# シミュレーション物理

## 演習課題 (5)

February 14, 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

本実験では過去の演習課題を組み合わせて引力を含んだ銀河の膨張をシュミレーションした。詳細には引力を含まない場合における銀河の膨張をシュミレーションしたものに、重力場を計算しそれぞれの天体が受ける力を加えたシュミレーションを行った。

## 2 実験方法

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。　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

以下のグラフを得た。

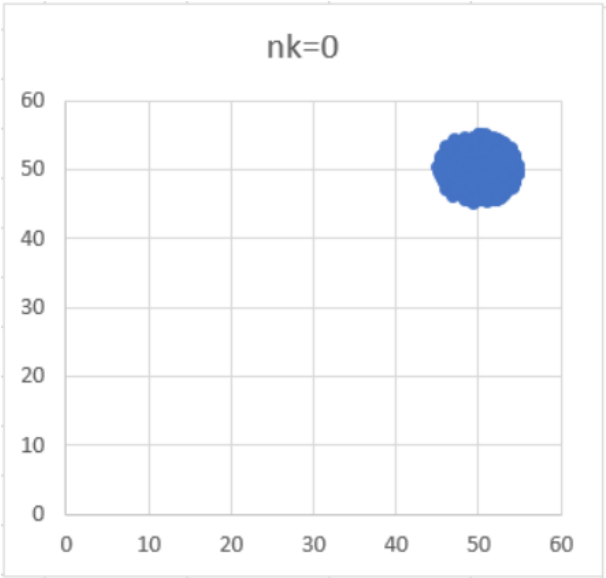


Figure 1: nk = 0 の天体の分布図

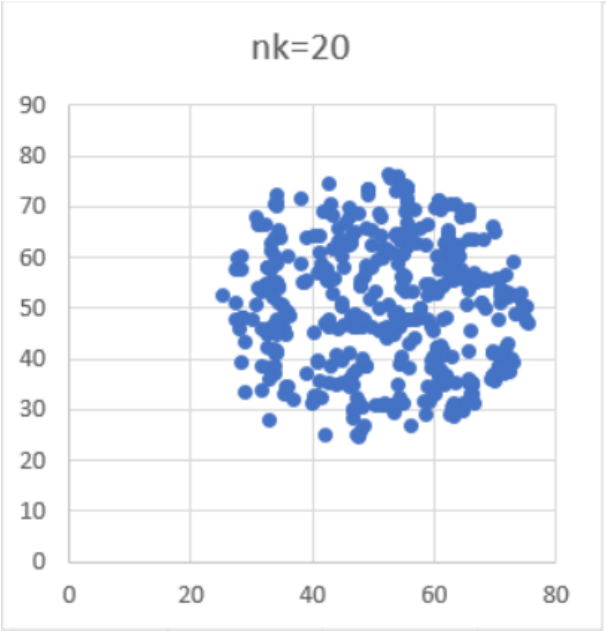


Figure 2: nk = 20 の天体の分布図

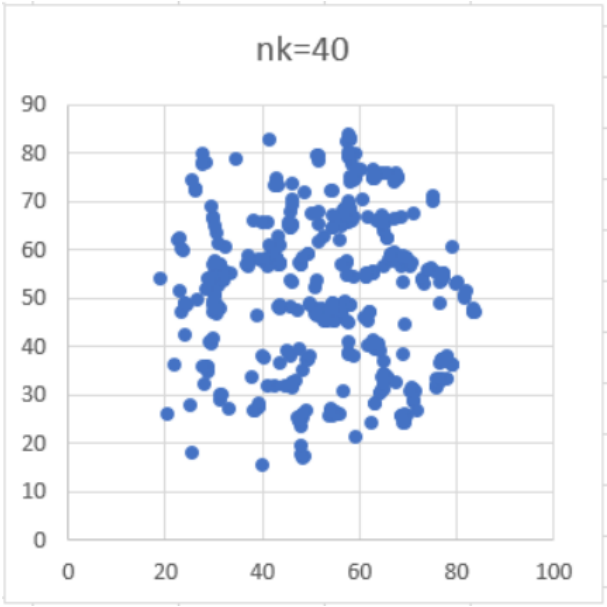


Figure 3: nk = 40 の天体の分布図

## 4 考察

天体のばらつき具合と拡散具合を確認した限り、銀河の膨張に関するシュミレーションを行うことが出来たと考えられる。また この場合の宇宙は 3 つの図を見る限りでは平坦な宇宙に近いと考えられる。この予測は特に nk = 20, nk = 40 の比較で銀河のサイズがほとんど変化していないことから考えられる。更に nk = 40 の図に注目すると大まかな天体の集合体が確認できることから、この後の進行もこの分布の形に沿って進んでいくことが、引力が計算されているという観点から 考えられる。

## 5 プログラムのリスト

extension.c

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

const int nm = 99;
const double M = 1.0;
const double G = 1.0;
const double H = 3.5;
const int ni = 99;
const double delta_t = 0.1;
const int seed = 449;
const double org = 50.0;
const double delta_x = 1.0;
const double delta_y = 1.0;

double X [500];
double Y [500];

double rho [500][500];
double phi [500][500];

double Fx[500][500];
double Fy[500][500];
double Fpx[500];
double Fpy[500];
double vx[500];
double vy[500];

void init_place();
int check(double, double);
void move_a ();
void calc_rho ();
void calc_phi ();
void calc_power_field ();
void calc_move ();
void calc_power ();
int find_point (double);
void calc_velocity ();
void calc_position ();
void move_b ();
```

```
41  void print_move ();
42
43  int check (double x, double y) {
44    if ((x * x + y * y) > 25.0) {
45      return 0;
46    } else {
47      return 1;
48    }
49  }
50
51  void init_place () {
52    double x, y;
53    srand(seed);
54    for (int t = 0; t < 500; ++t) {
55      x = -5.0 + (double) (10.0 * rand()/RAND_MAX);
56      y = -5.0 + (double) (10.0 * rand()/RAND_MAX);
57      while (0 == check (x, y)) {
58        x = -5.0 + (double) (10.0 * rand()/RAND_MAX);
59        y = -5.0 + (double) (10.0 * rand()/RAND_MAX);
60      }
61      X [t] = x;
62      Y [t] = y;
63    }
64  }
65
66  void init_phi () {
67    for (int i = 0; i < 500; ++i) {
68      for (int j = 0; j < 500; ++j) {
69        phi[i][j] = 0;
70      }
71    }
72  }
73
74  void move_a () {
75    for (int ip = 0; ip < 500; ++ip) {
76      X [ip] = X [ip] + vx[ip] * delta_t;
77      Y [ip] = Y [ip] + vy[ip] * delta_t;
78    }
79  }
80
81  void calc_rho () {
82    for (int i = 0; i < 500; ++i) {
83      for (int j = 0; j < 500; ++j) {
84        rho[i][j] = 0.0;
```

```
85      }
86    }
87    for (int i = 0; i < 500; ++i) {
88      rho[(int)floor(X [i] + 0.5)][(int)floor(Y [i] + 0.5)] += M;
89    }
90  }
91
92  void calc_phi () {
93    double p1;
94    double p2;
95    for (int i = 1; i <= ni; ++i) {
96      for (int ix = 1; ix <= ni; ++ix) {
97        for (int iy = 1; iy <= ni; ++iy) {
98          p1 = phi [ix + 1] [iy] + phi [ix - 1] [iy]
99            + phi [ix] [iy + 1] + phi [ix] [iy - 1];
100         p2 = G * rho [ix] [iy] * delta_x * delta_x;
101         phi [ix] [iy] = (p1 - p2) / 4.0;
102       }
103     }
104   }
105
106
107 void calc_power_field () {
108   for (int ix = 0; ix < 500; ++ix) {
109     for (int iy = 0; iy < 500; ++iy) {
110       Fx[ix][iy] = ( - ((phi[ix + 1][iy]) - (phi[ix][iy]))) / delta_x;
111       Fy[ix][iy] = ( - ((phi[ix][iy + 1]) - (phi[ix][iy]))) / delta_y;
112     }
113   }
114 }
115
116 int find_point (double p) {
117   return (int) floor(p + 0.5);
118 }
119
120 void calc_power () {
121   int x, y;
122   for (int i = 0; i < 500; i++) {
123     x = find_point (X [i]);
124     y = find_point (Y [i]);
125     Fpx [i] = M * Fx [x] [y];
126     Fpy [i] = M * Fy [x] [y];
127   }
128 }
```

```
129
130 void calc_velocity () {
131   for (int ip = 0; ip < 500; ip++) {
132     vx [ip] = vx [ip] + (Fpx [ip] / M) * delta_t;
133     vy [ip] = vy [ip] + (Fpy [ip] / M) * delta_t;
134   }
135 }
136
137 void calc_position () {
138   for (int i = 0; i < 500; ++i) {
139     X[i] += vx[i] * delta_t;
140     Y[i] += vy[i] * delta_t;
141   }
142 }
143
144 void calc_move () {
145   calc_power ();
146   calc_velocity ();
147   calc_position ();
148 }
149
150 void move_b () {
151   calc_rho ();
152   calc_phi ();
153   calc_power_field ();
154   calc_move ();
155 }
156
157 void print_move () {
158   printf ("\n");
159   for (int t = 0; t < 500; ++t) {
160     printf ("%f,%f\n", X [t], Y [t]);
161   }
162 }
163
164 int main (void) {
165   init_place ();
166
167   for (int ip = 0; ip < 500; ++ip) {
168     vx[ip] = H * X [ip];
169     vy[ip] = H * Y [ip];
170
171     X [ip] = X [ip] + org;
172     Y [ip] = Y [ip] + org;
```

```
173   }
174
175   print_move ();
176   init_phi ();
177
178   for (int i = 0; i < 20; ++i) {
179     move_b ();
180   }
181   print_move ();
182   for (int i = 0; i < 20; ++i) {
183     move_b ();
184   }
185   print_move ();
186   return 0;
187 }
```