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：2023/11/15
第 4 回レポート提出日：2023/11/21

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 $r_1, ..., r_n$ 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：$(i) r_i <= p$ 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

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

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．

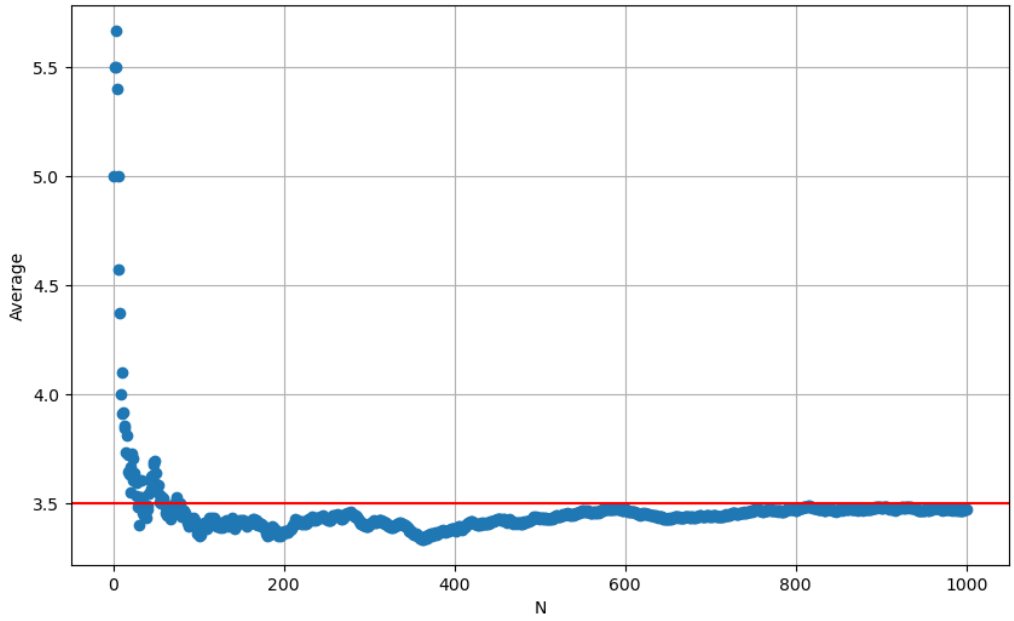

図 1　実験結果のグラフは、課題１－１で作った関数を参照

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 $E[]$ 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 にちかず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．

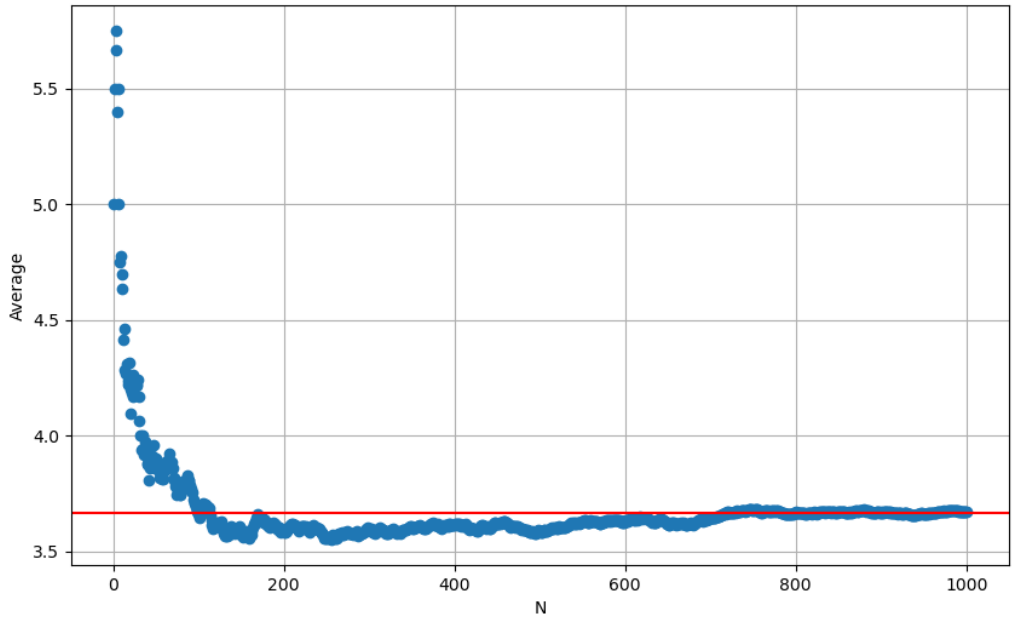

図 2　実験結果のグラフは、課題１－１で作った関数を参照

課題１－ A

実験の説明

課題 1-3 において各目 $i = 1, 2, ..., 1000$ 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 $P(i)$ は以下のように表される．

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

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．

平均値:3.67 (4)

となり、有効数値三桁では十分な精度である．また図 (2) からも理論値に知数い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．

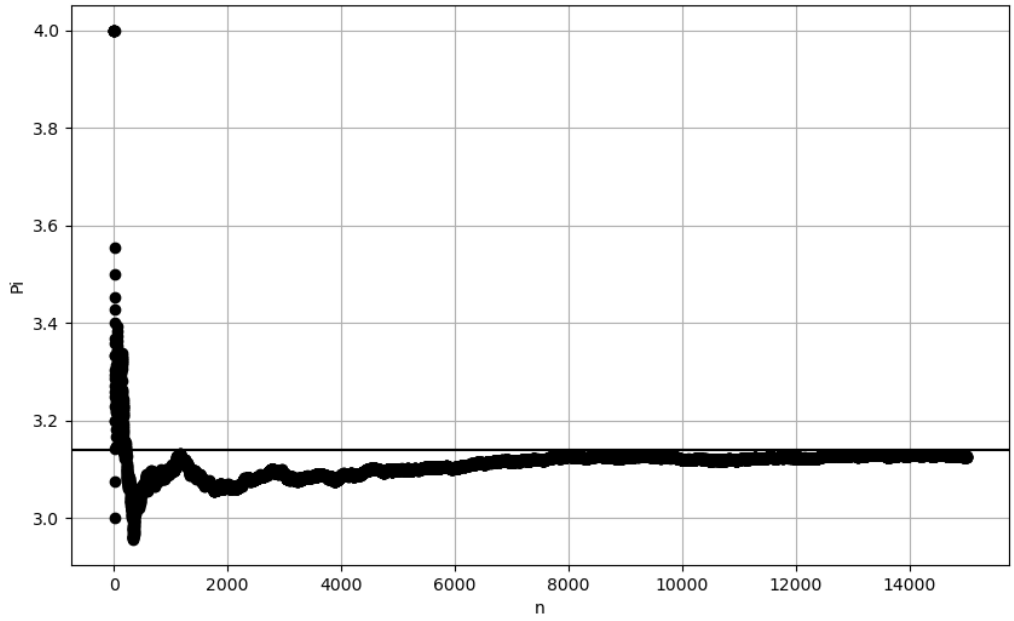

図 3　実験結果のグラフは、課題１－１で作った関数を参照

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 が求まる火をいかに示す．
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 円の扇形の面積の $\frac{1}{4}$ は $(\frac{\pi}{4})$ となるので 4 倍することで、円周率 π の近似値が得られる．

実験結果

実験の概要で説明した方法で得られた π の近似値が縦軸とし、n の要素数を横軸に示す．今回は $n = 15000$ で実験を行った．$n = 15000$ で実験を行った結果の最終的な近似値は 3.1267 となった．グラフから $n = 8000$ あたりから近似値がずれてきていることが分かる．

考察

今回の円周率の近似の結果はあまり正確なものではなかった．この理由として考えられるのは乱数が一様に生成されなかったことが考えられる．今回の実験で、再現性を持たせるためにシードを学習番号に固定したが乱数に偏りを持たせてしまった可能性がある．テストとして、シードを固定せずに実験を行った場合、$n = 10000$ の時点で有効数字 4 桁で円周率と近似値が一致した．より正確な近似値を得るためにはそれに適し

た一様な乱数を生成することが重要だろう.

実験２－２

１回の試行結果が確率変数 X_n によって表される（）した実験を繰り返す．その結果において、X_n の（平均値）を m と表すとき、どんなに小さな $h(h>0)$ であっても n が大きければ以下が成り立つ．

$$P\left(\left|\frac{X_1+X_2+\ldots+X_n}{n}-m\right|\geq h\right)\to 0 \quad (1)$$

この法則を適用して課題 2-1 を説明すると，確率変数 X_1 は 1 回の実験で円周の内側に入る事象 A が起これば 1，事象 A が起こらない（円周の外側に入る）場合は 0 をとるものとする．このときそれぞれの事象が起こる確率は

$$P(X_n=1)=\frac{\pi}{4},\quad P(X_n=0)=\frac{3\pi}{4} \quad (2)$$

となるため，m は以下のように計算できる．

$$m=1\cdot P(X_n=1)+0\cdot P(X_n=0)=\frac{\pi}{4} \quad (3)$$

この実験を何回も繰り返すことで確率変数 $X_1, X_2, \ldots$ を得ることができる．n 回までの試行で事象 A が起こる回数は $X_1+X_2+\ldots+X_n$ に等しいので，大数の法則に当てはめると

$$P\left(\left|\frac{X_1+X_2+\ldots+X_n}{n}-\frac{\pi}{4}\right|\geq h\right)\to 0 \quad (4)$$

となる．つまり，$\frac{X_1+X_2+\ldots+X_n}{n}$ は $n\to\infty$ のとき 1 に確率収束するため円周率を求めることが可能となる。

実験２ー３

実験の概要

今回の実験ではモンティホール問題をシミュレーションする．モンティホール問題には以下のステップがある．

1. 3 つのドアがあり、各ドアの後ろには 1 つの賞品が隠れている．
2. 参加者は最初に 1 つのドアを選ぶ．
3. ホスト（モンティ・ホール）は、参加者が選んだドア以外の 2 つのドアのうち、1 つを開けて、その後ろに賞品がないことを示す．
4. 参加者には、最初に選んだドアを変更するか、そのままにするかの選択肢が与えられる。

モンティホール問題はこのステップ４の時点で、ドアを変更するかどうかで商品の当たる確率がどのように変化するかについての問題である．変更する場合と変更しない場合でシミュレーションを行い、当選する確率を確かめる．

実験結果

グラフから変更するのとしないので大きな差があることが分かる．変更したほうがより当選回数が多いことが分かる．変更をした場合の当選回数は、変更をしない場合の当選回数と比べ２倍大きくなっていた．

考察

モンティホール問題はいくつかのステップを踏んでいるのでわかりずらいところがあるように思われる．ドアの数が１０００個で９９８個のドアを開けて残った１つと、自分が選んだ１では当たる確率が違うの感覚としてもわかるだろう．

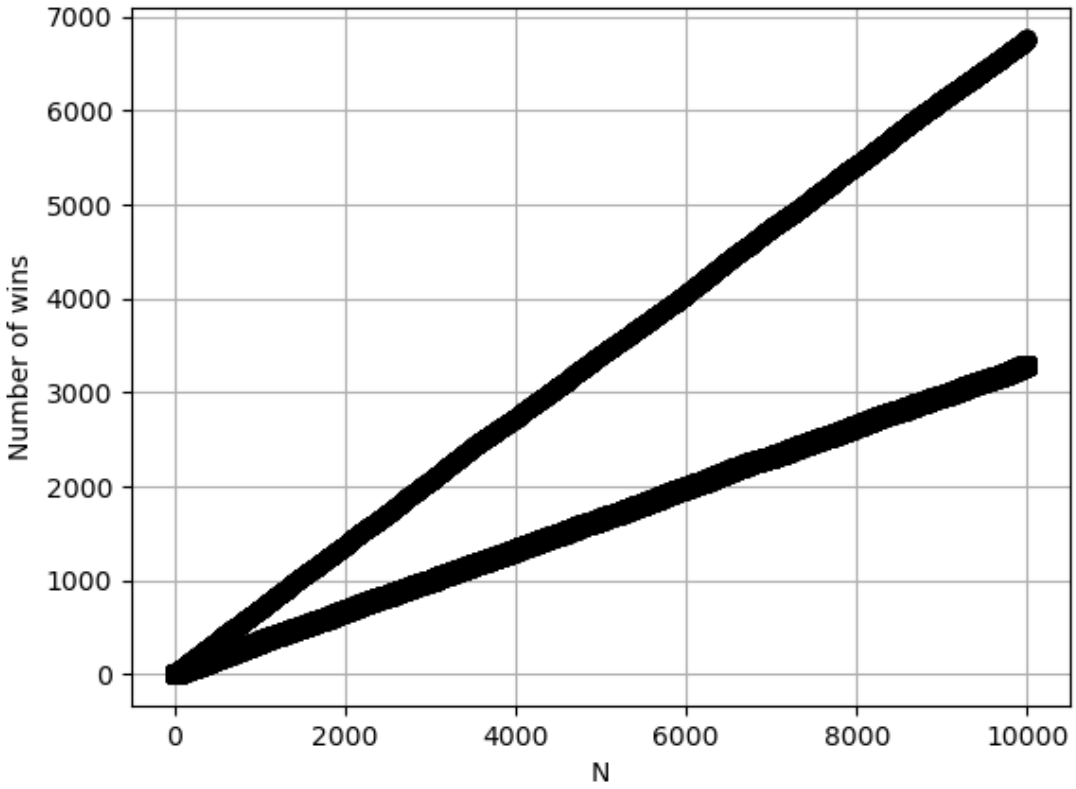

図 4　実験結果のグラフは、課題１－１で作った関数を参照

課題３－１

実験の概要

500 円で購入できるお菓子の特典でブロマイドカードが１枚もらえる．特典のカードは通常版と特別版の 2 種類で，それぞれの出る確率は $\frac{1}{2}$ とする．n 人のグループで特別版のカードを全員が入手するために必要な予算について考える．ここで，各消費者は特別版を引くまで購入を繰り返し，特別版を一度引いた時点で購入をやめることとする．また，お菓子は無限にあるとする．この手順を n 人全員が行ったとき，入手した全ブロマイドカードのうち特別版の比率がどうなるかをシミュレーションする．

実験の結果

あたり:0.501　　はずれ:0.499　　平均消費金額:998　　(5)

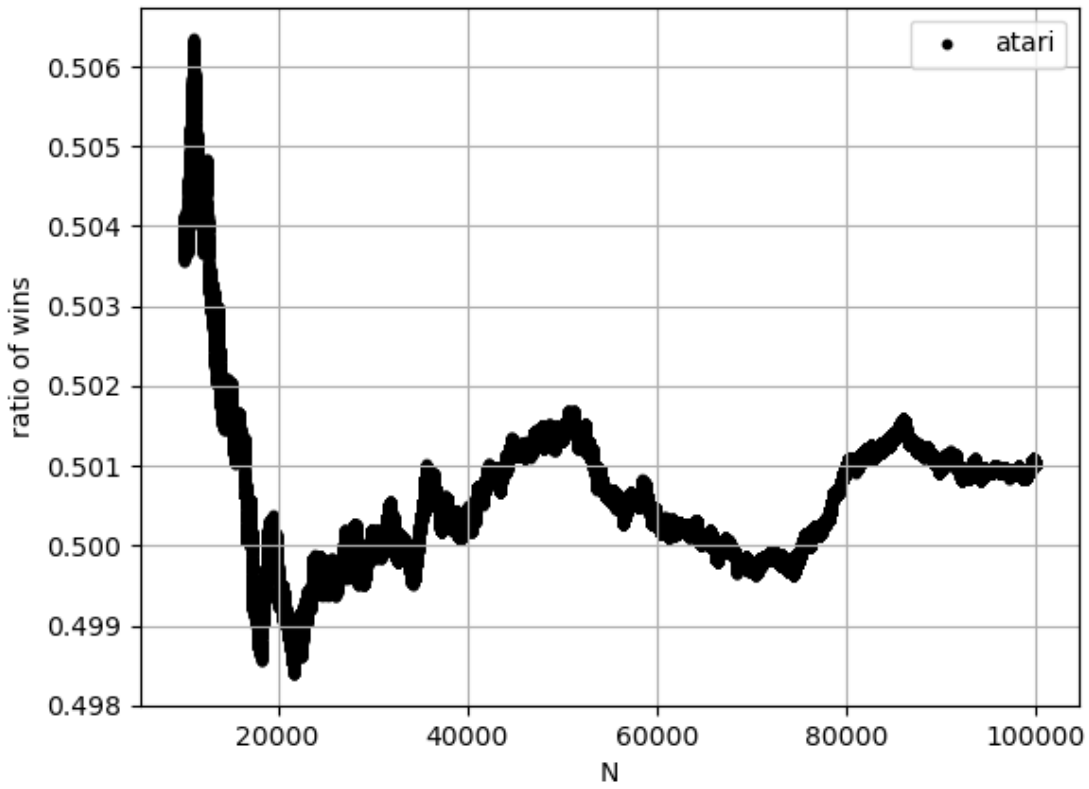

図５　横軸はグループの人数、縦軸は当たった確率を表す.

考察

図５より、特別ブロマイドカードの当選確率がおおよそ 0.5 に収束していることが分かる．これは、あたり、はずれが出る確率と一致している．課題の指定により有効数字３桁で行った結果、誤差が 0.01 であった．当たりが出る確率 $p = 0.5$ と設定しているため、この結果は妥当な結果であるといえる．理論的に、平均消費金額がどのようになるかを知るためには、期待値を計算する必要がある．１人の特別版ブロマイドカードが当たるまでの消費金額の期待値を E とすると以下の式により求めることができる.

$$
\begin{aligned}
E &= \sum_{i=1}^{\infty} 500\left(\frac{1}{2}\right)^n \\
&= 500 + 500 \cdot \frac{1}{2} + 500 \cdot \left(\frac{1}{2}\right)^2 + \ldots \\
&= 1000
\end{aligned}
$$

以上の式から理論的には、平均消費金額が１０００円になるはずである．今回の実験で得られた値は９９８円となっており２ずれているがこれは誤差の範囲出ある．この実験結果から、特別版ブロマイドカードを出すためには平均１０００円必要であることが分かった.
図（５)、図（６）ともに値にふり幅があるがこれは疑似乱数の偏りによる可能性がある.

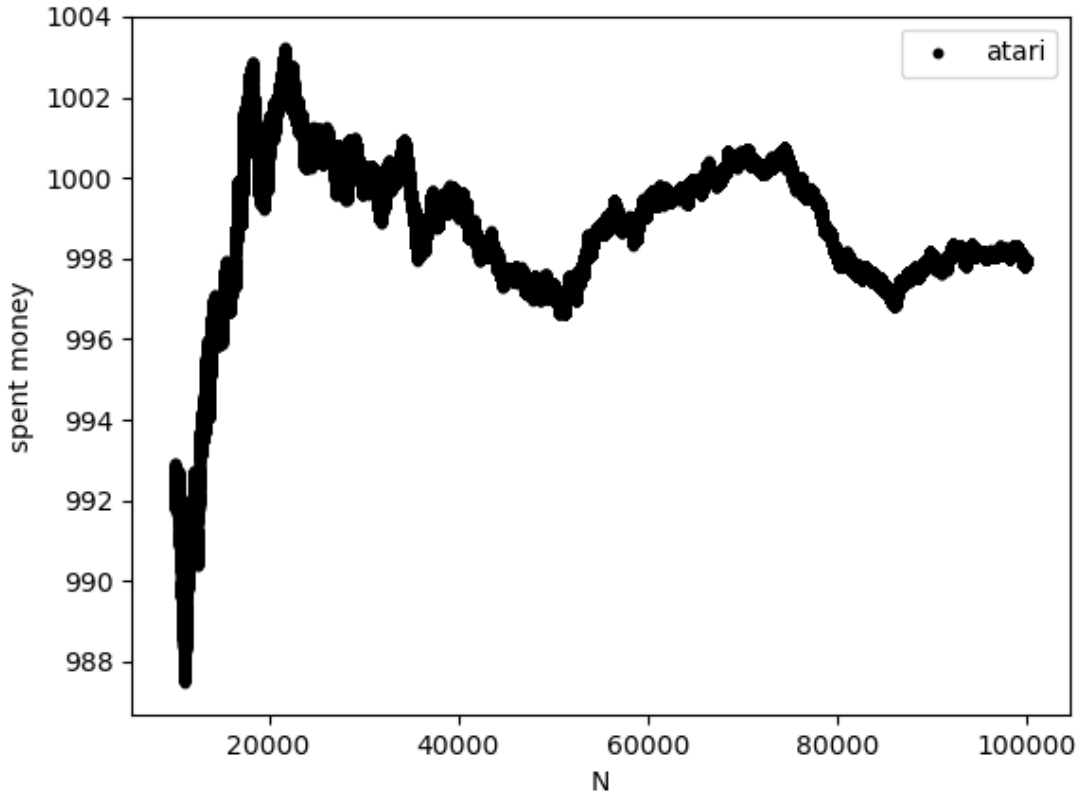

図 6　横軸はグループの人数、縦軸は平均消費金額を表す.

課題３－２

実験の概要

課題 3－1 の状況において，特別版と通常版の出る確率がそれぞれ 0.1 と 0.9 だった場合，特別版の比率と消費者の平均使用金額はどうなるかをシミュレーションする．

実験結果

当たる確率が0.1の時あたり:0.100はずれ:0.900平均消費金額:4992当たる確率が0.9の時あたり:0.900はずれ:0.100平均消費金額
(6)

考察

特別版が出る確率を 0.1, 0, 9 でシミュレーションを行うと大きな差がみられた．まず、特別版が当たる確率は設定した、0.1, 0.9 にちかずいた．これは、課題３－１デモ見られた傾向なので妥当な結果であるといえる．ま

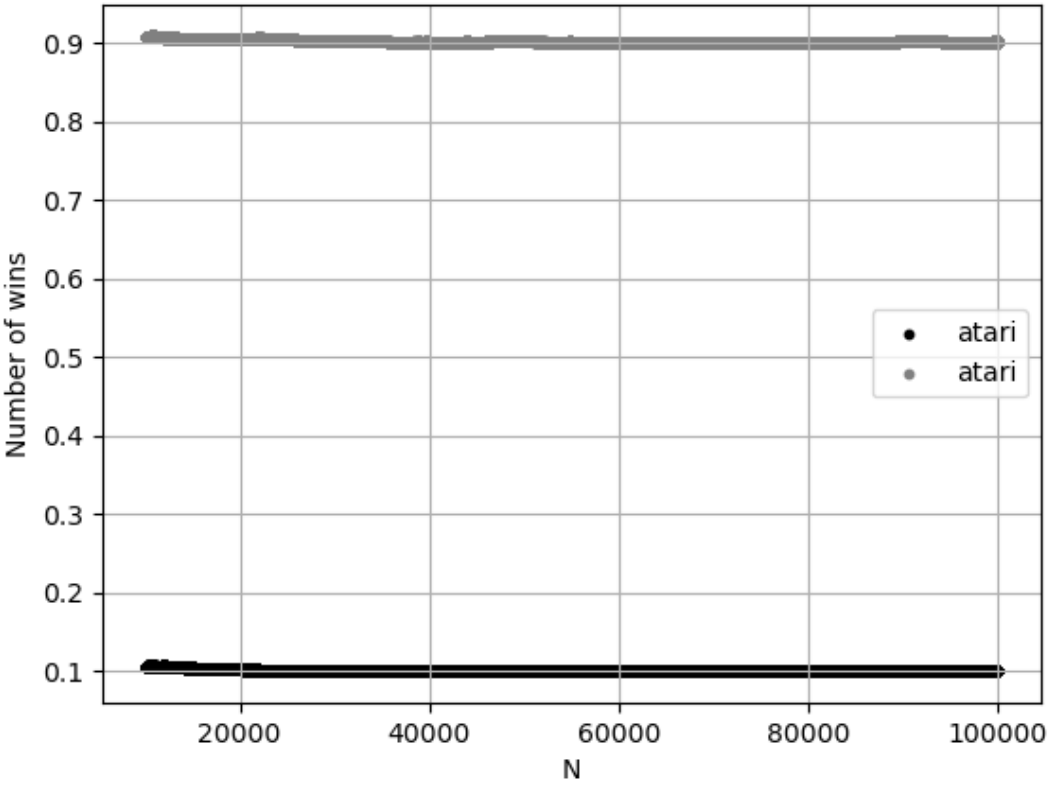

図７　横軸はグループの人数、縦軸は当たる比率を表す．黒の点は当たる確率が０．１灰色の点が当たる確率が０．９の時である．

た、平均消費金額はおよそ９倍ほどの差が出た．当たる確率が違うことによって、１０回中１回ほど当たる場合と１０回ほど９回当たる場合があるので、この結果は妥当である．また、平均消費金額と当たりはずれの比率の倍率がそれぞれ同じなのは今回正しくシミュレーションを行うことができたことを裏付けているだろう．

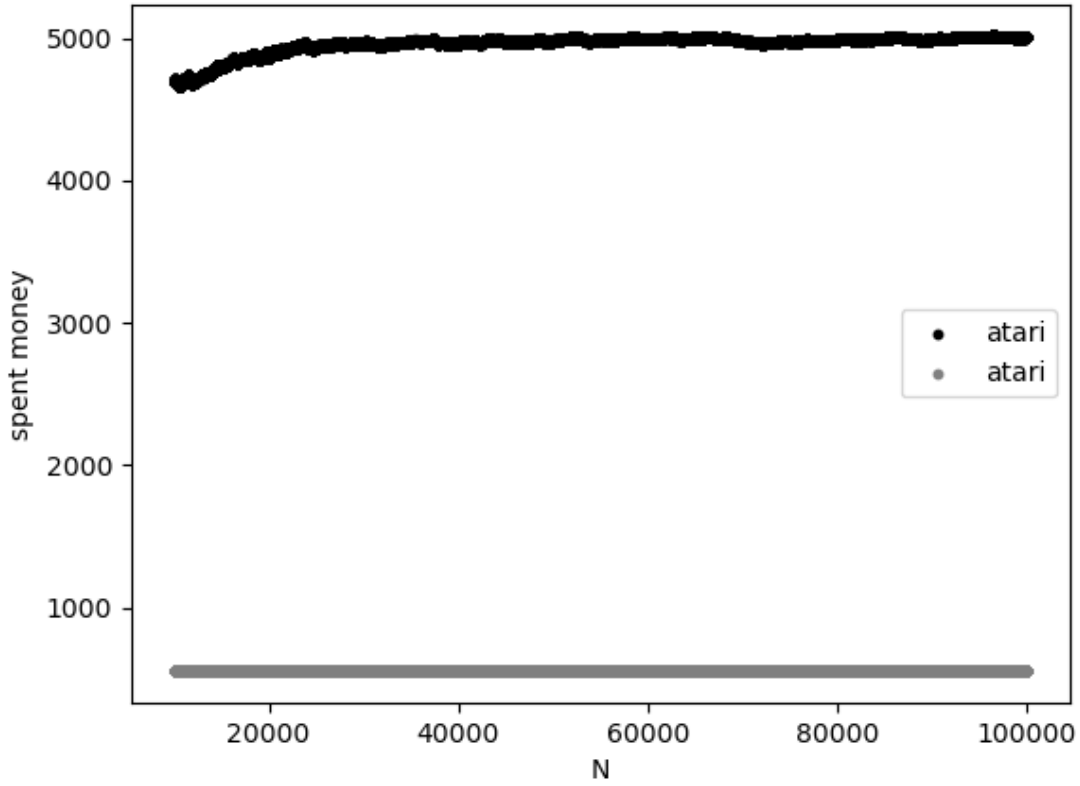

図８　横軸はグループの人数、縦軸は平均消費金額を表す．黒の点は当たる確率が０．１灰色の点が当たる確率が０．９の時である．

実験３－３

実験の概要

課題 3－1 の状況において，消費者 1 人あたり購入に使える予算が 2 千円だった場合，つまり各消費者が 4 個までしか購入することができないという制限がある場合を考える．この場合，特別版の比率と消費者の平均使用金額はどうなるかをシミュレーションせよ．

実験結果

あたり:0.501　　はずれ:0.499　　平均消費金額:936　　(7)

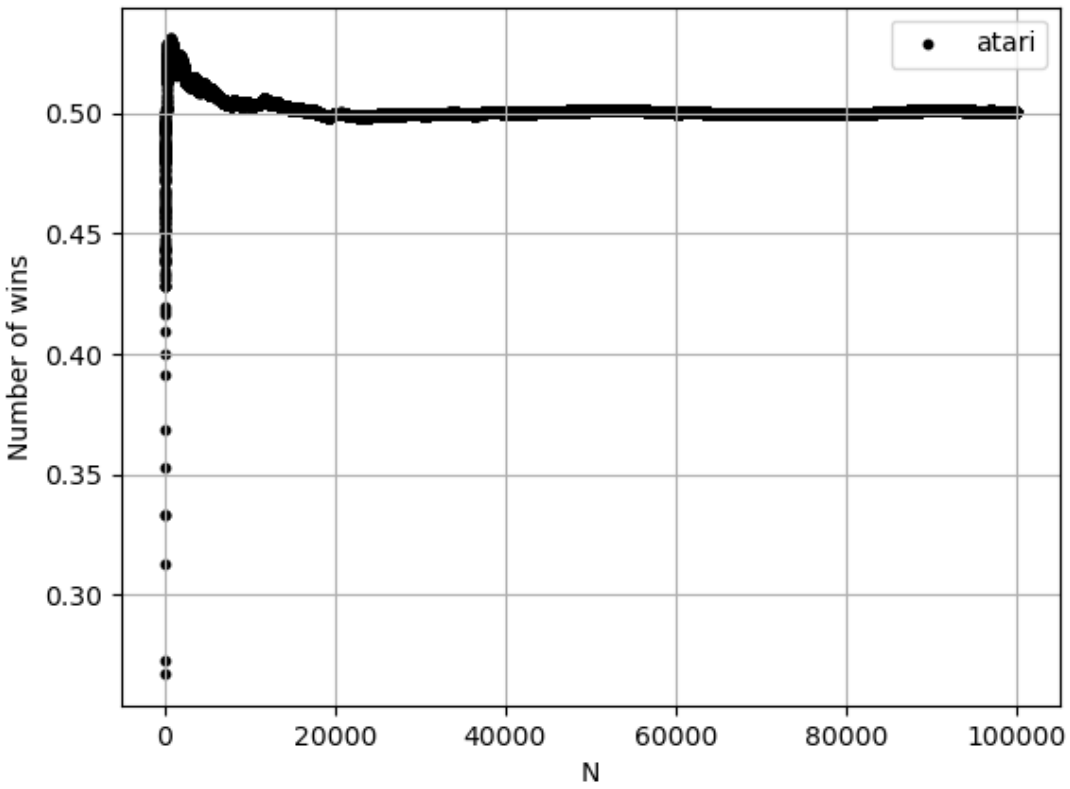

図９　横軸はグループの人数、縦軸は特別版ブロマイドカードが当たった比率を表す.

考察

結果から、消費する金額に予算を考慮しても当たりはずれの比率に変動は見られなかった．今回は消費者が４個までしか買えないという想定で行ったが、当たる確率は $\frac{1}{2}$ であり、４回連続ではずれる確率は $\frac{1}{16}$ であることから、当たりはずれの比率に対して、有意な差を生まなかったことが考えられる．課題３－１と比べて、比率に関しては有意な差がみられなかった一方で、平均消費金額はおよそ５０円ほど差がある．課題１では、１人の購入上限がないため、ずっとはずれを引き続け消費金額を挙げる固体が存在することが示唆される．それに対し、今回は購入の戸数に対して、制限をかけたため、２０００円以上一人が消費することがないことから、平均消費金額が下がったのではなかと思われる.

総合の考察

今回の実験では疑似乱数を用いて、確率的な実験を行った．実験を行う前に、自分で仮説を立てながら行っていたがおおよそ自分の仮設通りの結果となった．今回の実験のようにプログラミングを行って実際に確率の計算を行う手法は具体的な値を求めたいときに有効であると思う．例えば、課題３－３の場合、平均消費金額は課題３－１に比べて下がった．これは私が立てていた仮説と一致するものだが、どのくらい下がるのかなどを数学的に求めるのは困難である．そのため今回のようにシミュレーションを行うことが有効であると思った.

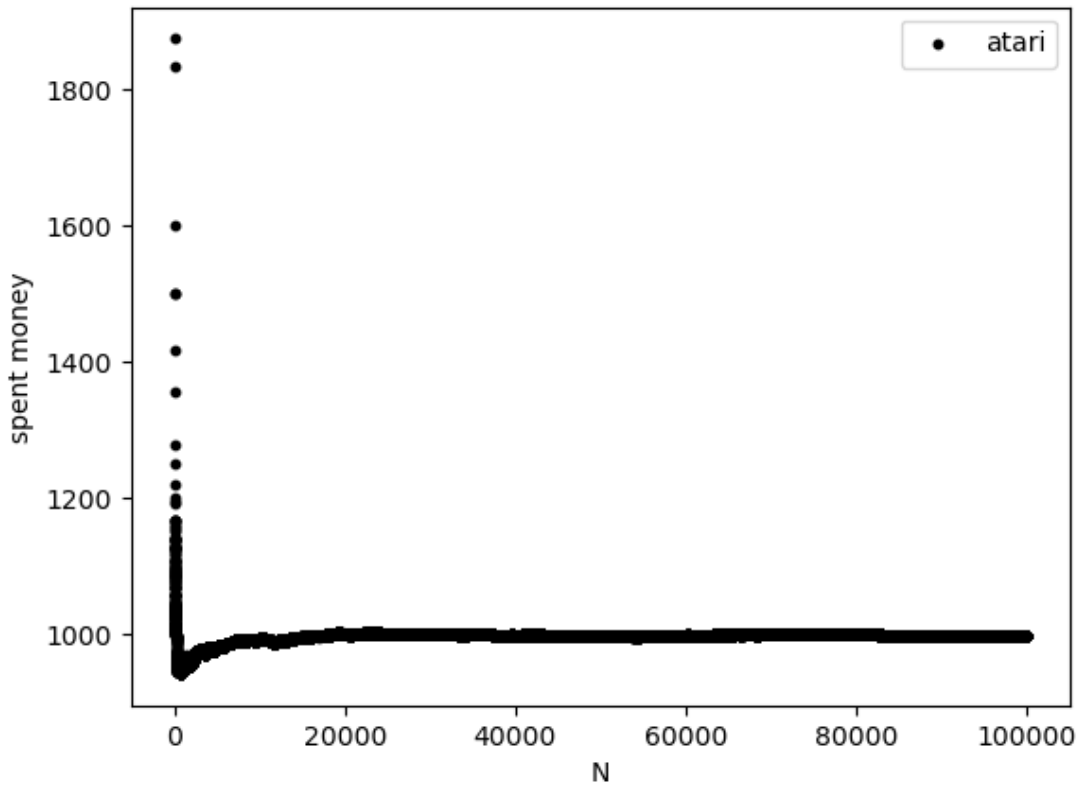

図 10　横軸はグループの人数、縦軸は平均消費金額を表す.

課題３－ A

実験の概要

課題 3 – 1 で行った 500 円のお菓子によるシミュレーションにおいて，消費者数を n = 10,000 人とし，お菓子 1 個につき企業の利益が 20 円だったとする．次の 2 つのそれぞれの場合において，特別版のブロマイドカードが出る確率を変化させたときの，消費者の平均使用金額及び企業の利益をシミュレーションせよ．なお，お菓子は無限にあるとする．

1. 各消費者は特別版を引くまで購入を繰り返し，特別版を一度引いた時点で購入をやめる．
2. 各消費者に購入上限額が設定されている場合．ここでは，n = 10, 000 人の消費者のうち，購入上限額が 500 円の消費者を 5,000 人，1,000 円の消費者を 3,000 人，2,000 円の消費者を 1,500 人，3,000 円の消費者を 500 人と設定せよ．

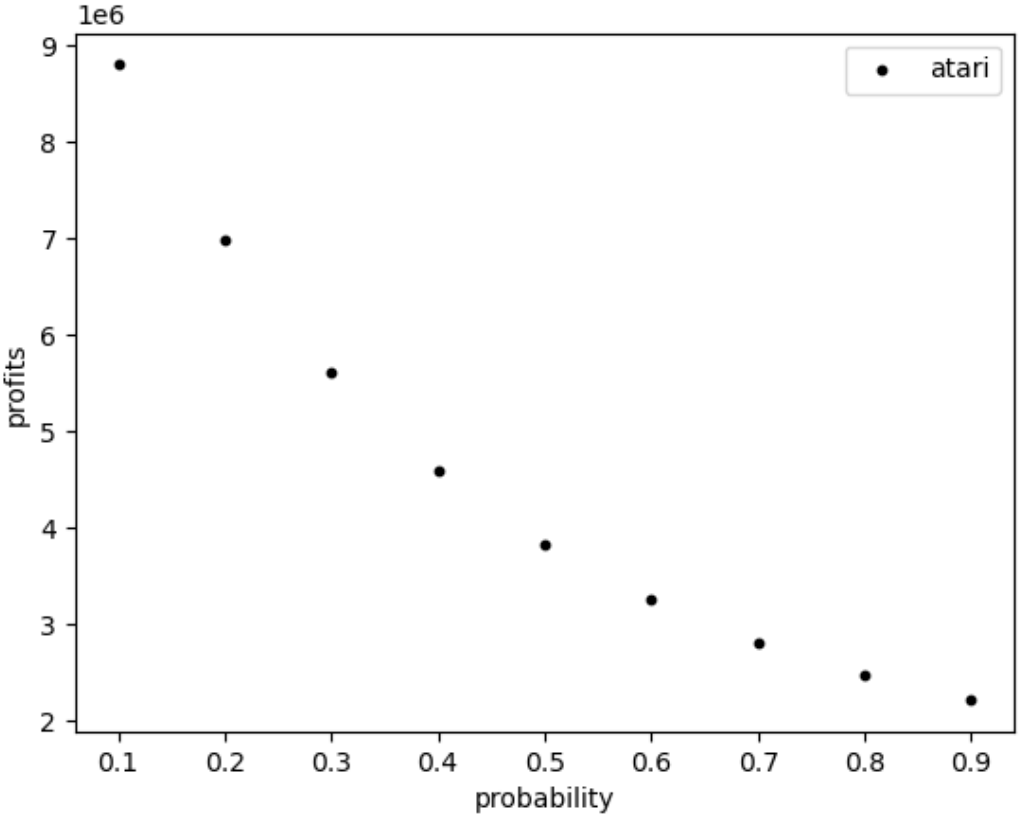

図 11　横軸は特別版ブロマイドが当たる確率、縦軸は消費者１人につき会社が得られる利益を表す.

実験結果

図９と図１０は同じ傾向を見せた．特別版ブロマイドが当たる確率をあげていくと消費者１人につき会社が得られる利益と、平均消費金額はともに下がっていった.

考察

特別版ブロマイドが当たる確率をあげると、当たるまでに購入する回数が減り、平均消費金額が下がることが予測される．また、購入あれた商品の数が減れば必然的に、利益も下がっていった．会社側としてより利益をあげるには、利益率と、値段、当たる確率の塩梅を考える必要がある.

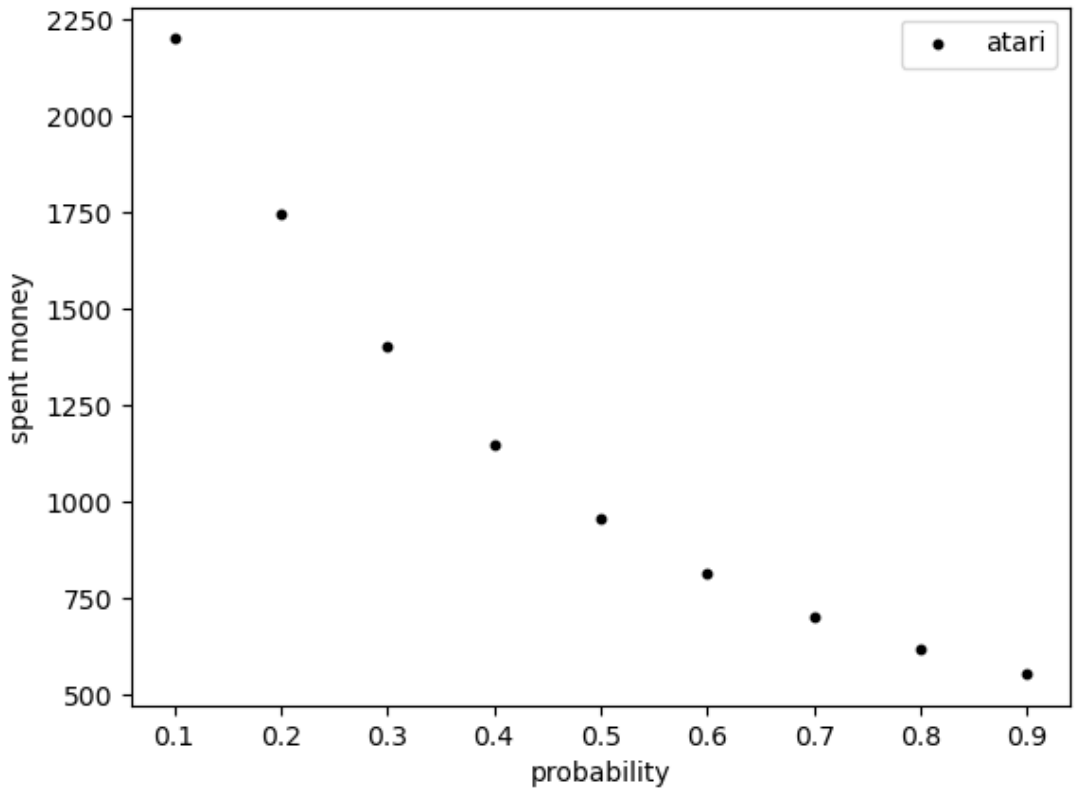

図 12　横軸は特別版ブロマイドが当たる確率、縦軸は平均消費金額を表す.

課題４－１

実験の概要

中心極限定理を確認するシミュレーションを作成し，実行する.

1. 平均 0，分散 1 の正規分布に従う乱数 $X_i(i = 1, ..., n)$ を独立に n 個生成し，n 個の平 $\overline{X}(n) = \frac{(\sum_{i=1}^{n} X_i)}{n}$ を求める関数 normal_rand を作成する
2. n 個の X の平均 $\overline{X}_n$ を求める試行を m 回行い，m 個の $\overline{X}(n)$ を求める.

実験結果

- m : 10
  - 平均値: 0.0000022
  - 分散: 0.0000000
- m : 100
  - 平均値: 0.0000219

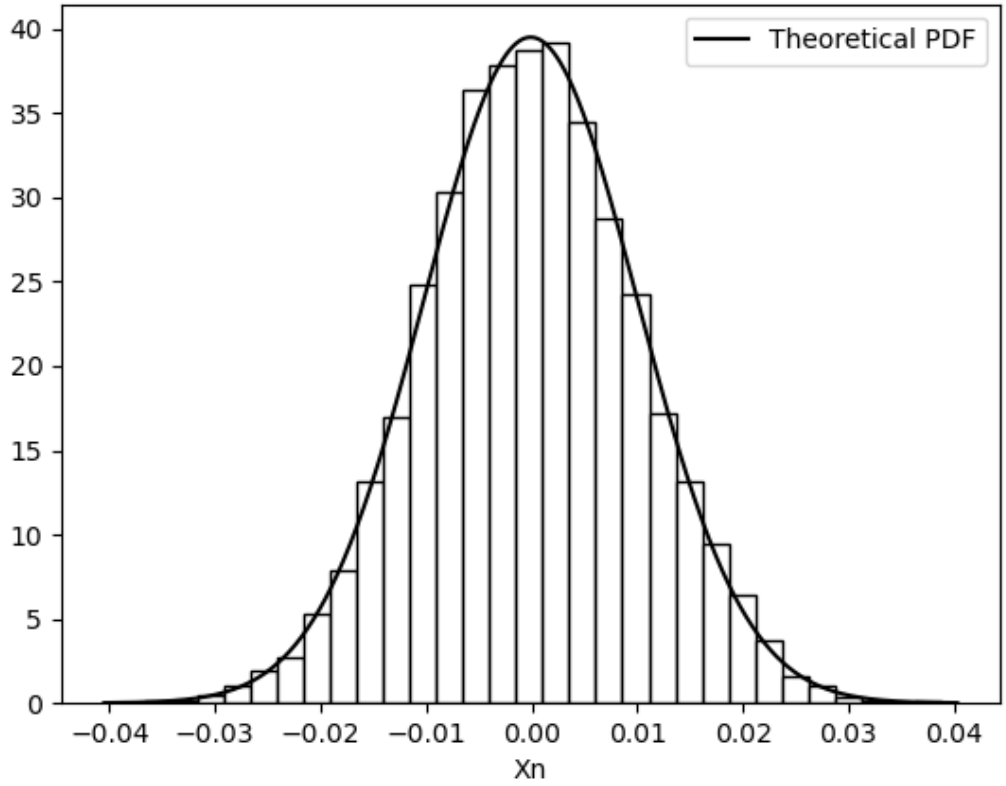

図 13　横軸は X_n の値、縦軸は標本数を表す.

  - 分散: 0.0000000
- m : 1000
  - 平均値: 0.000219
  - 分散: 0.0000004
- m : 10000
  - 平均値: 0.00219
  - 分散: −0.0000000

考察

図 (13) より、課題で生成した乱数から計算した $\overline{X}_n$ が正規分布に従っていることが分かる．理論的な正規分布の曲線にしたがっているのは中心極限定理から説明することができる．中心極限定理では、m を大きくすれば、標本の平均値は平均（元の集団の平均値)、分散（元の集団の分散を標本の数で割った値）の正規分布に従う．今回 $m = 10000$ でプロットしたが、正規分布に従ってプロットされていることから、$m = 10000$ は中心極限定理を示すのに十分大きな値であると言える．$\overline{X}(n)$ の平均値は理論とは異なった結果を示した．X の平均 $\overline{X}_n$ が m こ生成される平均は、0 になるはずであるが、今回は 0 とは違う値に近づいた．この値は m の値が大きくなるほど 0 に近づくはずであるが今回の結果では、逆に離れていった．これはシミュレーションの方法に

何か間違いがあるか、疑似乱数がうまく作動していないことが考えられる．平均値の値とは違い、分散の値は理論値に近い値となった．今回の実験では、分散の値は 0 になるはずであり、実験の結果でも 0 に非常に近い値となった．

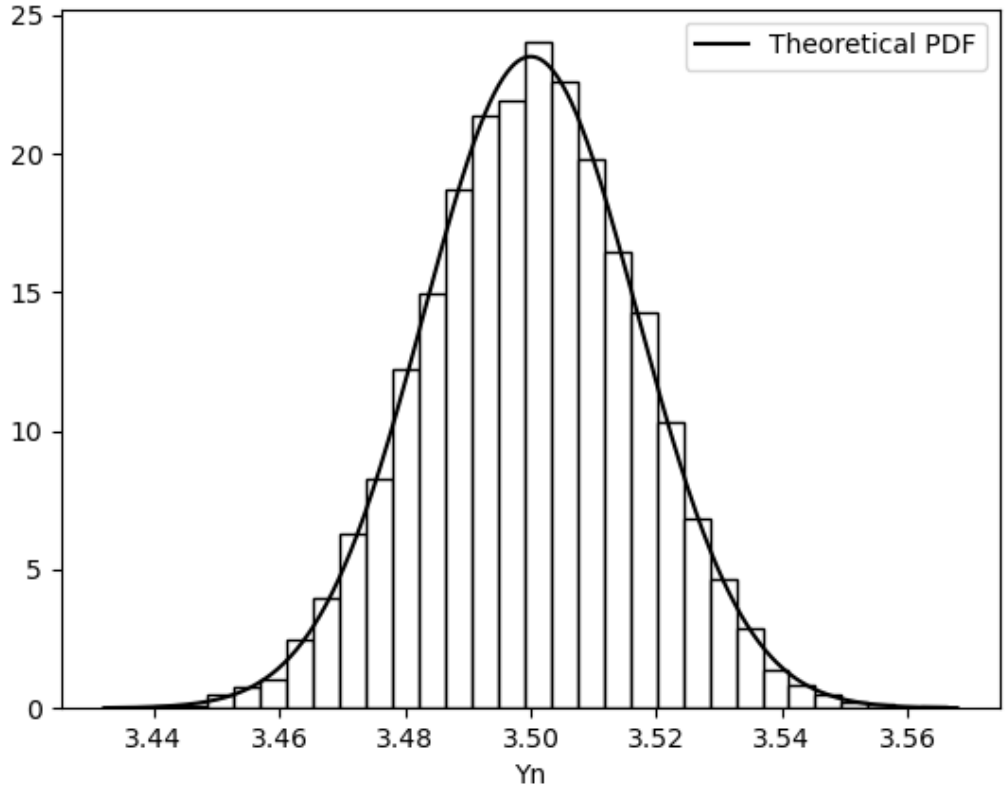

図 14　横軸は X_n の値、縦軸は標本数を表す.

課題４－２

実験の概要

第 1 回目のサイコロのシミュレーションを用いて中心極限定理を確認する.

1. サイコロを n 回独立に振り，n 個の出た目の平均 $\overline{Y}_n$ を求める.
2. n 個の出た目の平均 $\overline{Y}_n$ を求める試行を m 回行い，m 個の $\overline{Y}_n$ を求める.

実験結果

考察

今回の実験での平均と分散の理論値は、3.5 と 0 である．表 (1) より、平均値は m が大きくなるほど、3.5 に近づいていることが分かる．さらに、分散も m が大きくなるにつれ、理論値に近づいている．ただ $m = 1000$ の時と、$m = 10000$ の時ではあまり差がない．これは $m = 1000$ でも十分正確な値が出ることを示している．図 (14) から、課題で生成した乱数から計算した $\overline{Y}_n$ が正規分布に従っていることが分かる．中心極限定理は

m	平均値	分散
10	3.49096	0.0001866
100	3.49924	0.000303313
1000	3.50022	0.000282348
10000	3.49998	0.000284963

表 1　実験結果

$m \to \infty$ の時に、理論的な分布に従うことを示すが、$m = 10000$ でもある程度の近似であれば十分であることが分かった.

実験４－ A

実験の概要

母集団がベルヌーイ分布（二項分布）であることを想定して中心極限定理を確認する．ここでは，ベルヌーイ分布として第 1 回目の課題１－２ で作成したコイン投げを $p = 0.3$ としてシミュレーションを実行する．

1. コイン投げを n 回独立に行い，n 回のうち，表が出た割合 $\overline{P}_n$ を求める．
2. $\overline{P}_n$ を求める試行を m 回行い，m 個の $\overline{P}_n$ を求める．

実験結果

表 2　結果のまとめ

m	平均値	分散
10	0.29783	0.0000238
100	0.2998530	0.0000183
1000	0.3001655	0.0000212
10000	0.3000320	0.0000214

考察

図 (15) からわかるように今回の実験では理論に近い分布となった．P_n が 0.295 から 0.300 までのところに理論的な標本数より少し標本数が多くなったところがあるが、これは疑似乱数の生成の誤差によって収まる範囲である．また、平均値と分散の理論的な値に非常に近い結果となった．今回は表が当たる確率 $p = 0.3$ としていることから、理論的な平均値は 0.3 となる．表 2 から m の値が大きくなるほどに平均値が 0.3 に近づいている．分散の理論値は今回も 0 となるはずであり、すべての m の値に対して、理論値と近い値をとっている．ただ、m の増加とともに、理論値に近づく傾向はない．これは、P_n を生成する乱数が理想的な分布をなしていることから、$m = 10$ でも十分理論値に近いことが原因として考えられる．以上より母集団がベルヌーイ分布の場合でも中心極限定理が成立することが確認できた.

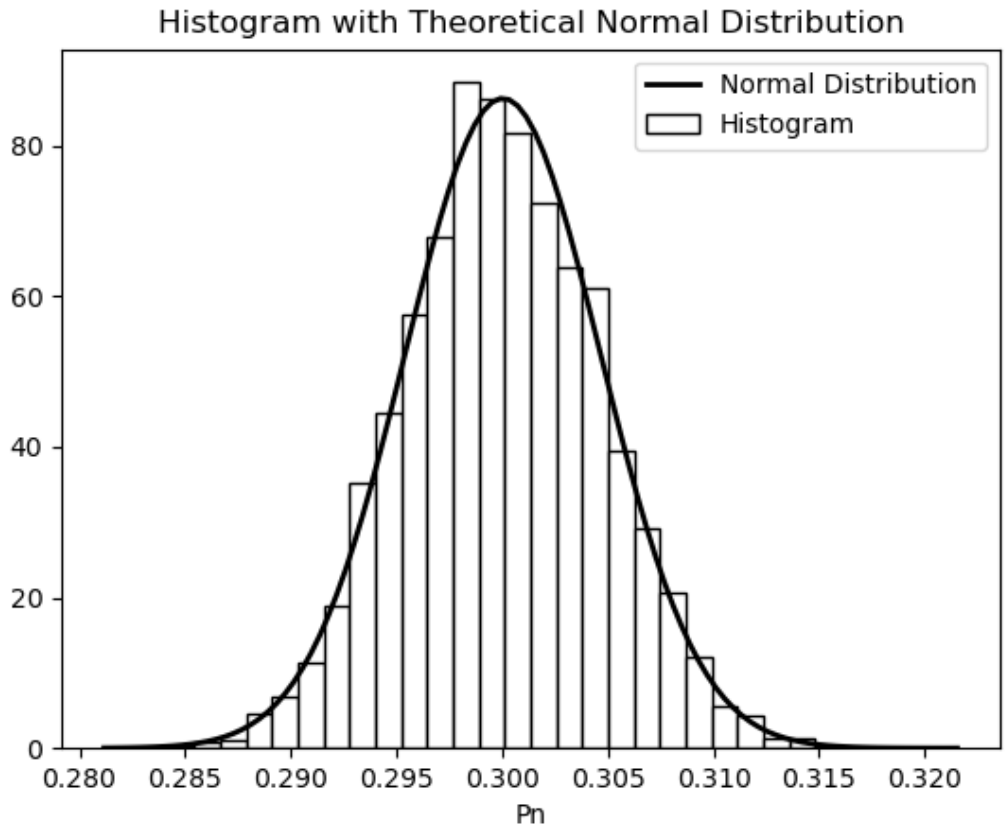

図 15　横軸は P_n の値、縦軸は標本数を表す.

付録

今回の課題に使用したソースコードを以下に示す.

```
  #include<iostream>
#include<random>
using namespace std;

//ランド関数を作る
void rnd_exp(double A[], int n){
    random_device rd;
    mt19937 gen(rd());
    uniform_real_distribution<double> distribution(0.0,1.0);
    for(int i = 0; i < n; i++){
        double r = distribution(gen);
        A[i] = r;
    }
}
int main(){
    int n = 1000;
    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
```

```
//ランド関数を作る
void rnd_exp(double A[], int n){
    random_device rd;
    mt19937 gen(rd());
    uniform_real_distribution<double> distribution(0.0,1.0);
    for(int i = 0; i < n; i++){
        double r = distribution(gen);
        A[i] = r;
    }
}
int GenDice(){
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> distribution(1, 9);
    int a = distribution(gen);
    if(a == 7) a = 2;
    if(a == 8) a = 4;
    if(a == 9) a = 6;
    return a;
}

int main(){
    ofstream of("DiceAverage1-A.csv");
    of << "N,Average" <<endl;
    int n = 1000;
    double A[n];
    rnd_exp(A,n);
    double sum = 0;

    for(int i = 0; i < n; i++){
        sum += GenDice();
        of << i+1 << "," << sum/(i+1) << endl;
    }
    return 0;
}
```

```
  #include<iostream>
#include<random>
#include<math.h>
#include<iomanip>
#include<fstream>
using namespace std;
void rnd_exp(double A[], double B[], int n){
    int rd = 22140003;
    mt19937 mt(rd);
    uniform_real_distribution<double> distribution(0.0,1.0);
    for(int i = 0; i < n; i++){
        double r = distribution(mt);
        double s = distribution(mt);
        A[i] = r;
        B[i] = s;
    }
}
```

```
int main(){
    ofstream of(”kadai_2_1.csv”);
    of << ”n,pi”<<endl;
    int n = 15000;
    double R1[n], R2[n];
    rnd_exp(R1,R2,n);
    int count = 0;
    for(int i = 0; i< n;i++){
        double r;
        r = pow(R1[i],2) + pow(R2[i],2);
        if(r < 1.000)count++;
        double a = i + 1.0;
        of << i+1 << ”,” << fixed << setprecision(4) << 4.0*(count/(a)) << endl;
    }
    double N = n + 0.0;
    double rat = count/N;
    cout << fixed << setprecision(4) << 4*rat << endl;
    return 0;
}
```

```
  #include<iostream>
#include<random>
#include<fstream>
using namespace std;
void rnd_exp(int A[], int B[], int n){
    int rd = 22140003;
    mt19937 mt(rd);
    uniform_int_distribution<int> distribution(1,3);
    for(int i = 0; i < n; i++){
        int r = distribution(mt);
        int s = distribution(mt);
        A[i] = r;
        B[i] = s;
    }
}
//bool MontyHallNOCHANGE(){}
int main(){
    ofstream of(”kadai_2_3.csv”);
    of << ”n,nochange,change” << endl;
    //1を正解のボックスとする.
    int n = 10000;
    int A[n], B[n];
    int change = 0;
    int Nochange = 0;
    rnd_exp(A,B,n);
    for(int j = 0;j<n;j++){
        if(A[j]==1)Nochange++;
        if(B[j]==2 || B[j] == 3)change++;
        of << j + 1 << ”,”<< Nochange<<”,”<< change << endl;
    }
    cout << ”変更なし” << Nochange/(n+0.0) << endl;
    cout << ”変更あり” << change/(n+0.0) << endl;
    return 0;
```

```

}
```

```
  #include<iostream>
#include<random>
#include<cmath>
#include<vector>
#include<iomanip>
#include<fstream>
using namespace std;

//make random numbers in an array


int main(){
    //set random numbers
    int n = 100000;
    //set probability
    double p = 0.5;
    int seed = 22140003;
    mt19937 mt(seed);
    uniform_real_distribution<double> distribution(0.0, 1.0);
    //set variable
    int count_loop = 0;
    int count_tousen = 0;
    //create a file for ratio
    ofstream of("kadai_3_1_ratio.csv");
    of << "n,tousen" << endl;

    //create a file for average spent
    ofstream file("kadai_3_1_average_expenditure.csv");
    file << "n,average" << endl;


    while(1){
        double prob = distribution(mt);
        if(prob > p){
            count_tousen++;
        }
        else if(prob < p){
        }
        count_loop++;
        if(10000 < count_tousen && count_tousen < 100000 ){
            double N = n;
            double tousen = count_tousen;
            double hazure = count_loop - count_tousen;
            of << count_tousen << "," << tousen/count_loop << endl;
            file << count_tousen << "," << (count_loop*500.0) / count_tousen << endl;
        }
        if(count_tousen == n)break;
    }
    double N = n;
    double tousen = count_tousen;
```

```
    double hazure = count_loop - count_tousen;
    cout << fixed << setprecision(3) << "あたり：" << tousen/count_loop << "はずれ：" << hazure/
        count_loop << endl;
    cout <<fixed << setprecision(0)<< "平均消費金額" << (count_loop*500) / N<< endl;
    return 0;
}
```

```
  #include<iostream>
#include<random>
#include<cmath>
#include<vector>
#include<iomanip>
#include<fstream>
using namespace std;

//make random numbers in an array


int main(){
    //set random numbers
    int n = 100000;
    //set probability
    double p = 0.9;
    int seed = 22140003;
    mt19937 mt(seed);
    uniform_real_distribution<double> distribution(0.0, 1.0);
    //set variable
    int count_loop = 0;
    int count_tousen = 0;
    double spnetmoney = 0;
    int stop_hazure = 0;
    //create a file for ratio
    ofstream of("kadai_3_2_ratio.csv");
    of << "n,tousen" << endl;

    //create a file for average spent
    ofstream file("kadai_3_2_average_expenditure.csv");
    file << "n,average" << endl;


    while(1){
        double prob = distribution(mt);
        spnetmoney += 500;
        count_loop++;
        if(prob > p){
            count_tousen++;
        }
        if(10000 < count_loop && count_loop < 100000 ){
            double N = n;
            double tousen = count_tousen;
            double hazure = count_loop - count_tousen;
            of << count_loop << "," << tousen/count_loop << endl;
            file << count_loop << "," << (count_loop*500.0) / count_tousen << endl;
```

```
47          }
48          if(count_loop - count_tousen == n){
49              stop_hazure = count_loop;
50          }
51          if(count_tousen == n)break;
52      }
53      double N = n;
54      double tousen = count_tousen;
55      double hazure = count_loop - count_tousen;
56      cout << "当たる確率が０．１の場合" << endl;
57      cout << fixed << setprecision(3) << "あたり：" << tousen/count_loop << "はずれ：" << hazure/
            count_loop << endl;
58      cout <<fixed << setprecision(0)<< "平均消費金額" << (count_loop*500) / N<< endl;
59      cout << "当たる確率が０．９の場合" << endl;
60      cout << fixed << setprecision(3) << "あたり：" << hazure/count_loop << "はずれ：" << tousen/
            count_loop << endl;
61      cout <<fixed << setprecision(0)<< "平均消費金額" << (stop_hazure*500) / N<< endl;
62
63      return 0;
64  }
65  #include<iostream>
66  #include<random>
67  #include<cmath>
68  #include<vector>
69  #include<iomanip>
70  #include<fstream>
71  using namespace std;
72
73
74  int main(){
75      //set random numbers
76      int n = 100000;
77      //set probability
78      double p = 0.5;
79      int seed = 22140003;
80      mt19937 mt(seed);
81      uniform_real_distribution<double> distribution(0.0, 1.0);
82      //set variable
83      int count_loop = 0;
84      int count_tousen = 0;
85      int count_people = 0;
86      int countfour = 0;
87      double spentmoney = 0;
88      //create a file for ratio
89      ofstream of("kadai_3_3_ratio.csv");
90      of << "n,tousen" << endl;
91
92      //create a file for average spent
93      ofstream file("kadai_3_3_average_expenditure.csv");
94      file << "n,average" << endl;
95
96
```

```
97      while(1){
98          spentmoney += 500.0;
99          countfour++;
100         count_loop++;
101         double prob = distribution(mt);
102         if(prob > p){
103             count_tousen++;
104             countfour = 0;
105             count_people++;
106             of << count_people << "," << count_tousen / double(count_loop) << endl;
107             file << count_people <<"," << spentmoney/ double(count_tousen) << endl;
108
109         }
110         else if(countfour == 4 && prob < p){
111             countfour = 0;
112             count_people++;
113             of << count_people << "," << count_tousen / double(count_loop) << endl;
114             file << count_people <<"," << spentmoney/ double(count_tousen) << endl;
115         }
116         if(count_people == n)break;
117     }
118     double N = n;
119     double tousen = count_tousen;
120     double hazure = count_loop - count_tousen;
121     cout << fixed << setprecision(3) << "あたり：" << tousen/count_loop << "はずれ：" << hazure/
            count_loop << endl;
122     cout <<fixed << setprecision(0)<< "平均消費金額" << spentmoney/ n<< endl;
123     return 0;
124 }
125 #include<iostream>
126 #include<random>
127 #include<cmath>
128 #include<vector>
129 #include<iomanip>
130 #include<fstream>
131 using namespace std;
132
133 int main(){
134
135     //set probability
136     int seed = 22140003;
137     mt19937 mt(seed);
138     uniform_real_distribution<double> distribution(0.0, 1.0);
139
140     //create csv files
141     ofstream of("kadai_3_A_average_expenditure.csv");
142     ofstream file("kadai_3_A_profits.csv");
143
144     //header
145     of << "prob,average" << endl;
146     file << "prob,profit" << endl;
147
```

```
148     for(double p = 0.1; p<= 0.9; p += 0.1){
149         //set variables
150         int count_people = 0;
151         double price = 500;
152         int n = 100000;
153         double prof = 20;
154         double reven = 0;
155         int tousen = 0;
156         int count = 0;
157         int count_loop = 0;
158         double spentmoney = 0;
159         //main loop
160         while(1){
161             spentmoney += 500;
162             count_loop++;
163             reven += prof;
164             if(0 <= count_people && count_people < 5000){
165                 if(p > distribution(mt)){
166                     tousen++;
167                     count_people++;
168                 }
169                 else{
170                     count_people++;
171                 }
172             }
173             else if (5000 <= count_people && count_people < 8000){
174                 count++;
175                 if(p > distribution(mt)){
176                     tousen++;
177                     count_people++;
178                     count = 0;
179                 }
180                 else if(count == 2){
181                     count_people++;
182                     count = 0;
183                 }
184             }
185             else if(8000 <= count_people && count_people < 9500){
186                 count++;
187                 if(p > distribution(mt)){
188                     tousen++;
189                     count_people++;
190                     count = 0;
191                 }
192                 else if(count == 4){
193                     count_people++;
194                     count = 0;
195                 }
196             }
197             else{
198                 count++;
199                 if(p > distribution(mt)){
```

```
                    tousen++;
                    count_people++;
                    count = 0;
                }
                else if(count == 6){
                    count_people++;
                    count = 0;
                }
            }
            if(count_people == n){
                break;
            }
        }

        of << p << "," << spentmoney / count_people << endl;
        file << fixed << setprecision(1) << p << "," << reven << endl;
        }
    return 0;
}
```

```
#include<iostream>
#include<random>
#include<fstream>
#include<iomanip>
#include<cmath>
using namespace std;

double normal_rand(int n){
    random_device rd;
    mt19937 mt(rd());
    //set mean and median
    double mean = 0.0;
    double var = 1.0;
    //define normal distribution
    normal_distribution<double> dist(mean,var);

    double sum = 0;
    //generate random number for n times
    for(int i =0; i < n; i ++){
        sum += dist(mt);
    }
    return sum / n;
}
int main(){
    //create a file
    ofstream of("kadai_4_1.csv");
    of << "Xn" << endl;
    //set variables
    int n = 10000;
    //
    for(int m = 10; m<= 10000; m *= 10){
        double sum = 0;
        double sum2 = 0;
```

```
34          for(int i = 0;i < m; i++){
35           double rnd = normal_rand(n);
36           sum += rnd;
37           sum2 += pow(rnd,2);
38           if(m == 10000){
39                of << rnd << endl;
40           }
41          }
42          cout << "m:" << m << endl;
43          cout << fixed << setprecision(7) << "平均値" << sum / m << endl;
44          cout << "分散" << (sum2 / m) - pow((sum/m),2) << endl;
45      }
46
47
48      return 0;
49  }
50  #include<iostream>
51  #include<random>
52  #include<fstream>
53  using namespace std;
54
55  double rnd_int(int n){
56      random_device rd;
57      mt19937 mt(rd());
58      //define normal distribution
59      uniform_int_distribution<int> dist(1,6);
60      double sum = 0;
61      //generate random number for n times
62      for(int i =0; i < n; i ++){
63          sum += dist(mt);
64      }
65      return sum / n;
66  }
67  int main(){
68      int n = 10000;
69      //file
70      ofstream of("kadai_4_2.csv");
71      of << "Yn" << endl;
72      for(int m = 10; m <= 10000; m *= 10){
73          double sum = 0;
74          double sum2 = 0;
75          for(int i = 0; i < m; i++){
76              double Y = rnd_int(n);
77              sum += Y;
78              sum2 += pow(Y,2);
79              if(m == 10000){
80                   of << Y << endl;
81              }
82          }
83          cout << "m:" << m << endl;
84          cout << "平均値" << sum / m << endl;
85          cout <<"分散" << (sum2/m) - pow(sum/m,2) << endl;
```

```
86      }
87      return 0;
88  }
89
90  #include<iostream>
91  #include<random>
92  #include<fstream>
93  #include<iomanip>
94  using namespace std;
95
96  double rnd_coin(int n){
97      random_device rd;
98      mt19937 mt(rd());
99      //define normal distribution
100     uniform_real_distribution<double> dist(0.0,1.0);
101     double sum = 0;
102     double p =0.3;
103     int head=0;
104     int loop=0;;
105     while(1){
106         loop++;
107         if(p > dist(mt)){
108             head++;
109         }
110         if(loop == n){
111             break;
112         }
113     }
114     return double(head) / n;
115 }
116 int main(){
117     int n = 10000;
118     //file
119     ofstream of("kadai_4_A.csv");
120     of << "Pn" << endl;
121     for(int m = 10; m <= 10000; m *= 10){
122         double sum = 0;
123         double sum2 = 0;
124         for(int i = 0; i < m; i++){
125             double P = rnd_coin(n);
126             sum += P;
127             sum2 += pow(P,2);
128             if(m == 10000){
129                 of << P << endl;
130             }
131         }
132         cout << "m:" << m << endl;
133         cout << "平均値" << sum / m << endl;
134         cout << fixed <<setprecision(7) << "分散" << (sum2/m) - pow(sum/m,2) << endl;
135     }
136     return 0;
137 }
```

(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．)