シミュレーション物理

演習課題 (1)

January 25, 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

本実験はヒットミス法を用いた半径=1 の円の面積を求めるにあたり、その視覚的特徴とヒットミス法の精度について学習するためのものである。具体的には、 ① モンテカルロ法によって得られる半径 1 の円内に分布する点群を確認しこの分布がどのようになっているのかを視覚的に確認する、 ② ヒットミス法によって得られる点群がどの程度円周率と関連づいているのかを確認する、というそれぞれの目的に基づいて出題されている。

2 実験方法

作成したそれぞれのプログラムを実行する。実験 ① で用いたプログラムは gen_graph.c であり、実験 ② で用いたプログラムは hitmiss.c である。

実験 ① の結果確認は出力を csv ファイルとして受け取りこれを Excel で読み込んで Scatter plot を行い、実験 ② は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 17.1.2
  - gcc (GCC) 7.2.1 20171224
  - GNU Emacs 25.3.1 ($x86_{64}$-pc-linux-gnu, GTK+ Version 3.22.26) of 2017-12-05
- Excel
  - Microsoft ® Excel ® 2016 MSO (16.0.8827.2131) 32bit

3 実験結果

- 実験 ①
  以下のグラフを得た。

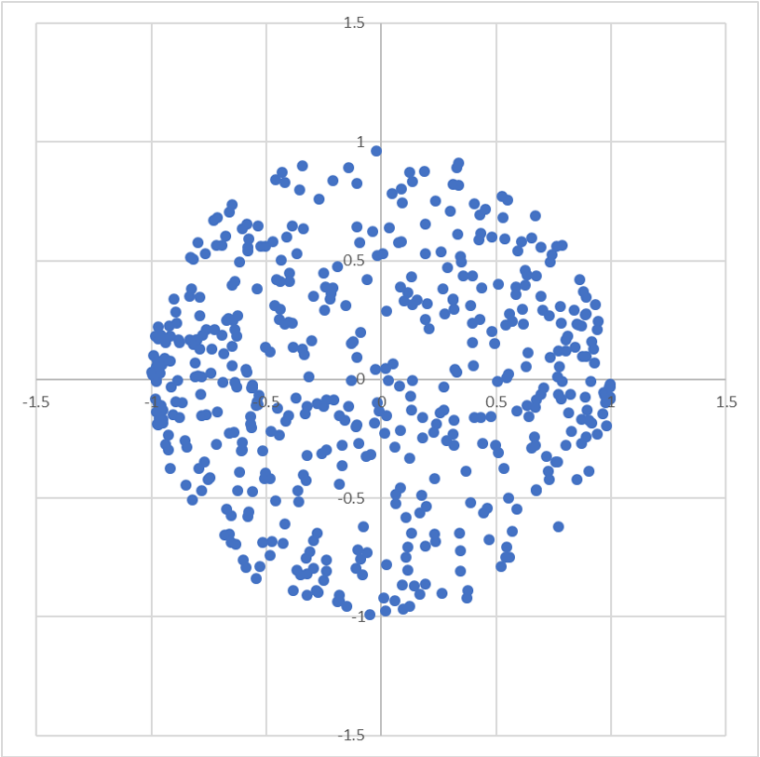

Figure 1: 実験 ① の Scatter Plot

- 実験 ②
  以下の標準出力を得た。

```
  100: 3.116000 : 0.008146
 1000: 3.166400 : -0.007897
10000: 3.143320 : -0.000551
```

これを指定されている表に合わせると以下のようになる。

Table 1: 実験 ② の 表

乱数の組 (点の数)	100	1000	10000
面積の平均値 (10 試行)	3.116000	3.166400	3.143320
(π - 面積の平均値) $\div \pi$	0.008146	-0.007897	-0.000551

4 考察

実験 ① については、図から密度がほぼ一様な円の形をはっきりと認識できる。点群そのものが一様乱数によって発生しているため、円外の点を除いた点群が円の形に見えるこの結果は当然のものだと考えられる。C 言語で実装したために配列の長さを柔軟に変えることが難しく、乱数の組 (x, y) の片方 x の条件に合うような y を選ぶようなアルゴリズムになっているが、別言語での実装を行うならばある無限長の長さの乱数の組 (x, y) を作成し、これを先頭から条件を満たすものだけ取り出していく形を取った方がより題意に沿っているのではないかと考えている。

実験 ① については、ミスヒット法がどの程度円周率に近似できるかを確認することができたが、これは円の面積分だけ面積一定な小さいセルをはめ込んでいると考えれば $\pi * 1 * 1 = \pi$ に値が等しくなることが理解できる。またセル面積が小さいほど、つまりセルの数が多くなるほど誤差が小さくなることはラスター画像を拡大した際に見ることができるジャギーを想像すれば理解できる。この実験において乱数の組の数を 100 倍して誤差が 1/10 未満に下がったことは素晴らしいことだと考えられる。また一様乱数の生成を独立して行うことができる場合、これを並列処理することができればより短い時間で同じ数の点を打つことができ、それに伴って精度も向上すると考えている。

5 プログラムのリスト

gen _ graph.c

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

void result_to_csv (int, double*, double*);
void set_correct_values (int, double*, double*);
int check (double, double);

int main (void) {
  int N = 500;
  double x [N];
  double y [N];

  set_correct_values(N, x, y);
```

```
  result_to_csv(N, x, y);
  return 0;
}

void set_correct_values (int N, double* x, double* y) {
  srand(691);
  int i, j;
  double x_, y_;
  for (i = 0; i < N; ++i) {
    x[i] = x_ =  -1.0 + (2.0 * rand() / (RAND_MAX + 1.0));
    y_ = -                      1.0 + (2.0 * rand() / (RAND_MAX + 1.0));
    while (check (x_, y_)) {
      y_ = -1.0 + (2.0 * rand() / (RAND_MAX + 1.0));
    }
    y [i] = y_;
  }
}

int check (double x, double y) {
   if (1.0 >= ((x * x) + (y * y))) {
     return 0;
   }
  return 1;
}

void result_to_csv (int N, double* x, double* y) {
  FILE *fo;
  int i;
  char* fname = "data.csv";
  if ((fo = fopen(fname, "w")) == NULL) {
    printf ("File[%s] does not open!! \n", fname);
    exit (0);
  }
  for (i = 0; i < N; ++i) {
    fprintf (fo, "%f, %f\n", x [i], y [i]);
  }
}
```

hitmiss.c

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

```

```
4  #define MY_PI 3.14159
5
6  void set_values (int, int, double*, double*);
7  double hit_miss_test (int, double*, double*);
8
9  int main (void) {
10   double *x;
11   double *y;
12
13   int t[3] = {100, 1000, 10000};
14   int seed [10] = {149, 193, 251, 383, 457, 503, 691, 761, 829, 991};
15   int i, j;
16   double hits;
17   for (i = 0; i < sizeof (t) / sizeof (t [0]); ++i) {
18     hits = 0.0;
19     x = (double*) malloc (sizeof (double) * t[i]);
20     y = (double*) malloc (sizeof (double) * t[i]);
21
22     if ((x == NULL) || (y == NULL)) {
23       printf ("malloc error \n");
24       exit (0);
25     }
26
27     for (j = 0; j < 10; ++j) {
28       set_values(seed [j], t[i], x, y);
29       hits += hit_miss_test (t [i], x, y);
30     }
31     hits = (hits / 10.0) * 4.0;
32     printf ("%5d: %f : %f\n",
33             t [i], hits, ((MY_PI - hits) / MY_PI));
34   }
35   free(x);
36   free(y);
37   return 0;
38 }
39
40 double hit_miss_test (int N, double* x, double* y) {
41   int i;
42   int hit = 0;
43   for (i = 0; i < N; ++i) {
44     if (1.0 >= ((x [i] * x [i]) + (y [i] * y [i]))) {
45       hit++;
46     }
47   }
```

```
48    return ((double) hit / N);
49  }
50
51  void set_values (int seed, int N, double *x, double *y) {
52    srand(seed);
53    int i;
54    for (i = 0; i < N; ++i) {
55      x[i] = -1.0 + (2.0 * rand() / (RAND_MAX + 1.0));
56      y[i] = -1.0 + (2.0 * rand() / (RAND_MAX + 1.0));
57    }
58  }
```