計算機実習 問題 6.12 - エノン写像

早稲田大学先進理工学部物理学科 B4 藤本將太郎

2014/05/14

1 シミュレーションの目的

これまでは、さまざまな点で教育的であるロジスティック写像について議論してきた。今回は 2 次元の写像

$$x_{n+1} = y_n + 1 - ax_n^2 \tag{1}$$

$$y_{n+1} = bx_n \tag{2}$$

で与えられる点 $(x_n,\, y_n)$ の例を考える。写像 (1),(2) はエノン (Hénon) により最初に考えられたもので、小惑星や衛星の振る舞いとの関連がその動機であった。

2 作成したプログラム

本シミュレーションで作成・使用したプログラムを以下に示す。

2.1 パラメータの設定ダイアログ (SetParameter.py)

以前にも何度か使用しているプログラムであるため、説明は省略する。

```
1   #!usr/bin/env python
2   # -*- coding: utf-8 -*-
3   #
4   # written by Shotaro Fujimoto, Mqy 2014.
5   #
6
7   from Tkinter import *
8
9   class SetParameter():
10
11      def show_setting_window(self, parameters, commands):
12          """ Show a parameter setting window.
13
14          parameters: A list of dictionaries {'parameter name':default_value}
```

```
15          commands: A dictionary {'name of button': command}
16          """
17          self.root = Tk()
18          self.root.title('Control Widget')
19
20          frame1 = Frame(self.root, padx=5, pady=5)
21          frame1.pack(side='top')
22          self.entry = []
23          for i, parameter in enumerate(parameters):
24              label = Label(frame1, text=parameter.items()[0][0] + ' = ')
25              label.grid(row=i, column=0, sticky=E)
26              self.entry.append(Entry(frame1, width=10))
27              self.entry[i].grid(row=i, column=1)
28              self.entry[i].delete(0, END)
29              self.entry[i].insert(0, parameter.items()[0][1])
30          self.entry[0].focus_set()
31
32          frame2 = Frame(self.root, padx=5, pady=5)
33          frame2.pack(side='bottom')
34          for name, command in commands.items():
35              button = Button(frame2, text=name, command=command)
36              button.pack(side='left', fill='x')
37
38          self.root.mainloop()
39
40      def quit(self):
41          self.root.destroy()
42
```

2.2 エノン写像の描画を行うプログラム (Henon.py)

このプログラムでは、実際の写像の値の計算の反復とその描画の指示を行っており、シミュレーションの中核となる部分である。大きく分けて 3 つの部分からなっており、SetParameter から値を代入する部分 (assignment)、式 (1)(2) に基づいて時間発展を計算する部分 (calc)、そしてその値をプロットする部分 (plot_x_and_y) である。それぞれの部分は互いに独立していて、本体の方で関数を呼び出してやる必要がある。これはテストの際の効率化と、関数の再利用性を考慮に入れたためである。

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   #
```

```
4  # written by Shotaro Fujimoto, May 2014.
5
6  import matplotlib.pylab as plt
7  import array
8
9  class Henon(object):
10
11      def __init__(self):
12          global ntransient, nplot
13          ntransient = 1000
14          nplot = 10000
15
16      def assignment(self, module):
17          self.x0 = float(module.entry[0].get())
18          self.y0 = float(module.entry[1].get())
19          self.a  = float(module.entry[2].get())
20          self.b  = float(module.entry[3].get())
21
22      def func(self, x_i, y_i):
23          return y_i + 1.0 - self.a*x_i**2, self.b*x_i
24
25      def calc(self):
26          global x, y
27          x = array.array('d')
28          y = array.array('d')
29          x.append(self.x0)
30          y.append(self.y0)
31          for n in range(ntransient+nplot):
32              x.append(self.func(x[n], y[n])[0])
33              y.append(self.func(x[n], y[n])[1])
34          return (x[ntransient:ntransient+nplot],y[ntransient:ntransient+nplot])
35
36      def plot_x_and_y(self, x, y, color='blue', showing=True):
37          self.margin = 0.1
38          plt.scatter(x, y, s=1.0, marker='o', color=color,
39                              label=r'$x_{0}=$' + str(self.x0) + ' : '
40                              + r'$y_{0}=$' + str(self.y0) + ' : '
41                              + r'$a=$' + str(self.a) + ' : '
42                              + r'$b=$' + str(self.b)
43                      )
```

```
44          plt.gca().set_xlim(min(x) - self.margin, max(x) + self.margin)
45          plt.gca().set_ylim(min(y) - self.margin, max(y) + self.margin)
46          plt.xlabel(r'$x$', fontsize=16)
47          plt.ylabel(r'$y$', fontsize=16)
48          plt.title(r"$\mathrm{H\'{e}non\ map}$")
49          plt.legend(loc="best")
50          if showing: plt.show()
51          else: pass
52
```

2.3 実習課題 a で用いるプログラム (6-12_henon-a.py)

課題 a を実行するため、単純に与えたパラメータからエノン写像を描画するプログラムとなっている。

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   #
4   # written by Shotaro Fujimoto, May 2014.
5
6   import SetParameter as sp
7   import Henon
8
9   def clicked():
10      henon.assignment(module=window)
11      window.quit()
12      henon.plot_x_and_y(henon.calc()[0], henon.calc()[1], color='black')
13
14  if __name__ == '__main__':
15      parameters = [{'x0':0.0}, {'y0':0.0}, {'a':1.4}, {'b':0.3}]
16      window = sp.SetParameter()
17      henon = Henon.Henon()
18      window.show_setting_window(parameters, {'OK':clicked})
19
```

2.4 実習課題 c で用いるプログラム (6-12_henon-c.py)

課題 c を実行するため、設定したパラメータと、そこからプログラム内で与えた Δx_0、Δy_0 だけ初期条件のずれた 2 つのエノン写像を描画するプログラムである。Henon の関数 plot_x_and_y に showing=False を指定することで、プロットしたグラフが表示されないようにでき、2 つ目の曲線を重ねて描くことができる。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# written by Shotaro Fujimoto, May 2014.

import SetParameter as sp
import Henon

delta_x0 = 0.0000001
delta_y0 = -0.00001

def clicked():
    henon.assignment(module=window)
    window.quit()

    # --- 1つ目のエノン写像の描画 ---
    henon.plot_x_and_y(henon.calc()[0], henon.calc()[1],
                       color='red', showing=False
                       )

    # --- 近傍点の設定 ---
    henon.x0 = henon.x0 + delta_x0
    henon.y0 = henon.y0 + delta_y0

    # --- 2つ目のエノン写像の描画 ---
    henon.plot_x_and_y(henon.calc()[0], henon.calc()[1],
                       color='blue', showing=True
                       )

if __name__ == '__main__':

    parameters = [{'x0':0.0}, {'y0':0.0}, {'a':1.4}, {'b':0.3}]
    window = sp.SetParameter()
    henon = Henon.Henon()
    window.show_setting_window(parameters, {'OK':clicked}) # パラメータの設定

```

3 実習課題

a. $a = 1.4$ および $b = 0.3$ として、式 (1),(2) の計算を反復して行え。$x_0 = 0$, $y_0 = 0$ から始め、10^4 回の反復の結果をプロットせよ。始めの過渡的な期間の結果はプロットしない。第 2 の初期条件 $x_0 = 0.63135448$、$y_0 = 0.18940634$ から始めて同様の図を描いて、それら 2 つの図の形を比較せよ。2 つの曲線の形は初期条件によらないか。

$a = 1.4$、$b = 0.3$ として式 (1)(2) の計算を行い、初期値 $x_0 = 0$、$y_0 = 0$ から始めて 10^4 回の反復の結果をプロットする。2.3 で示したプログラムによりこれを実行し、得られたグラフを図 1 に示す。このグラフから、x-y で描かれる曲線は円弧状の拡がりの中で折りたたまれたような形となっていることが分かる。次に、初期条件を $x_0 = 0.63135448$、$y_0 = 0.18940634$ としたときのグラフを図 2 に示す。このグラフは先の図と同じ形状をしており、曲線の形は初期条件によらないことが分かる。

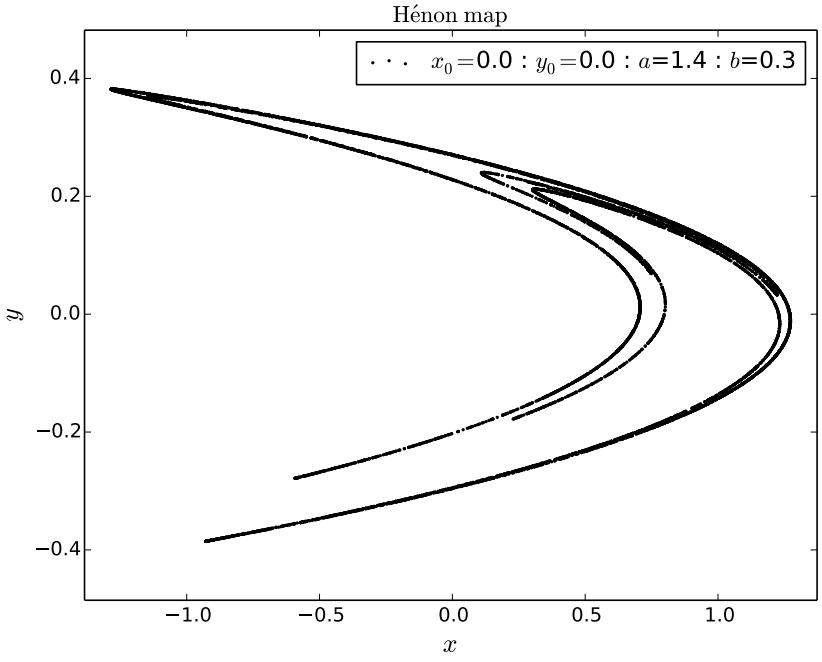

図 1　初期値 $x_0 = 0$、$y_0 = 0$ でのエノン写像

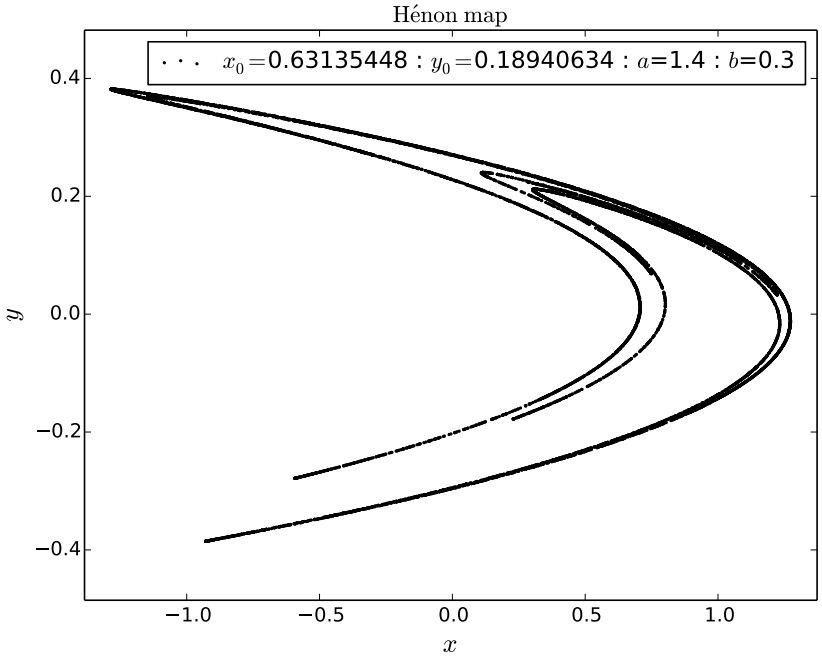

図 2　初期値 $x_0 = 0.63135448$、$y_0 = 0.18940634$ でのエノン写像

c. 系がカオス的であるかどうか、つまり初期条件に敏感であるかどうかを決定せよ。2 つの互いに非常に近い 2 点から始めて、それらの軌跡を観察せよ。2 つの軌跡に異なる色を付けよ。

互いに近い 2 点を初期値として反復して計算を行い、それらの軌跡を同時にプロットしたものを図 3 に示す。図から、非常に近い 2 つの初期条件から開始した曲線は、a でも見たようにその曲線の形は初期条件によらない。しかしながら、その曲線の中で離散的にプロットされる場所は異なっていることが分かる。

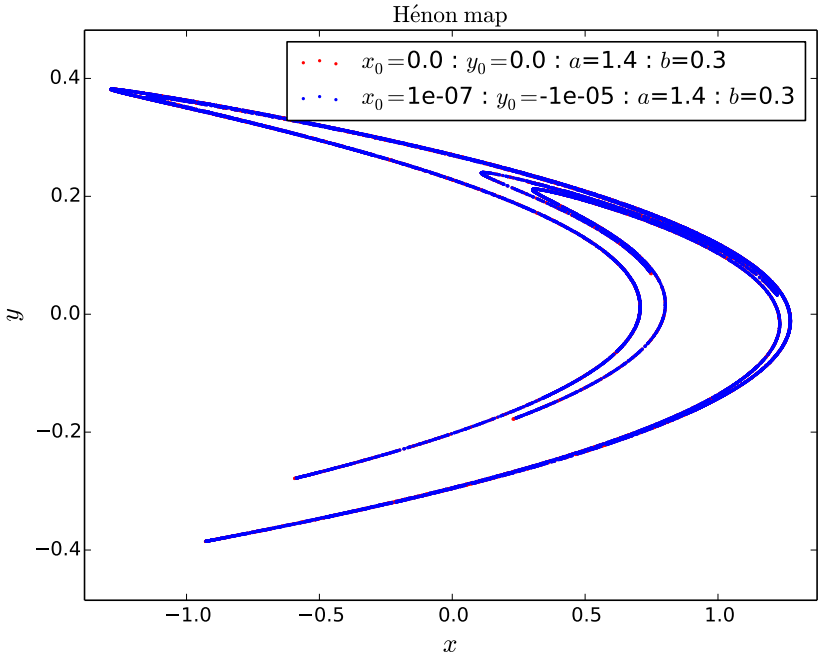

図 3　初期値 $x_0 = 0$、$y_0 = 0$ とその近傍点について、パラメータ $a = 1.4$、$b = 0.3$ としたときのエノン写像

4　まとめ

2 次元の写像であるエノン写像について、その曲線の形状は初期値によらないこと、すなわちエノンアトラクタの存在について知ることができた。

5　参考文献

- ハーベイ・ゴールド, ジャン・トボチニク, 石川正勝・宮島佐介訳『計算物理学入門』, ピアソン・エデュケーション, 2000.