

Национальный исследовательский университет ИТМО  
Факультет информационных технологий и программирования  
Информационные системы и технологии

**Алгоритмы одномерной минимизации**  
*Отчёт по лабораторной работе №3*

**Работу выполнили:**

Дерёвицкая П. К., студентка М32081  
Фирсова Д. А., студентка М32081  
Борзун А. В., студентка М32051

**Преподаватель:**

Свинцов М. В.

Санкт-Петербург,  
2023

## **Оглавление**

<b>Теоретическая часть .....</b>	<b>4</b>
<b>Алгоритм градиентного спуска с постоянным шагом .....</b>	<b>4</b>
<b>Алгоритм градиентного спуска с дроблением шага .....</b>	<b>6</b>
<b>Метод наискорейшего спуска .....</b>	<b>8</b>
<b>Метод сопряженных градиентов.....</b>	<b>10</b>
<b>Практическая часть .....</b>	<b>11</b>
<b>Тестирование алгоритмов.....</b>	<b>11</b>
<b>Первая функция (Химмельблау) .....</b>	<b>11</b>
<b>Вторая функция .....</b>	<b>20</b>
<b>Генератор квадратичных функций.....</b>	<b>27</b>
<b>Вывод.....</b>	<b>28</b>

**Цель работы:** знакомство с методами спуска

**Задачи, решаемые во время выполнения работы:**

1. Реализация данных методов на Python 3.
2. Тестирование реализованных алгоритмов для задач минимизации
3. Исследование сходимости градиентного спуска
4. Сравнение эффективности методов

## Теоретическая часть

### Алгоритм градиентного спуска с постоянным шагом

Градиентный метод с постоянным шагом — это один из простых алгоритмов оптимизации, который использует производную функции, чтобы находить оптимальное значение. Он заключается в том, чтобы на каждом шаге двигаться в направлении наиболее быстрого возрастания (убывания) функции, т. е. в направлении градиента (антиградиента) функции с заданным шагом, который остается постоянным на всем пространстве оптимизации.

Градиентом функции в точке  $X = \{x_1, x_2, \dots, x_n\}$  называется вектор, проекциями которого на координатные оси являются частные производные функции по координатам:

$$\nabla f(x_1, x_2, \dots, x_n) = \left( \frac{df}{dx_1} i + \frac{df}{dx_2} j + \dots + \frac{df}{dx_n} n \right)$$

где  $i, j, \dots, n$  — единичные векторы, параллельные координатным осям.

Стратегия решения задачи состоит в построении последовательности точек  $k = 0, 1, 2, \dots, n$ , таких что  $f(x^{k+1}) < f(x^k)$ ,  $k = 0, 1, 2, \dots, n$ . Точки последовательности вычисляются по правилу:  $x^{k+1} = x^k - t_k$

Величина шага выбирается из условия минимума целевой функции  $f(x)$  в направлении движения, т. е. в результате решения задачи одномерной оптимизации в направлении градиента или антиградиента.

Другими словами, величину шага  $\lambda$  определяют при решении данного уравнения:

$$\frac{df(X_k \pm \lambda S_k)}{d\lambda} = 0$$

Таким образом, шаг расчета выбирается такой величины, что движение выполняется до тех пор, пока происходит улучшение функции, достигая, таким образом, экстремума в некоторой точке. В этой точке вновь определяют направление поиска (с помощью градиента) и ищут новую точку оптимума целевой функции и т. д. Таким образом, в данном методе поиск происходит более крупными шагами, и градиент функции вычисляется в меньшем числе точек.

Проблема: достаточно малый шаг обеспечивает убывание функции, но может привести к неприемлемо большому количеству итераций, необходимых для достижения точки минимума. С другой стороны, слишком большой шаг может вызвать неожиданный рост функции (невыполнение условия убывания) либо привести к колебаниям около точки минимума. Однако проверка условия убывания на каждой итерации — трудоемкая задача, поэтому мы изначально зададим постоянным и достаточно малым чтобы можно было использовать этот шаг на любой итерации. Но приходиться мириться с возможно большим количеством итераций. Утешением является лишь то, что трудоемкость каждой итерации, в этом случае, минимальна (нужно вычислять только градиент  $f'(x_k)$ ).

Достоинства:

1. Простота реализации.
2. Не требует наличия производных второго порядка.
3. Применим для большого количества нелинейных задач оптимизации.

Недостатки:

1. Сходимость может быть медленной в ряде случаев.
2. Если шаг выбирается неверно, то метод может расходиться.
3. Не гарантирована сходимость к глобальному минимуму.

Время работы метода зависит от многих факторов, таких как размерность пространства оптимизации, структура объектной функции, точность результата, выбранная константа шага. В лучшем случае, когда функция имеет простую структуру и оптимальное значение не находится на границе области определения, метод сходится за несколько итераций. В худшем случае метод может зациклиться или не найти оптимальное значение вовсе.

## Алгоритм градиентного спуска с дроблением шага

Алгоритм градиентного спуска с дроблением шага — это метод оптимизации функции, который позволяет достичь минимума функции, минимизируя ошибку на каждой итерации.

Функция, которую мы хотим минимизировать, обозначается как  $f(x)$ , где  $x = [x_1, x_2, \dots, x_n]$  — набор параметров. Начальное приближение решения обозначается как  $x_0$ .

Алгоритм градиентного спуска работает по следующему принципу:

1. Вычисляем градиент функции  $f(x)$  в точке  $x_k$ :

$$\nabla f(x_k) = \left[ \frac{\partial f}{\partial x_1}(x_k), \frac{\partial f}{\partial x_2}(x_k), \dots, \frac{\partial f}{\partial x_n}(x_k) \right]$$

2. Выполняем дробление шага по формуле:

$$\alpha_k = \alpha_0 \cdot \beta^{j-1}$$

где  $\alpha_k$  - шаг на  $k$ -ой итерации,  $\alpha_0$  - начальный шаг,  $\beta$  - коэффициент дробления,  $j$  - номер итерации дробления шага.

4. Вычисляем новое приближение решения:

$$x_{k+1} = x_k - \alpha_k \cdot \nabla f(x_k)$$

5. Повторяем шаги 1-3, пока не достигнем заданного критерия останова (например, когда величина градиента становится меньше некоторого эпсилон).

Математический алгоритм градиентного спуска с дроблением шага может быть записан в следующем виде:

$$x_{k+1} = x_k - \alpha_k \cdot \nabla f(x_k)$$

$$\alpha_k = \alpha_0 \cdot \beta^{j-1}$$

$$\nabla f(x_k) = \left[ \frac{\partial f}{\partial x_1}(x_k), \frac{\partial f}{\partial x_2}(x_k), \dots, \frac{\partial f}{\partial x_n}(x_k) \right]$$

где  $j$  - номер итерации дробления шага.

Таким образом, в каждой итерации мы находим новое приближение решения, используя градиент функции в текущей точке и дробление шага. По мере продвижения к минимуму, мы уменьшаем шаг, что позволяет алгоритму сходиться быстрее.

Описание алгоритма:

1. Инициализируем вектор весов  $w$  и задаем начальный шаг (например, 1);

2. В цикле:

- a. Вычисляем градиент функции  $f(w)$  по вектору весов  $w$ ;
- b. Уменьшаем шаг до тех пор, пока  $f(w - \alpha \nabla f(w)) > f(w)$ , где  $\alpha$  - коэффициент шага;
- c. Обновляем веса заданным шагом:  $w := w - \alpha \nabla f(w)$ ;

Вычислительная сложность алгоритма зависит от сложности аналитического выражения для функции  $f(w)$ . Обычно функция градиента вычисляется за  $O(d)$ , где  $d$  - размерность пространства векторов весов.

Достоинства метода:

1. Позволяет находить локальные минимумы функций;
2. Функция градиента не требует задания производных в явном виде;
3. Можно выбирать различные коэффициенты шага для каждой итерации, что позволяет добиться наилучшего результата.

Недостатки метода:

1. Может сходиться к локальному минимуму, в зависимости от начальной инициализации вектора весов;
2. Может совершать лишние вычисления, если шаг будет настроен неправильно;
3. Не гарантирует глобальный минимум.

## Метод наискорейшего спуска

Метод наискорейшего спуска — это итерационный метод (первого порядка) оптимизации, который используется для поиска экстремума многомерной функции. Метод наискорейшего спуска является дальнейшим развитием метода градиентного спуска.

Он основывается на идее того, что на каждой итерации мы двигаемся в направлении наибольшего убывания функции. Это направление задается градиентом функции. Метод наискорейшего спуска заключается в том, что на каждой итерации мы выбираем шаг в направлении градиента, который минимизирует функцию на этом шаге. Мы можем использовать любой метод одномерной минимизации для нахождения этого шага.

Величина шага выбирается из условия минимума целевой функции  $f(x)$  в направлении движения, т. е. в результате решения задачи одномерной оптимизации в направлении градиента или антиградиента.

шаг расчета выбирается такой величины, что движение выполняется до тех пор, пока происходит улучшение функции, достигая, таким образом, экстремума в некоторой точке. В этой точке вновь определяют направление поиска (с помощью градиента) и ищут новую точку оптимума целевой функции и т.д. Таким образом, в данном методе поиск происходит более крупными шагами, и градиент функции вычисляется в меньшем числе точек.

Пусть дана функция  $f(x)$ , где  $x$  - вектор размерности  $n$ . Метод наискорейшего спуска математически можно записать в следующем виде:

1. Выбираем начальное приближение  $x_0$ .
2. Находим направление наискорейшего спуска  $d_k$ :

$$d_k = -\nabla f(x_k)$$

3. Находим оптимальный шаг  $\alpha_k$  в направлении  $d_k$ :

$$\alpha_k = \arg \min_{\alpha > 0} f(x_k + \alpha d_k)$$

4. Вычисляем новое приближение:

$$x_{k+1} = x_k + \alpha_k d_k$$

5. Повторяем шаги 2-4, пока не будет достигнуто условие остановки.

Алгоритм метода наискорейшего спуска состоит из следующих шагов:

1. Задать начальную точку  $i_p = (x_1, x_2)$ .
2. Вычислить частные производные функции  $f(x_1, x_2)$  по переменным  $x_1$  и  $x_2$ , обозначенные как  $f'(x_1)$  и  $f'(x_2)$  соответственно.

3. Найти направление движения  $s_1$  и  $s_2$ , которое является наискорейшим убыванием функции  $f(x_1, x_2)$  в заданной точке. Для этого необходимо взять отрицательные значения частных производных  $f'(x_1)$  и  $f'(x_2)$ , так как функция убывает в направлении отрицательных значений производных.
4. Найти длину шага  $dstar$  в направлении  $s_1$  и  $s_2$ , используя метод золотого сечения (golden\_section).
5. Перейти в новую точку  $(new_{x_1}, new_{x_2})$ , используя найденное направление движения и длину шага:  $new_{x_1} = x_1 + dstars_1$ ,  $new_{x_2} = x_2 + dstars_2$ .
6. Проверить условие остановки: если разница между значениями функции  $f(x_1, x_2)$  на текущей и предыдущей итерациях меньше заданной точности  $epsilon$ , то остановить алгоритм. Иначе продолжить итерации.
7. Сохранить значения на каждой итерации в массив results.

Вычислительная сложность метода наискорейшего спуска с дроблением шага составляет  $O(kn)$ , где  $k$  - количество итераций,  $n$  - размерность пространства функции.

Достоинства метода:

1. Прост в реализации
2. Высокая скорость сходимости на начальных этапах оптимизации

Недостатки метода:

1. Может быть медленным при сильно выпуклых или невыпуклых функциях
2. Может застрять в локальном минимуме
3. Не подходит для оптимизации функций с нелинейными ограничениями

В целом, метод наискорейшего спуска с дроблением шага является простым и эффективным методом оптимизации, который может использоваться в различных областях науки и техники.

## Метод сопряженных градиентов

Метод сопряженных градиентов — это итерационный численный метод (первого порядка) решения оптимизационных задач, который позволяет определить экстремум (минимум или максимум) целевой функции. Метод сопряженных градиентов является дальнейшим развитием метода наискорейшего спуска, который сочетает в себе два понятия: градиент целевой функции и сопряженное направление векторов.

Формула метода сопряженных градиентов:

$$x_{k+1} = x_k + \alpha_k p_k$$

$$p_{k+1} = -g_{k+1} + \beta_k p_k$$

где  $x_k$  — это приближенное решение на шаге  $k$ ,  $p_k$  — это направление поиска на шаге  $k$ ,  $g_k$  — это градиент функции на шаге  $k$ ,  $\alpha_k$  и  $\beta_k$  — это коэффициенты, которые определяются на каждом шаге для минимизации функции.

Метод сопряженных градиентов имеет вычислительную сложность  $O(n^2)$  для решения систем линейных уравнений размерности  $n$ . Это делает метод сопряженных градиентов особенно полезным для больших систем уравнений, где методы прямых решений неэффективны.

Достоинства метода сопряженных градиентов:

1. Эффективность: метод сопряженных градиентов является одним из наиболее эффективных методов для решения больших систем линейных уравнений.
2. Низкая память: метод сопряженных градиентов требует небольшого объема памяти, что делает его применимым для обработки больших объемов данных.
3. Универсальность: метод сопряженных градиентов может использоваться для решения различных задач, включая задачи оптимизации и решение систем линейных уравнений.

Недостатки метода сопряженных градиентов:

1. Чувствительность к начальному приближению: метод сопряженных градиентов может оказаться неустойчивым, если начальное приближение выбрано неправильно.
2. Ограничения на форму функции: метод сопряженных градиентов может работать только с квадратичными функциями или функциями, которые можно приблизить квадратичными.
3. Ограничения на точность вычислений: метод сопряженных градиентов может столкнуться с проблемами точности вычислений при работе с большими системами уравнений.

В целом, метод сопряженных градиентов является эффективным и универсальным методом для решения больших систем линейных уравнений и задач оптимизации.

## Практическая часть

### Тестирование алгоритмов

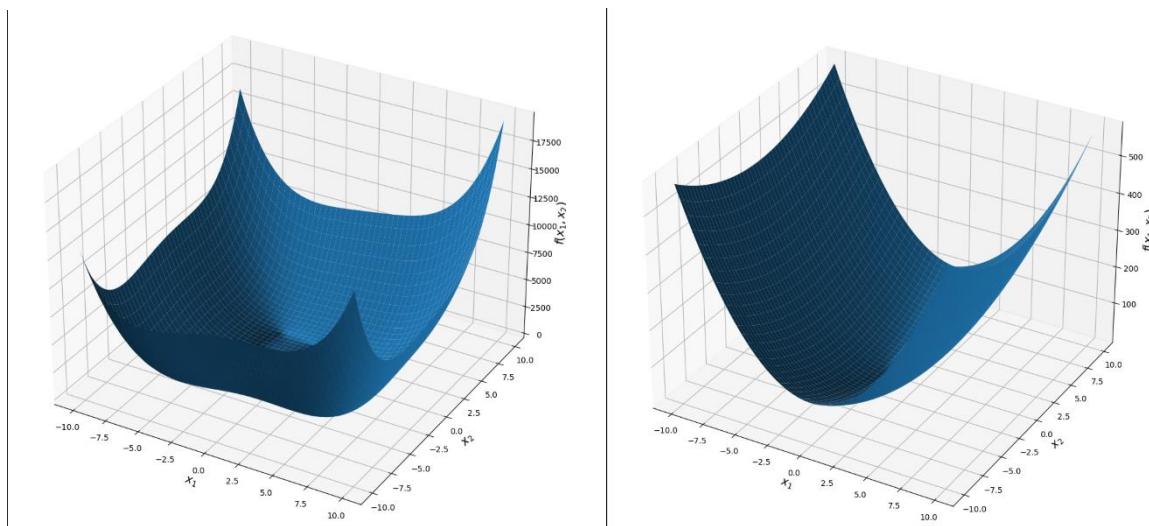
Для тестирования написанных алгоритмов нами были выбраны две функции, ведущие себя по-разному при использовании на них метода градиентного спуска:

$$f_1(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7) ** 2,$$

$$f_2(x_1, x_2) = 4,70x_1^2 + 1,06x_2^2 + 0,30x_2^2 + 0,39.$$

Первая функция представляет собой квадратичную функцию с одним минимумом, а вторая функция является нелинейной функцией с несколькими минимумами. При использовании градиентного спуска на этих функциях различия заключаются в том, что для первой функции градиентный спуск сходится к единственному минимуму, а для второй функции может существовать несколько локальных минимумов, к которым градиентный спуск может сойтись в зависимости от начальной точки и параметров алгоритма. Также для нелинейных функций может потребоваться больше итераций для сходимости к минимуму, чем для квадратичных функций.

Построим графики данных функций:

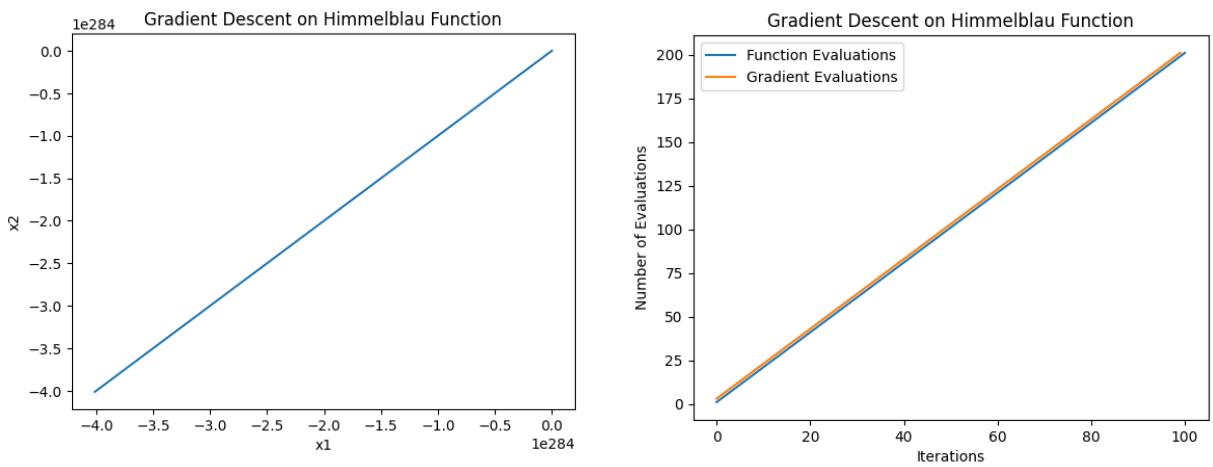


**Первая функция (Химмельблау)**

Для первой функции:

#### 1) Градиентный спуск с постоянным шагом.

1. Рассмотрим сходимость градиентного спуска с постоянным шагом. Для этого построим график изменения градиента и значений функции от количества итераций.



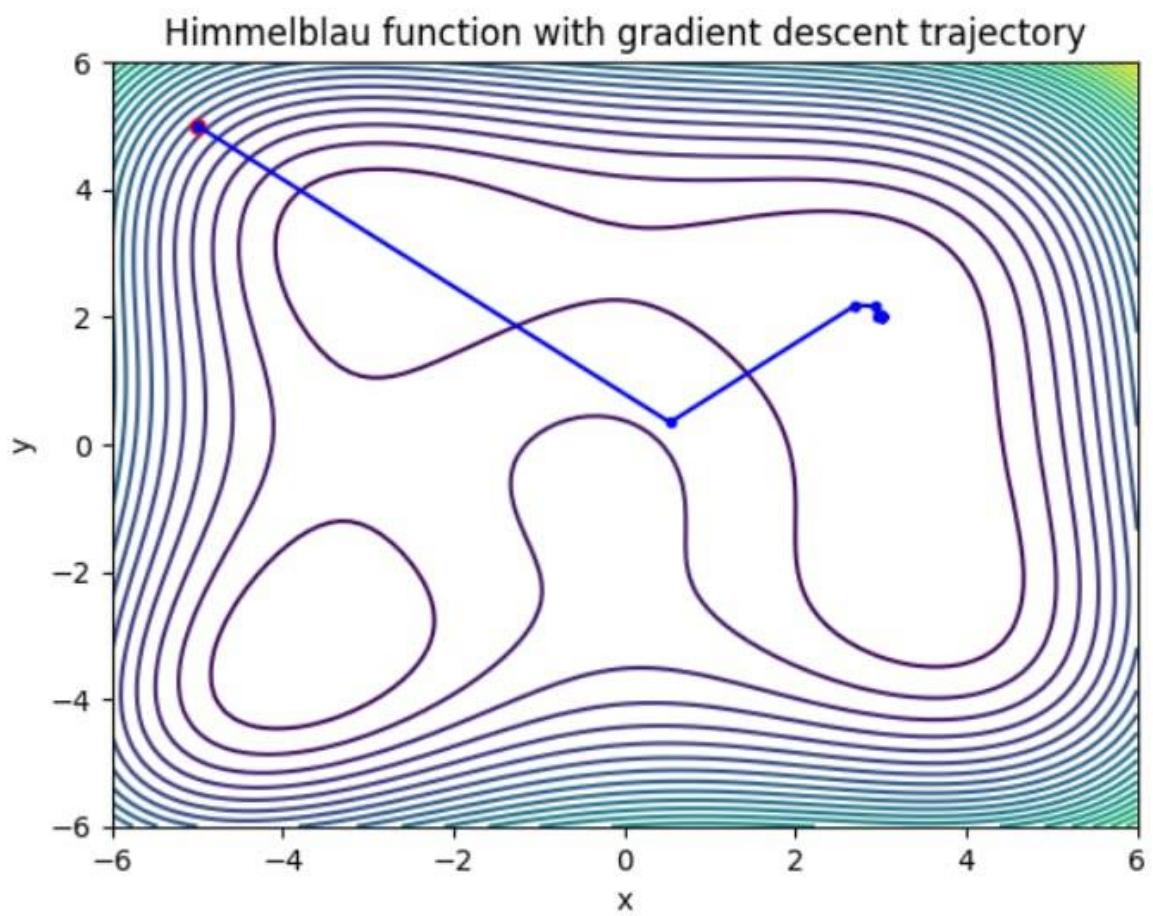
2. Сравним эффективность методов с точки зрения количества вычислений минимизируемой функции. И исследуем работу методов в зависимости от выбора начальной точки.

Iteration	x1	x2	current_point	grad
0	-4	[-4 -4]	[-6 -78]	[-3.94 -3.22]
1	-3.94	[-3.94 -3.22]	[-21.687936 9.969408]	[-3.72312064 -3.31969408]
2	-3.72312	[-3.72312064 -3.31969408]	[7.41624782 -4.86322519]	[-3.79728312 -3.27106183]
3	-3.79728	[-3.79728312 -3.27106183]	[-2.44738187 1.57149261]	[-3.7728093 -3.28677675]
4	-3.77281	[-3.7728093 -3.28677675]	[0.85529242 -0.50099798]	[-3.78136222 -3.28176677]
5	-3.78136	[-3.78136222 -3.28176677]	[-0.27886479 0.18310942]	[-3.77857358 -3.28359787]
6	-3.77857	[-3.77857358 -3.28359787]	[0.09726023 -0.05715973]	[-3.77954618 -3.28302627]
7	-3.77955	[-3.77954618 -3.28302627]	[-0.03194461 0.02075663]	[-3.77922673 -3.28323384]
8	-3.77923	[-3.77922673 -3.28323384]	[0.01106199 -0.00658127]	[-3.77933735 -3.28316802]
9	-3.77934	[-3.77933735 -3.28316802]	[-0.0036582 0.00235077]	[-3.77930077 -3.28319153]
10	-3.7793	[-3.77930077 -3.28319153]	[0.00125915 -0.00075685]	[-3.77931336 -3.28318396]

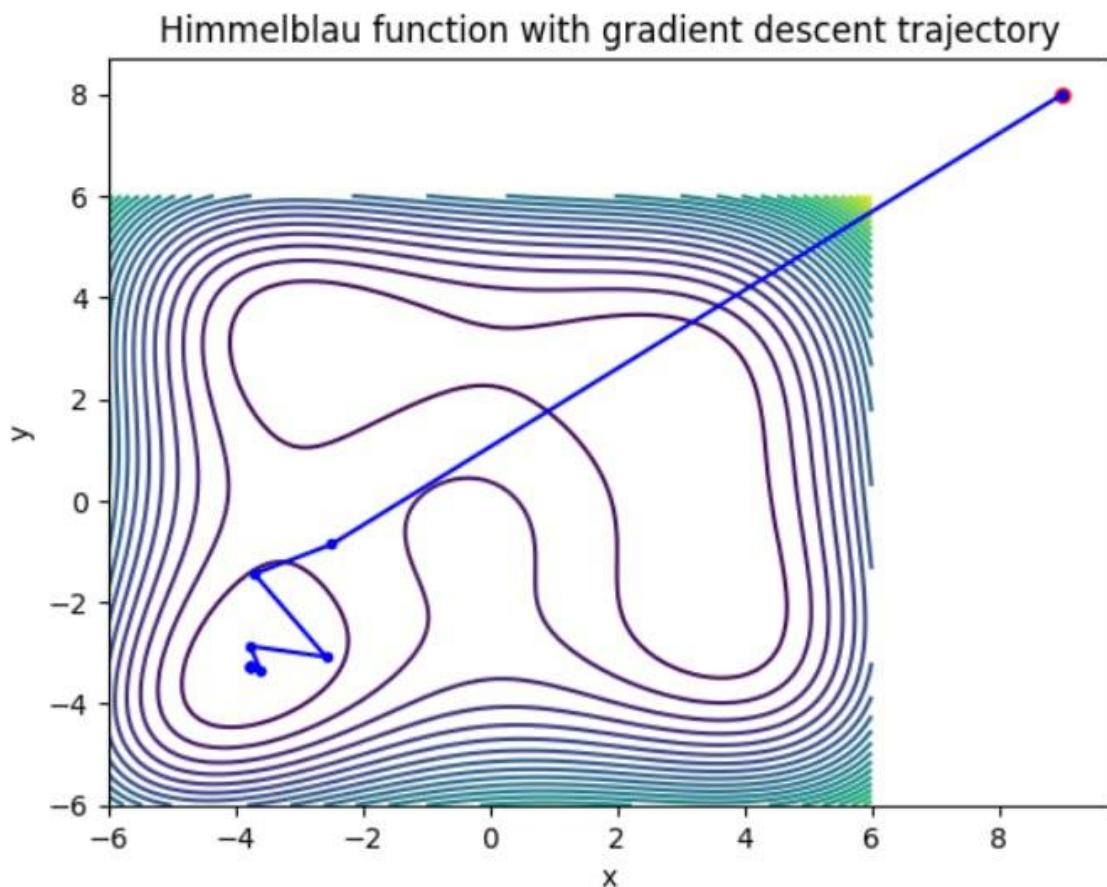
Если начальная точка выбрана близко к оптимальной точке, то метод градиентного спуска сходится быстро и требует меньшего количества вычислений минимизируемой функции. Однако, если начальная точка выбрана далеко от оптимальной точки, то метод градиентного спуска может сходиться медленно и требовать большего количества вычислений минимизируемой функции.

3. Нарисуем график с линиями уровня и траекториями методов.

Для начальной точки [-5, 5]



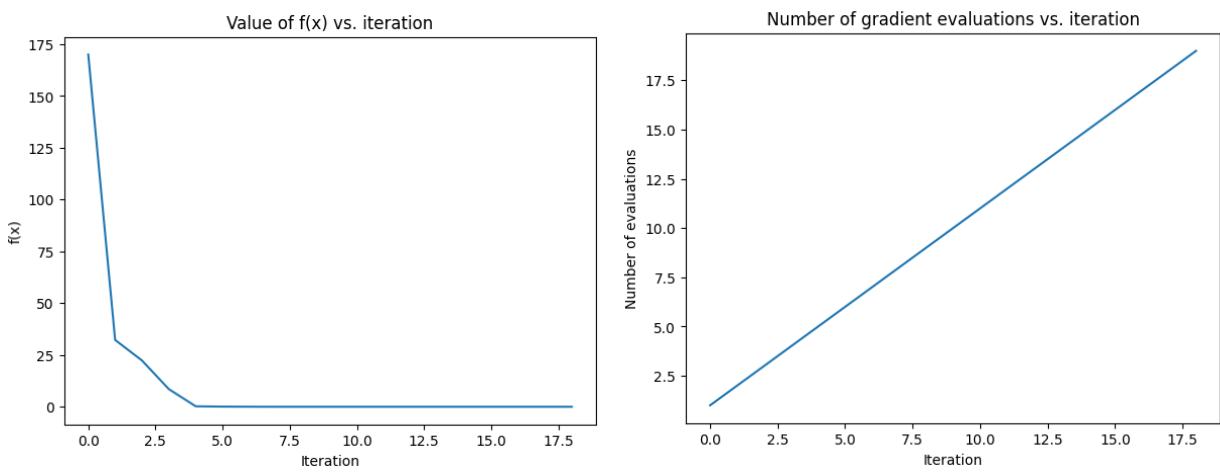
Для начальной точки [9, 8]



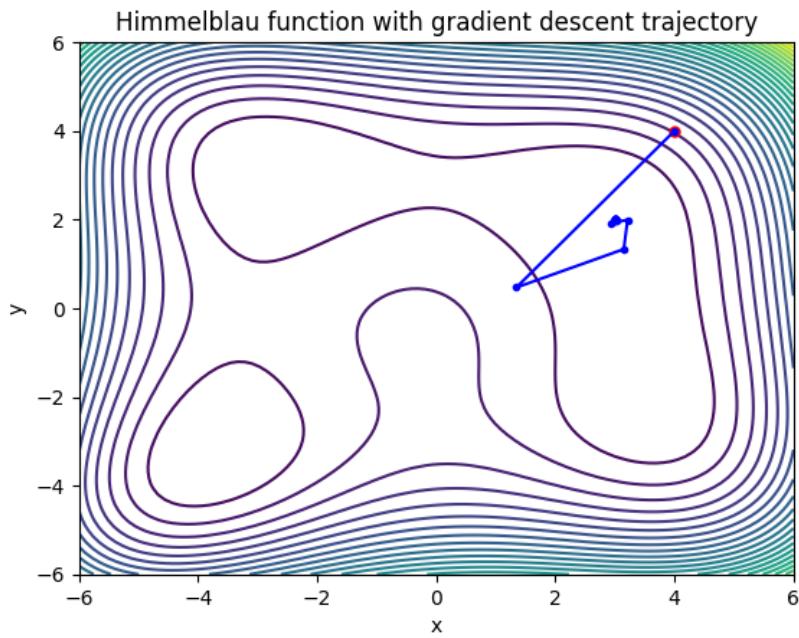
Траектории метода градиентного спуска сходятся к одной точке, то это означает, что алгоритм сошелся к оптимальной точке и достиг минимума функции.

## **2) Градиентный спуск с дроблением шага, с использованием условия Армихо.**

1. Сравним эффективность методов с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.

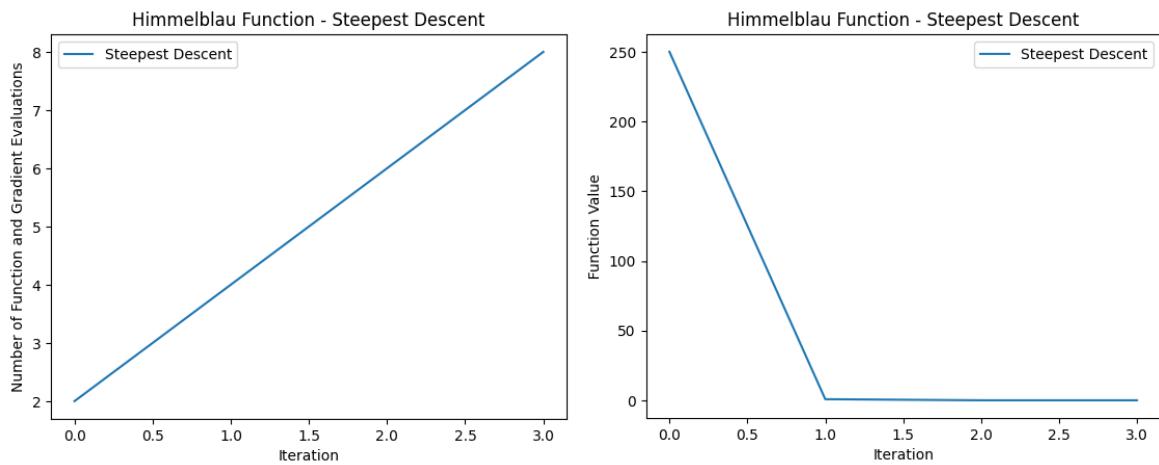


2. Нарисуем график с линиями уровня и траекториями методов.



### 3) Метод наискорейшего спуска с использованием метода Брента.

1. Сравним эффективность метода с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.



Для начальной точки [-5.5, 5.5]

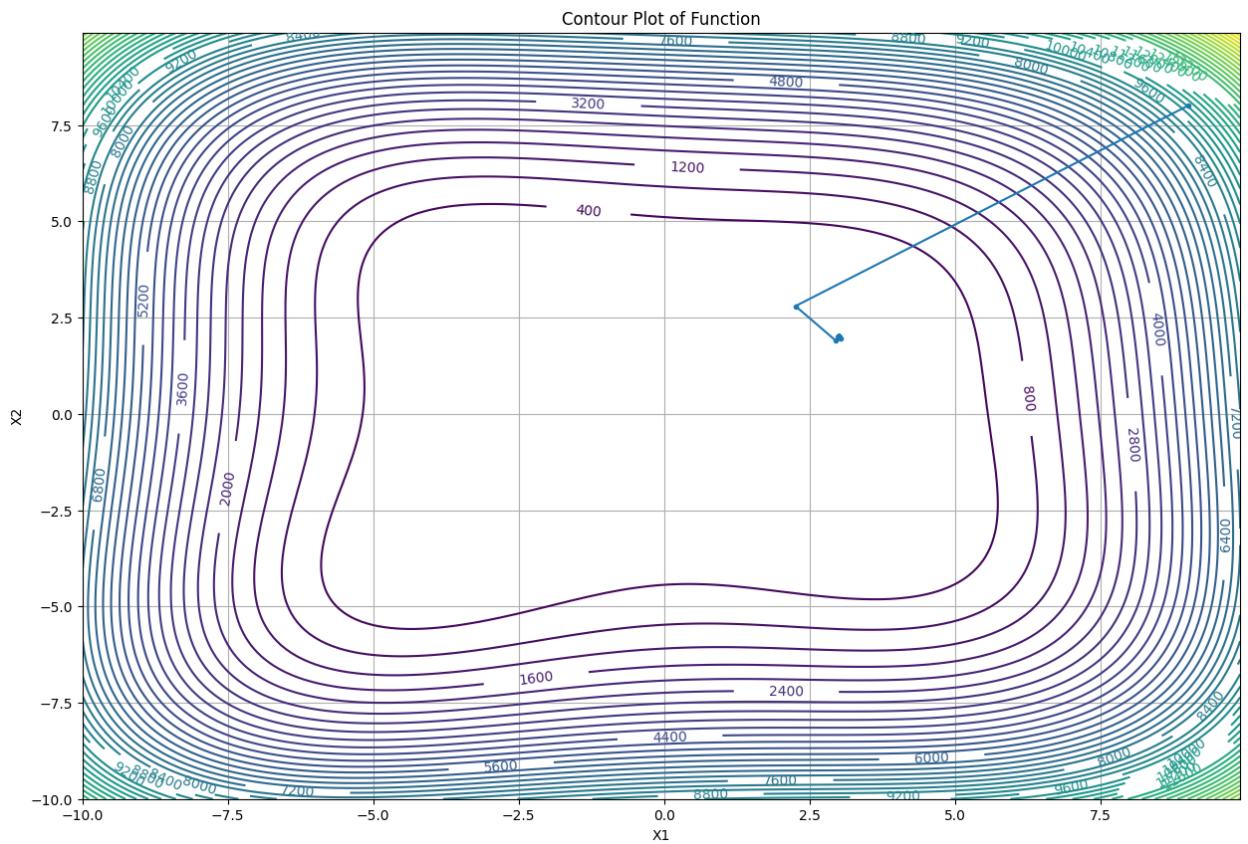
Iteration	x1	x2	s1	s2	f(x1,x2)	dstar	diff = fnext-fcurrent
0	-5.5	5.5	509	-440	927.625	0.00534	927.592
1	-2.78314	3.15144	-1.43991	-1.66572	0.0325738	0.01329	0.0321574
2	-2.80227	3.12931	-0.1819	0.15724	0.000416417	0.01425	0.000411978
3	-2.80486	3.13155	-0.0168	-0.01943	4.4392e-06	0.01331	4.39212e-06

Для начальной точки [9, 8]

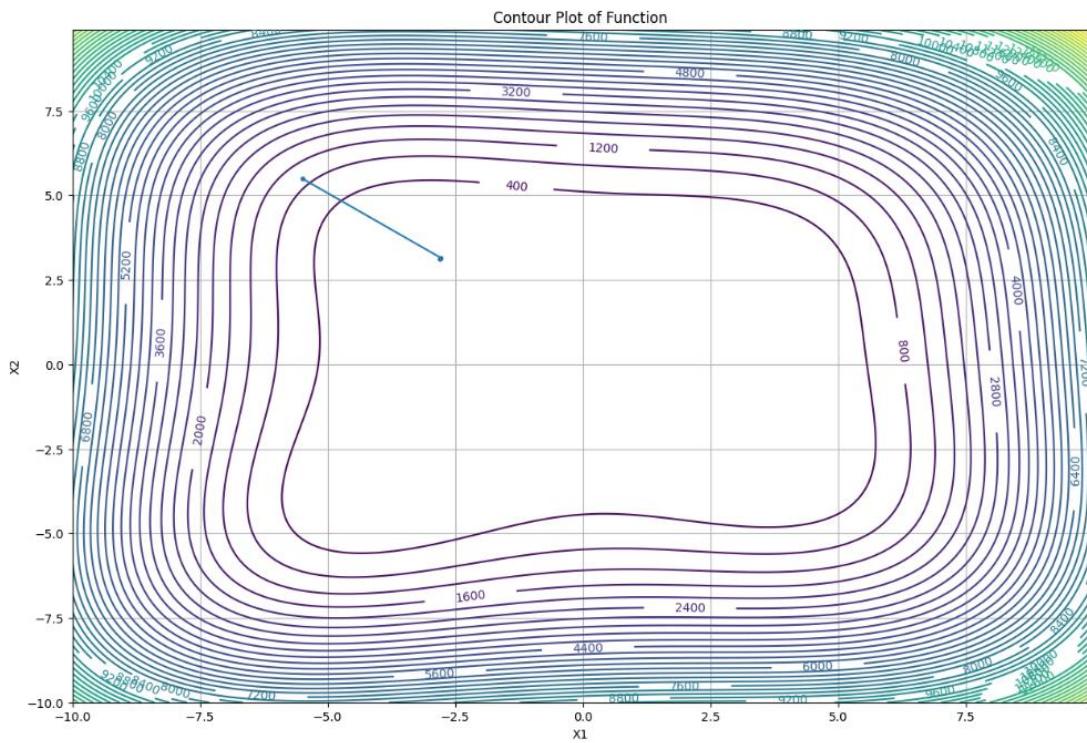
Iteration	x1	x2	s1	s2	f(x1,x2)	dstar	diff = fnext-fcurrent
0	9	8	-2940	-2268	10440	0.00229	10420.8
1	2.25718	2.7984	21.8735	-28.3545	19.1888	0.03181	18.8388
2	2.95307	1.89632	5.42678	4.18637	0.350059	0.01315	0.312218
3	3.02444	1.95138	-0.85756	1.11165	0.0378416	0.03446	0.0340231
4	2.99488	1.98969	0.58342	0.45006	0.00381851	0.01279	0.00347651
5	3.00235	1.99545	-0.08285	0.10739	0.000341999	0.03387	0.000311574
6	2.99954	1.99908	0.05223	0.04029	3.04253e-05	0.01276	2.77502e-05
7	3.00021	1.9996	-0.00734	0.00951	2.6751e-06	0.03381	2.44004e-06
8	2.99996	1.99992	0.00459	0.00354	2.351e-07	0.01275	2.14433e-07
9	3.00002	1.99996	-0.00064	0.00084	2.06e-08	0.03381	1.88232e-08
10	3	1.99999	0.0004	0.00031	1.8e-09	0.01275	1.65221e-09
11	3	2	-6e-05	7e-05	2e-10	0.03381	1.45008e-10
12	3	2	4e-05	3e-05	0	0.01275	1.27268e-11
13	3	2	-0	1e-05	0	0.03381	1.11698e-12
14	3	2	0	0	0	0.01275	9.80139e-14
15	3	2	-0	0	0	0.03381	8.60042e-15
16	3	2	0	0	0	0.01275	7.54659e-16
17	3	2	-0	0	0	0.0338	6.62086e-17
18	3	2	0	0	0	0.01275	5.81886e-18
19	3	2	-0	0	0	0.0338	5.11499e-19
20	3	2	0	0	0	0.01274	4.49517e-20
21	3	2	-0	0	0	0.03396	3.96926e-21

2. Нарисуем график с линиями уровня и траекториями методов.

Для начальной точки [9, 8]

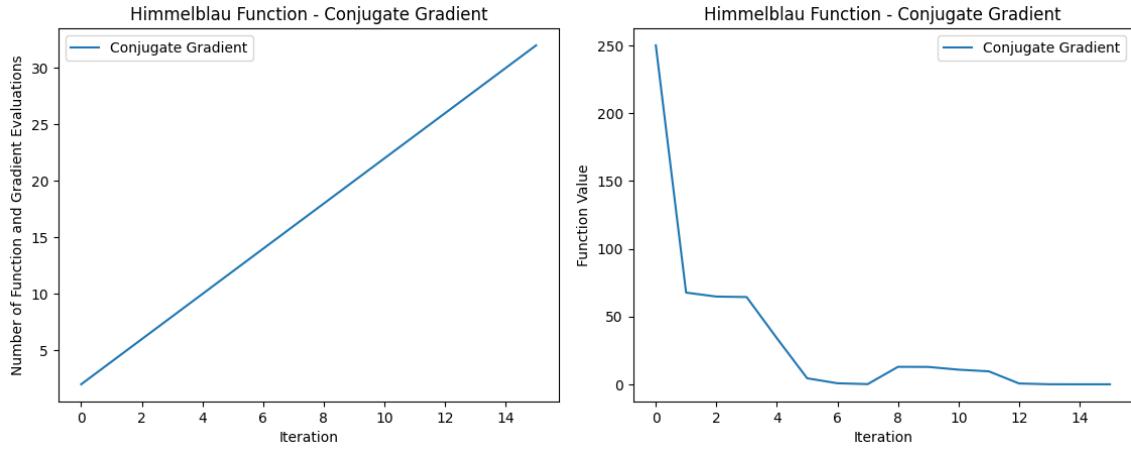


**Для начальной точки  $[-5, 5]$**



#### 4) Метод сопряженных градиентов.

- Сравним эффективность метода с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.



**Для начальной точки [-5, 5]**

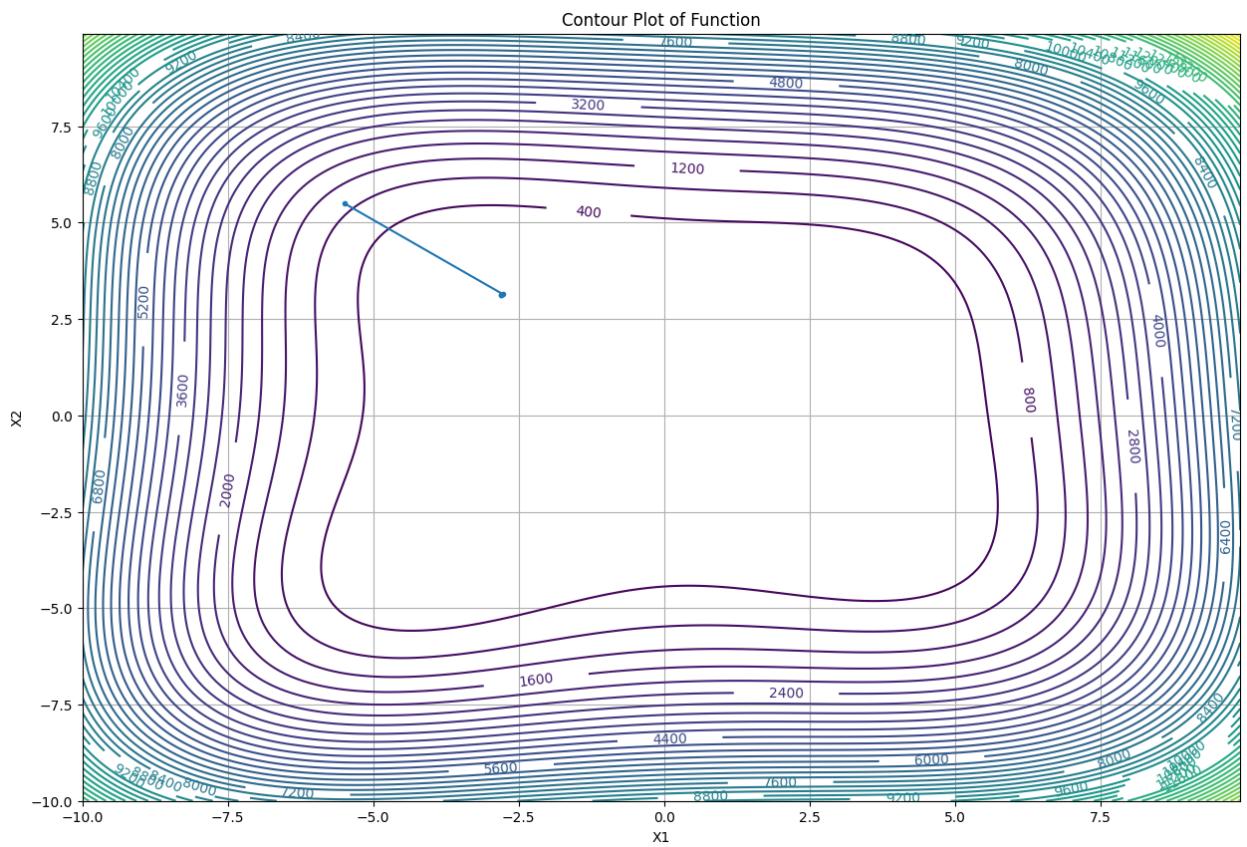
Iteration	x1	x2	s1	s2	f(x1,x2)	alpha_star	diff = fnext-fcurrent
0	-5.5	5.5	-1.97711	-1.0643	927.625	0.00535	927.588
1	-2.77438	3.14387	-0.15342	0.21665	0.0372538	0.01453	0.0367905
2	-2.80311	3.1284	0.00264	0.0021	0.000463357	0.01332	0.000463276
3	-2.80516	3.13129	0.00029	-0.00029	8e-08	0.01384	7.88597e-08

**Для начальной точки [9, 8]**

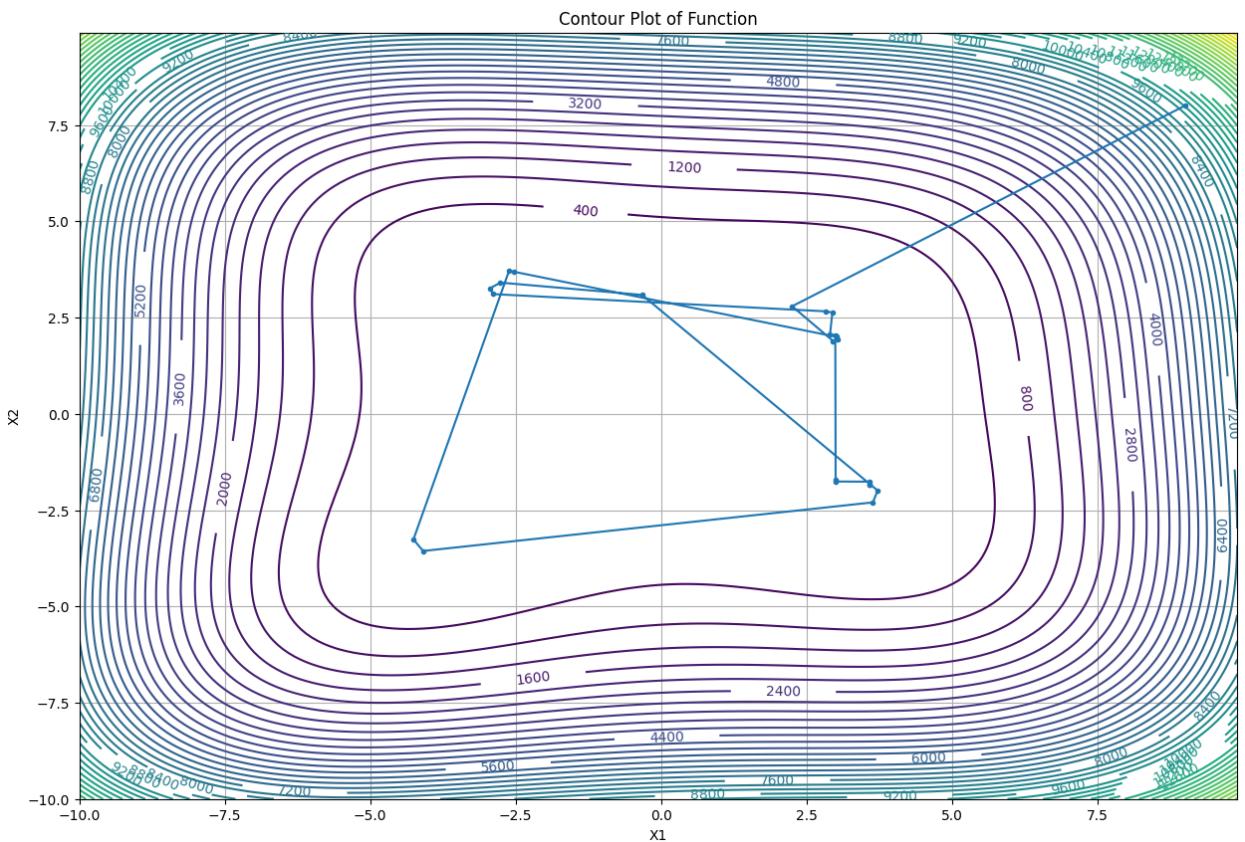
Iteration	x1	x2	s1	s2	f(x1,x2)	alpha_star	diff = fnext-fcurrent
0	9	8	21.91	-28.0303	10440	0.0023	10420.8
1	2.25048	2.79323	6.24855	3.20447	19.1912	0.03211	18.833
2	2.95393	1.89327	-0.41041	1.8442	0.358248	0.01205	0.289637
3	3.02924	1.93189	-1.17527	0.35854	0.0686118	0.03602	0.0612785
4	3.01446	1.99831	-2898.81	820.727	0.00733328	4.71087	17.4987
5	-2.52207	3.68733	-14.3149	-60.7682	17.506	3e-05	0.132513
6	-2.61449	3.71349	49.6183	-89.9261	17.6385	0.11477	2.31334
7	-4.25738	-3.26075	39.0883	6.38124	15.3252	0.00334	8.04311
8	-4.09182	-3.5608	5.15649	19.1297	7.28207	0.19776	3.58034
9	3.63841	-2.29883	-11.2931	14.1684	3.70173	0.01603	2.49621
10	3.72104	-1.99227	-26.8245	3.51633	1.20552	0.35861	64.5738
11	-0.32881	3.0887	-27.1496	-22.1643	65.7794	0.0907	62.355
12	-2.76185	3.40764	4.55832	-14.822	3.42433	0.0065	2.11142
13	-2.93829	3.26359	6.56042	-0.51411	1.3129	0.01037	1.04574
14	-2.89103	3.10991	138.849	-41.5072	0.267158	0.87274	8.3323
15	2.83454	2.66123	-2.36484	-32.5982	8.59946	0.00078	0.372219
16	2.94262	2.62892	5.52363	-1.383	8.22724	0.01723	7.93446
17	2.90187	2.06715	0.00219	-1.502	0.292785	0.01646	0.263093
18	2.9928	2.04439	48.8976	-1456.5	0.0296922	2.49955	14.9727
19	2.99827	-1.70993	46.9674	-0.42646	15.0024	3e-05	0.0466251
20	2.99983	-1.75636	0.11157	-2.33207	14.9558	0.01232	14.8526
21	3.5786	-1.76162	0.17778	-0.00437	0.103196	0.03706	0.183046
22	3.58273	-1.84803	0.00024	-0.00144	0.00014998	0.00952	0.000149943
23	3.58442	-1.84808	-0.00028	-0.00011	3.7e-08	0.03429	3.65528e-08

- Нарисуем график с линиями уровня и траекториями методов.

**Для начальной точки [-5, 5]**



**Для начальной точки [9, 8]**

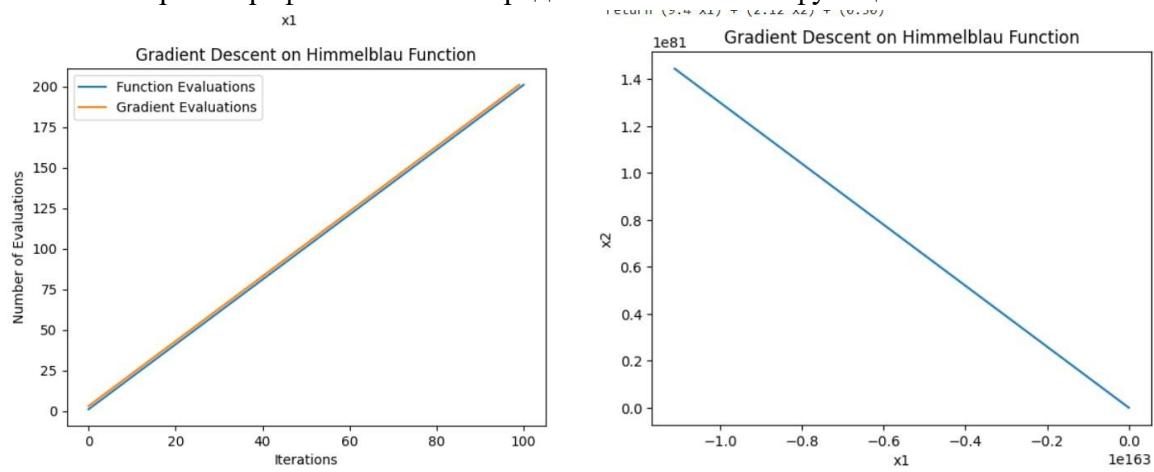


## Вторая функция

Для второй функции:

### 1) Градиентный спуск с постоянным шагом.

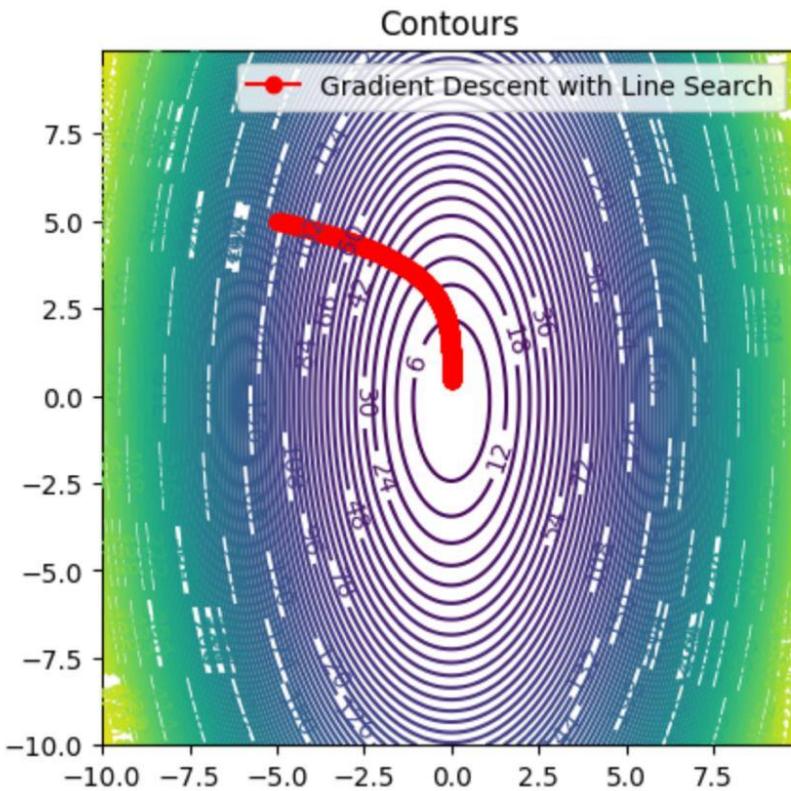
1. Рассмотрим сходимость градиентного спуска с постоянным шагом. Для этого построим график изменения градиента и значений функции от количества итераций.



2. Сравним эффективность методов с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.

Iteration	x1	x2	current_point	grad
0	-5	[- 5 5]	[145.89 -36.1 ]	[- 6.4589 5.361 ]
1	-6.4589	[-6.4589 5.361 ]	[228.53476955 -49.04834 ]	[ -8.7442477 5.8514834 ]
2	-8.74425	[-8.7442477 5.8514834]	[397.81047295 -69.49078353]	[ -12.72235242 6.54639124 ]
3	-12.7224	[-12.72235242 6.54639124]	[ 808.51425063 -105.41176338]	[ -20.80749493 7.60050887 ]
4	-20.8075	[-20.80749493 7.60050887]	[2098.77762481 -179.17737355]	[ -41.79527118 9.3922826 ]
5	-41.7953	[-41.79527118 9.3922826 ]	[8306.88561253 -372.66390996]	[ -124.8641273 13.1189217 ]
6	-124.864	[-124.8641273 13.1189217]	[73464.69450102 -1145.61068265]	[ -859.51107231 24.57502853 ]
7	-859.511	[-859.51107231 24.57502853]	[3472816.56258669 -8027.00501927]	[ -35587.67669818 104.84507872 ]
8	-35587.7	[-35587.67669818 104.84507872]	[ 5.95248053e+09 -3.34301589e+05]	[ -5.95603930e+07 3.44786097e+03 ]
9	-5.95604e+07	[-5.95603930e+07 3.44786097e+03]	[ 1.66729699e+16 -5.59860384e+08]	[ -1.66729759e+14 5.60205170e+06 ]
10	-1.6673e+14	[-1.66729759e+14 5.60205170e+06]	[ 1.30654419e+29 -1.56725972e+15]	[ -1.30654419e+27 1.56726028e+13 ]

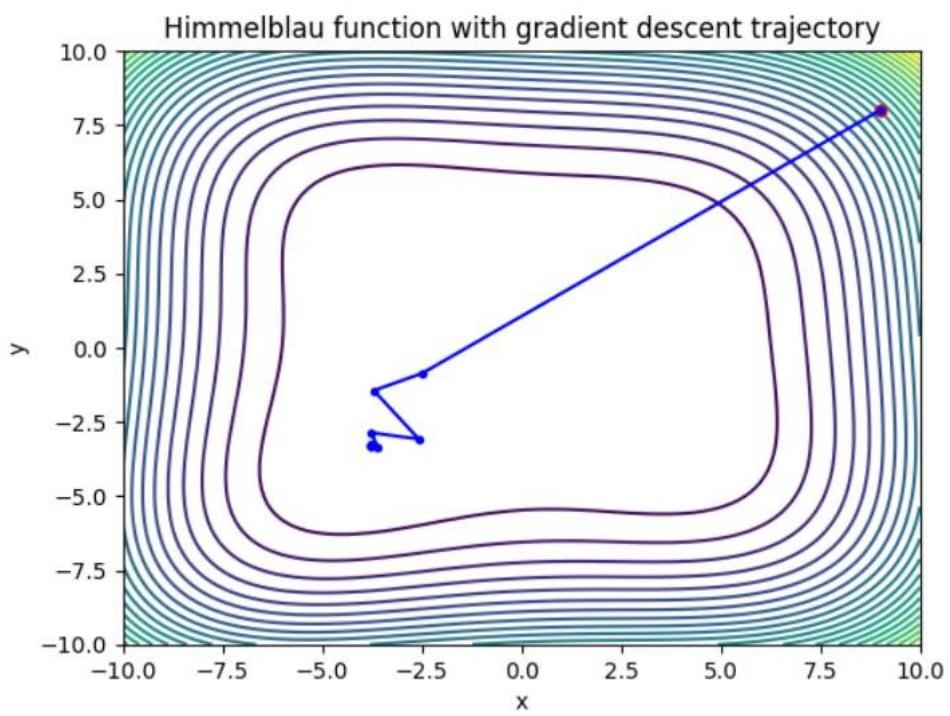
3. Нарисуем график с линиями уровня и траекториями методов.



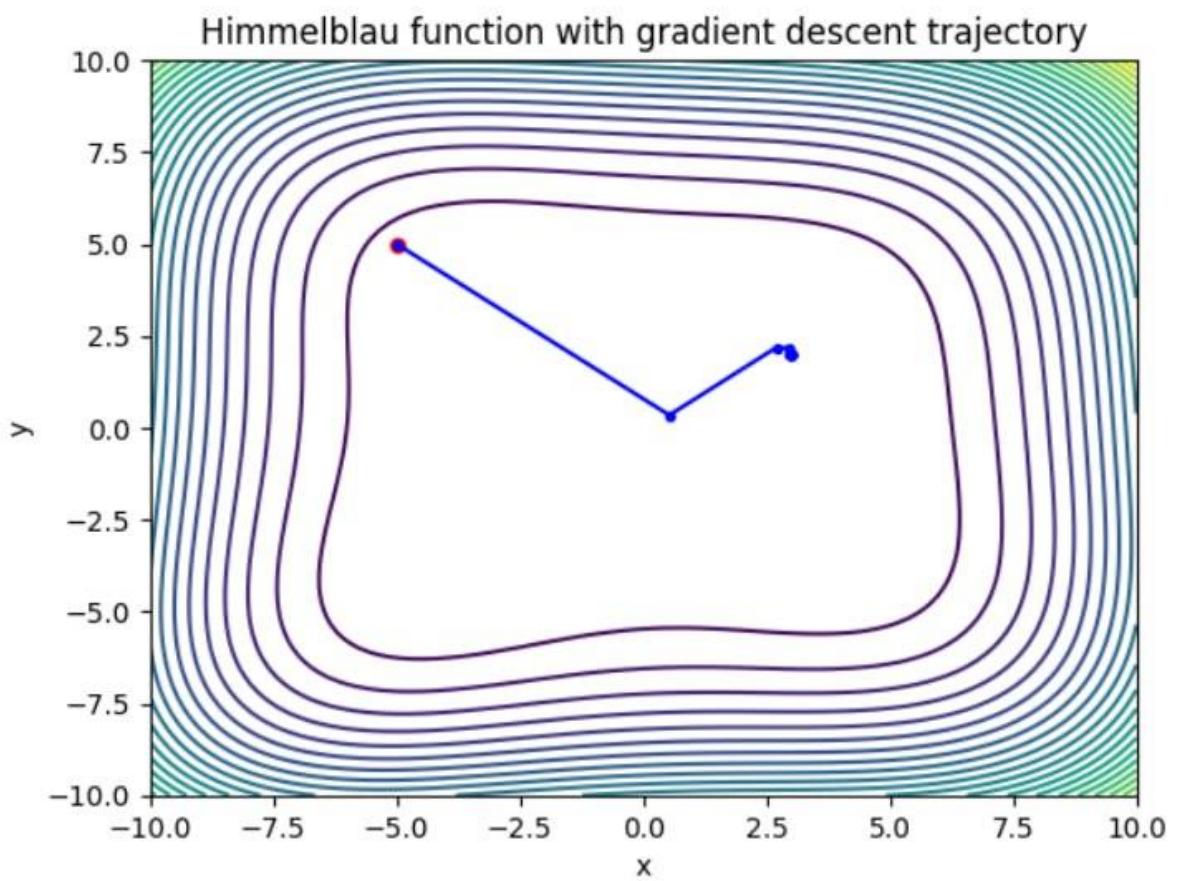
## 2) Градиентный спуск с дроблением шага, использующая условие Армихо.

1. Нарисуем график с линиями уровня и траекториями методов.

Для начальной точки [9, 8]

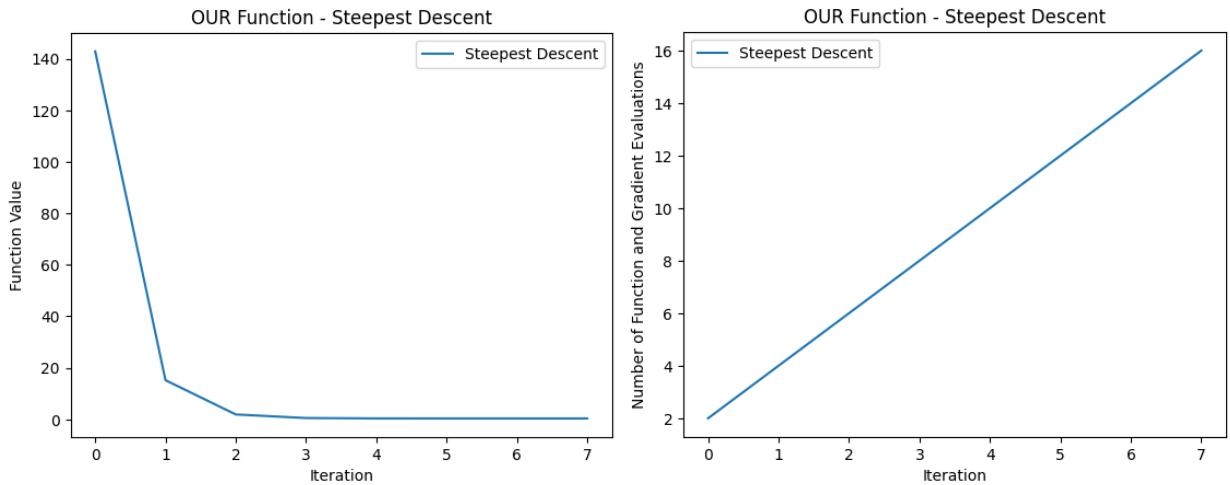


Для начальной точки [-5, 5]



### 3) Метод наискорейшего спуска с использованием метода Брента.

- Сравним эффективность методов с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.



Для начальной точки [-5, 5]

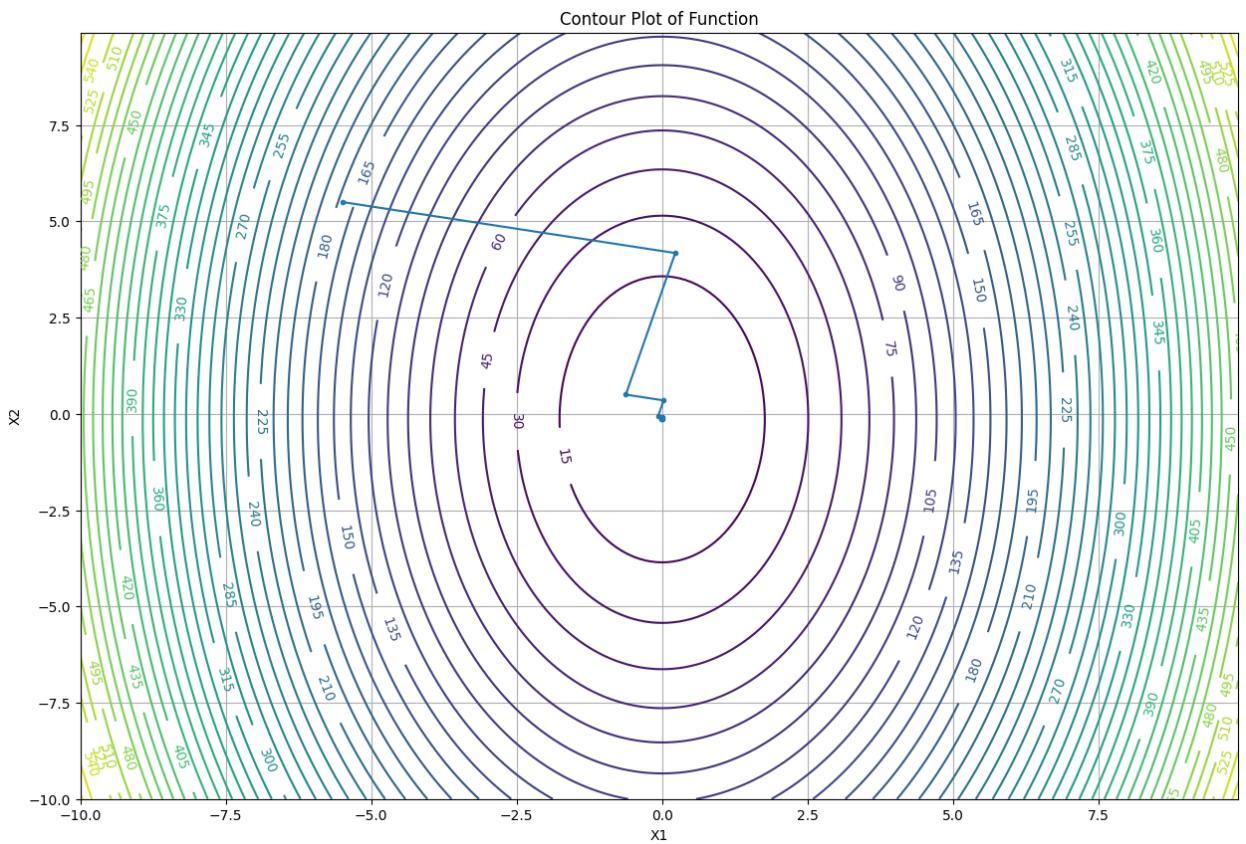
Iteration	x1	x2	s1	s2	f(x1,x2)	dstar	diff = fnext-fcurrent
0	-5.5	5.5	51.7	-11.96	176.28	0.11074	155.918
1	0.22524	4.17555	-2.11722	-9.15218	20.3625	0.40164	17.7213
2	-0.62512	0.49969	5.87611	-1.35935	2.64122	0.11074	2.01416
3	0.0256	0.34916	-0.24064	-1.04022	0.627055	0.40164	0.228925
4	-0.07105	-0.06863	0.66787	-0.1545	0.398129	0.11074	0.0260192
5	0.00291	-0.08574	-0.02735	-0.11823	0.37211	0.40164	0.00295729
6	-0.00808	-0.13323	0.07591	-0.01756	0.369153	0.11074	0.000336118
7	0.00033	-0.13517	-0.00311	-0.01344	0.368817	0.40164	3.82024e-05

Для начальной точки [9, 8]

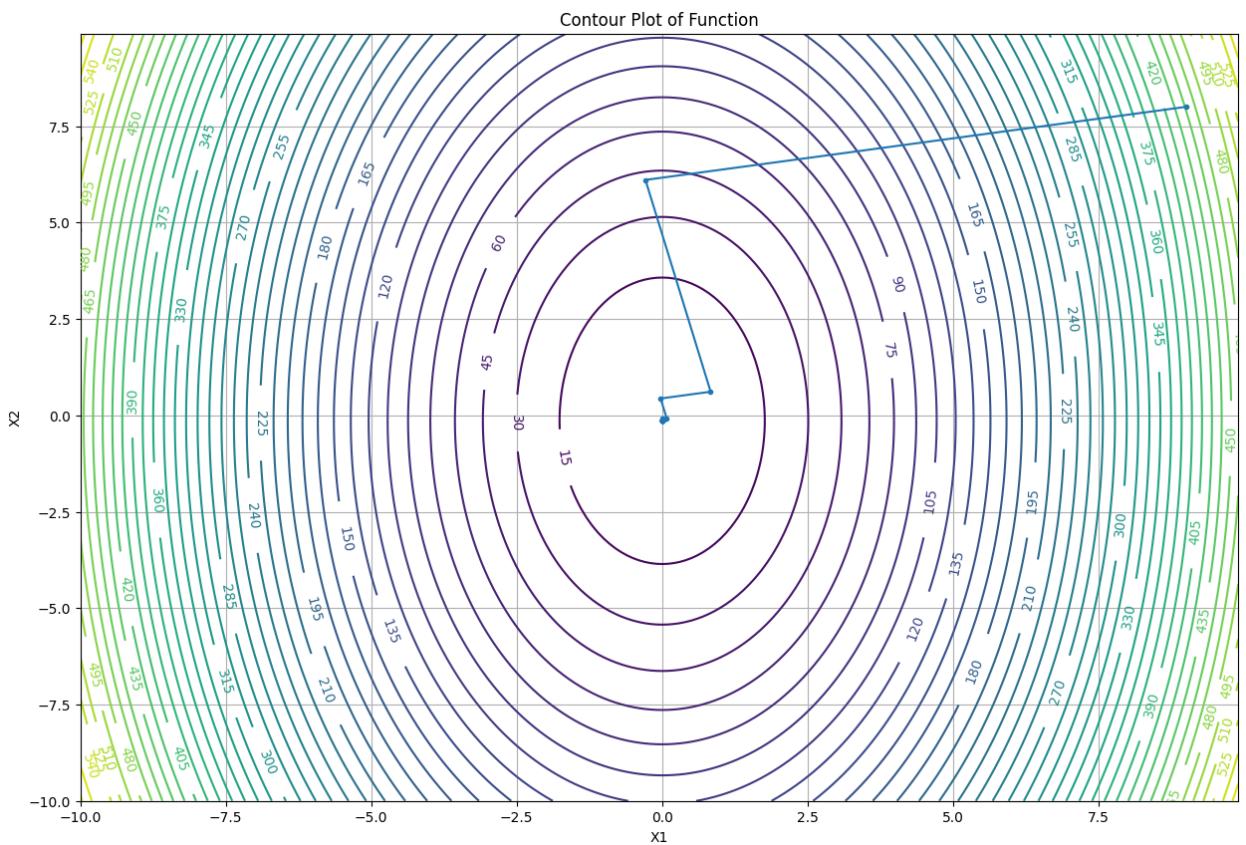
Iteration	x1	x2	s1	s2	f(x1,x2)	dstar	diff = fnext-fcurrent
0	9	8	-84.6	-17.26	451.33	0.10978	409.21
1	-0.28743	6.10519	2.70182	-13.243	42.1196	0.41478	37.8854
2	0.83324	0.61225	-7.83242	-1.59796	4.23414	0.10978	3.5075
3	-0.02661	0.43682	0.25014	-1.22606	0.726636	0.41478	0.324731
4	0.07714	-0.07173	-0.72514	-0.14794	0.401905	0.10978	0.0300642
5	-0.00246	-0.08797	0.02316	-0.11351	0.371841	0.41478	0.0027834
6	0.00714	-0.13505	-0.06713	-0.0137	0.369058	0.10978	0.000257692
7	-0.00023	-0.13655	0.00214	-0.01051	0.3688	0.41478	2.38576e-05
8	0.00066	-0.14091	-0.00622	-0.00127	0.368776	0.10978	2.20879e-06
9	-2e-05	-0.14105	0.0002	-0.00097	0.368774	0.41479	2.04501e-07
10	6e-05	-0.14145	-0.00058	-0.00012	0.368774	0.10978	1.89295e-08
11	-0	-0.14147	2e-05	-9e-05	0.368774	0.41479	1.7523e-09
12	1e-05	-0.14145	-5e-05	-1e-05	0.368774	0.1098	1.62208e-10
13	-0	-0.14151	0	-1e-05	0.368774	0.41414	1.49981e-11
14	0	-0.14151	-0	-0	0.368774	0.10983	1.40399e-12
15	-0	-0.14151	0	-0	0.368774	0.41415	1.3145e-13
16	0	-0.14151	-0	-0	0.368774	0.10983	1.23235e-14
17	-0	-0.14151	0	-0	0.368774	0.40671	1.16573e-15
18	0	-0.14151	-0	-0	0.368774	0.08478	5.55112e-17
19	0	-0.14151	-0	-0	0.368774	0	0

- Нарисуем график с линиями уровня и траекториями методов.

Для начальной точки [-5, 5]

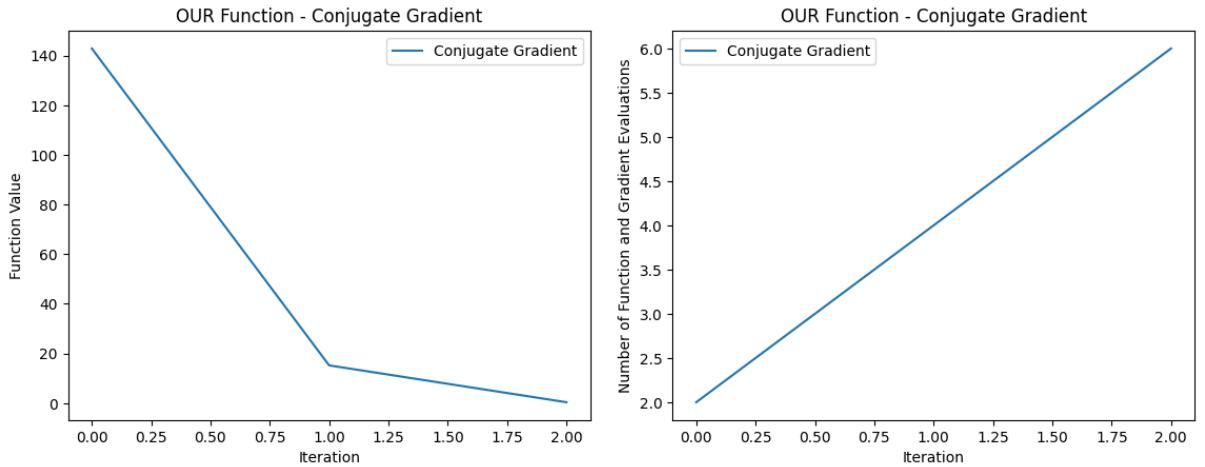


**Для начальной точки  $[9, 8]$**



#### 4) Метод сопряженных градиентов.

- Сравним эффективность методов с точки зрения количества вычислений минимизируемой функции. Исследуем работу методов в зависимости от выбора начальной точки.



**Для начальной точки [-5, 5]**

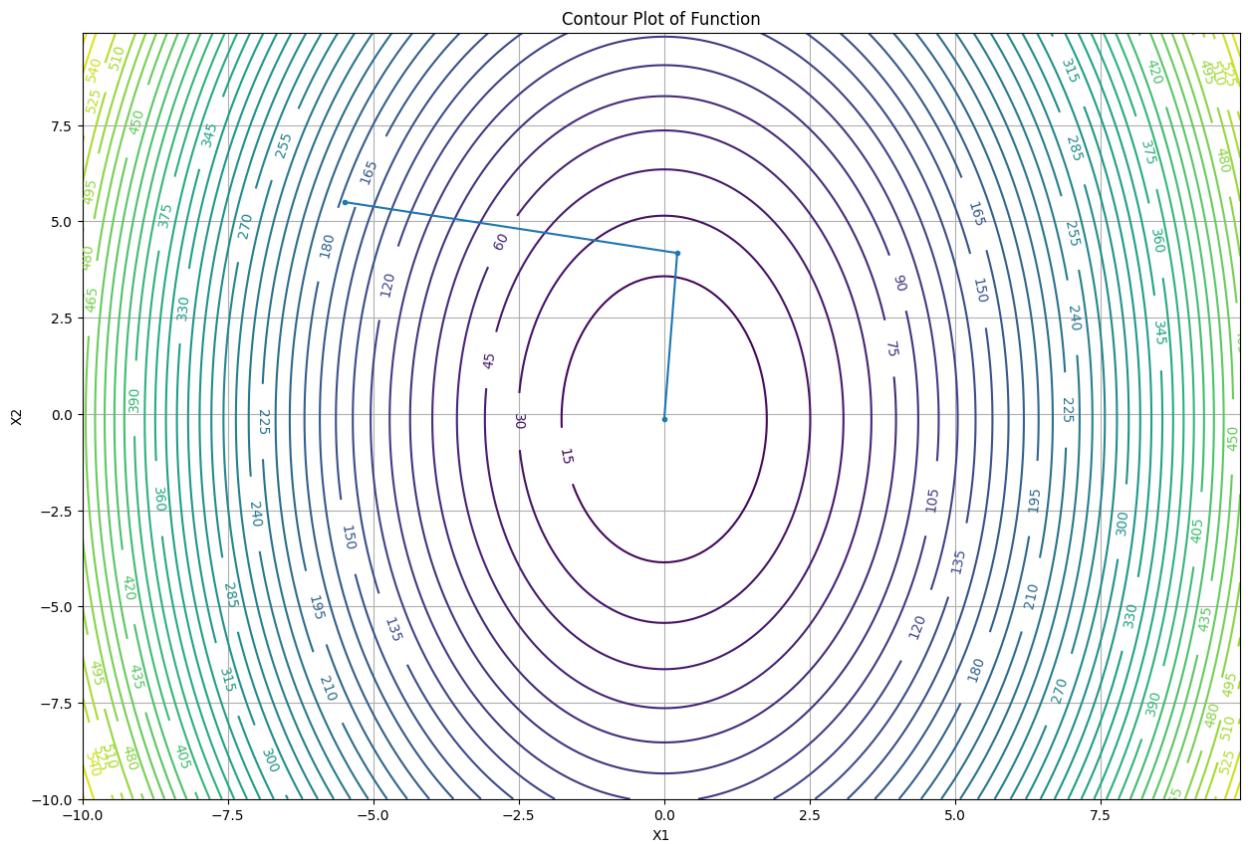
Iteration	x1	x2	s1	s2	f(x1,x2)	alpha_star	diff = fnext-fcurrent
0	-5.5	5.5	-0.49343	-9.52712	176.28	0.11073	155.918
1	0.22483	4.17565	-0.01141	0.00065	20.3625	0.45318	19.9937
2	0.00121	-0.14182	1e-05	0.00051	0.368781	0.10666	6.95887e-06

**Для начальной точки [9, 8]**

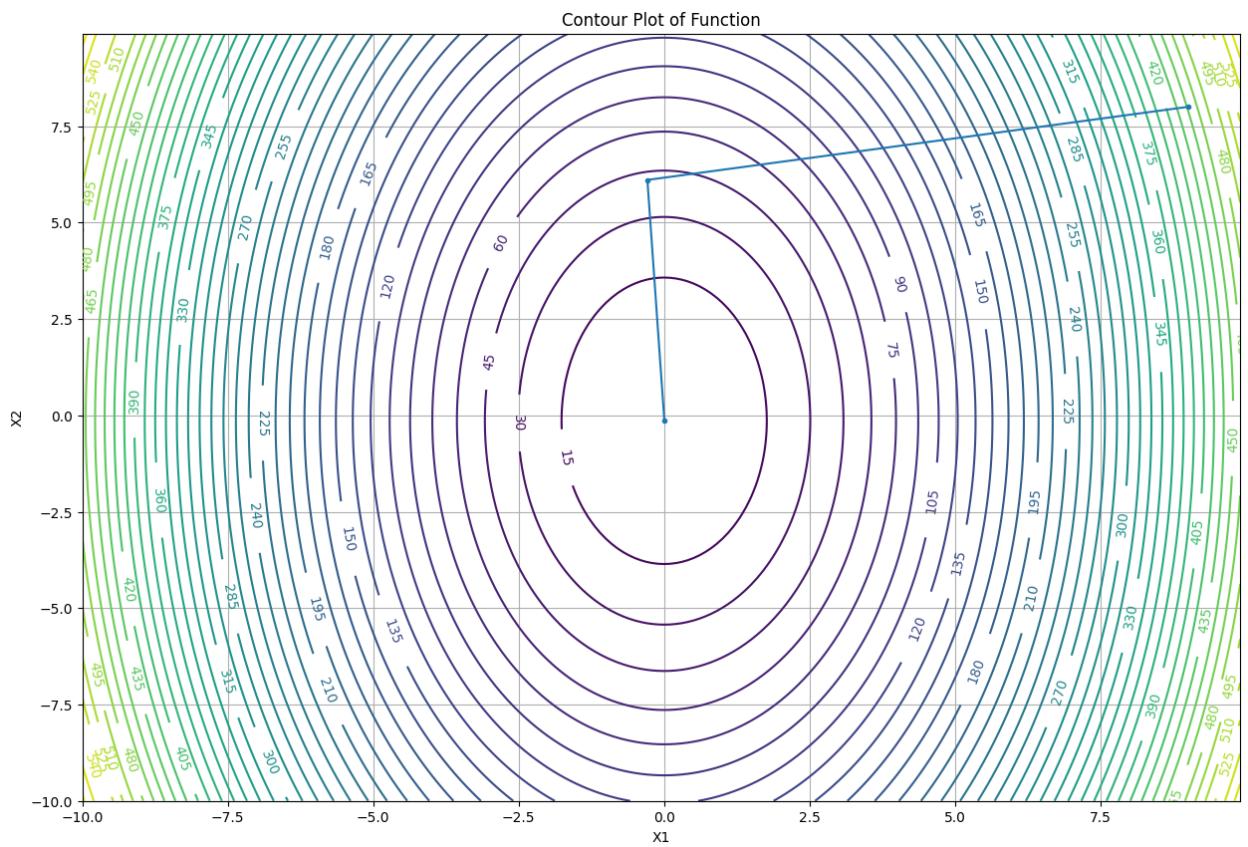
Iteration	x1	x2	s1	s2	f(x1,x2)	alpha_star	diff = fnext-fcurrent
0	9	8	0.63894	-13.6655	451.33	0.10979	409.21
1	-0.28856	6.10496	-0.03258	-0.00153	42.1196	0.45705	41.7508
2	0.00347	-0.14083	1e-05	-0.0011	0.368831	0.10655	5.669e-05

- Нарисуем график с линиями уровня и траекториями методов.

**Для начальной точки [-5, 5]**



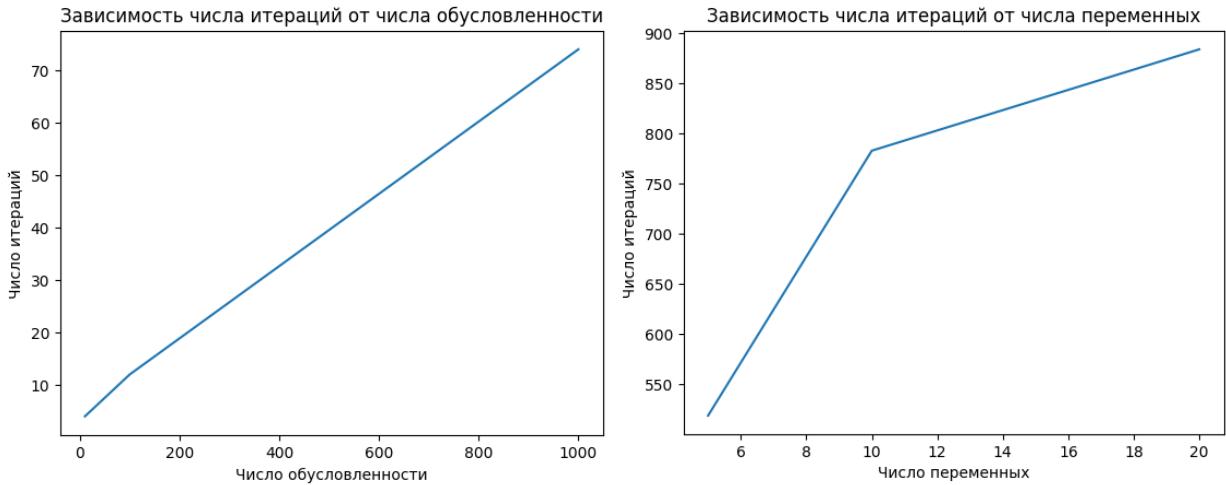
**Для начальной точки [9, 8]**



## Генератор квадратичных функций

Написав генератор случайный квадратичных функций и меняющимся количеством переменных и числом обусловленности, установили зависимость от числа итераций и размерности пространства, а также самим числом обусловленности.

Для наглядного представления, построили графики:



Таким образом, при использовании градиентного спуска с постоянным шагом для более сложных функций с большим количеством переменных, может потребоваться значительное количество итераций для достижения оптимального решения.

Число обусловленности матрицы характеризует ее чувствительность к небольшим изменениям входных данных, и может быть определено как отношение ее наибольшего и наименьшего сингулярных значений. Чем больше число обусловленности матрицы, тем более вытянутой (или "испорченной") будет ее форма, и тем больше итераций потребуется для достижения оптимального решения.

В частности, для квадратичной функции с матрицей, число обусловленности которой равно  $k$ , градиентный спуск с постоянным шагом потребует  $O(k)$  итераций для достижения оптимального решения. Таким образом, при увеличении числа обусловленности матрицы квадратичной функции, количество итераций для достижения оптимального решения также увеличивается.

Однако, для более сложных функций, зависимость числа итераций от числа обусловленности может быть более сложной, и может зависеть от других факторов, таких как выбранный шаг и точность, которую мы хотим достичь.

Таким образом, при использовании градиентного спуска с постоянным шагом для более сложных функций с высоким числом обусловленности матрицы, может потребоваться значительное количество итераций для достижения оптимального решения.

## **Вывод**

Метод градиентного спуска с постоянным шагом является простым и популярным методом оптимизации, но он может иметь низкую скорость сходимости и требовать большого количества вычислений функции и её градиента, особенно для функций с вытянутыми контурами.

Метод градиентного спуска с дроблением шага с условием Армихо может быть более эффективным, поскольку он позволяет динамически изменять шаг, достигая при этом точности, определённой условием Армихо, что позволяет ускорить сходимость метода. Это улучшает скорость сходимости метода и позволяет избежать проблемы слишком большого или слишком маленького шага.

Метод сопряженных градиентов может быть эффективным для квадратичных функций, но для общих функций он может иметь сходимость медленнее, чем метод градиентного спуска с дроблением шага и условием Армихо. Он позволяет быстрее достигать оптимальной точки и требует меньшего количества вычислений минимизируемой функции.

Метод наискорейшего спуска с использованием метода Брента может быть более эффективным для функций с вытянутыми контурами, поскольку он позволяет быстро определить оптимальный шаг и увеличить скорость сходимости.

Как правило, для достижения наилучшей эффективности и минимизации количества вычислений функции и её градиента, рекомендуется использовать метод градиентного спуска с дроблением шага и условием Армихо, поскольку он позволяет динамически изменять шаг и достигать при этом точности, определённой условием Армихо, что позволяет ускорить сходимость метода.

Эффективность методов градиентного спуска сильно зависит от начальной точки. Некоторые методы могут сходиться к оптимальному решению из любой начальной точки, тогда как другие могут сходиться только при определенных условиях.

Метод градиентного спуска с постоянным шагом может иметь низкую скорость сходимости при выборе неправильной начальной точки, особенно для функций с вытянутыми контурами.

Метод градиентного спуска с дроблением шага и условием Армихо может быть более устойчивым к выбору начальной точки, поскольку он позволяет динамически изменять шаг и достигать при этом точности, определённой условием Армихо.

Метод сопряженных градиентов может иметь проблемы с сходимостью, если начальная точка не является "хорошой".

Метод наискорейшего спуска с использованием метода Брента может быть более устойчивым к выбору начальной точки, поскольку он позволяет быстро определить оптимальный шаг и увеличить скорость сходимости.

Как правило, для достижения наилучшей эффективности при выборе начальной точки, рекомендуется использовать метод градиентного спуска с дроблением шага и условием Армихо, поскольку он позволяет динамически изменять шаг и достигать при этом точности, определённой условием Армихо, что позволяет ускорить сходимость метода и сделать его более устойчивым к выбору начальной точки.