NÁHODNÁ ČÍSLA V NUMERICKÝCH VÝPOČTECH

Náhodná čísla

- Proč potřebujeme náhodná čísla?
- Pro vytvoření náhodného stavu nebo náhodné události
  - náhodné souřadnice molekul
  - Brownův pohyb, chaos
  - fyzikální procesy (radioaktivní rozpad, šum, atd.)
  - pohyb lidí, rozhodovací procesy
  - náhodný směr výstřelu na videoherní postavu
  - náhodný videoherní quest
- Nejčastější aplikace
  - počítačové simulace
  - videohry
- **Chceme popsat různé jevy pomocí simulačních modelů**
- **Generování náhodných čísel**

Možnosti získání náhodných čísel

- Základ vždy pozorování reálného jevu
  - co použít při simulacích?
- Přímo hodnoty **měření reálného jevu**
  - skutečná náhodnost
  - malý počet náhodných hodnot
- **Odhad pravděpodobnostního rozdělení** pozorovaného jevu
  - a **generovat hodnoty** z tohoto rozdělení
  - dostatečný počet „náhodných“ hodnot
  - odhad rozdělení a jeho parametrů

Generátory náhodných čísel

- Generování náhodných čísel
  - náročný proces
  - pseudonáhodná čísla
    - John von Neumann, 1946 prvotní metoda pro rychlé získání náhodných číslic
- Nevýhody pseudonáhodných čísel
  - nemají zcela náhodné rozdělení
    - problém (šifrovací aplikace, hry, …)
  - zkoumáním minulé sekvence lze často určit následné číslo
    - např. neuronovou sítí
  - stále lepší metody , ale i rozvoj oblasti hlubokého učení
    - schopné rozpoznat následnou sekvenci čísel
- Dělení
  - fyzikální generátory
  - tabulky náhodných čísel
  - lineární kongruenční generátory
- Základní generátory
  - Randu, ZXSPECTRUM, Marsaglia XorShift, Mersene Twister, Park&Miller,...

Skutečná náhodná čísla

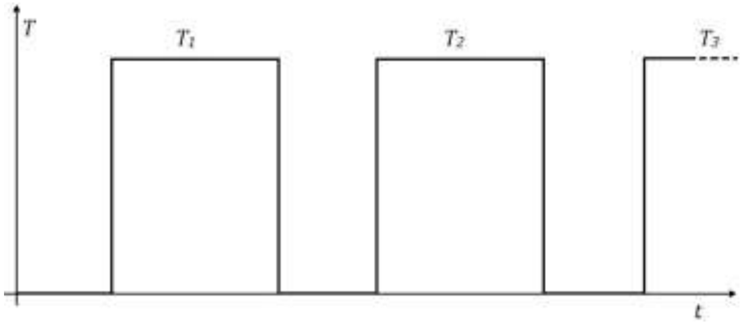
- Zařízení, která poskytují skutečně náhodná čísla
  - může být zpřístupněno webovými službami

- Generátory skutečně náhodných čísel
  - využívají nějaký **fyzický** nebo **fyzikální** fenomén

- Generátory založené na **fyzických** jevech
  - např. generování čísla ze sekvence pohybu myší nebo úhozů do klávesnice (či prodlevě mezi úhozy)
  - problémy
    - člověk může využívat podobný vzor pohybů a úhozů
    - komunikace s OS pomocí bufferů, které mohou náhodnost vyrušit
- Generátory založené na **fyzikálních** jevech
  - lepší, ale finančně náročnější

Fyzikální generátory

- Nejčastější jevy
  - radioaktivní rozpad nuklidů
    - Geiger-Müllerův počítač
  - atmosférický šum
    - elektromagnetické vlnění v daném prostoru a čase
    - lze získat citlivou anténou
  - akustický tlak v místnosti (šum z hluku)
    - slabší generátory
    - problém s prediktivními jevy jako hluk z otáček větráku

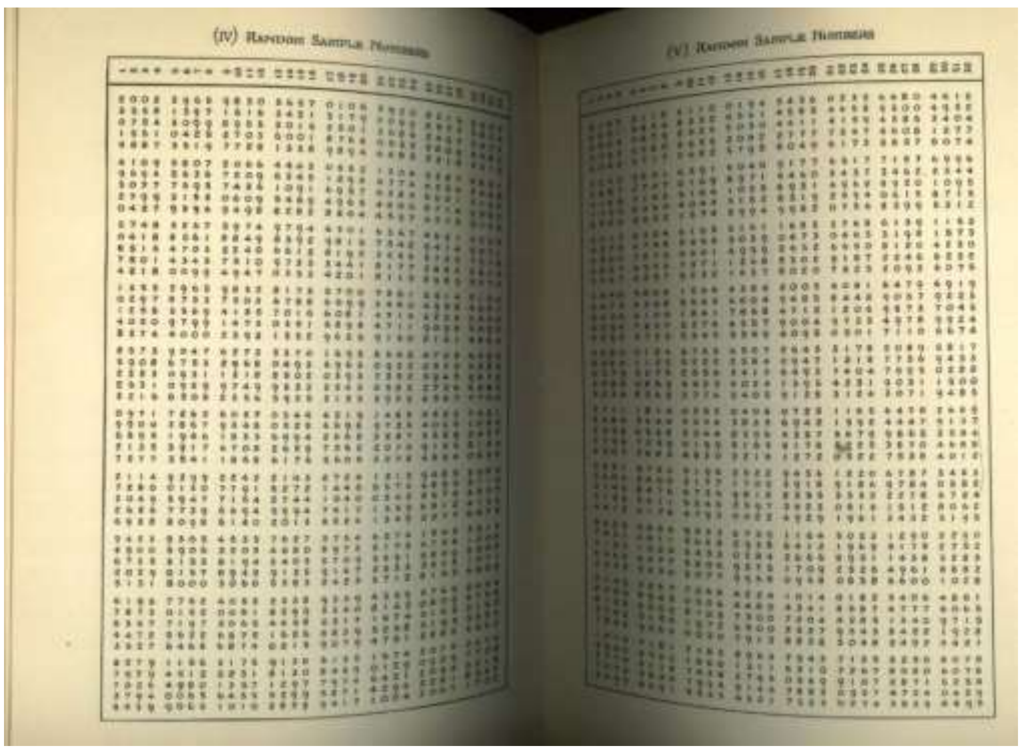


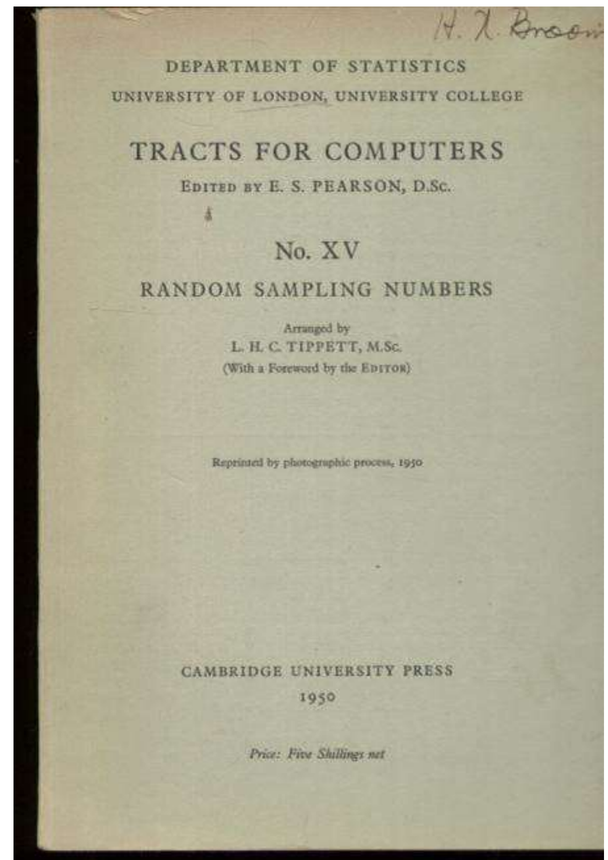
pokud $T_1 > T_2$ – zapíšeme 0
pokud $T_1 < T_2$ – zapíšeme 1

- Využití externích webových služeb přes jejich rozhraní REST
  - levné, spolehlivé
  - např. random.org
    - data pro vygenerování náhodných číslic z atmosférického šumu
    - potřebujete získat API klíč
    - uvádíte v požadavku ve formátu JSON pomocí metody POST z HTTP protokolu
    - klíč můžete vygenerovat po registraci na stránce: https://accounts.random.org/create
    - při volbě developer licence je registrace zdarma, ale denní limit vygenerovaných čísel 1000

Tabulky náhodných čísel

- Opravdu náhodná čísla
  - z fyzikálních generátorů
    - disk, CD nosič, páska
  - rozsáhlé soubory dat (tabulky)
    - 1927 – Tipper - 40 tis náhodných čísel
    - 1955 – RandCorp - 1 mil. náhodných čísel




H. L. Broom
DEPARTMENT OF STATISTICS
UNIVERSITY OF LONDON, UNIVERSITY COLLEGE

TRACTS FOR COMPUTERS

EDITED BY E. S. PEARSON, D.Sc.

No. XV

RANDOM SAMPLING NUMBERS

Arranged by
L. H. C. TIPPETT, M.Sc.
(With a Foreword by the EDITOR)

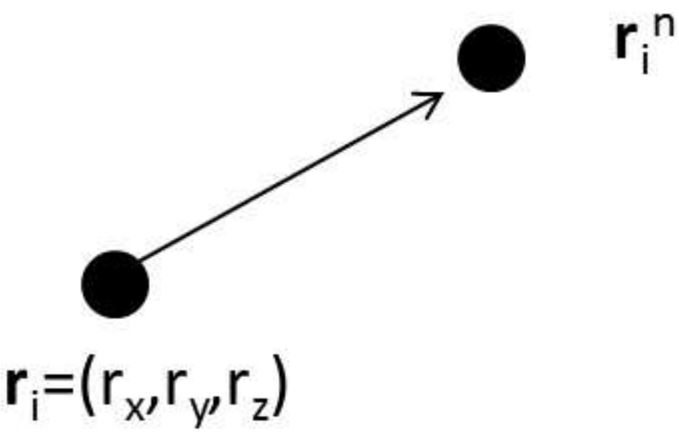
Reprinted by photographic process, 1950

CAMBRIDGE UNIVERSITY PRESS
1950

Price: Five Shillings net

Generování náhodného jevu – posun částice

- Náhodný posun částice v kapalině
- Generování náhodného čísla ξ v intervalu $(0,1)$
- Transformace ξ podle odhadnutého rozdělení
- Realizace náhodného jevu (sledovaná veličina)



- máme dané maximální posunutí $\boldsymbol{r}_{max}$
- generujeme náhodný směr

$$\underline{\xi} = (\xi_{x'}, \xi_{y'}, \xi_{z'})$$

- částici posuneme na nové (náhodné) místo

$$\boldsymbol{r}_i^n = \boldsymbol{r}_i + \underline{\xi} \cdot \boldsymbol{r}_{max}$$

- **Lze náhodný posun částice generovat i jiným způsobem?**

Generování náhodných čísel

- Náhodné číslo ξ
  - je v intervalu (0,1), $\langle 0,1 \rangle$,…
  - transformace na jiné rozdělení/interval
  - $\zeta^T = \zeta(b-a) + a, \qquad a, b \in \mathbb{R}$

- Základní dělení generátorů
  - Fyzikální generátory
  - Tabulky náhodných čísel
  - Vypočítaná náhodná čísla

Vypočítaná náhodná čísla

- Pseudonáhodná (kvazináhodná)
  - Algoritmus (posloupnost čísel), perioda P
  - náhodné číslo $\xi_{i+1} = f(\xi_i, \xi_{i-1}, \xi_{i-2}, \dots \xi_{i-n})$

- Von Neumannovy generátory
- Lineární kongruenční generátory
- Generátory s posuvnými registry

- Požadavky
  - co nejdelší perioda P
  - co největší náhodnost v rámci periody a rovnoměrné pokrytí
  - co největší rychlost generování čísla ξ_{i+1}

Vypočítaná náhodná čísla

- Von Neumannovy generátory (1946)
- „Každý, kdo se zabývá aritmetickými metodami vytváření náhodných čísel, se nepochybně dopouští hříchu“

- První známý generátor pseudonáhodných čísel
  - krátká perioda opakování náhodných čísel
  - řešil problém pomalého čtení náhodných čísel z děrných štítků pro počítač ENIAC
- Metoda prostředku čtverce (Middle-square)
    - (prostředních řádů druhé mocniny)
  - Zvolím počáteční číslo x_0 o $2k$ číslicích
  - Číslo se umocní
  - Z druhé mocniny se vybere prostředních 2k číslic
  - Získané číslo je dalším prvkem posloupnosti

Vypočítaná náhodná čísla

- Von Neumannovy generátory (1946)

x_0	x_0^2
1236	01527696

x_1	x_1^2
5276	27836176

x_2	x_2^2
8361	69906321

x_3	x_3^2
9063	81237969

- Krátká perioda P
- Malá náhodnost v rámci periody
- Pomalý proces generování

Lineární kongruenční generátory

- Historicky jeden z nejdůležitějších generátorů pseudonáhodných čísel
  - používán v mnoha implementacích novějších generátorů
    - např.: Park-Miller z C++11 standardní knihovny

- Princip:
  - 1. zvolíme parametr $\boldsymbol{M}$ (modulus)
    - prvočíslo nebo jeho mocninu
  - 2. zvolíme parametr $\boldsymbol{C}$ (inkrement)
    - pro $C = 0$ se nazývá generátor Lehmerův
  - 3. zvolíme parametr $\boldsymbol{a}$ (násobek)
  - 4. algoritmus vyžaduje semínko seed, které představuje první ξ_i
  - 5. další náhodné číslo ze vzorce: $\xi_{i+1} = (\boldsymbol{a} * \xi_i + \boldsymbol{C}) \bmod \boldsymbol{M}$

Lineární kongruenční generátory

- D. H. Lehmer (1948)

$$\xi_{i+1} = (a_0\xi_i + a_1\xi_{i-1} + \cdots + a_n\xi_{i-k} + b)(\mathrm{mod}\ M)$$

- Konstanty: a_j, b, M
  - vhodnou volbou konstant určujeme vlastnosti generátoru
  - $k > 0,\ \xi_i < M$
- Semínko (násada)
  - $\xi_0, \ldots, \xi_{-k}$ $(i = 0)$
  - $\xi_1 = (a_0\xi_0 + a_1\xi_{-1} + \cdots + a_n\xi_{-k} + b)(\mathrm{mod}\ M)$
- **Stejná násada = stejná posloupnost čísel**

$$\xi_i = 0, 1, 2, \ldots, M-1, \qquad \xi_i = \frac{\xi_i}{M} \rightarrow \qquad \xi_i \in (0,1)$$

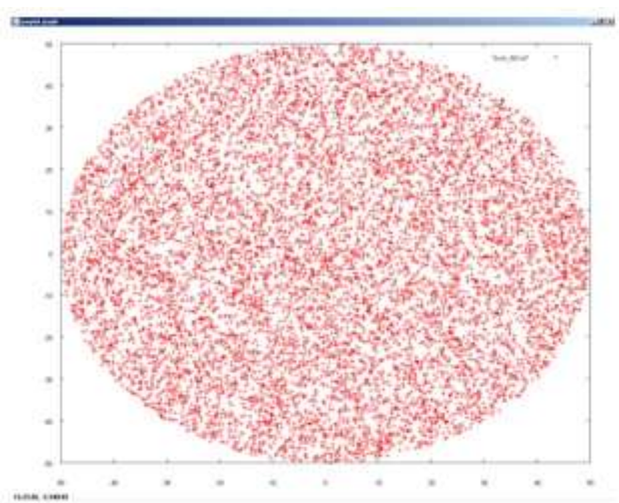
- Dělení
  - Multiplikativní generátory
  - Aditivní generátory
  - Smíšené generátory

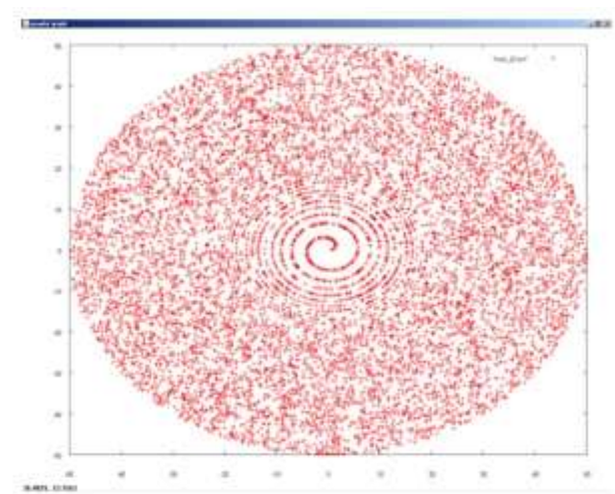
Multiplikativní LKG

- Multiplikativní generátory
  - velmi rychlé generování

D. H. Lehmer $\xi_{i+1} = a_0 \xi_i \pmod{M}$

- IBM 360: $\xi_{i+1} = 7^5 \xi_i \pmod{2^{31} - 1}$
- IBM 370: $\xi_{i+1} = 630360016\, \xi_i \pmod{2^{31} - 1}$
- Fortran: $\xi_{i+1} = 13^{13} \xi_i \pmod{2^{59}}$
- ZX SPECTRUM: $\xi_{i+1} = 75 \xi_i \pmod{65537}$
- RANDU: $\xi_{i+1} = a_0 \xi_i \pmod{2^{31}}$, $a_0 = 1$
- jazyk BASIC (nevhodná volba konstant)





Aditivní LKG

- Fibonacciho generátor
  - využívá Fibonacciho posloupnost
    - předposlední a poslední hodnotu
  - $\xi_{i+1} = (\xi_i + \xi_{i-1}) (\mathrm{mod}\ M)$
- Opožděný Fibonacciho generátor (Mitchell, Moore, 1958)
  - využívá obecné hodnoty opožděnosti
    - nepoužívá se, ale jednoduchý
  - $\xi_{i+1} = \left(\xi_{i-j} + \xi_{i-k}\right) (\mathrm{mod}\ M)$
  - místo "+" může být jiná operace
    - $+ \; - \; * \; /$ ap.
- Millerův-Prenticův generátor
  - $\xi_{i+1} = (\xi_{i-1} + \xi_{i-2n}) (\mathrm{mod}\ 3137) \qquad (P = 9.8 \cdot 10^6)$
- Další aditivní generátory
  - $\xi_{i+1} = (\xi_{i-5} + \xi_{i-17}) (\mathrm{mod}\ M)$

Volba M ovlivňuje periodu generátoru

$$M = 2^6 \rightarrow P = 1.6 \cdot 10^7$$
$$M = 2^{16} \rightarrow P = 4.3 \cdot 10^9$$
$$M = 2^{32} \rightarrow P = 2.8 \cdot 10^{14}$$

Smíšené LKG

- Smíšené generátory
  - kombinují vlastnosti multiplikativního a aditivního
    - tak, aby došlo ke zlepšení vlastností
  - $\xi_{i+1} = (a\xi_i + b)(\mathrm{mod}\ M)$

- Příklady
  - $\xi_{i+1} = (69069\xi_i + 1)(\mathrm{mod}\ 2^{32})$

Minimal Standard

- Série pravidel (parametrů)
  - generátor má plnou periodu (rovnoměrné pokrytí)
    - vybraná subsekvence čísel je nerozeznatelná od celé, generované sekvence
  - generátor projde **testy spolehlivosti**
  - lze jej **efektivně** implementovat pomocí 32-bitové architektury

- Lewsi, Goodman, Miller (1969)
  - $\xi_{i+1} = a_0 \xi_i (\mathrm{mod}\, M), \qquad a_0 = 16807, \qquad M = 2^{31} - 1$
  - snadná implementace do vysokoúrovňových jazyků
  - seed generátor: $\xi_0 \in (1,\ 2^{31} - 1)$, jsou rovnocenné

Generátory pseudonáhodných bitů

- Posuvné registry
  - k ukládání a posouvání jednotlivých bitů
    - o jeden nebo více míst
    - vpravo nebo vlevo (shift right, shift left)
    - realizace pomocí kombinační logiky nebo speciálních posuvných instrukcí procesoru
  - jeden nebo více registrů
    - každý uchovává určitý počet bitů
- pro výpočet následujícího čísla se používá kombinace bitů z několika registrů
  - některé bity se mohou použít jako zpětná vazba do registru
- počáteční stav registrů = semínko
  - ovlivňuje celou posloupnost čísel
  - pokud se použije stejné, bude posloupnost stejná
- Vlastnosti
  - konečná perioda
  - jednoduchost
  - malá paměťová náročnost
  - rychlost

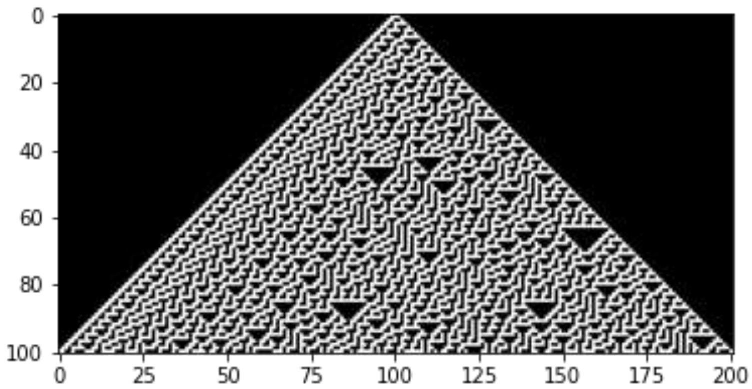
Pravidlo 30

- Stephen Wolfram (1983)
  - název podle decimální reprezentace schémata binárních operací
- Pravidlo 30
  - jedno z nejzajímavějších výpočetních schémat základních automatů
    - pro aktualizaci stavu buněk v 1D buněčném automatu
  - vytváří automat, obsahující pseudonáhodné sekvence (aperiodické chaotické sekvence bitů).
- Vlastnosti
  - jednoduchá definice
  - přesto velmi komplexní a chaotické vzorce
    - použijeme jako náhodnou posloupnost

Pravidlo 30

- Princip
  - pro každou buňku vypočítá nový stav na základě stavů tří sousedících buněk v předchozí řadě
    - kombinace stavů buněk v aktuální a předchozí řadě
    - např. pokud má buňka sousedy 1, 0 a 0, její nový stav bude 1

111 → 0
110 → 0
101 → 0
100 → 1
011 → 1
010 → 1
001 → 1
000 → 0



- Postup
  - 1. Nastav prvotní řádek buněčného automatu na binární 0, prostředek na binární 1
  - 2. Proveď vývoj buněčného automatu do zvolené generace pomocí pravidla 30
  - 3. Vyber prostřední sloupec, přeskoč N bitů (semínko) a získej 8 binárních číslic
  - 4. Vytvoř z M binárních číslic poseudonáhodné číslo

Generátory s posuvnými registry

- S posuvnými registry s lineární zpětnou vazbou
  - posuvný registr délky L
    - L = $\{R_0, R_1, \dots, R_{L-1}\}$ vnitřních registrů
    - časový signál
    - charakteristický mnohočlen C(x)

Iniciace generátoru

Pozor na zakázané stavy logické funkce

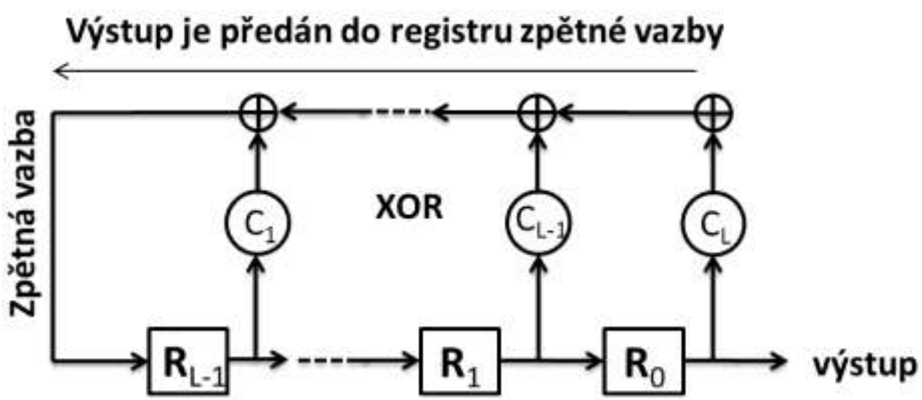
Zacyklení generátoru

Iniciace pomocí LKG (Lehmer,…)

- Princip
  - v každém časovém okamžiku se:
    - obsah R_i přesune do R_{i-1}, R_0 se předá na výstup
    - do registru R_{L-1} se uloží (výpočtem) nový obsah
  - R_i uchovává jednu jednotku informace
    - Jeden bit (0,1), 2^n bitů (velikost slova / jednotka času)
  - vnitřní stav generátoru S = $\{S_0, \dots, S_{L-1}\}$
  - charakteristický mnohočlen

$$C(x) = 1 + c_1 x + c_2 x^2 + \cdots + c_L x^L$$

  - koeficienty: zbytek po dělení modulo
    - 1 – liché, 0 – sudé

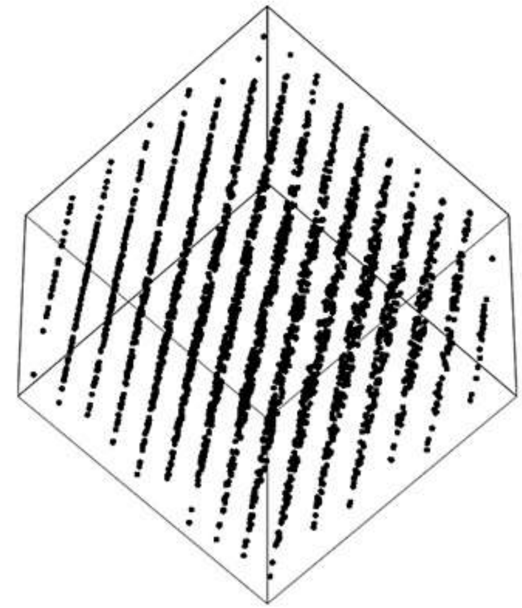


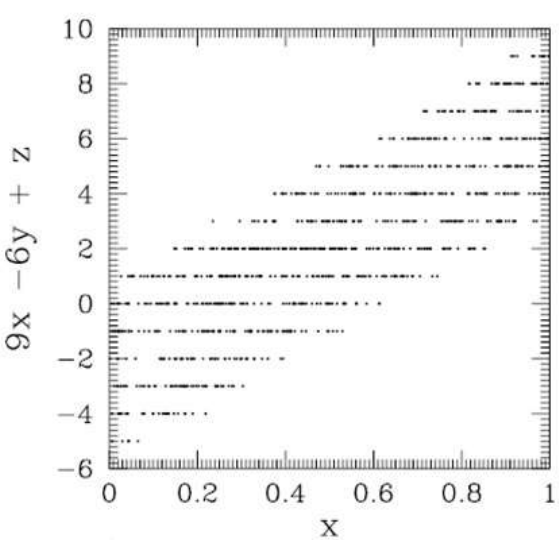
Marsaglia XORshift

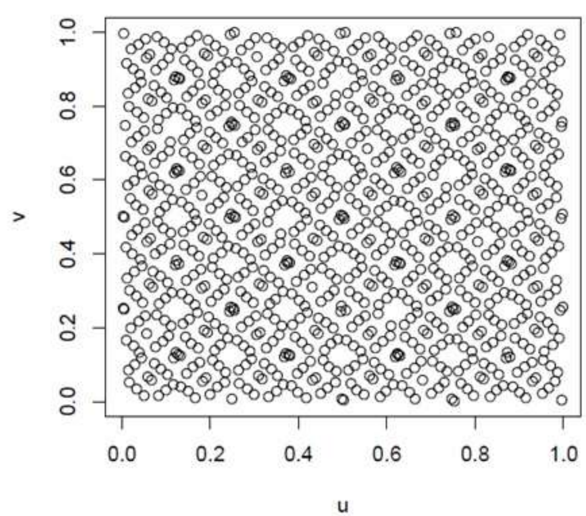
- George Marsaglia (2003)
  - na základě von Neumannova generátoru
- Užití operace XOR a bitového posunu
  - XOR (exkluzivní disjunkce)
    - logická operace, výstupem pravda, pokud vstupy unikátní
    - $A = (0,1,0,1), \quad B = (0,0,1,1), \quad A \oplus B = (0,1,1,0)$
  - logické shift (bitový posun)
    - levý posun: 001010111 → 01010111**0**
    - pravý posun: 001010111 → **0**00101011
- Postup
  - bitové posuny (různé podle implementace)
    - např. Xorshift128 o 23, 17, 26 a 11 míst vlevo
  - posunuté bity se použijí pro XOR s jinými bity v registru
    - aby se vytvořil nový stav generátoru
    - aplikujeme XOR na vektor β (binární) s posunutou verzí sebe sama
    - $\beta >> a$ – bitový posuv doprava o a pozic
      - a je parametr generátoru
    - $\beta \oplus (\beta >> a)$ – XORshift vektoru β o a pozic doprava

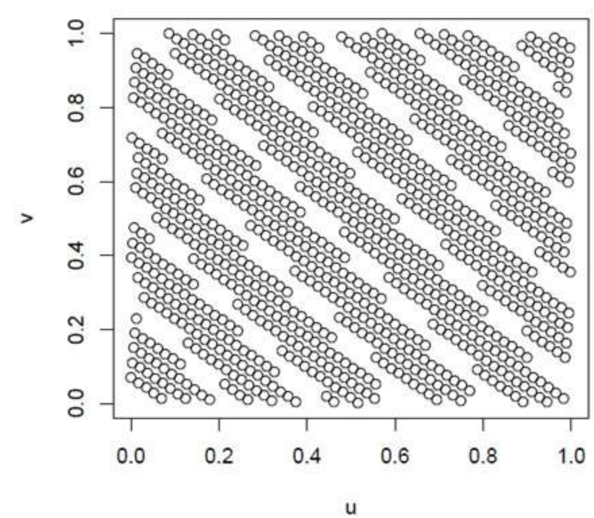
Marsaglia XORshift

- Produkujе sekvenci:
  - $2^{32}-1$ x celých čísel
  - $2^{64}-1$ x, y dvojic
  - $2^{96}-1$ x, y, z trojic
- George Marsaglia (1924 - 2011)
  - americký matematik (statistik)
  - algoritmy: XORshift, Ziggurat, …
  - „Random numbers fall mainly in the planes.“
    - multiplikativní generátory: „Crystalline“ nature









Mersenne Twister

- Makoto Mastumoto, Takuji Nishimura, 1998
  - Mersenneho prvočíslo
    - prvočíslo o jedničku menší než mocnina 2, $M = 2^n - 1.$
    - $2^3 - 1 = 7$ (je prvočíslo), $M = 2^4 - 1 = 15$ (není prvočíslo)
  - www.mersenne.org
    - největší ověřené Mersenneho prvočíslo (r. 2021) 57 885 161 (48. Mersenneho prvočíslo),
    - r. 2006 32 582 657 (44.)

- Použití
  - dnes jeden z nejpoužívanějších
  - různé verze MT
    - nejvyužívanější verze (nastavení parametrů) MT19937
  - používá Python v modulu Random
  - dále R, Ruby, Free Pascal, PHP, Maple, MATLAB, GAUSS, Julia, Microsoft Visual C++,...
    - numpy jiný (PCG64 z roku 2014)

Mersenne Twister

- Založen na generátoru s posuvnými registry
  - polynom $C(x)$ s řádem P, stavový vektor x o velikosti w bitů

- Generuje stavový vektor x

$$x_{k+n} = x_{k+m} \oplus \left(x_k^u \mid x_{k+1}^l\right)A$$

  - x_n je stavový vektor v kroku n
  - x_k^u je subvektor složený z w-r levých bitů vektoru **x**
  - x_{k+1}^l je subvektor složený z w-r pravých bitů vektoru **x**
  - | představuje operaci zřetězení
    - "Hello" | "world" → "hello world"
  - A je transformační matice
  - $\oplus$ XOR

- Vlastnosti
  - velice dlouhá perioda
    - až $P = 2^{19937} - 1$
  - TinyMT (2012)
    - $P = 2^{512}$, ale méně zatěžuje procesor

<u>Špatný Seed:</u>
trvá dlouho, než začne generovat náhodnou sekvenci

Testy generátorů náhodných čísel

- Mějme uspořádanou k-tici náhodných celých čísel $n(k)$
- Testy náhodnosti
  - Statistické testy (chí kvadrát test)
  - Transformace (Hadamard, Marsaglia)
  - Komplexita (složitost)
    - Kolmogorovova komplexita: měří složitost k-tice podle počtu znaků, délky programu, který takovou k-tici vyprodukuje
- DIEHARD testy
- TESTU01

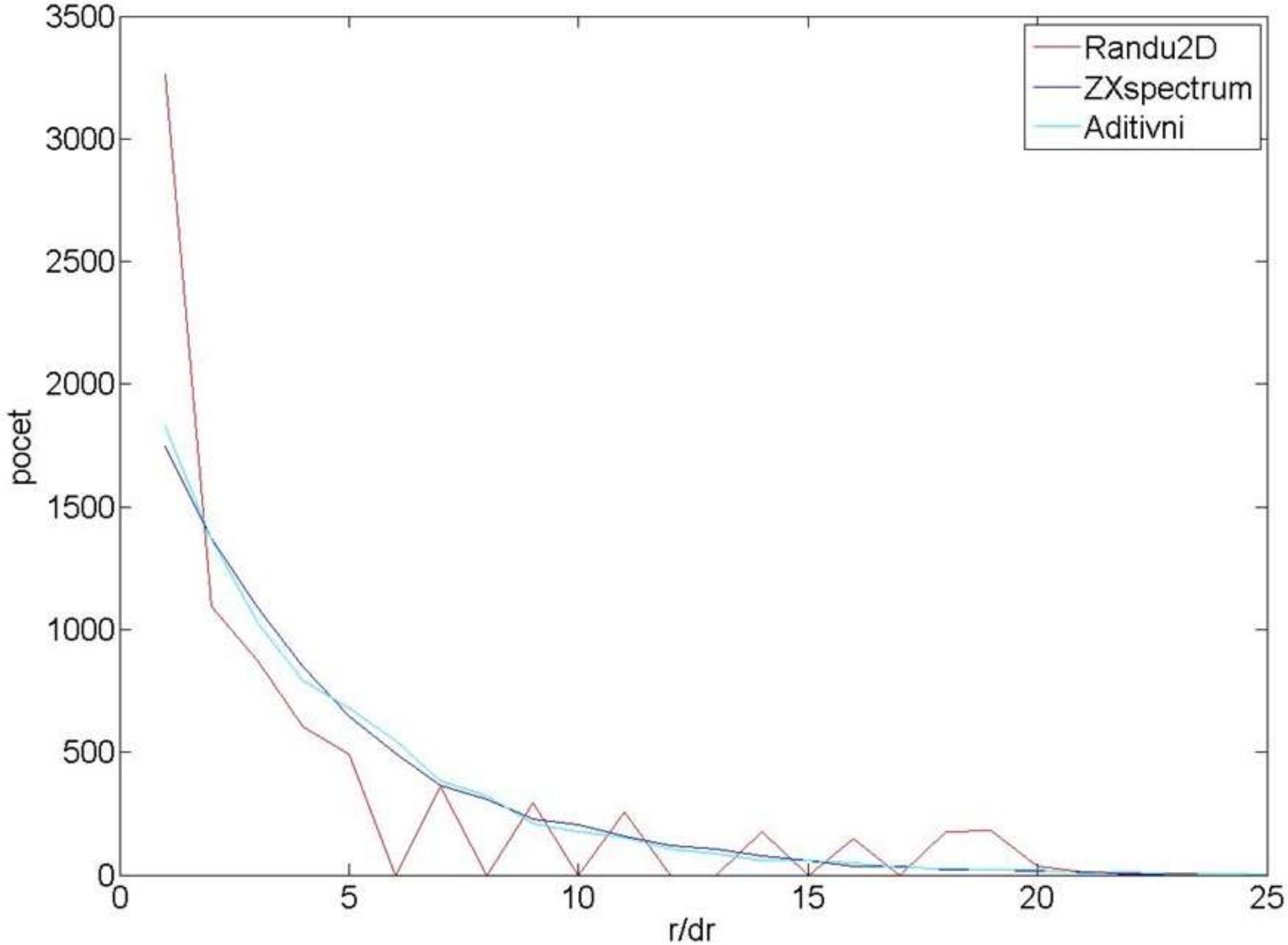
Testy generátorů náhodných čísel

- DIEHARD testy (Marsaglia 1995)
  - sada několika testů, např.
- Birthday spacings
  - Vyberte náhodně dva body na generovaném intervalu, vzdálenost mezi těmito body podléhá exponenciálnímu rozdělení
- Minimum distance test
  - Náhodně umístěte $n = 8\,000$ bodů do oblasti o velikosti 10 000 x 10 000 bodů, změřte vzdálenosti d mezi všemi $\frac{n(n-1)}{2}$ páry. Hledáme minimální vzdálenost. Veličina d^2 bude podléhat exponenciálnímu rozdělení se střední hodnotou 0.995
- Runs test
  - Generujte reálná náhodná čísla na intervalu (0,1); počítejte kdy dojde k růstu čísla a kdy k poklesu, tyto počty splňují určité rozdělení
- Parking lot test
  - Náhodně umístěte jednotkovou kružnici do čtverce o hraně 100 bodů; pokud se kružnice překrývají, zkuste to znovu; po 12,000 pokusech by počet úspěšně umístěných kružnic měl splňovat normální rozdělení
- Count-the-1's test on a stream of bytes
  - Uvažuje řetězec bajtů. Každý bajt může obsahovat 0 až 8 jedniček s pravděpodobností 1/256, 8/256, 28/256, 56/256, 70/256, 56/256, 28/256, 8/256, 1/256.
  - Nyní nechme vytvořit řetězec překrývajících se 5-písmenných slov. Každé písmeno nabývá hodnot A, B, C, D, a E. Písmena jsou určována počtem jedniček v bajtu: 0, 1 nebo 2 znamená A, 3 B, 4 C, 5 D a 6, 7 nebo 8 E. Takže je tu 55 možných 5-písmenných slov a v řetězce 256 000 (překrývajících se) 5-písmenných slov se spočítají výskyty jednotlivých slov

Testy generátorů náhodných čísel

- TestU01 (L'Ecuyer, Simard)
  - Knihovna v ANSI C.
  - Nástupce DIEHARD testů
  - Testy rozděleny na moduly
    - Implementace generátorů (předprogramované)
    - Implementace statistických testů
    - Implementace známých sad testů
    - Aplikace testů na generátor
    - SmallCrush, Crush, BigCrush: sady testů
  - Implementované generátory: http://simul.iro.umontreal.ca/indexe.html

Minimum distance test



GENEROVÁNÍ JINÝCH ROZDĚLENÍ

Rozehrání náhodné veličiny

- Generování a transformace náhodné veličiny (rozehrání NV)

- Co umíme
  - generace náhodných čísel z rovnoměrného rozdělení
- Co potřebujeme
  - náhodnou veličinu X s jiným typem rozdělení
  - se zadanou hustotou pravděpodobnosti $f_X(x)$ či distribuční funkcí $F_X(x)$

- Metody
  - pro diskrétní NV
  - pro spojité NV
    - metoda inverzní funkce
    - metoda výběru
    - metoda superpozice

Rozehrání diskrétní náhodné veličiny

- Náhodná veličina (NV) X

$$X = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ p_1 & p_2 & \dots & p_n \end{pmatrix} \qquad p_i = P(X = x_i)$$

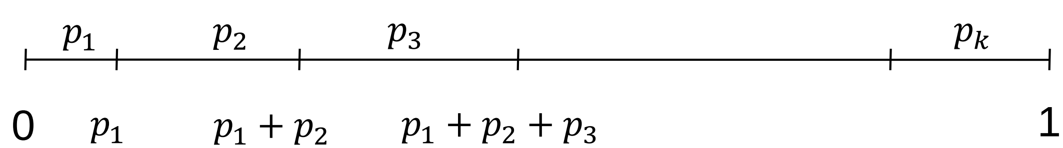
- Vytvoření vektoru (o n složkách)

$$(p_1, p_1 + p_2, p_1 + p_2 + p_3, \dots, 1)$$

- Vygenerujeme číslo y z rovnoměrného rozdělení $R(0,1)$
- Určíme, do kterého intervalu padne interval a jemu odpovídající NV X
  - podle podmínky

$$y < \sum_{i=1}^{j} p_i$$

  - První interval j, pro který bude tato podmínka splněna, určí příslušnou hodnotu $X = x_j$

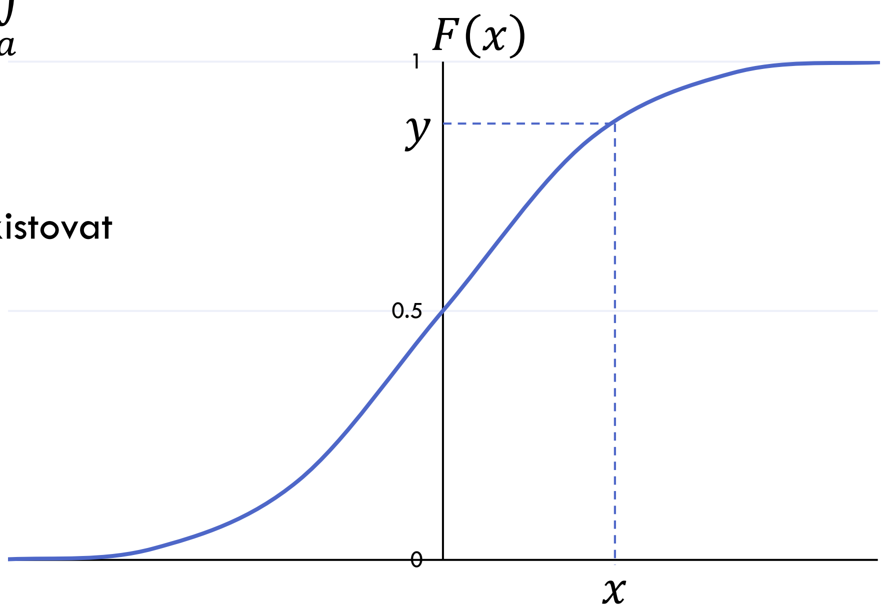


Rozehrání spojité náhodné veličiny

- Metoda inverzní funkce
  - hledáme NV X, hustota pravděpodobnosti $p(x)$, distr. funkce $F(x)$
  - náhodné číslo $y \epsilon \langle 0,1 \rangle$ (z NV Y s rovnoměrným rozdělením $R(0,1)$)
  - potom náhodná veličina $X = F^{-1}(Y)$ má rozdělení s distribuční funkcí $F(x)$

$$y = F(x) \quad \Longrightarrow \quad x = F^{-1}(y)$$

$$\int_a^x p(x)dx = y$$

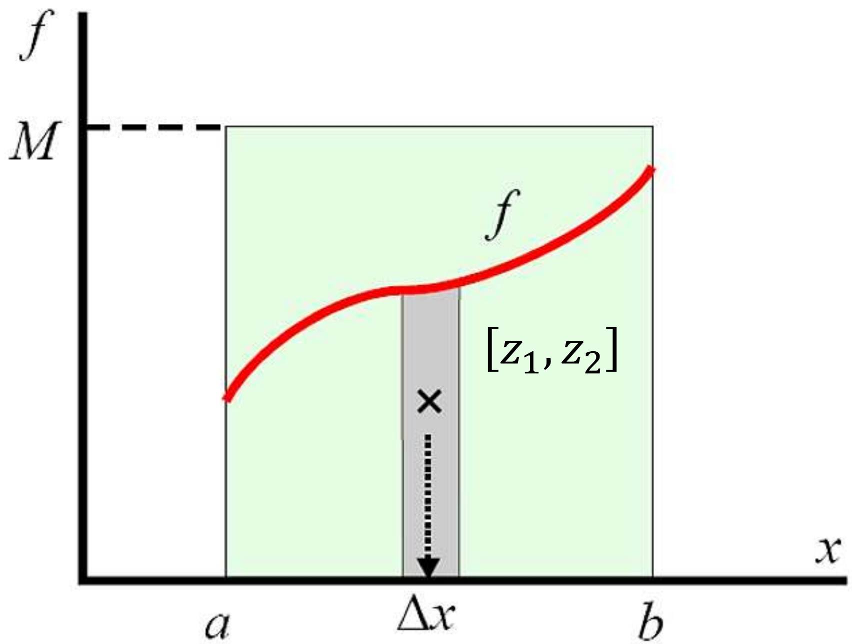


řešíme tuto rovnici, analytické řešení nemusí existovat
výsledkem transformační vztah $x = g(y)$

- např.
- $F(x) = \frac{x-a}{b-a} \Longrightarrow \quad x = y(b-a) + a$
- $F(x) = 1 - e^{-\lambda x} \Longrightarrow \quad x = -\frac{1}{\lambda} \ln y$

Rozehrání spojité náhodné veličiny

- Metoda výběru (von Neumannova)
  - vhodná, když nelze analyticky vyjádřit inverzní funkci
  - $p(x)$ veličiny X je omezená na intervalu $\langle a, b \rangle$
  - volíme $M \geq \sup\big(p(x)\big)$
  - generujeme dvě náhodná čísla veličiny Y: y_1, y_2 (rovnoměrné rozdělení 0 až 1)
  - $z_1 = a + y_1(b - a)$ (mezi a a b)
  - $z_2 = My_2$ (pod M)
  - pokud bod (z_1, z_2) leží pod křivkou $p(x)$, volíme $x = z_1$, jinak opakujeme



Rozehrání spojité náhodné veličiny

- Metoda superpozice (kompoziční)
  - NV X je lineární kombinací jiných spojitých náhodných veličin
  - rozložíme její distribuční funkci $F(x)$ do tvaru

$$F(x) = \sum_{i=1}^{m} p_i F_i(x)$$

    - $F_i(x)$ jsou distribuční funkce, p_i pravděpodobnosti
    - $p_1 + \cdots + p_m = 1, \qquad \mathrm{p}_i > 0$

  - Zavedeme diskrétní NV Z s rozdělením

$$Z = \begin{pmatrix} 1 & 2 & \dots & m \\ p_1 & p_2 & \dots & p_m \end{pmatrix}$$

  - generujeme dvě nezávislé hodnoty y_1 a y_2 veličiny Y (rovnoměrné rozdělení)
  - rozehrajeme číslem y_1 hodnotu $\mathrm{Z} = k$; $\quad k = 1, \dots, m$ (číslo intervalu)
    - rozehrání diskrétní NV
  - z rovnice $F_k(x) = y_2$ určíme x
    - distribuční funkce veličiny X je rovna $F(x)$