

# Лабораторная работа №4

## Методы стохастической оптимизации. Настройка гиперпараметров.

Задачи:

1. Описание одного из методов стохастической оптимизации и его реализация
2. Исследование метода на эффективность на нескольких функциях и сравнение с ранее реализованными методами
3. Исследование эффективности методов библиотеки `optuna` на ранее использованных примерах
4. Оптимизация выбора гиперпараметров для ранее реализованных методов.

## Часть 1. Описания используемых методов

Генетический алгоритм — эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе.

В качестве генома конкретной особи берется набор значений, на котором будет вызываться функция приспособленности – функция, значение которой показывает конкурентоспособность особи, что направляет эволюцию в сторону оптимального решения.

Этапы генетического алгоритма:



### 1. Селекция

На этапе селекции выбирается доля наиболее приспособленных особей, которые в дальнейшем будут давать потомство. При запуске алгоритма есть гиперпараметр, отвечающий за долю выживших.

### 2. Проверка критерия остановки

В качестве критерия остановки берется среднее арифметическое значение функции приспособленности, которое сравнивается с вычисленным на прошлом этапе.

### 3. Размножение

Выжившие особи размножаются, восстанавливая популяцию. Шанс размножения конкретной особи прямо пропорционален его приспособленности (у более приспособленных шанс дать потомство больше).

Способы размножения есть разные. Можно, например, в качестве генома ребенка брать среднее арифметическое генома родителей или использовать кроссинговер (часть генома первого родителя + оставшаяся часть генома второго родителя).

#### 4. Мутация

Гены у части популяции случайно мутируют, добавляя разнообразия в генофонд (доля мутирующих задается гиперпараметром). Разброс мутации может быть постоянным, а может и зависеть от номера поколения (например, у более поздних поколений мутации проявляются менее явно).

## Часть 2. Исследование эффективности

В качестве функций, на которых будет проводиться исследование эффективности генетического алгоритма, были взяты функции:

1. Из лабораторной 1:

- $x^2 + y^2 - xy$  (рис. 1)
- $(1 - x)^2 + 100(y - x^2)^2$  – функция Розенброка (рис. 2)

2. Из лабораторной 2:

- $e^{10x^2+y^2}$  (рис. 3)
- $\ln(1 + 100x^2 + y^2)$  (рис. 4)

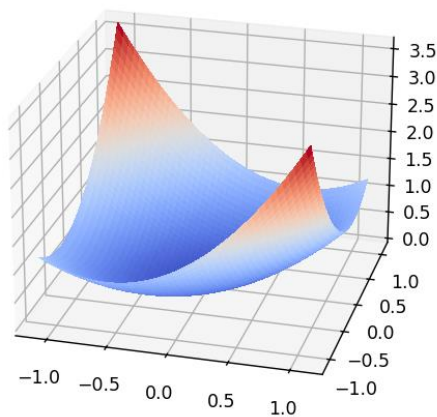


Рис. 1. График функции

$$x^2 + y^2 - xy$$

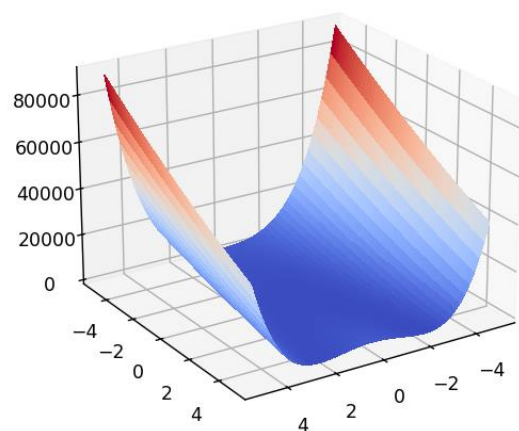


Рис. 2. График функции

$$(1 - x)^2 + 100(y - x^2)^2$$

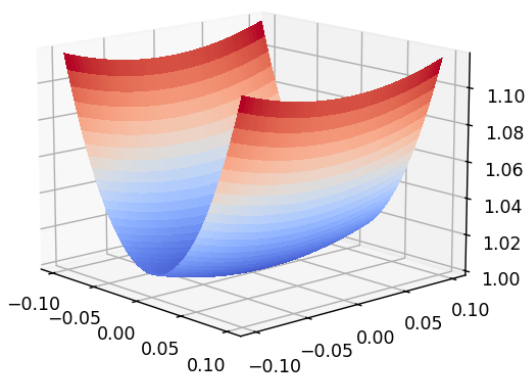


Рис. 3. График функции

$$e^{10x^2+y^2}$$

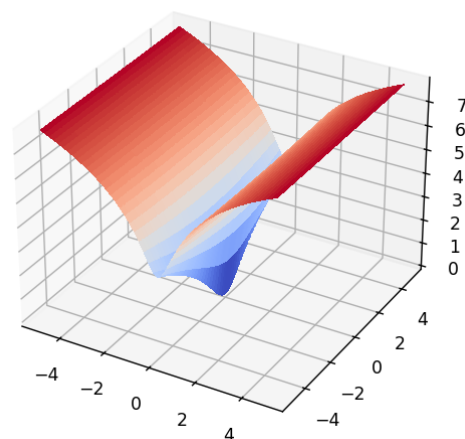
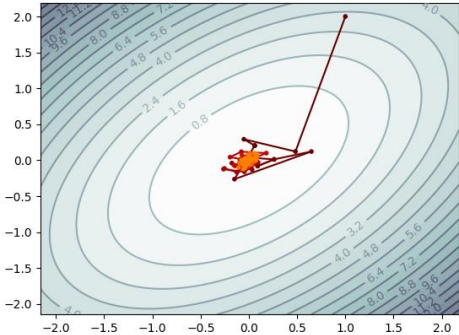
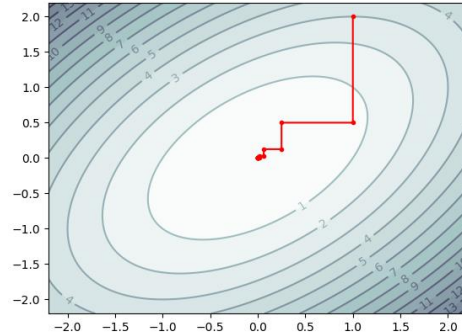
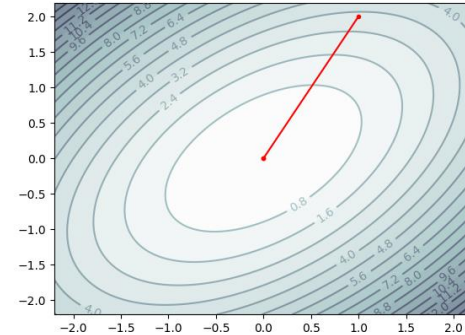
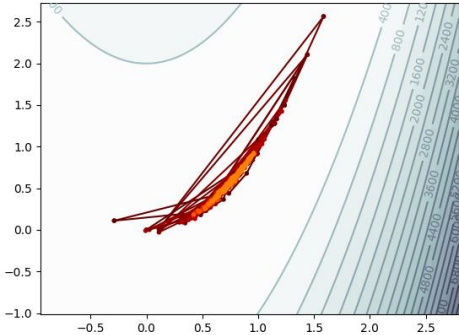
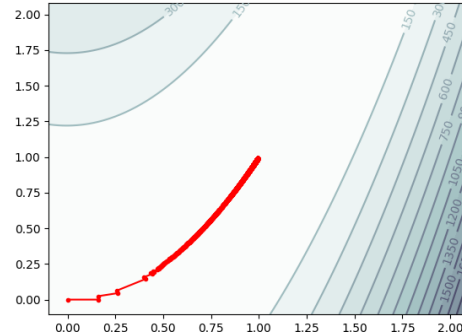
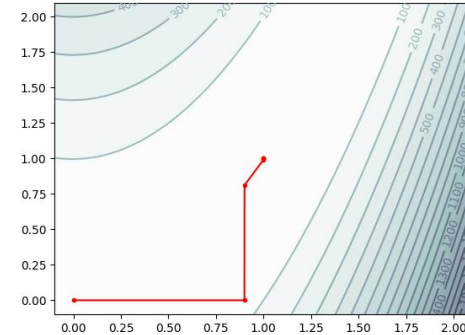


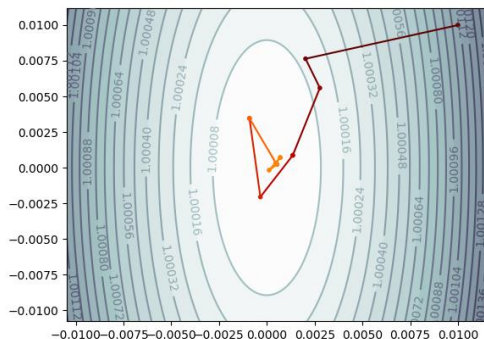
Рис. 4. График функции

$$\ln(1 + 100x^2 + y^2)$$

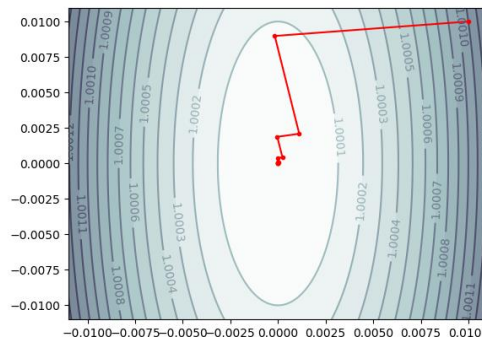
	Генетический алгоритм	Градиентный спуск (с золотым сечением)	Метод Ньютона (с условием Вольфе)
$x^2 + y^2 - xy$  (старт в (1, 2))	 <p>Итерации: 156 Выч. функ./град./гесс.: 15600</p>	 <p>Итерации: 19 Выч. функ./град./гесс.: 360</p>	 <p>Итерации: 4 Выч. функ./град./гесс.: 14</p>
$(1 - x)^2 + 100(y - x^2)^2$  (старт в (0, 0))	 <p>Итерации: 1000 (max) Выч. функ./град./гесс.: 100000</p>	 <p>Итерации: 3756 Выч. функ./град./гесс.: 31428</p>	 <p>Итерации: 12 Выч. функ./град./гесс.: 40</p>

$$e^{10x^2+y^2}$$

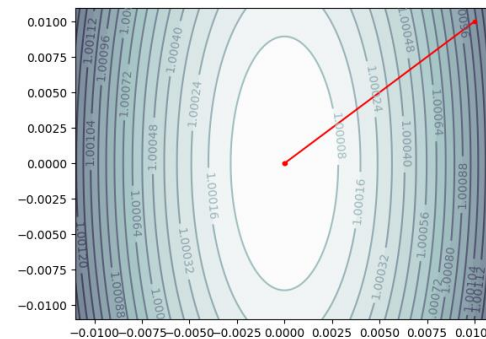
(старт в  
(1e-2, 1e-2))



Итерации: 18  
Выч. функ./град./гесс.: 1800



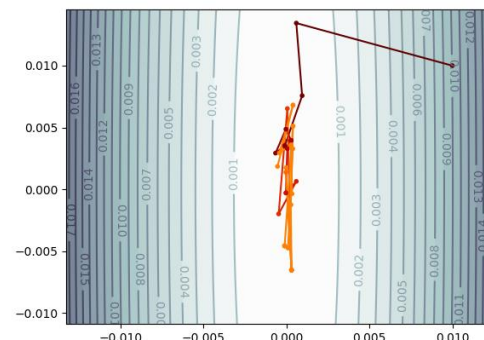
Итерации: 11  
Выч. функ./град./гесс.: 165



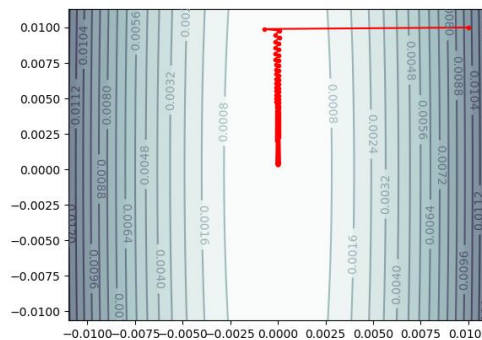
Итерации: 4  
Выч. функ./град./гесс.: 26

$$\ln(1 + 100x^2 + y^2)$$

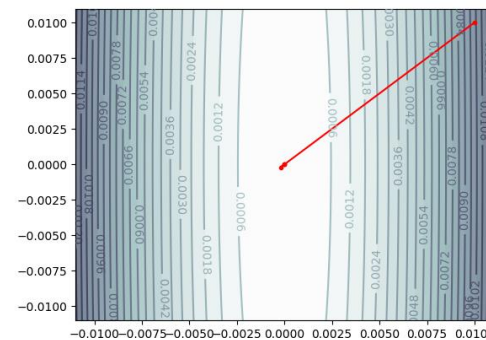
(старт в  
(1e-2, 1e-2))



Итерации: 45  
Выч. функ./град./гесс.: 4500



Итерации: 142  
Выч. функ./град./гесс.: 1378



Итерации: 4  
Выч. функ./град./гесс.: 24

**Вывод:** генетический метод работает хуже градиентного спуска и метода Ньютона, поскольку не учитывает особенности функции и основывается только на случайной выборке и её случайных модификациях, не гарантирующих быструю сходимость к оптимуму.

## Дополнительное задание 1.

Фреймворк optuna обычно используют как оптимизатор гиперпараметров, но никто не запрещает использовать ее для оптимизации любой функции.

Все обучение происходит через объект Study, который предоставляет интерфейсы для запуска нового исследования, доступа к истории испытаний, установки/получения определяемых пользователем атрибутов самого исследования.

Запуск обучения осуществляется с помощью функции optimize, которая принимает на вход целевую функцию, количество испытаний, направление оптимизации (минимизация / максимизация), sampler, pruner и др.

Sampler – алгоритм поиска оптимума. Реализовано несколько таких алгоритмов: RandomSampler (на основе случайной выборки), TPESampler (на основе алгоритма Tree-structured Parzen Estimator), CmaEsSampler (на основе CMA-ES алгоритма), GPSampler (байесовская оптимизация на основе гауссовского процесса), NSGAIISampler (недоминируемый генетический алгоритм сортировки) и др.

Зависимость работы методов от количества испытаний для различных функций (значение в таблице – норма разности найденного оптимума и реального оптимума):

Функция	Количество испытаний	RandomSampler	TPESampler	CmaEsSampler	GPSampler	NSGAIISampler
$x^2 + y^2 - xy$	5	0.7681	0.56355	0.84331	0.93239	1.13147
	10	0.49635	0.98559	0.30183	0.66138	1.16225
	20	0.57566	0.67172	0.0805	0.01852	0.38289
	30	0.5646	0.20239	0.01304	0.00181	0.15292
	40	0.54827	0.33618	0.00624	0.0011	0.13947
	50	0.31492	0.24502	0.18981	0.0006	0.27774
	60	0.59879	0.08924	0.03897	0.00049	0.30238

	70	0.35907	0.0954	0.02754	0.00037	0.29116
	80	0.30855	0.04176	0.00833	0.0014	0.20329
	90	0.14716	0.04317	0.0034	0.00151	0.05392
	100	0.25923	0.10742	0.02096	0.00079	0.2357
$(1 - x)^2 + 100(y - x^2)^2$	5	0.64545	1.16141	0.51036	0.58056	1.98118
	10	1.13655	0.24336	1.67329	0.87923	0.44025
	20	2.29972	1.16803	1.46591	0.60399	0.8863
	30	2.4315	1.1637	1.35962	0.28925	0.4067
	40	1.0576	0.4735	0.71919	0.1892	0.05812
	50	0.6832	0.81089	1.09604	0.11342	1.53341
	60	0.34864	1.204	1.29005	0.09278	1.41569
	70	0.45191	0.10583	0.23898	0.00855	0.15072
	80	0.38497	0.57403	0.13319	0.01283	0.74092
	90	0.51992	0.62663	0.60186	0.02006	1.44567
	100	0.61096	1.26863	0.8998	0.06939	0.61421
$e^{10x^2+y^2}$	5	0.40593	1.15397	0.75307	1.47379	1.52148
	10	0.90457	0.29365	0.29643	0.65302	0.31301
	20	0.52695	0.3907	0.17886	1.06326	0.98527
	30	0.0448	0.52711	0.22681	0.25284	0.29389
	40	0.6842	0.42703	0.06181	0.14182	0.41323
	50	0.33329	0.37947	0.12764	0.05271	0.31714
	60	0.10296	0.1796	0.00601	0.343	0.24023
	70	0.06711	0.20381	0.01773	0.24031	0.58175
	80	0.07239	0.01776	0.01572	0.11831	0.4401



	90	0.61757	0.10067	0.01786	0.16638	0.35326
	100	0.13201	0.04855	0.00443	0.30514	0.08677
$\ln(1 + 100x^2 + y^2)$	5	0.91428	1.06662	1.15662	1.81302	2.00026
	10	0.66147	0.4071	0.72462	1.6352	0.14565
	20	0.50861	0.18569	0.04083	0.05683	0.23596
	30	1.55672	0.25266	0.21814	0.01447	0.66276
	40	0.18113	0.52811	0.11586	0.00543	0.34949
	50	0.63663	0.15328	0.09451	0.0001	0.13492
	60	1.25609	0.02989	0.01082	0.01238	0.1524
	70	0.58589	0.61579	0.08361	0.01004	0.40136
	80	0.31319	0.07672	0.00467	0.00075	0.94084
	90	0.90614	0.0816	0.01635	0.00536	0.13659
	100	0.45053	0.12062	0.03415	0.00276	0.03939

Pruner - алгоритм для прореживания экспериментов. Pruning - это механизм который позволяет обрывать эксперименты, которые с большой долей вероятности приведут к не оптимальным результатам.

Основные алгоритмы:

- MedianPruner – останавливает половину худших, остальные продолжают
- NopPruner - никогда не останавливает испытания.
- PatientPruner - не останавливает бесперспективные испытания еще несколько эпох (затем используется другой переданный pruner).
- PercentilePruner - сохраняет определенный процентиль испытаний.
- SuccessiveHalvingPruner - алгоритм Asynchronous Successive Halving
- HyperbandPruner - алгоритм Hyperband

- ThresholdPruner - останавливает испытание, если значение целевой функции вышло за границы - превысило верхний порог или стало ниже, чем нижний порог.

Рекомендуемые в документации комбинации sampler – pruner:

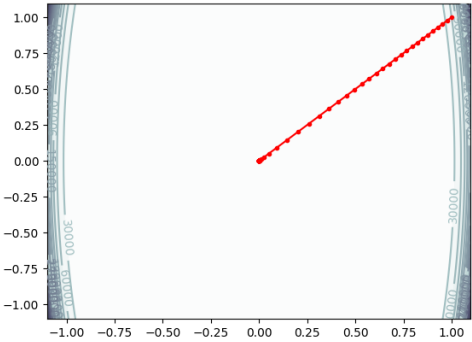
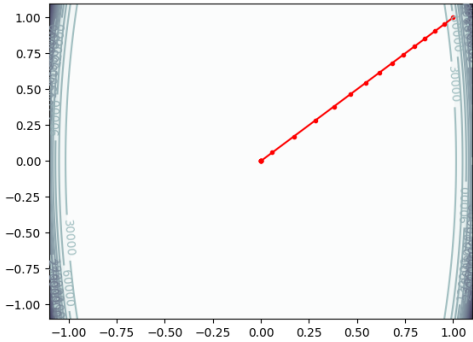
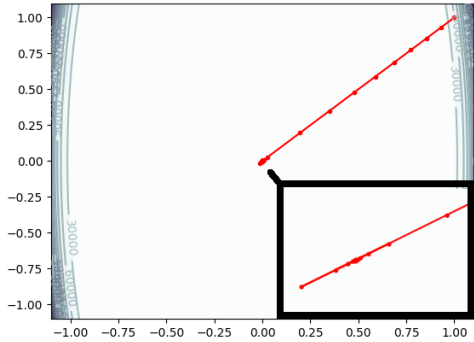
- RandomSampler – MedianPruner
- TRESampler – HyperbandPruner

Еще некоторые рекомендации из документации:

Parallel Compute Resource	Categorical / Conditional Hyperparameters	Recommended Algorithms
Limited	No	TPE. GP-EI if search space is low-dimensional and continuous.
	Yes	TPE. GP-EI if search space is low-dimensional and continuous
Sufficient	No	CMA-ES, Random Search
	Yes	Random Search or Genetic Algorithm

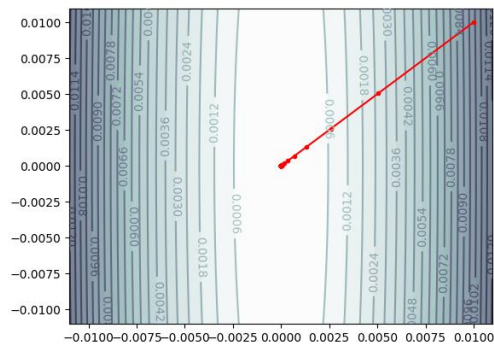
## Дополнительное задание 2.

1. Вычисление длины постоянного шага для наилучшей (наиболее эффективной) сходимости метода Ньютона

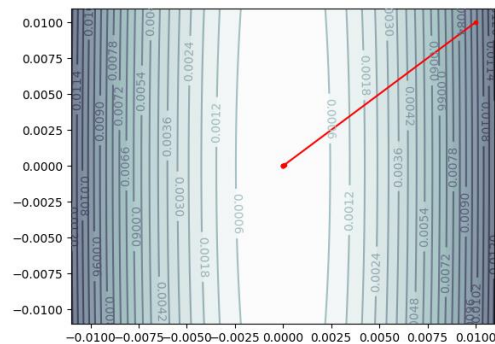
	$\frac{step}{2}$	$step$	$\frac{3 \cdot step}{2}$
$e^{10x^2+y^2}$  (старт в (1, 1))	 <p>Итерации: 100</p>	 <p>step = 1.076614591973265 Итерации: 40</p>	 <p>Итерации: 94</p>

$$\ln(1 + 100x^2 + y^2)$$

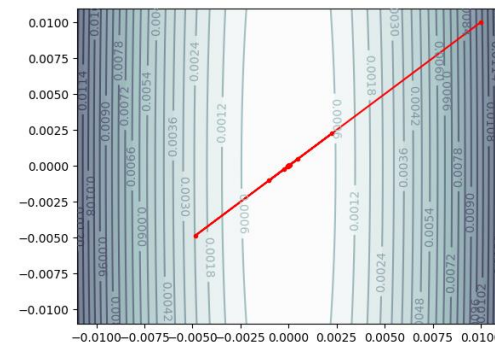
(старт в  
(1e-2, 1e-2))



Итерации: 50



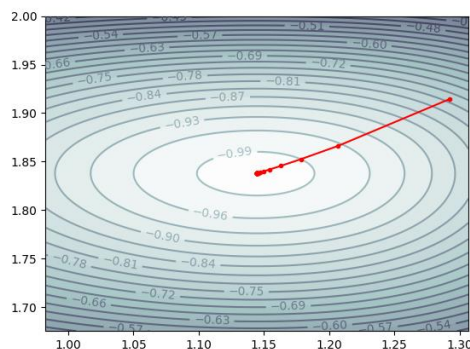
step = 0.9705589098621488  
Итерации: 12



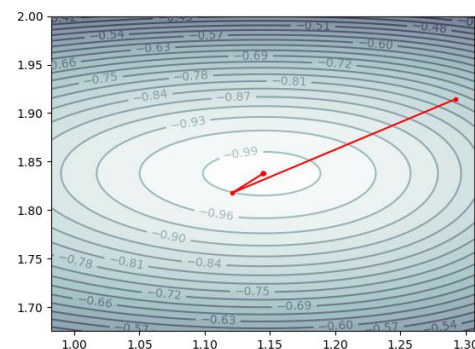
Итерации: 46

$$\cos(e^x) \cdot \cos(e^y)$$

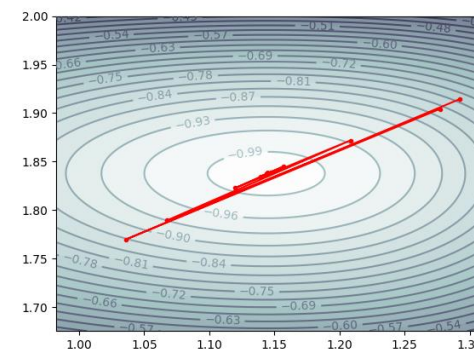
старт в  
(ln(π - 0.5),  
ln(2π + 0.5))



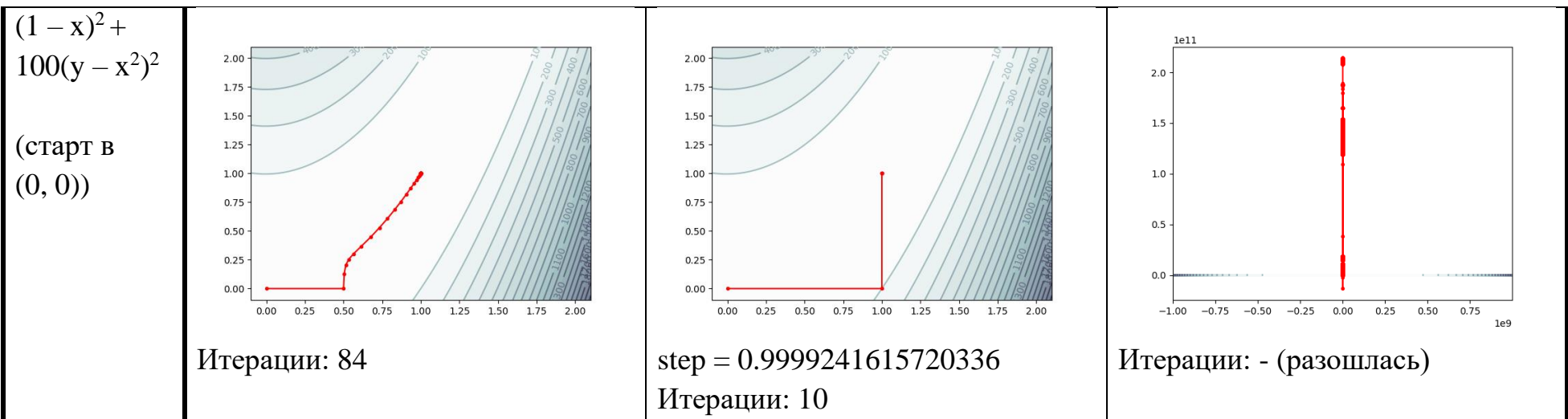
Итерации: 58



step = 0.9546245057297102  
Итерации: 14



Итерации: 56



2. Вычисление размера батча и длины шага для наиболее эффективного выполнения стохастического градиентного спуска (размер выборки = 800)

Линейная регрессия	По количеству итераций (размер батча)		По объему затраченной памяти (размер батча)		По количеству итераций (размер шага)	
	Размер батча	Итерации	Размер батча	Память, байт	Размер шага	Итерации
$3 + 6x$	543	35	12	44700	0.678	18
$-5 - 10x$	497	37	30	44864	0.799	20
$100 + 20x$	446	46	36	45048	0.558	26
$-75 - 0.5x$	471	46	41	45128	0.506	26
$10 + 0.8x$	681	41	42	44512	0.346	20

**Вывод:** стохастический градиентный спуск работает быстрее при большем размере батча, но при этом требует больше памяти (так как память минимальна при небольших размерах батча)