In [7]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams["figure.figsize"] = (10, 10)

%load_ext autoreload
%autoreload 2

from core.quadratic_analyzer import *
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Анализ работы градиентного спуска на квадратичных функциях двух переменных

Рассмотрим такие квадратичные фукнции: ax^2^ + bxy + cy^2^ + dx + ey. Можно добавить константу, но она никак не влияет решение текущей задачи, так как это просто добавляет вертикальное смещение, которое никак не влияет на производную, в том числе на численные методы ее вычисления.

Такие функции можно разделить на типы по сечению графика фунцкии по оси значений:

1. Положительно опрделенная форма. В сечении получаются элипсы.
2. Отрицательно опрделенная форма. В сечении получаются гипреболы.
3. a = 0, c = 0. В сечении получаются две прямые.
4. a = 0 или c = 0, то но не оба. В сечении получаются параболы.

Случай положительно определенной формы.

Это наиболее содержательный случай, так как у функции есть глобальный минимум.

Метод постоянного шага.

Сходимость методов с постоянным шагом будет сильно зависеть от выбора шага. Если шаг был выбран слишком большой, то метод разойдется, так как будет ходить вдоль отрезка. На больших значениях будет большая производная, а значит размер реального шага будет увеличиваться, поэтому траектория точек будет ходить вдоль прямой с потенциальным минимумом, но на каждой следующей итерации будет только отдаляться от него. При достаточно маленьком шаге метод будет сходиться, но для этого потребуется достаточно много итераций. Оссобенно заметно, что в точках близких к минимуму сдвиг точки будет невелик, так как длина градиента начинает постепенно уменьшаться. Это создает более плавную траекторию, что может хорошо работать для более сложных функций с большим количеством станционарных точек,

так как траектория не будет "кидаться" по сторонам, но это приводит к малой эффективности.

Методы линейного поиска минимума.

По графикам видно, что все методы линейного поиска минимума работают примерно одинаково. Это следует из общего свойства нахождения минимума вдоль градиента, а не идти в "случайную" точку. Также по графикам видно, что для положительно определенных форм траектория образуется из движения вдоль касательных элипсов, которые образованы сечениями функции по оси значений. Это создает "ломанную" траекторию, которая хорошо себя показывает на положительно определенных формах, так как у них есть ровно 1 станционарная точка, а все остальные точки имеют градиент глобально направленных в одно и тоже место. Для сходимости достаточно проведения всего нескольких итераций. Однако весь класс методов требует, чтобы сужение функции вдоль прямой имело производную, которая ровно 1 раз меняет свой знак. При невыполнении этого условия метод, может найти какую-то станционарную точку. Также графики показывают, что минимум вдоль прямой не обязательно находить точно: при запуске бинарного поиска с ограниченным числом итераций получается сопоставимый результат по числу итераций градиентного спуска.

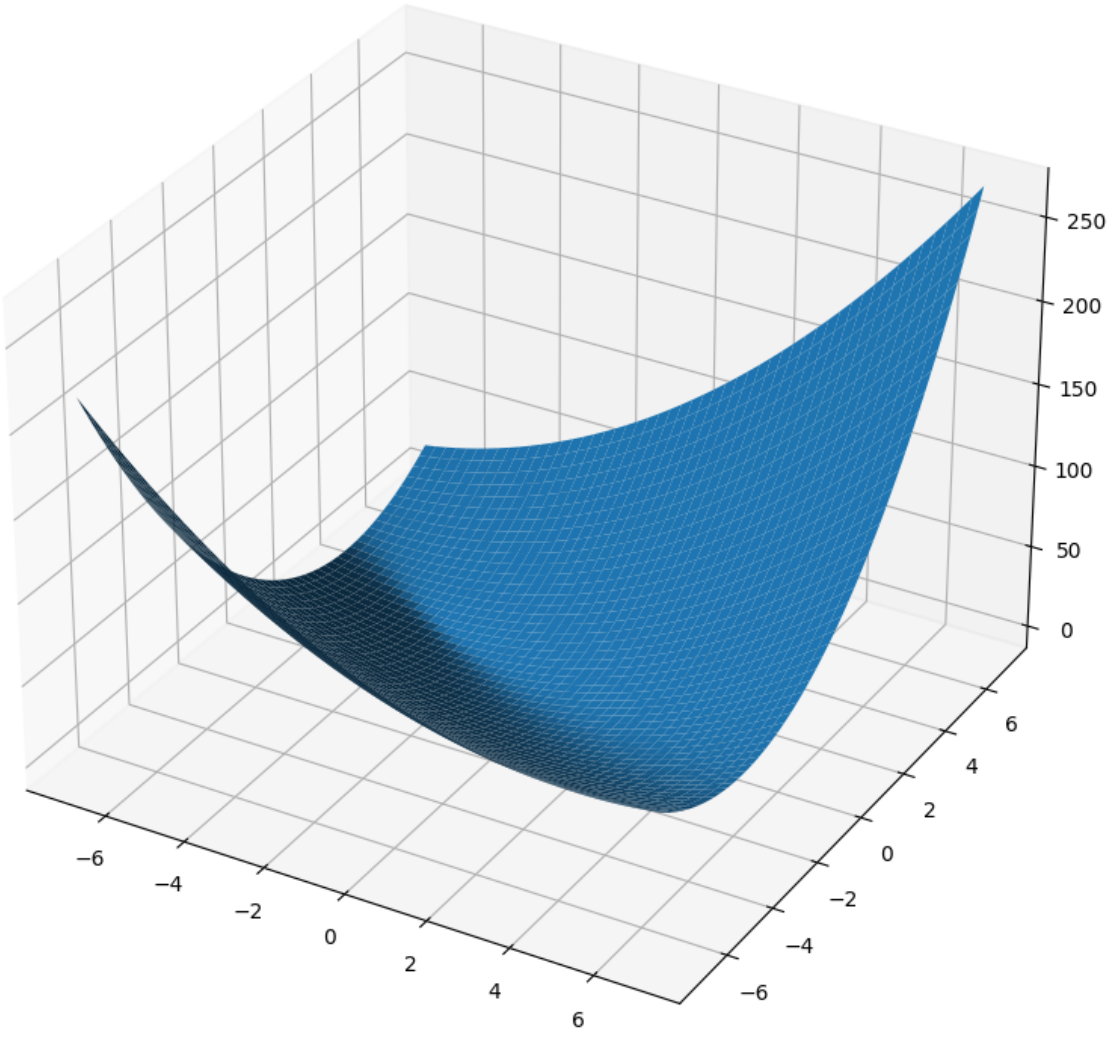
In [11]:
```
analyze_quadratic(
    roi=SearchRegion2d((-7, 7), (-7, 7)),
    fixed_steps=[0.1, 0.7],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=10,
    a=1, b=2, c=2, d=4, e=0
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [ 2.60000000e+00  2.00000000e+00]
 [ 1.28000000e+00  6.80000000e-01]
 [ 4.88000000e-01  1.52000000e-01]
 [-4.00000000e-02 -6.40000000e-03]
 [-4.30720000e-01  4.16000000e-03]
 [-7.45408000e-01  8.86400000e-02]
 [-1.01405440e+00  2.02265600e-01]
 [-1.25169664e+00  3.24170240e-01]
 [-1.46619136e+00  4.44841472e-01]
 [-1.66192138e+00  5.60143155e-01]
 [-1.84156574e+00  6.68470170e-01]
 [-2.00694662e+00  7.69395249e-01]
 [-2.15943635e+00  8.63026474e-01]
 [-2.30015437e+00  9.49703154e-01]
 [-2.43006413e+00  1.02985277e+00]
 [-2.55002186e+00  1.10392449e+00]
 [-2.66080238e+00  1.17235906e+00]
 [-2.76311372e+00  1.23557591e+00]
 [-2.85760616e+00  1.29396829e+00]
 [-2.94487859e+00  1.34790221e+00]]
Best value found: x* = [-2.94487859  1.34790221] with f(x*) = -7.412340428663294
Optimizing with fixed step = 0.7
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.18000000e+01 -1.60000000e+01]
 [ 2.43200000e+01  4.53200000e+01]
 [-7.59760000e+01 -1.15624000e+02]
 [ 1.89464000e+02  3.14489600e+02]
 [-5.18871040e+02 -8.31330880e+02]
 [ 1.36861165e+03  2.22281504e+03]
 [-3.66218572e+03 -5.91712338e+03]
 [ 9.74604702e+03  1.57778821e+04]
 [-2.59902537e+04 -4.20446536e+04]
 [ 6.92558165e+04  1.12066732e+05]
 [-1.84598551e+05 -2.98678260e+05]
 [ 4.91986184e+05  7.96058839e+05]
 [-1.31127965e+06 -2.12168657e+06]
 [ 3.49487026e+06  5.65482733e+06]
 [-9.31470917e+06 -1.50715076e+07]
 [ 2.48259914e+07  4.01693064e+07]
 [-6.61674284e+07 -1.07061140e+08]
 [ 1.76352564e+08  2.85344451e+08]
 [-4.70023260e+08 -7.60513602e+08]
 [ 1.25272834e+09  2.02695705e+09]]
Best value found: x* = [1.25272834e+09 2.02695705e+09] with f(x*) = 1.486489113169
1411e+19
Optimizing with binary search
Optimizer trajectory:
[[ 5.          5.        ]
 [ 0.35845947 -0.80192566]
 [-3.39523905  2.20143058]
 [-3.70708224  1.81167197]
```

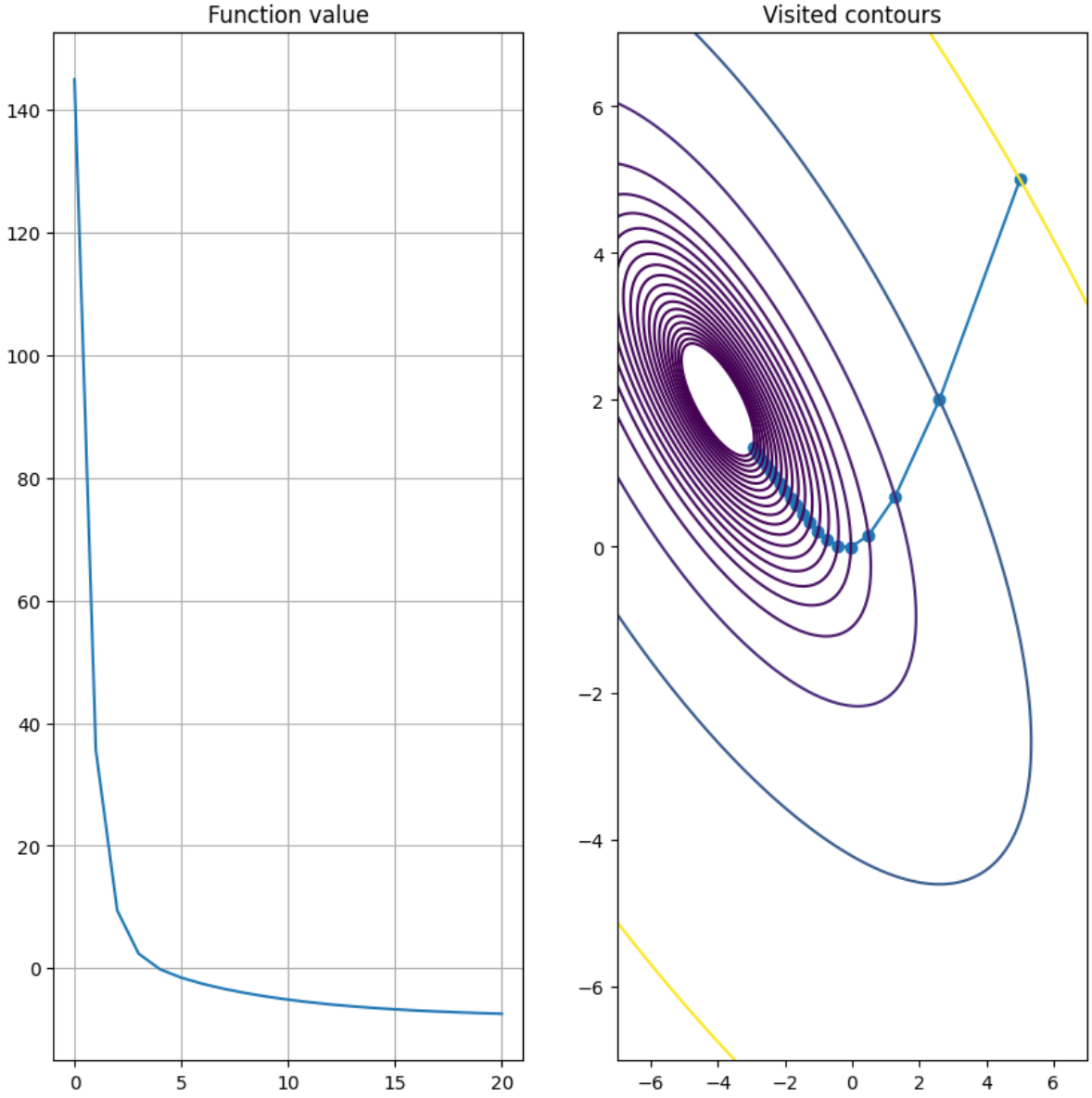
```
 [-3.95920291  2.01352887]
 [-3.98021758  1.98728089]
 [-3.99724287  2.00091356]
 [-3.99866284  1.99914018]
 [-3.99981294  2.00006171]
 [-3.99990918  1.9999416 ]
 [-3.99998729  2.00000419]
 [-3.99999383  1.99999603]
 [-3.99999913  2.00000028]
 [-3.99999958  1.99999973]
 [-3.99999994  2.00000002]
 [-3.99999997  1.99999998]
 [-3.99999999  2.        ]
 [-3.99999999  2.        ]
 [-4.          2.        ]
 [-4.          2.        ]
 [-4.          2.        ]]
Best value found: x* = [-4.  2.] with f(x*) = -8.0
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.          5.        ]
 [-0.25       -1.5625    ]
 [-0.35546875  0.3359375 ]
 [-1.84082031  0.09863281]
 [-2.00195312  1.12585449]
 [-2.77463531  0.95414734]
 [-2.89804983  1.54975653]
 [-3.30536652  1.42384201]
 [-3.39422314  1.76710388]
 [-3.60396856  1.68836253]
 [-3.67781328  1.88720059]
 [-3.79559364  1.8328699 ]
 [-3.82355083  1.93026028]
 [-3.89024424  1.9071542 ]
 [-3.90186984  1.95936009]
 [-3.93780125  1.94882863]
 [-3.94607178  1.97893662]
 [-3.96661231  1.97156071]
 [-3.97001433  1.98771069]
 [-3.98107456  1.98433128]
 [-3.9835171   1.99364028]]
Best value found: x* = [-3.9835171   1.99364028] with f(x*) = -7.999857075264955
Optimizing with golden ration
Optimizer trajectory:
[[ 5.          5.        ]
 [ 0.35844352 -0.8019456 ]
 [-3.39500537  2.20141477]
 [-3.70693122  1.81158002]
 [-3.95922105  2.01353756]
 [-3.98023217  1.98728973]
 [-3.99724075  2.00091264]
 [-3.99866121  1.99913913]
 [-3.99981269  2.0000618 ]
 [-3.99990907  1.99994151]
 [-3.99998718  2.00000422]
 [-3.99999378  1.99999599]
```

```
 [-3.9999991   2.00000031]
 [-3.99999956  1.99999974]
 [-3.99999998  1.99999993]
 [-3.99999996  1.99999999]
 [-3.99999999  1.99999997]
 [-3.99999997  2.00000001]
 [-3.99999999  1.99999999]
 [-3.99999998  2.00000001]
 [-4.00000001  1.99999997]]
Best value found: x* = [-4.00000001  1.99999997] with f(x*) = -7.999999999999998
Optimizing with fibonacci search limited by 10 iterations:
Optimizer trajectory:
[[ 5.          5.        ]
 [ 0.14606742 -1.06741573]
 [-1.06590077  1.16715061]
 [-2.10472092  0.54007623]
 [-2.49605154  1.46103664]
 [-3.01649294  1.2312721 ]
 [-3.19989987  1.70430698]
 [-3.48327465  1.58705187]
 [-3.56489718  1.83022   ]
 [-3.71991726  1.77439723]
 [-3.76154241  1.90514274]
 [-3.8486707   1.87556829]
 [-3.86801279  1.94570518]
 [-3.91689789  1.93098311]
 [-3.92702659  1.97048453]
 [-3.95437091  1.96171174]
 [-3.95948475  1.98327063]
 [-3.97445114  1.97883058]
 [-3.9776004   1.99090428]
 [-3.98597138  1.98825645]
 [-3.98756323  1.9948455 ]]
Best value found: x* = [-3.98756323  1.9948455 ] with f(x*) = -7.9999203995942825
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.          5.        ]
 [-1.         -2.5       ]
 [-0.25        0.5       ]
 [-2.5        -0.25      ]
 [-2.125       1.25      ]
 [-3.25        0.875     ]
 [-3.0625      1.625     ]
 [-3.625       1.4375    ]
 [-3.53125     1.8125    ]
 [-3.8125      1.71875   ]
 [-3.765625    1.90625   ]
 [-3.90625     1.859375  ]
 [-3.8828125   1.953125  ]
 [-3.953125    1.9296875 ]
 [-3.94140625  1.9765625 ]
 [-3.9765625   1.96484375]
 [-3.97070312  1.98828125]
 [-3.98828125  1.98242188]
 [-3.98535156  1.99414062]
 [-3.99414062  1.99121094]
```

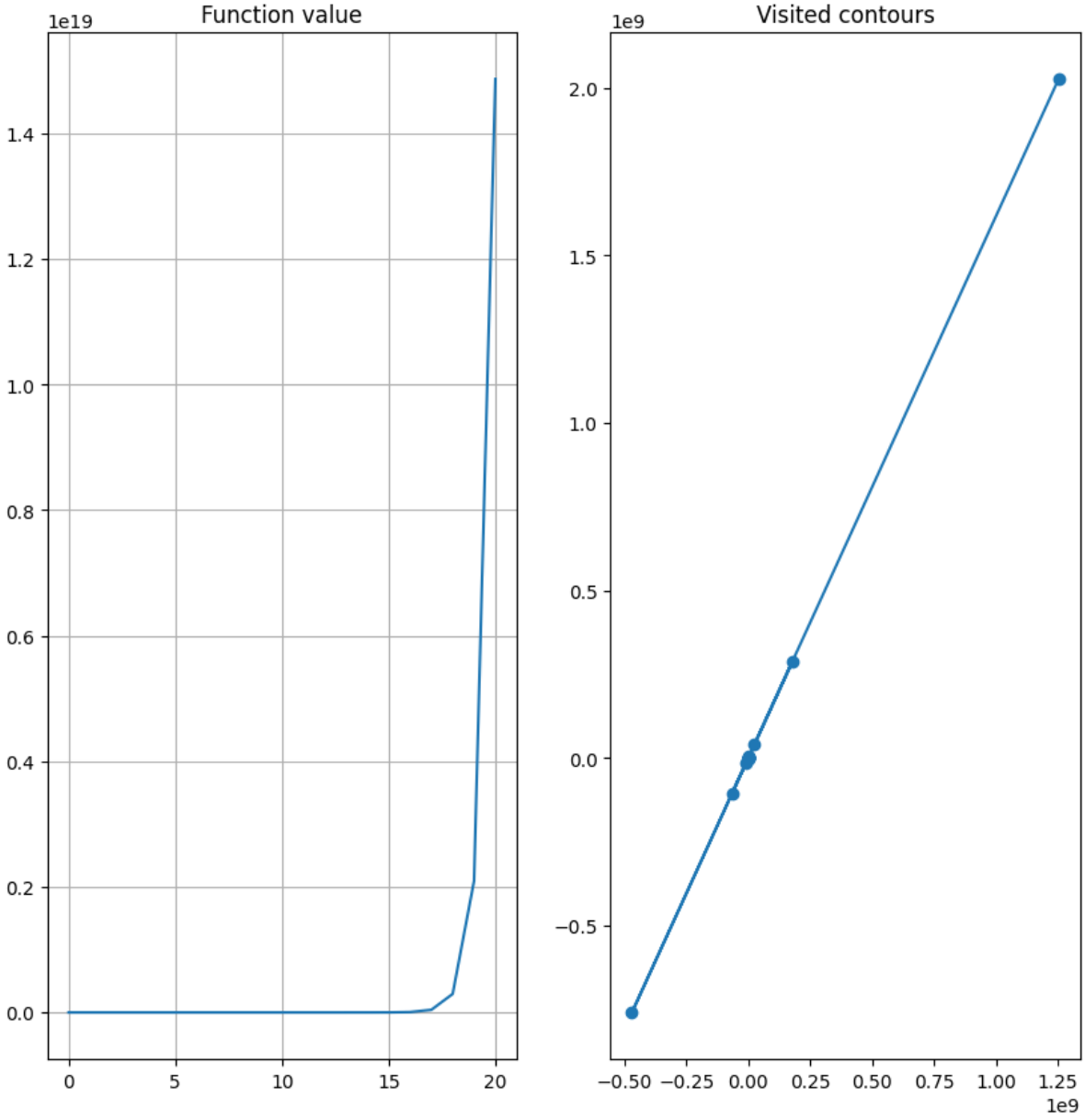
```
 [-3.99267578  1.99707031]]
Best value found: x* = [-3.99267578  1.99707031] with f(x*) = -7.999972105026245
```



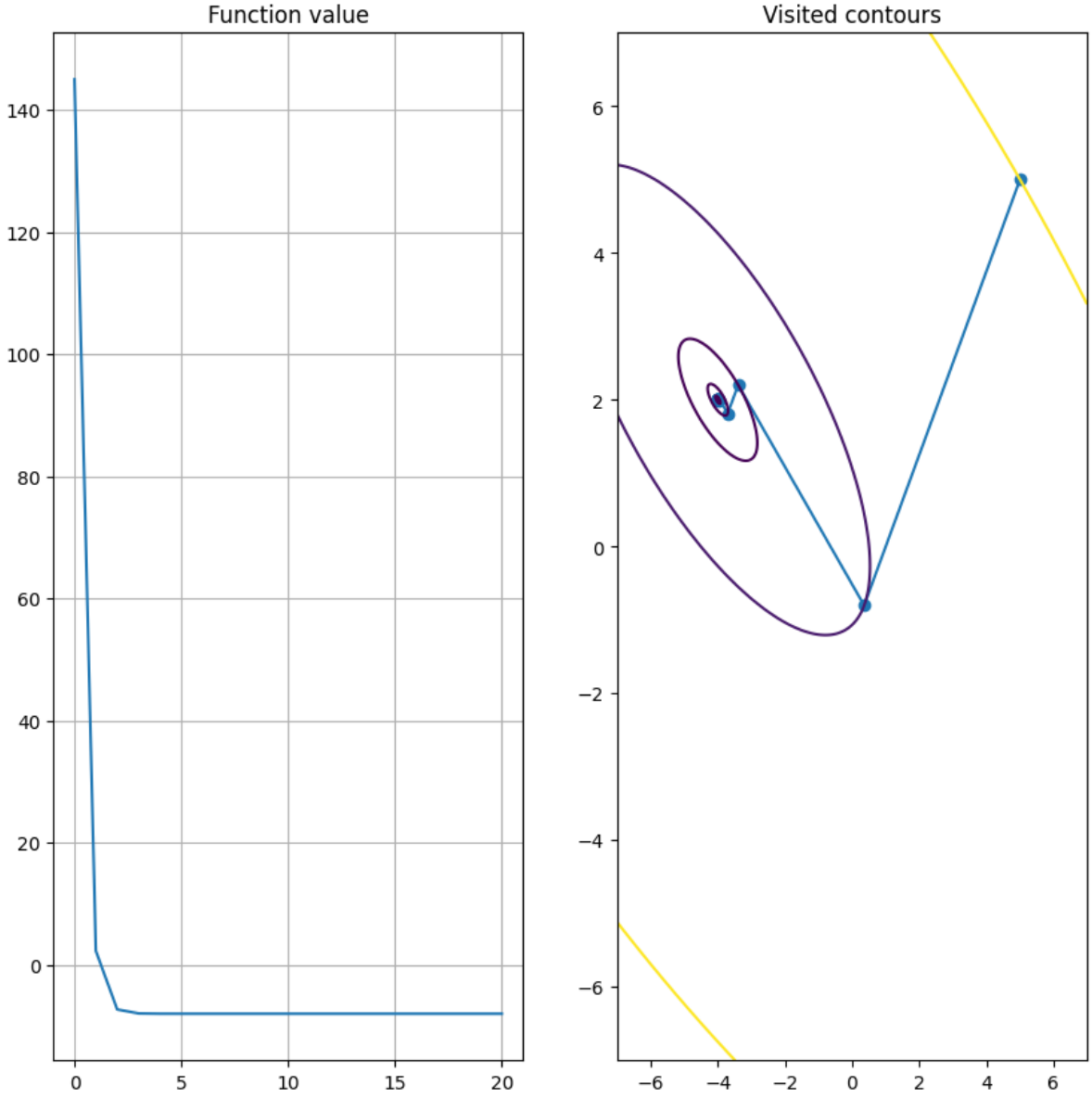
Optimizing with fixed step = 0.1



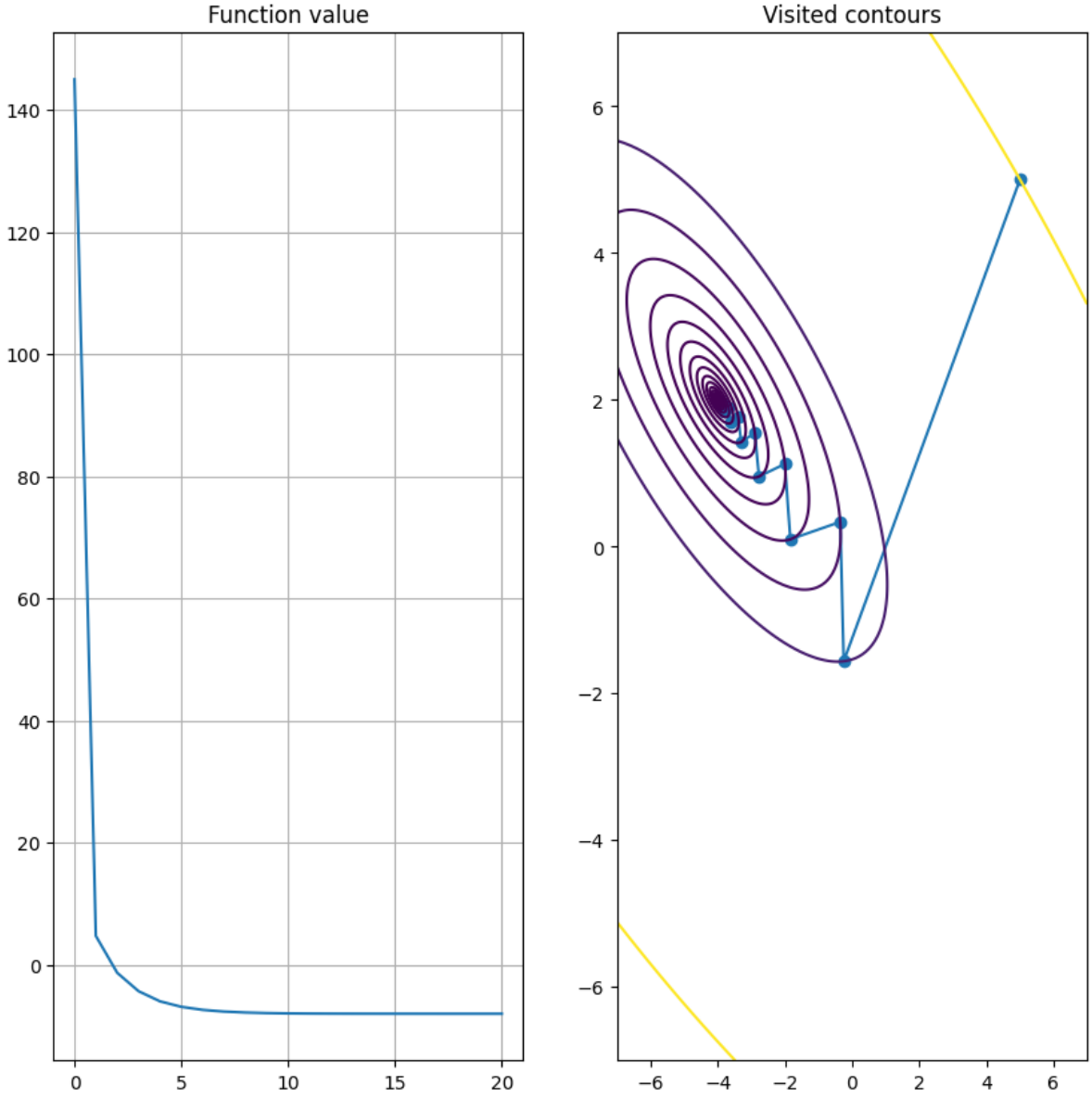
Optimizing with fixed step = 0.7



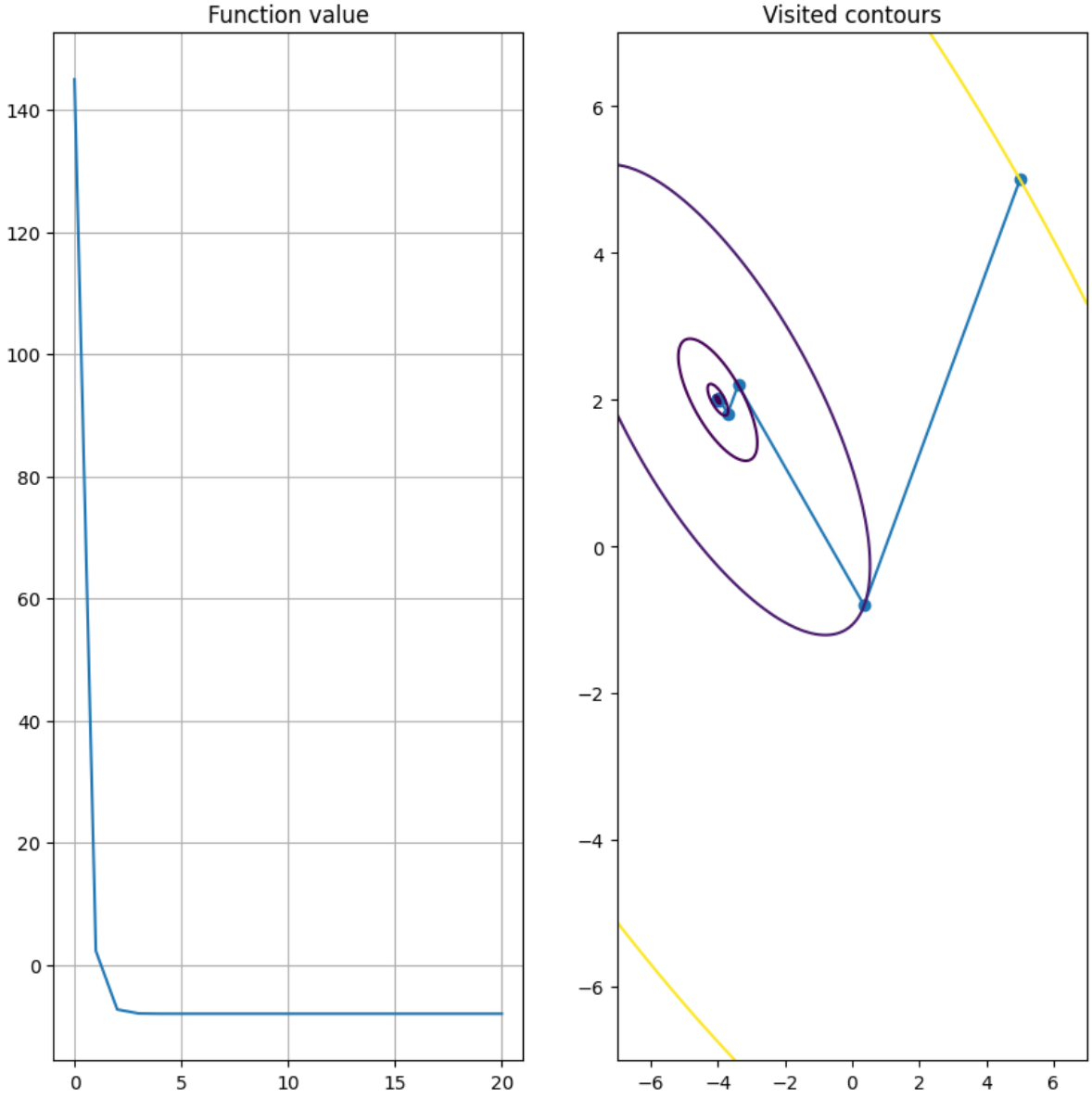
Optimizing with binary search



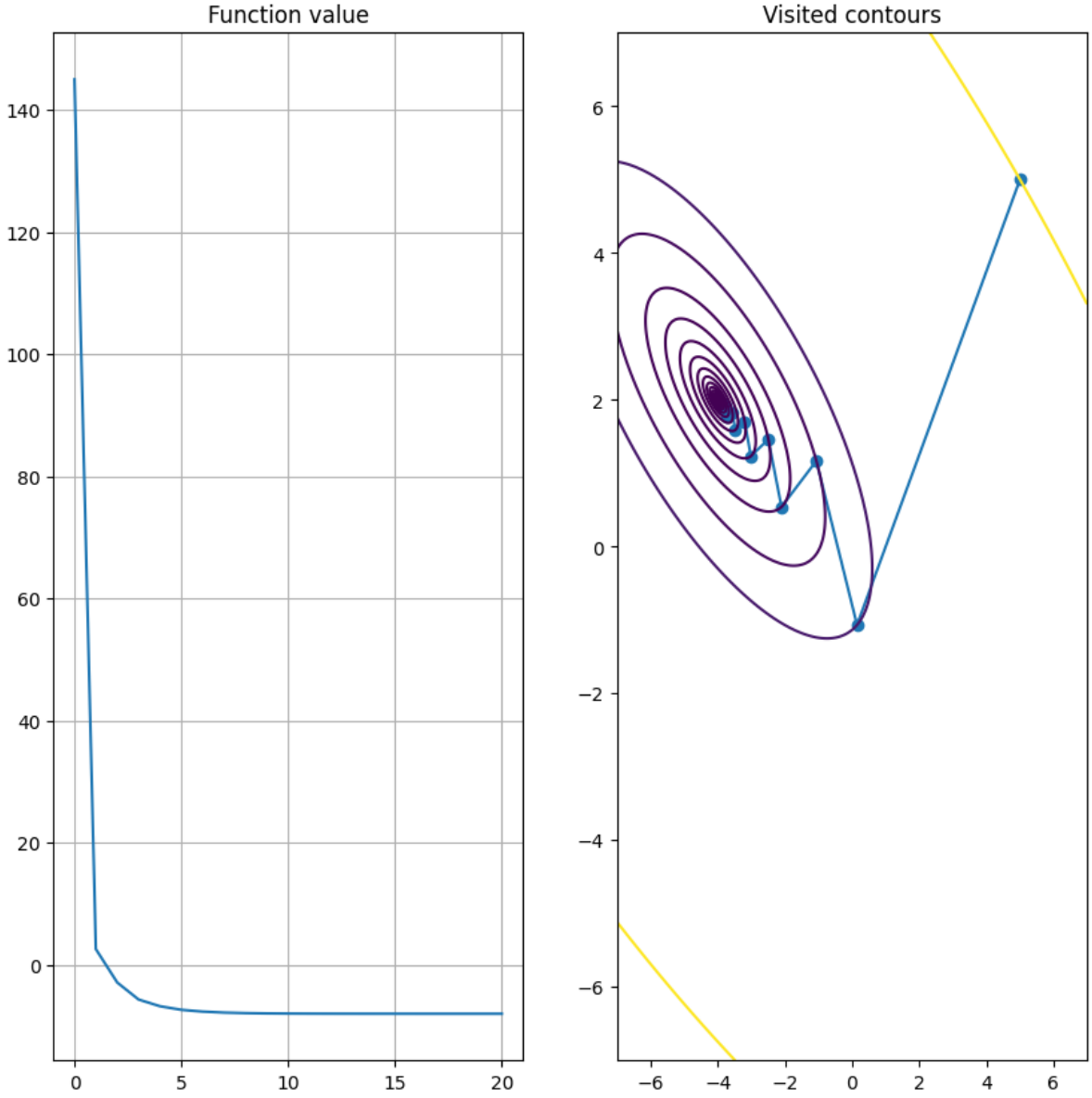
Optimizing with binary search limited by 5 iterations

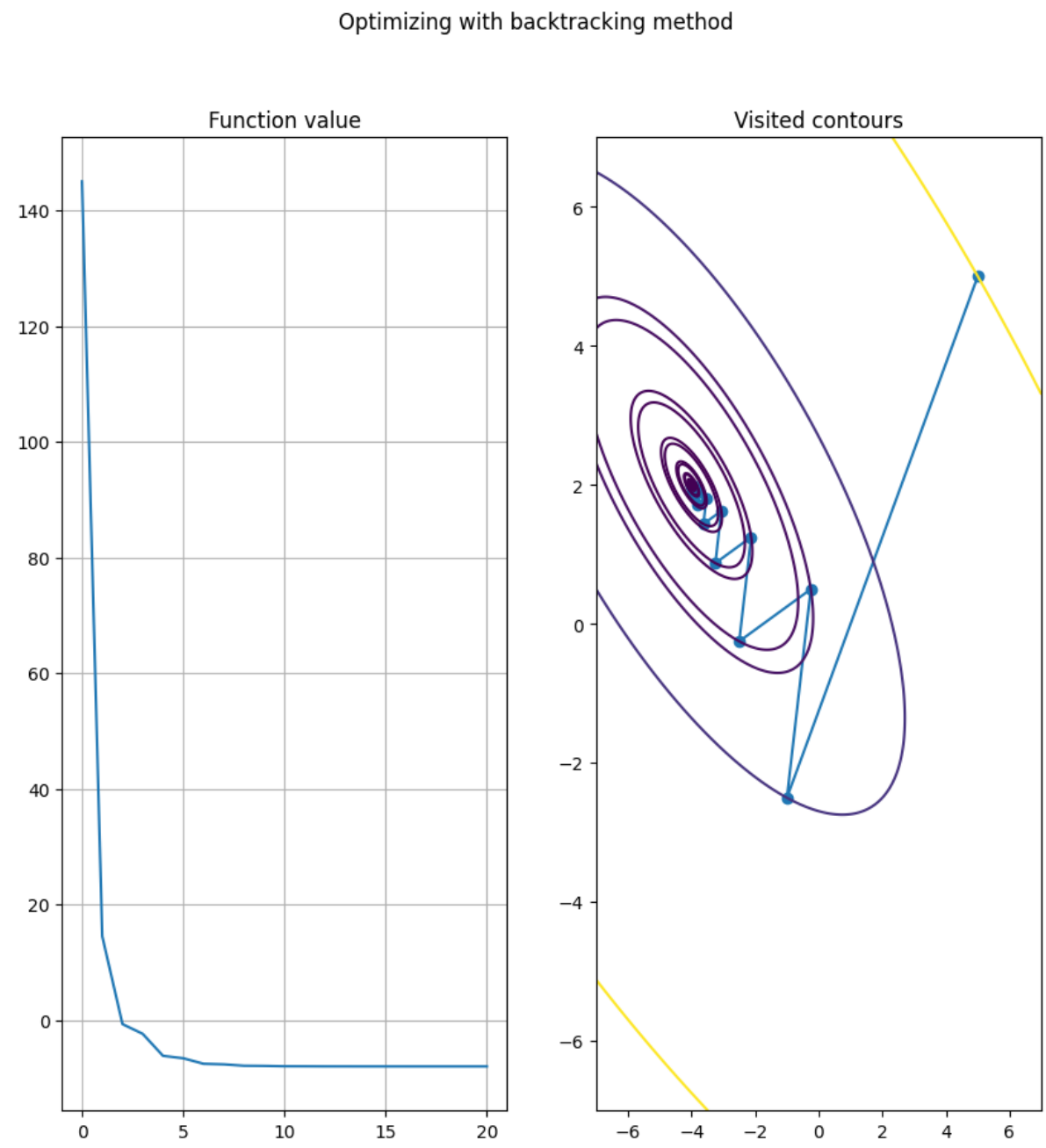


Optimizing with golden ration



Optimizing with fibonacci search limited by 10 iterations:





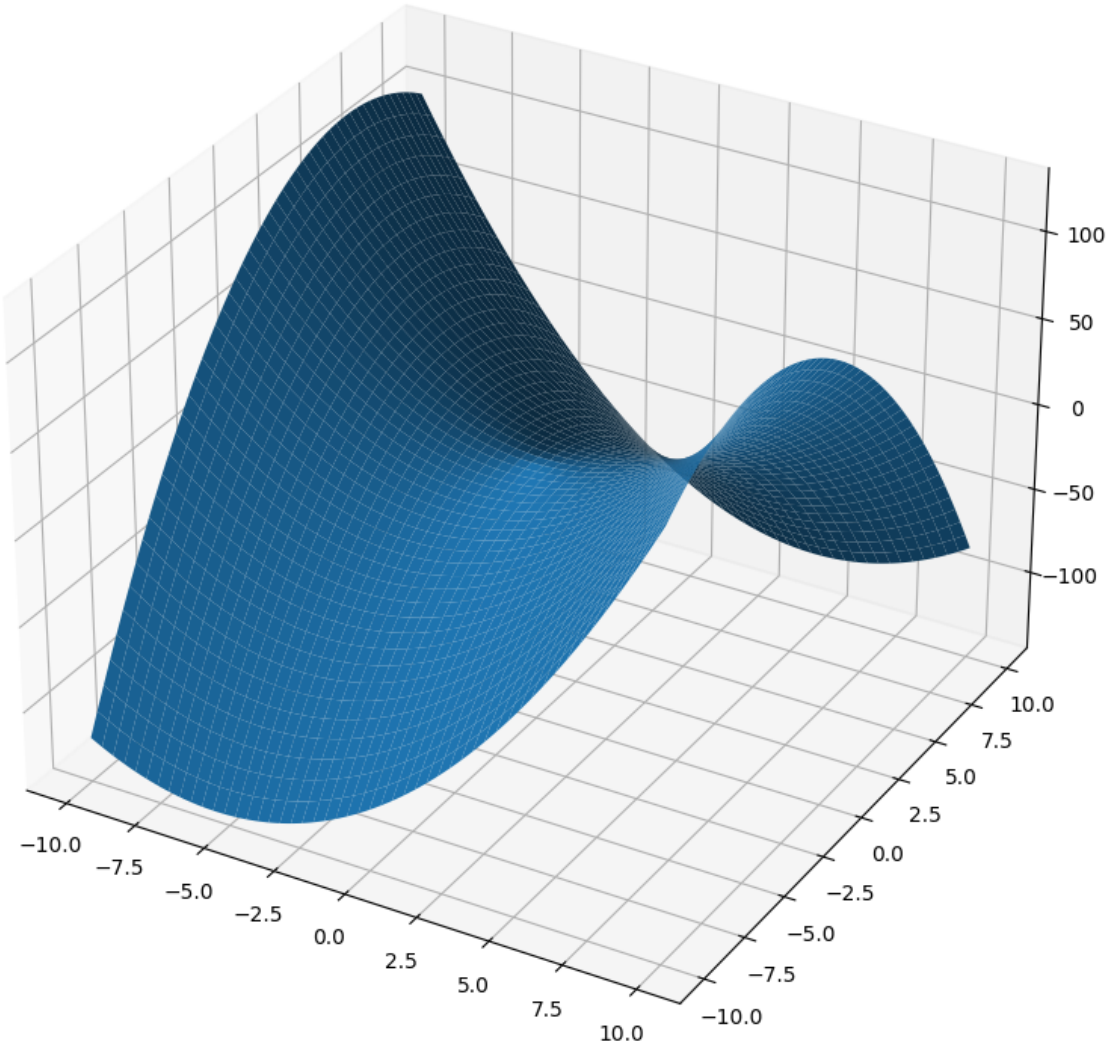
In [6]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.5, 0.1], x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=1, b=-1, c=-1, d=1, e=1
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[  5.           5.        ]
 [  4.4          6.4       ]
 [  4.06         8.02      ]
 [  3.95         9.93      ]
 [  4.053       12.211     ]
 [  4.3635      14.9585    ]
 [  4.88665     18.28655   ]
 [  5.637975    22.332525  ]
 [  6.6436325   27.2628275 ]
 [  7.94118875  33.27975625]
 [  9.58092663  40.62982638]
 [ 11.62772394  49.61388431]
 [ 14.16356758  60.59943357]
 [ 17.29079742  74.03567704]
 [ 21.13620564  90.47189219]
 [ 25.85615373 110.57989119]
 [ 31.64291211 135.18148481]
 [ 38.73247816 165.28207298]
 [ 47.41418983 202.11173539]
 [ 58.0425254  247.17550145]
 [ 71.05157047 302.31485428]]
Best value found: x* = [ 71.05157047 302.31485428] with f(x*) = -107452.5241993375
Optimizing with fixed step = 0.5
Optimizer trajectory:
[[5.00000000e+00 5.00000000e+00]
 [2.00000000e+00 1.20000000e+01]
 [5.50000000e+00 2.45000000e+01]
 [1.17500000e+01 5.12500000e+01]
 [2.51250000e+01 1.07875000e+02]
 [5.34375000e+01 2.27812500e+02]
 [1.13406250e+02 4.81843750e+02]
 [2.40421875e+02 1.01989062e+03]
 [5.09445312e+02 2.15949219e+03]
 [1.07924609e+03 4.57320703e+03]
 [2.28610352e+03 9.68553711e+03]
 [4.84226855e+03 2.05136260e+04]
 [1.02563130e+04 4.34478862e+04]
 [2.17234431e+04 9.20234290e+04]
 [4.60112145e+04 1.94908079e+05]
 [9.74535397e+04 4.12821266e+05]
 [2.06410133e+05 8.74368802e+05]
 [4.37183901e+05 1.85194217e+06]
 [9.25970585e+05 3.92247579e+06]
 [1.96123740e+06 8.30793638e+06]
 [4.15396769e+06 1.75964910e+07]]
Best value found: x* = [ 4153967.6888752  17596490.95362663] with f(x*) = -3654762
79429335.94
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[  5.           5.        ]
 [  4.4          6.4       ]
 [  4.06         8.02      ]
 [  3.95         9.93      ]
```

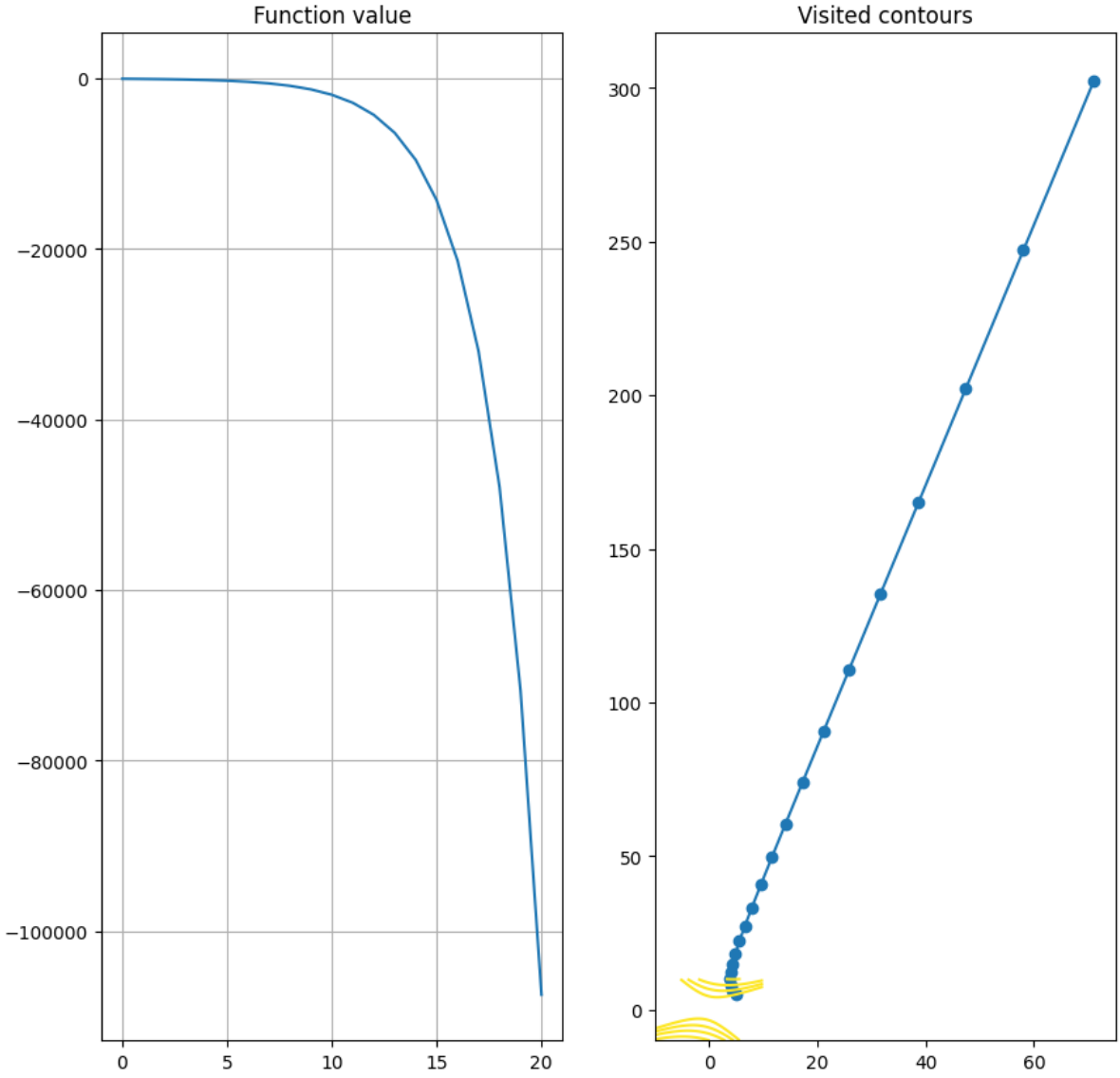
```
 [  4.053       12.211     ]
 [  4.3635      14.9585    ]
 [  4.88665     18.28655   ]
 [  5.637975    22.332525  ]
 [  6.6436325   27.2628275 ]
 [  7.94118875  33.27975625]
 [  9.58092663  40.62982638]
 [ 11.62772394  49.61388431]
 [ 14.16356758  60.59943357]
 [ 17.29079742  74.03567704]
 [ 21.13620564  90.47189219]
 [ 25.85615373 110.57989119]
 [ 31.64291211 135.18148481]
 [ 38.73247816 165.28207298]
 [ 47.41418983 202.11173539]
 [ 58.0425254  247.17550145]
 [ 71.05157047 302.31485428]]
Best value found: x* = [ 71.05157047 302.31485428] with f(x*) = -107452.5241993375
Optimizing with binary search
```

```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
Cell In[6], line 1
----> 1 analyze_quadratic(roi=SearchRegion2d((-10, 10), (-10, 10)), fixed_steps=[
0.1, 0.5, 0.1], x0=np.array([5, 5]), bin_iters=5, fib_iters=30, a=1, b=-1, c=-1, d
=1, e=1)

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:37, in analyze
_quadratic(roi, x0, fixed_steps, bin_iters, fib_iters, a, b, c, d, e)
     35 for title, linear in cases:
     36     print(title)
---> 37     visualize_optimizer_with(linear).suptitle(title)

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:22, in analyze
_quadratic.<locals>.visualize_optimizer_with(linear_search)
     19 else:
     20     true_min = None
---> 22 return visualize_optimizing_process(f, roi, np.array(gradient_descent(f, d
f, x0, linear_search, lambda f, points: len(points) > 20)), true_min)

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:22, in gradient_
descent(target_function, gradient_function, x0, linear_search, terminate_conditio
n)
     20     if np.linalg.norm(g) == 0:
     21         return points
---> 22     next_point = last_point - g * linear_search(lambda l: target_function(
last_point - g * l),
     23                                                 lambda l: -np.dot(g, gradi
ent_function(last_point - g * l)))
     24     points.append(next_point)
     25 return points

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:43, in bin_searc
h(f, derivative)
     40 def bin_search(f: Callable[[float], float], derivative: Callable[[float],
float]):
     41     # assume derivative(0) < 0 and derivative is rising
     42     l = 0
---> 43     r = find_upper_bound(f)
     45     while r - l > precision:
     46         m = (l + r) / 2

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:31, in find_uppe
r_bound(f)
     29 original = f(0)
     30 r = 1
---> 31 while f(r) < original:
     32     r *= 2
     33 return r

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:22, in gradient_
descent.<locals>.<lambda>(l)
     20     if np.linalg.norm(g) == 0:
     21         return points
---> 22     next_point = last_point - g * linear_search(lambda l: target_function(
last_point - g * l),
```

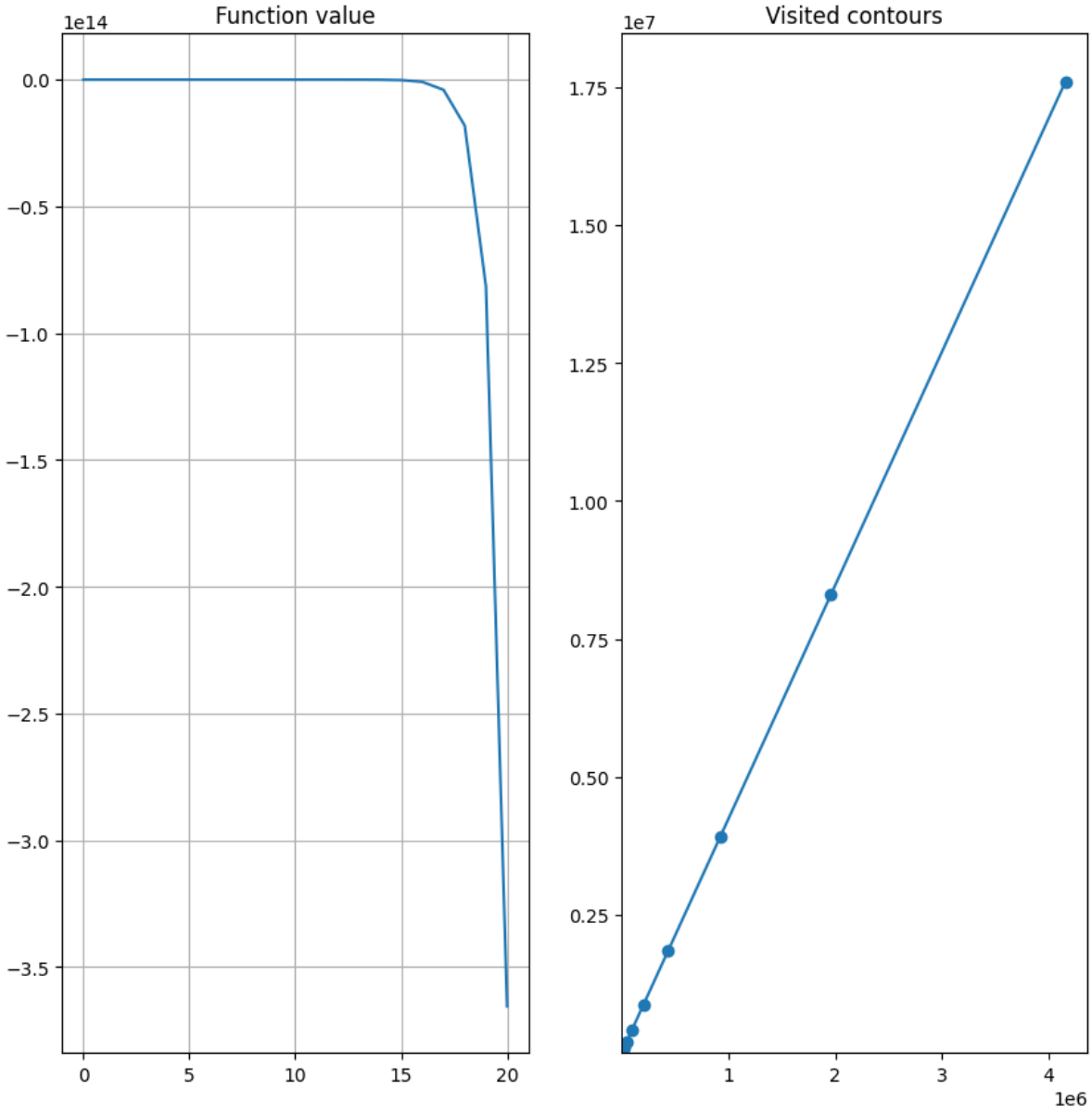
```
     23                                                 lambda l: -np.dot(g, gradi
ent_function(last_point - g * l)))
     24     points.append(next_point)
     25 return points

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:6, in create_q
uadratic.<locals>.<lambda>(x)
      5 def create_quadratic(a, b, c, d, e):
----> 6     return lambda x: a * x[0] ** 2 + b * x[0] * x[1] + c * x[1] ** 2 + d *
 x[0] + e * x[1]

OverflowError: (34, 'Result too large')
```

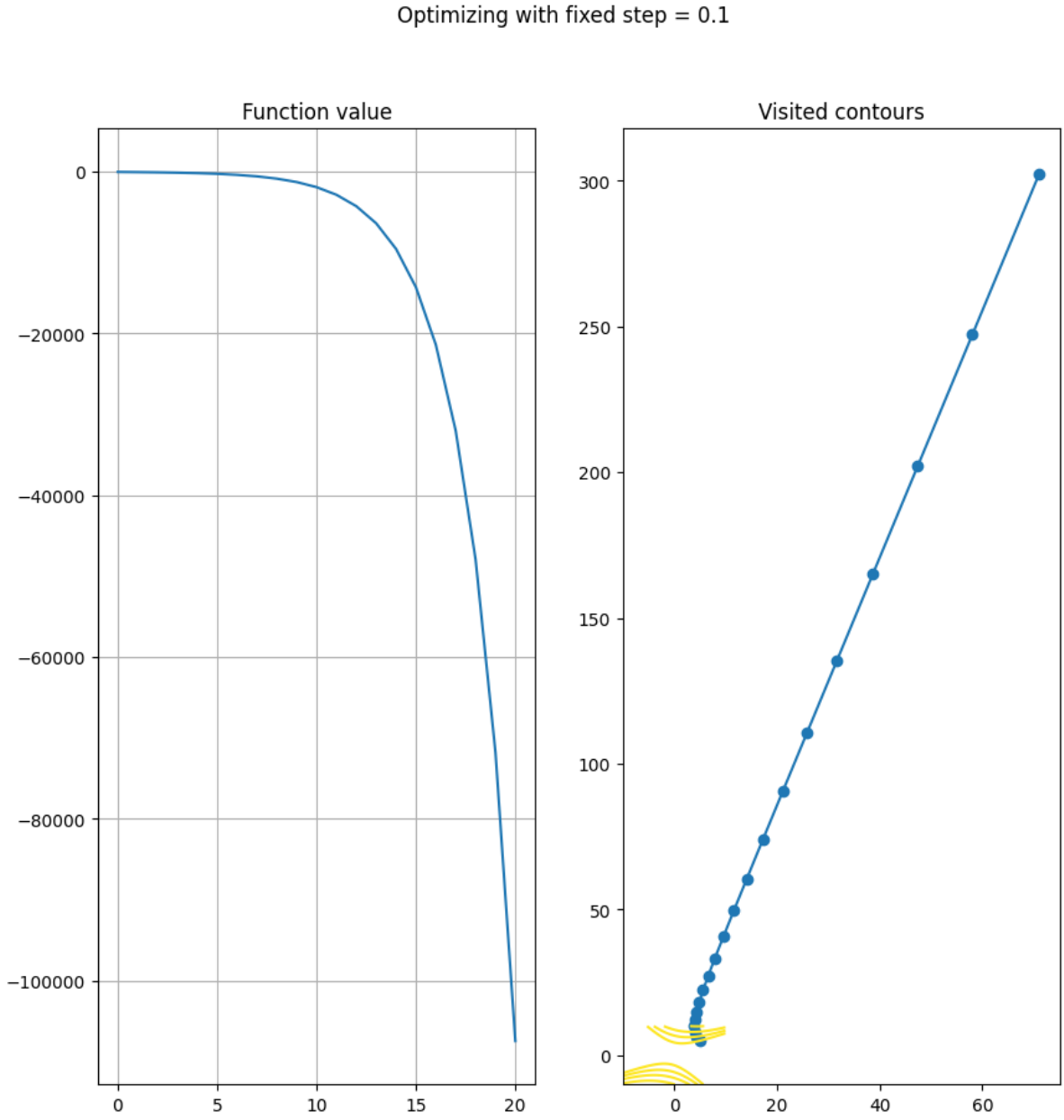


Optimizing with fixed step = 0.1



Optimizing with fixed step = 0.5





Случай неопределенной формы.

У таких функций есть ровно 1 стационарная точка, но сама функция не ограничена, поэтому нет глобального минимума, а единственная стационарная точка даже не является локальным минимумом.

На таком примере видно, что градиентный спуск сходится к стационарной точке, которая находится "ниже" остальных, но эта точка необязательно будет локальным минимумом. Помимо этого, видно, что на таких функциях очень важно стартовое условие. Если стартовать со "склона", то метод за доступные шаги будет "скатываться" по склону, пока не будет достигнуто терминально условия (в данном случае количество итераций просто ограничено).

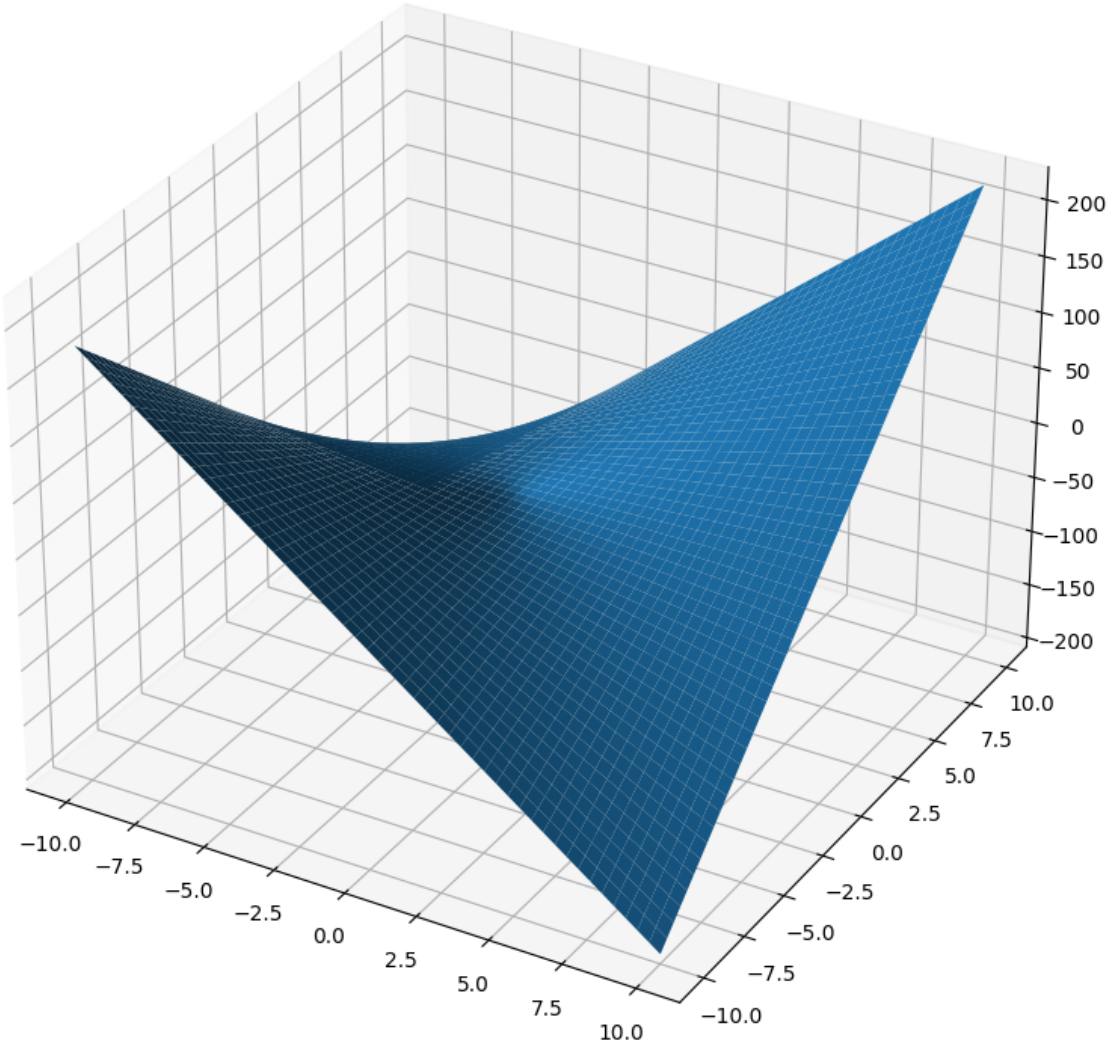
In [12]:
```
analyze_quadratic(
```

```
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.5, 1],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=0, b=2, c=0, d=1, e=1
)
```

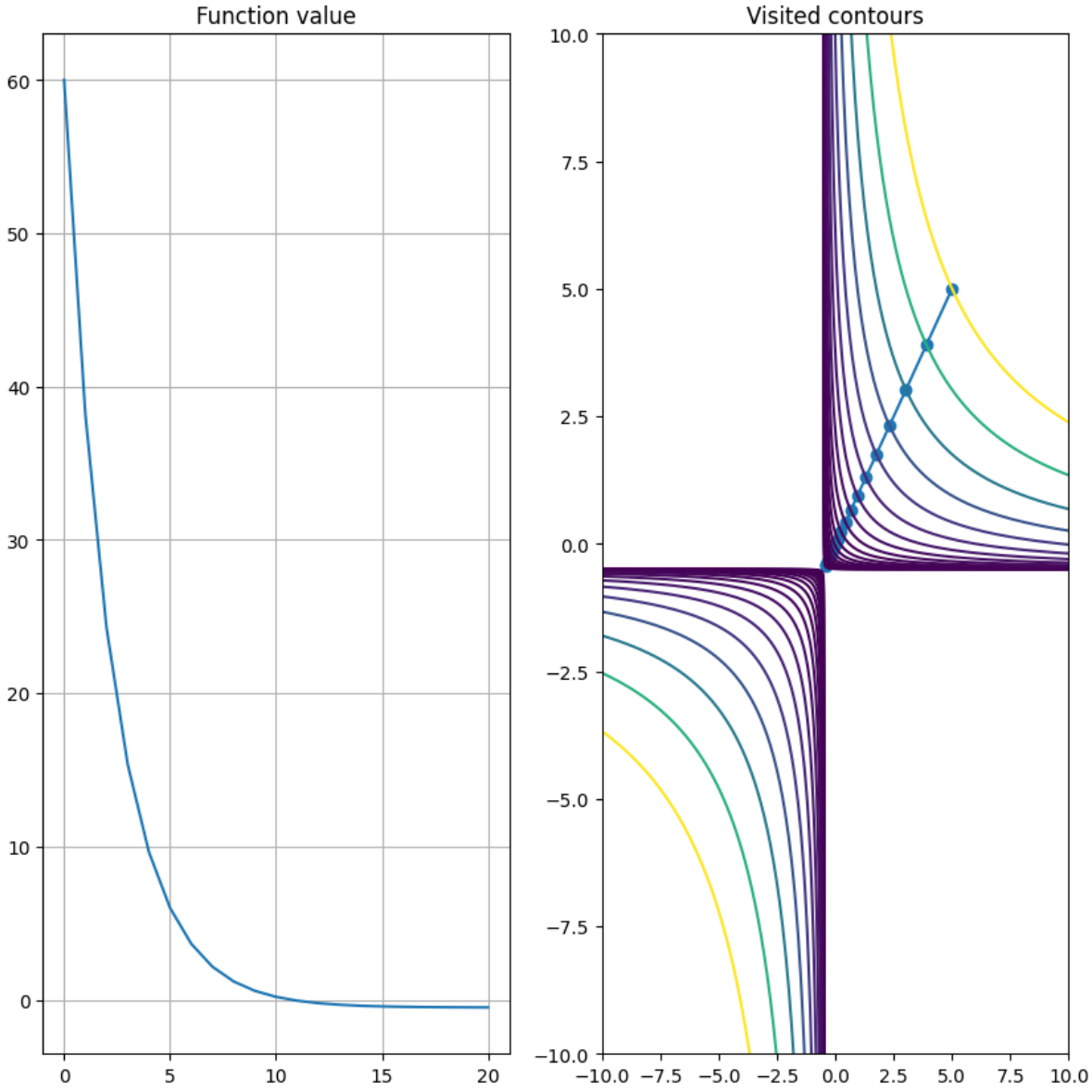
```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[ 5.          5.        ]
 [ 3.9         3.9       ]
 [ 3.02        3.02      ]
 [ 2.316       2.316     ]
 [ 1.7528      1.7528    ]
 [ 1.30224     1.30224   ]
 [ 0.941792    0.941792  ]
 [ 0.6534336   0.6534336 ]
 [ 0.42274688  0.42274688]
 [ 0.2381975   0.2381975 ]
 [ 0.090558    0.090558  ]
 [-0.0275536  -0.0275536 ]
 [-0.12204288 -0.12204288]
 [-0.1976343  -0.1976343 ]
 [-0.25810744 -0.25810744]
 [-0.30648595 -0.30648595]
 [-0.34518876 -0.34518876]
 [-0.37615101 -0.37615101]
 [-0.40092081 -0.40092081]
 [-0.42073665 -0.42073665]
 [-0.43658932 -0.43658932]]
Best value found: x* = [-0.43658932 -0.43658932] with f(x*) = -0.49195817062550123
Optimizing with fixed step = 0.5
Optimizer trajectory:
[[ 5.   5. ]
 [-0.5 -0.5]]
Best value found: x* = [-0.5 -0.5] with f(x*) = -0.5
Optimizing with fixed step = 1
Optimizer trajectory:
[[ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]
 [-6 -6]
 [ 5  5]]
Best value found: x* = [5 5] with f(x*) = 60
Optimizing with binary search
Optimizer trajectory:
```

```
[[ 5.   5. ]
 [-0.5 -0.5]]
Best value found: x* = [-0.5 -0.5] with f(x*) = -0.5
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.   5. ]
 [-0.5 -0.5]]
Best value found: x* = [-0.5 -0.5] with f(x*) = -0.5
Optimizing with golden ration
Optimizer trajectory:
[[ 5.          5.        ]
 [-0.50005305 -0.50005305]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]
 [-0.50000001 -0.50000001]
 [-0.5        -0.5       ]]
Best value found: x* = [-0.5 -0.5] with f(x*) = -0.49999999999999994
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.          5.        ]
 [-0.50000409 -0.50000409]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]
 [-0.50000001 -0.50000001]]
Best value found: x* = [-0.50000001 -0.50000001] with f(x*) = -0.5
```

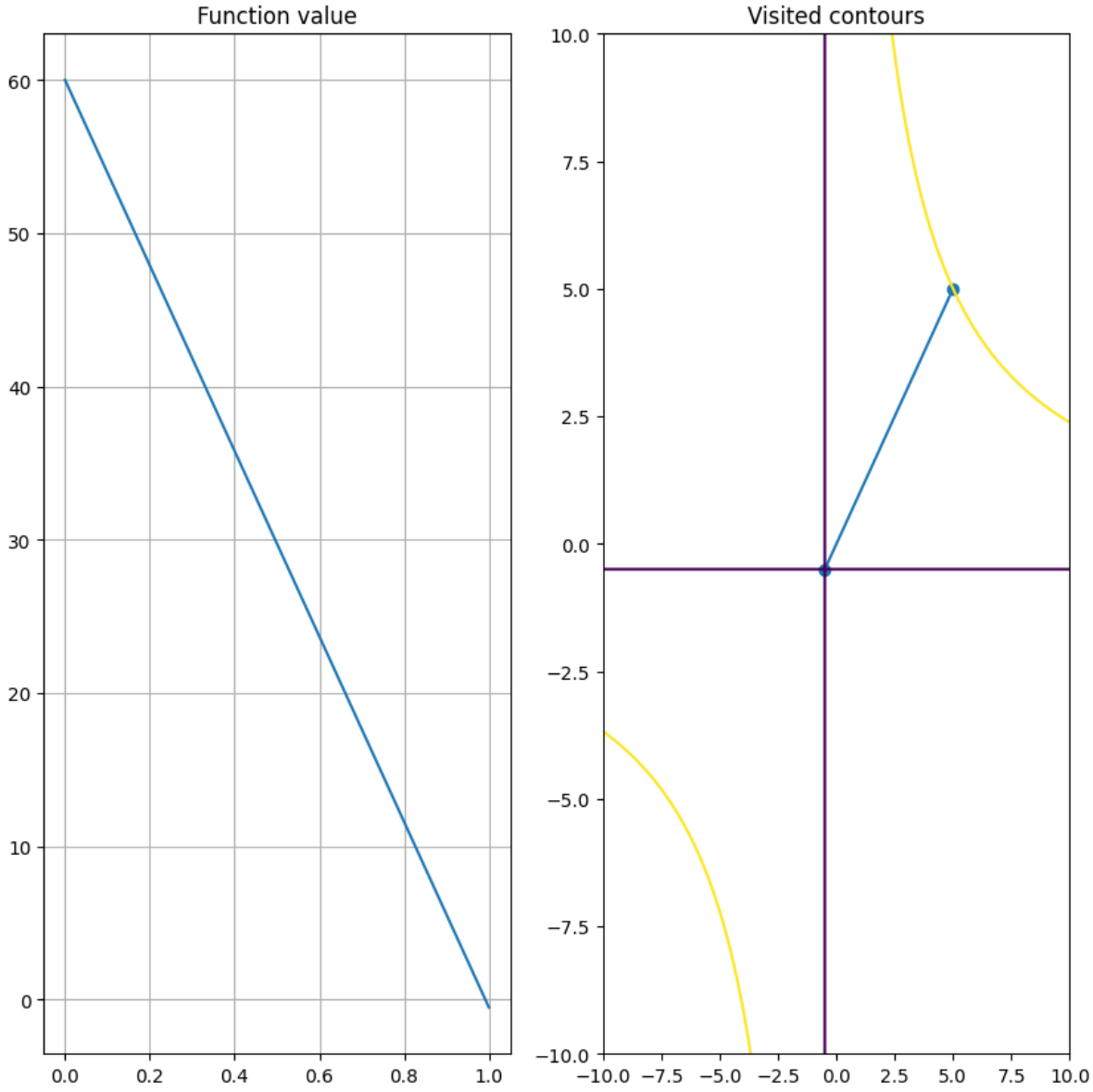
```
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.   5. ]
 [-0.5 -0.5]]
Best value found: x* = [-0.5 -0.5] with f(x*) = -0.5
```



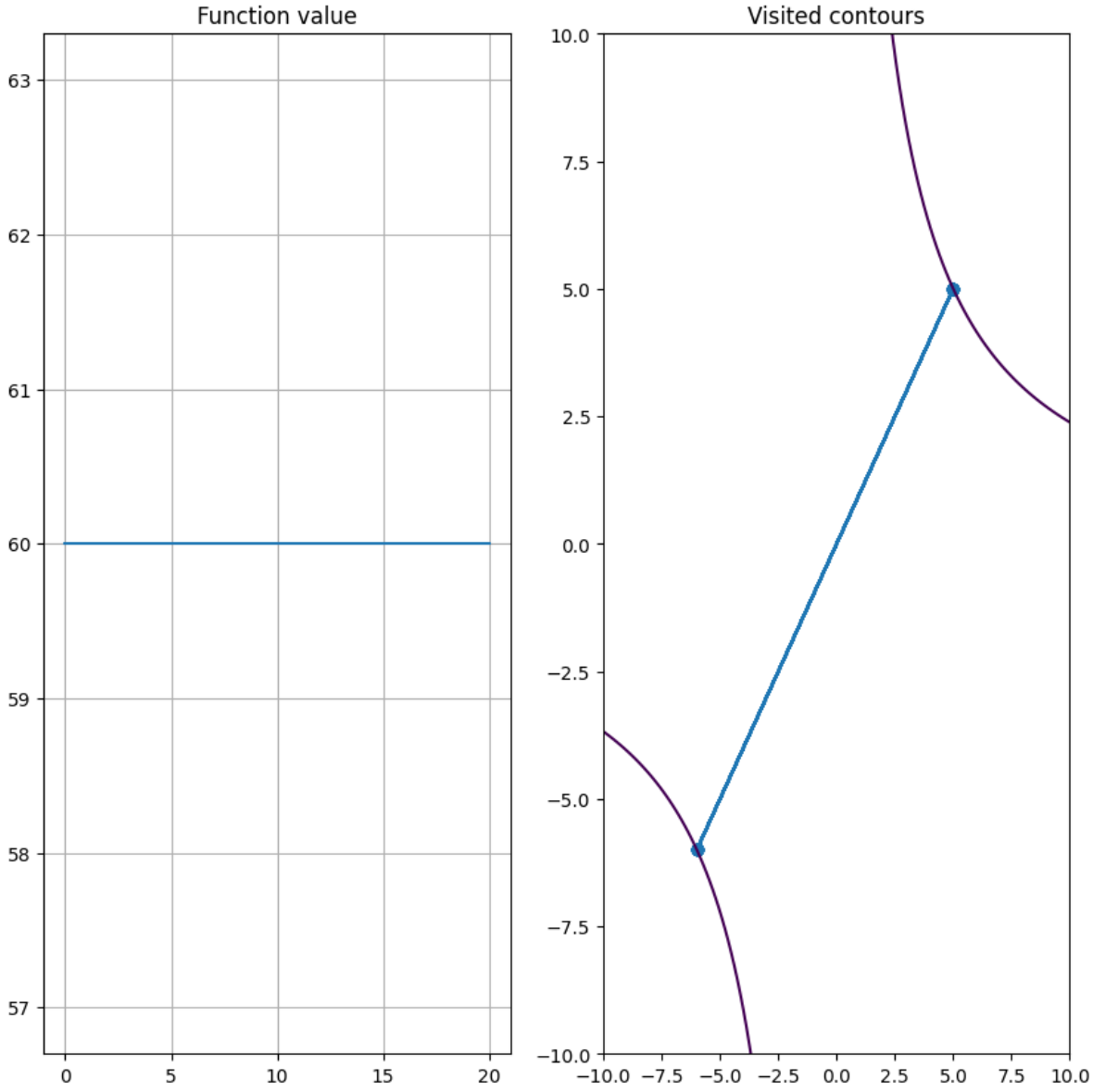
Optimizing with fixed step = 0.1



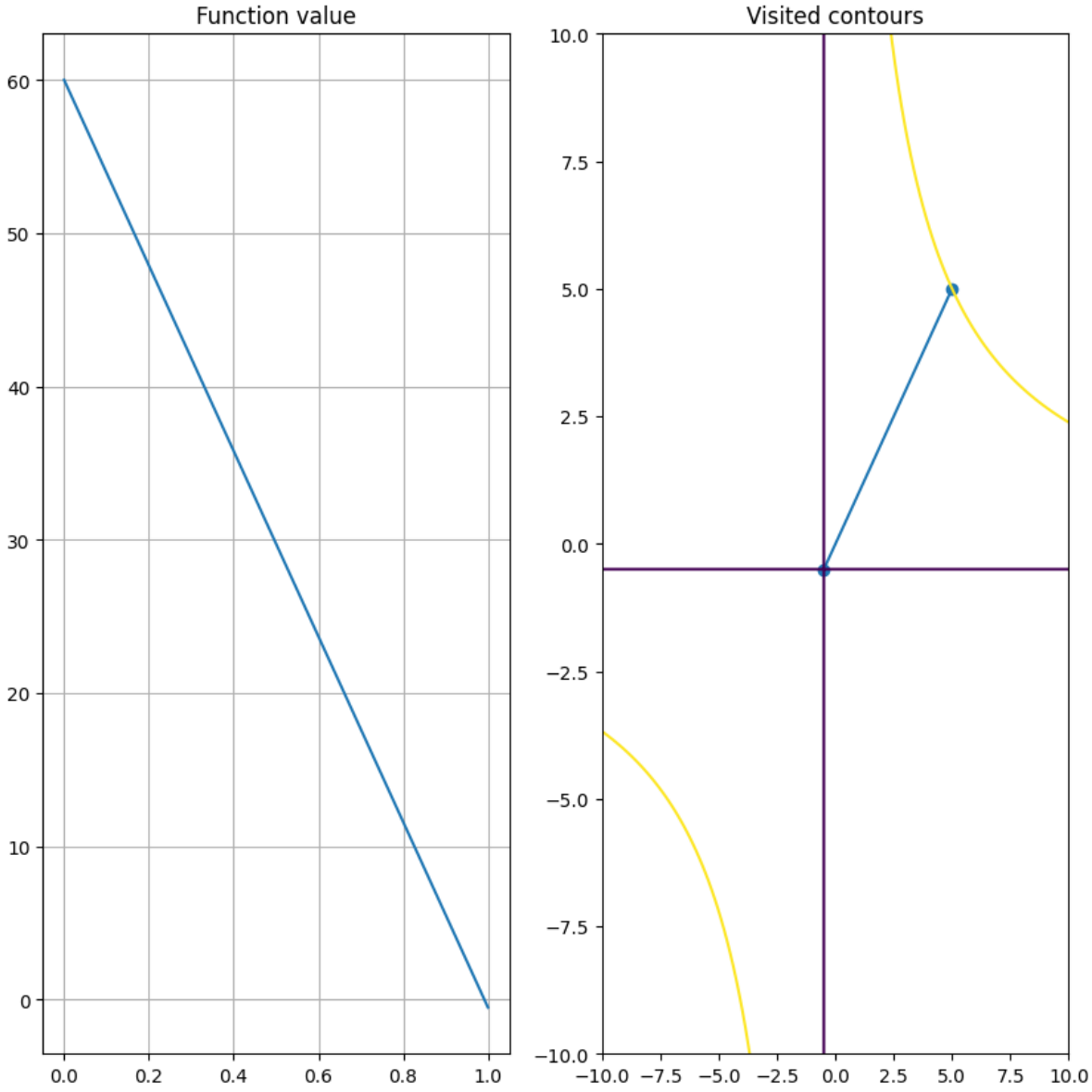
Optimizing with fixed step = 0.5



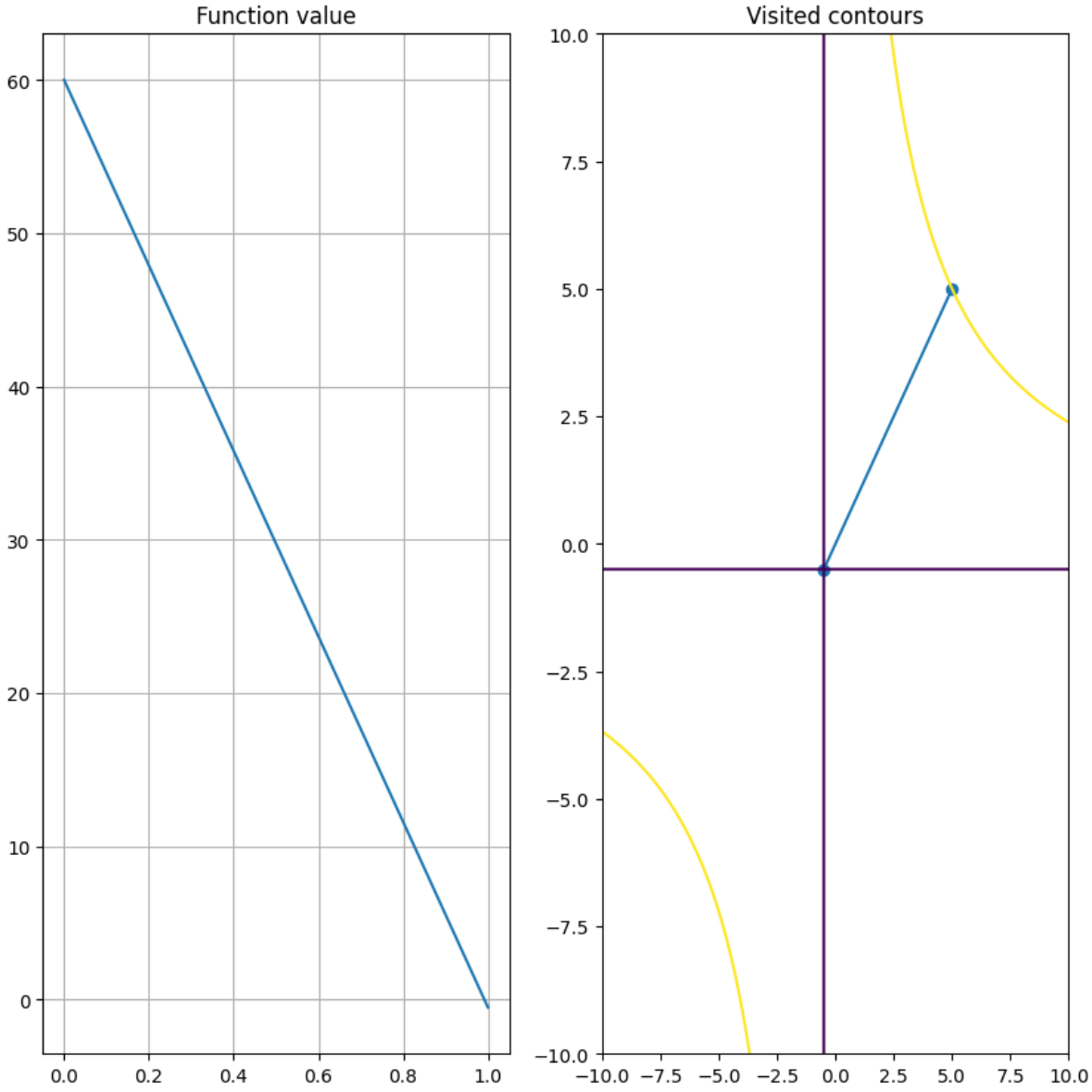
Optimizing with fixed step = 1



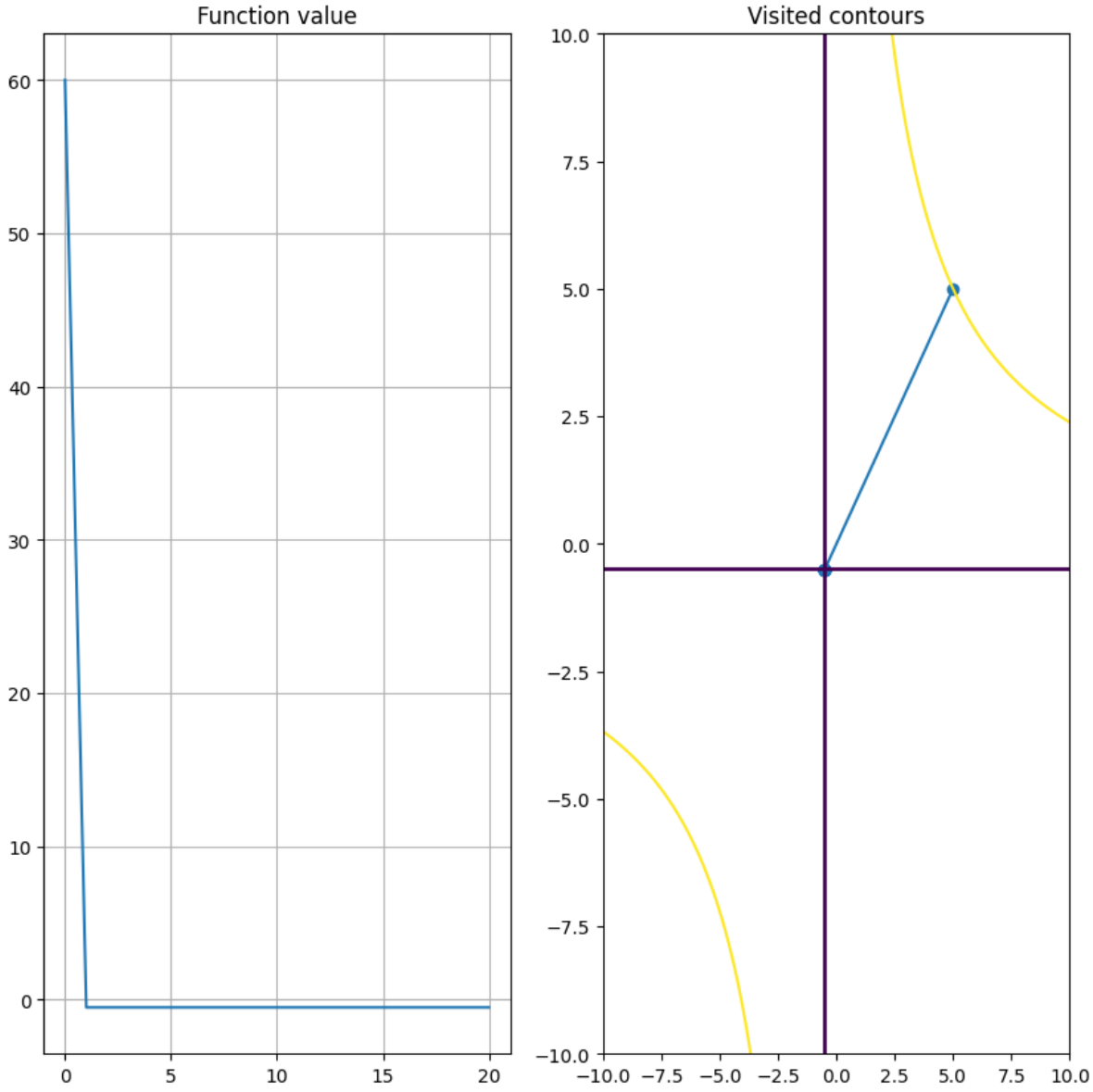
Optimizing with binary search



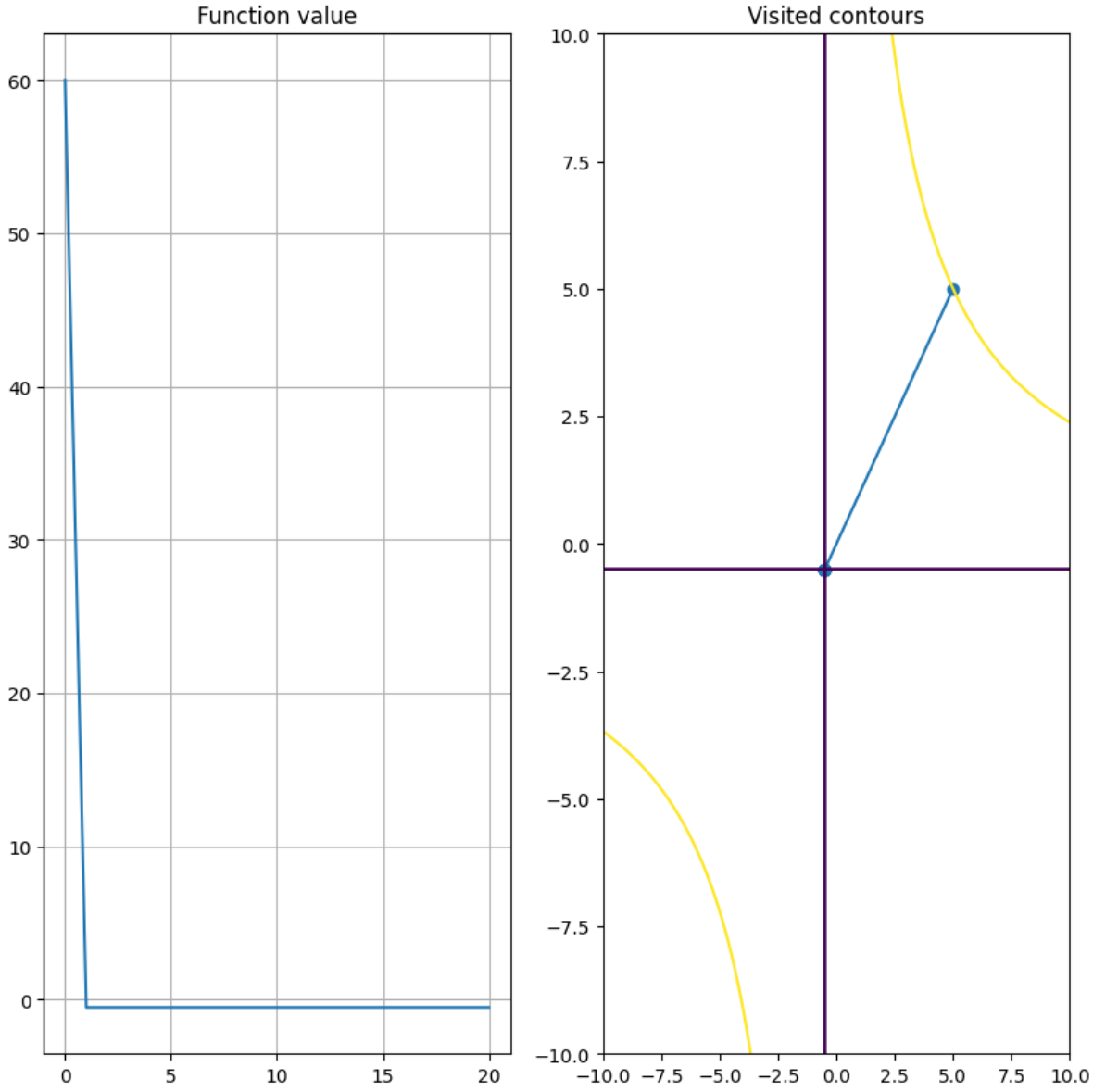
Optimizing with binary search limited by 5 iterations

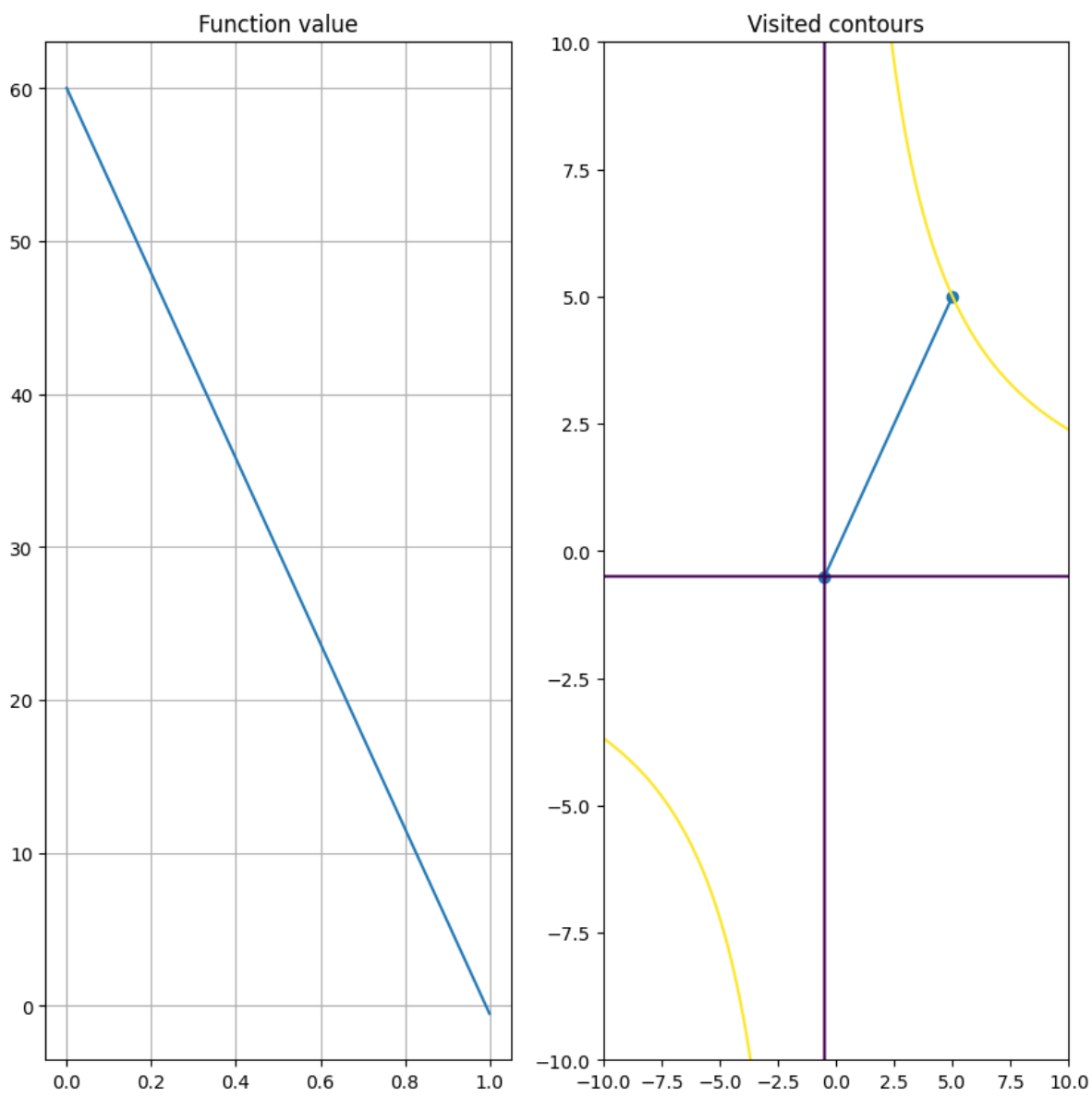


Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





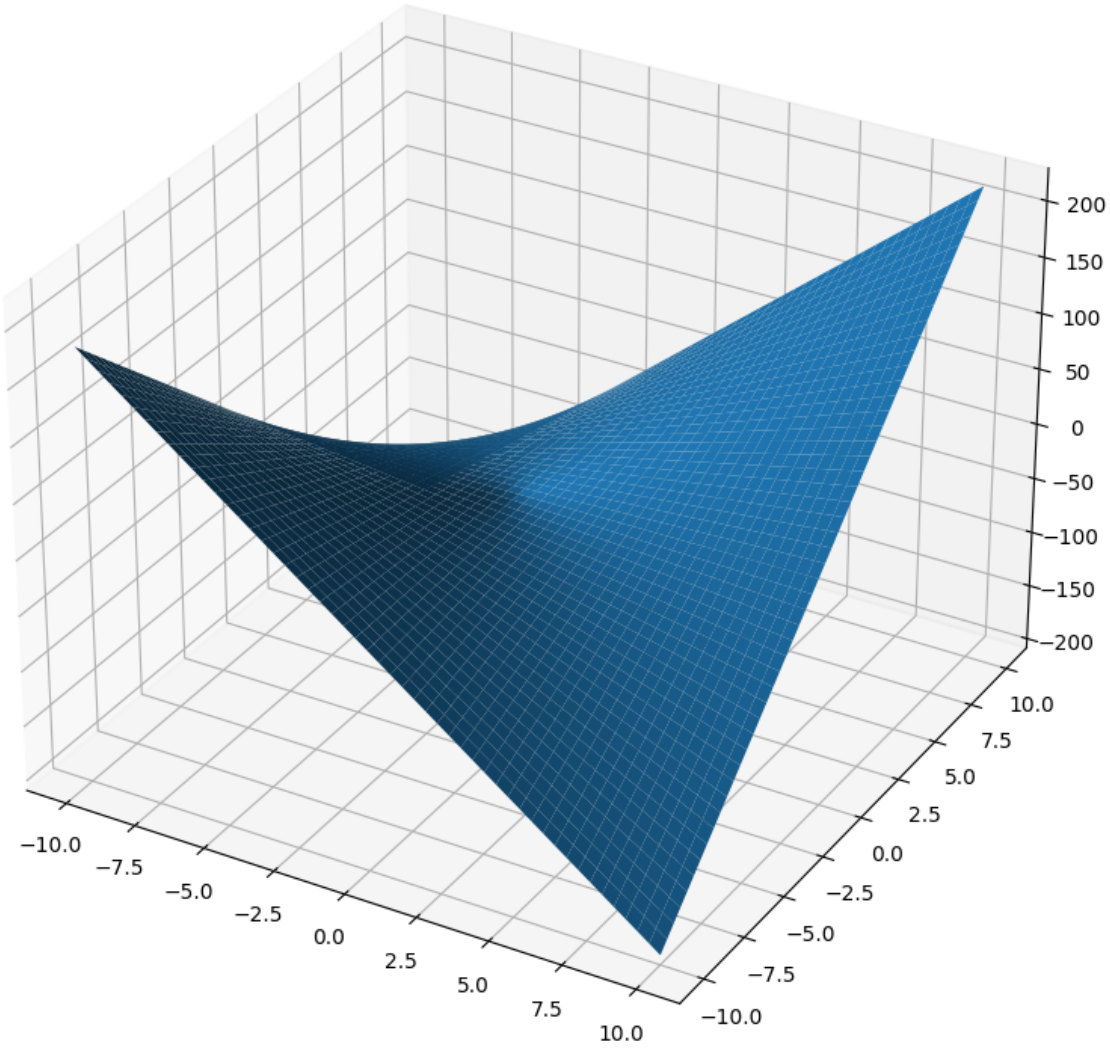
In [13]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.5, 1],
    x0=np.array([-5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=0, b=2, c=0, d=1, e=1
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[  -5.            5.        ]
 [  -6.1           5.9       ]
 [  -7.38          7.02      ]
 [  -8.884         8.396     ]
 [ -10.6632       10.0728    ]
 [ -12.77776      12.10544   ]
 [ -15.298848     14.560992  ]
 [ -18.3110464    17.5207616 ]
 [ -21.91519872   21.08297088]
 [ -26.2317929    25.36601062]
 [ -31.40499502   30.5123692 ]
 [ -37.60746886   36.69336821]
 [ -45.0461425    44.11486198]
 [ -53.9691149    53.02409048]
 [ -64.67393299   63.71791346]
 [ -77.51751569   76.55270006]
 [ -92.9280557    91.9562032 ]
 [-111.41929634  110.44181434]
 [-133.60765921  132.6256736 ]
 [-160.23279393  159.24720544]
 [-192.18223501  191.19376423]]
Best value found: x* = [-192.18223501  191.19376423] with f(x*) = -73489.078331993
02
Optimizing with fixed step = 0.5
Optimizer trajectory:
[[-5.0000000e+00  5.0000000e+00]
 [-1.0500000e+01  9.5000000e+00]
 [-2.0500000e+01  1.9500000e+01]
 [-4.0500000e+01  3.9500000e+01]
 [-8.0500000e+01  7.9500000e+01]
 [-1.6050000e+02  1.5950000e+02]
 [-3.2050000e+02  3.1950000e+02]
 [-6.4050000e+02  6.3950000e+02]
 [-1.2805000e+03  1.2795000e+03]
 [-2.5605000e+03  2.5595000e+03]
 [-5.1205000e+03  5.1195000e+03]
 [-1.0240500e+04  1.0239500e+04]
 [-2.0480500e+04  2.0479500e+04]
 [-4.0960500e+04  4.0959500e+04]
 [-8.1920500e+04  8.1919500e+04]
 [-1.6384050e+05  1.6383950e+05]
 [-3.2768050e+05  3.2767950e+05]
 [-6.5536050e+05  6.5535950e+05]
 [-1.3107205e+06  1.3107195e+06]
 [-2.6214405e+06  2.6214395e+06]
 [-5.2428805e+06  5.2428795e+06]]
Best value found: x* = [-5242880.5  5242879.5] with f(x*) = -54975581388800.5
Optimizing with fixed step = 1
Optimizer trajectory:
[[         -5           5]
 [        -16          14]
 [        -45          45]
 [       -136         134]
```

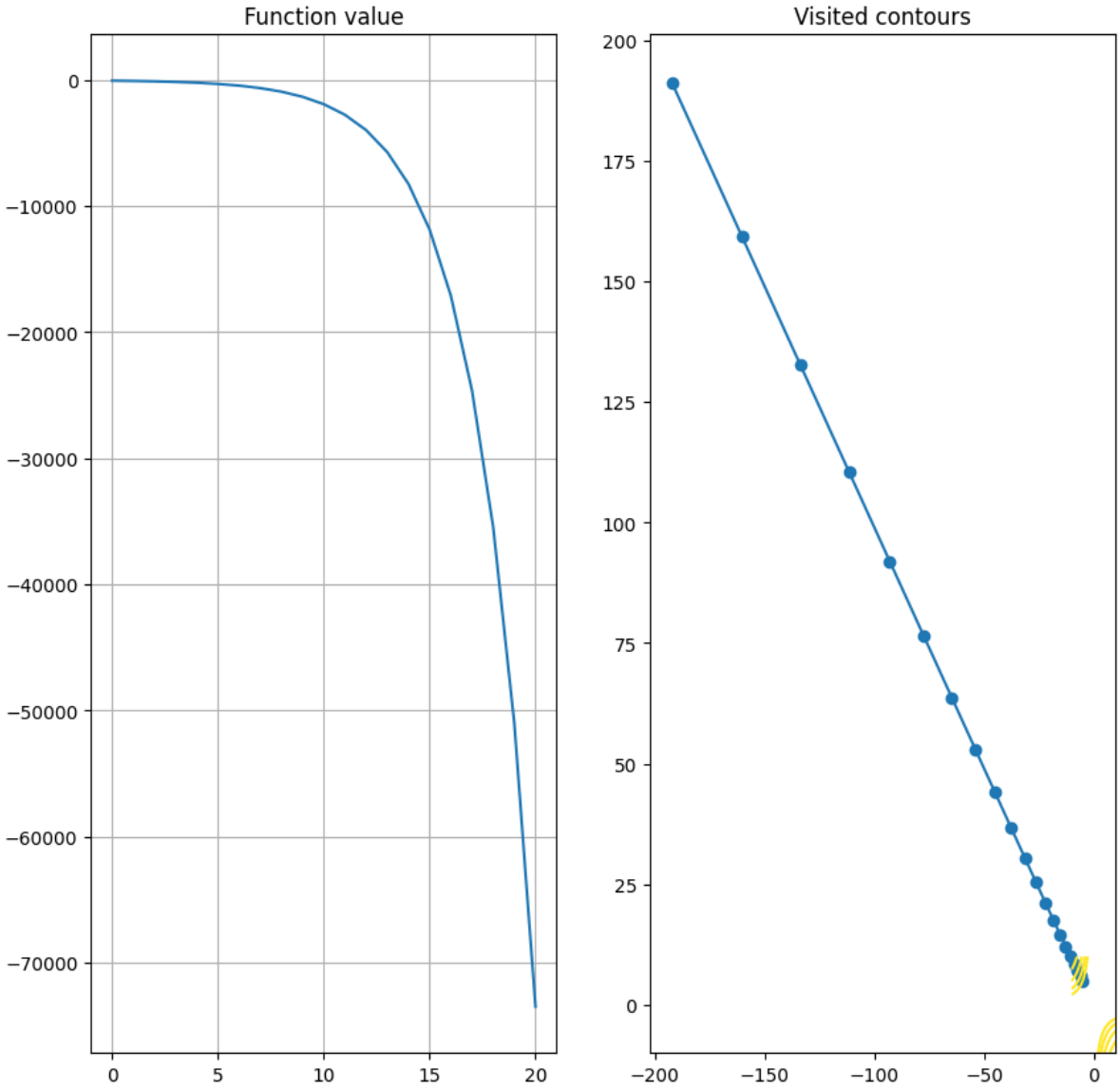
```
 [       -405         405]
 [      -1216        1214]
 [      -3645        3645]
 [     -10936       10934]
 [     -32805       32805]
 [     -98416       98414]
 [    -295245      295245]
 [    -885736      885734]
 [   -2657205     2657205]
 [   -7971616     7971614]
 [  -23914845    23914845]
 [  -71744536    71744534]
 [ -215233605   215233605]
 [ -645700816   645700814]
 [-1937102445  1937102445]
 [-1516340040  1516340038]
 [ -254052821   254052821]]
Best value found: x* = [-254052821  254052821] with f(x*) = -138106482
Optimizing with binary search
```

```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
Cell In[13], line 1
----> 1 analyze_quadratic(roi=SearchRegion2d((-10, 10), (-10, 10)), fixed_steps=[
0.1, 0.5, 1], x0=np.array([-5, 5]), bin_iters=5, fib_iters=30, a=0, b=2, c=0, d=1,
 e=1)

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:37, in analyze
_quadratic(roi, x0, fixed_steps, bin_iters, fib_iters, a, b, c, d, e)
     35 for title, linear in cases:
     36     print(title)
---> 37     visualize_optimizer_with(linear).suptitle(title)

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:22, in analyze
_quadratic.<locals>.visualize_optimizer_with(linear_search)
     19 else:
     20     true_min = None
---> 22 return visualize_optimizing_process(f, roi, np.array(gradient_descent(f, d
f, x0, linear_search, lambda f, points: len(points) > 20)), true_min)

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:22, in gradient_
descent(target_function, gradient_function, x0, linear_search, terminate_conditio
n)
     20     if np.linalg.norm(g) == 0:
     21         return points
---> 22     next_point = last_point - g * linear_search(lambda l: target_function(
last_point - g * l),
     23                                                 lambda l: -np.dot(g, gradi
ent_function(last_point - g * l)))
     24     points.append(next_point)
     25 return points

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:43, in bin_searc
h(f, derivative)
     40 def bin_search(f: Callable[[float], float], derivative: Callable[[float],
float]):
     41     # assume derivative(0) < 0 and derivative is rising
     42     l = 0
---> 43     r = find_upper_bound(f)
     45     while r - l > precision:
     46         m = (l + r) / 2

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:31, in find_uppe
r_bound(f)
     29 original = f(0)
     30 r = 1
---> 31 while f(r) < original:
     32     r *= 2
     33 return r

File ~\DataSpellProjects\GradientDescent\core\gradient_descent.py:22, in gradient_
descent.<locals>.<lambda>(l)
     20     if np.linalg.norm(g) == 0:
     21         return points
---> 22     next_point = last_point - g * linear_search(lambda l: target_function(
last_point - g * l),
```

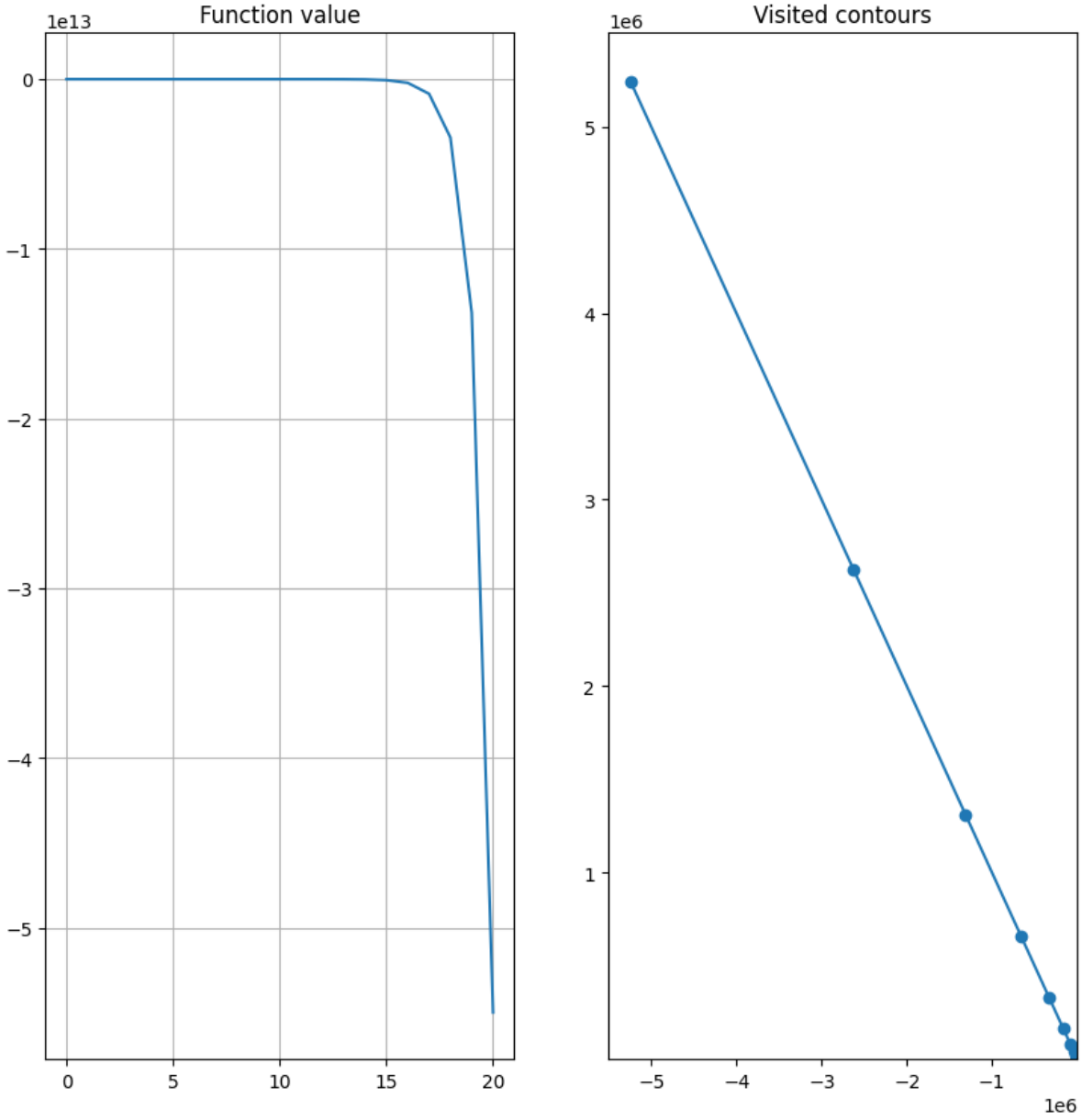
```
     23                                                 lambda l: -np.dot(g, gradi
ent_function(last_point - g * l)))
     24     points.append(next_point)
     25 return points

File ~\DataSpellProjects\GradientDescent\core\quadratic_analyzer.py:6, in create_q
uadratic.<locals>.<lambda>(x)
      5 def create_quadratic(a, b, c, d, e):
----> 6     return lambda x: a * x[0] ** 2 + b * x[0] * x[1] + c * x[1] ** 2 + d *
 x[0] + e * x[1]

OverflowError: (34, 'Result too large')
```

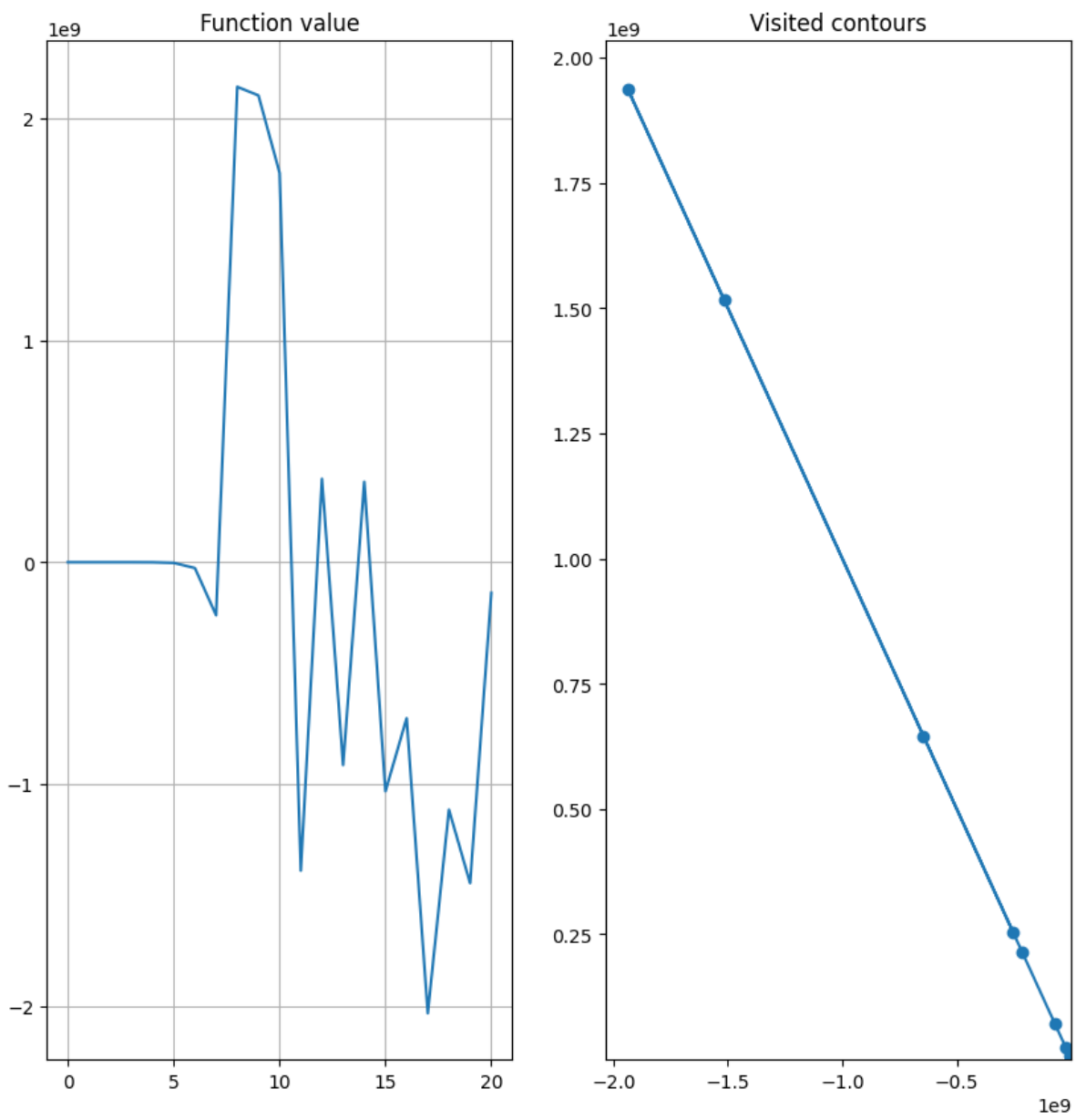


Optimizing with fixed step = 0.1



Optimizing with fixed step = 0.5





Случай параболы в сечении

У функции нет глобального минимума, но есть направление для убывания. В виду метод будет сходиться, кроме случая слишком большого шага.

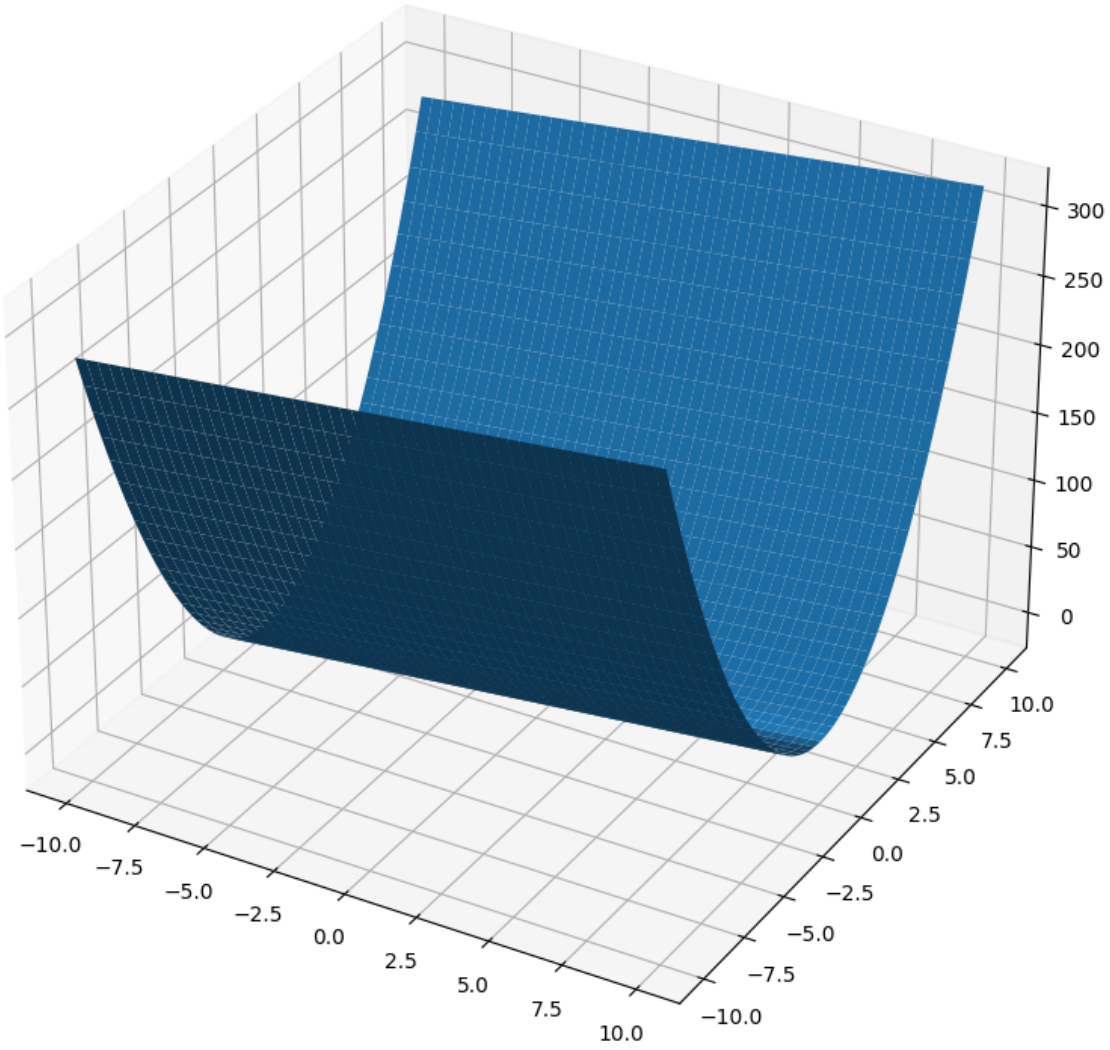
In [14]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.5, 1], x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=0, b=0, c=3, d=2, e=0
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[5.00000000e+00 5.00000000e+00]
 [4.80000000e+00 2.00000000e+00]
 [4.60000000e+00 8.00000000e-01]
 [4.40000000e+00 3.20000000e-01]
 [4.20000000e+00 1.28000000e-01]
 [4.00000000e+00 5.12000000e-02]
 [3.80000000e+00 2.04800000e-02]
 [3.60000000e+00 8.19200000e-03]
 [3.40000000e+00 3.27680000e-03]
 [3.20000000e+00 1.31072000e-03]
 [3.00000000e+00 5.24288000e-04]
 [2.80000000e+00 2.09715200e-04]
 [2.60000000e+00 8.38860800e-05]
 [2.40000000e+00 3.35544320e-05]
 [2.20000000e+00 1.34217728e-05]
 [2.00000000e+00 5.36870912e-06]
 [1.80000000e+00 2.14748365e-06]
 [1.60000000e+00 8.58993459e-07]
 [1.40000000e+00 3.43597384e-07]
 [1.20000000e+00 1.37438953e-07]
 [1.00000000e+00 5.49755814e-08]]
Best value found: x* = [1.00000000e+00 5.49755814e-08] with f(x*) = 2.000000000000
0044
Optimizing with fixed step = 0.5
Optimizer trajectory:
[[ 5.00000e+00  5.00000e+00]
 [ 4.00000e+00 -1.00000e+01]
 [ 3.00000e+00  2.00000e+01]
 [ 2.00000e+00 -4.00000e+01]
 [ 1.00000e+00  8.00000e+01]
 [ 0.00000e+00 -1.60000e+02]
 [-1.00000e+00  3.20000e+02]
 [-2.00000e+00 -6.40000e+02]
 [-3.00000e+00  1.28000e+03]
 [-4.00000e+00 -2.56000e+03]
 [-5.00000e+00  5.12000e+03]
 [-6.00000e+00 -1.02400e+04]
 [-7.00000e+00  2.04800e+04]
 [-8.00000e+00 -4.09600e+04]
 [-9.00000e+00  8.19200e+04]
 [-1.00000e+01 -1.63840e+05]
 [-1.10000e+01  3.27680e+05]
 [-1.20000e+01 -6.55360e+05]
 [-1.30000e+01  1.31072e+06]
 [-1.40000e+01 -2.62144e+06]
 [-1.50000e+01  5.24288e+06]]
Best value found: x* = [-1.50000e+01  5.24288e+06] with f(x*) = 82463372083170.0
Optimizing with fixed step = 1
Optimizer trajectory:
[[          5           5]
 [          3         -25]
 [          1         125]
 [         -1        -625]
```

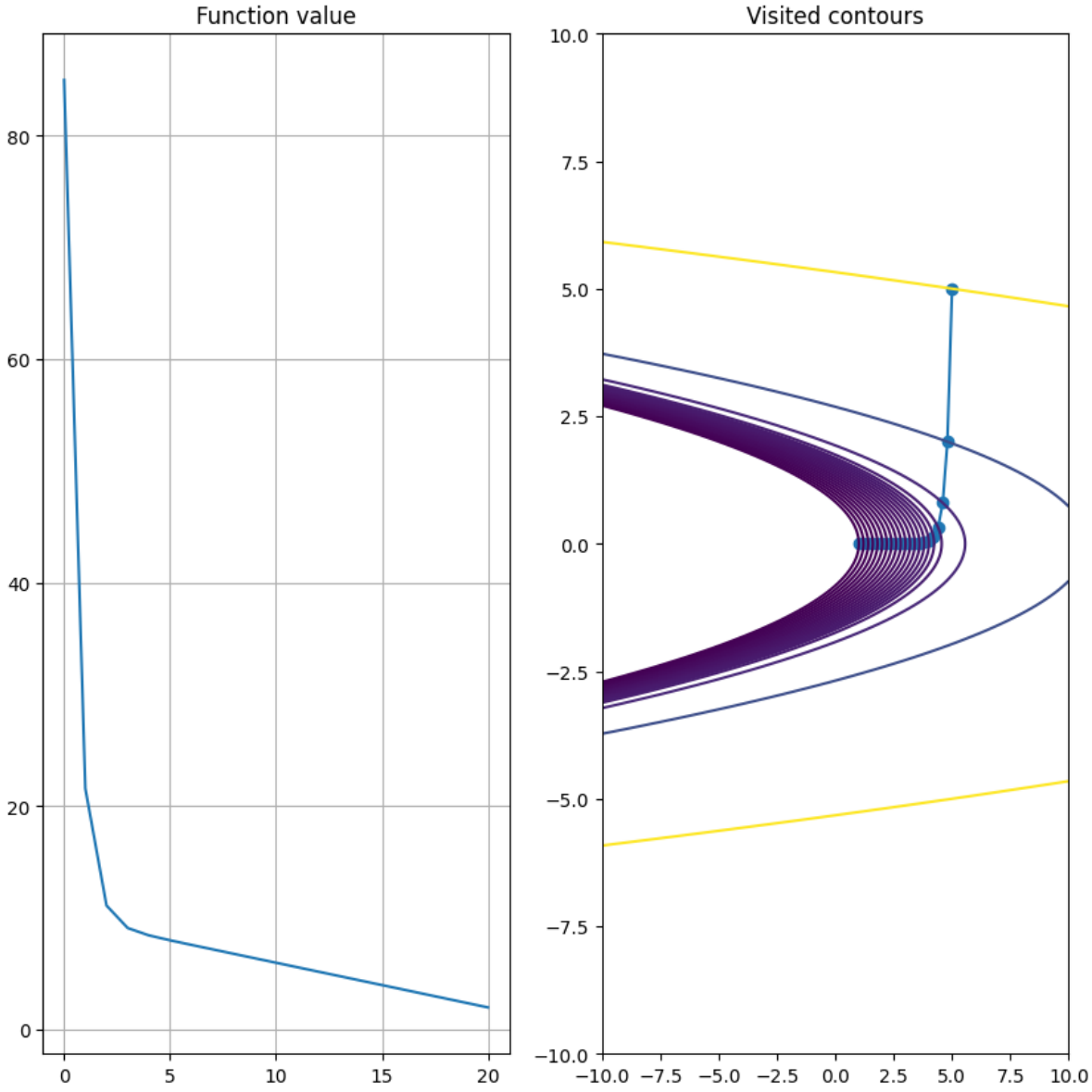
```
 [         -3        3125]
 [         -5      -15625]
 [         -7       78125]
 [         -9     -390625]
 [        -11     1953125]
 [        -13    -9765625]
 [        -15    48828125]
 [        -17  -244140625]
 [        -19  1220703125]
 [        -21 -1808548329]
 [        -23   452807053]
 [        -25  2030932031]
 [        -27 -1564725563]
 [        -29  -766306777]
 [        -31  -463433411]
 [        -33 -1977800241]
 [        -35  1299066613]]
Best value found: x* = [       -35 1299066613] with f(x*) = 1333453605
Optimizing with binary search
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [ 4.66517639e+00 -2.23541260e-02]
 [-6.97856750e+01  4.97049701e+00]
 [-7.01205139e+01 -2.24497546e-02]
 [-1.43941284e+02  4.94932477e+00]
 [-1.44276138e+02 -2.25806901e-02]
 [-2.17247131e+02  4.92062545e+00]
 [-2.17582001e+02 -2.26750013e-02]
 [-2.89950012e+02  4.90015926e+00]
 [-2.90284897e+02 -2.28050014e-02]
 [-3.61833984e+02  4.87222613e+00]
 [-3.62168884e+02 -2.28980354e-02]
 [-4.33140457e+02  4.85243073e+00]
 [-4.33475372e+02 -2.30271295e-02]
 [-5.03657135e+02  4.82522648e+00]
 [-5.03992065e+02 -2.31189135e-02]
 [-5.73620331e+02  4.80607062e+00]
 [-5.73955276e+02 -2.32471372e-02]
 [-6.42821228e+02  4.77956154e+00]
 [-6.43156189e+02 -2.33377028e-02]
 [-7.11491272e+02  4.76101388e+00]]
Best value found: x* = [-711.49127197    4.76101388] with f(x*) = -1354.9807844441
236
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.          5.        ]
 [ 4.625      -0.625     ]
 [ 4.1875      0.1953125 ]
 [ 2.8125     -0.61035156]
 [ 2.375       0.19073486]
 [ 1.         -0.59604645]
 [ 0.5         0.29802322]
 [-0.3125     -0.42840838]
 [-0.875       0.29453076]
 [-1.6875     -0.42338797]
 [-2.25        0.29107923]
```

```
 [-3.0625     -0.4184264 ]
 [-3.625       0.28766815]
 [-4.4375     -0.41352296]
 [-5.          0.28429704]
 [-5.8125     -0.40867699]
 [-6.375       0.28096543]
 [-7.1875     -0.40388781]
 [-7.75        0.27767287]
 [-8.625      -0.45121841]
 [-9.1875      0.31021266]]
Best value found: x* = [-9.1875      0.31021266] with f(x*) = -18.08630432511025
Optimizing with golden ration
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [ 4.66517240e+00 -2.24140708e-02]
 [-6.93897713e+01  4.95720418e+00]
 [-6.97246182e+01 -2.25090948e-02]
 [-1.43158432e+02  4.93627697e+00]
 [-1.43493299e+02 -2.26997296e-02]
 [-2.15704452e+02  4.89482123e+00]
 [-2.16039337e+02 -2.27923529e-02]
 [-2.87667483e+02  4.87492954e+00]
 [-2.88002380e+02 -2.28740814e-02]
 [-3.59121969e+02  4.85751172e+00]
 [-3.59456878e+02 -2.29660835e-02]
 [-4.30010468e+02  4.83805279e+00]
 [-4.30345396e+02 -2.31540571e-02]
 [-4.99763463e+02  4.79877560e+00]
 [-5.00098411e+02 -2.32437851e-02]
 [-5.68984131e+02  4.78025079e+00]
 [-5.69319098e+02 -2.34306861e-02]
 [-6.37115527e+02  4.74211989e+00]
 [-6.37450518e+02 -2.35829906e-02]
 [-7.04378379e+02  4.71149435e+00]]
Best value found: x* = [-704.37837893    4.71149435] with f(x*) = -1342.1622209752
757
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [ 4.66518430e+00 -2.22355265e-02]
 [-7.05786429e+01  4.99702281e+00]
 [-7.09134616e+01 -2.22668277e-02]
 [-1.45946787e+02  4.98999556e+00]
 [-1.46281609e+02 -2.22799925e-02]
 [-2.21226702e+02  4.98704838e+00]
 [-2.21561527e+02 -2.23112858e-02]
 [-2.96297449e+02  4.98005230e+00]
 [-2.96632277e+02 -2.23243762e-02]
 [-3.71280919e+02  4.97712870e+00]
 [-3.71615749e+02 -2.23556342e-02]
 [-4.46056741e+02  4.97017112e+00]
 [-4.46391575e+02 -2.23686847e-02]
 [-5.20746236e+02  4.96727927e+00]
 [-5.21081073e+02 -2.23999457e-02]
 [-5.95229226e+02  4.96034386e+00]
 [-5.95564066e+02 -2.24128846e-02]
```

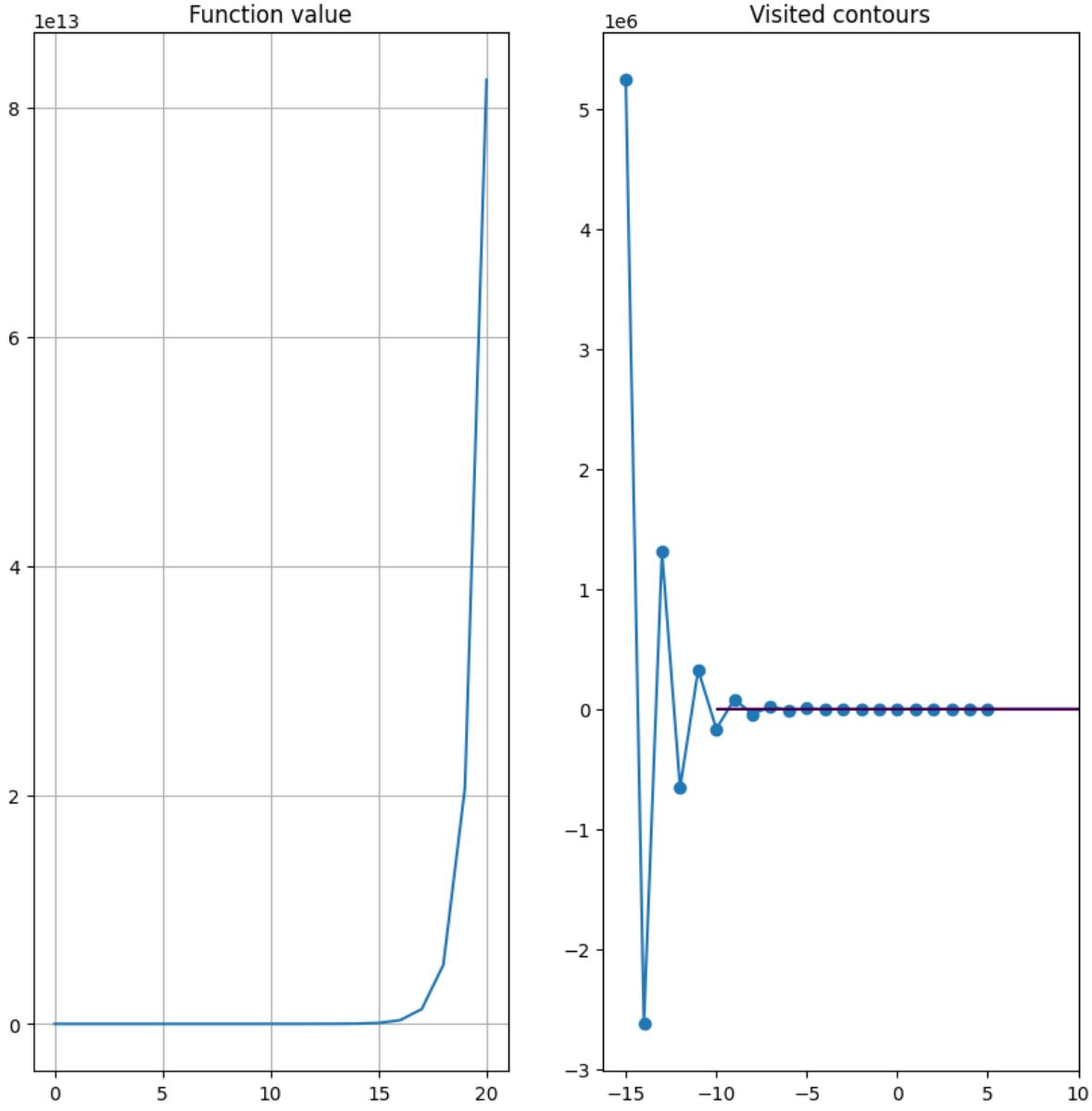
```
 [-6.69627029e+02  4.95748108e+00]
 [-6.69961872e+02 -2.24441380e-02]
 [-7.43819658e+02  4.95057890e+00]]
Best value found: x* = [-743.81965788    4.9505789 ] with f(x*) = -1414.1146215161
357
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.      5.    ]
 [ 4.5    -2.5   ]
 [ 4.      1.25  ]
 [ 3.5    -0.625 ]
 [ 3.      0.3125]
 [ 2.     -0.625 ]
 [ 1.5     0.3125]
 [ 0.5    -0.625 ]
 [ 0.      0.3125]
 [-1.     -0.625 ]
 [-1.5     0.3125]
 [-2.5    -0.625 ]
 [-3.      0.3125]
 [-4.     -0.625 ]
 [-4.5     0.3125]
 [-5.5    -0.625 ]
 [-6.      0.3125]
 [-7.     -0.625 ]
 [-7.5     0.3125]
 [-8.5    -0.625 ]
 [-9.      0.3125]]
Best value found: x* = [-9.      0.3125] with f(x*) = -17.70703125
```


300
250
200
150
100
50
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0

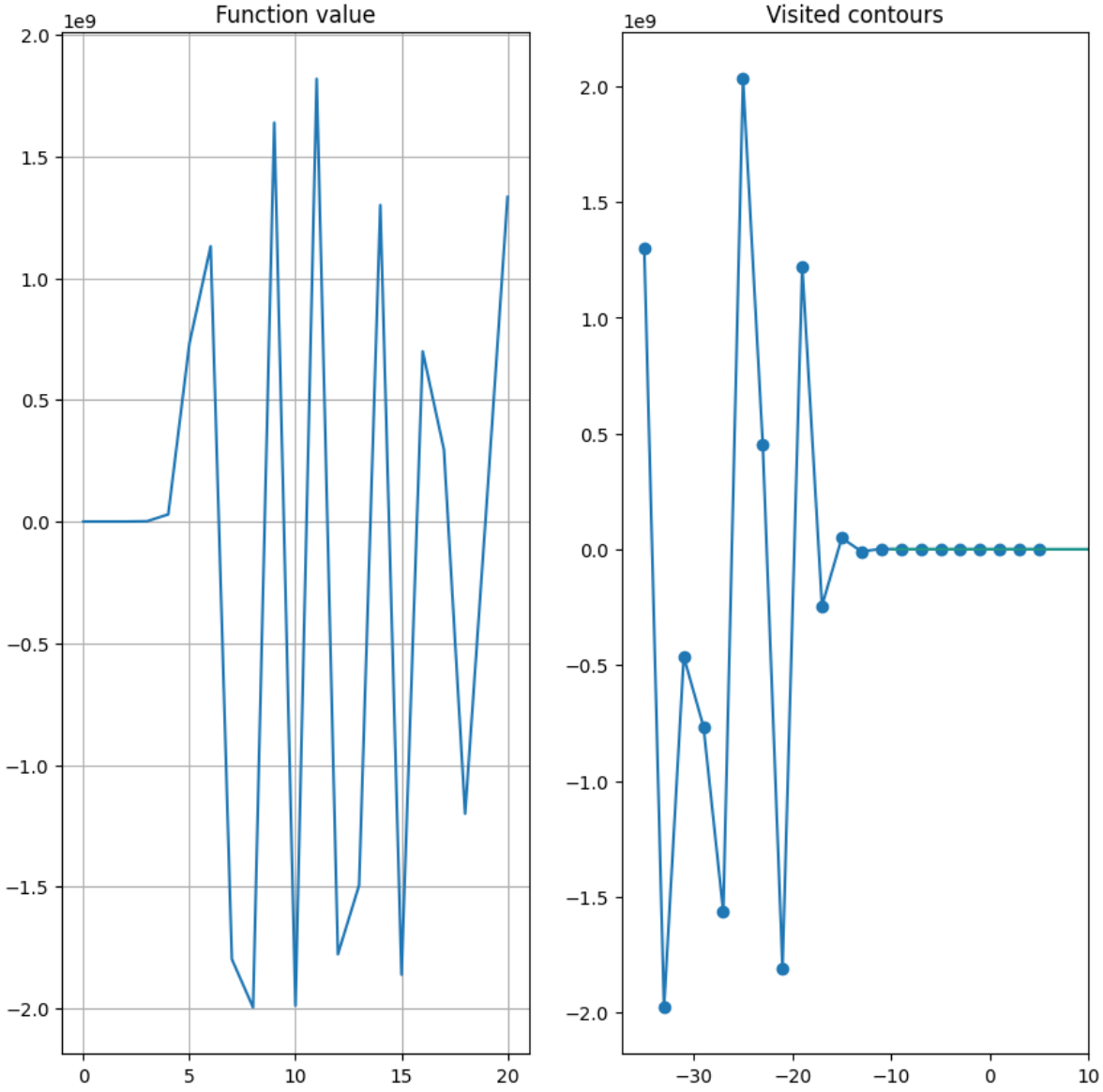
Optimizing with fixed step = 0.1



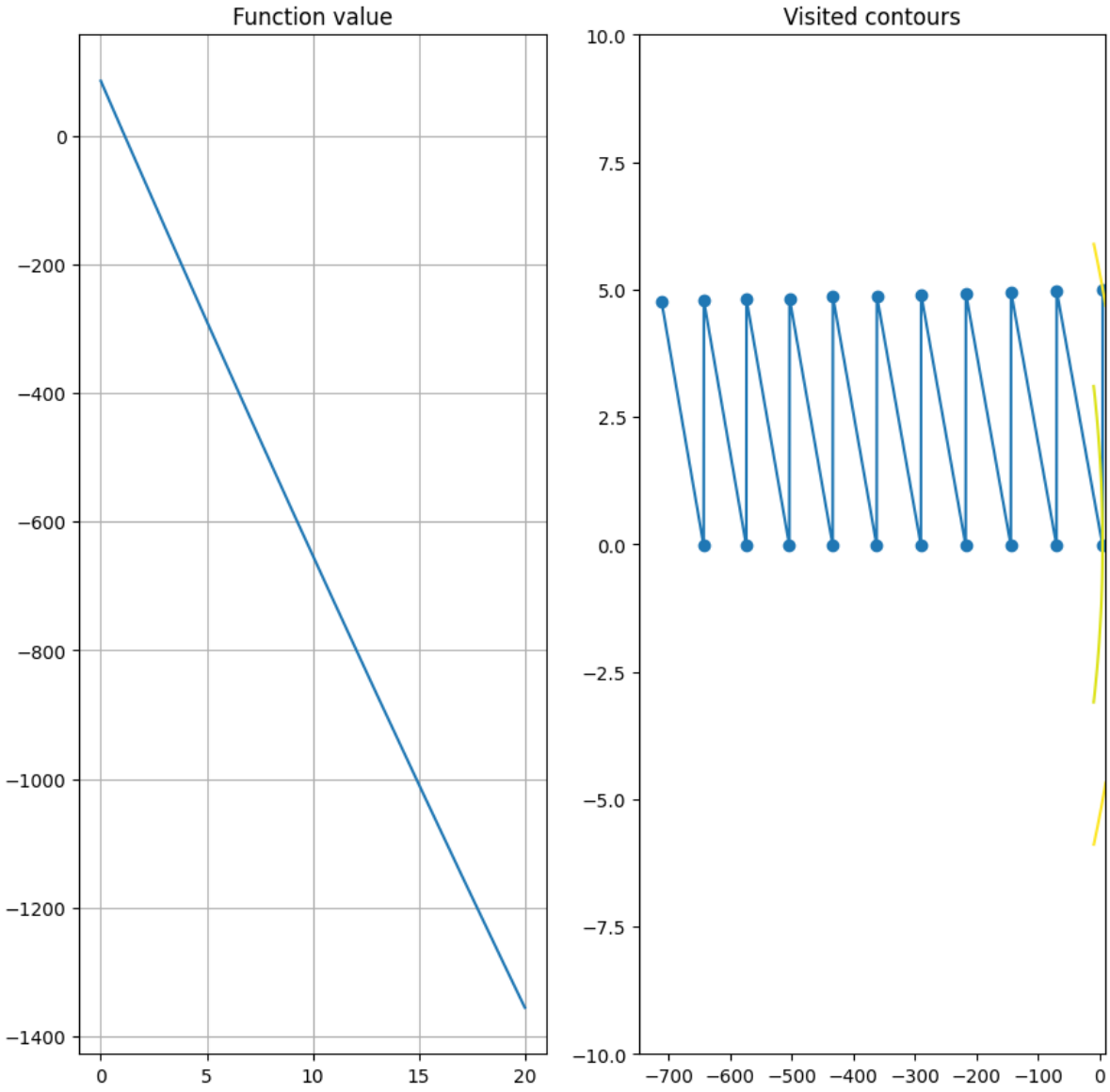
Optimizing with fixed step = 0.5



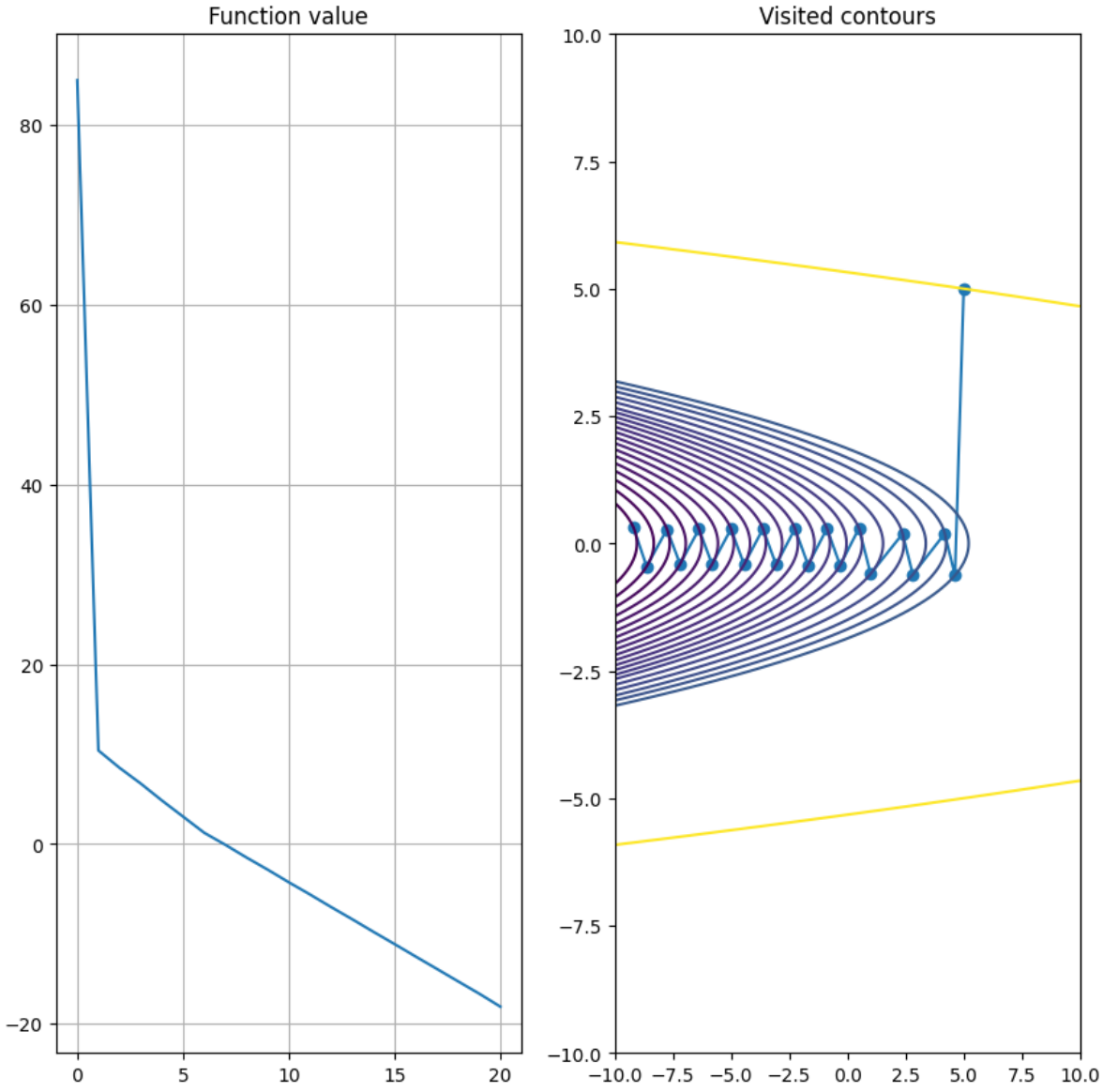
Optimizing with fixed step = 1



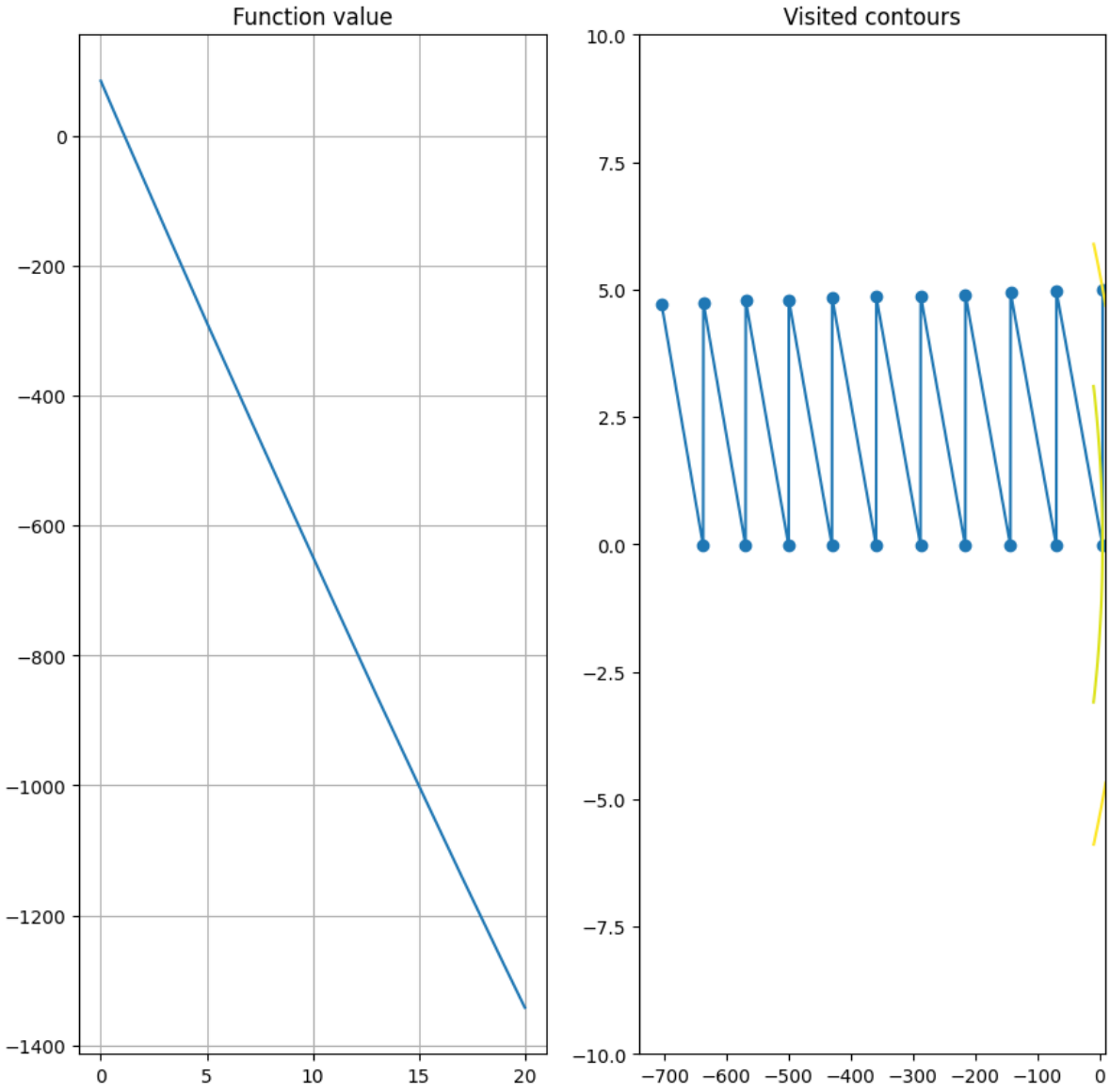
Optimizing with binary search



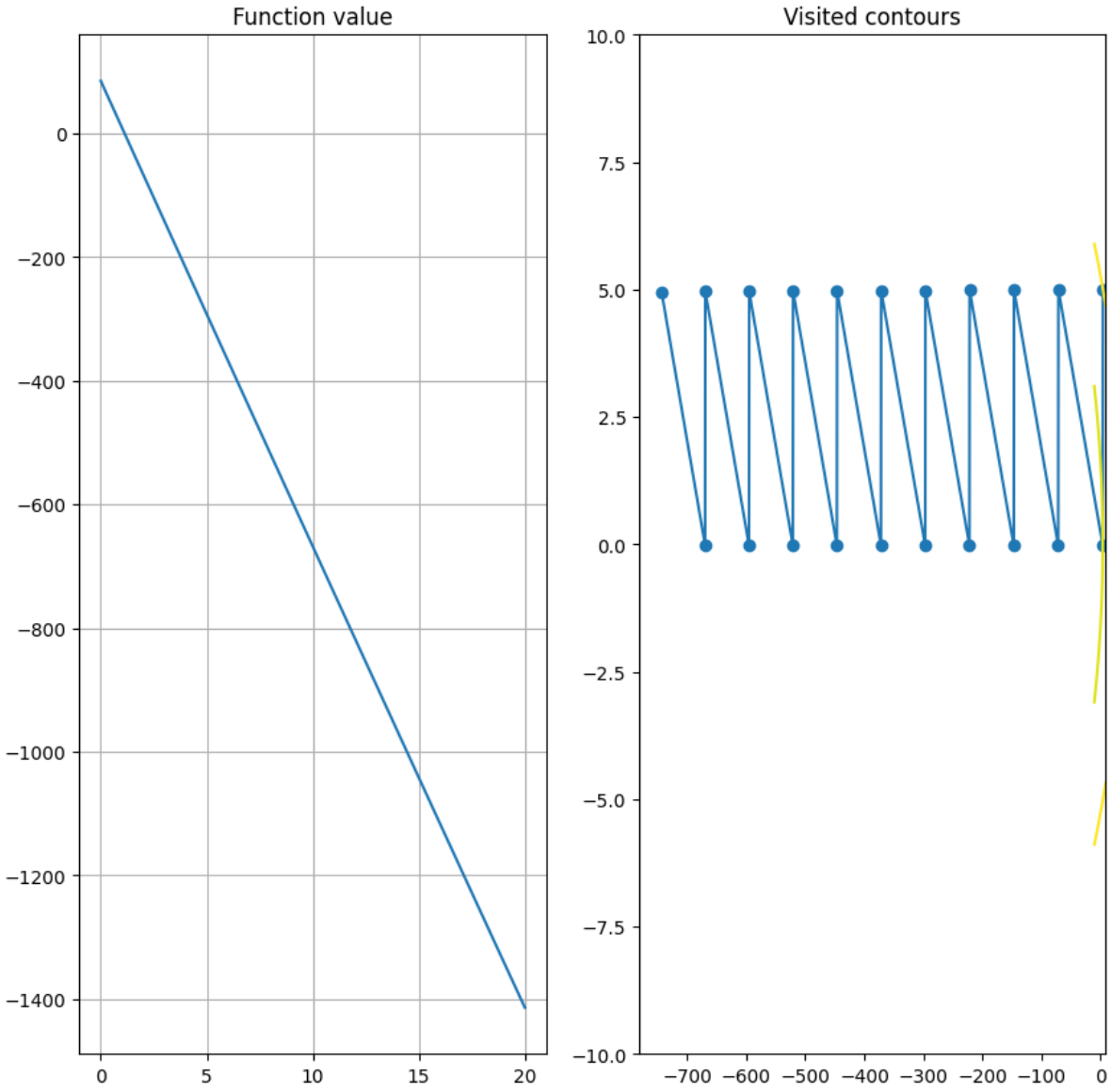
Optimizing with binary search limited by 5 iterations

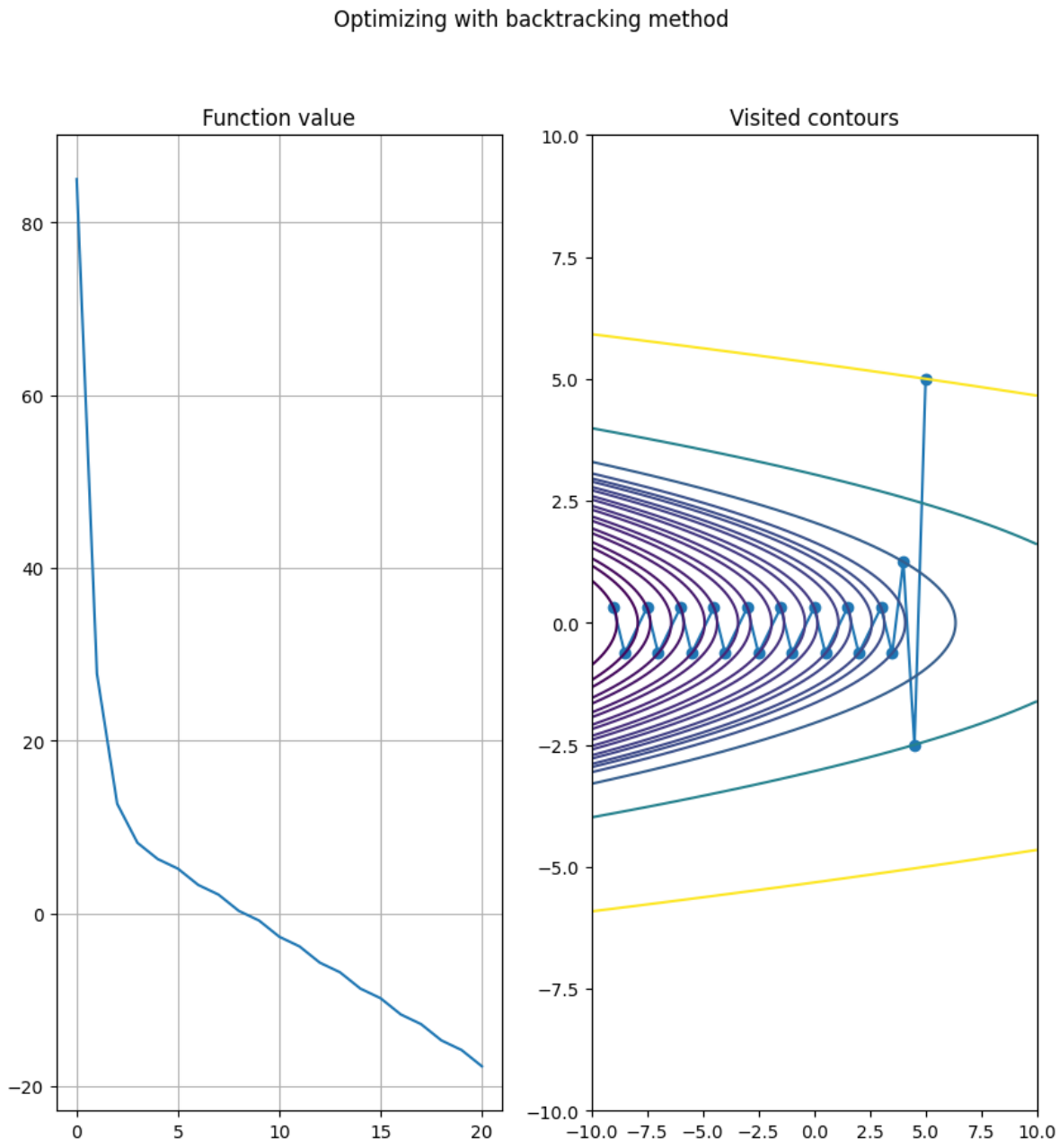


Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





Анализ преобразований над квадратичными формами

Возможные преобразования:

1. Добавление константы -- минимум меняется на ту же константу, траектория поиска никак не меняется.
2. Сдвиг координат -- траектория не меняется.
3. Равномерное растяжение координат -- траектория не меняется.
4. Неравномерное растяжение координат -- траектория не меняется.

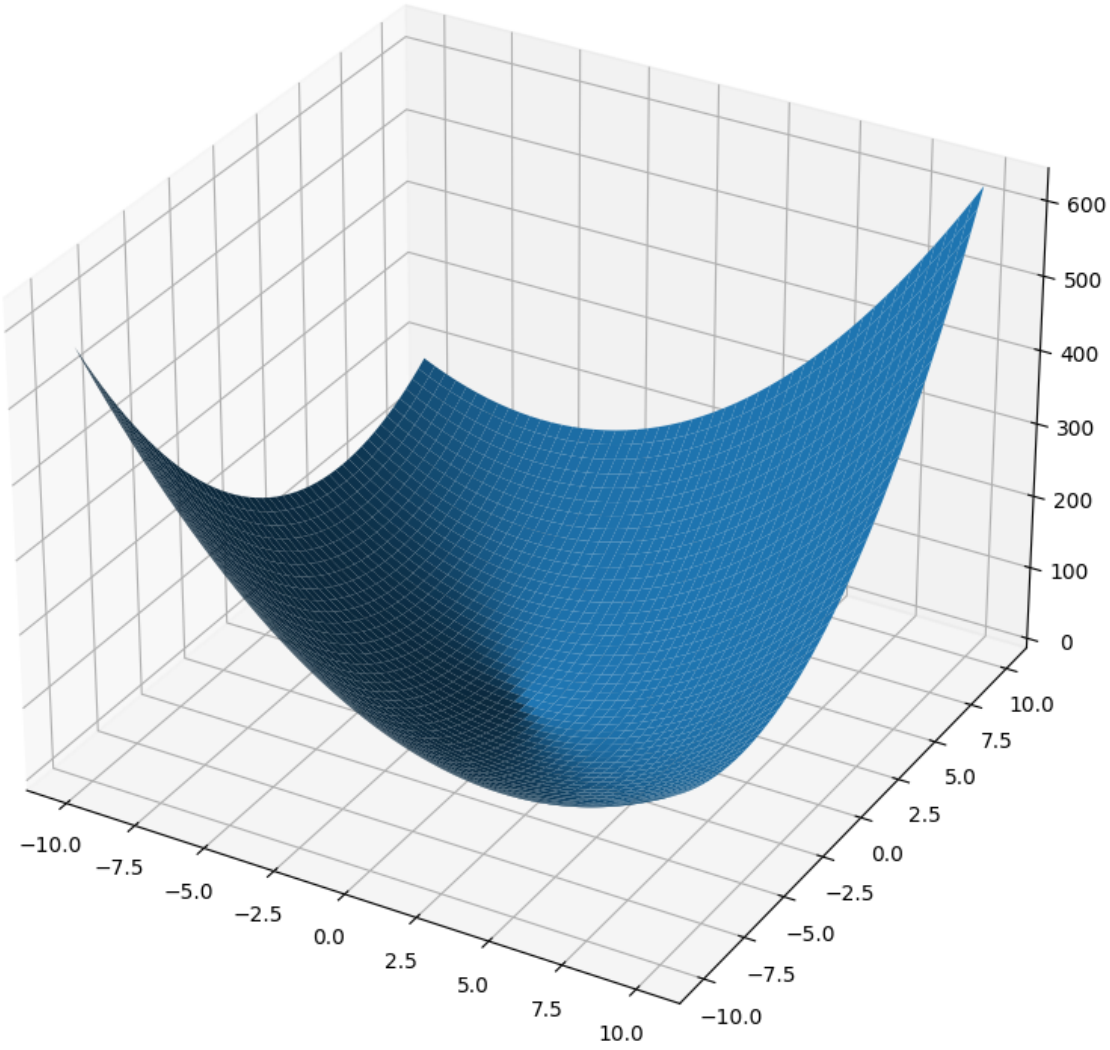
Сдвиг координат

In [15]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.3],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=2, b=2, c=2, d=2, e=1
)
```

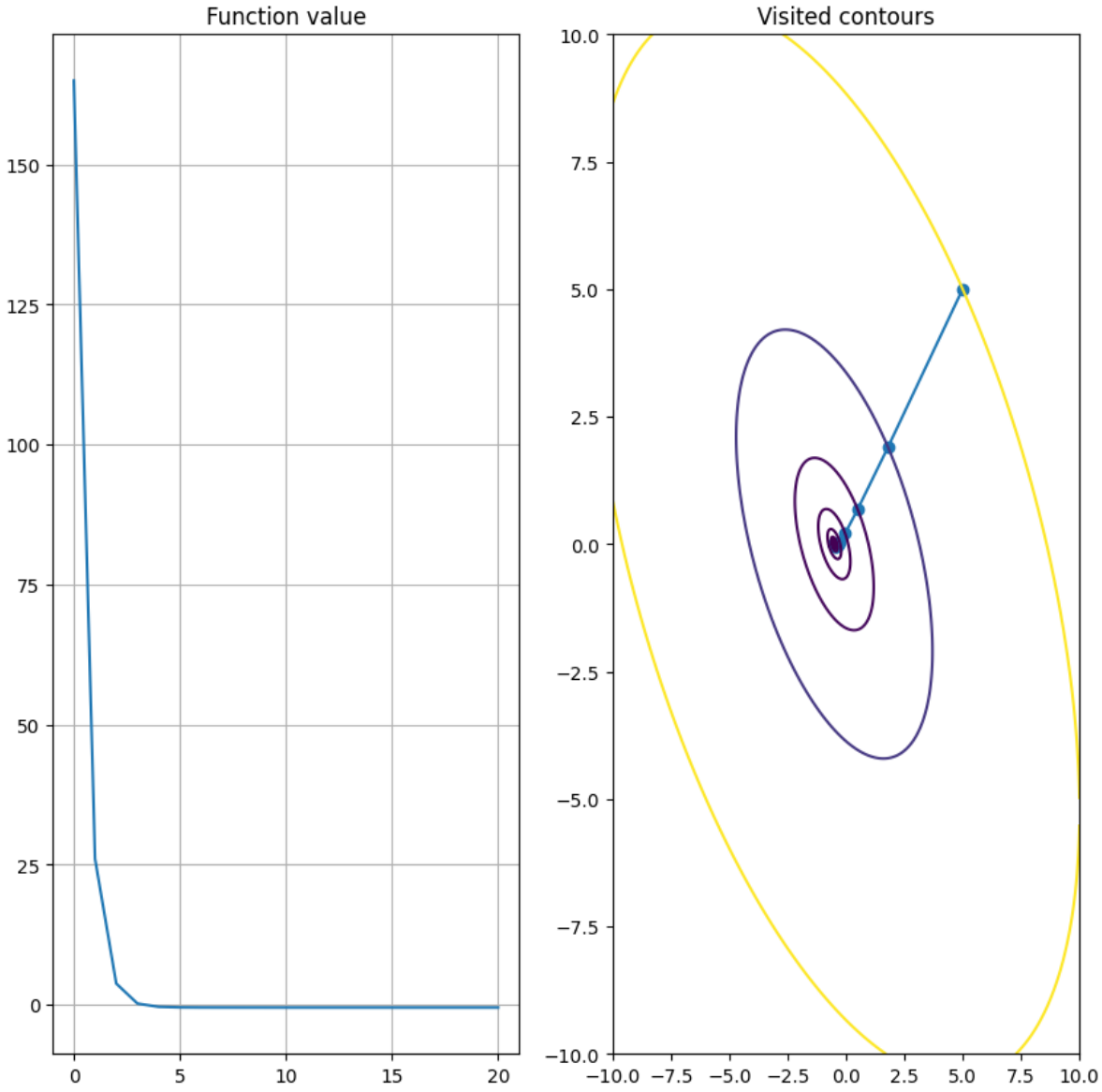
```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [ 1.80000000e+00  1.90000000e+00]
 [ 5.00000000e-01  6.80000000e-01]
 [-3.60000000e-02  2.08000000e-01]
 [-2.63200000e-01  3.20000000e-02]
 [-3.64320000e-01 -2.81600000e-02]
 [-4.12960000e-01 -4.40320000e-02]
 [-4.38969600e-01 -4.38272000e-02]
 [-4.54616320e-01 -3.85024000e-02]
 [-4.65069312e-01 -3.21781760e-02]
 [-4.72605952e-01 -2.62930432e-02]
 [-4.78304963e-01 -2.12546355e-02]
 [-4.82732050e-01 -1.70917888e-02]
 [-4.86220872e-01 -1.37086632e-02]
 [-4.88990791e-01 -1.09810234e-02]
 [-4.91198270e-01 -8.79045588e-03]
 [-4.92960871e-01 -7.03461956e-03]
 [-4.94369599e-01 -5.62859759e-03]
 [-4.95496040e-01 -4.50323885e-03]
 [-4.96396976e-01 -3.60273539e-03]
 [-4.97117639e-01 -2.88224604e-03]]
Best value found: x* = [-0.49711764 -0.00288225] with f(x*) = -0.49998338465003267
Optimizing with fixed step = 0.3
Optimizer trajectory:
[[ 5.          5.        ]
 [-4.6        -4.3       ]
 [ 2.9         3.32      ]
 [-3.172      -2.704     ]
 [ 1.6568      2.144     ]
 [-2.21776    -1.72288   ]
 [ 0.87728     1.375232  ]
 [-1.6005952  -1.1014144 ]
 [ 0.38096768  0.88064   ]
 [-1.20457754 -0.70470861]
 [ 0.06374067  0.56368824]
 [-0.95096108 -0.45098205]
 [-0.13921855  0.36077306]
 [-0.78862012 -0.28862348]
 [-0.26910189  0.23089677]
 [-0.68471769 -0.18471822]
 [-0.35222553  0.14777426]
 [-0.61821945 -0.11821953]
 [-0.40542439  0.09457557]
 [-0.57566047 -0.07566048]
 [-0.43947162  0.06052838]]
Best value found: x* = [-0.43947162  0.06052838] with f(x*) = -0.4780178920197069
Optimizing with binary search
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-3.34472656e-01 -1.67770386e-01]
 [-4.97624907e-01  2.10554944e-03]
 [-4.99910850e-01 -9.05176976e-05]
 [-4.99998542e-01  1.27204369e-06]
```

```
 [-4.99999938e-01 -6.26092710e-08]
 [-4.99999999e-01  8.90381007e-10]
 [-5.00000000e-01 -4.46122379e-11]
 [-5.00000000e-01  6.99596608e-13]
 [-5.00000000e-01 -3.90284042e-14]
 [-5.00000000e-01  6.52687167e-16]
 [-5.00000000e-01 -5.55289389e-17]
 [-5.00000000e-01 -5.55103042e-17]
 [-5.00000000e-01 -5.55103042e-17]
 [-5.00000000e-01 -2.77547286e-17]]
Best value found: x* = [-5.00000000e-01 -2.77547286e-17] with f(x*) = -0.5
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.00000000e+00 -8.12500000e-01]
 [-3.20312500e-01 -1.56250000e-02]
 [-4.49218750e-01 -7.12890625e-02]
 [-4.71923828e-01 -2.44140625e-03]
 [-4.92065430e-01 -1.11389160e-02]
 [-4.95613098e-01 -3.81469727e-04]
 [-4.98760223e-01 -1.74045563e-03]
 [-4.99499321e-01  5.00679016e-04]
 [-5.00250340e-01 -2.50339508e-04]
 [-4.99499321e-01  5.00679016e-04]
 [-5.00250340e-01 -2.50339508e-04]
 [-4.99499321e-01  5.00679016e-04]
 [-5.00250340e-01 -2.50339508e-04]
 [-4.99499321e-01  5.00679016e-04]
 [-5.00250340e-01 -2.50339508e-04]
 [-4.99499321e-01  5.00679016e-04]
 [-5.00250340e-01 -2.50339508e-04]
 [-4.99499321e-01  5.00679016e-04]]
Best value found: x* = [-0.49949932  0.00050068] with f(x*) = -0.49999849592313694
Optimizing with golden ration
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-3.34398550e-01 -1.67698595e-01]
 [-4.97788088e-01  1.97796051e-03]
 [-4.99922566e-01 -7.85135908e-05]
 [-4.99998854e-01  1.00947529e-06]
 [-4.99999956e-01 -4.75864323e-08]
 [-4.99999994e-01  1.19341705e-09]
 [-5.00000003e-01 -4.60941188e-09]
 [-4.99999997e-01  2.59860858e-09]
 [-5.00000004e-01 -3.96997528e-09]
 [-4.99999999e-01  7.71886794e-10]
 [-5.00000002e-01 -1.90626298e-09]
 [-4.99999996e-01  3.61104961e-09]
 [-5.00000005e-01 -4.66567464e-09]
 [-4.99999996e-01  3.65971683e-09]
 [-5.00000005e-01 -4.84800064e-09]
 [-4.99999997e-01  2.90400578e-09]
 [-5.00000005e-01 -5.10320680e-09]
 [-4.99999998e-01  1.92173159e-09]
```

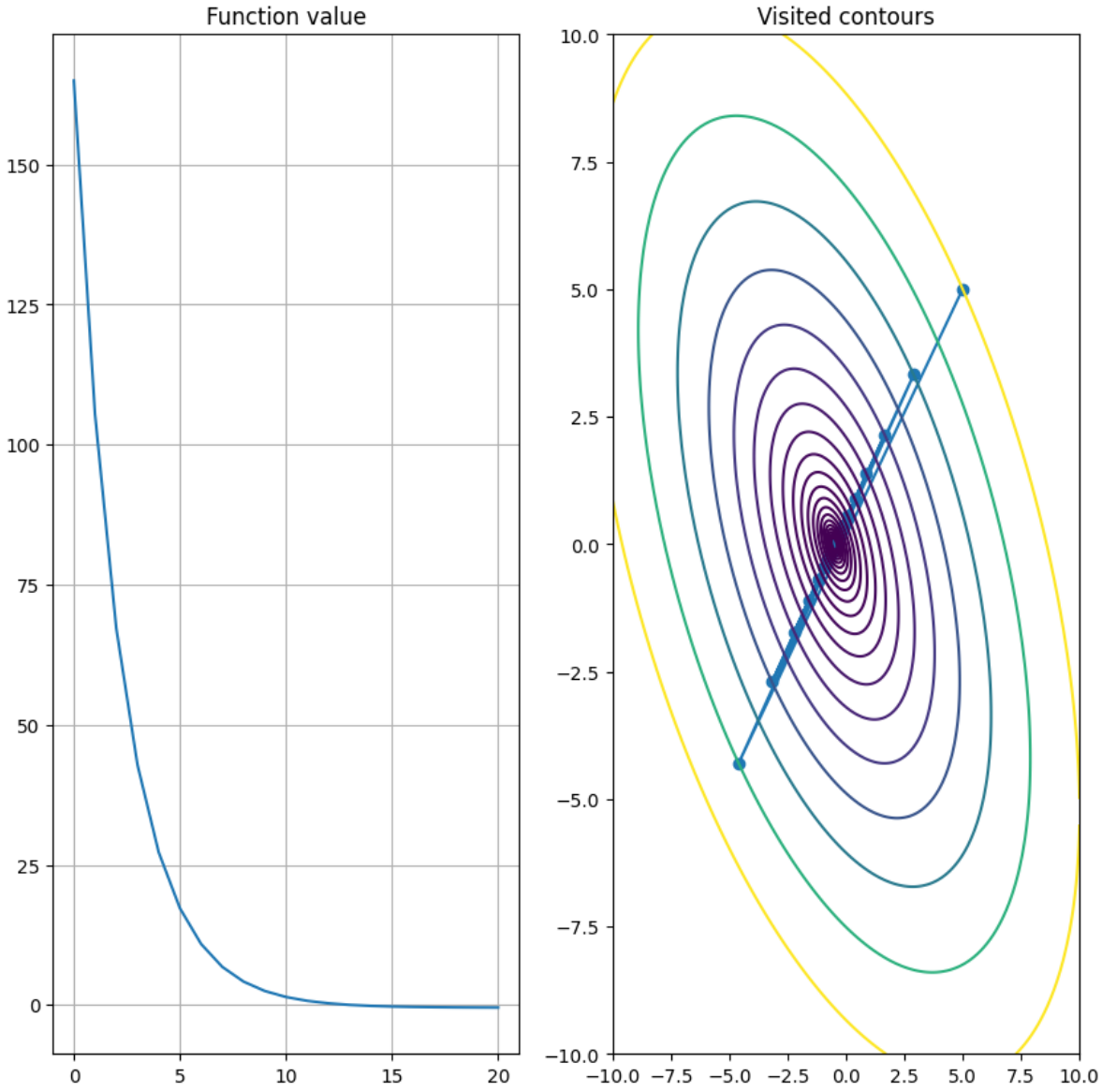
```
 [-5.00000004e-01 -3.63810254e-09]
 [-4.99999996e-01  3.71269818e-09]]
Best value found: x* = [-4.99999996e-01  3.71269818e-09] with f(x*) = -0.499999999
99999994
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-3.34256378e-01 -1.67560866e-01]
 [-4.98095451e-01  1.72703664e-03]
 [-4.99941165e-01 -5.94958314e-05]
 [-4.99999307e-01  6.27870546e-07]
 [-4.99999976e-01 -1.89734117e-08]
 [-4.99999997e-01 -9.04689356e-09]
 [-4.99999997e-01 -6.29476099e-09]
 [-4.99999997e-01 -5.34599208e-09]
 [-4.99999997e-01 -3.77168770e-09]
 [-5.00000000e-01  1.74819919e-09]
 [-5.00000000e-01  1.54893944e-09]
 [-5.00000001e-01 -7.16120057e-10]
 [-4.99999998e-01  1.91300175e-09]
 [-5.00000002e-01 -2.33862366e-09]
 [-4.99999999e-01  9.66294075e-10]
 [-5.00000000e-01  2.43489968e-10]
 [-5.00000003e-01 -2.84238400e-09]
 [-5.00000002e-01 -2.19184973e-09]
 [-5.00000002e-01 -2.06372645e-09]
 [-5.00000001e-01 -1.35691457e-09]]
Best value found: x* = [-5.00000001e-01 -1.35691457e-09] with f(x*) = -0.5
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-3.00000000e+00 -2.75000000e+00]
 [ 8.75000000e-01  1.25000000e+00]
 [-1.12500000e+00 -6.87500000e-01]
 [-1.56250000e-01  3.12500000e-01]
 [-6.56250000e-01 -1.71875000e-01]
 [-4.14062500e-01  7.81250000e-02]
 [-5.39062500e-01 -4.29687500e-02]
 [-4.78515625e-01  1.95312500e-02]
 [-5.09765625e-01 -1.07421875e-02]
 [-4.94628906e-01  4.88281250e-03]
 [-5.02441406e-01 -2.68554688e-03]
 [-4.98657227e-01  1.22070312e-03]
 [-5.00610352e-01 -6.71386719e-04]
 [-4.99664307e-01  3.05175781e-04]
 [-5.00152588e-01 -1.67846680e-04]
 [-4.99916077e-01  7.62939453e-05]
 [-5.00038147e-01 -4.19616699e-05]
 [-4.99979019e-01  1.90734863e-05]
 [-5.00009537e-01 -1.04904175e-05]
 [-4.99994755e-01  4.76837158e-06]]
Best value found: x* = [-4.99994755e-01  4.76837158e-06] with f(x*) = -0.499999999
8494786
```


600
500
400
300
200
100
0
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

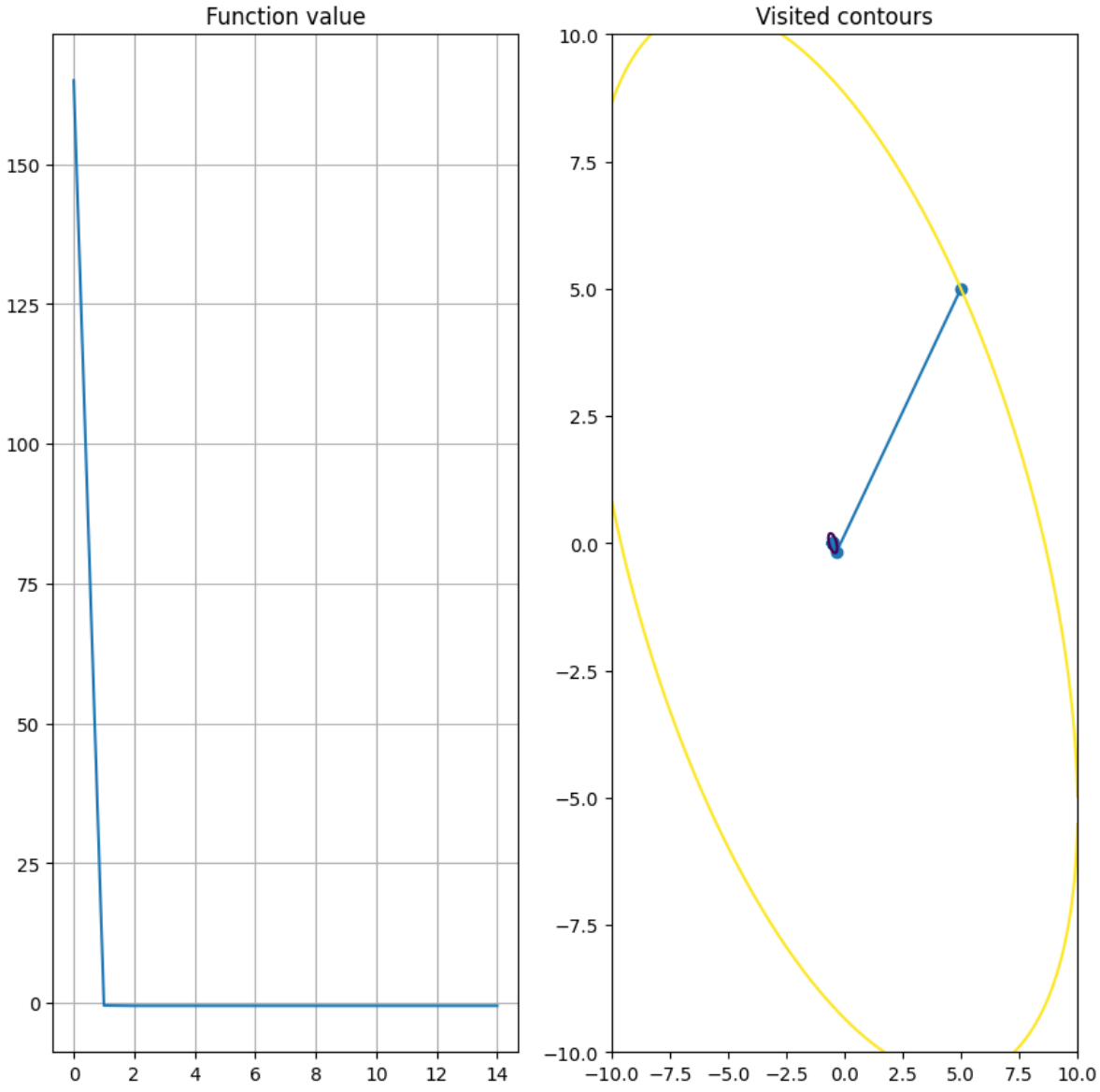
Optimizing with fixed step = 0.1



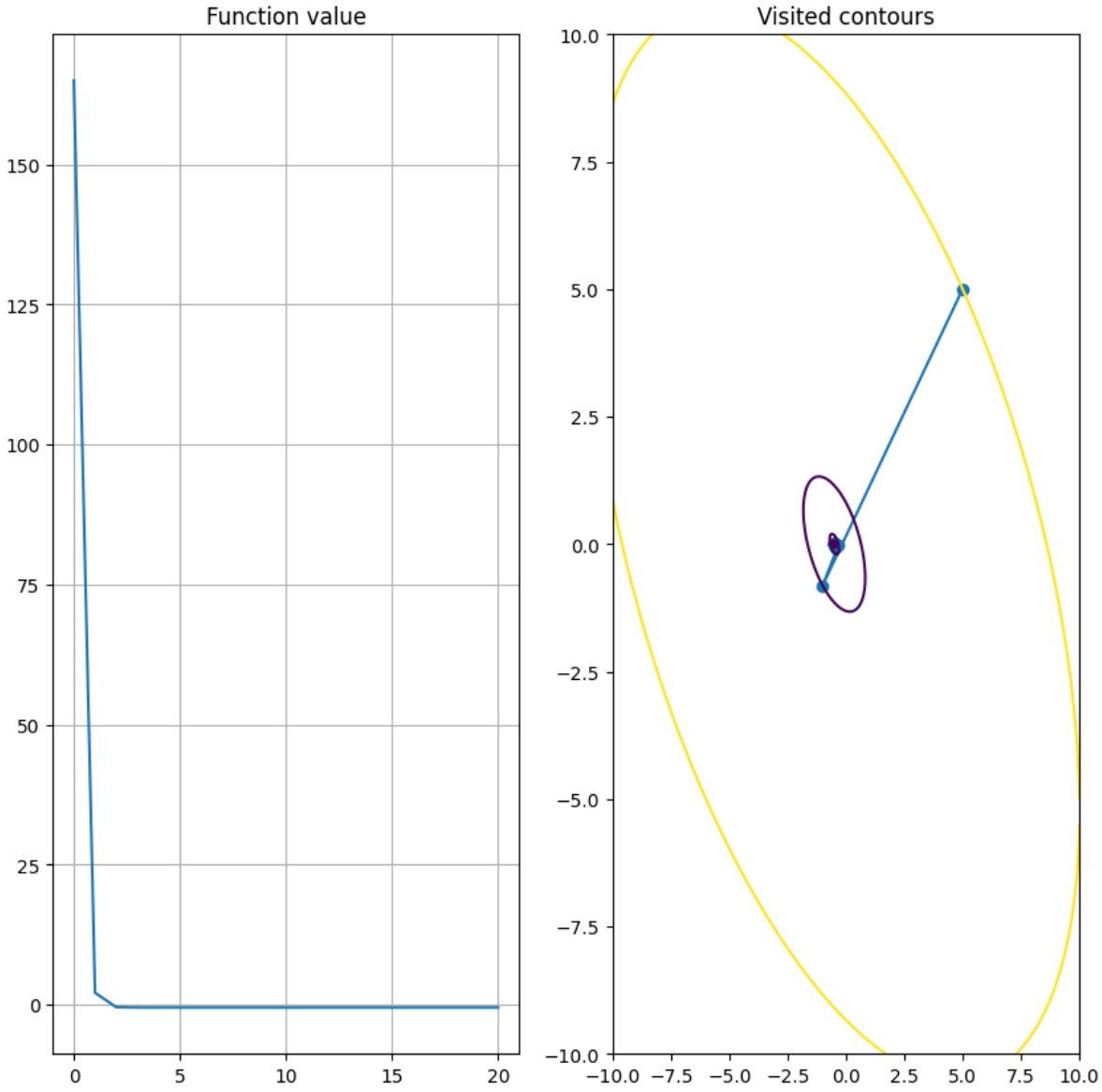
Optimizing with fixed step = 0.3



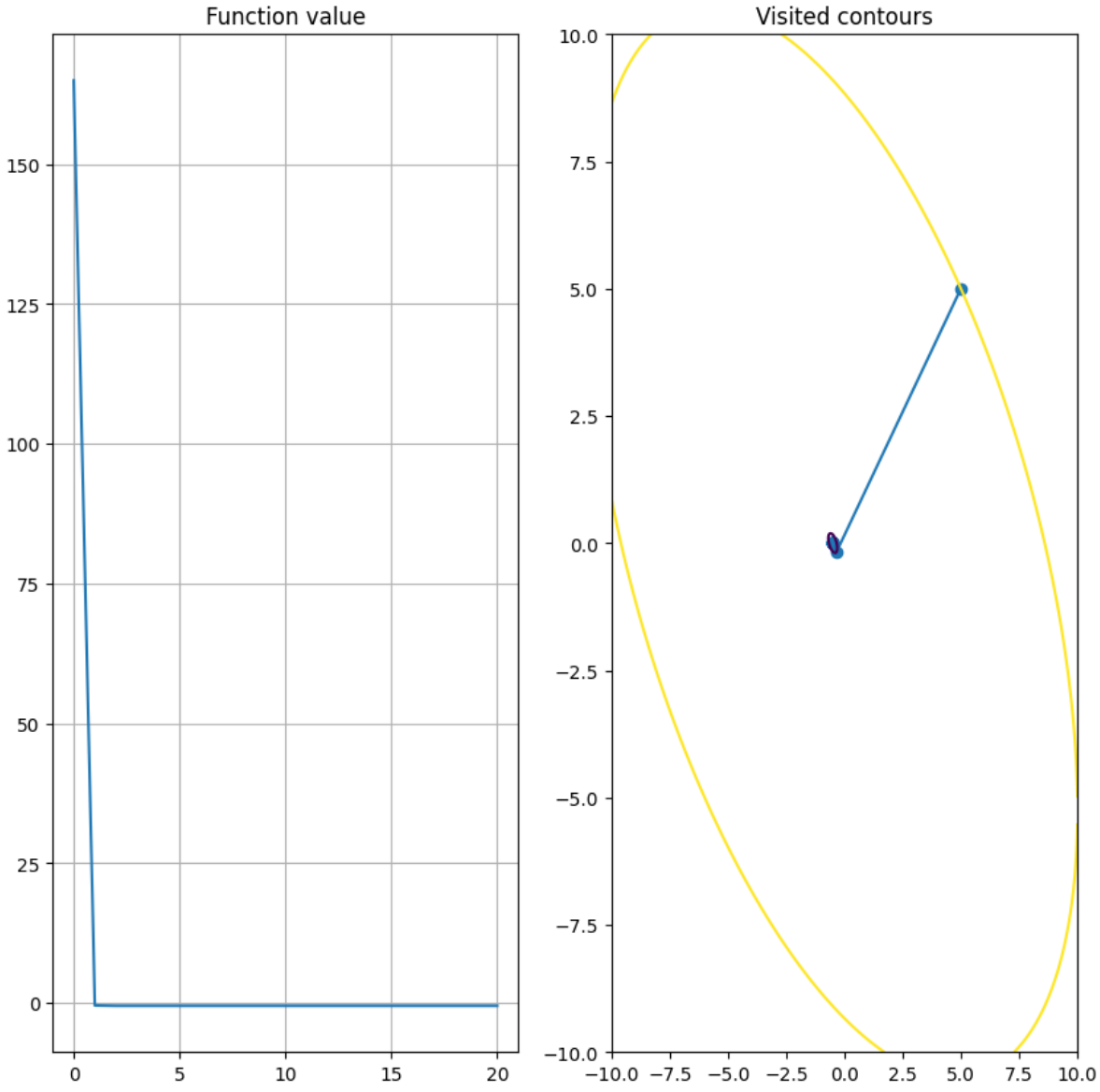
Optimizing with binary search



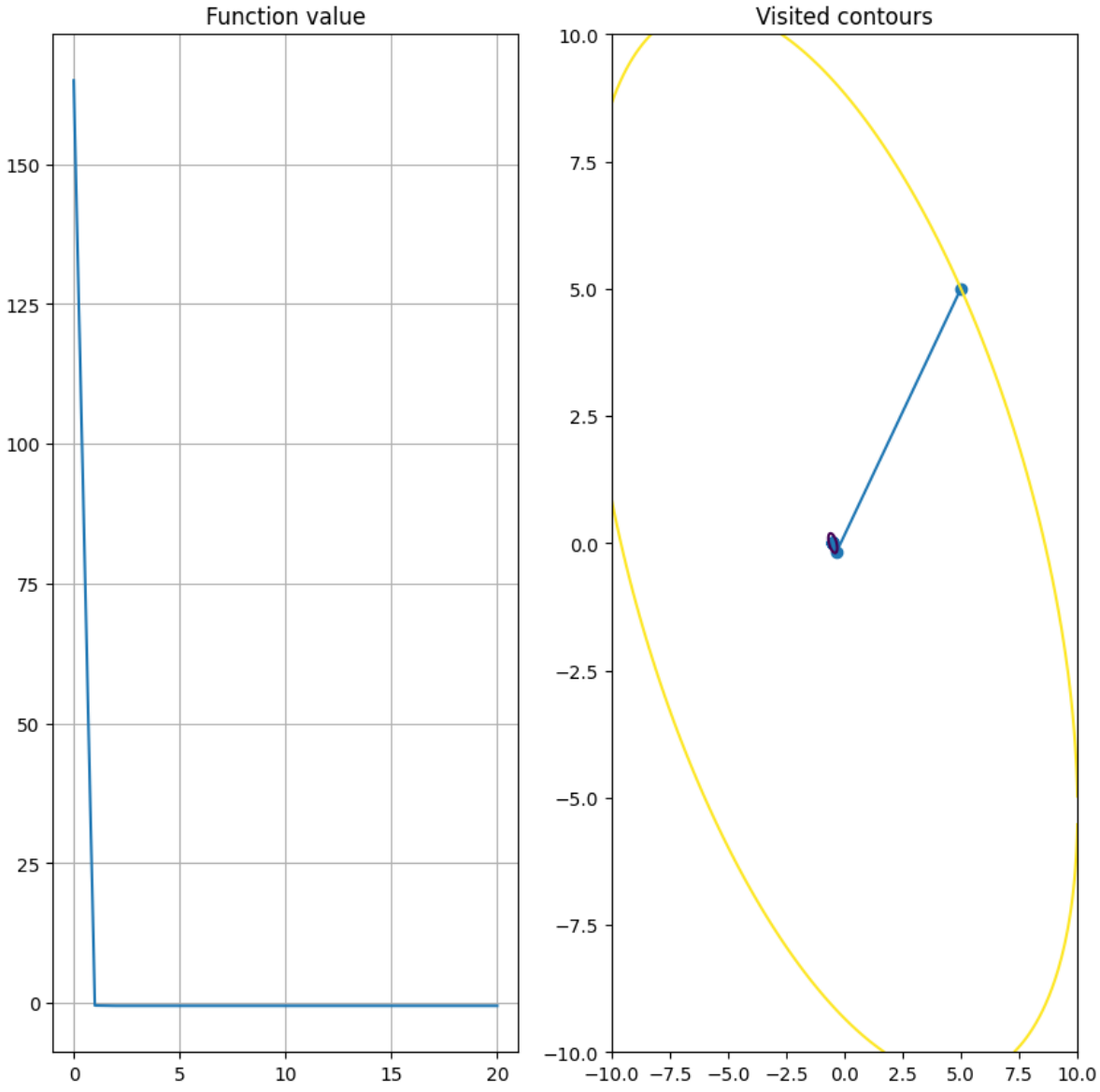
Optimizing with binary search limited by 5 iterations

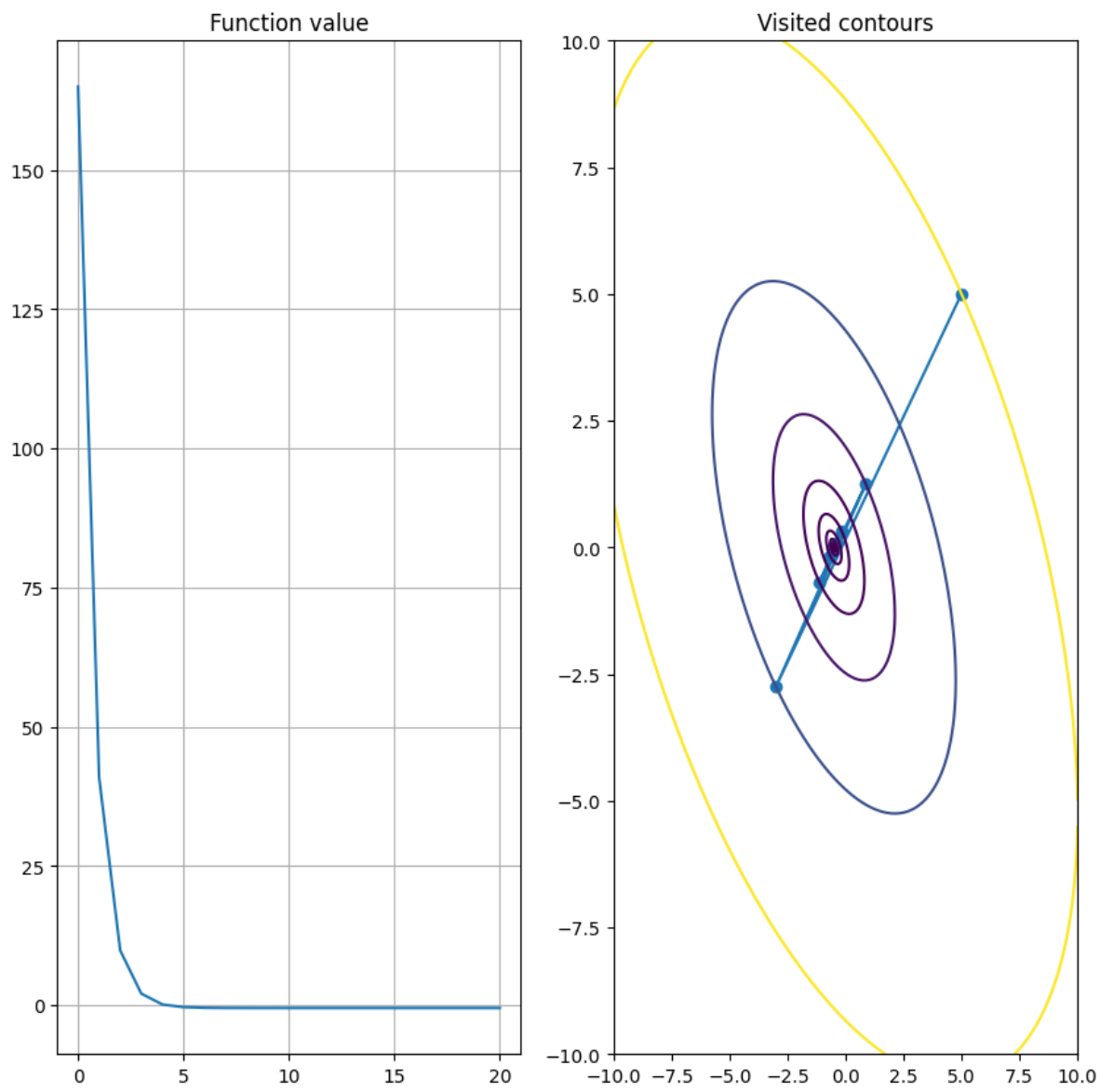


Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





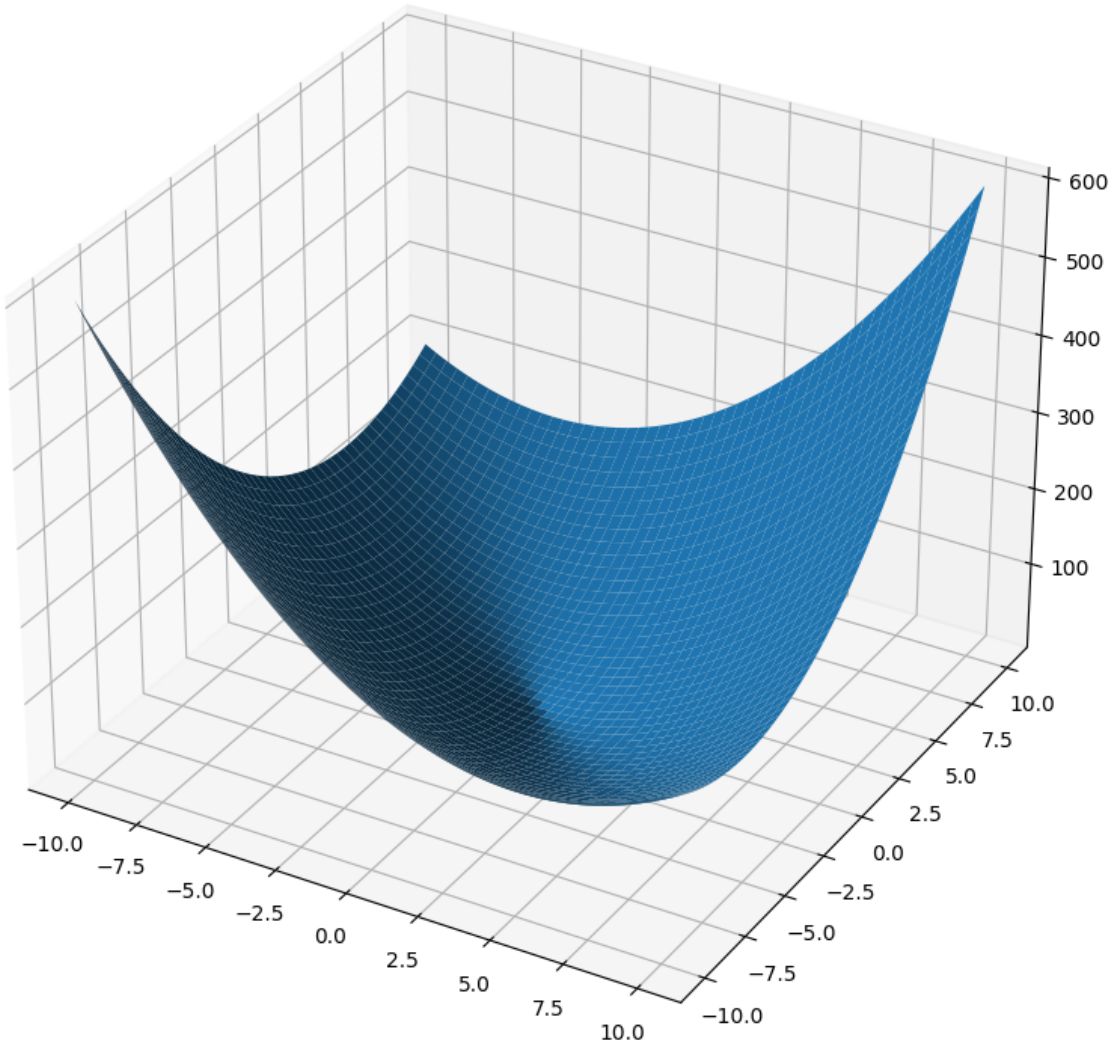
In [16]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.3],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=2, b=2, c=2, d=0, e=0
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[5.00000000e+00 5.00000000e+00]
 [2.00000000e+00 2.00000000e+00]
 [8.00000000e-01 8.00000000e-01]
 [3.20000000e-01 3.20000000e-01]
 [1.28000000e-01 1.28000000e-01]
 [5.12000000e-02 5.12000000e-02]
 [2.04800000e-02 2.04800000e-02]
 [8.19200000e-03 8.19200000e-03]
 [3.27680000e-03 3.27680000e-03]
 [1.31072000e-03 1.31072000e-03]
 [5.24288000e-04 5.24288000e-04]
 [2.09715200e-04 2.09715200e-04]
 [8.38860800e-05 8.38860800e-05]
 [3.35544320e-05 3.35544320e-05]
 [1.34217728e-05 1.34217728e-05]
 [5.36870912e-06 5.36870912e-06]
 [2.14748365e-06 2.14748365e-06]
 [8.58993459e-07 8.58993459e-07]
 [3.43597384e-07 3.43597384e-07]
 [1.37438953e-07 1.37438953e-07]
 [5.49755814e-08 5.49755814e-08]]
Best value found: x* = [5.49755814e-08 5.49755814e-08] with f(x*) = 1.813388729421
935e-14
Optimizing with fixed step = 0.3
Optimizer trajectory:
[[ 5.          5.        ]
 [-4.         -4.        ]
 [ 3.2         3.2       ]
 [-2.56       -2.56      ]
 [ 2.048       2.048     ]
 [-1.6384     -1.6384    ]
 [ 1.31072     1.31072   ]
 [-1.048576   -1.048576  ]
 [ 0.8388608   0.8388608 ]
 [-0.67108864 -0.67108864]
 [ 0.53687091  0.53687091]
 [-0.42949673 -0.42949673]
 [ 0.34359738  0.34359738]
 [-0.27487791 -0.27487791]
 [ 0.21990233  0.21990233]
 [-0.17592186 -0.17592186]
 [ 0.14073749  0.14073749]
 [-0.11258999 -0.11258999]
 [ 0.09007199  0.09007199]
 [-0.07205759 -0.07205759]
 [ 0.05764608  0.05764608]]
Best value found: x* = [0.05764608 0.05764608] with f(x*) = 0.0199384199367737
Optimizing with binary search
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.52587891e-04 -1.52587891e-04]
 [ 4.65661287e-09  4.65661287e-09]
 [-1.42108547e-13 -1.42108547e-13]
```

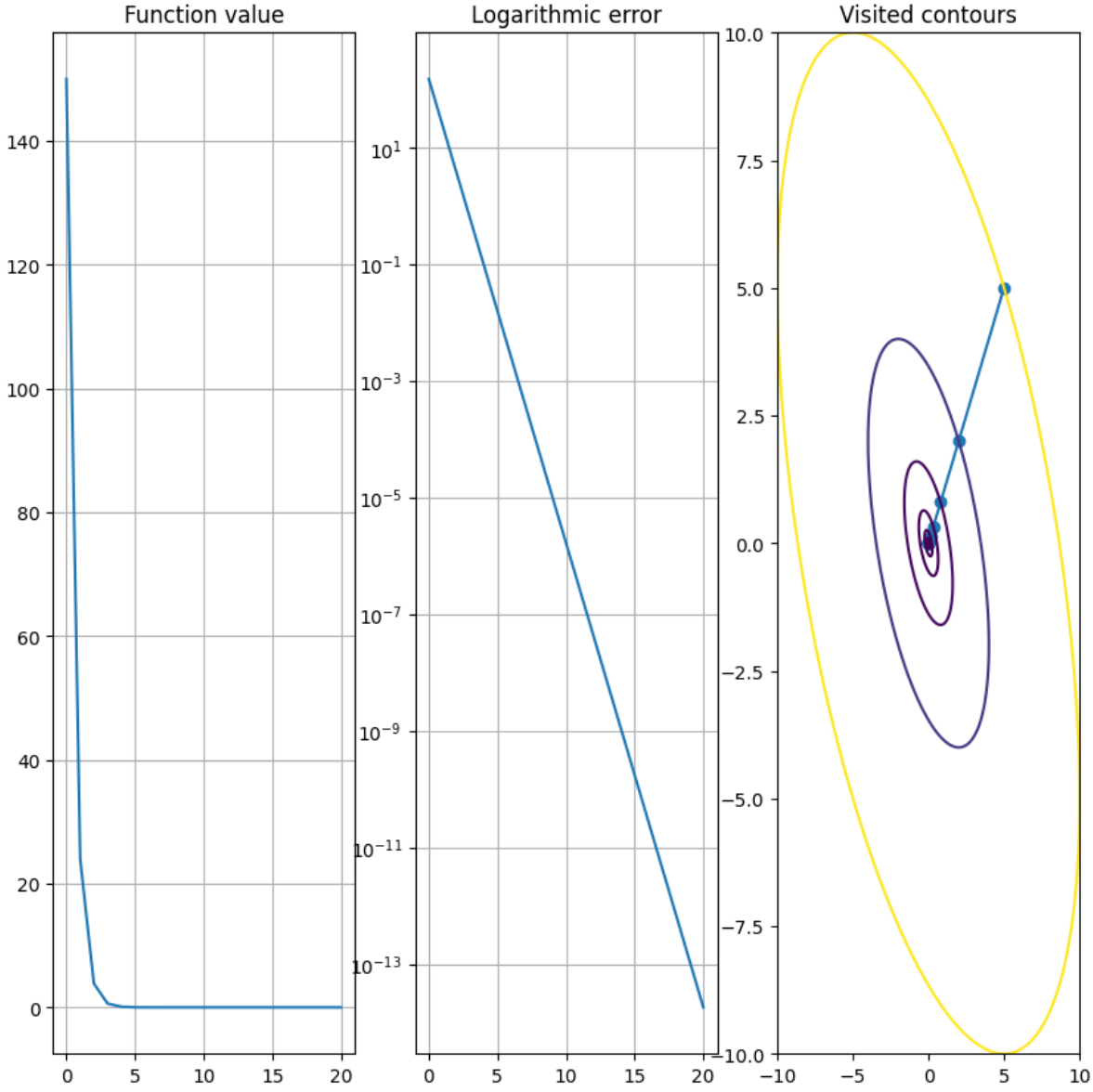
```
 [ 4.33680869e-18  4.33680869e-18]
 [-1.32348898e-22 -1.32348898e-22]
 [ 4.03896783e-27  4.03896783e-27]
 [-1.23259516e-31 -1.23259516e-31]
 [ 3.76158192e-36  3.76158192e-36]
 [-1.14794370e-40 -1.14794370e-40]
 [ 3.50324616e-45  3.50324616e-45]
 [-1.06910588e-49 -1.06910588e-49]
 [ 3.26265223e-54  3.26265223e-54]
 [-9.95682444e-59 -9.95682444e-59]
 [ 3.03858168e-63  3.03858168e-63]
 [-9.27301538e-68 -9.27301538e-68]
 [ 2.82989971e-72  2.82989971e-72]
 [-8.63616856e-77 -8.63616856e-77]
 [ 2.63554949e-81  2.63554949e-81]
 [-8.04305873e-86 -8.04305873e-86]
 [ 2.45454673e-90  2.45454673e-90]]
Best value found: x* = [2.45454673e-90 2.45454673e-90] with f(x*) = 3.614879797654
326e-179
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-6.25000000e-01 -6.25000000e-01]
 [ 7.81250000e-02  7.81250000e-02]
 [-9.76562500e-03 -9.76562500e-03]
 [ 1.22070312e-03  1.22070312e-03]
 [-1.52587891e-04 -1.52587891e-04]
 [ 7.62939453e-04  7.62939453e-04]
 [-9.53674316e-05 -9.53674316e-05]
 [ 4.76837158e-04  4.76837158e-04]
 [-2.38418579e-04 -2.38418579e-04]
 [ 4.76837158e-04  4.76837158e-04]
 [-2.38418579e-04 -2.38418579e-04]
 [ 4.76837158e-04  4.76837158e-04]
 [-2.38418579e-04 -2.38418579e-04]
 [ 4.76837158e-04  4.76837158e-04]
 [-2.38418579e-04 -2.38418579e-04]
 [ 4.76837158e-04  4.76837158e-04]
 [-2.38418579e-04 -2.38418579e-04]
 [ 4.76837158e-04  4.76837158e-04]]
Best value found: x* = [0.00047684 0.00047684] with f(x*) = 1.3642420526593924e-06
Optimizing with golden ration
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-6.22960624e-05 -6.22960624e-05]
 [ 7.76159877e-10  7.76159877e-10]
 [-9.67034082e-15 -9.67034082e-15]
 [ 1.20484831e-19  1.20484831e-19]
 [-1.50114611e-24 -1.50114611e-24]
 [ 1.87030983e-29  1.87030983e-29]
 [-2.33025876e-34 -2.33025876e-34]
 [ 2.90331890e-39  2.90331890e-39]
 [-3.61730671e-44 -3.61730671e-44]
 [ 4.50687928e-49  4.50687928e-49]
```

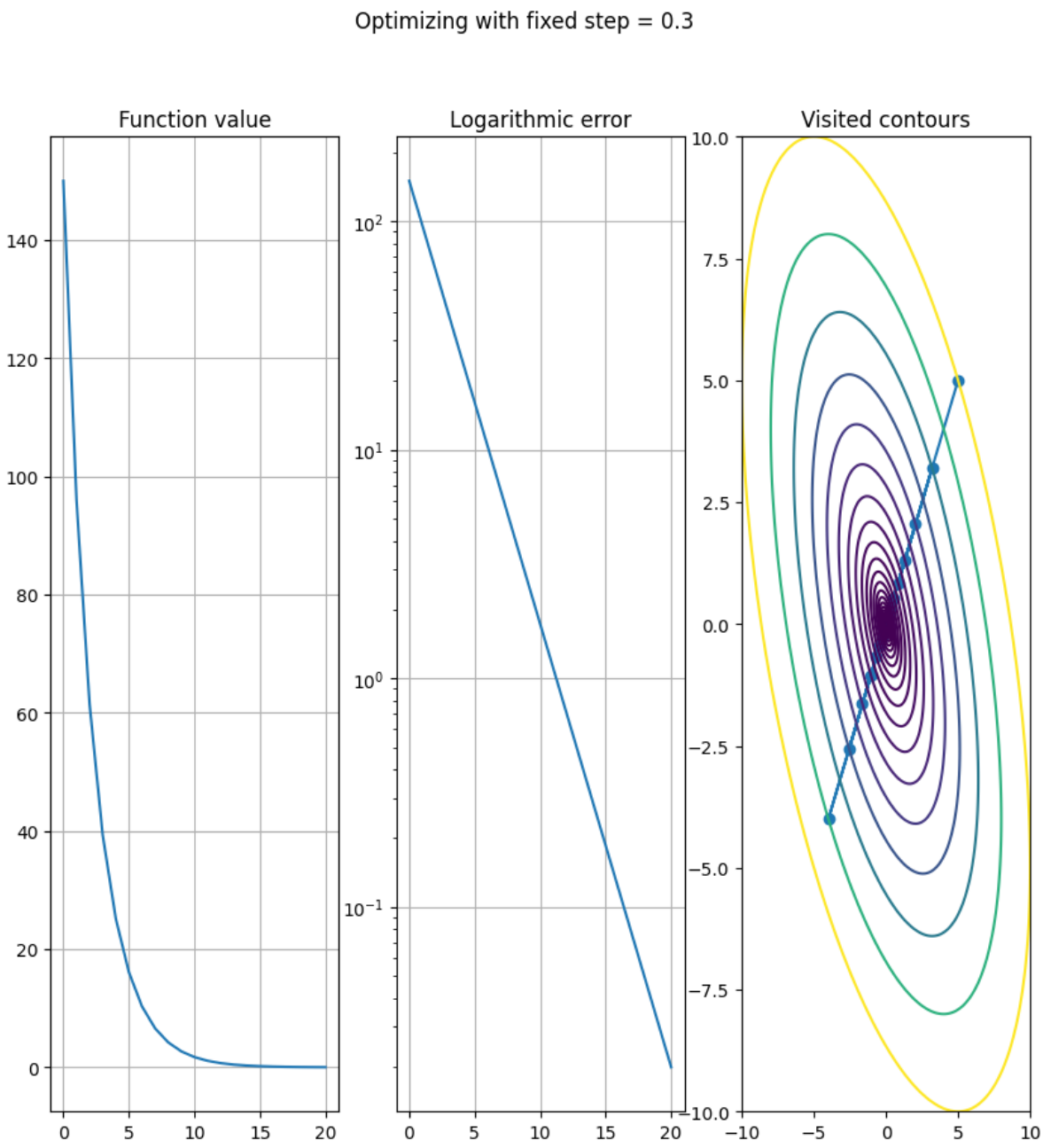
```
 [-5.61521666e-54 -5.61521666e-54]
 [ 6.99611774e-59  6.99611774e-59]
 [-8.71661174e-64 -8.71661174e-64]
 [ 1.08602118e-68  1.08602118e-68]
 [-1.35309686e-73 -1.35309686e-73]
 [ 1.68585213e-78  1.68585213e-78]
 [-2.10043899e-83 -2.10043899e-83]
 [ 2.61698156e-88  2.61698156e-88]
 [-3.26055293e-93 -3.26055293e-93]
 [ 4.06239217e-98  4.06239217e-98]]
Best value found: x* = [4.06239217e-98 4.06239217e-98] with f(x*) = 9.901818100106
845e-195
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.00000000e+000  5.00000000e+000]
 [-1.85698401e-005 -1.85698401e-005]
 [ 6.89677919e-011  6.89677919e-011]
 [-2.56144173e-016 -2.56144173e-016]
 [ 9.51311265e-022  9.51311265e-022]
 [-3.53313961e-027 -3.53313961e-027]
 [ 1.31219675e-032  1.31219675e-032]
 [-4.87345674e-038 -4.87345674e-038]
 [ 1.80998624e-043  1.80998624e-043]
 [-6.72223101e-049 -6.72223101e-049]
 [ 2.49661509e-054  2.49661509e-054]
 [-9.27234860e-060 -9.27234860e-060]
 [ 3.44372061e-065  3.44372061e-065]
 [-1.27898682e-070 -1.27898682e-070]
 [ 4.75011613e-076  4.75011613e-076]
 [-1.76417793e-081 -1.76417793e-081]
 [ 6.55210041e-087  6.55210041e-087]
 [-2.43342913e-092 -2.43342913e-092]
 [ 9.03767796e-098  9.03767796e-098]
 [-3.35656468e-103 -3.35656468e-103]
 [ 1.24661739e-108  1.24661739e-108]]
Best value found: x* = [1.24661739e-108 1.24661739e-108] with f(x*) = 9.3243294443
44603e-216
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-2.50000000e+00 -2.50000000e+00]
 [ 1.25000000e+00  1.25000000e+00]
 [-6.25000000e-01 -6.25000000e-01]
 [ 3.12500000e-01  3.12500000e-01]
 [-1.56250000e-01 -1.56250000e-01]
 [ 7.81250000e-02  7.81250000e-02]
 [-3.90625000e-02 -3.90625000e-02]
 [ 1.95312500e-02  1.95312500e-02]
 [-9.76562500e-03 -9.76562500e-03]
 [ 4.88281250e-03  4.88281250e-03]
 [-2.44140625e-03 -2.44140625e-03]
 [ 1.22070312e-03  1.22070312e-03]
 [-6.10351562e-04 -6.10351562e-04]
 [ 3.05175781e-04  3.05175781e-04]
 [-1.52587891e-04 -1.52587891e-04]
 [ 7.62939453e-05  7.62939453e-05]
```

```
 [-3.81469727e-05 -3.81469727e-05]
 [ 1.90734863e-05  1.90734863e-05]
 [-9.53674316e-06 -9.53674316e-06]
 [ 4.76837158e-06  4.76837158e-06]]
Best value found: x* = [4.76837158e-06 4.76837158e-06] with f(x*) = 1.364242052659
3924e-10
```

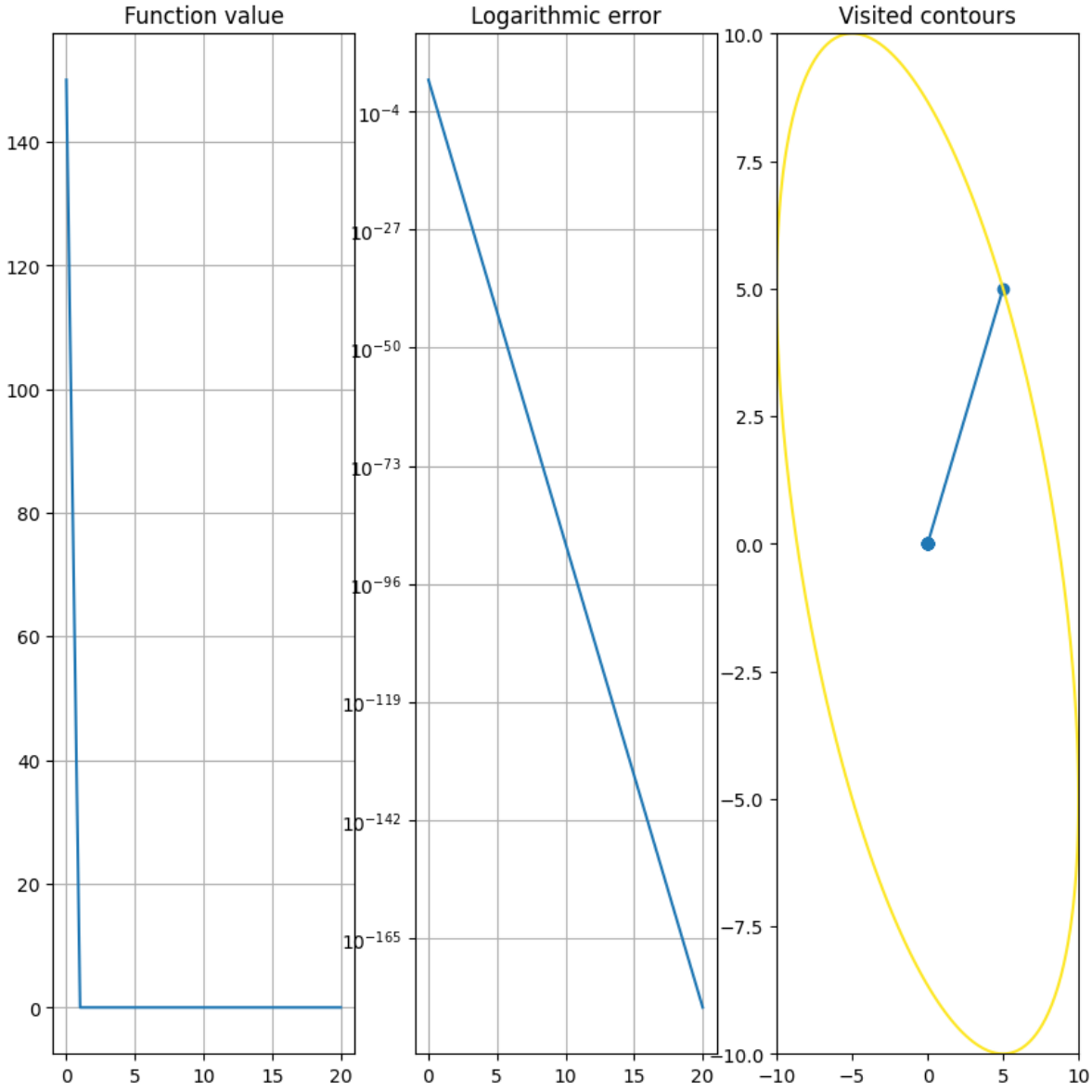


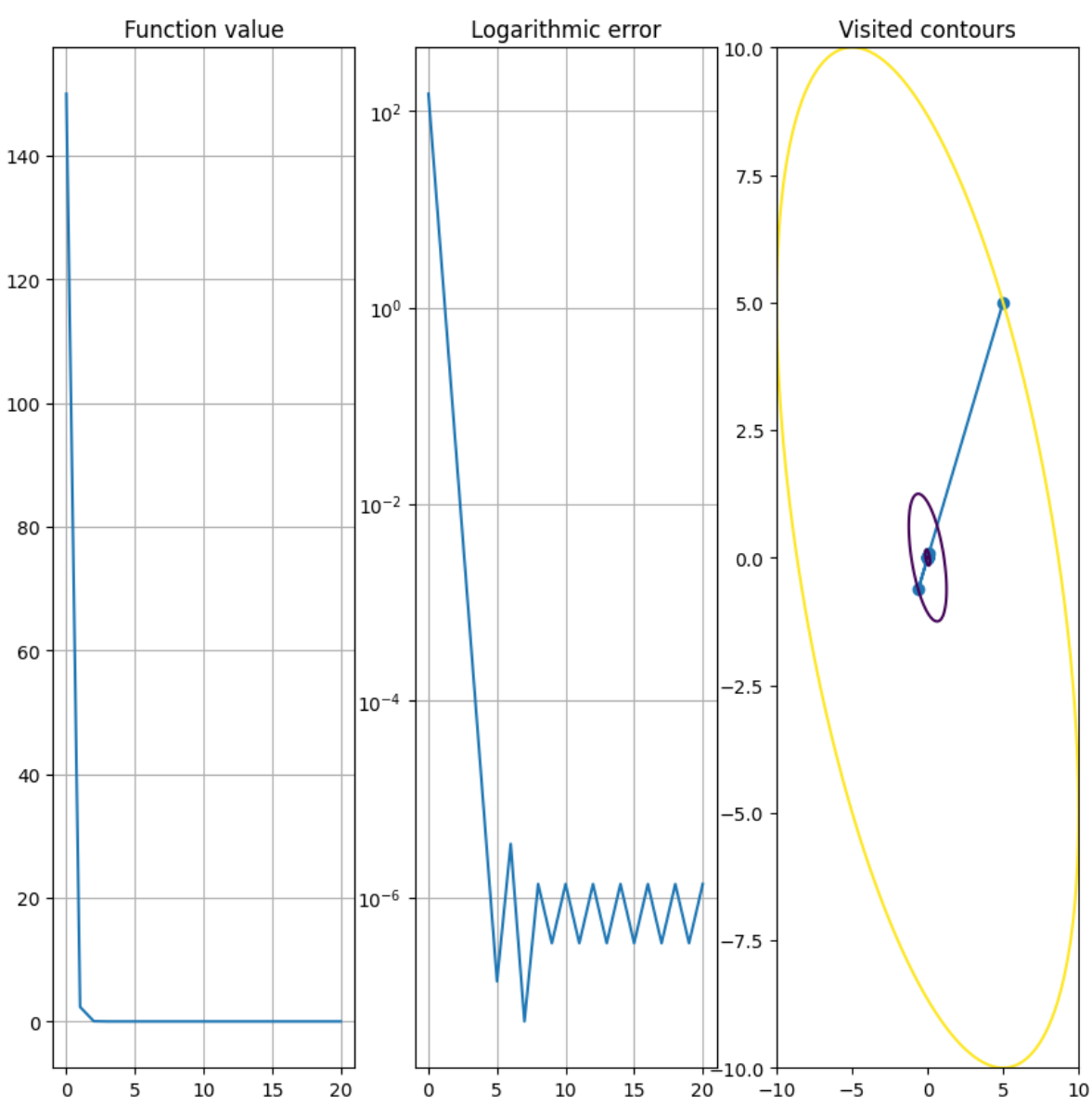
Optimizing with fixed step = 0.1



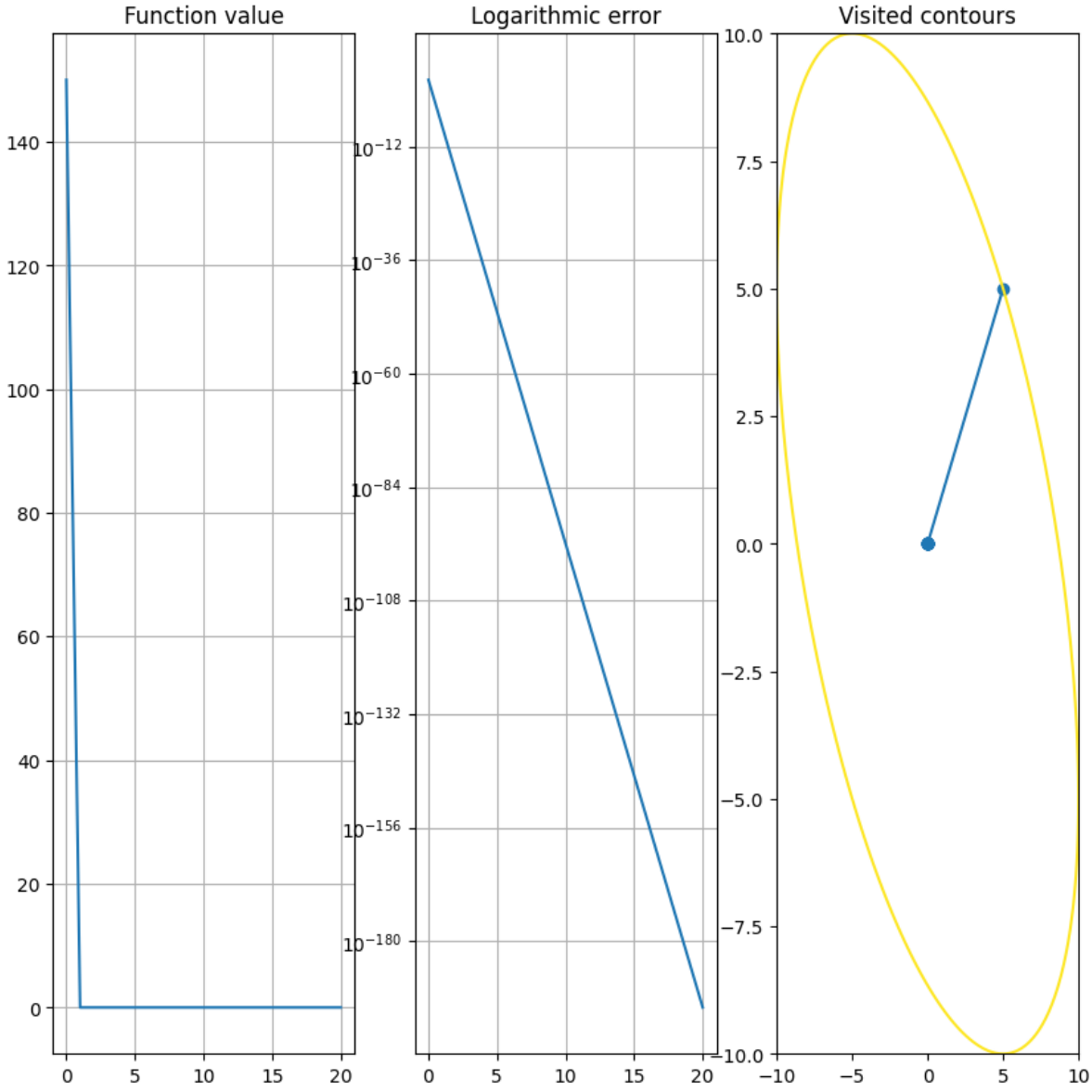

Optimizing with fixed step = 0.3
Function value
Logarithmic error
Visited contours

Optimizing with binary search

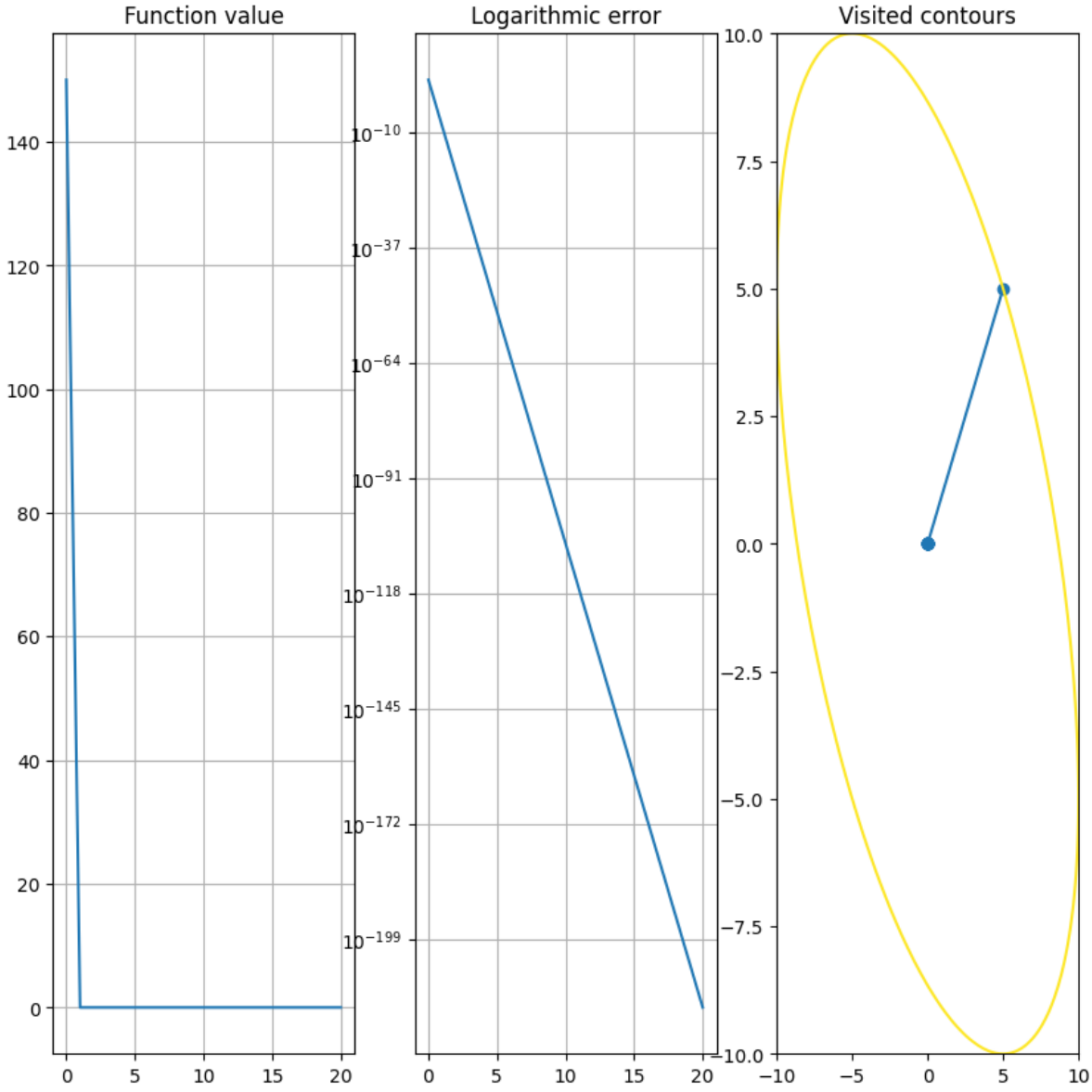


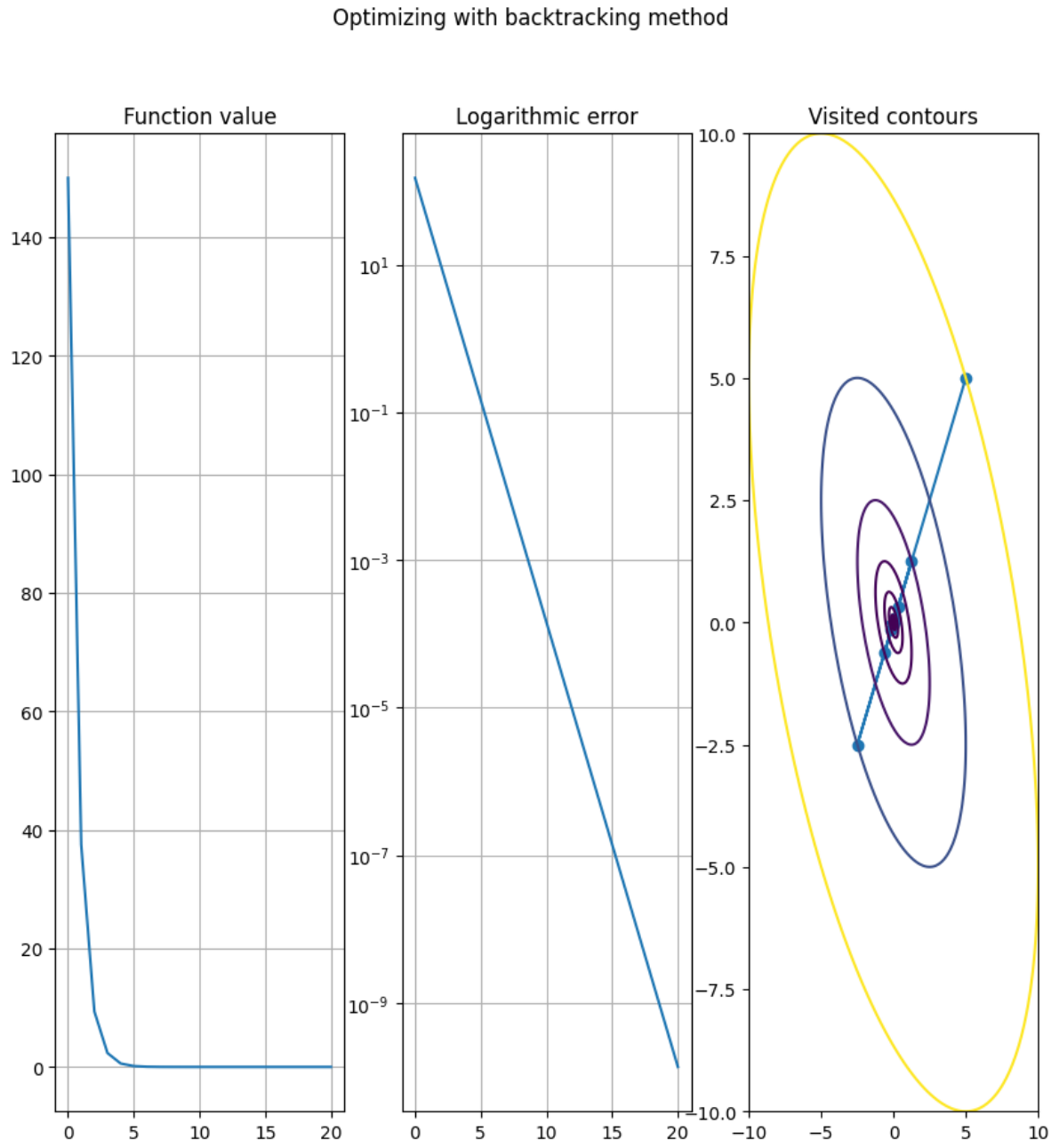

Optimizing with binary search limited by 5 iterations
Function value
Logarithmic error
Visited contours
140
120
100
80
60
40
20
0
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0
5
10
15
20
−10
−5

Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





Равномерное растяжение координат

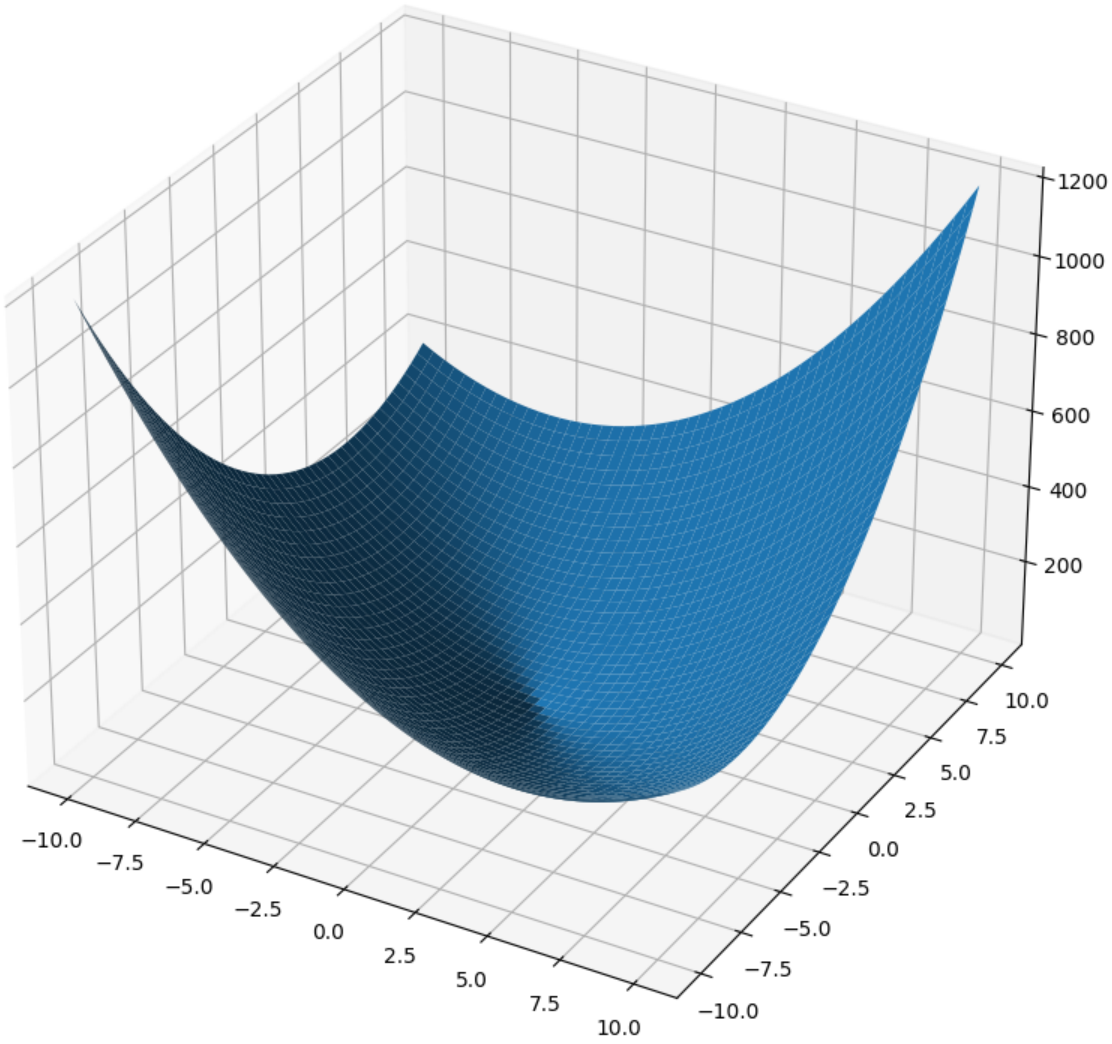
In [17]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.3],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=4, b=4, c=4, d=0, e=0
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[ 5.00000e+00  5.00000e+00]
 [-1.00000e+00 -1.00000e+00]
 [ 2.00000e-01  2.00000e-01]
 [-4.00000e-02 -4.00000e-02]
 [ 8.00000e-03  8.00000e-03]
 [-1.60000e-03 -1.60000e-03]
 [ 3.20000e-04  3.20000e-04]
 [-6.40000e-05 -6.40000e-05]
 [ 1.28000e-05  1.28000e-05]
 [-2.56000e-06 -2.56000e-06]
 [ 5.12000e-07  5.12000e-07]
 [-1.02400e-07 -1.02400e-07]
 [ 2.04800e-08  2.04800e-08]
 [-4.09600e-09 -4.09600e-09]
 [ 8.19200e-10  8.19200e-10]
 [-1.63840e-10 -1.63840e-10]
 [ 3.27680e-11  3.27680e-11]
 [-6.55360e-12 -6.55360e-12]
 [ 1.31072e-12  1.31072e-12]
 [-2.62144e-13 -2.62144e-13]
 [ 5.24288e-14  5.24288e-14]]
Best value found: x* = [5.24288e-14 5.24288e-14] with f(x*) = 3.2985348833280665e-
26
Optimizing with fixed step = 0.3
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.30000000e+01 -1.30000000e+01]
 [ 3.38000000e+01  3.38000000e+01]
 [-8.78800000e+01 -8.78800000e+01]
 [ 2.28488000e+02  2.28488000e+02]
 [-5.94068800e+02 -5.94068800e+02]
 [ 1.54457888e+03  1.54457888e+03]
 [-4.01590509e+03 -4.01590509e+03]
 [ 1.04413532e+04  1.04413532e+04]
 [-2.71475184e+04 -2.71475184e+04]
 [ 7.05835478e+04  7.05835478e+04]
 [-1.83517224e+05 -1.83517224e+05]
 [ 4.77144783e+05  4.77144783e+05]
 [-1.24057644e+06 -1.24057644e+06]
 [ 3.22549874e+06  3.22549874e+06]
 [-8.38629671e+06 -8.38629671e+06]
 [ 2.18043714e+07  2.18043714e+07]
 [-5.66913658e+07 -5.66913658e+07]
 [ 1.47397551e+08  1.47397551e+08]
 [-3.83233633e+08 -3.83233633e+08]
 [ 9.96407445e+08  9.96407445e+08]]
Best value found: x* = [9.96407445e+08 9.96407445e+08] with f(x*) = 1.191393355168
9069e+19
Optimizing with binary search
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.52587891e-04 -1.52587891e-04]
 [ 4.65661287e-09  4.65661287e-09]
```

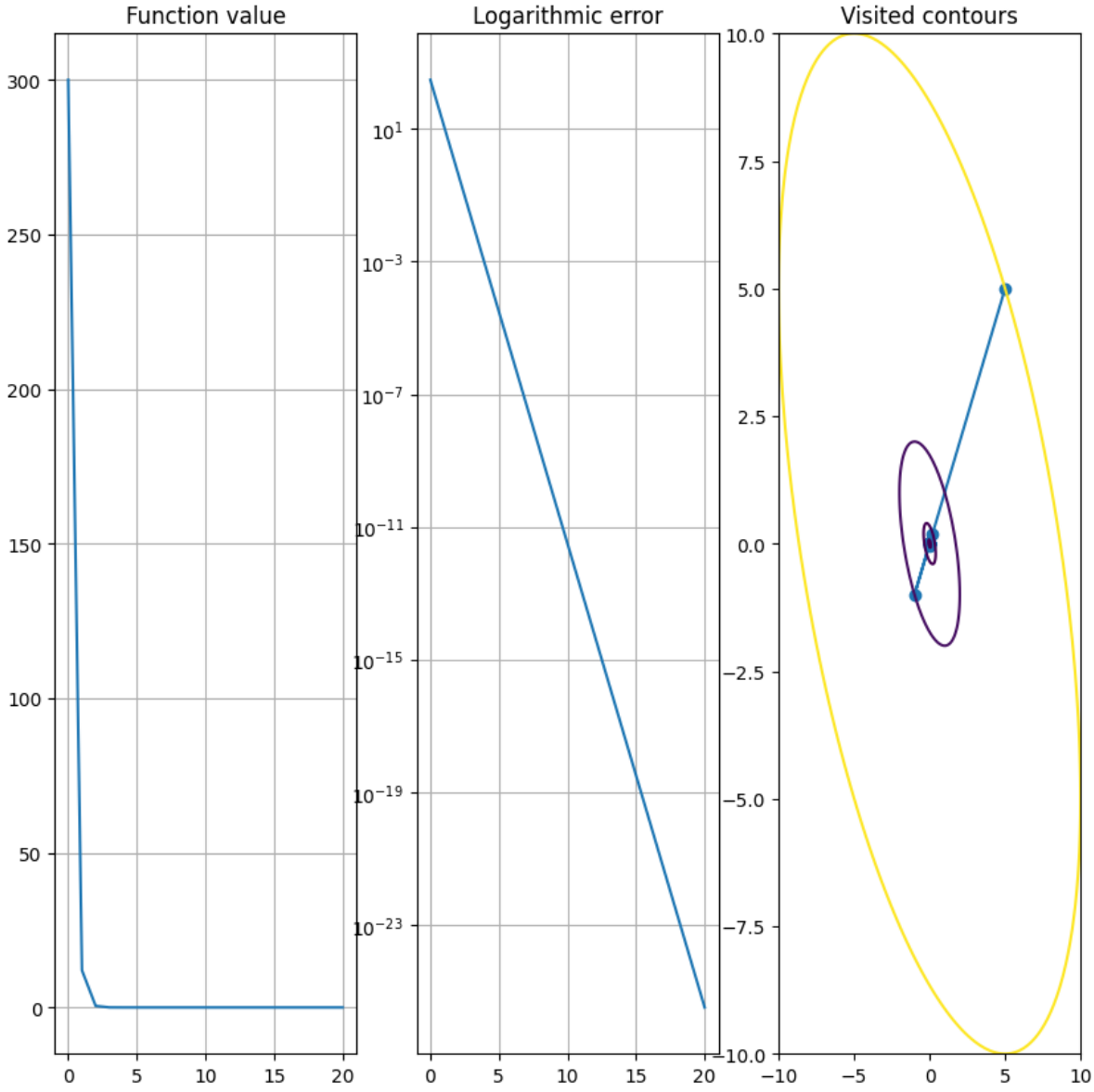
```
 [-1.42108547e-13 -1.42108547e-13]
 [ 4.33680869e-18  4.33680869e-18]
 [-1.32348898e-22 -1.32348898e-22]
 [ 4.03896783e-27  4.03896783e-27]
 [-1.23259516e-31 -1.23259516e-31]
 [ 3.76158192e-36  3.76158192e-36]
 [-1.14794370e-40 -1.14794370e-40]
 [ 3.50324616e-45  3.50324616e-45]
 [-1.06910588e-49 -1.06910588e-49]
 [ 3.26265223e-54  3.26265223e-54]
 [-9.95682444e-59 -9.95682444e-59]
 [ 3.03858168e-63  3.03858168e-63]
 [-9.27301538e-68 -9.27301538e-68]
 [ 2.82989971e-72  2.82989971e-72]
 [-8.63616856e-77 -8.63616856e-77]
 [ 2.63554949e-81  2.63554949e-81]
 [-8.04305873e-86 -8.04305873e-86]
 [ 2.45454673e-90  2.45454673e-90]]
Best value found: x* = [2.45454673e-90 2.45454673e-90] with f(x*) = 7.229759595308
652e-179
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-6.25000000e-01 -6.25000000e-01]
 [ 7.81250000e-02  7.81250000e-02]
 [-9.76562500e-03 -9.76562500e-03]
 [ 1.22070312e-03  1.22070312e-03]
 [-1.52587891e-04 -1.52587891e-04]
 [ 7.62939453e-05  7.62939453e-05]
 [-3.81469727e-04 -3.81469727e-04]
 [ 4.76837158e-05  4.76837158e-05]
 [-5.24520874e-04 -5.24520874e-04]
 [ 6.55651093e-05  6.55651093e-05]
 [-3.27825546e-04 -3.27825546e-04]
 [ 4.09781933e-05  4.09781933e-05]
 [-4.50760126e-04 -4.50760126e-04]
 [ 5.63450158e-05  5.63450158e-05]
 [-2.81725079e-04 -2.81725079e-04]
 [ 3.52156349e-05  3.52156349e-05]
 [-3.87371983e-04 -3.87371983e-04]
 [ 4.84214979e-05  4.84214979e-05]
 [-5.32636477e-04 -5.32636477e-04]
 [ 6.65795596e-05  6.65795596e-05]]
Best value found: x* = [6.65795596e-05 6.65795596e-05] with f(x*) = 5.319405314752
847e-08
Optimizing with golden ration
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.72816503e-04 -1.72816503e-04]
 [ 5.97310875e-09  5.97310875e-09]
 [-2.06450353e-13 -2.06450353e-13]
 [ 7.13560563e-18  7.13560563e-18]
 [-2.46630082e-22 -2.46630082e-22]
 [ 8.52434968e-27  8.52434968e-27]
 [-2.94629661e-31 -2.94629661e-31]
 [ 1.01833735e-35  1.01833735e-35]
```

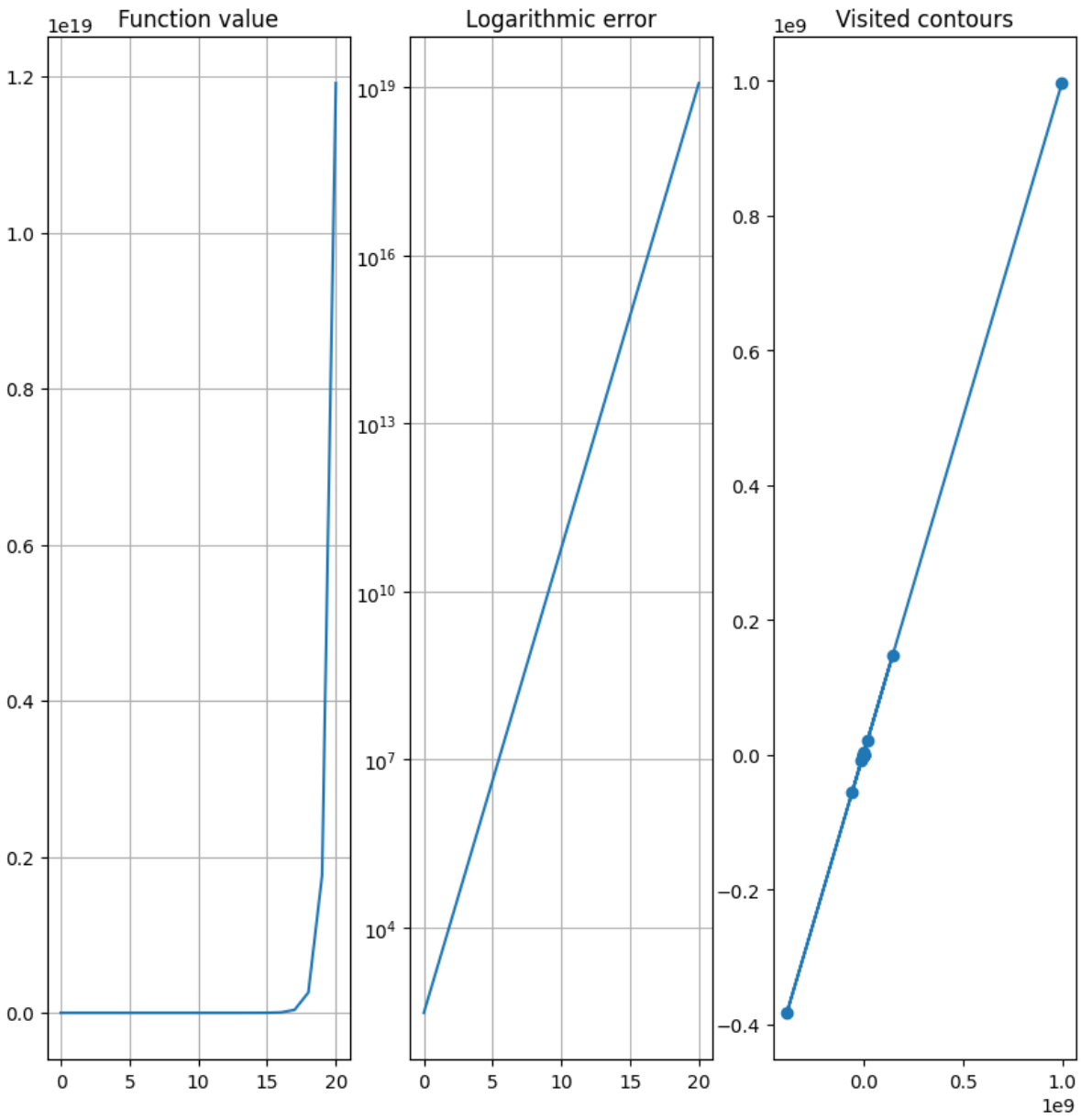
```
 [-3.51971001e-40 -3.51971001e-40]
 [ 1.21652795e-44  1.21652795e-44]
 [-4.20472213e-49 -4.20472213e-49]
 [ 1.45329075e-53  1.45329075e-53]
 [-5.02305251e-58 -5.02305251e-58]
 [ 1.73613274e-62  1.73613274e-62]
 [-6.00064778e-67 -6.00064778e-67]
 [ 2.07402193e-71  2.07402193e-71]
 [-7.16850435e-76 -7.16850435e-76]
 [ 2.47767171e-80  2.47767171e-80]
 [-8.56365122e-85 -8.56365122e-85]
 [ 2.95988051e-89  2.95988051e-89]]
Best value found: x* = [2.95988051e-89 2.95988051e-89] with f(x*) = 1.051307118942
4218e-176
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.00000000e+000  5.00000000e+000]
 [-4.08536481e-005 -4.08536481e-005]
 [ 3.33804113e-010  3.33804113e-010]
 [-2.72742315e-015 -2.72742315e-015]
 [ 2.22850372e-020  2.22850372e-020]
 [-1.82085013e-025 -1.82085013e-025]
 [ 1.48776741e-030  1.48776741e-030]
 [-1.21561453e-035 -1.21561453e-035]
 [ 9.93245763e-041  9.93245763e-041]
 [-8.11554258e-046 -8.11554258e-046]
 [ 6.63099042e-051  6.63099042e-051]
 [-5.41800298e-056 -5.41800298e-056]
 [ 4.42690375e-061  4.42690375e-061]
 [-3.61710336e-066 -3.61710336e-066]
 [ 2.95543736e-071  2.95543736e-071]
 [-2.41480796e-076 -2.41480796e-076]
 [ 1.97307429e-081  1.97307429e-081]
 [-1.61214566e-086 -1.61214566e-086]
 [ 1.31724063e-091  1.31724063e-091]
 [-1.07628170e-096 -1.07628170e-096]
 [ 8.79400678e-102  8.79400678e-102]]
Best value found: x* = [8.79400678e-102 8.79400678e-102] with f(x*) = 9.2801466247
43394e-202
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-2.50000000e+00 -2.50000000e+00]
 [ 1.25000000e+00  1.25000000e+00]
 [-6.25000000e-01 -6.25000000e-01]
 [ 3.12500000e-01  3.12500000e-01]
 [-1.56250000e-01 -1.56250000e-01]
 [ 7.81250000e-02  7.81250000e-02]
 [-3.90625000e-02 -3.90625000e-02]
 [ 1.95312500e-02  1.95312500e-02]
 [-9.76562500e-03 -9.76562500e-03]
 [ 4.88281250e-03  4.88281250e-03]
 [-2.44140625e-03 -2.44140625e-03]
 [ 1.22070312e-03  1.22070312e-03]
 [-6.10351562e-04 -6.10351562e-04]
 [ 3.05175781e-04  3.05175781e-04]
```

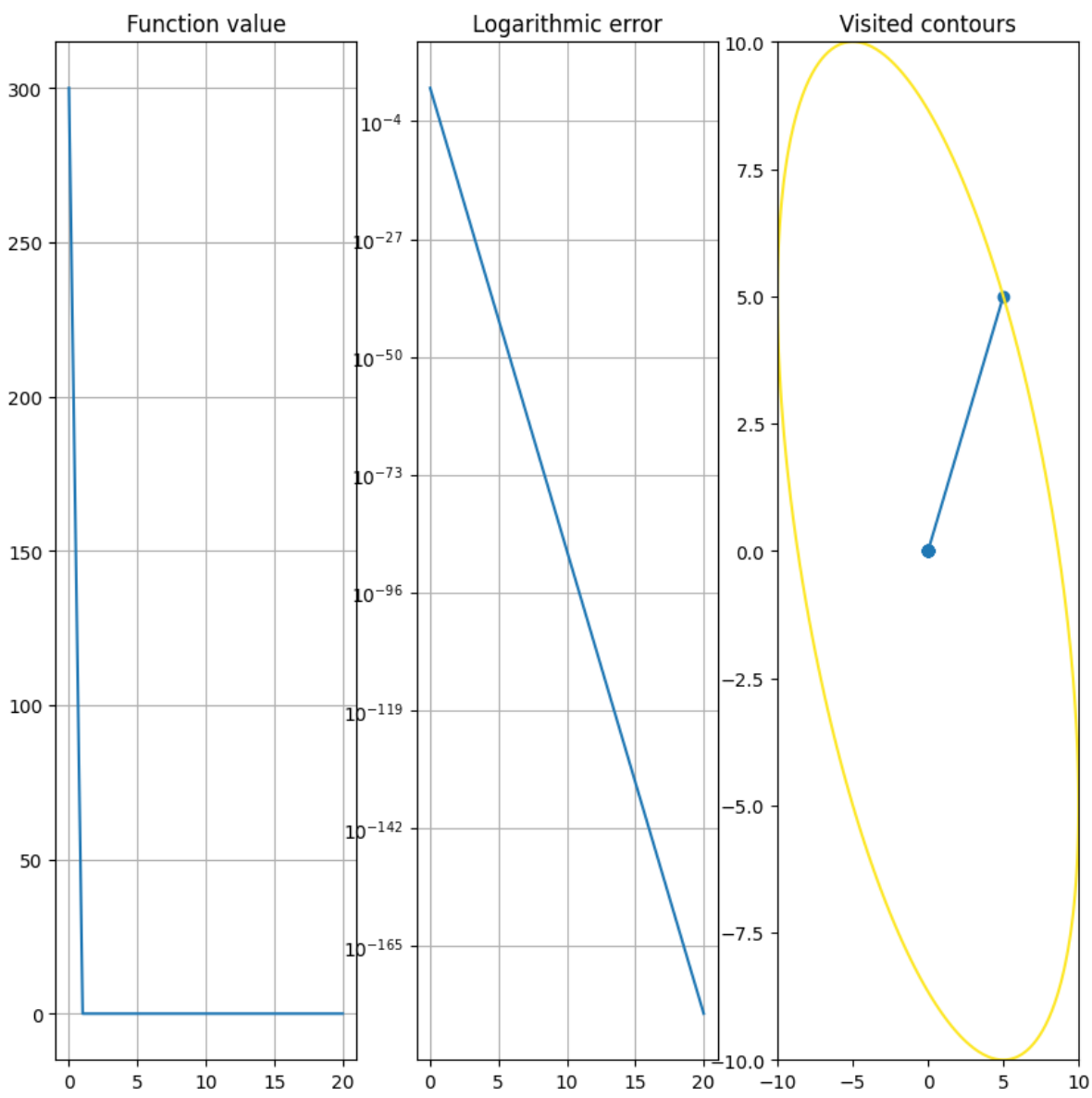
```
 [-1.52587891e-04 -1.52587891e-04]
 [ 7.62939453e-05  7.62939453e-05]
 [-3.81469727e-05 -3.81469727e-05]
 [ 1.90734863e-05  1.90734863e-05]
 [-9.53674316e-06 -9.53674316e-06]
 [ 4.76837158e-06  4.76837158e-06]]
Best value found: x* = [4.76837158e-06 4.76837158e-06] with f(x*) = 2.728484105318
7847e-10
```

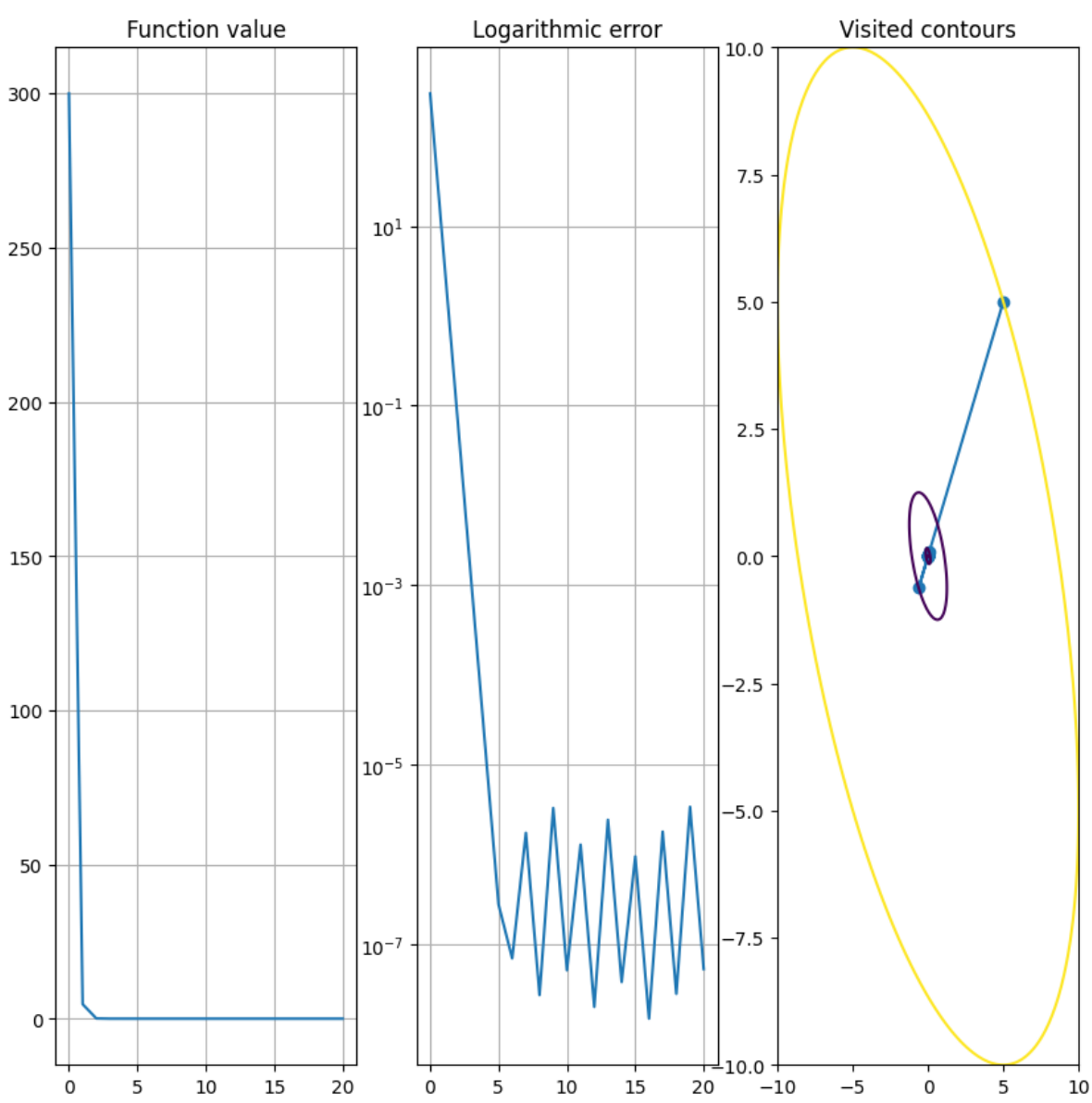


Optimizing with fixed step = 0.1

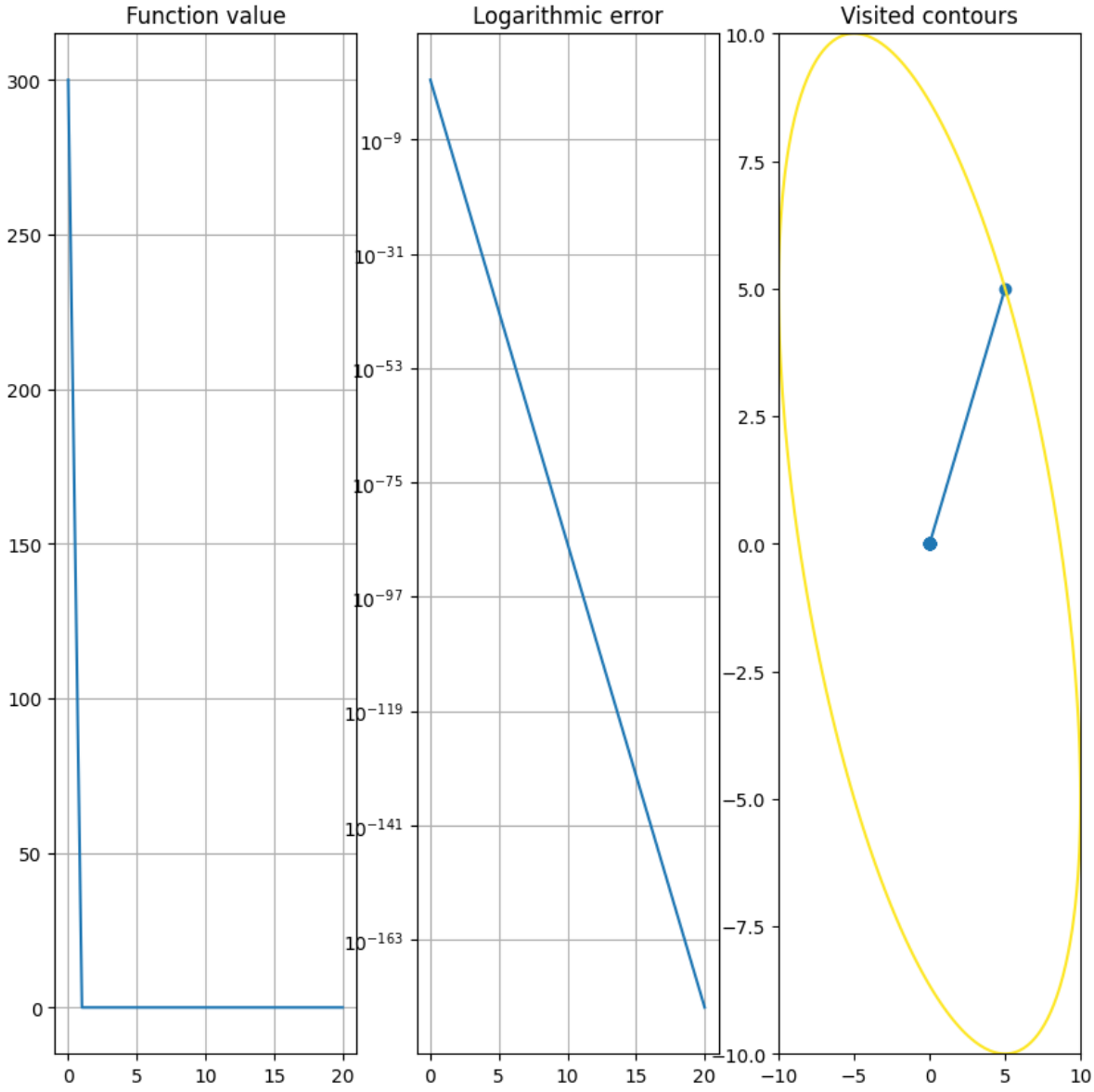



Optimizing with fixed step = 0.3
Function value
1e19
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Logarithmic error
10^{19}
10^{16}
10^{13}
10^{10}
10^{7}
10^{4}
0
5
10
15
20
Visited contours
1e9
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0.0
0.5
1.0
1e9

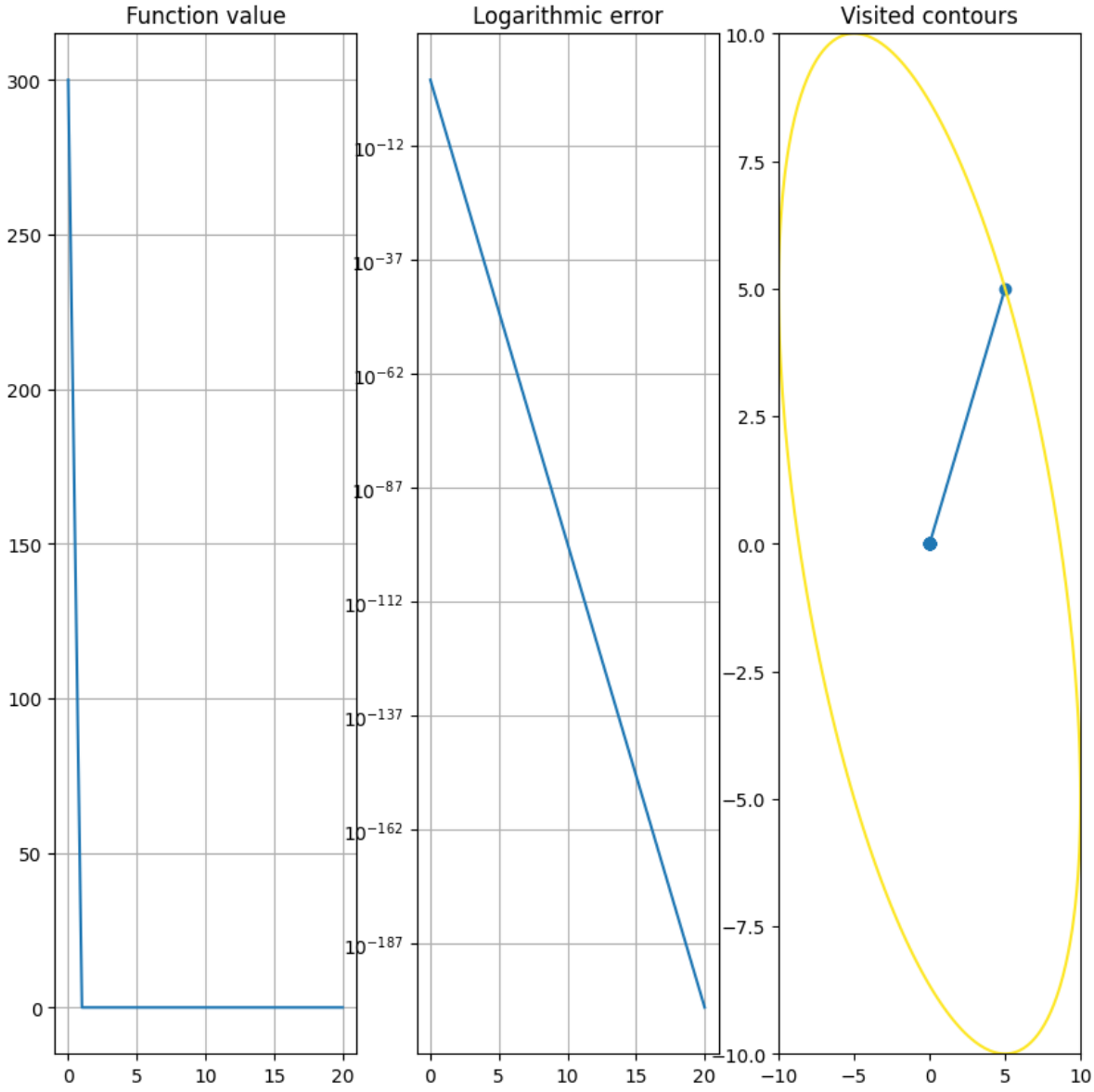

Optimizing with binary search
Function value
Logarithmic error
Visited contours
300
250
200
150
100
50
0
10^{-4}
10^{-27}
10^{-50}
10^{-73}
10^{-96}
10^{-119}
10^{-142}
10^{-165}
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0
5
10
15
20
−10
−5

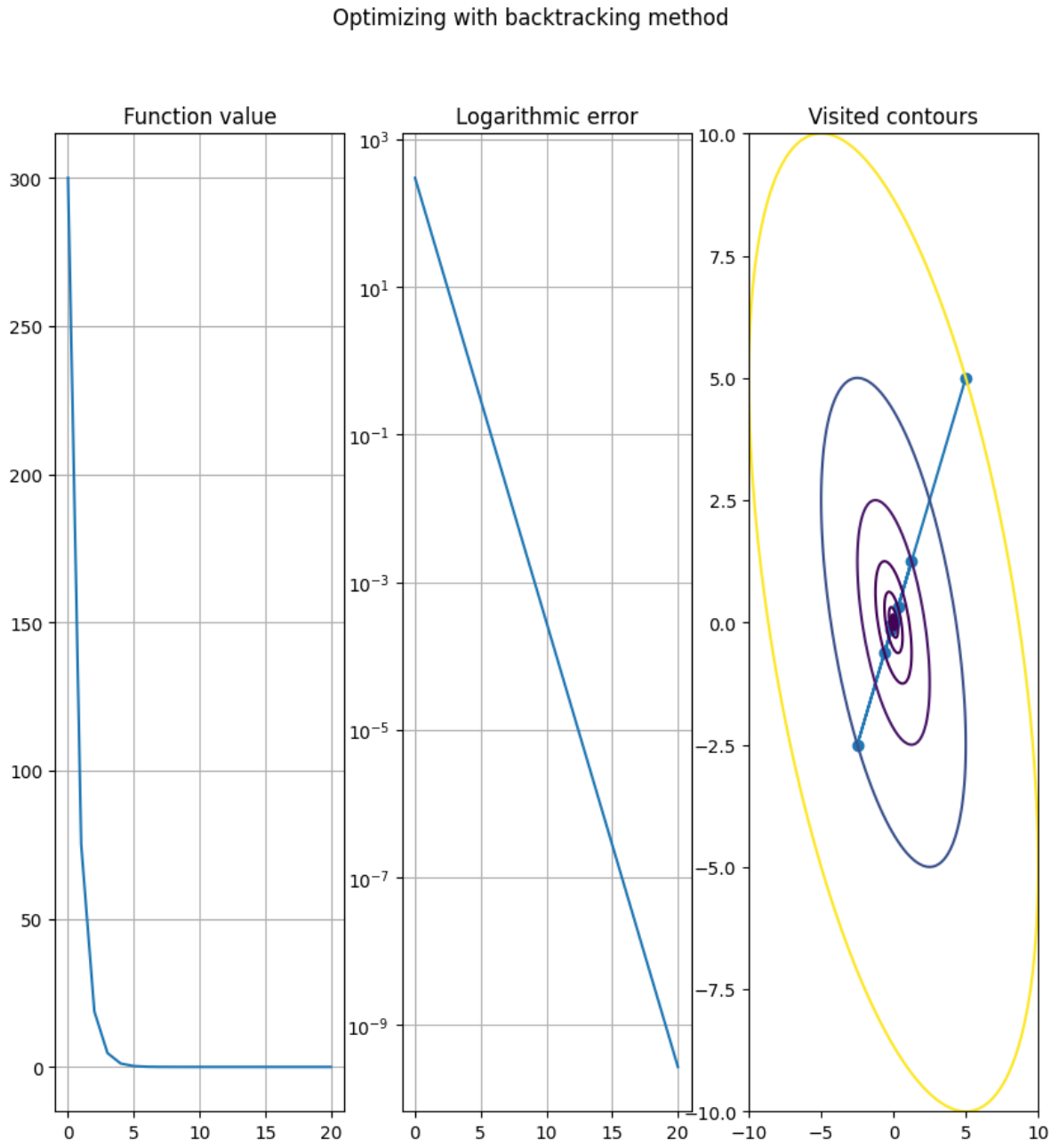

Optimizing with binary search limited by 5 iterations
Function value
Logarithmic error
Visited contours
300
250
200
150
100
50
0
10^{1}
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0
5
10
15
20
−10
−5
0
5
10

Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





Неравномерное растяжение координат

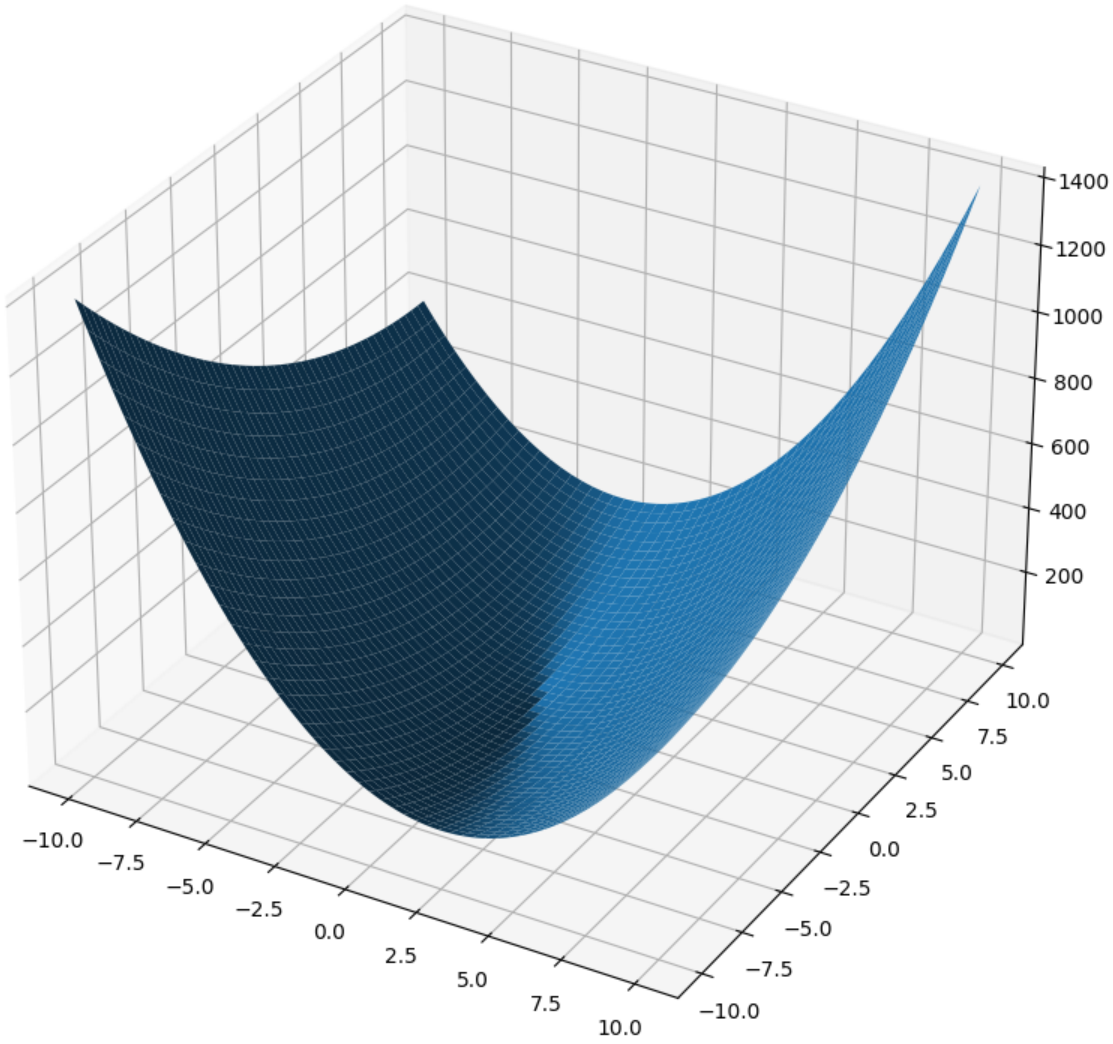
In [18]:
```
analyze_quadratic(
    roi=SearchRegion2d((-10, 10), (-10, 10)),
    fixed_steps=[0.1, 0.3],
    x0=np.array([5, 5]),
    bin_iters=5,
    fib_iters=30,
    a=8, b=4, c=2, d=0, e=0
)
```

```
Function plot:
Optimizing with fixed step = 0.1
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-5.00000000e+00  1.00000000e+00]
 [ 2.60000000e+00  2.60000000e+00]
 [-2.60000000e+00  5.20000000e-01]
 [ 1.35200000e+00  1.35200000e+00]
 [-1.35200000e+00  2.70400000e-01]
 [ 7.03040000e-01  7.03040000e-01]
 [-7.03040000e-01  1.40608000e-01]
 [ 3.65580800e-01  3.65580800e-01]
 [-3.65580800e-01  7.31161600e-02]
 [ 1.90102016e-01  1.90102016e-01]
 [-1.90102016e-01  3.80204032e-02]
 [ 9.88530483e-02  9.88530483e-02]
 [-9.88530483e-02  1.97706097e-02]
 [ 5.14035851e-02  5.14035851e-02]
 [-5.14035851e-02  1.02807170e-02]
 [ 2.67298643e-02  2.67298643e-02]
 [-2.67298643e-02  5.34597285e-03]
 [ 1.38995294e-02  1.38995294e-02]
 [-1.38995294e-02  2.77990588e-03]
 [ 7.22775530e-03  7.22775530e-03]]
Best value found: x* = [0.00722776 0.00722776] with f(x*) = 0.0007313662529580116
Optimizing with fixed step = 0.3
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-2.50000000e+01 -7.00000000e+00]
 [ 1.03400000e+02  3.14000000e+01]
 [-4.30600000e+02 -1.30360000e+02]
 [ 1.79271200e+03  5.42792000e+02]
 [-7.46365600e+03 -2.25981280e+03]
 [ 3.10736682e+04  9.40834976e+03]
 [-1.29369959e+05 -3.91700717e+04]
 [ 5.38609929e+05  1.63077965e+05]
 [-2.24241129e+06 -6.78947508e+05]
 [ 9.33589991e+06  2.82668305e+06]
 [-3.88684393e+07 -1.17684165e+07]
 [ 1.61822169e+08  4.89958105e+07]
 [-6.73719215e+08 -2.03985765e+08]
 [ 2.80491594e+09  8.49260211e+08]
 [-1.16777928e+10 -3.53575117e+09]
 [ 4.86185141e+10  1.47205016e+10]
 [-2.02414955e+11 -6.12863172e+10]
 [ 8.42720411e+11  2.55155210e+11]
 [-3.50852382e+12 -1.06229554e+12]
 [ 1.46071451e+13  4.42268769e+12]]
Best value found: x* = [1.46071451e+13 4.42268769e+12] with f(x*) = 2.004481209866
267e+27
Optimizing with binary search
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-8.47167969e-01  2.66113281e+00]
 [ 1.56728476e-01  1.58129260e-01]
 [-2.69066513e-02  8.44786084e-02]
```

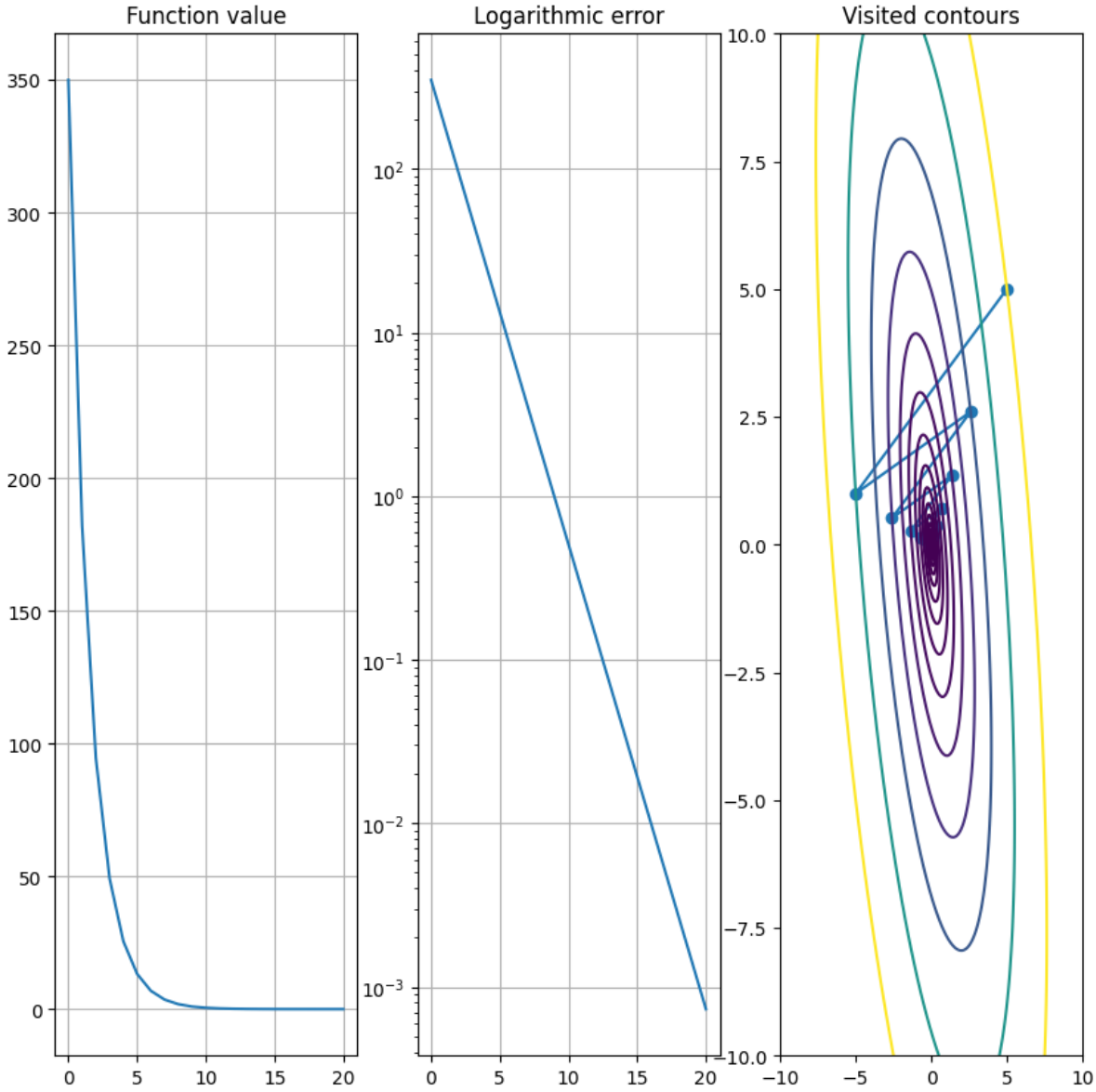
```
 [ 5.01085321e-03  5.09579081e-03]
 [-8.70407019e-04  2.73136414e-03]
 [ 1.63243655e-04  1.67493618e-04]
 [-2.87280852e-05  9.01082828e-05]
 [ 5.41988435e-06  5.60561332e-06]
 [-9.65080729e-07  3.02554996e-06]
 [ 1.83241590e-07  1.91147157e-07]
 [-3.30372967e-08  1.03525474e-07]
 [ 6.30982611e-09  6.63004376e-09]
 [-1.14982662e-09  3.60121069e-09]
 [ 2.21032117e-10  2.34298109e-10]
 [-4.07919269e-11  1.27705147e-10]
 [ 7.88143278e-12  8.41431211e-12]
 [-1.46976387e-12  4.59897536e-12]
 [ 2.85736616e-13  3.07578079e-13]
 [-5.39202084e-14  1.68646693e-13]
 [ 1.05364292e-14  1.14229527e-14]]
Best value found: x* = [1.05364292e-14 1.14229527e-14] with f(x*) = 1.630526937812
0126e-27
Optimizing with binary search limited by 5 iterations
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.25000000e+00  2.50000000e+00]
 [ 0.00000000e+00  1.87500000e+00]
 [-7.03125000e-01  1.17187500e+00]
 [-8.78906250e-02  9.96093750e-01]
 [-3.29589844e-01  6.55517578e-01]
 [ 1.83105469e-03  4.92553711e-01]
 [-1.85623169e-01  3.07159424e-01]
 [-2.23731995e-02  2.61583328e-01]
 [-8.69071484e-02  1.71879530e-01]
 [ 9.67383385e-04  1.29393339e-01]
 [-4.90061939e-02  8.05080682e-02]
 [-5.68742864e-03  6.86948653e-02]
 [-2.29168602e-02  4.50670766e-02]
 [-5.44172362e-03  3.67607454e-02]
 [-1.29386491e-02  2.11012345e-02]
 [-1.44363840e-03  1.80402650e-02]
 [-6.04328017e-03  1.18165300e-02]
 [-1.40955867e-03  9.65156132e-03]
 [-3.41622199e-03  5.53056000e-03]
 [-3.65849004e-04  4.73768324e-03]]
Best value found: x* = [-0.00036585  0.00473768] with f(x*) = 3.9028942216420154e-
05
Optimizing with golden ration
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-8.47196374e-01  2.66112145e+00]
 [ 1.56826886e-01  1.58286544e-01]
 [-2.69443378e-02  8.45731825e-02]
 [ 5.03661253e-03  5.14671226e-03]
 [-8.81102558e-04  2.76417053e-03]
 [ 1.65839843e-04  1.70957232e-04]
 [-2.93809460e-05  9.21504695e-05]
 [ 5.54777064e-06  5.74240819e-06]
 [-9.89020973e-07  3.10036295e-06]
```

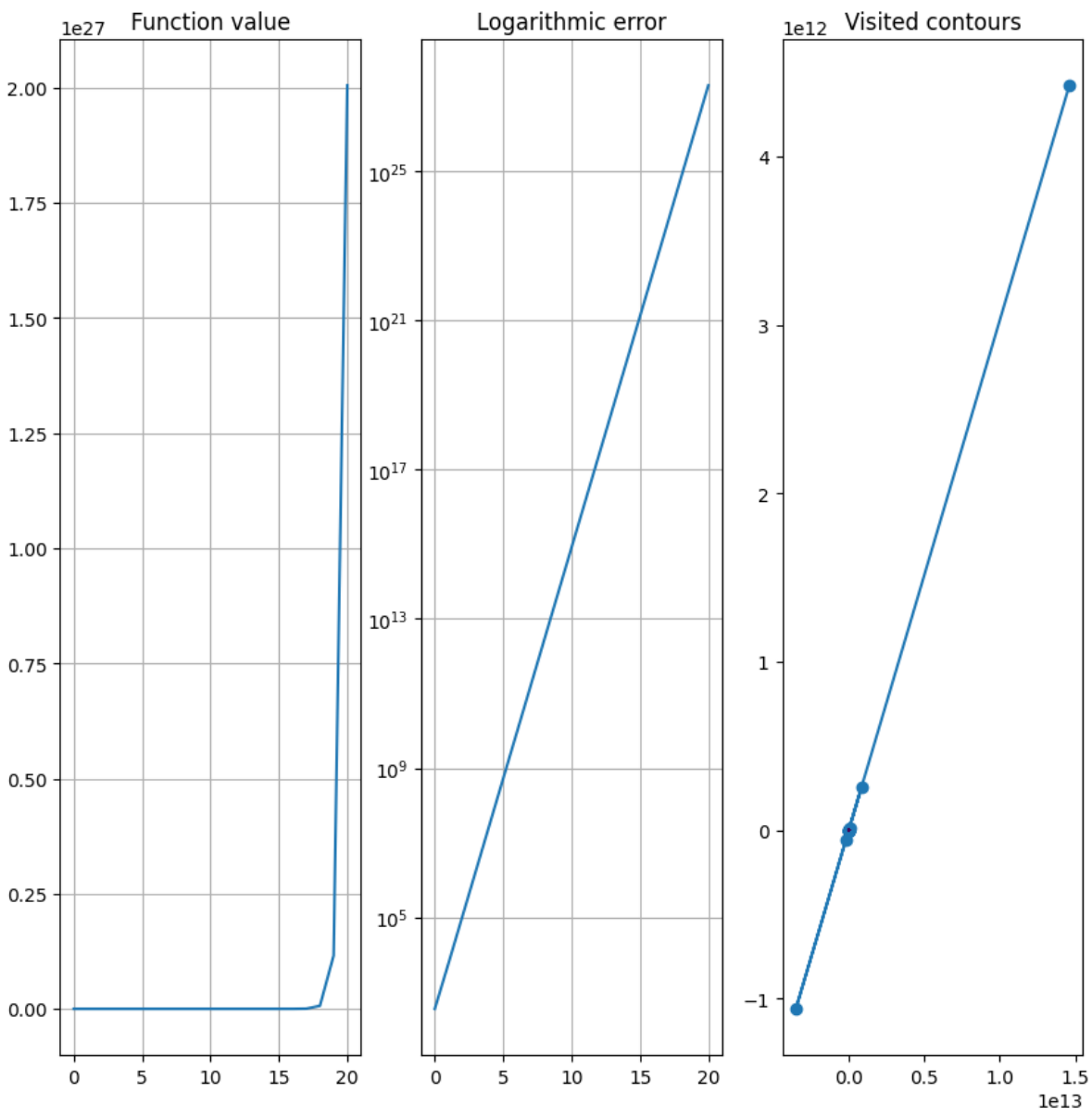
```
 [ 1.87987014e-07  1.96301320e-07]
 [-3.39519666e-08  1.06358144e-07]
 [ 6.50972824e-09  6.87752499e-09]
 [-1.19567611e-09  3.74370476e-09]
 [ 2.30667145e-10  2.45759969e-10]
 [-4.28952077e-11  1.34214833e-10]
 [ 8.34404579e-12  8.99003964e-12]
 [-1.57711215e-12  4.93081066e-12]
 [ 3.09604607e-13  3.37819258e-13]
 [-5.95734389e-14  1.86183068e-13]
 [ 1.17478872e-14  1.28987164e-14]]
Best value found: x* = [1.17478872e-14 1.28987164e-14] with f(x*) = 2.042987269538
964e-27
Optimizing with fibonacci search limited by 30 iterations:
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-8.46825560e-01  2.66126978e+00]
 [ 1.55688569e-01  1.55863956e-01]
 [-2.64138570e-02  8.29983737e-02]
 [ 4.86500808e-03  4.88144576e-03]
 [-8.28096259e-04  2.60189731e-03]
 [ 1.52660090e-04  1.53344616e-04]
 [-2.60290051e-05  8.17729157e-05]
 [ 4.80714775e-06  4.83966178e-06]
 [-8.22339360e-07  2.58330398e-06]
 [ 1.52004333e-07  1.53197944e-07]
 [-2.60460014e-08  8.18102972e-08]
 [ 4.82314805e-09  4.87193956e-09]
 [-8.29145592e-10  2.60417445e-09]
 [ 1.53670831e-10  1.55398317e-10]
 [-2.64625225e-11  8.31025249e-11]
 [ 4.91311234e-12  4.97917285e-12]
 [-8.48730551e-13  2.66516395e-12]
 [ 1.57718316e-13  1.60022936e-13]
 [-2.72932666e-14  8.56948355e-14]
 [ 5.08047675e-15  5.16588059e-15]]
Best value found: x* = [5.08047675e-15 5.16588059e-15] with f(x*) = 3.648431412509
384e-28
Optimizing with backtracking method
Optimizer trajectory:
[[ 5.00000000e+00  5.00000000e+00]
 [-1.25000000e+00  2.50000000e+00]
 [ 0.00000000e+00  1.87500000e+00]
 [-9.37500000e-01  9.37500000e-01]
 [-2.34375000e-01  9.37500000e-01]
 [-2.34375000e-01  2.34375000e-01]
 [-5.85937500e-02  2.34375000e-01]
 [-5.85937500e-02  5.85937500e-02]
 [-1.46484375e-02  5.85937500e-02]
 [-1.46484375e-02  1.46484375e-02]
 [-3.66210938e-03  1.46484375e-02]
 [-3.66210938e-03  3.66210938e-03]
 [-9.15527344e-04  3.66210938e-03]
 [-9.15527344e-04  9.15527344e-04]
 [-2.28881836e-04  9.15527344e-04]
 [-2.28881836e-04  2.28881836e-04]
```

```
 [-5.72204590e-05  2.28881836e-04]
 [-5.72204590e-05  5.72204590e-05]
 [-1.43051147e-05  5.72204590e-05]
 [-1.43051147e-05  1.43051147e-05]
 [-3.57627869e-06  1.43051147e-05]]
Best value found: x* = [-3.57627869e-06  1.43051147e-05] with f(x*) = 3.0695446184
83633e-10
```

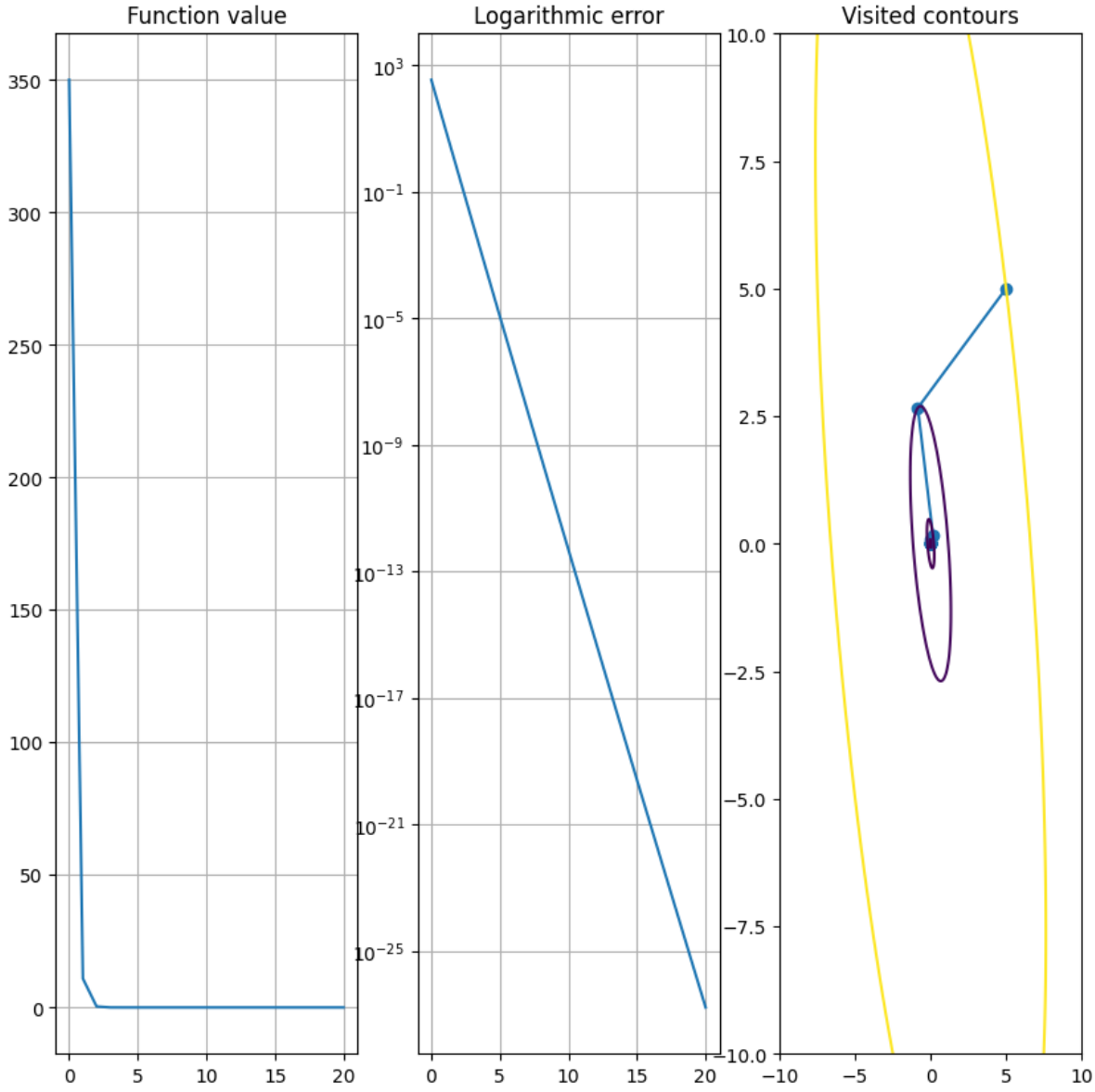


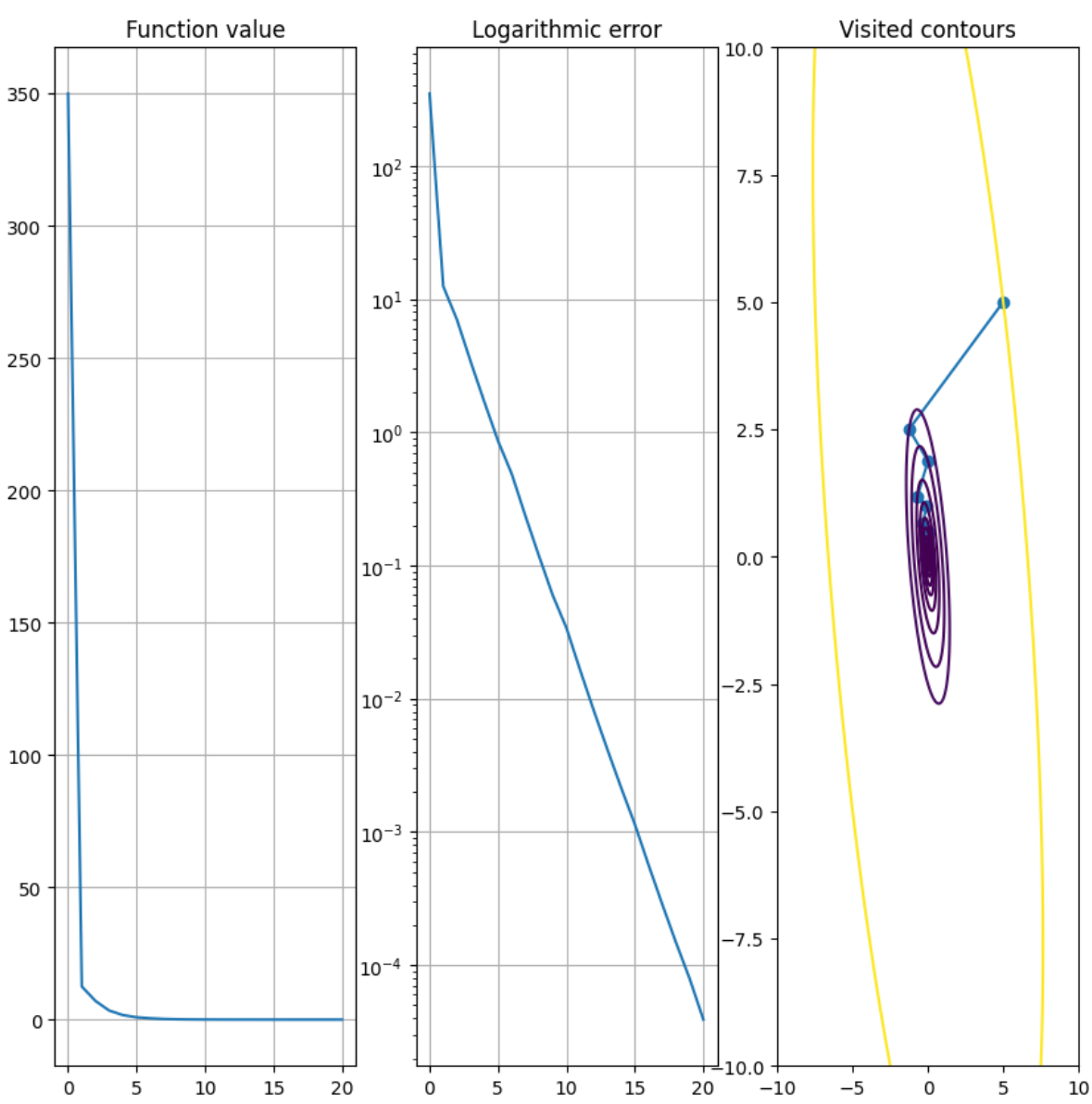
Optimizing with fixed step = 0.1



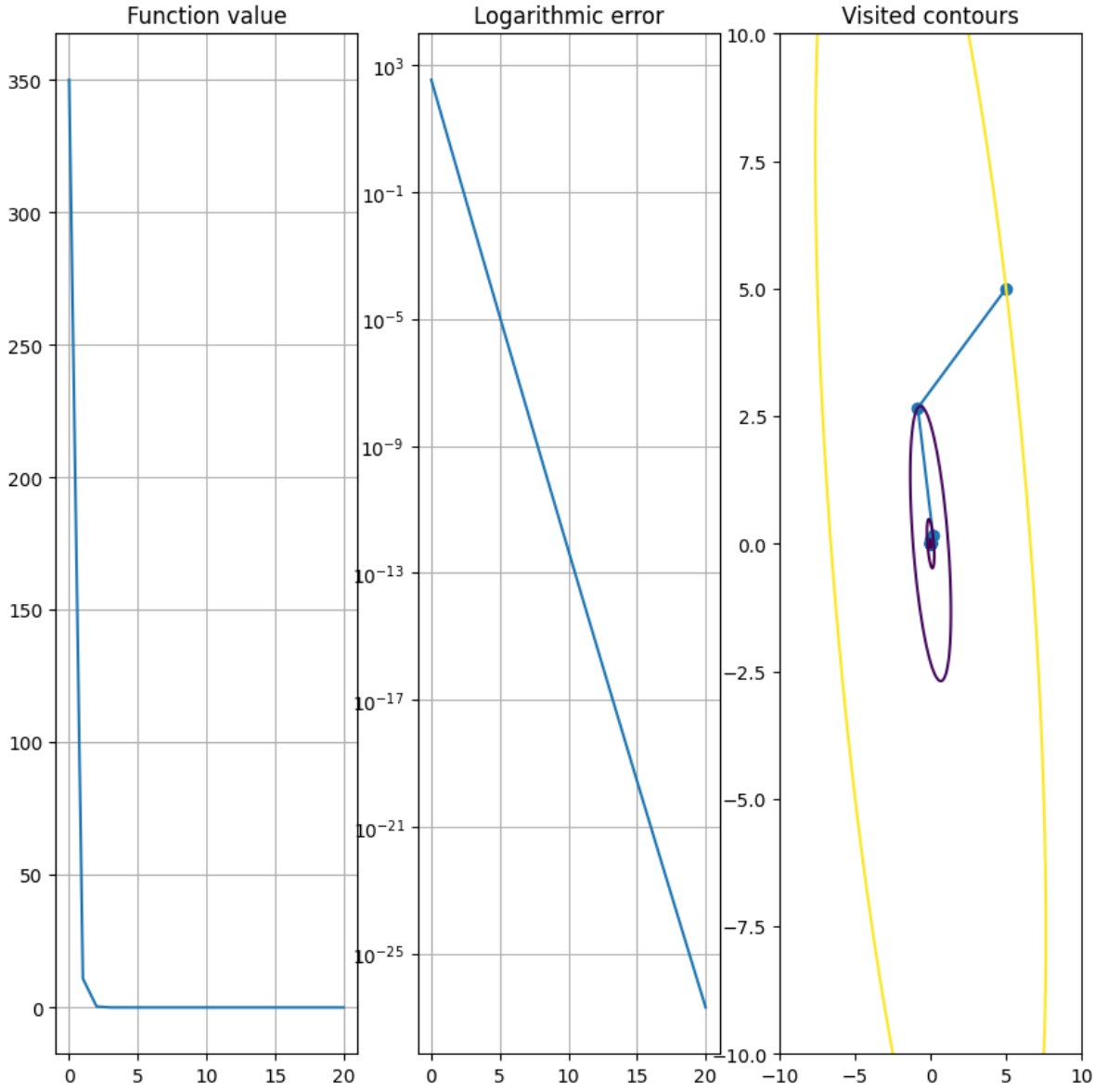

Optimizing with fixed step = 0.3
Function value
1e27
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
Logarithmic error
10^{25}
10^{21}
10^{17}
10^{13}
10^{9}
10^{5}
0
5
10
15
20
Visited contours
1e12
4
3
2
1
0
−1
0.0
0.5
1.0
1.5
1e13

Optimizing with binary search

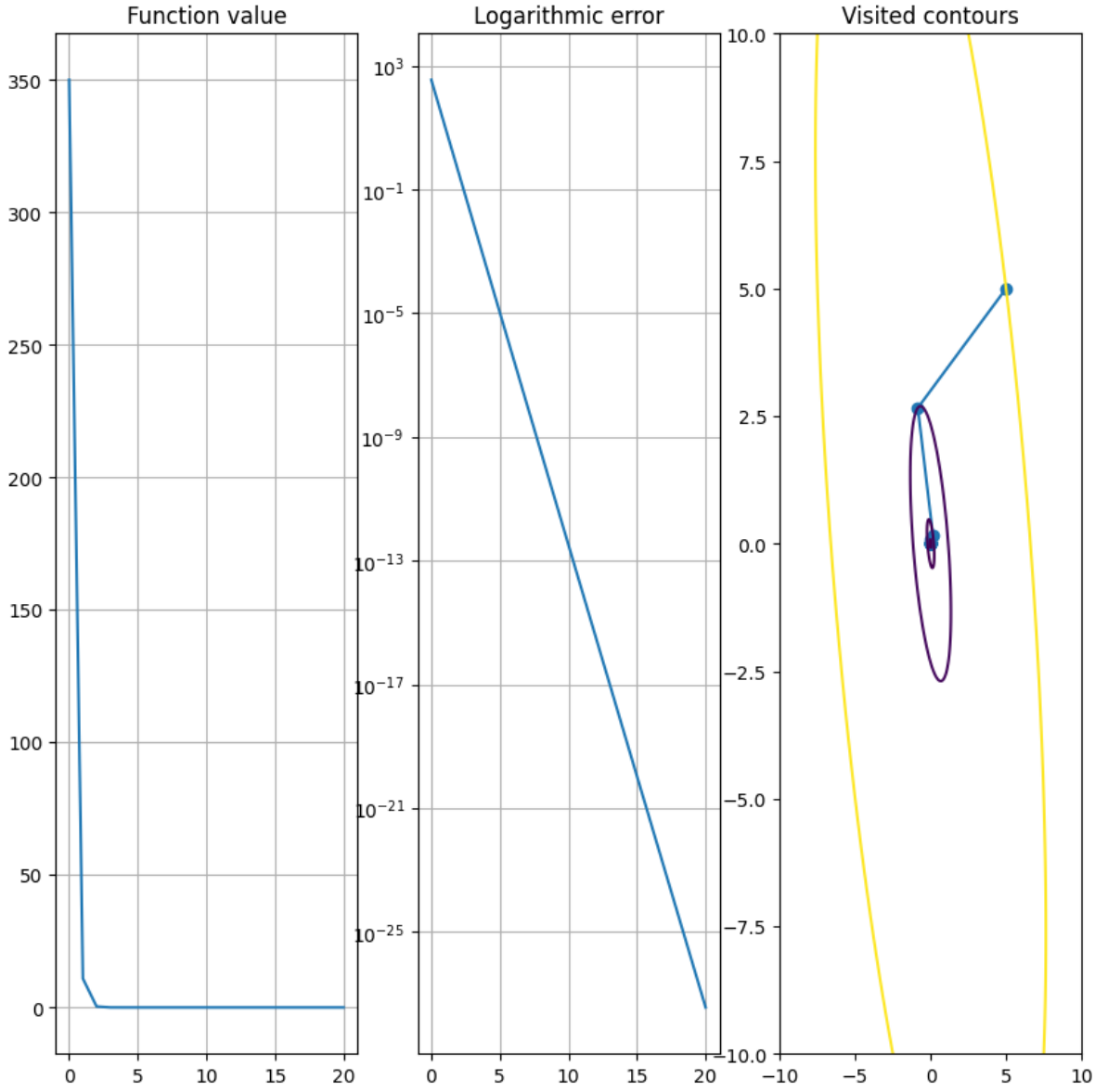


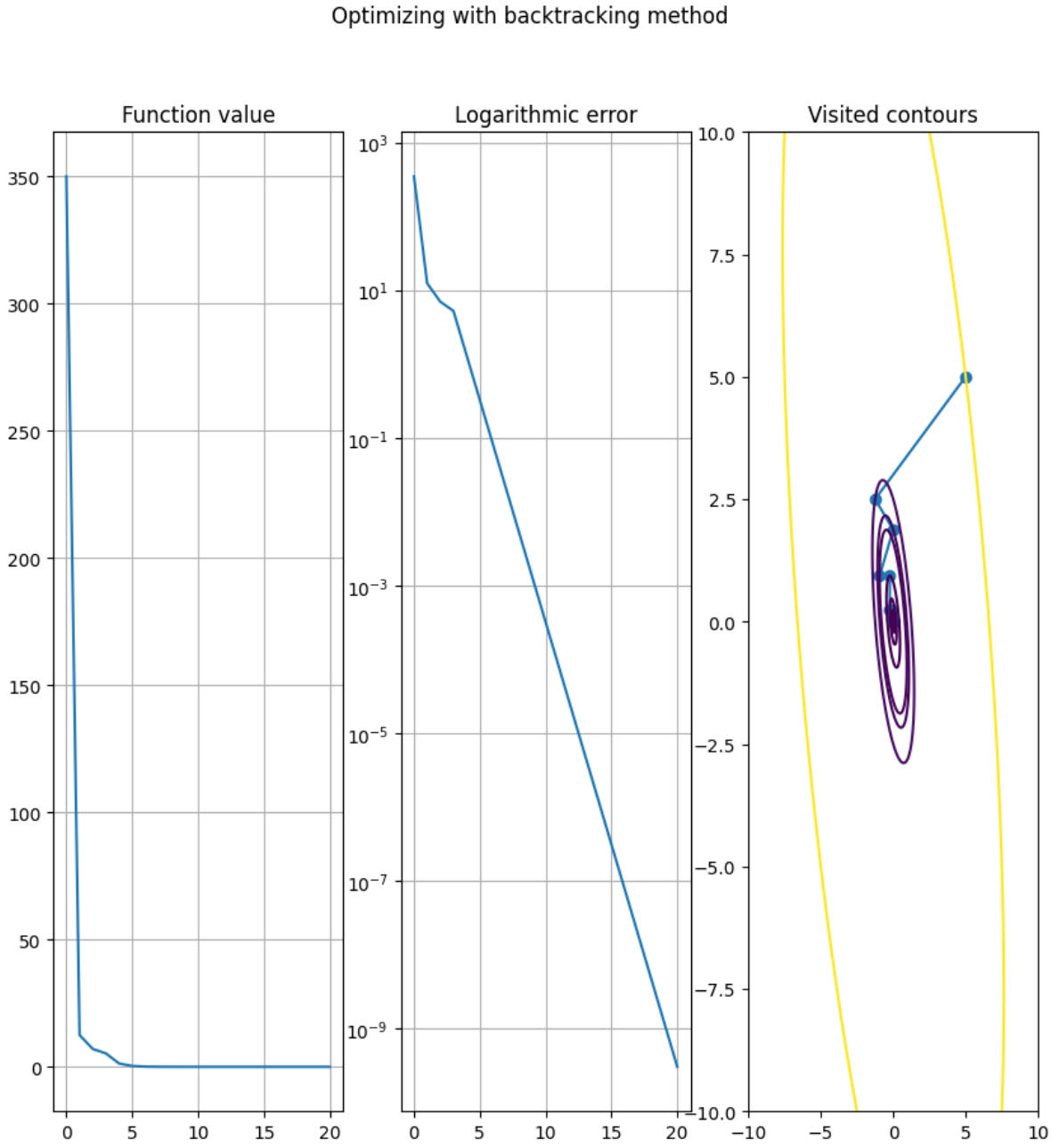

Optimizing with binary search limited by 5 iterations
Function value
Logarithmic error
Visited contours

Optimizing with golden ration



Optimizing with fibonacci search limited by 30 iterations:





Канонизация квадратичных функций от двух переменных.

Пусть дана квадратичная функция от двух переменных $ax^2 + 2bxy + cy^2 + dx + ey + f$ и требуется найти $\min$. Очевидно, что константу f можно вынести из операции нахождения минимума, поэтому задача сводится к $\min ax^2 + 2bxy + cy^2 + dx + ey$. Хочется избавиться от линейной части: $ax^2 + 2bxy + cy^2 + dx + ey = a(x - x_0)^2 + 2b(x - x_0)(y - y_0) + c(y - y_0)^2 + C$. Из этого получается набор уравнений:

$$\begin{cases} ax_0 + by_0 = d \\ bx_0 + cy_0 = e \end{cases} \tag{1}$$

Из которой по теореме Крамера:

$$x_0 = \frac{\begin{vmatrix} d & b \\ e & c \end{vmatrix}}{\begin{vmatrix} a & b \\ b & c \end{vmatrix}} \; y_0 = \frac{\begin{vmatrix} a & d \\ b & e \end{vmatrix}}{\begin{vmatrix} a & b \\ b & c \end{vmatrix}}$$

Итого: нахождение минимума квадратичной функции от двух переменных сводится к нахождению минимума функции

$$ax^2 + 2bxy + cy^2$$

Классификация квадратичных форм.

1. Положительно определенная форма. У такой формы минимум находиться в точке $(0, 0)$, потому что во всех остальных точках форма положительна.
2. Отрицательно определенная форма. Если форма $f(x, y)$ отрицательна, то форма $-f(x, y)$ положительна, а значит у нее есть глобальный максимум в точке $(0, 0)$ и нет минимума, что делает задачу нахождения минимума бессмысленной.
3. Неопределенная форма. У такой формы нет минимума и максимума, но есть стационарная точка в $(0, 0)$. Поэтому в зависимости от начальной точки градиентный спуск либо уйдет в сторону бесконечного убывания функции, либо сойдется к стационарной точке.
4. Исключение линейной части невозможно в случае, когда определитель формы равен 0. Получается параболический цилиндр, у которого может быть минимум вдоль прямой или не быть минимума. Если выполнено условие $\frac{a}{b} = \frac{b}{c} = \frac{d}{e}$, то минимум будет достигаться вдоль прямой $2ax + 2by + d = 0$, иначе у функции отсутствует минимум.