計算機実習 問題 12.16 拡散が支配する 1 次元化学反応系のモンテカルロ・シミュレーション

早稲田大学先進理工学部物理学科 B4 藤本將太郎

2014/06/17

1　シミュレーションの目的

単一の種類 A の粒子からなる系を考える、全ての粒子は拡散し、2 個の粒子が衝突すると “反応 “が起こり、1 個の粒子が消えるか 2 個の粒子が結合して、もはや反応には関わらない不活性な種類の粒子になる。後者の場合の化学反応は

$$A + A \to 0 \quad (\text{不活性}) \tag{1}$$

と表すことができる。A 粒子の密度の空間的ゆらぎを無視すると、その時間変化は簡単な反応速度方程式

$$\frac{dA(t)}{dt} = -kA^2(t) \tag{2}$$

で与えることができる。ここで A は A 粒子の時刻 t の密度であり、k は反応速度定数である。簡単のために、全ての反応物質が $t = 0$ に加えられ、その後には反応物質は何も加えられないとする (閉じた系)。1 階微分方程式 (2) の解は

$$A(t) = \frac{1}{kt + 1/A(0)} \tag{3}$$

であり、長時間の極限では

$$A(t) \sim t^{-1} \tag{4}$$

となる。

上に述べた 1 種類の粒子が消滅する過程における A の時間依存性は簡単であるが、空間ゆらぎが無視されている。ここでは、この過程の時間発展のシミュレーションを行い、仮定が正しいか調べることにする。

2　作成したプログラム

本シミュレーションで作成したプログラムを以下に示す。

2.1 1 次元格子上で反応して不活性化する粒子の拡散

```
1  #! /usr/bin/env python
2  # -*- coding:utf-8 -*-
3  #
4  # written by Shotaro Fujimoto, June 2014.
5
6  import numpy as np
7
8  def main_rw_d1(L, N, pmax):
9
10     lattice = np.identity(L, dtype=int)
11     T = [2**p for p in range(1, pmax+1)]
12     A = []
13     random = np.random.rand
14     np_sum = np.sum
15     np_roll = np.roll
16
17     for t in xrange(1,max(T)+1):
18         # 1 step ============================================================
19         for n in xrange(N):
20             if np_sum(lattice[n]):
21                 if random() > 0.5: lattice[n] = np_roll(lattice[n], 1)
22                 else: lattice[n] = np_roll(lattice[n], -1)
23
24         count = np_sum(lattice, axis=0)
25
26         for l in xrange(L):
27             if count[l] == 2:
28                 for n in xrange(N):
29                     lattice[n][l] = 0
30         # ===================================================================
31         if t in T: A.append(L/(1.*np_sum(lattice)) - 1)
32
33     return T, A
34
35 def plot_graph(x_data, y_data, x_labels, y_labels,
36                xscale='linear', yscale='linear', aspect='auto'):
37     """ Plot the graph about y_data for each x_data.
```

```
38      """
39      import matplotlib.pyplot as plt
40
41      d = len(y_data)
42      if not len(x_data) == len(y_data) == len(x_labels) == len(y_labels):
43          raise ValueError("Arguments must have the same dimension.")
44      if d == 0:
45          raise ValueError("At least one data for plot.")
46      if d > 9:
47          raise ValueError("""So much data for plot in one figure.
48                              Please divide two or more data sets.""")
49
50      fig = plt.figure(figsize=(9,8))
51      subplot_positioning = ['11','21','22','22','32','32','33','33','33']
52      axes = []
53      for n in range(d):
54          lmn = int(subplot_positioning[d-1] + str(n+1))
55          axes.append(fig.add_subplot(lmn))
56
57      for i, ax in enumerate(axes):
58          ymin, ymax = min(y_data[i]), max(y_data[i])
59          ax.set_aspect(aspect)
60          ax.set_xscale(xscale)
61          ax.set_yscale(yscale)
62          ax.set_xlabel(x_labels[i], fontsize=16)
63          ax.set_ylabel(y_labels[i], fontsize=16)
64          ax.set_ymargin(0.05)
65          ax.plot(x_data[i], y_data[i], 'o-')
66
67      fig.subplots_adjust(wspace=0.2,hspace=0.5)
68      fig.tight_layout()
69      plt.show()
70
71  def fitting(fit_func, parameter0, trial=10):
72      import scipy.optimize as optimize
73
74      a = np.zeros(trial)
75      for i in range(trial):
76          T, A = main_rw_d1(L, N, pmax)
77          x, y = np.array(T), np.array(A)
```

```
78          result = optimize.leastsq(fit_func, parameter0, args=(x, y))
79          a[i] = result[0][0]
80
81      print 'a =', np.average(a), 'sigma(a) =', np.std(a)
82
83  if __name__ == '__main__':
84
85      L = 1000
86      N = L
87      pmax = 9
88
89      T, A = main_rw_d1(L, N, pmax)
90      plot_graph([T], [A], [r'$t$'], [r'$1/A(t)-1$'],
91                 xscale='log', yscale='log', aspect='equal')
92
93      def fit_func(parameter0, t, y):
94          a = parameter0[0]
95          b = parameter0[1]
96          residual = y - b*(t**a)
97          return residual
98      parameter0 = [1, 0] # a, b
99  #    fitting(fit_func, parameter0, trial=10)
100
```

このプログラムは 3 つの関数からなっている。関数 main_rw_d1 は、1 次元格子上を複数の粒子がランダムウォークし、同じ格子点上に 2 個の粒子が来たときその粒子を取り除くような挙動を実現する。関数 plot_graph は、問題 12 − 12 でも用いたグラフ描画のための関数である。関数 fitting は、$1/A(t) - 1$ の t に関する両対数グラフを直線でフィッティングしたときのその傾きを求めるものである。何回か試行を繰り返し、その平均値と偏差を返すようにしてある。

関数 main_rw_d1 の動きについての解説を加えると、まず lattice は行方向が粒子を表し、列方向は座標を表している。単位行列によって $A(t = 0) = 1$ を表現し、時間発展ごとに右向きか左向きに等確率でシフトする。その後、列方向に和を計算し、その値が 2 であるときはその列の要素をすべて 0 にしている (2 以上の値となることはない)。この操作を繰り返し、時刻 $t(= 2^p (p = 1, 2, 3, \cdots, \mathrm{pmax}))$ のときの $1/A(t) - 1$ の値を A に追加している。ここで t は対数プロットしたときのデータが等間隔になるように、指数関数的にとった。

3 実習課題

a. N 個の粒子が、長さ L の 1 次元格子上を周期的境界条件の下でランダムウォークを行う場合を考え、全ての格子点が占有されている状態 $A(t = 0) = 1$ からシミュレーションを始めよ。量 $1/A(t) - 1$ を時間 t に対して両対数でプロットせよ。対数プロットのデータが等間隔になるように、時間の区間を指数

関数的にとれ。長時間の極限では、得られた両対数プロットは直線的か、直線ならばその傾きを求めよ。平均場近似は 1 次元で正しいといえるか。$L = 100$ 程度の小さい格子と、$t = 10^2$ 程度の時間で大雑把な結果を得ることができる。10 パーセント以内の精度で結果を得るためには、$L = 10^4$ 程度の大きさの格子と $t = 2^{13}$ 程度の時間が必要である。

上記のプログラムを用いて、N 個の粒子が 1 次元周期境界条件のもとでランダムウォークを行い、反応によって不活性化する問題をシミュレーションした。このとき、条件として、格子の大きさ $L = 1000$ で、時刻 $t = 512(= 2^9)$ までの計算を行った。$A(t)$ を時刻 t での粒子の密度として、量 $1/A(t) - 1$ を時間 t に対して両対数プロットし、これを図 1 に示した。図から分かるように、この両対数プロットは直線的であり、$A(t)$ と t の間にベキ乗則が成り立つことが分かる。実際にグラフにおける直線の傾き a を算出してみると、10 回の試行の平均値として $a = 0.512988987355$ が得られた。このときの分散 σ は $\sigma = 0.0521364760422$ であった。したがって、時刻 t とその時刻での粒子の存在密度 $A(t)$ の間には、およそ以下のような関係が成り立つと分かる:

$$A(t) \sim t^{-\frac{1}{2}} \tag{5}$$

この結果は、空間ゆらぎを考慮せず解析的に求めた、長時間後の $A(t)$ と t の関係 (3) と異なっている。したがって、式 (2) のように平均場近似をすることは妥当でなく、粒子密度の空間的なゆらぎの効果は無視できないことが分かった。

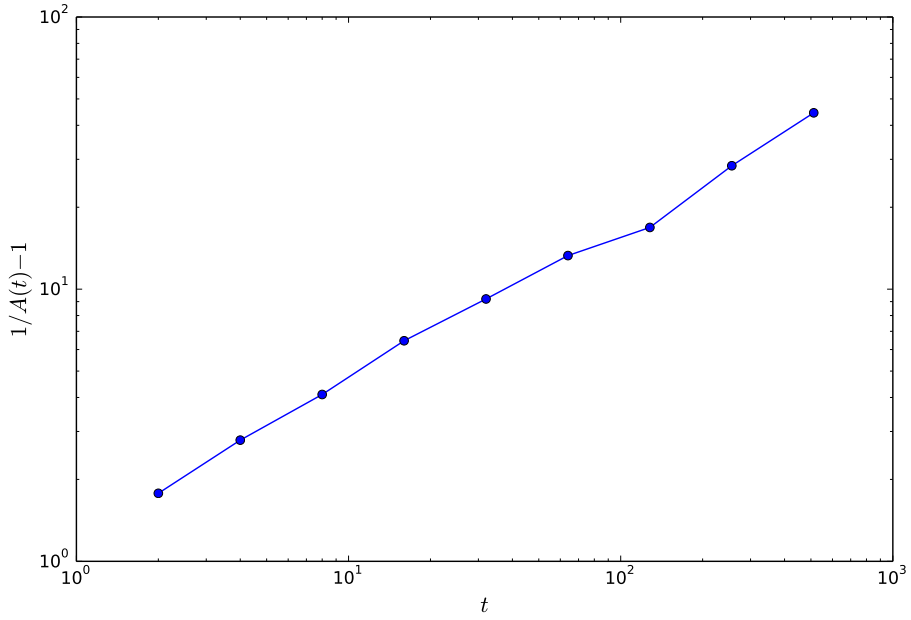

図 1　$1/A(t) - 1$ の時間 t に対する両対数プロット

4　まとめ

1 次元の化学反応系の簡単な例について考え、粒子の存在密度 $A(t)$ と時刻 t の間にベキ乗則が成り立ち、その指数の値が平均場近似より求められた値と異なることを示した。

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.