

2023 年度 システムプログラミング実験

確率プログラミング

レポート

学修番号: 22140003

氏名: 佐倉仙汰郎

第 1 回レポート提出日: 2023/10/24

第 2 回レポート提出日: 2023/10/31

第 3 回レポート提出日: yyyy/mm/dd

はじめに

本書ではシステムプログラミング実験第三回の課題を実験した結果を報告する。課題は乱数の生成、また乱数を用いた確率の解析である。解析結果をグラフと数値により示し、その結果について考察を行う。

実験の概要

本実験ではモンテカルロ法を用いて解析を進める。モンテカルロ法では乱数を用いてシミュレーションを行う手法のことである。課題 1 ではモンテカルロシミュレーションを行うために必要な乱数の生成を行う関数をつくる。課題 2 では課題 1 で作った関数を用いて、コイン投げのシミュレーションを行う。課題 3 では課題 1 で作った関数を用いて、さいころ投げのシミュレーションを行う。追加課題 A では、課題 1 で作った関数を用いて、さいころ投げの趣味レーションを課題 3 とは違う設定で行う。それぞれの課題で得られた結果をもとに考察を行う。

実験環境

前節で説明した方法を C++ 言語^{*1}により実装した。実験環境の仕様を次に示す。

- Central Processing Unit: 11th Gen Intel(R) Core(TM) i7-1167G7 @ 2.80 GHz
- 主記憶: Double Data Rate 4 Synchronous Dynamic Random-Access Memory
- コンパイラ: g++ version 11.2.0
- Operating System: Arch Linux^{*2}
- 数値型: 倍精度浮動小数点数^{*3}

^{*1} <https://isocpp.org/std/the-standard>

^{*2} <https://archlinux.org/>

^{*3} <https://www.gnu.org/software/gsl/doc/html/ieee754.html>

課題 1 - 1

実験の説明

区間 $[0, 1)$ の一様乱数を独立に n 個生成する関数を作成する。作成した関数を用いて $n = 1000$ 個の乱数を生成し、その平均値と分散を計算する。今回は標準ライブラリに搭載されている、`std::random` 内の関数 `std::random_device`, `std::mt19937` および `std::uniform_real_distribution` を使用した。(ファイル `kadai_1_1_sentaro_sakura.cpp` を参照)

実験結果

メルセンヌツイスター法を用いて 1000 個の乱数から、平均値と分散を求めた結果が以下のとおりである。
平均値:0.498

分散:0.0858

平均値はおおよそ理論値に近くなった。分散が 0.0858 と非常に小さな値になっていることから一様に乱数が分布していることが分かる。

考察

今回の実験で得られた値は理想地に近い値となった。このことからメルセンヌツイスター法が乱数の生成に妥当である。

課題 1 - 2

実験の説明


課題 1-1 で作成したプログラムをもとにコイン投げのシミュレーションプログラムを作成する。作成したプログラムを用いて 1,000 回のコイン投げのシミュレーションを行い、表・裏それぞれが出た確率を求める。ただし、表が出る確率を p 、裏が出る確率を $1-p$ とする。レポートには、 $p = 0.2, 0.5, 0.7$ の 3 通りそれぞれについて、関数 `rnd exp` を用いて生成した $n = 1,000$ 個のコイン投げに対する結果から考察を行う。シミュレーションで得られたコインの表・裏の確率は、有効数字 3 桁で報告する。

生成した n 個の乱数 r_1, \dots, r_n を用いて、 i 回目のコイン投げ試行の結果を以下のように定める：(i) $r_i \leq p$ であれば、 i 回目のコイン投げ試行で表が出たとする。(ii) $r_i > p$ であれば、 i 回目のコイン投げ試行で裏が出たとする。

実験結果

$$p = 0.3 \text{ の時 } 0.704 \qquad p = 0.5 \text{ の時 } 0.499 \qquad p = 0.7 \text{ の時 } 0.282 \qquad (1)$$

実験結果は以上の通りになった。すべての p の値に対して理論値に近い値である。このことから、乱数生成が適切に稼働していることが分かった。



my_plot1-3.png

図 1 実験結果のグラフは、課題 1 - 1 で作った関数を参照

課題 1 - 3

実験の説明

サイコロ投げのシミュレーションプログラムを作成する。各目の出る確率は等確率とする。作成したプログラムを用いて 1000 回の試行を行い、1000 回の試行で出たサイコロの目の平均値を求め、その結果を用いて、横軸を試行回数 n 、縦軸を n 回の試行の平均値としたグラフを作成し、結果を考察する。

実験結果

さいころの目を確率変数 X とし、確率変数の期待値を $E[X]$ とする。さいころの期待値は以下の式で表せる。

$$\begin{aligned} E[X] &= \frac{1+2+3+4+5+6}{6} \\ &= \frac{21}{6} \\ &= \frac{7}{2} \\ &= 3.5 \end{aligned}$$

平均値:3.53 (2)

1000 回さいころの目をふった平均値は図 (1) の通りになる。平均値が n の値が大きくなるにつれ 3.5 にちかづいていることがわかり、今回の実験で妥当な値が出ている。

課題 1 - A

実験の説明

課題 1-3 において各目 $i = 1, 2, \dots, 1000$ の出る確率 p_i が次のように偏っていたとする． $p_1 = p_3 = p_5 = 1/9, p_2 = p_4 = p_6 = 2/9$ ．このとき，課題 1-3 の結果がどう変わるかをグラフを作成して示し、その結果から考察を行う．

実験結果

各目の出る確率が $p_1 = p_3 = p_5 = 1/9$ および $p_2 = p_4 = p_6 = 2/9$ の場合、各目が出る確率 $P(i)$ は以下のよう表される．

$$P(1) = P(3) = P(5) = \frac{1}{9} \qquad P(2) = P(4) = P(6) = \frac{2}{9} \qquad (3)$$

期待値 E を計算するには以下のようになる．

$$\begin{aligned} E &= \sum_{i=1}^6 i \cdot P(i) \\ &= 1 \cdot \left(\frac{1}{9}\right) + 2 \cdot \left(\frac{2}{9}\right) + 3 \cdot \left(\frac{1}{9}\right) + 4 \cdot \left(\frac{2}{9}\right) + 5 \cdot \left(\frac{1}{9}\right) + 6 \cdot \left(\frac{2}{9}\right) \\ &= \frac{1}{9} + \frac{4}{9} + \frac{3}{9} + \frac{8}{9} + \frac{5}{9} + \frac{12}{9} \\ &= \frac{1+4+3+8+5+12}{9} \\ &= \frac{11}{3} \\ &= 3.67 \end{aligned}$$

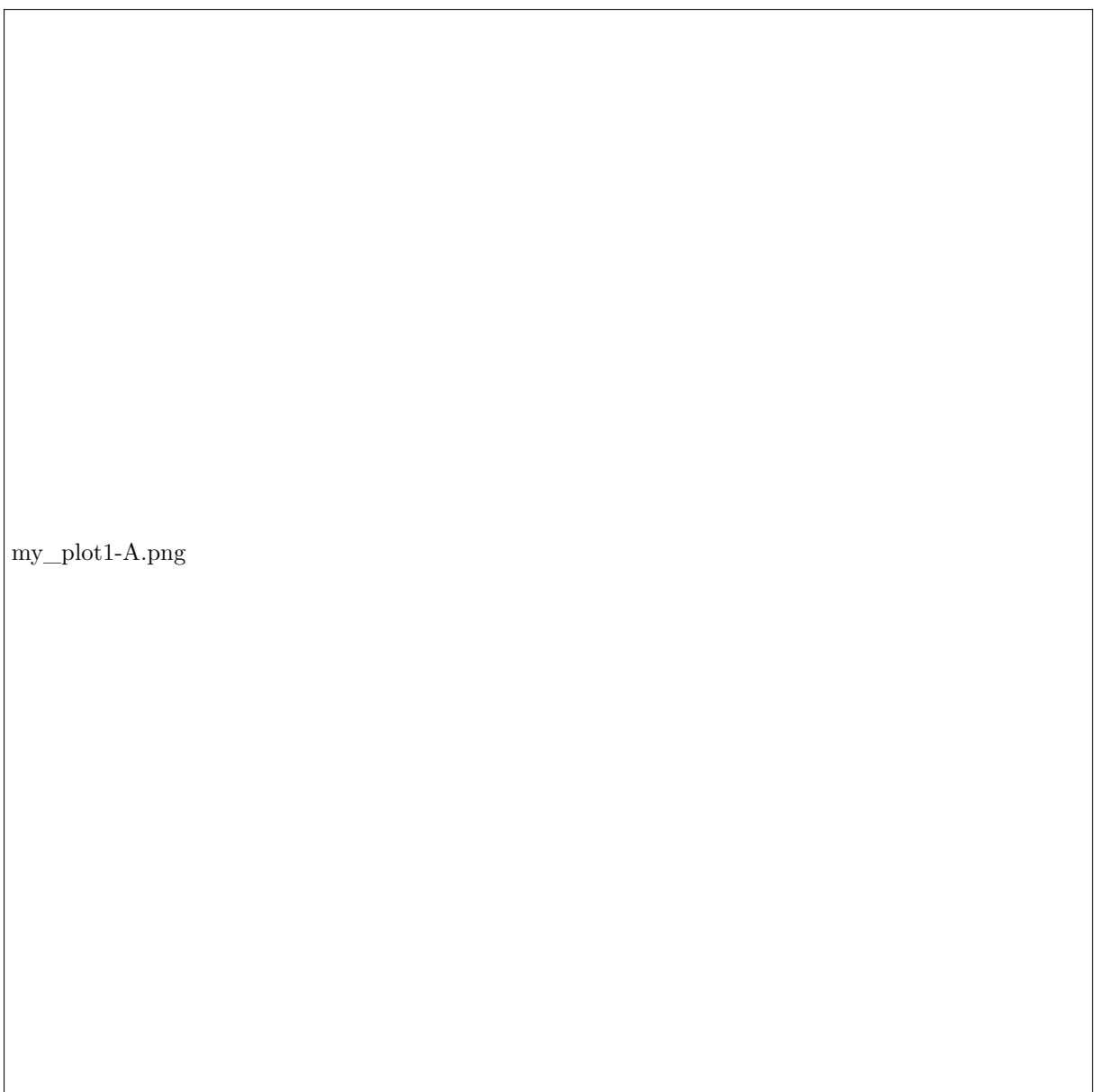
実験で得られた値は以下のとおりである．

$$\text{平均値: } 3.67 \qquad (4)$$

となり、有効数値三桁では十分な精度である．また図 (2) から理論値に知数していることが分かる．

おわりに

今回の実験ではたくさんのソースコードを作り、様々なライブラリなどを制作した．課題の目標であった、実践的なコーディング能力上昇に大きく貢献したと思う．課題が複数ある慣れない形式ではあったが、それぞれについて適切な考察を行えたと思う．



my_plot1-A.png

図 2

1 課題 2 - 1

実験の概要円周率の近似をモンテカルロ法を用いて求める．どのようにして円周率 π が求まるかをいかに示す．

区間 $[0, 1)$ の乱数の組 (r_1, r_2) を独立に n 個生成する．一辺が 1cm の正方形の中に半径 1cm の円が四等分されたものがあるとする．生成した乱数の組がその扇内にいくつ存在するかを調べ、その個数を総数で割ると、扇形の面積の近似値が出る．半径 1 の扇形の面積の $1/4$ は $\frac{\pi}{4}$ となるので寄れを 4 倍することで、円周率 π の近似値が得られる．

付録

今回の課題に使用したソースコードを以下に示す.

```
1  #include<iostream>
2  #include<random>
3  using namespace std;
4
5  //ランド関数を作る
6  void rnd_exp(double A[], int n){
7      random_device rd;
8      mt19937 gen(rd());
9      uniform_real_distribution<double> distribution(0.0,1.0);
10     for(int i = 0; i < n; i++){
11         double r = distribution(gen);
12         A[i] = r;
13     }
14 }
15 int main(){
16     int n = 1000;
17     double A[n];
18     rnd_exp(A,n);
19     double sum1 = 0;
20     double sum2 = 0;
21     for(int i = 0; i < n ; i++){
22         sum1 += A[i];
23         sum2 += A[i] * A[i];
24     }
25     double ave = sum1 / n;
26     cout << "平均値： " << sum1/n << endl;
27     cout << "分散： " << sum2/n - ave*ave << endl;
28     return 0;
29 }
```

```
1  #include<iostream>
2  #include<random>
3  using namespace std;
4
5  //ランド関数を作る
6  void rnd_exp(double A[], int n){
7      random_device rd;
8      mt19937 gen(rd());
9      uniform_real_distribution<double> distribution(0.0,1.0);
10     for(int i = 0; i < n; i++){
11         double r = distribution(gen);
12         A[i] = r;
13     }
14 }
15
16 int main(){
17     int n = 1000;
18     double A[n];
```



```

19  rnd_exp(A,n);
20  double p1 = 0.3;
21  double p2 = 0.5;
22  double p3 = 0.7;
23  double R1, R2, R3;
24  R1 = 0;
25  R2 = 0;
26  R3 = 0;
27  for(int i = 0; i < n; i++){
28      if(A[i] < p1){
29          }
30      else if(p1 <= A[i] && A[i] <= p2 ){
31          R1++;
32      }
33      else if(p2 < A[i] && A[i] <= p3){
34          R1++;
35          R2++;
36      }
37      else if(p3 < A[i]){
38          R1++;
39          R2++;
40          R3++;
41      }
42  }
43  cout << "p = 0.3の時" << R1/n << endl;
44  cout << "p = 0.5の時" << R2/n << endl;
45  cout << "p = 0.7の時" << R3/n << endl;
46
47  return 0;
48 }

```

```

1  #include<iostream>
2  #include<random>
3  #include<fstream>
4  using namespace std;
5
6  //ランド関数を作る
7  void rnd_exp(double A[], int n){
8      random_device rd;
9      mt19937 gen(rd());
10     uniform_real_distribution<double> distribution(0.0,1.0);
11     for(int i = 0; i < n; i++){
12         double r = distribution(gen);
13         A[i] = r;
14     }
15 }
16 int GenDice(){
17     random_device rd;
18     mt19937 gen(rd());
19     uniform_int_distribution<int> distribution(1, 6);
20     return distribution(gen);
21 }
22

```

```

23 int main(){
24     ofstream of("DiceAverage.csv");
25     of << "N,Average" << endl;
26     int n = 1000;
27     double A[n];
28     rnd_exp(A,n);
29     double sum = 0;
30
31     for(int i = 0; i < n; i++){
32         sum += GenDice();
33         of << i+1 << ", " << sum/(i+1) << endl;
34     }
35     cout << sum/n << endl;
36     return 0;
37 }

```

```

1  #include<iostream>
2  #include<random>
3  #include<fstream>
4  using namespace std;
5
6  //ランド関数を作る
7  void rnd_exp(double A[], int n){
8      random_device rd;
9      mt19937 gen(rd());
10     uniform_real_distribution<double> distribution(0.0,1.0);
11     for(int i = 0; i < n; i++){
12         double r = distribution(gen);
13         A[i] = r;
14     }
15 }
16 int GenDice(){
17     random_device rd;
18     mt19937 gen(rd());
19     uniform_int_distribution<int> distribution(1, 9);
20     int a = distribution(gen);
21     if(a == 7) a = 2;
22     if(a == 8) a = 4;
23     if(a == 9) a = 6;
24     return a;
25 }
26
27 int main(){
28     ofstream of("DiceAverage1-A.csv");
29     of << "N,Average" << endl;
30     int n = 1000;
31     double A[n];
32     rnd_exp(A,n);
33     double sum = 0;
34
35     for(int i = 0; i < n; i++){
36         sum += GenDice();
37         of << i+1 << ", " << sum/(i+1) << endl;

```

```
38 | }  
39 | return 0;  
40 | }
```

(参考文献. [?]などを参考に, bib ファイルの使い方はウェブなどで勉強すること.)