Martin–Luther–Universität
Halle–Wittenberg
Fachbereich Mathematik und Informatik

Computer oder echter Zufall?
Pseudozufallszahlen im Mathematikunterricht

Elvira Malitte

Report No. 12 (2000)

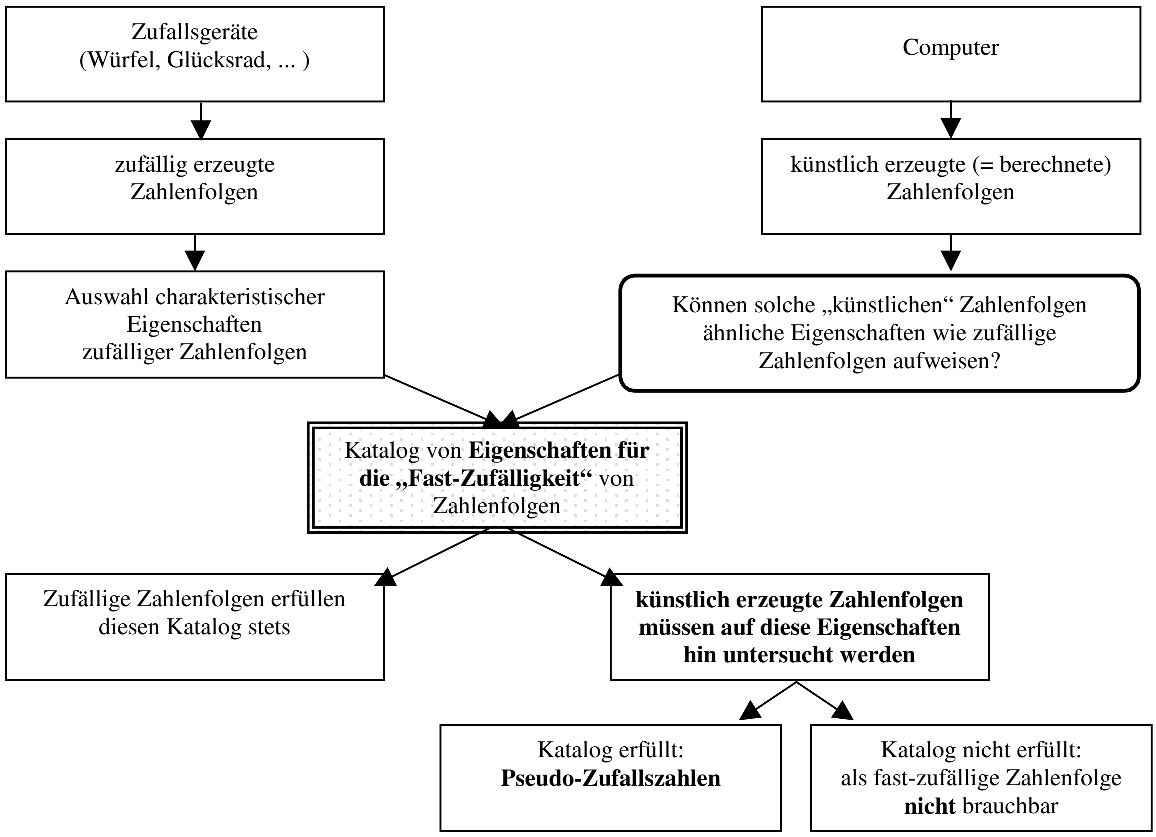
Reports on Didactics and
History of Mathematics

Computer oder echter Zufall? Pseudozufallszahlen im Mathematikunterricht

Elvira Malitte

Ziehen der Lottozahlen, Drehen eines Glücksrades, Werfen eines Würfels – das liefert *„Zufallszahlen"*. Durch Taschenrechner oder PC dagegen werden Zahlen berechnet. Inwieweit sind die Ergebnisse dann noch *„zufällig"*? Wie können Algorithmen aufgebaut sein, die Zahlenfolgen erzeugen, die *„wenigstens einigermaßen zufällig"* aussehen? Wie kann man die *„Zufälligkeit"* einer vorgelegten Zahlenfolge *„messen"*?
Dazu werden zunächst unterschiedliche Modelle zum Erzeugen von *„Zufall"* diskutiert. Im weiteren werden Verfahren zum Beurteilen der Güte von *„Zufall"* vorgestellt, die sich nicht nur auf quantitative sondern vor allem auf qualitative Bewertungsverfahren stützen.

Um Zufallszahlen für Simulationen zu gewinnen, bewährt es sich nur in sehr begrenztem Maße, zu würfeln oder Roulette zu spielen. Zu groß ist der Bedarf an Zahlen, zu lange dauert das Würfeln. Moderne Supercomputer können heute in einer einzigen Sekunde mehr als eine Million Zufallszahlen verschlingen – sie mit Ergebnissen von realen Zufallsexperimenten füttern zu wollen, ist daher aussichtslos. Einziger Ausweg ist, von den Computern selbst Folgen von Zahlen erzeugen zu lassen, die wenigstens „zufällig aussehen" (vgl. Malitte [6]).



Zufallszahlenfolgen erzeugen mit einem ***Computer*** – das kann doch nur ein Irrtum sein! Denn Computer sind Maschinen, arbeiten nach vorgegebenen Rechenvorschriften oder Regeln; mit ihrer Hilfe werden somit Zahlen ***berechnet*** – von Zufall also keine Spur. Wie können überhaupt geeignete Vorschriften zur Erzeugung von „**Fast-Zufallszahlen**“ aussehen?

Dazu stellen wir einige Modelle zum Erzeugen von „Zufall“ vor. Daraus ergeben sich dann unmittelbar zwei grundsätzliche Fragen:
Um eine mathematisch zufriedenstellende Lösung zu erhalten, müssen wir einerseits festlegen, was wir unter „fast zufällig“ verstehen, und zum anderen, wie wir erzeugte Zahlenfolgen auf (wenigstens) Fast-Zufälligkeit überprüfen.

Modelle zum Erzeugen von „Zufall“

Zufallszahlen-Generatoren werden oft als „Herstellungsgeheimnis“ durch den Produzenten betrachtet.
Eine sehr alte und traditionelle Methode ist die Verwendung von Ziffernfolgen transzendenter Zahlen, z. B. wurden π oder e verwendet. Über die Verteilung der einzelnen Ziffern dieser Zahlen ist trotz Berechnungen auf über vier Milliarden Stellen keine allgemeine Aussage möglich. Somit ist ihre Nutzung als Zufalls-Ziffernfolge eigentlich nicht gerechtfertigt, weil sie nicht begründbar ist.
Den ersten rekursiven Zufallsgenerator entwickelten J. v. Neumann, N. Metropolis und S. Ulam (etwa 1946). Das damals verwendete Verfahren, die mittleren Ziffern von Quadraten herzunehmen, erwies sich jedoch als problematisch (vgl. Gundel, Schupp, Schweizer [4]).
Heute werden häufig lineare Kongruenzgeneratoren verwendet, die die folgende allgemeine Struktur, vorgeschlagen von D. H. Lehmer (1949), besitzen:

1. Vorgabe eines Startwertes x_0 und ganzer Zahlen a, b, m mit $0 \le x_0 < m$, $0 \le a < m$ und $0 \le b < m$,
2. Erzeugen einer Zahlenfolge (x_i) nach der Vorschrift
   x_{i+1} := Rest von $(a\, x_i + b)$ bei Division durch m (i = 0, 1, 2, ...).

Häufig erwähnt wird der **147-Kongruenz-Generator**:

Man wählt für die Parameter die Werte a = 147 , b = 0 und $m = 10^k$ und erhält so k-stellige natürliche Zahlen.
Für k = 5 und bei Verwendung der Startzahl 54327 erhält man die nachfolgende Menge für die ersten 16 Werte der zugehörigen Zahlenfolge:

```
{54327, 86069, 52143, 65021, 58087, 38789, 1983, 91501,
 50647, 45109, 31023, 60381, 76007, 73029, 35263, 83661}
```

Die Berechnung wurde hier dem Computeralgebrasystem *MATHEMATICA* übertragen. Durch die folgenden Anweisungen erhält man die obige Ausgabe:

```
zufall[n_] :=
 Module[{}, pseudo = {}; pseudo = Append[pseudo, x0];
  Do[x0 = Mod[a x0 + b, m];
   pseudo = Append[pseudo, x0], {i, 1, n}]]
a = 147; b = 0; m = 10^5; x0 = 54327; zufall[15]; pseudo
```

Abb. 1

Bei anderer Wahl der Parameter wird schnell eine besondere Problematik deutlich, die mit Kongruenzverfahren grundsätzlich und unvermeidbar verbunden ist:
Mathematica liefert bei den Vorgaben

```
a = 1; b = 3; m = 10; x0 = 6; zufall[22]; pseudo
a = 7; b = 7; m = 10; x0 = 7; zufall[22]; pseudo
a = 6; b = 18; m = 40; x0 = 5; zufall[17]; pseudo
```

jeweils die folgenden Werte:

```
{6, 9, 2, 5, 8, 1, 4, 7, 0, 3, 6, 9, 2, 5, 8, 1, 4, 7, 0, 3, 6, 9}
{7, 6, 9, 0, 7, 6, 9, 0, 7, 6, 9, 0, 7, 6, 9, 0, 7, 6, 9, 0, 7, 6}
{5, 8, 26, 14, 22, 30, 38, 6, 14, 22, 30, 38, 6, 14, 22, 30, 38}
```

Abb. 2

Die Zahlenfolgen zeigen schließlich ein zyklisches Verhalten. Diese Eigenschaft lässt sich bei der Anwendung von Kongruenzverfahren grundsätzlich nicht vermeiden:
Die erzeugte Zahlenfolge kann nach Konstruktion nur ganze Zahlen aus dem Bereich [0,m) enthalten. Da dieser Zahlenbereich endlich ist, muss nach spätestens m Schritten der Iteration in der bisher erzeugten Zahlenfolge mindestens eine der berechneten Zahlen wenigstens zweimal auftreten. Mit dem zweiten Auftreten einer der Zahlen beginnt auf Grund des Generators, bei dem jede Zahl allein durch ihren Vorgänger bestimmt ist, ein neuer Zyklus. Somit ist m die maximale Zykluslänge.

Durch geeignete Wahl der Parameter versucht man nun, möglichst große Zyklenlängen zu erreichen.

Es lässt sich zeigen (siehe Greene, Knuth [2]), dass die drei folgenden Bedingungen an die Parameter a, b und m des Kongruenzverfahrens zusammen hinreichend sind für maximale Zykluslänge (= m):
1. m und b sind teilerfremd,
2. wenn 4 ein Teiler von m ist, dann ist 4 auch Teiler von a-1,
3. jeder Primfaktor von m ist auch Teiler von a-1.

Die genannten Bedingungen sichern mithin, dass dann jeder Zyklus der erzeugten Zahlenfolge jede der Zahlen 0, 1, ..., m-1 enthält.

Man überzeugt sich leicht, dass bei einer Parameterwahl in der Weise

a	kongruent 1 mod 4
b	ungerade
m	Zweierpotenz

die Eigenschaften 1. bis 3. erfüllt sind.

Für die Besprechung im Unterricht dürfte es ausreichen, insbesondere den für praktische Belange oft herangezogenen DERIVE-Generator mit den Parametern

$a = 3\,141\,592\,653$, $b = 1$ und $m = 2^{32}$ (= 4 294 967 296)

zu betrachten.

Um einen Eindruck von der Fülle der Zahlen zu bekommen, die mit Hilfe des DERIVE-Generators erzeugt werden können, bietet es sich an, Schülerinnen und Schüler durch die folgenden Aufträge selbst erste Erfahrungen im Umgang damit sammeln zu lassen.

1. Erzeuge im Rahmen eines PC-Programmes verschiedene Zahlenfolgen mit dem DERIVE-Generator. Schreibe auf, was dir dabei auffällt.

Eine Möglichkeit ist die Zahlenwerte zu betrachten, die sich für unterschiedliche Startwerte und verschiedene Anzahlen der berechneten Folgenglieder ergeben. MATHEMATICA *liefert bei Verwendung des Moduls in Abb. 1 und mit*

```
a = 3141592653; b = 1; m = 2^32; x0 = 12345; zufall[31]; pseudo
```

die Zahlenwerte

```
{12345, 3701585702, 850410095, 1145130340, 2553028373, 2238854994,
 2072924075, 3642857584, 1779399089, 2235342910, 3836995239,
 303458876, 4147108877, 3680034282, 966458211, 3141045960,
 3930759721, 56859734, 1793804767, 3924005396, 4021348357,
 2968143746, 2302037531, 1329709088, 1365477793, 3956738926,
 1500726295, 1234707692, 1928274685, 4187804698, 4282066899}
```

Diese wenigen berechneten Werte zeigen bereits, wie die erzeugten Zahlen im Bereich $[0,2^{32})$ *„streuen".*

2. Überlege dir eine experimentelle Möglichkeit zur näherungsweisen Bestimmung der Zeit, die dein PC benötigen würde, bis ein vollständiger Zyklus mit dem DERIVE-Generator durchlaufen ist.

Z. B. iterative Berechnung von 1000 Zahlen mit Hilfe eines Programmes – man stellt fest, dass (in Abhängigkeit von der verwendeten Hard- und Software!) ca. 1 sec dafür benötigt wird. Auf einem Pentium mit 230 MHz liefert MATHEMATICA *für*

```
a = 3141592653; b = 1; m = 2^32; x0 = 12345; Timing[zufall[1000]]
```

als Ergebnis

```
{0.87 Second, Null}
```

Für die 2^{32} *ganzen Zahlen des Bereichs* $[0, 2^{32})$ *benötigt dieser PC mithin etwa* $0.87 \cdot 1000^{-1} \cdot 2^{32}$ *sec, d.h. rund 1000 h.*

Zahlreiche Vorschläge für veränderte lineare Kongruenz-Generatoren bzw. für **nichtlineare Kongruenz-Generatoren** sind in Knuth [5] bzw. Mertsch [7] enthalten.

Durch die Wahl der Parameter lassen sich die Eigenschaften der jeweils erzeugten Zahlenfolge beeinflussen. Es bestehen somit prinzipielle Unterschiede zwischen mechanisch (d. h. vermittels eines realen zufälligen Versuchs) erzeugten Zufallszahlenfolgen und algorithmisch berechneten (in unserem Fall: durch Kongruenzverfahren erzeugten) Zahlenfolgen. Wenn man dennoch auf algorithmisch erzeugte Zahlenfolgen angewiesen ist, wird man sich für diejenigen entscheiden, die gewisse *ähnliche Eigenschaften* wie *Zufalls*zahlenfolgen aufweisen.
Dieser Gedanke nimmt in der Begriffsbildung der Pseudo-Zufallszahlenfolge Gestalt an (siehe M. Fisz [1]):

Folgen ganzer Zahlen aus dem Bereich [0,m) werden ***Pseudo-Zufallszahlenfolgen*** genannt, wenn folgende Eigenschaften erfüllt sind:

1. In jedem hinreichend langen Abschnitt der Zahlenfolge treten alle Zahlen aus dem Bereich [0, m) mit jeweils annähernd der gleichen relativen Häufigkeit auf.
2. Analoges gilt für die Zahlenpaare, -tripel usw. einer Pseudo-Zufallszahlenfolge.
3. Die aufeinanderfolgenden Zahlen einer Pseudo-Zufallszahlenfolge sollen fast-unabhängig voneinander sein.

Die oben angeführten Eigenschaften 1. bis 3. sind wichtige Merkmale für echt zufällig erzeugte (d.h. durch ein Zufallsexperiment erzeugte) Zahlenfolgen. Diese Eigenschaften werden somit als Charakteristika für Fast-Zufälligkeit genutzt. Ziel ist es, solche Pseudo-Zufallszahlen zu generieren, die sich in ihren Eigenschaften sehr nah den oben angeführten Merkmalen nähern. Je höher der Grad der Annäherung an diese Eigenschaften ist, umso größer ist die Erfüllung des Anspruches an „Zufälligkeit", also die Güte der Pseudo-Zufallszahlen.

Verfahren zum Beurteilen der Güte von „Zufall"

Nun bleibt „nur noch" die Aufgabe, für konkrete Zahlenfolgen (gleich welcher Herkunft) zu untersuchen, inwieweit sie den genannten drei Bedingungen genügen und wir sie demgemäß als Pseudo-Zufallszahlenfolge akzeptieren können oder wollen.

Ausgehend vom empirischen Gesetz der großen Zahlen werden zur Überprüfung der Güte einer gegebenen Zahlenfolge im ersten Schritt die relativen Häufigkeiten ihrer Zahlen $x \in [0, m)$ mit der Wahrscheinlichkeit $\frac{1}{m}$ für den Idealfall der Gleichverteilung auf der Menge {0, 1, ..., m-1} verglichen.

Dazu könnten zunächst folgende Schüleraufträge vergeben werden:

3. Erzeuge „Zufallszahlen" aus der Menge {0, 1, ..., 9}
a) mit dem 147-Kongruenz-Generator,
b) mit dem eingebauten Zufallszahlen-Generator des Taschenrechners oder einer Computer-Software.
Welche Abweichungen zur Forderung 1 für Pseudo-Zufallszahlen stellst du fest?

a) *Vorgabe einer Startzahl;*
Berechnung x_{k+1} := Rest der Zahl $(147 \cdot x_k)$ bei Division durch 10;
Verwendung unterschiedlicher Werte für n, z.B. n = 500, 1000, 5000;
Berechnen der Abweichungen $\left|h_n(x) - \frac{1}{10}\right|$ *für x = 0, 1, 2, ..., 9;*

b) *Erzeugen einer Zahlenfolge unter Verwendung des eingebauten Zufallszahlen_Generators (Random);*
n = 500, 1000, 5000;
Berechnen der Abweichungen $\left|h_n(x) - \frac{1}{10}\right|$ *für x = 0, 1, 2, ..., 9;*

4. Die zufällige Bewegung eines Teilchens auf einem Gitter soll im Rahmen eines Computerprogrammes simuliert werden. Um gleichverteiltes „Hüpfen“ in die vier Himmelsrichtungen zu erzeugen, benötigt man entsprechend „ausgewürfelte“ natürliche Zahlen.
Erzeuge dafür 10000 „Zufallszahlen“ und überprüfe ihre Eignung durch Ermittlung
a) der absoluten Häufigkeiten,
b) der absoluten Abweichungen vom theoretischen Mittelwert,
c) der relativen Schwankungen um die theoretische Wahrscheinlichkeit.

Die Erzeugung von 10000 „Zufallszahlen“ mit vier verschiedenen Werten und die Auswertung entsprechend obiger Aufgabenstellung leisten die folgenden MATHEMATICA-Anweisungen:

```
<< Statistics`DataManipulation`; anz = 10000; zli = Table[Random[Integer, {1, 4}], {i, 1, anz}];
fzl = Frequencies[zli]
abw = Table[(fzl[[i, 1]] - anz / 4), {i, 1, 4}]
abw / anz / 4 // N
```

Sie liefern folgende Ergebnisse

für a): `{{2556, 1}, {2503, 2}, {2426, 3}, {2515, 4}}`

für b): `{56, 3, -74, 15}`

für c): `{0.0014`, 0.000075`, -0.00185`, 0.000375`}`

Dabei wurde der Zufallszahlen-Generator Random von MATHEMATICA genutzt, der in diesem Fall Schwankungsbreiten um die theoretischen Werte liefert, die kleiner als 0,2% (bei 10000 Zahlen) sind.

Wie sind die Abweichungen zwischen den relativen Häufigkeiten und den Wahrscheinlichkeiten zu bewerten? Insbesondere: Können wir die Forderung 1 an Pseudo-Zufallszahlen als erfüllt betrachten oder sind diese Abweichungen zu deutlich? Welche (Maximalwerte für die) Abweichungen können als „hinreichend klein“ angesehen werden?

♦ Quantitative Bewertungsverfahren

Der Übergang zu dem üblicherweise herangezogenen χ^2-Test (vgl. Gundel, Schupp, Schweizer [4]) (oder eines der anderen gebräuchlichen Testverfahren wie etwa Poker- oder Sammlertest) sollte wohl überlegt und vorbereitet sein. Stehen den Schülerinnen und Schülern Kenntnisse über Zufallsgrößen, Erwartungswertberechnung und Testverfahren auf Gleichverteilung zur Verfügung, so bietet sich die Entwicklung des χ^2-Tests für das Zahlenfolgen-Prüfproblem an. Aber auch ein nur heuristisches Nachvollziehen dieser Testgedanken scheint möglich und überlegenswert. Ein guter Kompromiss ist die Darstellung in Strick [8].

♦ Qualitative Bewertungsverfahren

Neben und ergänzend oder auch einführend zu diesen quantitativen Häufigkeitsuntersuchungen für Zahlenfolgen existiert ein weiteres recht wirksames Mittel, die Zufälligkeit einer vorgelegten Zahlenfolge durch **Veranschaulichung** qualitativ zu bewerten – dies ist auch für Schülerinnen und Schüler jüngeren Alters zugänglich.

Es bieten sich dazu verschiedene Veranschaulichungsmöglichkeiten (**ein-, zwei- und dreidimensional**) an, bei denen aus den Zahlenfolgen alle einzelnen Werte oder Zusammenfassungen zu Paaren bzw. Tripeln dargestellt werden.

Die bekannteste Möglichkeit ist sicher die Darstellung der Werte als Punkte in einem **Rechteck**, bei dem die Seitenlängen festgelegt sind durch die Anzahl der erzeugten Zahlen sowie durch deren Maximum.

Wenn einige Programmierkenntnisse der Schülerinnen und Schüler vorhanden sind, können dazu die folgenden Aufträge vergeben werden.

5. Erzeuge Zahlenfolgen der Länge n unter Verwendung des linearen Kongruenzverfahrens mit den Parametern

I	**a = 211, b = 1663, m = 7875,**
II	**a = 106, b = 1283, m = 6075,**
III	**des DERIVE-Generators**

und der Startzahl x_0 = 1243. Wähle verschiedene Werte für n.
Stelle (k, x_k) für k = 0, ..., n als Punkte in einem Koordinatensystem dar.

„Rechteck-Darstellung“
Es werden jeweils die ersten 1000 Werte, die die Varianten I bis III liefern, in Abb. 3_I – 3_{III} dargestellt. Mit MATHEMATICA *erhält man diese Abbildungen unter Verwendung des Moduls* **zufall** *(Abb. 1) und Anweisungen, die analog sind zu den folgenden:*

```
a = 211; b = 1663; m = 7875; x0 = 1243; zufall[1000];
s1 = ListPlot[pseudo, Frame -> True];
```

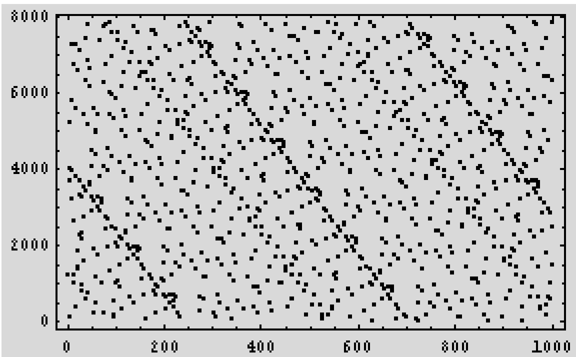


Abb. 3_I

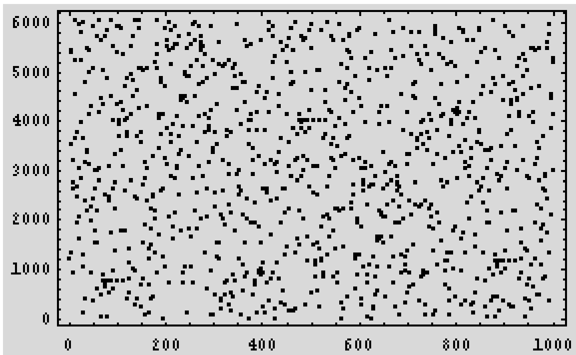


Abb. 3_{II}

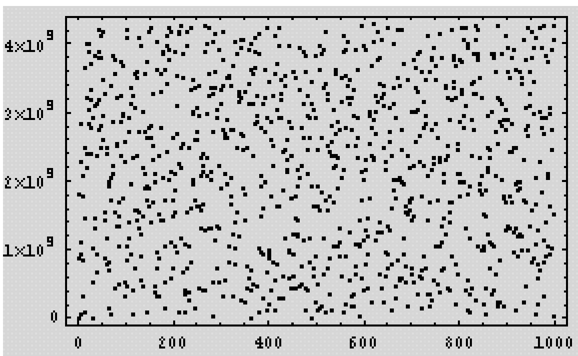


Abb. 3_{III}

In Abb. 3_I ist deutlich ein Muster zu erkennen, und das schon bei 1000 berechneten Zahlen – ***augenscheinlich*** *nicht akzeptabel für “Fast-Zufälligkeit”. Wie wirken die Abb. 3_{II} , 3_{III}? Lässt sich ein Muster entdecken – ein „Kreuz” in der Abb. 3_{II}? Bei mehr Berechnungen fällt die Entscheidung vielleicht leichter. In den folgenden Abbildungen sind 2000 Zahlen für die Varianten II bzw. III dargestellt.*

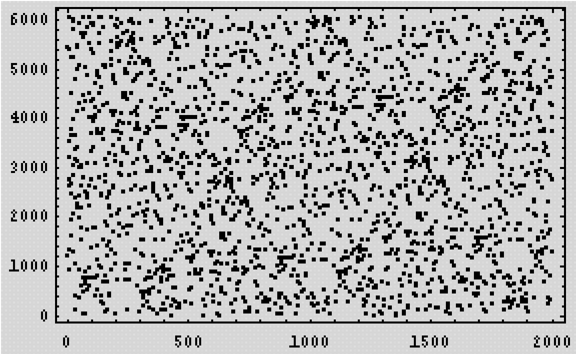


Abb. $3_{II\,2000}$

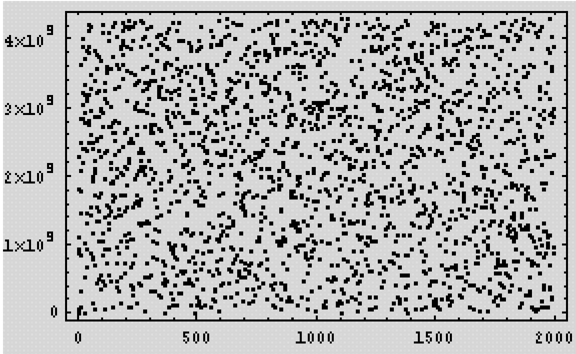


Abb. $3_{III\,2000}$

In Abb. $3_{II\,2000}$ sind die – nunmehr zwei „Kreuze” – vielleicht leichter zu erkennen.
Der DERIVE-Generator erscheint uns bisher ziemlich regellos, also bisher akzeptabel.

Eine Darstellung der erzeugten Zahlen kann aber auch in einem **Kreis** erfolgen:
Man berechnet aus jeder erzeugten Zahl $x \in [0, 1)$ mittels $x \cdot 2\pi$ einen Winkel in $[0, 2\pi)$ und legt damit einen Punkt auf dem Kreis fest. Eine Veranschaulichung der Ergebnisse ist nun auf sehr vielfältige Weise möglich (Abb. dazu auf S.9-11).

a) Es werden lediglich die Punkte auf dem Kreis dargestellt (Abb. 4).
b) Es werden die Radien gezeichnet, die zu diesen Punkten gehören (Abb. 5).
c) Es werden Sehnen gezeichnet, die folgendermaßen festgelegt werden:
   i) Man startet immer vom gleichen Punkt des Kreises und wählt nur den zweiten Punkt für eine Sehne aus den erzeugten Zahlen (Abb. 6).
   ii) Man verwendet jeden ermittelten Punkt des Kreises zum Zeichnen einer horizontalen Sehne (Abb. 7).
   iii) Man bestimmt jeweils zwei Punkte des Kreises durch das gewählte Verfahren und verbindet diese (Abb. 8).
   iv) Man startet von einem Punkt und verbindet diesen mit dem jeweils nächsten berechneten Punkt auf dem Kreis, der dann wieder als Ausgangspunkt einer neuen Sehne verwendet wird (Polygonzug) (Abb. 9).

Der optische Eindruck einer annähernd „gleichmäßig“ streuenden Verteilung der Punkte oder auch Strecken in jeder der Veranschaulichungen deutet dann auf eine genügende „Zufälligkeit“ der betrachteten Zahlenfolge hin.

Der geschlossene Zyklus mit der ersten Wiederholung einer Zahl führt bei einer Veranschaulichung der Ergebnisse in einem Kreis zu Abbildungen, die sich bei Berechnung weiterer Werte nicht mehr verändern – es kommen keine neuen Zahlen hinzu.

6. Erzeuge 10 (und anschließend 100) Zahlen unter Verwendung des linearen Kongruenzverfahrens mit den Parametern a = 6, b = 18, m = 40 und der Startzahl x_0=5.
Stelle die Ergebnisse in einem Kreis grafisch dar.

Nachfolgend eine Realisierung mit MATHEMATICA, wobei im Modul **zufk[n_]** *in der Liste* **ll** *letztlich alle berechneten Punkte auf dem Kreis und in der Liste li die Radien gespeichert werden. Durch die Anweisungen* **pz** *und* **pa** *werden die Abbildungen für die Varianten a), b) erzeugt.*

```
zufk[n_] := Module[{}, ll = {}; li = {};
  Do[x1 = Mod[a x1 + b, m];
     wink = 2 π x1 / m; x = Cos[wink]; y = Sin[wink];
     ll = Append[ll, {x, y}];
     li = Append[li, {Line[{{0, 0}, {x, y}}]}], {i, 1, n}]];
a = 6; b = 18; m = 40; x1 = 5; zufk[100];
pz = ListPlot[ll, AspectRatio -> 1, AxesOrigin -> {0, 0},
    PlotRegion -> {{0, 1}, {0, 1}}, Axes -> None, PlotStyle -> AbsolutePointSize[5] ];
ra = Show[Graphics[li], AspectRatio -> 1, DisplayFunction -> Identity];
kr =  ParametricPlot[{Sin[fi], Cos[fi]}, {fi, 0, 2 π}, AspectRatio -> 1,
    PlotRegion -> {{0, 1}, {0, 1}}, Axes -> None, DisplayFunction -> Identity];
pa = Show[ra, kr, DisplayFunction -> $DisplayFunction];
```

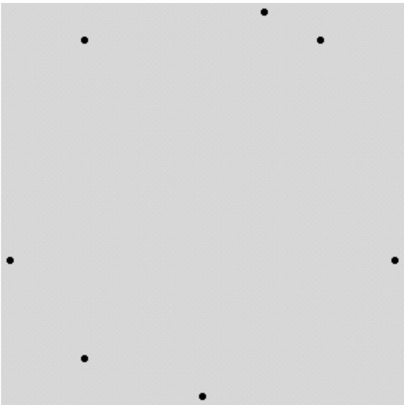


Abb. 4

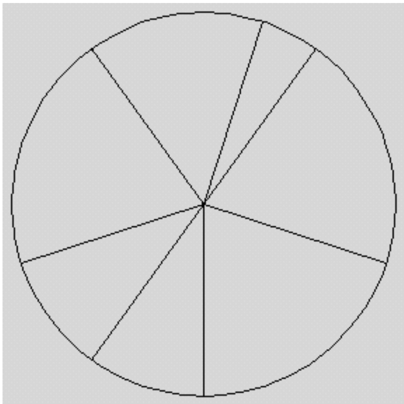


Abb. 5

100 oder auch mehr erzeugte Werte mit dem linearen Kongruenzverfahren unter Verwendung der obigen Parameter liefern nur 7 verschiedene Punkte bzw. Radien. Das entpricht sicher nicht unserem Wunsch nach „zufälligen“ Ergebnissen.

Die ersten 200 Werte unter Verwendung des ***147-Kongruenz-Generators*** *mit* $m = 10^5$ *und* $x_0 = 54327$ *sind in den nachfolgenden Abbildungen dargestellt.*

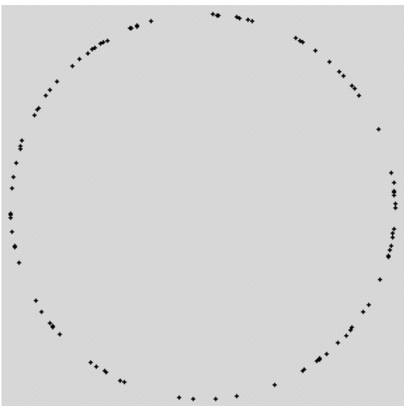


Abb. 4 $_{147}$

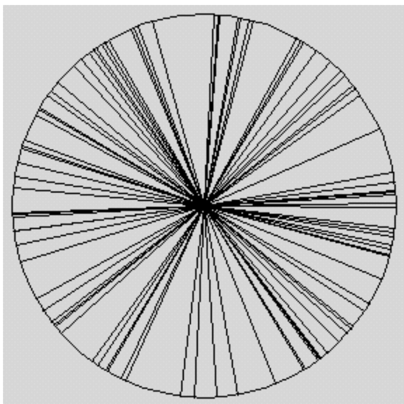


Abb. 5 $_{147}$

7. Erforsche die Zahlenfolgen, die mit dem
a) DERIVE-Generator,
b) in deiner PC-Software vorhandenen Zufallszahlen-Generator
erzeugt werden, durch verschiedene grafische Veranschaulichungen in einem Kreis.

Unterschiedliche Veranschaulichungen im Kreis (vgl. die Ausführungen zu den Abb. 4 - 9) für den von MATHEMATICA genutzten Zufallszahlen-Generator lassen sich in Analogie zum Modul **zufk[n_]** *mit dem Modul* **zrk[n_]** *und den nachfolgenden Anweisungen erreichen:*

```
zrk[n_] := Module[{}, lia = {}; lib = {}; lic = {}; lid = {}; lie = {}; xx = 0; yy = 1;
  Do[x1 = Random[]; x2 = Random[]; wink1 = 2 π x1; wink2 = 2 π x2;
    xa = Cos[wink1]; ya = Sin[wink1]; xb = Cos[wink2]; yb = Sin[wink2];
    lia = Append[lia, {Line[{{0, 0}, {xa, ya}}]}];
    lib = Append[lib, {Line[{{0, 1}, {xa, ya}}]}];
    lic = Append[lic, {Line[{{-xa, ya}, {xa, ya}}]}];
    lid = Append[lid, {Line[{{xa, ya}, {xb, yb}}]}];
    lie = Append[lie, {Line[{{xx, yy}, {xa, ya}}]}]; xx = xa; yy = ya   , {i, 1, n}]]
```

```
zrk[50];
al = Show[Graphics[lia], AspectRatio -> 1, DisplayFunction -> Identity];
alk = Show[k, al, DisplayFunction -> $DisplayFunction];
bl = Show[Graphics[lib], AspectRatio -> 1, DisplayFunction -> Identity];
blk = Show[k, bl, DisplayFunction -> $DisplayFunction];
cl = Show[Graphics[lic], AspectRatio -> 1, DisplayFunction -> Identity];
clk = Show[k, cl, DisplayFunction -> $DisplayFunction];
dl = Show[Graphics[lid], AspectRatio -> 1]; el = Show[Graphics[lie], AspectRatio -> 1];
```

Daraus resultieren die folgenden Abbildungen, bei denen zunächst nur 50 Durchläufe im Modul **zrk[n_]** *berücksichtigt wurden (Abb. 5 $_{Random\ 50}$ - 9 $_{Random\ 50}$), schließlich 200 Durchläufe (Abb. 5 $_{Random\ 200}$ - 9 $_{Random\ 200}$).*

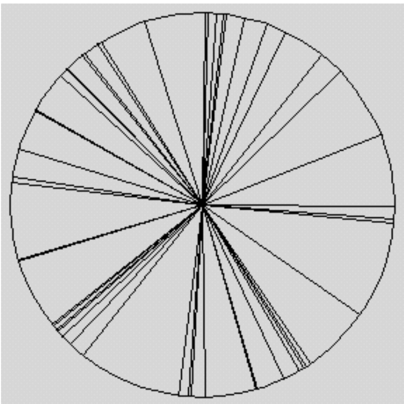


Abb. 5 $_{\textbf{Random 50}}$

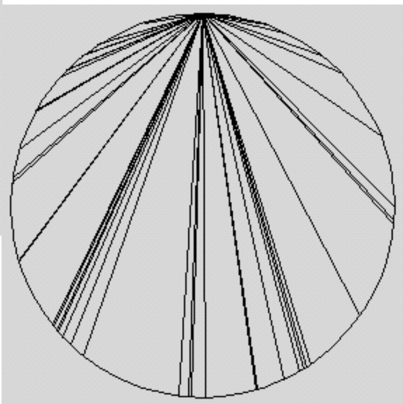


Abb. 6 $_{\textbf{Random 50}}$

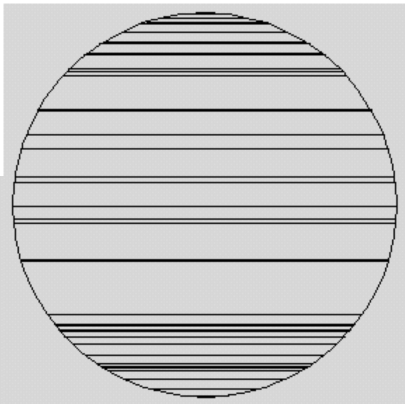


Abb. 7 $_{\textbf{Random 50}}$

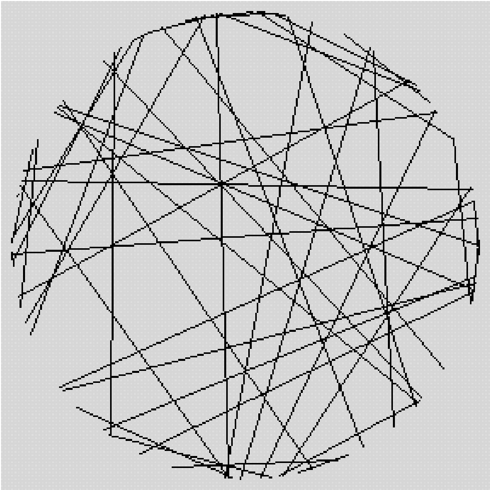


Abb. 8 ***Random 50***

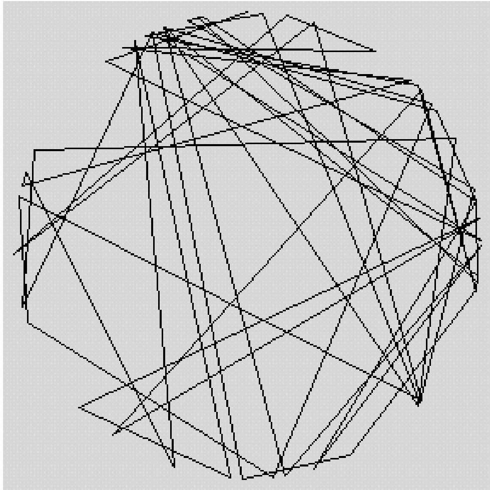


Abb. 9 ***Random 50***

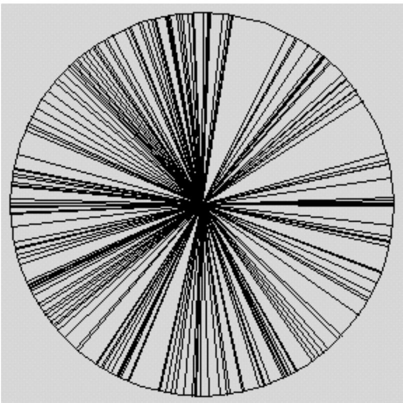


Abb. 5 ***Random*** *200*

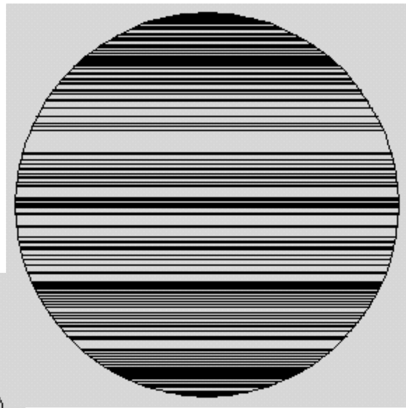


Abb. 7 ***Random*** *200*

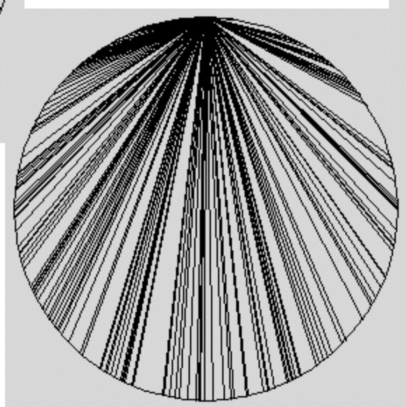


Abb. 6 ***Random*** *200*

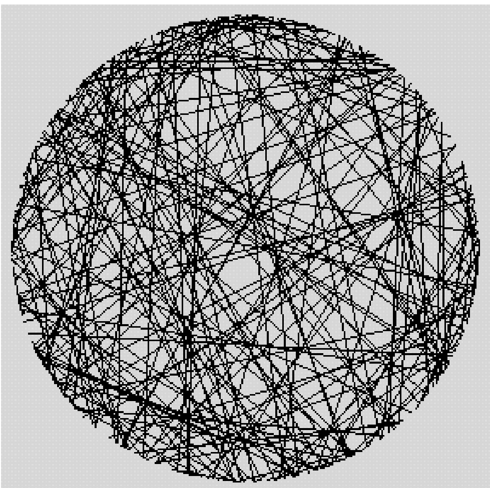


Abb. 8 ***Random*** *200*

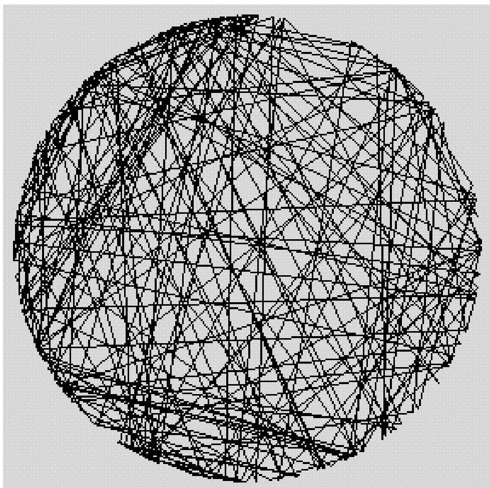


Abb. 9 ***Random*** *200*

Um die Unterschiede zwischen der speziellen Sehnenschar (Abb. 8) und dem Polygonzug (Abb. 9) leichter zu erkennen, wurde in diesen Fällen auf die Darstellung des Kreises verzichtet, das gilt auch für Abb. 4.
Die verschiedenen Strecken im Kreis haben den Vorteil, dass bereits bei relativ wenigen Berechnungen sowohl „leere Bereiche“ als auch „Klumpenbildungen“ (Häufungen) sehr deutlich ins Auge fallen. Der optische Eindruck für eine genügende Zufälligkeit einer großen Anzahl berechneter Zahlen ist mit diesen Veranschaulichungsverfahren jedoch nicht gegeben. Aus Abb. 4 - 8 ist auch nicht die Reihenfolge zu erkennen, in der die Zahlen ermittelt wurden. Diese Information liefert nur ein Polygonzug, der sich mit dem Auge noch verfolgen lässt.

Im weiteren geht es darum, wie gut die Bedingung 2 für Pseudo-Zufallszahlenfolgen bei der Verwendung unterschiedlicher Generatoren erfüllt wird. Dazu werden die Häufigkeiten für Zahlenpaare, -tripel veranschaulicht.
Interpretiert man je zwei aufeinanderfolgende Zahlen der Folge als Koordinaten eines Punktes der Ebene, so ergibt sich eine Punktwolke in einem **Quadrat** (Abb. 10).
Werden je drei aufeinander folgende Zahlen zu Raumkoordinaten zusammengefasst und entsprechend dargestellt, so erhält man Punkte in einem **Würfel** (Abb. 11).

8. Erzeuge n Zahlen (n = 500, 1000, 5000) unter Verwendung des linearen Kongruenzverfahrens mit den Parametern
I a = 211, b = 1663, m = 7875 und der Startzahl x_0 = 1243.
II des DERIVE-Generators und der Startzahl x_0 = 54327.
Stelle als Punkte in einem Koordinatensystem grafisch dar:
a) die Paare (x_{2k}, x_{2k+1}) für k = 0, ..., $\frac{n}{2}$, die sich bei Zusammenfassung von je zwei aufeinander folgender Zahlen ergeben;
b) die Tripel $(x_{3k}, x_{3k+1}, x_{3k+2})$ für k = 0, ..., $\frac{n}{3}$, die sich bei Zusammenfassung von je drei aufeinander folgender Zahlen ergeben.

Veranschaulichung der Paare als Menge von Punkten in dem Quadrat [0,m)x[0,m) für n = 1000 (Abb. $10_{I\,1000}$ bzw. $10_{II\,1000}$) bzw. n = 5000 (Abb. $10_{I\,5000}$ bzw. $10_{II\,5000}$). Dazu werden die folgenden bzw. analoge MATHEMATICA*-Anweisungen verwendet:*

```
a = 211; b = 1663; m = 7875; x0 = 1243; zufall[1000];
pseudo2 = Partition[pseudo, 2];
s2 = ListPlot[pseudo2, Frame -> True, AspectRatio -> Automatic, PlotStyle -> AbsolutePointSize[2]];
```

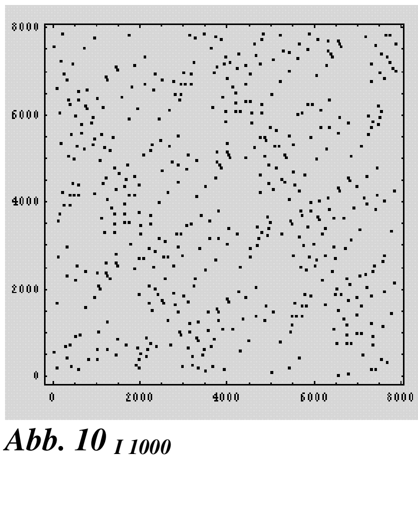


Abb. $10_{I\,1000}$

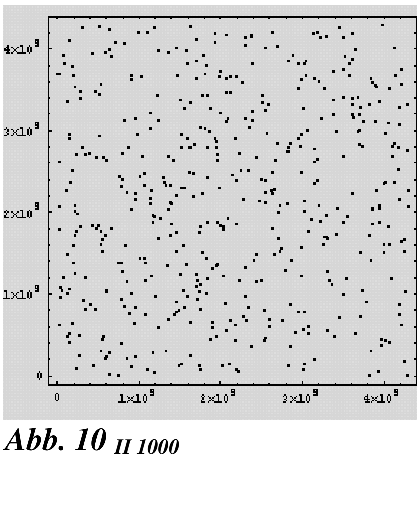


Abb. $10_{II\,1000}$

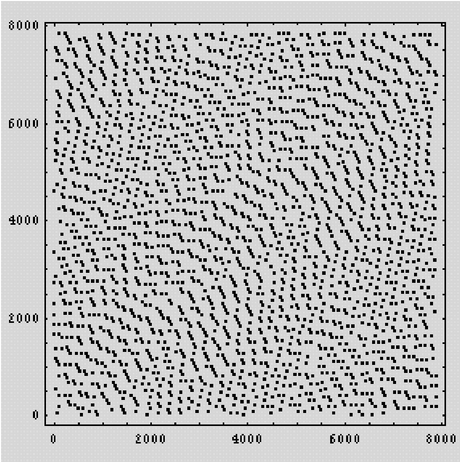


Abb. 10 $_{I\ 5000}$

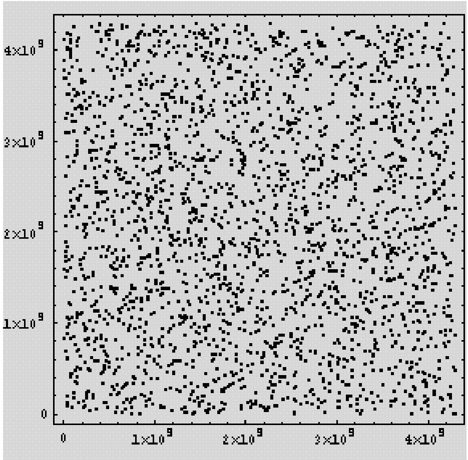


Abb. 10 $_{II\ 5000}$

Die folgenden Abb. 11 $_I$ bzw. 11 $_{II}$, zeigen zunächst die Tripel als Menge von Punkten in dem Würfel [0,m)x[0,m)x[0,m) für n = 1000, darunter für n = 5000.

```
a = 211; b = 1663; m = 7875; x0 = 1243; zufall[1000]; pseudo3 = Partition[pseudo, 3];
s3 = ScatterPlot3D[pseudo3, PlotStyle -> AbsolutePointSize[2]];
```

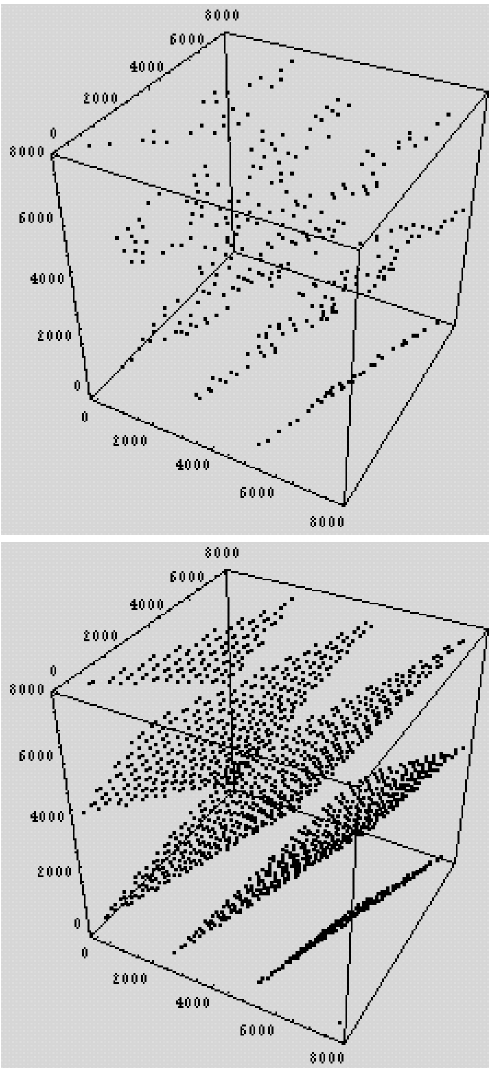


Abb. 11 $_I$

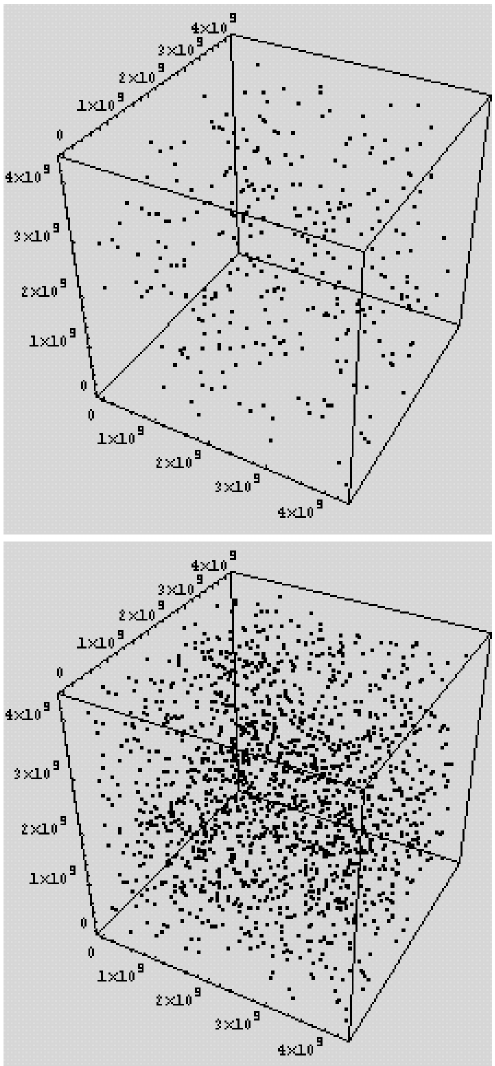


Abb. 11 $_{II}$

Entsprechen diese Abbildungen unserer Vorstellung von Zufälligkeit (= Regellosigkeit) oder ist ein Muster zu erkennen?
Für die Beantwortung dieser Frage ist nicht allein der verwendete Zahlengenerator entscheidend, sondern auch die Tatsache, wie viele Zahlen erzeugt wurden und zusätzlich, wie sie dargestellt werden.
Bei geringen Anzahlen sehen alle Grafiken oft noch recht zufällig aus. Erst bei großen Zahlenkolonnen ist unsere „Sichtkontrolle“ wirklich von Bedeutung: Ein Muster wird in einigen Fällen sichtbar. Die Abbildung verändert sich nun nicht mehr, obwohl die Zahlenkolonne wächst – es kommen keine neuen Zahlen hinzu. Es ist aber auch möglich, dass die eindimensionale Darstellung durchaus noch unserer Vorstellung von Zufälligkeit entspricht, aber im zwei- bzw. dreidimensionalem Fall Muster deutlich erkennbar werden. Das unterstreicht die Bedeutung der Bedingungen 2 und 3 für Pseudo-Zufallszahlenfolgen.
Mit größer werdendem n ist aus den Abbildungen immer deutlicher zu erkennen, ob eine „Fast-Zufälligkeit“ der berechneten Zahlen gelungen ist. Die verwendeten Parameter spielen offensichtlich die entscheidende Rolle. Die Variante I im Auftrag 8 weist auf einen schlechten Generator, die Variante II auf einen besseren, wohl sogar recht guten Generator hin.
Die nachfolgenden Abbildungen für den **147-Kongruenz-Generator** mit m = 10^5 und 5000 berechneten Zahlen lassen vermuten, dass dieser besser als Variante I und schlechter als Variante II ist.

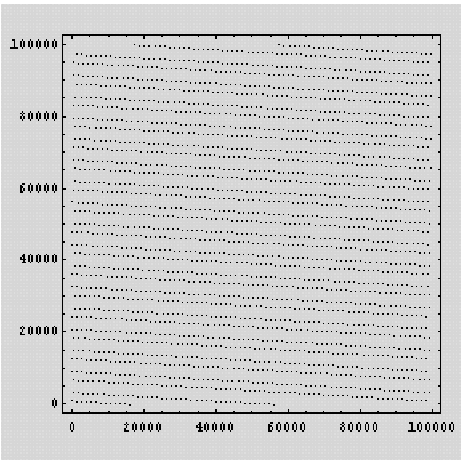


Abb. 10 *147-5000*

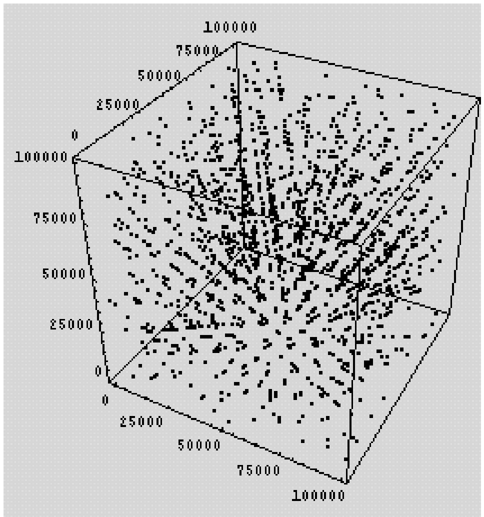


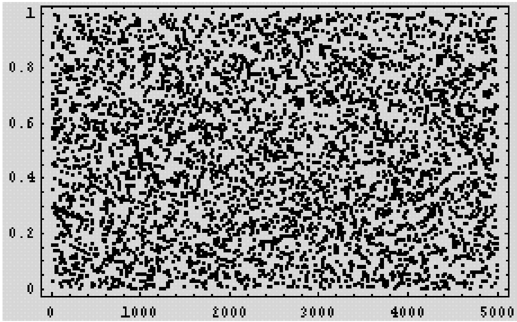
Abb. 11 *147-5000*

**9. Nutze nun den in deiner PC-Software vorhandenen Zufallszahlen-Generator zur Erzeugung von Zahlen.
Veranschauliche wieder wie eben die Ergebnisse für die Einzelwerte, für die sich ergebenden Paare und für die sich ergebenden Tripel.**

Verwendet man den eingebauten Zufallszahlen-Generator von MATHEMATICA, lassen sich die nachfolgenden Abbildungen erzeugen, wobei von 5000 Zahlen ausgegangen wurde.

```
pt = Table[Random[], {5000}]; pt2 = Partition[pt, 2]; pt3 = Partition[pt, 3];
s1 = ListPlot[pt, Frame -> True];
s2 = ListPlot[pt2, Frame -> True, AspectRatio -> Automatic, PlotStyle -> AbsolutePointSize[2]];
s3 = ScatterPlot3D[pt3, PlotStyle -> AbsolutePointSize[2]];
```

Abb. 3 *Random*



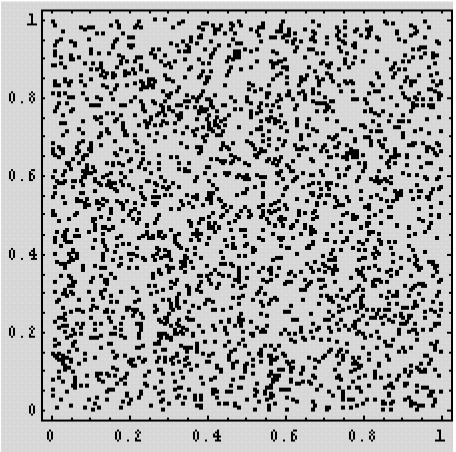


Abb. 10 *Random*

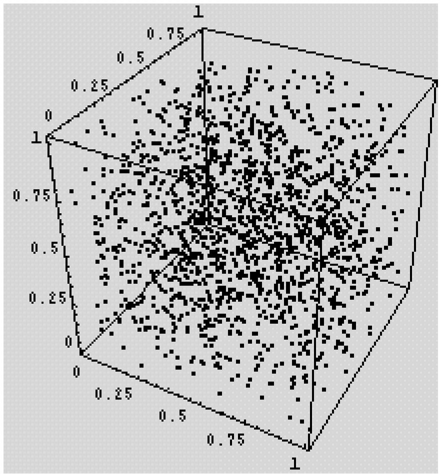


Abb. 11 *Random*

10. Verbinde 50 (und anschließend 200) Punkte im Raum, die durch einen Zufallszahlen-Generator erzeugt werden, durch einen Polygonzug. Interpretiere die entstehenden Abbildungen.

Bei Verwendung des Zufallszahlen-Generators von MATHEMATICA kann man mit den folgenden Anweisungen die Abb. 12 erhalten (n = 50 bzw. n = 200):

```
rkoord := {Random[], Random[], Random[]}; n = 50;
s31 = Show[Graphics3D[Line[Table[rkoord, {n}]]]];
```

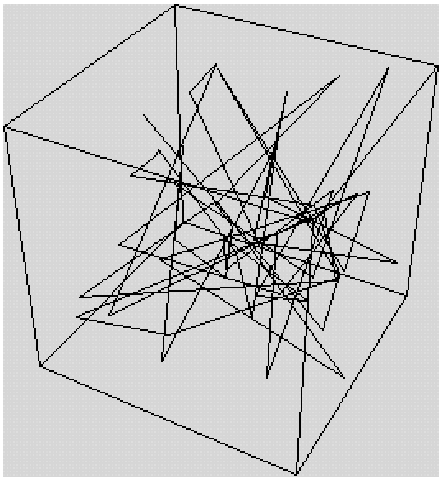


Abb. 12 *Random 50*

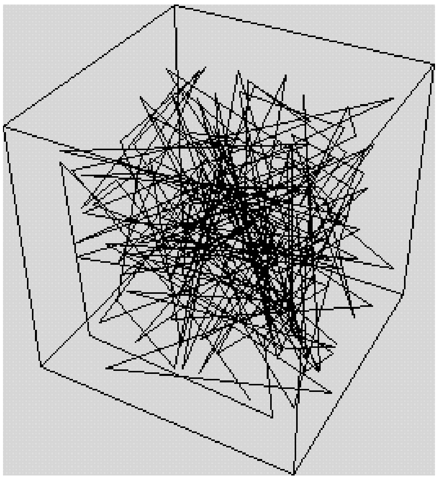


Abb. 12 *Random 200*

Dieser Polygonzug kann als Weg, den ein Teilchen zurücklegt, interpretiert werden. Wird der Eindruck, dass das Teilchen „regellos umherflitzt“, vermittelt? Ist eine räumliche Darstellung zusätzlich hilfreich (vgl. dazu auch die Abb. 9 *Random 50* *bzw. Abb. 9* *Random 200* *)?*

Eine räumliche Darstellung im Würfel für eine geringe Anzahl einfließender Punkte lässt wohl am deutlichsten den Weg des Teilchens erkennen – im dargestellten Fall tatsächlich sehr regellos.

Zusammenfassend kann festgestellt werden, dass es verschiedene Vorschläge gibt, die eine Einschätzung der Güte von Pseudo-Zufallszahlen ermöglichen. Hier wurden vor allem qualitative Kriterien betrachtet. Es gibt keine optimale Methode zur Beurteilung von Pseudo-Zufallszahlen: Die verschiedenen Methoden erfassen jeweils unterschiedliche Aspekte der generierten Pseudo-Zufallszahlen. Daher ist die parallele Anwendung mehrerer Gütekriterien zur Beurteilung von Pseudo-Zufallszahlen zwingend.

Literaturhinweise

[1] Fisz, M.: Wahrscheinlichkeitsrechnung und mathematische Statistik, VEB Deutscher Verlag der Wissenschaften, Berlin 1989.

[2] Greene, D. H., Knuth, D. E.: Mathematics for the analysis of algorithms, Boston 1982.

[3] Gundel, H., Schupp, P., Schweizer, U.: Einführung in die beurteilende Statistik, Deutsches Institut für Fernstudien an der Universität Tübingen, Beltz Verlag, Weinheim und Basel 1981.

[4] Gundel, H., Schupp, P., Schweizer, U.: Zufallszahlen, Monte-Carlo-Methode und Simulation, Deutsches Institut für Fernstudien an der Universität Tübingen, Beltz Verlag, Weinheim und Basel 1983.

[5] Knuth, D. E.: The Art of Computer Programming, Volume 2 / Seminumerical Algorithms, Addison-Wesley, Don Mills, Ontario 1998.

[6] Malitte, E.: Zufall mit dem Computer – geht das überhaupt? In: H. Hischer, Modellbildung, Computer und Mathematikunterricht, Bericht über die 16. Arbeitstagung des Arbeitskreises „Mathematik und Informatik“ in der GDM, Franzbecker, Hildesheim 1998.

[7] Mertsch, M.: Methoden zur Bestimmung und Begrenzung des Einflusses algorithmischer Zufallszahlgeneratoren auf simulative Untersuchungen, VDI-Verlag GmbH, Düsseldorf 1992.

[8] Strick, H. K.: Einführung in die beurteilende Statistik, Schroedel, Hannover 1998.