Отчет

1 Авторы

Студенты группы M3439:

- Тепляков Валерий
- Плешаков Алексей
- Филипчик Андрей

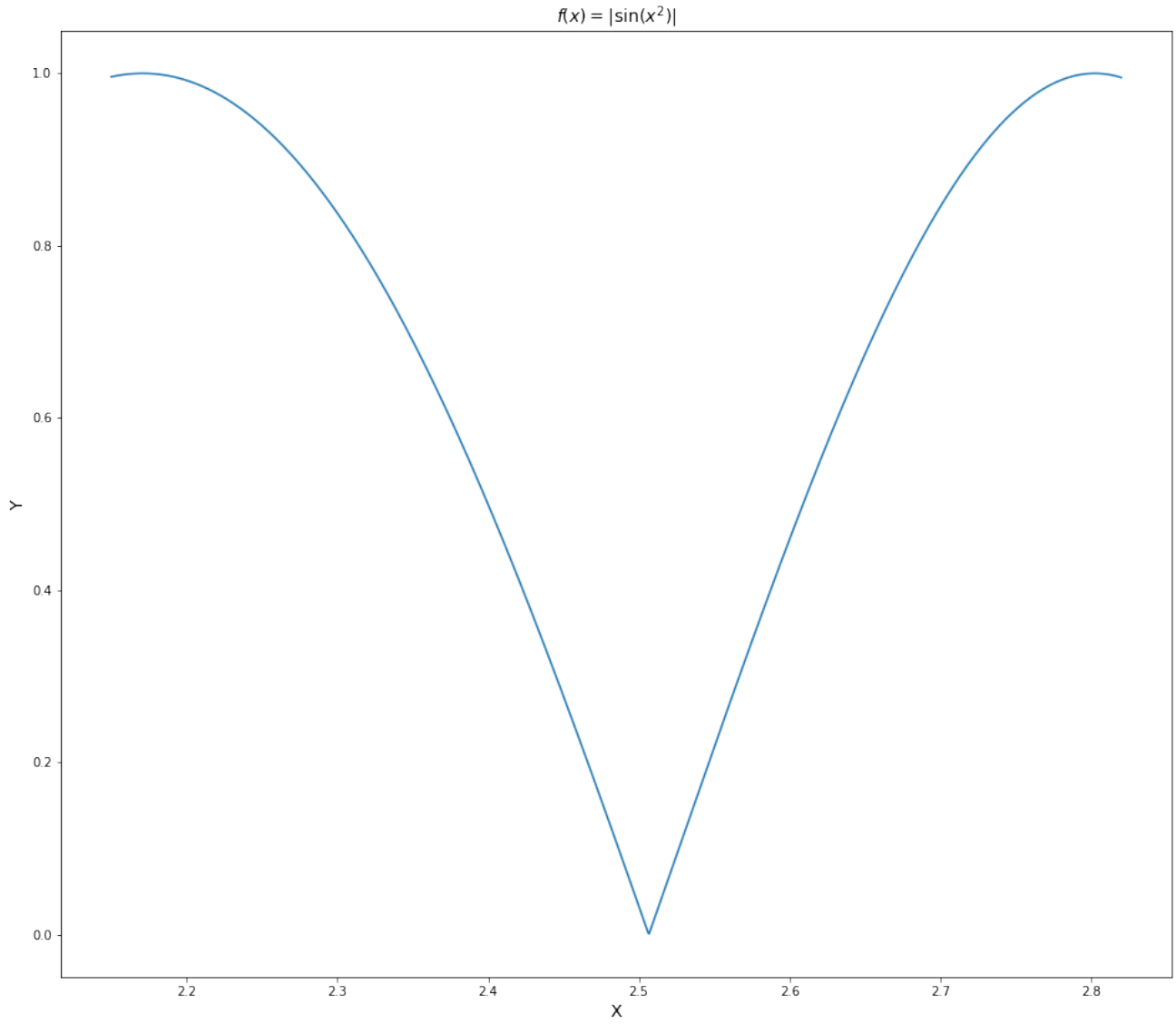
2 Source code

Исходный код можно посмотреть тут

3 Задание 1

Для сравнения методов одномерного поиска возьмем следующую унимодальную на отрезке [2.2, 2.8] функцию:

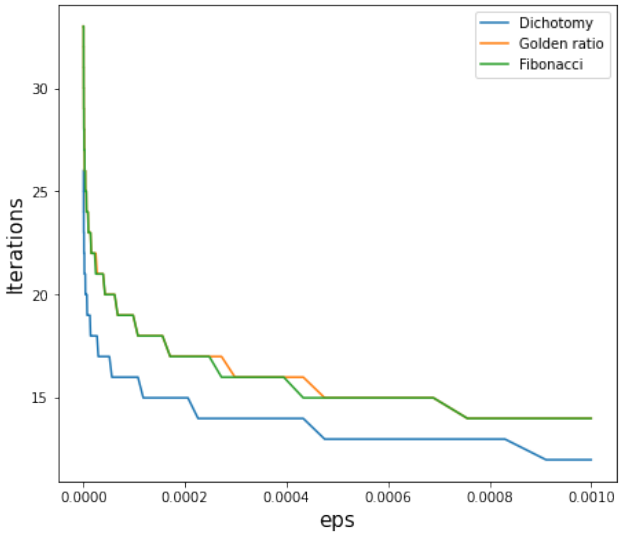
$$f(x) = |\sin(x^2)|$$

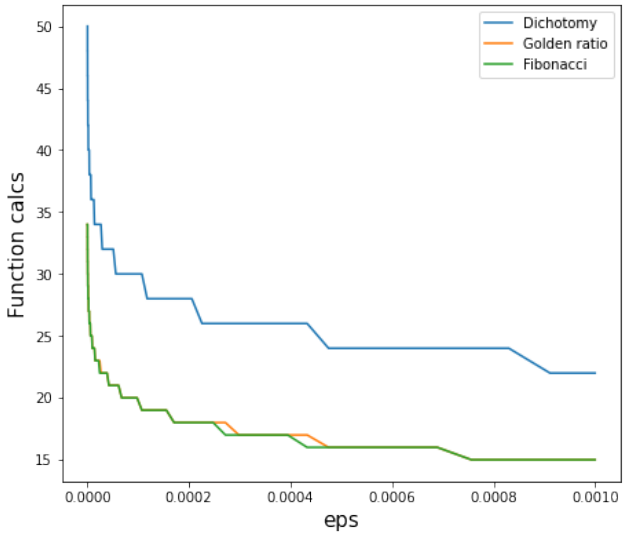


Сравним метод дихотомии, метод золотого сечения и метод Фибоначчи по количеству итераций и количеству вычислений функции в зависимости от запрашиваемой точности.

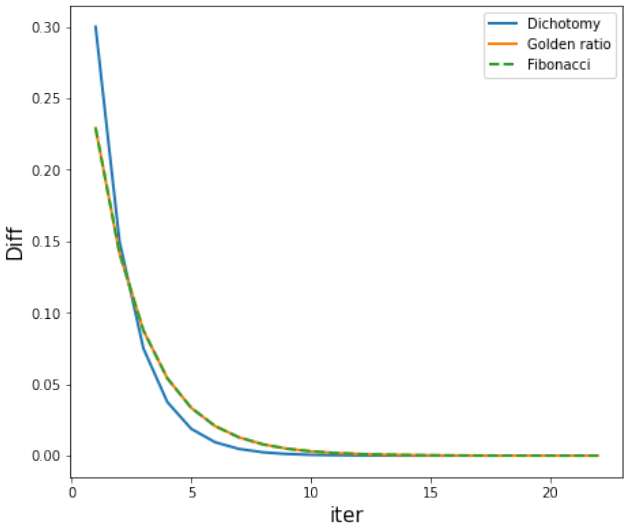
На графиках видно, что метод золотого сечения и метод Фибоначчи почти не отличаются по данным характеристикам. Метод Фибонначи на данной функции всегда показывает результат не хуже метода золотого сечения.

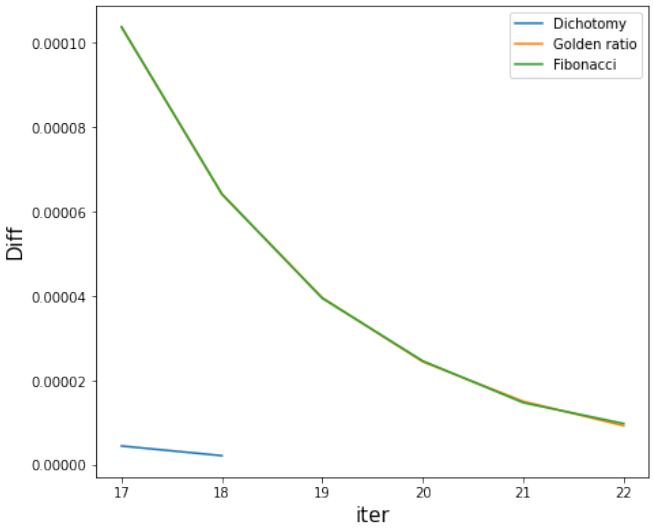
Метод дихотомии же использует меньшее количество итераций, но требует в 2 раза больше вычислений функции. Это ожидаемый результат, так как другие два метода переиспользуют уже вычисленные значения.





Также посмотрим, как изменяется отрезок при переходе к следующей итерации (правый график приближение левого). Видно, что на первых итерациях метод дихотомии сокращает интервал сильнее и быстро сходится. Другие 2 метода идут примерно наравне, лишь на большом масштабе видна разница (возможно связанная просто с погрешностями вычислений)





4 Задание 2

Используется наискорейший градиентный спуск. В качестве процедур одномерной оптимизации используются методы из задания 1 + метод средней точки (он же просто бинарный поиск) на основе производной функции

Для экспериментов использовались следующие функции:

0. $f(x, y) = 3x^2 + 7y^2 + 2yx - x$
1. $f(x, y) = (14 - x)^2 + 88(y - 4x + 7)^2$
2. $f(x, y) = xe^{-x^2-y^2}$

Метод запускался из следующих начальных точек:

0. (0.5, 0.1)
1. (0, 1)
2. (-3, 0.4)
3. (-1, 1)

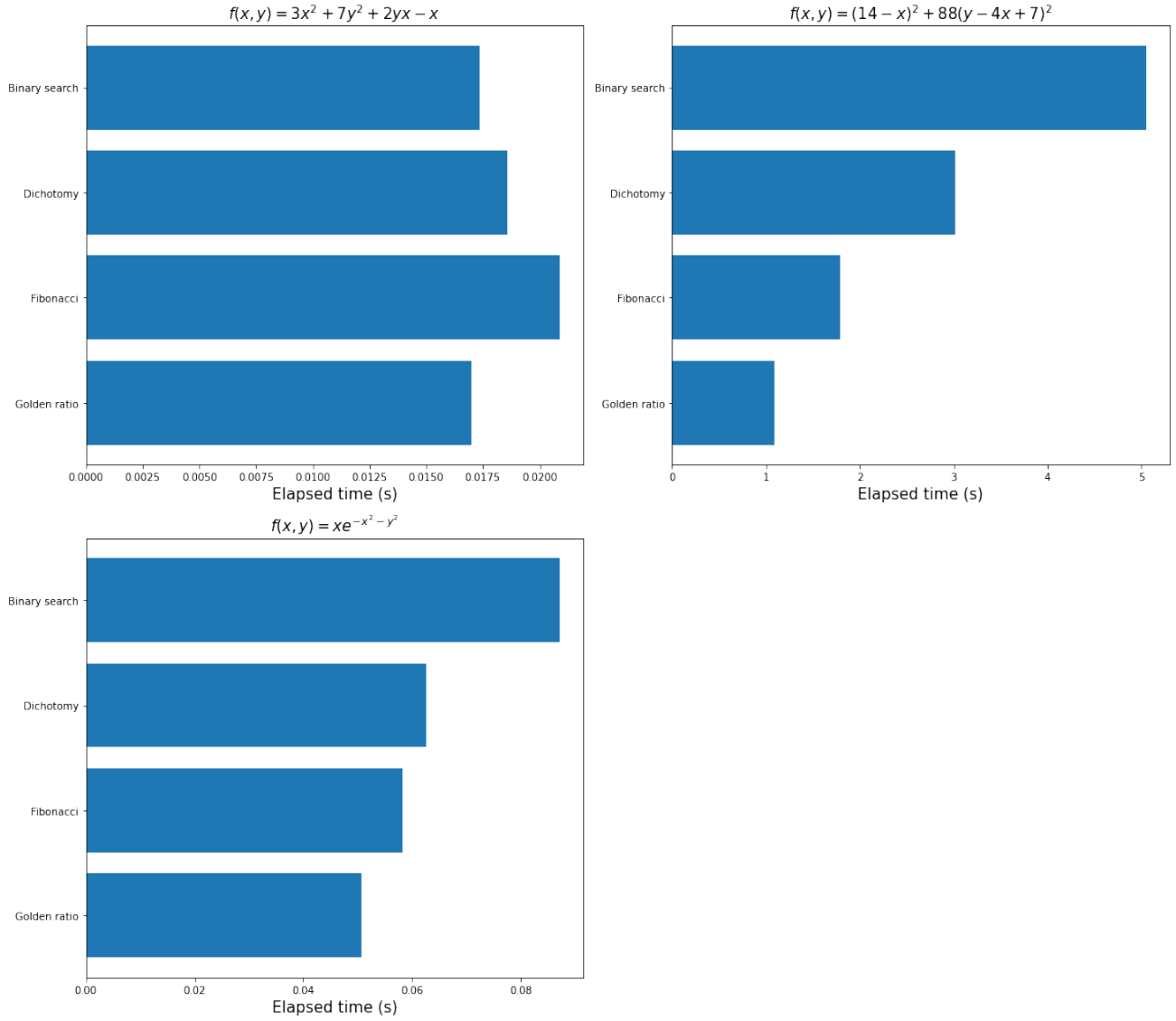
Подробные результаты запусков можно изучить в таблице:

[7]:

Function	Optimizer	Start point	Result point	Iterations	Function calls	Gradient calls	Elapsed time
0	Golden ratio	[0.5, 0.1]	(0.1686819, -0.0120458)	12	455	13	0.5s
0	Fibonacci	[0.5, 0.1]	(0.1686819, -0.0120458)	12	455	13	0.6s
0	Dichotomy	[0.5, 0.1]	(0.1686818, -0.0120457)	12	650	13	0.6s
0	Binary search	[0.5, 0.1]	(0.1686819, -0.0120458)	12	0	338	0.6s
0	Golden ratio	[0, 1]	(0.1686744, -0.0120465)	6	245	7	0.3s
0	Fibonacci	[0, 1]	(0.1686744, -0.0120465)	6	245	7	0.6s
0	Dichotomy	[0, 1]	(0.1686745, -0.0120465)	6	350	7	0.3s
0	Binary search	[0, 1]	(0.1686744, -0.0120465)	6	0	182	0.3s
0	Golden ratio	[-3, 0.4]	(0.1686736, -0.0120548)	3	140	4	0.1s
0	Fibonacci	[-3, 0.4]	(0.1686736, -0.0120548)	3	140	4	0.1s
0	Dichotomy	[-3, 0.4]	(0.1686736, -0.0120548)	3	200	4	0.1s
0	Binary search	[-3, 0.4]	(0.1686736, -0.0120548)	3	0	104	0.2s
0	Golden ratio	[-1, 1]	(0.1686655, -0.0120402)	12	455	13	0.6s
0	Fibonacci	[-1, 1]	(0.1686655, -0.0120402)	12	455	13	0.6s
0	Dichotomy	[-1, 1]	(0.1686656, -0.0120402)	12	650	13	0.7s
0	Binary search	[-1, 1]	(0.1686655, -0.0120402)	12	0	338	0.5s
1	Golden ratio	[0.5, 0.1]	(13.9988590, 48.9953925)	174	6125	175	0.60s
1	Fibonacci	[0.5, 0.1]	(13.9937287, 48.9748573)	3278	114765	3279	1.348s
1	Dichotomy	[0.5, 0.1]	(13.9932726, 48.9730319)	4558	227950	4559	1.759s
1	Binary search	[0.5, 0.1]	(13.9977413, 48.9909183)	606	0	15782	0.276s
1	Golden ratio	[0, 1]	(13.9982797, 48.9930759)	402	14105	403	0.116s
1	Fibonacci	[0, 1]	(13.9975424, 48.9901220)	692	24255	693	0.246s
1	Dichotomy	[0, 1]	(13.9957280, 48.9828597)	1944	97250	1945	0.755s
1	Binary search	[0, 1]	(13.9963823, 48.9854789)	1448	0	37674	0.587s
1	Golden ratio	[-3, 0.4]	(13.9956479, 48.9825392)	2048	71715	2049	0.631s
1	Fibonacci	[-3, 0.4]	(13.9986909, 48.9947223)	234	8225	235	0.82s
1	Dichotomy	[-3, 0.4]	(13.9982071, 48.9927832)	404	20250	405	0.161s
1	Binary search	[-3, 0.4]	(13.9916443, 48.9665149)	6948	0	180674	2.981s
1	Golden ratio	[-1, 1]	(13.9971676, 48.9886222)	900	31535	901	0.276s
1	Fibonacci	[-1, 1]	(13.9983873, 48.9935053)	340	11935	341	0.114s
1	Dichotomy	[-1, 1]	(13.9971682, 48.9886251)	940	47050	941	0.344s
1	Binary search	[-1, 1]	(13.9946088, 48.9783802)	3024	0	78650	1.200s
2	Golden ratio	[0.5, 0.1]	(-0.7071046, 0.0000059)	10	385	11	0.3s
2	Fibonacci	[0.5, 0.1]	(-0.7071046, 0.0000059)	10	385	11	0.3s
2	Dichotomy	[0.5, 0.1]	(-0.7071085, 0.0000078)	10	550	11	0.3s
2	Binary search	[0.5, 0.1]	(-0.7071046, 0.0000059)	10	0	286	0.5s
2	Golden ratio	[0, 1]	(-0.7071090, 0.0000027)	11	420	12	0.4s
2	Fibonacci	[0, 1]	(-0.7071090, 0.0000027)	11	420	12	0.5s
2	Dichotomy	[0, 1]	(-0.7071113, 0.0000025)	11	600	12	0.5s
2	Binary search	[0, 1]	(-0.7071090, 0.0000027)	11	0	312	0.6s
2	Golden ratio	[-3, 0.4]	(-0.7071040, 0.0000075)	126	4445	127	0.38s
2	Fibonacci	[-3, 0.4]	(-0.7071040, 0.0000075)	126	4445	127	0.43s
2	Dichotomy	[-3, 0.4]	(-0.7071028, 0.0000066)	126	6350	127	0.46s
2	Binary search	[-3, 0.4]	(-0.7071040, 0.0000075)	126	0	3302	0.65s
2	Golden ratio	[-1, 1]	(-0.7071035, 0.0000059)	11	420	12	0.4s
2	Fibonacci	[-1, 1]	(-0.7071034, 0.0000059)	11	420	12	0.5s
2	Dichotomy	[-1, 1]	(-0.7071101, 0.0000096)	11	600	12	0.7s
2	Binary search	[-1, 1]	(-0.7071035, 0.0000059)	11	0	312	0.9s

Сравнение скорости сходимости проводилось по затраченному процессорному времени, так как количество итераций и количество вычислений функции не очень репрезентативная информация. Нас все равно интересует, как быстро мы получим численный ответ в реальной жизни.

Агрегированные результаты можно видеть ниже:



Видно, что на 0 и 2 функции методы ведут себя сравнимо.

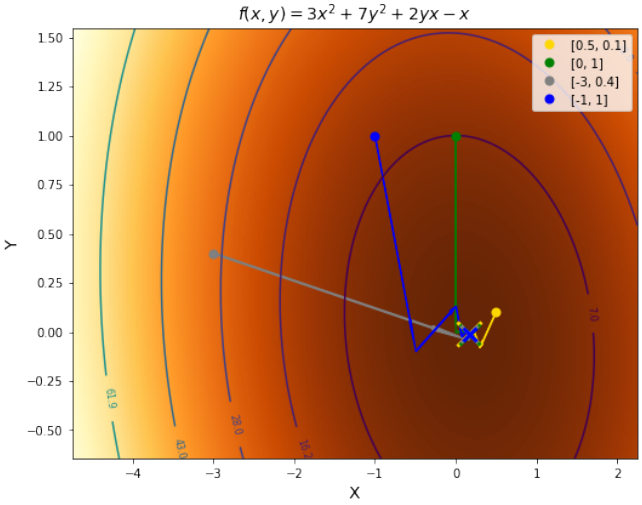
Разница во времени исполнения заметна на 1 функции. Бинарный поиск проигрывает другим методам скорее всего из-за не самой эффективной реализации вычисления градиента, который этот метод вычисляет много раз. Время выполнения остальных методов, вероятнее всего, отличается из-за того, что данная функция плохо обусловлена (будет видно далее по линиям уровня), из-за чего градиентный спуск в целом сходится плохо. Методы используют разные операции с числами, которые по-разному накапливают ошибку, и поведение градиетного спуска сильно меняется из-за накопленной ошибки

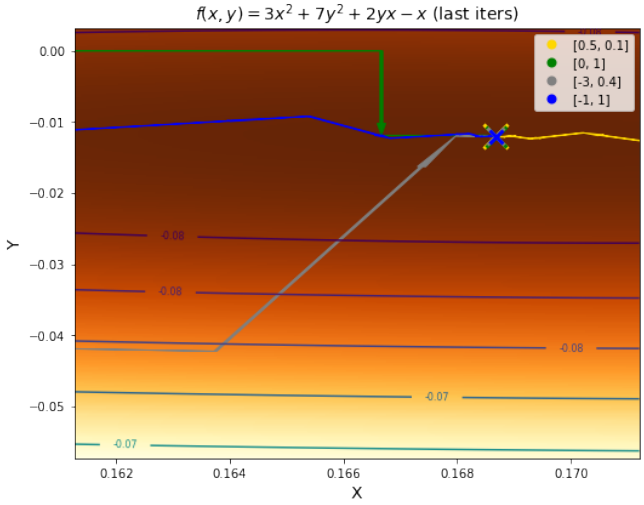
5 Задание 3

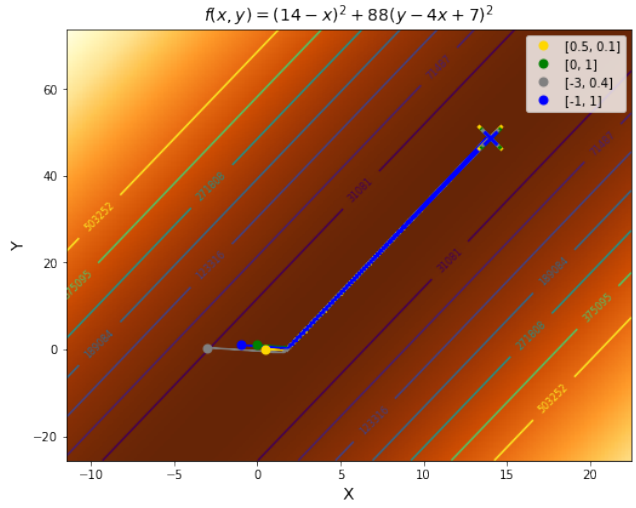
На графиках ниже можно наблюдать траектории градиентного спуска (точка на графике — стартовое значение, крестик — результат алгоритма)

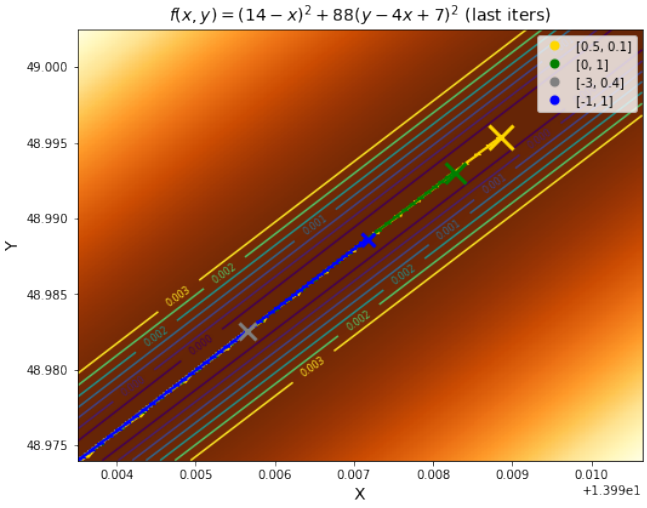
Также на графиках изображены линии уровня, и сетка раскрашена по значениям функции в точках (чем темнее — тем меньше значение)

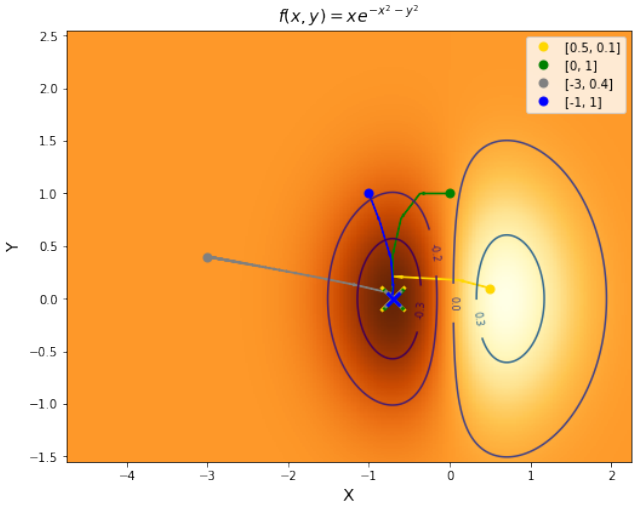
Правые графики являются приближением левых с концентрацией на последних итерациях


$f(x, y) = 3x^2 + 7y^2 + 2yx - x$
[0.5, 0.1]
[0, 1]
[-3, 0.4]
[-1, 1]
X
Y


$f(x, y) = 3x^2 + 7y^2 + 2yx - x$ (last iters)
[0.5, 0.1]
[0, 1]
[-3, 0.4]
[-1, 1]
X
Y


$f(x, y) = (14 - x)^2 + 88(y - 4x + 7)^2$
[0.5, 0.1]
[0, 1]
[-3, 0.4]
[-1, 1]
X
Y


$f(x, y) = (14 - x)^2 + 88(y - 4x + 7)^2$ (last iters)
[0.5, 0.1]
[0, 1]
[-3, 0.4]
[-1, 1]
X
Y
+1.399e1

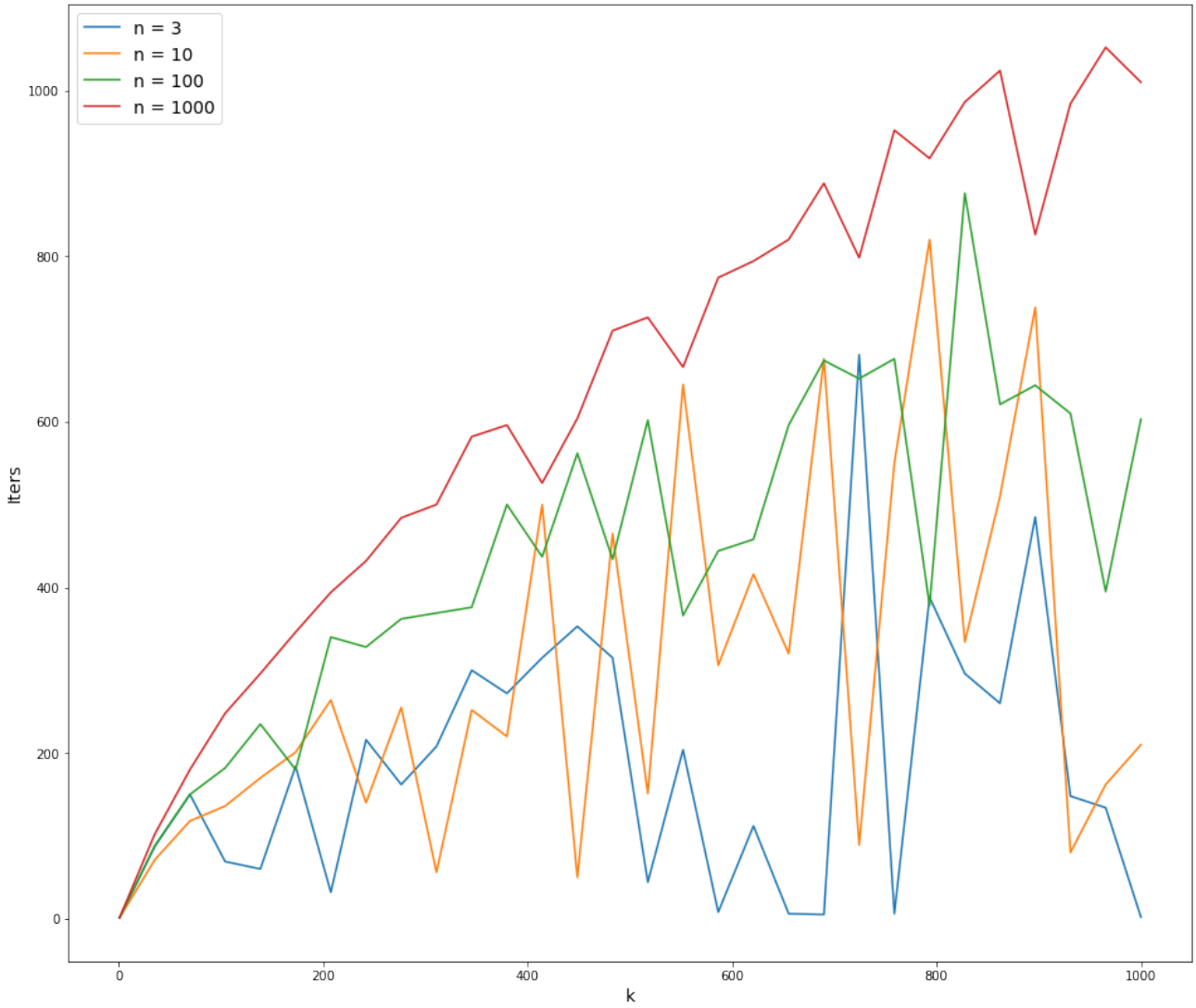

$f(x, y) = xe^{-x^2 - y^2}$
[0.5, 0.1]
[0, 1]
[-3, 0.4]
[-1, 1]
X
Y

6 Задание 4

Будем генерировать квадратичную задачу следующим образом:

1. Возьмем произвольную матрицу $Q^{'}_{[n\times n]}$
2. Произведем над ней сингулярное разложение $Q^{'} = US^{'}V^T$
3. Пусть s_{min} — минимальное сингулярное число, а s_{max} — максимальное. Отмасштабируем диагональные элементы матрицы $S^{'} -> S$: $[s_{min}, s_{max}] -> [1, \sqrt{k}]$
4. Посчитаем матрицу $Q_1 = USV^T$
5. Построим матрицу $Q = Q_1Q_1^T$
6. Сгенерируем случайный вектор b
7. Задача $f(x) = (Qx, x) + (b, x)$ является искомой и обладает числом обусловленности k, так как матрица Q является гессианом, и ее минимальное собственное число равно 1, а максимальное — k

Посчитаем количество итераций в зависимости от числа обусловленности при $n \in \{3, 10, 100, 1000\}$:



На графиках видно, что при увеличении числа обусловленности и размерности множества, число итераций градиент-

ного спуска также растет.