МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

НГТУ НЭТИ

Кафедра прикладной математики

Практическая работа №4

по дисциплине «Численные методы»

ФПМИ

Группа	ПМ-92
Вариант	7
Студенты	Кутузов Иван
	Иванов Владислав
Преподаватель	Задорожный А. Г.
Дата	16.10.2021

Новосибирск

Цель работы

Разработать программу решения системы нелинейных уравнений (СНУ) методом Ньютона. Провести исследования метода для нескольких систем размерности от 2.

Вариант 7: Производные при формировании матрицы Якоби вычислять численно.

Анализ

Пусть дана СНУ в виде:

$$
\begin{aligned}
&F_1(x_1, x_2, \ldots, x_n) = 0;\\
&F_2(x_1, x_2, \ldots, x_n) = 0;\\
&\qquad \ldots\\
&F_m(x_1, x_2, \ldots, x_n) = 0.
\end{aligned}
$$

Обозначим через x^k решение, полученное на k-ой итерации процесса Ньютона. Запишем исходную систему в виде $F_i(x^k + \Delta x) = 0,\ i = 1\ldots m$, где $\Delta x = \bar{x} - x^k$, $\bar{x}$ - искомое решение. Выполним линеаризацию системы в окрестности точки x^k:

$$A^k \Delta x^k = -F^k,$$

где F^k - значение вектор-функции F при $x = x^k$; A^k - матрица Якоби:

$$\left(A_{ij}^k = \left.\frac{\partial\left(F_i(x)\right)}{\partial x_j}\right|_{x=x^k} \right).$$

Это система уравнений, линейных относительно приращений Δx_j^k. Решив эту систему, найдем направление Δx^k поиска решения.
Для поиска следующего приближения организуем итерационный процесс:

$$x_\nu^{k+1} = x^k + \beta_\nu^k \Delta x^k$$

где β^k - параметр итерационного процесса, $(0 < \beta^k \leq 1)$, ν - номер итерации поиска оптимального значения β^k. Параметр β^k будем искать следующим образом: после нахождения направления Δx^k принимаем β^k равным 1 и вычисляем значение:

$$F_\nu^k = F(x^k + \beta_\nu^k \Delta x^k)$$

Далее, пока норма F_ν^k больше, чем норма F^{k-1}, β^k уменьшается вдвое.

Заметим, что в СЛАУ матрица A^k при несовпадении числа неизвестных и числа уравнений становится прямоугольной. В этом случае формируют СЛАУ с квадратной матрицей, решение которой является решением изначальной СЛАУ.

Вариант 6:

$m \geq n$. Для нахождения Δx^k из системы исключаются $(m - n)$ уравнения, для которых абсолютные значения $F_i(x^k)$ минимальны.

Вариант 7:

$m \geq n$. Для нахождения Δx^k из системы для тех ее уравнений, для которых абсолютные значения $F_i(x^k)$ минимальны, производится свертка. Т.е вместо исключаемых уравнений берется уравнение, получающееся возведением в квадрат исключаемых уравнений и их сложением.

Вариант 8:

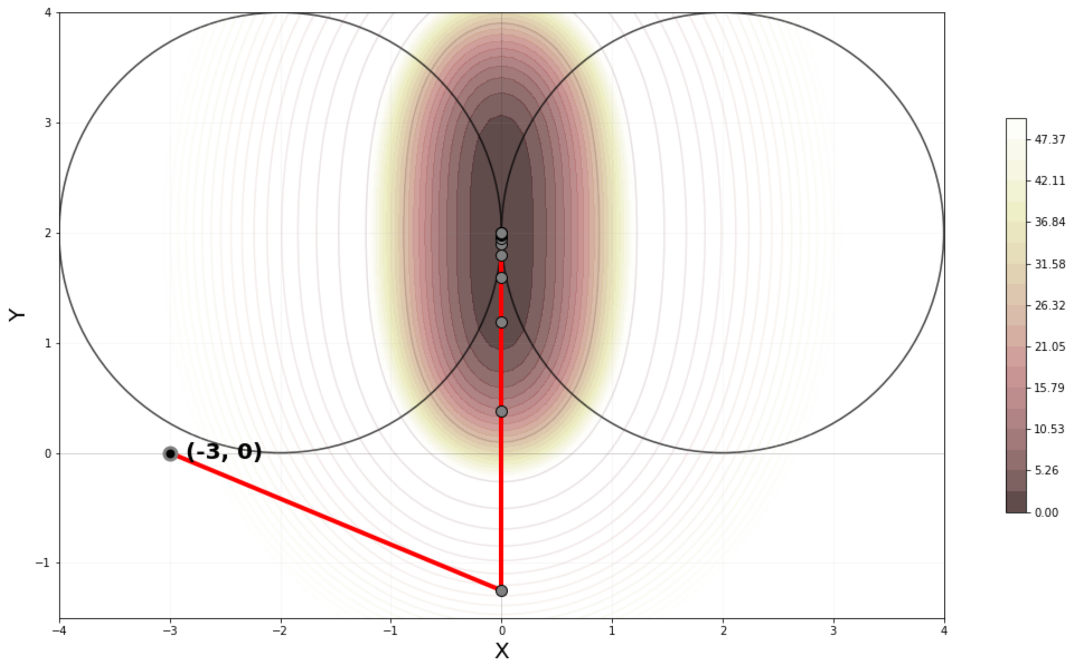
$m \geq n$. Для нахождения Δx^k из системы применяется процедура симметризации, заключающаяся в следующем. Вместо исходной системы решается система $(A^k)^T A^k \Delta x^k = -(A^k)^T F^k$.

Исследования

- Окружности пересекаются в точке

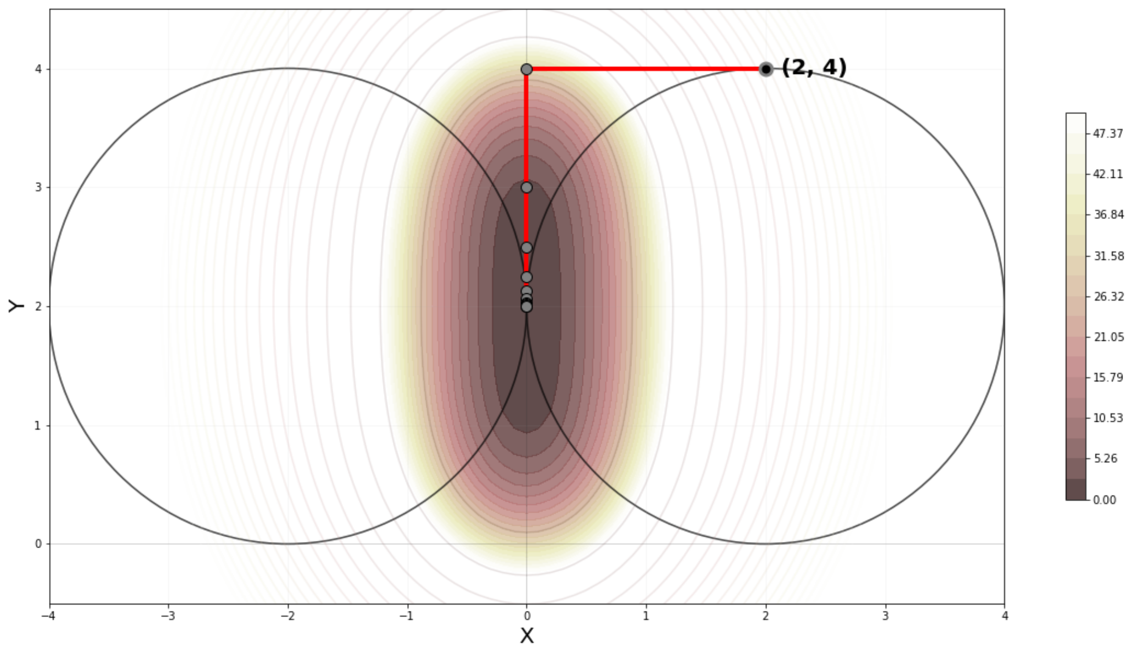
Начальное приближение (-3, 0)

k	x_1	x_2	β	Невязка
1	0.000000e+00	-1.250000e+00	1.000000e+00	5.970278e-01
2	0.000000e+00	3.750000e-01	1.000000e+00	1.492569e-01
3	0.000000e+00	1.187500e+00	1.000000e+00	3.731424e-02
4	0.000000e+00	1.593750e+00	1.000000e+00	9.328559e-03
5	0.000000e+00	1.796875e+00	1.000000e+00	2.332140e-03
6	0.000000e+00	1.898438e+00	1.000000e+00	5.830350e-04
7	0.000000e+00	1.949219e+00	1.000000e+00	1.457587e-04
8	0.000000e+00	1.974609e+00	1.000000e+00	3.643968e-05
9	0.000000e+00	1.987305e+00	1.000000e+00	9.109921e-06
10	0.000000e+00	1.993652e+00	1.000000e+00	2.277480e-06
11	0.000000e+00	1.996826e+00	1.000000e+00	5.693701e-07
12	0.000000e+00	1.998413e+00	1.000000e+00	1.423425e-07
13	0.000000e+00	1.999207e+00	1.000000e+00	3.558563e-08



Начальное приближение (2, 4)

k	x_1	x_2	β	Невязка
1	0.000000e+00	4.000000e+00	1.000000e+00	3.535534e-01
2	0.000000e+00	3.000000e+00	1.000000e+00	8.838835e-02
3	0.000000e+00	2.500000e+00	1.000000e+00	2.209709e-02
4	0.000000e+00	2.250000e+00	1.000000e+00	5.524272e-03
5	0.000000e+00	2.125000e+00	1.000000e+00	1.381068e-03
6	0.000000e+00	2.062500e+00	1.000000e+00	3.452670e-04
7	0.000000e+00	2.031250e+00	1.000000e+00	8.631675e-05
8	0.000000e+00	2.015625e+00	1.000000e+00	2.157919e-05
9	0.000000e+00	2.007812e+00	1.000000e+00	5.394797e-06
10	0.000000e+00	2.003906e+00	1.000000e+00	1.348699e-06
11	0.000000e+00	2.001953e+00	1.000000e+00	3.371748e-07
12	0.000000e+00	2.000977e+00	1.000000e+00	8.429370e-08

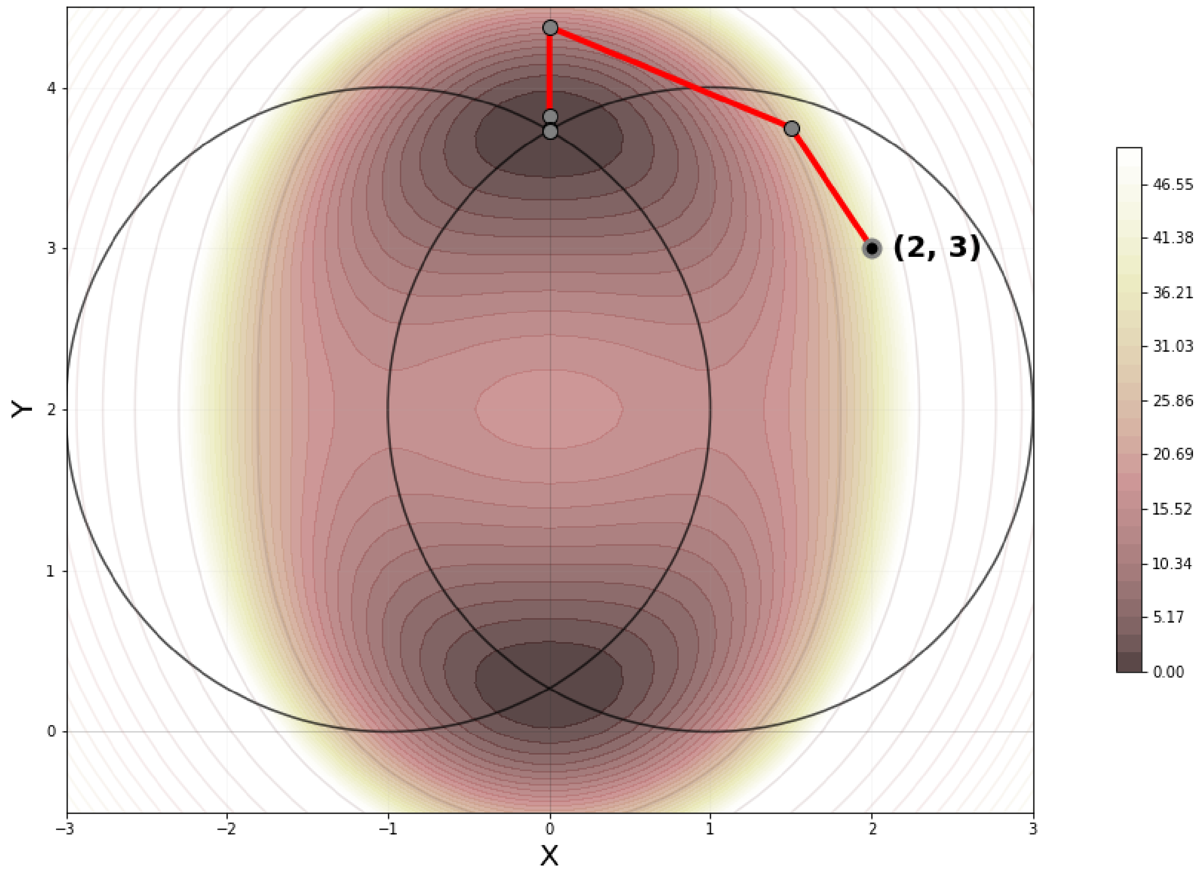


- Окружности пересекаются

Начальное приближение (2, 3)

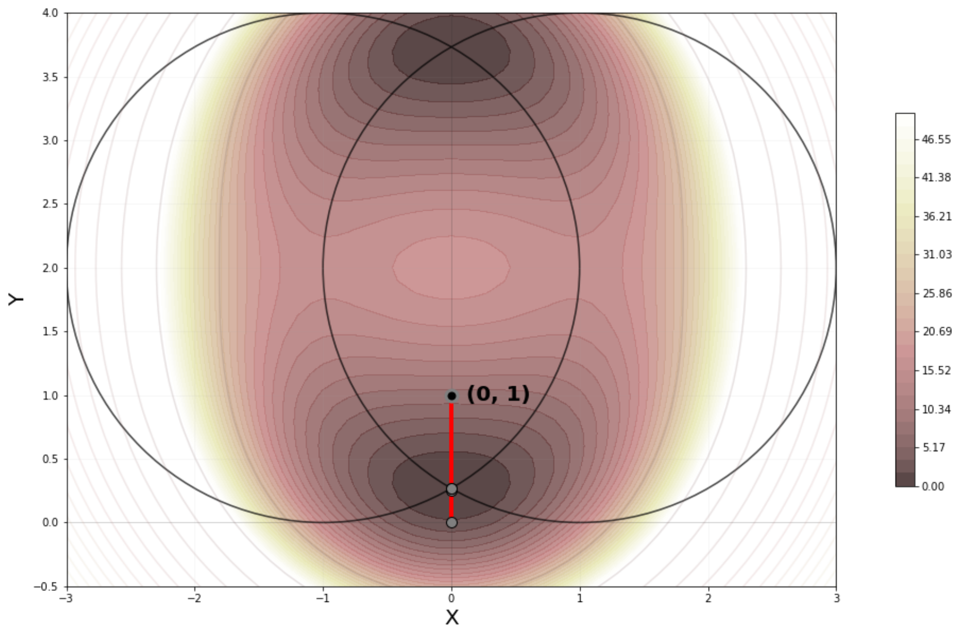
k	x_1	x_2	β	Невязка

1	1.500000e+00	3.750000e+00	2.500000e-01	8.469845e-01
2	0.000000e+00	4.375000e+00	1.000000e+00	5.904617e-01
3	0.000000e+00	3.819079e+00	1.000000e+00	6.910528e-02
4	0.000000e+00	3.734133e+00	1.000000e+00	1.613520e-03
5	0.000000e+00	3.732052e+00	1.000000e+00	9.679194e-07
6	0.000000e+00	3.732051e+00	1.000000e+00	3.491436e-13



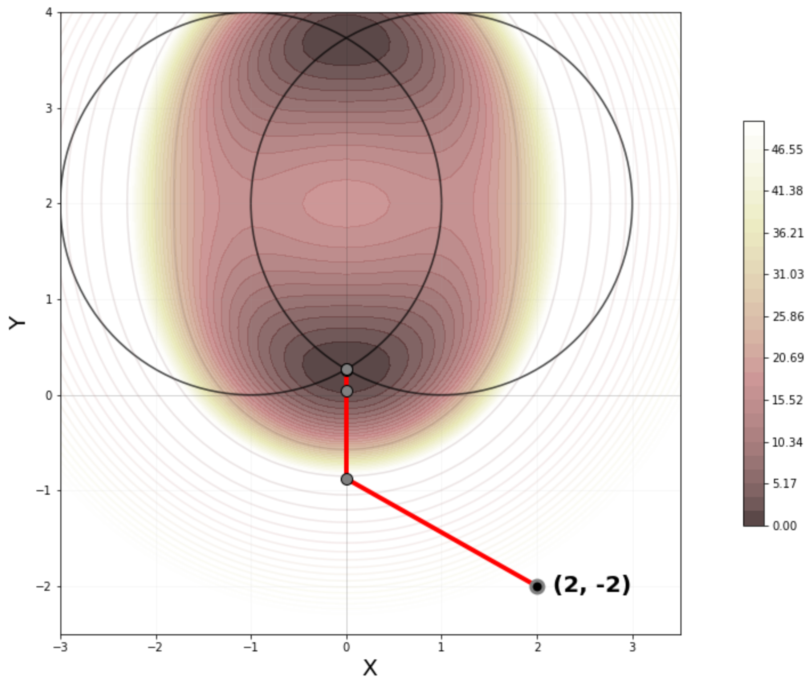
Начальное приближение (0, 1)

k	x_1	x_2	β	Невязка
1	0.000000e+00	0.000000e+00	1.000000e+00	5.000000e-01
2	0.000000e+00	2.500000e-01	1.000000e+00	3.125000e-02
3	0.000000e+00	2.678571e-01	1.000000e+00	1.594388e-04
4	0.000000e+00	2.679492e-01	1.000000e+00	4.236337e-09



Начальное приближение (2, -2)

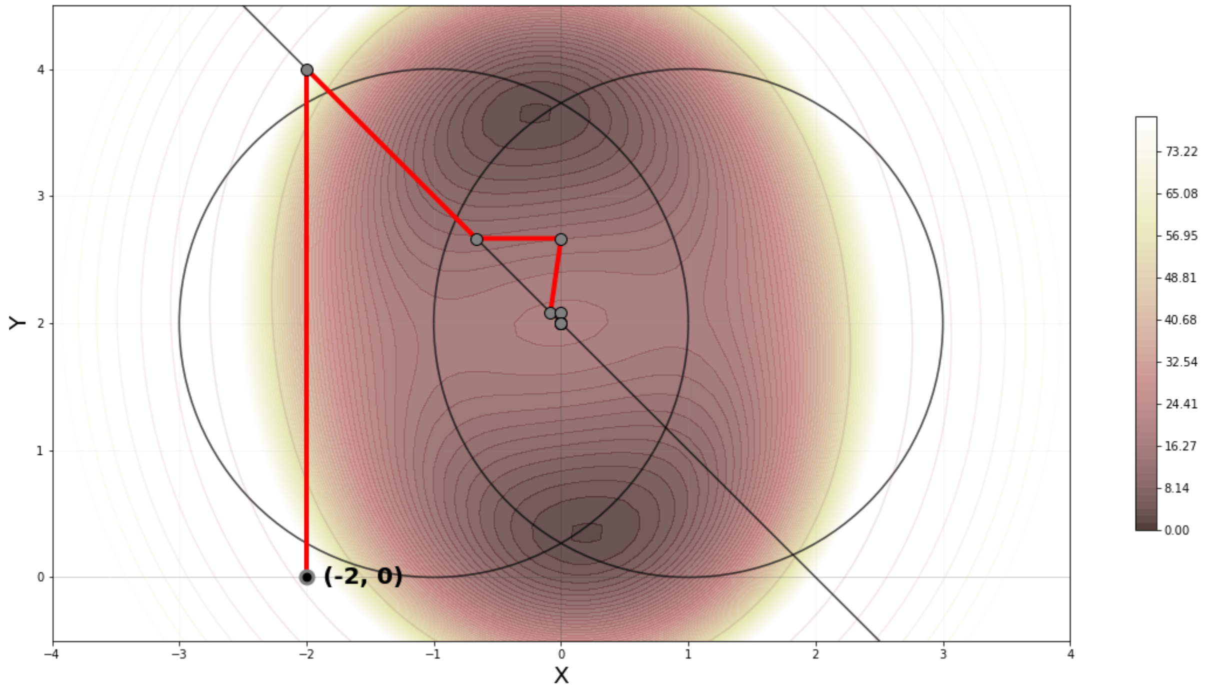
k	x_1	x_2	β	Невязка
1	-4.440892e-16	-8.750000e-01	1.000000e+00	3.015088e-01
2	-4.440892e-16	4.076087e-02	1.000000e+00	4.801913e-02
3	0.000000e+00	2.547771e-01	1.000000e+00	2.622669e-03
4	0.000000e+00	2.678995e-01	1.000000e+00	9.859961e-06
5	0.000000e+00	2.679492e-01	1.000000e+00	1.414797e-10



- Окружности пересекаются с прямой

Начальное приближение (-2, 0): Исключение строк

k	x_1	x_2	β	Невязка
1	-2.000000e+00	4.000000e+00	1.000000e+00	9.701425e-01
2	-6.666667e-01	2.666667e+00	1.000000e+00	2.410338e-01
3	0.000000e+00	2.666667e+00	1.000000e+00	5.555556e-02
4	-8.333333e-02	2.083333e+00	1.000000e+00	2.860790e-02
5	0.000000e+00	2.083333e+00	1.000000e+00	5.087794e-03
6	-1.666667e-03	2.001667e+00	1.000000e+00	5.716621e-04
7	-1.071192e-16	2.001667e+00	1.000000e+00	1.010568e-04
8	-6.938662e-07	2.000001e+00	1.000000e+00	2.379942e-07
9	7.976170e-17	2.000001e+00	1.000000e+00	4.205745e-08

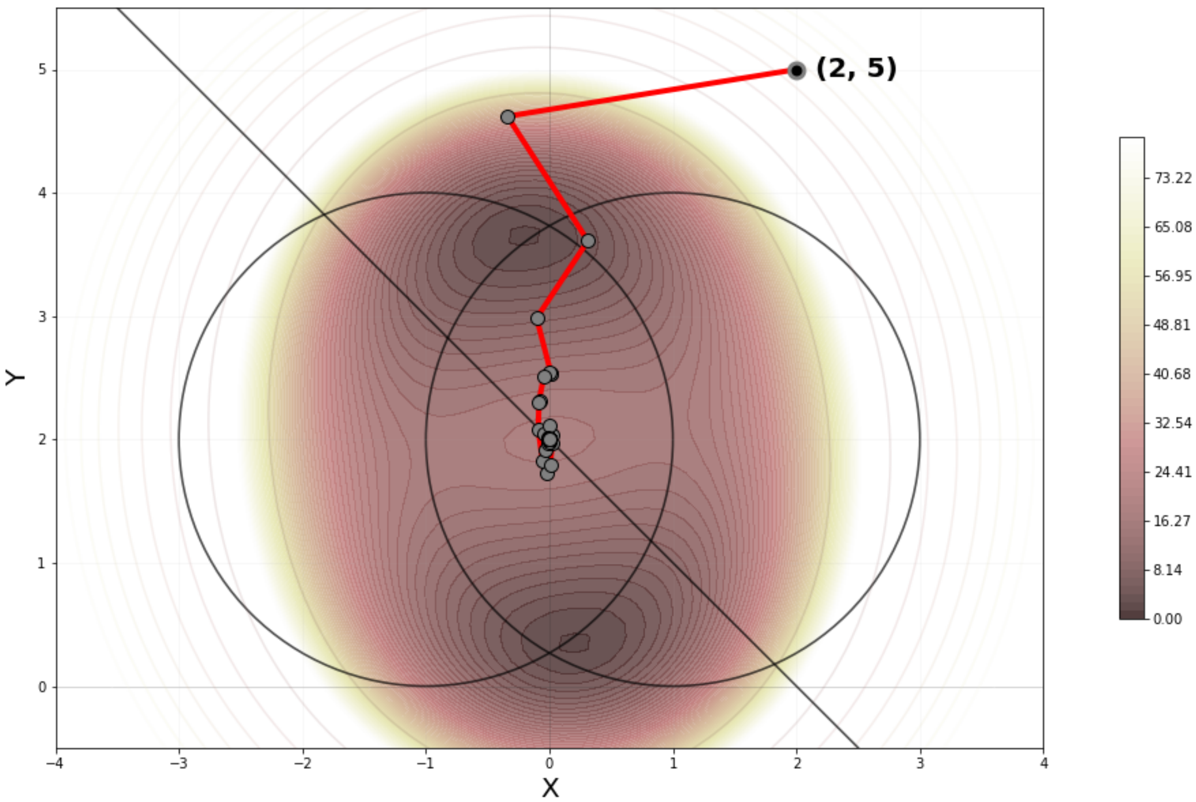


Начальное приближение (2, 5): Конволюция

k	x_1	x_2	β	Невязка
1	-3.400000e-01	4.620000e+00	1.000000e+00	4.653764e-01
2	3.115958e-01	3.610357e+00	1.000000e+00	2.081668e-01
3	-1.036440e-01	2.984101e+00	1.000000e+00	7.864176e-02
4	1.496882e-02	2.529506e+00	1.000000e+00	3.064469e-02

5	1.018552e-02	2.533226e+00	1.953125e-03	3.062222e-02
6	-4.160216e-03	2.539081e+00	1.562500e-02	3.046216e-02
7	-4.510816e-02	2.513164e+00	1.250000e-01	2.942186e-02
8	-8.243014e-02	2.316458e+00	5.000000e-01	2.451454e-02
9	-8.385528e-02	2.306406e+00	9.765625e-04	2.450785e-02
10	-9.078350e-02	2.081281e+00	2.500000e-01	2.319947e-02
11	-5.551679e-02	1.825866e+00	5.000000e-01	1.768698e-02
12	-1.636878e-02	1.730529e+00	5.000000e-01	1.433538e-02
13	-4.493768e-02	2.044938e+00	1.000000e+00	1.147505e-02
14	-3.644876e-02	1.920001e+00	2.500000e-01	1.069785e-02
15	6.130029e-03	1.795652e+00	1.000000e+00	9.464977e-03
16	2.525134e-02	1.974749e+00	1.000000e+00	6.446935e-03
17	2.287407e-02	2.037239e+00	1.250000e-01	6.440069e-03
18	-1.719040e-03	2.119026e+00	1.000000e+00	5.388582e-03
19	-1.050327e-02	2.010503e+00	1.000000e+00	2.681420e-03
20	-1.037997e-02	2.002691e+00	1.562500e-02	2.672538e-03
21	-7.828171e-03	1.975091e+00	2.500000e-01	2.485660e-03
22	-3.920381e-03	2.003920e+00	1.000000e+00	1.000837e-03
23	-3.914634e-03	2.002944e+00	1.953125e-03	1.000329e-03
24	-3.680029e-03	1.996167e+00	6.250000e-02	9.987860e-04
25	2.852838e-05	1.981403e+00	1.000000e+00	8.382919e-04
26	3.961266e-04	1.999604e+00	1.000000e+00	1.011273e-04
27	3.961220e-04	1.999612e+00	1.525879e-05	1.011267e-04
28	3.960312e-04	1.999642e+00	2.441406e-04	1.011174e-04
29	3.945063e-04	1.999765e+00	3.906250e-03	1.009707e-04
30	3.699102e-04	2.000267e+00	6.250000e-02	9.871782e-05
31	-2.206671e-07	2.002040e+00	1.000000e+00	9.205567e-05
32	-7.432914e-06	2.000007e+00	1.000000e+00	1.897551e-06
33	-7.432914e-06	2.000007e+00	5.960464e-08	1.897552e-06
34	-7.432688e-06	2.000007e+00	3.051758e-05	1.897536e-06
35	-7.425435e-06	2.000006e+00	9.765625e-04	1.897391e-06
36	-7.193412e-06	1.999998e+00	3.125000e-02	1.884875e-06

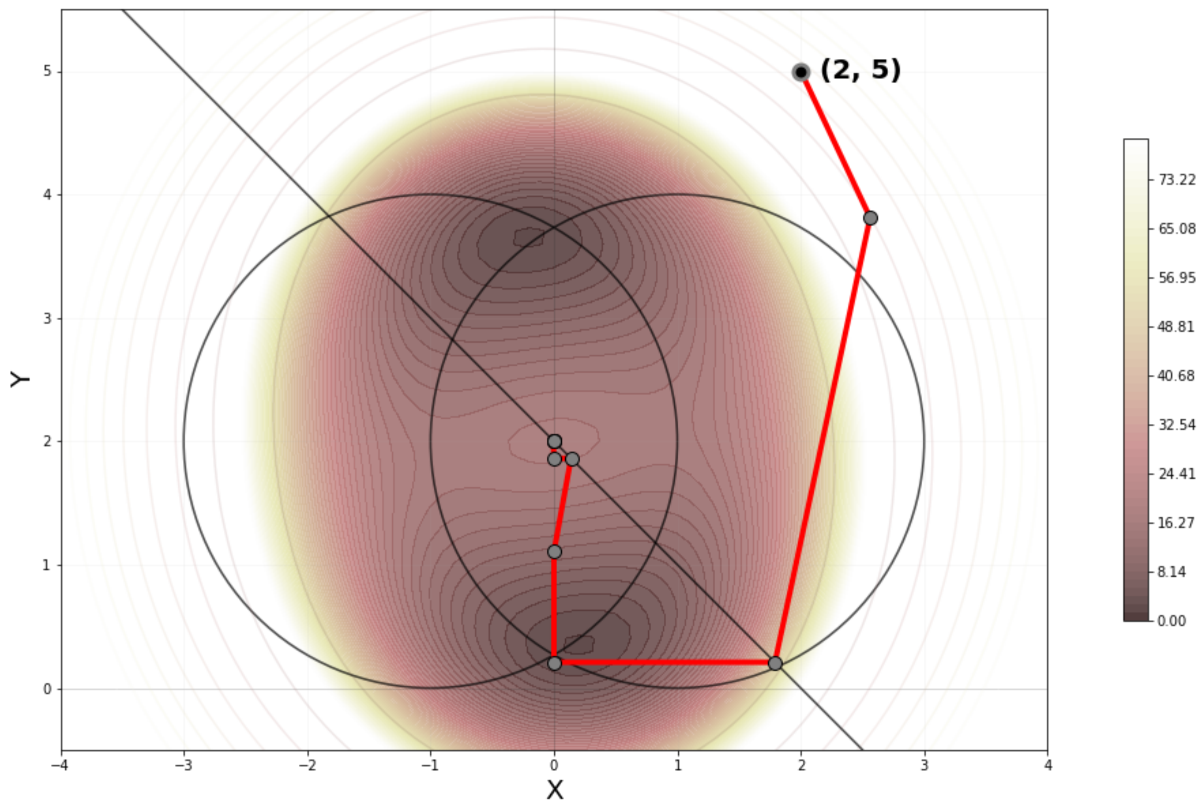
37	-3.596686e-06	1.999975e+00	5.000000e-01	1.600683e-06
38	-1.798343e-06	2.000002e+00	1.000000e+00	4.590995e-07
39	-1.798343e-06	2.000002e+00	5.960464e-08	4.591073e-07
40	-1.797465e-06	2.000002e+00	4.882812e-04	4.590387e-07
41	-1.769381e-06	2.000000e+00	1.562500e-02	4.586146e-07
42	-8.846908e-07	1.999992e+00	5.000000e-01	4.429808e-07
43	-4.423454e-07	2.000000e+00	1.000000e+00	1.129265e-07
44	-4.423454e-07	2.000000e+00	5.960464e-08	1.129585e-07
45	-4.388896e-07	2.000000e+00	7.812500e-03	1.126785e-07
46	-3.840285e-07	1.999999e+00	1.250000e-01	1.074816e-07
47	3.779393e-13	1.999998e+00	1.000000e+00	7.657208e-08



Начальное приближение (2, 5): Исключение строк

k	x_1	x_2	β	Невязка
1	2.562500e+00	3.812500e+00	1.250000e-01	9.285823e-01
2	1.791193e+00	2.088068e-01	1.000000e+00	6.138543e-01
3	2.220446e-16	2.088068e-01	1.000000e+00	2.201448e-01

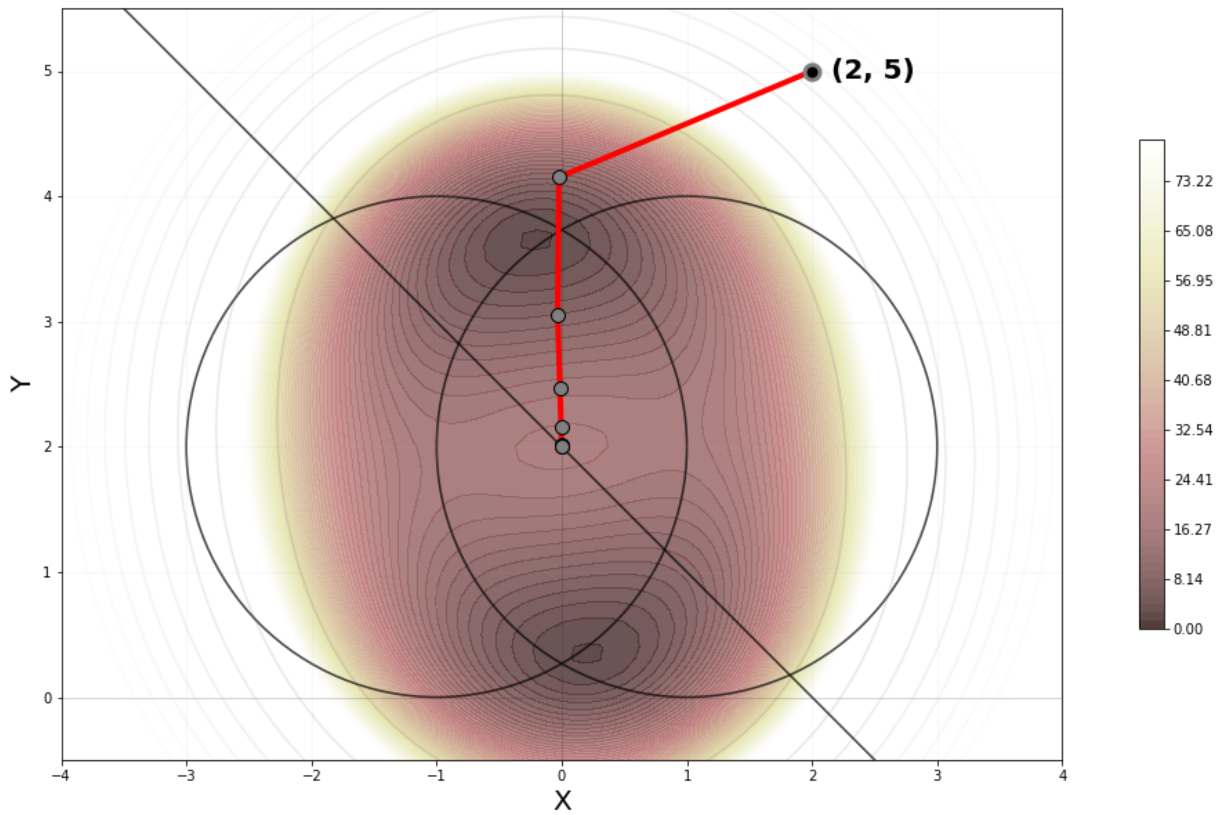
4	3.330669e-16	1.104403e+00	1.000000e+00	6.522404e-02
5	1.385022e-01	1.861498e+00	1.000000e+00	3.544296e-02
6	0.000000e+00	1.861498e+00	1.000000e+00	6.369293e-03
7	-5.152537e-03	2.005153e+00	1.000000e+00	1.315397e-03
8	1.040834e-16	2.005153e+00	1.000000e+00	2.325369e-04
9	-6.620104e-06	2.000007e+00	1.000000e+00	1.690048e-06
10	-3.104715e-17	2.000007e+00	1.000000e+00	2.987622e-07
11	-1.095648e-11	2.000000e+00	1.000000e+00	2.797105e-12



Начальное приближение (2, 5): Симметризация

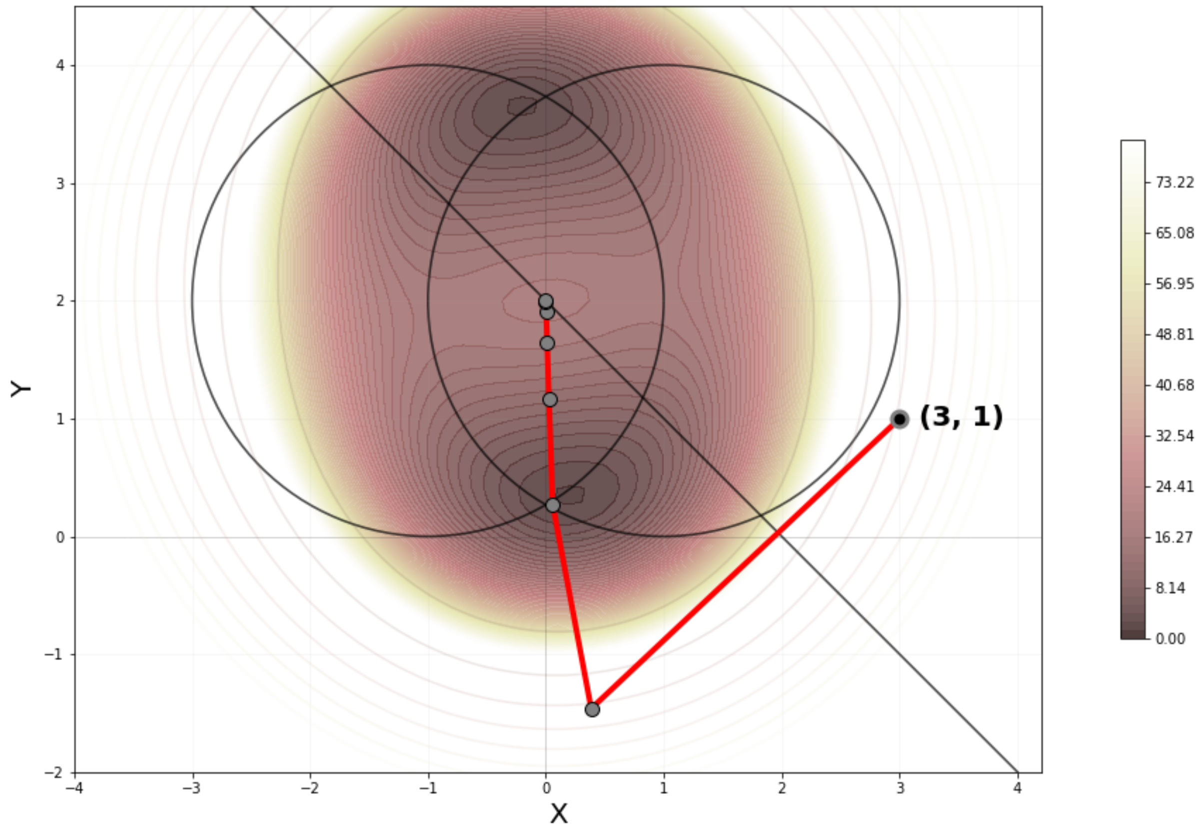
k	x_1	x_2	β	Невязка
1	-2.218430e-02	4.151877e+00	1.000000e+00	3.108523e-01
2	-3.208383e-02	3.048290e+00	1.000000e+00	8.425254e-02
3	-1.473193e-02	2.472154e+00	1.000000e+00	2.535967e-02
4	-4.777450e-03	2.153030e+00	1.000000e+00	6.963418e-03
5	-3.874061e-04	2.012409e+00	1.000000e+00	5.515599e-04
6	-2.462363e-07	2.000008e+00	1.000000e+00	3.505170e-07

7	-2.018692e-16	2.000000e+00	1.000000e+00	1.079266e-16



Начальное приближение (3, 1): Симметризация

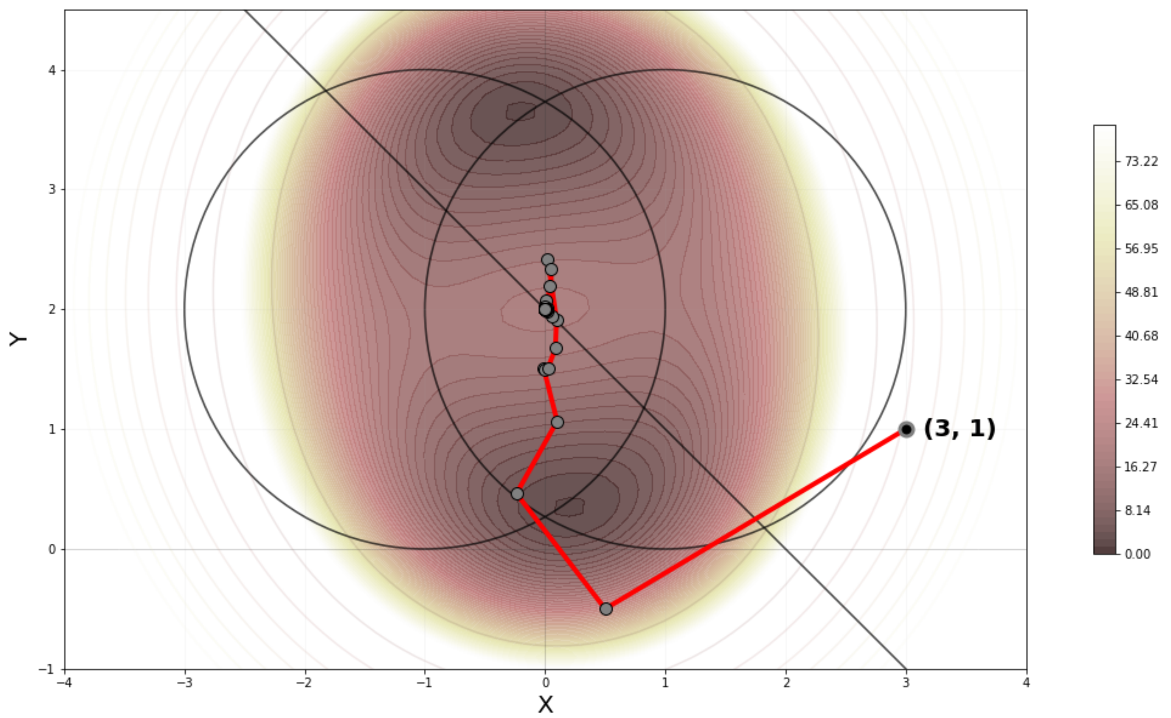
k	x_1	x_2	β	Невязка
1	3.846154e-01	-1.461538e+00	1.000000e+00	7.918328e-01
2	5.799558e-02	2.717317e-01	1.000000e+00	2.055180e-01
3	2.598677e-02	1.169436e+00	1.000000e+00	5.742739e-02
4	1.103120e-02	1.646680e+00	1.000000e+00	1.759319e-02
5	2.800879e-03	1.910285e+00	1.000000e+00	4.015972e-03
6	8.735985e-05	1.997202e+00	1.000000e+00	1.242312e-04
7	2.826911e-09	2.000000e+00	1.000000e+00	4.020011e-09



Начальное приближение (3, 1): Конволюция

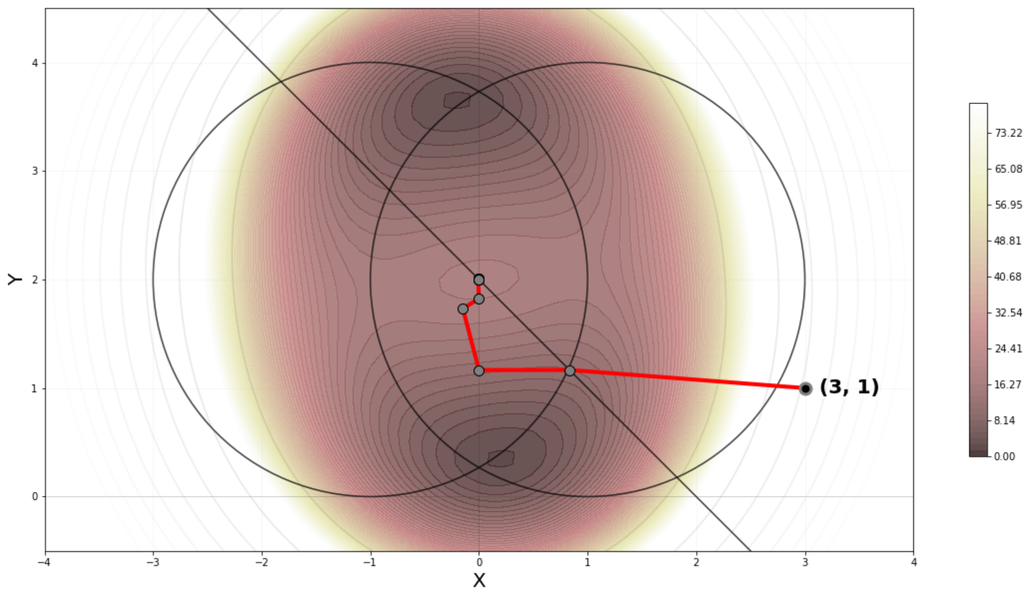
k	x_1	x_2	β	Невязка
1	5.000000e-01	-5.000000e-01	1.000000e+00	4.428749e-01
2	-2.381250e-01	4.618750e-01	1.000000e+00	1.842746e-01
3	1.001892e-01	1.066720e+00	1.000000e+00	7.224271e-02
4	-1.055352e-02	1.504233e+00	1.000000e+00	2.782255e-02
5	-2.151385e-04	1.497850e+00	7.812500e-03	2.777433e-02
6	2.367140e-02	1.505362e+00	6.250000e-02	2.705060e-02
7	8.663422e-02	1.677882e+00	5.000000e-01	2.551853e-02
8	9.534950e-02	1.911992e+00	2.500000e-01	2.434298e-02
9	2.168113e-02	2.420542e+00	1.000000e+00	2.357708e-02
10	4.995737e-02	2.336988e+00	1.250000e-01	2.283417e-02
11	5.482448e-02	1.945176e+00	1.000000e+00	1.398718e-02
12	3.407745e-02	2.194988e+00	5.000000e-01	1.372656e-02
13	2.176551e-02	1.978234e+00	1.000000e+00	5.551202e-03
14	2.074155e-02	2.009481e+00	6.250000e-02	5.462495e-03

15	1.010769e-02	2.074528e+00	5.000000e-01	4.618929e-03
16	5.374852e-03	1.994625e+00	1.000000e+00	1.370757e-03
17	5.359091e-03	1.996578e+00	3.906250e-03	1.369523e-03
18	4.712343e-03	2.009828e+00	1.250000e-01	1.368972e-03
19	-6.897987e-05	2.020114e+00	1.000000e+00	9.042463e-04
20	-2.532111e-04	2.000253e+00	1.000000e+00	6.457661e-05
21	-2.532097e-04	2.000249e+00	7.629395e-06	6.457647e-05
22	-2.531511e-04	2.000223e+00	2.441406e-04	6.457524e-05
23	-2.511882e-04	2.000091e+00	7.812500e-03	6.446646e-05
24	-1.884323e-04	1.999263e+00	2.500000e-01	6.365043e-05
25	-9.421632e-05	2.000094e+00	1.000000e+00	2.402805e-05
26	-9.421625e-05	2.000094e+00	9.536743e-07	2.402804e-05
27	-9.421354e-05	2.000090e+00	3.051758e-05	2.402801e-05
28	-9.412230e-05	2.000073e+00	9.765625e-04	2.402241e-05
29	-9.118497e-05	1.999965e+00	3.125000e-02	2.394199e-05
30	-4.558850e-05	1.999687e+00	5.000000e-01	1.989608e-05
31	-2.279427e-05	2.000023e+00	1.000000e+00	5.813239e-06
32	-2.279427e-05	2.000023e+00	5.960464e-08	5.813239e-06
33	-2.279419e-05	2.000023e+00	3.814697e-06	5.813232e-06
34	-2.279142e-05	2.000021e+00	1.220703e-04	5.813195e-06
35	-2.270248e-05	2.000013e+00	3.906250e-03	5.807893e-06
36	-1.986501e-05	1.999960e+00	1.250000e-01	5.748606e-06
37	1.160262e-09	1.999917e+00	1.000000e+00	3.720695e-06
38	4.328442e-09	2.000000e+00	1.000000e+00	1.103886e-09



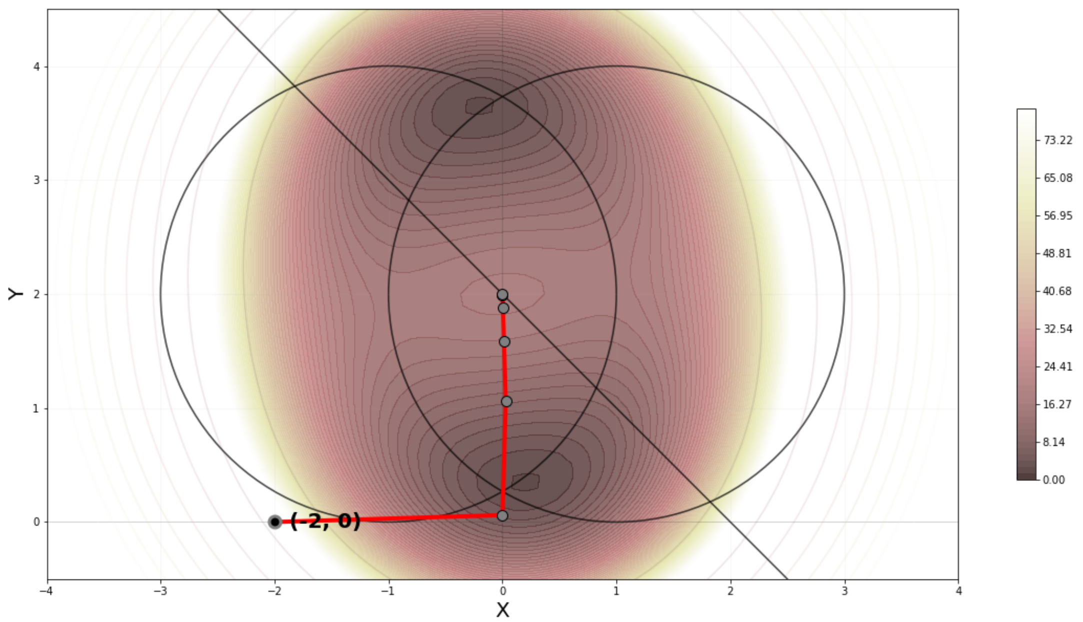
Начальное приближение (3, 1): Исключение строк

k	x_1	x_2	β	Невязка
1	8.333333e-01	1.166667e+00	1.000000e+00	2.302360e-01
2	-1.110223e-16	1.166667e+00	1.000000e+00	5.806764e-02
3	-1.488095e-01	1.732143e+00	5.000000e-01	4.276660e-02
4	0.000000e+00	1.824735e+00	1.000000e+00	8.140635e-03
5	-8.417018e-03	2.008417e+00	1.000000e+00	2.146617e-03
6	7.979728e-17	2.008417e+00	1.000000e+00	3.794954e-04
7	-1.763732e-05	2.000018e+00	1.000000e+00	4.498058e-06
8	-1.510090e-16	2.000018e+00	1.000000e+00	7.951523e-07
9	-7.776829e-11	2.000000e+00	1.000000e+00	1.983331e-11



Начальное приближение (-2, 0): Симметризация

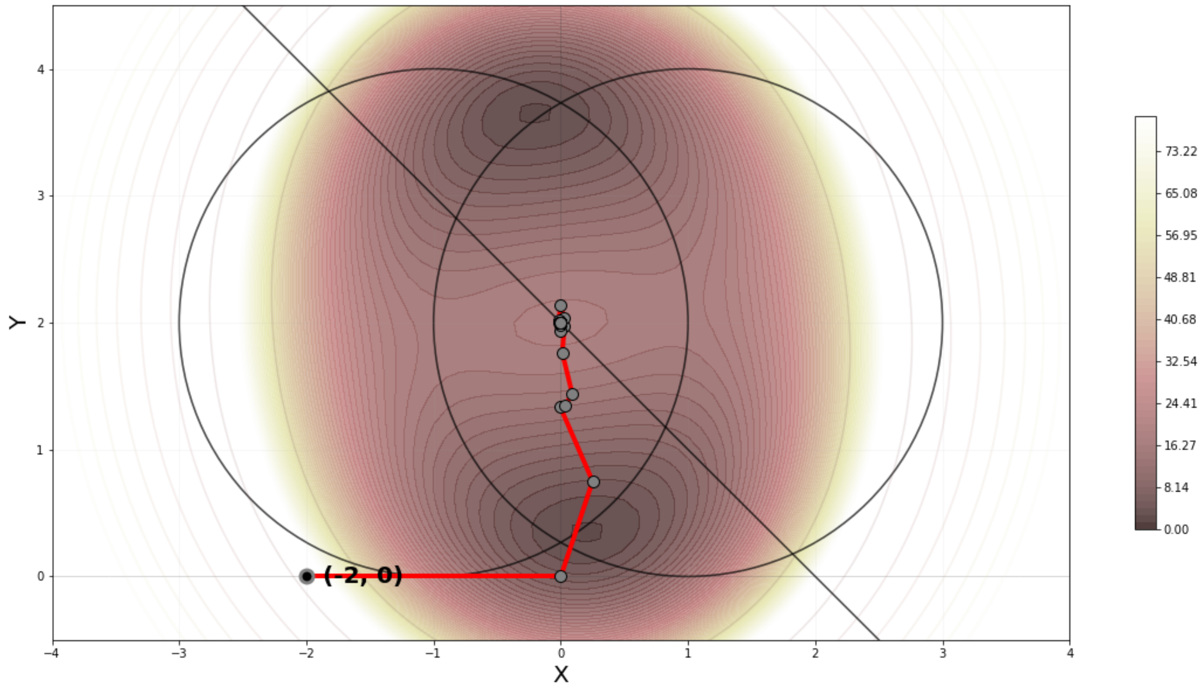
k	x_1	x_2	β	Невязка
1	0.000000e+00	6.060606e-02	1.000000e+00	3.432931e-01
2	2.846735e-02	1.060577e+00	1.000000e+00	9.425206e-02
3	1.288474e-02	1.586930e+00	1.000000e+00	2.868444e-02
4	3.772751e-03	1.879151e+00	1.000000e+00	7.323840e-03
5	2.031960e-04	1.993491e+00	1.000000e+00	3.886403e-04
6	3.556308e-08	1.999999e+00	1.000000e+00	6.801623e-08



Начальное приближение (-2, 0): Конволюция

k	x_1	x_2	β	Невязка
1	0.000000e+00	0.000000e+00	1.000000e+00	3.638034e-01
2	2.500000e-01	7.500000e-01	1.000000e+00	1.744872e-01
3	-5.809295e-03	1.339543e+00	1.000000e+00	5.509335e-02
4	3.189636e-02	1.343479e+00	6.250000e-02	5.409766e-02
5	8.909752e-02	1.442434e+00	2.500000e-01	4.988148e-02
6	1.221885e-02	1.759880e+00	1.000000e+00	1.526720e-02
7	2.762585e-02	1.972374e+00	1.000000e+00	9.476494e-03
8	2.502401e-02	2.034862e+00	1.250000e-01	9.320986e-03
9	-1.913815e-03	2.137579e+00	1.000000e+00	8.410222e-03
10	-1.579023e-02	2.015790e+00	1.000000e+00	5.416174e-03
11	-1.541918e-02	2.000166e+00	3.125000e-02	5.369028e-03
12	-7.744404e-03	1.935558e+00	5.000000e-01	5.132671e-03
13	-4.102864e-03	2.004103e+00	1.000000e+00	1.407274e-03
14	-4.096850e-03	2.003126e+00	1.953125e-03	1.406442e-03
15	-3.974561e-03	1.999484e+00	3.125000e-02	1.390195e-03
16	-1.985362e-03	1.984521e+00	5.000000e-01	1.259149e-03
17	-9.956873e-04	2.000996e+00	1.000000e+00	3.415180e-04
18	-9.955962e-04	2.000935e+00	1.220703e-04	3.415067e-04
19	-9.918992e-04	2.000524e+00	3.906250e-03	3.414012e-04
20	-8.684978e-04	1.998369e+00	1.250000e-01	3.342183e-04
21	2.150543e-06	1.996315e+00	1.000000e+00	2.233357e-04
22	9.091009e-06	1.999991e+00	1.000000e+00	3.118190e-06
23	9.091008e-06	1.999991e+00	5.960464e-08	3.118192e-06
24	9.090732e-06	1.999991e+00	3.051758e-05	3.118185e-06
25	9.081861e-06	1.999993e+00	9.765625e-04	3.117501e-06
26	8.798090e-06	2.000003e+00	3.125000e-02	3.105934e-06
27	4.399009e-06	2.000030e+00	5.000000e-01	2.585908e-06
28	2.199504e-06	1.999998e+00	1.000000e+00	7.544238e-07
29	2.199504e-06	1.999998e+00	5.960464e-08	7.544323e-07
30	2.198430e-06	1.999998e+00	4.882812e-04	7.544004e-07

31	2.164082e-06	2.000000e+00	1.562500e-02	7.519078e-07
32	1.623062e-06	2.000005e+00	2.500000e-01	6.808675e-07
33	-9.197820e-12	2.000006e+00	1.000000e+00	3.936567e-07
34	-2.270808e-11	2.000000e+00	1.000000e+00	7.788800e-12

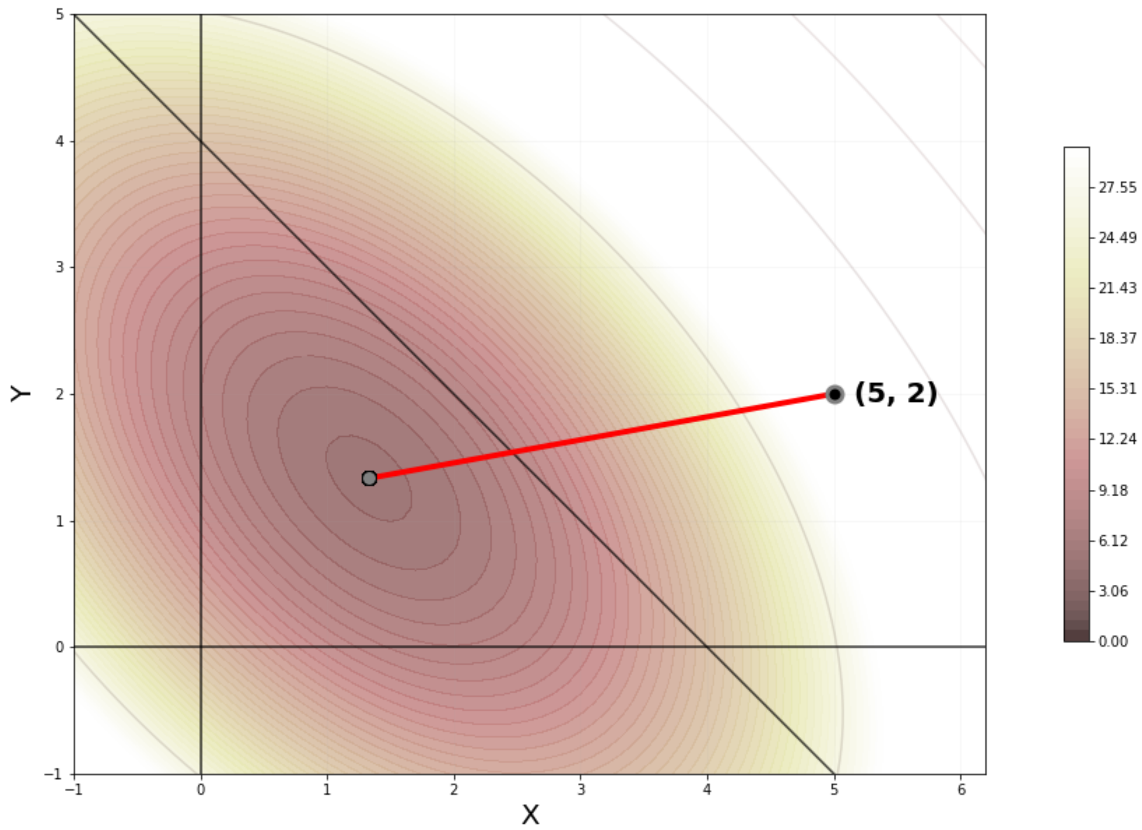


- Три прямые

Начальное приближение (5, 2): Симметризация

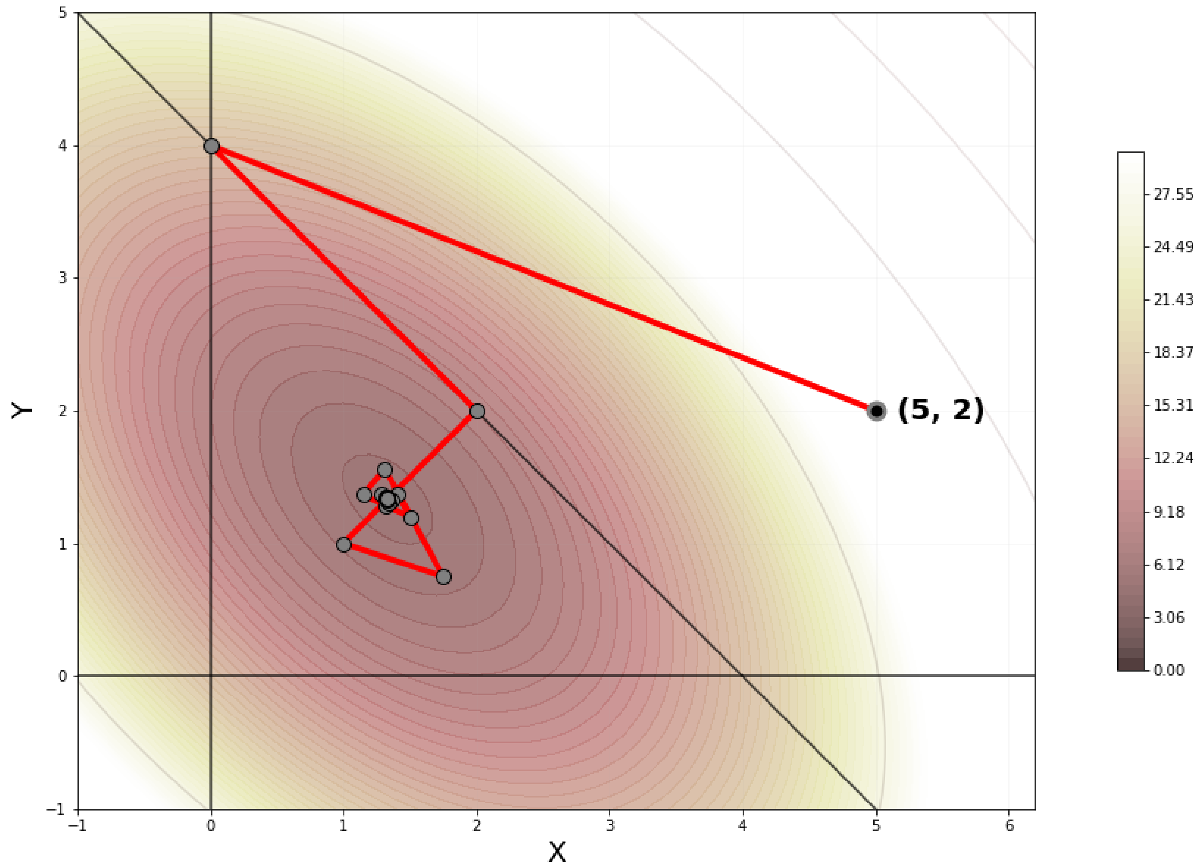
k	x_1	x_2	β	Невязка
1	1.333333e+00	1.333333e+00	1.000000e+00	3.746343e-01
2	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
3	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
4	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
5	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
...	...	...	...	...
996	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
997	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
998	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01
999	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01

1000	1.333333e+00	1.333333e+00	5.960464e-08	3.746343e-01



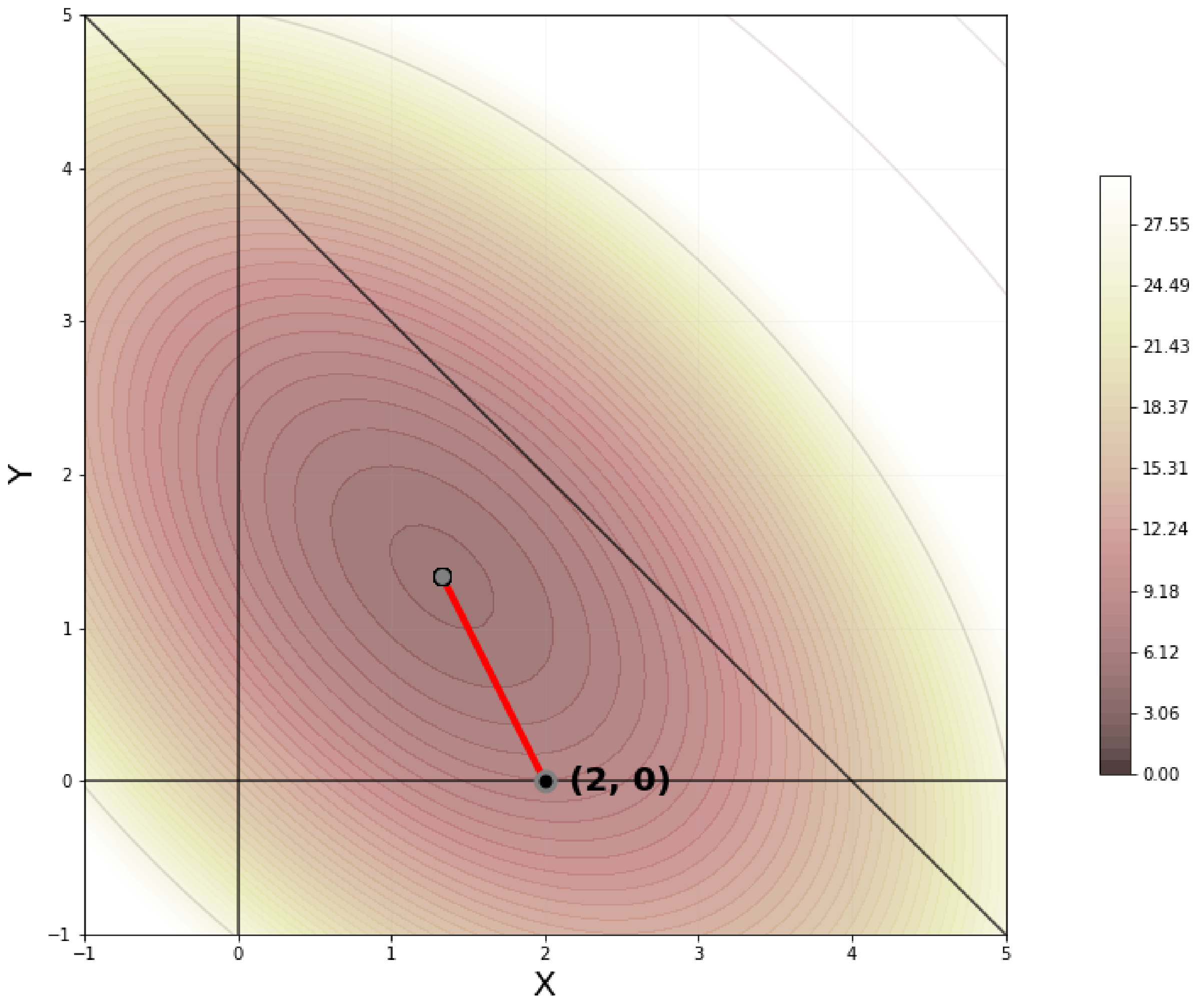
Начальное приближение (5, 2): Исключение строк

k	x_1	x_2	β	Невязка
1	0.000000e+00	4.000000e+00	1.000000e+00	6.488857e-01
2	2.000000e+00	2.000000e+00	5.000000e-01	4.588315e-01
3	1.000000e+00	1.000000e+00	5.000000e-01	3.973597e-01
4	1.750000e+00	7.500000e-01	2.500000e-01	3.931988e-01
5	1.312500e+00	1.562500e+00	2.500000e-01	3.780033e-01
...	...	...	...	...
996	1.333333e+00	1.333333e+00	1.192093e-07	3.746343e-01
997	1.333334e+00	1.333333e+00	5.960464e-08	3.746343e-01
998	1.333333e+00	1.333333e+00	1.192093e-07	3.746343e-01
999	1.333333e+00	1.333333e+00	1.192093e-07	3.746343e-01
1000	1.333334e+00	1.333333e+00	5.960464e-08	3.746343e-01



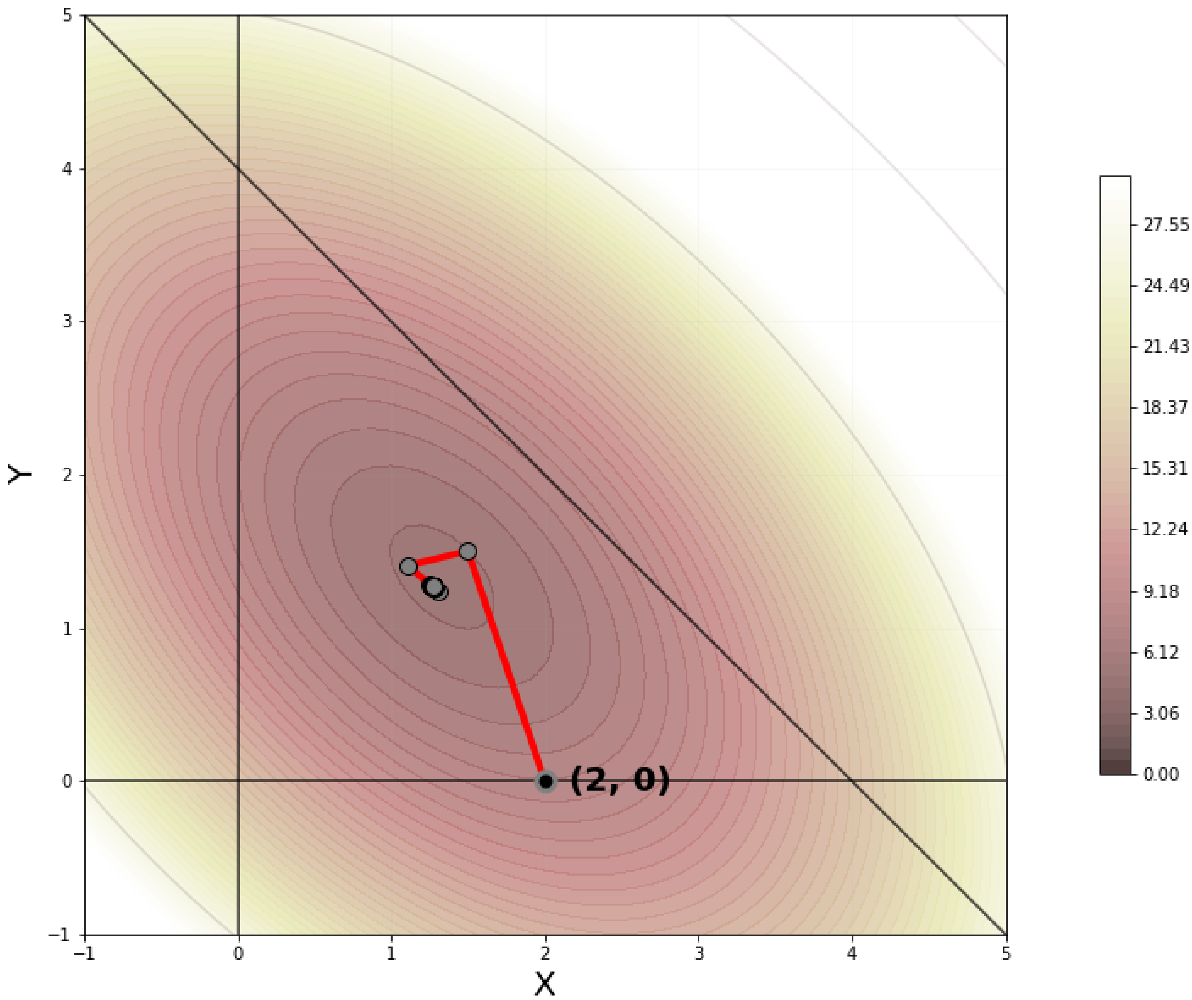
Начальное приближение (2, 0): Симметризация

k	x_1	x_2	β	Невязка
1	1.333333e+00	1.333333e+00	1.000000e+00	8.164966e-01
2	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
3	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
4	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
5	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
...	...	...	...	...
996	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
997	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
998	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
999	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01
1000	1.333333e+00	1.333333e+00	5.960464e-08	8.164966e-01



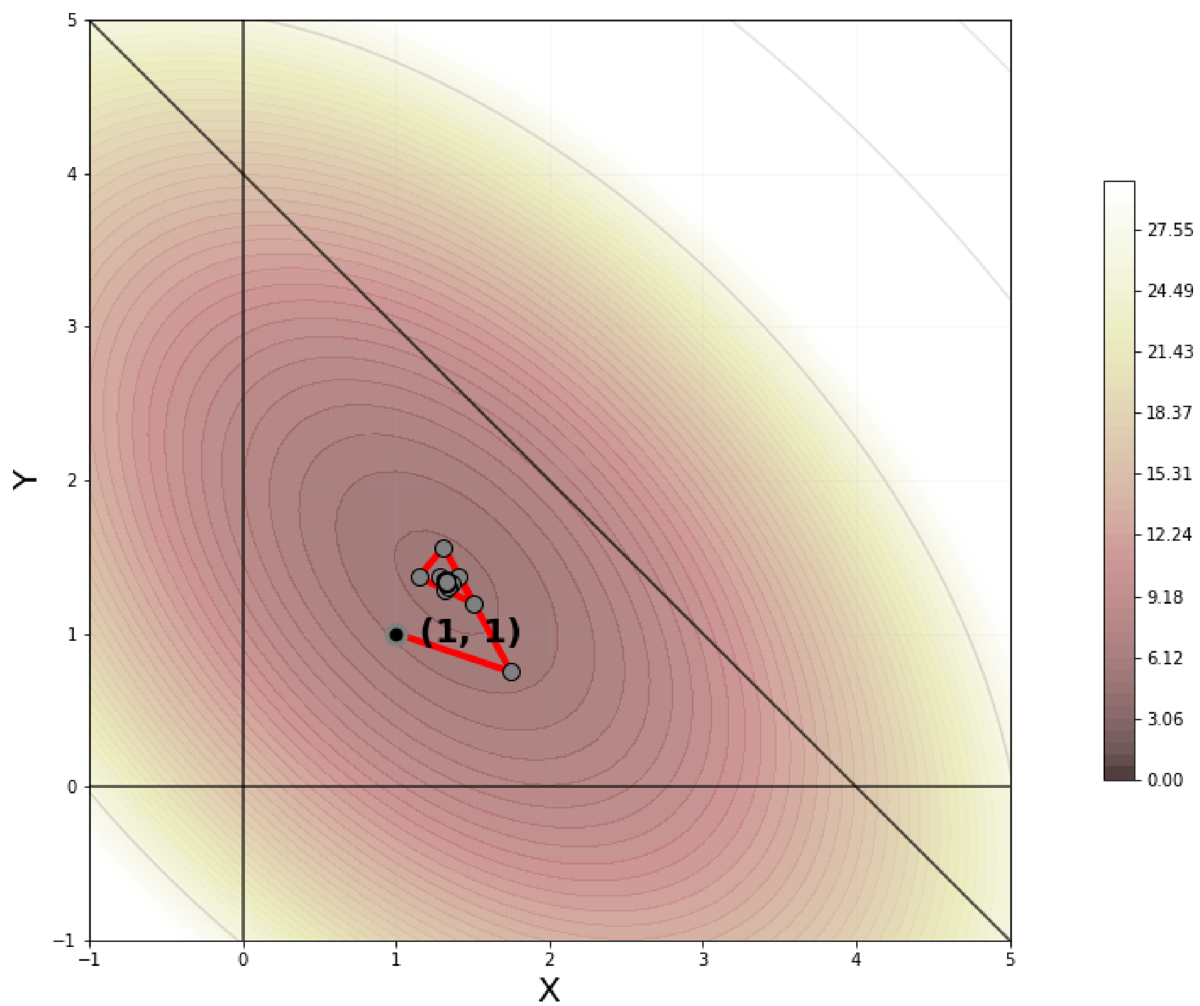
Начальное приближение (2, 0): Конволюция

k	x_1	x_2	β	Невязка
1	1.500000e+00	1.500000e+00	5.000000e-01	8.291562e-01
2	1.109375e+00	1.406250e+00	6.250000e-02	8.224674e-01
3	1.303666e+00	1.235152e+00	1.562500e-02	8.185504e-01
4	1.254956e+00	1.285289e+00	9.765625e-04	8.183647e-01
5	1.283044e+00	1.257558e+00	2.441406e-04	8.183441e-01
...	...	...	...	...
996	1.270777e+00	1.271816e+00	4.768372e-07	8.182623e-01
997	1.271573e+00	1.271020e+00	2.384186e-07	8.182622e-01
998	1.270824e+00	1.271769e+00	5.960464e-08	8.182623e-01
999	1.271699e+00	1.270894e+00	2.384186e-07	8.182622e-01
1000	1.271186e+00	1.271408e+00	1.192093e-07	8.182622e-01



Начальное приближение (1, 1): Исключение строк

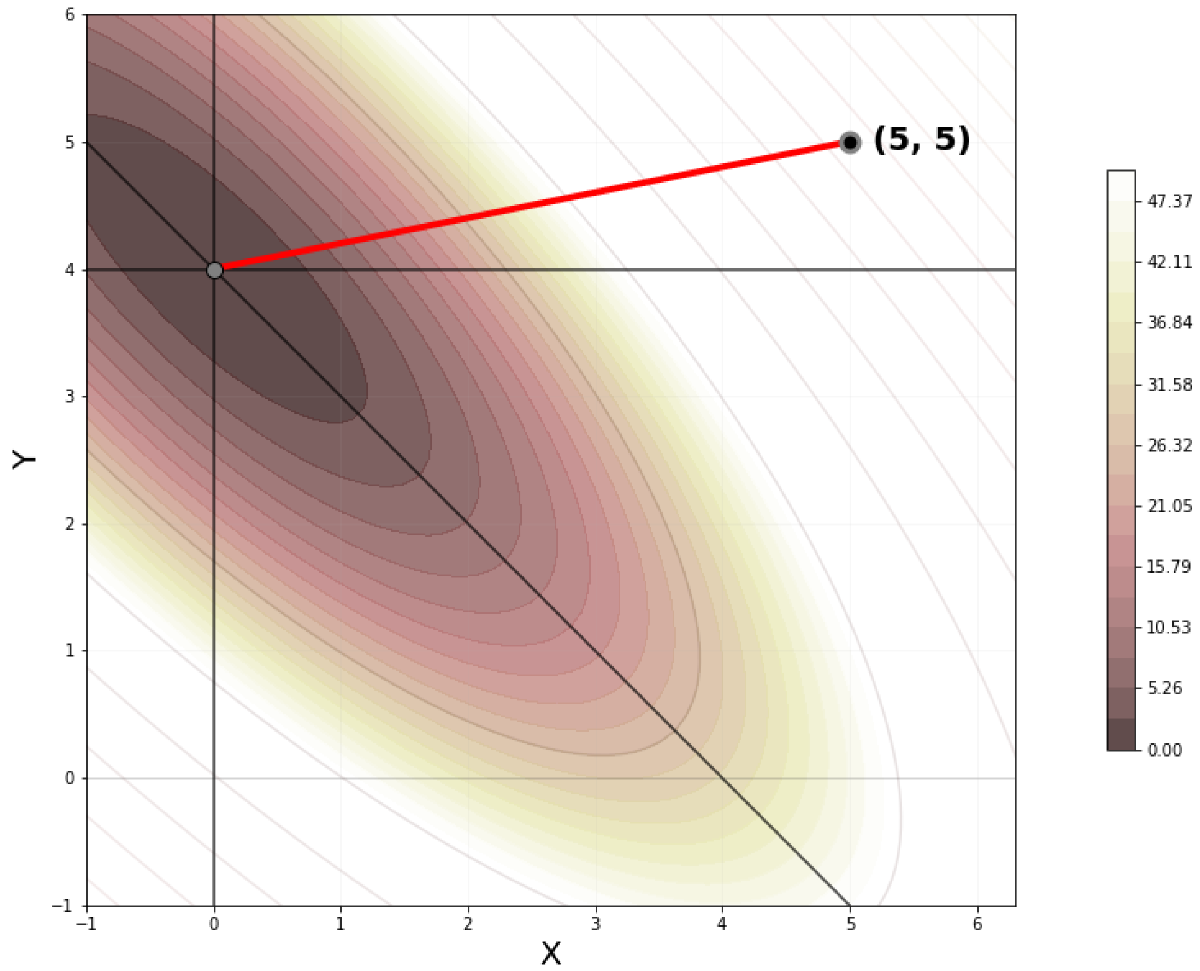
k	x_1	x_2	β	Невязка
1	1.750000e+00	7.500000e-01	2.500000e-01	9.895285e-01
2	1.312500e+00	1.562500e+00	2.500000e-01	9.512875e-01
3	1.148438e+00	1.367188e+00	1.250000e-01	9.479346e-01
4	1.504883e+00	1.196289e+00	1.250000e-01	9.471654e-01
5	1.410828e+00	1.371521e+00	6.250000e-02	9.446498e-01
...	...	...	...	...
996	1.333333e+00	1.333333e+00	1.192093e-07	9.428090e-01
997	1.333334e+00	1.333333e+00	5.960464e-08	9.428090e-01
998	1.333333e+00	1.333333e+00	1.192093e-07	9.428090e-01
999	1.333333e+00	1.333333e+00	1.192093e-07	9.428090e-01
1000	1.333334e+00	1.333333e+00	5.960464e-08	9.428090e-01



- Взвешенные прямые

Начальное приближение (5, 5): Симметризация

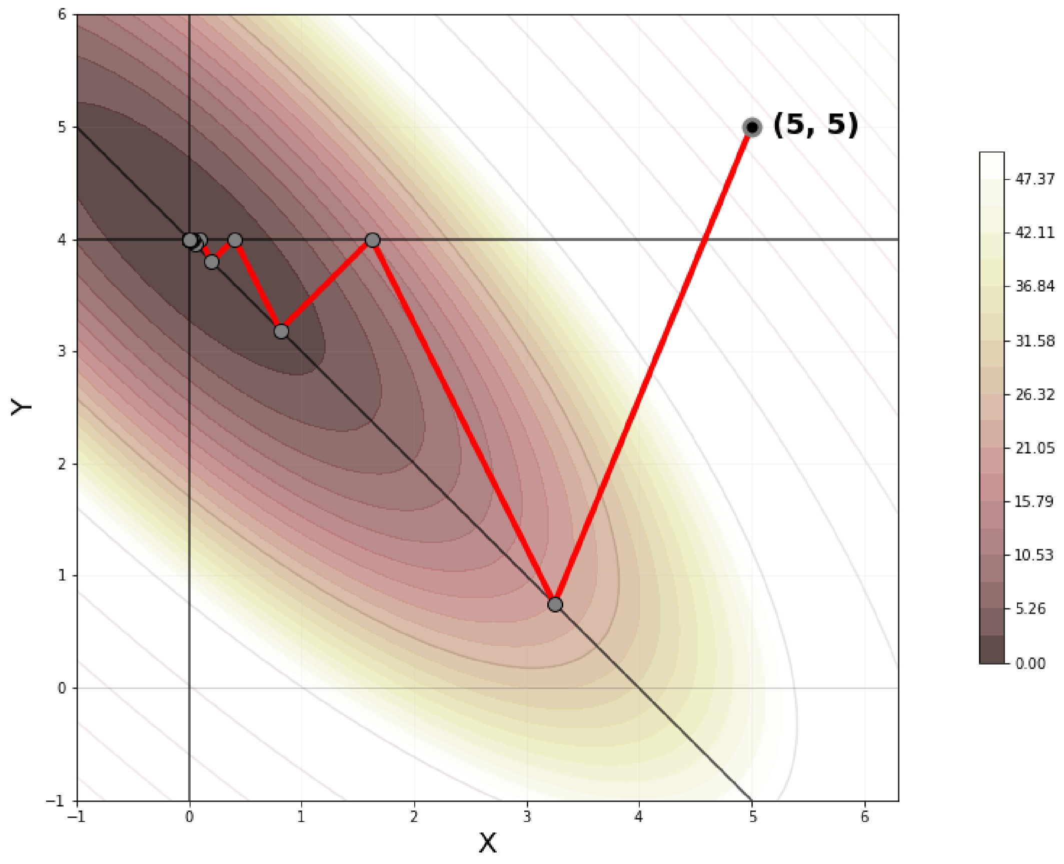
k	x_1	x_2	β	Невязка
1	8.881784e-16	4.000000e+00	1.000000e+00	7.616067e-17



Начальное приближение (5, 5): Конволюция

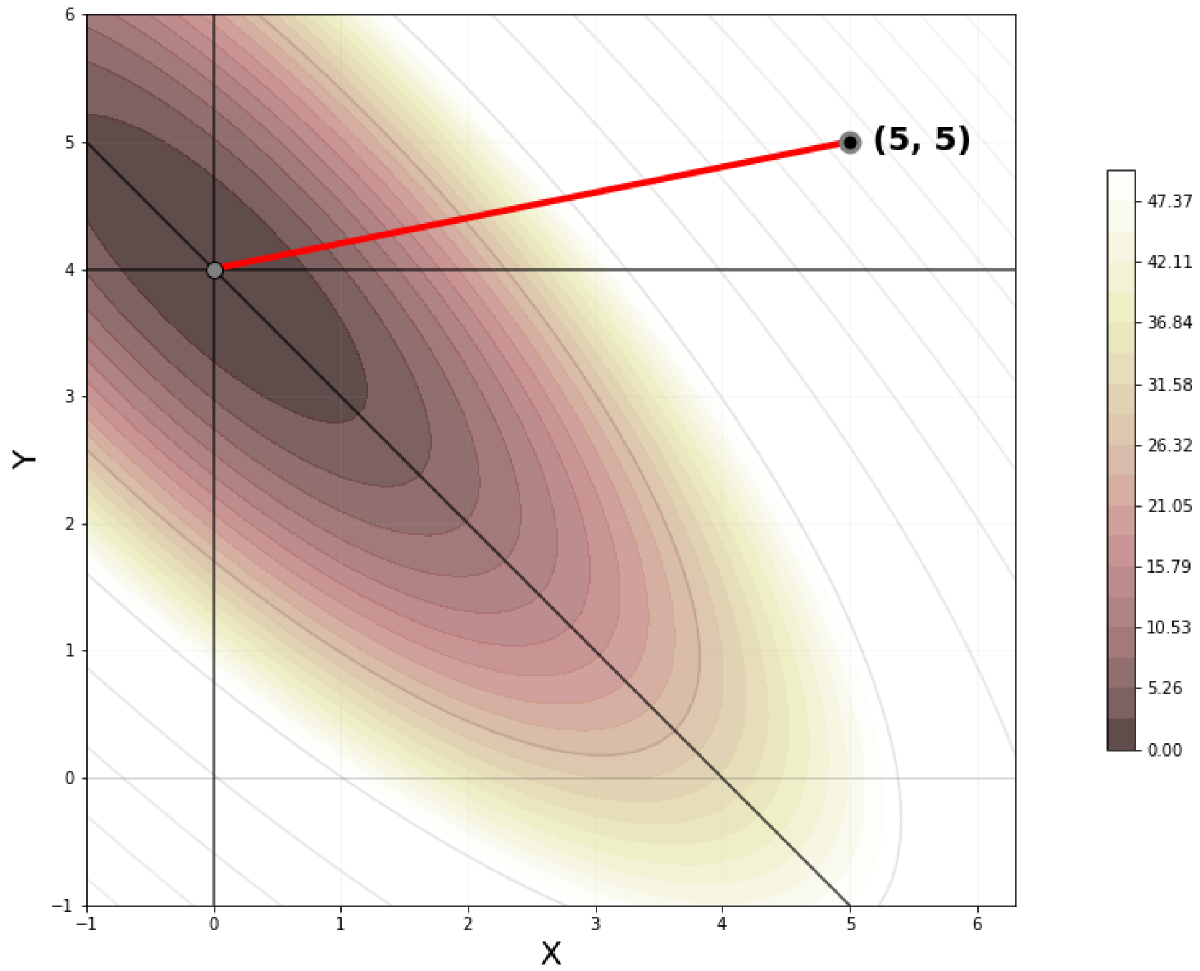
k	x_1	x_2	β	Невязка
1	3.250000e+00	7.500000e-01	1.000000e+00	3.525120e-01
2	1.625000e+00	4.000000e+00	1.000000e+00	2.786852e-01
3	8.125000e-01	3.187500e+00	1.000000e+00	8.812800e-02
4	4.062500e-01	4.000000e+00	1.000000e+00	6.967130e-02
5	2.031250e-01	3.796875e+00	1.000000e+00	2.203200e-02
6	1.015625e-01	4.000000e+00	1.000000e+00	1.741783e-02
7	5.078125e-02	3.949219e+00	1.000000e+00	5.508000e-03
8	2.539062e-02	4.000000e+00	1.000000e+00	4.354456e-03
9	1.269531e-02	3.987305e+00	1.000000e+00	1.377000e-03
10	6.347656e-03	4.000000e+00	1.000000e+00	1.088614e-03
11	3.173828e-03	3.996826e+00	1.000000e+00	3.442500e-04
12	1.586914e-03	4.000000e+00	1.000000e+00	2.721535e-04

13	7.934570e-04	3.999207e+00	1.000000e+00	8.606250e-05
14	3.967285e-04	4.000000e+00	1.000000e+00	6.803838e-05
15	1.983643e-04	3.999802e+00	1.000000e+00	2.151562e-05
16	9.918213e-05	4.000000e+00	1.000000e+00	1.700959e-05
17	4.959106e-05	3.999950e+00	1.000000e+00	5.378906e-06
18	2.479553e-05	4.000000e+00	1.000000e+00	4.252399e-06
19	1.239777e-05	3.999988e+00	1.000000e+00	1.344727e-06
20	6.198883e-06	4.000000e+00	1.000000e+00	1.063100e-06
21	3.099442e-06	3.999997e+00	1.000000e+00	3.361816e-07
22	1.549721e-06	4.000000e+00	1.000000e+00	2.657749e-07
23	7.748604e-07	3.999999e+00	1.000000e+00	8.404541e-08



Начальное приближение (5, 5): Исключение строк

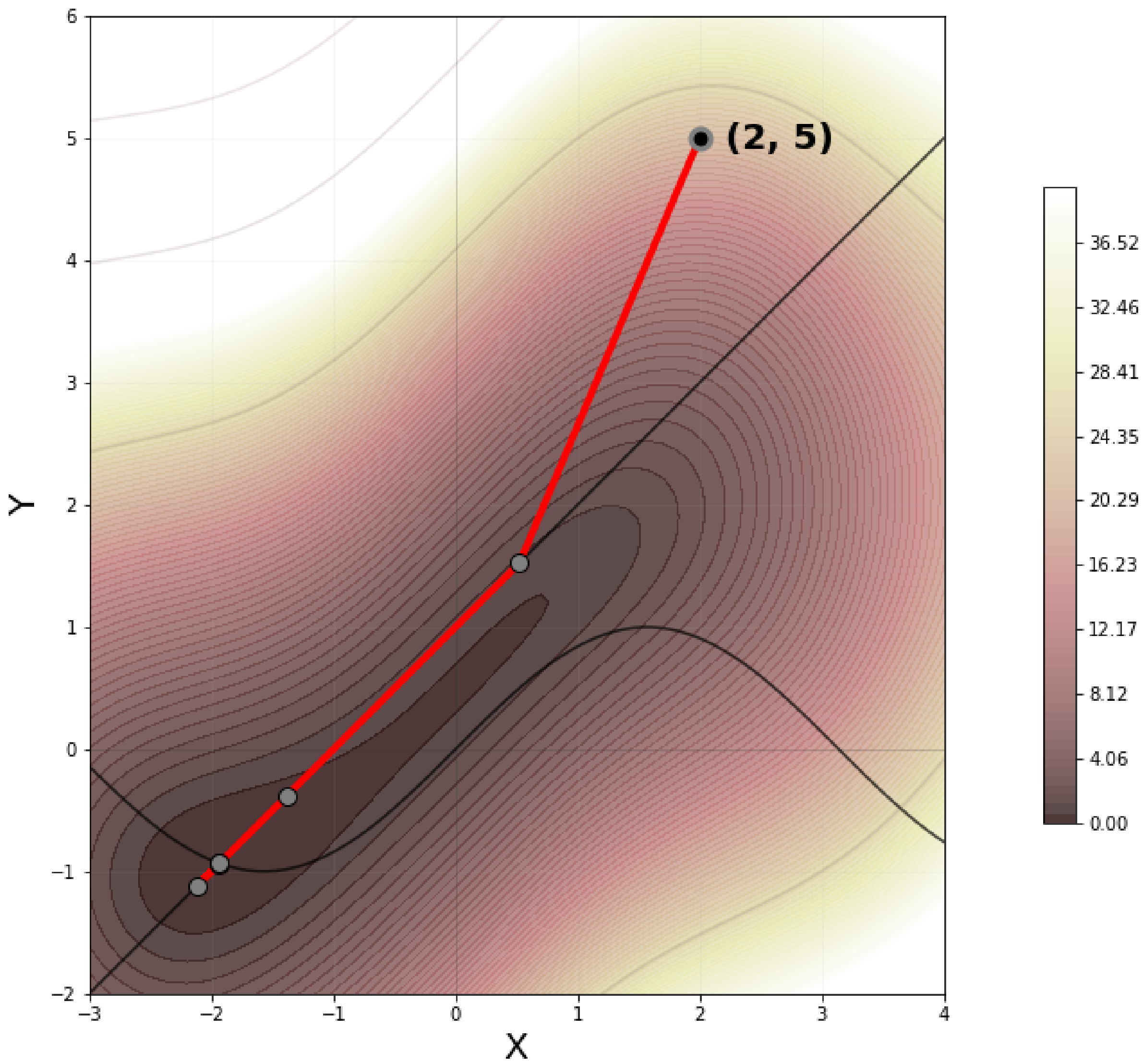
k	x_1	x_2	β	Невязка
1	0.000000e+00	4.000000e+00	1.000000e+00	0.000000e+00



- Прямая и синусоида

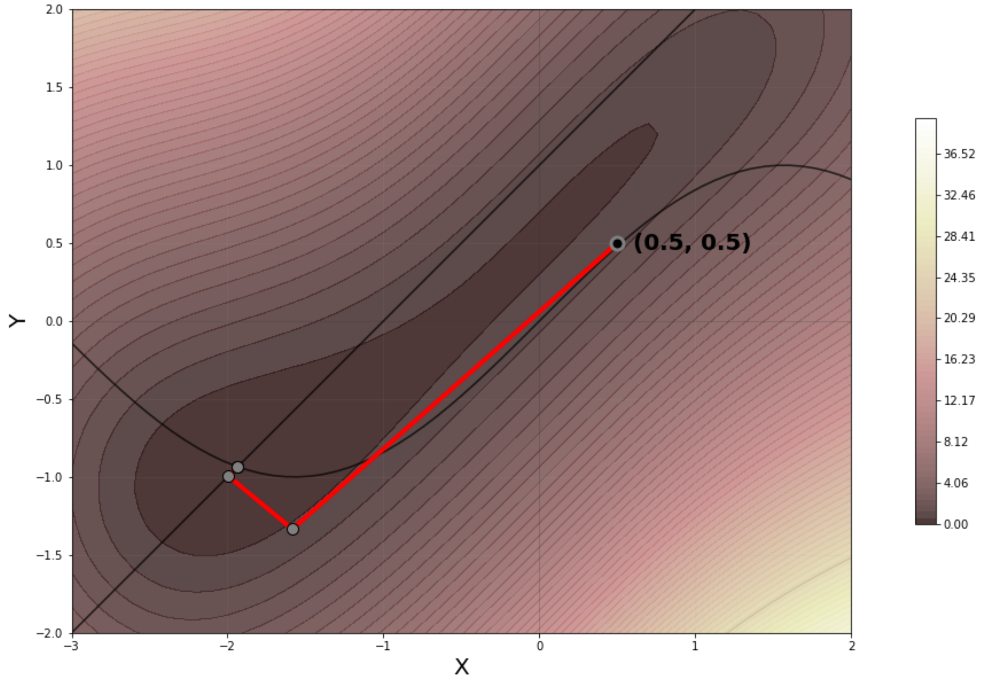
Начальное приближение (2, 5)

k	x_1	x_2	β	Невязка
1	5.236682e-01	1.523668e+00	1.000000e+00	2.247987e-01
2	-1.385915e+00	-3.859155e-01	2.500000e-01	1.311189e-01
3	-2.117432e+00	-1.117432e+00	1.000000e+00	5.779243e-02
4	-1.944284e+00	-9.442836e-01	1.000000e+00	2.903947e-03
5	-1.934595e+00	-9.345955e-01	1.000000e+00	9.607981e-06
6	-1.934563e+00	-9.345632e-01	1.000000e+00	1.068490e-10



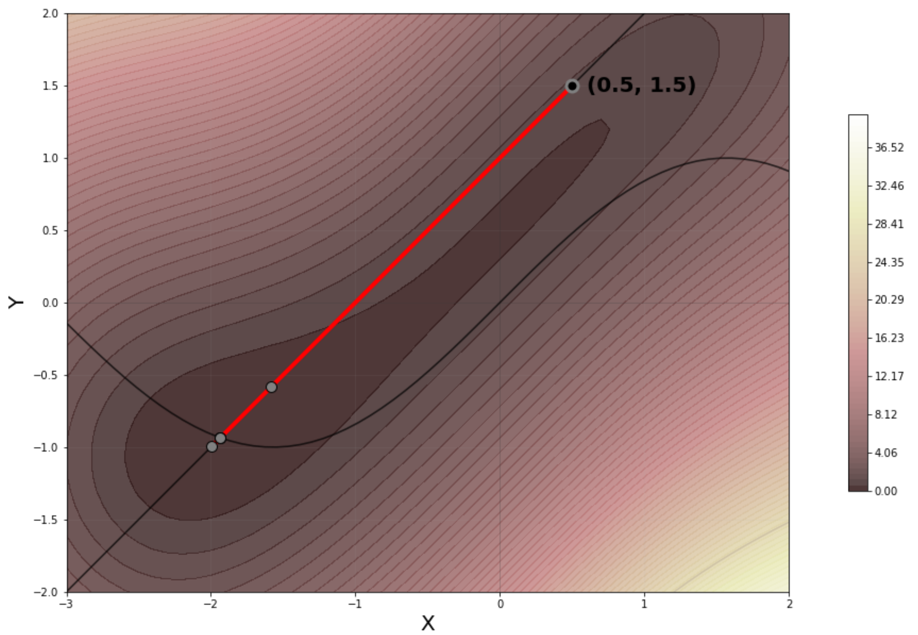
Начальное приближение (0.5, 0.5)

k	x_1	x_2	β	Невязка
1	-1.584210e+00	-1.334210e+00	2.500000e-01	8.209574e-01
2	-1.994408e+00	-9.944081e-01	1.000000e+00	8.278026e-02
3	-1.935730e+00	-9.357302e-01	1.000000e+00	1.582435e-03
4	-1.934564e+00	-9.345637e-01	1.000000e+00	6.354855e-07
5	-1.934563e+00	-9.345632e-01	1.000000e+00	1.026739e-13



Начальное приближение (0.5, 1.5)

k	x_1	x_2	β	Невязка
1	-1.584210e+00	-5.842097e-01	2.500000e-01	4.073199e-01
2	-1.994408e+00	-9.944081e-01	1.000000e+00	8.112861e-02
3	-1.935730e+00	-9.357302e-01	1.000000e+00	1.550861e-03
4	-1.934564e+00	-9.345637e-01	1.000000e+00	6.228061e-07
5	-1.934563e+00	-9.345632e-01	1.000000e+00	1.006253e-13



- Численное вычисление матрицы Якоби

Начальное приближение (2, 4). h = 1

k	x_1	x_2	β	Невязка
1	0.000000e+00	4.400000e+00	1	5.091169e-01
2	0.000000e+00	3.406897e+00	1	1.749522e-01
3	-5.551115e-17	2.887897e+00	1	6.968189e-02
4	-5.551115e-17	2.603884e+00	1	3.223310e-02
5	-5.551115e-17	2.438705e+00	1	1.701144e-02
6	-5.551115e-17	2.336191e+00	1	9.990017e-03
7	-5.551115e-17	2.268608e+00	1	6.377236e-03
…	…	…	…	…
948	-5.299647e-17	2.001065e+00	1	1.002302e-07
949	-5.299650e-17	2.001064e+00	1	1.000173e-07
950	-5.299652e-17	2.001063e+00	1	9.980503e-08

Начальное приближение (2, 4). h = 1e-10

k	x_1	x_2	β	Невязка
1	1.654807e-07	4.000000e+00	1	3.535534e-01
2	1.376677e-14	3.000000e+00	1	8.838836e-02
3	0.000000e+00	2.500000e+00	1	2.209709e-02
4	0.000000e+00	2.250000e+00	1	5.524273e-03
5	0.000000e+00	2.125000e+00	1	1.381068e-03
6	8.673617e-19	2.062501e+00	1	3.452793e-04
7	8.673617e-19	2.031251e+00	1	8.631983e-05
8	8.673617e-19	2.015625e+00	1	2.157996e-05
9	8.538092e-19	2.007812e+00	1	5.393456e-06
10	8.538092e-19	2.003906e+00	1	1.348364e-06
11	8.538092e-19	2.001954e+00	1	3.374742e-07
12	8.535975e-19	2.000977e+00	1	8.436860e-08

Начальное приближение (2, 4). h = 1e-15

k	x_1	x_2	β	Невязка
1	-2.517998e-01	4.000000e+00	1	3.700263e-01
2	3.170157e-02	2.856254e+00	1	6.585337e-02
3	-3.991225e-03	2.442950e+00	1	1.740090e-02
4	3.582534e-02	2.080734e+00	1	1.268493e-02
5	-6.706308e-03	2.088131e+00	1	2.469535e-03
6	-nan(ind)	inf	5.960464e-08	-nan(ind)

Вывод:

По графикам мы выяснили, что искомые направления вектора для поиска x_{k+1} соответствуют касательным к изолиниям функции $Enorm(F(x_k))$. Из этого следует что существуют такие точки, в которых нельзя найти такой вектор, при котором $Enorm(F(x_k))$ бы уменьшилась (вертикальные касательные). В алгоритме это выражается вырожденной матрицей Якоби.

В рамках работы мы рассмотрели три способа приведения СЛАУ к квадратному виду. Рассмотрим их отдельно:

1. Исключение строк

Исключаем те уравнения $F_i(x_k)$, значения которых минимально, поэтому их легко можно вычислить на графиках (одно из уравнений не учитывается при приближении и наблюдаются “скачки”).

2. Конволюция (Свертка)

“Скачки” также наблюдаются, однако видно, что исключаемые уравнения также учитываются.

3. Симметризация

На графиках ведет себя также, как тесты с квадратными СЛАУ.

Наименьшего количества итераций удалось добиться с использованием симметризации, однако сама операция симметризации очевидно алгоритмически сложнее, чем исключение строк.

При численном нахождении матрицы Якоби на результат влияет порядок h. Если порядок слишком большой по сравнению с $F(x_k)$, то решение будет находится долго или не будет найдено вообще, если порядок слишком мал, то можно получить переполнение. Поэтому порядок должен вычисляться относительно $F(x_k)$.

Текст программы

LinearAlgebra.h

```
#pragma once
#include <vector>
#include "common.h"

using namespace std;

class Vector
{
public:
	Vector(int size);

	int Size();
	real EuqlideanNorm();

	real& operator()(const int index);
	friend Vector operator *(real constant, const Vector& vector);
	friend Vector operator *(const Vector& vector, real constant);
	friend real operator *(const Vector& first, const Vector& second);
	friend Vector operator +(const Vector& first, const Vector& second);

private:
	vector<real> data;
};

class Matrix
{
public:
	int Rows();
	int Columns();
	Matrix Transpose();

	Matrix(int size);
	Matrix(int rows, int columns);

	real& operator()(const int row, const int column);
	friend Matrix operator*(real constant, const Matrix& matrix);
	friend Matrix operator*(const Matrix& matrix, real constant);
	friend Vector operator*(const Matrix& matrix, Vector vector);
	friend Matrix operator*(const Matrix& first, const Matrix& second);

private:
	vector<vector<real>> data;
};
```

LinearAlgebra.cpp

```
#include "LinearAlgebra.h"
#include <fstream>

Vector::Vector(int size)
{
	data.resize(size, 0.0);
}

int Vector::Size()
{
	return data.size();
```

```
}

real Vector::EuqlideanNorm()
{
	real sum = 0.0;

	for (int i = 0; i < data.size(); i++)
	{
		sum += data[i] * data[i];
	}

	return sqrt(sum);
}

real& Vector::operator()(int index)
{
	return data[index];
}

Vector operator*(real constant, const Vector& vector)
{
	Vector result = vector;

	for (int i = 0; i < result.Size(); i++)
	{
		result.data[i] *= constant;
	}

	return result;
}

Vector operator*(const Vector& vector, real constant)
{
	Vector result = vector;

	for (int i = 0; i < result.Size(); i++)
	{
		result(i) *= constant;
	}

	return result;
}

real operator *(const Vector& first, const Vector& second)
{
	real result = 0.0;

	for (int i = 0; i < second.data.size(); i++)
	{
		result += first.data[i] * second.data[i];
	}

	return result;
}

Vector operator +(const Vector& first, const Vector& second)
{
	Vector result(first.data.size());

	for (int i = 0; i < first.data.size(); i++)
	{
		result.data[i] = first.data[i] + second.data[i];
	}

	return result;
```

```
}

Matrix::Matrix(int size)
{
	data.resize(size);

	for (int i = 0; i < size; i++)
	{
		data[i].resize(size, 0.0);
	}
}

Matrix::Matrix(int rows, int colums)
{
	data.resize(rows);

	for (int i = 0; i < rows; i++)
	{
		data[i].resize(colums, 0.0);
	}
}

int Matrix::Columns()
{
	return data[0].size();
}

int Matrix::Rows()
{
	return data.size();
}

Matrix Matrix::Transpose()
{
	Matrix result(Columns(), Rows());

	for (int i = 0; i < Rows(); i++)
	{
		for (int j = 0; j < Columns(); j++)
		{
			result.data[j][i] = data[i][j];
		}
	}

	return result;
}

Matrix operator*(real constant, const Matrix& matrix)
{
	Matrix result(matrix.data.size());

	for (int i = 0; i < result.data.size(); i++)
	{
		for (int j = 0; j < result.data[i].size(); j++)
		{
			result.data[i][j] *= constant;
		}
	}

	return result;
}

Matrix operator*(const Matrix& matrix, real constant)
{
	Matrix result(matrix.data.size());
```

```
	for (int i = 0; i < result.data.size(); i++)
	{
		for (int j = 0; j < result.data[i].size(); j++)
		{
			result.data[i][j] *= constant;
		}
	}

	return result;
}

Matrix operator*(const Matrix& first, const Matrix& second)
{
	Matrix result(first.data.size());

	for (int i = 0; i < first.data.size(); i++)
	{
		for (int j = 0; j < first.data.size(); j++)
		{
			for (int k = 0; k < second.data.size(); k++)
			{
				result.data[i][j] += first.data[i][k] *
second.data[k][j];
			}
		}
	}

	return result;
}

real& Matrix::operator()(const int row, const int column)
{
	return data[row][column];
}

Vector operator*(const Matrix& matrix, Vector vector)
{
	Vector result(matrix.data.size());

	for (int i = 0; i < matrix.data.size(); i++)
	{
		for (int j = 0; j < matrix.data[0].size(); j++)
		{
			result(i) += matrix.data[i][j] * vector(j);
		}
	}

	return result;
}
```

SystemOfNonlinearEquations.h

```
#pragma once
#include "common.h"
#include "LinearAlgebra.h"

class VectorOfFunctions
{
public:
	virtual Vector ComputeInPoint(Vector point) = 0;
};

class DisjointCircles : public VectorOfFunctions
```

```
{
private:
	int size = 2;

public:
	Vector ComputeInPoint(Vector point) override;
};

class IntersectingCirclesAtPoint : public VectorOfFunctions
{
private:
	int size = 2;

public:
	Vector ComputeInPoint(Vector point) override;
};

class IntersectingCircles : public VectorOfFunctions
{
private:
	int size = 2;

public:
	Vector ComputeInPoint(Vector point) override;
};

struct SystemParameters
{
	int n;
	int m;
	int maxiter;
	int maxiterBeta;
	real epsF;
	real epsBeta;
	VectorOfFunctions* F;
	Vector x0 = Vector(1);

	SystemParameters(int n, int m, int maxiter, int maxiterBeta, real epsF,
real epsBeta, VectorOfFunctions* Function, Vector x0);
};

class Squaring
{
public:
	virtual void LeadToSquare(Matrix& matrix, Vector& vector);
};

class ExcludingRows : public Squaring
{
public:
	void LeadToSquare(Matrix& matrix, Vector& vector) override;
};

class Convolution : public Squaring
{
public:
	void LeadToSquare(Matrix& matrix, Vector& vector) override;
};

class SystemOfNonlinearEquations
{
private:
	int n;
	int m;
	int maxiter;
```

```
	int maxiterBeta;
	real epsF;
	real epsBeta;
	VectorOfFunctions* F;
	Vector x0 = Vector(1);

	Squaring* squaring;

public:
	SystemOfNonlinearEquations(struct SystemParameters parameters, Squaring*
squaring);
	Vector Solve();

private:
	Matrix FormJacobiMatrix(Vector x);
	Vector ComputeDirectionByGauss(Matrix matrix, Vector vector);
};
```

SystemOfNonlinearEquations.cpp

```
#pragma once
#include "SystemOfNonlinearEquations.h"
#include <iostream>

SystemParameters::SystemParameters(int n, int m, int maxiter, int maxiterBeta,
real epsF, real epsBeta, VectorOfFunctions* Function, Vector x0)
{
	this->n = n;
	this->m = m;
	this->maxiter = maxiter;
	this->maxiterBeta = maxiterBeta;
	this->epsF = epsF;
	this->epsBeta = epsBeta;
	this->F = Function;
	this->x0 = x0;
}

Vector DisjointCircles::ComputeInPoint(Vector point)
{
	Vector value(size);

	value(0) = pow(point(0) + 2, 2) + pow(point(1) - 2, 2) - 4;
	value(1) = pow(point(0) - 2, 2) + pow(point(1) - 2, 2) - 4;

	return value;
}

Vector IntersectingCirclesAtPoint::ComputeInPoint(Vector point)
{
	return Vector(2);
}

Vector IntersectingCircles::ComputeInPoint(Vector point)
{
	return Vector(2);
}

void Squaring::LeadToSquare(Matrix& matrix, Vector& vector)
{
	Matrix temp = matrix.Transpose();
	matrix = temp * matrix;
	vector = (-1.0 * temp) * vector;
}
```

```
void ExcludingRows::LeadToSquare(Matrix& matrix, Vector& vector)
{
	int m = matrix.Rows();
	int n = matrix.Columns();
	int rowsToDelete = m - n;
	int row;
	real min;

	for (int i = 0; i < rowsToDelete; i++)
	{
		row = 0;
		min = fabs(vector(0));
		for (int j = 0; j < m - i; j++)
		{
			if (fabs(vector(j)) < min)
			{
				min = fabs(vector(j));
				row = j;
			}
		}

		swap(vector(row), vector(m - i - 1));
		for (int j = 0; j < n; j++)
		{
			swap(matrix(row, j), matrix(m - i - 1, j));
		}
	}
}

void Convolution::LeadToSquare(Matrix& matrix, Vector& vector)
{
	int m = matrix.Rows();
	int n = matrix.Columns();
	int rowsToDelete = m - n + 1;
	int row;
	real min;
	real sum = 0.0;
	Vector rowsSum(n);

	for (int i = 0; i < rowsToDelete; i++)
	{
		row = 0;
		min = fabs(vector(0));
		for (int j = 0; j < m - i; j++)
		{
			if (fabs(vector(j)) < min)
			{
				min = fabs(vector(j));
				row = j;
			}
		}

		sum += pow(vector(row), 2);
		for (int j = 0; j < n; j++)
		{
			rowsSum(j) += pow(matrix(row, j), 2);
		}

		swap(vector(row), vector(m - i - 1));
		for (int j = 0; j < n; j++)
		{
			swap(matrix(row, j), matrix(m - i - 1, j));
		}
	}
```

```
	vector(n - 1) = sum;
	for (int i = 0; i < n; i++)
	{
		matrix(n - 1, i) = rowsSum(i);
	}
}

SystemOfNonlinearEquations::SystemOfNonlinearEquations(struct SystemParameters
parameters, Squaring* squaring)
{
	this->n = parameters.n;
	this->m = parameters.m;
	this->maxiter = parameters.maxiter;
	this->maxiterBeta = parameters.maxiterBeta;
	this->epsF = parameters.epsF;
	this->epsBeta = parameters.epsBeta;
	this->F = parameters.F;
	this->x0 = parameters.x0;

	this->squaring = squaring;
}

Matrix SystemOfNonlinearEquations::FormJacobiMatrix(Vector x)
{
	Matrix Jacobi(m, n);
	real h = 1e-10;

	Vector temp = x;

	Vector Fp = F->ComputeInPoint(x);

	for (int i = 0; i < m; i++)
	{
		for (int j = 0; j < n; j++)
		{
			temp(j) += h;
			Vector Fp_h = F->ComputeInPoint(temp);

			for (int k = 0; k < m; k++)
			{
				Fp_h(k) -= Fp(k);
			}

			Jacobi(i, j) = Fp_h(i) / h;
			temp(j) = x(j);
		}
	}

	return Jacobi;
}

Vector SystemOfNonlinearEquations::ComputeDirectionByGauss(Matrix matrix, Vector
vector)
{
	vector = -1 * vector;

	int i;

	for (i = 0; i < n; i++)
	{
		real mainElement = 0.0;
		int row = 0;

		for (int j = i; j < n; j++)
		{
```

```
                    if (mainElement < fabs(matrix(j, i)))
                    {
                          mainElement = matrix(j, i);
                          row = j;
                    }
              }

              if (row != i)
              {
                    swap(vector(i), vector(row));
                    for (int j = 0; j < n; j++)
                    {
                          swap(matrix(i, j), matrix(row, j));
                    }
              }

              vector(i) /= mainElement;
              for (int j = i + 1; j < n; j++)
              {
                    matrix(i, j) /= mainElement;
              }

              for (int j = i + 1; j < n; j++)
              {
                    mainElement = matrix(j, i);

                    for (int k = i; k < n; k++)
                    {
                          matrix(j, k) -= mainElement * matrix(i, k);
                    }

                    vector(j) -= mainElement * vector(i);
              }
       }

       for (i -= 2; i >= 0; i--)
       {
              for (int j = i + 1; j < n; j++)
              {
                    vector(i) -= vector(j) * matrix(i, j);
              }
       }

       return vector;
}

Vector SystemOfNonlinearEquations::Solve()
{
       real F0Norm = F->ComputeInPoint(x0).EuqlideanNorm();
       real FNorm;
       real FkNorm = F0Norm;
       Vector xk = x0;
       Vector xk1(n);
       real discrepancy = FkNorm / F0Norm;

       for (int k = 0; k < maxiter && discrepancy > epsF; k++)
       {
              Vector Fk = F->ComputeInPoint(xk);
              Matrix Jacobi = FormJacobiMatrix(xk);

              if (m != n)
              {
                    squaring->LeadToSquare(Jacobi, Fk);
              }
```

```
            Vector dx = ComputeDirectionByGauss(Jacobi, Fk);
            real beta = 1.0;
            FNorm = FkNorm;
            for (int v = 0; v < maxiterBeta && beta > epsBeta; v++)
            {
                  for (int i = 0; i < n; i++)
                  {
                        xk1(i) = xk(i) + beta * dx(i);
                  }

                  FkNorm = F->ComputeInPoint(xk1).EuqlideanNorm();
                  if (FkNorm < FNorm)
                  {
                        break;
                  }

                  beta /= 2;
            }

            for (int i = 0; i < n; i++)
            {
                  xk(i) = xk1(i);
            }

            discrepancy = FkNorm / F0Norm;

            cout << "beta: " << beta << endl;
            cout << "discrepancy: " << discrepancy << endl;
            cout << "xk: ";
            for (int i = 0; i < n; i++)
            {
                  cout << fixed << xk(i) << " ";
            }
            cout << endl << endl;
      }

      return xk;
}
```