シミュレーション物理

演習課題 (3,4)

February 15, 2018

筑波大学 情報学群 情報科学類 二年
江畑 拓哉 (201611350)

Contents

1 実験の目的

本実験は、演習課題 (3) として引力を無視した場合の銀河の膨張のシュミレーションを行い、演習課題 (4) としてはガウス・ザイデル法を用いて偏微分方程式を解いた。前者はまず引力を考えずに銀河の膨張を考えることで引力を考えた場合の宇宙の膨張をシュミレーションする場合の足がかりとするための実験であり、後者は重力場を計算する場合等に必要となる偏微分方程式を解決するための手段を身につけるための実験である。

2 実験方法

演習課題 (3) についてはプログラム extension.c を作成し、実行した出力を Excel を用いて確認した。演習課題 (4) についてはプログラム gaus_seidel.c を作成し、実行した出力を標準出力から確認した。

実行環境についての情報を以下に列挙する。

- プログラムとコンパイル
  - Manjaro Linux 17.1.4
  - gcc (GCC) 7.2.1 20180116
  - GNU Emacs 26.0.91 (build 1, x86_64-pc-linux-gnu, GTK+ Version 3.22.26) of 2018-01-26
- Excel
  - Microsoft ® Excel ® 2016 MSO (16.0.8827.2131) 32bit

3 実験結果

- 演習課題 (3)　以下のグラフを得た。

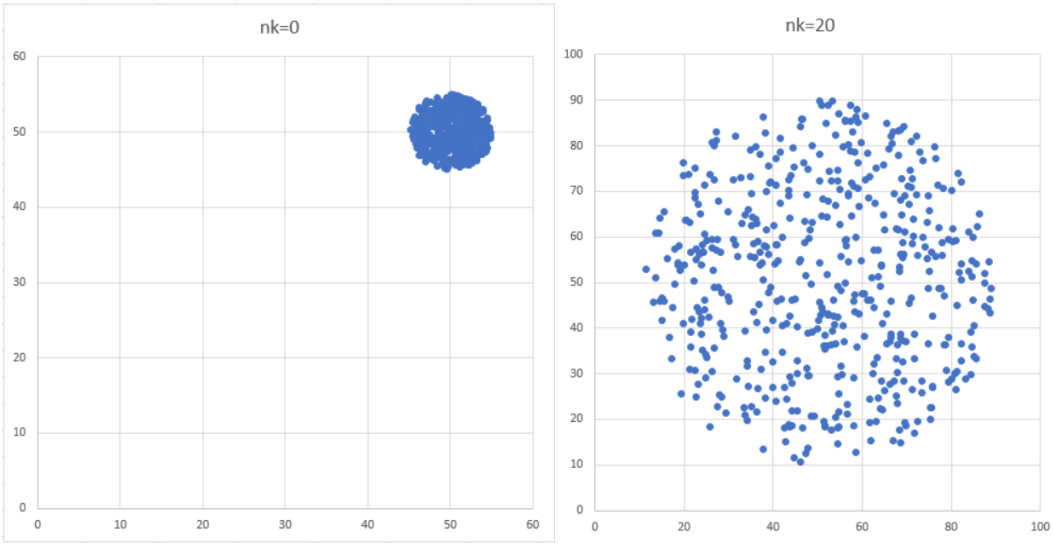

Figure 1: 演習課題 (3) の Scatter Plot

- 演習課題 (4)　標準出力から以下の出力を得た。

Listing 1: Standard Output

```
$ ./a.out
0.000000 22.500000 36.000000 0.000000
0.000000 9.000000 12.000000 9.000000
0.000000 1.500000 0.000000 -4.500000
0.000000 0.000000 0.000000 0.000000
```

解析解の式 $\Phi(x, y) = 3xy^2 - 1.5x^2y$ による出力予測を以下に示す。

Listing 2: 出力予測

```
0 22.5 36 40.5
0    9 12    9
0  1.5  0 -4.5
0    0  0    0
```

これにより出力が期待したものであると考えることが出来る。

4 考察

演習課題 (3) については単純な拡大図に近い出力が得られた。実際それぞれの点に対して同じ写像を適用しているのでこのような出力になることは当然であると考えられる。

演習課題 (4) については反復法の一種であるガウスザイデル法を行ったが、反復法の条件について気になった。反復法の多くは収束条件があり、ガウスザイデル法は係数行列が対角優位であることが求められるが、今回は特にそれについて検査を行わなかった。実用上はガウスザイデル法を適用する前に係数行列に検査を加えるか、問題を収束するものに限定する必要があるのではないかと考えている。

5 プログラムのリスト

extension.c

```
#include <stdio.h>
#include <stdlib.h>

const double H = 3.5;
const int org = 50;
const double delta_t = 0.1;
const int nk = 20;
const int seed = 449;
double X [500];
double Y [500];


void init_place();
int check (double, double);
void move ();
void print_move ();

double vx[500];
double vy[500];


void init_v () {
  for (int i = 0; i < 500; ++i) {
    vx[i] = H * X [i];
    vy[i] = H * Y [i];
  }
}

int check (double x, double y) {
```

```
30    if ((x * x + y * y) > 25.0) {
31      return 0;
32    } else {
33      return 1;
34    }
35  }
36
37  void init_place () {
38    double x, y;
39    srand(seed);
40    for (int t = 0; t < 500; ++t) {
41      x = -5.0 + (double) (10.0 * rand()/RAND_MAX);
42      y = -5.0 + (double) (10.0 * rand()/RAND_MAX);
43      while (0 == check (x, y)) {
44        x = -5.0 + (double) (10.0 * rand()/RAND_MAX);
45        y = -5.0 + (double) (10.0 * rand()/RAND_MAX);
46      }
47      X [t] = x;
48      Y [t] = y;
49    }
50  }
51
52
53  void move () {
54    for (int ip = 0; ip < 500; ++ip) {
55
56      X [ip] = X [ip] + vx[ip] * delta_t;
57      Y [ip] = Y [ip] + vy[ip] * delta_t;
58    }
59  }
60
61
62  void print_move () {
63    printf ("\n");
64    for (int t = 0; t < 500; ++t) {
65      printf ("%f,%f\n", X [t], Y [t]);
66    }
67  }
68
69  int main (void) {
70    init_place ();
71    init_v ();
72    for (int ip = 0; ip < 500; ++ip) {
73      X [ip] = X [ip] + org;
```

```
74      Y [ip] = Y [ip] + org;
75    }
76    print_move ();
77    for (int i = 0; i < nk; ++i) {
78      move ();
79    }
80    print_move ();
81    return 0;
82  }
```

gaus_seidel.c

```
1   #include <stdio.h>
2
3   double rho_func(int, int);
4   void init_phi();
5   void init_rho();
6   void gaus_xaiel(int);
7   void print_phi();
8
9   #define G  1.0
10  const int ni = 200;
11  const int nm = 2;
12  const double delta_x = 1;
13  const double delta_y = 1;
14  double phi[4][4];
15  double rho[4][4];
16
17  double rho_func(int x, int y) {
18    return (6 * x) - (3 * y);
19  }
20
21  void init_phi () {
22    for (int t = 0; t < 4 ; t++) {
23      phi[t][0] = 0.0;
24      phi[0][t] = 0.0;
25    }
26    phi[1][3] = 22.5;
27    phi[2][3] = 36.0;
28    phi[3][1] = -4.5;
29    phi[3][2] = 9.0;
30  }
31
32  void init_rho () {
```

```
33    for (int ix = 0; ix < 4; ix++) {
34      for(int iy = 0; iy < 4; iy++) {
35        rho[ix][iy] = rho_func(ix, iy);
36      }
37    }
38  }
39
40  void gaus_xaiel (int nm) {
41    double p1;
42    double p2;
43    for (int i = 1; i <= ni; i++) {
44      for (int ix = 1; ix <= nm; ix++) {
45        for (int iy = 1; iy <= nm; iy++) {
46          p1 = phi[ix+1][iy] + phi[ix-1][iy] + phi[ix][iy+1] + phi[ix][iy-1];
47          p2 = G * rho[ix][iy] * delta_x * delta_y;
48          phi[ix][iy] = p1 / 4.0 - p2 / 4.0;
49        }
50      }
51    }
52  }
53
54  void print_phi(){
55    for (int iy = 3; iy > -1; iy--) {
56      for (int ix = 0; ix < 4; ix++) {
57        printf("%f ",phi[ix][iy]);
58      }
59      printf("\n");
60    }
61  }
62  int main (void) {
63    init_phi();
64    init_rho();
65    gaus_xaiel(nm);
66    print_phi();
67    return 0;
68  }
```