滋賀大学教育学部紀要
No.65, pp. 119-128, 2015

モンテカルロシミュレーションによる円周率πの計算

水　上　善　博

On Calculations of π by Monte Carlo Simulations

Yoshihiro MIZUKAMI

Abstract

Monte Carlo simulations are performed on personal computers to estimate π. RND function of VBA(Visual Basic for Application) in Microsoft Excel® is used for pseudorandom number generation. Mersenne Twister is used for pseudorandom number generation, too. Central limit theorem is adapted to estimate accuracy of Monte Carlo simulations. The limit of frequency of random number generation by RND function is discussed. Monte Carlo simulations are performed on various types of personal computers to check machine time. Personal computers, which are commonly used in class room, will be available for Monte Carlo simulation of π in reasonable machine time in lessons.

キーワード：円周率、モンテカルロシミュレーション、中心極限定理、計算時間

1．はじめに

高等学校の教科「情報」には、「モデル化とシミュレーション」という単元 があり、シミュレーションの例として、乱数を用いたモンテカルロ法がしばしば取り上げられている。乱数を利用したシミュレーションは、パーソナルコンピュータ (パソコン) に標準的にインストールされているアプリケーション、例えば、エクセル（Microsoft Excel®）のような表計算ソフトでも実行できるので、比較的手軽に授業で実施できる。しかし、パソコンの性能によってシミュレーションに必要な時間が異なり、古いパソコンでは、計算時間が長くなって授業で行うには適さないこともある。

また、シミュレーションを実行する対象の選択は重要である。円周率（3.14）は高等学校の生徒のみならず、小学校の高学年の児童や中学校の生徒にとっても比較的身近な定数であり、シミュレーションの結果得られる円周率πの値を正確なπの値（3.14159…）と比較することによって、どのくらいの規模のシミュレーションでどのくらいの精度の値が得られるかを実感することができることから、児童・生徒から興味をもってもらえる対象であると思われる。大学の演習授業においてエクセルを用いてπのモンテカルロシミュレーションを実践した試みでは[1)]、学生たちがシミュレーションに興味を抱いたとの報告がなされている。

本研究では、高等学校の教科「情報」や中学校の技術科において、円周率πの値をシミュレーションで計算する授業を実施する際に必要となる、精度や計算時間などの基礎的な事項を検証することを目的とする。具体的には、エクセルの乱数を利用して、円周率πの値を求めるモンテカルロシミュレーションのプログラムをエクセルの VBA（Visual Basic for Applications）で作成し、シミュレーションの回数とπの計算結果の精度について中心極限定理などの統計数学の観点から検証し、適切なシミュレーションの回数を探った。さらに、様々な性能のパソコンでシミュレーションを実施し、結果が得られるまでの計算時間を比較した。これらを基に、現在、教育現場で一般的に使用されているパソコンの性能でどのくらいのシ

ミュレーションが可能であるかを考察した。

2．方法

モンテカルロ法による円周率πの計算

円周率πの計算の歴史については、ベックマンの著作「πの歴史」[2]に詳しい。その中の第15章「モンテ・カルロ法」では、シミュレーションによってπを求める方法が紹介されている。いわゆる「ビュフォンの針」というもので、「長さLの針を、Lより大きい間隔dで平行な多数の直線が引かれている床の上に、でたらめに落下させる。このとき、針が直線のどれかと交わる確率をPとすると、円周率$\pi = 2L/Pd$と求まる」というものである。本で紹介されている実験結果では12000回のシミュレーションでπとして3.14程度の値が得られている。詳しくは「πの歴史」[2]を参照のこと。

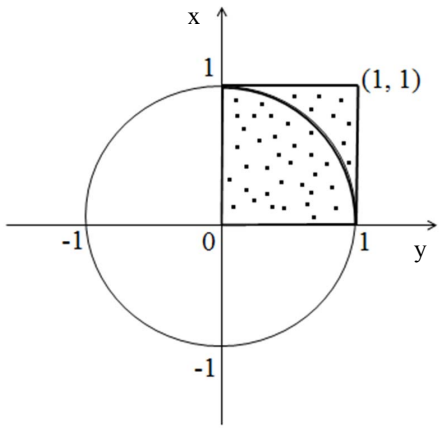

図1　円周率πのシミュレーションに用いられたモンテカルロ法

本研究では、コンピュータシミュレーションでより実行しやすい方法を用いる。図1に示すように、一辺が長さ1の正方形の内側に内接する四分の一の円を描く。正方形内でランダムな点をN個発生させて、そのうち四分の一の円の中に入る点の数Sを数える。ランダムな点を多数発生させれば、Nは正方形の面積1に対応し、Sは半径1の単位円の四分の一の面積$\pi/4$に対応する。よって、$S/N \cong (\pi/4)/1 = \pi/4$となり、

$$\pi \cong 4S/N \qquad (1)$$

とπの近似値が求まる。

VBAのプログラミング

具体的なシミュレーションはエクセルのVBAの乱数を用いたコンピュータシミュレーションで行う。エクセルのマクロでVBAのプログラムを作成したが、エクセルのマクロを実行するには、メニューに「開発タブ」を表示させる必要がある。Excel 2010以降のバージョンでは、「オプション　→　リボンのユーザー設定　→　開発　にチェックを入れる　→　OK」で、開発タブを表示できる。マクロでVBAのプログラム（例えば、paiという名前の関数）を作成するには、「開発　→　Visual Basic　→　挿入　→　標準モジュール」として、図2のようなプログラムを入力する。本来ならば、変数のデータ型の指定を厳密に行うべきであるが、中学生や高校生にわかりやすいように、巨大な整数（例えば、百万：10^6）を代入するnumという変数のみLong（長整数型）を指定している。また、このプログラムでは、繰り返し処理を実行する、For ～ Nextステートメントや、条件分岐を実行する、If 条件式 Thenステートメントなどの基本的な文法が用いられており、高校生のみならず、中学校3年生程度がプログラミングの基本を理解する教材としても適していると思われる。

プログラムの入力が終わったら、「ファイル　→　終了してMicrosoft Excelへ戻る」でマクロを閉じて、エクセルのシートに戻って計算を実行する。シミュレーションの実行は、例えば、100回ランダムな点を発生してπを計算する場合、セルに　= pai(100)　と入力する。あるいは、セルA1に100と入力して、セルB1に　= pai(A1)　と入力してもよい。すなわち、エクセルで標準に用意されている関数と同様に、関数paiが利用できるということである。もし、うまく実行できない場合は、プログラムの入力にミスがないかどうかを確認するとともに、マクロが有効になっているかどうかをチェックするとよい。

注意が必要なのは作業が終わってファイルを保存するときに、通常のエクセルファイルとしてではなく、「ファイルの種類　→　Excel マ

クロ有効ブック　→　保存」のようにマクロ有効ブックとして保存することである。
なお、本研究では、計算開始と計算終了の時刻を表示するマクロを作成してシミュレーションにかかる計算時間を算出した。

乱数について

エクセルの VBA では乱数（疑似乱数）を生成するのに Visual Basic の RND 関数を用いるが、RND 関数が生成する乱数には様々な問題があることが指摘されている[3)]。まず、単精度なので 10 進数表現で 7 ケタまでしか精度が保障されない。また、乱数を生成する周期が 1677 万（周期は 10 進法で 7 桁程度）なので、10^7 回～ 10^8 回以上の乱数を発生させると同じ数が出現して、乱数としての意味がなくなると思われる。すなわち 10^6 回程度の乱数を発生させるシミュレーションが限界であると予想される。また、65536 種類の系列の乱数しか得られないので、かなりの確率でまったく同じ系列の

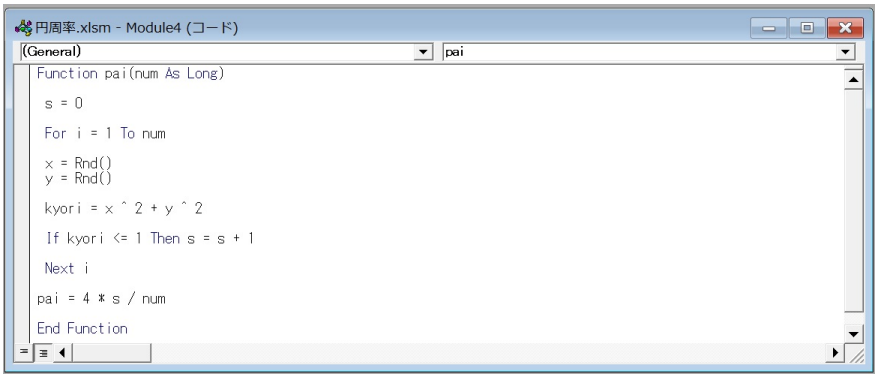

図 2　VBA で作成した、モンテカルロ法で円周率πを計算する関数のプログラムの一例

表 1　乱数の発生回数が異なるモンテカルロシミュレーションによる π の値。上は RND 関数で乱数を生成したシミュレーションの結果。下はメルセンヌ・ツイスタで乱数を生成したシミュレーションの結果。平均値、標準偏差、最大値、最小値は 100 回実行して求めた統計量。
誤差 ε は、$\varepsilon = (\mu - \pi) / \pi$　ただし、π はエクセルの関数 pi() で得られる円周率の値 3.1415926535897。

RND					
乱数の発生回数(N)	平均値(μ)	標準偏差(σ)	最大値	最小値	誤差(ε) %
10^2	3.1316	0.148947776	3.48	2.72	-0.318076043
10^3	3.12788	0.052139674	3.24	2.98	-0.436487320
10^4	3.140552	0.014351003	3.1704	3.1048	-0.033125033
10^5	3.1415944	0.005419461	3.15452	3.12572	0.000055590
10^6	3.1415902	0.001621922	3.14514	3.13792	-0.000078100
10^7	3.141582696	0.000190778	3.1419424	3.1412132	-0.000316960
10^8	3.141587	0.000014490	3.14162292	3.14155796	-0.000179959

メルセンヌ・ツイスタ					
乱数の発生回数(N)	平均値(μ)	標準偏差(σ)	最大値	最小値	誤差(ε) %
10^2	3.1636	0.148711264	3.44	2.84	0.700515593
10^3	3.14108	0.051851650	3.288	2.984	-0.016318271
10^4	3.141376	0.014010318	3.1688	3.0976	-0.006896298
10^5	3.1409432	0.004876898	3.154	3.1284	-0.020672750
10^6	3.14144408	0.001561359	3.145032	3.137992	-0.004729244
10^7	3.14162714	0.000512842	3.1427024	3.140452	0.001097737
10^8	3.141596535	0.000157429	3.14203112	3.14122144	0.000123555

乱数が生成される危険性がある。そこで、比較のため、現在、最も信頼性が高い疑似乱数を生成すると言われているメルセンヌ・ツイスタ法（周期は 10 進法で 6 千桁以上）によって生成した乱数によるモンテカルロシミュレーションもあわせて実施した。なお、エクセルでのメルセンヌ・ツイスタ法による乱数生成は、和田維作氏によって作成された、DLL[4] をエクセルにアドインして実行した。

3．結果と考察

シミュレーションの結果

表 1 に 10^2 から 10^8 まで 10 倍ずつ乱数の発生回数（N）を変えてモンテカルロシミュレーション実行して π の値を求めた結果を示す。表の上部分が RND 関数によって乱数を生成したシミュレーションの結果、表の下部分がメルセンヌ・ツイスタによって乱数を生成したシミュレーションの結果である。π の平均値、標準偏差、最大値および最小値は 100 回のシミュレーションから求めた統計量である。$\log_{10}N$ を横軸に、π の平均値を縦軸に描いたグラフを図 3 に示す。エクセルの pi 関数で求めた小数点以下 13 桁までの π（3.1415926535897：これを真値と呼ぶ）を合わせて示す。RND の乱数では発生回数が 10^2 と 10^3 のシミュレーションではあまり良い π の値は得られないが、10^4 からは真値に近づいて 10^5 から 10^8 では有効数字が小数点以下 4 桁まで真値に一致する。一方、メルセンヌ･ツイスタでは発生回数が 10^2 のシミュレーションではあまり良い π の値は得られないが、10^3 からは真値に近づいて 10^4 から 10^6 では有効数字が小数点以下 2 桁、10^7 では 3 桁、10^8 では 4 桁まで真値に一致する。

授業中に児童・生徒が 1 回だけシミュレーションをしたときに得られる π の値の目安として最大値、最小値を見ると、表 1 より、RND の場合は、乱数発生が 10^6 のシミュレーションの場合 3.14 ～ 3.15、10^7 で 3.141 ～ 3.142、10^8 で 3.1416 程度となっている。一方、メルセンヌ･ツイスタでは、10^6 で 3.14 ～ 3.15 程度、10^7 で 3.14 程度、10^8 でも 3.14 程度となっている。

文献 5) では、本研究と同様のシミュレーションによる π の計算結果が示されており（乱数は C++ の RAND 関数で生成）、11111 回の乱数発生で誤差 0.44％、111111 回の乱数発生で誤差 0.01% であったと報告されている。これを本研究の結果と比較する。表 1 で 10^3 の乱数を発生させたシミュレーションを 100 回実行して得られた π の平均値はトータルで 10^5 回乱数を発生したシミュレーション 1 回分に相当するので、表 1 の 10^3 の時の誤差（RND で 0.44%、メルセンヌ・ツイスタで 0.02%）が、文献 5) の 11111 回の乱数発生の誤差 0.44％と比較できる。また、

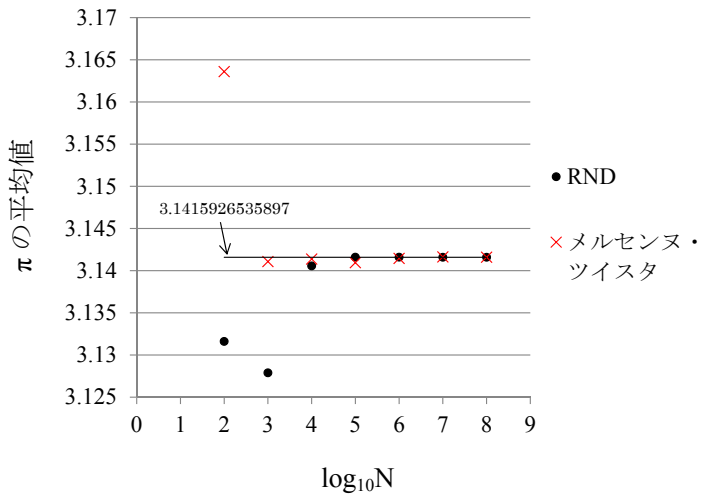

図 3　N 回の乱数発生のモンテカルロシミュレーションを 100 回実行して得られた π の平均値．黒丸は RND 関数の乱数を用いたシミュレーション、×はメルセンヌ・ツイスタによる乱数を用いたシミュレーション．

表1の10^4の時の誤差（RNDで0.03%、メルセンヌ・ツイスタで0.01%）が、文献5)の111111回の乱数発生の誤差0.01%と比較できる。

これらの結果を総合的に見ると、一見、RNDを用いたシミュレーションのほうがメルセンヌ・ツイスタのそれよりも精度がよいか、少なくとも同程度の精度があるように思われる。そこで、より詳しい計算精度のチェックを行うために、統計数学的な手法を用いた解析を行った。

中心極限定理

中心極限定理は「母集団の分布がどのようなものであっても（ただし、期待値や分散が定義できない分布を除く）、確率変数の和の確率分布の形は標本の大きさが大きいときは正規分布で近似できる」というものである[6)]。中心極限定理によると、標本の数がm倍になると標準偏差が$1/\sqrt{m}$倍になることがわかっている。

表1のシミュレーションは乱数の発生回数Nを10^2から10^8まで10倍ずつ増やしているので、標本の数は10倍ずつ増えている。よって、中心極限定理より、乱数の発生回数が10^3の標準偏差 は10^2の標準偏差σ_3の$1/\sqrt{10}$になる。同様に10^4の標準偏差σ_4は10^3の標準偏差σ_3の$1/\sqrt{10}$になる。これを、式で表すと、

$$\sigma_{n+1} = (1/\sqrt{10})\sigma_n \quad \text{ただし、n = 2, 3, ・・・, 8}$$

となる。両辺に$\log_{10}$を作用して、整理すると、

$$\log_{10}\sigma_{n+1} - \log_{10}\sigma_n = -0.5 \quad （2）$$

また、$n = \log_{10} N$なので、乱数の発生回数N（$=10^n$）と標準偏差σ_nのlog-logプロットを描くと、理論的には傾きが－0.5になると予想される。

図4にNとσ_nのlog-logプロットを示す。上図はRND関数で乱数を生成した結果である。$\log_{10}$N (= n) が2から6までのデータは、直線で近似でき、その傾きは－0.491となり、理論値の－0.5近い値となった。しかし、nが7と8では直線から外れ下落している。一方、下図

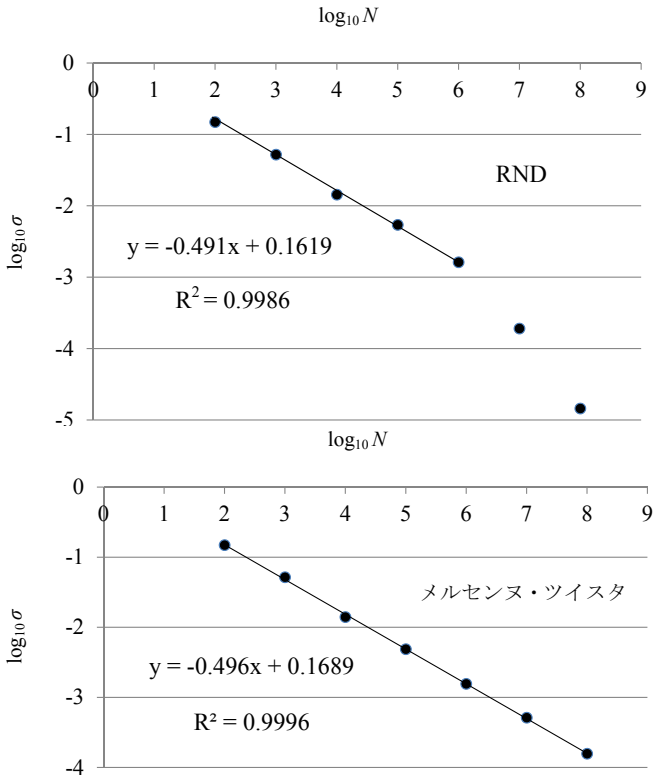

図4　乱数の生成回数(N)と標準偏差(σ)のlog-logプロット. 上はRND関数の結果. 下はメルセンヌ・ツイスタの結果.

はメルセンヌ・ツイスタで乱数を生成した結果である。N が 2 から 8 までのデータは直線で近似でき、その傾きは－ 0.496 となり、理論から予想される－ 0.5 に近い値となった。

RND の乱数発生回数が 10^7 および 10^8 のデータの下落は標準偏差が理論値よりも小さすぎるためと考えられる。π の有効数字は乱数発生の回数とほぼ一致するので、例えば乱数の発生回数が 10^7、10^8 のシミュレーションの有効数字は、それぞれ 7 桁、8 桁となるが、RND 関数の精度が単精度 10 進数でいえば 7 桁程度までとなっているため、10^7 回以上乱数を発生させると、同一の乱数が生じて、標準偏差が小さくなると考えられる。すなわち、RND 関数を用いて 10^7 回以上乱数を発生させるシミュレーションは信頼性に乏しいと言える。

一方で、メルセンヌ・ツイスタは少なくとも今回研究を行った乱数の発生回数が 10^8 までのシミュレーションでは、中心極限定理で予想されるゆらぎをもつバランスの良い乱数が生成されていると考えられる。

エクセルの RND 関数は手軽にモンテカルロシミュレーションができて便利であるが、乱数の発生回数としては 10^6 程度までが適当であり、10^7 回や 10^8 回の乱数発生のシミュレーションで良い結果が得られる場合は同一の乱数が発生しているための見かけ上のものであると考えられる。すなわち、エクセルの RND 関数で生成する乱数を用いたシミュレーションは最大でも 10^6 程度までの乱数発生にとどめるべきであるといえる。

図 5 に乱数の発生回数が 10^6 で π を求めるモンテカルロシミュレーションを 100 回実行した結果のヒストグラムを示す。上図が RND、下図がメルセンヌ・ツイスタの結果である。RND はあまりきれいな正規分布には見えないが、メルセンヌ・ツイスタでは比較的きれいな正規分布を示している。発生回数が 10^6 の場合でも、メルセンヌ・ツイスタの乱数を用いたシミュレーションの方が信頼性が高いと考えられる。

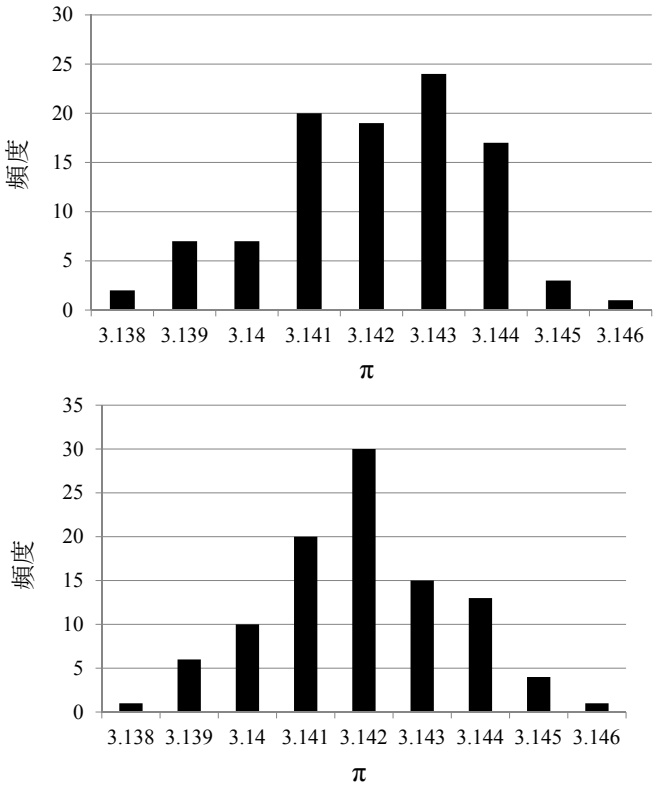

図 5　10^6 回の乱数発生のモンテカルロシミュレーションを 100 回実行したときの π の値のヒストグラム。上図は VB の RND 関数で乱数を生成した場合 . 下図はメルセンヌ・ツイスタで乱数を生成した場合 .

表2　様々な性能のパソコンにおいてエクセルのRND関数で10^6回乱数を発生させて円周率πを計算したモンテカルロシミュレーションの結果。（#1と#2は10回、#3～#14は100回のシミュレーションを実行した）

	Microsoft WindowsのOSのバージョン	Microsoft Excelのバージョン	パソコンタイプ	CPU	RAM	平均計算時間（秒）	πの平均値	標準偏差
#1	Windows 95	Excel 97	デスクトップ	Intel®Pentium® 133MHz	64MB	49.2	3.1415308	0.00180338
#2	Windows 95	Excel 97	ノート	Intel®Pentium® 120MHz	32MB	25.0	3.1414592	0.00180756
#3	Windows XP	Excel 2007	デスクトップ	Intel®Pentium®4 2.60GHz	1GB	1.66	3.1415766	0.00167359
#4	Windows XP	Excel 2007	デスクトップ	Intel®Pentium®4 3.00GHz	2GB	1.36	3.1415766	0.00167359
#5	Windows XP	Excel 2007	デスクトップ	Intel®Pentium®4 3.40GHz	3GB	1.21	3.1415766	0.00167359
#6	Windows Vista	Excel 2007	ノート	Intel®Core™ 2 2.53GHz	4GB	1.11	3.1415679	0.00168244
#7	Windows Vista	Excel 2010	ノート	Intel®Core™ 2 Duo 2.00GHz	2GB	0.88	3.1415766	0.00167359
#8	Windows 7	Excel 2007	デスクトップ	Intel®Core™ 2 Duo 2.66GHz	4GB	0.76	3.1415766	0.00167359
#9	Windows 7	Excel 2013	ノート	Intel®Core™ i5-3337 1.80GHz	4GB	0.70	3.1415766	0.00167359
#10	Windows 7	Excel 2013	デスクトップ	Intel®Core™ i7-960 3.20GHz	24GB	0.57	3.1415766	0.00167359
#11	Windows 7	Excel 2013	デスクトップ	Intel®Core™ i7-960 3.20GHz	12GB	0.56	3.1415766	0.00167359
#12	Windows 7	Excel 2013	デスクトップ	Intel®Core™ i7-3970X 3.50GHz	32GB	0.47	3.1415902	0.00162192
#13	Windows 8.1	Excel 2013	ノート	Intel®Core™ i3-3217U 1.80GHz	8GB	1.00	3.1415679	0.00168244
#14	Windows 8.1	Excel 2013	デスクトップ	Intel®Core™ i5-3570 3.40GHz	4GB	0.43	3.1415672	0.00169303

計算時間

様々な性能のパソコン上においてエクセルの RND 関数を用いて 10^6 回の乱数を発生させたモンテカルロシミュレーションで π の計算をした結果を表 2 に示す。Windows 95 のパソコンでは数十秒かかるが、Windows XP 以降のパソコンでは 2 秒以内で計算が終了することがわかる。また、RAM の大きさよりも CPU の性能の方が計算時間の短縮に重要であることがわかった。さらに、シミュレーションを 100 回実行して平均をとった π の値を見ると、#3, 4, 5, 7, 8, 9, 10, 11 で同じ値となっている。これは、初期化せずに乱数を発生させたため同一の系列の乱数が生成された結果である。#6 と #13 も同様である。なお、Windows XP 以降のパソコンの π の平均値は小数点以下 5 桁目を四捨五入すると、3.1416 になり、真値（3.1415926535897）の小数点以下 5 桁目を四捨五入した値と一致する。

教育情報化振興協会が 2013 年に実施した調査報告によると [7]、小学校 632、中学校 351 の合計 983 校で児童・生徒が使っているコンピュータの OS の割合は、Windows 98 が 0.3%、Windows XP が 31.4%、Windows Vista が 18.5%、Windows 7 が 50.1%、Windows 8 が 3.2% となっている。小、中学校の授業用のパソコンの OS は 99％以上が Windows XP 以降のバージョンとなっており、本研究の結果より 10^6 回の乱数発生のモンテカルロシミュレーションにかかる計算時間は 2 秒以内であるので、授業で十分に実行可能なものであると思われる。

学校の授業での活用

文献 7) によると、小、中学校におけるコンピュータ活用の目的としては、「インターネットを活用して「調べ学習」などをさせる」が 93.2% と最も多く、次いで、「文字を入力するなどの基本的な操作を行わせる」65.5%、「プレゼンテーション能力を高める」46.0%、「情報化社会の特徴（利点、危険、セキュリティ、個人情報保護など）を学習させる」44.2% となっており、シミュレーションやプログラミングにつながる「計算能力など児童・生徒の基礎学力を向上させる」は 12.7% にとどまっている。平成 27 年 6 月に閣議決定された「世界最先端 IT 国家創造宣言」[8] では、「初等・中等教育段階でのプログラミング、情報セキュリティ等の IT 教育の充実」および「高等教育段階での産業界と教育現場との連携の強化の推進」が述べられている。ICT（情報通信技術）は重要な教育課題となっており、小学校や中学校でもプログラミングの基礎について学ぶことが推奨されつつある。そこで、教育現場が現有している設備を利用して手軽にプログラミングの基礎を学べるような教材開発が望まれる。本研究で示したようなエクセルの VBA のプログラムを用いたシミュレーションは比較的簡単に実施できるので、小、中学校におけるプログラミングの教材としても適切であると思われる。

我々は人間同士のコミュニケーションの道具として日本語や英語などの自然言語を学ぶが、コンピュータとコミュニケーションをとるための人工言語でプログラミングを学ぶ意義のひとつは、プログラミング言語の文法やアルゴリズム (問題を解くための手順、ものの考え方) を学ぶことにある。プログラミング言語には、Basic、C 言語、Fortran、Java など数多く 存在するが、現在の小、中学校におけるパソコン環境で手軽に実施できるプログラミング教材としては、一つは、HTML 言語によるホームページ作成がある。HTML 言語は文法がシンプルで扱いやすく、プログラムもパソコンに標準装備されている「メモ帳」で書ける。また、作成したプログラム (すなわち作成したホームページ) は、やはりパソコンに標準装備されているインターネットのホームページを閲覧するブラウザを用いて確認できる。

もう一つのプログラミング教材としては、中学校の技術科で以前から取り上げられてきた Basic がある。Basic は数年前までは大学入試センター試験の数学の分野でも出題されていたが、新課程の数学の内容からコンピュータが削減されたことを受けて、2015 年実施の大学入試センター試験からは出題されなくなった。しかし、プログラミングを通じてアルゴリズムを学ぶことは重要なので、今後は、Basic が発展してビジュアル環境に適応した、Visual Basic でプログラミングを学ぶことが有意義であると

考えられる。Visual Basic は表計算ソフトのエクセルにおいてマクロ機能を利用したプログラミング言語 (VBA) としても用いられており、本研究で示した例のように、手軽に興味深いシミュレーションを実行できる。

大学の演習授業での実践

大学のコンピュータ実習におけるモンテカルロシミュレーションの実践として、滋賀大学教育学部で開講している教科「情報」の教員免許の取得に必要な科目の一つである「モデル化とシミュレーション」の授業において、エクセルによるπのモンテカルロシミュレーションを実施した。乱数の発生回数は 1 から 10 倍ずつ増やし、10、100、・・・、10^7、10^8 までとして、それぞれ、1 回ずつシミュレーションを実行した。乱数発生の回数が 10^8 回のシミュレーションでは、結果が出るまでに 44 秒程度かかった。結果が出るまでコンピュータが比較的長い時間計算する姿はあまり目にしたことがないようで、学生たちは興味を示していた。なお、誤って乱数の発生回数の入力を 1 桁間違えて 10^9 として、シミュレーションを実行してしまうと、計算結果が得られるまで、数十分程度コンピュータがかたまってしまうので、乱数の発生回数の入力には注意を促す必要がある。また、RND 関数で乱数を発生するシミュレーションの場合、乱数の発生回数が 10^7 回以上の結果は信頼性が低いことにも言及する必要がある。

謝辞

有益な助言をいただいた丸目諒氏と高田優介氏に感謝いたします。

文献

1) 「Excel を用いたシミュレーション演習授業」, 松永豊, Bulletin of Aichi Univ. of Education, 60 (Education Science), 187-190 (2011)
2) 「πの歴史」, ペートル・ベックマン著, 田尾陽一, 清水韶光訳, ちくま学芸文庫 (2006)
3) 「良い乱数・悪い乱数」 http://www001.upp.so-net.ne.jp/isaku/rand.html (2015 年 9 月 11 日閲覧)
4) 和田維作氏によって作成された、エクセルのためのダイナミック・リンク・ライブラリ (DLL) 一式（libZMT.zip） http://www001.upp.so-net.ne.jp/isaku/rand2.html (2015 年 9 月 11 日閲覧)
5) “Estimation of the value of π using Monte-Carlo Method and Related Study of Errors, Udayan Singh, Mathematics in School, 21-23 (2013)
6) 「統計学入門」, 東京大学教養学部統計学教室編, 東京大学出版会 (1991)
7) 「第 9 回教育用コンピュータ等に関するアンケート調査報告書」一般社団法人教育情報化振興協会（2014） http://www2.japet.or.jp/info/japet/report/ICTReport9.pdf (2015 年 9 月 11 日閲覧)
8) 「世界最先端 IT 国家創造宣言」の変更について（平成 27 年 6 月 30 日　閣議決定） https://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20150630/siryou1.pdf (2015 年 9 月 11 日閲覧)