疑似乱数発生に用いられる数学：
メルセンヌ・ツイスターを例に

松本　眞（東京大学数理科学研究科）

2010/4/23, 東京大学数理科学研究科談話会

matumoto “at mark と呼ばれるもの” ms.u-tokyo.ac.jp

This study is supported in part by JSPS Grant-In-Aid #16204002, #18654021, #19204002, #21654027 and JSPS Core-to-Core Program No.18005.

1. 乱数：なんに使うの？

- **モンテカルロ法:**

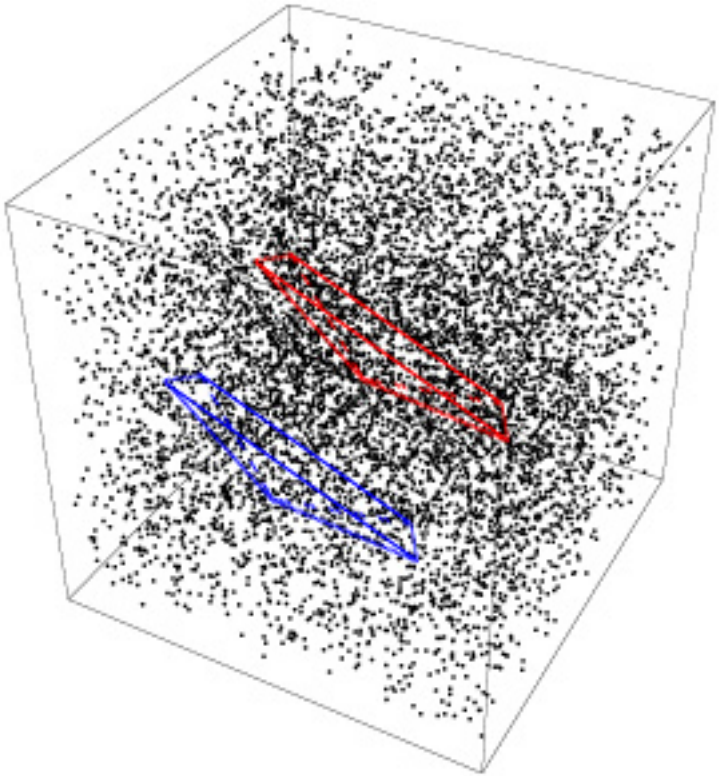
  例：ある領域の体積の近似

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

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

  (真の体積: $8.333\cdots \times 10^{-3}$)

  右図は3次元中の多面体だが
  もっと高次元・複雑な図形に
  モンテカルロ法が有効



3次元単位立方体に、でたらめにN個点を打つには：

1. 区間$[0,1] := \{x \mid 0 \leq x \leq 1\}$内から、
   一様にでたらめに実数xを取ってくる
2. 同様に、区間$[0,1]$から
   一様に独立にでたらめに実数y, zを取ってくる
3. 座標(x, y, z)の点を立方体に打つ
4. 1-3をN回繰り返す

定義(あいまい)

- $[0,1]$区間の中から、
  「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。
- 得られた数列を乱数列と呼ぶ。

注：
$[0,1]$区間に限らず、サイコロのように
「$\{1,2,3,4,5,6\}$ の中からでたらめな数を発生させる方法」
も乱数発生法という。

- **モンテカルロ法（つづき）**

  確率的現象を含んだあらゆる現象のシミュレーションに
  乱数は必要である。これらもモンテカルロ法と呼ばれる。
  （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）

- **暗号への応用** (略)
- **確率的アルゴリズム** (略)

2. 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、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

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

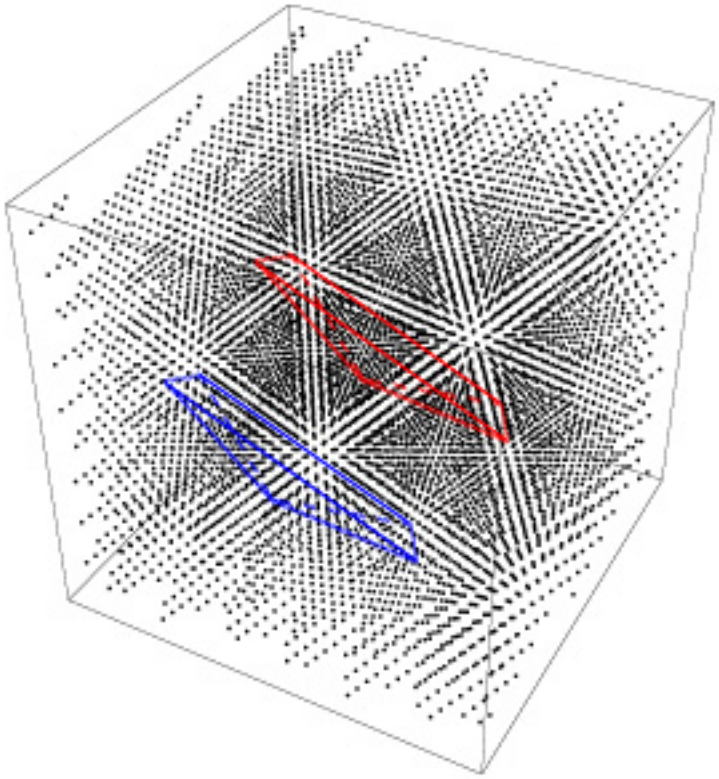
擬似乱数のメリット:

- 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」...⇐（実用的）定義の不在
疑似乱数の創始者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:

- 先の線形合同法による数列の周期は、
  初期シードの選び方によらず2^{32}。
- この生成法は、70年代から80年代にかけて
  ANSI-Cなどの標準擬似乱数であった。
- 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に見えるが、
  数千万個の出力を使うと、非乱数性が現れてくる

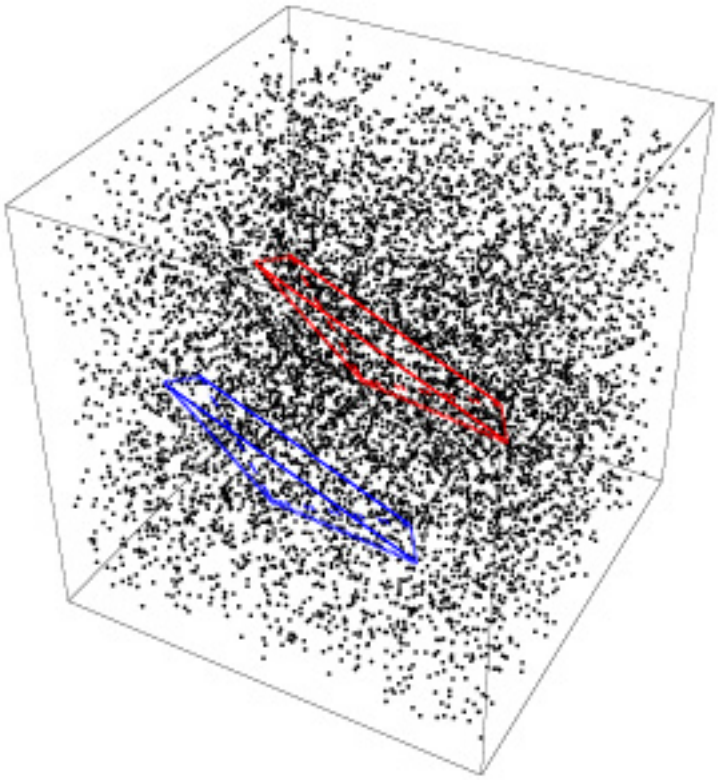


この線形合同法による2^{32}個の
全周期3次元ランダム点プロット
（70倍拡大図）

赤領域体積$\simeq \frac{62}{7253} = 8.5\times10^{-3}$.

青領域体積$\simeq \frac{45}{7253} = 6.2\times10^{-3}$.

(真の値: $8.333\cdots\times10^{-3}$)



Mersenne Twister(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: $8.333\cdots \times 10^{-3}$)

3. 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

- **乱数性**

  擬似乱数は、所詮漸化式による数列
  「乱数性」の十分に実用的な（実証可能な）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

4. 循環小数と線形合同法　$c = 0$のとき、線形合同法

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

は、「$x_1 \div M$のa進小数展開に現れる余りの列」である。

例$x_1 = 1, 1 \div 7$の10進小数展開

$1 \times 10 = 10, \div 7 = 1$ 余り $3 =: x_2$

$3 \times 10 = 30, \div 7 = 4$ 余り $2 =: x_3$

$2 \times 10 = 20, \div 7 = 2$ 余り $6 =: x_4$

$6 \times 10 = 60, \div 7 = 8$ 余り $4 =: x_5$

$4 \times 10 = 40, \div 7 = 5$ 余り $5 =: x_6$

$5 \times 10 = 50, \div 7 = 7$ 余り $1 =: x_7$

$$0.\dot{1}4285\dot{7}1$$

$$7\overline{)10}$$

7

30

28

20

14

60

56

40

35

50

49

10

7

30

5. 整数から$\mathbb{F}_2$多項式へ：

$\mathbb{F}_2$多項式割り算での循環小数が有用

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）。

1+1=0から1を移項して

$$1 = -1.$$

$\mathbb{F}_2$多項式$\mathbb{F}_2[t]$

$$\mathbb{F}_2[t] := \{\sum_{i=0}^{n} a_i t^i | a_i \in \mathbb{F}_2,\ n \in \mathbb{N}\}$$

を考える。掛け算・足し算は通常の多項式同様

$$\begin{aligned}(t+1) \times (t+1) &= t(t+1) + 1(t+1) \\ &= t^2 + t + t + 1 \ = \ t^2 + 1\end{aligned}$$

といった具合に計算できる。

（$1+1=0$より$t+t=(1+1)t=0$。）

係数のみを表記することにして、「t進くらい取り」で

$$t^3 + t^2 + 1 = 1t^3 + 1t^2 + 0t + 1 = 1101$$

と表わすことにする。

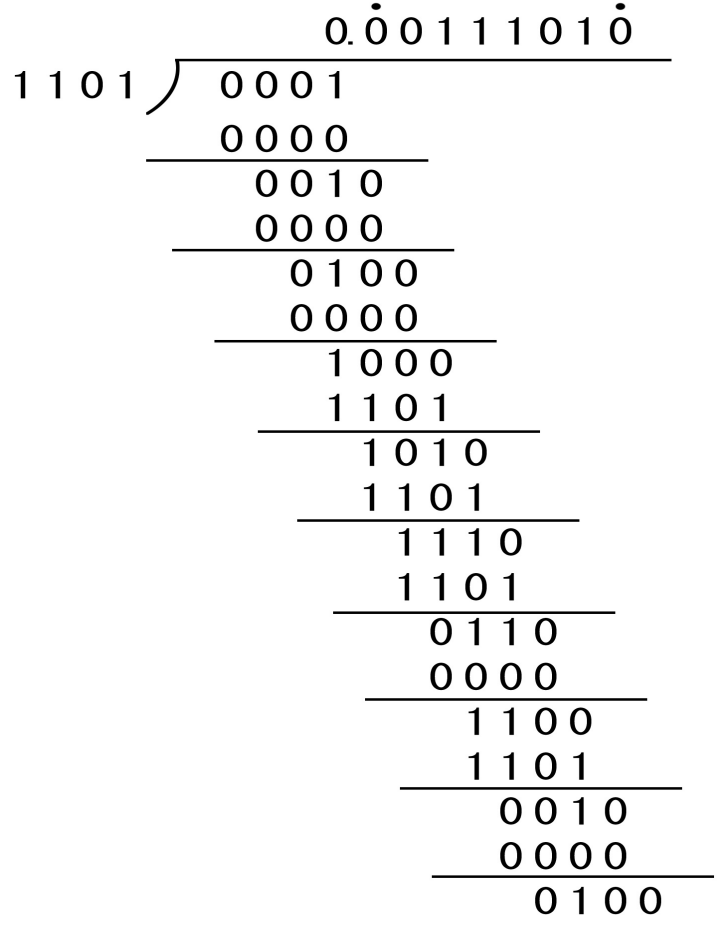
$\mathbb{F}_2$多項式での和差積商は、

「繰り上がり・繰り下がりのない世界での計算」

になる。

$$\begin{array}{rrr} & 1 & 1 \\ \times & 1 & 1 \\ \hline & 1 & 1 \\ 1 & 1 & \\ \hline 1 & 0 & 1 \end{array} \qquad \begin{array}{rrr} & t & +1 \\ \times & t & +1 \\ \hline & t & +1 \\ t^2 & +t & \\ \hline t^2 & +0t & +1 \end{array}$$

$\mathbb{F}_2[t]$ の世界で $1 \div (t^3 + t^2 + 1) = 1 \div 1101$ を小数展開すると



検算：

$$
\begin{array}{r}
0.00111010011101\cdots \\
\times\ 1101 \\
\hline
0.00111010011101\cdots \\
00.00000000000000\cdots \\
000.11101001110100\cdots \\
0001.11010011101001\cdots \\
\hline
0001.00000000000000\cdots
\end{array}
$$

$1 \div (t^3 + t^2 + 1) =$

$0 + 0t^{-1} + 0t^{-2} + 1t^{-3} + 1t^{-4} + 1t^{-5} + 0t^{-6} + 1t^{-7} \cdots$

$1 \div 1101 = 0.x_1x_2x_3x_4\cdots$ とおくと $\mathbb{F}_2$ での漸化式

$$x_{n+3} + x_{n+2} + x_n = 0 \quad (n \geq 1) \quad \text{を満たす}$$

検算：

$$\begin{array}{rrrrcccccc}
 & & & 0 & . & x_1 & x_2 & x_3 & x_4 & x_5 & \cdots \\
\times & 1 & 1 & 0 & 1 & & & & & & \\
\hline
 & & & 0 & . & x_1 & x_2 & x_3 & x_4 & x_5 & \cdots \\
 & & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & \cdots \\
 & 0 & x_1 & x_2 & . & x_3 & x_4 & x_5 & x_6 & x_7 & \cdots \\
0 & x_1 & x_2 & x_3 & . & x_4 & x_5 & x_6 & x_7 & x_8 & \cdots \\
\hline
0 & 0 & 0 & 1 & . & 0 & 0 & 0 & 0 & 0 & \cdots
\end{array}$$

同じ理屈で、$1 \div (t^{607} + t^{273} + 1)$を小数展開したものを
$0.x_1x_2x_3x_4\cdots$とおくと漸化式

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

を満たす。逆に、この式を使ってx_nが求められる。

（実は、この漸化式による0-1列の周期は$2^{607} - 1$となる。）

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

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

- 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）

周期保証に現れる初等数論

Proposition 1

次数pの$\mathbb{F}_2$多項式$\varphi(t)$に付随する漸化式の周期$\leq 2^p - 1$.

上限達成$\Leftrightarrow$ tの$\mathbb{F}_2[t]/(\varphi(t))$での乗法位数$= 2^p - 1$.

特に$2^p - 1$が素数（この形の素数をMersenne素数という）で
tが$\varphi(t)$を割り切らないならば、$\equiv$で　$\mod \varphi(t)$を表して

$$g_0(t) := t, g_1(t) :\equiv g_0(t)^2, \ldots g_p(t) :\equiv g_{p-1}(t)^2 \equiv t^{2^p}$$

を計算して、$g_0 = g_p$となることが最大周期の必要十分条件。
$\Rightarrow$ pがMersenne 指数なら周期チェックは高速

6. ベクトル化：$\mathbb{F}_2$多項式から$\mathbb{F}_2$線形変換へ

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$1101 = 0100 + 1001$

整定数p, qをうまく選ぶと周期$2^p - 1$にできる

- 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）

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: $2^{32p} - 1$が達成可能(32:ワード長)

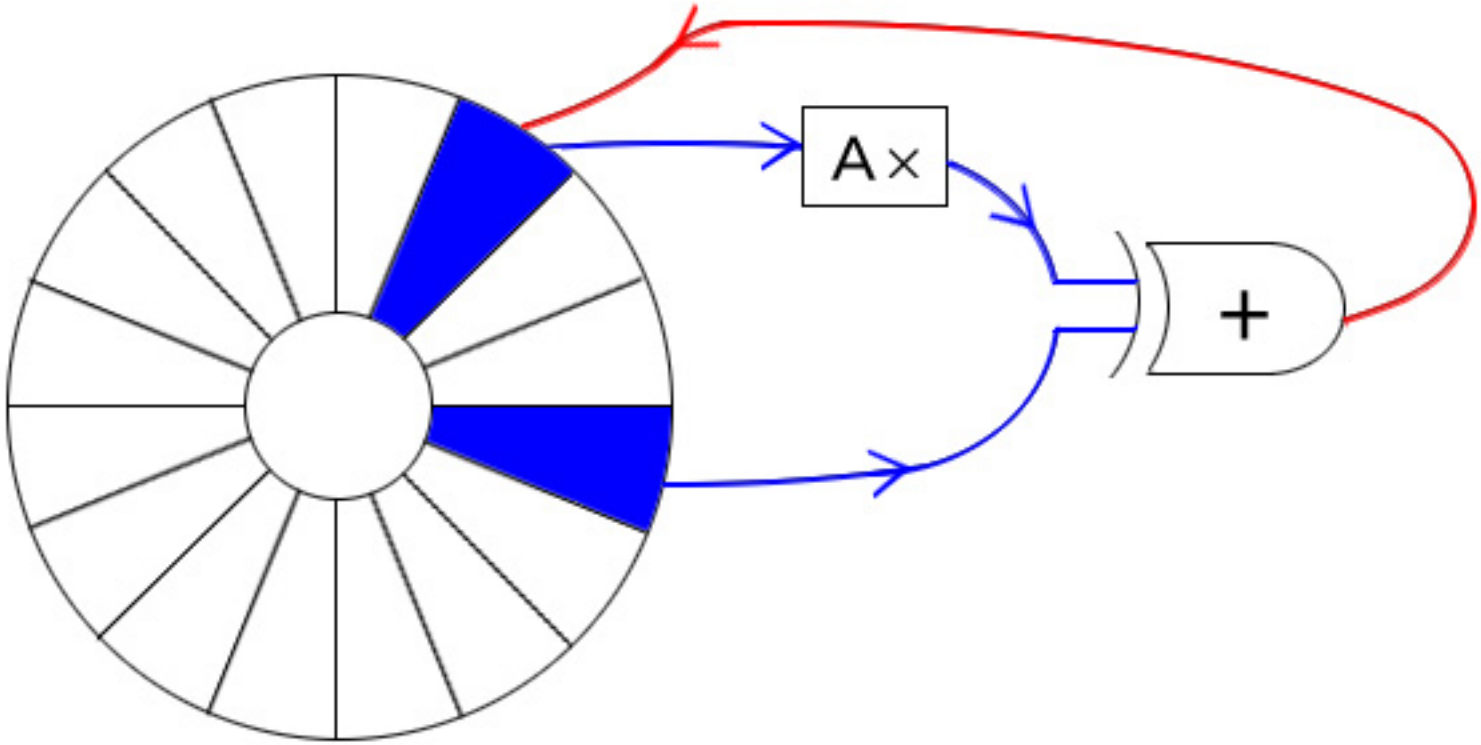
- Aは次のようなものを選ぶ:定数ベクトル$\vec{a}$により

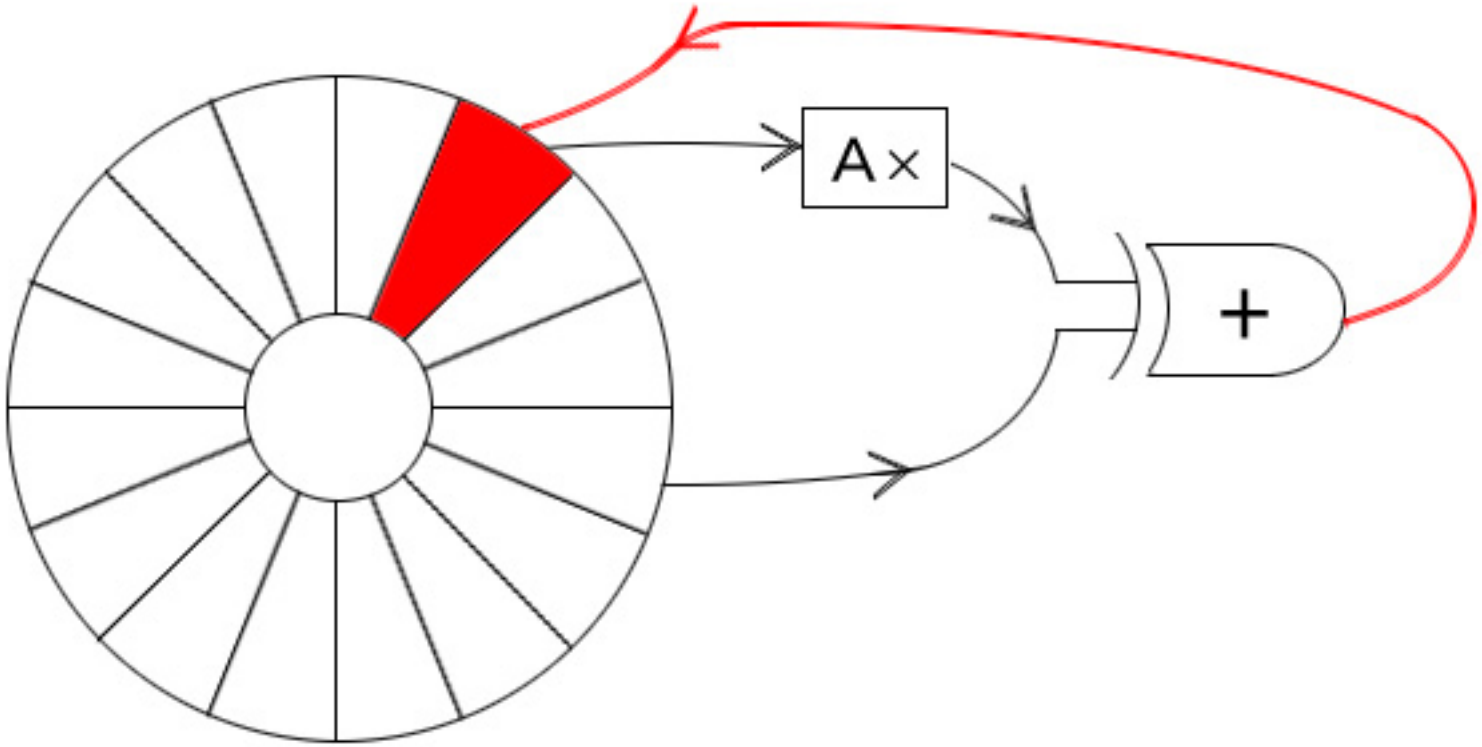
$$\vec{x}A = \begin{cases} \text{shiftright}(\vec{x}) & (\vec{x}\text{の最下位ビットが}0\text{の場合}) \\ \text{shiftright}(\vec{x}) + \vec{a} & (\vec{x}\text{の最下位ビットが}1\text{の場合}) \end{cases}$$

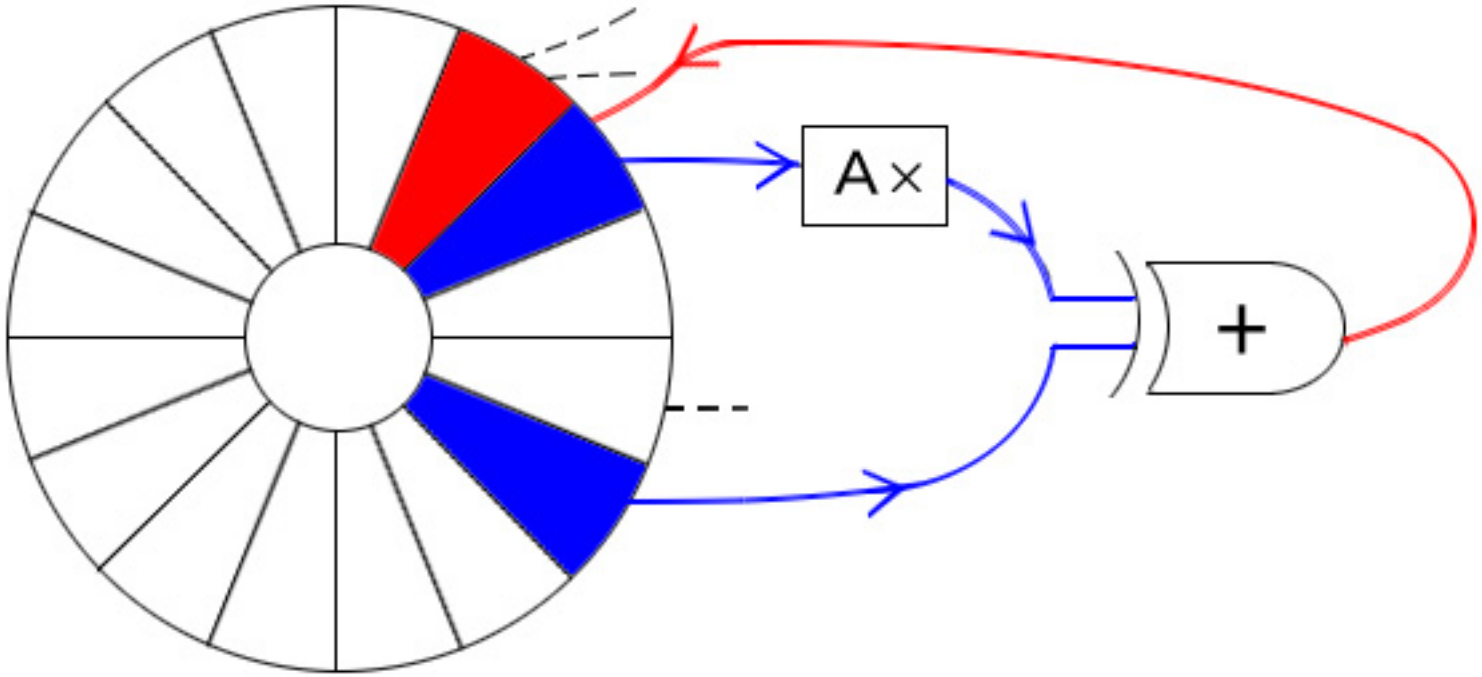
  $\Rightarrow$高速に計算可能, 十分一般: $A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix}$.

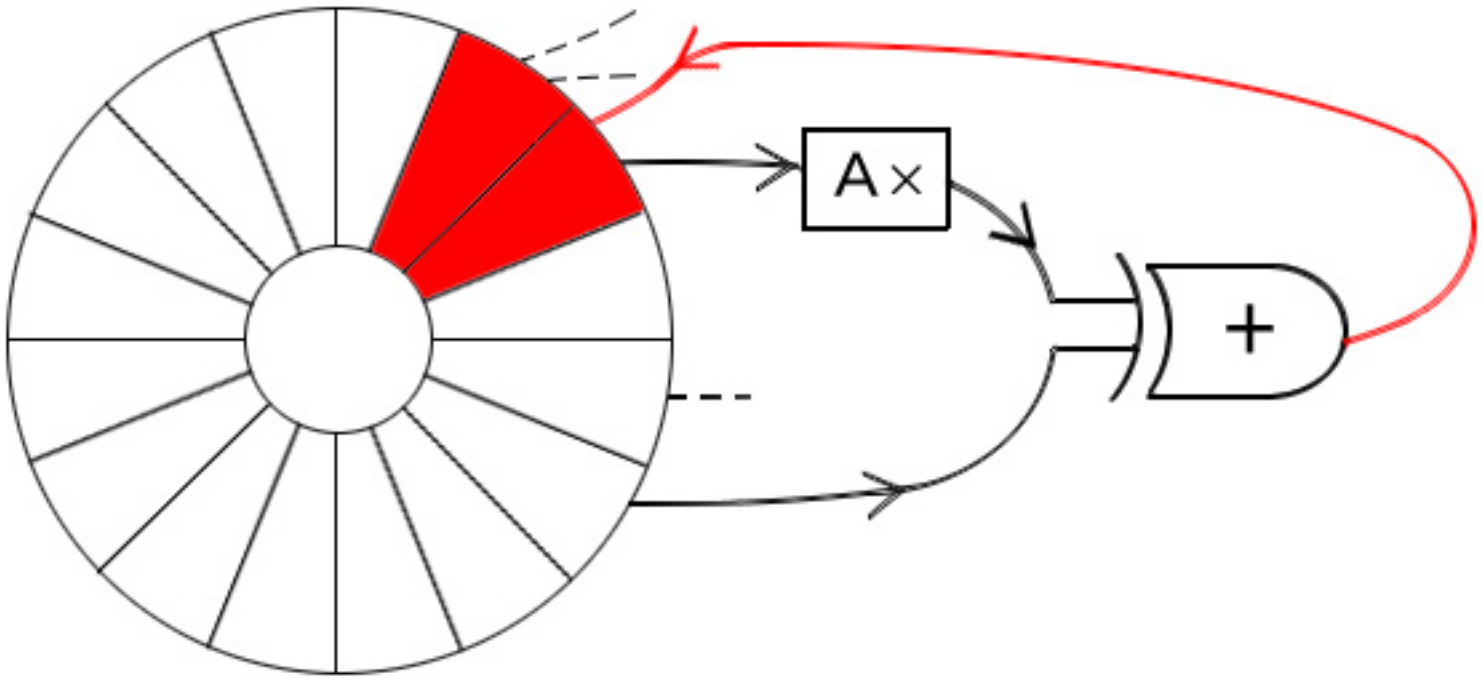
- 高次元均等分布性を改善するため$\vec{x}_n T$を出力('94)

なぜMT(TGFSR)は高速か？⇐ラウンドロビン実装




A×
+


A×
+


A×
+

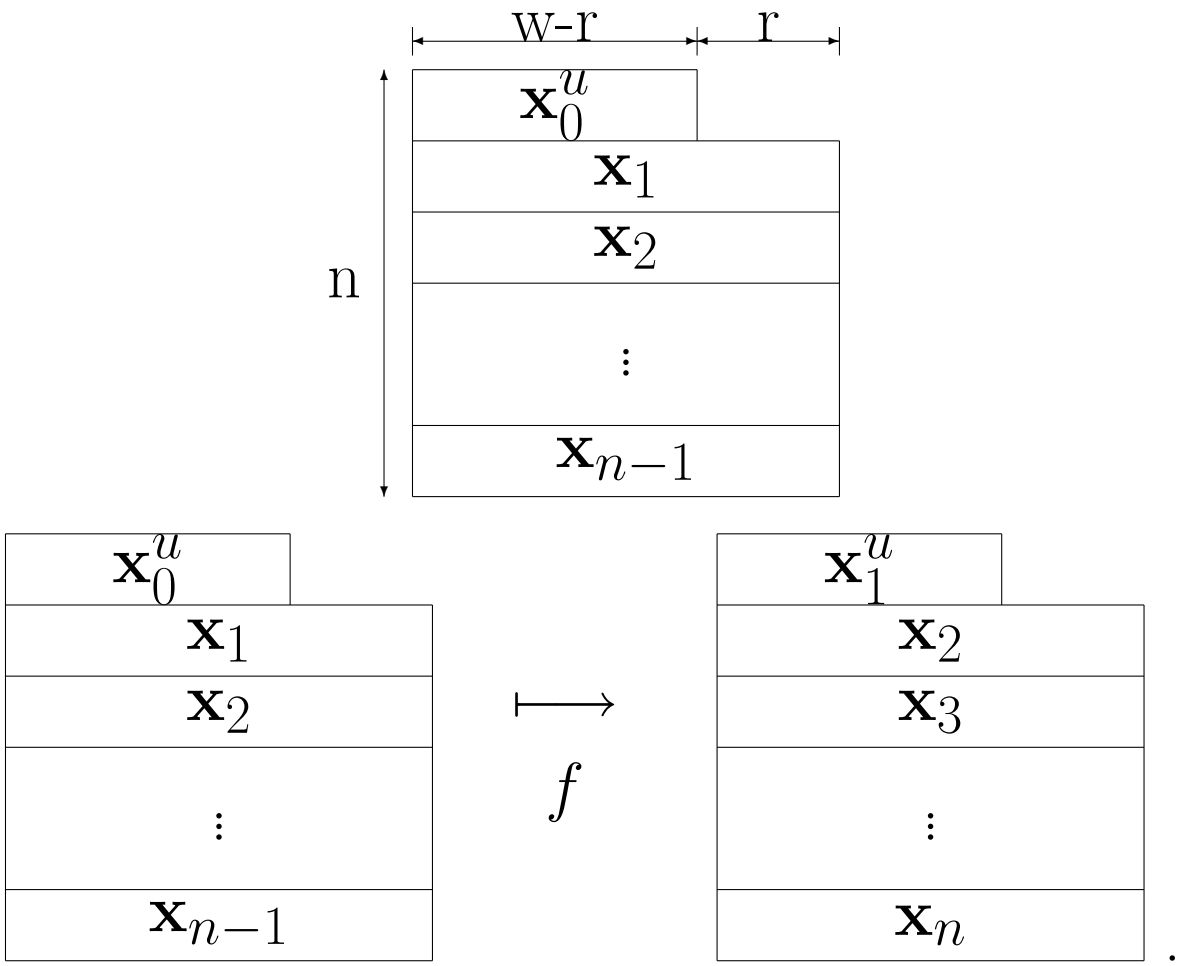
Mersenne Twister: (松本-西村拓士 '98):
TGFSRの周期を大きな素数$2^{32p-r}-1$の形にするため、

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

と二個行列を導入、状態空間$32p$次元中r次元を「使わない」
$\Rightarrow$ 素数周期の判定は容易なため、
超長周期が実現可能: $2^{624\times 32-31}-1 = 2^{19937}-1$.

注：19937は24番目のメルセンヌ指数、現在47個知られており2010年3月時点で知られている最大の指数は43,112,609
未解決問題：無限個あるか？

Mersenne Twisterの状態配列と遷移



7. Mersenne Twisterの性能１:周期保証の数学

- 生成速度は、近年の線形合同法（ $\bmod 2^{48}$）よりも高速
- 周期：$2^{19937}-1$ (初期値$\neq 0$なら）

次状態関数fの特性多項式$\varphi(t)$さえ求めれば、前述の方法で周期が最大か否か確かめられる。
が、実は$\varphi(t)$を求めるだけでも工夫しないと時間がかかりすぎる：

- サイズpの行列式$\chi(t)=\det(tI_n-F)$の展開は$O(p^3)$回の多項式演算を使う：$p=19937$では遅すぎる

  $\Rightarrow$ Berlekamp-Massey法('68)という

  数列から特性多項式を求めるアルゴリズムの利用

  ($\mathbb{F}_2((t^{-1}))$における$\mathbb{F}_2[t]$係数互除法)

- Berlekamp-Massey法の概説
  - （Euclid）既知の実数xが未知の既約分数

    $$x = b/N$$

    となるはずの場合、「$x = b/N$と1の整数係数互除法」の進み方は「bとNの互除法」の進み方と同じ、停止したらbとNが分かる。
  - (BM)数列を母関数とする形式べき級数$x(t)$が未知の既約分数式

    $$x(t) = b(t)/\varphi(t)$$

    となるはずの場合、「$x(t)$と1の多項式係数互除法」の進みかたで$b(t), \varphi(t)$が分かる。
  - $x(t)$を有限項で打ち切っても、項数が$2\deg(\varphi(t)) - 1$以上あれば正確に$\varphi(t)$がもとまる。

8 Mersenne-Twisterの性能２：高次元均等分布性

定義（あらまし）
擬似乱数列がk次元均等分布しているとは、
出力列をk個ずつに区切ってk次元立方体内の点を
一周期に渡り発生したとき、均等に分布していること

定義（正確）
有限集合Xに値をとる擬似乱数列が
「k次元均等分布している」とは、
「k個ずつ組にしてX^k内の点を一周期に渡り発生すると
X^k内のどの元も同数回ずつ現れる」こと

例：周期８の１ビット(0,1)列

$$000101110001011\cdots$$

連続3組を一周期に渡り並べると

$$000, 001, 010, 101, 011, 111, 110, 100,$$

となり、可能な８パターンを全て一度ずつとる
$\Rightarrow$ 3次元均等分布している

例：周期16の2ビット(0,1,2,3)列

$$00102031121322330010203112 13\cdots$$

$$00, 01, 10, 02, 20, 03, 31, 11, 12, 21, 13, 32, 22, 23, 33, 30$$

$\Rightarrow$ 2次元均等分布している

高次元均等分布性の意義:

経験的には大抵：

均等分布の次元　＞
シミュレーションに現れる多変数関数の変数の個数の最大値

⇓

シミュレーションの結果は正しい.

MTの場合は

- 32ビット整数列として623次元均等分布 $(X = \mathbb{F}_2^{32})$
- 高位ビットはさらに高次元
  （例えば上位3ビットは6240次元均等分布:$X = \mathbb{F}_2^{3}$）

注:

- MT以前はせいぜい20次元均等分布程度
- 実用上は、そこまで高次元均等分布していなくても
  大抵大丈夫

均等分布の次元の計算法

準備：$\mathbb{F}_2$線形疑似乱数発生法のモデル：

- 状態空間と呼ばれる$\mathbb{F}_2$-線形空間S
- 次状態関数$f : S \to S$
- 出力集合と呼ばれる$\mathbb{F}_2$-線形空間O
- 出力関数$o : S \to O$（f, oの$\mathbb{F}_2$-線形をここでは仮定）

初期値s_0に対し、

$$s_1 := f(s_0), s_2 := f(s_1), \ldots$$

と状態を変化させつつ、

$$o(s_0), o(s_1), o(s_2), \ldots$$

を出力列とする。

状態遷移が周期最大（0以外の状態が一軌道）の仮定のもと、
k-次元均等分布性$\Leftrightarrow$次の線形写像の全射性:

$$o^{(k)} : S \to O^k, \quad s \mapsto (o(s), o(f(s)), \ldots, o(f^{k-1}(s))).$$

$\because$この写像は、状態sからの相続くk個の出力の組を見るという線形写像。

- 応用上、$O = \mathbb{F}_2^v$で、oは「32ビット出力の上位vビットを見る」写像$(1 \leq v \leq 32)$.
  このとき、これが全射になる最大のkを
  「vビット精度での均等分布の次元」といい、$k(v)$であらわす。
- 次元比較$\Rightarrow$自明な上限：$k(v) \leq \lfloor \dim(S)/v \rfloor$

$\mathbb{F}_2[t]$-格子の利用

線形写像の全射性を示す＝表現行列のランク計算 $O(p^3)$ は遅すぎる

（$\because$ よいものを探すために、ランダムサーチを何度もやる）

$\Rightarrow$ Couture-L'Ecuyer-Tezuka ('93) による「格子の利用」

格子法(Couture-L'Ecuyer-Tezuka '93).

$$F := \mathbb{F}_2((t^{-1}))$$
$$Z := \mathbb{F}_2[t] \subset F$$
$$I := t^{-1} \cdot \mathbb{F}_2[[t^{-1}]] \subset F$$

Fのウルトラノルムを$|t| := 2$で定義

Eg. $|1| = 1$, $|t^{-1}| = 1/2$,

$$|a_n t^n + a_{n-1} t^{n-1} + \cdots| = 2^n.$$

$F = Z \oplus I$ ($\mathbb{F}_2$-ベクトル空間として),

$$F \ni \underbrace{a_2 t^2 + a_1 t^1 + a_0}_{\in Z} + \underbrace{a_{-1} t^{-1} + a_{-2} t^{-2} + \cdots}_{\in I = B(0,1/2)}$$

これは$\mathbb{R} = \mathbb{Z} + [0, 1)$の類似。

$O = \mathbb{F}_2^v$ とおき、初期状態 $s \in S$ からの無限出力列を

$$\begin{array}{c}(x_{11}, x_{12}, \ldots, x_{1v}) \in \mathbb{F}_2^v \\ (x_{21}, x_{22}, \ldots, x_{2v}) \in \mathbb{F}_2^v \\ \vdots\end{array}$$

とし、そのベクトル値生成母関数を以下で定義：

$$o^{(\infty)}(s) := (\sum_{i=1}^{\infty} x_{i1}t^{-i}, \sum_{i=1}^{\infty} x_{i2}t^{-i}, \ldots, \sum_{i=1}^{\infty} x_{iv}t^{-i}) \in I^v.$$

$$I^v = \begin{array}{c}(*******)t^{-1} \\ (*******)t^{-2} \\ (*******)t^{-3} \\ (*******)t^{-4} \\ \vdots\end{array} \qquad t^{-k}I^v = \begin{array}{c}(0000000)t^{-1} \\ \vdots \\ (0000000)t^{-k} \leftarrow k\text{-th} \\ (*******)t^{-k-1} \\ \vdots\end{array}$$

すると連続k出力写像$o^{(k)} : S \to (\mathbb{F}_2^v)^k$は以下の合成：

$$o^{(k)} : S \overset{o^{(\infty)}}{\to} I^v \to I^v/(t^{-k}I^v) \overset{\text{linearly}}{\cong} (\mathbb{F}_2^v)^k.$$

したがって、（周期の最大性のもと）

k-次元均等分布

$\Updownarrow$ 証明済み

$o^{(k)}$の全射性

上で$\Updownarrow$ 準同型定理

$o^{(\infty)}(S) + t^{-k}I^v = I^v$

$\Updownarrow$幾何の目で

出力の離散集合＋半径2^{-k-1}の球 $=$ 半径2^{-1}の球.

$$o^{(\infty)}(S) + t^{-k} I^v = I^v$$
$$\Updownarrow \text{直和だから}$$
$$Z^v \oplus (o^{(\infty)}(S) + t^{-k} I^v) = Z^v \oplus I^v \quad (= F^v)$$

主張.

$$L := Z^v \oplus o^{(\infty)}(S) \subset F^v \text{ は } Z\text{-格子.}$$

$\because$ 閉じていること：$t \cdot o^{(\infty)}(s) \in o^{(\infty)}(S) + Z^v$ をいう.

$$\begin{aligned} t \cdot o^{(\infty)}(s) &= t \cdot \textstyle\sum_{i=1}^{\infty} \mathbf{x}_i t^{-i} \\ &= \textstyle\sum_{i=2}^{\infty} \mathbf{x}_i t^{-(i-1)} + \mathbf{x}_1 \cdot 1 \\ &= o^{(\infty)}(f(s)) + \mathbf{x}_1 \cdot 1 \in L. \end{aligned}$$

離散的：特性多項式$\varphi(t)$を掛ければ全成分Zにはいる。

k-次元均等分布

見た $\Leftrightarrow$ $L + t^{-k}I^v = F^v$

比較的簡単 $\Leftrightarrow_*$ L が Z-基底 B であって$|B| \leq 2^{-k}$なるものを持つ

ここで$|\cdot|$は$\sup$ノルム:

$|B| := \max\{|\mathbf{b}| \mid \mathbf{b} \in B\}$,

$|\mathbf{b}| := \max\{|b_1|, |b_2|, \ldots, |b_v|\}$ where $\mathbf{b} = (b_1, \ldots, b_v) \in F^v$.

$\because (\Leftarrow_*)$ $F^v = \langle B \rangle_F = \langle B \rangle_Z \oplus \langle B \rangle_I = L \oplus \langle B \rangle_I$.

したがって

$$F^v = L + t^{-k}I^v \Leftarrow \langle B \rangle_I \subset t^{-k}I^v \Leftrightarrow |B| \leq 2^{-k}. \quad \square$$

$\because$ ($\Rightarrow_*$, スケッチ) 仮定は
「F^v の任意の元が L の元で近似でき、
誤差は $t^{-k}I^v$ に入れられる。」

$B_0 :=$ 格子 $t^{-k}Z^v$ の標準基底

B_0 を $B' \subset L$ で近似する（誤差 $\in t^{-k}I^v$）

$\overset{\text{Ultra norm}}{\Rightarrow}$ $|B'| \leq 2^{-k}$, and B' は自動的に F 一次独立

$|B|$ を最小化する L の Z 基底は $|B| \leq |B'| \leq 2^{-k}$ を満たす。
□

格子法の要約

均等分布次元

$\Uparrow$

Lの最短基底B

$\Uparrow$

Lenstra's (or Lenstra-Lenstra-Lovasz) の基底簡約法

(Euclidean algorithmの高次元版.)

基底簡約法の高速化

- MTでは、格子点を状態ベクトルで表現することで格子簡約を高速化（従来は、$\varphi(t)$を掛けて多項式化、あるいは双対格子をとって多項式化していた)。
- MT以後:高速な格子簡約アルゴリズム(Mulders-Storjohann'03)に加え、v-ビット精度(F^vの格子最短基底）から

  $v-1$ビット精度の最短基底を求めるアルゴリズムの開発をした

  (原瀬晋-斎藤睦夫-M 2010、原瀬晋(in preparation))

  Couture-L'Ecuyer('00)と$k(1), \ldots, k(w)$を求める計算量が違う：

$$O(w^2 \dim(S)^2) \to O(w \dim(S)^2),$$

  全ての$k(v)$を求めるのに10倍以上高速になった。

9 MTに用いられている数学（その他）

- 0-1のバランスの計算(M-西村'02, '03)：
  符号理論におけるMacWilliams恒等式('77)の利用：
  離散フーリエ反転により、固定した長さの
  擬似乱数列出力の1-0の個数分布（高次元、計算困難）が
  数列の満たす関係式の1-0個数分布（低次元、計算可能）
  により求まる。

10 まとめ：数学の予期せぬ効用

- $1+1=0$ の数学の研究は、ガロア $(1830$ ごろ$)$ に遡る
- 当時は応用の見えなかった純粋数学が、
  現在実用されている。
- $\mathbb{F}_2[t]$ と整数は良く似ており、
  前者が扱いやすい（現代整数論の指導原理の一つ）

11 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。

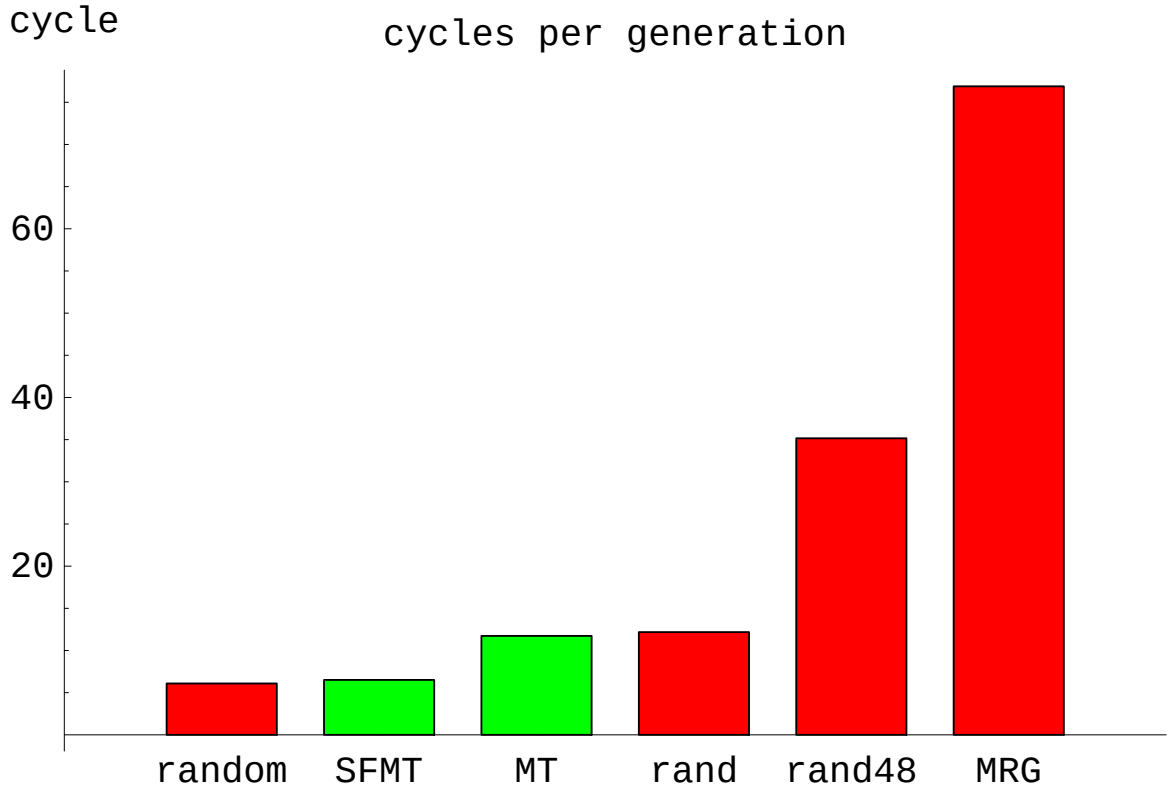
  SIMD命令やパイプライン処理を最大限に生かした
  漸化式を考案。

  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(mod 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。

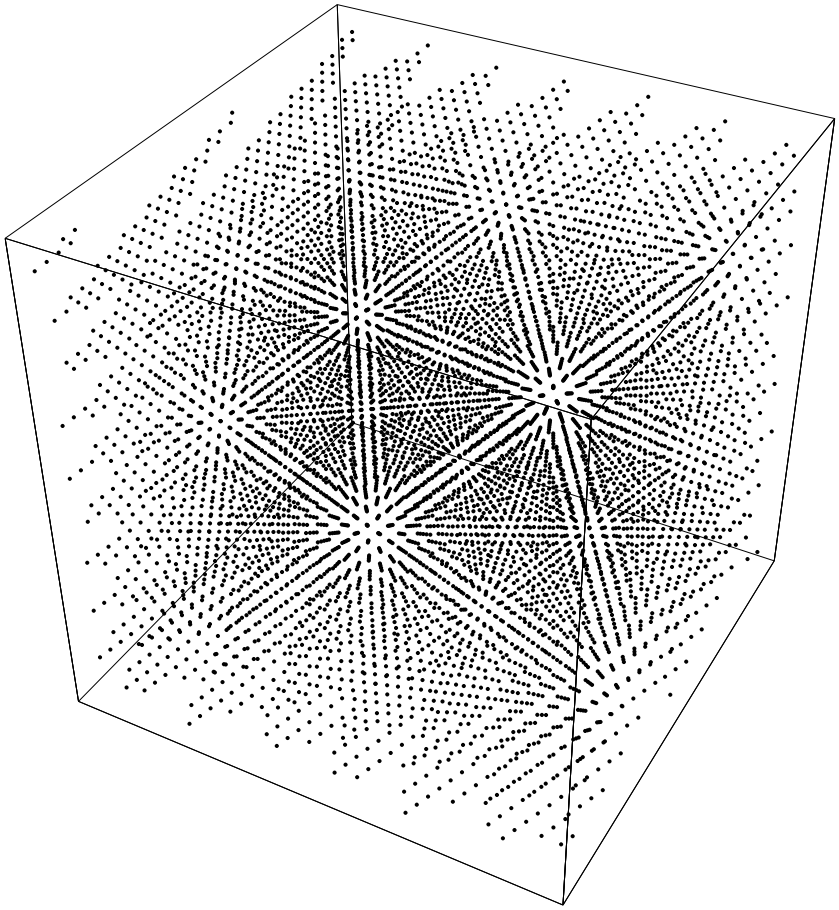
速度比較



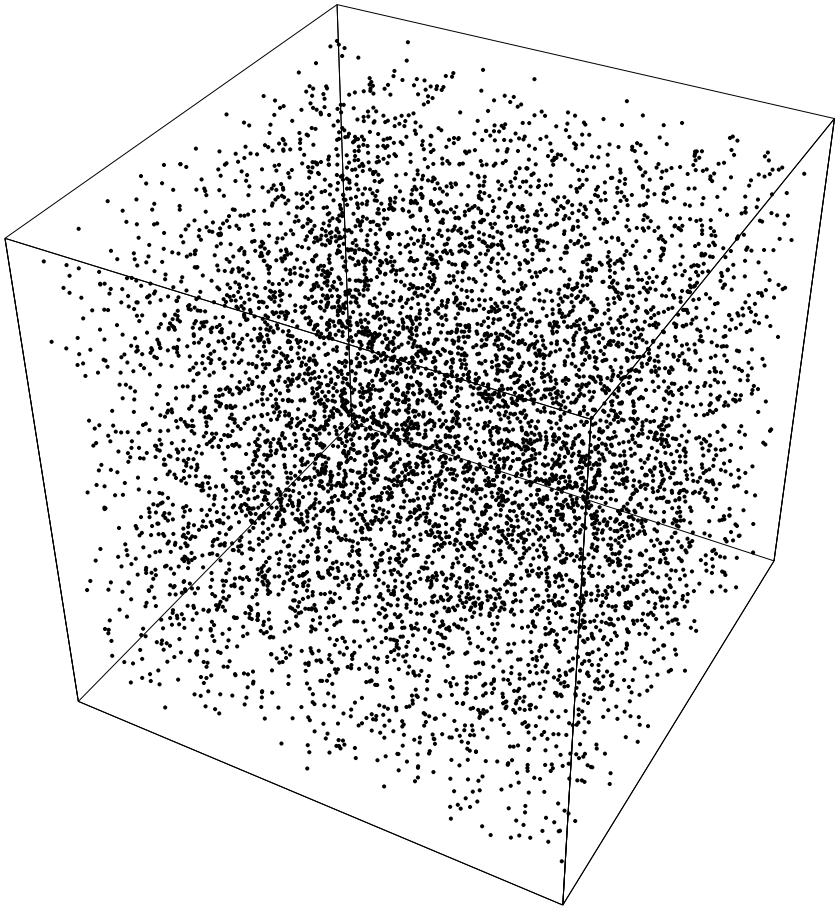
random: ラグ付き
フィボナッチ周期 $\sim 2^{63}$
rand: LCG 周期 2^{32}
SFMT SIMD Fast MT
MT: Mersenne Twister
　周期 $2^{19937} - 1$
rand48: LCG 周期 2^{48}
MRG: L'Ecuyer
　周期 $\sim 2^{186}$

12 終わりに：注意喚起

現在も、品質の悪い擬似乱数が広く使われている



ANSI-C標準擬似乱数('70-'90)



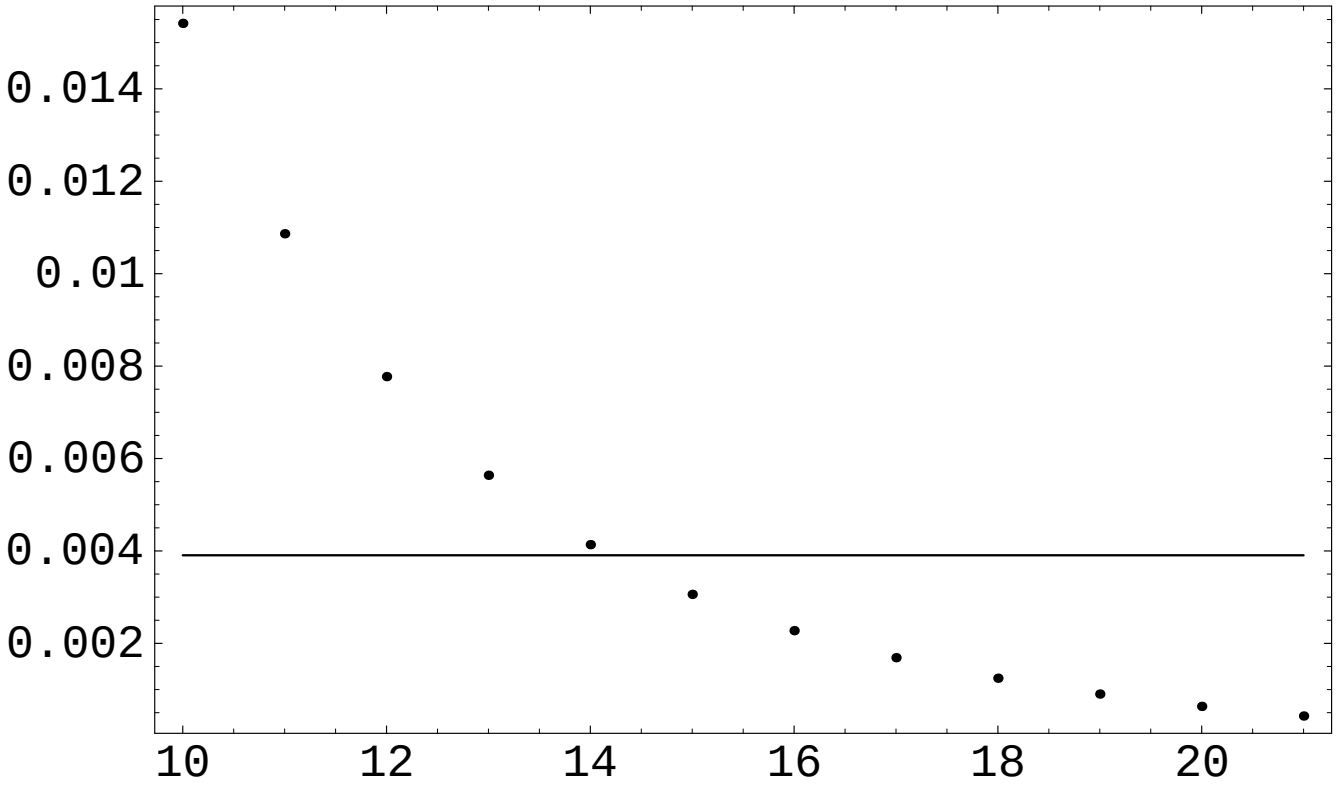
mt19937('98)

新しく提唱されたものの中にも悪いものが多い

`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)

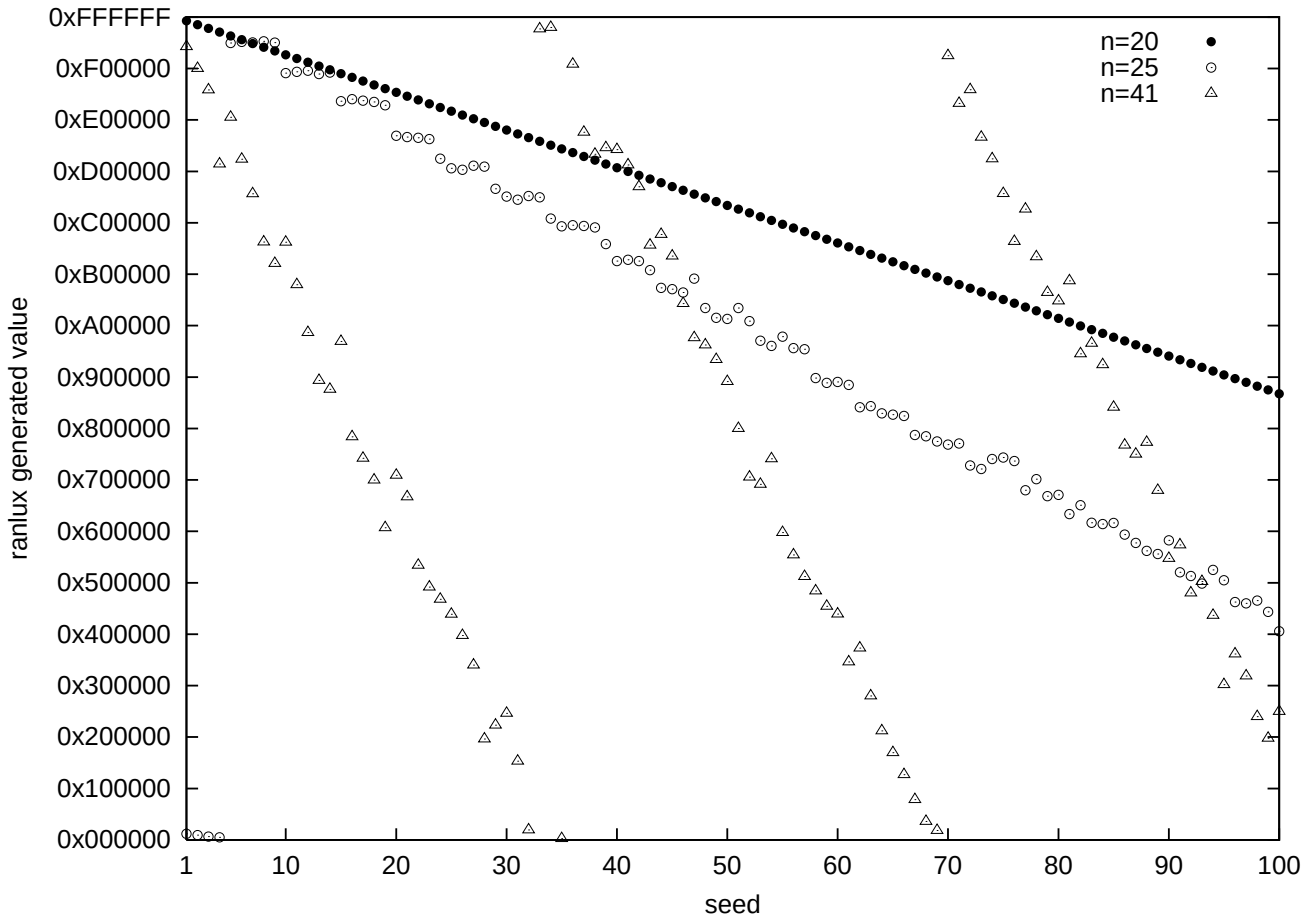
横軸：過去31回中の1の個数

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$1, 2, 3, \ldots, 100$（横軸）と動かしてプロット



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）

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。

擬似乱数研究の混迷

$\Leftarrow$ 理論的かつ実用的な「擬似乱数の定義」がないため

「擬似乱数の定義」へのまるで異なる３アプローチ：

1. 記録しておかない限り再生不能なものを乱数という
   (Kolmogorov-Chaitin, '60末)
2. 数列の一部から、他の部分が
   計算量的に計算できないものをいう (Blum-Blum-Shub, '86)
3. 周期や高次元分布性といった指標を用いて、
   良い漸化式を探す (古典的, '45-)

MT は古典的な3番だが、漸化式の「良さ」の評価に
$\mathbb{F}_2((t^{-1}))$ など現代数学の手法を用いた。

終わり