Generatory liczb losowych

- **Liczby losowe i pseudolosowe.**
- **Podstawy generowania liczb pseudolosowych.**
- **Podstawowe typy generatorów liczb (pseudo)losowych o rozkładzie równomiernym.**
  - ▷ **Generatory liniowe.**
  - ▷ **Generatory SR (oparte na rejestrach przesuwnych).**
  - ▷ **Generatory Fibonacciego.**
  - ▷ **Generatory SWB (oparte o odejmowanie z pożyczką).**
  - ▷ **Generatory MWC (oparte o mnożenie z przeniesieniem).**
  - ▷ **Generatory nieliniowe.**
- **Kombinacje generatorów.**
- **Generator RANLUX i *„operacyjna”* definicja losowości.**

$\Rightarrow$ http://th-www.if.uj.edu.pl/~placzek/dydaktyka/MMC/

Liczby losowe i pseudolosowe

“there is no such thing as a random number – there are only methods to produce random numbers”

John von Neumann

Liczba losowa – konkretna wartość przyjmowana przez zmienną losową (⟶ nieprzewidywalna!).

⇒ **Sekwencja liczb prawdziwie losowych** – nieprzewidywalna, a zatem niereprodukowalna!

- **Źródła liczb prawdziwie losowych – generatory fizyczne:**
  - ∗ **„mechaniczne”** – np. rzut monetą, losowanie z urny, ruletka itp.
  - ∗ **oparte o procesy fizyczne** – np. szum w urządzeniach elektronicznych (szczególnie tzw. szum biały), rozpad radioaktywny, promienie kosmiczne itp.

▶ **Wady generatorów fizycznych:**

- ∗ **zbyt wolne** dla typowych potrzeb obliczeniowych (szczególnie „mechaniczne”);
- ∗ **problemy ze stabilnością** – szczególnie generatory oparte o procesy fizyczne, np. niewielka zmiana warunków fizycznych źródła lub otoczenia może spowodować istotne zmiany własności probabilistycznych otrzymanych liczb

  ⟶ wymaga to dodatkowych urządzeń testujących i korygujących!

▷ Dawniej w użytku były **tablice liczb losowych** – niezbyt praktyczne!

⟶ Dziś wracają (?) – duże i tanie urządzenia pamięci masowej (twarde dyski, dyski CD/DVD itp.).

1995: G. Marsaglia, CD-ROM 650MB liczb losowych (pt. „White & Black Noise”): kompilacja szumu elektronicznego z muzyką rap.

Liczby losowe i pseudolosowe

Uwaga: Oferowane komercyjnie tzw. ,,fizyczne generatory liczb prawdziwie losowych" (najczęściej oparte o szumy elektroniczne) często nie spełniają prostych testów statystycznych!

⟶ Zanim zacznie się ich używać do poważnych obliczeń Monte Carlo należy poddać je gruntownym testom!

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

John von Neumann

Liczby pseudolosowe – liczby generowane według ścisłej formuły matematycznej (zatem reprodukowalne, a więc w ogóle nielosowe w sensie matematycznym), ale mające **,,wygląd" losowości**, tzn. ich własności statystyczne są bardzo bliskie własnościom liczb prawdziwie losowych (ktoś kto nie zna formuły, według której są generowane, nie powinien być w stanie stwierdzić, że nie są to liczby prawdziwie losowe).

► **Źródła liczb pseudolosowych – generatory matematyczne (programowe):**

∗ dobre własności statystyczne generowanych liczb,

∗ łatwość użycia (proste, szybkie, wygodne, . . .).

⟶ Wyparły prawie zupełnie generatory fizyczne!

Dlatego też liczby pseudolosowe są nazywane po prostu **liczbami losowymi** (ang. *random numbers*), a matematyczne algorytmy do ich otrzymywania nazywane są **generatorami liczb losowych** (ang. *random number generators – RNG*).

- **Pierwszy generator matematyczny: generator „środka kwadratu" Johna von Neumanna:**

  → Formuła: $$X_n = \lfloor X_{n-1}^2 \cdot 10^{-m} \rfloor - \lfloor X_{n-1}^2 \cdot 10^{-3m} \rfloor \cdot 10^{2m}$$

  gdzie: X_i, m – liczby naturalne, X_0 – stała, $\lfloor \cdot \rfloor$ – obcięcie do liczby całkowitej.

▶ Schemat działania – przykład:

Niech: $m = 2$, $X_0 = 2061$,

$\Rightarrow$ X_0^2 – liczba 8-cyfrowa (po ewentualnym dopisaniu zer po lewej stronie):

$$X_0^2 = \underbrace{04}_{\text{odrzucamy}}\ 2477\ \underbrace{21}_{\text{odrzucamy}} \qquad \Rightarrow\ X_1 = 2477,$$

$$X_1^2 = \underbrace{06}_{\text{odrzucamy}}\ 1355\ \underbrace{29}_{\text{odrzucamy}} \qquad \Rightarrow\ X_2 = 1355,$$

itd.

▶ Generuje $2m$-cyfrowe sekwencje liczb – niestety krótkie i w dodatku silnie zależne od X_0!

▷ Np. dla $m = 1$ możemy otrzymać następujące ciągi:

(1) $25,\ 62,\ 84,\ 05,\ 02,\ 00,\ 00,\ \ldots$ ← 6 różnych liczb;

(2) $47,\ 20,\ 40,\ 60,\ 60,\ \ldots$ ← 4 różne liczby.

(3) $50,\ 50,\ \ldots$ ← tylko 1 liczba!

Podstawy generowania liczb pseudolosowych

- **Typowy schemat generatora liczb losowych:**

  1) Wybieramy początkowe stałe: $X_0, X_1, \ldots, X_{k-1}$.

  2) Jeżeli wygenerowaliśmy już $(n-1)$ liczb, to n-tą liczbę obliczamy według pewnej formuły:

$$X_n = f(X_{n-1}, X_{n-2}, \ldots, X_{n-k}), \;\; n \geq k.$$

▷ Najczęściej generowane są liczby całkowite lub bity $(0/1)$ $\Rightarrow$ następnie konwertowane są one na liczby zmiennopozycyjne o **rozkładzie równomiernym** na przedziale $[0, 1)$, ozn. $\mathcal{U}(0, 1)$.

- **Okres generatora liczb losowych:**

  Sekwencje liczb z generatora matematycznego – **ciągi okresowe!**

  ▷ Niech P, ν – liczby całkowite, a $X_0, X_1, \ldots$ – sekwencja liczb losowych,

$$P \text{ – } \textbf{okres generatora} \text{ (ciągu)} \iff \exists_{\nu,P} : X_i = X_{i+jP} \; (j = 1, 2, \ldots) \; \forall_{i \geq \nu}.$$

  Zwykle okres może być wyznaczony teoretycznie (choć czasami może to być trudne!).

  ▶ **Wymagania co do okresu generatora:**

  Jeżeli N – ilość liczb losowych użyta w obliczeniach, wówczas:

$$N \ll P.$$

  $\rightarrow$ W praktyce na ogół wymaga się: $N < P^{2/3}$.

  ▷ Popularny generator: **„Mersenne Twister”** (Matsumoto & Nishimura, 1998): $P \approx 10^{6000}$.

Generatory liniowe

▶ **Ogólna postać:**

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \ldots + a_k X_{n-k} + c) \bmod m \,,$$

gdzie: $a_1, \ldots, a_k, c, m$ – parametry generatora (ustalone liczby całkowite ≥ 0),

$a \bmod b$ – reszta z dzielenia całkowitego a przez b.

▷ **Inicjowanie generatora:** dostarczenie danych początkowych: $X_0, X_1, \ldots, X_{k-1}$.

▷ **Okres generatora:** $P \leq m^k - 1$ (maksymalny okres osiągany tylko dla odpowiednio dobranych parametrów).

▶ **Popularne implementacje** (np. w starych bibliotekach języków C/C++, Pascal):

$$k = 1: \quad X_n = (a X_{n-1} + c) \bmod m \,,$$

$$c \begin{cases} = 0: & \textbf{generator multiplikatywny,} \\ \neq 0: & \textbf{generator mieszany.} \end{cases}$$

▷ Okres generatora: $P = \min\{i : X_i = X_0,\ i \in \mathbb{N}\} \leq m - 1\,.$

Maksymalny okres P_{max} może być osiągnięty tylko przy odpowiednio dobranych a i c.

$\longrightarrow$ Np. dla generatorów multiplikatywnych:

$$P_{\text{max}} = \begin{cases} 2^{L-2}: & \text{gdy} \quad m = 2^L, \\ m - 1: & \text{gdy} \quad m \text{ – liczba pierwsza.} \end{cases}$$

Generatory liniowe

Parametry kilku prostych generatorów liniowych o maksymalnych okresach:

a	c	m	Nazwa/autor
$2^{16}+3$	0	2^{31}	RANDU (IBM360/370, PDP11)
$2^2 \cdot 23^7 + 1$	0	2^{35}	Zieliński (1966)
69069	1	2^{32}	Marsaglia (1972)
16807	0	$2^{31}-1$	Park, Miller (1980)
40692	0	$2^{31}-249$	L' Ecuyer (1988)
68909602460261	0	2^{48}	Fishman (1990)

m – liczba pierwsza $\Rightarrow$ na ogół lepsze własności statystyczne generatora.

▷ Wypracowano kilka reguł doboru parametrów, aby uzyskiwać maksymalne okresy generatorów.

Uwaga: Okres $\sim 2^{32} \approx 4 \cdot 10^9$ zbyt mały dla obecnych potrzeb obliczeniowych!

▷ W praktyce wymaga się dla generatorów liniowych: $N < \sqrt{P}$.

▶ **Główna wada:** **Proste generatory liniowe nie spełniają nowszych testów statystycznych!**

Generatory liniowe

▶ **Ogólniejsze generatory liniowe – Marsaglia (1995):**

(1) $X_n = (1176X_{n-1} + 1476X_{n-2} + 1776X_{n-3}) \bmod m, \quad m = 2^{32} - 5;$

(2) $X_n = 2^{13}\,(X_{n-1} + X_{n-2} + X_{n-3}) \bmod m, \quad m = 2^{32} - 5;$

(3) $X_n = (1995X_{n-1} + 1998X_{n-2} + 2001X_{n-3}) \bmod m, \quad m = 2^{35} - 849;$

(4) $X_n = 2^{19}\,(X_{n-1} + X_{n-2} + X_{n-3}) \bmod m, \quad m = 2^{32} - 1629;$

▷ Okres: $P = m^3 - 1.$

⟶ Uzyskały dobrą ocenę statystyczną.

● **Postawowa wada generatorów liniowych:**

Wielowymiarowe rozkłady wyglądają **„bardzo nielosowo”**! ⟶ tzw. **„efekt Marsaglii”**:

▷ Niech: $U_i = X_i/m,\ i = 1, 2, \ldots, \quad \Rightarrow U_i \in [0, 1).$

Punkty: $(U_1, U_2, \ldots, U_k),\ (U_2, U_3, \ldots, U_{k+1}),\ \ldots,$

a także punkty: $(U_1, U_2, \ldots, U_k),\ (U_{k+1}, U_{k+2}, \ldots, U_{2k}),\ \ldots$

układają się na **regularnych hiperpłaszczyznach** wewnątrz kostki $[0, 1]^k$.

⟶ Opracowano metody znajdowania odległości między hiperpłaszczyznami (analiza Fouriera).

▶ **Uogólnienie na wiele wymiarów:**

$$\boxed{\vec{X}_{n+1} = \mathbf{A}\vec{X}_n \bmod m} \qquad \vec{X}_1, \vec{X}_2, \ldots \in \mathbb{R}^n,\ \mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n,\ n > 1.$$

▷ Odpowiedni wybór macierzy $\mathbf{A}$ $\Rightarrow$ zależności między składowymi generowanych wektorów.

$\Rightarrow$ Ćw. N5.1: Zaimplementować przedstawione przykłady generatorów liniowych.

Generatory SR (shift-register)

- **Generatory SR (oparte na rejestrach przesuwnych) – dla bitów:**

$$b_n = (a_1 b_{n-1} + \ldots + a_k b_{n-k}) \bmod 2,$$

gdzie: $a_1, \ldots, a_k \in \{0, 1\}$ – stałe binarne.

▷ Łatwe do implementacji, ponieważ: $(a+b) \bmod 2 = a \operatorname{xor} b$, xor – różnica symetryczna.

a	b	$a \operatorname{xor} b$
0	0	0
0	1	1
1	0	1
1	1	0

▶ Wówczas: $b_n = b_{n-i_1} \operatorname{xor} b_{n-i_2} \operatorname{xor} \ldots \operatorname{xor} b_{n-i_l}$

gdy: $a_{i_1} = \ldots = a_{i_l} = 1$, pozostałe $a_i = 0$.

▷ Dla k-elementowych układów: $(b_1, b_2, \ldots, b_k) \Rightarrow$ okres: $P \leq 2^k - 1$.

▶ Szczególny przypadek – **generator Tauswortha** (1965):

$a_p = a_q = 1$, pozostałe $a_i = 0$, $p > q$: $\boxed{b_n = b_{n-p} \operatorname{xor} b_{n-q}}$.

▷ Jak z ciągów bitów uzyskiwać liczby losowe $U_i \in \mathcal{U}(0,1)$?

→ Np. $U_i = \sum_{j=1}^{L} 2^{-j} b_{is+j} = 0.b_{is+1} \ldots b_{is+L}$, $s \leq L$, (s – ustalona liczba całkowita).

Jeżeli s nie ma wspólnych dzielników z $2^k - 1$, to ciąg $\{U_i\}$ ma maks. okres: $P_{\max} = 2^k - 1$.

Generatory SR (shift-register)

- **Efektywny algorytm Tezuki dla generatora Tauswortha (1995):**

  Niech: A – L-bitowa liczba całkowita o bitach początkowych: $b_1, \ldots, b_L$ $(\Rightarrow U_0)$,

  a B – L-bitowa zmienna pomocnicza oraz $q < p/2$ i $0 < s < p - q$:

  1: $B = ((A \ll q) \operatorname{xor} A) \ll (L - p)$

  2: $A = (A \ll s) \operatorname{xor} (B \gg (L - s))$

  3: Return A; goto 1

  gdzie $(A \ll k)$ oznacza przesunięcie bitów reprezentacji binarnej liczby A o k pozycji w lewo.

▶ Główna wada generatorów SR: **Nie spełniają niektórych nowszych testów statystycznych!**

- **Generator Tezuki (1995)** – kombinacja 3 generatorów Tauswortha: $L = 32$,

Generator	p	q	s	
			Inicjalizacja	Generacja
1.	28	9	4	13
2.	29	2	3	20
3.	31	6	1	17

▷ Do inicjalizacji potrzeba 3 liczby:

$l_1 < 2^{28},\ l_2 < 2^{29},\ l_3 < 2^{31}$.

▷ Generacja $\longrightarrow$ 3 liczby: $X_i,\ Y_i,\ Z_i \in \mathbb{N}$,

$\Rightarrow\ U_i = (X_i \operatorname{xor} Y_i \operatorname{xor} Z_i)/2^{32}$.

▷ Okres:

$P = (2^{28}-1)(2^{29}-1)(2^{31}-1) \approx 3 \cdot 10^{26}$.

▶ **Dobre własności statystyczne!**

$\Rrightarrow$ Ćw. N5.2: Zaimplementować generator Tezuki.

▷ **„Mersenne Twister”** (Matsumoto & Nishimura) – ulepszony generator SR: $P = 2^{19937} - 1$.

Generatory Fibonacciego

- **Liczby Fibonacciego (Fibonacci – Leonardo z Pizy, 1202):**

$$f_n = f_{n-2} + f_{n-1}, \quad n \geq 2, \quad f_0 = f_1 = 1.$$

▶ Pierwszy generator Fibonacciego (Taussky i Todd, 1956):

$$X_n = (X_{n-2} + X_{n-1}) \bmod m, \quad n \geq 2.$$

▷ **Nie spełnia testów niezależności liczb losowych!**

- **Uogólnione generatory Fibonacciego** $F(r, s, \odot)$:

$$X_n = (X_{n-r} \odot X_{n-s}) \bmod m, \quad n \geq r > s \geq 1,$$

gdzie operator: $\odot \in \{+, -, \times, \text{xor}\}$ → Łatwe w implementacji!

Maksymalny okres dla $m = 2^L$:

$\odot$	P_{max}	Własności statystyczne
$+, -$	$(2^r - 1)2^{L-1}$	dobre
$\times$	$(2^r - 1)2^{L-3}$	bardzo dobre
xor	$2^r - 1$	niezbyt dobre

Przykładowe wartości r i s dające maksymalne okresy:

r	17	31	55	68	97	607	1279
s	5	13	24	33	33	273	418

Przykład: generator MZT

- Popularny generator **MZT**, znany również jako **RANMAR** (Marsaglia, Zaman, Tsang, 1990):
  - ▷ **Uniwersalny** – daje identyczne wyniki na komputerach, gdzie:
    - ∗ liczby całkowite – co najmniej 16 bitów,
    - ∗ liczby zmiennopozycyjne – co najmniej 24 bity mantysy.
  - ▶ Kombinacja 2 generatorów:

1. Generator typu Fibonacciego:

$$F(97, 33, \bullet) \quad \longrightarrow \quad V_n \in [0, 1),$$

gdzie:

$$x \bullet y = \begin{cases} x - y, & x \geq y, \\ x - y + 1, & x < y. \end{cases}$$

▷ Inicjalizacja generatora – wyznaczenie liczb: $V_1, V_2, \ldots, V_{97}$

za pomocą ciągu bitów: $V_1 = 0.b_1 b_2 \ldots b_{24}$, $V_2 = 0.b_{25} b_{26} \ldots b_{48}$, $\ldots$

$\longrightarrow$ Ciąg bitów $\{b_n\}$ generowany za pomocą dwóch generatorów:

$$\left.\begin{aligned} y_n &= (y_{n-3} \cdot y_{n-2} \cdot y_{n-1}) \bmod 179 \\ z_n &= (53 z_{n-1} + 1) \bmod 169 \end{aligned}\right\} \Rightarrow b_n = \begin{cases} 0, & (y_n \cdot z_n) \bmod 64 < 32, \\ 1, & (y_n \cdot z_n) \bmod 64 \geq 32. \end{cases}$$

▷ Inicjowanie – dostarczenie 4 liczb: $y_1, y_2, y_3 \in \{1, 2, \ldots, 178\}$, $z_1 \in \{0, 1, \ldots, 168\}$.

$(y_1 \cdot y_2 \cdot y_3 \neq 1)$

▶ Okres: $P = 2^{120}$.

Przykład: generator MZT

2. Generator liczb losowych $c_n \in (0,1)$:

$$c_n = c_{n-1} \circ (7654321/16777216), \quad n \geq 2, \quad c_1 = 362436/16777216,$$

gdzie:

$$c \circ d = \left\{ \begin{array}{ll} c-d, & c \geq d \\ c-d+(16777213/16777216), & c < d \end{array} \right\} \; c,d \in [0,1).$$

▷ Okres: $P = 2^{24} - 3$.

- **Ostatecznie generator MZT:**

$$U_n = V_n \bullet c_n \, .$$

▷ Okres: $P \simeq 2^{144} \approx 2 \cdot 10^{43}$.

► **Spełnia wszystkie znane testy statystyczne!**

⇛ Ćw. N5.3: Zaimplementować wybrany uogólniony generator Fibonacciego.

⇛ Ćw. N2* (nadobowiązkowe): Zaimplementować generator MZT.

Generatory SWB (subtract-with-borrow)

- **Generatory SWB (oparte na odejmowaniu z pożyczką) (Marsaglia & Zaman, 1991):**

  Ozn. c – bit przeniesienia, tzn. $c \in \{0, 1\}$.

  Na początku: $c = 0$,

  $\longrightarrow$ Odejmowanie z pożyczką (SWB):

$$x \ominus y \bmod m = \begin{cases} x - y - c + m, & \text{oraz } c = 1, \quad \text{gdy } x - y - c < 0, \\ x - y - c, & \text{oraz } c = 0, \quad \text{gdy } x - y - c \geq 0. \end{cases}$$

  ▶ Schemat:

$$X_n = X_{n-r} \ominus X_{n-s} \bmod m \qquad r, s \in \mathbb{N},\ r > s.$$

  ▷ Inicjalizacja:

  Ciąg liczb całkowitych $X_1, \ldots, X_r \in (0, m)$ oraz $c = 0$.

  ▶ Zalety: Proste i szybkie!

  ▶ Wady: Nie spełniają niektórych nowszych testów statystycznych (niezależności)!

  ▷ Przykłady:

  - Generator **RCARRY** (F. James, 1990): $r = 24,\ s = 10,\ m = 2^{24}$ $\longrightarrow$ Okres: $P \approx 5 \cdot 10^{171}$.
  - Generator **ULTRA** (Marsaglia, Zaman): $r = 37,\ s = 24,\ m = 2^{32}$ $\longrightarrow$ Okres: $P \approx 10^{346}$.

Generatory MWC (multiply-with-carry)

- **Generatory MWC (oparte na mnożeniu z przeniesieniem) – G. Marsaglia:**

  ▶ Schemat:

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \ldots + a_r X_{n-r} + c) \bmod m \,,$$

  gdzie: $a_1, a_2, \ldots, a_r \in \mathbb{N}$ – ustalone parametry,

  c – tzw. **wartość przeniesienia** (do następnego kroku):

$$c = \lfloor (a_1 X_{n-1} + a_2 X_{n-2} + \ldots + a_r X_{n-r} + c)/m \rfloor,$$

  gdzie: $\lfloor a \rfloor$ ozn. część całkowitą liczby a.

  ▷ Inicjalizacja – dowolne wartości początkowe: $X_1, X_2, \ldots, X_r$ i c.

  ▶ **Zalety:**

  ∗ Proste, szybkie, łatwe w implementacji.

  ∗ Mają długie okresy.

  ∗ Bardzo dobre własności statystyczne.

  ▷ Wiele generatorów zaproponowanych przez Marsaglię.

Przykłady generatorów MWC

1. **MWC1:**

$$\left.\begin{aligned} X_n &= (18000\, X_{n-1} + c_X) \bmod 2^{16} \\ Y_n &= (30903\, Y_{n-1} + c_Y) \bmod 2^{16} \end{aligned}\right\} \text{ liczby 16-bitowe}$$

$$\Rightarrow\ Z_n = b_1^{X_n} \ldots b_{16}^{X_n} b_1^{Y_n} \ldots b_{16}^{Y_n} \quad \text{– liczba 32-bitowa.}$$

▷ Inicjalizacja: Dwie liczby 32-bitowe $\Rightarrow X_0, Y_0$ oraz c_X, c_Y, np. $\underbrace{b_1 \ldots b_{16}}_{X_0}\underbrace{b_{17} \ldots b_{32}}_{c_X}$ itd.

▶ Okres: $P \simeq 2^{60} \approx 10^{18}$.

2. **MWC2:**

$$\begin{aligned} X_n = (&12013\, X_{n-8} + 1066\, X_{n-7} + 1215\, X_{n-6} + 1492\, X_{n-5} + 1776\, X_{n-4} \\ &+ 1812\, X_{n-3} + 1860\, X_{n-2} + 1941\, X_{n-1} + c_X) \bmod 2^{16} \\ Y_n = (&9272\, Y_{n-8} + 7777\, Y_{n-7} + 6666\, Y_{n-6} + 5555\, Y_{n-5} + 4444\, Y_{n-4} \\ &+ 3333\, Y_{n-3} + 2222\, Y_{n-2} + 1111\, Y_{n-1} + c_Y) \bmod 2^{16} \end{aligned}$$

$\Rightarrow$ 32-bitowa liczba Z_n z połączenia bitów 16-bitowych liczb X_n, Y_n.

▷ Inicjalizacja: 16 liczb całkowitych 16-bitowych, np. z prostego generatora liniowego.

▶ Okres: $P \simeq 2^{250} \approx 2 \cdot 10^{75}$.

$\Rrightarrow$ Ćw. N5.4: Zaimplementować generatory MWC1 i MWC2 (użyć operacji bitowych).

Generatory nieliniowe

▶ Naturalnym pomysłem na uniknięcie problemów związanych z generatorami liniowymi wydaje się generowanie ciągów liczb w oparciu o formuły nieliniowe.

▷ Dziedzina generatorów nieliniowych rozwija się od połowy lat 1980-tych.

- **Generator inwersyjny Eichenauera i Lehna (1986):**

$$X_n = (aX_{n-1}^{-1} + b) \bmod m,$$

gdzie: c^{-1} – liczba całkowita: $c \cdot c^{-1} \bmod m = 1$, m – liczba pierwsza.

$\rightarrow X_n \in \{0, 1, \ldots, m-1\} \Rightarrow U_n = (X_n/m) \in [0, 1).$

- **Generator inwersyjny Eichenauera-Hermanna (1993):**

$$X_n = [a(n + n_0) + b]^{-1} \bmod m, \quad a \in \{1, 2, \ldots, m\}$$

$\rightarrow$ Liczba X_n może być otrzymana niezależnie od innych elementów ciągu; okres: $P = m$.

▷ Użyteczne dla obliczeń równoległych.

- **Generator kwadratowy – L. Blum, M. Blum, Shub (1986):**

$$X_n = X_{n-1}^2 \bmod m,$$

▷ Zastosowania w kryptografii.

▶ **Zalety:** Bardzo dobre własności statystyczne (spełniają wszystkie znane testy).

▶ **Wady:** Są wolniejsze od generatorów liniowych.

Kombinacje generatorów

- **Kombinacje kilku generatorów – zwykle dają lepsze wyniki, ale nie zawsze!**

▶ Przykłady: Przedstawione wcześniej generatory Tezuki i MZT.

▶ **Kombinacje nieliniowych generatorów inwersyjnych:**

Weźmy $r \geq 5$ generatorów inwersyjnych (przedstawionych powyżej):

$$X_m^{(j)}, \quad j = 1, \ldots, r$$

z parametrami $m_j,\ j = 1, \ldots, r,$ będącymi liczbami pierwszymi.

$$\Rightarrow U_n = U_n^{(1)} + \ldots + U_n^{(r)} \bmod 1, \quad n = 0, 1, 2, \ldots,$$

gdzie $U_n^{(j)} = X_n^{(j)}/m_j,\ j = 1, \ldots, r.$

▷ Długi okres: $P = m_1 m_2 \ldots m_r.$

▷ Bardzo dobre własności statystyczne!

▶ Więcej szczegółow nt. generatorów nieliniowych i nie tylko: http://random.mat.sbg.ac.at/

⇛ Ćw. N6.1: Zastosować powyższe implementacje generatorów liczb losowych do obliczania całek Φ i Ψ z wykładu 2. Porównywać otrzymywane wyniki z wynikami poprzednimi.

Generator RANLUX i operacyjna definicja losowości

- Wszystkie przedstawione wyżej matematyczne generatory liczb losowych opierają się na pewnych formułach rekurencyjnych!

▷ **Typowy schemat konstrukcji generatora liczb losowych:**

W oparciu o teorię liczb i statystykę matematyczną skonstruować odpowiednią formułę rekurencyjną, a następnie liczby otrzymywane według tej formuły poddać szeregowi testów statystycznych. Jeżeli wyniki testów są pozytywne, to akceptujemy dany generator.

$\longrightarrow$ Ale to nie daje nam odpowiedzi na pytanie, dlaczego te liczby wyglądają losowo!

▶ Właściwie nie ma żadnego wyraźnego powodu by formuły rekurencyjne miały dawać liczby losowe lub nawet wyglądające na losowe! $\longrightarrow$ To, że tak jest wydaje się raczej zaskakujące!

- **1993: M. Lüscher** – fizyk teoretyk, specjalizujący się w kwantowej teorii pola na siatkach.
  - ▷ Artykuł: hep-lat/9309020, Comput. Phys. Commun. **79** (1994) 100.
  - ▶ **„Wreszcie pojawiła się teoria generowania liczb losowych”** (F. James)
    - ▷ Artykuł: F. James, *“Finally, a theory of random number generation”,* Proceedings of the Fifth Workhop on Mathematical Methods in Scattering Theory and Biomedical Technology, Corfu, Greece, October 2001, D. Fiotiadis and C. Massalas (eds.), World Scientific (2002).

Generator RANLUX i operacyjna definicja losowości

- **Operacyjna definicja losowości w sensie wymaganym w rachunkach Monte Carlo:** oparta o chaotyczne zachowanie w klasycznych układach dynamicznych (teorie Kołmogorowa i Arnolda) – użycie wykładników Lapunowa i entropii Kołmogorowa do badania chaotycznego zachowania liczb produkowanych przez generator.

- **Generator RANLUX:** oparty o generator SWB o nazwie **RCARRY** (patrz: str. 14) – uzupełniony o algorytm odrzucania pewnych sekwencji liczb w celu osiągnięcia wystarczająco ,,chaotycznych" własności generowanych liczb losowych ($\longrightarrow$ usuwanie korelacji krótkozasięgowych).

▷ Okres: $P \approx 5 \cdot 10^{171}$.

▶ **Przeszedł pomyślnie najsurowsze testy statystyczne!**

▷ Niestety, pomijany milczeniem przez niemal wszystkich matematyków – ekspertów od generatorów liczb losowych (wg. F. Jamesa nie znają oni teorii chaosu Kołmogorowa i Arnolda).

▶ Cytowany w książce Donalda E. Knutha, ,,*Sztuka programowania*" tom 2, WNT 2002: potwierdzenie jego dobrych własności statystycznych, ale brak zrozumienia dla teorii.

$\Rrightarrow$ Ćw. N6.2: Zastosować generatory **RANLUX (TRandom1)** i **Mersenne Twister (TRandom3)** z systemu ROOT do obliczania całek Φ i Ψ z wykładu 2. Porównywać otrzymywane wyniki z wynikami poprzednimi.