Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра ТПИ

МЕТОДЫ ОПТИМИЗАЦИИ

Лабораторная работа № 3

Решение нелинейных начально-краевых задач

Факультет: ПМИ

Преподаватели:
Лемешко Борис Юрьевич
Чимитова Екатерина Владимировна

Группа: ПМ-81

Студенты: Ефремов А.
Ртищева К.
Бортникова А.

Бригада: 2

Новосибирск
2021

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 задача а)** с использованием **метода барьерных функций**.

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

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

Сформулировать выводы.

Первая задача (а)	Вторая задача (б)
$f(x,y) = 10(y-x)^2 + y^2 \to min$ $x + y \geq 1$	$f(x,y) = 10(y-x)^2 + y^2 \to min$ $x = 2 - y$

3. Таблицы с исследованиями и выводы для метода штрафных функций

3.1. Для функции G

$$funct_0 = G\big(g(\bar{x})\big) = 0.5(g(\bar{x}) + |g(\bar{x})|)$$

$$funct_1 = G\big(g(\bar{x})\big) = \big(0.5(g(\bar{x}) + |g(\bar{x})|)\big)^2$$

$$funct_2 = G\big(g(\bar{x})\big) = \big(0.5(g(\bar{x}) + |g(\bar{x})|)\big)^4$$

$$funct_3 = G\big(g(\bar{x})\big) = \frac{1}{g(x)}$$

$$rg_k^1 = rg_{k-1}^1 + 1$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	1	1	-5	2	1	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	1	2	1670828	0.473684	0.451128	0.208604	4.139578e-02
2	1	1	-5	2	1	2	842985	0.356169	0.339209	0.117939	1.320610e-01
3	1	1	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1	1	-5	2	0.1	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	0.1	2	1670828	0.473684	0.451128	0.208604	4.139578e-02
2	1	1	-5	2	0.1	2	842985	0.356169	0.339209	0.117939	1.320610e-01
3	1	1	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1	1	-5	2	0.01	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	0.01	3	2507068	0.482759	0.45977	0.216673	3.332673e-02
2	1	1	-5	2	0.01	3	1677983	0.368782	0.351221	0.12644	1.235599e-01
3	1	1	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1	1	-5	2	0.001	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	0.001	11	6695051	0.501992	0.478088	0.234282	1.571800e-02
2	1	1	-5	2	0.001	21	12575104	0.426888	0.40656	0.169424	8.057632e-02
3	1	1	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1	1	-5	2	0.0001	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	0.0001	34	15083201	0.508651	0.484429	0.240538	9.461692e-03
2	1	1	-5	2	0.0001	100	53652383	0.459589	0.437704	0.196374	5.362564e-02
3	1	1	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1	1	-5	2	1e-05	2	6162	0.521216	0.478784	0.247239	2.761235e-03
1	1	1	-5	2	1e-05	100	36226456	0.510961	0.48663	0.242729	7.271296e-03
2	1	1	-5	2	1e-05	100	53652383	0.459589	0.437704	0.196374	5.362564e-02
3	1	1	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03

$$rg_k^2 = rg_{k-1}^2 + 100$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	1	2	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	1	2	1669949	0.511574	0.487214	0.243312	6.688410e-03
2	1	2	-5	2	1	2	838089	0.470057	0.447674	0.205422	4.457791e-02
3	1	2	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761236e-03
0	1	2	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	0.1	2	1669949	0.511574	0.487214	0.243312	6.688410e-03
2	1	2	-5	2	0.1	2	838089	0.470057	0.447674	0.205422	4.457791e-02
3	1	2	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761236e-03
0	1	2	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	0.01	2	1669949	0.511574	0.487214	0.243312	6.688410e-03
2	1	2	-5	2	0.01	2	838089	0.470057	0.447674	0.205422	4.457791e-02
3	1	2	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761236e-03
0	1	2	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	0.001	2	1669949	0.511574	0.487214	0.243312	6.688410e-03
2	1	2	-5	2	0.001	9	4185674	0.486347	0.463188	0.219906	3.009351e-02
3	1	2	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761236e-03
0	1	2	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	0.0001	4	2524139	0.511884	0.487508	0.243606	6.393991e-03
2	1	2	-5	2	0.0001	47	22642202	0.497182	0.473507	0.229814	2.018590e-02
3	1	2	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761236e-03
0	1	2	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1	2	-5	2	1e-05	12	3738521	0.512091	0.487706	0.243803	6.196596e-03
2	1	2	-5	2	1e-05	100	46145565	0.500496	0.476663	0.232888	1.711196e-02
3	1	2	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761236e-03

$$rg_k^3 = 2rg_{k-1}^3$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	3	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	1	2	1673725	0.497041	0.473373	0.229684	2.031616e-02
2	2	3	-5	2	1	2	1671860	0.395667	0.376825	0.145547	1.044527e-01
3	2	3	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	3	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	0.1	2	1673725	0.497041	0.473373	0.229684	2.031616e-02
2	2	3	-5	2	0.1	2	1671860	0.395667	0.376825	0.145547	1.044527e-01
3	2	3	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	3	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	0.01	3	2506227	0.504505	0.48048	0.236633	1.336697e-02
2	2	3	-5	2	0.01	7	4188391	0.473234	0.450699	0.208208	4.179209e-02
3	2	3	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	3	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	0.001	6	2531059	0.511221	0.486877	0.242976	7.024417e-03
2	2	3	-5	2	0.001	17	10852098	0.508237	0.484035	0.240147	9.852967e-03
3	2	3	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	3	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	0.0001	10	3577326	0.512134	0.487747	0.243844	6.155645e-03
2	2	3	-5	2	0.0001	27	14397128	0.511801	0.48743	0.243528	6.472318e-03
3	2	3	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	3	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	3	-5	2	1e-05	13	5091141	0.512188	0.487798	0.243895	6.104823e-03
2	2	3	-5	2	1e-05	37	17297580	0.512156	0.487768	0.243865	6.134763e-03
3	2	3	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03

$$rg_k^4 = 200rg_{k-1}^4$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	4	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	1	2	1664950	0.512895	0.487103	0.243921	6.078814e-03
2	2	4	-5	2	1	2	833514	0.506323	0.482212	0.238342	1.165804e-02
3	2	4	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	0.1	2	1664950	0.512895	0.487103	0.243921	6.078814e-03
2	2	4	-5	2	0.1	2	833514	0.506323	0.482212	0.238342	1.165804e-02
3	2	4	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	0.01	2	1664950	0.512895	0.487103	0.243921	6.078814e-03
2	2	4	-5	2	0.01	3	859254	0.511188	0.486846	0.242944	7.055981e-03
3	2	4	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	0.001	2	1664950	0.512895	0.487103	0.243921	6.078814e-03
2	2	4	-5	2	0.001	4	989242	0.512023	0.487641	0.243738	6.261671e-03
3	2	4	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	0.0001	4	3324970	0.521215	0.478785	0.247238	2.761661e-03
2	2	4	-5	2	0.0001	6	2490068	0.512727	0.487263	0.24391	6.090450e-03
3	2	4	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	1e-05	5	4154980	0.521216	0.478784	0.247239	2.761237e-03
2	2	4	-5	2	1e-05	12	7470128	0.521215	0.478785	0.247238	2.761776e-03
3	2	4	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	1e-06	5	4154980	0.521216	0.478784	0.247239	2.761237e-03
2	2	4	-5	2	1e-06	13	8300138	0.521216	0.478784	0.247239	2.761328e-03
3	2	4	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	4	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	4	-5	2	1e-07	6	4984990	0.521216	0.478784	0.247239	2.761235e-03
2	2	4	-5	2	1e-07	14	9130148	0.521216	0.478784	0.247239	2.761251e-03
3	2	4	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03

$$rg_k^5 = \left(rg_{k-1}^5\right)^2$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	5	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1	2	1668742	0.504505	0.48048	0.236633	1.336697e-02
2	2	5	-5	2	1	2	1670988	0.417998	0.398093	0.16244	8.755965e-02
3	2	5	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	0.1	2	1668742	0.504505	0.48048	0.236633	1.336697e-02
2	2	5	-5	2	0.1	2	1670988	0.417998	0.398093	0.16244	8.755965e-02
3	2	5	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	0.01	3	1688747	0.511708	0.487341	0.243438	6.561651e-03
2	2	5	-5	2	0.01	5	3473667	0.512039	0.487656	0.243754	6.246341e-03
3	2	5	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	0.001	4	2518757	0.512592	0.487404	0.243907	6.092792e-03
2	2	5	-5	2	0.001	7	5133687	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	0.0001	6	3351599	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	0.0001	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1e-05	6	3351599	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	1e-05	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1e-06	6	3351599	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	1e-06	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1e-07	7	3354431	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	1e-07	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	1e-08	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1e-08	7	3354431	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	1e-08	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	1e-08	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	5	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	5	-5	2	1e-09	7	3354431	0.521216	0.478784	0.247239	2.761235e-03
2	2	5	-5	2	1e-09	8	5136519	0.521216	0.478784	0.247239	2.761235e-03
3	2	5	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03

$$rg_k^6 = (rg_{k-1}^6)^3$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	6	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1	2	1672416	0.511951	0.487573	0.24367	6.329772e-03
2	2	6	-5	2	1	2	1672353	0.481101	0.458192	0.215188	3.481189e-02
3	2	6	-5	2	1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	0.1	2	1672416	0.511951	0.487573	0.24367	6.329772e-03
2	2	6	-5	2	0.1	2	1672353	0.481101	0.458192	0.215188	3.481189e-02
3	2	6	-5	2	0.1	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	0.01	3	2502426	0.521202	0.478798	0.247229	2.771381e-03
2	2	6	-5	2	0.01	4	2552007	0.521205	0.478795	0.247231	2.769345e-03
3	2	6	-5	2	0.01	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	0.001	4	2505258	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	0.001	5	2554839	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	0.001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	0.0001	4	2505258	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	0.0001	5	2554839	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	0.0001	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-05	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-05	5	2554839	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-05	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-06	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-06	6	2557671	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-06	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-07	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-07	6	2557671	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-07	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-08	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-08	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-08	6	2557671	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-08	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-09	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-09	6	2557671	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03

Лучше всего себя показали функции $funct_0 = G\big(g(\bar{x})\big) = 0.5(g(\bar{x}) + |g(\bar{x})|)$, $funct_3 = G\big(g(\bar{x})\big) = \frac{1}{g(x)}$ – они показывают примерно одинаковую и лучшую среди всех функций точность и скорость схождения. Предпочтение стоить отдать $funct_0$, так как в $funct_3$ присутствует вычислительно затратная операция деления.

Лучше всего себя показала стратегия изменения коэффициента штрафа $rg_k^6 = (rg_{k-1}^6)^3$- как минимум с точки зрения количества итераций метода штрафных функций.

Проведем дополнительные исследования с помощью $rg_k^6 = (rg_{k-1}^6)^3$ с варьированием начального коэффициента штрафа.

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	1	6	-5	2	1e-09	2	7324	0.521216	0.478784	0.247239	2.761235e-03
1	1	6	-5	2	1e-09	2	1673276	0.411765	0.392157	0.157632	9.236832e-02
2	1	6	-5	2	1e-09	2	21698	0.29974	0.285467	0.0835285	1.664715e-01
3	1	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	1.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	1.5	6	-5	2	1e-09	5	3329857	0.521216	0.478784	0.247239	2.761235e-03
2	1.5	6	-5	2	1e-09	6	4161196	0.521216	0.478784	0.247239	2.761235e-03
3	1.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	2	6	-5	2	1e-09	5	2508090	0.521216	0.478784	0.247239	2.761235e-03
2	2	6	-5	2	1e-09	6	2557671	0.521216	0.478784	0.247239	2.761235e-03
3	2	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	2.5	6	-5	2	1e-09	4	11328	0.521216	0.478784	0.247239	2.761235e-03
1	2.5	6	-5	2	1e-09	5	1883660	0.521216	0.478784	0.247239	2.761235e-03
2	2.5	6	-5	2	1e-09	6	2505352	0.521216	0.478784	0.247239	2.761235e-03
3	2.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	3	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	3	6	-5	2	1e-09	4	1668843	0.521216	0.478784	0.247239	2.761235e-03
2	3	6	-5	2	1e-09	5	3327864	0.521216	0.478784	0.247239	2.761235e-03
3	3	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	3.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	3.5	6	-5	2	1e-09	4	2495355	0.521216	0.478784	0.247239	2.761235e-03
2	3.5	6	-5	2	1e-09	5	1674506	0.521216	0.478784	0.247239	2.761235e-03
3	3.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	4	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	4	6	-5	2	1e-09	4	844503	0.521216	0.478784	0.247239	2.761235e-03
2	4	6	-5	2	1e-09	5	1669855	0.521216	0.478784	0.247239	2.761235e-03
3	4	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	4.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	4.5	6	-5	2	1e-09	4	1668176	0.521216	0.478784	0.247239	2.761235e-03
2	4.5	6	-5	2	1e-09	5	2505653	0.521216	0.478784	0.247239	2.761235e-03
3	4.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	5	6	-5	2	1e-09	4	847338	0.521216	0.478784	0.247239	2.761235e-03
2	5	6	-5	2	1e-09	5	1680140	0.521216	0.478784	0.247239	2.761235e-03
3	5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
0	5.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03
1	5.5	6	-5	2	1e-09	4	848997	0.521216	0.478784	0.247239	2.761235e-03
2	5.5	6	-5	2	1e-09	5	1669356	0.521216	0.478784	0.247239	2.761235e-03
3	5.5	6	-5	2	1e-09	2	5664	0.521216	0.478784	0.247239	2.761235e-03

Вывод: для данной задачи лучшей функцией является $funct_0 = G\big(g(\bar{x})\big) = 0.5(g(\bar{x}) + |g(\bar{x})|)$, оптимальной стратегией выбора коэффициента штрафа является степенная функция $rg_k^6 = (rg_{k-1}^6)^3$, оптимальный начальный коэффициент штрафа для этой функции выбирается достаточно свободно в диапазоне от 1.5.

3.2. Для функции H

$$funct_0 = H\big(h(\bar{x})\big) = |h(\bar{x})|$$
$$funct_1 = H\big(h(\bar{x})\big) = h(\bar{x})^2$$
$$funct_2 = H\big(h(\bar{x})\big) = h(\bar{x})^4$$
$$rg_k^1 = rg_{k-1}^1 + 1$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	1	1	-5	2	1	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	1	2	1560000	0.947368	0.902256	0.834417	1.655831e-01
2	1	1	-5	2	1	2	7020	0.818493	0.779517	0.622838	3.771617e-01
0	1	1	-5	2	0.1	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	0.1	2	1560000	0.947368	0.902256	0.834417	1.655831e-01
2	1	1	-5	2	0.1	2	7020	0.818493	0.779517	0.622838	3.771617e-01
0	1	1	-5	2	0.01	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	0.01	7	3903588	0.994083	0.946746	0.918735	8.126466e-02
2	1	1	-5	2	0.01	8	3131700	0.878237	0.836416	0.717082	2.829183e-01
0	1	1	-5	2	0.001	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	0.001	27	6251700	1	0.974359	0.95595	4.404997e-02
2	1	1	-5	2	0.001	48	18767441	0.939425	0.894691	0.820483	1.795171e-01
0	1	1	-5	2	0.0001	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	0.0001	100	6285933	1	0.991071	0.98302	1.698023e-02
2	1	1	-5	2	0.0001	100	36744777	0.957209	0.911628	0.851842	1.481583e-01
0	1	1	-5	2	1e-05	2	624	1	1	1	1.776357e-15
1	1	1	-5	2	1e-05	100	6285933	1	0.991071	0.98302	1.698023e-02
2	1	1	-5	2	1e-05	100	36744777	0.957209	0.911628	0.851842	1.481583e-01

$$rg_k^2 = rg_{k-1}^2 + 100$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	1	2	-5	2	1	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	1	2	938	1	0.995283	0.990811	9.189214e-03
2	1	2	-5	2	1	2	780938	0.97073	0.924505	0.876077	1.239233e-01
0	1	2	-5	2	0.1	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	0.1	2	938	1	0.995283	0.990811	9.189214e-03
2	1	2	-5	2	0.1	2	780938	0.97073	0.924505	0.876077	1.239233e-01
0	1	2	-5	2	0.01	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	0.01	2	938	1	0.995283	0.990811	9.189214e-03
2	1	2	-5	2	0.01	4	784528	0.981606	0.934863	0.895818	1.041821e-01
0	1	2	-5	2	0.001	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	0.001	5	2345	1	0.998047	0.996136	3.864289e-03
2	1	2	-5	2	0.001	17	6259049	0.997816	0.950301	0.925648	7.435207e-02
0	1	2	-5	2	0.0001	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	0.0001	15	7035	1	0.999339	0.998682	1.317939e-03
2	1	2	-5	2	0.0001	100	14876416	1	0.968103	0.947397	5.260269e-02
0	1	2	-5	2	1e-05	2	938	1	1	1	1.776357e-15
1	1	2	-5	2	1e-05	46	21574	1	0.999783	0.999567	4.331345e-04
2	1	2	-5	2	1e-05	100	14876416	1	0.968103	0.947397	5.260269e-02

$$rg_k^3 = 2rg_{k-1}^3$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	3	-5	2	1	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	1	2	781872	0.994083	0.946746	0.918735	8.126466e-02
2	2	3	-5	2	1	2	782652	0.872705	0.831147	0.708076	2.919243e-01
0	2	3	-5	2	0.1	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	0.1	2	781872	0.994083	0.946746	0.918735	8.126466e-02
2	2	3	-5	2	0.1	2	782652	0.872705	0.831147	0.708076	2.919243e-01
0	2	3	-5	2	0.01	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	0.01	7	784216	1	0.996255	0.992664	7.336334e-03
2	2	3	-5	2	0.01	11	3130616	1	0.9593	0.936821	6.317866e-02
0	2	3	-5	2	0.001	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	0.001	10	785623	1	0.999514	0.999031	9.687507e-04
2	2	3	-5	2	0.001	25	3137182	1	0.998061	0.996163	3.836913e-03
0	2	3	-5	2	0.0001	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	0.0001	14	787499	1	0.999969	0.999939	6.100437e-05
2	2	3	-5	2	0.0001	36	3142341	1	0.999846	0.999693	3.071644e-04
0	2	3	-5	2	1e-05	2	624	1	1	1	1.776357e-15
1	2	3	-5	2	1e-05	17	788906	1	0.999996	0.999992	7.628869e-06
2	2	3	-5	2	1e-05	45	3146562	1	0.999981	0.999962	3.844297e-05

$$rg_k^4 = 200rg_{k-1}^4$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	4	-5	2	1	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	1	2	938	1	0.999988	0.999975	2.499488e-05
2	2	4	-5	2	1	2	3434	1	0.982822	0.96889	3.111019e-02
0	2	4	-5	2	0.1	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	0.1	2	938	1	0.999988	0.999975	2.499488e-05
2	2	4	-5	2	0.1	2	3434	1	0.982822	0.96889	3.111019e-02
0	2	4	-5	2	0.01	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	0.01	2	938	1	0.999988	0.999975	2.499488e-05
2	2	4	-5	2	0.01	4	4372	1	0.999462	0.998928	1.071912e-03
0	2	4	-5	2	0.001	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	0.001	3	1407	1	1	1	1.249944e-07
2	2	4	-5	2	0.001	5	4841	1	0.999908	0.999816	1.840461e-04
0	2	4	-5	2	0.0001	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	0.0001	3	1407	1	1	1	1.249944e-07
2	2	4	-5	2	0.0001	7	5779	1	0.999997	0.999995	5.385954e-06
0	2	4	-5	2	1e-05	2	938	1	1	1	1.776357e-15
1	2	4	-5	2	1e-05	4	1876	1	1	1	6.248566e-10
2	2	4	-5	2	1e-05	8	6248	1	1	0.999999	9.209925e-07

$$rg_k^5 = \left(rg_{k-1}^5\right)^2$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	5	-5	2	1	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1	2	2340	1	0.962963	0.941015	5.898491e-02
2	2	5	-5	2	1	2	4056	0.902636	0.859654	0.757479	2.425207e-01
0	2	5	-5	2	0.1	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.1	2	2340	1	0.962963	0.941015	5.898491e-02
2	2	5	-5	2	0.1	4	6086	1	0.981732	0.967135	3.286492e-02
0	2	5	-5	2	0.01	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.01	4	3278	1	0.999985	0.999969	3.050995e-05
2	2	5	-5	2	0.01	6	7024	1	1	0.999999	6.007758e-07
0	2	5	-5	2	0.001	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.001	5	3747	1	1	1	4.655440e-10
2	2	5	-5	2	0.001	6	7024	1	1	0.999999	6.007758e-07
0	2	5	-5	2	0.0001	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.0001	5	3747	1	1	1	4.655440e-10
2	2	5	-5	2	0.0001	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-05	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-05	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-05	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-06	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-06	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-06	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-07	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-07	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-07	9	8137	-2.81e-16	2	44	4.300000e+01
0	2	5	-5	2	1e-08	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-08	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-08	9	8137	-2.81e-16	2	44	4.300000e+01
0	2	5	-5	2	1e-09	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-09	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-09	9	8137	-2.81e-16	2	44	4.300000e+01

$$rg_k^6 = (rg_{k-1}^6)^3$$

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	5	-5	2	1	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1	2	2340	1	0.962963	0.941015	5.898491e-02
2	2	5	-5	2	1	2	4056	0.902636	0.859654	0.757479	2.425207e-01
0	2	5	-5	2	0.1	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.1	2	2340	1	0.962963	0.941015	5.898491e-02
2	2	5	-5	2	0.1	4	6086	1	0.981732	0.967135	3.286492e-02
0	2	5	-5	2	0.01	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.01	4	3278	1	0.999985	0.999969	3.050995e-05
2	2	5	-5	2	0.01	6	7024	1	1	0.999999	6.007758e-07
0	2	5	-5	2	0.001	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.001	5	3747	1	1	1	4.655440e-10
2	2	5	-5	2	0.001	6	7024	1	1	0.999999	6.007758e-07
0	2	5	-5	2	0.0001	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	0.0001	5	3747	1	1	1	4.655440e-10
2	2	5	-5	2	0.0001	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-05	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-05	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-05	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-06	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-06	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-06	7	7493	1	1	1	2.289280e-13
0	2	5	-5	2	1e-07	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-07	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-07	9	8137	-2.81e-16	2	44	4.300000e+01
0	2	5	-5	2	1e-08	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-08	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-08	9	8137	-2.81e-16	2	44	4.300000e+01
0	2	5	-5	2	1e-09	2	624	1	1	1	1.776357e-15
1	2	5	-5	2	1e-09	7	4391	-2.81e-16	2	44	4.300000e+01
2	2	5	-5	2	1e-09	9	8137	-2.81e-16	2	44	4.300000e+01

Лучше всего себя показала функция $funct_0 = H\big(h(\bar{x})\big) = |h(\bar{x})|$ - она ищет правильный экстремум за минимальное и константное число итераций, причем с максимальной точностью. Также стоит отметить, что функция $funct_1 = H\big(h(\bar{x})\big) = h(\bar{x})^2$ также показывает хороший результат, точность которого можно контролировать параметром **eps**.

Лучше всего себя показала стратегия изменения коэффициента штрафа $rg_k^4 = 200rg_{k-1}^4$. Использование степенных функций нежелательно, так как приводит к нахождению ложной точки минимума.

Проведем дополнительные исследования с помощью $rg_k^4 = 200rg_{k-1}^4$ с варьированием начального коэффициента штрафа.

funct	r0	rh_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	140	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	140	4	-5	2	1e-09	5	2345	1	1	1	4.352074e-14
2	140	4	-5	2	1e-09	12	5628	1	1	1	1.910152e-10
0	150	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	150	4	-5	2	1e-09	5	2345	1	1	1	4.063416e-14
2	150	4	-5	2	1e-09	12	5628	1	1	1	1.865716e-10
0	160	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	160	4	-5	2	1e-09	5	2345	1	1	1	3.352874e-14
2	160	4	-5	2	1e-09	12	5628	1	1	1	1.826532e-10
0	170	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	170	4	-5	2	1e-09	5	2345	1	1	1	3.352874e-14
2	170	4	-5	2	1e-09	12	5628	1	1	1	1.790303e-10
0	180	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	180	4	-5	2	1e-09	5	2345	1	1	1	3.352874e-14
2	180	4	-5	2	1e-09	12	5628	1	1	1	1.756799e-10
0	190	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	190	4	-5	2	1e-09	5	2345	1	1	1	3.108624e-14
2	190	4	-5	2	1e-09	12	5628	1	1	1	1.725058e-10
0	200	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	200	4	-5	2	1e-09	5	2345	1	1	1	3.108624e-14
2	200	4	-5	2	1e-09	12	5628	1	1	1	1.697658e-10
0	210	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	210	4	-5	2	1e-09	5	2345	1	1	1	2.819966e-14
2	210	4	-5	2	1e-09	12	5628	1	1	1	1.669287e-10
0	220	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	220	4	-5	2	1e-09	5	2345	1	1	1	2.819966e-14
2	220	4	-5	2	1e-09	12	5628	1	1	1	1.642926e-10
0	230	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	230	4	-5	2	1e-09	5	2345	1	1	1	2.819966e-14
2	230	4	-5	2	1e-09	12	5628	1	1	1	1.618972e-10
0	240	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	240	4	-5	2	1e-09	5	2345	1	1	1	2.287059e-14
2	240	4	-5	2	1e-09	12	5628	1	1	1	1.596183e-10
0	250	4	-5	2	1e-09	2	938	1	1	1	1.776357e-15
1	250	4	-5	2	1e-09	4	1876	1	1	1	4.978240e-12
2	250	4	-5	2	1e-09	12	5628	1	1	1	1.574634e-10

Вывод: для данной задачи лучшими функциями являются $funct_0 = H\big(h(\bar{x})\big) = |h(\bar{x})|$ и $funct_1 = H\big(h(\bar{x})\big) = h(\bar{x})^2$, оптимальной стратегией выбора коэффициента штрафа является мультипликативная функция $rg_k^4 = 200rg_{k-1}^4$, оптимальный начальный коэффициент штрафа для этой функции примерно равен **240**.

4. Таблицы с исследованиями и выводы для метода барьерных функций

$$funct_0 = -\frac{1}{g(x)}$$

$$funct_1 = -\ln(-g(x))$$

$$rg_k^1 = rg_{k-1}^1 - 0.01$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	1	-0.5	0.2	1	2	1667365	1.189	1.133	1.315	1.0652e+00
1	2	1	-0.5	0.2	1	2	1660010	1.319	1.256	1.618	1.3682e+00
0	2	1	-0.5	0.2	0.1	2	1667365	1.189	1.133	1.315	1.0652e+00
1	2	1	-0.5	0.2	0.1	2	1660010	1.319	1.256	1.618	1.3682e+00
0	2	1	-0.5	0.2	0.01	2	1667365	1.189	1.133	1.315	1.0652e+00
1	2	1	-0.5	0.2	0.01	2	1660010	1.319	1.256	1.618	1.3682e+00
0	2	1	-0.5	0.2	0.001	58	30175941	0.8	0.2	3.64	3.3900e+00
1	2	1	-0.5	0.2	0.001	100	48664665	1.033	0.9837	0.9918	7.4185e-01
0	2	1	-0.5	0.2	0.0001	58	30175941	0.8	0.2	3.64	3.3900e+00
1	2	1	-0.5	0.2	0.0001	100	48664665	1.033	0.9837	0.9918	7.4185e-01
0	2	1	-0.5	0.2	1e-05	58	30175941	0.8	0.2	3.64	3.3900e+00
1	2	1	-0.5	0.2	1e-05	100	48664665	1.033	0.9837	0.9918	7.4185e-01

$$rg_k^2 = \frac{rg_{k-1}^2}{2}$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	2	-0.5	0.2	1	5	1720160	0.1026	0.09775	0.009794	2.4021e-01
1	2	2	-0.5	0.2	1	2	840821	0.8344	0.7947	0.6474	3.9736e-01
0	2	2	-0.5	0.2	0.1	6	1757234	0.03834	0.03651	0.001366	2.4863e-01
1	2	2	-0.5	0.2	0.1	5	1684931	0.5711	0.5439	0.3032	5.3181e-02
0	2	2	-0.5	0.2	0.01	6	1757234	0.03834	0.03651	0.001366	2.4863e-01
1	2	2	-0.5	0.2	0.01	8	3349420	0.5203	0.4955	0.2517	1.6543e-03
0	2	2	-0.5	0.2	0.001	8	1827665	0.008482	0.008078	6.688e-05	2.4993e-01
1	2	2	-0.5	0.2	0.001	11	5025883	0.5132	0.4888	0.2449	5.1220e-03
0	2	2	-0.5	0.2	0.0001	9	1859719	0.004169	0.003971	1.616e-05	2.4998e-01
1	2	2	-0.5	0.2	0.0001	15	5245987	0.5123	0.4879	0.244	6.0365e-03
0	2	2	-0.5	0.2	1e-05	11	2730175	0.00103	0.0009805	9.854e-07	2.5000e-01
1	2	2	-0.5	0.2	1e-05	18	6699770	0.5122	0.4878	0.2439	6.0899e-03
0	2	2	-0.5	0.2	1e-06	12	2766107	0.0005137	0.0004893	2.454e-07	2.5000e-01
1	2	2	-0.5	0.2	1e-06	20	8365671	0.5119	0.4881	0.2439	6.0929e-03
0	2	2	-0.5	0.2	1e-07	14	2837519	0.0001282	0.0001221	1.529e-08	2.5000e-01
1	2	2	-0.5	0.2	1e-07	39	24177194	0.5033	0.4967	0.2472	2.8265e-03
0	2	2	-0.5	0.2	1e-08	16	3707176	3.205e-05	3.052e-05	9.549e-10	2.5000e-01
1	2	2	-0.5	0.2	1e-08	43	27503764	0.5033	0.4967	0.2472	2.8264e-03
0	2	2	-0.5	0.2	1e-09	17	3743680	1.602e-05	1.526e-05	2.387e-10	2.5000e-01
1	2	2	-0.5	0.2	1e-09	44	28335393	0.5033	0.4967	0.2472	2.8264e-03

$$rg_k^3 = \frac{rg_{k-1}^3}{10}$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	3	-0.5	0.2	1	2	41318	0.02302	0.02193	0.0004928	2.4951e-01
1	2	3	-0.5	0.2	1	2	832988	0.5324	0.507	0.2635	1.3523e-02
0	2	3	-0.5	0.2	0.1	3	876325	0.002117	0.002017	4.169e-06	2.5000e-01
1	2	3	-0.5	0.2	0.1	3	838478	0.5143	0.4898	0.2459	4.1016e-03
0	2	3	-0.5	0.2	0.01	3	876325	0.002117	0.002017	4.169e-06	2.5000e-01
1	2	3	-0.5	0.2	0.01	4	876798	0.5124	0.488	0.2441	5.8976e-03
0	2	3	-0.5	0.2	0.001	3	876325	0.002117	0.002017	4.169e-06	2.5000e-01
1	2	3	-0.5	0.2	0.001	5	1193115	0.5122	0.4878	0.2439	6.0776e-03
0	2	3	-0.5	0.2	0.0001	4	914446	0.0002102	0.0002002	4.107e-08	2.5000e-01
1	2	3	-0.5	0.2	0.0001	6	2025894	0.512	0.488	0.2439	6.0936e-03
0	2	3	-0.5	0.2	1e-05	4	914446	0.0002102	0.0002002	4.107e-08	2.5000e-01
1	2	3	-0.5	0.2	1e-05	11	6186508	0.5033	0.4967	0.2472	2.8269e-03
0	2	3	-0.5	0.2	1e-06	5	953972	2.1e-05	2e-05	4.101e-10	2.5000e-01
1	2	3	-0.5	0.2	1e-06	12	7018421	0.5033	0.4967	0.2472	2.8265e-03
0	2	3	-0.5	0.2	1e-07	5	953972	2.1e-05	2e-05	4.101e-10	2.5000e-01
1	2	3	-0.5	0.2	1e-07	13	7850036	0.5033	0.4967	0.2472	2.8264e-03
0	2	3	-0.5	0.2	1e-08	6	992227	2.1e-06	2e-06	4.101e-12	2.5000e-01
1	2	3	-0.5	0.2	1e-08	14	8681580	0.5033	0.4967	0.2472	2.8264e-03
0	2	3	-0.5	0.2	1e-09	6	992227	2.1e-06	2e-06	4.101e-12	2.5000e-01
1	2	3	-0.5	0.2	1e-09	15	9513408	0.5033	0.4967	0.2472	2.8264e-03

$$rg_k^4 = (rg_{k-1}^4)^{0.5}$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	4	-0.5	0.2	1	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1	2	24458	1.096	1.044	1.116	8.6643e-01
0	2	4	-0.5	0.2	0.1	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	0.1	3	856957	1.064	1.013	1.052	8.0173e-01
0	2	4	-0.5	0.2	0.01	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	0.01	6	1710297	1.037	0.9873	0.9991	7.4908e-01
0	2	4	-0.5	0.2	0.001	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	0.001	9	2564718	1.033	0.9841	0.9927	7.4275e-01
0	2	4	-0.5	0.2	0.0001	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	0.0001	13	2606648	1.033	0.9837	0.9919	7.4190e-01
0	2	4	-0.5	0.2	1e-05	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1e-05	16	3459436	1.033	0.9837	0.9919	7.4185e-01
0	2	4	-0.5	0.2	1e-06	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1e-06	19	5131983	1.033	0.9837	0.9918	7.4185e-01
0	2	4	-0.5	0.2	1e-07	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1e-07	23	6814629	1.033	0.9837	0.9918	7.4185e-01
0	2	4	-0.5	0.2	1e-08	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1e-08	25	7657447	1.033	0.9837	0.9918	7.4185e-01
0	2	4	-0.5	0.2	1e-09	2	664	0.8	0.2	3.64	3.3900e+00
1	2	4	-0.5	0.2	1e-09	29	8522044	1.033	0.9837	0.9918	7.4185e-01

$$rg_k^5 = \left(rg_{k-1}^5\right)^{0.25}$$

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	2	5	-0.5	0.2	1	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1	2	22142	1.048	0.9982	1.021	7.7121e-01
0	2	5	-0.5	0.2	0.1	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	0.1	2	22142	1.048	0.9982	1.021	7.7121e-01
0	2	5	-0.5	0.2	0.01	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	0.01	4	45124	1.034	0.9846	0.9936	7.4365e-01
0	2	5	-0.5	0.2	0.001	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	0.001	6	66252	1.033	0.9838	0.992	7.4196e-01
0	2	5	-0.5	0.2	0.0001	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	0.0001	7	76900	1.033	0.9837	0.9919	7.4188e-01
0	2	5	-0.5	0.2	1e-05	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1e-05	9	919211	1.033	0.9837	0.9918	7.4185e-01
0	2	5	-0.5	0.2	1e-06	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1e-06	11	2580899	1.033	0.9837	0.9918	7.4185e-01
0	2	5	-0.5	0.2	1e-07	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1e-07	12	3412564	1.033	0.9837	0.9918	7.4185e-01
0	2	5	-0.5	0.2	1e-08	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1e-08	15	4265685	1.033	0.9837	0.9918	7.4185e-01
0	2	5	-0.5	0.2	1e-09	2	664	0.8	0.2	3.64	3.3900e+00
1	2	5	-0.5	0.2	1e-09	28	5222638	1.033	0.9837	0.9918	7.4185e-01

Лучше всего себя показывает $funct_1 = -\ln(-g(x))$.

Лучше всего себя показали мультипликативные стратегии изменения коэффициента штрафа, а именно $rg_k^3 = \frac{rg_{k-1}^3}{10}$. Использование степенных функций нежелательно, так как приводит к нахождению ложной точки минимума.

Проведем дополнительные исследования с помощью $rg_k^3 = \frac{rg_{k-1}^3}{10}$ с варьированием начального коэффициента штрафа.

funct	r0	rg_num	x0x	x0y	eps	iterations	f_calc_count	x	y	f	df
0	0.1	3	-0.5	0.2	1e-09	5	973903	1.05e-06	1e-06	1.025e-12	2.5000e-01
1	0.1	3	-0.5	0.2	1e-09	14	9850083	0.5033	0.4967	0.2472	2.8264e-03
0	0.35	3	-0.5	0.2	1e-09	6	225389	3.675e-07	3.5e-07	1.256e-13	2.5000e-01
1	0.35	3	-0.5	0.2	1e-09	16	11671641	0.5033	0.4967	0.2472	2.8264e-03
0	0.6	3	-0.5	0.2	1e-09	6	1043966	6.3e-07	6e-07	3.691e-13	2.5000e-01
1	0.6	3	-0.5	0.2	1e-09	15	10115970	0.5033	0.4967	0.2472	2.8264e-03
0	0.85	3	-0.5	0.2	1e-09	6	2609207	8.926e-07	8.501e-07	7.407e-13	2.5000e-01
1	0.85	3	-0.5	0.2	1e-09	15	10756896	0.5033	0.4967	0.2472	2.8264e-03
0	1.1	3	-0.5	0.2	1e-09	6	2621034	1.155e-06	1.1e-06	1.24e-12	2.5000e-01
1	1.1	3	-0.5	0.2	1e-09	15	8975881	0.5033	0.4967	0.2472	2.8264e-03
0	1.35	3	-0.5	0.2	1e-09	6	1804556	1.418e-06	1.35e-06	1.868e-12	2.5000e-01
1	1.35	3	-0.5	0.2	1e-09	15	11339881	0.5033	0.4967	0.2472	2.8264e-03
0	1.6	3	-0.5	0.2	1e-09	6	1020362	1.68e-06	1.6e-06	2.624e-12	2.5000e-01
1	1.6	3	-0.5	0.2	1e-09	15	10867369	0.5033	0.4967	0.2472	2.8264e-03
0	1.85	3	-0.5	0.2	1e-09	6	1813996	1.943e-06	1.85e-06	3.509e-12	2.5000e-01
1	1.85	3	-0.5	0.2	1e-09	15	9552393	0.5033	0.4967	0.2472	2.8264e-03
0	2.1	3	-0.5	0.2	1e-09	6	187426	2.205e-06	2.1e-06	4.521e-12	2.5000e-01
1	2.1	3	-0.5	0.2	1e-09	15	12482389	0.5033	0.4967	0.2472	2.8264e-03
0	2.35	3	-0.5	0.2	1e-09	6	2615435	2.468e-06	2.35e-06	5.661e-12	2.5000e-01
1	2.35	3	-0.5	0.2	1e-09	15	9462601	0.5033	0.4967	0.2472	2.8264e-03
0	2.6	3	-0.5	0.2	1e-09	6	1808724	2.73e-06	2.6e-06	6.93e-12	2.5000e-01
1	2.6	3	-0.5	0.2	1e-09	15	9443394	0.5033	0.4967	0.2472	2.8264e-03
0	2.85	3	-0.5	0.2	1e-09	6	980300	2.993e-06	2.85e-06	8.326e-12	2.5000e-01
1	2.85	3	-0.5	0.2	1e-09	17	11093037	0.5033	0.4967	0.2472	2.8264e-03
0	3.1	3	-0.5	0.2	1e-09	6	184829	3.255e-06	3.1e-06	9.851e-12	2.5000e-01
1	3.1	3	-0.5	0.2	1e-09	17	12513741	0.5033	0.4967	0.2472	2.8264e-03
0	3.35	3	-0.5	0.2	1e-09	7	1023680	3.518e-07	3.35e-07	1.151e-13	2.5000e-01
1	3.35	3	-0.5	0.2	1e-09	17	11684893	0.5033	0.4967	0.2472	2.8264e-03
0	3.6	3	-0.5	0.2	1e-09	7	227020	3.78e-07	3.6e-07	1.329e-13	2.5000e-01
1	3.6	3	-0.5	0.2	1e-09	17	11856939	0.5033	0.4967	0.2472	2.8264e-03
0	3.85	3	-0.5	0.2	1e-09	7	1817362	4.043e-07	3.85e-07	1.52e-13	2.5000e-01
1	3.85	3	-0.5	0.2	1e-09	17	12501995	0.5033	0.4967	0.2472	2.8264e-03

Вывод: для данной задачи лучшей функцией является $funct_1 = -\ln(-g(x))$, оптимальной стратегией выбора коэффициента штрафа является мультипликативная функция $rg_k^3 = \frac{rg_{k-1}^3}{10}$, оптимальный начальный коэффициент штрафа для этой функции выбирается достаточно свободно в диапазоне от **0.1**.

5. Общие выводы

Большое число вычисления функций объясняется необходимостью для каждой итерации метода штрафных или барьерных функций вызывать функцию метода Гаусса, который в свою очередь для каждой своей итерации два раза вызывает функцию одномерного поиска по компонентам методом золотого сечения, который в свою очередь вызывает функцию поиска отрезка с минимумом.

Высокая погрешность решения объясняется несовершенством метода Гаусса, работоспособность которого зависит от выбора начальной точки, а также вариацией параметра delta в функции поиска отрезка с минимумом.

6. Текст программы

main.cpp

```
#include <iostream>
#include <fstream>
#include "Gauss.h"
#include "PenaltyMethodData.h"
#include "PenaltyMethod.h"
#include "BarrierMethodData.h"
#include "BarrierMethod.h"
#include "Test.h"

using namespace std;

void PenaltyGTest(const string& file_name)
{
   // Объект для хранения тестовых функций
   Test test = Test(0);

   // Объект с данными для метода штрафных функций
   PenaltyMethodData penalty_method_data = PenaltyMethodData(test);

   // Объект метода штрафных функций
   PenaltyMethod penalty_method = PenaltyMethod(penalty_method_data);

   // Поток вывода
   ofstream fout(file_name);

   int rg_num = 6;
   vector<double> x0 = { -5, 2 };

   fout << setw(5) << "funct" << setw(6) << "r0" << setw(7) << "rg_num";
   fout << setw(6) << "x0x" << setw(6) << "x0y";
   fout << setw(8) << "eps" << setw(12) << "iterations" << setw(14) << "f_calc_count";
   fout << setw(10) << "x" << setw(10) << "y" << setw(10) << "f";
   fout << setw(14) << "df" << endl;

   for(double i = 1; i < 10; i+=0.5)
   {
      double r0 = i;
      fout << "-------------------------------------------------------------";
      fout << "-----------------------------------------------" << endl;

      double f_eps = pow(10, -9);
      for(int funct_n = 0; funct_n < 4; funct_n++)
      {
         penalty_method.barrier_method_data.funct_n = funct_n;
         vector<double> x = penalty_method.FindExtremum(x0, r0, rg_num, 0, f_eps);

         // Блок вывода
```

```
            fout << setw(5) << penalty_method.barrier_method_data.funct_n << setw(6) << r0 <<
setw(7) << rg_num;
            fout << setw(6) << x0[0] << setw(6) << x0[1];
            fout << setw(8) << f_eps << setw(12) << penalty_method.iterations_count;
            fout << setw(14) << penalty_method.f_calc_cout;
            fout << setw(10) << x[0] << setw(10) << x[1] << setw(10) << penal-
ty_method_data.test.f(x) << scientific;
            fout << setw(14) << abs(penalty_method_data.test.f(x) - penal-
ty_method_data.test.f_min1) << defaultfloat << endl;
        }
    }

    fout.close();
}

void PenaltyHTest(const string& file_name)
{
    // Объект для хранения тестовых функций
    Test test = Test(0);

    // Объект с данными для метода штрафных функций
    PenaltyMethodData penalty_method_data = PenaltyMethodData(test);

    // Объект метода штрафных функций
    PenaltyMethod penalty_method = PenaltyMethod(penalty_method_data);

    // Поток вывода
    ofstream fout(file_name);

    int rh_num = 4;
    vector<double> x0 = { -5, 2 };

    fout << setw(5) << "funct" << setw(6) << "r0" << setw(7) << "rh_num";
    fout << setw(6) << "x0x" << setw(6) << "x0y";
    fout << setw(8) << "eps" << setw(12) << "iterations" << setw(14) << "f_calc_count";
    fout << setw(10) << "x" << setw(10) << "y" << setw(10) << "f";
    fout << setw(14) << "df" << endl;

    for(int i = 140; i < 260; i += 10)
    {
        double r0 = i;
        fout << "------------------------------------------------------------";
        fout << "------------------------------------------------" << endl;

        double f_eps = pow(10, -9);
        for(int funct_n = 0; funct_n < 3; funct_n++)
        {
            penalty_method.barrier_method_data.funct_n = funct_n;
            vector<double> x = penalty_method.FindExtremum(x0, r0, 0, rh_num, f_eps);

            // Блок вывода
            fout << setw(5) << penalty_method.barrier_method_data.funct_n << setw(6) << r0 <<
setw(7) << rh_num;
            fout << setw(6) << x0[0] << setw(6) << x0[1];
            fout << setw(8) << f_eps << setw(12) << penalty_method.iterations_count;
            fout << setw(14) << penalty_method.f_calc_cout;
            fout << setw(10) << x[0] << setw(10) << x[1] << setw(10) << penal-
ty_method_data.test.f(x) << scientific;
            fout << setw(14) << abs(penalty_method_data.test.f(x) - penal-
ty_method_data.test.f_min2) << defaultfloat << endl;
        }
    }

    fout.close();
```

```
}

void BarrierGTest(const string& file_name)
{
   // Объект для хранения тестовых функций
   Test test = Test(0);

   // Объект с данными для метода барьерных функций
   BarrierMethodData barrier_method_data = BarrierMethodData(test);

   // Объект метода штрафных функций
   BarrierMethod penalty_method = BarrierMethod(barrier_method_data);

   // Поток вывода
   ofstream fout(file_name);

   int rg_num = 3;
   vector<double> x0 = { -0.5, 0.2 };

   fout << setw(5) << "funct" << setw(6) << "r0" << setw(7) << "rg_num";
   fout << setw(6) << "x0x" << setw(6) << "x0y";
   fout << setw(8) << "eps" << setw(12) << "iterations" << setw(14) << "f_calc_count";
   fout << setw(10) << "x" << setw(10) << "y" << setw(10) << "f";
   fout << setw(14) << "df" << endl;

   fout << setprecision(4);

   for(double i = 0.1; i < 5; i += 0.25)
   {
      double r0 = i;
      fout << "------------------------------------------------------------";
      fout << "------------------------------------------------" << endl;

      double f_eps = pow(10, -9);
      for(int funct_n = 0; funct_n < 2; funct_n++)
      {
         penalty_method.barrier_method_data.funct_n = funct_n;
         vector<double> x = penalty_method.FindExtremum(x0, r0, rg_num, f_eps);

         // Блок вывода
         fout << setw(5) << penalty_method.barrier_method_data.funct_n << setw(6) << r0 <<
setw(7) << rg_num;
         fout << setw(6) << x0[0] << setw(6) << x0[1];
         fout << setw(8) << f_eps << setw(12) << penalty_method.iterations_count;
         fout << setw(14) << penalty_method.f_calc_cout;
         fout << setw(10) << x[0] << setw(10) << x[1] << setw(10) << barri-
er_method_data.test.f(x) << scientific;
         fout << setw(14) << abs(barrier_method_data.test.f(x) - barri-
er_method_data.test.f_min1) << defaultfloat << endl;
      }
   }

   fout.close();
}

int main()
{
   PenaltyGTest("results/penalty_g_test.txt");
   //PenaltyHTest("results/penalty_h_test.txt");
   //BarrierGTest("results/barrier_g_test.txt");
   cout << "Done!";
}
```

vector.h

```
#pragma once
```

```cpp
#include <vector>
#include <iomanip>
#include <fstream>

using namespace std;

// Умножение вектора на число
vector<double> operator * (const double& val, vector<double> vec)
{
   const size_t size = vec.size();

   for (size_t i = 0; i < size; ++i)
      vec[i] *= val;
   return vec;
}

// Деление вектора на число
vector<double> operator / (const double& val, vector<double> vec)
{
   const size_t size = vec.size();

   for(size_t i = 0; i < size; ++i)
      vec[i] /= val;
   return vec;
}

vector<double>& operator *= (vector<double>& vec, const double& val)
{
   const size_t size = vec.size();

   for(size_t i = 0; i < size; ++i)
      vec[i] *= val;

   return vec;
}

// Сложение векторов
vector<double> operator + (vector<double> vec1, const vector<double>& vec2)
{
   const size_t size = vec1.size();

   for (size_t i = 0; i < size; ++i)
      vec1[i] += vec2[i];

   return vec1;
}

// Вычитание векторов
vector<double> operator - (vector<double>vec1, const vector<double>& vec2)
{
   const size_t size = vec1.size();

   for (size_t i = 0; i < size; ++i)
      vec1[i] -= vec2[i];
   return vec1;
}

// Скалярное произведение векторов
double operator * (const vector<double>& vec1, const vector<double>& vec2)
{
   const size_t size = vec1.size();

   double res = 0;
```

```
    for(size_t i = 0; i < size; ++i)
        res += vec1[i] * vec2[i];

    return res;
}

// Норма вектора
double Norm(const vector<double>& vec)
{
    const size_t size = vec.size();

    double res = 0;
    for(int i = 0; i < size; i++)
        res += vec[i] * vec[i];

    return sqrt(res);
}
```

Function.h

```
#pragma once
#include <vector>

class Function
{
public:
    virtual double GetValue(const std::vector<double>&) const = 0;
};
```

Test.h

```
#pragma once
#include <vector>

using namespace std;

// Класс с информацией о тестовых функциях и коэффициентах r
class Test
{
public:
    int test_n;

    Test() : test_n(0) {}

    Test(int t_test_n) : test_n(t_test_n) {}

    double f(const vector<double>& x) const
    {
        switch(test_n)
        {
            case 0: return 10 * (x[1] - x[0]) * (x[1] - x[0]) + x[1] * x[1];
            case 1: return (x[0] - 1) * (x[0] - 1) + (x[1] - 3) * (x[1] - 3);
            case 2: return x[0] * x[0] + x[1] * x[1];
            case 3: return 5 * (x[1] + x[0]) * (x[1] + x[0]) + (x[0] - 2) * (x[0] - 2);
        }
    }

    double h(const vector<double>& x) const
    {
        return x[0] + x[1] - 2;
    }

    double g(const vector<double>& x) const
    {
        return 1 - x[0] - x[1];
    }
```

```
    // Точные значения минимума задач и точек
    // в которых эти значения достигаются
    double f_min1 = 0.25;
    double x_min1 = 0.5;
    double y_min1 = 0.5;

    double f_min2 = 1;
    double x_min2 = 1;
    double y_min2 = 1;
};
```

Gauss.h

```
#include "Function.h"

using namespace std;

/// <summary>
/// Класс для поиска экстремума функции многомерного параметра
/// методом Гаусса
/// </summary>
class Gauss
{
public:
    size_t size = 0;            // Размерность вектора
    vector<double> prev;        // Приближение на текущем шаге
    vector<double> curr;        // Новое приближение,
    double delta = 10;

    int max_iter_count = 5000;  // Максимальное количество итераций

    ///<param name = "t_size">- размерность вектора функции</param>
    Gauss(const size_t& t_size);

    /// <summary>
    /// Поиск экстремума функции методом Гаусса
    /// </summary>
    /// <param name="funct">- объект класса-наследника класса Function с информацией о функ-
ции</param>
    /// <param name="x0">- вектор начального приближения</param>
    /// <param name="f_eps">- точость по изменению функции</param>
    /// <param name="xs_eps">- точность по компонентам</param>
    /// <param name="fout">- поток вывода</param>
    /// <returns>Количество итераций</returns>
    int FindExtremum(const Function& funct,
                     const vector<double>& x0,
                     const double& f_eps, const double& xs_eps);

    const double SQRT5 = sqrt(5);
    const double PI = 3.141592653589793238462;
    int f_calc_count = 0;           // Количество вычислений функции

    /// <summary>
    /// Поиск отрезка с минимумом функции для метода Гаусса
    /// </summary>
    /// <param name="funct">- объект класса-наследника класса Function с информацией о функ-
ции</param>
    /// <param name="x">- вектор компонент</param>
    /// <param name="comp_n">- номер компоненты</param>
    /// <returns>Пара чисел - отрезок с минимумом функции</returns>
    pair<double, double> FindSegmentWithMin(const Function& funct,
                                            vector<double> x,
```

```
                                        const int& comp_n);

    /// <summary>
    /// Поиск аргумента минимума функции для метода Гаусса
    /// </summary>
    /// <param name="funct">- объект класса-наследника класса Function с информацией о функ-
ции</param>
    /// <param name="x">- вектор компонент</param>
    /// <param name="comp_n">- номер компоненты</param>
    /// <param name="eps">- точность поиска</param>
    /// <returns>Аргумент минимума функции</returns>
    double FindMinArgGolden(const Function& funct,
                            vector<double> x,
                            const int& comp_n,
                            const double& eps);
};
```

Gauss.cpp

```
#include <vector>
#include <fstream>
#include "Gauss.h"

Gauss::Gauss(const size_t& t_size) : size(t_size)
{
    prev = vector<double>(t_size);
    curr = vector<double>(t_size);
}

int Gauss::FindExtremum(const Function& funct,
                  const vector<double>& x0,
                  const double& f_eps, const double& xs_eps)
{
    prev = x0;
    curr = x0;

    bool result_fit = false;
    int iter_count = 0;
    do
    {
        for(size_t i = 0; i < size; i++)
            curr[i] = FindMinArgGolden(funct, curr, i, 1e-15);

        iter_count++;
        // Расчет изменения решения на текущей итерации
        if(abs(funct.GetValue(prev) - funct.GetValue(curr)) < f_eps)
            result_fit = true;

        for(int i = 0; i < size && result_fit; i++)
            if(abs(prev[i] - curr[i]) > xs_eps)
                result_fit = false;

        prev = curr;
    } while(iter_count < max_iter_count && !result_fit);

    return iter_count;
}

pair<double, double> Gauss::FindSegmentWithMin(const Function& funct,
                                               vector<double> x,
                                               const int& comp_n)
{
    pair<double, double> result;
    double x0 = 0;
    double xk, xk1, xk_1, h = 1;
```

```
x[comp_n] = x0;
double f = funct.GetValue(x);
f_calc_count += 1;

x[comp_n] = x0 + delta;
if(f == funct.GetValue(x))
{
   result.first = x0;
   result.second = x0 + delta;
   f_calc_count = 2;

   return result;
}
else
{
   x[comp_n] = x0 - delta;
   if(f == funct.GetValue(x))
   {
      result.first = x0 - delta;
      result.second = x0;
      f_calc_count = 3;

      return result;
   }
   else
   {
      x[comp_n] = x0 + delta;
      if(f > funct.GetValue(x))
      {
         xk = x0 + delta;
         h = delta;
         f_calc_count++;
      }
      else
      {
         x[comp_n] = x0 - delta;
         if(f > funct.GetValue(x))
         {
            xk = x0 - delta;
            h = -delta;
            f_calc_count += 2;
         }
         else
         {
            result.first = x0 - delta;
            result.second = x0 + delta;
            f_calc_count += 2;

            return result;
         }
      }

      xk_1 = x0;

      bool exit = false;
      do
      {
         h *= 2;
         xk1 = xk + h;

         x[comp_n] = xk;
         double f1 = funct.GetValue(x);

         x[comp_n] = xk1;
```

```
            double f2 = funct.GetValue(x);

            if(f1 > f2)
            {
               xk_1 = xk;
               xk = xk1;
            }
            else
               exit = true;

            f_calc_count += 2;
         } while(!exit);

         result.first = xk_1;
         result.second = xk;
      }
   }
   return result;
}

double Gauss::FindMinArgGolden(const Function& funct,
                        vector<double> x,
                        const int& comp_n,
                        const double& eps)
{
   pair<double, double> segment = FindSegmentWithMin(funct, x, comp_n);
   double a = segment.first, b = segment.second;

   double x1 = a + (3 - SQRT5) / 2 * (b - a);
   double x2 = a + (SQRT5 - 1) / 2 * (b - a);

   x[comp_n] = x1;
   double f1 = funct.GetValue(x);
   f_calc_count += 1;

   x[comp_n] = x2;
   double f2 = funct.GetValue(x);
   f_calc_count += 1;

   double a1, b1;

   int iter_count = 0;
   for(; abs(b - a) > eps; iter_count++)
   {
      a1 = a, b1 = b;
      if(f1 < f2)
      {
         b = x2;
         x2 = x1;
         x1 = a + (3 - SQRT5) / 2 * (b - a);
         f2 = f1;

         x[comp_n] = x1;
         f1 = funct.GetValue(x);
         f_calc_count += 1;
      }
      else
      {
         a = x1;
         x1 = x2;
         x2 = a + (SQRT5 - 1) / 2 * (b - a);
         f1 = f2;

         x[comp_n] = x2;
```

```
            f2 = funct.GetValue(x);
            f_calc_count += 1;
        }
    }

    return a;
}
```

PenaltyMethodData.cpp

```
#pragma once
#include<vector>
#include "Test.h"
#include "Function.h"

using namespace std;

class PenaltyMethodData : public Function
{
public:
   Test test;
   int funct_n = 0;

   double rg = 0;
   double rh = 0;

   PenaltyMethodData(const Test& t_test, int t_funct_n) :
      test(t_test), funct_n(t_funct_n) { }

   PenaltyMethodData(const Test& t_test) : test(t_test) { }

   double G(const vector<double>& x) const
   {
      if(test.g(x) <= 0)
         return 0;
      else
      {
         switch(funct_n)
         {
            case 0: return 0.5 * (test.g(x) + abs(test.g(x)));
            case 1: return pow(0.5 * (test.g(x) + abs(test.g(x))), 2);
            case 2: return pow(0.5 * (test.g(x) + abs(test.g(x))), 4);
            case 3: return 1.0 / test.g(x);
            default: return 0;
         }
      }
   }

   double H(const vector<double>& x) const
   {
      switch(funct_n)
      {
         case 0: return abs(test.h(x));
         case 1: return pow(test.h(x), 2);
         case 2: return pow(test.h(x), 4);
         default: return 0;
      }
   }

   double GetValue(const vector<double>& x) const override
   {
      return test.f(x) + rg * G(x) + rh * H(x);
   }
};
```

PenaltyMethod.cpp

```
#pragma once
#include <iomanip>
#include "PenaltyMethodData.h"

using namespace std;

class PenaltyMethod
{
public:
   PenaltyMethodData barrier_method_data;

   int max_iter_count = 100;  // Максимальное количество итераций
   int iterations_count = 0;
   int f_calc_cout = 0;

   double CalcNewR(const int& num, const double& r)
   {
      switch(num)
      {
         case 0: return 0.0;

         case 1: return r + 1.0;
         case 2: return r + 100;

         case 3: return r * 2;
         case 4: return r * 200;

         case 5: return r * r;
         case 6: return r * r * r;
      }
   }

   PenaltyMethod(const PenaltyMethodData& t_penalty_method_data):
      barrier_method_data(t_penalty_method_data) { }

   /// <summary>
   /// Поиск экстремума методом штрафных функций
   /// </summary>
   /// <param name="x0">- начальная точка</param>
   /// <param name="r0">- начальное значение коэффициента штрафа</param>
   /// <param name="rg_num">- номер функции для вычисления параметра штрафа для функции
G</param>
   /// <param name="rh_num">- номер функции для вычисления параметра штрафа для функции
H</param>
   /// <param name="f_eps">- точость по изменению функции</param>
   /// <returns>Точка с найденным экстремумом</returns>
   vector<double> FindExtremum(const vector<double>& x0,
                               const double& r0,
                               const int& rg_num,
                               const int& rh_num,
                               const double& f_eps)
   {
      barrier_method_data.rg = r0;
      barrier_method_data.rh = r0;

      // Значения функции на предыдущей и текущей итерациях
      double prev_f = barrier_method_data.test.f(x0);
      double curr_f;

      // Найденная точка
      vector<double> x(2);
```

```
        f_calc_cout = 0;
        iterations_count = 0;

        bool result_fit = false;
        do
        {
            Gauss gauss = Gauss(2);

            // Изменение коэффициентов штрафа
            barrier_method_data.rg = CalcNewR(rg_num, barrier_method_data.rg);
            barrier_method_data.rh = CalcNewR(rh_num, barrier_method_data.rh);

            // Поиск экстремума методом Гаусса
            gauss.FindExtremum(barrier_method_data, x0, 1e-20, 1e-20);

            // Полученная точка
            x = gauss.curr;

            // Функция в этой точке
            curr_f = barrier_method_data.test.f(gauss.curr);

            iterations_count++;
            f_calc_cout += gauss.f_calc_count;

            if(abs(prev_f - curr_f) < f_eps)
                result_fit = true;

            prev_f = curr_f;

        } while(!result_fit && iterations_count < max_iter_count);

        return x;
    }
};
```

BarrierMethodData.cpp

```
#pragma once
#include<vector>
#include "Test.h"
#include "Function.h"

using namespace std;

class BarrierMethodData : public Function
{
public:
    Test test;
    int funct_n = 0;

    double rg = 0;

    BarrierMethodData(const Test& t_test, int t_funct_n) :
        test(t_test), funct_n(t_funct_n) { }

    BarrierMethodData(const Test& t_test) : test(t_test) { }

    double G(const vector<double>& x) const
    {
        switch(funct_n)
        {
            case 0: return - 1.0 / test.g(x);
            case 1: return -1.0 * log(-1.0 * test.g(x));
            default: return 0;
        }
    }
```

```
    double GetValue(const vector<double>& x) const override
    {
        return test.f(x) + rg * G(x);
    }
};
```

BarrierMethod.cpp

```
#pragma once
#include <iomanip>
#include "BarrierMethodData.h"

using namespace std;

class BarrierMethod
{
public:
    BarrierMethodData barrier_method_data;

    int max_iter_count = 100;  // Максимальное количество итераций
    int iterations_count = 0;
    int f_calc_cout = 0;

    double CalcNewR(const int& num, const double& rg)
    {
        switch(num)
        {
            case 0: return 0.0;

            case 1: return rg - 0.01;

            case 2: return rg / 2;
            case 3: return rg / 10;

            case 4: return pow(rg, 0.5);
            case 5: return pow(rg, 0.25);
        }
    }

    BarrierMethod(const BarrierMethodData& t_barrier_method_data) :
        barrier_method_data(t_barrier_method_data) { }

    /// <summary>
    /// Поиск экстремума методом барьерных функций
    /// </summary>
    /// <param name="x0">- начальная точка</param>
    /// <param name="r0">- начальное значение коэффициента штрафа</param>
    /// <param name="rg_num">- номер функции для вычисления параметра штрафа для функции
G</param>
    /// <param name="f_eps">- точость по изменению функции</param>
    /// <returns>Точка с найденным экстремумом</returns>
    vector<double> FindExtremum(const vector<double>& x0,
                                const double& r0,
                                const int& rg_num,
                                const double& f_eps)
    {
        barrier_method_data.rg = r0;

        // Значения функции на предыдущей и текущей итерациях
        double prev_f = barrier_method_data.test.f(x0);
        double curr_f;

        // Найденная точка
        vector<double> x(2);
```

```
        f_calc_cout = 0;
        iterations_count = 0;

        bool result_fit = false;
        do
        {
            Gauss gauss = Gauss(2);

            // Изменение коэффициентов штрафа
            barrier_method_data.rg = CalcNewR(rg_num, barrier_method_data.rg);

            // Поиск экстремума методом Гаусса
            gauss.FindExtremum(barrier_method_data, x0, 1e-20, 1e-20);

            // Полученная точка
            x = gauss.curr;

            // Функция в этой точке
            curr_f = barrier_method_data.test.f(gauss.curr);

            iterations_count++;
            f_calc_cout += gauss.f_calc_count;

            if(abs(prev_f - curr_f) < f_eps)
                result_fit = true;

            prev_f = curr_f;

        } while(!result_fit && iterations_count < max_iter_count);

        return x;
    }
};
```