Назарова К.А. Группа № в5130904/30030

Расчетная работа №4. Параметрическая идентификация модели

1. **Задание. Вариант 10.**

$$W(s) = \frac{k(1-as)}{(1+b_1 s)(1+b_2 s)}$$

x(t) = const = 3 k=3 a=2 b_1=0.8 b_2=4

2. **Применим метод введения дополнительной переменной:**

$$W(s) = \frac{k(1-as)}{(1+b_1 s)(1+b_2 s)} = \frac{y(s)}{x(s)} = \frac{P(s)}{Q(s)}$$

$W(s) = W_1(s) * W_2(s)$ - при последовательном включении.

В решаемом случае $W_1(s) = \frac{1}{Q(s)}$, $W_2(s) = P(s)$

$W_1(s) = \frac{1}{Q(s)} = \frac{u(s)}{x(s)}$,

$W_2(s) = P(s) = \frac{y(s)}{u(s)}$

$$\begin{cases} \bar{x}(s) = (1+b_1 s)(1+b_2 s)\bar{U}(s) \\ \bar{y}(s) = k(1-as)\bar{U}(s) \end{cases}$$

Раскрыв скобки и приведя подобные, получим:

$$\begin{cases} \bar{x}(s) = (1+b_1 s + b_2 s + b_1 b_2 s)\bar{U}(s) \\ \bar{y}(s) = (k - kas)\bar{U}(s) \end{cases}$$

Перейдём в вещественную форму:

$\bar{x}(s) \longrightarrow x(t)$

$\bar{y}(s) \longrightarrow y(t)$ $\qquad S^n \longrightarrow \frac{d^n}{dt^n}$

$\bar{U}(s) \longrightarrow U(t)$

$$\begin{cases} \bar{x}(t) = U(t) + b_1 U'(t) + b_2 U'(t) + b_1 b_2 U''(t) \\ \bar{y}(t) = kU'(t) - kaU''(t) \end{cases}$$

Выразим из уравнения с x(t) старшую производную:

$$U''(t) = \frac{\bar{x}(t) - U(t) - (b_1 + b_2)U'(t)}{b_1 b_2}$$

Чтобы применить метод Эйлера, приведем систему к канонической форме. Обозначим:

$$z_1 = U(t)$$
$$z_2 = U'(t) = z_1'(t)$$

Переписав всё в соответствии с произведенной заменой, получим систему уравнений:

$$z_1' = z_2$$
$$z_2' = \frac{x - z_1 - (b_1 + b_2)z_2}{b_1 b_2}$$

Далее решаем систему явным методом Эйлера (при нулевых начальных условиях), начальные условия: $z_{1,0} = 0$; $z_{2,0} = 0$.

$$z_{1,n+1} = z_{1,n} + hz_{2,n}$$
$$z_{2,n+1} = z_{2,n} + h\frac{x - z_1 - (b_1 + b_2)z_2}{b_1 b_2}$$

И далее, на каждом шаге, подставив полученные значения, рассчитываем

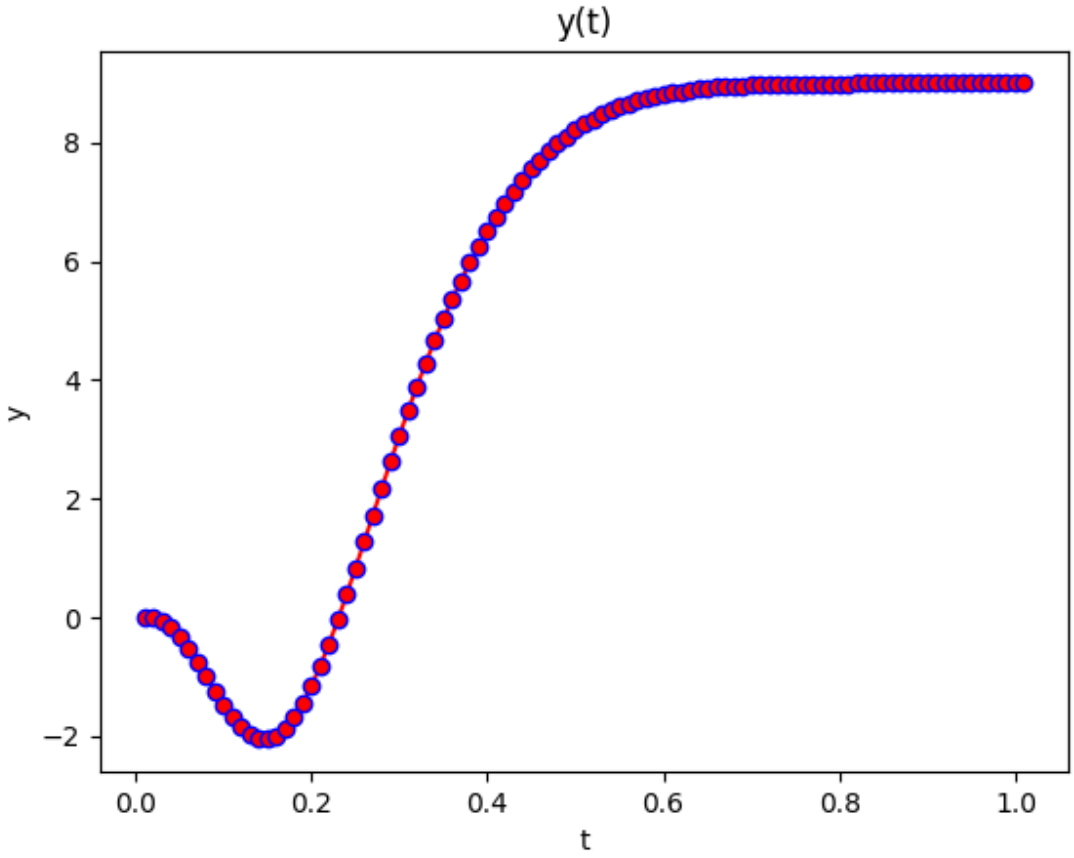
$$y = k(z_{1,n} - az_{2,n})$$

3. **Таблица выходных значений Y(t)**

	t	y		t	y		t	y
0	0.01	0.00000	35	0.36	5.35527	70	0.71	8.97261
1	0.02	0.00000	36	0.37	5.67375	71	0.72	8.97735
2	0.03	-0.05622	37	0.38	5.97288	72	0.73	8.98140
3	0.04	-0.16647	38	0.39	6.25278	73	0.74	8.98470
4	0.05	-0.32517	39	0.40	6.51369	74	0.75	8.98752
5	0.06	-0.52380	40	0.41	6.75606	75	0.76	8.98977
6	0.07	-0.75141	41	0.42	6.98040	76	0.77	8.99172
7	0.08	-0.99513	42	0.43	7.18740	77	0.78	8.99331
8	0.09	-1.24104	43	0.44	7.37769	78	0.79	8.99457
9	0.10	-1.47513	44	0.45	7.55211	79	0.80	8.99562
10	0.11	-1.68381	45	0.46	7.71138	80	0.81	8.99643
11	0.12	-1.85490	46	0.47	7.85634	81	0.82	8.99715
12	0.13	-1.97814	47	0.48	7.98789	82	0.83	8.99772
13	0.14	-2.04567	48	0.49	8.10684	83	0.84	8.99823
14	0.15	-2.05212	49	0.50	8.21400	84	0.85	8.99859
15	0.16	-1.99491	50	0.51	8.31030	85	0.86	8.99892
16	0.17	-1.87392	51	0.52	8.39652	86	0.87	8.99913
17	0.18	-1.69143	52	0.53	8.47350	87	0.88	8.99931
18	0.19	-1.45164	53	0.54	8.54193	88	0.89	8.99946
19	0.20	-1.16031	54	0.55	8.60262	89	0.90	8.99961
20	0.21	-0.82410	55	0.56	8.65626	90	0.91	8.99967
21	0.22	-0.45036	56	0.57	8.70357	91	0.92	8.99979
22	0.23	-0.04674	57	0.58	8.74509	92	0.93	8.99982
23	0.24	0.37935	58	0.59	8.78142	93	0.94	8.99985
24	0.25	0.82080	59	0.60	8.81310	94	0.95	8.99994
25	0.26	1.27128	60	0.61	8.84064	95	0.96	8.99988
26	0.27	1.72494	61	0.62	8.86449	96	0.97	8.99997
27	0.28	2.17683	62	0.63	8.88516	97	0.98	8.99997
28	0.29	2.62266	63	0.64	8.90298	98	0.99	8.99997
29	0.30	3.05892	64	0.65	8.91828	99	1.00	8.99997
30	0.31	3.48285	65	0.66	8.93139	100	1.01	8.99997
31	0.32	3.89214	66	0.67	8.94255	101	1.02	8.99997
32	0.33	4.28511	67	0.68	8.95200			
33	0.34	4.66050	68	0.69	8.96007			
34	0.35	5.01741	69	0.70	8.96688			

4. **Полученный график Y(t)**

Рис. 1: График Y(t)

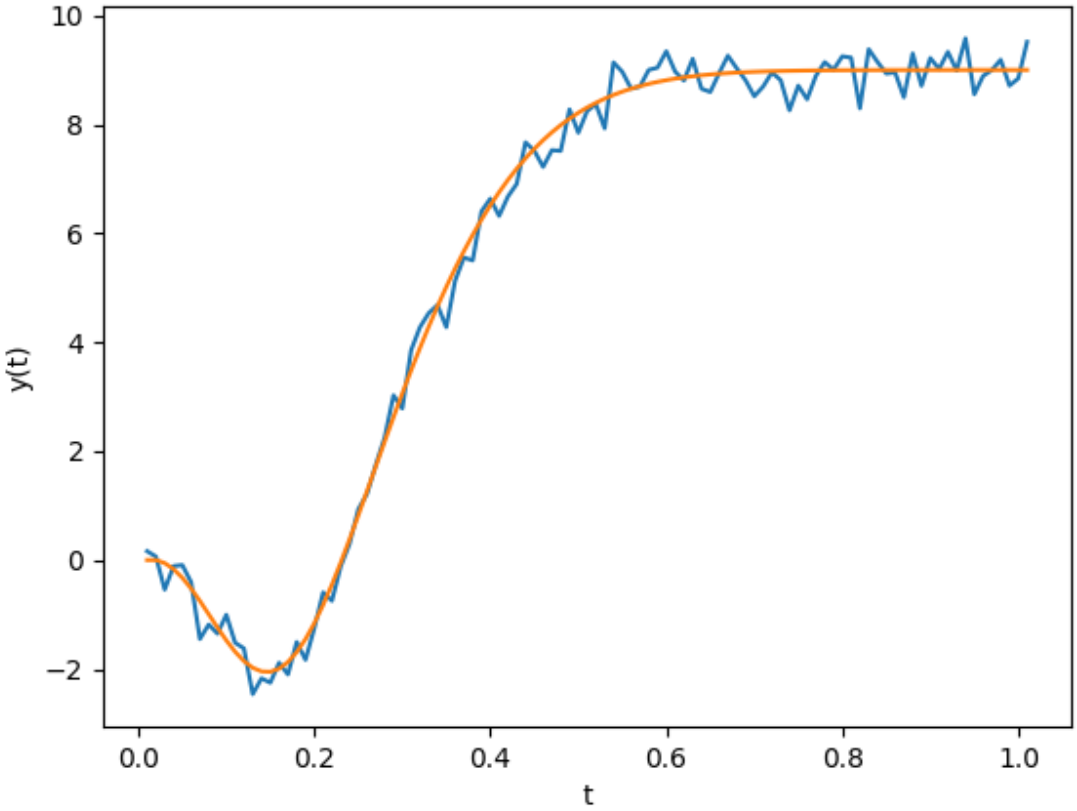


5. **Получение «зашумленных» значений выходного сигнала и их график**

Используя ГСЧ из лабораторной работы №3 был получен параметр $\sigma_x = \Delta y$, который суммируется с полученными в первой работе значениями y(t).

$$\sigma_x = 0,05 * MAX|y(t)| = 0.05 * 8.99997 = 0,45$$

Рис. 2: "Зашумленный"график



6. **Осуществление параметризации модели**

Пусть неизвестны a, b_1 в передаточной функции:

$$W(s) = \frac{k(1-as)}{(1+b_1 s)(1+b_2 s)}$$

Целевая функция будет иметь следующий вид:

$$CF = \frac{1}{n+1}\sum_{i=0}^{n}(y_i^E - y_i^M)^2 \to min = 1$$

Далее требуется применить **метод поисковой оптимизации** из лабораторной работы №2. С его помощью будет осущетсвляться поиск минимума целевой функции. Причем заданые значения $a = 2.0$, $b_1 = 0.8$

7. **Результат вычисления 1 итерации**

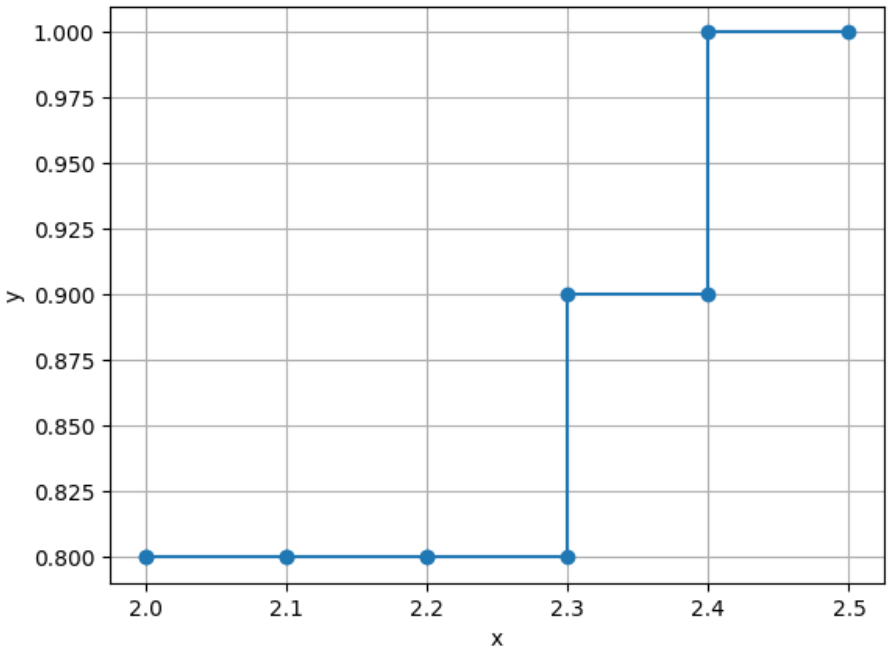
Начальная точка - (2.5, 1.0). Шаг - 0.1.

Таблица 2: Результат вычисления 1 итерации

a	b1	cf
2.5	1.0	0.2301273907921569
2.4	1.0	0.1897149552127452
2.4	0.9000000000000001	0.16124511697941177
2.3	0.9000000000000001	0.13048205677352942
2.3	0.8000000000000003	0.11382225191764711
2.1999999999999997	0.8000000000000003	0.09310974705980395
2.1999999999999997	0.7000000000000002	0.09063502251470593
2.0999999999999996	0.7000000000000002	0.08031409626666673
2.0999999999999996	0.7000000000000001	0.08031409626666673
1.9999999999999998	0.7000000000000001	0.07869046586372551
1.9999999999999998	0.8	0.07652744229313732
1.9999999999999998	0.8	0.07652744229313732
1.9999999999999998	0.8	0.07652744229313732

Минимальная точка - (1.9999999999999998, 0.8).

Рис. 3: Траектория движения к минимуму



8. **Результат вычисления 2 итерации**

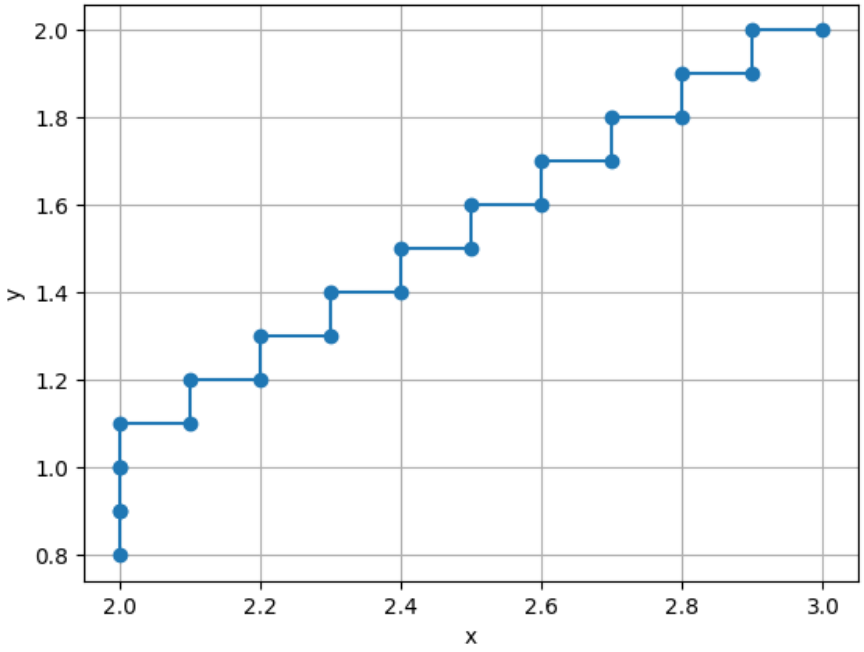
Начальная точка - (3.0, 2.0). Шаг - 0.1.

Таблица 3: Таблица полученных значений 2 итерации

a	**b1**	**cf**
3.0	2.0	1.201904971866667
2.9	2.0	1.1127685212333334
2.9	1.9000000000000001	1.0356386276480396
2.8	1.9000000000000001	0.9533535891509803
2.8	1.8	0.8793103465049015
2.6999999999999997	1.8	0.8042140837529408
2.6999999999999997	1.7000000000000002	0.7336457771000001
2.5999999999999996	1.7000000000000002	0.6660998683901964
2.5999999999999996	1.6000000000000003	0.5995822421372549
2.4999999999999996	1.6000000000000003	0.5399665155725494
2.4999999999999996	1.5000000000000004	0.47809897869901946
2.3999999999999995	1.5000000000000004	0.426833586738235
2.3999999999999995	1.4000000000000006	0.37037641911862734
2.2999999999999994	1.4000000000000006	0.3278830399284313
2.2999999999999994	1.3000000000000007	0.27763181773235285
2.1999999999999993	1.3000000000000007	0.24437297556666676
2.1999999999999993	1.2000000000000008	0.20120734023235287
2.099999999999999	1.2000000000000008	0.17766961523235295
2.099999999999999	1.100000000000001	0.1427107843107842
1.9999999999999993	1.100000000000001	0.129398180704902
1.9999999999999993	1.000000000000001	0.10381219364607844
1.8999999999999992	1.000000000000001	0.10127277428431372
1.8999999999999992	0.9000000000000012	0.08652726585392159
1.9999999999999993	0.9000000000000012	0.08565130717254904
1.9999999999999993	0.8000000000000014	0.07652744229313732
1.9999999999999993	0.8000000000000014	0.07652744229313732
1.9999999999999993	0.8000000000000014	0.07652744229313732

Минимальная точка - (1.9999999999999993, 0.8000000000000014).

Рис. 4: Траектория движения к минимуму



9. **Результат вычисления 3 итерации**

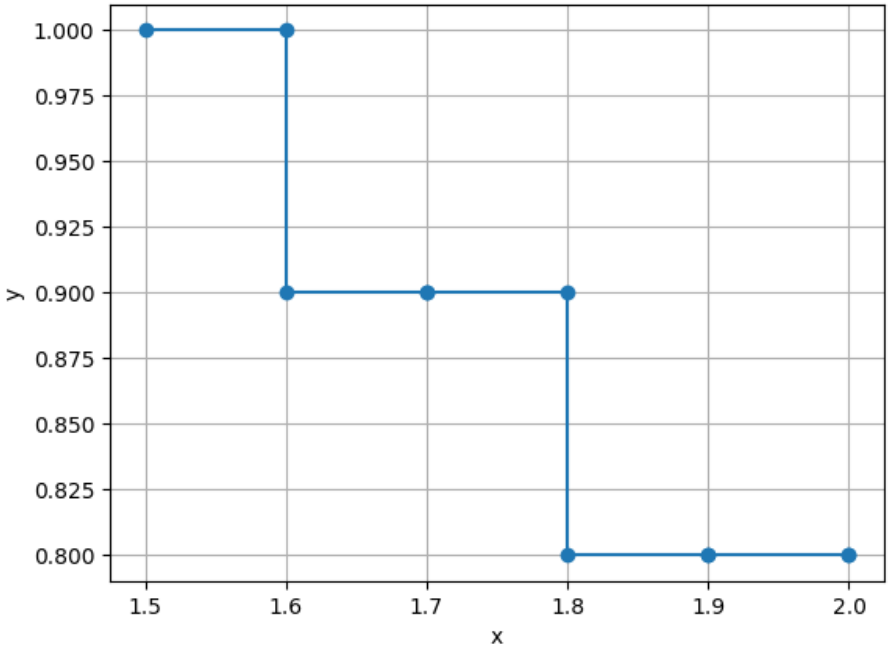
Начальная точка - (1.5, 0.8). Шаг - 0.1.

Таблица 4: Таблица полученных значений 3 итерации

a	**b1**	**cf**
1.5	1.0	0.15959993137254908
1.6	1.0	0.1330966274509805
1.6	0.9000000000000001	0.1317313627450981
1.7000000000000002	0.9000000000000001	0.10848305882352945
1.7000000000000002	0.9000000000000002	0.10848305882352945
1.8000000000000003	0.9000000000000002	0.09313244117647067
1.8000000000000003	0.8000000000000003	0.09251376470588243
1.9000000000000004	0.8000000000000003	0.08188862745098044
1.9000000000000004	0.8000000000000002	0.08188862745098044
2.0000000000000004	0.8000000000000002	0.07948902941176471
2.0000000000000004	0.8000000000000002	0.07948902941176471
2.0000000000000004	0.8000000000000002	0.07948902941176471
2.0000000000000004	0.8000000000000002	0.07948902941176471

Минимальная точка - (2.0000000000000004, 0.8000000000000002).

Рис. 5: Траектория движения к минимуму



10. **Результат вычисления 4 итерации**

Начальная точка - (1.0, 0.5). Шаг - 0.1.

Таблица 5: Таблица полученных значений 4 итерации

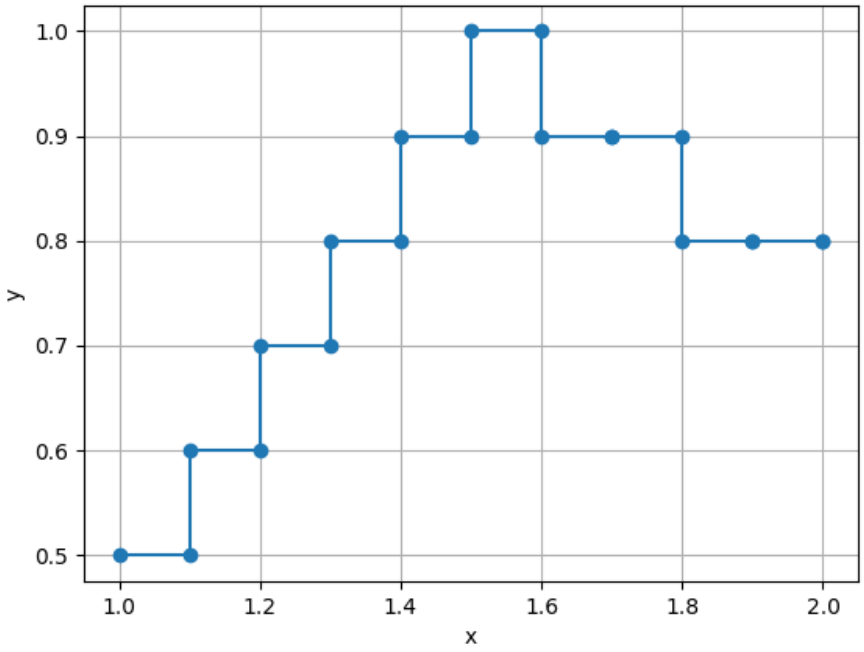
a	**b1**	**cf**
1.0	0.5	0.6654829705882351
1.1	0.5	0.5694585000000001
1.1	0.6	0.5004751568627451
1.2000000000000002	0.6	0.41966953921568645
1.2000000000000002	0.7	0.3695251470588235
1.3000000000000003	0.7	0.3034094117647058
1.3000000000000003	0.7999999999999999	0.2700151470588236
1.4000000000000004	0.7999999999999999	0.21799217647058827
1.4000000000000004	0.8999999999999999	0.201830294117647
1.5000000000000004	0.8999999999999999	0.16277373529411776
1.5000000000000004	0.9999999999999999	0.15931699019607848
1.6000000000000005	0.9999999999999999	0.13291063725490207
1.6000000000000005	0.8999999999999999	0.1317313627450981

Продолжение далее

a	b1	cf
1.7000000000000006	0.8999999999999999	0.10848305882352945
1.7000000000000006	0.8999999999999998	0.10848305882352945
1.8000000000000007	0.8999999999999998	0.09313244117647067
1.8000000000000007	0.7999999999999998	0.09251376470588243
1.9000000000000008	0.7999999999999998	0.08188862745098044
1.9000000000000008	0.7999999999999997	0.08188862745098044
2.000000000000001	0.7999999999999997	0.07948902941176471
2.000000000000001	0.7999999999999997	0.07948902941176471
2.000000000000001	0.7999999999999997	0.07948902941176471
2.000000000000001	0.7999999999999997	0.07948902941176471

Минимальная точка - (2.000000000000001, 0.7999999999999997).

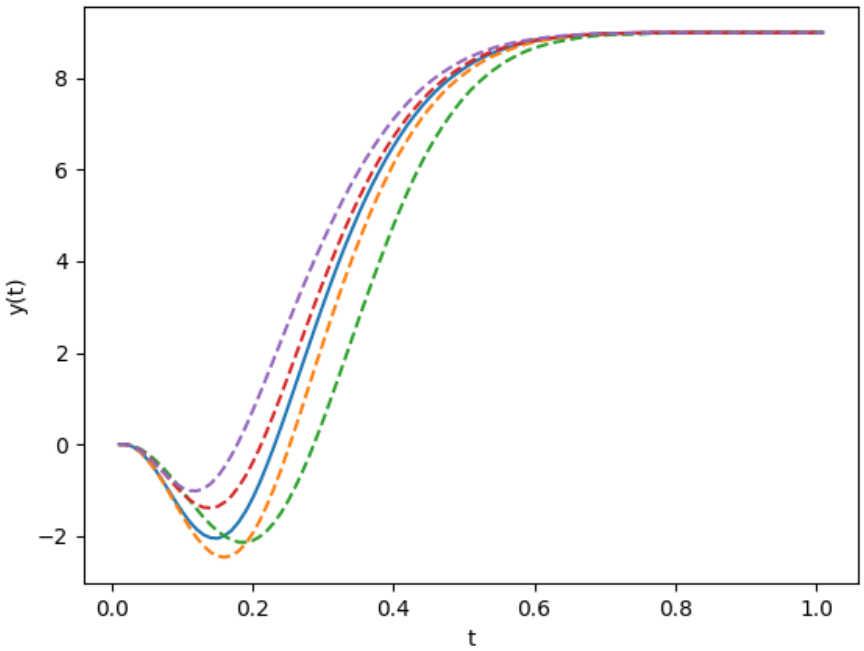
Рис. 6: Траектория движения к минимуму



11. **Общие результаты**

Таблица 6: Таблица сводных результатов вычислений

	Начальная точка	Полученные результаты
a	2.5	1.9999999999999998
b1	1.0	0.8000000000000002
CF		0.07652744229313732
a	3.0	1.8999999999999995
b1	2.0	0.8000000000000014
CF		0.07652744229313732
a	1.5	2.0000000000000004
b1	0.8	0.8000000000000002
CF		0.07948902941176471
a	1.0	2.000000000000001
b1	0.5	0.7999999999999997
CF		0.07948902941176471



12. Код программы

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import csv
import random
import math

def calc(a, b_1):
    z_1 = [0]
    z_2 = [0]
    s = 0.01
    h = s
    t = [h]
    y = [0]

    for i in range(1, 101):
        z_1.append(round(z_1[i - 1] + h * z_2[i - 1], 5))
        z_2.append(round(z_2[i - 1] + h * ((3.0 - z_1[i - 1] - (b_1
            + 4.0) * z_2[i - 1]) / (b_1 * 4.0)), 5))
        y.append(round(3.0 * z_1[i - 1] - (3.0 * a) * z_2[i - 1],
           5))
        h += s
        t.append(round(h, 5))
    func_dict = {'t': t, 'y': y}
    return func_dict


def drawGraph(dest_file, dest_csv, func_dict, line_style='-', flag=
    False):
    dataframe = pd.DataFrame(func_dict)
    dataframe.to_csv(dest_csv, index=False)
    df = pd.read_csv(dest_csv)
    x = df['t']
    y = df['y']
    plt.xlabel('t')
    plt.ylabel('y(t)')
    plt.plot(x, y, line_style)
    plt.savefig(dest_file)
    if flag:
```

```
        plt.close()


delta_y = round(0.05 * 8.99997, 5)
iter = 100000

def calcVariables(num_list: list):
    m_calc = round(sum(num_list) / iter, 5)
    d_calc = round(sum([(num - m_calc) ** 2 for num in num_list]) /
        iter, 5)
    sigma_calc = round(d_calc ** 0.5, 5)
    return m_calc, d_calc, sigma_calc


def mullerGenerator():
    r1, r2 = random.uniform(0, 1), random.uniform(0, 1)
    z = math.cos(2 * math.pi * r1) * (-2 * math.log10(r2)) ** 0.5
    x = z * delta_y
    return x

def create_hist(numbers: list, dest_graph, bins):
    counts, edges, bars = plt.hist(numbers, bins=bins, edgecolor='
       black', rwidth=0.9, color="green")
    plt.xlabel(' ')
    plt.ylabel(' ')
    plt.savefig(dest_graph)
    return dest_graph

eps = 0.01

def ellipsoid(x: list):
    A = 3.0
    B = 2.0
    return (float(x[0]) / A) ** 2 + (float(x[1]) / B) ** 2

def rosenbrok(x: list):
    return (1 - x[1]) ** 2 + 100 * (x[0] - x[1] ** 2) ** 2

def func(coordinates):
    mathFunc = calc(coordinates[0], coordinates[1])
```

```
75      return sum([(y_a - y_e) ** 2 for y_a, y_e in zip(mathFunc['y'],
            y_noised)]) / (len(mathFunc['y']) + 1)
76
77  def calc_min(func, s, coordinates, max_iter):
78      result = [coordinates.copy()]
79      current_iter = 0
80      f_1 = 0
81      f_2 = 1
82      while abs(f_1 - f_2) > eps and current_iter < max_iter:
83          for i in range(len(coordinates)):
84              value = func(coordinates)
85              coordinates[i] += s
86              new_value = func(coordinates)
87              if new_value >= value:
88                  coordinates[i] -= 2 * s
89                  new_value = func(coordinates)
90                  if new_value > value:
91                      coordinates[i] += s
92              result.append(coordinates.copy())
93          f_1 = f_2
94          f_2 = func(coordinates)
95          current_iter += 1
96      result.append(coordinates)
97      print("Iter Amount: ", current_iter)
98      print(coordinates)
99      return result
100
101 def exportResults (func, dest_file, dest_graph, selection, result)
        :
102     newResult = result [0::selection]
103     newResult.append(result[len(result)-1])
104     columns = ['a', 'b1', 'cf']
105     for coord in newResult :
106         coord.append(func(coord))
107     with open (dest_file, 'w') as f :
108         write = csv.writer(f)
109         write.writerow(columns)
110         write.writerows(newResult)
111     f.close()
112     df = pd.read_csv(dest_file, on_bad_lines ='skip')
113     x = df['a']
```

```
114     y = df['b1']
115     plt.xlabel('x')
116     plt.ylabel('y')
117     plt.grid(True)
118     plt.plot(x, y , "-o")
119     plt.savefig(dest_graph)
120     plt.close()
121 
122 def noise(a, b_1):
123     mathFunc = calc(a, b_1)
124     iter = len(mathFunc['y'])
125     noiseNum = [mullerGenerator() for c in range(iter)]
126     y_noised = [round(x, 5) + round(y, 5) for x, y in zip(mathFunc[
            'y'], [noiseNum[c] for c in range(iter)])]
127     noisedCoord = {'t': mathFunc['t'], 'y': y_noised}
128     drawGraph('laba_4_noize.png', 'laba_4_noize.csv', noisedCoord,
            flag=False)
129     drawGraph('laba_4_function_with_noize.png', '
            laba_4_function_with_noize.csv', mathFunc, flag=False)
130     return noisedCoord
131 
132 def cf(y_actual: list , y_noised: list):
133     return sum([(y_a - y_e) ** 2 for y_a, y_e in zip(y_actual,
            y_noised)]) / (len(y_actual) + 1)
134 
135 def function(coordinates):
136     mathFunc = calc(coordinates[0], coordinates[1])
137     return cf(y_actual=mathFunc['y'], y_noised=y_noised)
138 
139 def drawResult(a, b_1, line_style):
140     mathFunc = calc(a, b_1)
141     drawGraph('lab4_result.png', 'lab4_result.csv', mathFunc,
            line_style=line_style, flag=False)
142 
143 if __name__ == '__main__':
144     y_noised = noise(a=2.0, b_1=0.8)['y']
145 
146     find_min = calc_min(func=function , s=0.01, coordinates=[2.5,
            1.0], max_iter=100000)
147     exportResults(function , "exp1.csv", "exp1.png", 1, find_min)
148 
```

```
149     find_min = calc_min(func=function, s=0.01, coordinates=[3.0,
            2.0], max_iter=100000)
150     exportResults(function, "exp2.csv", "exp2.png", 1, find_min)
151
152     find_min = calc_min(func=function, s=0.01, coordinates=[1.8,
            0.8], max_iter=100000)
153     exportResults(function, 'exp3.csv', 'exp3.png', 1, find_min)
154
155     find_min = calc_min(func=function, s=0.01, coordinates=[1.0,
            0.5], max_iter=100000)
156     exportResults(function, 'exp4.csv', 'exp4.png', 1, find_min)
157
158     drawResult(2.0, 0.8, line_style='-')
159     drawResult(2.5, 1.0, line_style='--')
160     drawResult(3.0, 2.0, line_style='--')
161     drawResult(1.5, 0.8, line_style='--')
162     drawResult(1.0, 0.5, line_style='--')
```