МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

НГТУ НЭТИ

Кафедра прикладной математики и информатики

Лабораторная работа №3
по дисциплине «Методы оптимизации»
Метод штрафных функций

ФПМИ

Группа	ПМ-92
Бригада	08
Студенты	БЕГИЧЕВ АЛЕКСАНДР
	ШИШКИН НИКИТА
Преподаватель	ФИЛИППОВА Е.В.
Дата	19.04.2022

Новосибирск

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.

Задание

Вариант 8:

- Первая задача (а):
  $f(x, y) = 5(x + y)^2 + (x - 2)^2 \to \min$
  $x + y \geqslant 1$
- Вторая задача (б):
  $f(x, y) = 5(x + y)^2 + (x - 2)^2 \to \min$
  $x = y$

1. Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с использованием **метода штрафных функций**.
2. Исследовать сходимость **метода штрафных функций** в зависимости
   - от выбора штрафных функций,
   - начальной величины коэффициента штрафа,
   - стратегии изменения коэффициента штрафа,
   - начальной точки,
   - задаваемой точности ε.

   Сформулировать выводы.
3. Применяя методы поиска минимума 0-го порядка, реализовать программу для решения заачи нелинейного программирования с ограничением типа неравенства (**только задача а**) с использованием **метода барьерных функций**.
4. Исследовать сходимость **метода барьерных функций** (только задача а) в зависимости
   - от выбора барьерных функций,
   - начальной величины коэффициента штрафа,
   - стратегии изменения коэффициента штрафа,
   - начального приближения,
   - задаваемой точности ε.

   Сформулировать выводы.

Исследование на сходимость метода штрафных функций

Первая задача (а), аналитическое значение

$\min\{5(x+y)^2 + (x-2)^2 \,|\, x+y \geqslant 1\} = 5$, точка $(x, y) = (2, -1)$

Вторая задача (б), аналитическое значение

$\min\{5(x+y)^2 + (x-2)^2 \,|\, x = y\} = \dfrac{80}{21}$, точка $(x, y) = \left(\dfrac{2}{21}, \dfrac{2}{21}\right)$

Описание столбцов

- № – номер теста.
- name – название метода спуска.
- x_{01} и x_{02} – начальная точка.
- ε – точность вычислений.
- iters – количество итераций.
- evals – количество вычислений функции.
- x_1 и x_2 – приближённые значения точек.
- $f(x_1, x_2)$ – приближённое минимальное значени функции.
- coef – конечный коэффциент штрафа.

Если не указана штрафная функция, то:

- Для первой задачи: $G_j = \left[\frac{1}{2}\left\{g_j(\overline{x} + |g_j(\overline{x})|\right\}\right]^\alpha$, $\alpha = 1$.
- Для второй задачи: $H_j = |h_j(\overline{x}|^\alpha$, $\alpha = 1$

Разные начальные точки

Коэффициент для первой задачи: $r = 1.5$, $r_{k+1} = r_k \cdot 4$
Коэффициент для второй задачи: $r = 3$, $r_{k+1} = r_k \cdot 3$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-0.50	2.50	$1.00 \cdot 10^{-7}$	86	935	2.00005856	-1.00005855	5.00000009	$2.244867 \cdot 10^{51}$
2	simplex	2.00	2.00	$1.00 \cdot 10^{-7}$	75	815	2.00010731	-1.00010729	5.00000020	$5.352179 \cdot 10^{44}$
3	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	75	812	1.99975136	-0.99975136	5.00000007	$5.352179 \cdot 10^{44}$
4	simplex	10.00	0.00	$1.00 \cdot 10^{-7}$	73	802	1.99603163	-0.99603162	5.00001579	$3.345112 \cdot 10^{43}$
5	simplex	10.00	10.00	$1.00 \cdot 10^{-7}$	78	849	1.99969841	-0.99969841	5.00000013	$3.425394 \cdot 10^{46}$

Таблица 1: Разные начальные точки для первой задачи.

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	116	1,281	0.09522163	0.09522163	3.80952382	$3.697552 \cdot 10^{55}$
2	simplex	-1.00	-1.00	$1.00 \cdot 10^{-7}$	120	1,323	0.09526497	0.09526497	3.80952382	$2.995017 \cdot 10^{57}$
3	simplex	3.00	3.00	$1.00 \cdot 10^{-7}$	113	1,248	0.09520510	0.09520510	3.80952383	$1.369464 \cdot 10^{54}$
4	simplex	5.00	5.00	$1.00 \cdot 10^{-7}$	109	1,204	0.09536883	0.09536883	3.80952417	$1.690696 \cdot 10^{52}$
5	simplex	10.00	10.00	$1.00 \cdot 10^{-7}$	115	1,269	0.09528085	0.09528085	3.80952385	$1.232517 \cdot 10^{55}$

Таблица 2: Разные начальные точки для второй задачи

Разная начальная величина коэффициента штрафа

Коэффициент: $r = [5, 15, 25, 35, 45]$, $r_{k+1} = r_k \cdot 2$
Штрафная функция для первой задачи: $G_j = \left[\frac{1}{2}\left\{g_j(\overline{x} + |g_j(\overline{x})|\right\}\right]^{\alpha}$, $\alpha = 2$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	1.50	0.50	$1.00 \cdot 10^{-7}$	71	775	2.00002985	-1.00002983	5.00000024	$5.902958 \cdot 10^{21}$
2	simplex	1.50	0.50	$1.00 \cdot 10^{-7}$	61	666	1.99814749	-0.99814740	5.00000430	$1.729382 \cdot 10^{19}$
3	simplex	1.50	0.50	$1.00 \cdot 10^{-7}$	76	823	1.99996656	-0.99996656	5.00000002	$9.444733 \cdot 10^{23}$
4	simplex	1.50	0.50	$1.00 \cdot 10^{-7}$	116	1,264	1.99991817	-0.99991815	5.00000020	$1.453843 \cdot 10^{36}$
5	simplex	1.50	0.50	$1.00 \cdot 10^{-7}$	116	1,264	1.99991817	-0.99991815	5.00000020	$1.869227 \cdot 10^{36}$

Таблица 3: Разная начальная величина коэффициента штрафа для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	116	1,280	0.09521562	0.09521562	3.80952382	$2.076919 \cdot 10^{35}$
2	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	128	1,413	0.09519145	0.09519145	3.80952386	$2.552118 \cdot 10^{39}$
3	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	110	1,216	0.09519533	0.09519533	3.80952385	$1.622593 \cdot 10^{34}$
4	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	110	1,216	0.09519533	0.09519533	3.80952385	$2.271630 \cdot 10^{34}$
5	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	110	1,216	0.09519533	0.09519533	3.80952385	$2.920667 \cdot 10^{34}$

Таблица 4: Разная начальная величина коэффициента штрафа для второй задачи

Разная точность вычислений

Коэффициент для первой задачи: $r = 1.5$, $r_{k+1} = r_k \cdot 4$
Коэффициент для второй задачи: $r = 10$, $r_{k+1} = r_k + 10$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	2.00	2.00	$1.00 \cdot 10^{-3}$	37	405	2.00664381	-1.00645049	5.00197755	$7.083550 \cdot 10^{21}$
2	simplex	2.00	2.00	$1.00 \cdot 10^{-4}$	48	525	1.99953842	-0.99953505	5.00003397	$2.971056 \cdot 10^{28}$
3	simplex	2.00	2.00	$1.00 \cdot 10^{-5}$	55	600	1.99897336	-0.99897242	5.00001040	$4.867778 \cdot 10^{32}$
4	simplex	2.00	2.00	$1.00 \cdot 10^{-6}$	63	685	1.99892398	-0.99892394	5.00000163	$3.190147 \cdot 10^{37}$
5	simplex	2.00	2.00	$1.00 \cdot 10^{-7}$	75	815	2.00010731	-1.00010729	5.00000020	$5.352179 \cdot 10^{44}$

Таблица 5: Разная точность вычислений для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	3.00	3.00	$1.00 \cdot 10^{-3}$	32	356	0.09191878	0.09191632	3.80975066	320
2	simplex	3.00	3.00	$1.00 \cdot 10^{-4}$	40	444	0.09659969	0.09659977	3.80956289	400
3	simplex	3.00	3.00	$1.00 \cdot 10^{-5}$	46	509	0.09503654	0.09503655	3.80952470	460
4	simplex	3.00	3.00	$1.00 \cdot 10^{-6}$	54	596	0.09521233	0.09521233	3.80952383	540
5	simplex	3.00	3.00	$1.00 \cdot 10^{-7}$	60	662	0.09522541	0.09522541	3.80952381	600

Таблица 6: Разная точность вычислений для второй задачи

Разная стратегия изменения коэффициента штрафа

Коэффициент: $r = [3, 3, 3, 3, 3]$, $r_{k+1} = [r_k + 10, r_k \cdot 2, r_k \cdot 4, r_k \cdot 8, r_k \cdot 16]$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	81	886	2.00025119	-1.00025119	5.00000007	803
2	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	66	723	2.00075589	-1.00075588	5.00000060	1,953
3	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	63	684	1.99984913	-0.99984913	5.00000007	$1.383506 \cdot 10^{19}$
4	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	82	896	2.00017655	-1.00017653	5.00000022	$4.240433 \cdot 10^{73}$
5	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	82	896	2.00017655	-1.00017653	5.00000022	$1.025274 \cdot 10^{98}$

Таблица 7: Разная стратегия изменения коэффициента штрафа для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	80	882	0.09523209	0.09523209	3.80952381	793
2	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	73	807	0.09523961	0.09523961	3.80952381	2,163
3	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	115	1,269	0.09527600	0.09527600	3.80952384	$6.230756 \cdot 10^{34}$
4	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	116	1,280	0.09521562	0.09521562	3.80952382	$2.150155 \cdot 10^{104}$
5	simplex	-0.50	-0.50	$1.00 \cdot 10^{-7}$	116	1,280	0.09521562	0.09521562	3.80952382	$8.931394 \cdot 10^{138}$

Таблица 8: Разная стратегия изменения коэффициента штрафа для второй задачи

Разные штрафные функции

Коэффициент для первой задачи: $r = 1.5$, $r_{k+1} = r_k \cdot 4$, $G_j = \left[\frac{1}{2}\left\{g_j(\overline{x} + |g_j(\overline{x})|\right\}\right]^{\alpha}$, $\alpha = [1, 2, 4, 8, 16]$
Коэффициент для второй задачи: $r = 1.5$, $r_{k+1} = r_k \cdot 4$, $H_j = |h_j(\overline{x}|^{\alpha}$, $\alpha = [1, 2, 4, 8, 16]$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	75	812	1.99975136	-0.99975136	5.00000007	$5.352179 \cdot 10^{44}$
2	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	75	812	1.99975136	-0.99975136	5.00000007	$5.352179 \cdot 10^{44}$
3	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	82	892	2.00014259	-1.00014258	5.00000009	$8.769010 \cdot 10^{48}$
4	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	91	986	1.99996370	-0.99996371	4.99999992	$2.298743 \cdot 10^{54}$
5	simplex	4.00	3.00	$1.00 \cdot 10^{-7}$	178	1,899	1.99990515	-0.99990522	4.99999928	$5.504397 \cdot 10^{106}$

Таблица 9: Разные штрафные функции для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	116	1,281	0.09522163	0.09522163	3.80952382	$2.588155 \cdot 10^{69}$
2	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	116	1,281	0.09522163	0.09522163	3.80952382	$2.588155 \cdot 10^{69}$
3	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	148	1,623	0.09519701	0.09519701	3.80952384	$4.774303 \cdot 10^{88}$
4	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	139	1,506	0.09523807	0.09523807	3.80952381	$1.821252 \cdot 10^{83}$
5	simplex	-3.00	-3.00	$1.00 \cdot 10^{-7}$	162	1,729	0.09519997	0.09519969	3.80952330	$1.281592 \cdot 10^{97}$

Таблица 10: Разные штрафные функции для второй задачи

Исследование сходимости барьерных функций

$G_j = -\ln\left(-g_j(\overline{x})\right)$
Коэффициент для первой задачи: $r = 1.5$, $r_{k+1} = r_k/2$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	-6.00	8.00	$1.00 \cdot 10^{-7}$	82	891	2.00003031	-1.00003030	5.00000004	$6.203855 \cdot 10^{-25}$
2	simplex	0.50	1.00	$1.00 \cdot 10^{-7}$	72	785	1.99948912	-0.99948912	5.00000027	$6.352747 \cdot 10^{-22}$
3	simplex	2.00	1.00	$1.00 \cdot 10^{-7}$	76	829	2.00028413	-1.00028412	5.00000016	$3.970467 \cdot 10^{-23}$
4	simplex	3.00	3.00	$1.00 \cdot 10^{-7}$	71	770	1.99956247	-0.99956245	5.00000034	$1.270549 \cdot 10^{-21}$
5	simplex	10.00	0.00	$1.00 \cdot 10^{-7}$	73	802	1.99603163	-0.99603162	5.00001579	$3.176374 \cdot 10^{-22}$

Таблица 11: Разные начальные точки для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	68	742	2.00023089	-1.00023087	5.00000029	$1.355253 \cdot 10^{-20}$
2	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	68	742	2.00023089	-1.00023087	5.00000029	$2.710505 \cdot 10^{-20}$
3	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	68	743	2.00023089	-1.00023087	5.00000029	$5.421011 \cdot 10^{-20}$
4	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	73	803	2.01048110	-1.01048110	5.00010986	$3.388132 \cdot 10^{-21}$
5	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	68	741	2.00023089	-1.00023087	5.00000029	$2.168404 \cdot 10^{-19}$

Таблица 12: Разные начальные штрафные коэффциенты для первой задачи

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	2.00	3.00	$1.00 \cdot 10^{-3}$	26	288	2.01351464	-1.01299253	5.00540508	$4.470348 \cdot 10^{-8}$
2	simplex	2.00	3.00	$1.00 \cdot 10^{-4}$	42	461	2.00493809	-1.00493398	5.00006547	$6.821210 \cdot 10^{-13}$
3	simplex	2.00	3.00	$1.00 \cdot 10^{-5}$	51	558	2.00205798	-1.00205699	5.00001411	$1.332268 \cdot 10^{-15}$
4	simplex	2.00	3.00	$1.00 \cdot 10^{-6}$	61	666	2.00044708	-1.00044700	5.00000094	$1.301043 \cdot 10^{-18}$
5	simplex	2.00	3.00	$1.00 \cdot 10^{-7}$	68	742	2.00023089	-1.00023087	5.00000029	$1.016440 \cdot 10^{-20}$

Таблица 13: Разная точность для первой задачи

Коэффициент для первой задачи: $r = 10$, $r_{k+1} = [r_k/2, r_k/4, r_k/8, r_k/16, r_k/32]$

№	name	x_{01}	x_{02}	ϵ	iters	evals	x_1	x_2	$f(x_1, x_2)$	coef
1	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	75	819	2.00009533	-1.00009531	5.00000019	$5.293956 \cdot 10^{-22}$
2	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	83	907	1.99982123	-0.99982120	5.00000035	$4.276424 \cdot 10^{-49}$
3	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	83	907	1.99982123	-0.99982120	5.00000035	$8.843437 \cdot 10^{-74}$
4	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	83	907	1.99982123	-0.99982120	5.00000035	$1.828780 \cdot 10^{-98}$
5	simplex	1.50	-0.50	$1.00 \cdot 10^{-7}$	83	907	1.99982123	-0.99982120	5.00000035	$3.781828 \cdot 10^{-123}$

Таблица 14: Разная стратегия изменения коэффициента штрафа для первой задачи

Вывод

Выбор начальной точки влияет на количество итераций и вычислений целевой функции. Выбор разного начального коэффициента штрафа повлиял только на конечный коэффициент штрафа. Чем выше точность вычислений, тем больше итераций и вычислений целевой функции. Стратегия изменения штрафа сильно влияет на результат при условии разных производимых над коэффициентом операций, но не от побочных коэффицентов.

Листинги

Programs.cs

```
using oMethods_3;

Solver solverWithGauss =
    Solver.CreateBuilder().SetPoint(Argument.ReadJson("point.json")!)
.SetFunction(new FunctionA(2, 1,
    MethodTypes.InteriorPointReverse)).SetMinMethod(new
    GaussAlgorithm(1000, 1E-5))
.SetMinMethod1D(new Fibonacci(1E-5)).SetStrategyType((StrategyTypes.Div,
    0.5));

Solver solverWithSimplex =
    Solver.CreateBuilder().SetPoint(Argument.ReadJson("point.json")!)
.SetFunction(new FunctionA(2, 1, MethodTypes.Penalty)).SetMinMethod(new
    SimplexMethod(1000, 1, 1E-4))
.SetStrategyType((StrategyTypes.Mult, 2));

solverWithGauss.Compute();
solverWithSimplex.Compute();
```

Solver.cs

```
namespace oMethods_3;

public enum MethodTypes {
    Penalty,
    InteriorPointLog,
    InteriorPointReverse
}

public enum StrategyTypes {
    Add,
    Mult,
    Div,
}

public class Solver {
    public class SolverBuilder {
        private readonly Solver _solver = new();

        public SolverBuilder SetPoint(Argument point) {
            _solver._initPoint = point;
            return this;
        }

        public SolverBuilder SetFunction(IFunction function) {
            _solver._function = function;
            return this;
        }

        public SolverBuilder SetMinMethod(IMinMethodND method) {
            _solver._methodND = method;
            return this;
        }

        public SolverBuilder SetMinMethod1D(IMinMethod1D method) {
            _solver._method1D = method;
            return this;
        }

        public SolverBuilder SetStrategyType((StrategyTypes, double)
 ↪  strategy) {
            _solver._strategy = strategy;
            return this;
        }

        public static implicit operator Solver(SolverBuilder builder)
            ⇒ builder._solver;
    }
```

```
47
48      private IFunction _function = default!;
49      private IMinMethodND _methodND = default!;
50      private IMinMethod1D _method1D = default!;
51      private Argument _initPoint = default!;
52      private (StrategyTypes, double)? _strategy;
53
54      public void Compute() {
55          try {
56              ArgumentNullException.ThrowIfNull(_initPoint,
    ↪ $"{nameof(_initPoint)} cannot be null, set the init point");
57
58              ArgumentNullException.ThrowIfNull(_function,
59                                                $"{nameof(_function)} cannot
    ↪ be null, set the function");
60
61              ArgumentNullException.ThrowIfNull(_methodND,
62                                                $"{nameof(_methodND)} cannot
    ↪ be null, set the method of minimization");
63
64              if (_methodND.Need1DSearch && _method1D is null)
65                  throw new ArgumentNullException(nameof(_method1D), "Set
    ↪ the one dimensional search method");
66
67              if (!_function.Limitation(_initPoint))
68                  throw new Exception("The starting point does not satisfy
    ↪ the limitation of the function");
69
70              if (_function.Degree is null && _function.MethodType ==
    ↪ MethodTypes.Penalty)
71                  throw new Exception("When selecting the penalty function
    ↪ method, the value of the degree cannot be null");
72
73              if (_function.MethodType == MethodTypes.Penalty &&
    ↪ _function.Coef ≠ 0 && _strategy?.Item1 ≠ StrategyTypes.Add &&
    ↪ _strategy?.Item1 ≠ StrategyTypes.Mult)
74                  throw new Exception("With the chosen method of penalty
    ↪ functions, the strategy for changing the coefficient should be
    ↪ addition or multiplication");
75
76              if ((_function.MethodType == MethodTypes.InteriorPointLog ||
    ↪ _function.MethodType == MethodTypes.InteriorPointReverse) &&
77                  _strategy?.Item1 ≠ StrategyTypes.Div && _function.Coef ≠
    ↪ 0)
78                  throw new Exception("With the chosen method of barrier
    ↪ functions, the strategy for changing the coefficient should be
    ↪ division");
79
80              if (_function.MethodType == MethodTypes.Penalty &&
    ↪ _function.Degree ≠ 1 && (_function.Degree % 2 ≠ 0 ||
    ↪ _function.Degree == 0))
```

```
                throw new Exception("The degree of the penalty function must be even or equal to one");

            if (_function.Coef ≠ 0 && _strategy is null)
                throw new Exception("Need to choose a strategy to change the penalty coefficient");

            if ((_strategy?.Item1 == StrategyTypes.Mult || _strategy?.Item1 == StrategyTypes.Add) && _strategy?.Item2 < 1)
                throw new Exception("With this strategy, the coefficient of change should be > 1");

            if (_strategy?.Item1 == StrategyTypes.Div && _strategy?.Item2 < 1)
                throw new Exception("With this strategy, the coefficient of change should be > 1");

            _methodND.Compute(_initPoint, _function, _method1D, _strategy);

        } catch (Exception ex) {
            Console.WriteLine($"We had problem: {ex.Message}");
        }
    }

    public static SolverBuilder CreateBuilder()
        ⇒ new();
}
```

Argument.cs

```
namespace oMethods_3;

public class ArgumentConverter : JsonConverter {
    public override void WriteJson(JsonWriter writer, object? value, JsonSerializer serializer) {
        if (value is null) {
            writer.WriteNull();
            return;
        }

        Argument arg = (Argument)value;

        writer.WriteStartObject();
        writer.WritePropertyName("Extremum");
        writer.WriteStartArray();
        Array.ForEach(arg.Variables, (x) ⇒ writer.WriteValue(x));
        writer.WriteEndArray();
        writer.WriteEndObject();
```

```
    }

    public override object? ReadJson(JsonReader reader, Type objectType,
↪   object? existingValue, JsonSerializer serializer) {
        if (reader.TokenType == JsonToken.Null || reader.TokenType ≠
↪   JsonToken.StartObject)
            return null;

        Argument value;

        JObject? maintoken = JObject.Load(reader);
        JToken? token = maintoken["Number"];
        int number = Convert.ToInt32(token);

        value = new(number);

        token = maintoken["Point"];

        if (token is not null) {
            double[]? variables =
↪   serializer.Deserialize<double[]>(token.CreateReader());

            if (variables is not null)
                value.Variables = variables;
        }

        return value;
    }

    public override bool CanConvert(Type objectType)
        ⇒ objectType == typeof(Argument);
}

[JsonConverter(typeof(ArgumentConverter))]
public class Argument : ICloneable {
    [JsonProperty("Point")]
    public double[] Variables { get; set; }
    public int Number { get; init; }

    public double this[int index] {
        get ⇒ Variables[index];
        set ⇒ Variables[index] = value;
    }

    public Argument(int number) {
        Variables = new double[number];
        Number = number;
    }

```

```
    public void Fill(double value) {
        for (int i = 0; i < Number; i++)
            Variables[i] = value;
    }

    public double Norm() {
        double result = 0.0;

        for (int i = 0; i < Number; i++)
            result += Variables[i] * Variables[i];

        return Math.Sqrt(result);
    }

    public static Argument operator -(Argument fstArg, Argument sndArg) {
        Argument result = new(fstArg.Number);

        for (int i = 0; i < result.Number; i++)
            result[i] = fstArg[i] - sndArg[i];

        return result;
    }

    public static Argument operator +(Argument fstArg, Argument sndArg) {
        Argument result = new(fstArg.Number);

        for (int i = 0; i < result.Number; i++)
            result[i] = fstArg[i] + sndArg[i];

        return result;
    }

    public static Argument operator *(double constant, Argument arg) {
        Argument result = new(arg.Number);

        for (int i = 0; i < result.Number; i++)
            result[i] = constant * arg[i];

        return result;
    }

    public static Argument? ReadJson(string jsonPath) {
        try {
            if (!File.Exists(jsonPath))
                throw new Exception("File does not exist");

            var sr = new StreamReader(jsonPath);
            using (sr) {
                return
↪   JsonConvert.DeserializeObject<Argument>(sr.ReadToEnd());
```

```
            }

        } catch (Exception ex) {
            Console.WriteLine($"We had problem: {ex.Message}");
        }

        return null;
    }

    public object Clone() {
        Argument arg = new(Number);
        Array.Copy(Variables, arg.Variables, Number);

        return arg;
    }
}
```

Interval.cs

```
namespace oMethods_3;

public record struct Interval {
    public double Left { get; init; }
    public double Right { get; init; }
    public double Center { get; init; }

    public Interval(double left, double right) {
        Left = left;
        Right = right;
        Center = (left + right) / 2;
    }

    public bool Contain(double point)
        => point ⩾ Left && point ⩽ Right;

    public override string ToString()
        => $"[{Left}; {Right}]";
}
```

SearchInterval.cs

```
namespace oMethods_3;

public static class SearcherInterval {
    // for report
    public static int Call = 0;
```

```
    //
    public static Interval Search(IFunction function, double x0, double
↪   delta, Argument direction, Argument argument) {
        double x, xk, xk1, h;

        if (function.Value(argument + x0 * direction) +
↪   function.PenaltyValue(argument + x0 * direction) >
            function.Value(argument + (x0 + delta) * direction) +
↪   function.PenaltyValue(argument + (x0 + delta) * direction)) {

            Call += 2;

            xk = x0 + delta;
            h = delta;
        } else {
            xk = x0 - delta;
            h = -delta;
        }

        do {
            h *= 2;
            xk1 = xk + h;
            x = x0;
            x0 = xk;
            xk = xk1;

            Call += 2;

        } while (function.Value(argument + x0 * direction) +
↪   function.PenaltyValue(argument + x0 * direction) >
                function.Value(argument + xk * direction) +
↪   function.PenaltyValue(argument + xk * direction));

        return (x < xk1) ? new(x, xk1) : new(xk1, x);
    }
}
```

Methods/MinMethods1D.cs

```
namespace oMethods_3;

public interface IMinMethod1D {
    public double? Min { get; }
    public double Eps { get; init; }

    // fore report
    public int Call { get; set; }
    //
```

```
10
11      public void Compute(Interval interval, IFunction function, Argument
    ↪   argument, Argument direction);
12  }
13
14  public class Fibonacci : IMinMethod1D {
15      private double? _min;
16      public double? Min => _min;
17      public double Eps { get; init; }
18
19      // for report
20      public int Call { get; set; }
21      //
22
23      public Fibonacci(double eps)
24          => (Eps, Call) = (eps, 0);
25
26      public void Compute(Interval interval, IFunction function, Argument
    ↪   argument, Argument direction) {
27          int n = 1;
28
29          while ((interval.Right - interval.Left) / Eps > Fib(n + 2))
30              n++;
31
32          double x1 = interval.Left + Fib(n) / Fib(n + 2) * (interval.Right
    ↪   - interval.Left);
33          double x2 = interval.Left + Fib(n + 1) / Fib(n + 2) *
    ↪   (interval.Right - interval.Left);
34          double y1 = function.Value(argument + x1 * direction) +
    ↪   function.PenaltyValue(argument + x1 * direction);
35          double y2 = function.Value(argument + x2 * direction) +
    ↪   function.PenaltyValue(argument + x2 * direction);
36
37          Call += 2;
38
39          for (int k = 1; k < n; k++) {
40              if (y1 < y2) {
41                  interval = new(interval.Left, x2);
42                  x2 = x1;
43                  y2 = y1;
44                  x1 = interval.Left + Fib(n - k + 1) / Fib(n - k + 3) *
    ↪   (interval.Right - interval.Left);
45                  y1 = function.Value(argument + x1 * direction) +
    ↪   function.PenaltyValue(argument + x1 * direction);
46                  Call++;
47              } else {
48                  interval = new(x1, interval.Right);
49                  x1 = x2;
50                  y1 = y2;
```

```
                x2 = interval.Left + Fib(n - k + 2) / Fib(n - k + 3) *
    (interval.Right - interval.Left);
                y2 = function.Value(argument + x2 * direction) +
    function.PenaltyValue(argument + x2 * direction);
                Call++;
            }
        }

        _min = interval.Center;
    }

    private static double Fib(int n)
        => (Math.Pow((1 + Math.Sqrt(5)) / 2, n) -
            Math.Pow((1 - Math.Sqrt(5)) / 2, n)) / Math.Sqrt(5);
}
```

Methods/MinMethodsND.cs

```
namespace oMethods_3;

public interface IMinMethodND {
    public Argument? Min { get; }
    public int MaxIters { get; init; }
    public double Eps { get; init; }
    public bool Need1DSearch { get; }

    public void Compute(Argument initPoint, IFunction function,
    IMinMethod1D method, (StrategyTypes, double)? strategy);
}

public class GaussAlgorithm : IMinMethodND {
    private Argument? _min;
    public Argument? Min => _min;
    public int MaxIters { get; init; }
    public double Eps { get; init; }
    public bool Need1DSearch => true;

    // for report
    public int Call { get; private set; }
    //

    public GaussAlgorithm(int maxIters, double eps) {
        MaxIters = maxIters;
        Eps = eps;
        Call = 0;
    }

    public void Compute(Argument initPoint, IFunction function,
    IMinMethod1D method, (StrategyTypes, double)? strategy) {
```

```
        Argument direction = new(initPoint.Number);
        Argument nextPoint;
        int iters;

        nextPoint = (Argument)initPoint.Clone();

        for (iters = 0; iters < MaxIters; iters++) {
            for (int i = 0; i < initPoint.Number; i++) {
                direction.Fill(0);
                direction[i] = 1;
                method.Compute(SearcherInterval.Search(function, 0, Eps,
↪   direction, nextPoint),
                               function, nextPoint, direction);
                nextPoint[i] = initPoint[i] + method.Min!.Value;
            }

            Call += 2;

            if (function.PenaltyValue(nextPoint) < Eps &&
                Math.Abs(function.Value(nextPoint) +
↪   function.PenaltyValue(nextPoint) -
                (function.Value(initPoint) +
↪   function.PenaltyValue(initPoint))) < Eps) {

                _min = (Argument)nextPoint.Clone();
                break;
            }

            initPoint = (Argument)nextPoint.Clone();

            if (strategy is not null) {
                function.Coef = strategy.Value.Item1 switch {
                    StrategyTypes.Mult ⇒ function.Coef *=
↪   strategy.Value.Item2,
                    StrategyTypes.Add ⇒ function.Coef == 0 ? 0 :
↪   function.Coef += strategy.Value.Item2,
                    StrategyTypes.Div ⇒ function.Coef /=
↪   strategy.Value.Item2,

                    _ ⇒ throw new InvalidEnumArgumentException($"This
↪   type of coefficient change strategy does not exist:
↪   {nameof(strategy.Value.Item1)}")
                };
            }
        }

        if (iters == MaxIters)
            _min = (Argument)nextPoint.Clone();

```

```
        Console.WriteLine($"Function evaluations: {Call + method.Call +
↪   SearcherInterval.Call}");
        Console.WriteLine($"Iterations: {iters}");
        Console.WriteLine(JsonConvert.SerializeObject(_min));
        Console.WriteLine($"f(extremum) = {function.Value(_min!)}");
    }
}

public class SimplexMethod : IMinMethodND {
    private Argument? _min;
    public Argument? Min ⇒ _min;
    public bool Need1DSearch ⇒ false;
    public int MaxIters { get; init; }
    public double Step { get; init; }
    public double Eps { get; init; }

    // for report
    public int Call { get; private set; }
    //

    public SimplexMethod(int maxIters, double step, double eps) {
        MaxIters = maxIters;
        Step = step;
        Eps = eps;
        Call = 0;
    }

    public void Compute(Argument initPoint, IFunction function,
↪   IMinMethod1D method, (StrategyTypes, double)? strategy) {

        Argument[] points = new Argument[initPoint.Number + 1];

        int iters;
        int n = initPoint.Number;

        Argument xG = new(n);
        Argument xR = new(n);
        Argument xC = new(n);
        Argument xE = new(n);

        double d1 = Step * (Math.Sqrt(n + 1) + n - 1) / (n * Math.Sqrt(2));
        double d2 = Step / (n * Math.Sqrt(2) * (Math.Sqrt(n + 1) - 1));

        points[^1] = (Argument)initPoint.Clone();

        for (int i = 0; i < points.Length - 1; i++)
            points[i] = new(n);

        for (int i = 0; i < n; i++)
```

```
            for (int j = 0; j < n; j++) {
                if (i == j) {
                    points[i][j] = d1;
                    continue;
                }

                points[i][j] = d2;
            }

        for (iters = 0; iters < MaxIters; iters++) {
            points = points.OrderBy(point => function.Value(point) +
↪ function.PenaltyValue(point)).ToArray();
            xG.Fill(0);

            Call += n + 1;

            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    xG[i] += points[j][i] / n;

            if (Criteria(points, xG, function) &&
↪ function.PenaltyValue(points[0]) < Eps) {
                _min = points[0];
                break;
            }

            if (iters != 0) {
                if (strategy is not null) {
                    function.Coef = strategy.Value.Item1 switch {
                        StrategyTypes.Mult => function.Coef *=
↪ strategy.Value.Item2,
                        StrategyTypes.Add => function.Coef == 0 ? 0 :
↪ function.Coef += strategy.Value.Item2,
                        StrategyTypes.Div => function.Coef /=
↪ strategy.Value.Item2,

                        _ => throw new InvalidEnumArgumentException($"This
↪ type of coefficient change strategy does not exist:
↪ {nameof(strategy.Value.Item1)}")
                    };
                }
            }

            Reflection(points, xG, xR);

            double valueXR = function.Value(xR) +
↪ function.PenaltyValue(xR);
            double valueBestPoint = function.Value(points[0]) +
↪ function.PenaltyValue(points[0]);
```

```
            double valueN = function.Value(points[n]) +
↪   function.PenaltyValue(points[n]);

            Call += 3;

            if (valueBestPoint ⩽ valueXR
                && valueXR < function.Value(points[n - 1]) +
↪   function.PenaltyValue(points[n - 1])) {
                points[n] = (Argument)xR.Clone();

                Call++;
            } else if (valueXR < valueBestPoint) {
                Expansion(xG, xR, xE);

                if (function.Value(xE) + function.PenaltyValue(xE) <
↪   valueXR) {
                    points[n] = (Argument)xE.Clone();

                    Call++;
                } else
                    points[n] = (Argument)xR.Clone();
            } else if (valueXR < valueN) {
                OutsideContraction(xG, xR, xC);

                if (function.Value(xC) + function.PenaltyValue(xC) <
↪   valueXR) {
                    points[n] = (Argument)xC.Clone();

                    Call++;
                } else
                    Shrink(points);
            } else {
                InsideContraction(points, xG, xC);

                if (function.Value(xC) + function.PenaltyValue(xC) <
↪   valueN) {
                    points[n] = (Argument)xC.Clone();

                    Call++;
                } else
                    Shrink(points);
            }
        }

        Console.WriteLine($"Function evaluations: {Call}");
        Console.WriteLine($"Iterations: {iters}");
        Console.WriteLine(JsonConvert.SerializeObject(_min = points[0]));
        Console.WriteLine($"f(extremum) = {function.Value(_min)}");
    }
```

```
202
203     private bool Criteria(Argument[] points, Argument xG, IFunction
    ↪   function) {
204         double sum = 0;
205         double valueXG = function.Value(xG) + function.PenaltyValue(xG);
206
207         Call++;
208
209         for (int i = 0; i < xG.Number + 1; i++) {
210             double valuePoint = function.Value(points[i]) +
    ↪   function.PenaltyValue(points[i]);
211
212             sum += (valuePoint - valueXG) *
213                    (valuePoint - valueXG);
214
215             Call++;
216         }
217
218         if (Math.Sqrt(sum / (xG.Number + 1)) < Eps)
219             return true;
220         else
221             return false;
222     }
223
224     private static void Reflection(Argument[] points, Argument xG,
    ↪   Argument xR) {
225         for (int i = 0; i < xG.Number; i++)
226             xR[i] = xG[i] - points[xG.Number][i] + xG[i];
227     }
228
229     private static void Expansion(Argument xG, Argument xR, Argument xE) {
230         for (int i = 0; i < xG.Number; i++)
231             xE[i] = xG[i] + 2 * (xR[i] - xG[i]);
232     }
233
234     private static void OutsideContraction(Argument xG, Argument xR,
    ↪   Argument xC) {
235         for (int i = 0; i < xG.Number; i++)
236             xC[i] = xG[i] + 0.5 * (xR[i] - xG[i]);
237     }
238
239     private static void InsideContraction(Argument[] points, Argument xG,
    ↪   Argument xC) {
240         for (int i = 0; i < xG.Number; i++)
241             xC[i] = xG[i] + 0.5 * (points[xG.Number][i] - xG[i]);
242     }
243
244     private static void Shrink(Argument[] points) {
245         for (int i = 1; i ⩽ points[0].Number; i++)
```

```
246             points[i] = (Argument)(points[0] + 0.5 * (points[i] -
    ↪ points[0])).Clone();
247     }
248 }
```

Functions/Function.cs

```
1  namespace oMethods_3;
2
3  public interface IFunction {
4      public double Coef { get; set; }
5      public int? Degree { get; init; }
6      public MethodTypes? MethodType { get; init; }
7      public double Value(Argument arg);
8      public double PenaltyValue(Argument arg);
9      public bool Limitation(Argument arg);
10 }
11
12 public class FunctionA : IFunction {
13     public double Coef { get; set; }
14     public int? Degree { get; init; }
15     public MethodTypes? MethodType { get; init; }
16
17     public FunctionA(double? coef, int? degree, MethodTypes? methodType)
18         => (Coef, Degree, MethodType) = (coef ?? 0, degree, methodType ??
    ↪ MethodTypes.Penalty);
19
20     public double Value(Argument arg)
21         => 5 * (arg[0] + arg[1]) * (arg[0] + arg[1]) +
22            (arg[0] - 2) * (arg[0] - 2);
23
24     // public double PenaltyValue(Argument arg) => MethodType switch {
25     //     MethodTypes.Penalty => Coef * Math.Pow(0.5 * (-arg[0] - arg[1]
    ↪ + 1 + Math.Abs(-arg[0] - arg[1] + 1)), Degree!.Value),
26
27     //     MethodTypes.InteriorPointLog => (-Coef * Math.Log(arg[0] +
    ↪ arg[1] - 1)).Equals(Double.NaN) ? 0
28     //                                     : -Coef * Math.Log(arg[0] +
    ↪ arg[1] - 1),
29
30     //     MethodTypes.InteriorPointReverse => -Coef / (arg[0] + arg[1] -
    ↪ 1),
31
32     //     _ => throw new InvalidEnumArgumentException($"This type of
    ↪ method does not exist: {nameof(MethodType)}")
33     // };
34
35     public double PenaltyValue(Argument arg) {
```

```
        switch (MethodType) {
            case MethodTypes.InteriorPointLog:
                //
↪   Console.WriteLine($"ValuePenaltyFunction={Math.Log(arg[0] + arg[1] -
↪   1)}");
                return -Coef * Math.Log(arg[0] + arg[1] - 1);
            case MethodTypes.InteriorPointReverse:
                // Console.WriteLine($"ValuePenaltyFunction={-1.0 /
↪   (-arg[0] - arg[1] + 1)}");
                return -Coef / (-arg[0] - arg[1] + 1);
            case MethodTypes.Penalty:
                return Coef * Math.Pow(0.5 * (-arg[0] - arg[1] + 1 +
↪   Math.Abs(-arg[0] - arg[1] + 1)), Degree!.Value);
            default:
                return 0;
        }

    }

    public bool Limitation(Argument arg)
        => -arg[0] - arg[1] + 1 ⩽ 0;
}

public class FunctionB : IFunction {
    public double Coef { get; set; }
    public int? Degree { get; init; }
    public MethodTypes? MethodType { get; init; }

    public FunctionB(double? coef, int? degree, MethodTypes? methodType)
        => (Coef, Degree, MethodType) = (coef ?? 0, degree, methodType ??
↪   MethodTypes.Penalty);

    public double Value(Argument arg)
        => 5 * (arg[0] + arg[1]) * (arg[0] + arg[1]) +
           (arg[0] - 2) * (arg[0] - 2);

    public double PenaltyValue(Argument arg)
         => Coef * Math.Pow(Math.Abs(arg[0] - arg[1]), Degree!.Value);

    public bool Limitation(Argument arg)
        => Math.Abs(arg[0] - arg[1]) < 1E-12;
}
```