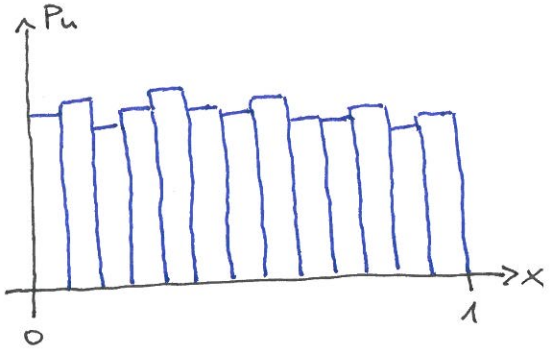
Zufallszahlen

In der heutigen Vorlesung wollen wir uns mit "Zufallszahlen" beschäftigen. Insbesondere wollen wir klären, was unter Zufallszahlen zu verstehen ist und wie man diese erzeugen kann.
In den nächsten Vorlesungen werden wir dann einige der mächtigsten und anwendungsstärksten Algorithmen kennenlernen, die allesamt auf ebendiesen Zufallszahlen aufbauen.

Was sind Zufallszahlen?

Unter Zufallszahlen verstehen wir eine Folge reeller Zahlen $\{x_i\} = x_1, x_2, \dots, x_N$ mit $x_i \in]0,1]$, welche die folgenden Eigenschaften besitzt:

1) die Zufallszahlen sind gleichverteilt im Intervall $]0,1]$



P_n: Anzahl der x_i im n-ten Bin $\left]\frac{n}{M}, \frac{n+1}{M}\right]$
mit $n = 0, 1, 2, \dots, M-1$
M = # des Bins.

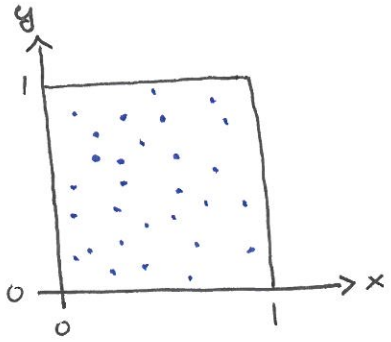
Eine Gleichverteilung stellt sich im Limes $N \to \infty$ ein, d.h.

$$\lim_{N\to\infty} p_n = \frac{1}{M}$$

unabhängig von n

2) es gibt keine Korrelationen zwischen aufeinanderfolgenden Zufallszahlen

Etwaige Korrelationen lassen sich z.B. durch einen Spektraltest ausfindig machen: Trage (x_i, x_{i+1}) als Koordinatentupel (x,y) auf.
Regelmäßige Muster → Korrelation



Ein quantitatives Maß für Korrelationen:

$$\chi = \langle x_i\, x_{i+1} \rangle - \langle x_i \rangle^2$$

mit den Mittelwerten

$$\langle x_i \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i \qquad \lim_{N\to\infty} \langle x_i \rangle = \frac{1}{2}$$

$$\langle x_i\, x_{i+1} \rangle = \frac{1}{N-1} \sum_{i=1}^{N-1} x_i\, x_{i+1}$$

Falls es keine Korrelationen gibt, gilt:

$$x_i\, x_{i+1} \xrightarrow{\text{im Mittel}} x_i \langle x_{i+1} \rangle$$

$$\langle x_i\, x_{i+1} \rangle \longrightarrow \langle x_i \rangle^2$$

$$\lim_{N\to\infty} \chi(N) = 0$$

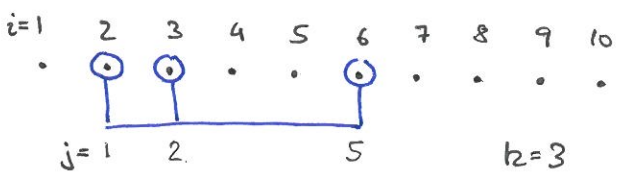
3) Verallgemeinerung von 2):

Es gibt keine Korrelationen zwischen beliebiger Anzahl von Zufallszahlen mit beliebigen Abständen innerhalb der Folge

$$\chi_{[j_1, j_2, \dots, j_k]} = \langle x_{i+j_1} \cdot x_{i+j_2} \cdot \dots \cdot x_{i+j_k} \rangle - \langle x_i \rangle^k$$

$$\to 0 \quad \text{für} \quad N \to \infty$$

Bsp.:



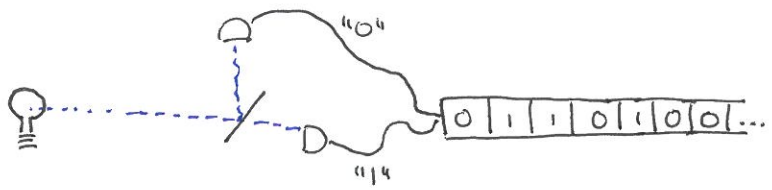
$$\to \chi_{[1,2,5]} = \langle x_{i+1} x_{i+2} x_{i+5} \rangle - \langle x_i \rangle^3$$

Erzeugung von Zufallszahlen

Echte Zufallszahlen sind hart zu erzeugen. Derzeit gibt es zwei große Klassen von Generatoren:

1) physikalische Prozesse

- deterministisch: klassische chaotische Systeme
  typischerweise große Systeme & Variation der Anfangsbedingungen → etwa atmosphärisches Rauschen
- nicht-deterministisch: quantenmechanische Prozesse
  → etwa atomarer Zerfall, Photon-Splitter



Kommerzieller USB-Stick (idquantique.com) mit Generationsrate von 4 Mbits/s.
(Oftmals z. B. kosmisch)

2) algorithmische Generatoren

Deterministische Algorithmen können "pseudo-zufällige" Zahlenfolgen erzeugen, etwa über ein iteratives Verfahren $x_{i+1} = f(x_i)$.
Vorteil ist, daß Zufallszahlen schnell und reproduzierbar erzeugt werden können, d.h. identische Startworte ("Seeds") x_0 liefern immer dieselbe Sequenz an Zufallszahlen.

Warnung: Pseudo-Zufallszahlen sind nicht zufällig, sondern komplett deterministisch erzeugt. Sie sehen lediglich zufällig aus, wenn der erzeugende Algorithmus unbekannt ist.

Für unsere Zwecke mögen Pseudo-Zufallszahlen ausreichen. Dennoch: Traue niemals Zufallszahlengeneratoren!

Erzeugung von Pseudo-Zufallszahlen

Die einfachste Form, Pseudo-Zufallszahlen, numerisch zu erzeugen ist anhand eines linearen kongruenten Generators mit Iterationsformel

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

GGL (IBM und Apple):

$$a = 16\,807$$
$$c = 0$$
$$m = 2^{31} - 1$$

Ein solcher 32-bit Generator ist periodisch – die Sequenz wiederholt sich nach $2^{31} - 1$ Iterationen. Auf heutigen CPUs können wir etwa 500 Millionen Pseudo-Zufallszahlen pro Sekunde via GGL erzeugen, d.h. unsere Periode ist lediglich 4 Sekunden. Derartige Verfahren sollten heutzutage nicht mehr angewandt werden!

Eine bessere Alternative sind sogenannte Lagged Fibonacci-Generatoren, deren Iterationsformel gegeben ist als

$$X_n = X_{n-p} \otimes X_{n-q} \mod m$$

wobei $p > q$ sei und $\otimes$ eine der Binäroperationen $+, -, \times, \oplus$ (exclusive or). m ist typischerweise eine 2er-Potenz, häufig 2^{32} oder 2^{64}, allgemein 2^M. Desweiteren sei eine Seed-Sequenz der ersten p Elemente $x_1, x_2, \ldots, x_p$ gegeben.

Die maximale Periode ist

$(2^p - 1) \cdot 2^{M-1}$ für $+, -$

$(2^p - 1) \cdot p$ für $\oplus$

$(2^p - 1) \cdot 2^{M-3}$ für $\times$ (oder 1/4 des ersten Falls für $+, -$)

Eine gute Wahl für $(p, q, \otimes)$ sind

$(2281, 1252, +)$ $(9689, 5502, +)$ $(44497, 23463, +)$

Alle diese Optionen haben extrem lange Perioden aufgrund der großen Seed-Sequenzen. Letztere werden typischerweise durch einen linearen kongruenten Generator (siehe oben) erzeugt.
Der Lagged Fibonacci-Generator ist zudem ein sehr schneller Generator, der sich vektorisieren und pipelinen läßt.

Noch ausgefeiltere Generatoren nutzen zahlentheoretische Zusammenhänge, wie etwa der "Mersenne twister" (Matsumoto & Nishimura, 1997) oder der "Well Generator" (Panneton & L'Ecuyer, 2004).