Pseudozufallszahlen

Die erste Methode zur rechnerischen Erzeugung von Zufallszahlen war der so genannte Quadratmitten-Generator (J.v.Neumann,1946), der jedoch zu große Nachteile hatte.

Heute benutzt man vorwiegend „lineare und nichtlineare Kongruenzgeneratoren“.

Linearer Kongruenzgenerator nach D.H.Lehmer (1948)

Dieser Generator erzeugt natürliche Zahlen x_i mittels der Iteration

$$x_{n+1} = (a \cdot x_n + c) \bmod m \;;$$

$0 \le x_0 < m,\ 0 < a < m,\ 0 \le c < m,\ 0 < m$; alle natürliche Zahlen.
x_0=Saat, a=Multiplikator, c = Inkrement, m = Modul

mod (Modulo) ist der ganzzahlige Rest bei der Division. Da bei der Modulo-Division nur ganze Zahlen im Bereich 0; 1; 2; … ; m-1 entstehen, kann man maximal **m** verschiedene Zahlen mit diesem Generator erzeugen. Die Folge $\{x_n\}$ läuft also nach spätestens m Iterationen in eine Periode.
Die Punkte $(x_n ; x_{n+1})$ liegen auf (parallelen) Geraden. (siehe Beispiele weiter unten)
Stellt man die Tripel $(x_n ; x_{n+1} ; x_{n+2})$ in einem 3D-System dar, so ergeben sich parallele Ebenen !

Beispiele:

1) x_0=1 a=4 c=2 m=9
1 6 8 7 3 5 4 0 2 1 6 8 7 3 5 4 0 2 1 6 8 7 3 5 4 0 2 1 6 8 7
Hier ergibt sich die Periodenlänge 9 (also m)

2) x_0=2 a=5 c=2 m=9
2 3 8 6 5 0 2 3 8 6 5 0 2 3 8 6 5 0 2 3 8 6 5 0 2 3 8 6 5 0 2
Hier ist die Periodenlänge gleich 6 (< m)

3) x_0=4 a=6 c=2 m=9
4 8 **5** 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
Hier ist die Periodenlänge gleich 1; außerdem entsteht eine Vorperiode !

4) x_0=5 a=6 c=18 m=40
5 8 26 **14 22 30 38 6** 14 22 30 38 6 14 22 30 38 6 14 22 30 38 6 14 22 30
Hier ist die Periodenlänge gleich 5; auch hier entsteht eine Vorperiode !

*Soll die **Periode = m** sein, so beachte man 3 Bedingungen (nach Prof. D.Knuth):*
- *c und m teilerfremd*
- *a-1 ist ein Vielfaches jedes Primteilers von m*
- *falls m ein Vielfaches von 4 ist, so ist auch a-1 ein Vielfaches von 4*

Für einen guten Generator ist zusätzlich zu beachten, dass die Korrelation aufeinanderfolgender Zahlen minimiert wird. Dies ist der Fall, wenn gilt: $a \approx \sqrt{m + 6 \cdot \frac{c}{m} \cdot (\frac{c}{m} - 1)}$
Ist die Periodenlänge kleiner als m, so kann (muss aber nicht) eine **Vorperiode** auftreten.

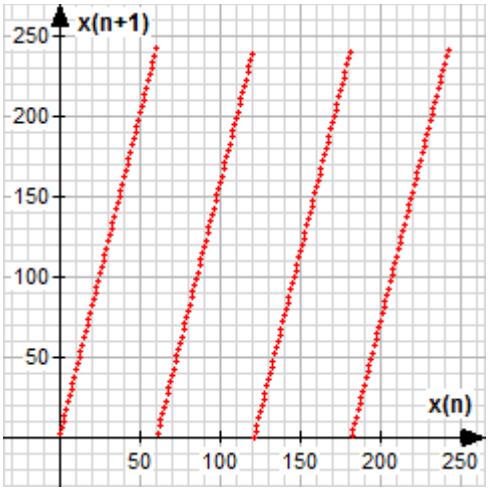
Ein guter Generator sollte gleichmäßig verteilte Zahlen erzeugen, sogar die Zahlen aus Teilfolgen sollten bereits gleichmäßig verteilt sein. Ein Gegenbeispiel ist : x_0=79 a=81 c=41 m=200
79 40 81 2 3 84 45 86 7 8 89 50 91 12 13 94 55 96 17 18 99 60 101 22 23 104 65 106 27 28 109 70 111 32 33 114 75 116 37 38 119 80 121 42 43 124 85 126 47 48 129 90 131 52 53 134 95 136 57 58 139
Die Periodenlänge ist zwar 200, jedoch sind die ersten 40 erzeugten Zahlen alle kleiner als 116 !!

Außerdem sollten bei einem guten Generator die Punkte $(x_n;x_{n+1})$ auf möglichst vielen Parallelen liegen, da somit die Verteilung auch besser wird.

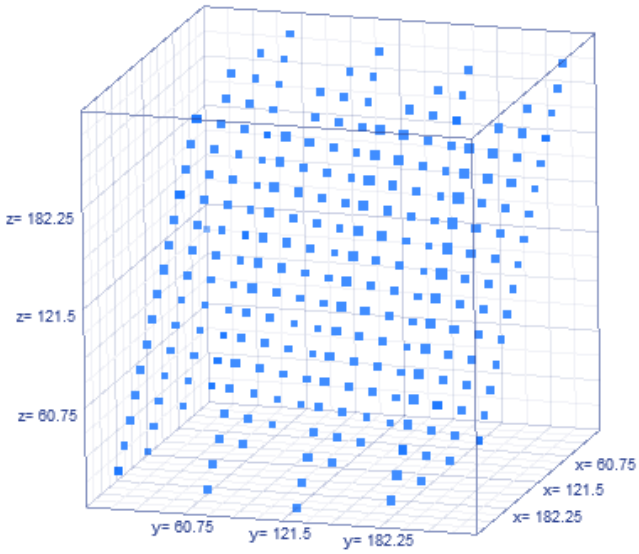
Beispiele :

a) $x_0=0$ a=4 c=2 m=243 = 3^5
Periodenlänge = m = 243

Alle Punkte liegen auf 4 Geraden mit
y = 4x+k·243 ; k=0 , - 1 , -2 , -3 .

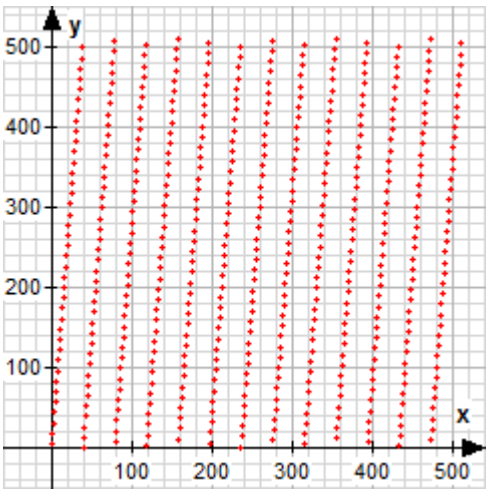


Die 3-dimensionale Darstellung zeigt, dass die Punkte in Ebenen liegen !



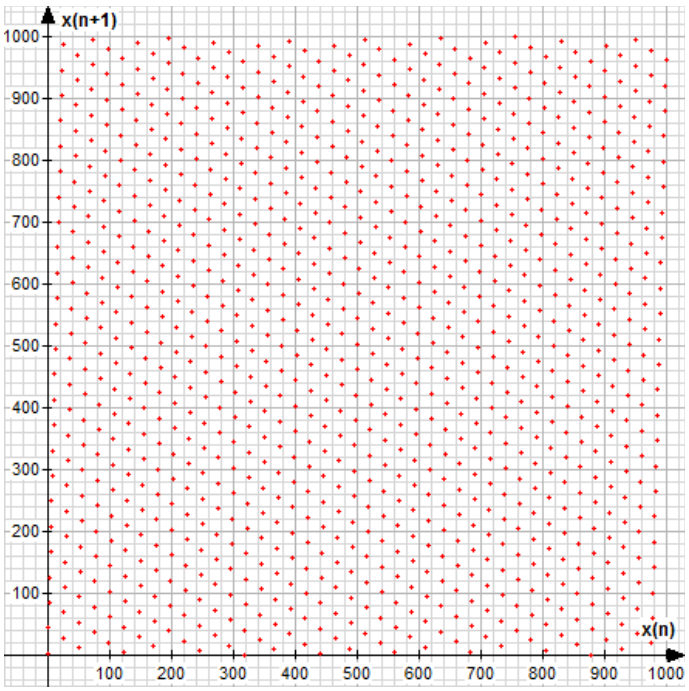
b) $x_0=3$ a=13 c=5 m=512 = 2^9
Periodenlänge = m = 512

Alle Punkte liegen auf 13 Geraden mit y = 13x+k·512 ;
k=0 , - 1 , -2 , … , -12 .



c) $x_0=1$ a=41 c=3 m=1000 = $2^3 \cdot 5^3$
Periodenlänge = 1000

Alle Punkte liegen auf 41 Geraden mit
y = 41x+k·1000 ; k=0 , - 1 , -2 , … , -40 .



Nur beim Beispiel c) sieht man eine recht gute Verteilung !

Der Ende der 70er Jahre gebaute Taschencomputer TI59 verwendete übrigens:
x_0=9991 a=24298 c=99991 m=$3^7 \cdot 7 \cdot 13$=199017 (Periodenlänge 199017)
Die Bedingungen des Informatik-Pioniers D.Knuth sind erfüllt, denn

- c=99991 und m=199017 sind teilerfremd
- a-1=24297=$3 \cdot 7 \cdot 13 \cdot 89$ und damit ein Vielfaches jedes Primteilers von m
- m ist kein Vielfaches von 4 , daher muss auch a-1 kein Vielfaches von 4 sein

Gilt zusätzlich $a \approx \sqrt{m - 6 \cdot \frac{c}{m} \cdot (1 - \frac{c}{m})}$? Nein, denn $24298 \neq 446$

Java.util.random verwendet $\mathbf{x_{n+1} = (25214903917 \cdot x_n + 11) \bmod 2^{48}}$
Als Startwert wird die aktuelle Systemzeit (Typ Long) verwendet.
Von den 48 Bits (0 .. 47) des Ergebnisses verwendet Java jedoch nur die Bits Nr. 16 bis 47.

Wie findet man die **Periodenlänge** heraus ??

Wir betrachten das Beispiel : $x_0=0$ $a=5$ $c=3$ $m=1000$

```
Folge xn :    0  3  18 93 468 343 718 593 968 843 218 93 468 343 ...
```

Nach R.W. Floyd kann man die Länge der Periode sowie ggfs. der Vorperiode mithilfe von zwei Spielmarken ermitteln:
Die langsame Spielmarke ('slow') rückt in Einserschritten vor , $x0 \rightarrow x1 \rightarrow x2 \rightarrow \ldots$ und die schnelle Spielmarke ('quick') in Zweierschritten , $x0 \rightarrow x2 \rightarrow x4 \rightarrow \ldots$.

Irgendwann gilt „quick = slow“, d.h. quick hat slow überrundet und somit haben beide die gleiche Zahl z (hier: 968) erreicht. Ist nun $z = x_0$, so liegt eine reine Periode vor, andernfalls muss slow nochmals von z an auf die Reise geschickt werden, bis er z zum zweiten mal erreicht. Damit ist die **Länge der Periode** bekannt.

```
slow:  0  3  18 | 93 468 343 718 593 968 843 218 | 93 468 343 718 593 968 843 218 | 93
quick: 0 18 468  718 968 218 468 718 968
```

Um die **Länge der Vorperiode** zu ermitteln, wird quick zum Start geschickt, um sich von dort aus mit Einserschritten zu bewegen. Gleichzeitig bewegt sich slow von z aus weiter. Haben beide die gleiche Zahl erreicht (hier: 93), so ist die zurückgelegte Strecke die Länge der Vorperiode.

```
slow neu:  |968| 843 218  93  468  343  718  593  968  843  218  93 ...
quick neu:    0    3  18  93  468  343  718  593  968  843  218  93 ...
```

JAVA-Methode mit f = (a*x+c) % m :

```
public static long[] periodenLaenge(long x0, long a, long c, long m) {
	long pLaenge = 0, vorpLaenge = 0;
	long slow = x0, quick = x0;
	do {
		slow = f(slow, a, c, m);
		quick = f(f(quick, a, c, m), a, c, m);
		pLaenge = pLaenge + 1;
	} while (slow != quick);
	if (slow != x0) { // es gibt eine Vorperiode !
		pLaenge = 0; // nochmals Periodenlänge bestimmen
		do {
			slow = f(slow, a, c, m);
			pLaenge = pLaenge + 1;
		} while (slow != quick);
		quick = x0; // Vorperiodenlänge bestimmen
		do {
			quick = f(quick, a, c, m);
			slow = f(slow, a, c, m);
			vorpLaenge = vorpLaenge + 1;
		} while (slow != quick);
	}
	long[] feld = { pLaenge, vorpLaenge };
	return feld;
}
```

Bemerkung: Mit dem gleichen Algorithmus kann man die Periode jeder beliebigen Funktion f ermitteln, z.B. auch von **f(x) = (x·x+c) mod n** beim „Pollard-Rho“ .

Beispiel: $n=817=19\cdot43$ $c = 15$ $x_0=0$

```
0 15 240 425 83 368 634 7 64 26 691 368 634 7 64 26 691 368 634 7

PeriodenlängeRho = 6
VorPeriodenlängeRho = 5
```