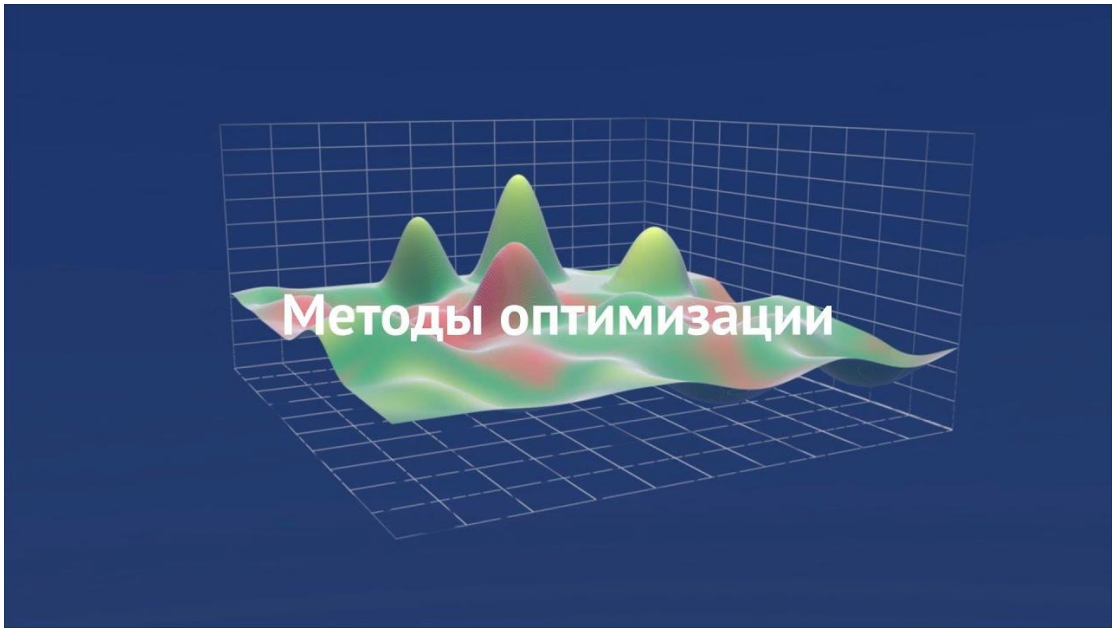


Методы Оптимизации

ЛАБОРАТОРНАЯ РАБОТА 1

Авторы: Куприянов, Долматова, Шайдулин,
группы М3233,М3232,М3233

Цели работы

- Исследовать различные численные методы для нахождения *локального* минимума
- Понять какой метод стоит использовать в зависимости от ситуаций
- Научиться реализовывать алгоритмы на языке Python 3
- Изучить возможности библиотеки scipy.optimize

РАБОТА

Мы решили что для данной работы нам стоит рассмотреть функцию Розенброка поскольку она является достаточно важной и часто используемой при исследовании алгоритмов оптимизации и имеет достаточно удобную структуру, чтобы показать плюсы и минусы разных видов. Как вторую функцию мы решили использовать какую-нибудь несложную квадратичную функцию, и взяли $(x-2)^2 + (y-3)^2$

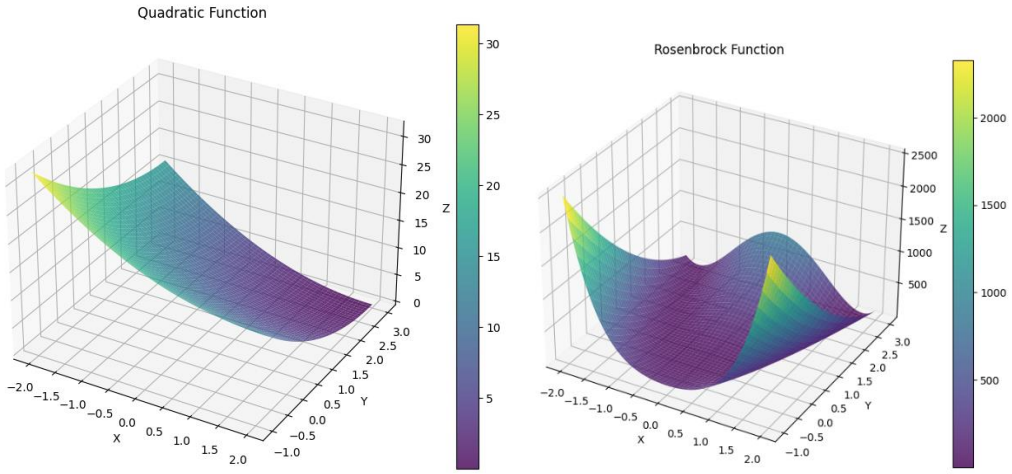
В качестве начальных точек мы выбрали (10000,1888); (-2;-2);(0.001;0.003) так как надо было посмотреть как будут вести себя функции в зависимости от координатной четверти исходной точки, но при этом не стали брать слишком большие или малые значения, чтобы не ухудшать асимптотику алгоритмов и было легче предсказать результат.

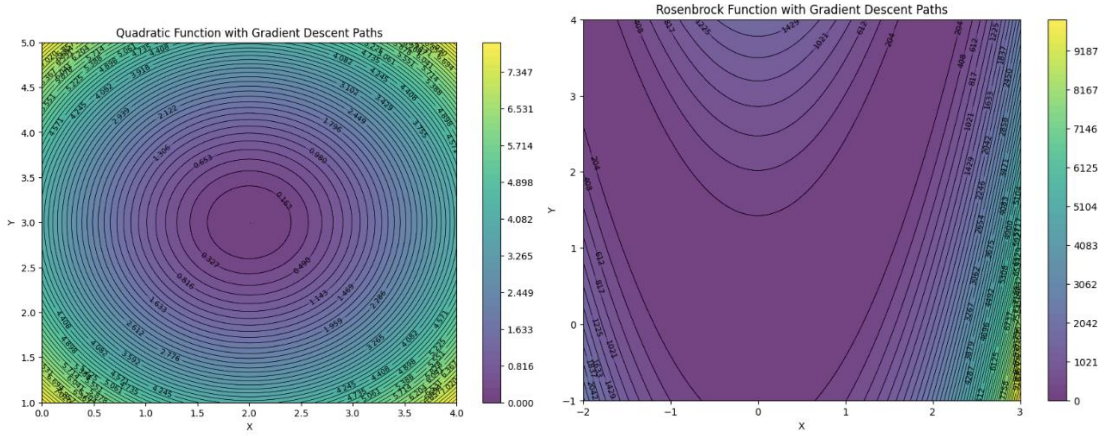
При работе мы рассматривали три точности: 1e-2, 1e-4, 1e-6.

Ссылка на реализацию всех методов

МЕТОД ГРАДИЕНТНОГО СПУСКА

И так, начнем с метода градиентного спуска:





Результаты работы программы:

Table 1: Optimization Results

Function	Initial Point	Tolerance	Minimum	Function Value at Minimum	Iterations
Rosenbrock	(10000, 1888)	0.010000	(10000.0000, 1888.0000)	999962240456434432.000000	2
Quadratic	(10000, 1888)	0.000100	(9990.0000, 1888.0005)	995968311934480512.000000	1001
Rosenbrock	(10000, 1888)	0.000001	(9990.0000, 1888.0005)	995968311934480512.000000	1001
Quadratic	(10000, 1888)	0.010000	(-1.9903, -1.9976)	3559.746798	3
Rosenbrock	(10000, 1888)	0.000100	(0.8500, 0.7318)	0.031138	1001
Quadratic	(10000, 1888)	0.000001	(0.8500, 0.7318)	0.031138	1001
Rosenbrock	(-2, -2)	0.010000	(0.0505, 0.0017)	0.901564	7
Quadratic	(-2, -2)	0.000100	(0.8924, 0.8063)	0.021291	1001
Rosenbrock	(-2, -2)	0.000001	(0.8924, 0.8063)	0.021291	1001
Quadratic	(-2, -2)	0.010000	(10000.0000, 1888.0000)	103513229.000000	2
Rosenbrock	(-2, -2)	0.000100	(9990.1731, 1886.1473)	103309846.097530	1001
Quadratic	(-2, -2)	0.000001	(9990.1731, 1886.1473)	103309846.097530	1001
Rosenbrock	(0.001, 0.003)	0.010000	(-1.2004, -1.0005)	26.246402	130
Quadratic	(0.001, 0.003)	0.000100	(1.9969, 2.9962)	0.000024	821
Rosenbrock	(0.001, 0.003)	0.000001	(1.9999, 2.9999)	0.000000	1001
Quadratic	(0.001, 0.003)	0.010000	(0.0176, 0.0280)	12.762760	5
Rosenbrock	(0.001, 0.003)	0.000100	(1.9973, 2.9959)	0.000024	541
Quadratic	(0.001, 0.003)	0.000001	(2.0000,	0.000000	769

Как мы можем видеть, что метод хорошо и быстро работает при небольшой точности, но затем возникают проблемы:

Таким образом, из плюсов можем отметить:

Плюсы метода градиентного спуска:

1. Эффективность на больших наборах данных: Градиентный спуск может хорошо работать на больших объемах данных, ведь не надо храненить весь набор данных в памяти.

2. Возможность нахождения локальных минимумов: Градиентный спуск может быть решить задачу нахождения локальных минимумов функций

Минусы метода градиентного спуска:

1. Зависимость от начальной точки: Результаты градиентного спуска могут зависеть от начальной точки, с которой начинается оптимизация, что видно из графика.

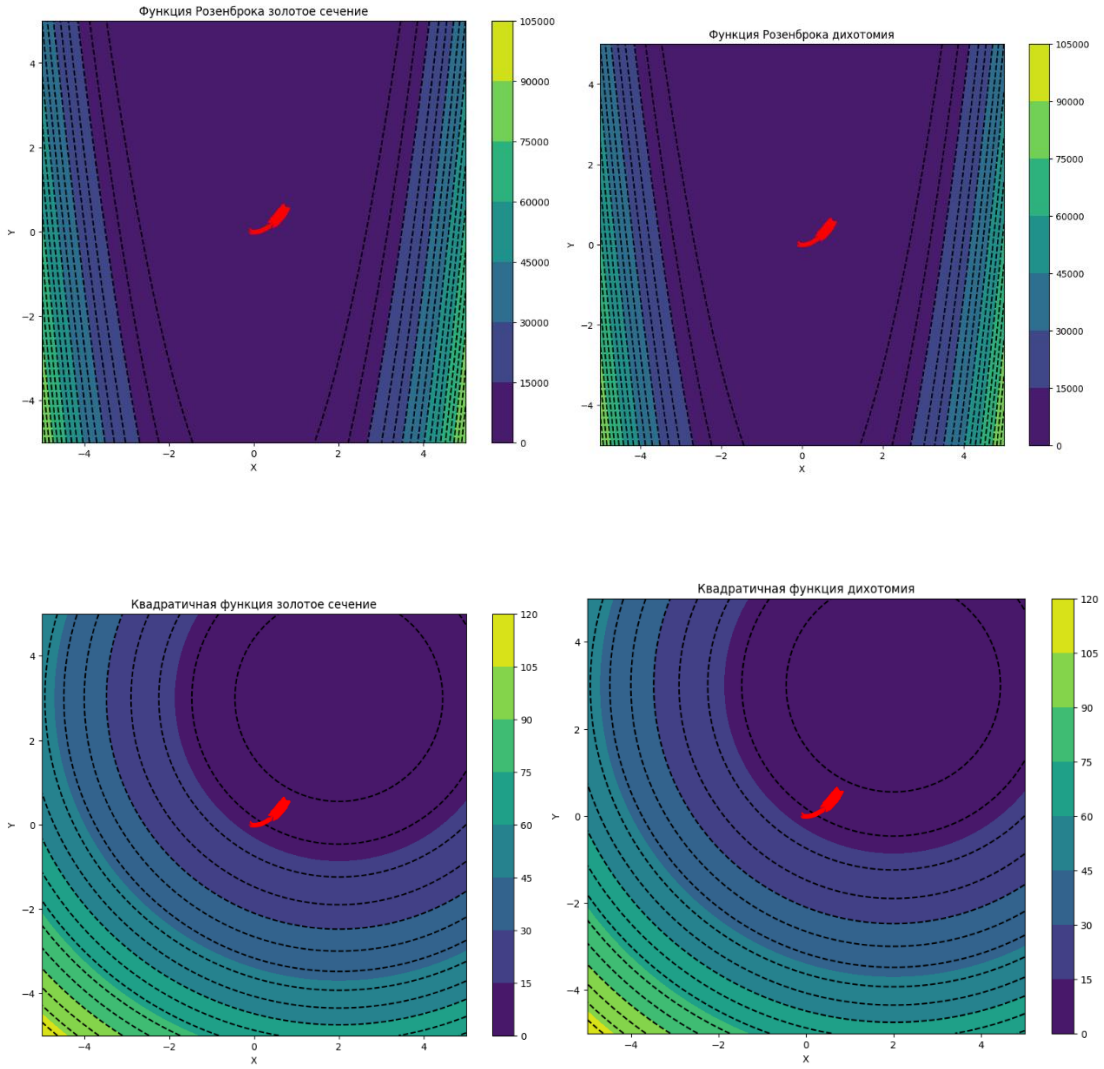
2. Чувствительность к выбору скорости обучения: Выбор правильной скорости обучения (learning rate) градиентного спуска. Слишком большая скорость обучения может привести к расходимости, а слишком маленькая к медленной сходимости или застреванию в локальном минимуме, поэтому нам важно было подобрать этот параметр корректно.

3. Вычислительная сложность на больших наборах данных: На больших наборах данных вычисление градиента требует большого числа итераций

МЕТОД ДИХАТОМИИ И ЗОЛОТОГО СЕЧЕНИЯ

Теперь перейдем ко второму пункту работы:

Рассмотрим графики полученные при работе программы.



Теперь перейдем к результатам работы программы:

Table 1: Optimization Results: Golden Section

Function	Initial Point	Tolerance	Point	Value	Iterations
Rosenbrock	(10000, 1888)	0.01	(10000, 1888)	9.9996e17	2
Quadratic	(10000, 1888)	0.0001	(9939.99, 1888.003)	9.7617e17	1001
Rosenbrock	(10000, 1888)	1e-6	(9939.95, 1888.003)	9.7616e17	1001
Quadratic	(10000, 1888)	0.01	(-2, -2)	3609	2
Rosenbrock	(10000, 1888)	0.0001	(0.5613, 0.2318)	0.885387	1001
Quadratic	(10000, 1888)	1e-6	(0.5611, 0.2315)	0.885808	1001
Rosenbrock	(-2, -2)	0.01	(0.001, 0.003)	0.9989	2
Quadratic	(-2, -2)	0.0001	(0.4922, 0.3187)	0.840806	1001
Rosenbrock	(-2, -2)	1e-6	(0.4919, 0.3183)	0.841072	1001
Quadratic	(-2, -2)	0.01	(10000, 1888)	1.035e8	2
Rosenbrock	(-2, -2)	0.0001	(9941.0296, 1876.8819)	1.023e8	1001
Quadratic	(-2, -2)	1e-6	(9940.99, 1876.8744)	1.023e8	1001
Rosenbrock	(0.001, 0.003)	0.01	(0.001, 0.003)	41	2
Quadratic	(0.001, 0.003)	0.0001	(1.9993, 2.9991)	1e-6	334
Rosenbrock	(0.001, 0.003)	1e-6	(2, 3)	0	383
Quadratic	(0.001, 0.003)	0.01	(0.001, 0.003)	12.978010	2
Rosenbrock	(0.001, 0.003)	0.0001	(1.9993, 2.9989)	2e-6	262
Quadratic	(0.001, 0.003)	1e-6	(2, 3)	0	320

Table 2: Optimization Results: Dichotomy

Function	Initial Point	Tolerance	Point	Value	Iterations
Rosenbrock	(10000, 1888)	0.01	(10000, 1888)	9.9996e17	2
Quadratic	(10000, 1888)	0.0001	(9942.5, 1888.003)	9.7716e17	1001
Rosenbrock	(10000, 1888)	1e-6	(9942.45, 1888.003)	9.7714e17	1001
Quadratic	(10000, 1888)	0.01	(-2, -2)	3609	2
Rosenbrock	(10000, 1888)	0.0001	(0.4997, 0.3252)	0.819319	1001
Quadratic	(10000, 1888)	1e-6	(0.5785, 0.252)	0.86129	1001
Rosenbrock	(-2, -2)	0.01	(0.001, 0.003)	0.9989	2
Quadratic	(-2, -2)	0.0001	(0.5127, 0.3391)	0.819277	1001
Rosenbrock	(-2, -2)	1e-6	(0.5123, 0.3387)	0.819713	1001
Quadratic	(-2, -2)	0.01	(10000, 1888)	1.035e8	2
Rosenbrock	(-2, -2)	0.0001	(9943.4955, 1877.3468)	1.023e8	1001
Quadratic	(-2, -2)	1e-6	(9943.4469, 1877.3376)	1.023e8	1001
Rosenbrock	(0.001, 0.003)	0.01	(0.001, 0.003)	41	2
Quadratic	(0.001, 0.003)	0.0001	(1.9993, 2.9991)	1e-6	354
Rosenbrock	(0.001, 0.003)	1e-6	(2, 3)	0	403
Quadratic	(0.001, 0.003)	0.01	(0.001, 0.003)	12.978010	2
Rosenbrock	(0.001, 0.003)	0.0001	(1.9992, 2.9988)	2e-6	280
Quadratic	(0.001, 0.003)	1e-6	(2, 3)	0	340

Как мы видим, оба из методов достаточно схожи по результатам, как в плане значений так и по результатам, но на наших данных метод золотого сечения чуть эффективнее, причем оба работают не хуже, а где-то даже лучше чем метод градиентного спуска, например при точках небольших значений, как (0.001;0.003) и максимальной точности,

Плюсы:

1. Метод золотого сечения достаточно эффективен, когда функция имеет один экстремум на отрезке, как в нашем случае
2. Этот метод не требует непрерывной дифференцируемости функции, поэтому он может использоваться для оптимизации негладких функций.

Минусы:

1. Точность: В некоторых случаях метод золотого сечения может работать медленнее по сравнению с другими методами, так как он не всегда обеспечивает быстрое сходимость к минимуму.

Метод дихотомии:

Плюсы:

1. эффективен, даже когда функция имеет несколько экстремумов на заданном интервале.

2. Метод дихотомии не так чувствителен к начальным условиям, как некоторые другие методы.

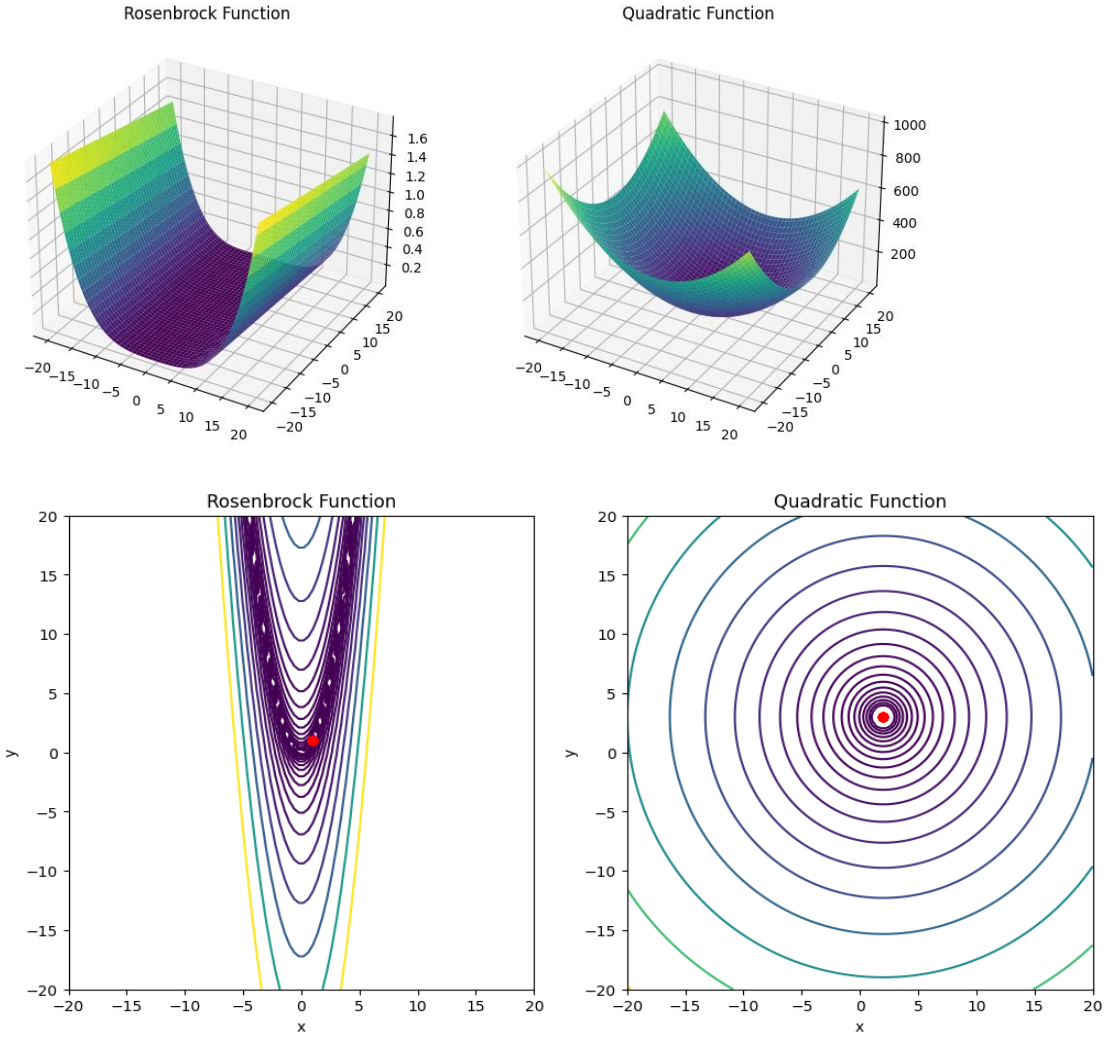
Минусы:

1. Метод дихотомии может иметь медленную скорость сходимости на больших интервалах.

2. Метод дихотомии может быть не очень эффективен для оптимизации негладких функций.

3. Требует больше вычислений по сравнению с методом золотого сечения, метод дихотомии требует больше вычислений за каждую итерацию, как видно из выходных данных

МЕТОД НЕЛДЕРА-МИДА



Результаты метода:

Table 1: Optimization Results

Function	Initial Point	Tolerance	Minimum	Function Value at Minimum	Iterations
Rosenbrock	(10000, 1888)	0.01	[1.00312, 1.00633]	1.03542e-05	646
Quadratic	(-2, -2)	0.0001	[1.00002, 1.00003]	5.00977e-10	660
Rosenbrock	(0.001, 0.003)	1e-6	[1, 1]	1.14087e-13	673
Quadratic	(10000, 1888)	0.01	[0.997731, 0.9957]	1.05479e-05	64
Rosenbrock	(-2, -2)	0.0001	[1.00001, 1.00003]	1.81794e-10	79
Quadratic	(0.001, 0.003)	1e-6	[1, 1]	4.29682e-14	94
Rosenbrock	(10000, 1888)	0.01	[1.00098, 1.00226]	9.69496e-06	72
Quadratic	(-2, -2)	0.0001	[1.00002, 1.00004]	4.29235e-10	87
Rosenbrock	(0.001, 0.003)	1e-6	[1, 1]	6.44232e-14	101
Quadratic	(10000, 1888)	0.01	[2.00262, 2.99748]	1.32008e-05	56
Rosenbrock	(-2, -2)	0.0001	[2, 3]	2.63024e-10	71
Quadratic	(0.001, 0.003)	1e-6	[2, 3]	2.52297e-13	85
Rosenbrock	(10000, 1888)	0.01	[2.00302, 2.99667]	2.02160e-05	35
Quadratic	(-2, -2)	0.0001	[2.00003, 3.00001]	9.32861e-10	52
Rosenbrock	(0.001, 0.003)	1e-6	[2, 3]	1.14531e-13	65
Quadratic	(10000, 1888)	0.01	[2.00372, 3.00174]	1.68383e-05	69
Rosenbrock	(-2, -2)	0.0001	[2, 3]	3.86720e-10	84
Quadratic	(0.001, 0.003)	1e-6	[2, 3]	9.68225e-14	99

Как мы видим, ситуация по количеству итераций, при небольшой точности для всех точек кроме (0.001;0.003) ,этот метод выходит не очень эффективным по сравнению с предыдущими методами , зато в остальных случаях видно, что эффективность сильно возрастает, но точность слегка ниже и в некоторых случаях отстает примерно на 0.1 от предыдущих методов. Из данных, а также известной теории можем отметить:

Плюсы:

1. Метод Нелдера-Мида не требует наличия градиентов функции. Это будет важно полезно, если функция недифференцируема или тяжело ее дифференцировать

2. относительно прост в реализации, тем более есть уже готовая библиотека которую мы и использовали

3. Метод Нелдера-Мида может быть использован для функций в многомерных пространствах

Минусы:

1. Метод Нелдера-Мида может быть медленным при сходимости к оптимальному решению, например в случае большого числа параметров или сложной функции
2. Также он не гарантирует нахождение глобального минимума функции и результаты могут зависеть от начального приближения
3. В высокоразмерных пространствах метод Нелдера-Мида может столкнуться с плохой эффективности из-за большого объема вычислений, (к сожалению не успели доделать доп. Пункт 2)

ЗАКЛЮЧЕНИЕ

Подводя итог к работе, хотелось бы отметить , что мы освоили основную теорию алгоритмов предложенных в работе, а также реализовали их на практике, что помогло узнать их плюсы и минусы в зависимости от начальных данных и исходных функций, показав в каких случаях лучше применять тот или иной из методов.