

Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации
Отчет по лабораторной работе №1

Работу выполнил:
Абрамов Илья Дмитриевич,
Группа М32341

Преподаватель:
Ким. С. Е.

Санкт-Петербург
2023

Постановка задачи

1. Реализуйте градиентный спуск с постоянным шагом (learning rate).
2. Реализуйте метод одномерного поиска (метод дихотомии, метод Фибоначчи, метод золотого сечения) и градиентный спуск на его основе.
3. Проанализируйте траекторию градиентного спуска на примере квадратичных функций. Для этого придумайте две-три квадратичные функции от двух переменных, на которых работа методов будет отличаться.
4. Для каждой функции:
 - (а) исследуйте сходимость градиентного спуска с постоянным шагом, сравните полученные результаты для выбранных функций;
 - (б) сравните эффективность градиентного спуска с использованием одномерного поиска с точки зрения количества вычислений минимизируемой функции и ее градиентов;
 - (с) исследуйте работу методов в зависимости от выбора начальной точки;
 - (д) исследуйте влияние нормализации (scaling) на сходимость на примере масштабирования осей плохо обусловленной функции;
 - (е) в каждом случае нарисуйте графики с линиями уровня и траекториями методов;
5. Реализуйте генератор случайных квадратичных функций n переменных с числом обусловленности k .
6. Исследуйте зависимость числа итераций $T(n,k)$, необходимых градиентному спуску для сходимости в зависимости от размерности пространства $2 \leq n \leq 10^3$ и числа обусловленности оптимизируемой функции $1 \leq k \leq 10^3$.
7. Для получения более корректных результатов проведите множественный эксперимент и усредните полученные значения числа итераций.
8. Реализуйте одномерный поиск с учетом условий Вольфе и исследуйте его эффективность. Сравните полученные результаты с реализованными ранее методами.

Цель работы:

1. Научиться программно реализовывать метод градиентного спуска с постоянным шагом, метод градиентного спуска с использованием методов одномерного поиска, метод градиентного спуска с учетом условий Вольфе.
2. Сравнить производительность перечисленных методов на примере квадратичных функций
3. Исследовать влияние нормализации на сходимость градиентного спуска.

4. Градиентный спуск с постоянным шагом.

Реализован в функции optimization.optimize.gradient_descent (листинг 1, 2). На вход передаются параметры: lr – шаг градиентного спуска, epoch – количество итераций, start – начальная точка, function – функция, минимум которой хотим найти. На каждом шаге вычисляется градиент функции и происходит смещение текущего положения точки в сторону, обратную направлению градиента.

Листинг 1

```
optimization.optimize.gradient_descent
def gradient_descent(lr, epoch, start, function,
linear_search=None, logging=False, draw=False, save_image=False):
    points = np.zeros((epoch, len(start)))
    points[0] = start
    for i in range(1, epoch):
        start = grad_descent_once(function, lr, start,
linear_search=linear_search)
        points[i] = start

    if logging:
        print(points)
    if draw:
        t = np.linspace(-50, 50, 1000)
        X, Y = np.meshgrid(t, t)
        plt.plot(points[:, 0], points[:, 1], 'o-')
        plt.contour(X, Y, function.calc(np.array([X, Y])), levels=sorted([function.calc(p) for p in points]))
        if save_image.__class__ is str:
            plt.savefig(save_image)
        plt.show()

    return start
```

Листинг 2

```
optimization.optimize.gradient_descent_once
def grad_descent_once(function, lr, start, linear_search=None):
    grad = function.grad(start)
    right = start - lr * np.array(grad)

    match linear_search:
        case "dichotomy":
            epoch = 5
            return dichotomy(start, right, grad, epoch, lr / (2 ** epoch), function)
        case "wolfe":
            return wolfe(start, function)
        case _:
            return right
```

5. Метод дихотомии.

Реализован в функции optimization.optimize.dichotomy (листинг 3). На вход передаются параметры: left, right – левая и правая границы

отрезка, на котором нужно найти минимум функции соответственно, epoch - число итераций для вычисления минимума, lr - шаг, с которым точка двигается в сторону минимума, grad – градиент в левом конце отрезка, function – функция, минимум на отрезке которой хотим найти. На каждой итерации цикла: находим точку в середине отрезка. Находим точки, находящиеся ближе к каждому из концов отрезка. Перемещаем конец отрезка, соответствующая которому точка посередине больше, в середину отрезка. Так как мы работаем с квадратичными функциями, на каждом шаге мы будем выбирать ту половину отрезка, на котором точно нет максимума.

Листинг 3

```
optimization.optimize.dichotomy
def dichotomy(left, right, grad, epoch, lr, function):
    for _ in range(epoch):
        mid = (left + right) / 2
        mid1 = mid + grad * lr / 2 # closer to l
        mid2 = mid - grad * lr / 2 # closer to r
        value1 = function.calc(mid1)
        value2 = function.calc(mid2)
        if value1 > value2:
            left = mid
        else:
            right = mid
    return right
```

6. Траектории градиентного спуска.

$$f(x, y) = 16x^2 + y^2 + 4xy$$

$$f_{min}(x, y) = (0, 0)$$

Рассмотрим поведение градиентного спуска с различными параметрами lr и epoch = 20.

lr = 0.06:

Метод с постоянным шагом приходит в точку (-5.4, 0.47) (рис. 1)

Метод с использованием дихотомии приходит в точку (-0.2, 1.2) (рис. 2)

Видно, что градиентный спуск с использованием метода дихотомии приблизился к минимуму лучше, чем метод с постоянным шагом.

lr = 0.1:

Градиентный спуск с постоянным шагом расходится при таком значении lr. Градиентный спуск с использованием метода дихотомии приходит в точку (-0.21, 1.19) (рис. 3).

lr = 0.01:

Оба метода приходят в точку (-1.45, 11.1) (рис. 4)

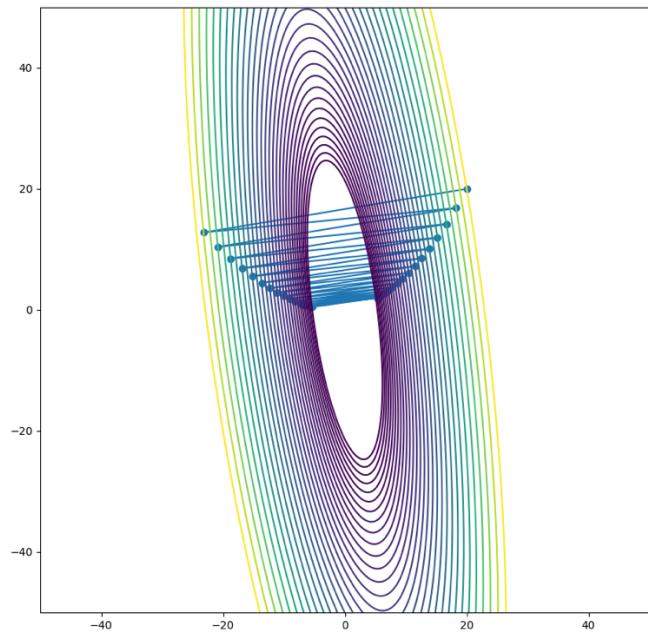


Рисунок 1

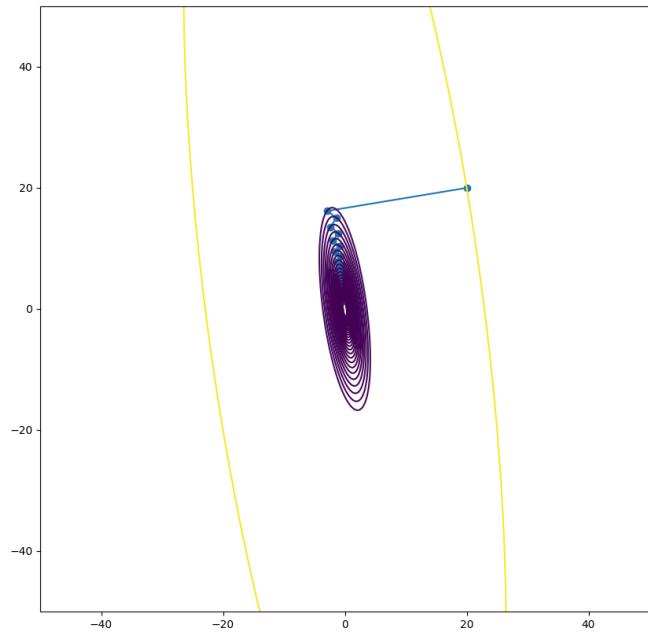


Рисунок 2

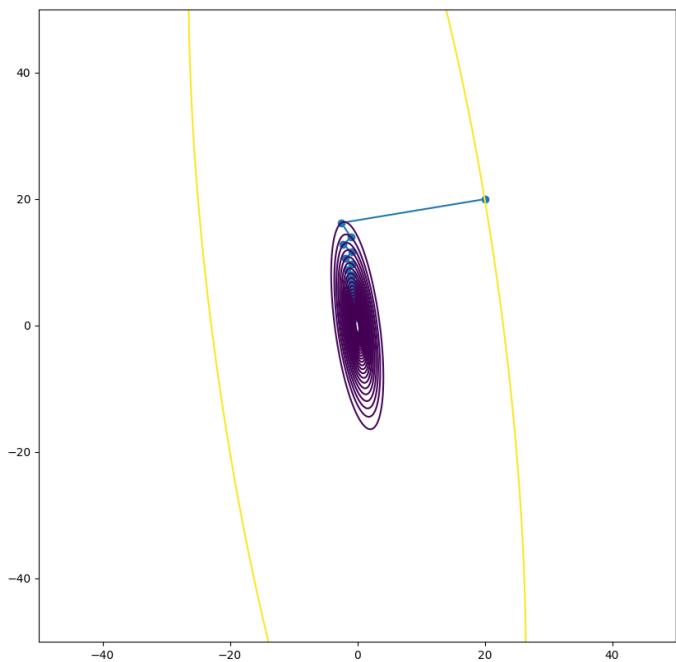


Рисунок 3

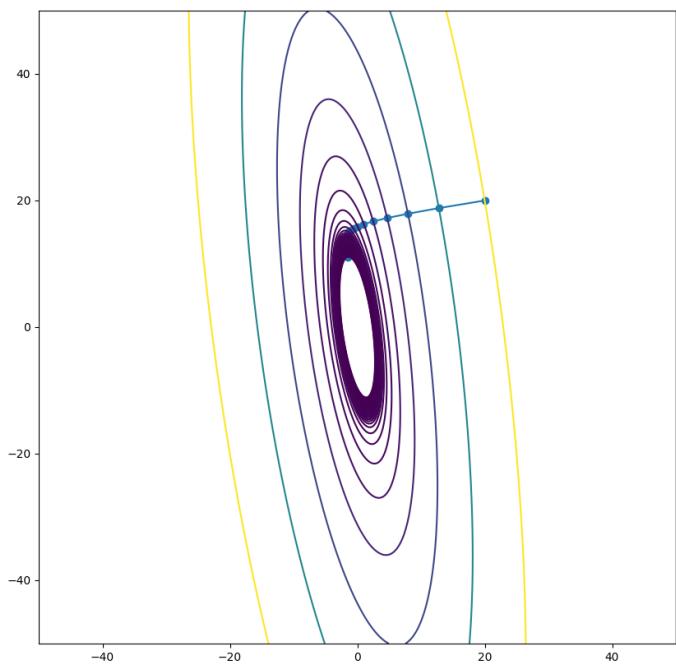


Рисунок 4

$$g(x, y) = 100(x - 2)^2 + (y + 1)^2$$

$$g_{min}(x, y) = (2, -1)$$

Рассмотрим поведение градиентного спуска с различными параметрами lr и epoch = 30.

lr = 0.01:

Метод с постоянным шагом приходит в точку (-16, 10.6) (рис. 5)

Метод с использованием дихотомии приходит в точку (1.9, 10.9) (рис. 6). Оба метода приходят в далекие от минимума точка.

lr = 0.05:

Градиентный спуск с постоянным шагом расходится при таком значении lr. Градиентный спуск с использованием метода дихотомии приходит в точку (-0.21, 1.19).

lr = 0.01:

Оба метода приходят в точку (2, 18.8) (рис. 7)

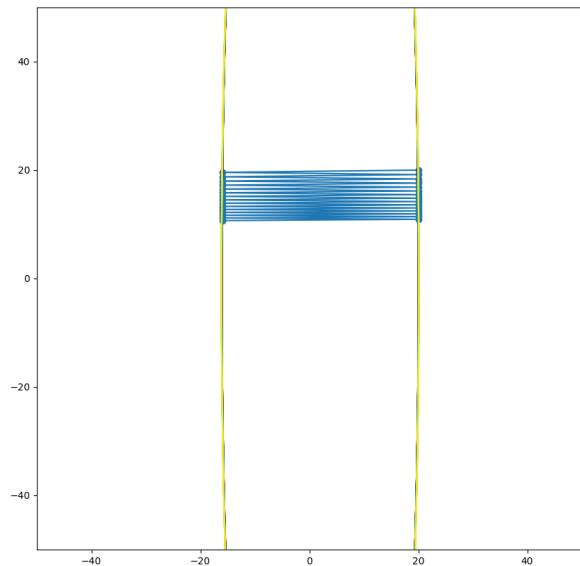


Рисунок 5

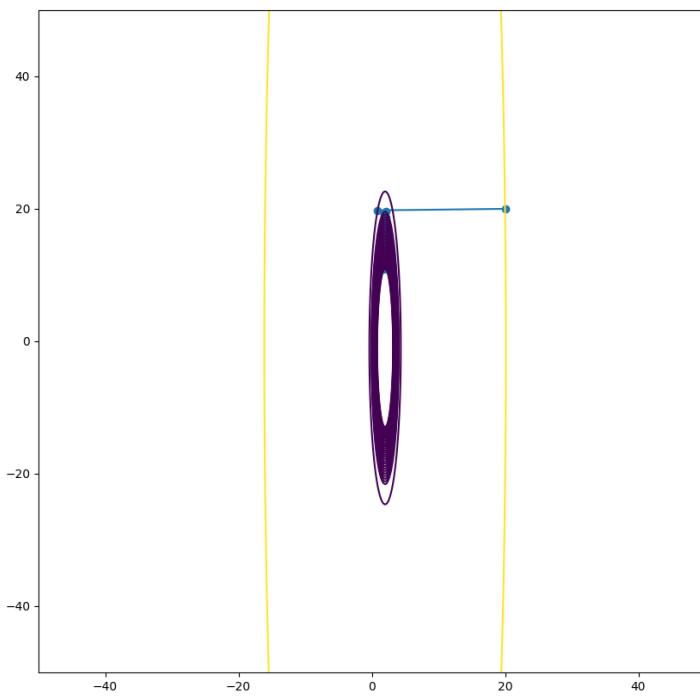


Рисунок 6

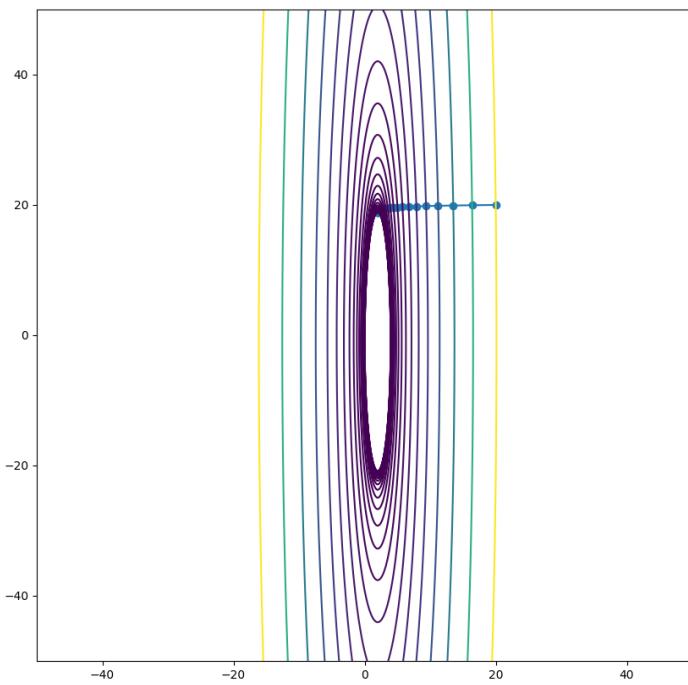


Рисунок 7

Описанные выше наблюдения говорят о том, что в случае квадратичных функций применение метода дихотомии может помочь сойтись градиентному спуску в случае больших l_r и не дает существенных преимуществ при небольших.

7. Сходимость градиентного спуска с постоянным шагом

Выберем несколько шагов для первой функции, на которых сходимость различается: [0.007, 0.06, 0.1]. Получили соответственно следующие результаты: 534, 179, метод расходится. Графики представлены на рис. 8, рис. 9 соответственно.

Выберем несколько шагов для второй функции, на которых сходимость различается: [0.001, 0.005, 0.007]. Получили соответственно следующие результаты: 1867, 372, метод расходится. Графики представлены на рис. 10, рис. 11 соответственно.

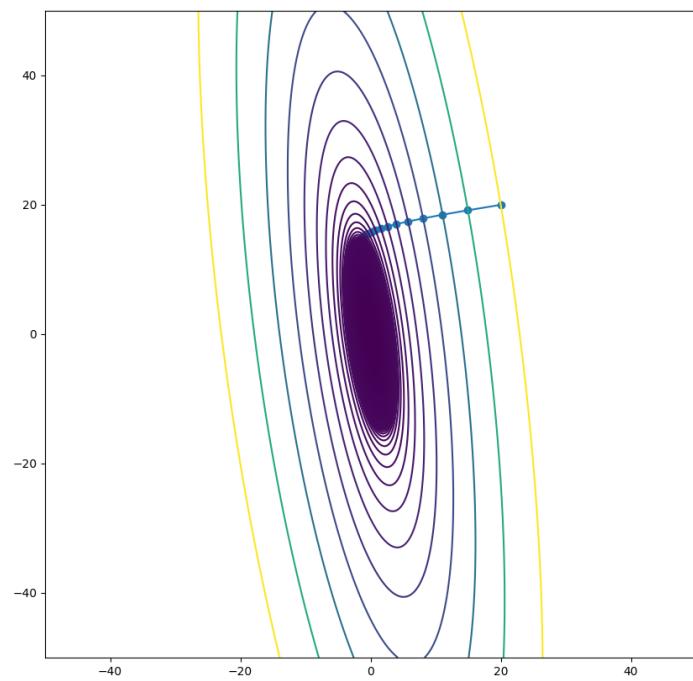


Рисунок 8

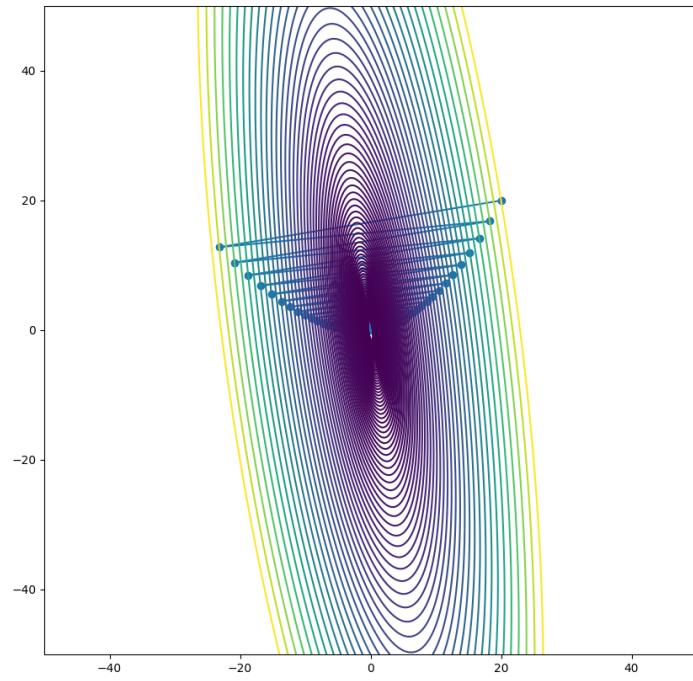


Рисунок 9

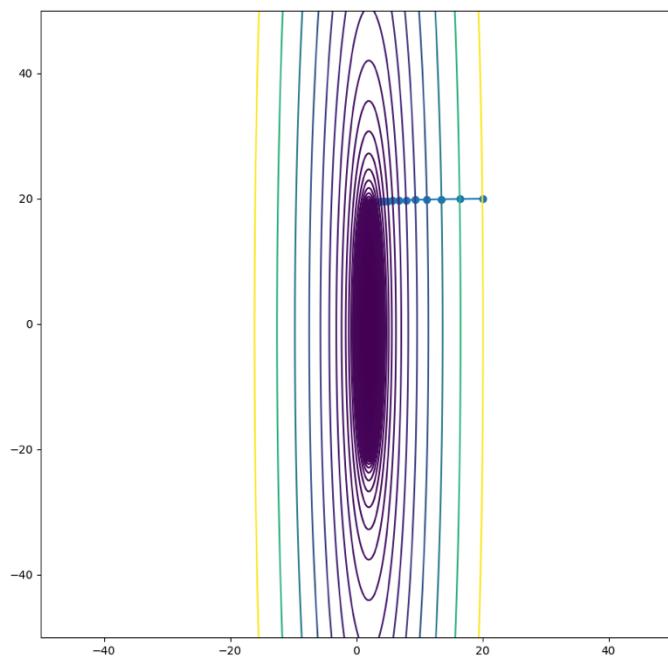


Рисунок 10

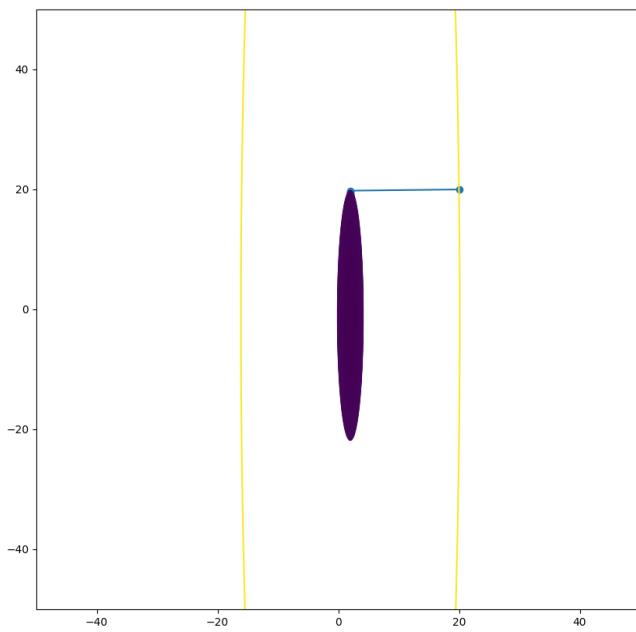


Рисунок 11

Можно сделать вывод, что для каждой конкретной функции на сходимость градиентного спуска влияет выбранный learning rate, причем lr стоит выбирать из строго заданного диапазона, выход из которого может значительно увеличить время сходимости (в случае уменьшения lr) или метод вообще перестанет сходиться (в случае увеличения lr)

Эффективность градиентного спуска с использованием одномерного поиска с точки зрения количества вычислений минимизируемой функции и ее градиентов

Для сходимости градиентного спуска с постоянным шагом и lr = 0.06 в случае первой функции требуется вычислить градиент 178 раз (число итераций для сходимости) (рис. 12)

Для сходимости градиентного спуска с использованием метода дихотомии требуется вычислить градиент 65 раза (число итераций для сходимости) и функцию 650 раз. (рис. 13)

Для сходимости градиентного спуска с постоянным шагом и lr = 0.005 в случае второй функции требуется вычислить градиент 601 раз (число итераций для сходимости) (рис. 14)

Для сходимости градиентного спуска с использованием метода дихотомии требуется вычислить градиент 601 раза (число итераций для сходимости) и функцию 6010 раз. (рис. 14)

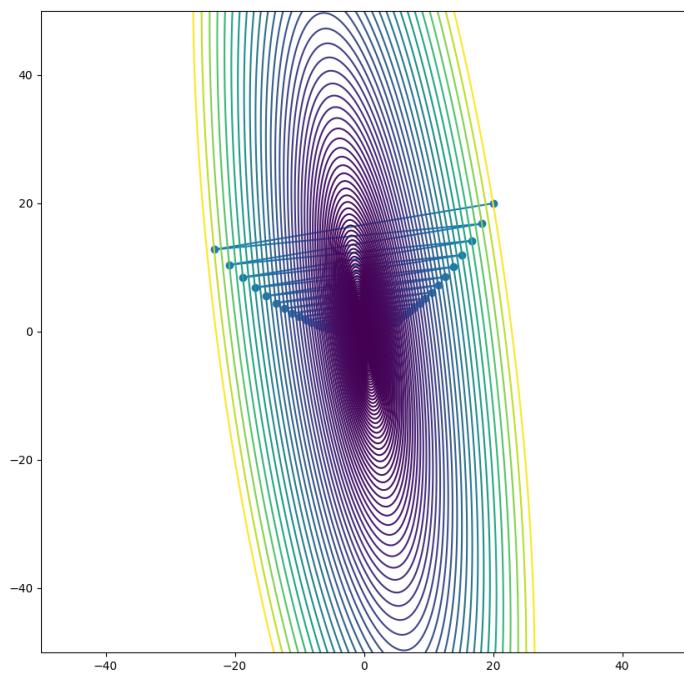


Рисунок 12

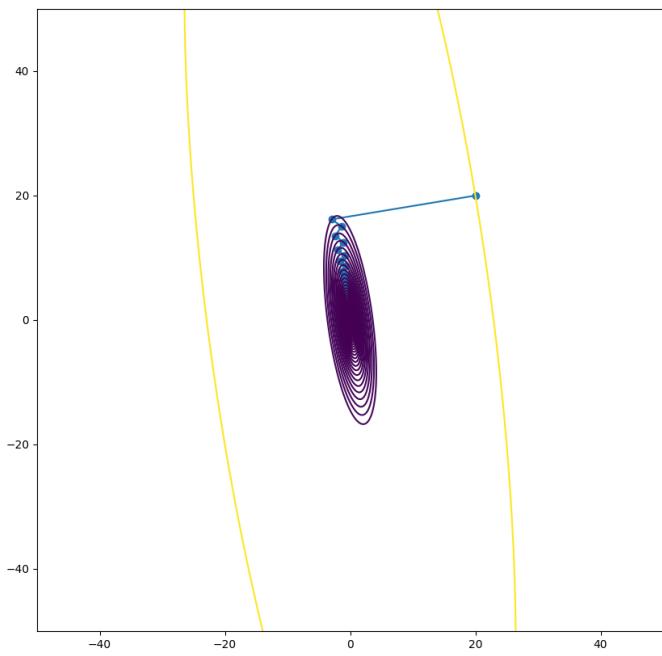


Рисунок 13

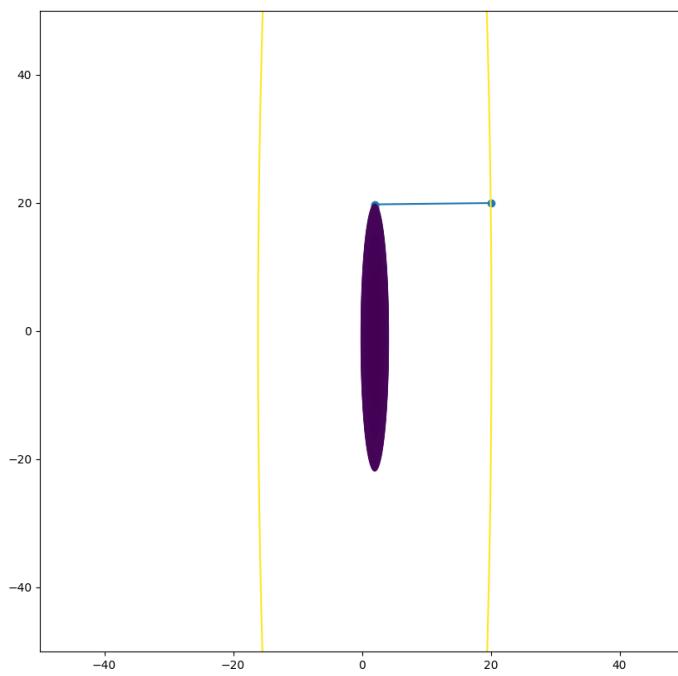


Рисунок 14

Выходит, что метод дихотомии дает выигрыш в производительности только в случае, когда вычисление функции стоит дешевле вычисления градиента, или метод дихотомии позволяет существенно сократить число итераций для сходимости.

Работа методов в зависимости от выбора начальной точки.

Составим таблицу зависимости числа итераций до сходимости метода поиска минимума первой функции в зависимости от начальной точки (табл. 1). Рисунки к соответствующим строкам таблицы соответствуют рисункам 15 - 22

Таблица 1

<i>Начальная точка</i>	<i>Сходимость гр. спуска с постоянным шагом</i>	<i>Сходимость гр. спуска с использованием метода дихотомии</i>
(68, 11)	202	43
(-54, -10)	197	46
(52, -33)	194	75
(0, 74)	162	82

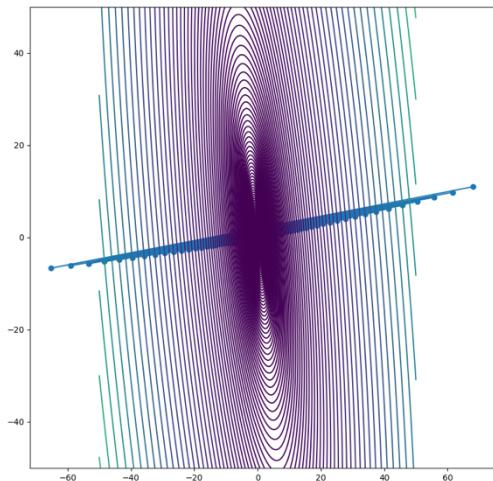


Рисунок 15

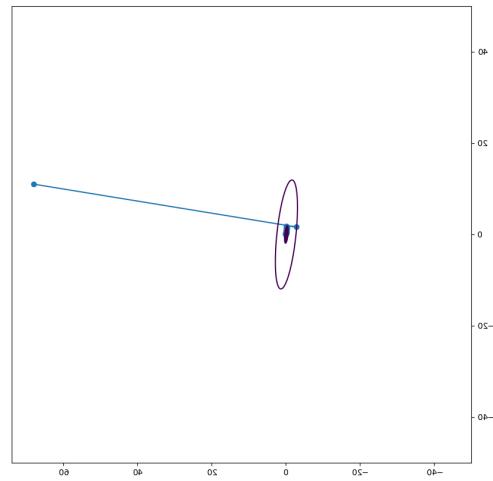


Рисунок 16

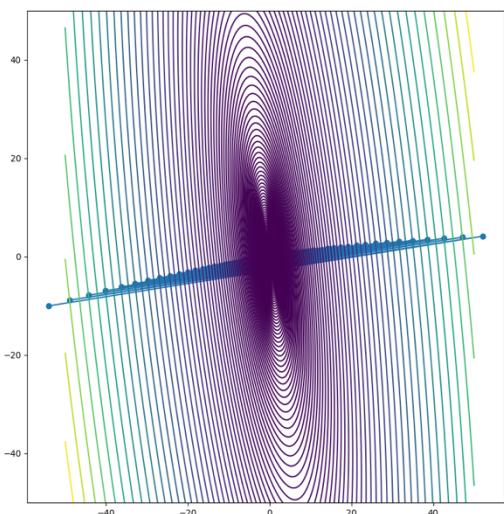


Рисунок 17

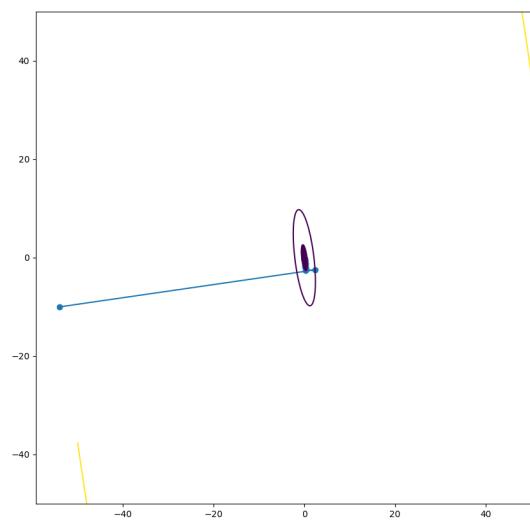


Рисунок 18

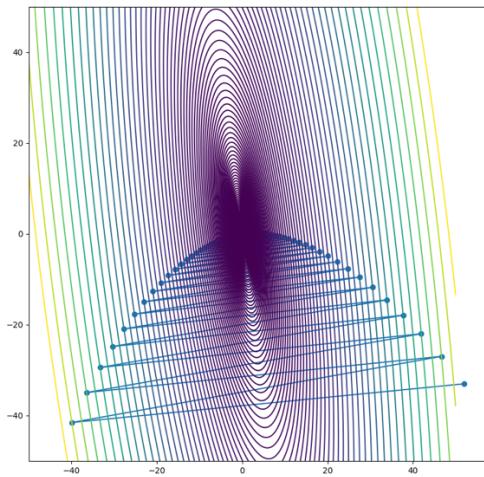


Рисунок 19

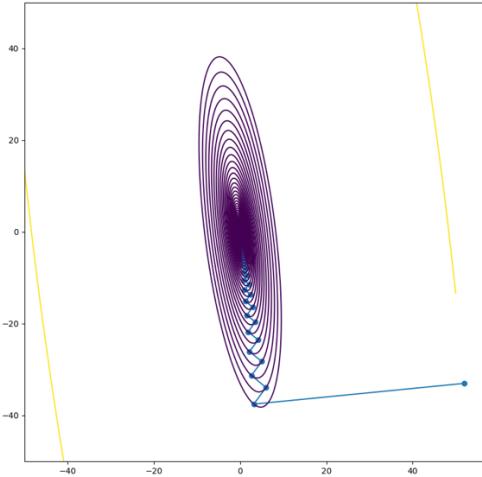


Рисунок 20

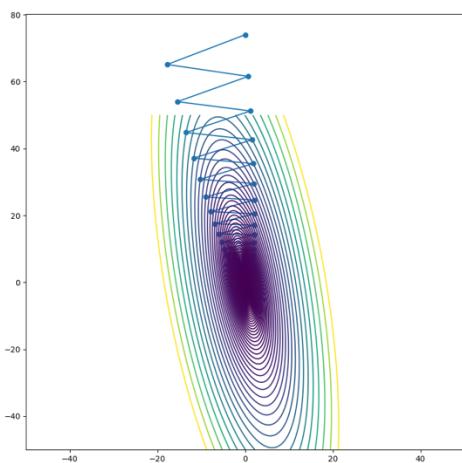


Рисунок 21

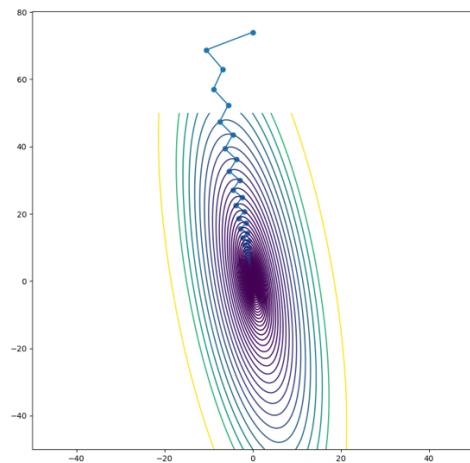


Рисунок 22

Составим таблицу зависимости числа итераций до сходимости метода поиска минимума первої функції в залежності від початкової точки (табл. 2). Рисунки к відповідаючим строкам таблиці відповідають рисункам 23 - 30

Таблица 2

<i>Начальная точка</i>	<i>Сходимость гр. спуска с постоянным шагом</i>	<i>Сходимость гр. спуска с использованием метода дихотомии</i>
(8, 43)	374	374
(-73, 38)	367	368
(-28, -2)	165	166
(54, 13)	311	312

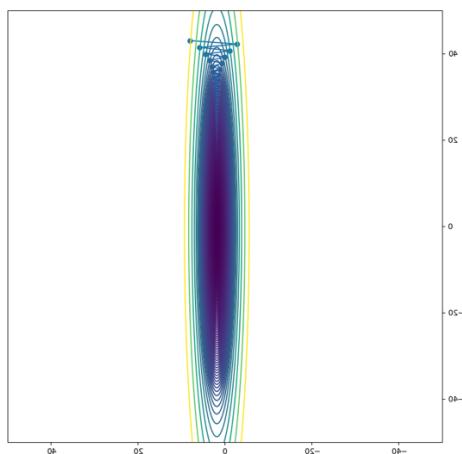


Рисунок 23

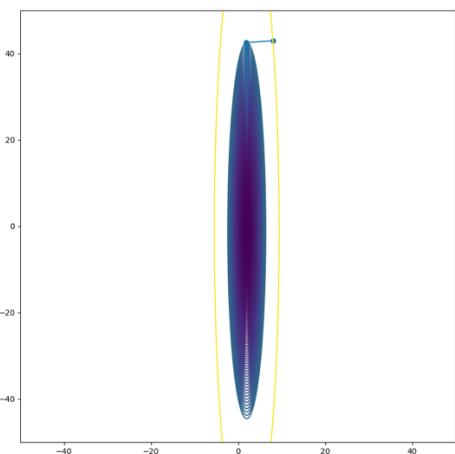


Рисунок 24

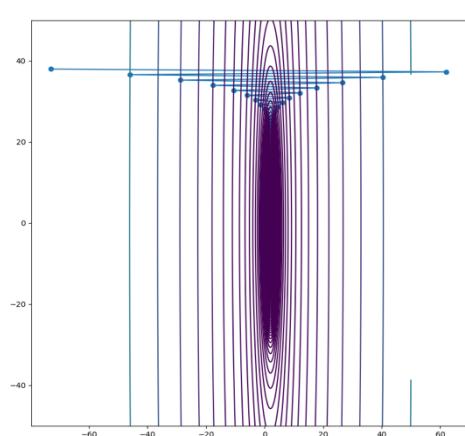


Рисунок 25

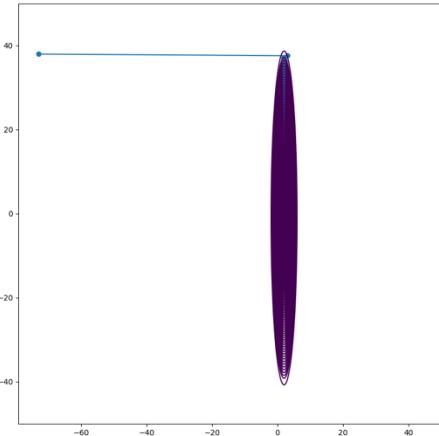


Рисунок 26

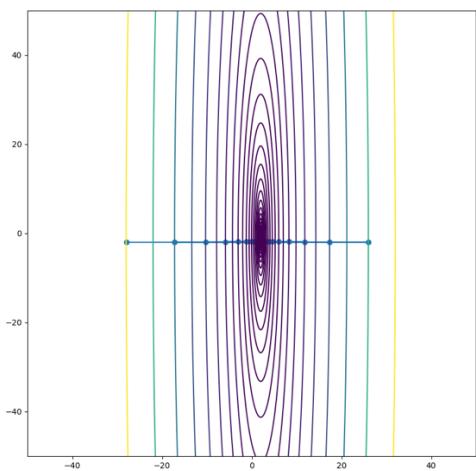


Рисунок 27

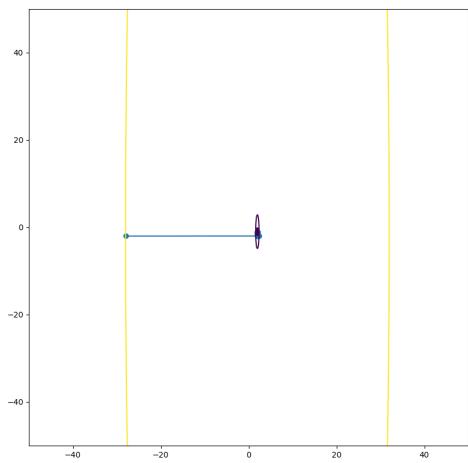


Рисунок 28

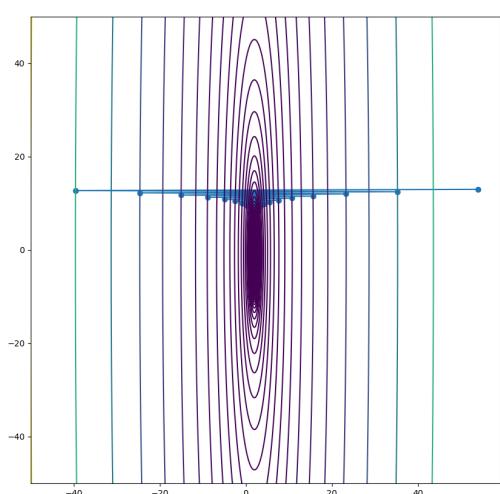


Рисунок 29

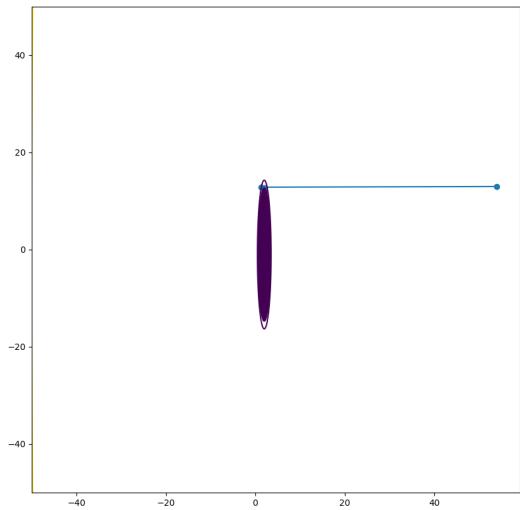


Рисунок 30

Вывод: начальное положение точки в случае квадратичных функций почти не влияет или несильно влияет на сходимость градиентного спуска с постоянным шагом и градиентного спуска с использованием метода дихотомии.

Влияние нормализации на сходимость.

Число обусловленности первой функции: 22

При lr = 0.06 метод сходится за 179 итераций (рис. 8)

При lr = 0.04 метод сходится за 92 итерации (рис. 9)

При lr = 0.1 метод расходится

Масштабируем оси функции следующим образом:

$$x' = \frac{x}{4} \quad y' = y$$

Получим функцию $f(x, y) = x^2 + y^2 + xy$

Ее число обусловленности: 3

При lr = 0.06 метод сходится за 34 итерации (рис. 31)

При lr = 0.04 метод сходится за 53 итерации (рис. 32)

При lr = 0.1 метод сходится за 19 итераций (рис. 33)

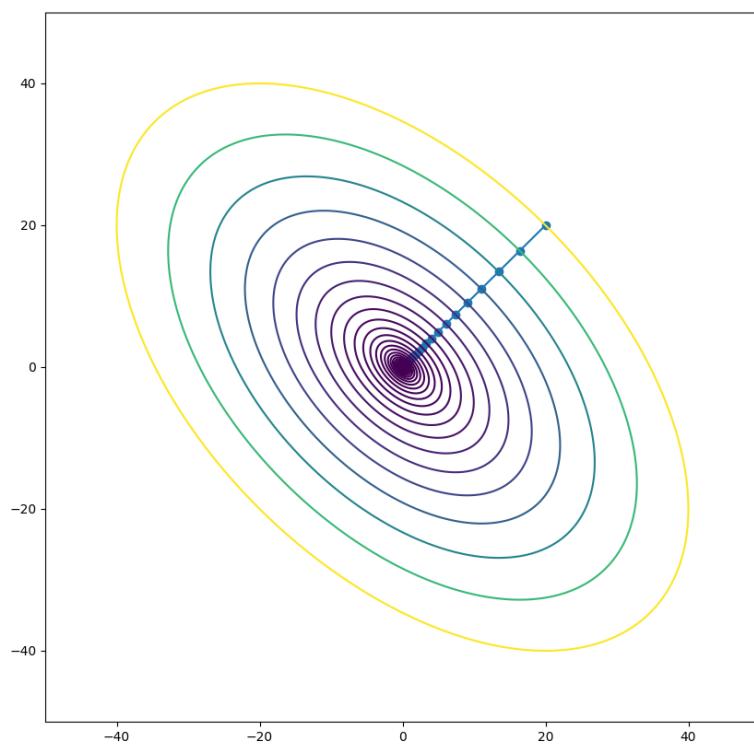


Рисунок 31

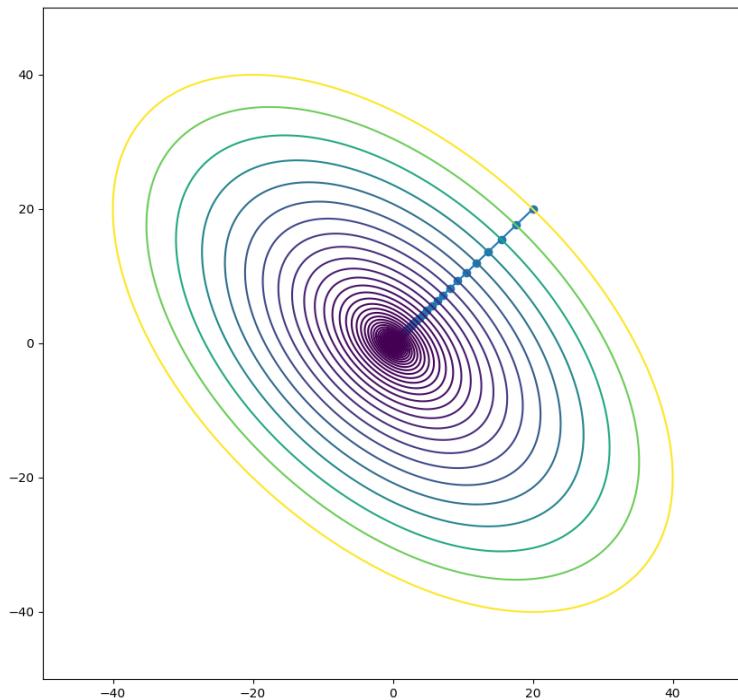


Рисунок 32

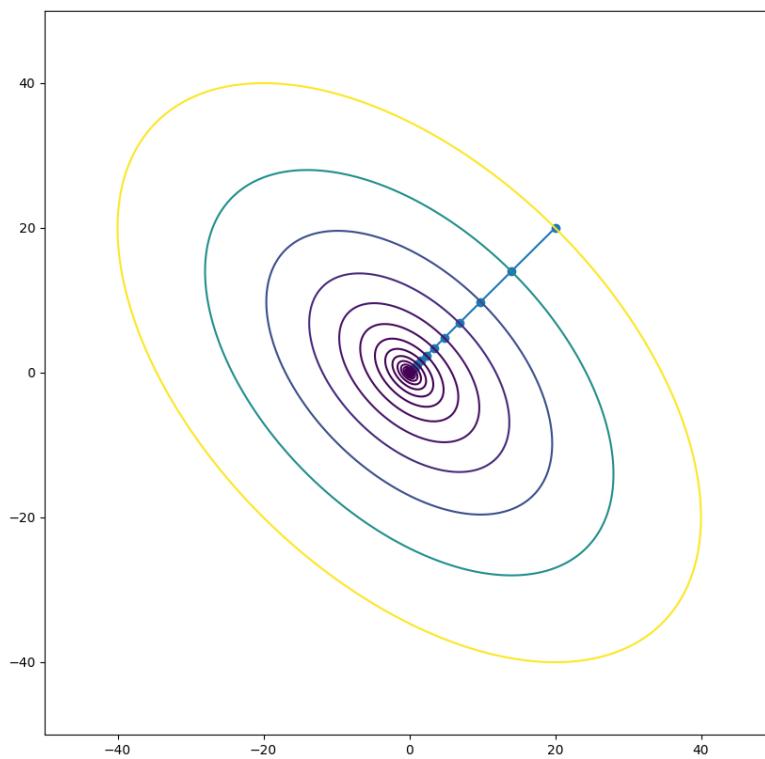


Рисунок 33

Число обусловленности второй функции: 100

При lr = 0.001 метод сходится за 1867 итераций (рис. 10)

При lr = 0.005 метод сходится за 372 итерации (рис. 11)

При lr = 0.007 метод расходится

Масштабируем оси функции следующим образом:

$$x' = \frac{x-2}{10} \quad y' = y + 1$$

Получим функцию $g(x, y) = x^2 + y^2$

Ее число обусловленности: 1

При lr = 0.005 метод сходится за 631 итерацию (рис. 34)

При lr = 0.15 метод сходится за 18 итераций (рис. 35)

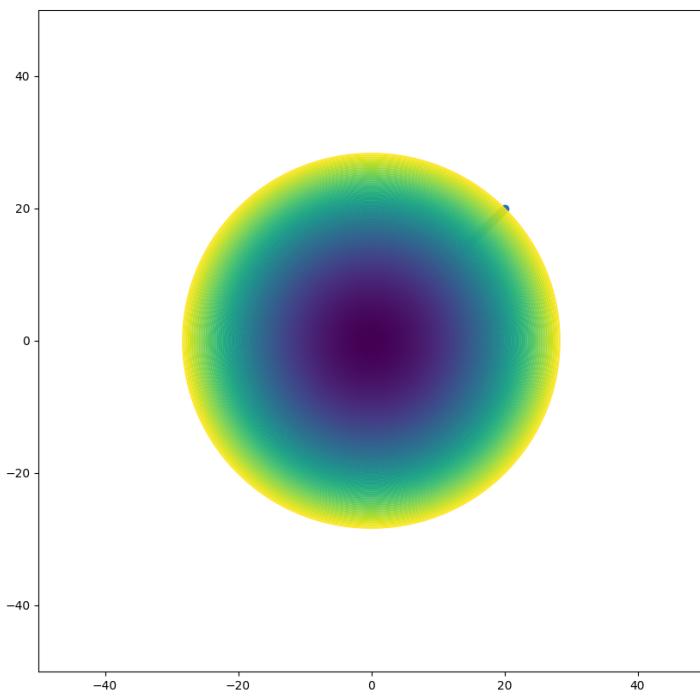


Рисунок 34

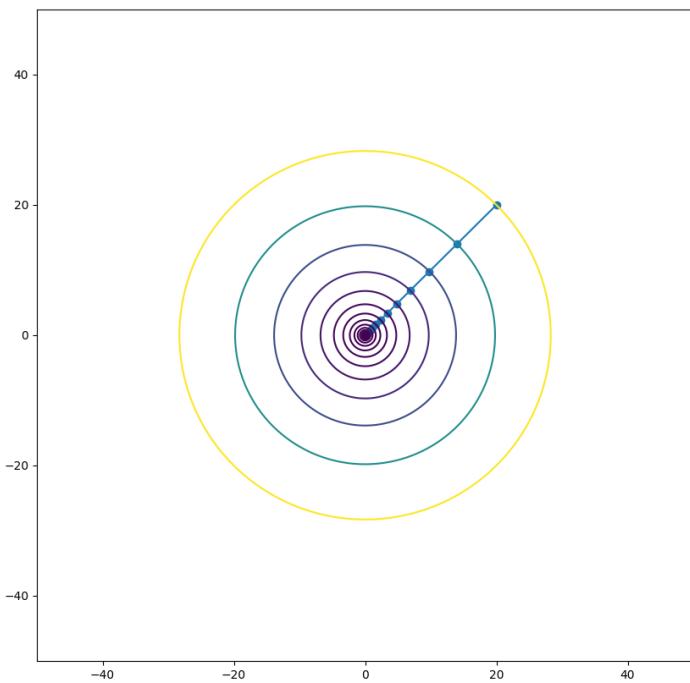


Рисунок 35

Вывод: Масштабирование осей способно существенно улучшить сходимость метода градиентного спуска в случае плохо обусловленных функций.

8. Генератор случайных квадратичных функций n переменных с числом обусловленности k.

Реализован в функции `functions.function_generator.generate_function` (листинг 4).

На вход передаются параметры n , k – количество аргументов сгенерированной функции и число обусловленности соответственно. Функция генерирует случайную ортогональную матрицу с помощью QR разложения и случайную диагональную матрицу (A) с заданным числом обусловленности. После чего приводит ее в матрицу $Q^{-1}AQ$ – матрицу квадратичной формы.

Листинг 4

```
functions.function_generator.generate_function
def generate_function(n, k):
    matrix = np.random.randint(-100, 100, (n, n))
    S = np.diag(np.linspace(k, 1, n))
    Q, R = np.linalg.qr(matrix)
    matrix = np.dot(np.dot(Q, S), np.transpose(Q))

    f = lambda x: np.dot(np.dot(np.transpose(x), matrix), x)
    h = 1e-5
    grad = lambda x: (f(x + h * np.eye(n)) - f(x - h * np.eye(n))) / (2 * h)

    return Function(f, grad)
```

9. Зависимость числа итераций, необходимых градиентному спуску для сходимости, от размерности пространства и числа обусловленности оптимизируемой функции

Зависимость приведена на рис. 36.1 Оси абсцисс соответствует n – размерность пространства. Оси ординат соответствует k – число обусловленности функции. Теплота цветовой схемы соответствует числу итераций, необходимых для сходимости.

Из графика можно понять, что сходимость градиентного спуска в случае квадратичных форм никак (или почти никак) не зависит от размерности пространства. На сходимость сильно влияет число обусловленности функции. Так происходит потому, что чем больше обусловленность функции, тем меньше приходится брать шаг (lr) (иначе – градиентный спуск вообще не будет сходиться). С достаточно маленьким шагом приходится на каждой итерации точка едва приближается к минимуму, а значит, чтобы его достигнуть потребуется

много итераций. Аналогичная ситуация возникает и с использованием метода дихотомии. (рис. 36.2)

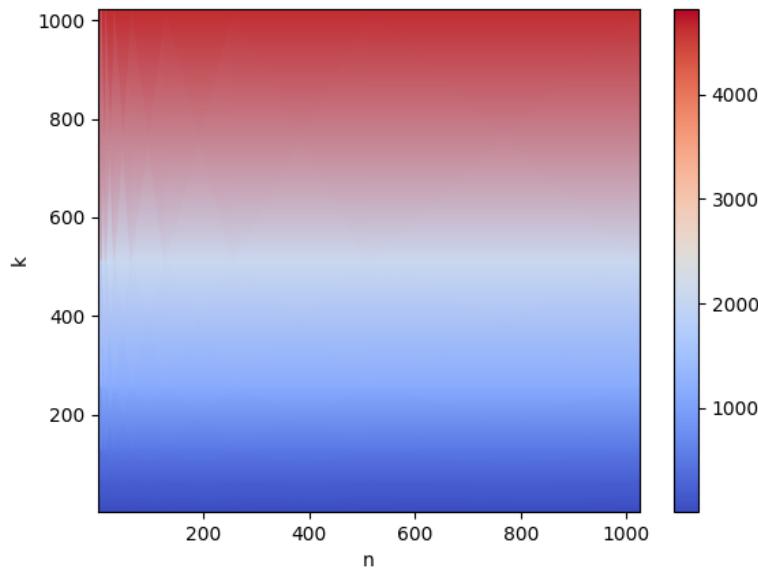


Рисунок 36.1

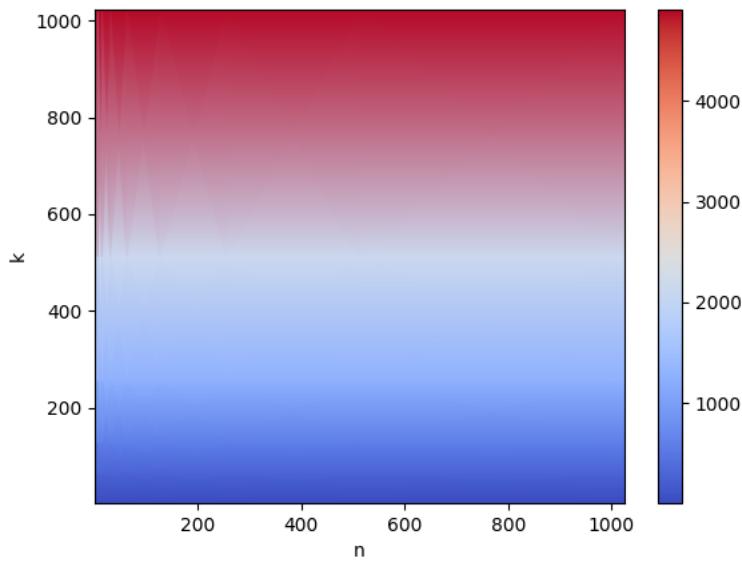


Рисунок 36.2

Поиск с учетом условий Вольфе.

Реализован в функции `optimization.optimize.wolfe` (листинг 5). Функция принимает на вход начальную точку и функцию, минимум которой надо найти.

На каждой итерации мы шагаем в направлении, противоположном градиенту. Если в полученной точке не выполняются условия Вольфе, то уменьшаем шаг и начинаем снова из текущей точки. Иначе – полученная точка – искомая.

Листинг 5

```
optimization.optimize.wolfe
def wolfe(start, func: Function, c1=10e-4, c2=0.9, tau=0.5):
    f = func.calc(start)
    grad = func.grad(start)
    p = -grad
    phi = np.dot(grad, p)
    alpha = 1

    while True:
        x = start + alpha * p
        if func.calc(x) > f + c1 * alpha * phi or np.dot(func.grad(x),
p) < c2 * phi:
            alpha *= tau
        else:
            return x
```

Сравним работу градиентного спуска с учетом условий Вольфе с градиентным спуском с постоянным шагом и градиентным спуском с использованием дихотомии.

$$1. \quad f(x, y) = 16x^2 + y^2 + 4xy, \quad lr = 0.06$$

Таблица 3

	Гр. спуск с постоянным шагом	Гр. спуск с методом дихотомии	Гр. спуск с условиями Вольфе
Число итераций до сходимости	180	67	15
Число вычислений функции	0	660	76
Число вычислений градиента	179	66	42

Траектории градиентного спуска для обозначенных методов приведены на рисунках 37 – 39.

$$2. \quad g(x, y) = 100(x - 2)^2 + (y + 1)^2, \quad lr = 0.005$$

Таблица 4

	Гр. спуск с постоянным шагом	Гр. спуск с методом дихотомии	Гр. спуск с условиями Вольфе
Число итераций до сходимости	603	603	288
Число вычислений функции	0	6020	2467
Число вычислений градиента	602	602	861

Траектории градиентного спуска для обозначенных методов приведены на рисунках 40 – 42.

$$3. \quad f(x, y) = x^2 + y^2, \quad lr = 0.15$$

Таблица 5

	Гр. спуск с постоянным шагом	Гр. спуск с методом дихотомии	Гр. спуск с условиями Вольфе
Число итераций до сходимости	19	19	2
Число вычислений функции	0	180	3
Число вычислений градиента	18	18	3

Траектории градиентного спуска для обозначенных методов приведены на рисунках 43 – 45.

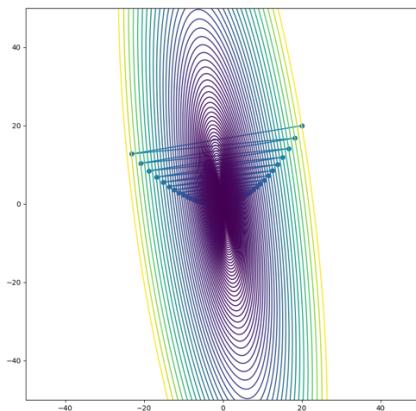


Рисунок 37

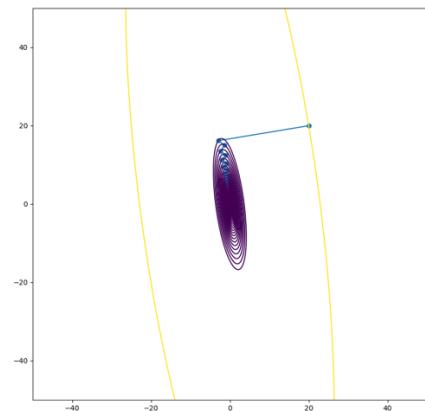


Рисунок 38

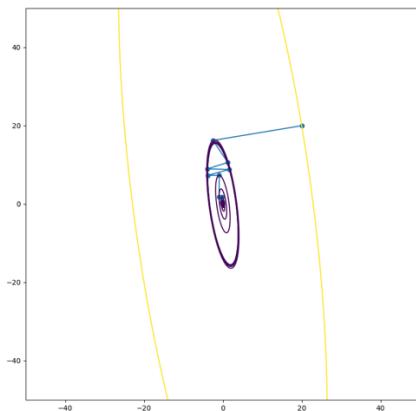


Рисунок 39

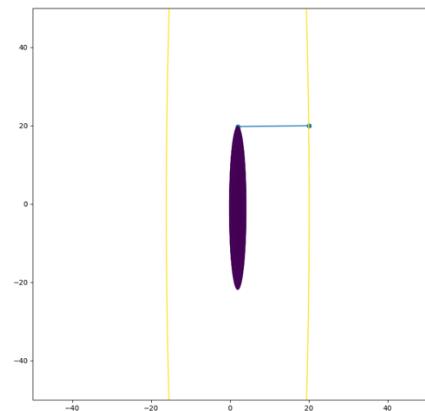


Рисунок 40

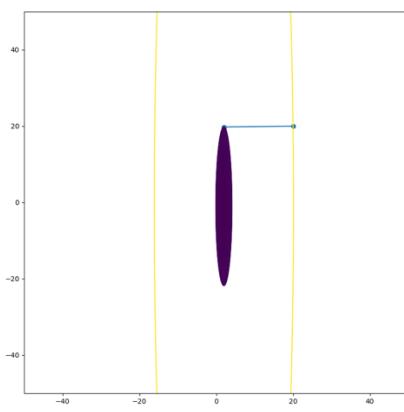


Рисунок 41

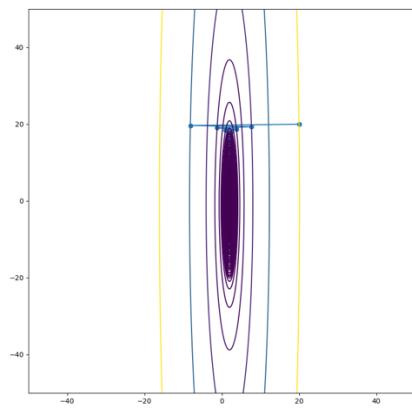


Рисунок 42

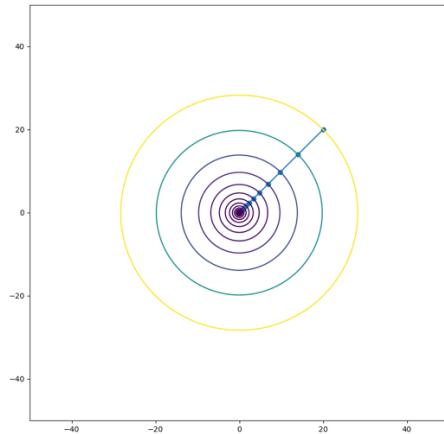


Рисунок 43

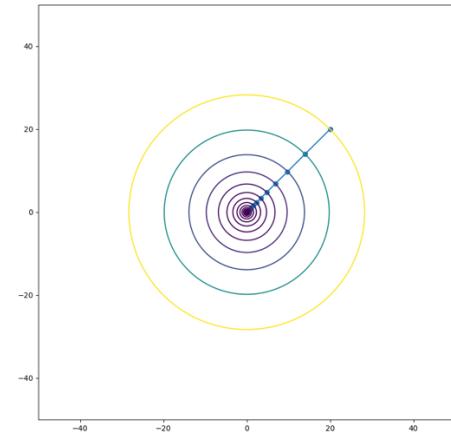


Рисунок 44

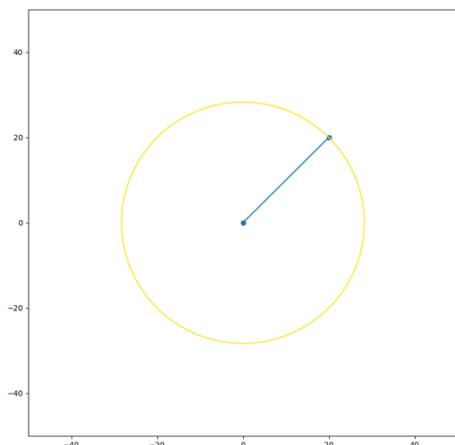


Рисунок 45

График зависимости числа итераций до сходимости от размерности пространства и числа обусловленности приведен на рисунке 46

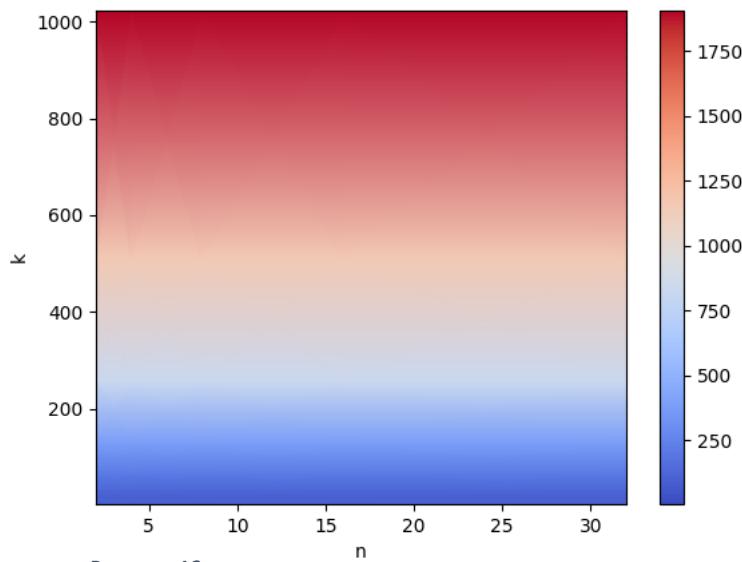


Рисунок 46

Вывод:

Условия Вольфе могут существенно сократить число итераций до сходимости в случае квадратичных функций. Однако во время работы метода требуется немалое количество раз считать как функцию, так и ее градиент, что снижает производительность.

Вывод:

В ходе данной лабораторной работы я реализовал метод градиентного спуска с постоянным шагом, метод дихотомии, линейных поиск с условиями Вольфе и градиентный спуск на их основе. Сравнил эти методы по различным

критериям. Изучил влияние масштабирования осей на сходимость функции. На основу приведенных результатов я могу сделать вывод о том, что градиентный спуск с постоянным шагом работает эффективно только в случае хорошо обусловленных квадратичных функций. Градиентный спуск с использованием метода дихотомии и градиентный спуск с учетом условий Вольфе требуют меньше итераций на сходимость, однако при их использовании требуется существенно больше раз вычислять саму функцию и ее градиент, что может сказываться на времени работы алгоритма.