Отчет по Лабораторной работе #2

Методы первого и высших порядков

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

Постановка задачи

Реализация и анализ эффективности различных методов оптимизации первого и высших порядков

Основное задание

Реализуйте и исследуйте на эффективность следующие методы:

1. Метод Ньютона с постоянным шагом;
2. Метод Ньютона с одномерным поиском (любым методом)
3. `scipy.optimize`: метод `Newton-CG` и один-два квазиньютоновских метода. Изучите параметры вызываемых библиотечных функций.

Описание методов

1. Метод Ньютона с постоянным шагом

Принцип работы для одномерного случая: рассматриваем разложение функции в ряд Тейлора до второй производной
Одномерный случай:

$$f(x_k + p) \approx f(x_k) + pf'(x_k) + \frac{1}{2}p^2 f''(x_k)$$

Минимизируем $f(x_k + p)$ по p:

$$f'(x_k) + pf''(x_k) = 0$$

$$p = -\frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k + p = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Многомерный случай: Аналогично, для функции нескольких переменных, используют градиент и гессиан:

$$x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k)$$

где $H(x_k)$ — гессиан функции в точке x_k, $\nabla f(x_k)$ — градиент функции в этой точке.

Логика остановки итераций: Итерации продолжаются до тех пор, пока не произойдет что-то из этого:

1. абсолютная разница между значениями функции в новой и текущей точках не станет меньше некоторого заданного порога
2. либо пока определитель матрицы гессиана не станет равен нулю, что сделает невозможным дальнейшее вычисление x_{k+1}. В таком случае досчитываем при помощи градиентного спуска.

2. Метод Ньютона с одномерным поиском (любым методом)

В методе Ньютона с одномерным поиском новая точка рассчитывается по формуле:

$$x_{k+1} = x_k + \alpha p_k$$

где α — это оптимальное значение, находящееся с помощью метода дихотомии, для минимизации функции в направлении вектора p_k. Регулировка параметра α позволяет адаптивно управлять длиной шага в зависимости от текущих условий, что способствует более точному приближению к точке минимума

3. `scipy.optimize`: метод `Newton-CG`

Функция `minimize` предназначена для минимизации функции нескольких переменных. Она является частью библиотеки оптимизации, входящей в состав SciPy.

Параметры функции `minimize`

- **fun** – Целевая функция, минимум которой требуется найти. Должна принимать один аргумент – вектор переменных.
- **x0** – Начальное приближение к минимуму. Массив или список, содержащий начальные значения переменных.
- **args** – Дополнительные аргументы, передаваемые в функцию. Кортеж значений, которые будут переданы функции `fun`.
- **method** – Строка, указывающая метод оптимизации, который будет использоваться.
- **jac** – Функция, вычисляющая градиент целевой функции. Если установлено в `True`, градиент будет оценен численно.
- **hess** – Функция, вычисляющая гессиан (матрицу вторых частных производных) функции.
- **hessp** – Функция, вычисляющая произведение гессиана на вектор.
- **bounds** – Границы для переменных оптимизации. Список пар `(min, max)`, определяющих границы для каждой переменной.
- **constraints** – Ограничения, которым должно удовлетворять решение. Может быть представлено списком.
- **tol** – Допуск к остановке.
- **callback** – Функция обратного вызова, которая вызывается на каждом шаге итерации.

Так же существует параметр **options**, позволяющий настраивать метод с большей спецификой.

- **maxiter** – Максимальное количество итераций алгоритма.

- **disp** – Если `True`, отображает процесс оптимизации во время выполнения.
- **tol** – Точность, с которой должен быть найден минимум. Также используется как критерий остановки.
- **gtol** – Толерантность градиента, при достижении которой алгоритм будет остановлен для метода BFGS.
- **norm** – Норма для вычисления размера шага для BFGS.
- **adaptive** – Указывает, следует ли использовать адаптивный алгоритм для регулировки параметров алгоритма для метода Нейлера-Мида.
- **scale** – Параметры масштабирования для каждой переменной в процессе оптимизации для метода TNC.
- **factr** – Коэффициент для оценки точности решения для метода L-BFGS-B.
- **pgtol** – Толерантность градиента для остановки L-BFGS-B.

Используемый нами квазиньютоновский метод - Алгоритм Бройдена — Флетчера — Гольдфарба — Шанно (BFGS)

4. Дополнительное задание 1: Одномерный поиск по правилу Вольфе и метод Ньютона на его основе

Правило Вольфе состоит из двух частей: условия достаточного уменьшения функции и условия кривизны.

Условие достаточного уменьшения Гарантирует, что шаг оптимизации действительно уменьшает значение функции. Формула:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k$$

где α — это размер шага, p_k — направление шага, а c_1 — небольшое положительное число.

Условие кривизны Проверяет, что угол между градиентом функции и направлением шага достаточно велик, чтобы избежать слишком маленьких или неверно направленных шагов:

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k$$

где c_2 — это число, большее c_1, но меньшее 1.

Описание результатов

Для исследования были выбраны следующие функции:

1. **Функция Розенброка:**

$$f(x) = 100 \cdot (x_1 - x_0^2)^2 + (1 - x_0)^2$$

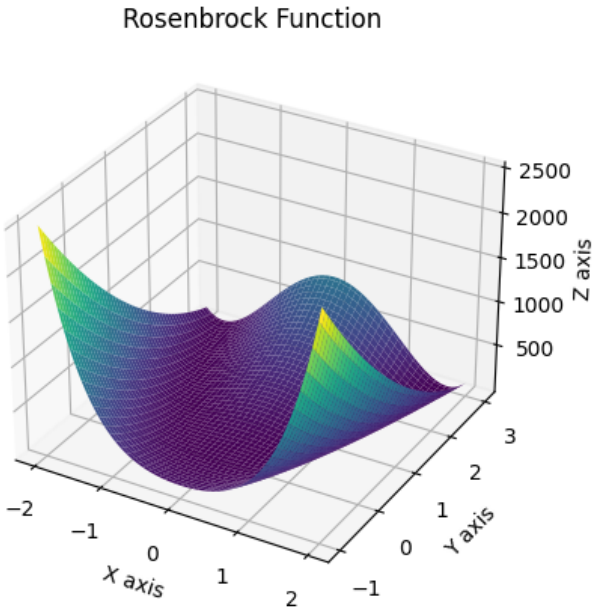

Рис. 1: 3D график функции

2. **Функция кубических квадратов:**

$$f(x) = -\left(\frac{1}{(x_0^2+1)^2} + \frac{1}{(x_1^2+1)^2}\right)^3$$

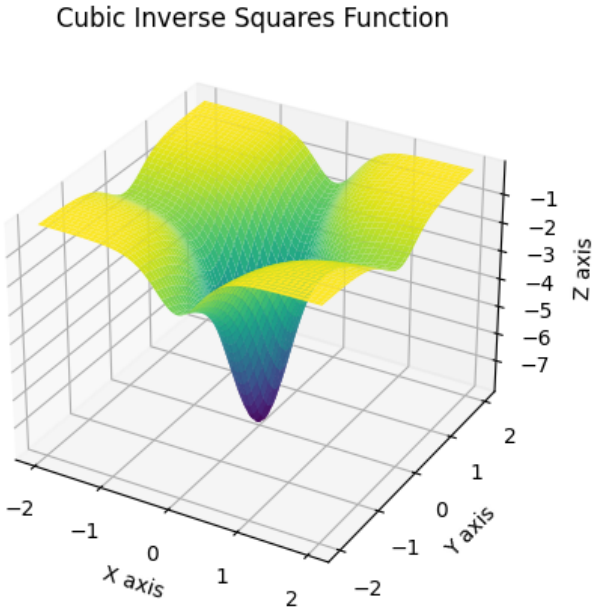

Рис. 2: 3D график функции

Шаги исследования

1. Сравнить реализацию метода Ньютона с методом `Newton-CG` из библиотеки `scipy.optimize` по точности и скорости.

2. Сравнить эффективность методов нулевого порядка и градиентного спуска из предыдущих лабораторных работ с методом Ньютона и квазиньютоновскими методами.

3. Сравнить эффективность методов нулевого порядка с квазиньютоновскими методами, если в последних производная вычисляется разностным методом.

4. Проиллюстрировать примеры на графиках.

```
------------MAIN__TASK-------------
START = [-2, 1]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2
    + y)**2
++
FINAL                [0.997643770252781, 0.995252916921813]
RESULT               0.00000571322491724373
ITERATION            351
GRADIENT EVAL COUNT  351
FUNCTION EVAL COUNT  702.0
++
Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2
    + y)**2
++
FINAL                [0.999689378372377, 0.999376146822458]
RESULT               9.72182600226012E-8
ITERATION            49
GRADIENT EVAL COUNT  49
FUNCTION EVAL COUNT  2058.0
++
Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                [0.999999856836437, 0.999999713094231]
RESULT               2.052929092387938e-14
ITERATION            95
GRADIENT EVAL COUNT  110
FUNCTION EVAL COUNT  305
++
Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
    **2 + y)**2
++
FINAL                [0.9999955481911366, 0.9999910883821501]
RESULT               1.9825034102903414e-11
ITERATION            35
GRADIENT EVAL COUNT  45
FUNCTION EVAL COUNT  135
++
-----------------------------
-----------------------------
```

```
START = [-2, 1]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 + 1)**(-2))**3
++
FINAL                [-5.39477083034862, 6.76208024232295]
RESULT               -0.0000113981676522378
ITERATION            217
GRADIENT EVAL COUNT  217
FUNCTION EVAL COUNT  434.0
++
Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2 + 1)**(-2))**3
++
FINAL                [-2, 1]
RESULT               -0.15746400000000002
ITERATION            1
GRADIENT EVAL COUNT  1
FUNCTION EVAL COUNT  42.0
++
Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)**(-2))**3
++
FINAL                [-2.622349691406688e-19, -4.387299794532255e-19]
RESULT               -8.0
ITERATION            11
GRADIENT EVAL COUNT  21
FUNCTION EVAL COUNT  42
++
Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x**2 + 1)**(-2))**3
++
FINAL                [-8.347416398916561e-09, -1.1072128563335669e-08]
RESULT               -7.999999999999997
ITERATION            10
GRADIENT EVAL COUNT  27
FUNCTION EVAL COUNT  81
++
------------------------------
------------------------------
START = [1, -2]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2 + y)**2
++
FINAL                [0.999713950853545, 0.999198951381365]
RESULT               0.00000532739668015190
ITERATION            181
GRADIENT EVAL COUNT  181
FUNCTION EVAL COUNT  362.0
++
Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2 + y)**2
++
FINAL                [0.999999000937193, 0.999996071771669]
RESULT               3.73528161330276E-10
```

```
ITERATION             3
GRADIENT EVAL COUNT   3
FUNCTION EVAL COUNT   126.0
++
Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                 [0.9999999896813918, 0.9999999793217728]
RESULT                1.0664186496686661e-16
ITERATION             28
GRADIENT EVAL COUNT   36
FUNCTION EVAL COUNT   98
++
Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
    **2 + y)**2
++
FINAL                 [0.9999978441740618, 0.9999955856764816]
RESULT                5.701827520087579e-12
ITERATION             27
GRADIENT EVAL COUNT   75
FUNCTION EVAL COUNT   237
++
------------------------------
------------------------------
START = [0, -2]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
     1)**(-2))**3
++
FINAL                 [-5.00086371455102E-7, -445.374451038327]
RESULT                -1.00001512413731
ITERATION             329
GRADIENT EVAL COUNT   329
FUNCTION EVAL COUNT   658.0
++
Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2
    + 1)**(-2))**3
++
FINAL                 [0, -2]
RESULT                -1.7279999999999998
ITERATION             1
GRADIENT EVAL COUNT   1
FUNCTION EVAL COUNT   42.0
++
Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                 [0.0, -6.977552487619915e-16]
RESULT                -8.0
ITERATION             3
GRADIENT EVAL COUNT   8
FUNCTION EVAL COUNT   11
++
Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x
    **2 + 1)**(-2))**3
++
FINAL                 [-4.6462954148138415e-08, -9.003681359057637e-08]
```

```
RESULT                -7.999999999999848
ITERATION             5
GRADIENT EVAL COUNT   11
FUNCTION EVAL COUNT   33
++
------------------------------
------------------------------------
___________MAIN_SUBTASK____________
------------------------------
START = [-2, 1]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2
    + y)**2
++
FINAL                 [0.997643770252781, 0.995252916921813]
RESULT                0.00000571322491724373
ITERATION             351
GRADIENT EVAL COUNT   351
FUNCTION EVAL COUNT   702.0
++
Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2
     + y)**2
++
FINAL                 [0.999689378372377, 0.999376146822458]
RESULT                9.72182600226012E-8
ITERATION             49
GRADIENT EVAL COUNT   49
FUNCTION EVAL COUNT   2058.0
++
Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                 [0.999999856836437, 0.999999713094231]
RESULT                2.052929092387938e-14
ITERATION             95
GRADIENT EVAL COUNT   110
FUNCTION EVAL COUNT   305
++
Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
    **2 + y)**2
++
FINAL                 [0.9999955481911366, 0.9999910883821501]
RESULT                1.9825034102903414e-11
ITERATION             35
GRADIENT EVAL COUNT   45
FUNCTION EVAL COUNT   135
++
------------------------------
------------------------------
START = [-2, 1]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
     1)**(-2))**3
++
FINAL                 [-5.39477083034862, 6.76208024232295]
RESULT                -0.0000113981676522378
```

```
ITERATION             217
GRADIENT EVAL COUNT   217
FUNCTION EVAL COUNT   434.0
++
Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2
    + 1)**(-2))**3
++
FINAL                 [-2, 1]
RESULT                -0.15746400000000002
ITERATION             1
GRADIENT EVAL COUNT   1
FUNCTION EVAL COUNT   42.0
++
Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                 [-2.622349691406688e-19, -4.387299794532255e-19]
RESULT                -8.0
ITERATION             11
GRADIENT EVAL COUNT   21
FUNCTION EVAL COUNT   42
++
Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x
    **2 + 1)**(-2))**3
++
FINAL                 [-8.347416398916561e-09, -1.1072128563335669e-08]
RESULT                -7.999999999999997
ITERATION             10
GRADIENT EVAL COUNT   27
FUNCTION EVAL COUNT   81
++
------------------------------
------------------------------
START = [1, -2]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2
    + y)**2
++
FINAL                 [0.999713950853545, 0.999198951381365]
RESULT                0.00000532739668015190
ITERATION             181
GRADIENT EVAL COUNT   181
FUNCTION EVAL COUNT   362.0
++
Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2
     + y)**2
++
FINAL                 [0.999999000937193, 0.999996071771669]
RESULT                3.73528161330276E-10
ITERATION             3
GRADIENT EVAL COUNT   3
FUNCTION EVAL COUNT   126.0
++
Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                 [0.9999999896813918, 0.9999999793217728]
```

```
RESULT                1.0664186496686661e-16
ITERATION             28
GRADIENT EVAL COUNT   36
FUNCTION EVAL COUNT   98
++
Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
    **2 + y)**2
++
FINAL                 [0.9999978441740618, 0.9999955856764816]
RESULT                5.701827520087579e-12
ITERATION             27
GRADIENT EVAL COUNT   75
FUNCTION EVAL COUNT   237
++
------------------------------
------------------------------
START = [0, -2]
STEP =  0.1
EPS =  1e-06
Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
    1)**(-2))**3
++
FINAL                 [-5.00086371455102E-7, -445.374451038327]
RESULT                -1.00001512413731
ITERATION             329
GRADIENT EVAL COUNT   329
FUNCTION EVAL COUNT   658.0
++
Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2
    + 1)**(-2))**3
++
FINAL                 [0, -2]
RESULT                -1.7279999999999998
ITERATION             1
GRADIENT EVAL COUNT   1
FUNCTION EVAL COUNT   42.0
++
Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                 [0.0, -6.977552487619915e-16]
RESULT                -8.0
ITERATION             3
GRADIENT EVAL COUNT   8
FUNCTION EVAL COUNT   11
++
Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x
    **2 + 1)**(-2))**3
++
FINAL                 [-4.6462954148138415e-08, -9.003681359057637e-08]
RESULT                -7.999999999999848
ITERATION             5
GRADIENT EVAL COUNT   11
FUNCTION EVAL COUNT   33
++
------------------------------
------------------------------------
------------FIRST_TASK-------------
```

```
------------------------------
Starting point: [-2, 1]
Const step:  0.1
Required accuracy :  1e-06
Running method Wolfe Rule Newton Method for (1 - x)**2 + 100*(-x**2 + y
    )**2
++
FINAL                [0.999675298812091, 0.999348110324489]
RESULT               1.06103086602393E-7
ITERATION            53
GRADIENT EVAL COUNT  53
FUNCTION EVAL COUNT  477.0
++
Running method Scipy Newton-CG Optimizer for (1 - x)**2 + 100*(-x**2 +
    y)**2
++
FINAL                [0.999999856836437, 0.999999713094231]
RESULT               2.052929092387938e-14
ITERATION            95
GRADIENT EVAL COUNT  110
FUNCTION EVAL COUNT  305
++
Running method Dichotomy Newton Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                [0.999689378372377, 0.999376146822458]
RESULT               9.72182600226012E-8
ITERATION            49
GRADIENT EVAL COUNT  49
FUNCTION EVAL COUNT  2058.0
++
------------------------------
------------------------------
Starting point: [-2, 1]
Const step:  0.1
Required accuracy :  1e-06
Running method Wolfe Rule Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                [-6.81211489594668, 9.80729965789102]
RESULT               -0.00000123705033079946
ITERATION            29
GRADIENT EVAL COUNT  29
FUNCTION EVAL COUNT  261
++
Running method Scipy Newton-CG Optimizer for -(1/(y**2 + 1) + (x**2 +
    1)**(-2))**3
++
FINAL                [-2.622349691406688e-19, -4.387299794532255e-19]
RESULT               -8.0
ITERATION            11
GRADIENT EVAL COUNT  21
FUNCTION EVAL COUNT  42
++
Running method Dichotomy Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                [-2, 1]
```

```
RESULT                -0.15746400000000002
ITERATION             1
GRADIENT EVAL COUNT   1
FUNCTION EVAL COUNT   42.0
++
------------------------------
------------------------------
Starting point: [1, -2]
Const step:  0.1
Required accuracy :  1e-06
Running method Wolfe Rule Newton Method for (1 - x)**2 + 100*(-x**2 + y
    )**2
++
FINAL                 [0.999710628356721, 0.999382436739065]
RESULT                2.35085815646672E-7
ITERATION             15
GRADIENT EVAL COUNT   15
FUNCTION EVAL COUNT   135.0
++
Running method Scipy Newton-CG Optimizer for (1 - x)**2 + 100*(-x**2 +
    y)**2
++
FINAL                 [0.9999999896813918, 0.9999999793217728]
RESULT                1.0664186496686661e-16
ITERATION             28
GRADIENT EVAL COUNT   36
FUNCTION EVAL COUNT   98
++
Running method Dichotomy Newton Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
++
FINAL                 [0.999999000937193, 0.999996071771669]
RESULT                3.73528161330276E-10
ITERATION             3
GRADIENT EVAL COUNT   3
FUNCTION EVAL COUNT   126.0
++
------------------------------
------------------------------
Starting point: [0, -2]
Const step:  0.1
Required accuracy :  1e-06
Running method Wolfe Rule Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                 [-5.00018590452804E-7, -1554.69929871097]
RESULT                -1.00000124116044
ITERATION             47
GRADIENT EVAL COUNT   47
FUNCTION EVAL COUNT   423
++
Running method Scipy Newton-CG Optimizer for -(1/(y**2 + 1) + (x**2 +
    1)**(-2))**3
++
FINAL                 [0.0, -6.977552487619915e-16]
RESULT                -8.0
ITERATION             3
GRADIENT EVAL COUNT   8
```

```
FUNCTION EVAL COUNT  11
++
Running method Dichotomy Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
++
FINAL                [0, -2]
RESULT               -1.7279999999999998
ITERATION            1
GRADIENT EVAL COUNT  1
FUNCTION EVAL COUNT  42.0
++
------------------------------
------------------------------------
___________SECOND__TASK____________
------------------------------
START = [0, 1]
STEP =  0.01
EPS =  1e-06
Running method Constant Step Newton Method for (x + y**2 - 7)**2 + (x
    **2 + y - 11)**2
++
FINAL                [-0.270892976011036, -0.906258976479230]
RESULT               181.614134541362
ITERATION            1001
GRADIENT EVAL COUNT  1001
FUNCTION EVAL COUNT  2002.0
++
Running method Dichotomy Step Newton Method for (x + y**2 - 7)**2 + (x
    **2 + y - 11)**2
++
FINAL                [-0.000159133709973960, 0.999204389491602]
RESULT               136.036912058947
ITERATION            1001
GRADIENT EVAL COUNT  1001
FUNCTION EVAL COUNT  42042.0
++
Running method Scipy Newton-CG Method for (x + y**2 - 7)**2 + (x**2 + y
     - 11)**2
++
FINAL                [3.000000000148095, 1.9999999998679516]
RESULT               7.167977428491776e-19
ITERATION            9
GRADIENT EVAL COUNT  13
FUNCTION EVAL COUNT  32
++
Running method Scipy BFGS Quasi-Newton Method for (x + y**2 - 7)**2 + (
    x**2 + y - 11)**2
++
FINAL                [2.999999993244534, 1.9999999979321275]
RESULT               2.0406263010115856e-15
ITERATION            13
GRADIENT EVAL COUNT  21
FUNCTION EVAL COUNT  63
++
------------------------------
------------------------------------
```

Графики

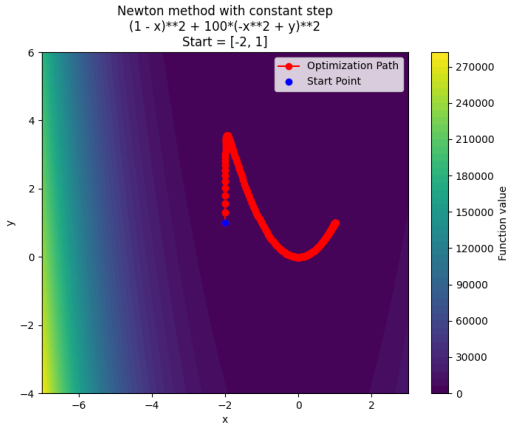

Рис. 3: График 4

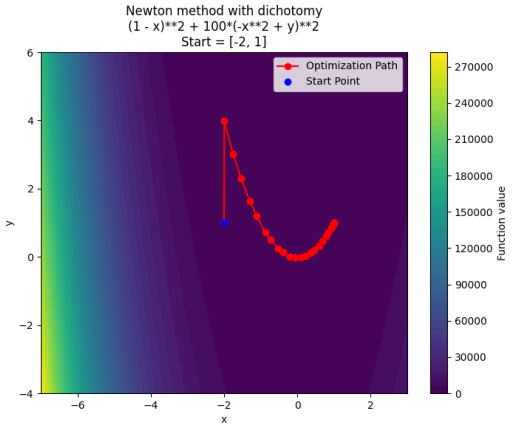

Рис. 4: График 5

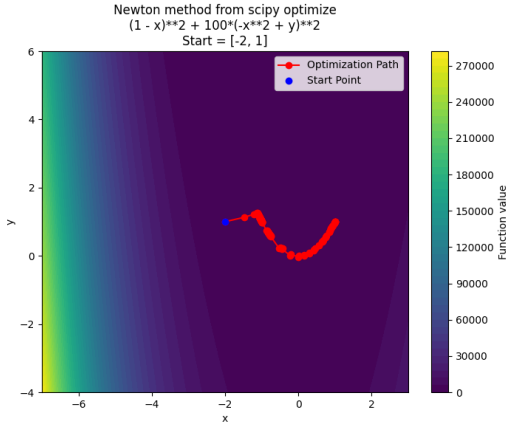

Рис. 5: График 6

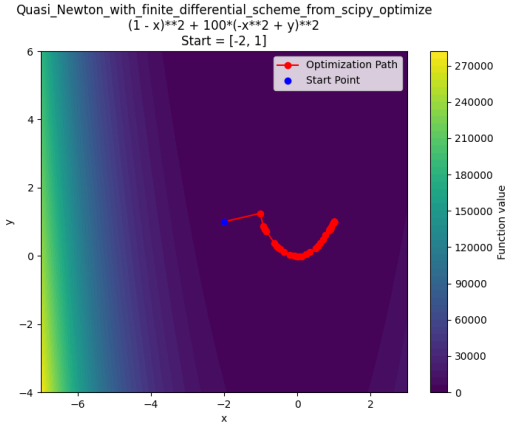

Рис. 6: График 7

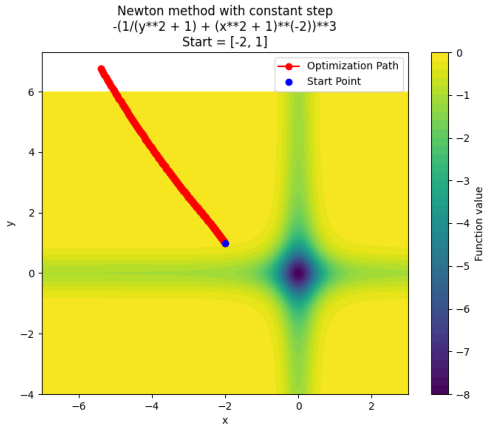

Рис. 7: График 8

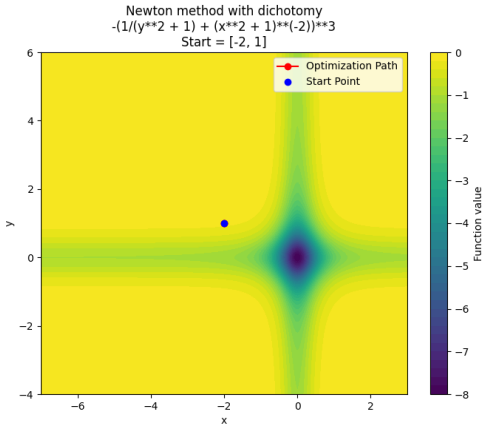

Рис. 8: График 9

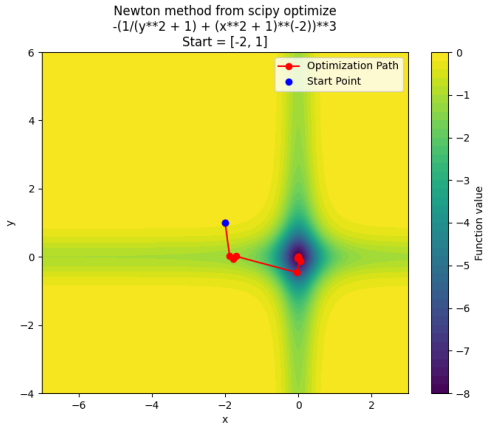

Рис. 9: График 10

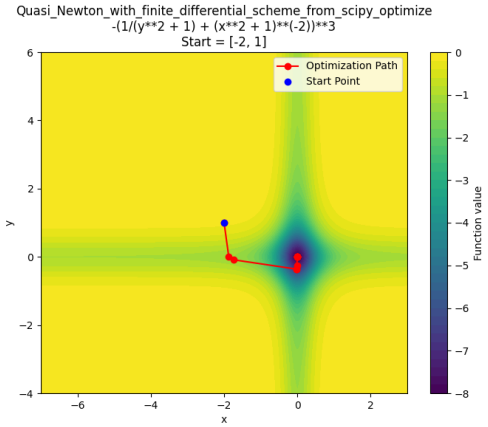

Рис. 10: График 11

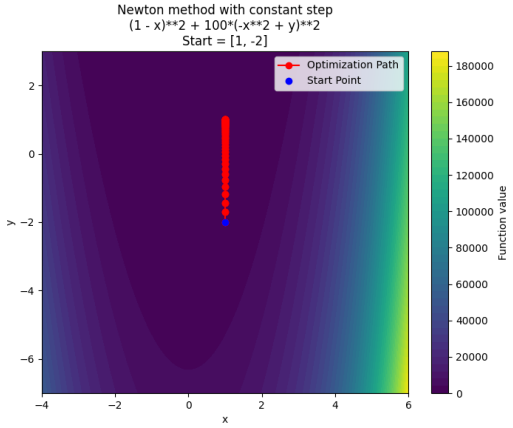

Рис. 11: График 12

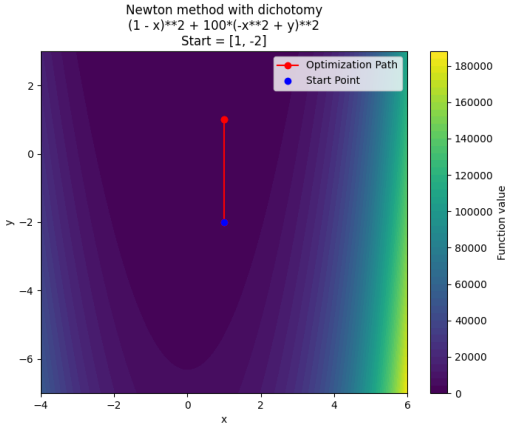

Рис. 12: График 13

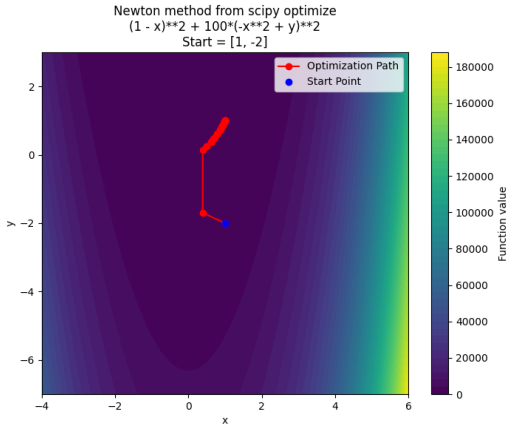

Рис. 13: График 14

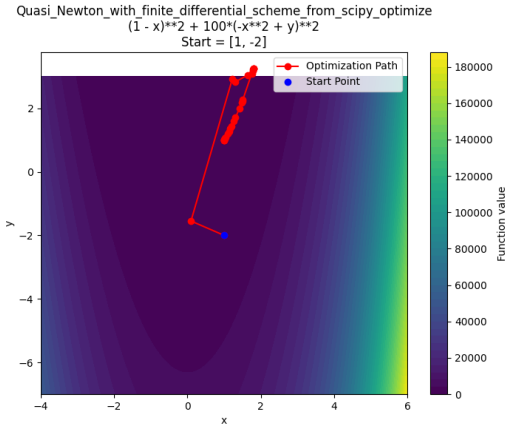

Рис. 14: График 15

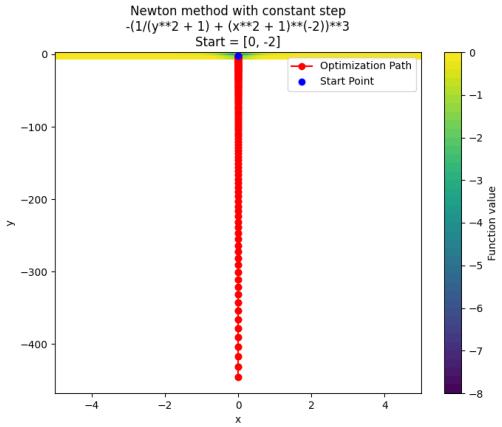

Рис. 15: График 16

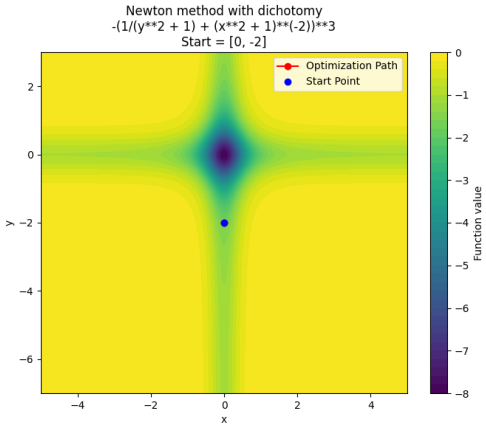

Рис. 16: График 17

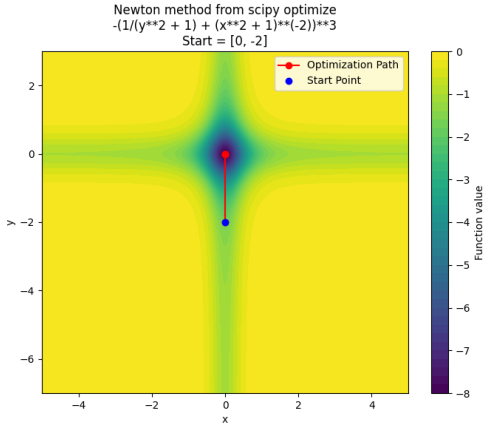

Рис. 17: График 18

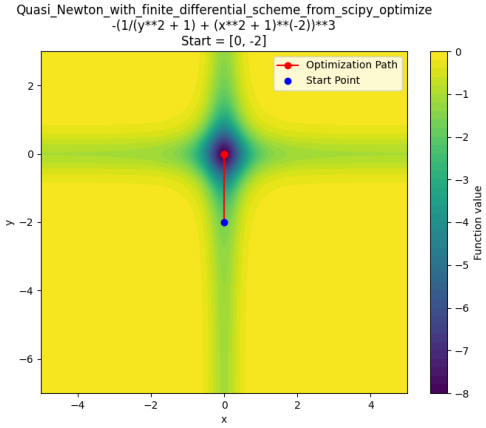

Рис. 18: График 19

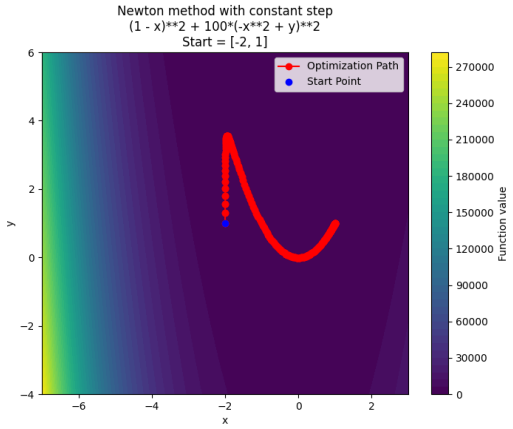

Рис. 19: График 20

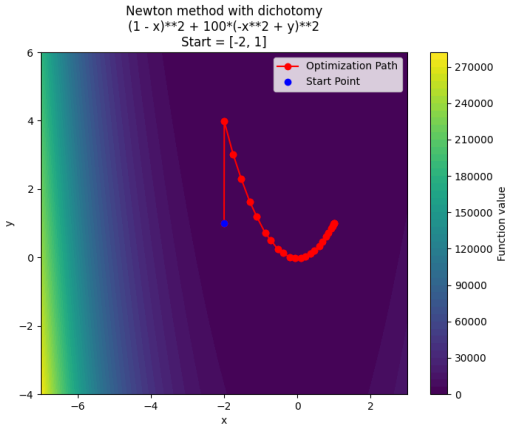

Рис. 20: График 21

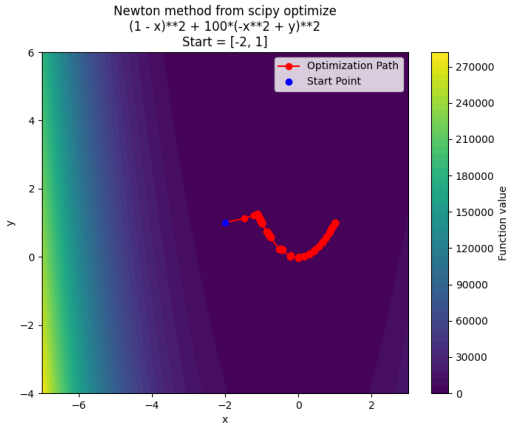

Рис. 21: График 22

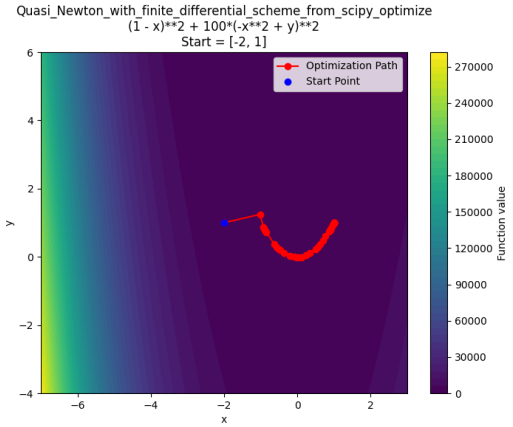

Рис. 22: График 23

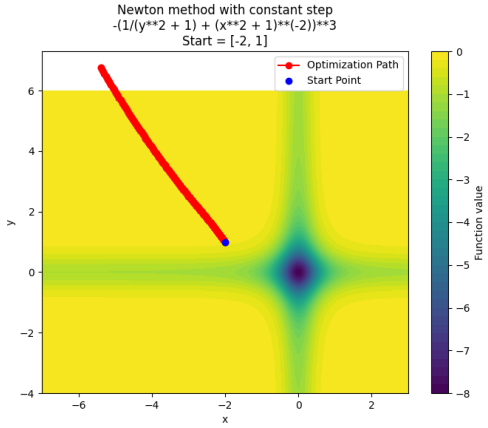

Рис. 23: График 24

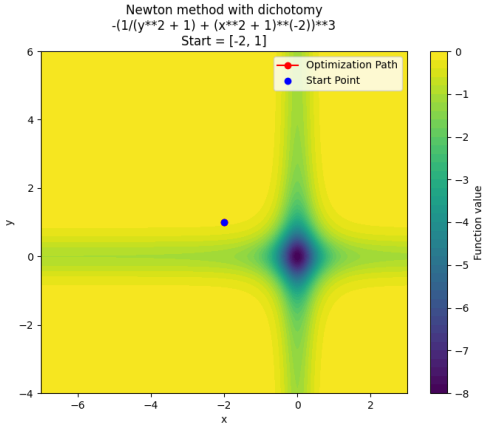

Рис. 24: График 25

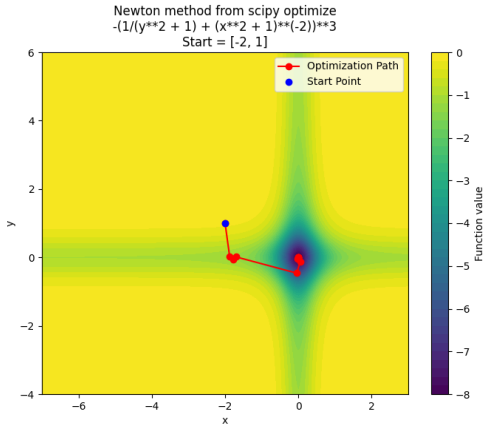

Рис. 25: График 26

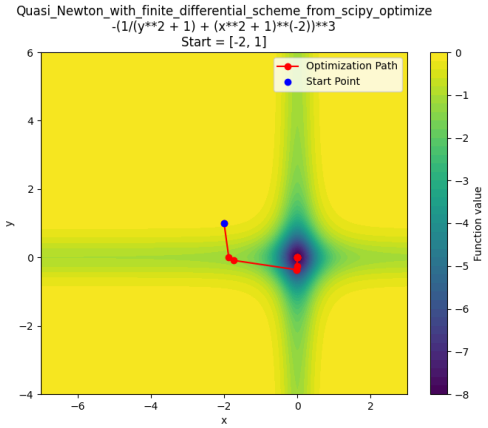

Рис. 26: График 27

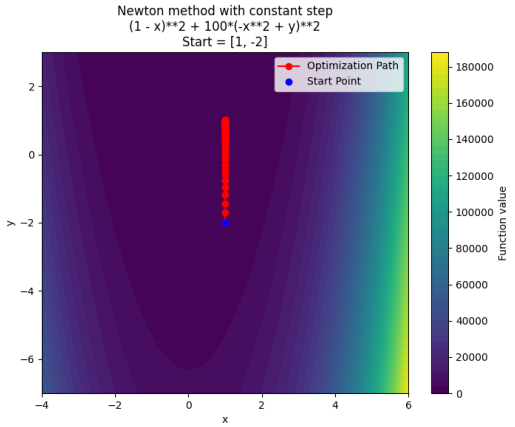

Рис. 27: График 28

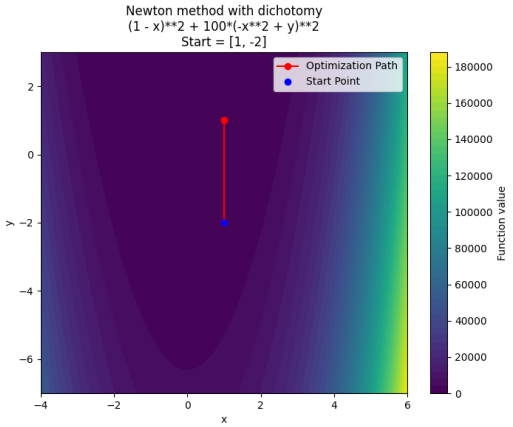

Рис. 28: График 29

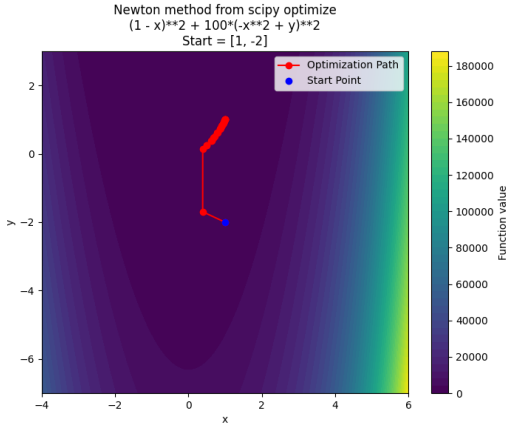

Рис. 29: График 30

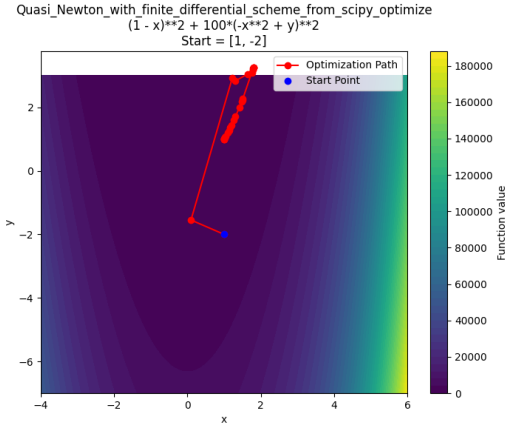

Рис. 30: График 31

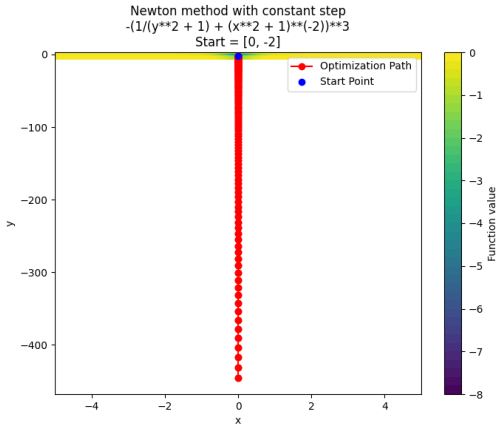

Рис. 31: График 32

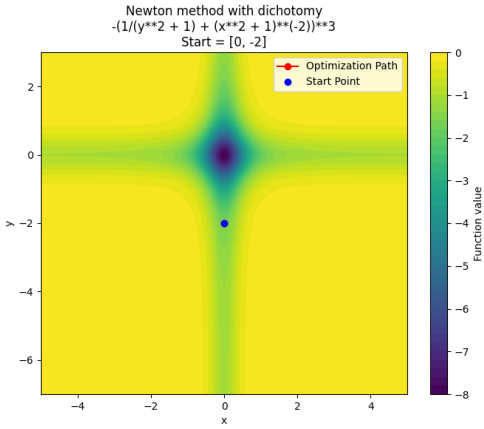

Рис. 32: График 33

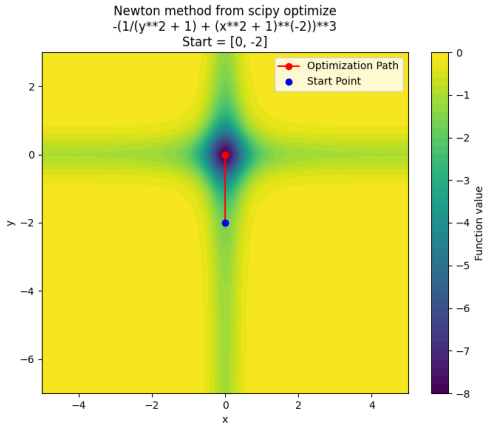

Рис. 33: График 34

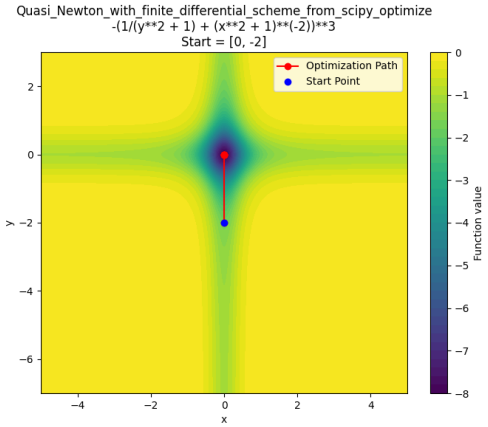

Рис. 34: График 35

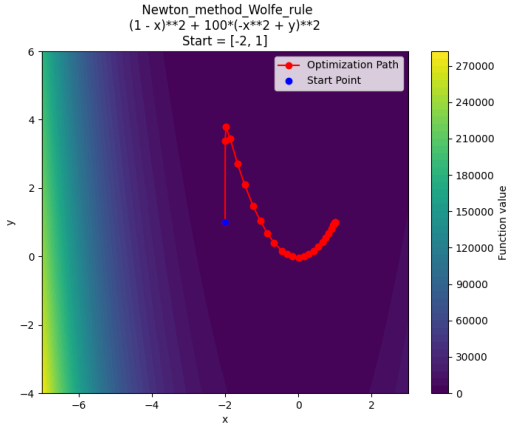

Рис. 35: График 36

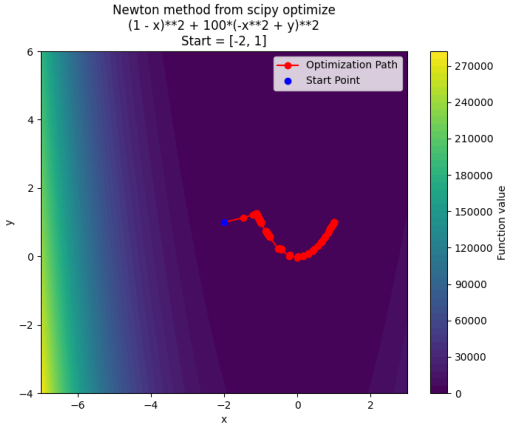

Рис. 36: График 37

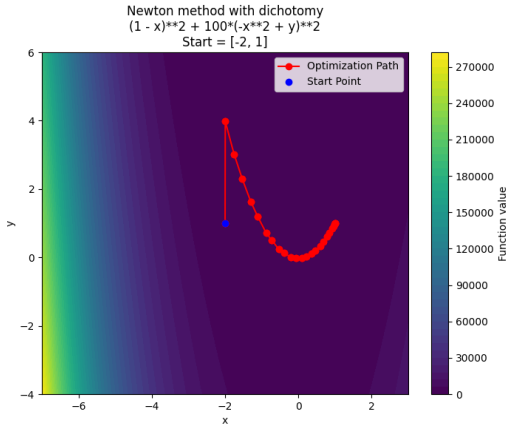

Рис. 37: График 38

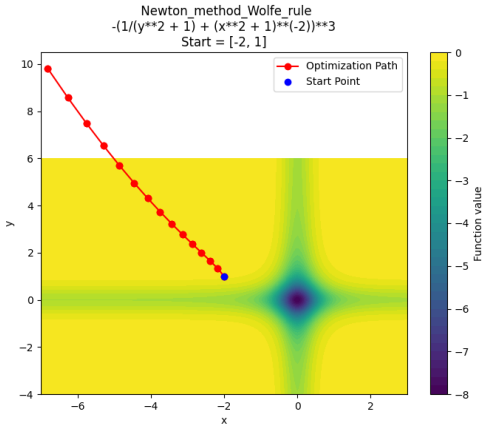

Рис. 38: График 39

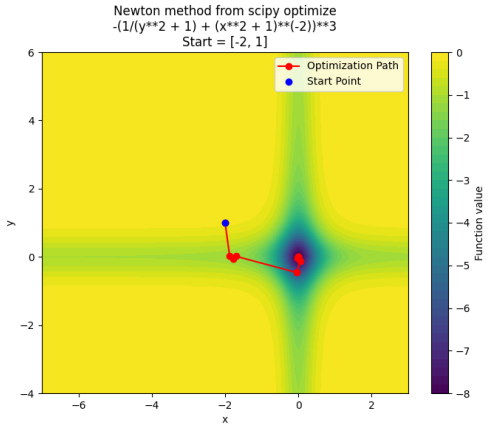

Рис. 39: График 40

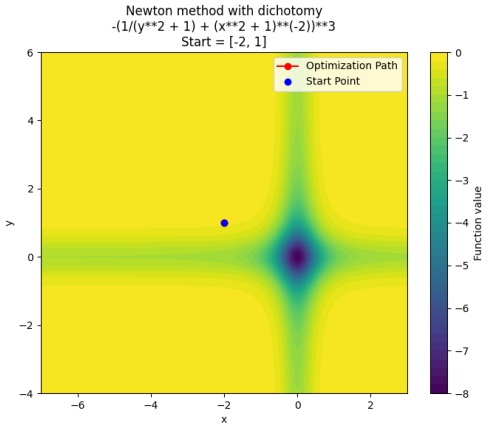

Рис. 40: График 41

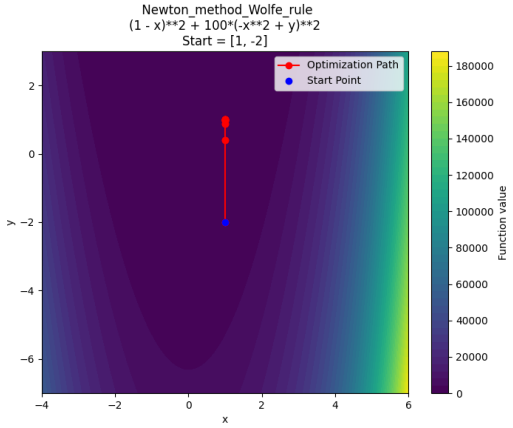

Рис. 41: График 42

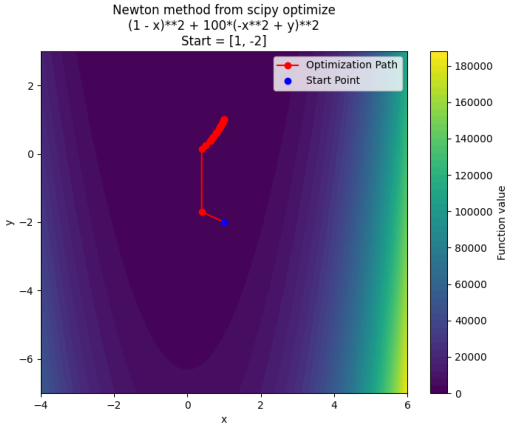

Рис. 42: График 43

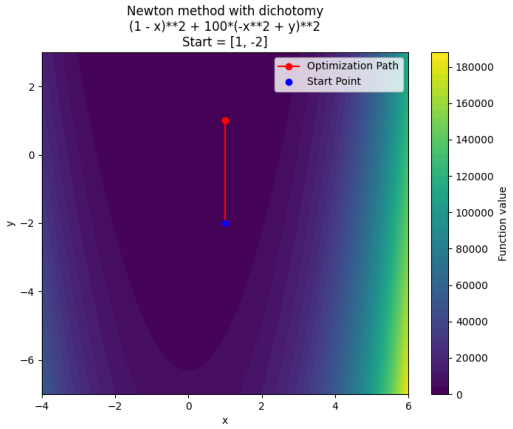

Рис. 43: График 44

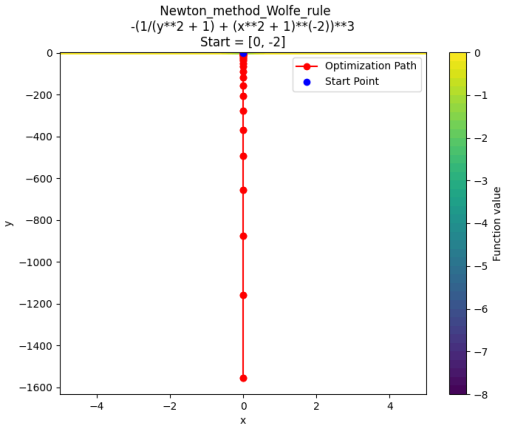

Рис. 44: График 45

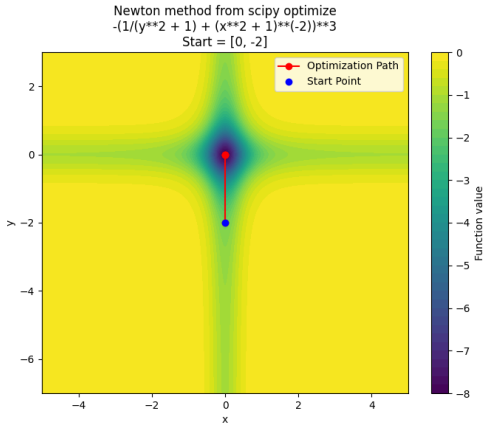

Рис. 45: График 46

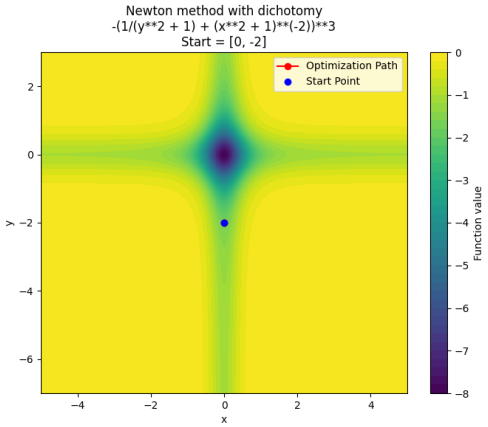

Рис. 46: График 47

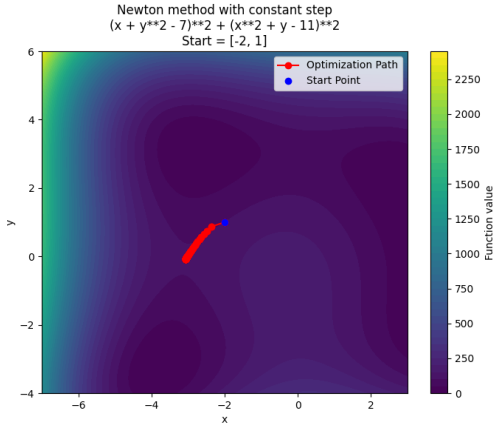

Рис. 47: График 48

Дополнительное задание 2

Шаги исследования

Проведите исследование сложных случаев:

1. найдите функцию, для которой метод Ньютона не работает для некоторой начальной точки. Попробуйте сначала использовать градиентный спуск в таком случае.

2. для функции со многими точками локального минимума найдите пример, где разные методы, из одной начальной точки, сходятся к разным точкам минимума.

Выводы

Метод Ньютона с константным шагом показывает различное поведение в зависимости от величины шага: При большом шаге метод часто пропускает ближайшие локальные минимумы, достигая других, потенциально менее очевидных минимальных точек. Это может быть полезным для задач с множественными локальными минимумами, но также может замедлить процесс сходимости. При меньшем шаге

метод сходится к тем же минимумам, что и библиотечные реализации. Метод Ньютона с одномерным поиском (дихотомия) обеспечивает более точное и регулируемое настраивание шага, что позволяет алгоритму точно и последовательно достигать ближайших минимумов, совпадающих с минимумами, найденными в библиотечных реализациях. Сравнение с методом Newton-CG из scipy.optimize: Метод Newton-CG показывает лучшую точность и большую скорость сходимости по сравнению с нашей реализацией метода Ньютона, что делает его предпочтительным выбором. Общее сравнение методов: Квазиньютоновские методы, такие как BFGS, и методы градиентного спуска в целом показывают лучшую производительность по сравнению с методами нулевого порядка, особенно когда производные функций могут быть эффективно вычислены. Методы нулевого порядка могут оказаться более выгодными в условиях, когда производные сложно вычислимы.