8 Zufallszahlenerzeugung (LK 7)

Zufallszahlen U([0;1]) sind die Basis um allgemeine Zufallsvariablen auswürfeln zu können.

8.1 Anforderungen

- U([0;1]) bedeutet gleichverteilt
- Voraussetzung: Unabhängikeit, also insbesondere auch Unkorreliertheit
- Schnelle Algorithmen mit wenig Speicherbedarf
- Reproduzierbarkeit des Zufallszahlenstroms
    - Debugging
    - Systemvergleiche
- Kontrollierbare Zerlegbarkeit des Zufallszahlenstroms in Unterbereiche
    - Systemvergleiche
        - Ein Teilstrom für gemeinsame externe Zufälle
        - Ein Teilstrom für Methoden-abhängige Zufälle

8.2 Verfahren

8.2.1 Physikalische Zufallszahlenerzeugung

- Ziehen aus einer Urne (Lottozahlen)
- Nutzung von radioaktiven Zerfallsprozessen
- Sperrrauschen von Dioden
- Tafelwerke mit Zufallszahlen

Nachteile offensichtlich.

8.2.2 Numerische Zufallszahlenerzeugung (Pseudozufallszahlen)

- Software: Numerische Verfahren (z.B. multiplikative oder gemischte Kongruenzmethoden)
- Hardware: rückgekoppelte Schieberegister mit primitivem Polynom als Rückkopplungsmuster

8.3 Numerische / Arithmetische Zufallszahlenerzeugung

8.3.1. Midsquare Method (von Neumann, Metropolis, 40er Jahre, erstes Verfahren)

$Z_0 = seed; \quad (4 - stellig)$

$Z_i = \left\lfloor \left(Z_{i-1}^2 / 10^2 \right) \right\rfloor \oplus 10^4;$

$U_i = Z_i / 10^4;$

Beispiel: 1009, 0180, 0324, 1049, 1004, 0080, 0064, 0040, 0016, 0002, 0000, … Problem offensichtlich

8.3.2 Linear Congruential Generators (LCGs, Lehmer 1951)

$$Z_0 = \text{seed};$$
$$Z_i = (a \cdot Z_{i-1} + c) \oplus m;$$
$$U_i = Z_i / m;$$
$$a, c, m \in N_0;$$
$$m > \max(a, c, Z_0, 0);$$

Man kann zeigen:

$$Z_i = \left(a^i \cdot Z_0 + \frac{c \cdot (a^i - 1)}{a - 1} \right) \oplus m;$$

- m muss groß gewählt werden, sonst sind die Zufallszahlen zu grob verteilt.
- Höchstens m verschiedene Zufallszahlen Z_i können erzeugt werden.
- Wegen der deterministischen Vorschrift ist die Folge periodisch.
- Falls die Folge eine maximale Periode m hat, erzeugt jede Seed dieselbe Folge.

Theorem

Der LCG hat genau dann maximale Periode, wenn folgende Bedingungen gelten:

1. GCD(m,c)=1 (ggT, „c ist relativ prim zu m“)
2. Alle Primfaktoren von m teilen a-1
3. Wenn m durch 4 teilbar ist, ist auch a-1 durch 4 teilbar

Falls c>0 heißt ein LCG „mixed Generator“, für c=0 heißt er „multiplicative“.

8.3.2.1 Mixed Generators (Gemischte Kongruenzmethode)

- Wahl der Parameter:
  - $m=2^{31}$ für signed integer bzw. $m=2^{32}$ für unsigned integer erlaubt Division durch integer overflow.
  - c muss ungerade sein.
  - a-1 muss durch 4 teilbar sein.
    - Spezialfall $a=2^d+1$ erlaubt effiziente „shift-and-add“ Implementierung (d ganze Zahl)
    - Wurde früher benutzt, hat aber schlechte statistische Eigenschaften.
- Beispiel
  - $z_i = 314159269 \cdot z_{i-1} + 453806245 \pmod{2^{31}}$ (Kobayashi, 1978)

8.3.2.2 Multiplicative Generators (Multiplikative Kongruenzmethode)

- Die einfacheren Spezialfälle des LCG (Multiplicative Generators) sind besser verstanden, mindestens genauso gut und werden häufiger angewandt.
- Problem: keine volle Periode möglich, da seed=0 im Null-Zyklus landet
  - Aber Periode m-1 möglich, falls a sorgfältig gewählt
- Problem: für $m=2^b$ ist Periode maximal 2^{b-2}
  - Allgemein: unabhängige Teilströme können üble statistische Eigenschaften aufweisen (e.g. für $a=2^d+j$)
  - „RANDU“: $m=2^{31}$, $a=2^{16}+3$, c=0 ist besonders schlecht
- Besser: Wähle m als große Primzahl, z.B. $2^{31}-1=2147483647$

- Periode ist m-1, falls a primitives Element Modulo m ist (kleinste Zahl, für die a^l-1 durch m teilbar ist, ist l=m-1)
  =„prime modulus multiplicative LCG“ (PMMLCG)
- Problem: Primitive Elemente Modulo m sind schwierig zu berechnen
- Beispiele:
  - Gute Werte für a (primitive Elemente modulo 2^{31}-1): 7^5=16807
    $z_i = 16807 \cdot z_{i-1} \pmod{2^{31}-1}$ (Lewis, Goodman, Miller, 1969)
  - Noch bessere statische Ergebnisse für 630360016
    $z_i = 630360016 \cdot z_{i-1} \pmod{2^{31}-1}$ (Payne, Rabung, Bogyo, 1969)

- Problem: Keine Schnelle Division durch Overflow-Mechanismus möglich
  Lösung: Simulierte Division für Teiler 2^b-q, damit
  $Z_i = (a \cdot Z_{i-1}) \oplus (2^b - q)$ schnell berechnet werden kann:

  $Z_i' = (a \cdot Z_{i-1}) \oplus 2^b$ (*use overflow without division*)
  $k = \lfloor (a \cdot Z_{i-1}) / 2^b \rfloor$
  $$Z_i = \begin{cases} Z_i' + k \cdot q & \text{if} \quad Z_i' + k \cdot q < 2^b - q \\ Z_i' + k \cdot q - (2^b - q) & \text{if} \quad Z_i' + k \cdot q \geq 2^b - q \end{cases}$$

Vorteile

- Reproduzierbarkeit, da Saat z_0 komplette Sequenz festlegt
- Zahlen einfach und schnell zu erzeugen

Nachteile

- Periode von a, c und m abhängig, maximal m bzw. m-1.
- Verteilungs- und Korrelationseigenschaften nicht offensichtlich ⇒ Tests notwendig

8.3.3 Andere Generatorarten

Generellere Kongruenzen
Allgemein: $Z_i = g(Z_{i-1}, Z_{i-2}, \ldots) \oplus m$ und $U_i = Z_i / m$.

- Quadratic congruential generators: $Z_i = \left(a_2 \cdot Z_{i-1}^2 + a_1 \cdot Z_{i-1} + a_0 \right) \oplus m$

  Maximale Periode auch m.

- Multiple recursive generators (MRGs): $Z_i = \left(\sum_{0<j\leq q} a_j \cdot Z_{i-j} \right) \oplus m$

  Perioden der Größe m^q-1 bei geeigneter Parameterwahl möglich.

  - Fibonacci-Generator: $Z_i = \left(Z_{i-1} + Z_{i-2} \right) \oplus m$

    Periode größer als m aber schlechte statistische Eigenschaften

Composite Generators
Beispiele

- Ein Vektor mit k Einträgen wird mit Zufallszahlen von Generator 1 besetzt, Generator 2 würfelt Zahlen zwischen 0 und k-1 aus. Die entsprechende Position im Vektor ist die gewünschte Ergebniszufallszahl und der Vektoreintrag wird durch die nächste Zufallszahl von Generator 1 wieder besetzt.

- Kann aus zwei schlechten Generatoren einen guten machen
- k darf klein sein (k=2 genügt oft)
- Aufwändiges Verfahren
- Gezielter Sprung an eine bestimmte Stelle des Zahlenstromes unmöglich

- Zwei durch unterschiedliche Moduli erzeugte Zufallszahlenströme $Z_{1,i}$ und $Z_{2,i}$. Sei m ein neuer Modulus. Die neue Zufallszahl ist dann $Z_i = \left(Z_{1,i} + Z_{2,i} \right) \oplus m$.
  - Schnell und gute statistische Ergebnisse
- Kombination mehrerer MRGs
  - n simultane MRGs erzeugen Zufallszahlenströme $Z_{j,i}$ ($0<j\leq n$).
  - m ist der Modulo des ersten MRG
  - $Z_i = \left(\sum_{0<j\leq n} \delta j \cdot Z_{j,i} \right) \oplus m$
  - Beispiel für n=2 (L'Ecuyer 1999):
    - $Z_{1,i} = \left(1403580 \cdot Z_{1,i-2} - 810728 \cdot Z_{1,i-3} \right) \oplus \left(2^{32} - 209 \right)$
    - $Z_{2,i} = \left(527612 \cdot Z_{2,i-1} - 1370589 \cdot Z_{2,i-3} \right) \oplus \left(2^{32} - 22853 \right)$
    - $m=2^{32}-209$
    - $Z_i = \left(Z_{1,i} - Z_{2,i} \right) \oplus m$
    - $seed = \left(Z_{1,0}, Z_{1,2}, Z_{1,3}, Z_{2,1}, Z_{2,2}, Z_{2,3} \right)$
    - Periodenlänge: ca $2^{191} \approx 3.1*10^{57}$
    - Exzellente statistische Eigenschaften
    - Große Anzahl von weit entfernten Zufallszahlenströmen
- Drei zufallszahlenströme $U_{j,i}$ von separaten Generatoren
  - $U'_i = \left(U_{1,i} + U_{2,i} + U_{3,i} \right)$
    $U_i = U'_i - \lfloor U'_i \rfloor$
  - Sehr lange Periode
  - Äquivalent zu LCG mit sehr großem m und a
  - Trotz großer Periode auch auf Maschinen mit kleinen Wortlängen einfach implementierbar

<u>Tausworthe and Related Generators (LK 7.3.3)</u>

- Tausworthe Generators
  - Basiert direkt auf Bitfolgen b_i, die Grundlage für Zufallszahlen bilden
  - $b_i = \left(\sum_{0<j\leq q} c_j \cdot b_{i-j} \right) \oplus 2$
  - In Schieberegistern (feedback shift register) implementierbar
  - Bei allgemeiner Software-Lösung enorm große Perioden implementierbar (bei nur zwei $c_j \neq 0$)
  - Möglichkeiten zur Zufallszahlengenerierung
    - Interpretation aufeinanderfolgender Bitfolgen als Zahlen
    - Auslassen von Bits
  - Nicht allzu sehr verbreitet

- Methode aus der Kryptographie zur Erzeugung von Bits
  - p, q große Primzahlen und p-3, q-3 durch 4 teilbar
  - $X_i = X_{i-1}^2 \oplus (p \cdot q)$
  - b_i ist das Bit mit der niedrigsten Signifikanz von X_i
  - Bitfolge sehr schwer vorhersagbar, äquivalent zur „Faktorisierung von m in p*q“ (= sehr rechenintensives Problem)

8.4 Testen von Zufallszahlengeneratoren

Eine Stichprobe wird durch einen Generator erzeugt und dann auf ihre Güte bezüglich Gleichverteilung und Unabhängigkeit getestet.

8.4.1 Empirische Tests

- χ^2-Test (siehe 4.3 in diesem Skript)
- Berechnung der empirischen Autokorrelation (siehe 2.4 und 2.5 in diesem Skript)

8.4.2 Theoretische Tests

- Theoretische globale Generatortests: Spektraltest, Lattice-Test (Fishman (1995))

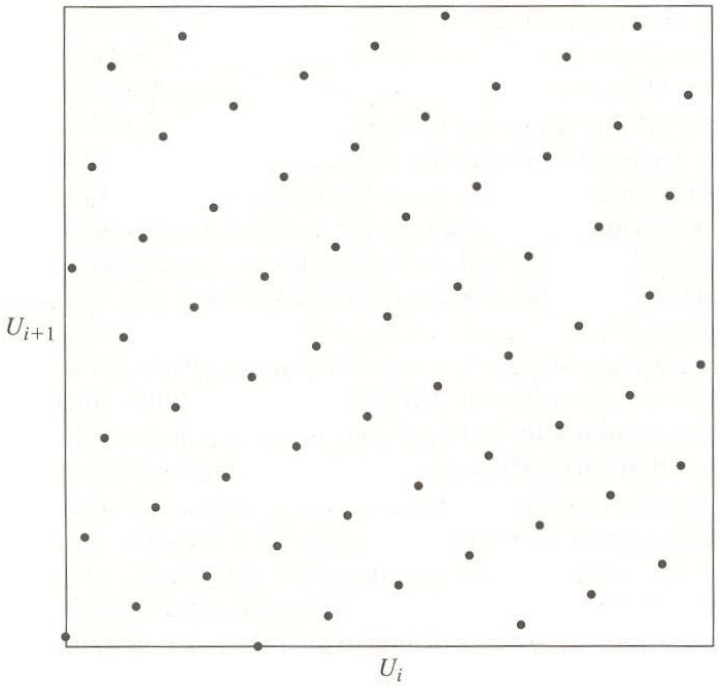
  Grundidee: Test der k-dimensionalen Uniformität (d.h. k-Tupel ist $[0,1]^k$-gleichverteilt)

  Gegeben: Generator für z_i, Bildung von Paaren (z_i, z_{i+1}) bzw. Tripeln (z_i, z_{i+1}, z_{i+2}), Verwendung als Koordinaten von Punkten im Einheitsquadrat bzw. Einheitswürfel.

  Überprüfen: gleichmäßige Verteilung der Punkte ohne auffällige Muster (nichtgefüllte Flächen bzw. Volumen deuten auf Mängel bei der Verteilung hin, Streifen oder Spuren deuten auf korrelierte Werte hin)

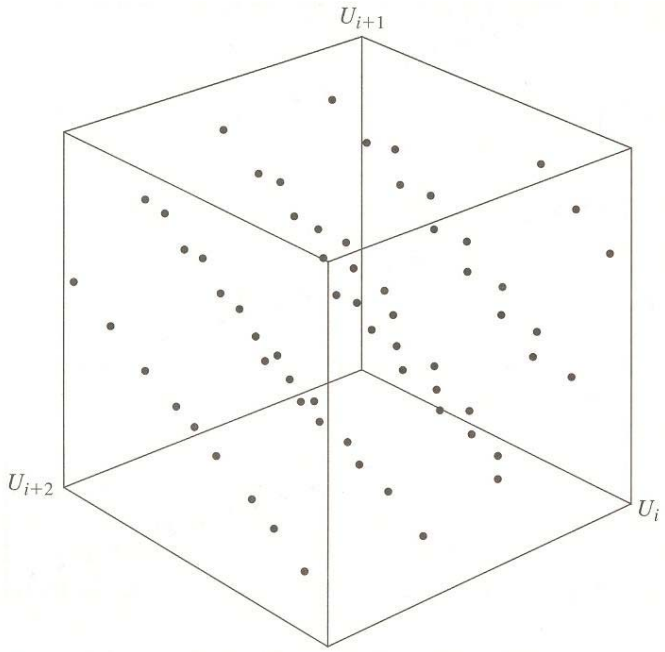
  Umsetzung der Idee: Messung des Abstandes von nebeneinanderliegenden Hyperebenen auf denen die k-Tupel liegen durch Lösung eines Minimierungsproblems (sehr aufwendig für $k > 10$)

Bemerkung: am besten nur getestete Generatoren und getestete Saaten verwenden!



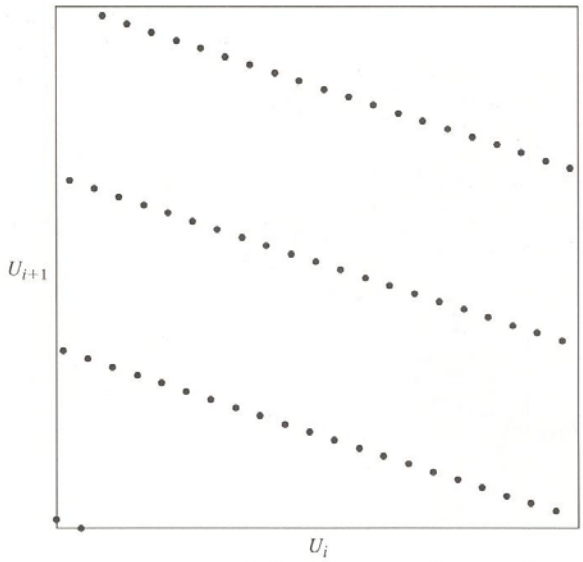
Two-dimensional lattice structure for the full-period LCG with $m = 64$, $a = 37$, and $c = 1$.

Abbildung 8.1: Spektraltest für Zufallszahlen des Generators $z_i = (37 \cdot z_{i-1} + 1) \pmod{64}$



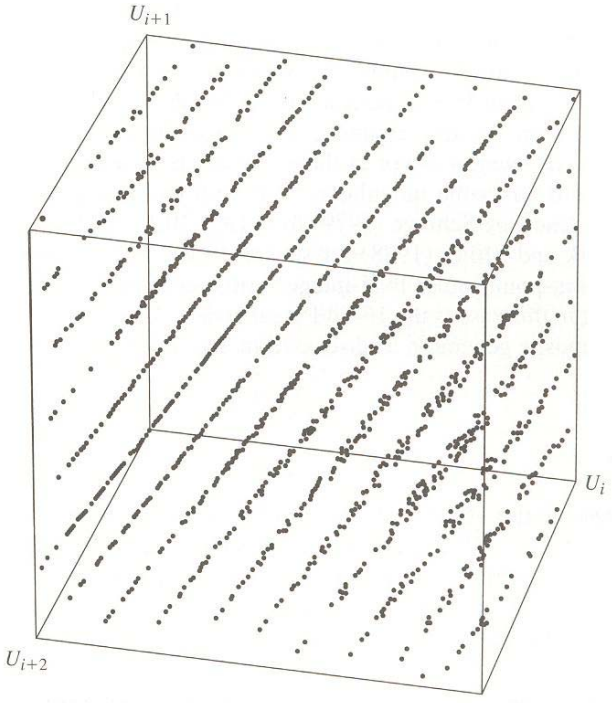
Three-dimensional lattice structure for the full-period LCG with $m = 64$, $a = 37$, and $c = 1$.

Abbildung 8.2: Spektraltest für Zufallszahlen des Generators $z_i = (37 \cdot z_{i-1} + 1) \pmod{64}$



Two-dimensional lattice structure for the full-period LCG with $m = 64$, $a = 21$, and $c = 1$.

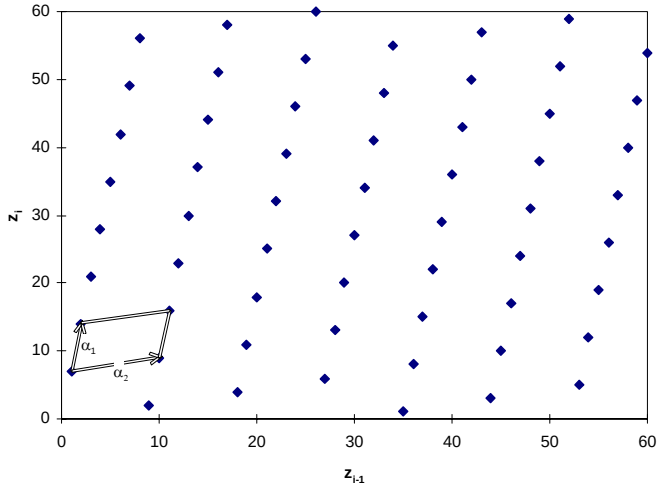
Abbildung 8.3: Spektraltest für Zufallszahlen des Generators $z_i = (21 \cdot z_{i-1} + 1) \pmod{64}$



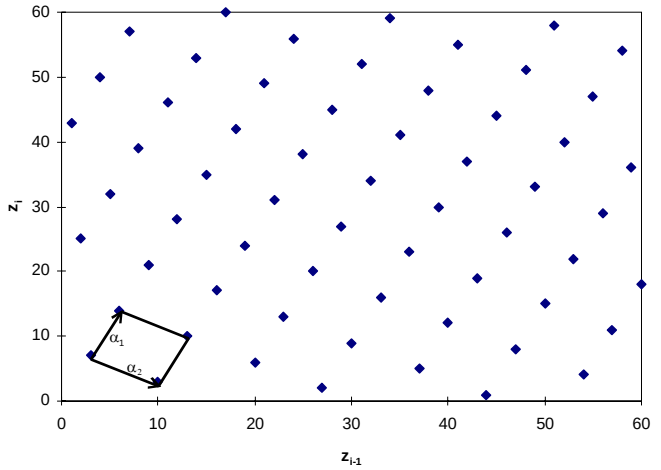
Three-dimensional lattice structure for 2000 triples from the LCG RANDU with $m = 2^{31}$, $a = 2^{16} + 3 = 65{,}539$, and $c = 0$.

Abbildung 8.4: Spektraltest für Zufallszahlen des Generators „RANDU“

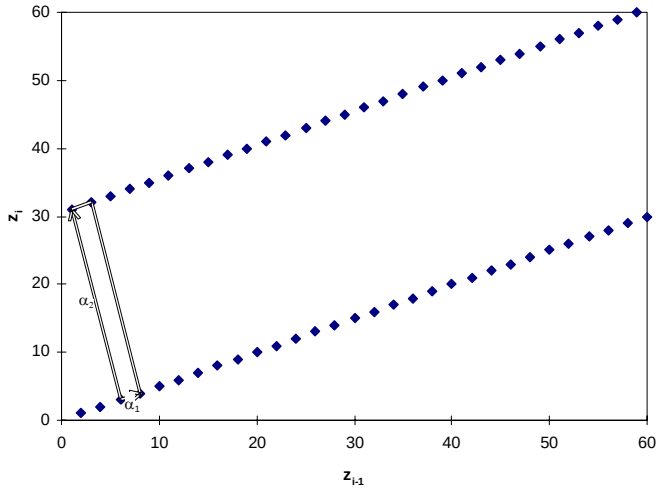
$z_i = \left(\left(2^{16} + 3\right) \cdot z_{i-1}\right) \pmod{2^{31}}$



a=7



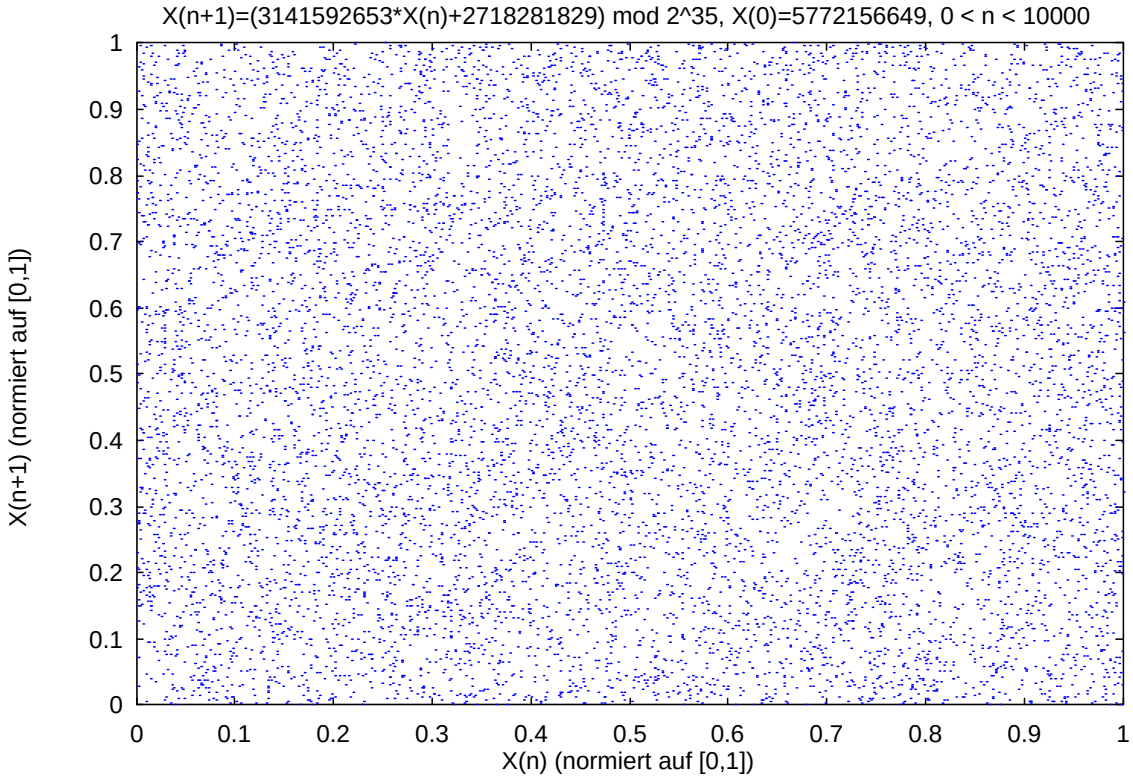
a=43

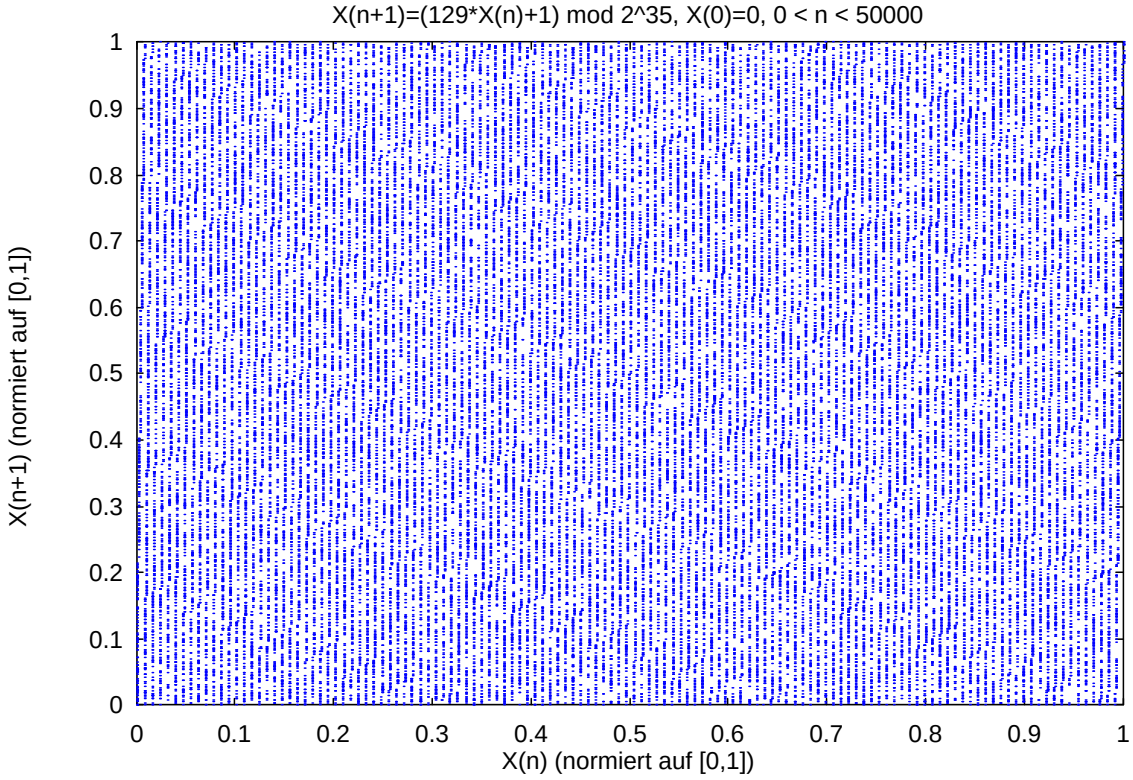


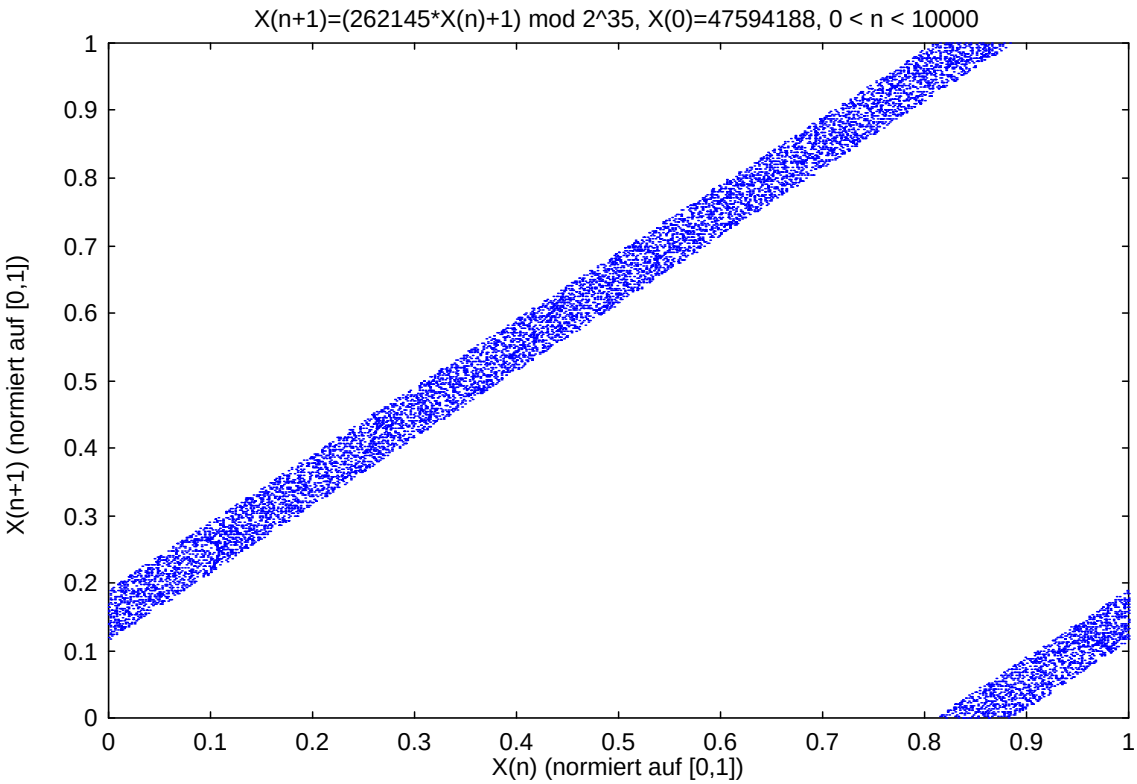
a=31

Abbildung 2: Beispiele für Zufallszahlengeneratoren nach der multiplikativen Kongruenzmethode mit $z_i = a \cdot z_{i-1} \pmod{61}$

Weitere Beispiele






X(n+1)=(262145*X(n)+1) mod 2^35, X(0)=47594188, 0 < n < 10000
X(n+1) (normiert auf [0,1])
X(n) (normiert auf [0,1])
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1