0 };
99     for (int i = 0; i < a.size(); i++)
100         for (int j = 0; j < a.size(); j++)
101             result[i] += a[i][j] * b[j];
102     return result;
103 }
104
105
106 // Умножение на константу
107 inline vector1D operator*(const vector1D& a, double b) {
108     vector1D result = a;
109     for (int i = 0; i < result.size(); i++)
110         result[i] *= b;
111     return result;
112 }
113
114
115 // Умножение на константу
116 inline vector1D operator*(double b, const vector1D& a) {
117     return operator*(a, b);
118 }
119
120
121 // Деление на константу
122 inline vector1D operator/(&const vector1D& a, double b) {
123     vector1D result = a;
124     for (int i = 0; i < result.size(); i++)
125         result[i] /= b;

```

```

126     return result;
127 }
128
129
130 // Деление на константу
131 inline vector1D operator/(double b, const vector1D& a) {
132     return operator/(a, b);
133 }
134
135
136 // Скалярное произведение
137 inline real operator*(const vector1D& a, const vector1D& b) {
138 #ifdef _DEBUG
139     if (a.size() != b.size())
140         throw std::exception();
141 #endif
142     real sum = 0;
143     for (int i = 0; i < a.size(); i++)
144         sum += a[i] * b[i];
145     return sum;
146 }
147
148
149 // Потоковый вывод вектора
150 inline std::ostream& operator<<(std::ostream& out, const vector1D& v) {
151
152     for (int i = 0; i < v.size() - 1; ++i)
153         out << v[i] << " ";
154     out << v.back();
155     return out;
156 }
157
158 // Потоковый вывод вектора для TeX
159 inline void printTeXVector(std::ofstream &fout, const vector1D &v) {
160     fout << "(";
161     for (int i = 0; i < v.size() - 1; ++i)
162         fout << v[i] << ", ";
163     fout << v.back() << ")^T$";
164 }
165
166 // Потоковый вывод матрицы
167 inline std::ostream& operator<<(std::ostream& out, const matrix2D& v) {
168     for (int i = 0; i < v.size() - 1; ++i)
169         out << v[i] << " ";
170     out << v.back();
171     return out;
172 }
173
174 // Евклидова норма
175 real calcNormE(const vector1D &x) {
176     return sqrt(x*x);
177 }
178
179
180 // Определитель матрицы
181 real det(const matrix2D &m) {
182     return m[0][0] * m[1][1] - m[0][1] * m[1][0];
183 }

```

main.cpp

```
1 #include "head.h"
```

```

2 int fCalcCount, gCalcCount, alpha, rMult;
3 real r;
4
5 //-----
6 struct methodResult
7 {
8     real E;
9     int iterationsCount;
10    int fCalcCount;
11    int r0;
12    vector1D x0, x, xPrev, S, gradf;
13    matrix2D A;
14    real fx, fxPrev, lambda;
15    void printResultE(std::ofstream &fout);
16    void printResultR(std::ofstream &fout);
17    void printResultRFfirst(std::ofstream &fout);
18 };
19
20
21 // Вывод результатов в поток
22 void methodResult::printResultE(std::ofstream &fout) {
23     fout << E << "\t"
24         << iterationsCount << "\t"
25         << fCalcCount << "\t"
26         << r << "\t";
27     printTeXVector(fout, x);
28     fout << "\t" << fx << endl;
29 }
30
31
32 // Вывод результатов в поток
33 void methodResult::printResultR(std::ofstream &fout) {
34     fout << rMult << "\t"
35         << iterationsCount << "\t"
36         << fCalcCount << "\t"
37         << r << "\t";
38     printTeXVector(fout, x);
39     fout << "\t" << fx << endl;
40 }
41
42
43 // Вывод результатов в поток
44 void methodResult::printResultRFfirst(std::ofstream &fout) {
45     fout << r0 << "\t"
46         << iterationsCount << "\t"
47         << fCalcCount << "\t"
48         << r << "\t";
49     printTeXVector(fout, x);
50     fout << "\t" << fx << endl;
51 }
52
53
54
55 //-----
56 // Функция для исследования min=0 в точке (1,1)
57 // x - вектор аргументов функции
58 inline real f(const vector1D &x) {
59     fCalcCount++;
60     return 4 * pow((x[1] - x[0]), 2) + 3 * pow((x[0] - 1), 2);
61 }
```

```

62
63
64
65 // Ограничение области
66 inline real g(const vector1D &x) {
67     gCalcCount++;
68     return x[0] + x[1] + 1;
69 }
70
71
72
73 // -----
74 // Штрафная функция G
75 inline real G1(const vector1D &x) {
76     if (g(x) > 0)
77         return 0.5*(g(x) + fabs(g(x)));
78     else
79         return 0;
80 }
81 inline real G2(const vector1D &x) {
82     if (g(x) > 0)
83         return pow(0.5*(g(x) + fabs(g(x))), 2);
84     else
85         return 0;
86 }
87 inline real G3(const vector1D &x) {
88     if (g(x) > 0)
89         return pow(0.5*(g(x) + fabs(g(x))), alpha);
90     else
91         return 0;
92 }
93 inline real G4(const vector1D &x) {
94     if (g(x) <= 0)
95         return -log(-g(x));
96     else
97         return DBL_MAX;
98 }
99 inline real G5(const vector1D &x) {
100    if (g(x) <= 0)
101        return -1.0 / g(x);
102    else
103        return DBL_MAX;
104 }
105
106
107
108 // -----
109 // Минимизируемая функция Q(x,y)
110 real Q1(const vector1D &x) { return f(x) + r * G1(x); }
111 real Q2(const vector1D &x) { return f(x) + r * G2(x); }
112 real Q3(const vector1D &x) { return f(x) + r * G3(x); }
113 real Q4(const vector1D &x) { return f(x) + r * G4(x); }
114 real Q5(const vector1D &x) { return f(x) + r * G5(x); }
115
116
117
118 // Поиск интервала, содержащего минимум
119 // f — целевая одномерная функция
120 // a,b — искомые границы отрезка
121 // x — начальное приближение

```

```

122 // S - орты направления()
123 void interval(const function<real(const vector1D &x)> &f, real &a, real &b,
124   vector1D &x, vector1D &S)
125 {
126     real lambda0 = 0.0;
127     real delta = 1.0e-8;
128     real lambda_k_minus_1 = lambda0;
129     real f_k_minus_1 = f(x + S * lambda_k_minus_1);
130     real lambda_k;
131     real f_k;
132     real lambda_k_plus_1;
133     real f_k_plus_1;
134     real h;
135     if (f(x + S * lambda0) > f(x + S * (lambda0 + delta)))
136     {
137         lambda_k = lambda0 + delta;
138         h = delta;
139     }
140     else
141     {
142         lambda_k = lambda0 - delta;
143         h = -delta;
144     }
145     f_k = f(x + S * lambda_k);
146     while (true)
147     {
148         h *= 2.0;
149         lambda_k_plus_1 = lambda_k + h;
150         f_k_plus_1 = f(x + S * lambda_k_plus_1);
151         if (f_k > f_k_plus_1)
152         {
153             lambda_k_minus_1 = lambda_k;
154             f_k_minus_1 = f_k;
155             lambda_k = lambda_k_plus_1;
156             f_k = f_k_plus_1;
157         }
158         else
159         {
160             a = lambda_k_minus_1;
161             b = lambda_k_plus_1;
162             if (b < a)
163                 swap(a, b);
164             return;
165         }
166     }
167
168
169
170 // Вычисление nго числа Фибоначчи
171 inline real fib(int n)
172 {
173     real sqrt5 = sqrt(5.0), pow2n = pow(2.0, n);
174     return (pow(1.0 + sqrt5, n) / pow2n - pow(1.0 - sqrt5, n) / pow2n) / sqrt5;
175 }
176
177
178
179 // Определение коэффициента лямбда методом Фибоначчи
180 // f - минимизируемая функция

```

```

181 // x - начальное значение
182 // S - базис
183 // E - точность
184 real fibonacci(const function<real(const vector1D &x)> &f, vector1D &x, vector1D
185   &S, real E)
186 {
187     real a, b;
188     interval(f, a, b, x, S);
189     int iter;
190     real len = fabs(a - b);
191     int n = 0;
192     while (fib(n) < (b - a) / E) n++;
193     iter = n - 3;
194     real lambda1 = a + (fib(n - 2) / fib(n)) * (b - a);
195     real f1 = f(x + S * lambda1);
196     real lambda2 = a + (fib(n - 1) / fib(n)) * (b - a);
197     real f2 = f(x + S * lambda2);
198     for (int k = 0; k < n - 3; k++)
199     {
200         if (f1 <= f2)
201         {
202             b = lambda2;
203             lambda2 = lambda1;
204             f2 = f1;
205             lambda1 = a + (fib(n - k - 3) / fib(n - k - 1)) * (b - a);
206             f1 = f(x + S * lambda1);
207         }
208         else
209         {
210             a = lambda1;
211             lambda1 = lambda2;
212             f1 = f2;
213             lambda2 = a + (fib(n - k - 2) / fib(n - k - 1)) * (b - a);
214             f2 = f(x + S * lambda2);
215         }
216         len = b - a;
217     }
218     lambda2 = lambda1 + E;
219     f2 = f(x + S * lambda2);
220     if (f1 <= f2)
221         b = lambda1;
222     else
223         a = lambda1;
224     return (a + b) / 2.0;
225 }
226
227
228 // Метод Розенброка
229 // f - оптимизируемая функция
230 // x0 - начальное приближение
231 // E - точность
232 // funcname - название функции
233 methodResult calcByRosenbrock(const function<real(const vector1D &x)> &f, const
234   function<real(const vector1D &x)> &G, const vector1D &x0, real r0, real E,
235   const string &funcname, bool isFineNotBarier) {
236
237     ofstream fout("report/tableRosenbrock_" + funcname + ".txt");
238     ofstream steps("steps/Rosenbrock_" + funcname + ".txt");
239     fout << scientific;

```

```

238     steps << fixed << setprecision(12);
239     steps << x0 << endl;
240
241     methodResult result;
242     fCalcCount = 0;
243     gCalcCount = 0;
244     vector1D xPrev, B, x = x0;
245     int maxiter = 1000;
246     int iterationsCount = 0, count = 0;
247     r = r0;
248     real lambda1, lambda2;
249     matrix2D A(2);
250     A[0] = { 1.0, 0.0 };
251     A[1] = { 0.0, 1.0 };
252
253
254 // Начальные ортогональные направления
255 matrix2D S(2);
256 S[0] = { 1.0, 0.0 };
257 S[1] = { 0.0, 1.0 };
258
259 do {
260     xPrev = x;
261
262     // Минимализируем функцию в направлениях S^k_1...S^k_n
263     lambda1 = fibonacci(f, x, S[0], E);
264     x = x + S[0] * lambda1;
265     lambda2 = fibonacci(f, x, S[1], E);
266     x = x + S[1] * lambda2;
267
268     // Построение новых ортогональных направлений при
269     // сортировке лямбд в порядке убывания по абсолютным значениям
270     A[0] = S[0] * lambda1 + S[1] * lambda2;
271     if (fabs(lambda1) >= lambda2)
272         A[1] = S[1] * lambda2;
273     else
274         A[1] = S[0] * lambda1;
275
276
277     // Ортогонализация ГраммШмидта
278     S[0] = A[0] / calcNormE(A[0]);
279     B = A[1] - S[1] * A[1] * S[1];
280     if (calcNormE(B) > E)
281         S[1] = B / calcNormE(B);
282     iterationsCount++;
283
284
285     fout << iterationsCount << "\t"
286     << fCalcCount << "\t"
287     << r << "\t"
288     << x << "\t"
289     << f(x) << endl;
290
291     steps << x << endl;
292
293     if (isFineNotBarier)
294         r *= rMult;
295     else
296         r /= rMult;
297

```

```

298     } while (abs(f(x) - f(xPrev)) > E && abs(calcNormE(x) - calcNormE(xPrev)) >
299             E && iterationsCount < maxiter && r*G(x) > E);
300
301     fout.close();
302     steps.close();
303
304     result.E = E;
305     result.iterationsCount = iterationsCount;
306     result.fCalcCount = fCalcCount;
307     result.x0 = x0;
308     result.x = x;
309     result.fx = f(x);
310     result.r0 = r0;
311
312     return result;
313 }
314
315
316 //////////////////////////////////////////////////////////////////
317 //////////////////////////////////////////////////////////////////
318 // Исследования
319 //////////////////////////////////////////////////////////////////
320 //////////////////////////////////////////////////////////////////
321
322 // Исследование сходимости
323 void researchConvergence()
324 {
325     cout << "Research of convergence" << endl;
326     ofstream fout("steps/Rosenbrock_Q.txt");
327     real E = 1e-12;
328     real r0 = 10;
329     rMult = 2;
330     alpha = 12;
331     methodResult result;
332     vector1D x0Fine = { -2, 0 };
333     vector1D x0Barrier = { 0, -2 };
334     fout << scientific;
335     result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1", true);
336     fout << result.iterationsCount << " " << result.fCalcCount << endl;
337     result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2", true);
338     fout << result.iterationsCount << " " << result.fCalcCount << endl;
339     result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3", true);
340     fout << result.iterationsCount << " " << result.fCalcCount << endl;
341     result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4", false);
342     fout << result.iterationsCount << " " << result.fCalcCount << endl;
343     result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5", false);
344     fout << result.iterationsCount << " " << result.fCalcCount << endl;
345
346     // Визуализация
347     string runVisualisation = "python plot.py "
348         + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
349         + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
350     system(runVisualisation.c_str());
351     fout.close();
352 }
353
354
355
356 // Зависимость скорости сходимости метода от заданной точности

```

```

357 void researchE()
358 {
359     cout << "Research E" << endl;
360     ofstream foutF("steps/tableEFines.txt");
361     ofstream foutB("steps/tableEBarriers.txt");
362     ofstream fout1("report/tableE1.txt");
363     ofstream fout2("report/tableE2.txt");
364     ofstream fout3("report/tableE3.txt");
365     ofstream fout4("report/tableE4.txt");
366     ofstream fout5("report/tableE5.txt");
367     fout1 << scientific << "E\titer\tfCalc\tr\tx\tfx" << endl;
368     fout2 << scientific << "E\titer\tfCalc\tr\tx\tfx" << endl;
369     fout3 << scientific << "E\titer\tfCalc\tr\tx\tfx" << endl;
370     fout4 << scientific << "E\titer\tfCalc\tr\tx\tfx" << endl;
371     fout5 << scientific << "E\titer\tfCalc\tr\tx\tfx" << endl;
372
373     methodResult result;
374     real r0 = 10;
375     rMult = 2;
376     alpha = 8;
377     vector1D x0Fine = { -2, 0 };
378     vector1D x0Barrier = { 0, -2 };
379
380     for (double E = 1e-3; E >= 1e-7; E /= 10)
381     {
382         result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1", true);
383         result.printResultE(fout1);
384         foutF << result.x << endl;
385         result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2", true);
386         result.printResultE(fout2);
387         foutF << result.x << endl;
388         result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3", true);
389         result.printResultE(fout3);
390         foutF << result.x << endl;
391
392         result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4", false);
393         result.printResultE(fout4);
394         foutB << result.x << endl;
395         result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5", false);
396         result.printResultE(fout5);
397         foutB << result.x << endl;
398     }
399
400     // Визуализация
401     string runVisualisation = "python plot.py "
402         + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
403         + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
404     system(runVisualisation.c_str());
405
406     foutF.close();
407     foutB.close();
408     fout1.close();
409     fout2.close();
410     fout3.close();
411     fout4.close();
412     fout5.close();
413 }
414
415
416

```

```

417 // Зависимость скорости сходимости метода от стратегии изменения коэффициента штрафа
418 void researchRMult()
419 {
420     cout << "Research r^mult" << endl;
421     ofstream foutF("steps/tableRMultFines.txt");
422     ofstream foutB("steps/tableRMultBarriers.txt");
423     ofstream fout1("report/tableRMult1.txt");
424     ofstream fout2("report/tableRMult2.txt");
425     ofstream fout3("report/tableRMult3.txt");
426     ofstream fout4("report/tableRMult4.txt");
427     ofstream fout5("report/tableRMult5.txt");
428     fout1 << scientific << "rMult\titer\tfCalc\tr\tx\tfx" << endl;
429     fout2 << scientific << "rMult\titer\tfCalc\tr\tx\tfx" << endl;
430     fout3 << scientific << "rMult\titer\tfCalc\tr\tx\tfx" << endl;
431     fout4 << scientific << "rMult\titer\tfCalc\tr\tx\tfx" << endl;
432     fout5 << scientific << "rMult\titer\tfCalc\tr\tx\tfx" << endl;
433
434     methodResult result;
435     real E = 1e-12;
436     real r0 = 10;
437     alpha = 8;
438     vector1D x0Fine = { -2, 0 };
439     vector1D x0Barrier = { 0, -2 };
440
441     for (size_t i = 2; i <= 10; i++)
442     {
443         rMult = i;
444         result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1", true);
445         result.printResultR(fout1);
446         foutF << result.x << endl;
447         result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2", true);
448         result.printResultR(fout2);
449         foutF << result.x << endl;
450         result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3", true);
451         result.printResultR(fout3);
452         foutF << result.x << endl;
453
454         result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4", false);
455         result.printResultR(fout4);
456         foutB << result.x << endl;
457         result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5", false);
458         result.printResultR(fout5);
459         foutB << result.x << endl;
460     }
461
462     // Визуализация
463     string runVisualisation = "python plot.py "
464         + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
465         + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
466     system(runVisualisation.c_str());
467
468     foutF.close();
469     foutB.close();
470     fout1.close();
471     fout2.close();
472     fout3.close();
473     fout4.close();
474     fout5.close();
475 }
476

```

```

477
478
479 // Зависимость скорости сходимости метода от стратегии изменения коэффициента штрафа
480 void researchRFirst()
481 {
482     cout << "Research r_0" << endl;
483     ofstream foutF("steps/tableRFFirstFines.txt");
484     ofstream foutB("steps/tableRFFirstBarriers.txt");
485     ofstream fout1("report/tableRFFirst1.txt");
486     ofstream fout2("report/tableRFFirst2.txt");
487     ofstream fout3("report/tableRFFirst3.txt");
488     ofstream fout4("report/tableRFFirst4.txt");
489     ofstream fout5("report/tableRFFirst5.txt");
490     fout1 << scientific << "rFirst\titer\tfCalc\tr\tx\tfx" << endl;
491     fout2 << scientific << "rFirst\titer\tfCalc\tr\tx\tfx" << endl;
492     fout3 << scientific << "rFirst\titer\tfCalc\tr\tx\tfx" << endl;
493     fout4 << scientific << "rFirst\titer\tfCalc\tr\tx\tfx" << endl;
494     fout5 << scientific << "rFirst\titer\tfCalc\tr\tx\tfx" << endl;
495
496     methodResult result;
497     real E = 1e-12;
498     rMult = 2;
499     alpha = 8;
500     vector1D x0Fine = { -2, 0 };
501     vector1D x0Barrier = { 0, -2 };
502
503     for (real r0 = 2; r0 <= 64; r0 *= 2)
504     {
505         result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1", true);
506         result.printResultRFFirst(fout1);
507         foutF << result.x << endl;
508         result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2", true);
509         result.printResultRFFirst(fout2);
510         foutF << result.x << endl;
511         result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3", true);
512         result.printResultRFFirst(fout3);
513         foutF << result.x << endl;
514
515         result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4", false);
516         result.printResultRFFirst(fout4);
517         foutB << result.x << endl;
518         result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5", false);
519         result.printResultRFFirst(fout5);
520         foutB << result.x << endl;
521     }
522
523     // Визуализация
524     string runVisualisation = "python plot.py "
525         + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
526         + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
527     system(runVisualisation.c_str());
528
529     foutF.close();
530     foutB.close();
531     fout1.close();
532     fout2.close();
533     fout3.close();
534     fout4.close();
535     fout5.close();
536 }

```

```

537
538
539
540 // Зависимость скорости сходимости метода от начального приближения
541 void researchXFirst()
542 {
543     cout << "Research x_0" << endl;
544     methodResult result;
545     real E = 1e-12;
546     real r0 = 10;
547     rMult = 2;
548     alpha = 8;
549     vector1D x0Fine = { 2, -2 };
550     vector1D x0Barrier = { 0, -2 };

551
552     ofstream fout1("steps/Rosenbrock_Q_1.txt");
553     result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1_1", true);
554     fout1 << result.iterationsCount << " " << result.fCalcCount << endl;
555     result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2_1", true);
556     fout1 << result.iterationsCount << " " << result.fCalcCount << endl;
557     result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3_1", true);
558     fout1 << result.iterationsCount << " " << result.fCalcCount << endl;
559     result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4_1", false);
560     fout1 << result.iterationsCount << " " << result.fCalcCount << endl;
561     result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5_1", false);
562     fout1 << result.iterationsCount << " " << result.fCalcCount << endl;

563
564 // Визуализация
565 string runVisualisation = "python plot.py "
566     + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
567     + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
568 system(runVisualisation.c_str());
569 fout1.close();

570
571
572     ofstream fout2("steps/Rosenbrock_Q_2.txt");
573     x0Fine = { -2, -2 };
574     x0Barrier = { -2, -2 };
575     result = calcByRosenbrock(Q1, G1, x0Fine, r0, E, "Q1_2", true);
576     fout2 << result.iterationsCount << " " << result.fCalcCount << endl;
577     result = calcByRosenbrock(Q2, G2, x0Fine, r0, E, "Q2_2", true);
578     fout2 << result.iterationsCount << " " << result.fCalcCount << endl;
579     result = calcByRosenbrock(Q3, G3, x0Fine, r0, E, "Q3_2", true);
580     fout2 << result.iterationsCount << " " << result.fCalcCount << endl;
581     result = calcByRosenbrock(Q4, G4, x0Barrier, r0, E, "Q4_2", false);
582     fout2 << result.iterationsCount << " " << result.fCalcCount << endl;
583     result = calcByRosenbrock(Q5, G5, x0Barrier, r0, E, "Q5_2", false);
584     fout2 << result.iterationsCount << " " << result.fCalcCount << endl;

585
586 // Визуализация
587 runVisualisation = "python plot.py "
588     + to_string(x0Fine[0]) + " " + to_string(x0Fine[1]) + " "
589     + to_string(x0Barrier[0]) + " " + to_string(x0Barrier[1]);
590 system(runVisualisation.c_str());
591 fout2.close();
592 }
593
594
595
596 void main()

```

```

597 {
598     researchConvergence();
599     researchE();
600     researchRFirst();
601     researchRMult();
602     researchXFirst();
603 }

```

plot.py

```

1 import pylab
2 import numpy
3 import sys
4 import matplotlib.pyplot as plt
5 import matplotlib.lines as lines
6
7 DPI = 200
8
9 xList = []
10 yList = []
11 data = []
12 text = []
13
14 folder = 'pics/'
15 steps = 'steps/'
16
17 # Считывание итогов работы метода
18 def inputMethodResults(filename):
19     global text
20     global data
21     text = []
22     data = []
23     with open(steps + filename, 'r') as f:
24         for line in f: # read rest of lines
25             data.append([ int(x) for x in line.split()])
26     for i in range(len(data)):
27         text.append('iter:' +str(data[i][0])+'\nfcalc: '+str(data[i][1]))
28
29
30 # Считывание хода метода
31 def inputSteps(filename):
32     global xList
33     global yList
34     global data
35     xList = []
36     yList = []
37     data = []
38     with open(steps + filename, 'r') as f:
39         for line in f: # read rest of lines
40             data.append([ float(x) for x in line.split()])
41     for i in range(len(data)):
42         xList.append(data[i][0])
43         yList.append(data[i][1])
44
45
46 # Целевая функция
47 def f(x, y):
48     return 4*(y-x)**2 + 3*(x-1)**2
49
50
51 # Рассчёт значений целевой функции на сетке
52 def buildIsoLines(f):

```

```

53     x = numpy.linspace (-3, 3, 100)
54     y = numpy.linspace (-4, 3, 100)
55     xgrid, ygrid = numpy.meshgrid(x, y)
56     zgrid = f(xgrid, ygrid)
57     return xgrid, ygrid, zgrid
58
59
60 # Закраска областей, где накладывается штраф или барьер уходит в бесконечность
61 def drawFines():
62     x1 = numpy.linspace(-3, 3, 100)
63     y1 = -1 - x1
64     yBorder = 3
65     plt.fill_between(x1,y1,yBorder, color='grey')
66     x2 = numpy.linspace(-3, 2, 100)
67     y2 = x2 + 1
68     plt.plot(x2, y2)
69
70
71 # Отрисовка сходимости метода
72 def drawMethodConvergence(name, f, index, x0, y0, xExp, yExp):
73     global text
74     inputSteps(name + '.txt')
75     drawFines()
76     x, y, z = buildIsoLines(f)
77     cs = pylab.contour(x, y, z, 25)
78     plt.plot(xList, yList, linewidth=1)
79     for i in range(len(xList)):
80         plt.scatter(xList[i], yList[i], s=2, color='black')
81     plt.title(name, fontsize=19)
82     plt.xlabel('X', fontsize=10)
83     plt.ylabel('Y', fontsize=10)
84     plt.tick_params(axis='both', labelsize=8)
85     plt.scatter(x0, y0, s=20)
86     plt.scatter(xExp, yExp, s=20)
87     plt.text(2, 2, text[index], size=12,
88             ha="center", va="center",
89             bbox=dict(boxstyle="square",
90                     ec=(.5, .5, .5),
91                     fc=(.8, .8, 0.8),
92                     ))
93     )
94     plt.savefig(folder + name + '.png', dpi=DPI)
95     plt.clf()
96
97
98 # Отрисовка точек в исследовании
99 def drawPointsAtResearch(name, f):
100     inputSteps(name + '.txt')
101     drawFines()
102     x, y, z = buildIsoLines(f)
103     cs = pylab.contour(x, y, z, 25)
104     plt.title(name, fontsize=19)
105     plt.xlabel('X', fontsize=10)
106     plt.ylabel('Y', fontsize=10)
107     plt.tick_params(axis='both', labelsize=8)
108     for i in range(len(xList)):
109         plt.scatter(xList[i], yList[i], s=5)
110     plt.savefig(folder + name + '.png', dpi=DPI)
111     plt.clf()
112

```

```

113
114
115 if __name__ == '__main__':
116     # парсим начальное приближение
117     params = sys.argv[1:5]
118     x0Fine = float(params[0])
119     y0Fine = float(params[1])
120     x0Barrier = float(params[2])
121     y0Barrier = float(params[3])
122     xExp = 1
123     yExp = 1
124
125     inputMethodResults('Rosenbrock_Q.txt')
126     drawMethodConvergence('Rosenbrock_Q1', f, 0, x0Fine, y0Fine, xExp, yExp)
127     drawMethodConvergence('Rosenbrock_Q2', f, 1, x0Fine, y0Fine, xExp, yExp)
128     drawMethodConvergence('Rosenbrock_Q3', f, 2, x0Fine, y0Fine, xExp, yExp)
129     drawMethodConvergence('Rosenbrock_Q4', f, 3, x0Barrier, y0Barrier, xExp,
yExp)
130     drawMethodConvergence('Rosenbrock_Q5', f, 4, x0Barrier, y0Barrier, xExp,
yExp)
131
132
133     inputMethodResults('Rosenbrock_Q_1.txt')
134     drawMethodConvergence('Rosenbrock_Q1_1', f, 0, x0Fine, y0Fine, xExp, yExp)
135     drawMethodConvergence('Rosenbrock_Q2_1', f, 1, x0Fine, y0Fine, xExp, yExp)
136     drawMethodConvergence('Rosenbrock_Q3_1', f, 2, x0Fine, y0Fine, xExp, yExp)
137     drawMethodConvergence('Rosenbrock_Q4_1', f, 3, x0Barrier, y0Barrier, xExp,
yExp)
138     drawMethodConvergence('Rosenbrock_Q5_1', f, 4, x0Barrier, y0Barrier, xExp,
yExp)
139
140
141     inputMethodResults('Rosenbrock_Q_2.txt')
142     drawMethodConvergence('Rosenbrock_Q1_2', f, 0, x0Fine, y0Fine, xExp, yExp)
143     drawMethodConvergence('Rosenbrock_Q2_2', f, 1, x0Fine, y0Fine, xExp, yExp)
144     drawMethodConvergence('Rosenbrock_Q3_2', f, 2, x0Fine, y0Fine, xExp, yExp)
145     drawMethodConvergence('Rosenbrock_Q4_2', f, 3, x0Barrier, y0Barrier, xExp,
yExp)
146     drawMethodConvergence('Rosenbrock_Q5_2', f, 4, x0Barrier, y0Barrier, xExp,
yExp)
147
148
149
150     drawPointsAtResearch('tableEFines', f)
151     drawPointsAtResearch('tableEBarriers', f)
152
153     drawPointsAtResearch('tableRMultFines', f)
154     drawPointsAtResearch('tableRMultBarriers', f)
155
156     drawPointsAtResearch('tableRFFirstFines', f)
157     drawPointsAtResearch('tableRFFirstBarriers', f)

```