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

John von Neumann, 1951

Randomisierte Algorithmen

14. Pseudo-Zufallszahlen

Thomas Worsch

Fakultät für Informatik
Karlsruher Institut für Technologie

Wintersemester 2018/2019

Überblick

Überblick

Wünsche

Wünsche

- schnelle Erzeugbarkeit
- lange (gar keine?) Periode
- Reproduzierbarkeit
- portable Implementierbarkeit
- schnelles Vorwärtsspringen

Wünsche

- schnelle Erzeugbarkeit
- lange (gar keine?) Periode
- Reproduzierbarkeit
- portable Implementierbarkeit
- schnelles Vorwärtsspringen

- auch von der zyklischen Folge $0, 1, \ldots, m-1, 0, 1 \ldots$ erfüllt.

- große Aufgabe: Zahlen sollen „zufällig“ aussehen

Wünsche: konkreter

1. Die Zufallszahlen sollen, etwa in $[0; 1[$, gleichverteilt sein.
2. Die Zufallszahlen sollen unabhängig voneinander sein.

1. unter Umständen noch zu garantieren;
2. im allgemeinen verletzt
   - man ist zufrieden, wenn die erzeugten Zahlen „unabhängig aussehen“
   - statistische Abhängigkeiten sollen nicht „einfach“ feststellbar sein

- diese/andere Eigenschaften abhängig von Anwendung
  - unterschiedliche Anforderungen z. B. für Monte-Carlo-Simulationen versus Kryptographie

Physikalische Prozesse

zur Erzeugung von Zufallszahlen:

- keine Gleichverteilung
- keine Unabhängigkeit
- keine Reproduzierbarkeit

(Pseudo-)Zufallszahlengenerator (PRNG)

Struktur $G = (S, s_0, T, U, G)$ mit folgenden Komponenten:

- einer endliche Menge S von *Zuständen*,
- einem *Anfangszustand* $s_0 \in S$,
- einer *Überführungsfunktion* $T : S \to S$,
- einer Menge U von *Ausgabewerten* und
- einer *Ausgabefunktion* $G : S \to U$.

- Generator
  - beginnt in Zustand s_0 und
  - durchläuft Zustände $s_1, s_2, \ldots,$ mit $s_{i+1} = T(s_i)$
  - produziert in Zustand s_i Ausgabe $u_i = G(s_i)$

PRNG (2)

- Zustandsmenge S endlich
- $\Longrightarrow$ Folge $s_0, s_1, s_2, \ldots$ und Folge der Ausgaben wird schließlich zyklisch
- Länge ρ des Zyklus heißt auch *Periodenlänge* des PRNG.

Bauplan vieler Generatoren

- Zustandsmenge: $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$
- Ausgabefunktion: $u_i = s_i / m$ *Rundung?*
- Ausgabewerte: reelle Zahlen im Intervall
  - $[0; 1[$ oder
  - $[0; 1]$
  - *immer überprüfen!*

- Bei aufwändigeren Generatoren:
- Zustand besteht aus $k > 1$ Zahlen $(x_i, x_{i-1}, \ldots, x_{i-k+1})$ kleiner m
- Ausgabefunktion: z. B. $u_i = x_i / m$.

Überblick

Von Neumanns *Middle-square-Generator*

- b-Bit Zahlen.
- Startwert x_0
- aus x_{n-1} ergibt sich x_n so:
  - Man schreibt das Quadrat von x_{n-1} als $2b$-Bit Zahl.
  - Die mittleren b Bits bilden x_n.
- Generator praktisch unbrauchbar: Periodenlänge zu klein

Muddle-square-Generator von Blum/Blum/Shub

Verallgemeinerung von Levin:

- Alle Zahlen sind r Bits lang.
- m: Produkt zweier großer Primzahlen der Form $4\ell + 3$.
- x_0 relativ prim zu m.
- Zahl z als „Maske“
- Sind die Dualzahldarstellungen $x = (a_{r-1} \cdots a_0)_2$ und $z = (b_{r-1} \cdots b_0)_2$ gegeben,
  so sei $x \cdot z = a_{r-1}b_{r-1} + \cdots + a_0 b_0$.

Muddle-square-Generator (2)

- Levins Generator:

$$\begin{aligned} x_{n+1} &= x_n^2 \pmod m \\ u_{n+1} &= x_{n+1} \cdot z \pmod 2 \end{aligned}$$

- **Satz**
  Bei zufälliger Wahl von x_0, m und z übersteht der Generator alle statistischen Tests, die nicht mehr Zeit benötigen als das Faktorisieren natürlicher Zahlen.
- Generator von Blum/Blum/Shub: $z = 1$
- in der Praxis anscheinend keine große Bedeutung

Lineare Kongruenzgeneratoren

- LCG festgelegt durch
  - Anfangswert x_0,
  - Modulus m,
  - Multiplikator a
  - Inkrement c
  - Regel

$$x_n = ax_{n-1} + c \pmod{m}$$

- LCG heißt *multiplikativ* (MLCG), wenn $c = 0$.

LCG (2)

Satz

Der LCG mit

$$x_n = ax_{n-1} + c \pmod{m}$$

hat genau dann volle Periodenlänge, wenn gilt:

- $\text{ggt}(m, c) = 1$,
- q prim und $q|m \implies q|a-1$ und
- $4|m \implies 4|a-1$.

RANDU

- LCG mit $a = 65539$, $c = 0$ und $m = 2^{31}$
- **schlecht**
- lange im IBM/360 Betriebssystem benutzt
- Periodenlänge 2^{29}
- schlechte Gitterstruktur (siehe später)

„Unix“ `rand` und Co.

- `rand`
  - Standard-LCG mit $a = 1\,103\,515\,245$, $c = 12345$, $m = 2^{31}$ und $x_0 = 12345$.
  - **schlecht**
  - vor allem die niedrigwertigen Bits der erzeugten Zahlen
- `drand48`
  - LCG mit $m = 2^{48}$, $a = 25\,214\,903\,917$ und $c = 11$.
  - volle Periodenlänge
  - Besser als `rand`, aber auch er besteht manche statistische Tests nicht.

guter LCG von Marsaglia

LCG mit

- $a = 69069$
- $c = 1$
- $m = 2^{32}$

Mehrfach rekursive Generatoren (MRG)

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k+1} \pmod{m}$$

Konstanten $a_1, \ldots, a_k$
aus dem Bereich $\{-(m-1), \ldots, m-1\}$

Lagged Fibonacci generators (LFG)

Spezialfälle der MRG:

- Name wegen:

$$x_n = x_{n-1} + x_{n-2} \pmod{m}$$

- **zu schlecht**
- besser:

$$x_n = x_{n-r} + x_{n-s} \pmod{m}$$

  für geeignete Zahlen r und s
- statt Addition auch andere Operationen möglich

Anmerkungen zu LFG

- Exklusives Oder in LFG ist **schlecht**.
- Lange Zeit galt $(r, s) = (24, 55)$ und Addition als gut.
  - für $m = 2^e$ Periodenlänge $2^{e-1}(2^{55} - 1)$
  - allerdings z. B. die niedrigstwertigen Bits der x_i nicht „gut“ verteilt.
- Marsaglia (1985) schlägt einige Varianten vor

MRG von L'Ecuyer

$$
\begin{aligned}
x_n = \\
&2\,620\,007\,610\,006\,878\,699\ x_{n-1} + \\
&4\,374\,377\,652\,968\,432\,818\ x_{n-2} + \\
&\quad 667\,476\,516\,358\,487\,852\ x_{n-3} \qquad \mod (2^{31}-1)(2^{31}-2000169)
\end{aligned}
$$

Satz

Ein MRG hat maximale Periodenlänge $\rho = m^k - 1$, falls m prim ist und das *charakteristische Polynom* $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$ primitives Polynom des Körpers $\mathbb{Z}_m$ ist.

Inverser Kongruenzgenerator (ICG)

- p sei prim
- ICG festgelegt durch

$$x_{n+1} = (ax_n^{-1} + c) \pmod{p}$$

- für $x_i = 0$ sei festgelegt: $0^{-1} = 0$

- bemerkenswert gute theoretische Eigenschaften
- bestehen empirische Tests mit sehr guten Ergebnissen.
- Nachteil: nicht ganz einfache Implementierung (Geschwindigkeit des Generators?)
- konkrete Parameter für gute ICG unter `http://random.mat.sbg.ac.at/generators/wsc95/inversive/`.

Mersenne-Twister (Matsumoto/Nishimura)

- alle Zahlen sind Bitvektoren der Länge w
- r ist eine Zahl mit $0 \leq r \leq w - 1$
- zwei Zahlen k und m mit $1 \leq m < k$
- $w \times w$-Matrix A geeignet gewählt
- x^l_{n+1}: die „unteren“ (lower) r Bits von x_{n+1}
- x^u_n: die „oberen“ (upper) $w - r$ Bits von x_n
- $x^u_n | x^l_{n+1}$: die Konkatenation der beiden Bitfolgen
- dann:

$$x_{n+k} = x_{n+m} \oplus (x^u_n | x^l_{n+1})A \; .$$

Mersenne-Twister (2)

- MT19937: konkrete Implementierung eines Mersenne-Twisters
  - Periodenlänge $2^{19937} - 1$
  - ausgezeichnete Eigenschaften bei statistischen Tests
  - mehr: `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`
  - wird z.B. in Python (seit 2.3), R, Ruby, matlab, C++ 11, etc. eingesetzt
- auch MT noch verbesserungsfähig
  - Arbeit von Panneton, L'Ecuyer, Matsumoto: `http://www.iro.umontreal.ca/~lecuyer/myftp/papers/wellrng.pdf`
  - WELL-Generatoren: teils noch längere Periode, und noch bessere Eigenschaften als MT: `http://www.iro.umontreal.ca/~panneton/WELLRNG.html`

Der Zellularautomat „Regel 30“ als PRNG

- Feld von k Bits $x_{k-1}, \ldots, x_1, x_0$
- berechne neue Werte $y_0, \ldots, y_{k-1}$ gemäß *„Regel 30“:*

$x_{i+1}x_ix_{i-1}$	111	110	101	100	011	010	001	000
y_i	0	0	0	1	1	1	1	0

- alle Indexrechnungen $i-1$ und $i+1$ modulo k
- die y_i bilden die Werte x_i für den nächsten Schritt.
- Name wegen der Folge 00011110 der Bits für die y_i
- in Mathematica eine der Möglichkeiten

Regel 30 (2)

- guter Generator, wenn
  - k groß genug (in Mathematica: einige Hundert)
  - Anfangskonfiguration mit mindestens einer 1
  - zunächst größere Zahl von Initialisierungsschritten
  - in einer fest gewählten Zelle die Folge der durchlaufenen Werte benutzt wird
- Generator besteht z. B. alle Test in Marsaglia's `diehard` Test (Schloissnig, 2003)

Kombination mehrerer Generatoren

- Idee: baue einen neuen Generator aus mehreren alten
- häufig: größere Periodenlänge
- manchmal besseres statistisches Verhalten
- zwei Kombinationsverfahren gerne verwendet: …

Shuffling

- zwei Generatoren für Zahlenfolgen x_i und y_i.
- Tabelle mit einer gewissen Anzahl zuletzt erzeugter x_i.
- wenn nächster Wert benötigt:
  - benutze nächstes y_j, um zufällig Wert aus Tabelle zu wählen
  - entsprechender Platz wird mit nächstem neuen x_i gefüllt
- Nachteile:
  - theoretisch nicht gut verstanden
  - schnelles Überspringen von Werten unklar

zweite Kombinationsmethode

- aus Zufallsfolgen $x_0, x_1, \ldots$ und $y_0, y_1, \ldots$
  erzeuge neue Folge $z_0 = x_0 \bullet y_0, z_1 = x_1 \bullet y_1, \ldots$
- Periodenlänge kann größer werden
- Uniformität der Verteilung im allgemeinen besser
- (Pseudo-)Unabhängigkeit aufeinanderfolgender Werte im allgemeinen größer

Überblick

Qualität von PRNG

- Wie stellt man fest, ob ein PRNG „gut“ ist?
- Was heißt überhaupt „gut“?

- verschiedene Tests

Tests

- Testergebnisse sind üblicherweise Zahlen
  (z.B. zwischen 0 und 1),
  die als Qualitätsangabe interpretiert werden können.
- theoretischer Test:
  - Werte explizit berechenbar
- empirischer Test:
  - untersuche lange Folge erzeugter Zufallszahlen

Chi-Quadrat-Statistik

- $s_1, \ldots, s_k$: mögliche Ergebnisse eines Zufallsexperiments,
- die mit Wahrscheinlichkeiten $p_1, \ldots, p_k$ auftreten.

- n unabhängige Zufallsexperimente:
  Y_i: absolute Häufigkeit, mit der Ergebnis s_i auftritt
- also $\sum_i Y_i = n$ und $\mathbf{E}[Y_i] = np_i$.
- χ^2-Statistik der beobachteten Größen $Y_1, \ldots, Y_k$:

$$V = \sum_{i=1}^{k} \frac{(Y_i - \mathbf{E}[Y_i])^2}{\mathbf{E}[Y_i]}$$

Chi-Quadrat-Statistik (2)

- Einsetzen von $\mathbf{E}[Y_i] = np_i$ ergibt

$$
\begin{aligned}
V &= \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i} \\
&= \sum_{i=1}^{k} \frac{Y_i^2 - 2Y_i np_i + n^2 p_i^2}{np_i} \\
&= \frac{1}{n} \sum_{i=1}^{k} \frac{Y_i^2}{p_i} - 2n + n \\
&= \frac{1}{n} \sum_{i=1}^{k} \frac{Y_i^2}{p_i} - n \ .
\end{aligned}
$$

Chi-Quadrat-Statistik (3)

- Wie ist die Größe V verteilt?
- Tabellen mit Näherungswerten enthalten
  - für jedes $v = k - 1$ (*Anzahl der Freiheitsgrade*) eine Zeile
  - für einige p, z. B. 1%, 5%, …, 99% eine Spalte
- Beachte: Anzahl n geht nicht ein; muss groß genug sein;
  Daumenregel: n so groß, dass jedes $s_i \geq 5$.
- dann besagt Wert x in Zeile $v = k - 1$ und Spalte p:
  V ist kleiner oder gleich x mit Wahrscheinlichkeit p.

Chi-Quadrat-Statistik: Beispiel 1

-

p	0.01	0.05	0.25	0.5	0.75	0.95	0.99
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.72

- ist $k = 12$, also $\nu = 11$, und
  hat man in einer Versuchsreihe Wert $V = 29.44$ ermittelt,
  dann liest man aus der Tabelle ab:
  In 99% der Fälle ist $V \leq 24.72$, also ist $V > 24.72$ und erst recht $V > 29.44$ in höchsten einem Prozent der Fälle.
- Also: Es ist sehr unwahrscheinlich, dass die Versuchsreihe der angenommenen Verteilung genügt.

Chi-Quadrat-Statistik: Beispiel 2

p	0.01	0.05	0.25	0.5	0.75	0.95	0.99
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.72
- Wenn die Y_i alle sehr sehr gut den Erwartungswerten entsprechen, dann ist V klein, etwa $V = 1.17$.
- Tabelle: auch das passiert in weniger als 1% der Fälle

Kolmogorov-Smirnov-Test (KS-Test)

- gegeben: n *unabhängige* Zufallsexperimente mit Ergebnissen $X_1, \ldots, X_n$.
- diese induzieren eine empirische Verteilungsfunktion

$$F_n(x) = \frac{|\{i \mid X_i \leq x\}|}{n} \; .$$

- Zum Vergleich mit einer vorgegebenen kontinuierlichen Verteilungsfunktion $F(x)$ berechnet man die Größen

$$K_n^+ = \sqrt{n} \max_{-\infty<x<+\infty} (F_n(x) - F(x)) \qquad \text{und}$$
$$K_n^- = \sqrt{n} \max_{-\infty<x<+\infty} (F(x) - F_n(x))$$

Kolmogorov-Smirnov-Test (2)

- Für die K_n^+ und K_n^- gibt es wieder Tabellen mit Zeilen für verschiedene n und Spalten für verschiedene p:
- Wert x in Zeile n und Spalte p: K ist kleiner oder gleich (bzw. größer) x mit Wahrscheinlichkeit p.

- mögliche Anwendung:
  - zunächst mehrere χ^2-Tests
  - Verteilung für χ^2 (jedenfalls näherungsweise) bekannt
  - auf die sich ergebenden Wahrscheinlichkeiten KS-Test anwenden
- analog:
  - Verteilung der K_n ist (näherungsweise für große n) bekannt und darauf weiterer KS-Test anwendbar.

Einfache empirische Tests

- Ziele: Informationen über
  - die vermutliche Art der Verteilung
  - die Un-/Abhängigkeit der erzeugten Zahlen
  - eventuelle Korrelationen
- Frei verfügbare Programmpakete zum Beispiel:
  - *Diehard*: `http://stat.fsu.edu/pub/diehard/`
  - *dieharder*: `https://www.phy.duke.edu/~rgb/General/dieharder.php`
  - *TestU01*: `http://www.iro.umontreal.ca/~simardr/testu01/tu01.html`
    - enthält auch Implementierungen von fast 200 PRNG („some are good and many are bad“)
  - *NIST statistical test suite*: `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`

Vereinbarung

- $U_0, U_1, \ldots$ Folge reeller Zahlen,
  angeblich unabhängig und gleichverteilt in $[0, \ldots, 1[$.
- Werden für einen Test nichtnegative ganze Zahlen benötigt, dann statt dessen die Folge der $Y_i = \lfloor dU_i \rfloor$ (für geeignetes $d \in \mathbb{N}$)

EinfacheTest

- am simpelsten:
  stelle fest, ob unter den Y_i jeder mögliche Wert $0, \ldots, d-1$ gleich oft vorkommt.

- *Serial Test*:
  - analog für Paare, Tripel, etc.
  - beachte: benutze *nicht überlappende* Vektoren
    $\mathbf{Y}_i = (Y_{si}, Y_{si+1}, \ldots Y_{si+s-1})$
    (und *nicht* Vektoren, die sich überlappen)

Lückentest (gap test)

- zwei relle Zahlen $0 \leq \alpha < \beta \leq 1$ gegeben
- Betrachte die Längen der maximalen Teilfolgen $U_j, \ldots, U_{j+r}$, so dass $\alpha \leq U_{j+r} \leq \beta$ aber alle vorherigen Werte nicht. Man spricht dann von einer Lücke der Länge r.
- Für $p = \beta - \alpha$ ist $p_r = p(1-p)^r$ die Wahrscheinlichkeit für eine Lücke der Länge r.
- χ^2-Test anwendbar

Permutationstest

- betrachte nicht überlappende Vektoren
  $\mathbf{U}_i = (U_{si}, U_{si+1}, \ldots U_{si+s-1})$
- bestimme Permutation π der Indizes, die die Komponenten sortiert.
- erstelle Statistik, welche Permutation wie häufig auftritt
- vergleiche mit der theoretisch gegebenen Gleichverteilung

Maximum-of-t Test

- Berechne die Größen $V_i = \max\{U_{si}, U_{si+1}, \ldots U_{si+s-1}\}$
- Falls die U_j gleichverteilt sind, ist

$$\begin{aligned} \mathbf{P}(V_i \leq x) &= \mathbf{P}(\max\{U_{si}, U_{si+1}, \ldots U_{si+s-1}\} \leq x) \\ &= \prod \mathbf{P}\left(U_{si+j} \leq x\right) \\ &= x^s \ . \end{aligned}$$

- Diese Hypothese kann mit einem KS-Test überprüft werden.

Geburtstagsabstandstest (birthday spacings test)

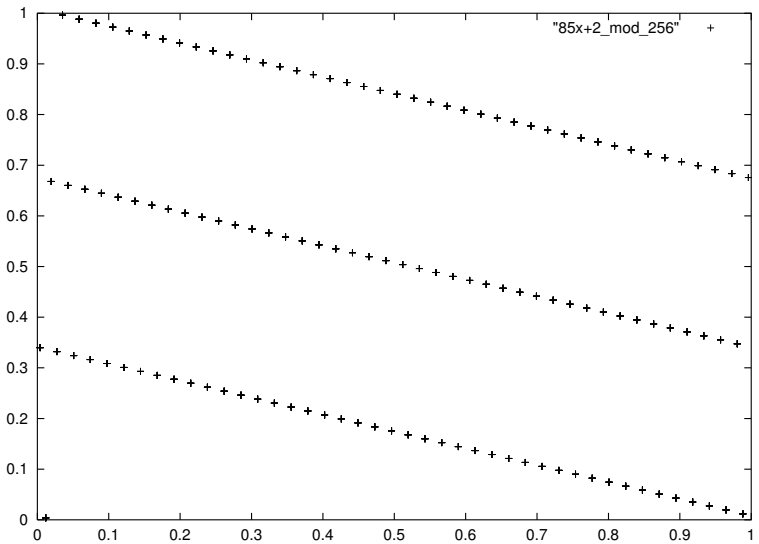
- gegeben: Pseudozufallszahlen $Y_1, Y_2, \ldots, Y_m$ aus $\{1, \ldots, n\}$
- $Z_1, Z_2, \ldots, Z_m$: die Zahlen aufsteigend sortiert
- berechne die Abstände $S_i = Z_{i+1} - Z_i$
- Sei $R = \left|\{S_i \mid \exists j < i : S_j = S_i\}\right|$ die Anzahl der mehrfach vorkommenden Abstände.
- R näherungsweise Poisson-verteilt mit Parameter $\lambda = m^3/(4n)$.
- χ^2-Test
- Lagged-Fibonacci-Generatoren (z. B. $x_n = x_{n-24} + x_{n-55} \pmod{2^e}$) haben üblicherweise Probleme mit dem Geburtstagsabstandstest.

Weitere Tests

- betrachte (besonders) schlechten PRNG $x_{n+1} = 85x_n + 2 \pmod{256}$ mit Ausgabefunktion $u_n = x_n/256$.
- zeichne überlappende Paare (u_i, u_{i+1}) im Einheitsquadrat:

Weitere Tests

- betrachte (besonders) schlechten PRNG $x_{n+1} = 85x_n + 2 \pmod{256}$ mit Ausgabefunktion $u_n = x_n/256$.
- zeichne überlappende Paare (u_i, u_{i+1}) im Einheitsquadrat:



- *Gitterstruktur*
- Tritt bei linearen Kongruenzgeneratoren aber auch bei einigen anderen Generatorarten auf.

Spektraltest für Dimension s

- s-dimensionale überlappende Vektoren $\mathbf{u}_i = (u_i, u_{i+1}, \ldots, u_{i+s-1})$ im Einheitshyperwürfel
- ermittle d_s = maximaler Abstand zwischen zwei Hyperebenen, genommen über alle Familien paralleler Hyperebenen im Einheitswürfel, die alle Punkte $\mathbf{u}_i$ beinhalten.
- je kleiner ein d_s ist, als desto „besser“ der Generator (hinsichtlich Dimension s)

Diskrepanz und Sterndiskrepanz

- N Vektoren $\mathbf{u}_i = (u_i, u_{i+1}, \ldots, u_{i+s-1})$ im Einheitshyperwürfel
- für Menge $R = \prod_{j=1}^{s} [\alpha_j, \beta_j[$ mit $0 \leq \alpha_j < \beta_j \leq 1$ sei
  - $I(R)$ die Anzahl der $\mathbf{u}_i$, die in R liegen, und
  - $V(R) = \prod_{j=1}^{s} (\beta_j - \alpha_j)$ das Volumen von R.
- s-dim. Diskrepanz $D_N^{(s)}$ der Punkte $\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}$ ist

$$D_N^{(s)} = \max_R |V(R) - I(R)/N| \ .$$

- Sterndiskrepanz $D_N^{*(s)}$: nur R mit einer Ecke im Nullpunkt
- Punkteverteilung „zu ungleich“: zu große Diskrepanz
- Punkteverteilung „zu gleichmäßig“: zu kleine Diskrepanz

Näheste-Paare-Test (nearest pair test)

- erzeuge zufällig n Punkte im s-dimensionalen Einheitshyperwürfel
- bestimme das Minimum D der (euklidischen) Abstände zwischen je zwei Punkten.
- bekannt: Für große n ist $T = n^2 D^s/2$ exponentiell verteilt mit Erwartungswert $1/V_s$,
  wobei V_s Volumen der s-dimensionalen Einheitskugel

Ränge zufälliger Boolescher Matrizen

- Eine echt zufällig mit Nullen und Einsen gefüllte $m \times n$-Matrix hat Rang r mit $1 \leq r \leq \min(m, n)$ mit Wahrscheinlichkeit

$$2^{-(n-r)(m-r)} \prod_{i=0}^{r-1} \frac{(1-2^{i-n})(1-2^{i-m})}{1-2^{i-r}} \,.$$

- insbesondere: nutze für die Bits jeweils einer Zeile der Matrix die Bits *einer* Pseudozufallszahl.

OPSO-Test (overlapping pairs sparse occupancy test)

- fasse je 10 Bits als ein Symbol aus einem 1024-elementigen Alphabet auf
- erzeuge $2^{21} + 1$ solche Symbole und prüfe alle 2^{21} überlappenden Paare auf fehlende Symbolpaare
- bekannt: Deren Zahl ist normalverteilt mit Erwartungswert 141 909 und Standardabweichung 290.

weitere Tests …

- Marsaglia schlägt auch noch andere Tests vor, bei denen allerdings nicht klar ist, welches die korrekten theoretischen Werte sind, mit denen die empirischen Ergebnisse verglichen werden müssen. Die Vergleichswerte bestimmt er daher ebenfalls mit der Hilfe von PRNGs.
- hmmm …

weitere Tests …

- Marsaglia schlägt auch noch andere Tests vor, bei denen allerdings nicht klar ist, welches die korrekten theoretischen Werte sind, mit denen die empirischen Ergebnisse verglichen werden müssen. Die Vergleichswerte bestimmt er daher ebenfalls mit der Hilfe von PRNGs.
- hmmm …
- L'Ecuyer: Das ist akzeptabel, solange viele vermutlich gute Generatoren zu übereinstimmenden Zahlen kommen, die als Ersatz für die fehlenden theoretischen Werte genommen werden.

OQSO Test (overlapping quadruples sparse occupancy)

- analog zum OPSO-Test,
  aber Quadrupel von Symbolen über einem 32-elementigem Alphabet.
- auch hier Erwartungswert nur experimentell bestimmbar.

Parking lot test

mein Lieblingstest: Hubschrauber parken

Parking lot test

mein Lieblingstest: Hubschrauber parken

- gegeben: z. B. ein Quadrat mit Seitenlänge 100
- darauf sollen Einheitskreise („Hubschrauber von oben gesehen“)
  an zufällig gewählten Stellen $\mathbf{u}_i$ positioniert („geparkt“) werden.
- wähle $\mathbf{u}_i$ als Mittelpunkt des Kreises, sofern der sich nicht mit schon festgelegten Kreisen schneidet.
- Man macht n Versuche und zählt, wie oft ein Kreis ohne Kollision positioniert werden konnte.

Auf Wiedersehen!