

間違いだらけの疑似乱数選び

松本 眞 (京都大学総合人間学部)
西村 拓士 (山形大学理学部)

2002/Feb./6 慶応大学湘南藤沢キャンパス

matumoto@math.h.kyoto-u.ac.jp
<http://www.math.h.kyoto-u.ac.jp/~matumoto>

1. 疑似乱数

疑似乱数とは：

計算機の中で

あたかもサイコロを振って得られるかのような
でたらめな数列を

高速に再現性があるように生成する方法の総称

使用目的：

- 確率的現象と関わる、あらゆる現象のシミュレーション（物理、金融、ゲーム...）
- 情報の暗号化（ここでは扱わない）

例：核分裂

- 一つ一つの原子核が、一定の確率 p 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、また別の確率で分裂

実際に実験してみる

環境に厳しい

計算機実験の必要性

一つ一つの確率現象をシミュレートするごとに
疑似乱数が消費される。

核分裂： $[0,1]$ に分布する疑似乱数 x を一つ生成し、
 $x < p$ なら分裂、 $x \geq p$ なら非分裂

中性子：中性子の飛ぶ方向も疑似乱数により決められる

- 計算機で作る必要性
- 高速性 10^8 個の原子核を単位時間シミュレートするのに 10^8 個の疑似乱数が消費される
- 再現性 同じ実験を何度も繰り返したい
(例: 条件をわずかに変えながら、あるいは追試)

周期性 :

再現性のある疑似乱数列は、かならず周期的になる。
(有限状態数の計算機を使って生成するかぎり)

∴ 計算機が有限個の状態しかもたない以上、
次々に数を生成していくうちにいつかは
かつてなったとの同じ状態になる
その後生成される数列は同じ
周期が存在する
(周期 \leq 状態の数)

2. 既存の方法とMersenne Twister

既存の生成法の問題点：

- 乱数性に問題があるか、さもなくば遅い
- 最新の方法でも同様
(古い生成法のマイナーチェンジ)

解決策

Mersenne Twister：1997年に松本-西村により提唱された高速高品質疑似乱数生成法

- 周期： $2^{19937} - 1$
- 623次元空間での均等分布が証明されている
- 計算機ハードウェアにフィット 高速

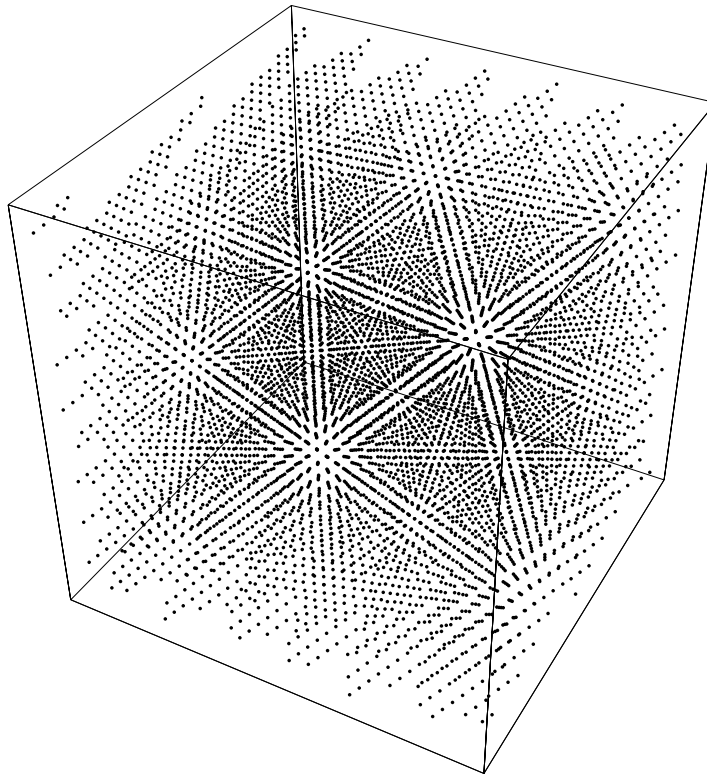
3. 古典的randとMersenne Twister

rand: C言語の標準乱数。線形合同法=漸化式

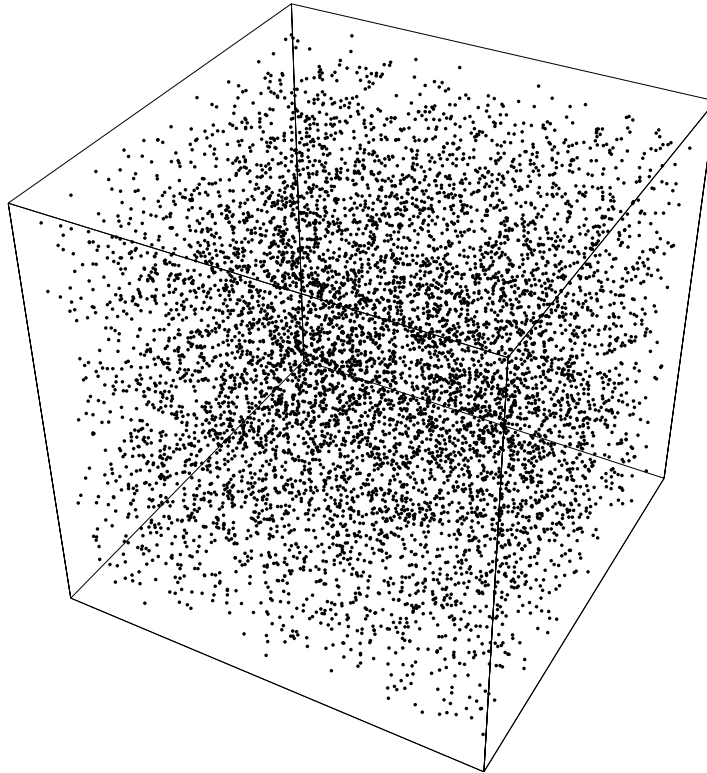
$$x_{n+1} := ax_n + c \pmod{M}$$

で0..M-1の整数列を発生して[0,1]に正規化する。

- 70年代-90年代までC言語の標準疑似乱数だった。
- $a = 1103515245$, $c = 12345$, $M = 2^{31}$ 。
- 周期 $M = 2^{31}$ 。



randを使って、3次元単位立方体の中に「ランダム」に点を打った図（70倍拡大図）



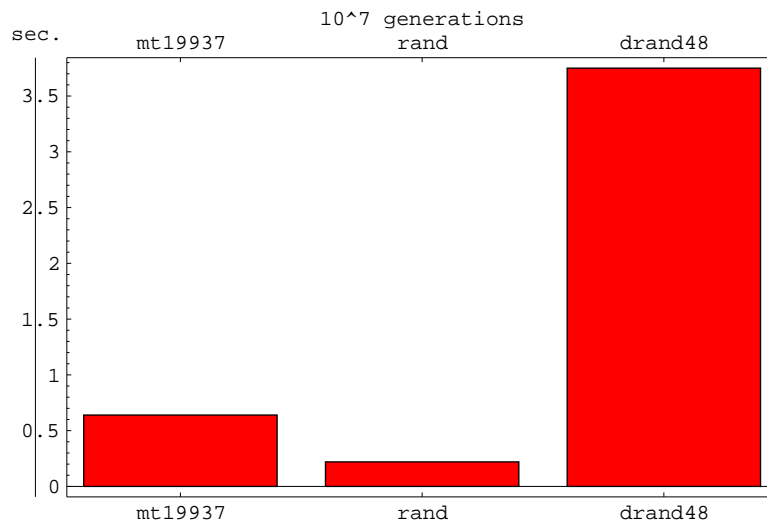
同じ図をメルセンヌ・ツイスター mt19937 で発生させたもの

上の3次元プロット図の作り方：

- $[0,1]$ に分布する疑似乱数列 $x_1, x_2, x_3, x_4, \dots$ を生成する。
 - $(x_1, x_2, x_3), (x_4, x_5, x_6), \dots$ を3次元空間の座標とする点を次々と打つ
 - 2^{31} 個の点を打ち、原点近くを70倍に拡大する
- 計算機能力の向上(数分で 2^{31} 個) 欠陥が明らかに

drand48: 80年代にrandの代替標準となった線形合同法

- $a = 5DEECE66_{16}$, $c = B_{16}$, $M = 2^{48}$.
- 周期= $M = 2^{48}$.
- まだ欠陥がある（視覚化はしにくい）。
遅い（32-bitマシンでは）。



mt19937, rand, drand48のスピード比較:
 10^7 個の生成にかかる秒数。

4. 90年代上位標準 : random

random : BSD他C言語の**現**標準乱数 (90年代 ~)

Lagged Fibonacci生成法の一つ :

整数列 x_n を次の漸化式で生成

$$x_{n+p} := x_{n+q} + x_n \pmod{M}.$$

- p, q, M : 整数定数, しばしば $M = 2^w$.
- 周期: $\leq 2^{w-1}(2^p - 1)$ (ここに w はワード長).
- 欠陥: Sum-test により、乱数に偏りが見られる。

Sum-Test: 連続する m 個の乱数の和 (sum) の、正規分布からのずれを計る統計的検定 (test)。

- $[0,1]$ 上の疑似乱数を m 個生成して和をとる (ほぼ正規分布する)
- N 回繰り返す (N をサンプルサイズと呼ぶ)
- N 個のサンプルの分布の、正規分布からのずれを χ^2 -検定する

現 C 言語標準乱数 random:

- $p = 31, q = 3, M = 2^{31}$.
- Period $\sim 2^{62}$.

Sum-Test の結果:

χ^2 -検定の結果 (確率値、三回)

0.9905	0.9435	0.9309
--------	--------	--------

サンプルサイズ $N = 2 \times 10^7$,
和の項数 $m = 50$.

- サンプルサイズを大きくしていくと、検定結果はどんどん悪くなる
- どのくらいサンプルサイズを大きくすると確率値がどのくらいになるか、計算する方法がある
(Levy の反転公式という Fourier 反転公式を使う、松本-西村'01)

5. Knuthの最新乱数：ran_array

ran_array: Lagged Fibonacci の改良 (Lüscher).

- Knuth '97.

- $p = 100, q = 63, M = 2^{30},$

$$x_{n+100} := x_{n+63} - x_n \bmod 2^{30}.$$

- 周期 $\sim 2^{129}$.
- L 個乱数を生成, (L :Luxury Level と呼ぶ整数)
「100 個乱数を使い $L-100$ 個捨てる」を繰り返す。
- $L \leq 200$ では乱数性が悪い。簡単なコイン投げで
どんどん儲けられる。

(Knuth :「この種の注意はほとんど必要ないが、 $L \sim 1000$ なら安全である」)

コインテスト:

$x_j = \text{偶数、奇数} \Leftrightarrow \text{コインの表、裏}$
により疑似乱数でコイン投げをシミュレーション。

戦略:

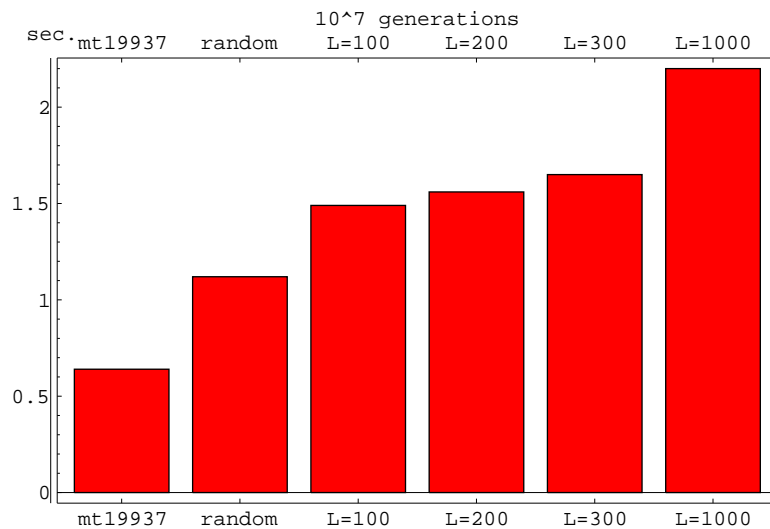
1. 100 回のコイン投げの結果を観察。
2. もし 60 回以上表だったら、次は表に 1 ドル賭ける。
もし 59 回以下だったら、賭けない。
3. これを 100 万回繰り返す。

ran_arrayによるコイン投げでの金の儲かりかた

L :Luxury	勝ち回数	負け回数	儲かったお金	確率
100	14740	13594	1146	5.11816×10^{-12}
200	14527	13955	572	0.000357862
300	14456	14359	97	0.285854
400	14440	14335	85	0.269908
MT521	14250	14322	-72	0.667081

ran_array(Luxury = 100, 200, 300, 400)によるコイン投げ
に対し、上の戦略で儲かった金とそれだけ儲かる確率

速度の比較



mt19937, random, および ran_array
($L = 100, 200, 300, 1000$) の速度比較

Knuth とのやりとり (97年秋)

計算機科学のバイブルと言われる Knuth の名著
“The art of computer programming” 第3版 (97年末
出版)

をめぐって交わされたやりとり

「ホットケーキとキャベツと旧約聖書」

松本 疑似乱数に『Luxury Level』が選択できるのは
良くない。ユーザーが混乱する。

デザイナーが責任もって指定すべきだ。

Knuth アメリカにはホットケーキパウダーがある。
これは使うときに卵を混ぜる。

...本当は、アメリカの技術では生卵と全く区別の
つかない『卵の粉』を作ることでもでき、それを混
ぜておけば済むことだがそうはしない。なぜか。
ユーザーは、生卵を自分の手で入れたいのだ。
その方がおいしい気がする。

このように、製作責任の一端をユーザーが担うと
いう考えは、気にいっている。

松本 大部分を捨てれば使える、というのは
ホットケーキではないだろう。

「ここに、くさったキャベツがある。

大部分を捨てれば、食べられる。

捨てる場所を増やすほど *luxual* だ。」

というのがあなたの主張だ。

だが、安くて新鮮なキャベツが売られているのだ。
なぜわざわざくさったキャベツを選ぶのか。

この本は『計算機科学のバイブル』と呼ばれる
高名な本だ。そこでこういう方法を推奨すれば
みんなそれに従ってしまう。あなたの責任は重大
だ。

Knuth 私の本を『バイブル』と呼びたい人は、

『Old Testament(旧約聖書)』と呼ぶべきであろう。

書かれたアルゴリズムは古く、乱数以外にもっと
緊急に書きなおすべき個所がふんだんにあるので、
そちらを優先する。

6. Mersenne Twister に到るまで

$\mathbb{F}_2 = \{0, 1\}$: 2元体上の線形代数を使う
($1 + 1 = 0$ の約束で四則演算を行う)

GFSR (Lewis-Payne '73) 次の漸化式によって
 w ビットの横ベクトル列 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ を生成する

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

各ビット間の情報が混じらず、周期は高々 $2^p - 1$ 。

Twisted GFSR (松本-栗田'92、'94)

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

A はツイスターと呼ばれる、 $w \times w$ 正則行列
(companion 行列という掛け算が極めて速い形)
ビット間の情報が混じり、周期 $2^{pw} - 1$ を実現

Mersenne Twister (松本-西村'97)

$$\mathbf{x}_{n+p} := \mathbf{x}_{n+q} + \mathbf{x}_{n+1}B + \mathbf{x}_nA.$$

ここに、 B, A はある $w \times w$ 行列で、状態空間のサイズをメルセンヌ冪 $P (\leq pw)$ にするもの。

$2^P - 1$ が素数であることが効いて、周期が $2^P - 1$ であるかどうか高速に判定できる。

- 掛け算、割り算など遅い演算を完全排除
(bit 演算と和差演算のみで生成が進む)
- $p = 624$: キャッシュメモリに入るサイズを一度に生成

高速性の実現

- 周期 $2^{19937} - 1$ が数学的に証明できる
(\mathbb{F}_2 係数多項式の新既約判定アルゴリズム)
- 数の幾何 (1800-) の手法で、高次元分布が解析・最適化できる

高品質を数学的に保証

まとめ

数学的解析が可能であること、
計算機で高速にできることと、
のインタラクションに生まれたのがMT

Epilogue: internet success story

92年、94年 Twisted GFSR法（松本-栗田）発表

96年、Mersenne Twister法（松本-西村）発表

既存の方法の問題点（乱数性の悪さ）を述べ、それらを解決する方法を提示した

が、誰も使わない

既存の（欠陥のある）方法のマイナーチェンジが発表され、使われ、問題点が指摘され、またマイナーチェンジ、を繰り返している。

「その方法よりこの方法が優れている」といっても伝わらない

- ユーザー（言語のデザイナーを含め）まで声が届かない
（届く声は「偉い人」の声, Knuth とのやりとり)
- 有限体など、背景にある理論を知らないとだめ
- 数学的証拠を見せてもだめ

インターネット・サクセス

MT 普及の障害

- 偉い先生の作る風評
- 分野の壁=ユーザーに声がとどかない
(物理、金融の人は乱数研究者の言に余り注意を払わない)

Break Through=インターネット

- 1997年5月 Bonn Max Planck Institut に滞在中、ザルツブルグ大学のホームページ pLab でニュースとして取り上げられる
- 1997年10月 慶応大学理工学部所属のころ、朝日新聞夕刊で大きく報じられる
- 電話での問い合わせが大学に殺到
MTのホームページ作成
英語版も作り、ダウンロード可能にする

- 毎日世界中から問い合わせあり：
 - カーネギーメロン大学コモンリスプの標準疑似乱数に採用
 - プラズマシミュレーションHAWKに採用
 - フォートラン版、パスカル版など、多くの言語に翻訳
 - JIS規格の標準乱数の一つに採用
 - 富士通計算処理システムAZIRの乱数に採用
 - 多くの金融機関が採用

欠けていたもの =

インターネットというメディア
個人から個人・世界へ壁を超えて正確に情報発信
できる
(ダウンロードダブル、各自比較実験可能)

どんなに良いものを作っても、うまく宣伝しなければ使ってもらえない

でも、そもそも「良いもの」がなければ意味がない

流通（メディア）も大切だが、
生産（例：数学・情報科学の基礎研究）も大切