Отчет по Лабораторной работе #1

Методы нулевого порядка

Авторы:
Ивченков Дмитрий, M3234
Тюленев Вадим, M3234
Веселкова Варвара, M3234

Постановка задачи

Исследовать эффективность методов нулевого порядка в задачах безусловной оптимизации.

Основное задание

Реализуйте и исследуйте на эффективность следующие методы:

1. Метод градиентного спуска с постоянным шагом (learning rate);
2. Любой метод одномерного поиска и градиентный спуск на его основе;
3. Метод Нелдера-Мида, используя готовую реализацию в Python библиотеке `scipy.optimize`.

Описание методов

1. Метод градиентного спуска с постоянным шагом

В качестве направления убывания берем $p_k = -\nabla f(x_k)$ (обратное градиенту), причем понятно, что в некоторой окрестности точки x_k оно будет обеспечивать наискорейшее убывание функции (из свойств градиента). В таком случае $x_{k+1} = x_k + \alpha_k p_k$, где α мы подбираем так, что $f(x_{k+1}) < f(x_k)$.
Так как у нас постоянный шаг, мы подбираем α (или, по-другому, learning rate) как константу. Понятно, что при выборе больших значений α для поиска минимума потребуется меньше итераций (мы будем быстрее двигаться к ответу), однако и точность будет ниже. При выборе меньших значений α будет увеличиваться как точность, так и количество итераций. α необходимо подобрать так, чтобы получать достаточную точность с не слишком большим числом итераций.

Код:

```python
def gradient_descent_with_path(grad_f, x0, learning_rate=0.002, tolerance=1e-6, max_iterations=100000):
    x = np.array(x0, dtype=np.float64)
    path = [x.copy()]
    for i in range(max_iterations):
        grad = grad_f(x)
        x_prev = x
        step = learning_rate * grad
        if np.linalg.norm(step) > 1:
            step = step / np.linalg.norm(step)
        x = x - step
        path.append(x.copy())
        if np.linalg.norm(x - x_prev) < tolerance:
            break
    return x, i + 1, np.array(path)
```

2. Метод одномерного поиска и градиентный спуск на его основе

Здесь мы также используем метод градиентного спуска, однако learning rate адаптивно изменяется на каждой итерации, что позволяет улучшить сходимость алгоритма.

Learning rate подбирается с целью минимизации значения функции вдоль направления антиградиента.
Для выбора подходящего α на каждой итерации мы использовали метод дихотомии (тернарный поиск).

Код:

```
def search_dichotomy(f, grad_f, x, tol=1e-5, max_iterations=100):
    a, b = 0, 1
    sigma = tol / 2

    for _ in range(max_iterations):
        midpoint = (a + b) / 2
        left = midpoint - sigma
        right = midpoint + sigma
        f_left = f(x - left * grad_f(x))
        f_right = f(x - right * grad_f(x))
        if f_left < f_right:
            b = midpoint
        else:
            a = midpoint
        if b - a < tol:
            break
    return (a + b) / 2
```

3. Метод Нелдера-Мида

Это метод, не использующий градиент вовсе. Если описывать простыми словами, то мы используем треугольник, который мы можем перемещать, отражать и сжимать для исследования функции. Если треугольник перестает существенно перемещаться, значит, мы достигли нужной точки.
Если описывать более формально, то мы используем симплекс для поиска минимума функции. Для начала выбираются $n+1$ начальные точки, формирующие симплекс, и вычисляются значения функции в этих точках. Дальше выбираются точки с наибольшим, следующим по величине и наименьшим значениями функции, точка с наибольшим значением функции отражается от центра масс остальных точек.
Дальше происходит адаптация симплекса по следующим правилам: если новая точка лучше всех текущих, мы растягиваем симплекс в направлении улучшения; если новая точка улучшает ситуацию, но не является лучшей, мы сжимаем симплекс в этом направлении; если отраженная точка хуже наихудших, мы сжимаем симплекс по направлению к лучшей из имеющихся точек. Этот процесс повторяется, пока не будет достигнута достаточная близость вершин симплекса друг к другу. В языке программирования Python метод Нейлера-Мида реализован в библиотеке `scipy.optimize`, им мы и пользуемся.

Код:

```
from scipy.optimize import minimize

def apply_nelder_mead_with_path(f, x0):
    path = [np.array(x0)]
    def callback(x):
        path.append(x.copy())
    result = minimize(f, x0, method='Nelder-Mead', callback=callback)
    return result.x, result.nit, result.nfev, np.array(path)
```

4. Дополнительное задание 1: Метод покоординатного спуска

Это алгоритм оптимизации, который ищем минимум функции многих переменных путём последовательного улучшения каждой координаты вектора переменных. Процесс оптимизации происходит следующим образом: сначала фиксируются значения всех переменных вектора $x = (x_1, \ldots, x_n)$, кроме одной выбранной переменной x_i. Дальше, производим оптимизацию по каждой координате: функция $f(x_i)$ минимизируется по переменной x_i методом одномерной оптимизации (мы используем уже реализованный тернарный поиск). Вектор x обновляется, изменяя только одну координату x_i так, чтобы достичь минимума функции f в этом направлении.
Критерием остановки является достаточная мелкота нормы разности последовательных приближений вектора x или абсолютная разность значений функции в последовательных приближениях.

Код:

```python
import numpy as np

def coordinate_descent(f, grad, x0, tol=1e-4, max_iter=100000):
    x = np.array(x0, dtype=np.float64)
    path = [x.copy()]
    for _ in range(max_iter):
        x_prev = x.copy()
        for ind in range(len(x)):
            def f_single(var):
                temp = x.copy()
                temp[ind] = var
                return f(temp)

            def grad_f_single(var):
                temp = x.copy()
                temp[ind] = var
                d = grad(temp)
                return d[ind] if isinstance(d, np.ndarray) else d

            x_single = x[ind]
            optimal_step = search_dichotomy(f_single, grad_f_single,
    x_single)
            x[ind] = x_single - optimal_step * grad_f_single(x_single)

        path.append(x.copy())
        if np.linalg.norm(x - x_prev) <= tol or np.abs(f(x) - f(x_prev)
    ) <= tol:
            break

    return x, _, np.array(path)
```

Сравнение эффективности для двух функций

Функция 1: $f = (x-2)^2 + (y-2)^2$

1. 3D График функции

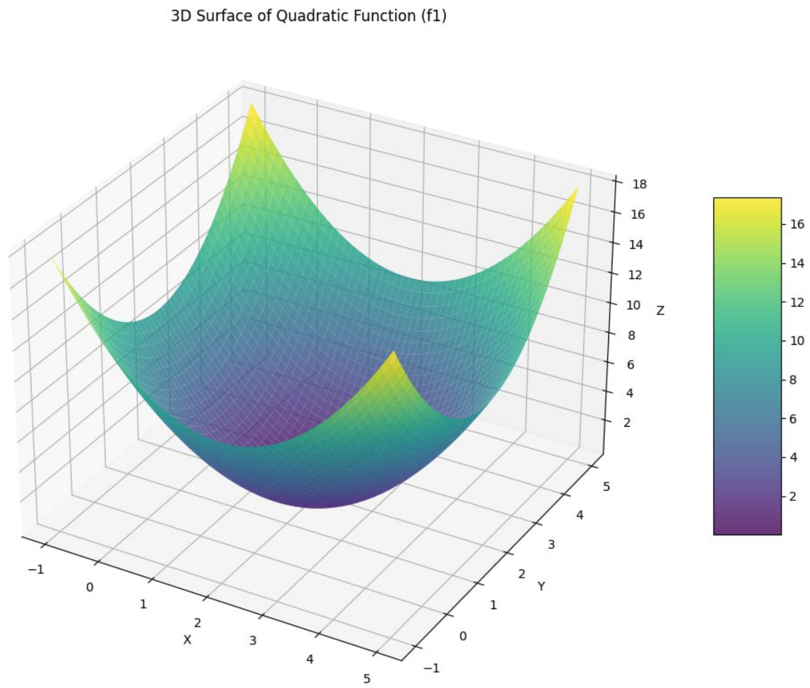

Рис. 1: 3D график функции

2. Сравнительная таблица

Таблица 1: Функция $f = (x-2)^2 + (y-2)^2$

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
Coordinate Descent	[50 50]	1e-06	-	4	[2. 2.]
Coordinate Descent	[-1 -1]	1e-06	-	4	[2. 2.]
Coordinate Descent	[50 50]	0.0001	-	4	[2. 2.]
Coordinate Descent	[-1 -1]	0.0001	-	4	[2. 2.]
Coordinate Descent	[1 0]	0.01	-	4	[2. 2.]
Coordinate Descent	[1 0]	1e-06	-	4	[2. 2.]
Coordinate Descent	[-1 -1]	0.01	-	4	[2. 2.]
Coordinate Descent	[1 0]	0.0001	-	4	[2. 2.]
Coordinate Descent	[50 50]	0.01	-	4	[2. 2.]
GD with Bin Search	[50 50]	0.01	-	69	[2. 2.]
GD with Bin Search	[50 50]	0.0001	-	69	[2. 2.]
GD with Bin Search	[-1 -1]	1e-06	-	7	[2. 2.]
GD with Bin Search	[1 0]	1e-06	-	5	[2. 2.]
GD with Bin Search	[-1 -1]	0.0001	-	6	[2. 2.]
GD with Bin Search	[1 0]	0.0001	-	4	[2. 2.]
GD with Bin Search	[-1 -1]	0.01	-	6	[2. 2.]
GD with Bin Search	[1 0]	0.01	-	4	[2. 2.]
GD with Bin Search	[50 50]	1e-06	-	70	[2. 2.]
GD with Constant Step	[50 50]	0.0001	0.005	879	[2.00699161 2.00699161]
GD with Constant Step	[50 50]	1e-06	0.001	5905	[2.00035244 2.00035244]
GD with Constant Step	[50 50]	0.0001	0.002	1974	[2.01758633 2.01758633]

Таблица 1 – Продолжение

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
GD with Constant Step	[50 50]	0.0001	0.001	3605	[2.03522432 2.03522432]
GD with Constant Step	[50 50]	0.01	0.002	825	[3.75871488 3.75871488]
GD with Constant Step	[50 50]	1e-06	0.002	3123	[2.00017586 2.00017586]
GD with Constant Step	[50 50]	1e-06	0.005	1338	[2.00006936 2.00006936]
GD with Constant Step	[-1 -1]	0.01	0.005	145	[1.30140661 1.30140661]
GD with Constant Step	[-1 -1]	0.01	0.002	133	[0.23959228 0.23959228]
GD with Constant Step	[50 50]	0.01	0.001	1304	[5.52750044 5.52750044]
GD with Constant Step	[-1 -1]	1e-06	0.001	4520	[1.99964749 1.99964749]
GD with Constant Step	[-1 -1]	1e-06	0.005	1062	[1.99993055 1.99993055]
GD with Constant Step	[1 0]	0.0001	0.001	1900	[1.97771419 1.95542838]
GD with Constant Step	[1 0]	0.0001	0.002	1123	[1.9889022 1.97780439]
GD with Constant Step	[1 0]	0.0001	0.005	540	[1.99560453 1.99120906]
GD with Constant Step	[1 0]	1e-06	0.001	4200	[1.99977702 1.99955403]
GD with Constant Step	[1 0]	1e-06	0.002	2272	[1.99988903 1.99977805]
GD with Constant Step	[1 0]	1e-06	0.005	998	[1.99995595 1.9999119]
GD with Constant Step	[1 0]	0.01	0.001	1	[1.002 0.004]
GD with Constant Step	[-1 -1]	0.01	0.001	1	[-0.994 -0.994]
GD with Constant Step	[1 0]	0.01	0.002	1	[1.004 0.008]
GD with Constant Step	[-1 -1]	0.0001	0.001	2220	[1.96476917 1.96476917]
GD with Constant Step	[-1 -1]	0.0001	0.002	1282	[1.98239674 1.98239674]
GD with Constant Step	[-1 -1]	0.0001	0.005	604	[1.99306927 1.99306927]
GD with Constant Step	[50 50]	0.01	0.005	421	[2.69768304 2.69768304]
GD with Constant Step	[-1 -1]	1e-06	0.002	2431	[1.99982398 1.99982398]
GD with Constant Step	[1 0]	0.01	0.005	82	[1.5613825 1.122765]
Nelder-Mead	[1 0]	-	-	65	[1.99995681 2.00001611]
Nelder-Mead	[-1 -1]	-	-	44	[2.00002848 1.99998398]
Nelder-Mead	[50 50]	-	-	51	[2.00003014 1.99996591]

3. Графики с линиями уровня и траекториями методов

Начальная точка [1, 0]

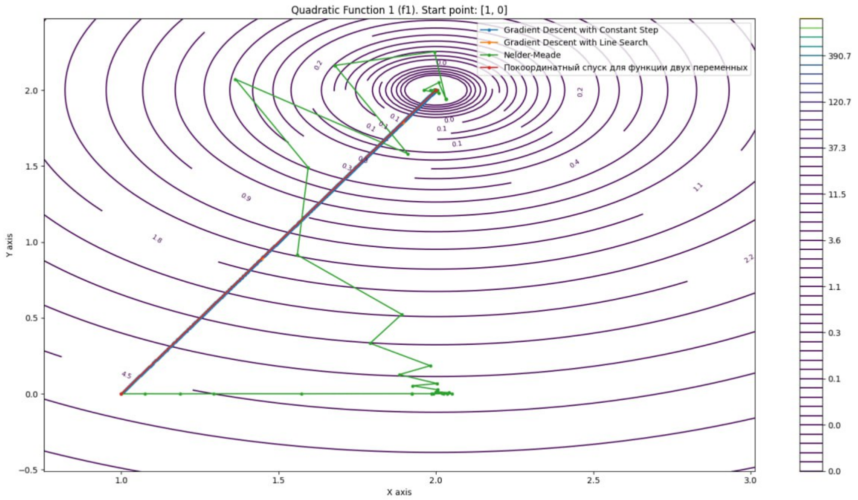

Рис. 2: Линии уровня и траектории методов для начальной точки 1

Начальная точка [-1, -1]

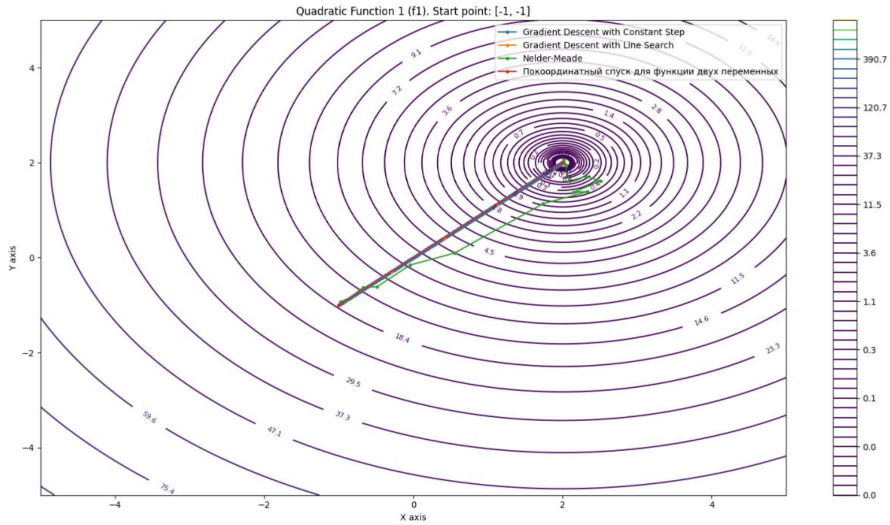

Рис. 3: Линии уровня и траектории методов для начальной точки 2

Начальная точка [50, 50]

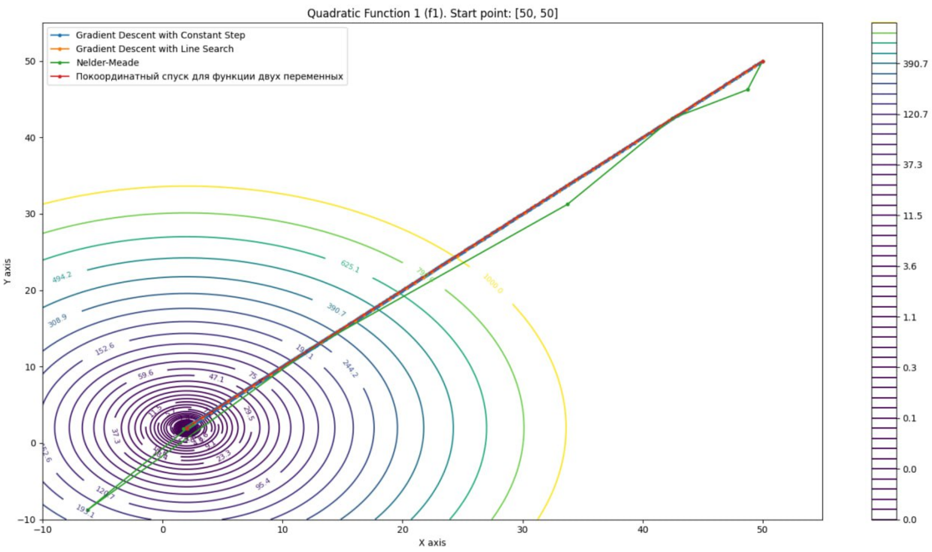

Рис. 4: Линии уровня и траектории методов для начальной точки 3

Функция 2: $f = 0.1 * (x^2 + y^2)$

1. 3D График функции

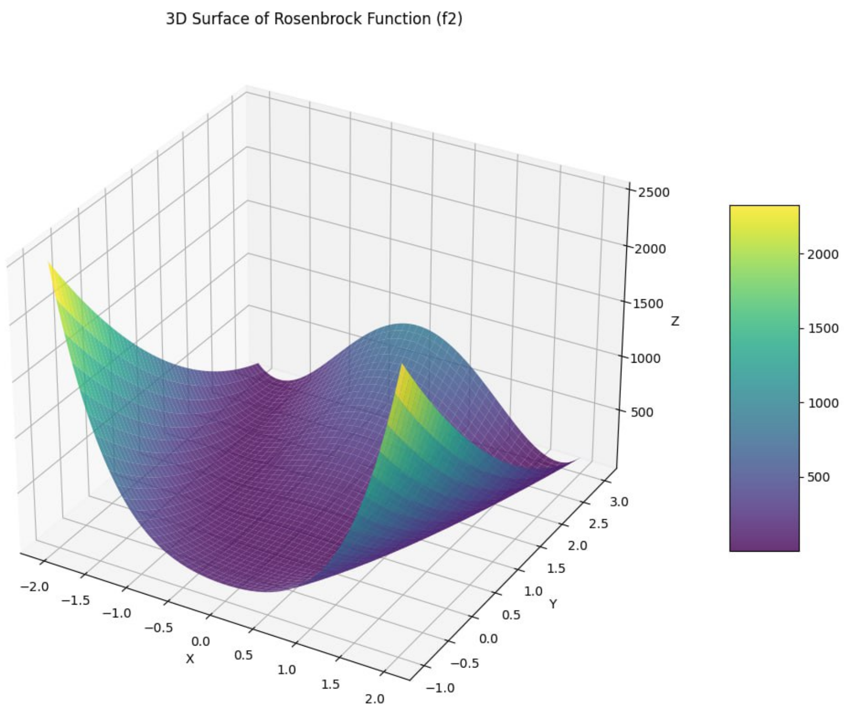

Рис. 5: 3D график функции

2. Сравнительная таблица

Таблица 2: Функция $f = (x-2)^2 + (y-2)^2$

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
Coordinate Descent	[50 50]	1e-06	-	86	[-0.00025535 0.00340252]
Coordinate Descent	[-1 -1]	1e-06	-	60	[-0.00262875 -0.00236765]
Coordinate Descent	[50 50]	0.0001	-	64	[-0.00207617 0.03961263]
Coordinate Descent	[-1 -1]	0.0001	-	38	[-0.03017431 -0.0272245]
Coordinate Descent	[1 0]	0.01	-	8	[4.09601563e-01 -5.64391874e-07]
Coordinate Descent	[1 0]	1e-06	-	50	[3.77798326e-03 -6.78150818e-07]
Coordinate Descent	[-1 -1]	0.01	-	16	[-0.30187573 -0.27741061]
Coordinate Descent	[1 0]	0.0001	-	30	[3.51848754e-02 -6.77326110e-07]
Coordinate Descent	[50 50]	0.01	-	44	[-0.0023819 0.36893984]
GD with Bin Search	[50 50]	0.01	-	84	[0.02378653 0.02378653]
GD with Bin Search	[50 50]	0.0001	-	104	[0.00027027 0.00027027]
GD with Bin Search	[-1 -1]	1e-06	-	62	[-2.31893384e-06 -2.31893384e-06]
GD with Bin Search	[1 0]	1e-06	-	57	[2.78012497e-06 -1.81798456e-06]
GD with Bin Search	[-1 -1]	0.0001	-	41	[-0.00025138 -0.00025138]
GD with Bin Search	[1 0]	0.0001	-	36	[0.00030144 -0.00019705]
GD with Bin Search	[-1 -1]	0.01	-	20	[-0.02680213 -0.02680213]
GD with Bin Search	[1 0]	0.01	-	15	[0.03295443 -0.02069573]
GD with Bin Search	[50 50]	1e-06	-	125	[2.49240601e-06 2.49240601e-06]
GD with Constant Step	[50 50]	0.0001	0.005	3372	[0.06827916 0.06827916]
GD with Constant Step	[50 50]	1e-06	0.001	31507	[0.00352811 0.00352811]
GD with Constant Step	[50 50]	0.0001	0.002	6357	[0.16329958 0.16329958]
GD with Constant Step	[50 50]	0.0001	0.001	9892	[0.30643868 0.30643868]
GD with Constant Step	[50 50]	0.01	0.002	738	[5.02089173 5.02089173]
GD with Constant Step	[50 50]	1e-06	0.002	17479	[0.00176519 0.00176519]
GD with Constant Step	[50 50]	1e-06	0.005	7909	[0.00070545 0.00070545]
GD with Constant Step	[-1 -1]	0.01	0.005	1	[-0.9995 -0.9995]
GD with Constant Step	[-1 -1]	0.01	0.002	1	[-0.9998 -0.9998]
GD with Constant Step	[50 50]	0.01	0.001	990	[7.4577109 7.4577109]
GD with Constant Step	[-1 -1]	1e-06	0.001	31671	[-0.00354109 -0.00354109]
GD with Constant Step	[-1 -1]	1e-06	0.005	7945	[-0.00070629 -0.00070629]
GD with Constant Step	[1 0]	0.0001	0.001	3518	[0.49211388 -0.12467661]
GD with Constant Step	[1 0]	0.0001	0.002	3545	[0.23777354 -0.09096412]
GD with Constant Step	[1 0]	0.0001	0.005	2359	[0.09151957 -0.04213717]
GD with Constant Step	[1 0]	1e-06	0.001	26857	[0.004458 -0.00226581]
GD with Constant Step	[1 0]	1e-06	0.002	15162	[0.00222624 -0.00113456]
GD with Constant Step	[1 0]	1e-06	0.005	6980	[0.00088923 -0.00045414]
GD with Constant Step	[1 0]	0.01	0.001	1	[9.998e-01 -1.000e-04]
GD with Constant Step	[-1 -1]	0.01	0.001	1	[-0.9999 -0.9999]
GD with Constant Step	[1 0]	0.01	0.002	1	[9.996e-01 -2.000e-04]
GD with Constant Step	[-1 -1]	0.0001	0.001	6060	[-0.45868247 -0.45868247]
GD with Constant Step	[-1 -1]	0.0001	0.002	5550	[-0.19590085 -0.19590085]
GD with Constant Step	[-1 -1]	0.0001	0.005	3268	[-0.07330174 -0.07330174]
GD with Constant Step	[50 50]	0.01	0.005	491	[2.87851094 2.87851094]
GD with Constant Step	[-1 -1]	1e-06	0.002	17572	[-0.00176842 -0.00176842]
GD with Constant Step	[1 0]	0.01	0.005	1	[9.99e-01 -5.00e-04]
Nelder-Mead	[1 0]	-	-	43	[2.05649609e-05 2.46854500e-05]
Nelder-Mead	[-1 -1]	-	-	38	[2.10235293e-05 -2.54845649e-05]

Таблица 2 – Продолжение

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
Nelder-Mead	[50 50]	-	-	52	[2.18382909e-05 -3.27508130e-05]

3. Графики с линиями уровня и траекториями методов

Начальная точка [1, 0]

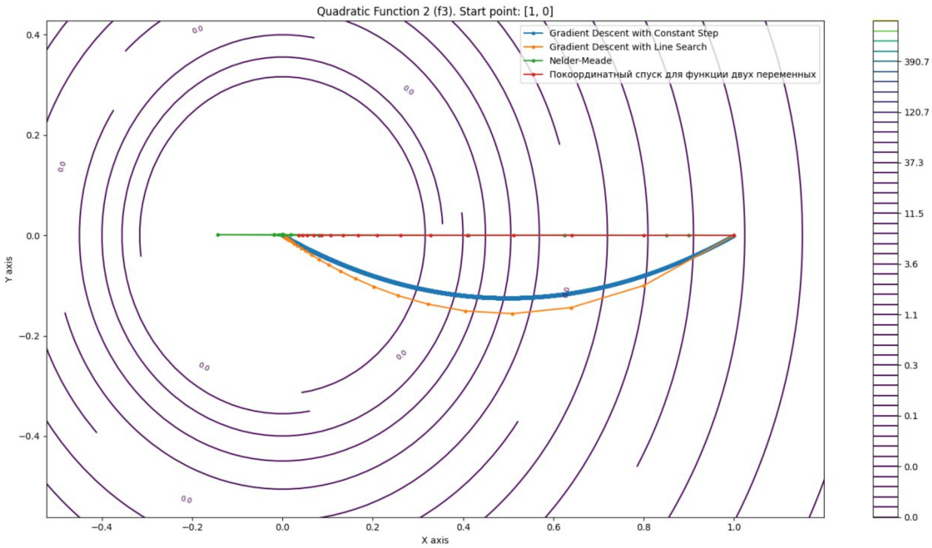

Рис. 6: Линии уровня и траектории методов для начальной точки 1

Начальная точка [-1, -1]

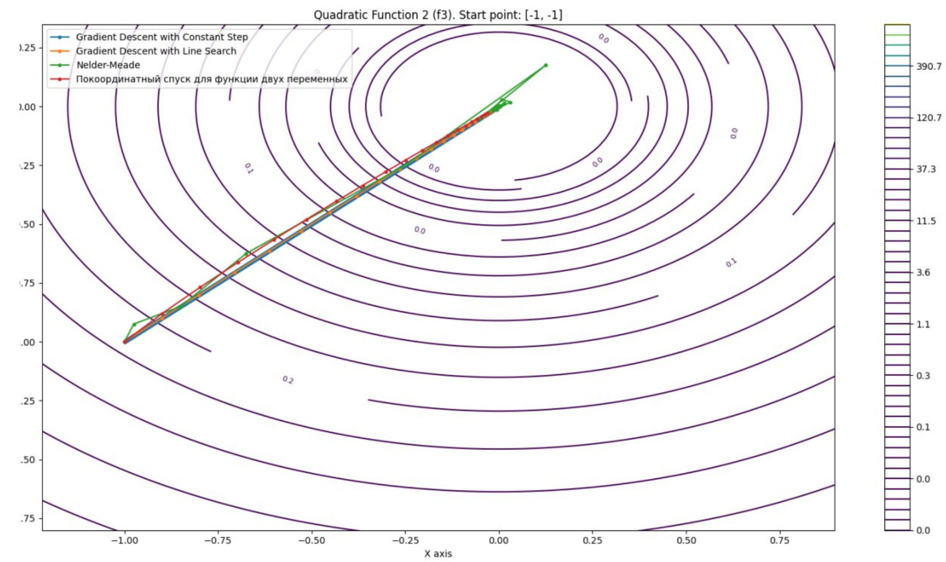

Рис. 7: Линии уровня и траектории методов для начальной точки 2

Начальная точка [50, 50]

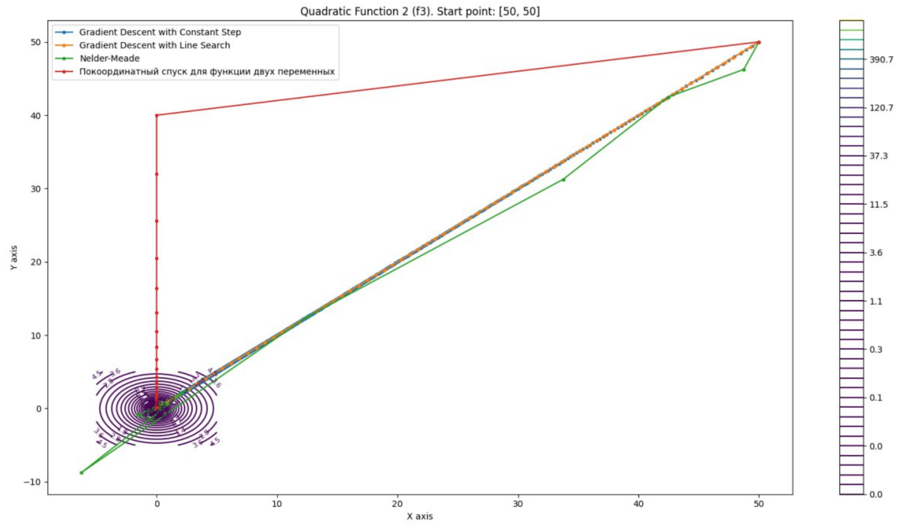

Рис. 8: Линии уровня и траектории методов для начальной точки 3

Выводы

1. **Влияние ландшафта функции на сходимость.** Для первой функции все методы демонстрируют достаточно быструю сходимость по сравнению с функцией 2.

2. **Влияние допустимой погрешности.** Установление слишком высокого порога допустимой погрешности может привести к тому, что методы оптимизации остановятся до того, как будет найдена достаточно точная точка минимума.

3. **Зависимость от начальной точки.** Расстояние от начальной точки до точки минимума оказывает существенное влияние на количество итераций, необходимых для сходимости. Чем дальше начальная точка, тем больше итераций может потребоваться.

4. **Выбор размера шага.** Слишком большой размер шага (*learning rate*) в методах градиентного спуска может привести к "перепрыгиванию"через минимум и, как следствие, к отсутствию сходимости.

5. **Эффективность методов.** Среди рассмотренных методов, градиентный спуск с фиксированным размером шага показал себя как наименее эффективный в сравнении с другими методами.

Доп. задание 2

Исследуйте эффективность методов на функциях n переменных, в зависимости от размерности пространства n

Функция определяется как $f_n(x) = \sum_{i=1}^{n}(x_i - 1)^2$, где x_i — i-я компонента вектора x, а n — размерность вектора x.

Начальная точка: [0, 0 ... 0]

Method	Dimension	Iterations	End point
Coordinate Descent	2	4	[1. 1.]
Coordinate Descent	5	10	[1. 1. 1. 1. 1.]
Coordinate Descent	10	20	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
GD with Constant Step	2	2157	[0.99982405 0.99982405]
GD with Constant Step	5	2272	[0.9998 0.9998 0.9998 0.9998 0.9
GD with Constant Step	10	2358	[0.9999 0.9999 0.9999 0.9999.
GD with Line Search	2	4	[1. 1.]
GD with Line Search	5	5	[1. 1. 1. 1. 1.]
GD with Line Search	10	6	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Nelder-Mead	2	44	[1.0002 1.0003]
Nelder-Mead	5	482	[1.0002 0.9999 1.0003 0.9999 0.9
Nelder-Mead	10	1429	[0.3109 1.0732 0.8251 1.6459 0.4125 0.7975 -0.147

Исследуйте эффективность методов на плохо обусловленных функциях двух переменных

Функция определяется как $f(x) = \frac{1}{2}x^T Ax$, где A — диагональная матрица с элементами диагонали $A_{ii} = \kappa^{i/(n-1)}$ для $i = 0, 1, \ldots, n-1$, κ — число обусловленности, а n — размерность вектора x. Значение A_{ii} линейно увеличивается от 1 до κ, обеспечивая изменение обусловленности функции.

Method	Dimension	Iterations	End point
Coordinate Descent	10	4	[7.27595761e-12 2.61934474e-10]
GD with Constant Step	10	6213	[9.98500651e-04 3.80569552e-28]
GD with Line Search	10	12	[8.09119027e-08 8.09103485e-08]
Nelder-Mead	10	43	[-2.02775418e-05 1.10370702e-05]
Coordinate Descent	100	4	[7.27595761e-12 1.40862539e-08]
GD with Constant Step	100	6213	[9.98500651e-004 2.55670812e-285]
GD with Line Search	100	11	[2.64415464e-06 9.37474630e-10]
Nelder-Mead	100	48	[-2.49075394e-05 2.65919629e-06]
Coordinate Descent	1000	6	[2.77555756e-17 -1.74089081e-08]
GD with Constant Step	1000	6213	[0.0009985 0.]
GD with Line Search	1000	1200	[2.66976352e-04 8.18740952e-08]
Nelder-Mead	1000	53	[-2.94638334e-05 -2.75296139e-06]

Исследуйте эффективность методов на функциях с зашумленными значениями и на мультимодальных функциях

Зашумленная функция

Функция определяется как $f_{\text{multy}}(x) = (x^2 + y^2 - 13)^2 + (x + y^2 - 5)^2$

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
Coordinate Descent	[1 0]	1e-06	-	200000	[-2.20224557 -2.29999997
Coordinate Descent	[1 0]	0.01	-	4	[0.99990845 -2.29999561
Coordinate Descent	[-1 -1]	1e-06	-	200000	[-3.13483038 -2.29999996
Coordinate Descent	[-1 -1]	0.01	-	4	[-1.00006103 -2.29999356
GD with Bin Search	[1 0]	1e-06	-	10000	[0.55926126 -0.16894985
GD with Bin Search	[1 0]	0.01	-	1	[9.99954224e-01 -1.75476074
GD with Bin Search	[-1 -1]	1e-06	-	10000	[-1.29382583 -1.09549339
GD with Bin Search	[-1 -1]	0.01	-	1	[-1.00003052 -1.00000992
GD with Constant Step	[1 0]	1e-06	0.001	4727	[-4.99953418 -2.29982143
GD with Constant Step	[1 0]	0.01	0.001	127	[-0.34703277 -0.51636256
GD with Constant Step	[-1 -1]	1e-06	0.001	4516	[-4.99952621 -2.29984602
GD with Constant Step	[-1 -1]	0.01	0.001	1	[-1.008 -1.0026]
Nelder-Mead	[1 0]	-	-	72	[4.99998377 -2.30001572
Nelder-Mead	[-1 -1]	-	-	51	[4.99996534 -2.29998578

Мультимодальная функция

Функция определяется как $f_{\text{multy}}(x) = (x^2 + y^2 - 13)^2 + (x + y^2 - 5)^2$

Method	*Startpoint*	*Tolerance*	*LR*	*Iterations*	*Endpoint*
Coordinate Descent	[1 0]	1e-06	-	4	[3.63158434 0.]
Coordinate Descent	[1 0]	0.01	-	4	[3.63158434 0.]
Coordinate Descent	[-1 -1]	1e-06	-	42	[3.37212262 -1.27606159]
Coordinate Descent	[-1 -1]	0.01	-	20	[3.35254808 -1.30535086]
GD with Bin Search	[1 0]	1e-06	-	5	[3.63158438 0.]
GD with Bin Search	[1 0]	0.01	-	4	[3.63158429 0.]
GD with Bin Search	[-1 -1]	1e-06	-	31	[-2.37228054 -2.71519503]
GD with Bin Search	[-1 -1]	0.01	-	11	[-2.36308934 -2.72102146]
GD with Constant Step	[1 0]	1e-06	0.001	130	[3.63157629 0.]
GD with Constant Step	[1 0]	0.01	0.001	49	[3.54850122 0.]
GD with Constant Step	[-1 -1]	1e-06	0.001	364	[-2.37225103 -2.7152083]
GD with Constant Step	[-1 -1]	0.01	0.001	41	[-2.04837964 -2.8079612]
Nelder-Mead	[1 0]	-	-	93	[3.37230234 1.27578877]
Nelder-Mead	[-1 -1]	-	-	45	[-2.37224693 -2.71520526]

Выводы

1. **Влияние зашумлённости.** Градиентные методы не могут найти точку при работе с функциями, подверженными зашумлению, в то время как метод Нелдера-Мида может. Это показывает что при непредсказуемомо поведении функции лучше работают методы не зависящие от градиента.

2. **Влияние обусловленности.** Для градиентного спуска с постоянным шагом

наблюдается увеличение числа итераций при росте числа обусловленности функции. Метод Нелдера-Мида работает примерно одинаково. Методы, использующие дихотомию для определения размера шага, используют больше итераций при большей обусловленности.

3. **Влияниеразмерности задачи.** С ростом размерности пространства поиска методы Нелдера-Мида и градиентного спуска с фиксированным шагом требуют большего числа итераций для достижения минимума. Методы, базирующиеся на поиске оптимального шага через дихотомию, показывают стабильность в количестве итераций независимо от размерности, поскольку тернарный поиск находит точку независимо от размера.

4. **Влияние мультимодальности** Все рассмотренные методы находят ближайшую точке начала точку минимума, а не точку глобального минимума.