

Отчет по Лабораторной работе #2

Методы первого и высших порядков

Авторы:

Ивченков Дмитрий, М3234

Тюленев Вадим, М3234

Веселкова Варвара, М3234

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 останова.
- **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 останова L-BFGS-B.

Используемый нами квазиньютоновский метод - Алгоритм Бroyдена — 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:

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

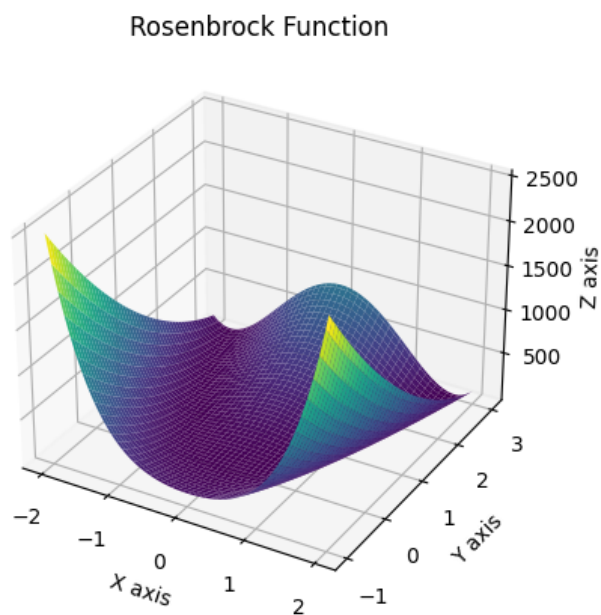


Рис. 1: 3D график функции

2. Функция кубических квадратов:

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

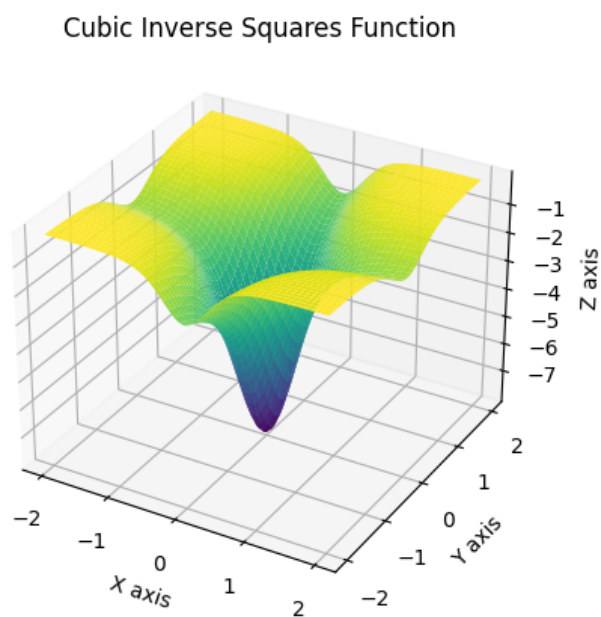


Рис. 2: 3D график функции

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.

```
1  -----MAIN__TASK-----
2  START = [-2, 1]
3  STEP = 0.1
4  EPS = 1e-06
5  Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2
   + y)**2
6  ++
7  FINAL          [0.997643770252781, 0.995252916921813]
8  RESULT          0.00000571322491724373
9  ITERATION       351
10 GRADIENT EVAL COUNT 351
11 FUNCTION EVAL COUNT 702.0
12 ++
13 Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2
   + y)**2
14 ++
15 FINAL          [0.999689378372377, 0.999376146822458]
16 RESULT          9.72182600226012E-8
17 ITERATION       49
18 GRADIENT EVAL COUNT 49
19 FUNCTION EVAL COUNT 2058.0
20 ++
21 Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
   **2
22 ++
23 FINAL          [0.999999856836437, 0.999999713094231]
24 RESULT          2.052929092387938e-14
25 ITERATION       95
26 GRADIENT EVAL COUNT 110
27 FUNCTION EVAL COUNT 305
28 ++
29 Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
   **2 + y)**2
30 ++
31 FINAL          [0.9999955481911366, 0.9999910883821501]
32 RESULT          1.9825034102903414e-11
33 ITERATION       35
34 GRADIENT EVAL COUNT 45
35 FUNCTION EVAL COUNT 135
36 ++
37 -----
38 -----
```

```

39 START = [-2, 1]
40 STEP = 0.1
41 EPS = 1e-06
42 Running method Constant Step Newton Method for  $-(1/(y^2 + 1) + (x^2 + 1)^{-2})^3$ 
43 ++
44 FINAL [-5.39477083034862, 6.76208024232295]
45 RESULT -0.0000113981676522378
46 ITERATION 217
47 GRADIENT EVAL COUNT 217
48 FUNCTION EVAL COUNT 434.0
49 ++
50 Running method Dichotomy Step Newton Method for  $-(1/(y^2 + 1) + (x^2 + 1)^{-2})^3$ 
51 ++
52 FINAL [-2, 1]
53 RESULT -0.157464000000000002
54 ITERATION 1
55 GRADIENT EVAL COUNT 1
56 FUNCTION EVAL COUNT 42.0
57 ++
58 Running method Scipy Newton-CG Method for  $-(1/(y^2 + 1) + (x^2 + 1)^{-2})^3$ 
59 ++
60 FINAL [-2.622349691406688e-19, -4.387299794532255e-19]
61 RESULT -8.0
62 ITERATION 11
63 GRADIENT EVAL COUNT 21
64 FUNCTION EVAL COUNT 42
65 ++
66 Running method Scipy BFGS Quasi-Newton Method for  $-(1/(y^2 + 1) + (x^2 + 1)^{-2})^3$ 
67 ++
68 FINAL [-8.347416398916561e-09, -1.1072128563335669e-08]
69 RESULT -7.999999999999997
70 ITERATION 10
71 GRADIENT EVAL COUNT 27
72 FUNCTION EVAL COUNT 81
73 ++
74 -----
75 -----
76 START = [1, -2]
77 STEP = 0.1
78 EPS = 1e-06
79 Running method Constant Step Newton Method for  $(1 - x)^2 + 100*(-x^2 + y)^2$ 
80 ++
81 FINAL [0.999713950853545, 0.999198951381365]
82 RESULT 0.00000532739668015190
83 ITERATION 181
84 GRADIENT EVAL COUNT 181
85 FUNCTION EVAL COUNT 362.0
86 ++
87 Running method Dichotomy Step Newton Method for  $(1 - x)^2 + 100*(-x^2 + y)^2$ 
88 ++
89 FINAL [0.999999000937193, 0.999996071771669]
90 RESULT 3.73528161330276E-10

```

```

91 ITERATION          3
92 GRADIENT EVAL COUNT 3
93 FUNCTION EVAL COUNT 126.0
94 ++
95 Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
  **2
96 ++
97 FINAL              [0.9999999896813918, 0.9999999793217728]
98 RESULT              1.0664186496686661e-16
99 ITERATION          28
100 GRADIENT EVAL COUNT 36
101 FUNCTION EVAL COUNT 98
102 ++
103 Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
  **2 + y)**2
104 ++
105 FINAL              [0.9999978441740618, 0.9999955856764816]
106 RESULT              5.701827520087579e-12
107 ITERATION          27
108 GRADIENT EVAL COUNT 75
109 FUNCTION EVAL COUNT 237
110 ++
111 -----
112 -----
113 START = [0, -2]
114 STEP = 0.1
115 EPS = 1e-06
116 Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
  1)**(-2))**3
117 ++
118 FINAL              [-5.00086371455102E-7, -445.374451038327]
119 RESULT              -1.00001512413731
120 ITERATION          329
121 GRADIENT EVAL COUNT 329
122 FUNCTION EVAL COUNT 658.0
123 ++
124 Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2
  + 1)**(-2))**3
125 ++
126 FINAL              [0, -2]
127 RESULT              -1.7279999999999998
128 ITERATION          1
129 GRADIENT EVAL COUNT 1
130 FUNCTION EVAL COUNT 42.0
131 ++
132 Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)
  **(-2))**3
133 ++
134 FINAL              [0.0, -6.977552487619915e-16]
135 RESULT              -8.0
136 ITERATION          3
137 GRADIENT EVAL COUNT 8
138 FUNCTION EVAL COUNT 11
139 ++
140 Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x
  **2 + 1)**(-2))**3
141 ++
142 FINAL              [-4.6462954148138415e-08, -9.003681359057637e-08]

```



```

143 RESULT -7.9999999999999848
144 ITERATION 5
145 GRADIENT EVAL COUNT 11
146 FUNCTION EVAL COUNT 33
147 ++
148 -----
149 -----
150 -----_MAIN_SUBTASK-----
151 -----
152 START = [-2, 1]
153 STEP = 0.1
154 EPS = 1e-06
155 Running method Constant Step Newton Method for (1 - x)**2 + 100*(-x**2
+ y)**2
156 ++
157 FINAL [0.997643770252781, 0.995252916921813]
158 RESULT 0.00000571322491724373
159 ITERATION 351
160 GRADIENT EVAL COUNT 351
161 FUNCTION EVAL COUNT 702.0
162 ++
163 Running method Dichotomy Step Newton Method for (1 - x)**2 + 100*(-x**2
+ y)**2
164 ++
165 FINAL [0.999689378372377, 0.999376146822458]
166 RESULT 9.72182600226012E-8
167 ITERATION 49
168 GRADIENT EVAL COUNT 49
169 FUNCTION EVAL COUNT 2058.0
170 ++
171 Running method Scipy Newton-CG Method for (1 - x)**2 + 100*(-x**2 + y)
**2
172 ++
173 FINAL [0.999999856836437, 0.999999713094231]
174 RESULT 2.052929092387938e-14
175 ITERATION 95
176 GRADIENT EVAL COUNT 110
177 FUNCTION EVAL COUNT 305
178 ++
179 Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
**2 + y)**2
180 ++
181 FINAL [0.9999955481911366, 0.9999910883821501]
182 RESULT 1.9825034102903414e-11
183 ITERATION 35
184 GRADIENT EVAL COUNT 45
185 FUNCTION EVAL COUNT 135
186 ++
187 -----
188 -----
189 START = [-2, 1]
190 STEP = 0.1
191 EPS = 1e-06
192 Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
1)**(-2))**3
193 ++
194 FINAL [-5.39477083034862, 6.76208024232295]
195 RESULT -0.0000113981676522378

```

```

196 ITERATION                217
197 GRADIENT EVAL COUNT     217
198 FUNCTION EVAL COUNT     434.0
199 ++
200 Running method Dichotomy Step Newton Method for  $-(1/(y^{**2} + 1) + (x^{**2} + 1)**(-2))^{**3}$ 
201 ++
202 FINAL                     [-2, 1]
203 RESULT                    -0.157464000000000002
204 ITERATION                 1
205 GRADIENT EVAL COUNT       1
206 FUNCTION EVAL COUNT       42.0
207 ++
208 Running method Scipy Newton-CG Method for  $-(1/(y^{**2} + 1) + (x^{**2} + 1)**(-2))^{**3}$ 
209 ++
210 FINAL                     [-2.622349691406688e-19, -4.387299794532255e-19]
211 RESULT                    -8.0
212 ITERATION                 11
213 GRADIENT EVAL COUNT       21
214 FUNCTION EVAL COUNT       42
215 ++
216 Running method Scipy BFGS Quasi-Newton Method for  $-(1/(y^{**2} + 1) + (x^{**2} + 1)**(-2))^{**3}$ 
217 ++
218 FINAL                     [-8.347416398916561e-09, -1.1072128563335669e-08]
219 RESULT                    -7.999999999999997
220 ITERATION                 10
221 GRADIENT EVAL COUNT       27
222 FUNCTION EVAL COUNT       81
223 ++
224 -----
225 -----
226 START = [1, -2]
227 STEP = 0.1
228 EPS = 1e-06
229 Running method Constant Step Newton Method for  $(1 - x)^{**2} + 100*(-x^{**2} + y)^{**2}$ 
230 ++
231 FINAL                     [0.999713950853545, 0.999198951381365]
232 RESULT                    0.00000532739668015190
233 ITERATION                 181
234 GRADIENT EVAL COUNT       181
235 FUNCTION EVAL COUNT       362.0
236 ++
237 Running method Dichotomy Step Newton Method for  $(1 - x)^{**2} + 100*(-x^{**2} + y)^{**2}$ 
238 ++
239 FINAL                     [0.999999000937193, 0.999996071771669]
240 RESULT                    3.73528161330276E-10
241 ITERATION                 3
242 GRADIENT EVAL COUNT       3
243 FUNCTION EVAL COUNT       126.0
244 ++
245 Running method Scipy Newton-CG Method for  $(1 - x)^{**2} + 100*(-x^{**2} + y)^{**2}$ 
246 ++
247 FINAL                     [0.9999999896813918, 0.9999999793217728]

```

```

248 RESULT                1.0664186496686661e-16
249 ITERATION              28
250 GRADIENT EVAL COUNT    36
251 FUNCTION EVAL COUNT    98
252 ++
253 Running method Scipy BFGS Quasi-Newton Method for (1 - x)**2 + 100*(-x
    **2 + y)**2
254 ++
255 FINAL                  [0.9999978441740618, 0.9999955856764816]
256 RESULT                5.701827520087579e-12
257 ITERATION              27
258 GRADIENT EVAL COUNT    75
259 FUNCTION EVAL COUNT    237
260 ++
261 -----
262 -----
263 START = [0, -2]
264 STEP = 0.1
265 EPS = 1e-06
266 Running method Constant Step Newton Method for -(1/(y**2 + 1) + (x**2 +
    1)**(-2))**3
267 ++
268 FINAL                  [-5.00086371455102E-7, -445.374451038327]
269 RESULT                -1.00001512413731
270 ITERATION              329
271 GRADIENT EVAL COUNT    329
272 FUNCTION EVAL COUNT    658.0
273 ++
274 Running method Dichotomy Step Newton Method for -(1/(y**2 + 1) + (x**2
    + 1)**(-2))**3
275 ++
276 FINAL                  [0, -2]
277 RESULT                -1.7279999999999998
278 ITERATION              1
279 GRADIENT EVAL COUNT    1
280 FUNCTION EVAL COUNT    42.0
281 ++
282 Running method Scipy Newton-CG Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
283 ++
284 FINAL                  [0.0, -6.977552487619915e-16]
285 RESULT                -8.0
286 ITERATION              3
287 GRADIENT EVAL COUNT    8
288 FUNCTION EVAL COUNT    11
289 ++
290 Running method Scipy BFGS Quasi-Newton Method for -(1/(y**2 + 1) + (x
    **2 + 1)**(-2))**3
291 ++
292 FINAL                  [-4.6462954148138415e-08, -9.003681359057637e-08]
293 RESULT                -7.999999999999848
294 ITERATION              5
295 GRADIENT EVAL COUNT    11
296 FUNCTION EVAL COUNT    33
297 ++
298 -----
299 -----
300 -----_FIRST_TASK-----

```

```

301 -----
302 Starting point: [-2, 1]
303 Const step: 0.1
304 Required accuracy : 1e-06
305 Running method Wolfe Rule Newton Method for (1 - x)**2 + 100*(-x**2 + y
    )**2
306 ++
307 FINAL [0.999675298812091, 0.999348110324489]
308 RESULT 1.06103086602393E-7
309 ITERATION 53
310 GRADIENT EVAL COUNT 53
311 FUNCTION EVAL COUNT 477.0
312 ++
313 Running method Scipy Newton-CG Optimizer for (1 - x)**2 + 100*(-x**2 +
    y)**2
314 ++
315 FINAL [0.999999856836437, 0.999999713094231]
316 RESULT 2.052929092387938e-14
317 ITERATION 95
318 GRADIENT EVAL COUNT 110
319 FUNCTION EVAL COUNT 305
320 ++
321 Running method Dichotomy Newton Method for (1 - x)**2 + 100*(-x**2 + y)
    **2
322 ++
323 FINAL [0.999689378372377, 0.999376146822458]
324 RESULT 9.72182600226012E-8
325 ITERATION 49
326 GRADIENT EVAL COUNT 49
327 FUNCTION EVAL COUNT 2058.0
328 ++
329 -----
330 -----
331 Starting point: [-2, 1]
332 Const step: 0.1
333 Required accuracy : 1e-06
334 Running method Wolfe Rule Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
335 ++
336 FINAL [-6.81211489594668, 9.80729965789102]
337 RESULT -0.00000123705033079946
338 ITERATION 29
339 GRADIENT EVAL COUNT 29
340 FUNCTION EVAL COUNT 261
341 ++
342 Running method Scipy Newton-CG Optimizer for -(1/(y**2 + 1) + (x**2 +
    1)**(-2))**3
343 ++
344 FINAL [-2.622349691406688e-19, -4.387299794532255e-19]
345 RESULT -8.0
346 ITERATION 11
347 GRADIENT EVAL COUNT 21
348 FUNCTION EVAL COUNT 42
349 ++
350 Running method Dichotomy Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
    **(-2))**3
351 ++
352 FINAL [-2, 1]

```

```

353 RESULT -0.157464000000000002
354 ITERATION 1
355 GRADIENT EVAL COUNT 1
356 FUNCTION EVAL COUNT 42.0
357 ++
358 -----
359 -----
360 Starting point: [1, -2]
361 Const step: 0.1
362 Required accuracy : 1e-06
363 Running method Wolfe Rule Newton Method for (1 - x)**2 + 100*(-x**2 + y
) **2
364 ++
365 FINAL [0.999710628356721, 0.999382436739065]
366 RESULT 2.35085815646672E-7
367 ITERATION 15
368 GRADIENT EVAL COUNT 15
369 FUNCTION EVAL COUNT 135.0
370 ++
371 Running method Scipy Newton-CG Optimizer for (1 - x)**2 + 100*(-x**2 +
y) **2
372 ++
373 FINAL [0.9999999896813918, 0.9999999793217728]
374 RESULT 1.0664186496686661e-16
375 ITERATION 28
376 GRADIENT EVAL COUNT 36
377 FUNCTION EVAL COUNT 98
378 ++
379 Running method Dichotomy Newton Method for (1 - x)**2 + 100*(-x**2 + y)
**2
380 ++
381 FINAL [0.999999000937193, 0.999996071771669]
382 RESULT 3.73528161330276E-10
383 ITERATION 3
384 GRADIENT EVAL COUNT 3
385 FUNCTION EVAL COUNT 126.0
386 ++
387 -----
388 -----
389 Starting point: [0, -2]
390 Const step: 0.1
391 Required accuracy : 1e-06
392 Running method Wolfe Rule Newton Method for -(1/(y**2 + 1) + (x**2 + 1)
**(-2))**3
393 ++
394 FINAL [-5.00018590452804E-7, -1554.69929871097]
395 RESULT -1.00000124116044
396 ITERATION 47
397 GRADIENT EVAL COUNT 47
398 FUNCTION EVAL COUNT 423
399 ++
400 Running method Scipy Newton-CG Optimizer for -(1/(y**2 + 1) + (x**2 +
1) **(-2))**3
401 ++
402 FINAL [0.0, -6.977552487619915e-16]
403 RESULT -8.0
404 ITERATION 3
405 GRADIENT EVAL COUNT 8

```

```

406 FUNCTION EVAL COUNT 11
407 ++
408 Running method Dichotomy Newton Method for  $-(1/(y^{**2} + 1) + (x^{**2} + 1)
      **(-2))^{**3}$ 
409 ++
410 FINAL [0, -2]
411 RESULT -1.7279999999999998
412 ITERATION 1
413 GRADIENT EVAL COUNT 1
414 FUNCTION EVAL COUNT 42.0
415 ++
416 -----
417 -----
418 -----_SECOND__TASK-----
419 -----
420 START = [0, 1]
421 STEP = 0.01
422 EPS = 1e-06
423 Running method Constant Step Newton Method for  $(x + y^{**2} - 7)^{**2} + (x
      **2 + y - 11)^{**2}$ 
424 ++
425 FINAL [-0.270892976011036, -0.906258976479230]
426 RESULT 181.614134541362
427 ITERATION 1001
428 GRADIENT EVAL COUNT 1001
429 FUNCTION EVAL COUNT 2002.0
430 ++
431 Running method Dichotomy Step Newton Method for  $(x + y^{**2} - 7)^{**2} + (x
      **2 + y - 11)^{**2}$ 
432 ++
433 FINAL [-0.000159133709973960, 0.999204389491602]
434 RESULT 136.036912058947
435 ITERATION 1001
436 GRADIENT EVAL COUNT 1001
437 FUNCTION EVAL COUNT 42042.0
438 ++
439 Running method Scipy Newton-CG Method for  $(x + y^{**2} - 7)^{**2} + (x^{**2} + y
      - 11)^{**2}$ 
440 ++
441 FINAL [3.0000000000148095, 1.99999999998679516]
442 RESULT 7.167977428491776e-19
443 ITERATION 9
444 GRADIENT EVAL COUNT 13
445 FUNCTION EVAL COUNT 32
446 ++
447 Running method Scipy BFGS Quasi-Newton Method for  $(x + y^{**2} - 7)^{**2} + (
      x^{**2} + y - 11)^{**2}$ 
448 ++
449 FINAL [2.9999999993244534, 1.99999999979321275]
450 RESULT 2.0406263010115856e-15
451 ITERATION 13
452 GRADIENT EVAL COUNT 21
453 FUNCTION EVAL COUNT 63
454 ++
455 -----
456 -----

```

Графики

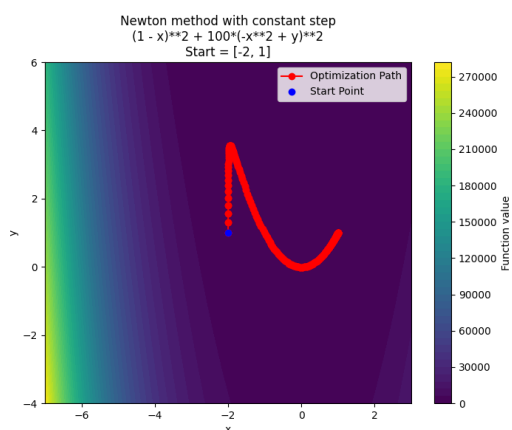


Рис. 3: График 4

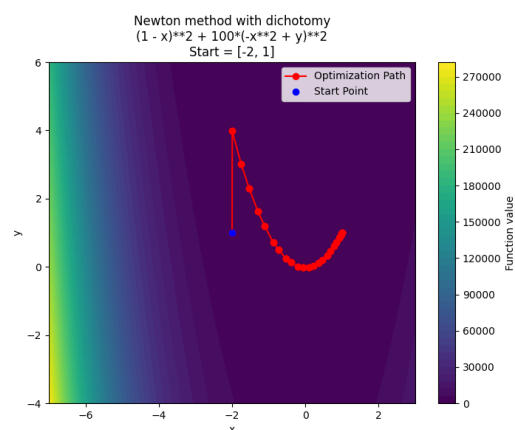


Рис. 4: График 5

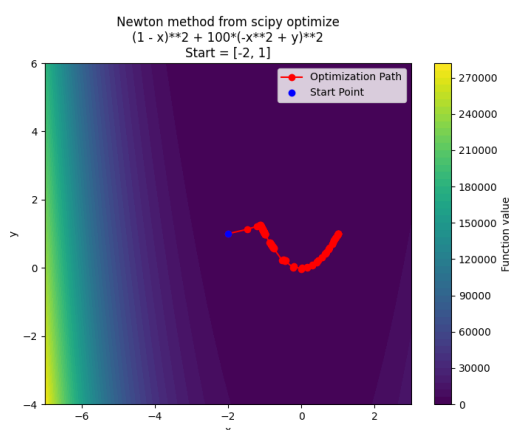


Рис. 5: График 6

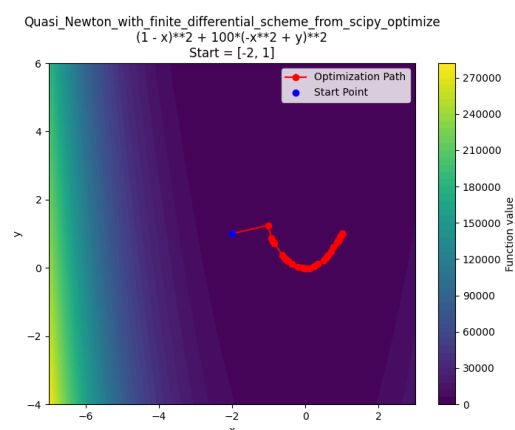


Рис. 6: График 7

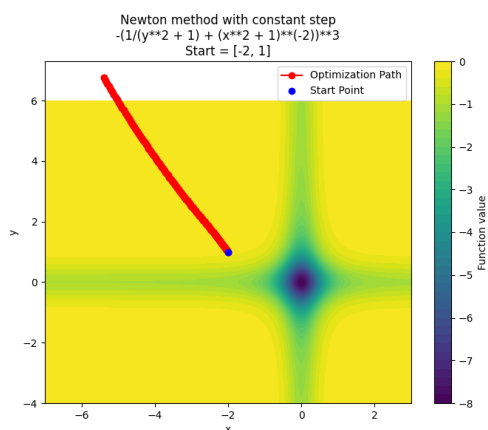


Рис. 7: График 8

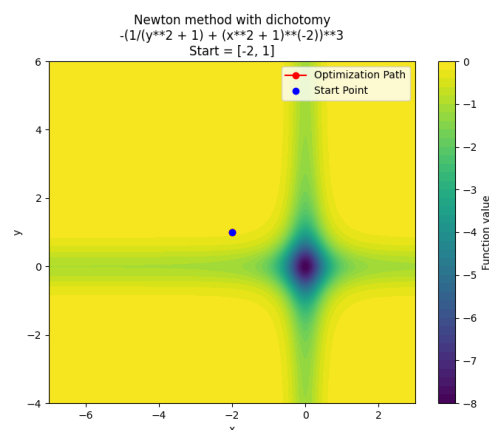


Рис. 8: График 9

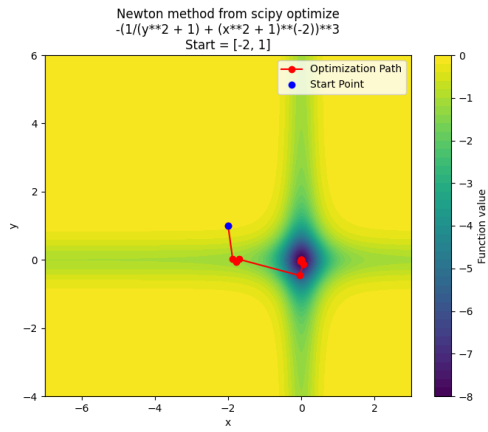


Рис. 9: График 10

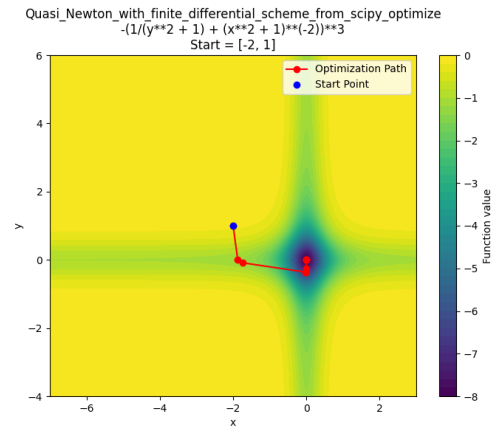


Рис. 10: График 11

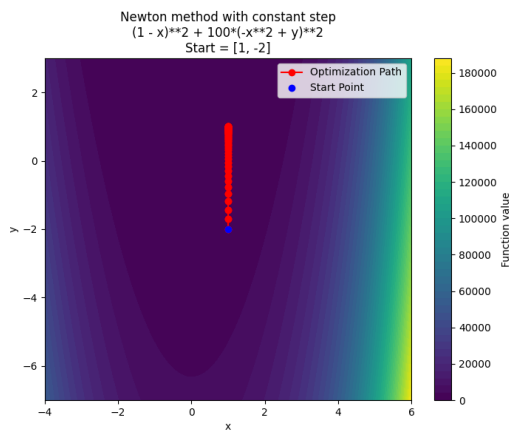


Рис. 11: График 12

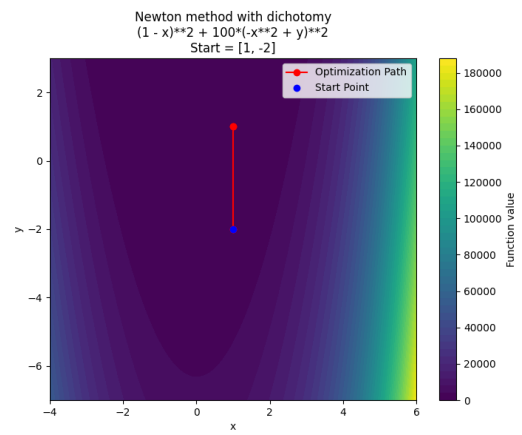


Рис. 12: График 13

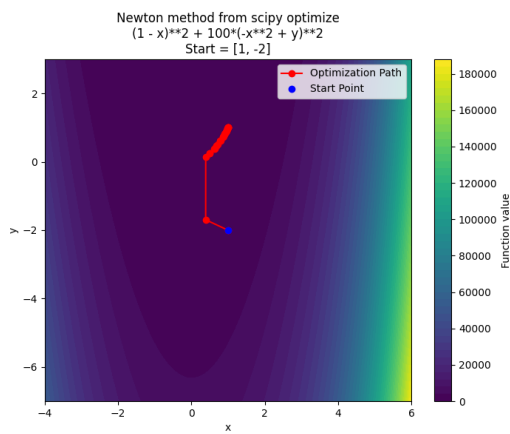


Рис. 13: График 14

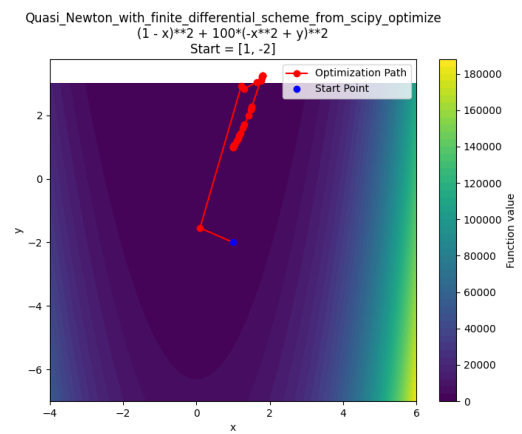


Рис. 14: График 15

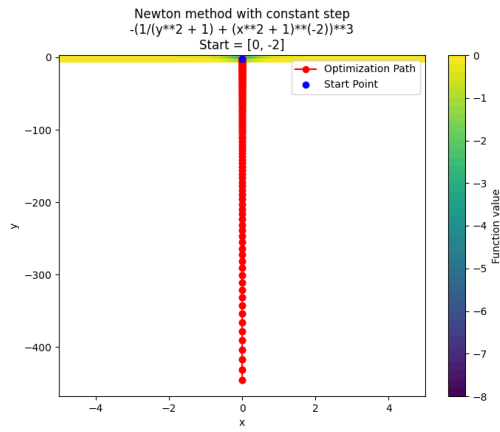


Рис. 15: График 16

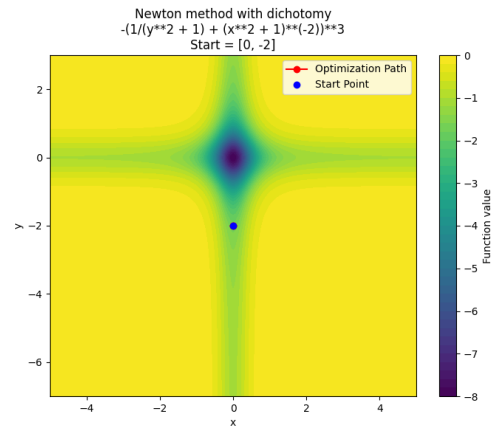


Рис. 16: График 17

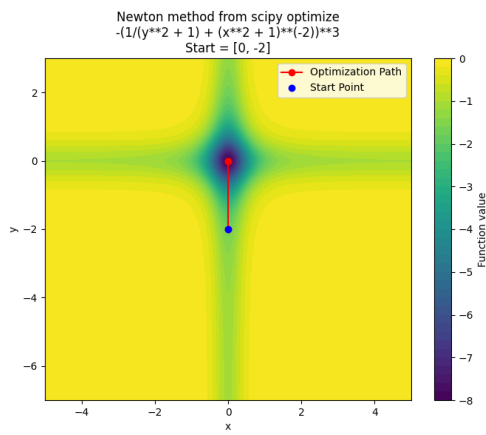


Рис. 17: График 18

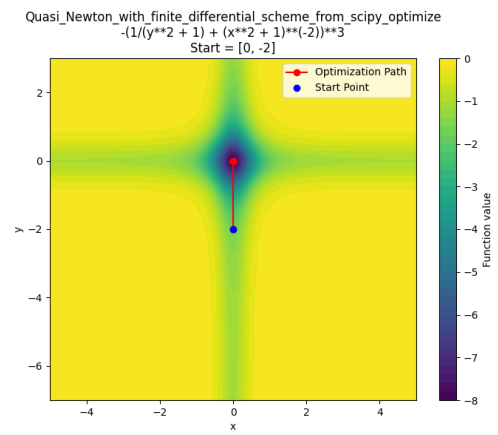


Рис. 18: График 19

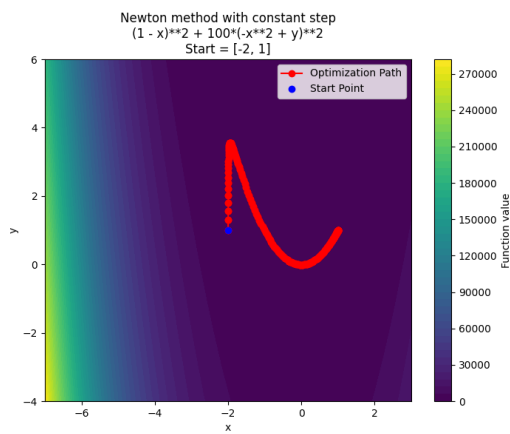


Рис. 19: График 20

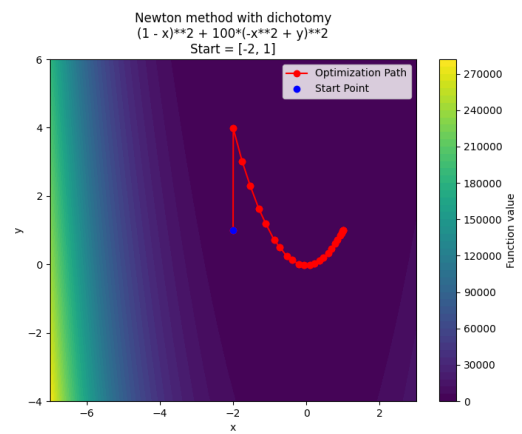


Рис. 20: График 21

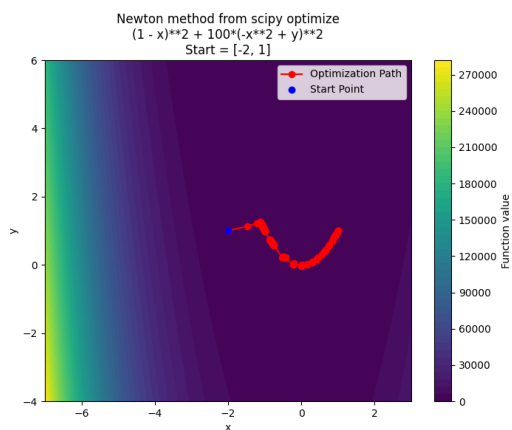


Рис. 21: График 22

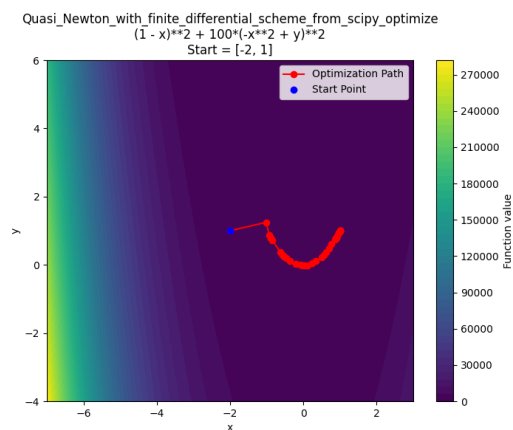


Рис. 22: График 23

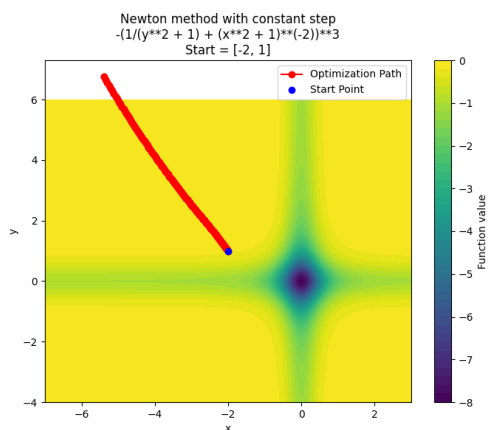


Рис. 23: График 24

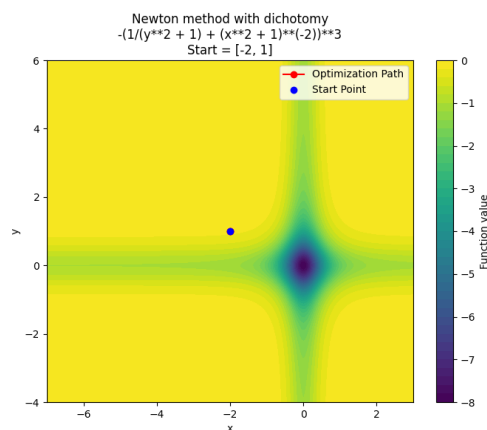


Рис. 24: График 25

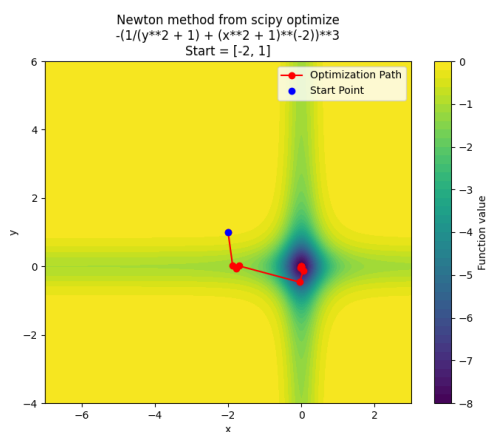


Рис. 25: График 26

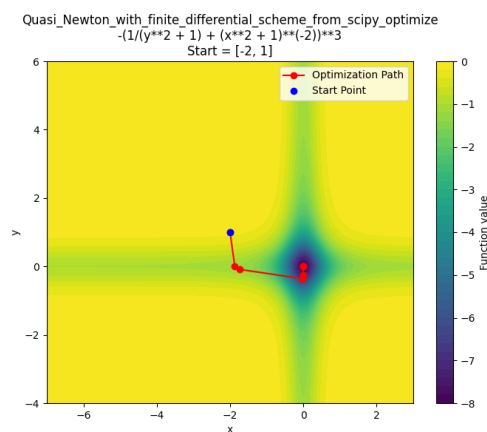


Рис. 26: График 27

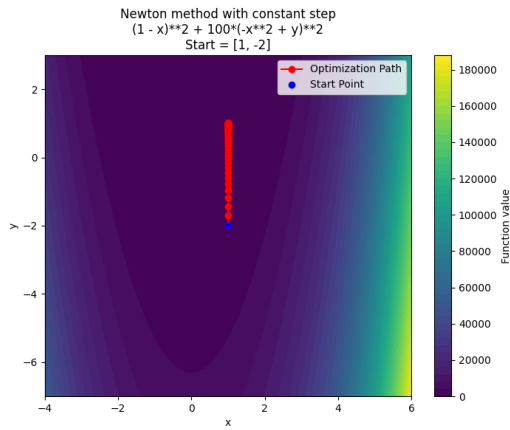


Рис. 27: График 28

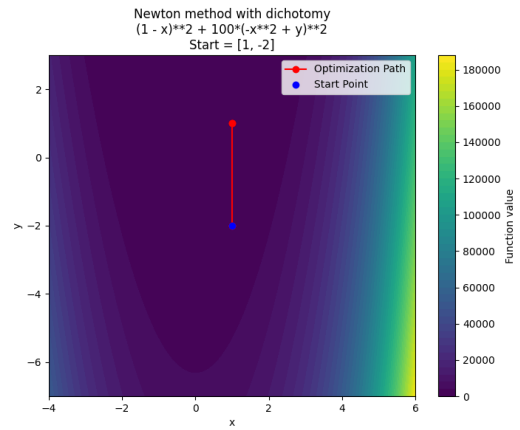


Рис. 28: График 29

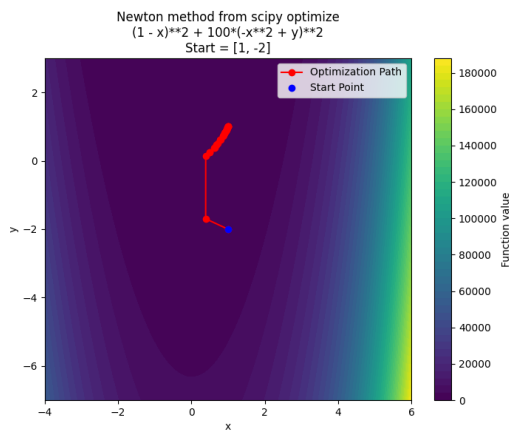


Рис. 29: График 30

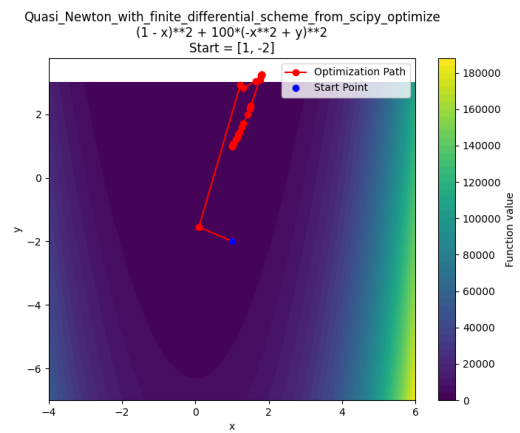


Рис. 30: График 31

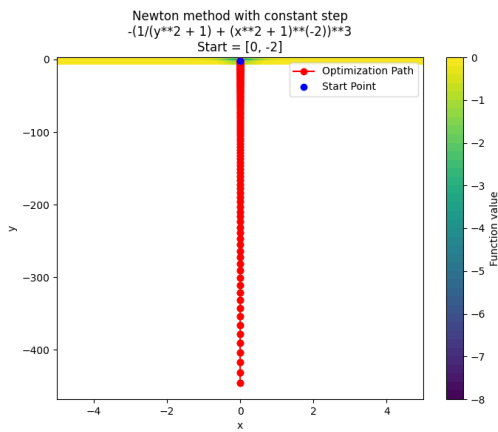


Рис. 31: График 32

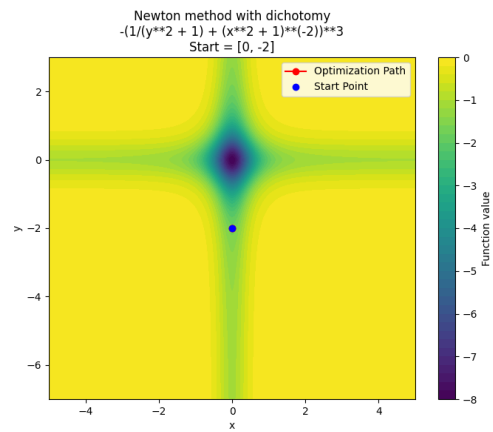


Рис. 32: График 33

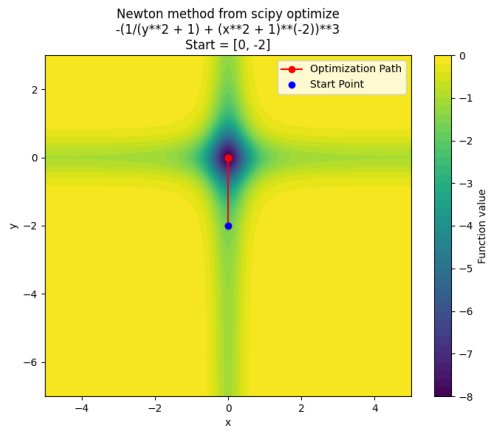


Рис. 33: График 34

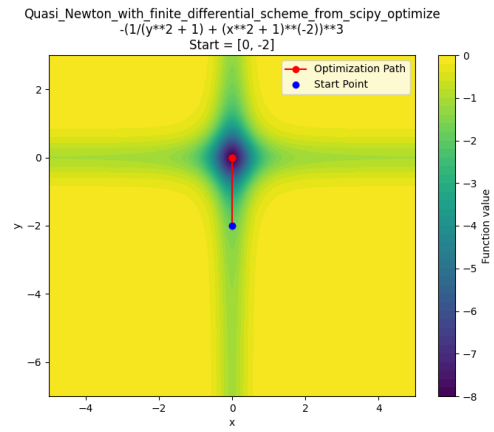


Рис. 34: График 35

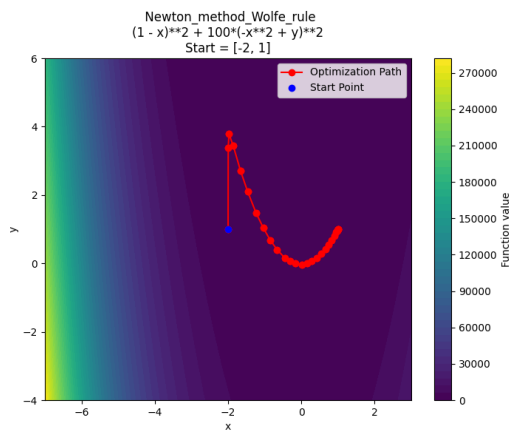


Рис. 35: График 36

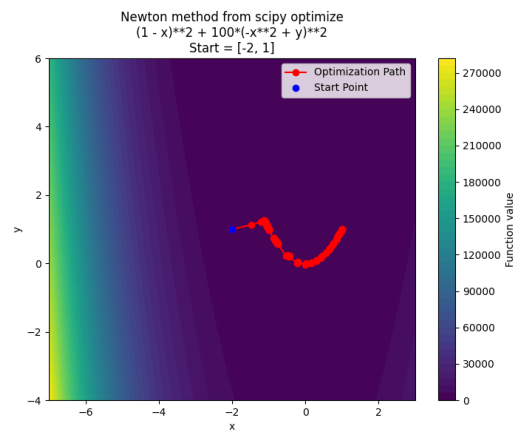


Рис. 36: График 37

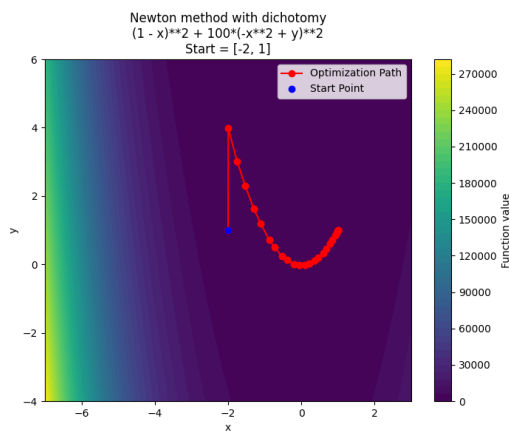


Рис. 37: График 38

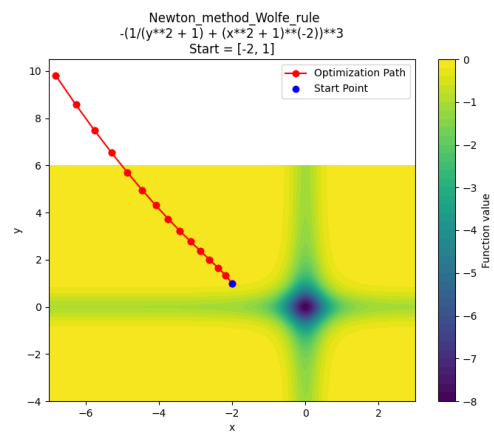


Рис. 38: График 39

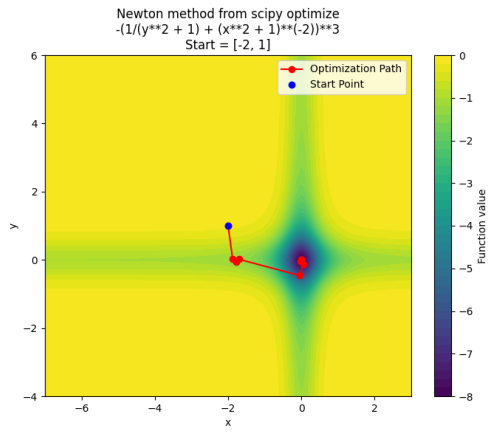


Рис. 39: График 40

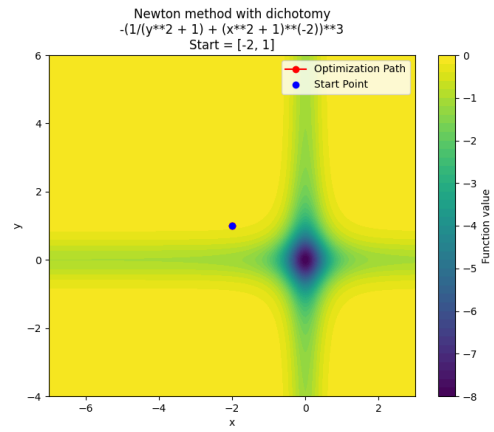


Рис. 40: График 41

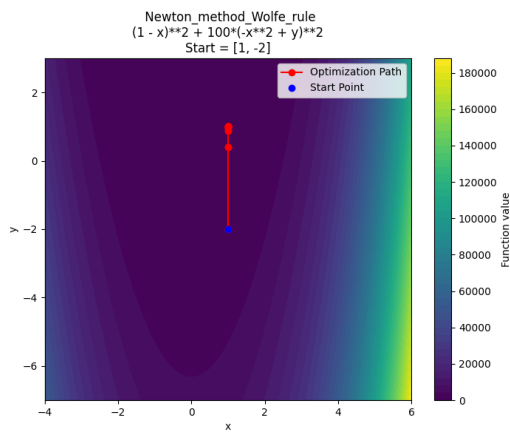


Рис. 41: График 42

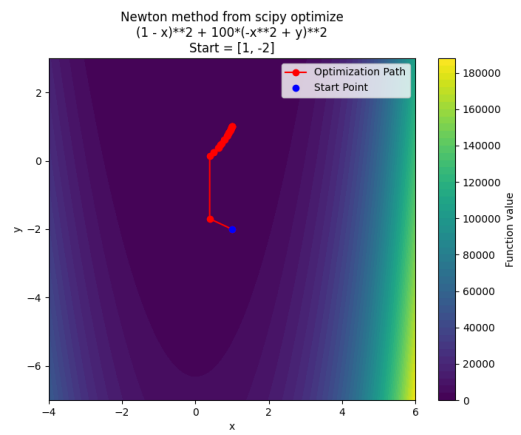


Рис. 42: График 43

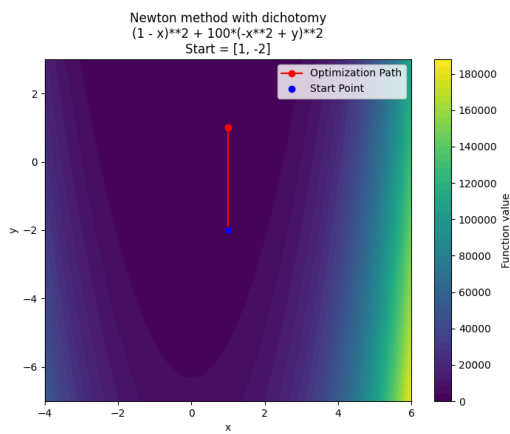


Рис. 43: График 44

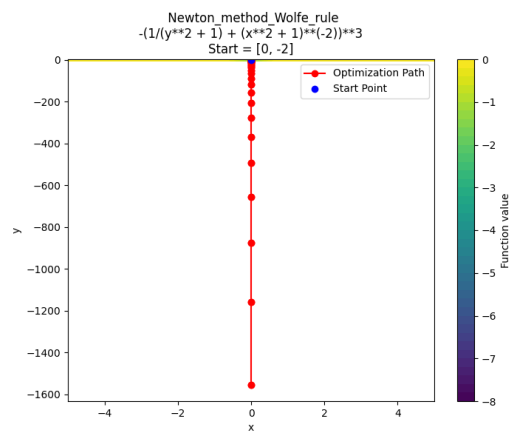


Рис. 44: График 45

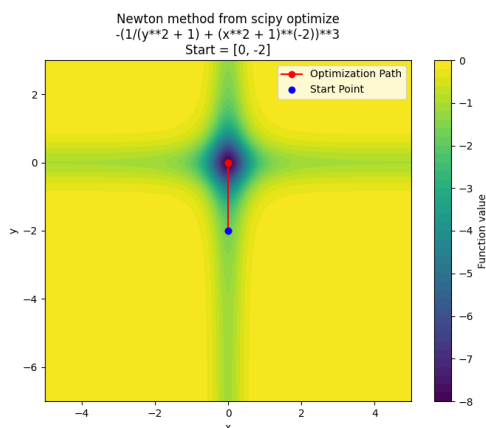


Рис. 45: График 46

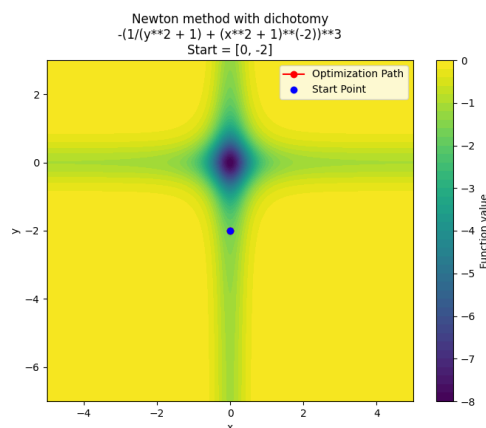


Рис. 46: График 47

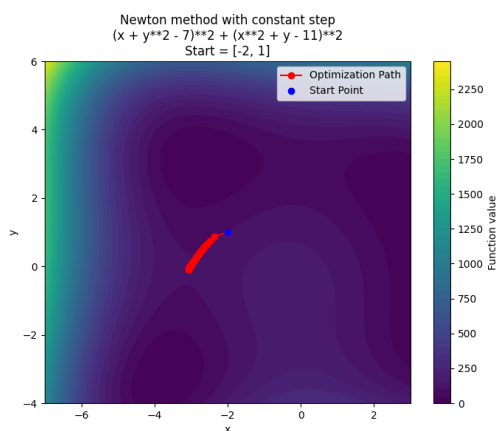


Рис. 47: График 48

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.