

複雑系科学演習 第2回 レポート

複雑系知能学科 複雑系コース 3年 Iクラス番号 1019086 岩上慎之介

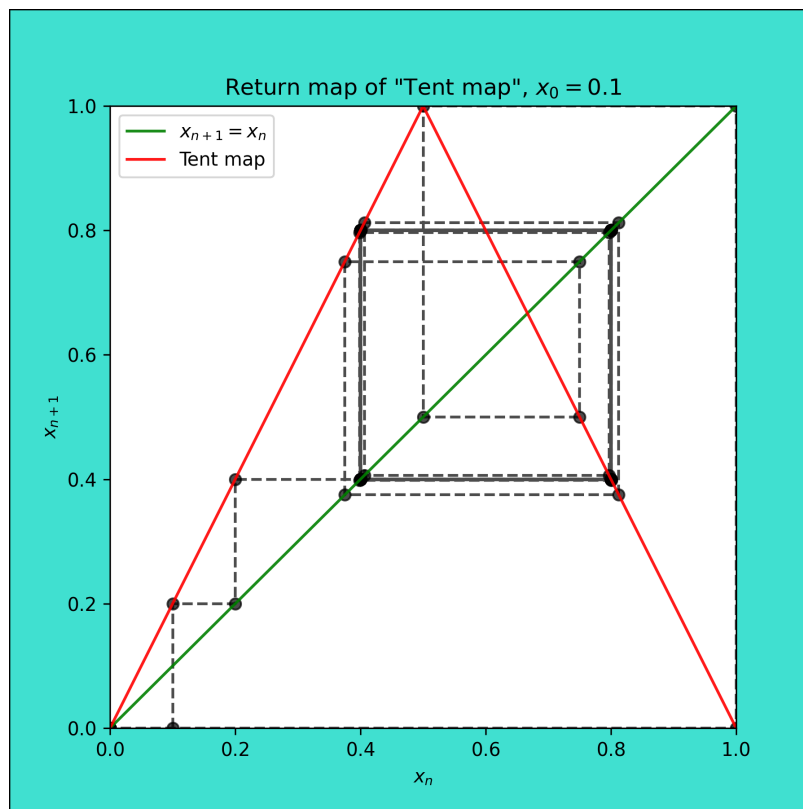
2021年12月6日

1 レポート課題 1st

1.1 課題 1

前頁の関数 `next` をテント写像に書き換え，リターンマップを描け

1.1.1 画像



1.1.2 結果と考察

1.1.3 ソースコード

Listing 1 task6

```
1 from matplotlib import pyplot as plt
2 import numpy as np
3
4
5 class Task6():
6     def __init__(self) -> None:
7         self.x = 0.1 # 初期値 x0 = 0.1
8         self.xn = np.linspace(0, 1, 1000) # 横軸の範囲と刻み幅
```

```

9         self.tent_y = []                                # テント写像のグラフを描くための配列
10        for i in self.xn:
11            if 0 <= i <= 0.5:
12                self.tent_y.append(2 * i)
13            else:
14                self.tent_y.append(2 * (1 - i))
15        self.filepath = 複雑系科学演習'/Week6/images/task6'
16
17    def tent(self) -> list:
18        リターンマップでのテント写像の座標を持つ配列を返す""
19        calc_x = self.x
20        x_array = [calc_x]
21        for _ in range(100):
22            if 0 <= calc_x <= 0.5:
23                calc_x = 2 * calc_x
24            else:
25                calc_x = 2 * (1 - calc_x)
26            x_array.append(calc_x)
27        return x_array
28
29    def plot_return_map(self) -> None:
30        リターンマップの描画 (テント写像) ""
31        plt.figure(figsize=(6, 6), facecolor='turquoise',
32                    linewidth=1, edgecolor='black')
33        n = self.tent()
34        spiper_plot_x = [self.x]                        # クモの巣図用の配列 (x)
35        spiper_plot_y = [0]                             # クモの巣図用の配列 (y)
36        for i in range(1, len(n)):
37            spiper_plot_x.append(n[i - 1])
38            spiper_plot_x.append(n[i])
39            spiper_plot_y.append(n[i])
40            spiper_plot_y.append(n[i])
41
42        plt.plot(spiper_plot_x, spiper_plot_y, marker='o',
43                linestyle='dashed', color='black', alpha=0.7)
44        plt.plot(self.xn, self.xn, color='green',
45                alpha=0.9, label="$x_{n+1} = x_n$")
46        plt.plot(self.xn, self.tent_y, color='red',
47                alpha=0.9, label="Tent map")
48        plt.title("Return map of \"Tent map\", $x_0 = $" + str(self.x))
49        plt.xlim(0, 1)
50        plt.ylim(0, 1)
51        plt.xlabel("$x_n$")
52        plt.ylabel("$x_{n+1}$")
53        plt.legend(loc='best')
54        plt.savefig(self.filepath, dpi=300)

```

```

55
56
57     task = Task6()
58     task.plot_return_map()

```

1.2 課題 2

$x_0 = \frac{1}{10}$ として、どのような周期解になるか厳密にて計算せよ。

手計算で x_i の値を計算すると、

$$\begin{aligned}
 x_0 &= \frac{1}{10} \\
 x_1 &= 2 \times \frac{1}{10} = \frac{1}{5} \\
 x_2 &= 2 \times \frac{1}{5} = \frac{2}{5} \\
 x_3 &= 2 \times \frac{2}{5} = \frac{2}{5} \\
 x_4 &= 2 \times \left(1 - \frac{4}{5}\right) = \frac{2}{5} \\
 x_5 &= 2 \times \frac{2}{5} = \frac{2}{5} \\
 x_6 &= 2 \times \left(1 - \frac{4}{5}\right) = \frac{2}{5} \\
 &\vdots \\
 &\vdots \\
 &\vdots
 \end{aligned} \tag{1}$$

となるため $\frac{2}{5}$ と $\frac{4}{5}$ の 2 つの周期解 (2 周期解) を取り続けていく。

1.3 課題 3

$x_0 = \frac{1}{10}$ として、前のプログラムを実行せよ。ただし $N = 100$ とし x_{100} まで数値計算せよ。どのような結果となったか。理由も述べよ。

1.3.1 出力

Listing 2 out

```

1  0.700000 0.600000
2  0.600000 0.600000
3  0.600000 0.800000
4  0.800000 0.800000
5  0.800000 0.400000
6  0.400000 0.400000
7  0.400000 0.800000

```

8	0.800000	0.800000
9	0.800000	0.400000
10	0.400000	0.400000
11	0.400000	0.800000
12	0.800000	0.800000
13	0.800000	0.400000
14	0.400000	0.400000
15	0.400000	0.800000
16	0.800000	0.800000
17	0.800000	0.400000
18	0.400000	0.400000
19	0.400000	0.800000
20	0.800000	0.800000
21	0.800000	0.400000
22	0.400000	0.400000
23	0.400000	0.800000
24	0.800000	0.800000
25	0.800000	0.400000
26	0.400000	0.400000
27	0.400000	0.800000
28	0.800000	0.800000
29	0.800000	0.400000
30	0.400000	0.400000
31	0.400000	0.800000
32	0.800000	0.800000
33	0.800000	0.400000
34	0.400000	0.400000
35	0.400000	0.800000
36	0.800000	0.800000
37	0.800000	0.400000
38	0.400000	0.400000
39	0.400000	0.800000
40	0.800000	0.800000
41	0.800000	0.400000
42	0.400000	0.400000
43	0.400000	0.800000
44	0.800000	0.800000
45	0.800000	0.400000
46	0.400000	0.400000
47	0.400000	0.800000
48	0.800000	0.800000
49	0.800000	0.400000
50	0.400000	0.400000
51	0.400000	0.800000
52	0.800000	0.800000
53	0.800000	0.400000

54	0.400000	0.400000
55	0.400000	0.800000
56	0.800000	0.800000
57	0.800000	0.400000
58	0.400000	0.400000
59	0.400000	0.800000
60	0.800000	0.800000
61	0.800000	0.400000
62	0.400000	0.400000
63	0.400000	0.800000
64	0.800000	0.800000
65	0.800000	0.400000
66	0.400000	0.400000
67	0.400000	0.799999
68	0.799999	0.799999
69	0.799999	0.400002
70	0.400002	0.400002
71	0.400002	0.800003
72	0.800003	0.800003
73	0.800003	0.399994
74	0.399994	0.399994
75	0.399994	0.799988
76	0.799988	0.799988
77	0.799988	0.400024
78	0.400024	0.400024
79	0.400024	0.800049
80	0.800049	0.800049
81	0.800049	0.399902
82	0.399902	0.399902
83	0.399902	0.799805
84	0.799805	0.799805
85	0.799805	0.400391
86	0.400391	0.400391
87	0.400391	0.800781
88	0.800781	0.800781
89	0.800781	0.398438
90	0.398438	0.398438
91	0.398438	0.796875
92	0.796875	0.796875
93	0.796875	0.406250
94	0.406250	0.406250
95	0.406250	0.812500
96	0.812500	0.812500
97	0.812500	0.375000
98	0.375000	0.375000
99	0.375000	0.750000

100	0.750000	0.750000
101	0.750000	0.500000
102	0.500000	0.500000
103	0.500000	1.000000
104	1.000000	1.000000
105	1.000000	0.000000
106	0.000000	0.000000
107	0.000000	0.000000
108	0.000000	0.000000
109	0.000000	0.000000
110	0.000000	0.000000
111	0.000000	0.000000
112	0.000000	0.000000
113	0.000000	0.000000
114	0.000000	0.000000
115	0.000000	0.000000
116	0.000000	0.000000
117	0.000000	0.000000
118	0.000000	0.000000
119	0.000000	0.000000
120	0.000000	0.000000
121	0.000000	0.000000
122	0.000000	0.000000
123	0.000000	0.000000
124	0.000000	0.000000
125	0.000000	0.000000
126	0.000000	0.000000
127	0.000000	0.000000
128	0.000000	0.000000
129	0.000000	0.000000
130	0.000000	0.000000
131	0.000000	0.000000
132	0.000000	0.000000
133	0.000000	0.000000
134	0.000000	0.000000
135	0.000000	0.000000
136	0.000000	0.000000
137	0.000000	0.000000
138	0.000000	0.000000
139	0.000000	0.000000
140	0.000000	0.000000
141	0.000000	0.000000
142	0.000000	0.000000
143	0.000000	0.000000
144	0.000000	0.000000
145	0.000000	0.000000

146	0.000000	0.000000
147	0.000000	0.000000
148	0.000000	0.000000
149	0.000000	0.000000
150	0.000000	0.000000
151	0.000000	0.000000
152	0.000000	0.000000
153	0.000000	0.000000
154	0.000000	0.000000
155	0.000000	0.000000
156	0.000000	0.000000
157	0.000000	0.000000
158	0.000000	0.000000
159	0.000000	0.000000
160	0.000000	0.000000
161	0.000000	0.000000
162	0.000000	0.000000
163	0.000000	0.000000
164	0.000000	0.000000
165	0.000000	0.000000
166	0.000000	0.000000
167	0.000000	0.000000
168	0.000000	0.000000
169	0.000000	0.000000
170	0.000000	0.000000
171	0.000000	0.000000
172	0.000000	0.000000
173	0.000000	0.000000
174	0.000000	0.000000
175	0.000000	0.000000
176	0.000000	0.000000
177	0.000000	0.000000
178	0.000000	0.000000
179	0.000000	0.000000
180	0.000000	0.000000
181	0.000000	0.000000
182	0.000000	0.000000
183	0.000000	0.000000
184	0.000000	0.000000
185	0.000000	0.000000
186	0.000000	0.000000
187	0.000000	0.000000
188	0.000000	0.000000
189	0.000000	0.000000
190	0.000000	0.000000
191	0.000000	0.000000


```
192  0.000000 0.000000
193  0.000000 0.000000
194  0.000000 0.000000
195  0.000000 0.000000
196  0.000000 0.000000
197  0.000000 0.000000
198  0.000000 0.000000
199  0.000000 0.000000
200  0.000000 0.000000
201  0.000000 0.000000
202  0.000000 0.000000
```

1.3.2 結果と考察

実行した結果、0.4 と 0.8 でずっとループすることはなく途中から値が変化した。最終的には、 $(x_n, x_{n+1}) = (0, 0)$ を取り続けるようになった。これは、コンピュータの丸め誤差が影響し値が変化したと考えられる。

1.3.3 ソースコード

Listing 3 tent.c

```
1  #include<stdio.h>
2  #define N 100
3  double next(double x, double r) {
4      if (0 <= x && x <= 0.5) {
5          return 2 * x;
6      } else {
7          return 2 * (1 - x);
8      }
9  }
10 int main(void){
11     int n;
12     double r;
13     double xn = 0.7;
14     double xn1;
15     scanf("%lf", &r);
16     for(n = 0; n <= N; n++) {
17         xn1 = next(xn, r);
18         printf("%lf %lf\n", xn, xn1);
19         printf("%lf %lf\n", xn1, xn1);
20         xn = xn1;
21     }
22     return 0;
23 }
```

1.4 課題 4

テント写像のリアプノフ指数を計算せよ。プログラミングがなければ手計算でよい。
リアプノフ指数 λ は、

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)| \quad (2)$$

と表すことができる。

今回はテント写像のリアプノフ指数について調べていくので、関数 $f(x_i)$ は、

$$f(x_i) = \begin{cases} 2x_i & \left(0 \leq x_i \leq \frac{1}{2}\right) \\ 2(1-x_i) & \left(\frac{1}{2} < x_i \leq 1\right) \end{cases} \quad (3)$$

となる。さらに (3) を用いて計算すると $f'(x_i)$ は、

$$f'(x_i) = \begin{cases} 2 & \left(0 \leq x_i \leq \frac{1}{2}\right) \\ -2 & \left(\frac{1}{2} < x_i \leq 1\right) \end{cases} \quad (4)$$

となる。よって、 $|f'(x_i)| = 2$ である。したがって、テント写像のリアプノフ指数 λ は、

$$\begin{aligned} \lambda &= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} \log 2 \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \times n \log 2 \\ &= \lim_{n \rightarrow \infty} \log 2 \\ &= \log 2 \end{aligned} \quad (5)$$

と計算することができ $\lambda = \log 2$ となる。

1.5 課題 5

テント写像は x_n から x_{n+1} を求める写像である。 $y_n = \sin^2\left(\frac{\pi}{2}x_n\right)$ と定義して、 y_{n+1} を y_n の式として表わせ。

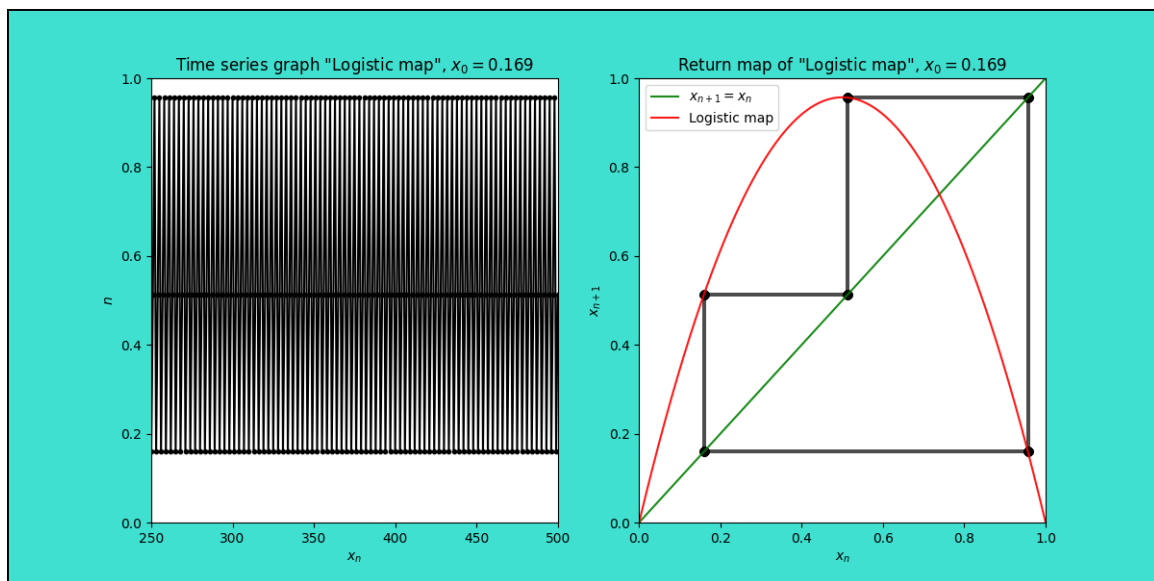
2 レポート課題 2nd

2.1 課題 1

2.1.1 問題

$r = 3.8285$ として、 x_n が $250 \leq n \leq 500$ の場合の時系列とリターンマップを描け、初期値は適当でよい。周期3を確認せよ（3周期の窓）

2.1.2 画像



2.1.3 考察

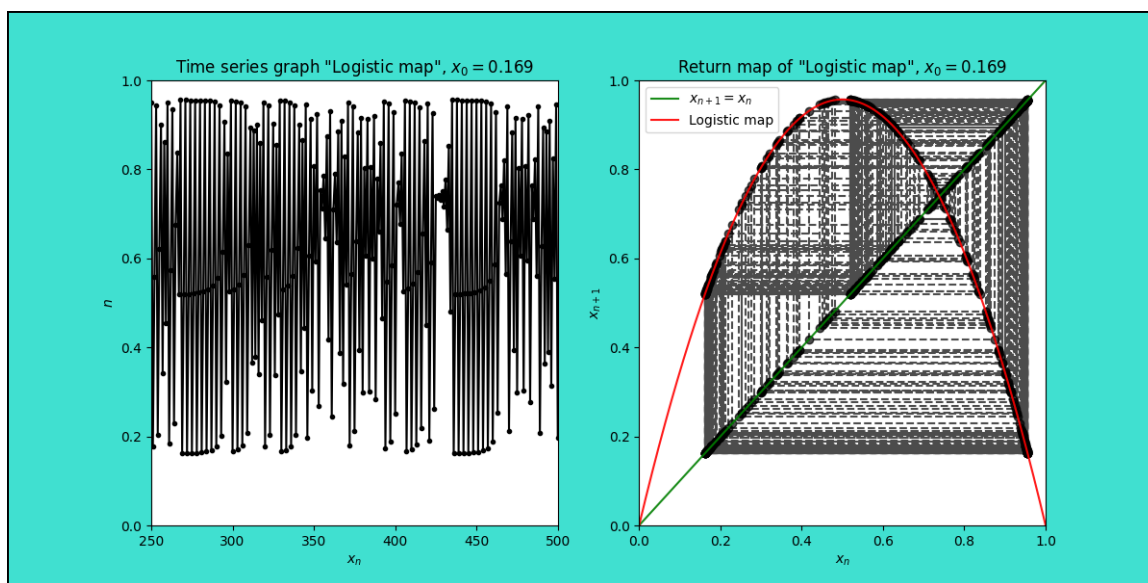
$r = 3.8285$ のときは、前回のレポート課題3に書いてあるロジスティック写像の分岐図を見ると、3周期解になっている範囲だとわかる。実際にグラフを描くと、時系列グラフ、リターンマップともに3つの解で周期性を持っていることが読み取れた。また、初期値をランダムにとって複数回プログラムを実行してもどちらのグラフにも変化は見られなかった。よって、 $r = 3.8285$ のときはカオスの条件には当てはまらず周期性（3周期）を持っていると考察した。

2.2 課題 2

2.2.1 問題

$r = 3.8284$ として、 x_n が $250 \leq n \leq 500$ の場合の時系列とリターンマップを描け、初期値は適当でよい。規則的な部分（ラミナー）と不規則の部分（バースト）を確認せよ。

2.2.2 画像



2.2.3 考察

課題 1 と比較すると r の変化は 0.0001 なのに対し、時系列グラフとリターンマップには大きな変化が見られた。また、初期値をランダムにとっているため複数回プログラムを実行させた結果、毎回違うグラフが出力された。よって、 $r = 3.8284$ のときは、カオスの条件に当てはまると考察した。また課題 1 の r の値から、 $r = 3.8284$ と $r = 3.8285$ の間に周期倍分岐が起きていることが考察できる。

2.3 課題 3

2.3.1 問題

次に説明する写像 $f^{(3)}$ について、リターンマップを描け（横軸 x_n , 縦軸 x_{n+3} ）

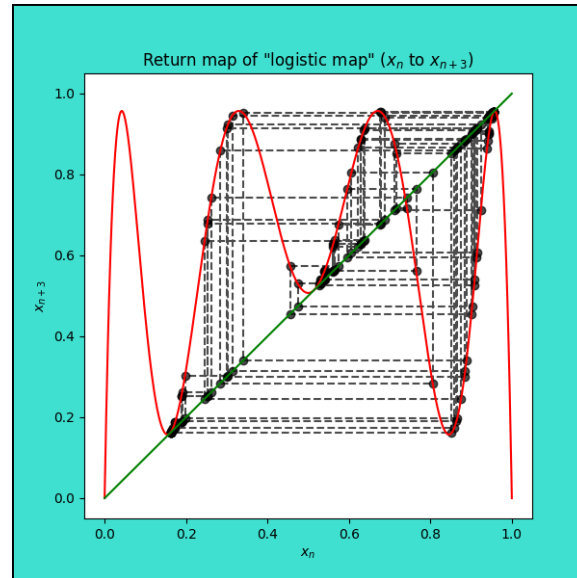
これまで $x_{n+1} = f(x_n)$ の写像について考えてきた（ここで関数 f は、上で定義したロジスティック写像）。時系列は $x_0, x_1, x_2, x_3, x_4, x_5, \dots$ を順に書いてきたが、上のパラメータでは、だいたいの部分において周期 3 の運動をしている。そこでここでは、3 つ飛ばしで時系列とリターンマップを考える。

いま $x_{n+1} = f(x_n)$ だから、同様に $x_{n+2} = f(x_{n+1}), x_{n+3} = f(x_{n+2})$ が成り立つ。これらをまとめると、 $x_{n+3} = f(x_{n+2}) = f(f(x_{n+1})) = f(f(f(x_n)))$ と書ける。

この、 x_n から x_{n+3} を求める式 $x_{n+3} = f(f(f(x_n)))$ を簡単のために $x_{n+3} = f^{(3)}(x_n)$ と書くことにする。これまでは、 x_n と x_{n+1} の間のリターンマップを書いてきたが、 x_n と x_{n+1} の間のリターンマップを

$r = 3.8284$ のときに書いてみよ。また、下右図のようなリターンマップの階段状にはさまれた部分は何を意味するか考えよ。

2.3.2 画像



2.3.3 考察

階段上に挟まれた部分はクモの巣図でとっている x_n の時の x_{n+3} の値であると考察した。その理由は範囲が $[0.9561, 0.9565]$ のため曲線の一番右の部分と $x_n = x_{n+3}$ のクモの巣図を拡大しているからである。また、グラフと拡大グラフにより収束したのちに再度発散するような動きになっている。そのため完全には交わっているわけではなく近似していると考察することができる。

2.3.4 ソースコード

課題1から課題3までのソースコードを記載している。

Listing 4 task7.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3  from random import uniform
4
5
6  class Task7():
7      def __init__(self) -> None:
8          # 初期値 x0 = 0.1
9          self.x = uniform(0, 1)
10         self.xn = np.linspace(0, 1, 1000)          # 横軸の範囲と刻み幅
11         self.fig = plt.figure(figsize=(12, 6), facecolor='turquoise',
12                                     linewidth=1, edgecolor='black')
```

```

13
14     def logistic(self, r) -> list:
15         リターンマップでのロジスティック写像の座標を持つ配列を返す""
16         calc_x = self.x
17         x_array = []
18         for i in range(1, 501):
19             calc_x = r * calc_x * (1 - calc_x)
20             if 250 <= i <= 500:
21                 x_array.append(calc_x)
22         return x_array
23
24     def plot_time_series_graph(self, r: float) -> None:
25         時系列グラフの描画（ロジスティック写像）""
26         ax1 = self.fig.add_subplot(1, 2, 1)
27         n = list(range(250, 501))
28         ax1.plot(n, self.logistic(r), marker='.', color='black')
29         ax1.set_title(
30             "Time series graph \"Logistic map\", $x_0 = $" + str(round(self.x, 3)))
31         ax1.set_xlim(250, 500)
32         ax1.set_ylim(0, 1)
33         ax1.set_xlabel("$x_n$")
34         ax1.set_ylabel("$n$")
35
36     def plot_return_map(self, r) -> None:
37         リターンマップの描画（ロジスティック写像）""
38         ax2 = self.fig.add_subplot(1, 2, 2)
39         n = self.logistic(r)
40         spider_array_x = [] # クモの巣図用の配列
41         spider_array_y = [] # クモの巣図用の配列
42         for i in range(1, len(n)):
43             spider_array_x.append(n[i - 1])
44             spider_array_x.append(n[i])
45             spider_array_y.append(n[i])
46             spider_array_y.append(n[i])
47         logistic_y = [] # テント写像のグラフを
描くための配列
48         for i in self.xn:
49             logistic_y.append(r * i * (1 - i))
50
51         ax2.plot(spider_array_x, spider_array_y, marker='o',
52                 linestyle='dashed', color='black', alpha=0.7)
53         ax2.plot(self.xn, self.xn, color='green',
54                 alpha=0.9, label="$x_{n+1} = x_n$")
55         ax2.plot(self.xn, logistic_y, color='red',
56                 alpha=0.9, label="Logistic map")

```

```

57         ax2.set_title(
58             "Return map of \"Logistic map\",  $x_0 = x$  + str(round(self.x, 3))
59         ax2.set_xlim(0, 1)
60         ax2.set_ylim(0, 1)
61         ax2.set_xlabel(" $x_n$ ")
62         ax2.set_ylabel(" $x_{n+1}$ ")
63         ax2.legend(loc='best')
64
65     def code_problem1(self) -> None:
66         r = 3.8285
67         self.fig = plt.figure(figsize=(12, 6), facecolor='turquoise',
68                                 linewidth=1, edgecolor='black')
69         self.plot_time_series_graph(r)
70         self.plot_return_map(r)
71         plt.savefig複雑系科学演習 ('/Week7/images/task7_1')
72
73     def code_problem2(self) -> None:
74         r = 3.8284
75         self.fig = plt.figure(figsize=(12, 6), facecolor='turquoise',
76                                 linewidth=1, edgecolor='black')
77         self.plot_time_series_graph(r)
78         self.plot_return_map(r)
79         plt.savefig複雑系科学演習 ('/Week7/images/tast7_2')
80
81     def preprocessing(self, x: float) -> float:
82         r = 3.8284
83         return(r * x * (1 - x))
84
85     def code_problem3(self):
86         r = 3.8284
87         self.fig = plt.figure(figsize=(6, 6), facecolor='turquoise',
88                                 linewidth=1, edgecolor='black')
89
90         fff_array_x = []
91         fff_array_y = []
92         for i in self.xn:
93             fff_array_x.append(i)
94             fff_array_y.append(self.preprocessing(
95                 self.preprocessing(self.preprocessing(i))))
96
97         n = self.logistic(r)
98         spider_array_x = [] # クモの巣図用の配列
99         spider_array_y = [] # クモの巣図用の配列
100        (x)
101        (y)
102        for i in range(3, len(n), 3):

```

```

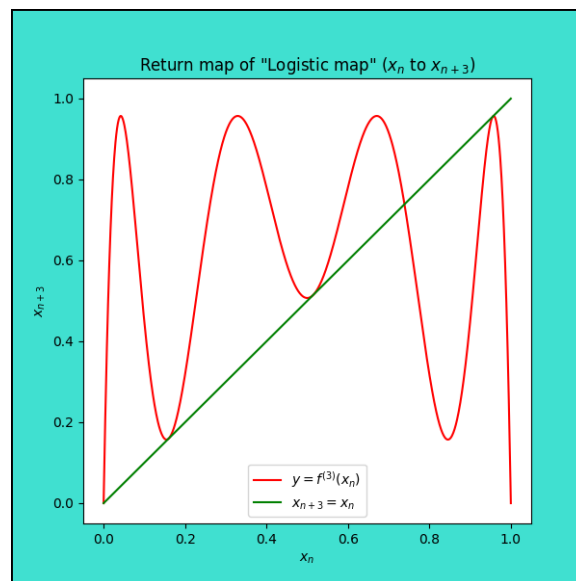
101         spider_array_x.append(n[i - 3])
102         spider_array_x.append(n[i])
103         spider_array_y.append(n[i])
104         spider_array_y.append(n[i])
105     plt.plot(spider_array_x, spider_array_y, marker='o',
106             linestyle='dashed', color='black', alpha=0.7)
107     plt.plot(fff_array_x, fff_array_y, color='red', label='$Logistic map$')
108     plt.plot(self.xn, self.xn, color='green', label='$x_{n+3} = x_n$')
109     plt.title('Return map of "logistic map" ($x_n$ to $x_{n+3}$)')
110     plt.xlabel('$x_n$')
111     plt.ylabel('$x_{n+3}$')
112     plt.savefig(複雑系科学演習 ('/Week7/images/tast7_3'))
113
114
115     task = Task7()
116     task.code_problem1()
117     task.code_problem2()
118     task.code_problem3()

```

2.4 課題 4

$x_{n+1} = f(x_n)$ である時、 $x_{n+2} = f(x_{n+1})$, $x_{n+3} = f(x_{n+2})$ である。従って、 $x_{n+3} = f(f(x_{n+1})) = f(f(f(x_n)))$ と書ける。 $y = f(f(f(x_n)))$ のグラフを描き、この曲線が $y = x$ と区間 $[0, 1]$ において互いに異なる3つの点で接する時の r の値は、 $r = 1 + \sqrt{8}$ であることをグラフを描いて確認せよ。

2.4.1 画像



2.4.2 考察

画像から、 $r = 1 + \sqrt{8}$ のとき $y = f^{(3)}(x_n)$ と $x_{n+1} = x_n$ は3点で接していると確認することができる。また、課題3の挟まれた部分のグラフを見ると、 $r = 3.8284$ のときには $y = f^{(3)}(x_n)$ と $x_{n+1} = x_n$ は接することなく収束したのち発散していることが読み取れる。さらに、 $1 + \sqrt{8} = 3.82842712474\dots$ である。以上のことから、 $r = 1 + \sqrt{8}$ を境に3周期の特徴を持つと考察することができる。

2.4.3 ソースコード

Listing 5 task7-4.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3  from random import uniform
4
5
6  class Week7Task4():
7      def __init__(self) -> None:
8          self.x = uniform(0, 1)
9          self.r = 1 + 8**(1 / 2)
10         self.xn = np.linspace(0, 1, 1000)
11         self.fig = plt.figure(figsize=(6, 6), facecolor='turquoise',
12                                     linewidth=1, edgecolor='black')
13
14     def preprocessing(self, x: float) -> float:
15         前処理""""""
16         return(self.r * x * (1 - x))
17
18     def code_problem4(self):
19         """"r = 1 +   のときのグラフ 8""""
20
21         fff_array_x = []
22         fff_array_y = []
23         for i in self.xn:
24             fff_array_x.append(i)
25             fff_array_y.append(self.preprocessing(
26                 self.preprocessing(self.preprocessing(i))))
27         plt.plot(fff_array_x, fff_array_y, color='red',
28                 label='$y = f^{\{3\}}(x_n)$')
29         plt.plot(self.xn, self.xn, color='green', label='$x_{n+3} = x_n$')
30         plt.title('Return map of "Logistic map" ($x_n$ to $x_{n+3}$)')
31         plt.xlabel('$x_n$')
32         plt.ylabel('$x_{n+3}$')
33         plt.legend(loc='best')
34         plt.savefig複雑系科学演習 ('/Week7/images/tast7_4')
35
```

```
36
37 task = Week7Task4()
38 task.code_problem4()
```

2.5 課題 5

2.5.1 問題

4の結果から r の値が $r = 1 + \sqrt{8}$ より大きい場合と小さい場合において、 $x_{n+1} = f(x_n)$ の安定固定点が幾つあるか理由を述べて答えよ。

2.5.2 考察

前回のレポート課題3であるロジスティック写像の分岐図を見て考察した。その結果、 $r > 1 + \sqrt{8}$ の場合は3周期をもつときとカオスの状態のときの2つの場合があることが読み取れる。具体的には、 $r = 1 + \sqrt{8}$ から進むにつれ周期倍分岐を起こし、その後カオスの状態へと変化していく。よって、安定固定点は3, 6, ... と変化していくと考察した。また、 $r < 1 + \sqrt{8}$ の場合は、カオスの状態になるため安定固定点はないと考察した。

2.6 課題 6

2.6.1 問題

4, 5から $r = 1 + \sqrt{8}$ の場合に生じたラミナーとバーストは $x_{n+3} = f(f(f(x_n)))$ の値と x_n の値がどのように変化する時に現れる現象であるか説明せよ。

2.6.2 考察

複数回実行するとラミナーとバーストの周期性は捉えることができなかった。

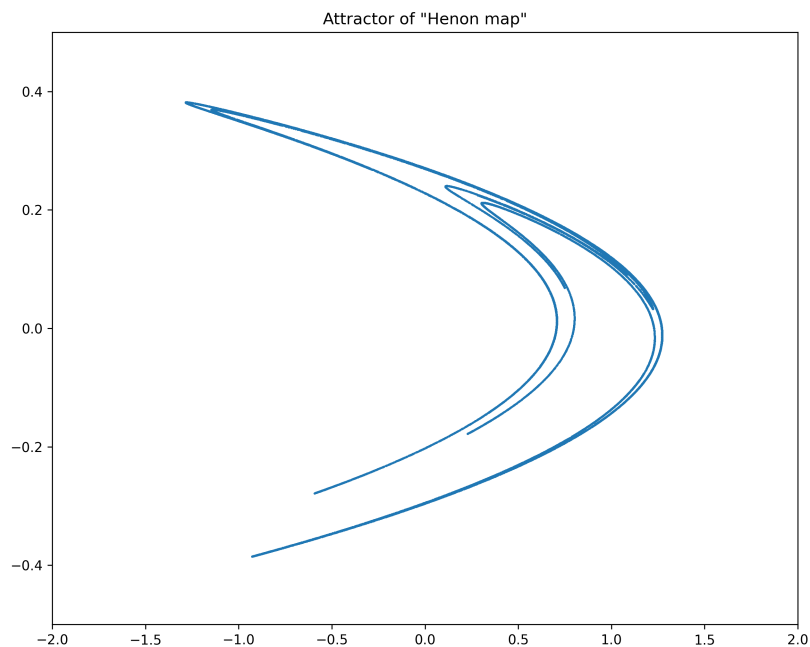
3 レポート課題 3rd

3.1 課題 1

3.1.1 問題

$a = 1.4, b = 0.3$ として、Henon 写像のアトラクタを描け。アトラクタへ至るまでの過渡状態は書かなくてよい。

3.1.2 画像



3.1.3 考察

漸化式を代入し Python で実行させた結果、画像のような結果になった。この画像を拡大していくと同様の解軌道を描いていくことが発見できた。そのため、エノン写像にはフラクタル構造があるのではないかと考察した。

3.1.4 ソースコード

Listing 6 task8-1.py

```
1 from matplotlib import pyplot as plt
2 from random import uniform
3
```

```

4
5 def henon(x: float, y: float):
6     return y + 1 - a * x ** 2, b * x
7
8
9 x = uniform(-1, 1)
10 y = uniform(-1, 1)
11 a = 1.4
12 b = 0.3
13 cnt = 30000
14 x_array = []
15 y_array = []
16 for i in range(cnt):
17     x, y = henon(x, y)
18     if i < 249:
19         continue
20     x_array.append(x)
21     y_array.append(y)
22 plt.figure(figsize=(10, 8))
23 plt.plot(x_array, y_array, linestyle='None', marker='.', markersize=1)
24 plt.title('Attractor of "Henon map"')
25 plt.xlim(-2, 2)
26 plt.ylim(-0.5, 0.5)
27 # plt.show()
28 plt.savefig複雜系科学演習 ('/Week8/images/task8_1', dpi=300)

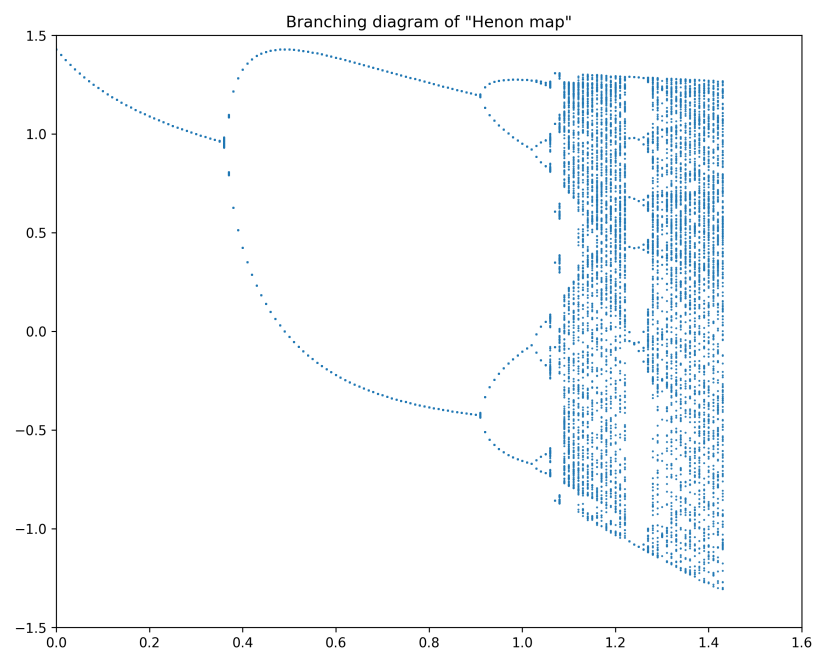
```

3.2 課題 2

3.2.1 問題

$b = 0.3$ に固定して、横軸 a 縦軸 x_n の分岐図を描け。 $a \in [0, 1.5]$ を 0.01 刻みで変化させたときの、 $x_n (n = 250 \text{ から } 500)$ の点を打つこと。時間とファイルサイズが許すなら、より刻み幅を小さくしてもよい。

3.2.2 画像



3.2.3 考察

考察については次の課題 3 に記述している。

3.2.4 ソースコード

Listing 7 task8-2.py

```
1 from matplotlib import pyplot as plt
2 import numpy as np
3 from random import uniform
4
5
6 def henon(a: float, x: float, y: float):
7     エノン写像の"""(x, y)座標を返す関数"""
```

```

8         return y + 1 - a * x ** 2, b * x
9
10
11     a = np.arange(0, 1.5, 0.01, dtype=object)    # 課題2の a の範囲
12     b = 0.3                                     # b = 0.3 に固定
13     x0 = uniform(-1, 1)                         # ランダムの初期値 (x0[-1, の範囲) 1]
14     y0 = uniform(-1, 1)                         # ランダムの初期値 (y0[-1, の範囲) 1]
15     x_array = []                                # x の値を保存するための配列
16     a_array = []                                # a の値を保存するための配列
17
18     for i in a:                                  # 各 a のときの振る舞いを計算する
19         x = x0
20         y = y0
21         for j in range(500):
22             x, y = henon(i, x, y)
23             if x < -1.5 or 1.5 < x:                # プロットの範囲外なら終了 (オーバーフ
               ローを避けるため)
24                 break
25             if j < 249:                            # 250 < xn < 500 の範囲外なら配列に入
               れない
26                 continue
27             x_array.append(x)
28             a_array.append(i)
29
30     plt.figure(figsize=(10, 8))
31     plt.plot(a_array, x_array, linestyle='None', marker='.', markersize=1)
32     plt.title('Branching diagram of "Henon map"')
33     plt.xlim(0, 1.6)
34     plt.ylim(-1.5, 1.5)
35     # plt.show()
36     plt.savefig複雑系科学演習 ('/Week8/images/task8_2', dpi=300)

```

3.3 課題3

3.3.1 問題

問2の分岐図よりわかることは何か？（例えば分岐の種類など）

3.3.2 考察

$r = 0.4$ 付近を境に周期倍分岐を起こしていることが読み取れた。

4 レポート課題 4

4.1 課題 1

4.1.1 問題

10×10 分割したマス目内にアトラクタの点が入っているかどうか調べる。点が入っているマス目の数を数えよ。

4.1.2 解答と考察

実際にレポート課題 4 の下にある参考ソースコードを埋め実行してみた。すると [10 34] が出力結果として出てきた。そのためマス目の数は 34 個である。ソースコードは $N = 1, 2, \dots, 500$ までの結果を一括で表示したため最後に掲載する。

4.2 課題 2

4.2.1 問題

$50 \times 50, 100 \times 100, 500 \times 500$ 分割したとき、同様の手順でアトラクタの点が入っているマス目を (プログラムを用いて) 数えよ。

4.2.2 解答と考察

同様に $N = 50, 100, 500$ で実行した。その結果 [50 254], [100 604], [500 4290] がそれぞれ出力された。よってマス目の数は、

- $N = 50$ のとき 254 個
- $N = 100$ のとき 604 個
- $N = 500$ のとき 4290 個

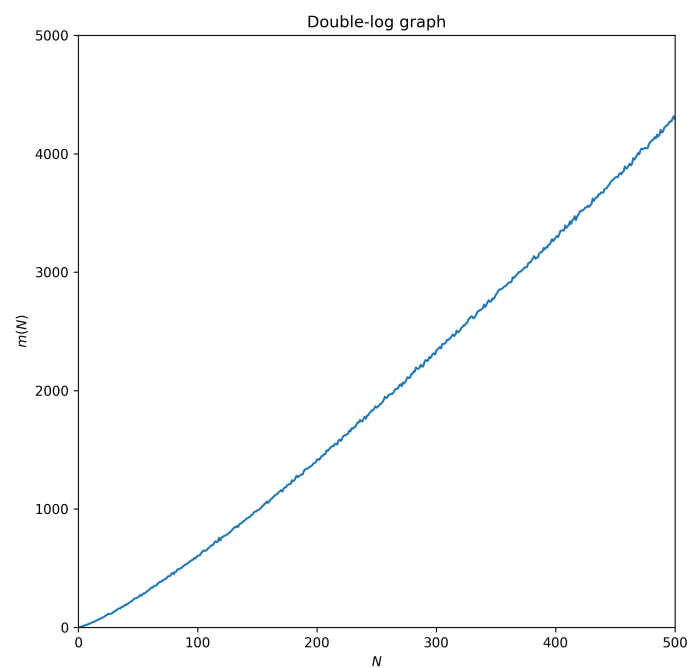
となった。また、ソースコードは $N = 1, 2, \dots, 500$ までの結果を一括で表示したため最後に掲載する。

4.3 課題 3

4.3.1 問題

$N \times N$ 分割したとき上で得た結果が $m(N)$ だったとする。横軸 N 、縦軸 $m(N)$ として両対数グラフを描け。何がわかるか。

4.3.2 画像



4.3.3 考察

Python では、それぞれの個数を計算するときの計算量に問題があったため、今回は個数計算を C 言語グラフの描画を Python で行った。考察としては、最初のみ直線とは違うような増加の仕方をするが、 $N = 50$ あたり以降からは直線的な増加をしていることが挙げられる。

4.3.4 ソースコード

Listing 8 ctest9.c

```
1  #include <stdio.h>
2  #define N 1000000
3  #define nSizeMax 1000
4
5  #define xmin -1.5
```



```

6  #define xmax 1.5
7  #define ymin -0.5
8  #define ymax 0.5
9
10 int hist[nSizeMax][nSizeMax];
11
12 void next(double* x, double* y, double a, double b) {
13     double xx = (*y) + 1 - a * (*x) * (*x);
14     double yy = b * (*x);
15     *x = xx;
16     *y = yy;
17 }
18
19 // initialization
20 void init(int nS) {
21     int i, j;
22     for ( i = 0; i < nS; i++ ) {
23         for ( j = 0; j < nS; j++ ) {
24             hist[i][j] = 0;
25         }
26     }
27 }
28
29 // print out
30 void print(int nS) {
31     int i, j, count = 0;
32     for ( i = 0; i < nS; i++ ) {
33         for ( j = 0; j < nS; j++ ) {
34             if (hist[i][j]) {
35                 count++;
36             }
37         }
38     }
39     printf("%d %d\n", nS, count);
40 }
41
42
43 int main(){
44     double a = 1.4, b = 0.3;
45     double x = 0.5, y = 0.5;
46
47     for (int nSize = 0; nSize <= 500; nSize++) {
48         init(nSize);
49         int px, py;
50         for (int n = 0; n <= N; n++) {
51             next(&x, &y, a, b);

```

```

52         if (n > 10000) {
53             px = (int)((x - xmin) / (xmax - xmin) * nSize);
54             py = (int)((y - ymin) / (ymax - ymin) * nSize);
55             hist[px][py] = 1;
56         }
57     }
58     print(nSize);
59 }
60 return 0;
61 }

```

Listing 9 task9.py

```

1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  filepath = '複雜系科学演習/Week9/ctest9.txt'
5  x_array = []
6  y_array = []
7  with open(filepath, encoding='UTF-8') as f:
8      while line := f.readline():
9          line = line.rstrip()
10         x, y = line.split(' ')
11         x_array.append(int(x))
12         y_array.append(int(y))
13
14  plt.figure(figsize=(8, 8))
15  plt.xlim(0, 500)
16  plt.ylim(0, 5000)
17  plt.plot(x_array, y_array)
18  plt.title('Double-log graph')
19  plt.xlabel('$N$')
20  plt.ylabel('$m(N)$')
21  # plt.show()
22  plt.savefig('複雜系科学演習/Week9/images/task9', dpi=300)

```
