あなたの使っている乱数、大丈夫？

-危ない標準乱数と、メルセンヌ・ツイスター開発秘話-

松本　眞

広島大学理学研究科数学専攻

2014/11/18, 第50回市村学術賞記念　先端技術講演会

m-mat “at mark と呼ばれるもの” math.sci.hiroshima-u.ac.jp

昔から、乱数発生は（ばくちに）必要だ。

- サイコロ：$\{1,2,3,4,5,6\}$の中からでたらめに
  数を発生させる装置
- 「一様かつ独立に次々とでたらめな数を発生させる方法」
  を**乱数発生法**という。
- コイン投げ、は表と裏で$\{0,1\}$の要素を発生
  ルーレットは$\{0,1,\ldots,36\}$の要素を発生
- くじ引きも、乱数発生法と言える。

Q. シミュレーションに乱数はなぜ必要？

A. 世界は、ばくちだから。

確率的現象のシミュレーションは、すごろくそのもの。

例１：連鎖核分裂

- 一つ一つの原子核が、一定の確率で分裂
- 中性子が放出される（放出される方向も確率的）
- 中性子がぶつかった核は、また分裂

一つ一つの確率現象をシミュレートするごとに
乱数が消費される。

核分裂 ：たとえば、各原子につきサイコロを10回振って、
全部１の目なら分裂、それ以外は非分裂。

中性子の方向 ：中性子の飛ぶ方向・速度もサイコロにより
決める。それにより別の原子核にぶつかるか決まる。

例２：人生ゲーム：ルーレットを回して、就職や結婚などの
イベントが確率的に決まっていく**人生シミュレータ**。

- **モンテカルロ法**

  確率的現象を乱数を用いてシミュレーションすることを
  モンテカルロ法と呼ぶ。

  （モンテカルロ市はモナコの首都、カジノのメッカ）

  物理、化学：核・化学反応シミュレーション

  生物科学：たんぱく質折りたたみシミュレーション

  金融工学：株価変動、金融商品の価格付

  遊び、芸術：ゲーム、漫画の模様（トーン）、
  　コンピュータグラフィックス

乱数を、どうやって生成するか

…それが問題だ。
“決定的な動作しかしない計算機では、乱数は生成できない”

物理乱数発生器:

- 物理雑音から拾ってくる: もっとも素朴な方法
- 例：サイコロ、熱雑音、株価の変動
- 問題点: コスト、スピード、再現不能性.

再現不能性とは、同じ乱数列を再現するのに、
それらをすべて記録しておかないとならないこと。

再現性が必要となる場合：

- 追試
- 最適化

参考：核シミュレーションでは乱数は何兆個も消費される
⇒ それらをすべて記録しておくのは効率が悪い。

擬似乱数発生法:
漸化式を用いて、乱数のように見える数列を生成する方法

例: **線形合同法** Linear Congruential Generator
(LCG, Lehmer '60)

- ある整数x_1を初期シードとして選ぶ。
- 次の例のような漸化式により$x_2, x_3, \ldots$を次々に生成：

$$x_{n+1} = 1103515245 x_n + 12345 \bmod 2^{32}.$$

例 $x_1 = 3$ならば

$$\begin{aligned}
3 \times 1103515245 + 12345 &= 3310558080 \ (\mathrm{mod} 2^{32}) \to 3310558080 = x_2 \\
3310558080 \times 1103515245 + 12345 &= 3653251310737941945 \ (\mathrm{mod} 2^{32}) \to 465823161 = x_3 \\
465823161 \times 1103515245 + 12345 &= 5140429596376017 90 \ (\mathrm{mod} 2^{32}) \to 679304702 = x_4 \\
679304702 \times 1103515245 + 12345 &= 749623094657194335 \ (\mathrm{mod} 2^{32}) \to 2692258143 = x_5
\end{aligned}$$

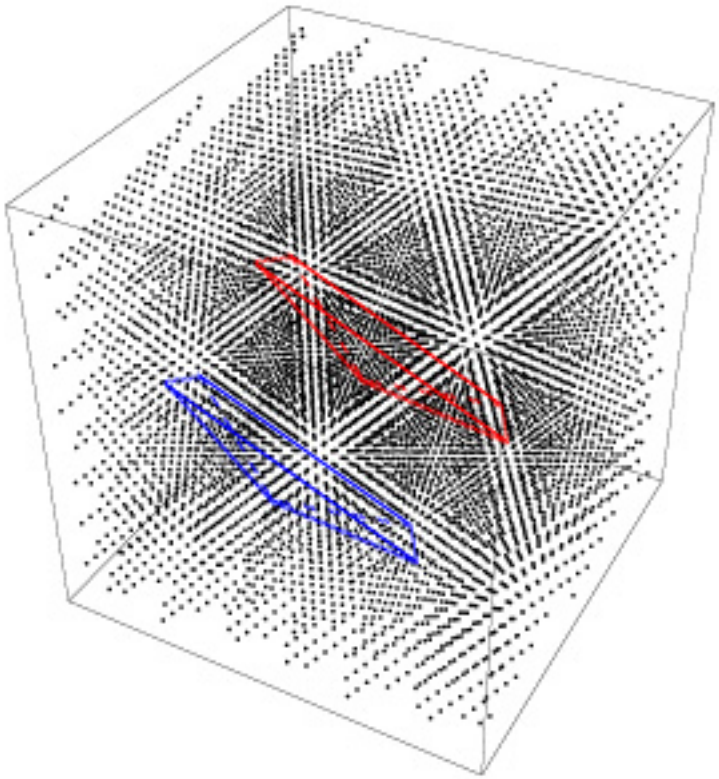
擬似乱数のメリット:

- 漸化式と初期シードを記録しておけば、
  誰でも同じ数列を再現できる
- 高速で低コスト

問題点：「乱数と呼んでいいのか」...
擬似乱数の創始者von Neumann
「漸化式で乱数を作るのはある種の罪」
“Anyone who considers arithmetical methods of
producing random digits is, of course, in a state of sin”

たとえば:

- 先の線形合同法は、70年代から80年代にかけて
  ANSI-Cなどの標準擬似乱数`rand()`であった。
  いまでも教科書にのっていて、広く使われている。
- この数列の周期は、初期シードの選び方によらず2^{32}。
- 現代のパソコンは数分で2^{32}個の乱数を使ってしまう
- 生成される数列はかなり乱数っぽく見えるが、
  数千万個の出力を使うと、非乱数性が現れてくる

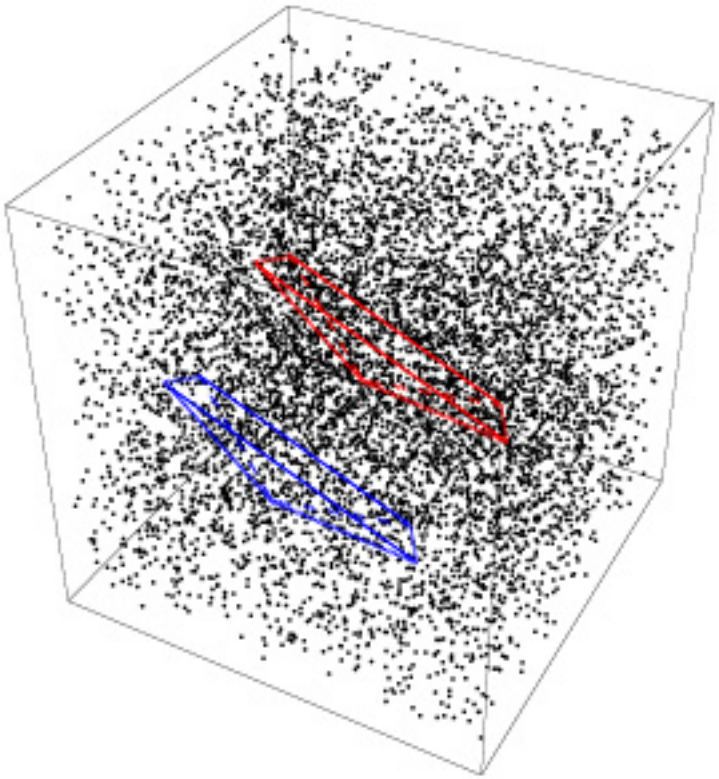


この線形合同法`rand`による
2^{32}個の3次元ランダム点プロット
（70倍拡大図）

赤領域体積$\simeq \frac{62}{7253} = 8.5 \times 10^{-3}$.

青領域体積$\simeq \frac{45}{7253} = 6.2 \times 10^{-3}$.

(真の値: $8.333\cdots \times 10^{-3}$)



メルセンヌ・ツイスター擬似乱数
(MT, 松本-西村 '98) による
3次元ランダム点プロット

赤領域 $\simeq \frac{61}{7248} = 8.4 \times 10^{-3}$.

青領域 $\simeq \frac{64}{7248} = 8.8 \times 10^{-3}$.

(真の値: $8.333\cdots \times 10^{-3}$)

擬似乱数の落とし穴

落とし穴その１：偶数と奇数が交互に出るものがある

先の生成法`rand()`

$$
\begin{aligned}
3 \times 1103515245 + 12345 &= 3310558080 \ (\mathrm{mod} 2^{32}) \rightarrow 3310558080 = x_2 \\
3310558080 \times 1103515245 + 12345 &= 3653251310737941945 \ (\mathrm{mod} 2^{32}) \rightarrow 465823161 = x_3 \\
465823161 \times 1103515245 + 12345 &= 514042959637601790 \ (\mathrm{mod} 2^{32}) \rightarrow 679304702 = x_4 \\
679304702 \times 1103515245 + 12345 &= 749623094657194335 \ (\mathrm{mod} 2^{32}) \rightarrow 2692258143 = x_5 \\
2692258143 \times 1103515245 + 12345 &= 2970947904275902380 \ (\mathrm{mod} 2^{32}) \rightarrow 3498995628 = x_6 \\
3498995628 \times 1103515245 + 12345 &= 3861195017686361205 \ (\mathrm{mod} 2^{32}) \rightarrow 1035215989 = x_7 \\
1035215989 \times 1103515245 + 12345 &= 1142376625729264650 \ (\mathrm{mod} 2^{32}) \rightarrow 273505290 = x_8 \\
273505290 \times 1103515245 + 12345 &= 301817257103158395 \ (\mathrm{mod} 2^{32}) \rightarrow 1018653819 = x_9
\end{aligned}
$$

- 奇数と偶数が交互に出る。
- この現象は、線形合同法を「mod 偶数」で使った場合必ず起きる。
- そして、知っていなければ気づきにくい。

(slashdot.jp より引用：)

- 2006年12月06日

  「カルドセプトサーガ」にダイス目が偶数と奇数を繰り返すバグ

  株式会社バンダイナムコゲームス発売のXbox360向けゲームソフト「カルドセプトサーガ」に「次のダイス目が偶数か奇数か推測できる」という致命的バグが見つかりました。

- 2006年12月16日

  ダイス目が偶数と奇数を繰り返すバグが取り上げられたXbox360向けゲーム「カルドセプトサーガ」ですが、CNETの記事によると店頭在庫の回収を行うことが決定したようです。

モンテカルロ法用擬似乱数への要請

- **高速性**

  素粒子シミュレーションなどでは、全体の計算時間の
  ４０％を擬似乱数生成が占めることもある
  (MTに変えたら全体が17%高速化したという報告あり)

- **乱数性**

  擬似乱数は、所詮漸化式による数列
  「乱数性」の実用的な定義がない
  ⇒「乱数である」ことを証明できない
  仕方がないので代わりに：

  – **周期, 分布**
    周期や高次元分布に着目して「良いもの」を使う

線形合同法の問題点：

$$x_{n+1} = ax_n + c \bmod M$$

なる数列の周期は高々M
（$\because x_n$が取りうる値は$0, 1, \ldots, M-1$のM個しかない）

メリット：周期や分布を求めるアルゴリズムがある
限界：周期を大きくするにはMを大きくするしかなく、
掛け算や割り算が必要で生成が遅くなる

擬似乱数の落とし穴その２：周期が短い

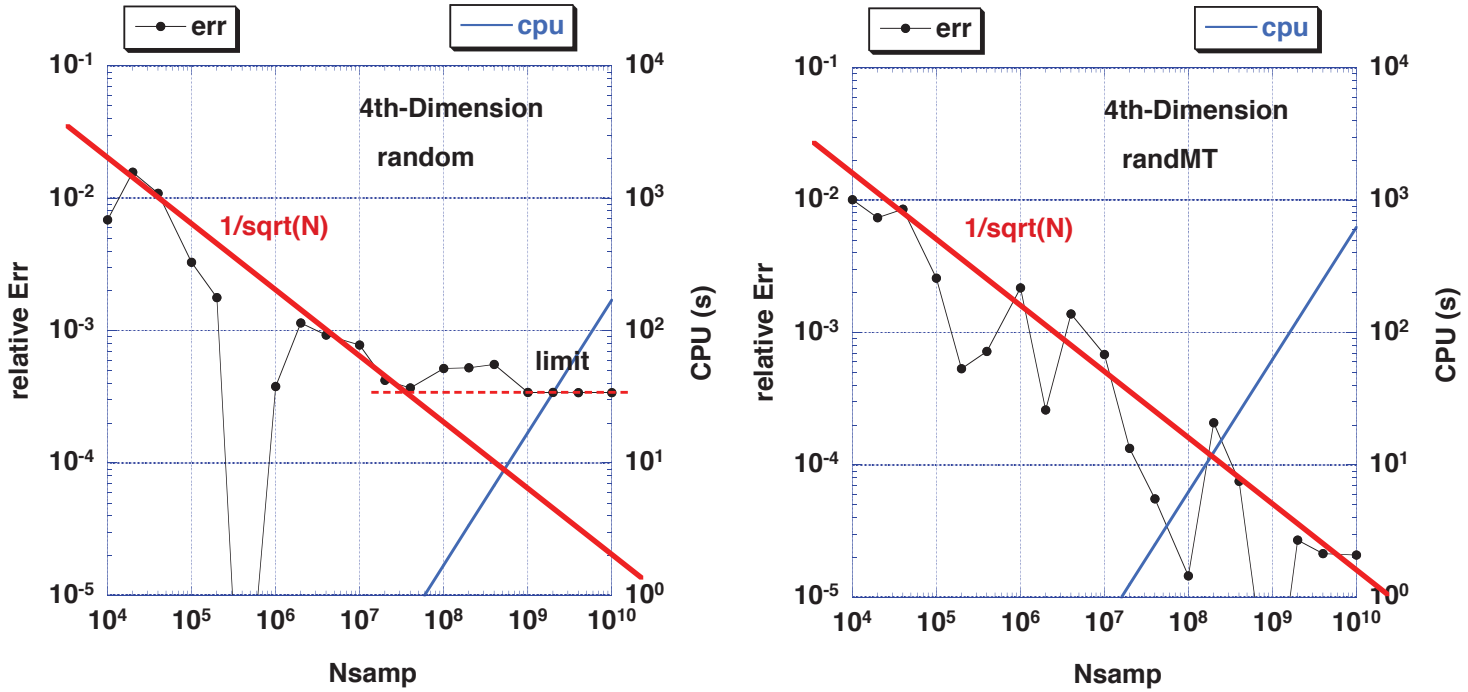
rand(), TurboC 1.5, Visual C++, Borland C++, エクセルの
Visual Basicの標準乱数は線形合同法
（2000年ごろの和田維作氏による調査、現在は改善か？）

周期は：$2^{32} = 4294967296$以下、エクセルは2^{24}以下。
→プログラム開発初期段階ではうまく行くが、
本格的に大量の乱数を使うとおかしくなるという
やっかいな状態

例：４次元球の体積のモンテカルロ積分誤差
左:rand(周期2^{31})　右:メルセンヌ・ツイスター(周期$> 10^{6000}$)

提供:日本原子力研究開発機構 核融合研究開発部門 清水勝宏氏・慶応大学 理工学部 物理学科

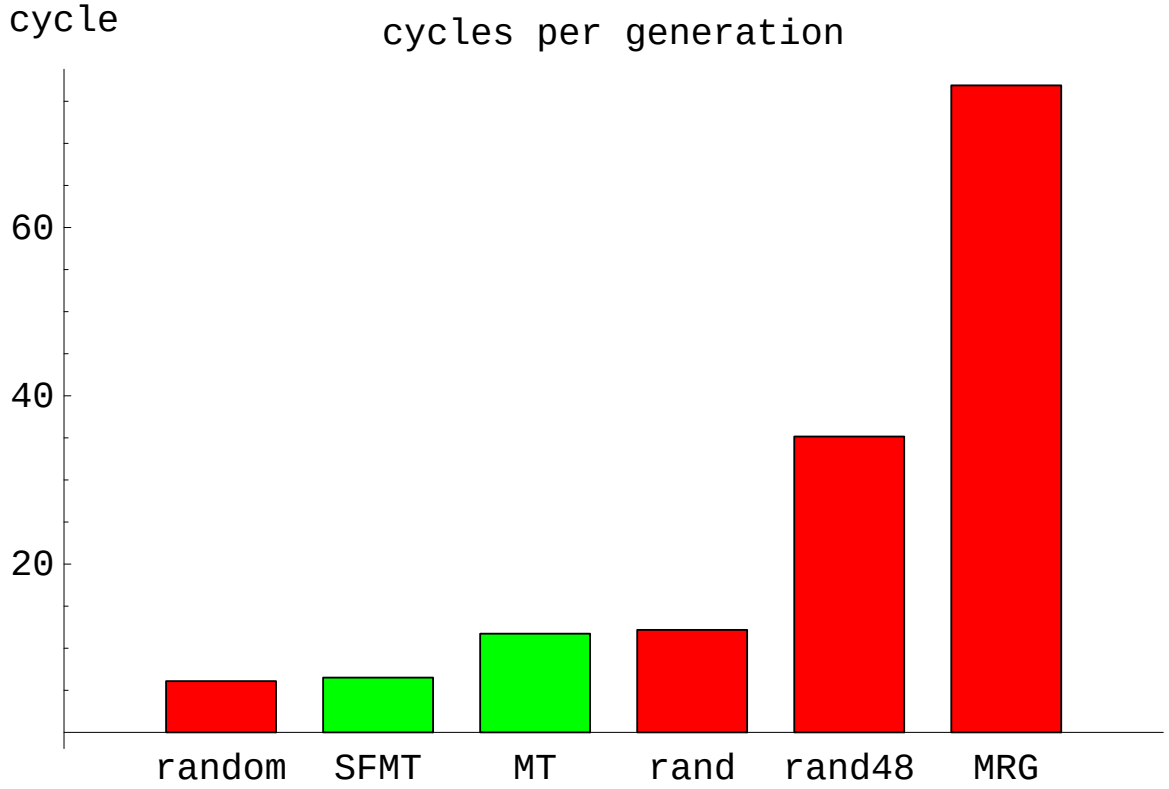
早川奈伊紀氏 (2014/8/27 松本宛電子メール「長周期乱数コードに感謝」より)



メルセンヌ・ツイスター法(松本-西村拓士 '98):

- 周期：$2^{19937}-1 \fallingdotseq 4.3 \times 10^{6001}$
- 1周期で623次元空間に均等分布することが証明
  （32ビット精度で）
- 生成速度は、近年の線形合同法（ $\bmod 2^{48}$）よりも高速
- 多くの計算機言語で標準擬似乱数として採用(Python, Ruby, R, PHP, MATLAB, C++（C++11から）など)
  他、広く用いられている
  （多くのソフト、ポケモンゲーム、任天堂Wiiなど）
- MTのWikipediaも見てください

速度比較 32ビット整数一個を生成するのにかかるCPUサイクル数



random: ラグ付きフィボナッチ周期 $\sim 2^{63}$

SFMT SIMD Fast MT

MT: Mersenne Twister
周期 $2^{19937}-1$

rand: LCG 周期 2^{31}

rand48: LCG 周期 2^{48}

MRG: L'Ecuyer
周期 $\sim 2^{186}$

メルセンヌ・ツイスターの動作原理:$1+1=0$の数学：

二元体$\mathbb{F}_2$

$\mathbb{F}_2 := \{0,1\}$とおく。

0,1の掛け算は普通に定義して、$\mathbb{F}_2$からはみ出ない。
足し算は$1+1=2$だけが$\mathbb{F}_2$からはみ出してしまうので、

$$1+1=0$$

と定義する（2で割ったあまりを見ている）。

$\mathbb{F}_2$の世界で漸化式

$$x_{n+3} = x_{n+2} + x_n \ (n \geq 1), \quad x_1 = 0, x_2 = 0, x_3 = 1$$

で$x_1x_2x_3x_4\cdots$を計算すると周期7で循環

123456789

0011

00111

001110

0011101

00111010

001110100

0011101001

00111010011

001110100111

同様の漸化式

$$x_{n+607} = x_{n+273} + x_n$$

で数列を作る（初期値として$x_1, x_2, \ldots, x_{607}$に607個の0または1を与える）と周期が$2^{607} - 1$となることが示せる。

注：この数は約5.3×10^{182}である。

１ペタFlopsの計算機は、１秒に10^{15}回の計算しかできない。計算機で一周期求めることはできない。

宇宙の素粒子の数でも10^{80}くらいである。

実験ではなく、数学的証明によってのみ周期を示しえる。

このような$\mathbb{F}_2$上の高階線形漸化式を用いて0-1列を生成、
擬似乱数として用いる方法を

Tausworthe法あるいは

Linear Feedbacked Shift Register法（LFSR法）

と言う (Tausworthe, 1965)。

メリット：

- 周期を長くしても、生成速度が遅くならない

  「二か所見て、足して、一か所に書く」という動作だから

$$x_{n+607} = x_{n+273} + x_n$$

- 周期が極大なので、「全て0」以外のビットパターンが

$$(x_n, \ldots, x_{n+607-1}) \quad (n = 1, 2, \ldots, 2^{607} - 1)$$

  のなかにちょうど一度だけ現れる（607次元均等分布性）

ベクトル化

GFSR (Lewis-Payne '73) 計算機ワード長の$\mathbb{F}_2$ベクトル列を

$$\vec{x}_{n+p} := \vec{x}_{n+q} + \vec{x}_n$$

で生成(+は$\mathbb{F}_2$ベクトルとしての和)

例 4 ビット計算機なら 4 ビット同時に作る

$$\begin{array}{llll}
0101 & \vec{x}_1 & & \\
0110 & & \vec{x}_2 & \\
1111 & \vec{x}_3 & & \\
1010 & \vec{x}_4 & \vec{x}_4 & \\
1100 & & \vec{x}_5 & \\
0011 & & & \\
1001 & & & \\
0101 & & &
\end{array}$$

整定数p, qをうまく選ぶと周期$2^p - 1$にできる

- pとして31〜607が良く使われていた
- 各桁はTausworthe法で生成される数列に一致
- 高速だが、各桁の間に情報のやり取りがない
- 乱数性に問題あり（特にランダムウォークで)

思いつき $\Longrightarrow$ 桁の間に情報の攪拌(Twist)を行ったら？

結果:非常に効果があった。

Twisted GFSR (松本-栗田良春 '92, '94):
Twisterと呼ぶ$\mathbb{F}_2$係数正方行列Aを導入する：

$$\vec{x}_{n+p} = \vec{x}_{n+q} + \vec{x}_n A.$$

- A は桁の間の情報を混ぜる

  $\Rightarrow$ より長周期: $2^{wp}-1$が達成可能(w:ワード長)

- Aは次のようなものを選ぶ:定数ベクトル$\vec{a}$により

$$\vec{x}A = \begin{cases} \mathrm{shiftright}(\vec{x}) & (\vec{x}\text{の最下位ビットが}0\text{の場合}) \\ \mathrm{shiftright}(\vec{x}) + \vec{a} & (\vec{x}\text{の最下位ビットが}1\text{の場合}) \end{cases}$$

  $\Rightarrow$高速に計算可能, 十分一般: $A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix}$.

- 高次元均等分布性を改善するため$\vec{x}_n T$を出力('94)

開発秘話その１：

’87学部4年生のときに、当時指導教員の米田信夫教授から
GFSR法とその問題点を聞いた。
GFSR法を統計検定しているつくば計量研究所の栗田良春
氏を訪問、いろいろ改良をこころみた。
（当時のスーパーコンピュータ・クレイが使い放題。
段ボール箱ふた箱に検定結果をプリントアウトした。）

根津の飲み屋「車屋」で米田研で飲んでいるとき、
松本「GFSRの1ワードを、rotateしてはどうでしょう」
米田「それは僕もやった。けど、あまり良くなかった。」
松本「では、companion行列を掛けたらどうでしょう。」
米田「...それはいい考えだ。早速、また栗田さんのところに
行きなさい。」

開発秘話その２

：Twisted GFSRに関する投稿論文は、投稿した学術雑誌が紛失した。

投稿後2年たっても「査読中」と言われた。3年目に、雑誌の編集長が変わったときに「紛失した」と言われた。

そのうえ、rejectされた。ので、ぐれて**ほったらかした**。

’90に擬似乱数の大家Pierre L’Eecuyer教授が来日したとき、直接Twsited GFSRを説明する機会があり、日の目を見ることになった。

(この「他力」がなければ、MTはなかったかもしれない。)

Mersenne Twister: (松本-西村拓士 '98):
TGFSRの周期を$2^{wp-r}-1$の形にするため、
状態空間wpビット中r次元を「使わない」工夫：

$$\vec{x}_{n+p} = \vec{x}_{n+q} + \vec{x}_{n+1}B + \vec{x}_n C$$

$\Rightarrow 2^{wp-r}-1$を素数にするように選ぶことができる。
　素数に対する周期判定は容易なため、
　超長周期が実現可能: $2^{624\times 32-31}-1 = 2^{19937}-1$.

注：2^p-1の形の素数をメルセンヌ素数、
そのときのpをメルセンヌ指数という。

現在48個知られており、
19937は24番目のメルセンヌ指数。
2013年1月時点で知られている最大の指数は257885161

開発秘話その３

メモリの一部を使わないことによって一般性を増し、
周期をメルセンヌ素数にする考えは'94ごろ思いついた。

が、自分でパラメータ探索プログラムを書いてもうまく動かなかった。

当時すでに研究を純粋数学分野にシフトしていたため、
このアイデアを**ほったらかした**。

'95慶応大学理工学部数理科学科講師として赴任。
榎本彦衛教授の修士学生であった西村拓士氏に
MTのアイデアを話し、実験を始めた。
理論と実験が合わなかったとき：
僕の証明の間違い＝西村氏のプログラムが正しい確率9割。
⇐ 彼の力が無ければMTは実現できなかった可能性大。

普及への道

96年国際会議でMTを口頭発表。

96年ザルツブルグ大学の乱数研究グループホームページでニュースとして取り上げられる。

97年10月27日朝日新聞夕刊記事（内村直之記者）

素数理論使いコンピューターに新たな計算手順

一層でたらめな乱数発生に成功

世界で利用進む動き

松本・慶応大講師ら

でたらめな数である「乱数」は、社会現象や物理現象の数値シミュレーションに幅広く応用されている。ところが、コンピューターで作った乱数は、意外にでたらめでなく、時には正確な答えが出ないことがある。慶応大理工学部数理科学科の松本眞講師と大学院生の西村拓士さんらは素数の理論を使い、これまでよりもはるかにでたらめな乱数を作ることに成功した。この計算手順を公開したため、世界的に利用が進み始めている。

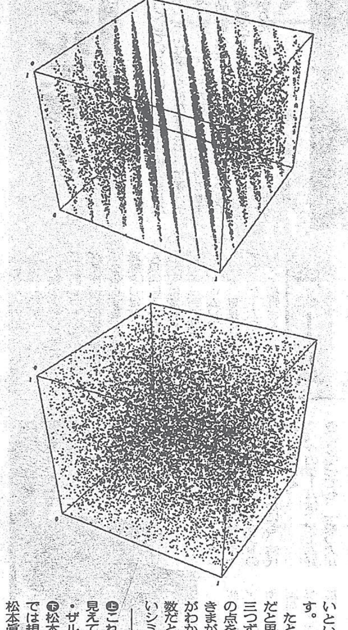
乱数はサイコロを何度も振って作るようなでたらめな数の列だ。この乱数を使って、偶然に起きる現実世界の現象を模擬する「モンテカルロ法」と呼ばれる数値シミュレーションでは、何十万、何百万個もの数を作る必要がある。このため、コンピューターで利用できる簡単な乱数の発生法が重要になる。

コンピューターで乱数を作るには、一つの数を元に、次の数をある計算手順で計算し、それをまた元に計算するという方法を使う。いつでも同じ乱数を作れ、同じシミュレーションが再現可能という利点があるが、コンピューターは使えるメモリーに限界があり、限られた数しか扱えない。長い周期になると、元と同じ数が出てくるため、でたらめな乱数ではなくなる。

四十年以上前に作られた「線形合同法」と呼ばれる乱数発生プログラムは今もパソコンなどで利用されているが、最近はそのでたらめさに疑問の目が向けられている。一九九二年に米国で、厳密に解くことの可能な物理の問題を、いろいろな乱数を使った数値シミュレーションで解き比べたところ、線形合同法など一部の方法では誤った答えが出てくることがわかった。

乱数に詳しい東大工学部の伏見正則教授（数理工学）は「一様にばらまかれているはずの数を、いくつか組みにして見ると、そんなにばらばらじゃないという事実があった」と話す。

たとえば、これまでよい乱数だと思われていた乱数を頭から三つずつとっていき、たくさんの点を三次元で表示すると、すきまが規則的にできていることがわかる（図㊤）。こうした乱数だと本当にでたらめさがほしいシミュレーションではおかしな結果が出る。



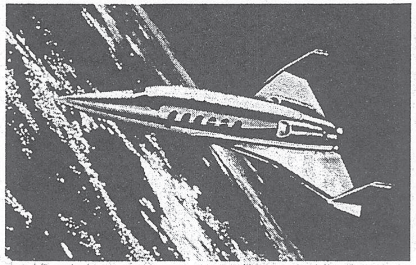
㊤これまでの乱数では規則性が見えてしまう。可視化はドイツ・ザルツブルグ大の協力による
㊦松本さんたちの開発した乱数では規則性は出ない＝いずれも松本眞講師提供

松本さんらは、メルセンヌ素数と呼ばれる大きな素数のひとつである「二の一万九千九百三十七乗引く一」という素数を使い、メモリー上にこれだけの数の状態を作り出す方法を考えた。現在のパソコンでは二進法の三十二けた（三十二ビット）で一つの数を表している。松本さんたちは、この二進法の数字を一万九千九百三十七個並べ、その一部分を変換して三十二ビットの乱数の列を発生させる新たな計算手順を開発した。

さらに「二の一万九千九百三十七乗引く一」が素数であることを利用し、これがこの方法で作った乱数の周期となることを数学的に証明した。この周期は十進法では約六千けたになり、今のコンピューターで計算し尽くすことは不可能だ。結果として使いきれないほどの乱数が手に入ったことになるという。

この乱数を使い、三つずつ数字をとった点を三次元プロットしても、すきま構造は現れない（図㊦）。松本さんたちは論文でこの計算手順を公開し、だれでも乱数を使えるようにした。米国の大学研究者からはプラズマのシミュレーションに使ったところ正確な答えを出すのに一七%もスピードアップしたと感謝されたという。

伏見教授は「コンピューターの限られたメモリーでこれだけの乱数が作れることは大変ユニークで面白い。さらに研究を進めてほしい」と話している。



無重量楽しむ宇宙旅行を　米で予約受け付け

米国シアトルにある旅行会社のゼグラム旅行社が宇宙旅行の予約受け付けを開始した。ロケットエンジン搭載の宇宙船で、オゾン層を超える地上約百キロの上空を飛び、二分間ほどの無重量状態を楽しむ＝写真AP。全体の飛行時間は二時間半から三時間で、参加費は九万八千ドル（約千百八十万円）。二〇〇一年から週二便を運航する計画で、一度に六人が飛べる。同社の話では、参加者は二日間の教育と三日間の宇宙飛行士訓練を受ける必要があるという。

予約受け付けを始めた今月二十日から二十三日までに少なくとも十五人が予約し、五千ドルの前払い金を払ったという。

同社は現在、ロケットエンジンをテスト中で、宇宙船二隻の建造は来年から始めるという。宇宙船建造のコストは明らかにされていない。（AP）

鳥の先祖、恐竜ではない

翼と前肢比較し独立進化説

米博士ら発表

鳥類の最古の祖先とされる始祖鳥は恐竜から進化したという説がこれまでは有力だったが、米ノースカロライナ大のアン・バーク博士らの研究グループは、鳥類の翼を恐竜の前肢の指の化石と比較した研究から、鳥類と恐竜は独立して進化した可能性が高いとの新説を打ち出し、二十四日発行の米科学誌サイエンスに発表した。

バーク博士らは、これまで発見されている獣脚類の恐竜の前肢の化石を分析したところ、五本の指のうち、薬指と小指は退化し、残った三本の指は親指、人さし指、中指に相当する部分であることが分かった。

一方、鳥類を対象に、受精卵から成長する過程の胚（はい）を生物発生学の手法で分析した結果、前肢が進化上発達したとみられる翼が成長する過程で三本の指の痕跡があった。この三本は人さし指、中指のほか、恐竜と異なり、親指ではなく薬指の名残があったという。

これまでの研究で恐竜と鳥類はあごや脚などの骨格の形態が異なることが分かっていた。研究グループは、今回の分析結果などから、鳥類は恐竜から枝分かれしたのではなく独立した進化系統を持っていると考えた方が自然だとしている。

恐竜は二億数千万年前の三畳紀末に、は虫類の一種から進化。始祖鳥は約一億五千万年前ごろのジュラ紀に登場し、恐竜から原鳥という「空飛ぶ恐竜」の形態を経て進化したとの説が有力だった。（共同）

科学

酔いざめの早さは遺伝子が関係

血中のアルコール濃度同じでも

理研など発表

酔いざめが早いか遅いかのかぎは脳内で働く遺伝子が握っているらしいことを、理化学研究所や岡崎国立共同研究機構生理学研究所などのグループが突き止め、二十四日発行の米科学誌サイエンスに発表した。酒に強いかどうかはアルコールを分解する酵素に関係するが、血中のアルコール濃度が同じでも、この遺伝子には頭を早くはっきりさせる働きがあるようだ。

理研脳科学総合研究センター情動機構研究チームの宮川剛研究員らの研究で、マウスを使って、脳内で発現し、記憶や学習にかかわるｆｙｎという遺伝子を研究した。この遺伝子が欠けたマウスは憶病になり、恐怖心にも関連するとみられる。

宮川さんらは、脳に作用する様々な薬物を、正常なマウスと、この遺伝子を欠損させたマウスに与えて反応を比較した。

マウスはあおむけになってもすぐに起き上がって腹ばいになる。しかし、体重一キロあたり三グラムのアルコールを注射すると、正常マウスは平均二十二分で腹ばいになったのに、欠損マウスは四十一分かかった。泥酔状態になる三・五グラムだと、それぞれ四十分と八十分だった。両者の血中アルコール濃度には差がなく、運動機能も正常だったため、腹ばいになる時間差は遺伝子の働きによると考えられた。

アルコールは脳内の神経伝達物質の受容体の一種の働きを抑制し、ｆｙｎ遺伝子がつくるたんぱく質はこの抑制を素早く回復させるとみられている。宮川さんらは、この遺伝子が欠けるとアルコールの作用が続き、酔った状態が長引くとみている。

だ液のDNA調べて遺伝性疾患など診断

米博士ら研究

米ニューヨーク州立大のマーク・ウィルソン博士らは、血液の代わりに、だ液のDNAを調べることにより、特定の遺伝性疾患の診断や、個人の識別が可能になると発表した。だ液による検査は、患者に血液検査のような負担をかけないため優れた検査方法になるという。

同博士らは子どもを対象に、呼吸器疾患や自己免疫型の皮膚疾患、若年性の歯周炎などに関係する遺伝子の異常を調べた。その結果、だ液による検査でも血液を使った場合と同様の有効性が確認されたという。

ウィルソン博士は「血液検査は負担があるため、子どもにはあまり勧められない。また、中には宗教的な理由で採血がむずかしい人もいる。これまで血液検査ができなかった人でも、だ液による検査なら可能だ」と話している。（ロイター）

薬の効かない結核菌が急増

WHOなど報告書

世界保健機関（WHO）と米疾病管理センター（CDC）などはこのほど、薬の効かない多剤耐性結核菌が急速に広がり始めているという報告書を発表、結核対策強化の必要性を改めて強調した。とくに途上国では、こうした結核菌への感染は死刑宣告にも等しい状態で、事実上の「結核非常事態宣言」といえる。

報告書によると、三十五カ国で五万人の患者を調べたところ、多剤耐性菌は、三分の一の国で見つかり、五大陸すべてに広がっていることがわかった。インドのデリーでは結核患者の一三%、ラトビアでは二二%が多剤耐性菌によるもので、ほかに、ロシア、ドミニカ、アルゼンチン、象牙海岸などで目立っている。

報告書をまとめた米コロラド大のマイケル・イスマン教授は「最も恐れていたことが証拠として示された。世界は再び、結核という不治の病の脅威にさらされている」と述べた。

結核の治療は、多剤の併用療法を六―八カ月程度続けることが必要だ。しかし、貧しい国では結核の治療そのものが不十分だったり、患者が経済的な困難などから、途中で薬の服用を止めることが少なくない。こうした不適切な治療が多剤耐性菌の流行につながっているとみられる。

WHOの責任者は「四年前、WHOは結核が緊急事態であることを宣言したが、今や結核との戦いに敗れつつあるのは弁明が許されないことだ」と話し、各国が緊急対策を取るよう訴えている。（アメリカ総局）

宇宙ステーション産業利用を模索へ

宇宙開発委に分科会

来年六月に打ち上げと建設が始まる予定の国際協力計画宇宙ステーションについて、産業分野への利用を模索する分科会が国の宇宙開発委員会の下に設けられることが決まった。米国ではすでに宇宙ステーションに商業利用枠が設定されるなどの動きがあり、日本でも検討していくことになった。

分科会の会長にはNTTアドバンステクノロジの池上徹彦社長が就任。新日鉄や三菱重工業、清水建設といった重厚長大型産業のほか、製薬会社などからもメンバーを求めた。大学や宇宙開発事業団の研究者らとともに、宇宙環境の産業利用について検討する。

どうして新聞に載ったか？
⇒ 96年夏ごろ朝日新聞に「秋の夜の数学」というシリーズ企画を内村さんが書いていた。
この企画で、慶応理工の教授が内村さんに取材を受けた。
その教授が、MTについて内村さんに売り込んだ。
内村さんが取材に来た。大きな記事になってびっくりした。
（東大工学部の伏見正則教授に、その重要性を確かめてくれていた。）

⇒ 慶応大学に電話での問い合わせが多数あった。
電話で回答するのには無理があった。(プログラムを下さい、という依頼。)

普及への道（つづき）

回答できないのでホームページを立ち上げた。
ついでに英文のホームページも立ち上げて、
無料でダウンロード可能とした。
ライセンスフリー、商用利用も許可なく可能とした。
⇒ 多数のボランティアが無償で多くの言語に移植し、
高速バージョンを作って送ってくれた。

それをまた、ホームページに載せた。

- なまけものなので、新聞記事にならなければ
  ホームページもつくらず忘れ去られた可能性大。
- インターネットがなければ、忘れ去られた可能性大。
  （乱数の大御所のKnuth, Compagner, Marsagliaらが、
  「整数の方が2元体より乱数生成に適している」
  と主張しつづけており、MTの発表後も、
  古典的なアルゴリズムを推奨しつづけた。）
- 有償にしていたら、普及しなかった可能性大。

普及への道（つづきその２）

ホームページで発表後、MTは世界中で使われ始めた。
多くの言語、ゲーム、商用プログラムで利用が広がった。

’98年JIS規格の「ランダムサンプリング」の改訂にあたり、伏見正則教授が改訂委員長となり、
「日本発の世界規格を目指して、MTをJIS規格乱数の目玉としよう」
という動きになり、JIS規格に入った。

’07年、ISO規格で「ランダムサンプリング」を改訂するにあたり、統計数理研究所椿広計教授らがISO規格委員に入り、JIS規格をもとにしてMTを含めた標準疑似乱数の提案を行い、採択された。

(ただし、規格になる前にすでに広く普及していた。)

数学の効用

周期と高次元分布性を保証するのに、ガロアに遡る近代数学が用いられている。

有限体、線形代数、メルセンヌ素数、Berlekamp-Massey法、格子縮約アルゴリズム、MacWilliams恒等式などなど。

研究された当時は応用など思いもつかなかった$1+1=0$の数学が、今、実用されている。

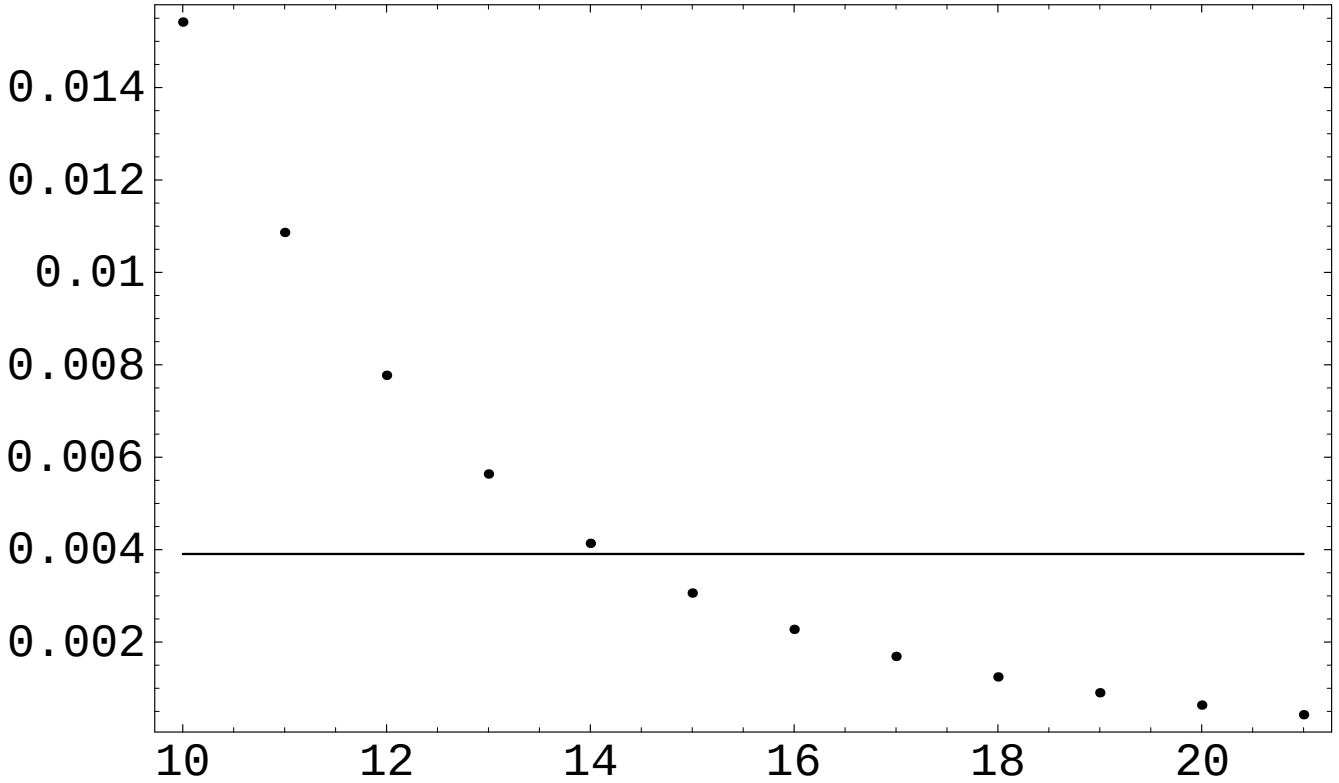
終わり

終わりに：今でも危険な疑似乱数が使われている

`random`: ’90-現在UNIX系C言語での標準的擬似乱数の最下位ビット0-1を見る。(原本博史-M ’07)

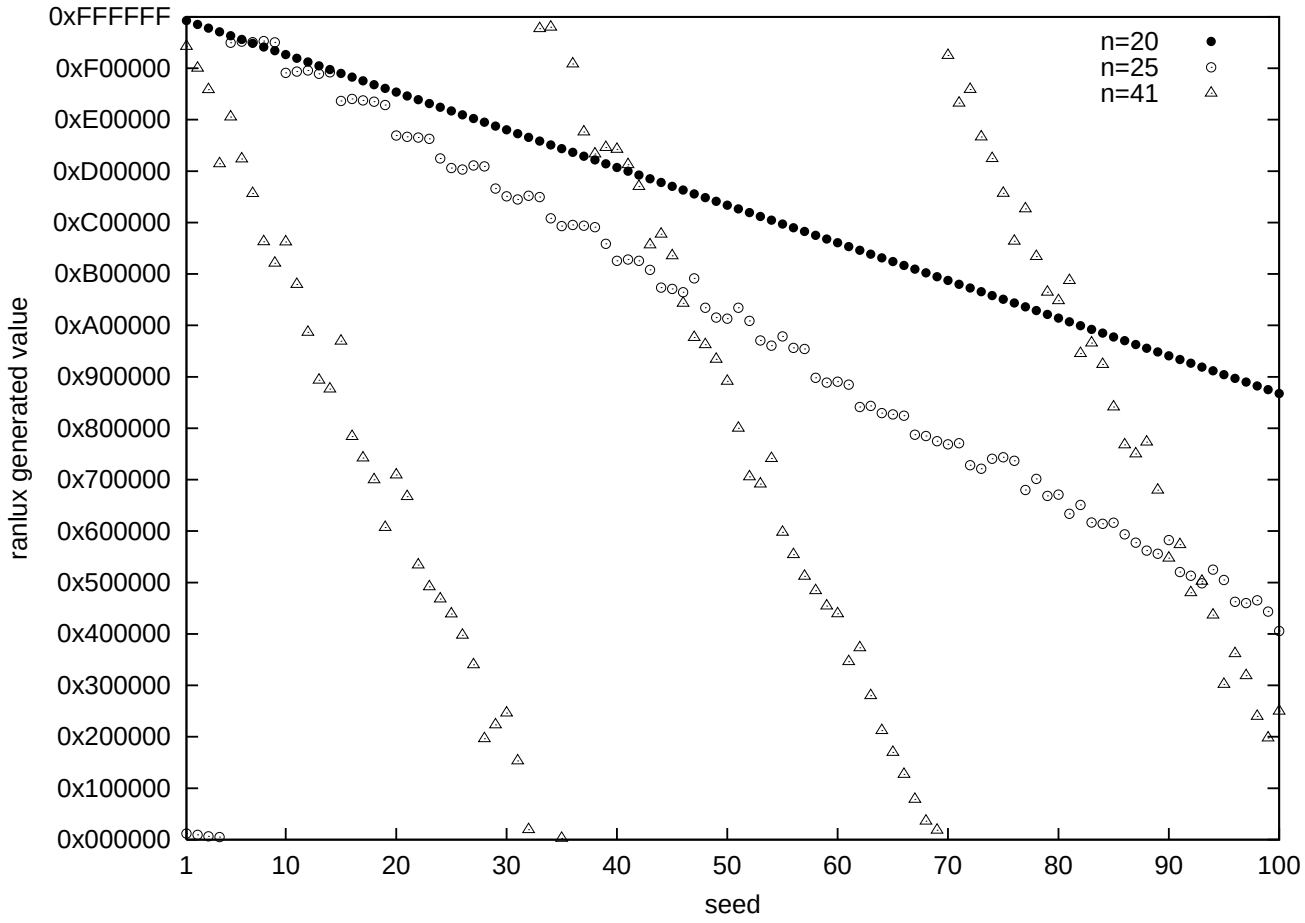
横軸：過去31回中の1の個数

縦軸：その条件下で、次の8回が全て0の確率：



初期値を系統的に選んだときの、非乱数性

`ranlux`(カオス理論に基づく擬似乱数, Lüscher '94) で、
20, 25, 41番目の出力（縦軸）を、初期シードを1, 2, 3, . . . , 100
（横軸）と動かしてプロット「20番目の打者がいつもヒット」



GNU Scientific Libraryに入っている
58個の擬似乱数発生法のうち、最新のものも含めて
45個にこのような問題が観測された。
Mersenne Twisterでは観測されなかった。
（M, 和田維作、倉本愛、芦原評 2007）

⇒ モンテカルロ法にはMTを使いましょう。

(注：MTは暗号用にはそのままでは使えません。)

蛇足

- 2008年5月16日 DebianのOpenSSLに脆弱性:
  SSL／SSH暗号鍵の基になる乱数が推測可能
  ⇐ 乱数発生アルゴリズムが悪かった。
- 2010年8月13日 乱数生成器の問題によって、
  Android版BitCoinアプリで生成したウォレット
  は脆弱な秘密鍵を利用している可能性がある
  ⇐ 初期値を十分ランダムにとっていなかった。

- 2010年10月02日ポケットモンスターBW
  「乱数調整で強力なモンスターをゲット」
  ⇐ MTを使っているが、初期値をMACアドレスと
  時刻のみから決めていた。
  時刻を選んで(MACアドレスとMTを使って)乱数を再現
  することで、望みのパラメータをもつモンスターの出現
  時期がわかる。

目的にあった疑似乱数を、正しく使いましょう。

御清聴ありがとうございました。

最近の研究

- SIMD-oriented Fast Mersenne Twister
  (SFMT, 斎藤睦夫-M, 2006)
  最近のCPUは128ビット演算命令
  (Single Instruction Multiple Data, SIMD)
  を持っている。

  SIMD命令やパイプライン処理を最大限に生かした
  漸化式を考案。

  SFMTは、SIMDを使わなくてもMTよりも高速。
  SIMDを使うと4倍程度高速。

  周期はMTと同じ、高次元均等分布性はMTより良い。
  （ホームページで公開中、ダウンロード数万件）

- WELL (Panneton, L'Ecuyer, M '06)
  高次元均等分布性について、理論的上限を達成し、MTに近い高速性を持つ
- CryptMT: 暗号用MT, MTの出力32ビット整数を奇数化して整数積算($\bmod 2^{32}$)し、最上位8ビットだけ使う
  (斎藤睦夫、西村拓士、萩田真理子-M, 2007)
- Graphic Processor (高並列性・低機能マルチプロセッサ)用のMersenne Twisterの開発、(斎藤睦夫 2010)
- Dynamic Creator:多数のマシンで別々の疑似乱数を使うため、多数のパラメータを生成。(M-西村98, 斎藤2010)
  特に原瀬らの「最短基底計算の高速化」が有用。
- WAFOM:準モンテカルロ法の新指標(M-斎藤-Matoba2013)

総まとめ

擬似乱数は極めて大量に用いられる。微細な統計的偏りや、初期値へのわずかな依存性が、計算機の高速化・大規模化に伴いシミュレーションを狂わせる可能性がある。

この際、使用者は擬似乱数が原因だと中々気づけない。

⇒　精密なデタラメさを高速生成する必要性

それに答えるのが、「1+1=0の数学」

講演者は、

「擬似乱数をMT系に変えたらうまく動いた」

というメールをたくさんもらっている。