乱数の仕組みを明かす

西山豊

〒533-8533　大阪市東淀川区大隅 2-2-8　大阪経済大学　経営情報学部

Tel: 06-6328-2431　E-Mail: nishiyama@osaka-ue.ac.jp

１．　身近な乱数

パソコンをログオンしたまま放置すると，しばらくしてスクリーン・セイバーが働く．これは画面の焼付けを防止するためのものである．スクリーン・セイバーの色や模様はさまざまであるが，共通していえることは模様や座標を決めているのに乱数が使われていることである．座標の値は乱数を発生させる組込み関数 RAND が使われているのだろう．関数の使い方は分かっていても，その仕組みを知る人は少ない．乱数は数学では初等整数論で議論されている素数と原始根に関係するが，今回は乱数のブラックボックスについて考えてみたい．

パソコンを使わなくても私たちは乱数や乱数の考え方を大いに利用している．たとえば，サッカーの試合で先攻か後攻かを決めるのにコインの裏表を使うし，双六や麻雀ではサイコロを使う．ビンゴゲームでは数字の記入したボールを使うし，宝くじでは回転板と矢を使う．これらに共通して言えることは何が出るかわからない，ということである．乱数は前の結果に影響されないので独立性といえるし，規則的でないので不規則的ともいえる．

乱数にはサイコロがよく使われる．サイコロは正六面体で１から６までの乱数を発生することができる．サイコロは穴が彫ってあるが，１の裏には６，２の裏には５，３の裏には４と合計が７になるようになっている．重心がど真ん中にくるように１の穴の大きさは大きい．これらは１から６の目が均等に出るようにするための工夫であろう．6 より大きい乱数はサイコロを２つ用いたり，他の正多面体（正 12 面体や正 20 面体）を用いたりすることもある．また，サイコロを振らなくてもよいように乱数ばかりが載っている本が売られたことも

ある．乱数表が商売になる時代があったのだ．

ここで，サイコロの数１から６について乱数を発生することを考えてみよう．いろいろな乱数が考えられるが，周期が６であることを限定しておこう．１から６までの数字をランダムに並べてみる．

1，2，3，4，5，6

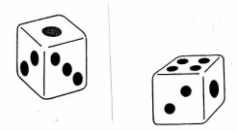
6，5，4，3，2，1

1，3，5，2，4，6

これだったら乱数でなさそうだ．乱数は独立していなければならない．前後の関連性があってはならない．ところが，数字が

2，6，4，5，1，3

というような並び方をしているなら乱数のように見える．このような乱数はどうして生成するのだろうか．今回のテーマは乱数の生成についてである．



２．素数と素数定理

素数とは自然数で１とそれ自身以外で割れない数のことである．たとえば２，３，５･･･などが素数である．

問１． １から 100 までの素数を求めよ．そして，その個数を求めよ．

自然数nが素数であるかどうかのチェックの方法は，nを２から$n-1$までの数字で割っていき，どの数字でも割り切れないときがそうであるが，よく考えれば$n-1$までのすべての数で割る必要はなく$\sqrt{n}$までの数で十分である．また，2 の倍数である偶数を飛ばすと効率よく求めることができる．nまでの素数を求める問題は Visual Basic の練習問題としてはよく使われている．

素数を求める方法は古代ギリシャ時代から「エラトステネスのふるい」として知られた方法がある（図１）．たとえば 30 までの素数を求める方法はつぎの

通りである．自然数２から 30 までの数字を並べておく．そして２は素数であるので，２の倍数である 4，6，8，･･･に斜線を入れていく．つぎに３は素数であるので，３の倍数である 6，9，12，･･･に斜線を入れていく．このようにして残った数が素数であることになる．この方法は原始的であるが，取りこぼしをなくすという長所がある．また，10^7 までの素数や個数を求めるといった大きい数を計算するときは威力を発揮する．

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13
~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23
~~24~~ 25 ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~

図１．エラトステネスのふるい

素数はいろいろなところに応用されている．私が以前に IBM につとめていたとき，高速フーリエ変換（FFT）には素数による素因数分解が応用されていることを知った．地震波の周波数分析をするとき，波を sin と cos のフーリエ級数に変換することがある．観測データが 1000 個あるなら，1000 回の計算が必要である．そこで，この近辺の数値で，1024 は 2^{10} と素因数分解され，普通だったら 1024 回の計算が必要だが，高速フーリエ変換だったら 10 回ですむというのだ．そのアルゴリズムの詳細は知らないが，素数は実際に役に立っているといえる．高速フーリエ変換は 1965 年にクーリーとテューキーによって発見された．

素数については昔からいろいろなことが研究されている．そのひとつに素数定理がある．自然数の中に素数がどのくらいの割合で含まれるかを述べる定理で，1792 年にガウスによって予想され，1896 年にアダマールとプーサンによって独立に証明された．具体的にはつぎの式で示される．１から x まで

x	実個数	$\frac{x}{\log x}$	$L_i(x)$
10	4	4	
100	25	22	
1000	168	145	
10^4	1229	1086	
10^5	9592	8686	9628
10^6	78498	72382	78627
10^7	664579	620421	664915

表１．素数の個数と近似式

の素数の個数 $\pi(x)$ について，

$$\pi(x) \approx \frac{x}{\log x} \quad （\log x は自然対数）$$

となる．公式にしたがって実際に値を計算してみると，$x<1000$ については比較的あっているが，$x>10^5$ になると，この式では十分でない．そこで，より精度の高い式は対数積分によって得られ，

$$Li(x) = \frac{x}{\log x} + \frac{1!x}{\log^2 x} + \cdots + \frac{(k-1)!x}{\log^k x} + O(\frac{x}{\log^{k+1} x})$$

となる．自然数 x までの素数の個数と素数の分布関数による近似値，さらに改良式による近似値を表１に示しておく．$Li(x)$ については，$x=10^5$ については第９項まで，$x=10^6$ については第 10 項まで，$x=10^7$ については第 12 項までの和とした．x が大きい時はこの公式がよく合っているように思える．

３．複素数の素数

素数の話のついでに複素数の素数について触れておこう．１とそれ自身以外で割ることができない数のことを素数といい，２，３，５･･･は素数であった．数を拡張して，複素数の世界ではどうであろうか．２，３，５のうち２と５は，

$$2=(1+i)(1-i)$$
$$5=(2+i)(2-i)$$

と素因数分解ができるので２や５は素数でなくなる．ちょっと変な感じであるが複素数ではそうだ．そして複素数の素因数分解というのがあって，$32+22i$ はつぎのようになる．

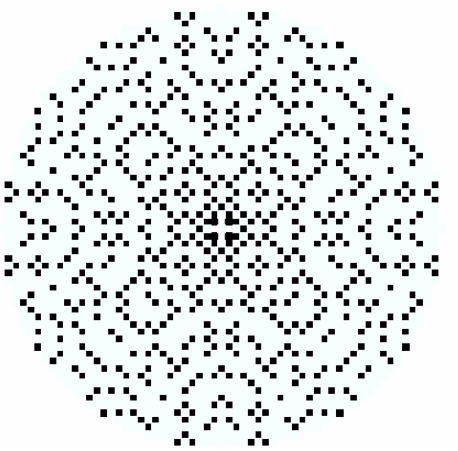


図２．複素数の素数

$$32+22i$$
$$=2(16+11i)=(1+i)(1-i)(2+3i)(5-2i)$$

ここで複素数 $1+i, 1-i, 2+3i, 5-2i$ は素数である．

実数の場合は基本となる数は１であった．複素数の場合は$1, -1, i, -i$の４つが基本となる．これらの数とそれ自身以外で割り切れない数を素数として，先に紹介したエラトステネスのふるいなどの手法でプログラムを作れば，複素数の素数である２次元の図を作ることができる（図２）．実際に作図してみるとテーブル・クロスのような模様ができ，数学っぽくてなかなかよい．BASIC プログラムの作り方は参考文献(1)を参照のこと．

４．素数と原始根

素数についての話はこれくらいにしておいて，冒頭で述べた擬似乱数について説明しよう．サイコロの目の１から６までが，ある方法で

2，6，4，5，1，3

のような並びで生成できれば，乱数として適している．では，その方法とはいかなるものであろうか．素数に対して原始根というのがある．たとえば，素数 7 に対する原始根は３である．この関係を使って擬似乱数を生成してみよう．

$3^1 = 3$

$3^2 = 9 \equiv 2 \bmod 7$

$3^3 = 2 \times 3 = 6$

$3^4 = 6 \times 3 = 18 \equiv 4 \bmod 7$

$3^5 = 4 \times 3 = 12 \equiv 5 \bmod 7$

$3^6 = 5 \times 3 = 15 \equiv 1 \bmod 7$

このようにして擬似乱数の列

3，2，6，4，5，1

が生成されたことになる．計算の仕方は原始根３を掛けていく．その答えが 7 以上になった場合は 7 の倍数を引いて剰余を答えとする．このように 6 回計算を繰り返していくと１に戻る．

これは，剰余を用いた合同法とも呼ばれている．この場合，最後の数が１になるのでそれを避けるために初期値を

p（素数）	r （原始根）
3	2
5	2，3
7	3，5
11	2，6，7，8
13	2，6，7，11

表２．素数と原始根

$3^2 \equiv 2 \bmod 7$のようにしておけば,

2，6，4，5，1，3

となる．素数 7 に対しては 1 から 6 までの乱数を生成することができる．言葉を変えれば 7 を法として$3^1, 3^2, 3^3, 3^4, 3^5, 3^6$は巡回群をなしているともいえる．素数 p と互いに素な数 r については，$r^{p-1} \equiv 1 \bmod p$ の関係が成り立ち，これを「フェルマーの小定理」という．定理に「小」という字がついているのは，有名なフェルマーの最終定理と区別するためである．この合同式で$r^i \equiv 1 \bmod p$となるのは$i < p-1$においても存在する．$i = p-1$のときのみ成立するとき r は原始根となる．素数 7 に対する原始根は 3 以外に 5 がある．素数 3，5，7，11，13 の原始根を示しておく（表２）．

問２． 素数 11 の原始根が 6 であることを利用して 1 から 10 までの擬似乱数を生成せよ．

５．コンピュータの乱数

以上は，乱数生成の原理についての説明であるが，実際のコンピュータの中ではどのようになっているのだろうか．パソコンには RAND などの組込み関数があり，$(0,1)$区間の乱数が生成される．まず整数の乱数が計算され，最大数で割って$(0,1)$区間の実数の乱数が計算されている．現在の Visual BASIC の組込み関数 RAND にどのようなアルゴリズムが使われているか，私は詳しく知らないが，擬似乱数生成のよく知られているアルゴリズムは次の通りである．どちらもつぎの合同式の考え方が乱数に取り入れられている．

$X_i = (aX_{i-1} + c) \mod M$　　（a, cは定数）

X_{i-1}にある値を入れておき，それに a を掛ける．その値が M を超えるなら M で割ってその余りを求めて X_i とする．このような漸化式でつぎつぎと乱数が計算される．

プログラムの中であつかう変数には実数タイプ変数と整数タイプ変数がある．乱数は整数タイプ変数で４バイト（＝32 ビット）が使われる．32 ビットで２進数の値が表されるが，先頭の１ビットは符号ビットであり，０が正，１が

負となっている．32 ビットから符号の 1 ビットを除いた 31 ビットで実際の数値が表される．乱数は正の数で考えるから，31 ビットで表せる最大数は $2^{31}-1=21474836$ である．1 から $2^{31}-1$ の区間で先に説明した合同式で乱数を生成し，最大数で割れば $(0,1)$ 区間の乱数が生成できたことになる．

そこで，問題になるのは素数 p と原始根 r をどのように選ぶかということになる．整数タイプ変数の最大数は $2^{31}-1$ であり，2^p-1 の形であらわせる素数はメルセンヌ数として知られている．メルセンヌ数は $p<11400$ の範囲には 23 個が存在して，

$$p=2,3,5,7,13,17,19,31,61,\cdots$$

などである．p はすべての素数でいえるとは限らない．

31 ビットで表せる最大数 $2^{31}-1$ は幸運にもメルセンヌ数であり素数である．そこで $2^{31}-1$ に対する原始根を求めればよいことになる．表 2 でも示したように，原始根の数は 1 つではない．素数が小さいなら原始根も容易に求められるが $2^{31}-1$ のように 10 進数で 10 桁のような大きな数値になると，そう簡単に求めることができずコンピュータの力を借りなければならない．

S.K.パークと K.W.ミラーは原始根 $a=7^5$，素数 $M=2^{31}-1$ として，

$$X_i=7^5 X_{i-1} \mod (2^{31}-1)$$

を用いている(1988)．参考文献(2)にもあるとおり，素数と原始根のよい組合せを見つけるのは難しいとある．$2^{31}-1=2147483647$ は素数であるが，原始根 $7^5=16807$ は素数でない．合同式において $M=2^{31}-1$ と $a=7^5$ は互いに素であるので，前述したフェルマーの小定理が成り立つ．つまり，

$$(7^5)^{2^{31}-2} \equiv 1 \mod (2^{31}-1)$$

となる．

これは計算するまでもなく成立するが，実際にパソコンで検査してみたところ，$2^{31}-2$ 回計算を繰り返してはじめて 1 に戻ることが確認できた．$2^{31}-2$ までの途中で 1 に戻ることはないので，すべての数値を通過するという極めて優れた原始根である．

以上の方法は周期が $2^{31}-2$ となり申し分ないのだが，割り算である $\mod (2^{31}-1)$ の計算に時間がかかるのが難点である．そこでこれを改良した，

よく知られる IBM 社の RANDU というサブルーチンがある（1970）. RANDU の合同式は次式で与えられる.

$X_i = 65539X_{i-1} \mod 2^{31}$

$M = 2^{31}(= 2147483648)$は素数でないのでフェルマーの小定理は成り立たないが，$a = 2^{16} + 3(= 65539)$は素数であり，Mとaは互いに素であるので，周期の大きい乱数を生成することが期待できる．パソコンで確認したところ

$(2^{16} + 3)^{2^{29}} \equiv 1 \mod 2^{31}$

となった．周期は最大数2^{31}の 4 分の 1 で比較的長い周期である.

数学的にはMを素数にしたほうがよいが，RANDU の場合は素数にしなかった理由は，割り算の除数を2^{31}としたほうが，計算の効率がよいためであろう．コンピュータの内部では2^{31}で割るという計算は 2 進数を 31 ビット右にずらすという機械命令で行われている．これにはシフト・レジスターが使われている．詳細は別の機会にゆずるとして，この方法は演算速度が非常に速いという長所がある．RANDU は 1970 年代の話しである。コンピュータの技術が発展すると演算速度が速くなり、このような工夫もいらなくなり、RANDU が生成する乱数の精度の悪さから、現在では別の方法が用いられている。

乱数は素数と原始根で生成される．この仕組みを知っておくと数学の素晴らしさがわかるのではないだろうか.

参考文献

(1) 何森仁『パソコンで楽しむ高校数学』サイエンス社，1991，47-51

(2) S. K. Park and K. W. Miller, Random number generators: Good ones are hard to find, Communication of the ACM, Vol.31, 1192-1201, 1988.