計算機実習課題 6-14 外力を受けた減衰振り子

早稲田大学先進理工学部物理学科 B4 藤本將太郎

2014/05/18

1　シミュレーションの目的

古典力学で記述される非線形力学系のよく知られた例である単振り子について、一次の減衰項があって、振り子の支点が垂直方向に動かされる場合を考えよう。この系に対するニュートンの第 2 法則は

$$\frac{d^2\theta}{dt^2} = -\gamma\frac{d\theta}{dt} - [\omega_0^2 + 2A\cos\omega t]\sin\theta \tag{1}$$

と書き表される。ここで θ は振り子が鉛直軸となす角度、γ は減衰定数、$\omega_0^2 = g/L$ は振り子の自然振動数、ω と A はそれぞれ外力の振動数と振幅である。支点を垂直方向に加速することは時間に依存する重力場を考えることと等価である。この外力を受けた減衰のある単振り子はどのように振る舞うと予想されるか。減衰項が存在するために、外力がないときには振り子は静止してしまう。つまり、($x = 0$、$y = 0$) は安定なアトラクタである。A を 0 から大きくしていくとき、小さな A に対してはこのアトラクタは安定である。しかし、A がある値 A_c を超えると、このアトラクタは不安定となる。振幅 A を大きくしていくとき、この外力を受けた非線形な振り子はどのように振る舞うだろうか。運動の (安定と不安定) 固定点に主に興味があるので、外力の周期 T ごとに位相空間に点をプロットして、運動の解析を行うのがよい。位相空間におけるそのような図はポアンカレ (Poincaré)・プロットと呼ばれている。したがって、$n = 1, 2, 3\cdots$ として、$t = n\mathrm{T}$ で $d\theta/dt$ を θ に対してプロットする。系の周期が T の時にはポアンカレ・プロットは単一の点からなり、周期が mT のときには m 個の点からなる。

2　作成したプログラム

2.1　パラメータの設定ダイアログ (SetParameter.py)

以前まで使用していたものを改良してある。show_setting_window に渡す引数は、パラメータの名前とデフォルト値をセットにして辞書にしたもののリストと、ボタンに表示するテキストをキーとし、コマンドをその値とした辞書である。これによって更に拡張した使い方ができるようになった。

```
1  #! /usr/bin/env python
2  # -*- coding: utf-8 -*-
3  #
4  # written by Shotaro Fujimoto, Mqy 2014.
5
```

```
6  from Tkinter import *
7
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()
```

2.2 4 次のルンゲクッタ法で 1 ステップの計算を行うプログラム (runge_kutta.py)

4 次のルンゲクッタ法を用いて、t の時の値 y をもとに t+h のときの値を返すプログラムとなっている。引数として渡す関数と可変長の変数 y を変えれば、一般に用いることのできるモジュールとなっている。

```
1  #! /usr/bin/env python
2  # -*- coding: utf-8 -*-
3  #
4  # written by Shotaro Fujimoto, May 2014.
5
6  import numpy as np
7
8
9  class RK4(object):
10
11     def __init__(self, function):
12         """ Initialize function."""
13         self.function = function
14
15     def solve(self, y, t, h):
16         """ Solve the system ODEs.
17
18         --- arguments ---
19         y: A list of initial values
20         t: Time (float)
21         h: Stepsize (float)
22         """
23         f = self.function
24         y = np.array(y)
25         k1 = h * f(t, y)
26         k2 = h * f(t + 0.5*h, y + 0.5*h*k1)
27         k3 = h * f(t + 0.5*h, y + 0.5*h*k2)
28         k4 = h * f(t + 1.0*h, y + 0.5*h*k3)
29
30         y = y + (k1 + 2*k2 + 2*k3 + k4)/6.0
31         return y
```

2.3 課題 a の実行プログラム (6-14_poincare_a.py)

課題 a で用いたプログラムで、外力の周期 T ごとに $d\theta/dt$ を θ に対してプロットする。アニメーションで表示させるために plt.ion() で pyplot のインタラクティブモードをオンにしている。しかし、そのままにすると最後のプロットが終了した時点でウィンドウが閉じてしまうので、最後のプロットでインタラクティブモードをオフにするようにした。

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
#
# written by Shotaro Fujimoto, May 2014.
"""外力を受けた減衰振り子のシミュレーション
外力の周期ごとにθ-d θ/dt のプロットを表示する。
初めの ntransient 回の計算結果は表示しない。
"""

from math import *
import numpy as np
import matplotlib.pylab as plt
import SetParameter as sp
import runge_kutta as RK

# --- parameters ---
omega0 = 1
omega = 2
T = pi                  # <-- (2*pi)/omega (time period)
nstep = 1000
dt = T / nstep            # stepsize in a time period
ntransient = 100
nplot = 100
nmax = ntransient + nplot


def assignment():
    """ Assign the values to variables."""
    global theta, ang_vel, gamma, A
    theta = float(window.entry[0].get())
    ang_vel = float(window.entry[1].get())
    gamma = float(window.entry[2].get())
```

```
33      A = float(window.entry[3].get())
34      window.quit()
35      plot(y=np.array([theta, ang_vel]), function=force)
36
37
38  def force(t, y):
39      theta = y[0]
40      ang_vel = y[1]
41      return np.array([ang_vel, -gamma * ang_vel -
42                       (omega0 ** 2 + 2 * A * cos(omega * t)) * sin(theta)
43      ])
44
45
46  def plot(y, function):
47      """ Show an animation of Poincare plot.
48
49      --- arguments ---
50      y: A list of initial values
51      function: function which is argument of Runge-Kutta solver
52      """
53      h = dt
54      fig = plt.figure()
55      ax = fig.add_subplot(111)
56      ax.grid()
57      time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)
58      plt.ion()
59
60      for i in range(nmax + 1):
61          for j in range(nstep):
62              rk4 = RK.RK4(function)
63              y = rk4.solve(y, j * h, h)
64              # -pi <= theta <= pi
65              while y[0] > pi:
66                  y[0] = y[0] - 2 * pi
67              while y[0] < -pi:
68                  y[0] = y[0] + 2 * pi
69
70          if ntransient <= i < nmax:          # <-- draw the poincare plots
71              plt.scatter(y[0], y[1], s=2.0, marker='o', color='blue')
72              time_text.set_text('n = %d' % i)
```

```
73              plt.draw()
74
75          if i == nmax:                       # <-- to stop the interactive mode
76              plt.ioff()
77              plt.scatter(y[0], y[1], s=2.0, marker='o', color='blue')
78              time_text.set_text('n = %d' % i)
79              plt.show()
80
81  if __name__ == '__main__':
82
83      default_params = [
84          {'theta': 0.3}, {'ang_vel': 0.0}, {'gamma': 0.2}, {'A': 0.85}]
85
86      window = sp.SetParameter()
87      window.show_setting_window(default_params, {'Run!': assignment})
```

2.4 課題 b の実行プログラム (6-14_poincare_b.py)

課題 b で用いたプログラムで、θ と $d\theta/dt$ を t の関数としてプロットする。同時に表示させるために subplot を利用している。また、a とは異なり、すべての演算を終えてから一度に表示させるようにしてある。

```
1   #! /usr/bin/env python
2   # -*- coding: utf-8 -*-
3   #
4   # written by Shotaro Fujimoto, May 2014.
5   """外力を受けた減衰振り子のシミュレーション
6   θ と d θ/dt を t の関数としてプロットする。
7   """
8
9   from math import *
10  import numpy as np
11  import matplotlib.pylab as plt
12  import SetParameter as sp
13  import runge_kutta as RK
14
15  # --- parameters ---
16  omega0 = 1
17  omega = 2
18  T = pi                  # <-- (2*pi)/omega (time period)
19  tmax = 10. * T
```

```
20  dt = 0.1                # stepsize in a time period
21
22
23  def assignment():
24      """ Assign the values to variables."""
25      global theta, ang_vel, gamma, A
26      theta = float(window.entry[0].get())
27      ang_vel = float(window.entry[1].get())
28      gamma = float(window.entry[2].get())
29      A = float(window.entry[3].get())
30      window.quit()
31      plot(function=force)
32
33
34  def force(t, y):
35      theta = y[0]
36      ang_vel = y[1]
37      return np.array([ang_vel,
38                       -gamma * ang_vel -
39                      (omega0 ** 2 + 2 * A * cos(omega * t)) * sin(theta)
40      ])
41
42
43  def plot(function):
44      """ Show an animation of Poincare plot.
45
46      --- arguments ---
47      y: A list of initial values
48      function: function which is argument of Runge-Kutta solver
49      """
50      y = np.array([theta, ang_vel])
51      h = dt
52      rk4 = RK.RK4(function)
53
54      def calc(y, t, h):
55          y = rk4.solve(y, t, h)
56          # -pi <= theta <= pi
57          while y[0] > pi:
58              y[0] = y[0] - 2 * pi
59          while y[0] < -pi:
```

```
60              y[0] = y[0] + 2 * pi
61          return y
62
63      t = np.arange(0, tmax, dt)
64      res_theta = np.zeros(len(t), np.float32)
65      res_ang_vel = np.zeros(len(t), np.float32)
66
67      for i, _t in enumerate(t):
68          res_theta[i] = y[0]
69          res_ang_vel[i] = y[1]
70          y = calc(y, _t, h)
71
72      fig = plt.figure()
73      fig.subplots_adjust(hspace=0.4)
74
75      ax1 = fig.add_subplot(211)
76      ax1.plot(t, res_theta, label=r'$\theta$')
77      ax1.set_xlim(0, tmax)
78      plt.xlabel(r'$t$', fontsize=16)
79      plt.ylabel(r'$\theta$', fontsize=16)
80      plt.title(r'$t\ -\ \theta$')
81      plt.legend(loc="best")
82
83      ax2 = fig.add_subplot(212)
84      ax2.plot(t, res_ang_vel, label=r'$d\theta /dt$')
85      ax2.set_xlim(0, tmax)
86      plt.xlabel(r'$t$', fontsize=16)
87      plt.ylabel(r'$d\theta /dt$', fontsize=16)
88      plt.title(r'$t\ -\ d\theta /dt$')
89      plt.legend(loc="best")
90
91      plt.show()
92
93  if __name__ == '__main__':
94
95      default_params = [
96          {'theta': 0.1}, {'ang_vel': 0.0}, {'gamma': 0.2}, {'A': 0.85}]
97
98      window = sp.SetParameter()
99      window.show_setting_window(default_params, {'Run!': assignment})
```

3　実習課題

a. 外力を受けた単純減衰振り子のシミュレーションを行え。プログラムでは、外力の周期 T が π に等しくなるように $\omega = 2$、また、$\omega_0 = 1$ を仮定してよい。$\gamma = 0.2$ と $A = 0.85$ を用いて、位相空間の軌跡を計算せよ。初期の過渡現象の後に、ポアンカレ・プロットには何個の点があるか。このことから振り子の周期はいくらか求めよ。θ と $d\theta/dt$ の初期値を変えてみよ。アトラクタは初期条件に依存しないか。初期の過渡現象を無視することを忘れないこと。

$\gamma = 0.2$ と $A = 0.85$ として初期値 $\theta = 0.3$、$d\theta/dt = 0$ として位相空間に外力の周期 T ごとに 200 回計算を行い、うち最後の 100 回をプロットした。このポアンカレ・プロットを図 1 に示す。図 1 にはプロットされている点が 2 つしかないため、振り子の周期は $2\mathrm{T} = 2\pi$ であることがわかる。また、自明なことではあるが、アトラクタの位置は初期条件に依存する。ただし、振り子の周期は、運動の固定点以外であれば変わらない。固定点では周期 1 である。

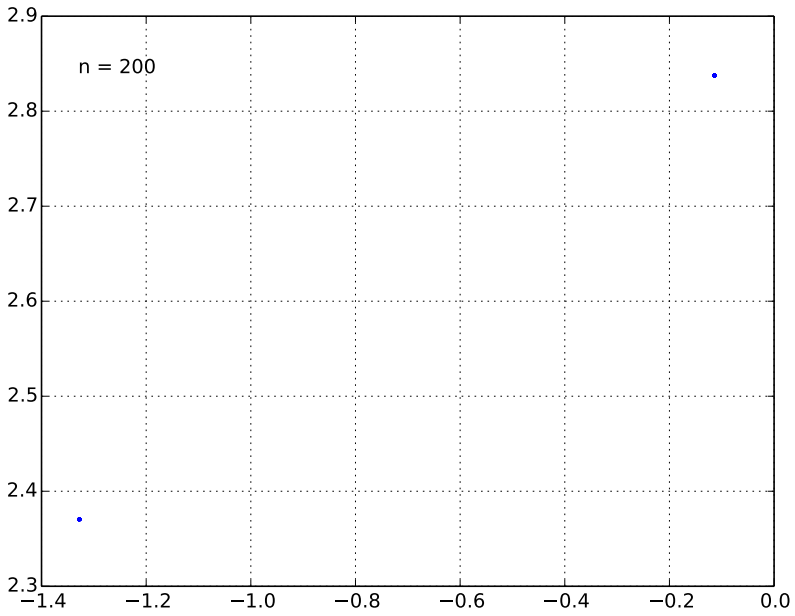

図 1　$\gamma = 0.2$、$A = 0.85$ としたときのポアンカレ・プロット

b. θ と $d\theta/dt$ を t の関数としてプロットせよ。そしてポアンカレ・プロット、位相空間の図、θ や $d\theta/dt$ の t 依存性の関係を定性的に述べよ。

θ と $d\theta/dt$ を t の関数としてプロットしたものを図 2 に示す。図 2 を見ると、t の大きいところでは θ と $d\theta/dt$ は t に対して周期関数として振舞っていることがわかる。これは先程ポアンカレ・プロットより得られた理解と一致する。

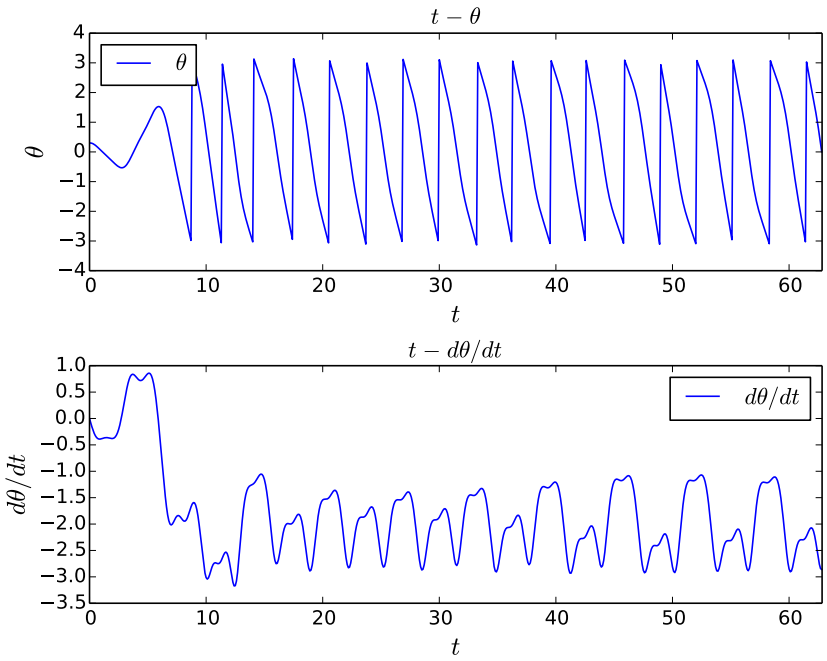

図 2　$\theta = 0.3$、$d\theta/dt = l0.0$、$\gamma = 0.2$、$A = 0.85$ のときの $t - \theta$、$t - d\theta/dt$ のグラフ

4　まとめ

4 次のルンゲクッタ法を用いた計算で、外力を受けた単純減衰振り子のシミュレーションを行うことができた。課題 c,d,e,f にあるように、この系は外力の振幅 A をパラメータとして周期倍化の性質を見ることができる。時間の都合上取り組むことができなかったが、知識として覚えておきたいと思う。

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.