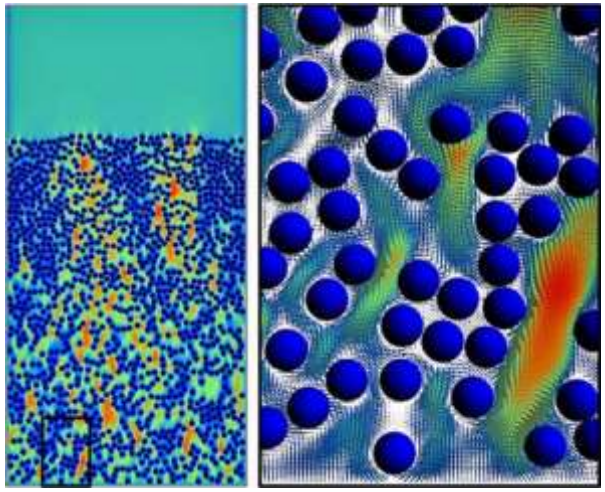
NÁHODNÁ ČÍSLA V NUMERICKÝCH VÝPOČTECH

Náhodná čísla

- Co jsou náhodná čísla:
  - postrádají jakýkoliv vzor (sekvenci, předvídatelnost)
- Proč potřebujeme náhodná čísla?
- Pro vytvoření náhodného stavu nebo náhodné události
  - kryptografie (generování klíčů)
  - simulace fyzikálních procesů
    - náhodné souřadnice molekul
    - Brownův pohyb, chaos
    - radioaktivní rozpad, šum, …
  - modelování lidského chování
    - pohyb lidí, rozhodovací procesy, …
  - videohry
    - nepředvídatelnost
    - události, pohyb nepřátel, směr výstřelu, …
- Nejčastější aplikace
  - počítačové simulace
  - videohry
  - …



Možnosti získání náhodných čísel

- **Generování náhodných čísel**
  - základem vždy pozorování reálného jevu
  - co použít při simulacích?

- Dvě možnosti
- Generátory skutečně náhodných čísel
  - přímé **měření reálného jevu**
  - skutečná náhodnost
    - házení mincí, kostkou, měření přírodních jevů, …
  - malý počet náhodných hodnot
  - náročný proces

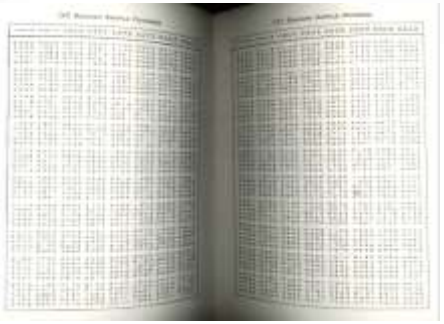
- Vypočtená pseudonáhodná čísla
  - **odhad pravděpodobnostního rozdělení** pozorovaného jevu
  - odhad rozdělení a jeho parametrů (na základě sběru dat)
  - **generování hodnot** z tohoto rozdělení
  - dostatečný počet „náhodných“ hodnot
    - John von Neumann, 1946 prvotní metoda pro rychlé získání náhodných číslic

$$\xi_{i+1} = (\boldsymbol{a} * \xi_i + \boldsymbol{C}) \bmod \boldsymbol{M}$$

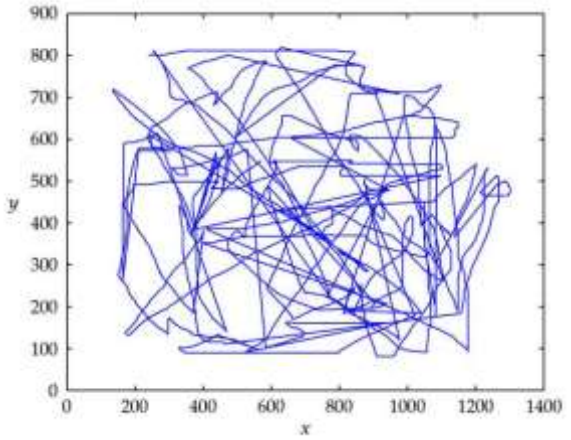
Skutečná náhodná čísla

- Skutečná náhodná čísla
  - založené na **fyzickém** jevu
  - založené na **fyzikálním** jevu
  - tabulky náhodných čísel

fyzikální generátory

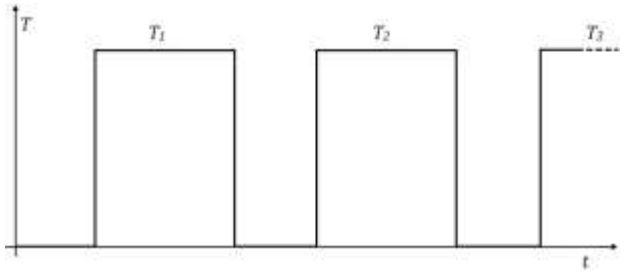


- Generátory založené na **fyzických** jevech
  - např. generování čísla ze sekvence pohybu myší
    - nebo úhozů do klávesnice (prodlevy)
  - problémy
    - člověk může využívat podobný vzor pohybů a úhozů
    - komunikace s OS pomocí bufferů, které mohou náhodnost vyrušit



Fyzikální generátory

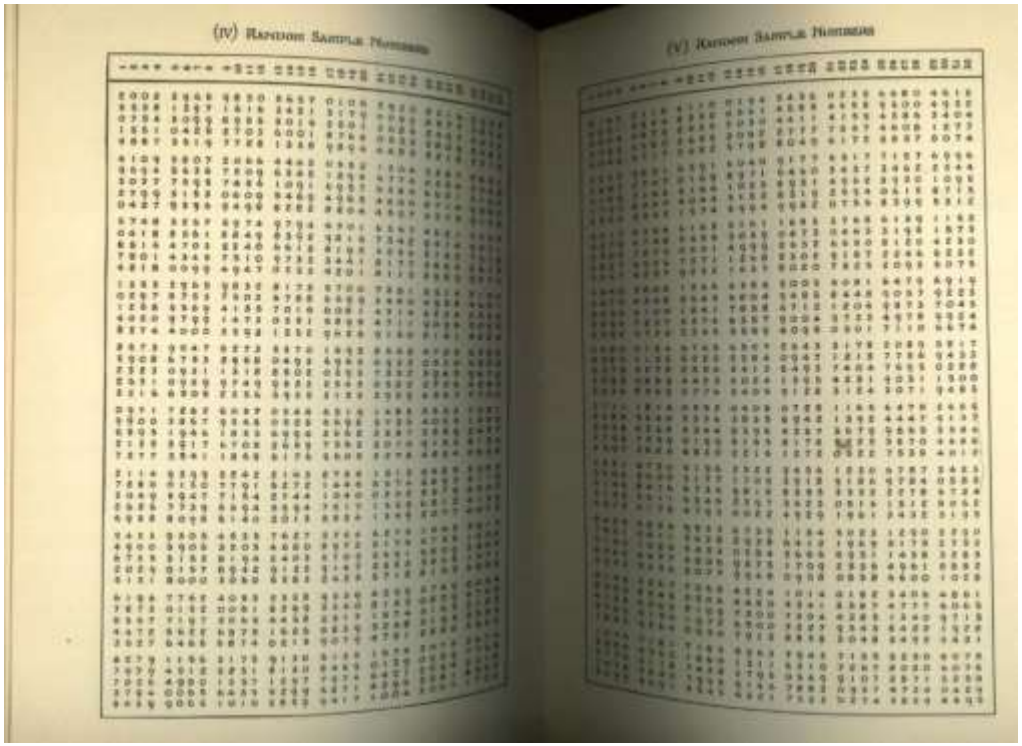
- Generátory založené na **fyzikálních** jevech
  - lepší, ale finančně náročnější
- Nejčastěji využívané jevy
  - radioaktivní rozpad nuklidů
    - Geiger-Müllerův počítač
  - atmosférický šum
    - elektromagnetické vlnění v daném prostoru a čase
    - lze získat citlivou anténou
  - akustický tlak v místnosti (šum z hluku)
    - slabší generátory
    - problém s prediktivními jevy jako hluk z otáček větráku

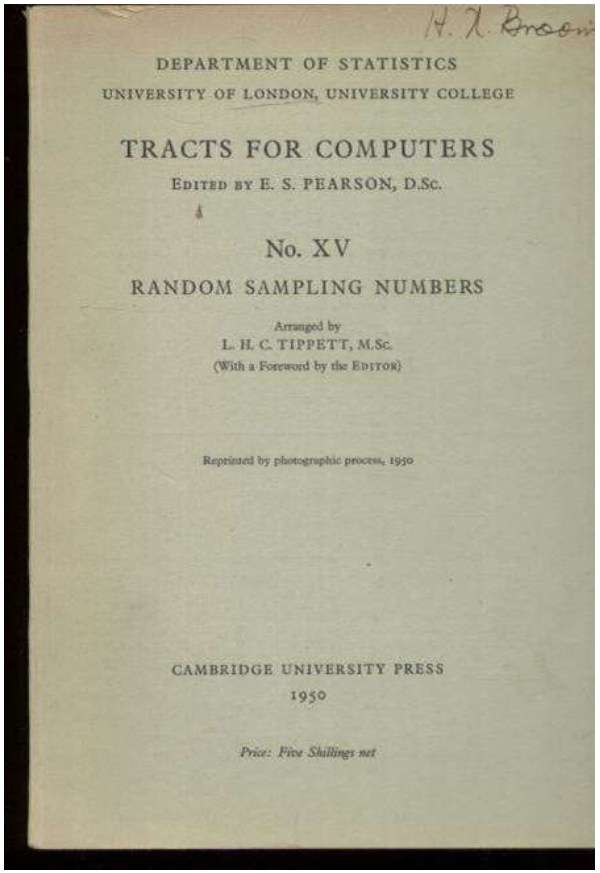


pokud $T_1 > T_2$ – zapíšeme 0
pokud $T_1 < T_2$ – zapíšeme 1

Tabulky náhodných čísel

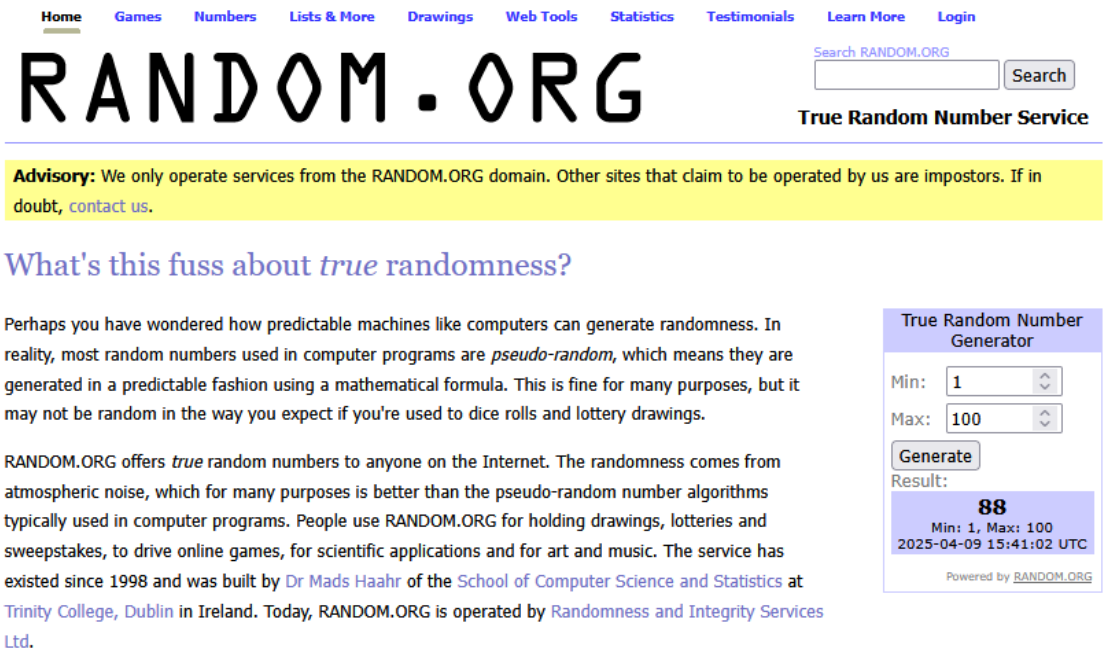
- Opravdu náhodná čísla
  - z fyzikálních generátorů
    - disk, CD nosič, páska
  - rozsáhlé soubory dat (tabulky)
    - 1927 – Tipper - 40 tis náhodných čísel
    - 1955 – RandCorp - 1 mil. náhodných čísel





Weby poskytující náhodná čísla

- Využití externích webových služeb přes jejich rozhraní REST
  - levné, spolehlivé
  - využívají nějaký **fyzický** nebo **fyzikální** jev
- např. random.org
  - generování náhodných číslic z atmosférického šumu
  - potřeba API klíče
    - pro metodu POST z HTTP
  - registrace na https://accounts.random.org/create
  - developer licence zdarma
    - denní limit 1000 čísel



Pseudonáhodná čísla

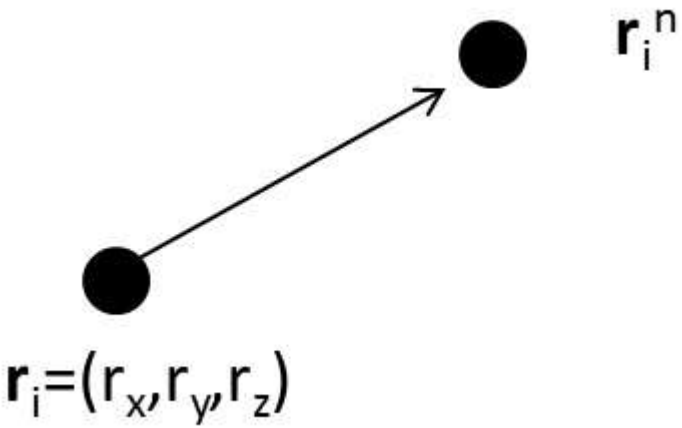
- Pseudonáhodná čísla
  - vypočtená
  - generovány předvídatelným způsobem
    - pomocí matematických metod
  - lineární kongruentní generátory

- 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

- Základní generátory
  - Randu, ZXSPECTRUM, Marsaglia XorShift, Mersene Twister, Park&Miller,...

Využití náhodných čísel

- Např. náhodný posun částice v kapalině
  - generování náhodného čísla ξ v intervalu $(0, 1)$
  - transformace ξ podle odhadnutého rozdělení
    - které lépe popisuje chování částice (např. normální rozdělení)
  - realizace náhodného jevu (sledovaná veličina)
    - posunutí o náhodný vektor



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

$$\underline{\xi} = \left(\xi_{x\prime}, \xi_{y\prime}, \xi_{z\prime}\right)$$

- částici posuneme na nové (náhodné) místo

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

Transformace náhodných čísel

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

- **Lze náhodný posun částice generovat i jiným způsobem?**
- Monte Carlo simulace
  - opakované generování náhodných posunů
    - a vyhodnocování jejich dopadu na chování
  - získáme distribuci možných poloh částice
    - a odhadneme pravděpodobnostní rozdělení
- deterministické algoritmy s náhodnými počátečními podmínkami
  - sledování vzájemných interakcí částic
    - stochastický charakter není zásadní

VYPOČÍTANÁ NÁHODNÁ ČÍSLA

Vypočítaná náhodná čísla

- Pseudonáhodná (kvazináhodná)
  - algoritmus (posloupnost čísel)
  - perioda P
  - náhodné číslo (NČ) $\xi_{i+1} = f(\xi_i, \xi_{i-1}, \xi_{i-2}, \dots \xi_{i-n})$
- Von Neumannovy generátory
- Lineární kongruentní 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
  - řešil problém pomalého čtení NČ
    - z děrných štítků pro počítač ENIAC
  - krátká perioda opakování
- Metoda prostředku čtverce (Middle-square)
    - (prostředních řádů druhé mocniny)
  - zvolíme 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 kongruentní 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: $\xi_{i+1} = (\boldsymbol{a} * \xi_i + \boldsymbol{C}) \bmod \boldsymbol{M}$

- Dělení
  - Multiplikativní generátory
  - Aditivní generátory
  - Smíšené generátory

Lineární kongruentní generátory

- D. H. Lehmer (1948); obecný vztah

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

- Konstanty: $a_0, \ldots, a_k, \quad b, \quad M$
  - volba konstant ovlivňuje vlastnosti (periodu, náhodnost)
  - $k > 0, \ \xi_i < M$
- Semínko (násada) – obecně vektor
  - $(\xi_0, \ldots, \xi_{-k})$ (dosadíme $i = 0$)
- První člen
  - $\xi_1 = (a_0\xi_0 + a_1\xi_{-1} + \cdots + a_k\xi_{-k} + b)(\mathrm{mod}\, M)$
- **Stejná násada = stejná posloupnost čísel**
- Možnost normalizace

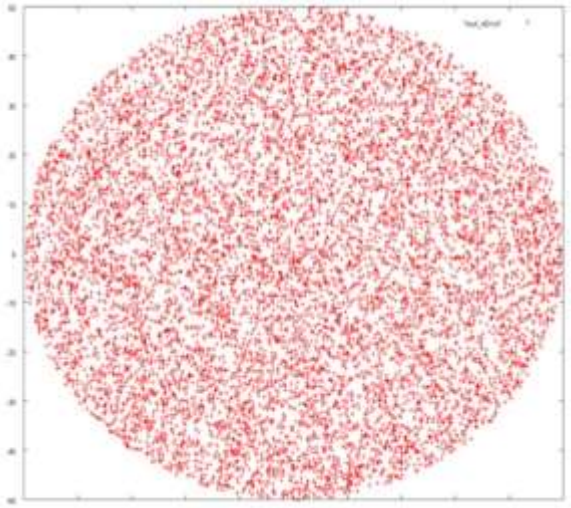
$$\xi_i' = \frac{\xi_i}{M} \rightarrow \quad \xi_i' \in (0,1)$$

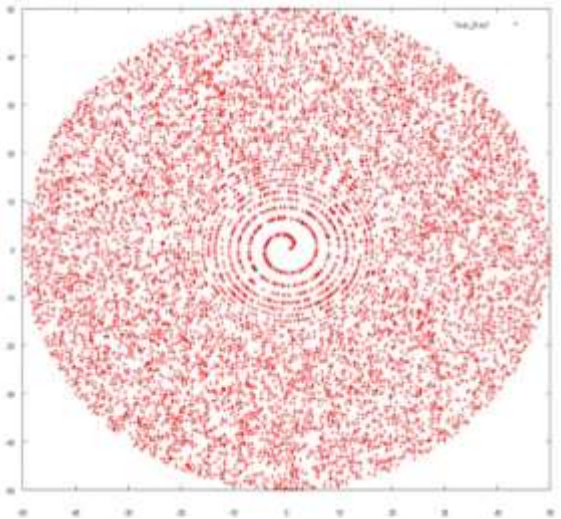
Multiplikativní LCG

- 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 LCG

- Fibonacciho generátor
  - využívá Fibonacciho posloupnost
    - předposlední a poslední hodnotu
  - $\xi_{i+1} = (\xi_i + \xi_{i-1}) (\text{mod } M)$
- Opožděný Fibonacciho generátor (Mitchell, Moore, 1958)
  - využívá obecné hodnoty opožděnosti
    - nepoužívá se, ale jednoduchý
  - $\xi_{i+1} = \left(\xi_{i-j} + \xi_{i-k}\right) (\text{mod } M)$
  - místo "+" může být jiná operace
    - $+ \; - \; * \; /$ ap.

Příklady opožděných Fibonacciho generátorů

- Millerův-Prenticův generátor
  - $\xi_{i+1} = (\xi_{i-1} + \xi_{i-2n}) (\text{mod } 3137) \qquad (P = 9.8 \cdot 10^6)$
- Další aditivní generátory
  - $\xi_{i+1} = (\xi_{i-5} + \xi_{i-17}) (\text{mod } M)$
- Volba M ovlivňuje periodu generátoru

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

Smíšené LCG

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

- Příklady
  - $\xi_{i+1} = (69069\xi_i + 1)(\mathrm{mod}\ 2^{32})$
    - dlouhá perioda
    - simulace, kryptografické funkce

Minimal Standard

- Jednoduchý, ale kvalitní LCG
  - Lewsi, Goodman, Miller (1969)
  - $\xi_{i+1} = a_0 \xi_i (\mathrm{mod}\, M), \qquad a_0 = 16\,807, \quad M = 2^{31} - 1 = 2\,147\,483\,647$
- Vlastnosti
  - plná perioda (rovnoměrné pokrytí)
  - $P = M - 1$
    - nejdelší možná perioda pro čistě multiplikativní LCG
  - projde **testy spolehlivosti**
  - uniformita
    - vybraná subsekvence je nerozeznatelná od celé sekvence
  - nezávislost periody na semínku
    - všechna $\xi_0 \in (1,\ 2^{31} - 1)$ rovnocenná
- Použití
  - **efektivní** implementace pomocí 32-bitové architektury
  - dnes zastaralý
    - stále se používá pro jednoduchost na MC

Posuvné registry

- Posuvné registry (Shift Registers)
  - struktury k ukládání sekvence bitů
    - jeden nebo více registrů
  - možnosti posouvání jednotlivých bitů
    - o jedno nebo více míst
    - vpravo nebo vlevo (shift right, shift left)
    - realizace pomocí kombinační logiky nebo speciálních posuvných instrukcí procesoru
- Generátory pseudonáhodných bitů
  - některé bity se používají jako zpětná vazba do registru ("taps")
  - pro výpočet následujícího čísla se používá kombinace bitů z několika registrů
    - registr "modifikuje sám sebe“ na základě svých minulých hodnot
- 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 → 010101110
    - pravý posun: 001010111 → 000101011

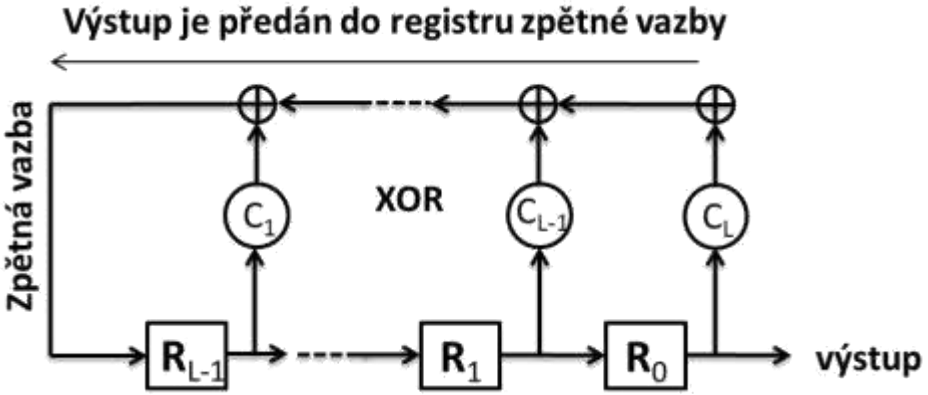
Posuvné registry

- Semínko
  - počáteční stav registrů
  - ovlivňuje celou posloupnost čísel
- Vlastnosti
  - náhodně vypadající sekvence bitů
    - ale je deterministická a periodická
  - konečná perioda
    - např. pro 4 bity 15 různých stavů
  - jednoduchost, extrémní rychlost
  - malá paměťová náročnost
- Použití
  - v kryptografii, simulacích nebo hardwaru
- Příklad 4bitového posuvného registru
  - všechny bity se posunou doprava
  - nový bit vlevo = XOR několika vybraných bitů (taps)
    - zde bity 1 a 4

Krok	Stav registru	Nový bit (vlevo)
0	1001	$1 \oplus 1 = 0$
1	0100	$0 \oplus 0 = 0$
2	0010	$0 \oplus 0 = 0$
3	0001	$0 \oplus 1 = 1$
4	1000	$1 \oplus 0 = 1$
5	1100	$1 \oplus 0 = 1$
6	1110	$1 \oplus 0 = 1$
…	…	…

Generátory s posuvnými registry

- S posuvnými registry s lineární zpětnou vazbou (LFSR)
  - posuvný registr délky L (bitů)
    - L = $\{R_0, R_1, \ldots, R_{L-1}\}$ vnitřních registrů
  - R_i uchovává jednu jednotku informace
    - 1 bit, příp. 2^n bitů



- Princip
  - v každém časovém okamžiku se:
    - obsah R_i přesune do R_{i-1} (doprava)
    - R_0 se předá na výstup
    - do R_{L-1} se uloží (výpočtem) nový obsah
  - vnitřní stav generátoru S = $\{S_0, \ldots, S_{L-1}\}$
    - 2^L různých stavů, kromě nuly (zakázaná)
  - charakteristický mnohočlen

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

  - např.

$$C(x) = x^4 + x + 1$$

    - udělej XOR mezi R_3 a R_3, přičti 1

Iniciace generátoru

Pozor na zakázané stavy logické funkce

Zacyklení generátoru

Iniciace pomocí LCG (Lehmer,…)

Pravidlo 30

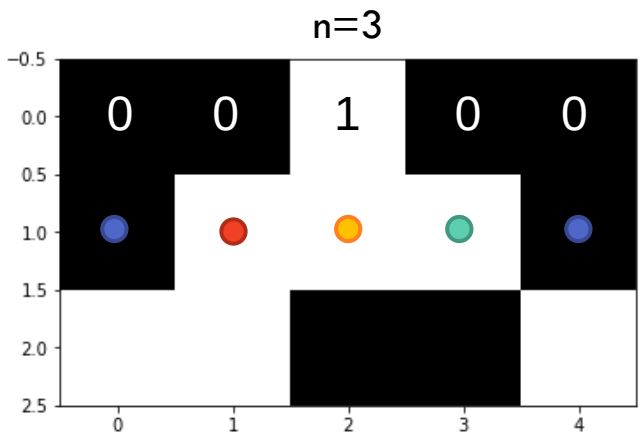
- Stephen Wolfram (1983)
  - název podle decimální reprezentace schémata binárních operací
- Pravidlo 30
  - jedno z nejzajímavějších výpočetních schémat základních automatů
    - pro aktualizaci stavu buněk v 1D buněčném automatu
  - vytváří automat, obsahující pseudonáhodné sekvence
    - aperiodické chaotické sekvence bitů
- Vlastnosti
  - jednoduchá definice
  - přesto velmi komplexní a chaotické vzorce
    - použijeme jako náhodnou posloupnost
- Princip
  - výpočet stavu každé buňky v novém řádku
    - na základě stavů 3 sousedících buněk v předchozí řadě
    - podle pravidla $a \text{ xor } (b \text{ or } c)$

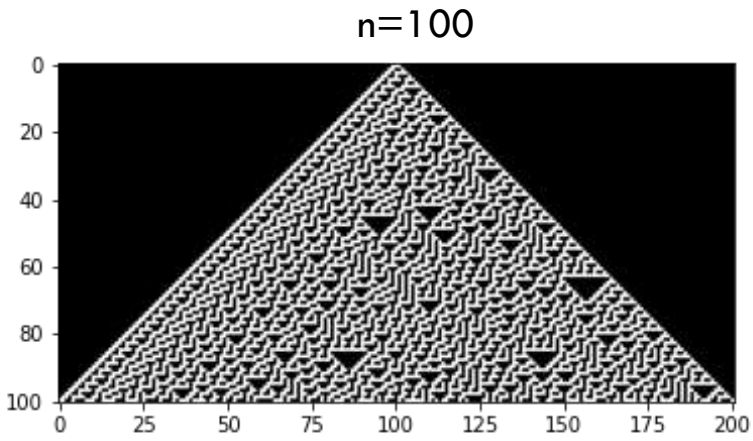
Pravidlo 30

- Stav každé buňky v novém řádku
  - na základě stavů tří sousedících buněk v předchozí řadě
    - podle pravidla $a \text{ xor } (b \text{ or } c)$

$a \text{ xor } (b \text{ or } c)$

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



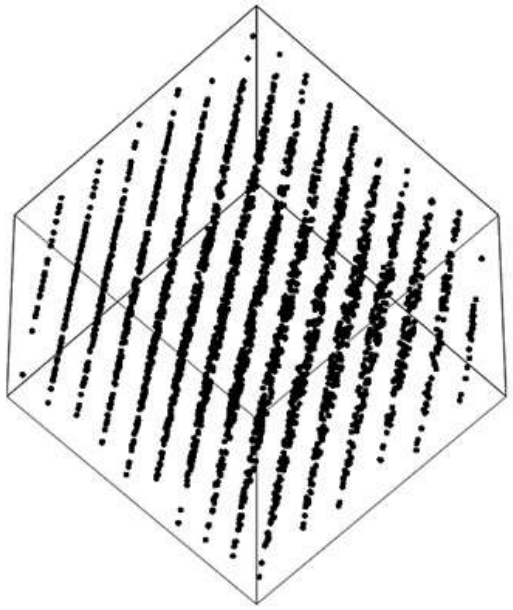


- Postup
  - 1. Nastav prvotní řádek na binární 0, prostřední bit na 1
  - 2. Proveď vývoj automatu do zvolené generace pomocí pravidla 30
  - 3. Vyber prostřední sloupec, přeskoč N bitů (semínko) a získej 8 binárních číslic
  - 4. Vytvoř z M binárních číslic pseudonáhodné číslo

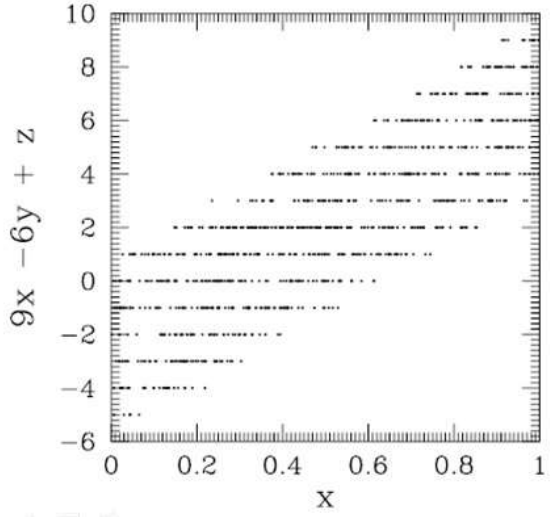
Marsaglia XORshift

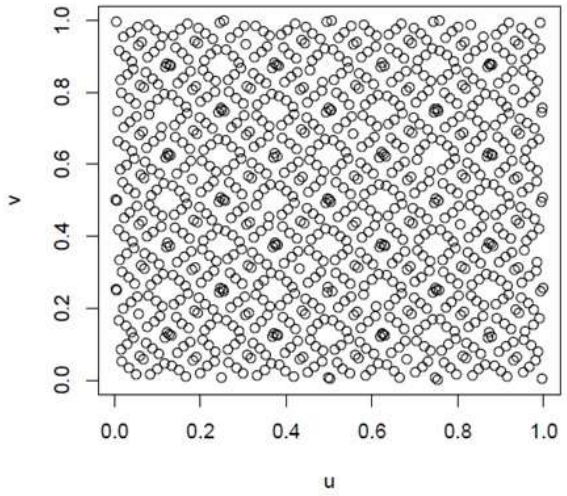
- George Marsaglia (2003)
  - americký matematik (statistik)
  - na základě von Neumannova generátoru
- Postup
  - volíme semínko
    - celé číslo 32 b, 64 b nebo 128 b
  - bitové posuny (různé podle implementace)
    - xa = (x << a)
    - xb = (x >> b)
    - xc = (x << c)
    - např. a = 13, b = 17, c = 5
    - např. Xorshift128 o 23, 17, 26 a 11 míst vlevo
  - posunuté bity se použijí pro XOR s původní hodnotou
    - jinými bity v registru
    - 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

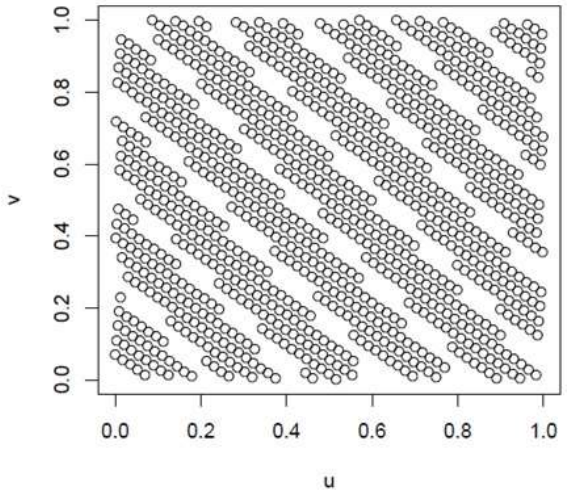
Marsaglia XORshift



- Marsaglia XORshift produkuje sekvenci:
  - $2^{32} - 1$ x celých čísel
  - $2^{64} - 1$ x, y dvojic
  - $2^{96} - 1$ x, y, z trojic
- Nevýhody
  - „Random numbers fall mainly in the planes.“
    - náhodná čísla padají častěji do nadrovin multidimenzionálního prostoru
- Další Marsagliovy algoritmy: Ziggurat, …
  - ke generování náhodných čísel s normálním rozdělením
  - multiplikativní generátory: „Crystalline“ nature







Mersenne Twister

- Založen na generátoru s posuvnými registry
  - rekurzivní sekvence 100 hodnot
  - míchání hodnot pomocí bitových operací (XOR a bitový posun)
  - polynom $C(x)$ s řádem P
  - stavový vektor x

- Generuje stavový vektor x
  - o velikosti w bitů
    - typicky 624 32bit čísel

- $x_{k+n} = x_{k+m} \oplus \left(x_k^u \mid x_{k+1}^l\right) A$
  - x_n je stavový vektor v kroku n
  - x_k^u je horních r bitů vektoru x
  - x_{k+1}^l je dolních r bitů vektoru x
  - | představuje operaci zřetězení
    - "Hello" | "world" → "Hello world"
  - A je transformační matice (konstantní matice 32×32 bitů)
  - $\oplus$ XOR

1. vezme dva po sobě jdoucí stavové prvky
2. vytvoří nové číslo z r horních bitů 1. prvku a r dolních bitů 2. prvku
3. transformační matice A bitově promíchá číslo
4. provede XOR s jiným prvkem z předchozího stavu
5. výsledek uloží jako nový prvek stavového vektoru

Mersenne Twister

- Makoto Mastumoto, Takuji Nishimura, 1998

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

Špatný Seed:
trvá dlouho, než začne generovat náhodnou sekvenci

- Různé verze MT
  - nejvyužívanější verze (nastavení parametrů) MT19937
  - perioda $2^{19937} - 1$ (Mersenneho číslo)
  - ~ 10 000× větší než počet atomů ve viditelné části vesmíru

- Použití
  - dnes jeden z nejpoužívanějších
  - používá Python v modulu Random
  - dále R, Ruby, Free Pascal, PHP, Maple, MATLAB, GAUSS, Julia, Microsoft Visual C++,…
    - numpy jiný (PCG64 z roku 2014)

Mersenneho prvočísla

- název podle Mersenneho prvočísla
  - prvočíslo o jedničku menší než mocnina 2, $M = 2^n - 1$
  - $2^3 - 1 = 7$ (je prvočíslo), $M = 2^4 - 1 = 15$ (není prvočíslo)

- http://www.mersenne.org
  - největší ověřené Mersenneho prvočíslo (r. 2021) 57 885 161 (48. Mersenneho prvočíslo)
  - r. 2006 32 582 657 (44.)

Co je „kvalitní“ RNG?

- Náhodnost
  - čísla nevykazují žádný vzor, trend ani periodicitu
- Uniformita
  - hodnoty pokrývají požadované rozmezí rovnoměrně
    - např. $\langle 0,1)$
- Nezávislost
  - žádné závislosti mezi po sobě jdoucími čísly
- Dlouhá perioda
  - neopakují se brzy
    - důležité pro pseudonáhodné generátory
- Reprodukovatelnost
  - možnost opakovat generaci se stejným semínkem
    - užitečné při ladění

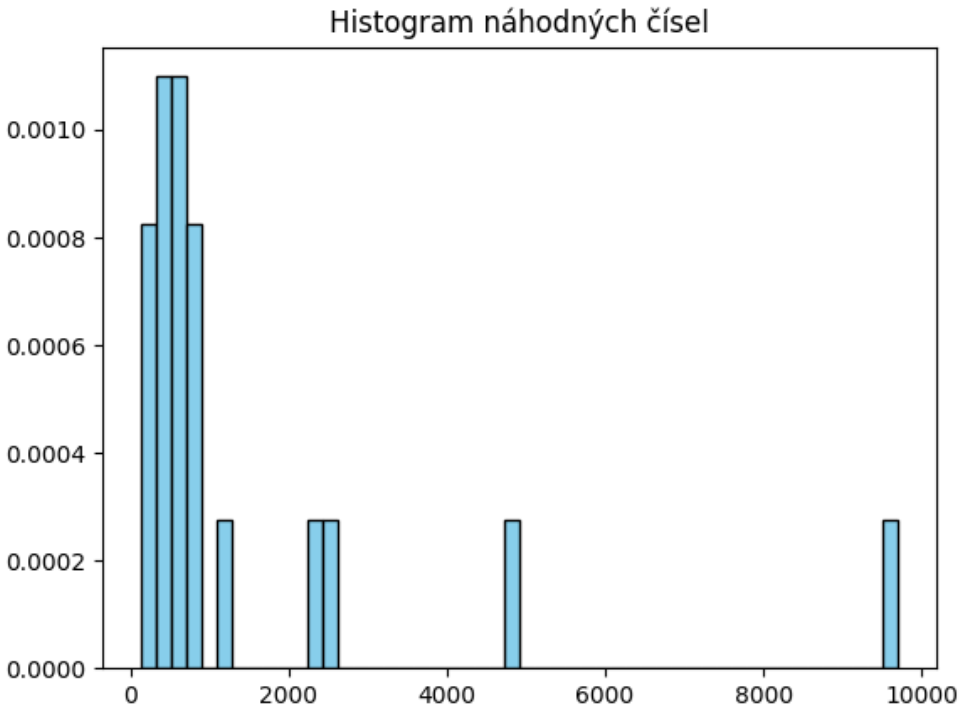
Testy generátorů náhodných čísel

- předpokládejme uspořádanou k-tici náhodných celých čísel $n(k)$

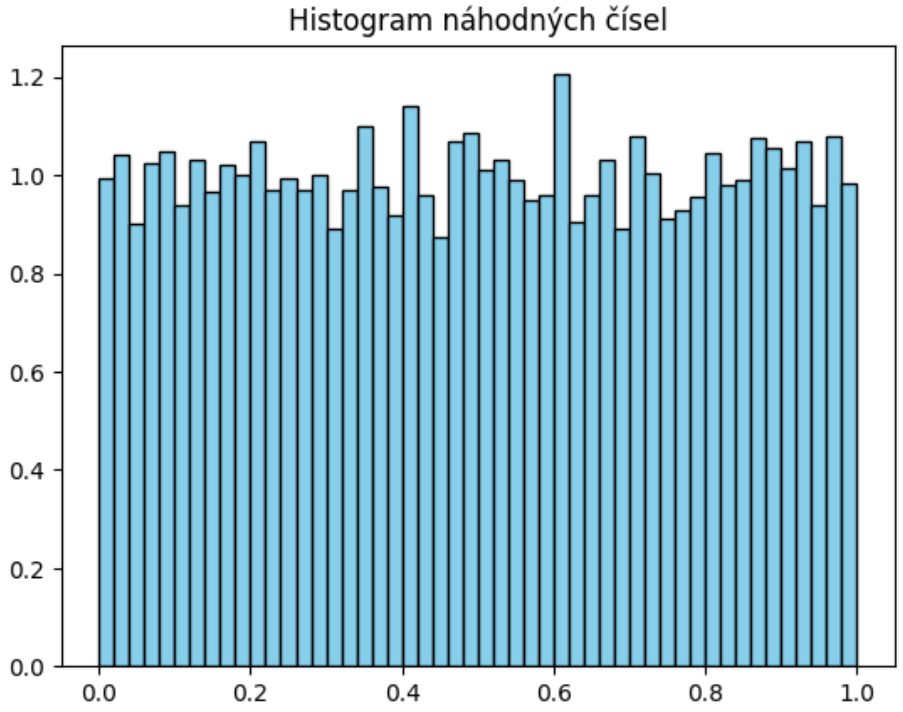
Histogram

- Histogram
  - ukazuje uniformitu

Middle-square (von Neumann)



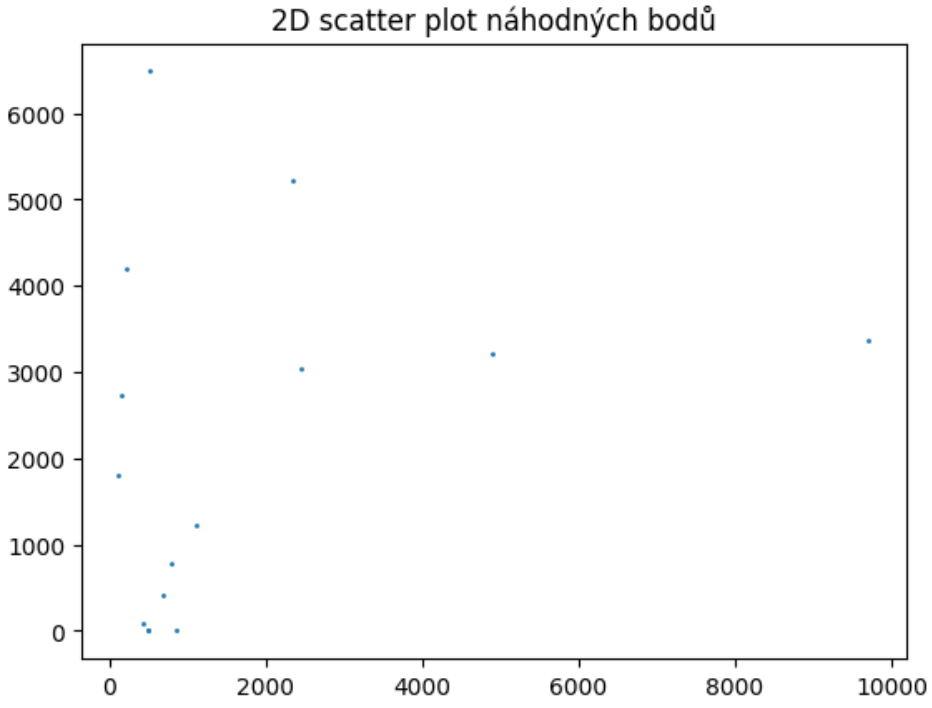
numpy.random



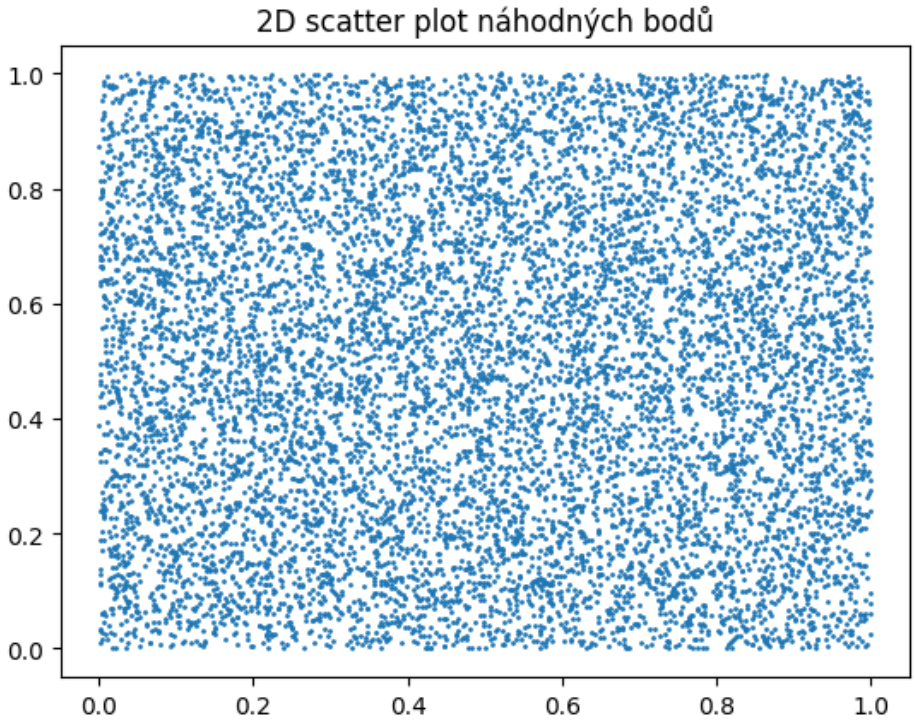
Spektrální analýza

- Spektrální analýza

Middle-square (von Neumann)



numpy.random



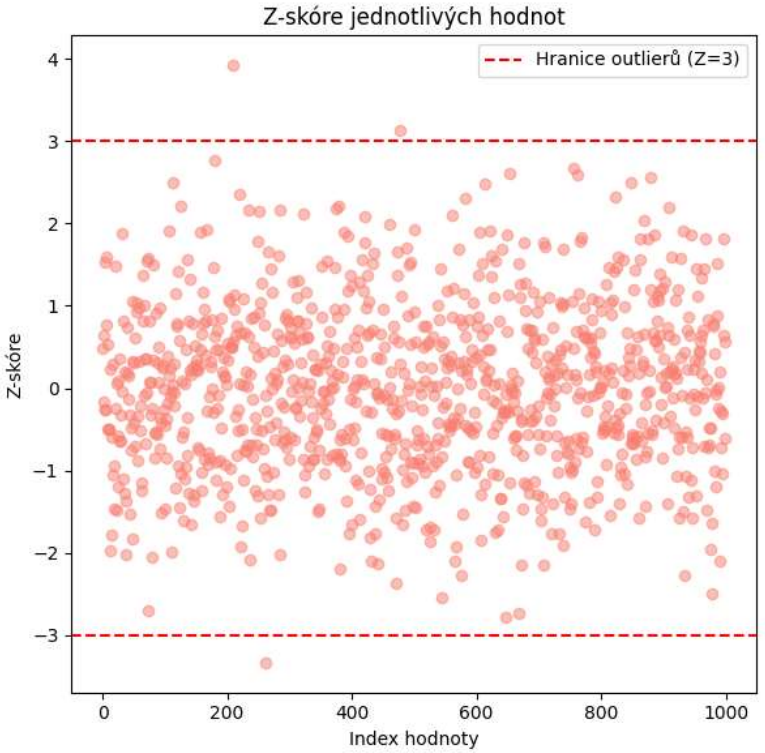
Standardizované (Z) skóre

- Standardizované (Z) skóre
  - pomáhá identifikovat anomálie nebo odchylky
- Princip Z-skóre
  - jak daleko je hodnota od průměru
  - vyjádření v jednotkách směrodatné odchylky

$$Z = \frac{X - \mu}{\sigma}$$

  - X –číslo vygenerované generátorem
  - μ – průměr očekávaného rozdělení
  - σ – směrodatná odchylka očekávaného rozdělení

- Identifikace odchylek
  - pokud Z-skóre překračuje kritickou hodnotu, hodnota je statisticky významně odlišná od očekávání
    - např. ±1,96 pro 95% interval spolehlivosti
    - možný problém s kvalitou generátoru



Další testy náhodnosti

- Statistické testy
  - chí kvadrát test
  - Kolmogorov–Smirnov test
  - srovnání s rovnoměrným rozdělením
- Runs test (test sérií)
  - testuje střídání hodnot nad/pod průměrem
- Autokorelace
  - zjišťuje závislosti mezi čísly
- Komplexita (složitost)
  - Kolmogorovova komplexita: měří složitost k-tice podle počtu znaků, délky programu, který takovou k-tici vyprodukuje
- Transformace (Hadamard, Marsaglia)

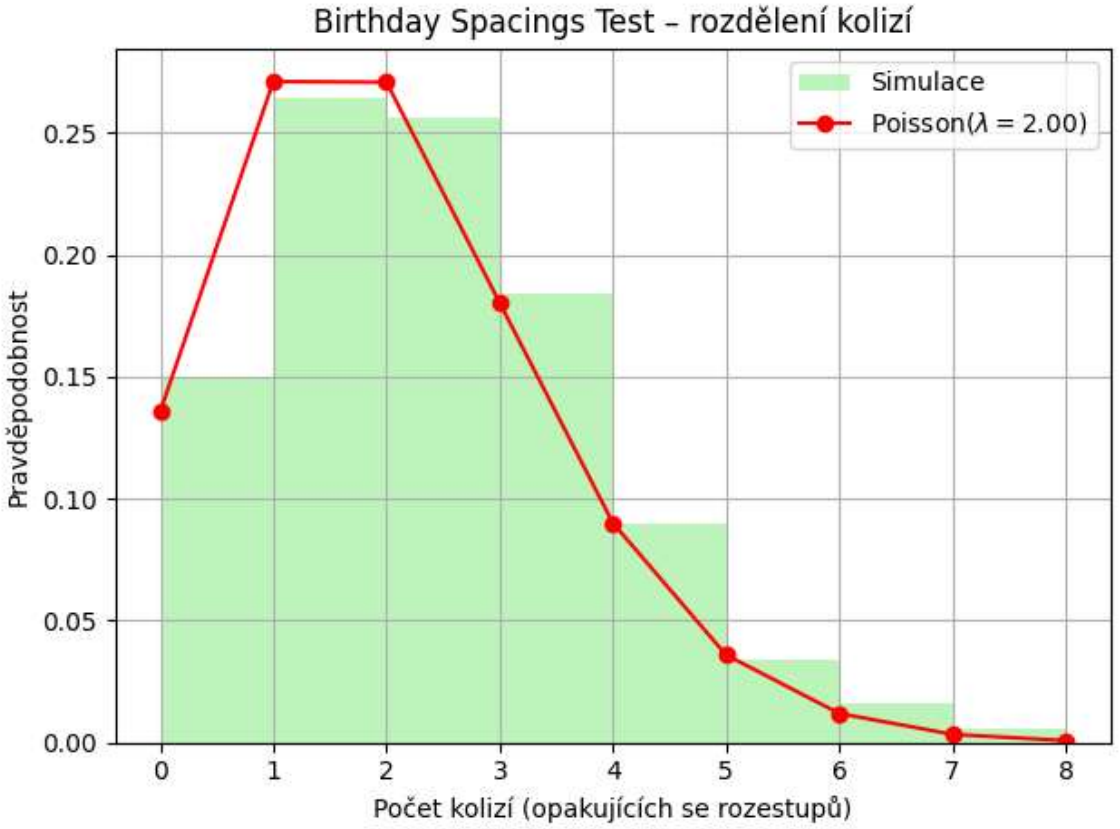
DIEHARD testy

- DIEHARD testy
  - vyvinul George Marsaglia 1995, Stanfordova univerzita
  - sada statistických testů a algoritmů
  - prověřují různé aspekty náhodnosti generovaných číselných posloupností
    - statistická podobnost skutečným náhodným posloupnostem
    - požadované vlastnosti náhodnosti

1. Testy na rovnoměrné rozložení
   - zda je rozložení čísel rovnoměrné v daném intervalu
2. Testy na korelaci
   - mezi jednotlivými čísly v posloupnosti
3. Testy na různorodost
   - různorodost hodnot v posloupnosti
   - další vlastnosti, např. počet unikátních hodnot
4. Testy na sériovou korelaci
   - detekce opakujících se vzorů
5. Testy na permutace
   - kontrola vlastností permutací čísel

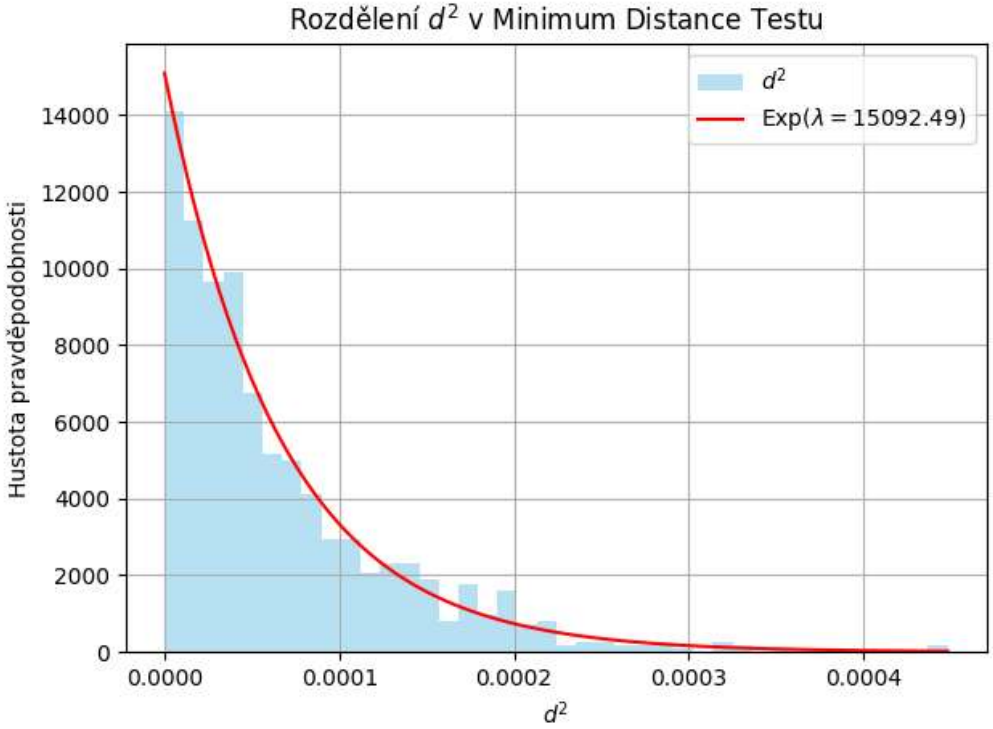
DIEHARD testy

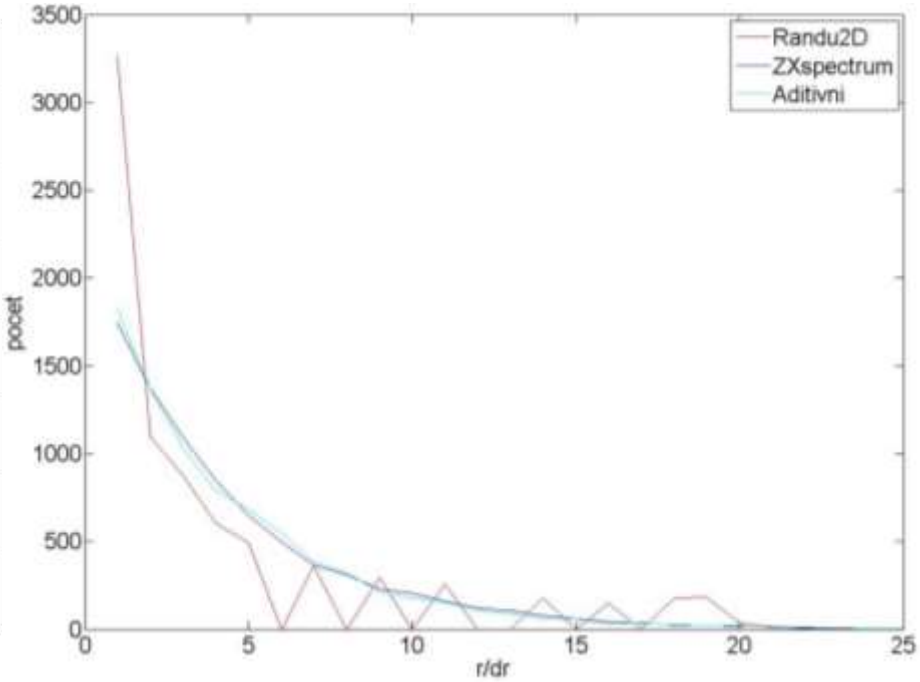
- Birthday spacings
  - vygenerujeme náhodné body v nějakém intervalu
    - např. $\langle 0, T \rangle$
  - setřídíme je a spočítáme rozdíly (spacings) mezi sousedními body
  - sledujeme, kolik těchto rozdílů se opakovalo
    - tj. kolik je "shod" (kolizí)
  - kolizí by mělo být málo
  - opakujeme
    - počty kolizí podle Poissonova rozdělení



DIEHARD testy

- Minimum distance test
  - náhodně umístíme n bodů do roviny
  - změříme vzdálenosti d mezi všemi $\frac{n(n-1)}{2}$ páry
  - hledáme minimální vzdálenost (pro různé pokusy)
    - veličina d_{min}^2 bude podléhat exponenciálnímu rozdělení se střední hodnotou 0.995





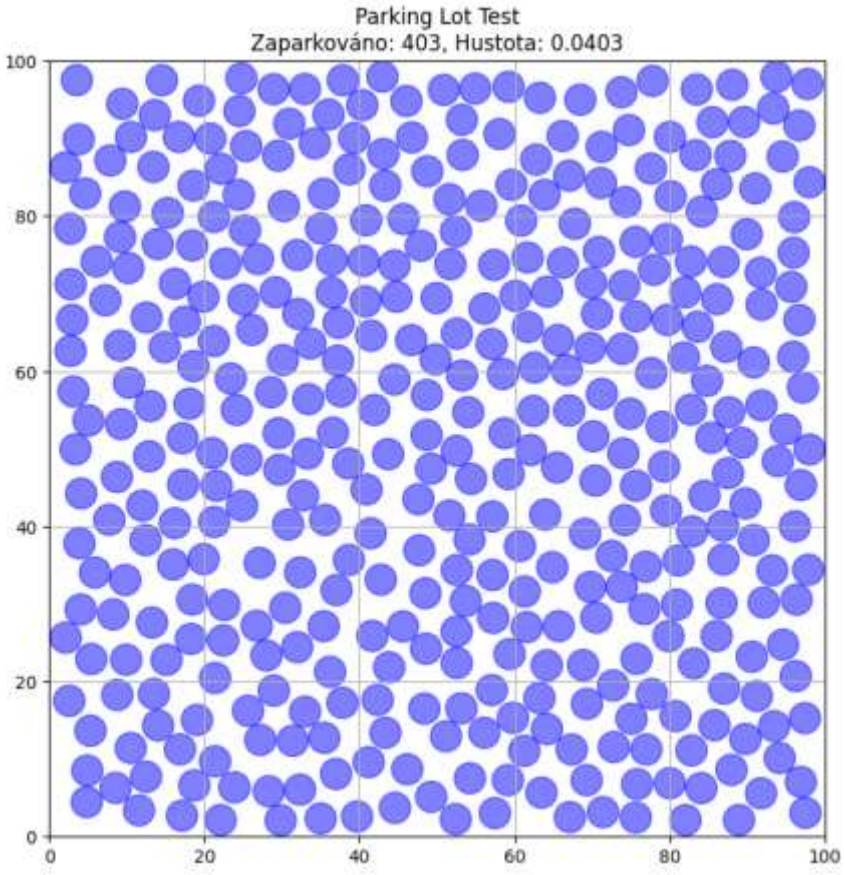
DIEHARD testy

- Runs test
  - Run – posloupnost stejných symbolů
    - ohraničená odlišným symbolem nebo začátkem/koncem sekvence
  - např. + + − − − + − − + + +
    - celkem 5 runs
  - převede se na binární posloupnost (např. pomocí mediánu)
    - nad mediánem hodnoty označeny jako 1 (horní čára)
    - pod mediánem jako 0 (dolní čára)

  - spočítáme standardizované (Z) skóre (p-hodnota)
  - $p > 0.05$ ⟶ data mohou být náhodná, střídání je přiměřené.
  - $p < 0.05$ ⟶ data nevypadají náhodně:
    - příliš málo runs ⟶ hodnoty se „shlukují“ (pomalé střídání, třeba 11110000)
    - příliš mnoho runs ⟶ střídání je „podezřele časté“ (např. 10101010)

DIEHARD testy

- Parking lot test
  - máme čtverec 100×100 bodů ("parkoviště")
  - opakovaně vybíráme souřadnice x, y ve čtverci a kolem nich vykreslíme kruh o poloměru r ("auto")
  - pokud se kružnice překrývají), zkusíme to znovu
    - nejde zaparkovat
  - ukončíme po 12 000 pokusech
    - nebo 1000 úspěšných zaparkování
  - počet úspěšně umístěných kružnic (hustota zaparkovaných aut) se nebude významně lišit od očekávané hodnoty (kolem 0.1)



DIEHARD testy

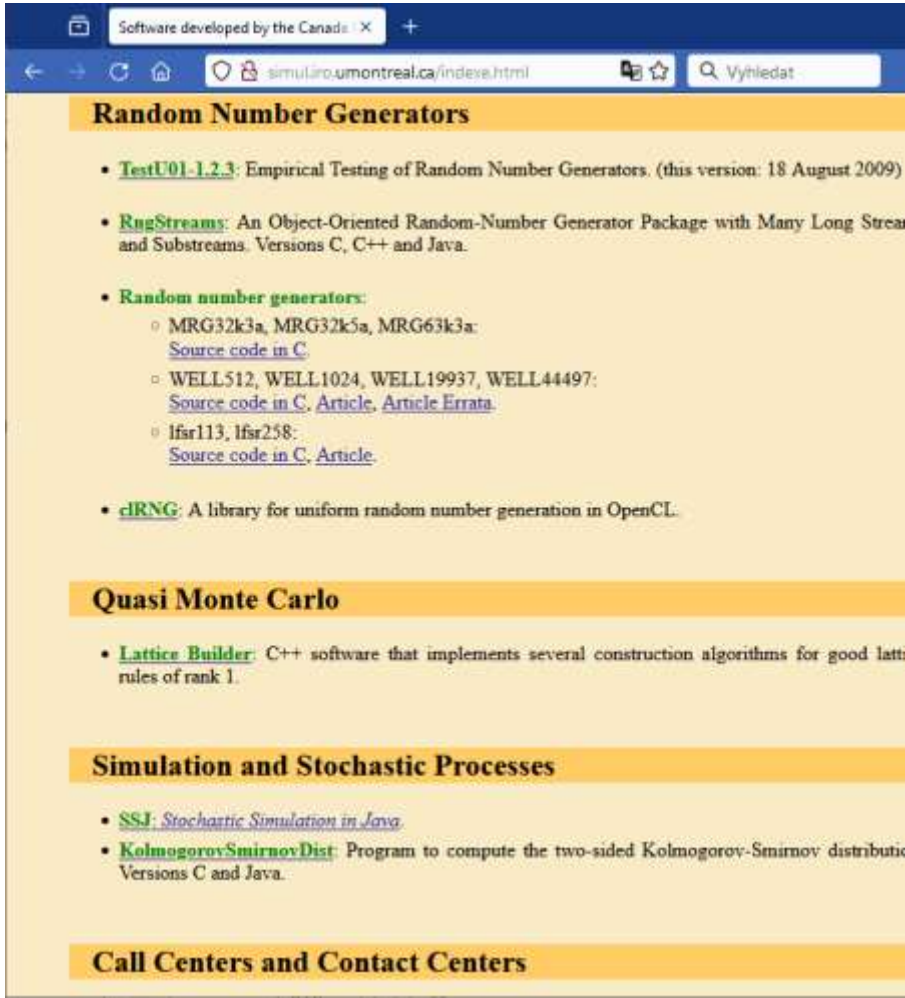
- Count-the-1's test on a stream of bytes
  - jak často se v jednotlivých bytech vyskytují jedničky (1)
  - teoreticky s pravděpodobností 0.5
  - počet jedniček v náhodně generovaném bytu by měl mít binomické rozdělení $(n = 8,\ p = 0.5)$
    - střední hodnota: $\mu = 4$
    - rozptyl: $\sigma^2 = 2$
  - pravděpodobnosti počtu jedniček v bajtu ukazuje tabulka

počet jedniček	P
0	1/256
1	8/256
2	28/256
3	56/256
4	70/256
5	56/256
6	28/256
7	8/256
8	1/256

- Konkrétní implementace
  - 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.
  - Existuje 55 možných 5-písmenných slov, jejich výskyty se spočítají z řetězce 256 000 5-písmenných slov.

Testy generátorů náhodných čísel

- TestU01 (L'Ecuyer, Simard, Université de Montréal 2007)
  - vylepšená verze DIEHARD testů
    - rovnoměrnost, nezávislost, uniformita, rozptyl, …
  - knihovna v ANSI C
  - několik modulů
    - 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



GENEROVÁNÍ JINÝCH ROZDĚLENÍ

Rozehrání náhodné veličiny

- Generování a transformace náhodné veličiny (rozehrání NV)

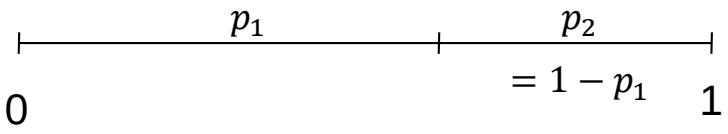
- Co umíme
  - generace náhodných čísel z rovnoměrného rozdělení
- Co potřebujeme
  - náhodnou veličinu X s jiným typem rozdělení
  - se zadanou hustotou pravděpodobnosti $f_X(x)$ či distribuční funkcí $F_X(x)$

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

Rozehrání binomické náhodné veličiny

- Binomická NV
  - diskrétní náhodná veličina
  - může nabývat jedné ze dvou hodnot (úspěch/neúspěch)
  - pravděpodobnost úspěchu p (známe)
  - pravděpodobnost x úspěchů v n pokusech

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



- Vygenerujeme číslo y z rovnoměrného rozdělení $R(0,1)$
- Určíme, do kterého intervalu padne
- Podle intervalu určíme odpovídající NV X

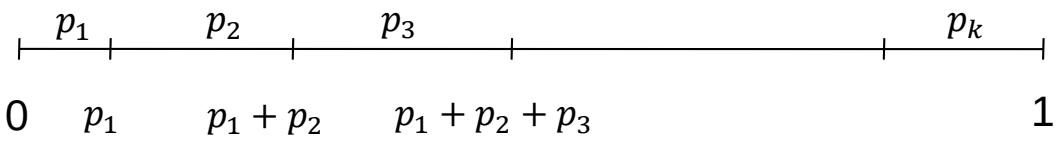
Rozehrání diskrétní náhodné veličiny

- 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)

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



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

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

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

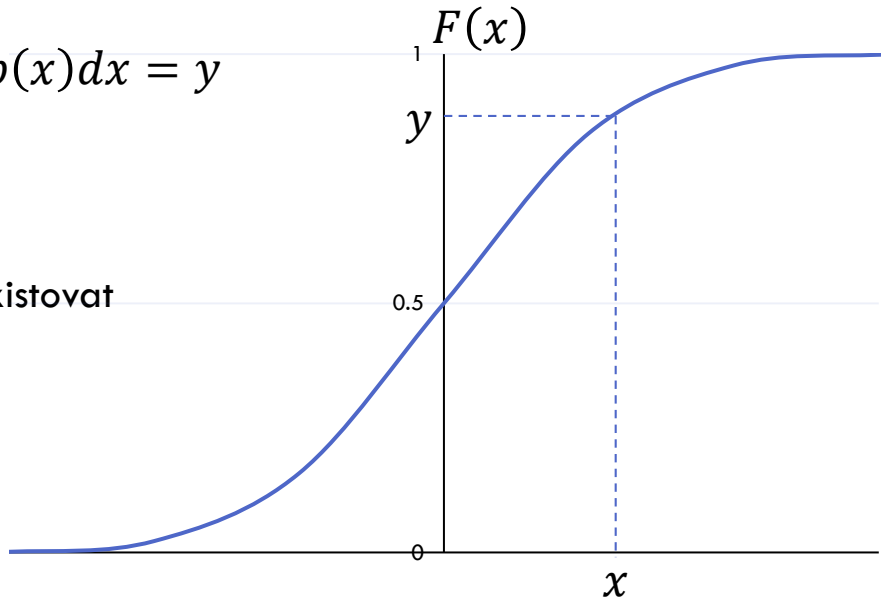
Metoda inverzní funkce

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

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

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} p(x)dx = y$$

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



Vyžaduje znalost analytické formy distribuce

Metoda inverzní funkce – příklady

- Mějme spojitou náhodnou veličinu X s kumulativní distribuční funkcí

$$F(x) = 1 - \exp(-\sqrt{x}) \qquad \text{pro } x \geq 0 \text{ (a rovno 0 jinak)}$$

  - najdeme inverzní funkci $F(x)$ vyjádřením x z rovnice $F(x) = y$

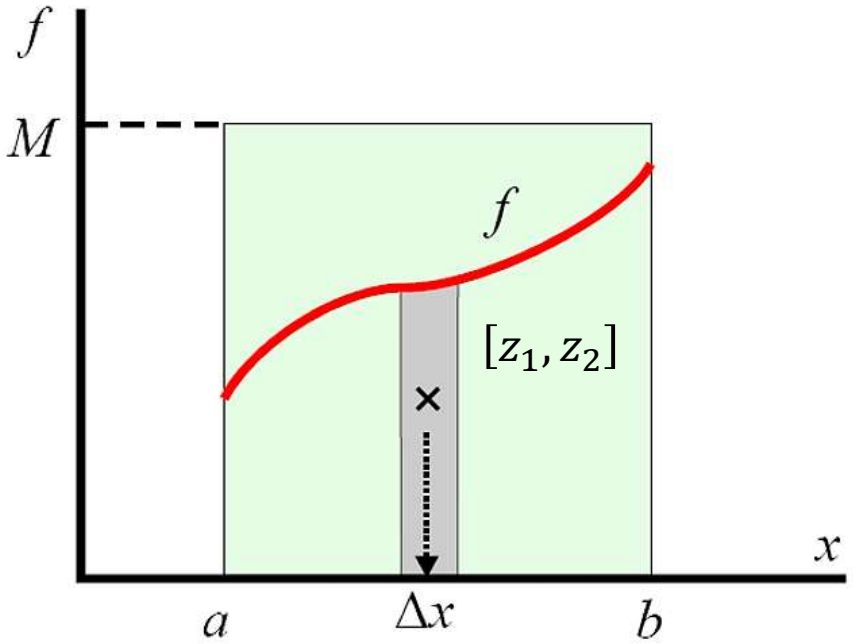
$$x = (\log(1 - y))^2$$

  - vygenerujeme náhodné číslo $y \in \langle 0, 1 \rangle$
  - transformujeme ho pomocí inverzní funkce
  - dostaneme náhodné číslo z rozdělení X

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

Metoda výběru (von Neumannova)

- 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)
  - $z_1 = a + y_1(b - a)$ (x, mezi a a b)
  - $z_2 = My_2$ (y, pod M)
  - pokud bod (z_1, z_2) leží pod křivkou $p(x)$, volíme $x = z_1$, jinak opakujeme



Metoda superpozice (kompoziční)

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

$$F(x) = \sum_{k=1}^{m} p_k F_k(x)$$

    - $F_k(x)$ jsou distribuční funkce, p_k pravděpodobnosti
    - $p_k > 0, \qquad p_1 + \cdots + p_m = 1$

  - Zavedeme diskrétní NV Z s rozdělením

$$Z = \begin{pmatrix} 1 & 2 & \dots & m \\ p_1 & p_2 & \dots & p_m \end{pmatrix} \qquad Z(z = k) = p_k \quad k = 1, \dots, m$$

  - generujeme 2 nezávislé hodnoty y_1 a y_2 veličiny Y (rovnom. rozdělení 0 až 1)
  - rozehrajeme číslem y_1 hodnotu $Z = k; \qquad k = 1, \dots, m$ (číslo intervalu)
    - rozehrání diskrétní NV
  - z rovnice $F_k(x) = y_2$ určíme x
    - distribuční funkce veličiny X je rovna $F(x)$

Metoda superpozice – příklad

- Distribuční funkce $F(x)$ má tvar

$$F(x) = \sum_{k=1}^{m} p_k F_k(x)$$

  - řešíme rovnice

$$p_k > 0, \quad p_1 + \cdots + p_m = 1$$

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

- Máme distribuční funkci odpovídající hustotě

$$p(x) = \frac{5}{12}(1 + (x-1)^4) \qquad x \in \langle 0, 2 \rangle$$

- rozložíme ji do tvaru

$$p(x) = \frac{5}{6} \cdot \left(\frac{1}{2}\right) + \frac{1}{6} \cdot \left(\frac{5}{2}(x-1)^4\right)$$

$$p_1(x) = \frac{1}{2} \qquad p_2(x) = \frac{5}{2}(x-1)^4$$

$$F_1(x) = \frac{x}{2} \qquad F_2(x) = \frac{1}{2}(x-1)^5$$

$$Z = \begin{pmatrix} 1 & 2 \\ \frac{5}{6} & \frac{1}{6} \end{pmatrix}$$

$$x = \begin{cases} 2y_1 & \text{pro } y_2 < \frac{5}{6} \\ 1 + \sqrt[5]{2y_1 - 1} & \text{pro } y_2 \geq \frac{5}{6} \end{cases}$$

Využití centrální limitní věty

- Nechť $X_1, \ldots, X_n$ jsou náhodné veličiny z $R(0,1)$
- potom pro střední hodnotu a rozptyl jejich součtu platí

$$E\left(\sum_{i=1}^{n} X_i\right) = \frac{1}{2}n$$

$$D\left(\sum_{i=1}^{n} X_i\right) = \frac{1}{12}n$$

- k rozdělení $N(0,1)$ se pro $n \to \infty$ blíží veličina

$$X_n = \sqrt{\frac{12}{n}}\left(\sum_{i=1}^{n} X_i - \frac{1}{2}n\right)$$

- prakticky použitelný je vztah pro $n = 12$

$$X_{12} = \sum_{i=1}^{12} X_i - \frac{1}{2}n$$

- Pozn.: platí pro jakékoliv nezávislé a stejně rozložené náhodné veličiny X_i