Universität Potsdam
Mathematisch-Naturwissenschaftliche Fakultät

Institut für Mathematik

Bachelorarbeit

im Studiengang Mathematik - Schwerpunkt Numerik

zur Erlangung des akademischen Grades
Bachelor of Education

Thema:	Pseudozufallszahlen
Autor:	Jasmin Sophie Pusch jpusch@uni-potsdam.de MatNr. 775752
Version vom:	14. Juni 2018
1. Betreuerin:	Dr. Jana de Wiljes
2. Betreuer:	Prof. Dr. Sebastian Reich

Inhaltsverzeichnis

1. Einleitung

Pseudozufallszahlen begleiten uns überall hin, sei es in Form eines Shuffle-Modus auf Musikprogrammen oder in kryptographischen Anwendungen zum Verschlüsseln von Daten. Sogar in Monte-Carlo-Methoden wie beispielsweise zum Modellieren von komplexeren Systemen wie Wetter- und Klimaprozesse oder auch in der Wirtschaftsmathematik sind Pseudozufallszahlen vonnöten.

Was macht Pseudozufallszahlen eigentlich aus? Hierbei handelt es sich um Zufallszahlen, die mithilfe eines Algorithmus generiert werden, wodurch sie nur für den Nutzenden eines Zufallsgenerators, der diesen Algorithmus nicht kennt, zufällig erscheinen. Echte Zufallszahlen hingegen werden nur durch Prozesse wie einen Würfelwurf oder das Messergebnis eines Quantenteilchens bezüglich des Ortes generiert.

Die Arten, Pseudozufallszahlen zu generieren, sind vielfältig und werden bis heute immer weiter verbessert: einer der ersten Generatoren war der Quadrat-Mitten-Generator von John von Neumann, der im 20. Jahrhundert konzipiert wurde. In der Mitte des selbigen Jahrhunderts fand Derrick Lehmer den linearen Kongruenzgenerator, der ihm zu Ehren auch den Namen „Lehmer-Generator“ trägt. Kongruenzgeneratoren im Allgemeinen beruhen auf den Regeln der Restklassenarithmetik und sind auch in mehreren Dimensionen anwendbar, wie der Matrixkongruenzgenerator oder der Mersenne-Twister, der Ende des 20. Jahrhunderts von Makoto Matsumoto und Takuji Nishimura erfunden wurde und auch heute noch zu den besten Pseudozufallsgeneratoren zählt und unter anderem bevorzugt für Monte-Carlo-Methoden eingesetzt wird. Es existieren auch Generatoren, deren Folgen gleichverteilt modulo Eins sind und den Kriterien von Weyl entsprechen sowie welche, die einen zufälligen Algorithmus besitzen wie Donald Knuths „Algorithmus K“. Gleichzeitig ist auch wichtig, wie gut diese Generatoren sind: Sind die Zufallszahlen wirklich zufällig genug? Sind sie gleichverteilt? Ist die Periodenlänge maximal? Daher werden sie statistischen Tests unterzogen wie dem Chi-Quadrat-Test oder ihre serielle Autokorrelation wird untersucht.

In dieser Bachelorarbeit wird zunächst auf die Grundlagen zum Erzeugen von Pseudozufallszahlen sowie auf stochastische Kenngrößen für die Überprüfung der Güte von Pseudozufallsgeneratoren eingegangen. Danach werden diverse Pseudozufallsgeneratoren vorgestellt: angefangen bei Generatoren von historischer Bedeutung wie dem Quadrat-Mitten-Generator, über Pseudozufallsgeneratoren, die auf Restklassenarithmetik und dem Satz von Weyl über die Gleichverteilung modulo Eins basieren bis hin zu Pseudozufallsgeneratoren, deren Algorithmen ebenso zufällig gewählt sind wie die generierte Pseudozufallszahl selbst. Des Weiteren werden auch Gütetests für Zufallsgeneratoren thematisiert. Abschließend werden einige Pseudozufallsgeneratoren mithilfe von Python implementiert und deren Verteilung simuliert sowie miteinander verglichen. Abschließend folgt kurzer Einblick über Monte-Carlo-Methoden.

Als Quellen wurden unter anderem Donald Knuths *The Art of Computer Programming, Vol.2*, für die Grundlagen in der Restklassenarithmetik Jürg Kramers *Zahlen für Einsteiger* und für die Grundlagen in Stochastik und Statistik Norbert Henzes *Stochastik für Einsteiger* sowie Peter Ecksteins *Repetitorium Statisik* genutzt.

2. Klärung der Grundbegriffe

2.1. Arithmetik

2.1.1. Primzahlen

Primzahlen sind gerade für Kongruenzgeneratoren von großer Bedeutung. Doch zunächst ist zu klären, was eine Primzahl ist (vgl. [Kra08, S. 25 ff.]).

Definition 2.1. *Eine natürliche Zahl $p > 1$ heißt Primzahl, wenn p nur triviale Teiler, das heißt 1 und p selbst, hat.*

Für die Teilbarkeit von anderen Primzahlen gilt nach folgendes Lemma:

Lemma 2.2. *Für jede natürliche Zahl $a > 1$ gibt es eine Primzahl p, die a teilt.*

Beweis. Siehe [Kra08, S. 25 f.]. □

Desweiteren gilt folgender Satz:

Satz 2.3 (Fundamentalsatz der elementaren Zahlentheorie)**.** *Jede natürliche Zahl $a > 0$ kann als Produkt von $r \in \mathbb{N}$ verschiedenen Primzahlpotenzen eindeutig dargestellt werden:*

$$a = p_1^{a_1} \cdot ... \cdot p_r^{a_r} \tag{2.1}$$

Dabei sind die Primzahlen $p_1, ...p_r$ paarweise verschieden und die Exponenten $a_i > 0$ mit $i \in \{1, ..., r\}$ natürlich.

Beweis. Der Beweis findet sich in [Kra08, S. 31 ff.] □

2.1.2. Restklassen

Für Kongruenzgeneratoren sind Restklassen wichtig. Im Folgenden werden Restklassen und ihre Rechenregeln betrachtet (vgl. [Kra08, S. 217 ff.]).

Definition 2.4. *Seien $m \in \mathbb{N}$ mit $m > 0$ und $a, b \in \mathbb{Z}$. Die Relation*

$$a \equiv b \ mod \ m \tag{2.2}$$

wird a kongruent b modulo m genannt, wobei m das Modul der Kongruenz ist.

b mod m bezeichnet auch die Restklasse von b modulo m. Die Restklassen werden in der Menge

$$\mathbb{Z}_m := \{\overline{0}, \overline{1}, ..., \overline{m-1}\} \tag{2.3}$$

zusammengefasst. Die Addition und Multiplikation mit Restklassen wird wie folgt definiert:

Definition 2.5. *Seien $\overline{a}, \overline{b} \in \mathbb{Z}_m$. Dann lässt sich wie folgt mit $\overline{a}, \overline{b}$ rechnen:*

$$\overline{a} + \overline{b} = \overline{a+b} \qquad \textit{(Addition)}$$
$$\overline{a} \bullet \overline{b} = \overline{a \cdot b} \qquad \textit{(Multiplikation)}$$

Bemerkung 2.6. Das neutrale Element in der Addition ist wie bei den ganzen Zahlen selbst die 0, und wie bei den ganzen Zahlen kann man zwei Restklassen miteinander subtrahieren, d.h. das Rechnen mit Inversen ist möglich. Für die Multiplikation ist 1 das neutrale Element analog zu den ganzen Zahlen. Anders als bei den ganzen Zahlen kann man hier Inverse bestimmen. Hierfür löse man die Kongruenz:

$$a \cdot a^{-1} \equiv 1 \bmod m \tag{2.4}$$

Definition 2.7. *Restklassen $\overline{a} \in \mathbb{Z}_m$, die ein inverses Element besitzen, werden Einheiten in $\mathbb{Z}_m$ genannt. Es gilt $ggT(a, m) = 1$.*

Wie die Anzahl der Einheiten in Z_m bestimmt werden können, wird in Kapitel 2.1.5 genauer erklärt.

2.1.3. Chinesischer Restsatz

Es wird nun das Lösen linearer Kongruenzen betrachtet. Wie beim Lösen linearer Funktionen in $\mathbb{Z}$ gibt es eine Lösung $\overline{x}$, so dass:

$$a \cdot x \equiv b \bmod m \qquad a, b \in \mathbb{Z} \tag{2.5}$$

Doch wie findet man eine Lösung $\overline{x}$, so dass mehrere Kongruenzen

$$x \equiv a_1 \bmod m_1, ..., \; x \equiv a_r \bmod m_r, \qquad r \in \mathbb{N} \tag{2.6}$$

gelöst werden?

Satz 2.8 (Chinesischer Restsatz)**.** *Seien $m_1, ..., \; m_r \in \mathbb{N}$ mit $m = \prod_{i=1}^{r} m_i$ paarweise teilerfremd und $a_1, ..., \; a_r \in \mathbb{Z}$ beliebig. Dann lässt sich* (2.6) *durch genau eine Restklasse x mod m lösen.*

Beweis. Ein guter Beweis ist nachzulesen in [Kra08, S. 225 ff.]. □

Der Chinesische Restsatz 2.8 und der Fundamentalsatz der elementaren Zahlentheorie 2.3 sind nun insofern anwendbar, dass wenn das Modul m wie in Gleichung (2.1) dargestellt werden kann, dass sich $\mathbb{Z}_m$ über einen Isomorphismus folgendermaßen ausdrücken lässt [Kra08, S. 228 ff.]:

$$\mathbb{Z}_m \cong \mathbb{Z}_{p_1^{k_1}} \times ... \times \mathbb{Z}_{p_r^{k_r}} \tag{2.7}$$

2.1.4. Eulersche Phi-Funktion

Definition 2.9. *Die Anzahl der Einheiten von $\mathbb{Z}_m$ abgebildet durch die die Eulersche φ-Funktion mit*

$$\varphi : \mathbb{N} \to \mathbb{N}.$$

Die Eulersche φ-Funktion kann folgende Formen annehmen [Rib06, S. 28]:

- für Primzahlen p ist
$$\varphi(p) = p - 1 \tag{2.8}$$
- für Primzahlpotenzen mit $k \in \mathbb{N}$ ist
$$\varphi(p^k) = p^{k-1}(p-1) = p^k(1 - \frac{1}{p}) \tag{2.9}$$
- für m in der Darstellung (2.1) ist es möglich, die jeweiligen Eulerschen φ-Funktionen miteinander zu multiplizieren, demnach:
$$\varphi(m) = \varphi(p_1^{k_1}) \cdot ... \cdot \varphi(p_r^{k_r}) \qquad r \in \mathbb{N}. \tag{2.10}$$

Es gelten folgende Sätze für die Eulersche φ-Funktion:

Satz 2.10 (Satz von Euler)**.** *Für Einheiten in $\mathbb{Z}_m$ gilt $a^{\varphi(m)} \equiv 1$ mod m .*

Satz 2.11 (Kleiner Satz von Fermat)**.** *Für $a \in Z$ teilerfremd zur Primzahl p gilt $a^{p-1} \equiv 1$ mod p,*

wobei Satz 2.11 ein Spezialfall von Satz 2.10 darstellt, da die Gleichung (2.8) für Primzahlen gilt und diese in der Gleichung aus dem Satz von Euler eingesetzt werden muss (vgl. [Rib06, S. 28]).

Beweis des Satzes von Euler. Ein guter Beweis dafür findet sich in [Rib06, S. 28 f.]. □

2.1.5. Carmichael-Funktion

Die Carmichael-Funktion wurde 1912 von Robert Carmichael [Car12, S. 23] eingeführt. In ihr kommt auch die Eulersche φ-Funktion zu tragen. Sie ist wie folgt definiert:

Definition 2.12. *Seien $a, m \in \mathbb{N}$ Es gilt:*

- *für $a \in \{0, 1, 2\}$: $\lambda(2^a) = \varphi(2^a)$*
- *für $a > 2$: $\lambda(2^a) = \frac{1}{2}\varphi(2^a)$*
- *für eine Primzahl $m = p > 2$ und a beliebig: $\lambda(p^a) = \varphi(p^a)$*

- *und für m als Darstellung von Primzahlpotenzen laut* (2.1)*:* $\lambda(m) = \lambda(2^a) \cdot \lambda(p_1^{a_1}) \cdot ... \cdot \lambda(p_r^{a_r})$.

Somit kann die Kongruenz von Satz 2.11 auch als

$$a^{\lambda(p)} \equiv 1 \bmod p \tag{2.11}$$

geschrieben werden.

2.1.6. Quadratischer Rest

Definition 2.13. *Sei $a \in \mathbb{Z}_m$ und teilerfremd zu $m \in \mathbb{N}$. Dann heißt a quadratischer Rest modulo m, falls*

$$x^2 \equiv a \ mod \ m \tag{2.12}$$

lösbar ist. Sonst heißt a quadratischer Nichtrest modulo m.

Mithilfe der Gleichungen (2.1) und (2.7) ist erkennbar, dass a genau dann quadratischer Rest modulo m ist, wenn a für alle $i \in \{1, ..., r\}$ auch quadratischer Rest von $p_i^{k_i}$ ist. Zudem ist a teilerfremd zu p ein quadratischer Rest von p^k, wenn a ein quadratischer Rest von p ist [Kra08, S. 236 f.]

Nun ergibt sich die Frage: wie kommt man an die quadratischen Reste einer Zahl als Produkt zweier Primfaktoren? Man betrachte hierfür den Restklassenkörper $\mathbb{F}_p = \{\overline{1}, ..., \overline{p}\}$ mit der zyklischen, multiplikativen Gruppe $\mathbb{F}_p^\times$.

Definition 2.14. *Eine Restklasse $\overline{w}$, welche $\mathbb{F}_p^\times$ erzeugt, heißt Primitivwurzel modulo p. Für sie gilt:*

$$\mathbb{F}_p^\times = \{\overline{w}^0, ..., \overline{w}^{p-2}\}. \tag{2.13}$$

Es gibt genau $\varphi(p-1)$ Primitivwurzeln modulo p [Kra08, S. 238 ff.]. Die Menge der quadratischen Reste ist gegeben durch [Kra08, S. 243]

$$\{\overline{w}^0, ..., \overline{w}^{p-3}\}$$

Anhand eines Beispiels, wie es in Kapitel 3.2.5 zur Anwendung kommt werden die quadratischen Reste mithilfe von Primitivwurzeln bestimmt.

Beispiel 2.15. Betrachte $\mathbb{Z}_{77} \cong \mathbb{Z}_7 \times \mathbb{Z}_{11}$. Die Menge der Quadrate in $\mathbb{Z}_{77}$ lautet:

$$Q_{\mathbb{Z}_{77}} = \{\overline{1}, \overline{4}, \overline{9}, \overline{16}, \overline{25}, \overline{36}, \overline{64}\}.$$

Um herauszufinden, ob es sich wirklich um die quadratischen Reste handelt, werden nun jeweils eine Primitivwurzel von $\mathbb{F}_7^\times$ und $\mathbb{F}_{11}^\times$ bestimmt, und somit die quadratischen Reste bestimmt. Eine Primitivwurzel von $\mathbb{F}_7$ ist $\overline{3}$, denn:

$$\{\overline{3}^0 = \overline{1}, \overline{3}^1 = \overline{3}, \overline{3}^2 = \overline{3}, \overline{3}^3 = \overline{6}, \overline{3}^4 = \overline{4}, \overline{3}^5 = \overline{5}\} = \mathbb{F}_7^\times$$

Die Menge der quadratischen Reste in $\mathbb{Z}_7$ lautet daher:

$$Q_{\mathbb{Z}_7} = \{\overline{3}^0 = \overline{1}, \overline{3}^2 = \overline{2}, \overline{3}^4 = \overline{4}\}$$

Eine Primitivwurzel aus $\mathbb{F}_{11}^\times$ ist $\overline{2}$, denn

$$\{\overline{2}^0 = \overline{1}, \overline{2}^1 = \overline{2}, \overline{2}^2 = \overline{4}, \overline{2}^3 = \overline{8}, \overline{2}^4 = \overline{5}, \overline{2}^5 = \overline{10}, \overline{2}^6 = \overline{9}, \overline{2}^7 = \overline{7}, \overline{2}^8 = \overline{3}, \overline{2}^9 = \overline{6}\} = \mathbb{F}_{11}^\times.$$

Die quadratischen Reste von $\mathbb{Z}_{11}$ lauten daher:

$$Q_{\mathbb{Z}_{11}} = \{\overline{2}^0 = \overline{1}, \overline{2}^2 = \overline{4}, \overline{2}^4 = \overline{5}, \overline{2}^6 = \overline{9}, \overline{2}^8 = \overline{3}\}.$$

Nun wird über (2.12) überprüft, ob die Elemente aus $Q_{\mathbb{Z}_{77}}$ jeweils in $Q_{\mathbb{Z}_7}$ und $Q_{\mathbb{Z}_{11}}$ enthalten sind, was der Fall ist.

2.2. Stochastik und Statistik

2.2.1. Zufall und Pseudozufall

Was ist der Unterschied zwischen einer Zufallszahl und einer Pseudozufallszahl?

Mithilfe der Kriterien von [Hen17, S. 1] wird erstmal klar gemacht, wie Zufallszahlen bei einem idealen Zufallsexperiment entstehen:

- unter Versuchsbedingungen, die genau festgelegt sind
- es ist bekannt, welche möglichen Zufallszahlen entstehen können
- der Vorgang der Erzeugung kann unter den gleichen Bedingungen beliebig oft wiederholt werden.

Zu den idealen Zufallsexperimenten zählen unter anderem der Würfelwurf und der Münzwurf. Auch quantenmechanische Vorgänge sind zufällig.

Was Pseudozufallszahlen von echten Zufallszahlen unterscheidet, ist, dass sie mithilfe eines Algorithmus erzeugt werden können, so dass sie zufällig scheinen, wenn man

als Beobachter diesen nicht kennt. Wiederholt man diesen den Vorgang mit den gleichen Parametern und Startwerten, so ist erkennbar, dass diese Zufallsreihe noch einmal wiedergegeben wird, was darauf schließen lässt, dass die Zufallszahlen sich nur dann verändern, wenn (vgl. [Gen02, S. 2 f.]) der Startwert sich verändert.

Die (Pseudo-)Zufallszahlen als Ergebnisse $\omega_1, ..., \omega_n$ werden als Ergebnismenge Ω zusammengefasst.

Definition 2.16. *Für Ω als Ergebnismenge heißt eine Abbildung*

$$X : \Omega \to \mathbb{R}$$

Zufallsvariable auf Ω.

Jedem Ergebnis ω wird eine Zufallsvariable $X(\omega)$ zugeordnet. Die Zufallsvariable kann man auch als Abbildungsvorschrift für den Startwert eines Generators deuten. Das Ereignis, dass $X = k$ (k bezeichnet dahingehend einen Folgenwert, der die Pseudozufallsfolge fortsetzen)wird, kann man auch bezeichnen als

$$\{X = k\} := \{\omega \in \Omega : X(\omega) = k\}. \tag{2.14}$$

Man unterscheidet zwischen diskreten und stetigen Variablen [Eck14, S. 217].

Definition 2.17. *Eine Zufallsvariable heißt diskret, wenn sie für $k = 1, 2, ..., n$ oder $k = 1, 2, ...$ k viele Werte x_k annehmen kann, und ihre Verteilung mithilfe der Einzelwahrscheinlichkeit*

$$\mathbb{P}(X = x_k) = p_k \tag{2.15}$$

für ein $a \in \mathbb{R}$ so dargestellt werden kann:

$$\mathbb{P}(X \leq a) = \sum_k \mathbb{P}(X = k). \tag{2.16}$$

Definition 2.18. *Sei Y eine Zufallsvariable. Sie heißt stetig, wenn sie in einem Intervall $(-\infty, a)$, $a, b \in \mathbb{R}$ jeden Wert annehmen kann und ihre Verteilungsfunktion mithilfe einer Dichtefunktion dargestellt werden kann für:*

$$F_X(k) = \mathbb{P}(X \leq k) = \int_{-\infty}^{a} f_X(t)\, dt \tag{2.17}$$

2.2.2. Wahrscheinlichkeitsverteilung

Definition 2.19. *Seien $A, B \subset \Omega$. Dann heißt die reelle Funktion $\mathbb{P}$ Wahrscheinlichkeitkeitsverteilung auf Ω. Sie besitzt folgende Eigenschaften [Hen17, S. 37]:*

- *Nichtnegativität:* $\mathbb{P}(A) \geq 0$ *für* $A \subset \Omega$
- *Normiertheit:* $\mathbb{P}(\Omega) = 1$
- *Additivität:* $\mathbb{P}(A + B) = \mathbb{P}(A) + \mathbb{P}(B)$, *wenn* $A \cap B = \emptyset$

Es existieren diverse Wahrscheinlichkeitsverteilungen für Zufallszahlen. Die wichtigsten werden im Folgenden beschrieben. Dabei unterscheidet man zwischen diskreten und stetigen Wahrscheinlichkeiten [Hen17], [Eck14, S. 239 ff.].

Diskrete Verteilungen Diskrete Verteilungen mit ihren Verteilungsparametern gelten in der Regel für diskrete Zufallsvariablen X. Die bekanntesten sind die Gleichverteilung, die Binomialverteilung, die hypergeometrische Verteilung sowie die Poissonverteilung.

Gleichverteilung Sei der Ergebnisraum eine n-elementige Menge $\Omega = \{\omega_1, ..., \omega_n\}$. Dann gilt für die Einzelwahrscheinlichkeit eines Ereignisses $\omega \in \Omega$:

$$p(\omega) = \frac{1}{|\Omega|} = \frac{1}{n} \tag{2.18}$$

und durch die Addition der Einzelwahrscheinlichkeiten $A \subset \Omega$:

$$\mathbb{P}(A) = \frac{|A|}{|\Omega|} = \frac{|A|}{n}. \tag{2.19}$$

Somit heißt $\mathbb{P}$ gleichverteilt auf Ω. Ein Experiment, bei dem die Ergebnisse und diskrete Zufallsvariablen $X = k$ gleichverteilt sind, heißt auch Laplace-Experiment.

Binomialverteilung Zunächst wird der Begriff des Binomialkoeffizienten geklärt:

Definition 2.20. *Ein Binomialkoeffizient hat die Form*

$$\binom{n}{k} := \frac{n!}{k!(n-k)!} \tag{2.20}$$

mit $n, k \in \mathbb{N}_0$ *und* $n \geq k$

Nun betrachte man die diskrete Zufallsvariable X mit ihren Parametern $n, p \in \mathbb{N}_0$. Wenn ihre Wahrscheinlichkeit für $k = 0, 1, ...n$ durch die Formel

$$\mathbb{P}(X = k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{2.21}$$

beschrieben wird, nennt man X binomialverteilt.

Hypergeometrische Verteilung Betrachtet man X mit den Parametern $N, M, n \in \mathbb{N}$ für die möglichen Zufallswerte $k = 0, ..., n,\ k \leq M,\ n - k \leq N - M$, ist X hypergeometrisch verteilt, wenn $\mathbb{P}$ die Darstellung

$$\mathbb{P}(X = k) = \frac{\binom{M}{k} \cdot \binom{N-M}{n-k}}{\binom{N}{n}} \tag{2.22}$$

besitzt.

Poisson-Verteilung Man betrachte X mit $\lambda > 0$ als Parameter. Für $k \in \mathbb{N}_0$ gilt X mit der Darstellung der Wahrscheinlichkeitsverteilung

$$\mathbb{P}(X = k) = \frac{\lambda^k}{k!} \cdot e^{-\lambda} \tag{2.23}$$

als poissonverteilt.

Stetige Verteilungen Stetige Verteilungen mit ihren jeweiligen Parametern gelten hingegen für stetige Zufallsvariablen Y. Die wichtigsten sind hier die Normalverteilung und die Exponentialverteilung.

Normalverteilung Man betrachte Y mit den Parametern $\mu \in \mathbb{R},\ \sigma > 0$. Wenn die Verteilung von Y mit $a \in \mathbb{R}$ mit der Dichtefunktion

$$f_Y(a) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} e^{-\frac{(a-\mu)^2}{2 \cdot \sigma}} \tag{2.24}$$

dargestellt werden kann, nennt man Y normalverteilt. Ihre Verteilungsfunktion lautet

$$\mathbb{Y} \leq \curvearrowright = \int_{-\infty}^{a} f_Y(t) \mathrm{d}t. \tag{2.25}$$

Für $\mu = 0,\ \sigma = 1$ ist die standardisierte Zufallsgröße $Z = \frac{X - \mathbb{E}(X)}{\sqrt{Var(X)}} = \frac{X-\mu}{\sigma}$ standardnormalverteilt mit der Dichtefunktion

$$\varphi(z) = \frac{1}{\sqrt{2 \cdot \pi}} e^{-\frac{z^2}{2}}. \tag{2.26}$$

Die Werte der Verteilungsfunktion $\phi(z)$ sind tabellarisch festgehalten.

Exponentialverteilung Für einen Parameter $\lambda > 0$ ist Y exponentialverteilt, wenn ihre Verteilung durch

$$f_X(a) = \lambda \cdot e^{-\lambda \cdot a} \tag{2.27}$$

für alle $a \geq 0$ als Dichtefunktion gegeben ist. Ihre Verteilungsfunktion lautet:

$$\mathbb{P}(X \leq a) = \begin{cases} 0 & a < 0 \\ 1 - e^{-\lambda \cdot a} & a \geq 0 \end{cases}. \tag{2.28}$$

Gleichverteilung Eine Zufallsvariable Y ist auf dem Intervall (a, b) stetig gleichverteilt, wenn die Dichtefunktion für $a < y < b$ die Form

$$f(y) = \frac{1}{b-a} \tag{2.29}$$

annimmt. Die Verteilungsfunktion hat die Form

$$F(x) = \begin{cases} 0 & y \leq a \\ \frac{y-a}{b-a} & a < y < b \\ 1 & y \geq b \end{cases}. \tag{2.30}$$

2.2.3. Gleichverteilung modulo Eins und der Satz von Weyl

In Kapitel 3.3 werden Zufallsgeneratoren verwendet, deren generierten Zufallszahlen gleichverteilt modulo Eins sind. Doch was bedeutet das eigentlich?

Definition 2.21. *Für eine Folge reeller Zahlen $(x_i)_{i \in \mathbb{N}}$ ist für alle i $y_i = x_i - [x_i]$ eine Bruchteilfolge. $[x_i]$ ist hierbei die größte ganze Zahl, die kleiner als x_i ist.*

Aus der Definition folgt, dass $y_i \in [0, 1)$. Die Bruchteilfolgenglieder gelten als Zufallsfolge, wenn sie gleichverteilt modulo Eins sind [JRBT09, S. 114].

Definition 2.22. *Eine Folge $(x_i)_{i \in \mathbb{N}}$ gilt als gleichverteilt modulo Eins, wenn für eine Anzahl an Bruchteilfolgenglieder y_i in einem Intervall $[a, b)$ der Grenzwert*

$$\lim_{N \to \infty} \frac{n_y = \#\{y_n \text{ in } [a,b)\}_{n \leq N}}{N} = b - a \tag{2.31}$$

ist.

Die Definition wurde von Hermann Weyl hergeleitet [Wey16, S. 313 f.]. Er ging davon aus, dass eine Gerade, die mit unendlich vielen Punkten $y_1, y_2, y_3, ...$markiert ist, zu einem Kreis mit dem Umfang Eins aufgerollt wird. Er stellte sich die Frage, ob die

Markierungen an Stellen y_n gleichmäßig verteilt sind. Dies würde zutreffen, wenn die Anzahl n_y unter den Markierungen $y_1, ..., y_n$ im Kreisteilabschnitt der Länge $a - b$ der Gleichung aus Definition 2.22 entsprechen. Er legte folgendes für Folgen gleichverteilt modulo Eins fest:

Satz 2.23. *Sei die Bruchteilfolge $(y_i)_{i\in[1,n]}$ gleichverteilt modulo eins. Dann gilt für jede riemannintegrierbare Funktion $f(x)$, die mit der Periode 1 periodisch ist*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) = \int_0^1 f(x)dx. \tag{2.32}$$

Beweis. Aus der Definition 2.22 folgt, dass die Gleichung (2.32) für jede stückweise stetige Funktion für verschiedene Teilintervalle in [0;1), wobei das prominenteste Beispiel eine Treppenfunktion ist, gilt. Daher gibt es zu der Funktion f zwei stückweise stetige Funktionen t_1, t_2, so dass gilt:

$$t_1 \leq f \leq t_2 \quad \text{sowie} \quad \int_0^1 t_2 \, \mathrm{d}x - \int_0^1 t_1 \, \mathrm{d}x = \varepsilon.$$

Betrachtet man zunächst t_1, so gilt:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} t_1(y_i) = \int_0^1 t_1 \, \mathrm{d}x \geq \int_0^1 f \, \mathrm{d}x - \varepsilon.$$

Wählt man n ausreichend groß, so folgt

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} t_1(y_i) > \int_0^1 f \, \mathrm{d}x - 2\varepsilon,$$

also auch

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) > \int_0^1 f \, \mathrm{d}x - 2\varepsilon,$$

da

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) > \frac{1}{n} \sum_{i=1}^{n} t_1(b_1)$$

Aus der Betrachtung mit t_2 folgt analog:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) < \int_0^1 f \, \mathrm{d}x - 2\varepsilon,$$

so dass

$$\int_0^1 f \, \mathrm{d}x - 2\varepsilon < \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) < \int_0^1 f \, \mathrm{d}x - 2\varepsilon.$$

Durch geeignete Wahl von t_1 und t_2 wird ε so klein, dass $\varepsilon \to 0$ und

$$\int_0^1 f \, \mathrm{d}x < \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(y_i) < \int_0^1 f \, \mathrm{d}x,$$

also auch die Behauptung folgt. □

Für das aus diesem Satz folgende Weylsche Kriterium benutze man die komplexe Exponentialfunktion

$$e(x) := e^{2\pi i x}$$

Setzt man $x = my_i,\ m \in \mathbb{Z} \setminus 0$, erhält man nach Satz2.23

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} e(my_i) = \int_0^1 e(mx)\mathrm{d}x = e(m1) - e(m0) = 1 - 1 = 0$$

Für $m = 0$ gilt $e^{0x} = 1$ und somit gilt Gleichung (2.32) ebenfalls.
Aus der Lehre der Fourierreihen, dass eine periodische Funktion $f(x)$ eine lineare Zusammensetzung von beliebig vielen Exponentialfunktionen ist, lässt sich folgender Satz formulieren:

Satz 2.24 (Weylsches Kriterium)**.** *Wenn für jedes $m \in \mathbb{Z} \setminus 0$ gilt:*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} e(my_i) = 0, \tag{2.33}$$

so sind die Zahlen y_i gleichverteilt modulo eins.

Beweis. Der Beweis lässt sich unter [Wey16, S. 315] nachlesen. □

2.2.4. Kenngrößen in der Wahrscheinlichkeitsrechnung

Mithilfe einer Wahrscheinlichkeitsverteilung lassen sich vier verschiedene Kenngrößen berechnen. Sie werden im Folgenden allgemein vorgestellt [Hen17, S. 77 ff].

Erwartungswert Für eine diskret verteilte Zufallsvariable X ist ihr Erwartungswert

$$\mathbb{E}(X) := \sum_{\omega\in\Omega} X(\omega)\mathbb{P}(\{\omega\}). \tag{2.34}$$

Für stetige Zufallsvariablen Y hingegen lässt sich der Erwartungswert folgendermaßen berechnen:

$$\mathbb{E}(Y) := \int_{-\infty}^{\infty} yf(y)\mathrm{d}y. \tag{2.35}$$

Varianz und Standardabweichung Die Varianz für eine diskret verteilte Zufallsvariable X lautet

$$Var(X) := \mathbb{E}(X - \mathbb{E}X)^2. \tag{2.36}$$

Die Standardabweichung ist im Allgemeinen die Wurzel aus der Varianz:

$$\sigma(X) := \sqrt{Var(X)}. \tag{2.37}$$

Für stetige Zufallsvariablen lautet die Varianz

$$Var(X) := \int_{-\infty}^{\infty} (y - \mathbb{E}Y)^2 f(y) \mathrm{d}y. \tag{2.38}$$

Kovarianz Für zwei gemeinsam diskret verteilte Zufallsvariablen X_1, X_2 ist die Kovarianz

$$Cov(X_1, X_2) := \mathbb{E}((X_1 - \mathbb{E}(X_1))(X_2 - \mathbb{E}X_2)) \tag{2.39}$$

Korrelation Der pearsonsche Korrelationskoeffizient von X_1, X_2 lässt sich wie folgt berechnen:

$$r(X_1, X_2) := \frac{Cov(X_1, X_2)}{\sqrt{Var(X_1)Var(X_2)}}. \tag{2.40}$$

2.2.5. Stichproben

Bei statistischen Erhebungen [Hen17, S. 20 ff.] werden Merkmalsausprägungen an Versuchseinheiten festgestellt. Es wird vor allem zwischen qualitativen und - für die Erfassung von Zufallszahlen interessanter - quantitativen Merkmalen unterschieden. Die Menge der Versuchseinheiten über mindestens ein Merkmal wird Grundgesamtheit genannt. Eine zufällig erhobene Teilmenge aus dieser Grundgesamtheit heißt Stichprobe oder auf englisch sample. Ihr Umfang ist n, d.h. sie hat n Elemente.

Für s mögliche Merkmalsausprägungen $a_1, ..., a_s$ in einer Stichprobe $x_1, ..., x_n$ werden s absolute Häufigkeiten gebildet. Ihre Bildungsvorschrift lautet [Hen17, S. 22]:

$$h_j = \sum_{i=1}^{n} \mathbf{1}\{x_i = a_j\} \qquad \text{mit } j = 1, ..., s, \quad h_1 + ... + h_j = n. \tag{2.41}$$

Die relative Häufigkeit ergibt sich aus

$$r_j = \frac{h_j}{n} \qquad \text{mit } j = 1, ..., s, \quad r_1 + ... + r_j = n. \tag{2.42}$$

Die statistischen Kennwerte [Hen17, S. 27 ff.], [Eck14, S. 97] sind ähnlich zu denen, die in Kapitel 2.2.4 für gleichverteilte (Pseudo-)Zufallszahlen auftreten:

- das arithmetische Mittel:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.43}$$

  oder das gewichtete Mittel der Merkmalsausprägungen (entspricht dem Erwartungswert):

$$\overline{x} = \sum_{j=1}^{s} g_j \cdot a_j \qquad g_i = \frac{h_j}{n} \text{ mit } j = 1, ...s \tag{2.44}$$

- die empirische Varianz

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2 \tag{2.45}$$

- die darausfolgende empirische Standardabweichung

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2} \tag{2.46}$$

- die Kovarianz für zwei Stichproben zweier unterschiedlicher Merkmalsausprägungen

$$d_{XY} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \tag{2.47}$$

- der Maßkorrelationskoeffizient nach Bravais und Pearson

$$r_{XY} = \frac{d_{XY}}{s_X \cdot s_Y} = \frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2} \cdot \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \overline{y})^2}} \tag{2.48}$$

3. Arten von Zufallsgeneratoren

3.1. Quadrat-Mitten-Generator

Der Quadrat-Mitten-Generator von John von Neumann (1903-1957) ist nach [JRBT09, S. 109 f.] einer der ersten Zufallsgeneratoren. Sein Prinzip beruht darauf, dass aus einem maximal achtstelligem Quadrat einer vierstelligen ganzen Startzahl x_0 die dritte bis sechste Zahl, beginnend beim Einer, ausgewählt werden. Diese Auswahl beschreibt die neue Zufallszahl x_1.

Für bestimmte Startwerte wird dieser Algorithmus periodisch, also wenn die Zufallszahl x_n gleich die Startzahl x_0 **oder** eine der anderen Zufallzahlen x_i für $i = \{1, ..., n-1\}$ ist, wiederholt sich die Abfolge der Zufallszahl, was bereits nach wenigen Iterationsschritten passieren kann. Für andere Startwerte endet der Algorithmus bei einer Zufallzahl, die den Wert Null annimmt. Die Periodenlängen sind zumeist kleiner als 110, was für einen Zufallsgenerator eher schlecht ist [JRBT09, ebd.]. Daher ist dieser nur von historischer Bedeutung.

3.2. Zufallsgeneratoren basierend auf der Restklassenarithmethik

3.2.1. Additiver Kongruenzgenerator

Die Bildungsvorschrift für Pseudozufallszahlen nach dem additiven Kongruenzgeneratoren lautet nach [JRBT09, S. 112]:

$$x_{n+2} = (x_{n+1} + x_n) \bmod m \tag{3.1}$$

Hierbei seien x_0 und x_1 die Startwerte dieses Generators und m der Modulparameter. Für den Fibonacci-Generator, welcher die Spezialform des additiven Kongruenzgenerators ist, gilt für die Startwerte $x_0 = x_1 = 1$ [JRBT09, ebd.]. Eine Periode, die durch den additiven Kongruenzgenerator erzeugt wird, endet dann genau dann, wenn die beiden Startwerte sowie der erste Zufallswert erzeugt werden. Dadurch wird die Periodenlänge größer als m selbst, was aber auch nach sich zieht, dass einige Zufallszahlen häufiger als andere auftreten, wie sich an folgendem Beispiel zeigt:

Beispiel 3.1. Es werden die Fibonaccizufallszahlen für $m = 3$ und $x_0 = x_1 = 1$ berechnet:

Iteration	x_n	x_{n+1}	x_{n+2}
1	1	1	2
2	1	2	0
3	2	0	2
4	0	2	2

Iteration	x_n	x_{n+1}	x_{n+2}
5	2	2	1
6	2	1	0
7	1	0	1
8	0	1	1
9	1	1	2

Die Periodendauer beträgt 8, was größer ist als m selbst.

3.2.2. Multiplikativer Kongruenzgenerator

Der multiplikative Kongruenzgenerator benötigt hingegen nur einen Startwert x_i, um eine Zufallszahl x_{i+1} zu generieren. Die Berechnungsvorschrift hierfür lautet:

$$x_{n+1} = (a \cdot x_n) \bmod m \tag{3.2}$$

Wie beim additiven Restklassengenerator sei $m \in \mathbb{N}$ der Modulparameter und $a \in \mathbb{Z}_m$ ein linearer Parameter.

Der multiplikative Kongruenzgenerator ist ein Sonderfall von Gleichung (3.3) mit $b = 0$. Das heißt, dass die Periodenlänge maximal nur $m - 1$ betragen kann, das heißt, dass es einen Wert gibt, der für eine Periodenlänge m nicht auftauchen kann, und zwar $x_i = 0$ [Knu97, S. 19]. Es gilt folgender Satz für die Periodenlänge eines multiplikativen Kongruenzgenerators [Knu97, S. 20]:

Satz 3.2. *Die maximale Periodenlänge eines multiplikativen Kongruenzgenerators beträgt $\lambda(m)$, wobei es sich um $\lambda(m)$ um die Carmichael-Funktion handelt, wenn*

1. *x_0 ist teilerfremd zu m*
2. *a ist eine Primitivwurzel modulo m*

Beispiel 3.3. Sei $m = 7$. Dann ist der Startwert $x_0 \in \{1, 2, 3, 4, 5, 6\}$ und der lineare Parameter $a \in \{3, 5\}$. Daraus folgt die maximale Periodenlänge $\lambda(7) = 6$.

3.2.3. Lehmer-Generator

Der Lehmer-Generator ist ein linearer Kongruenzgenerator. Er geht aus einer Kombination aus dem additiven und dem multiplikativen Kongruenzgenerator hervor. Dessen Vorschrift zur Bildung von Zufallszahlen lautet daher:

$$x_{n+1} = (a \cdot x_n + b) \bmod m \tag{3.3}$$

Hierbei seien a und m wie beim multiplikativen Kongruenzgenerator gegeben. Dazu kommt nun ein additiver Parameter $b \in \mathbb{Z}_m$. Eingeführt wurde der Lehmer-Generator, wie der Name schon sagt, von Derrick Lehmer 1949, vgl. [Leh51, S.145].

Bei der Programmierung eines Lehmer-Generators möchte man, dass die Periodenlänge desselben maximal, also m, wird. Trivialerweise erreicht man das bei beliebigen m und $a = b = 1$. Dabei tritt das Problem auf, dass die erzeugten Zufallszahlen $x_1, ..., x_n$ alles andere als zufällig erscheinen. Ein kurzes Beispiel hierfür ist folgendes:

Beispiel 3.4. Seien $x_0 = 2$, $m = 3$ und $a = b = 1$. Die Bildungsvorschrift lautet daher $x_{n+1} = (x_n + 1)$ *mod* 3. Für x_i mit $0 \leq i \leq 5$ sehen die erzeugten Zahlen so aus:

$$x_0 = 2$$
$$x_1 = 0$$
$$x_2 = 1$$
$$x_3 = 2$$
$$x_4 = 0$$
$$x_5 = 1.$$

Daher stellt sich die Frage: Wie müssen die Parameter ausgewählt sein, damit die erzeugten Zahlen sowohl zufälliger erscheinen als auch eine maximale Periodenlänge aufweisen? Hierfür wurde eine Lösung gefunden:

Satz 3.5. *Der Lehmer-Generator mit den Parametern a, b, m und dem Startwert x_0 hat genau dann die maximale Periodenlänge m, wenn folgende Bedingungen erfüllt sind:*

1. *b ist teilerfremd zu m*
2. *$(a-1)$ wird von jedem Primfaktor p von m geteilt*
3. *$(a-1)$ wird von 4 geteilt, wenn m ebenfalls von 4 geteilt wird*

Beweis. Ein Beweis findet sich in [Knu97, S. 17 f.]. □

Ein kurzes Beispiel für diesen Satz sähe wie folgt aus:

Beispiel 3.6. Sei $m = 8$. Dann besteht 8 aus dem einzigen Primfaktor $p = 2$. Der geeignete Lehmer-Generator für dieses m lautet daher:

$$x_{n+1} = (5x_n + 7) \; mod \; 8. \tag{3.4}$$

Für den Startwert $x_0 = 3$ ergibt sich folgende Zufallszahlenfolge:

$$x_0 = 3$$
$$x_1 = 6$$
$$x_2 = 5$$
$$x_3 = 0$$
$$x_4 = 7$$
$$x_5 = 2$$
$$x_6 = 1$$
$$x_7 = 4$$

mit der maximalen Periode von $m = 8$.

3.2.4. Inverser Kongruenzgenerator

Der inverse Kongruenzgenerator arbeitet mit dem multiplikativen Inversen x_0^{-1} einer Startzahl x_0, um an die Zufallszahlen x_i zu kommen. Hierfür sei der Modulparameter eine Primzahl $p \in \mathbb{N}$ und der additive sowie der lineare Parameter wie oben gegeben. Dann lautet die Iterationsvorschrift wie folgt [EL86, S. 316]:

$$x_{n+1} = \begin{cases} a \cdot x_n^{-1} + b \bmod p & x_n \geq 1 \\ b & x_n = 0 \end{cases} \tag{3.5}$$

Die Periodenlänge des Generators ist p selbst, wenn die Bedingungen aus Satz 3.5 eingehalten werden. Durch das zusätzliche Invertieren der neu generierten Zufallszahl ist es recht aufwändig, die Zufallszahlen zu generieren, was ein Nachteil bezüglich der Geschwindigkeit des Generators bringt, denn dieser braucht für das Ermitteln einer Pseudozufallszahl $12 \cdot \ln(\frac{2}{\pi^2}) \cdot \mathrm{ln}(p) + 1$-mal länger als der Lehmer-Generator. [EL86, S. 317].

3.2.5. Blum-Blum-Shub-Generator

Der Blum-Blum-Shub-Generator [BBS86] ist ein Zufallsbitgenerator und daher von kryptologischer Bedeutung. Die Pseudozufallszahlen, die in diesem Fall nur 0 oder 1 sein können, werden mithilfe von quadratischen Resten erzeugt. Sei $\mathbb{Z}_m$ ein Restklassenkörper mit $m = p \cdot q$. Dafür gilt, dass $p,\ q \in \mathbb{N}$ Primzahlen mit $p \equiv q \equiv 3 \bmod 4$ sind. Wie man sicherstellen kann, dass alle quadratischen Reste in $\mathbb{Z}_m$ enthalten sind, wurde bereits in Beispiel 2.15 gezeigt. Darüber hinaus sind in $\mathbb{Z}_m$ nur die Elemente drin, die teilerfremd zu p und q sind.

Die Menge der quadratischen Reste $Q_{\mathbb{Z}_m}$ ist gleichzeitig die Menge der Startwerte. Wird ein Element dieser Startwerte als x_0 ausgewählt, so wird dieses mit folgender Iterationsvorschrift verarbeitet:

$$x_{n+1} \equiv (x_n)^2 \bmod m. \tag{3.6}$$

Die Zufallsbits $b_0, ..., b_n$ werden mit folgender Iterationsvorschrift gebildet:

$$b_n \equiv x_n \bmod 2. \tag{3.7}$$

Die Periodenlängen $\pi(x_0)$ des Blum-Blum-Shub-Generators mit entsprechendem Startwert teilt $\lambda(\lambda(m))$ [BBS86, S. 377]. Dies zeigt ein kurzes Rechenbeispiel:

Beispiel 3.7. Sei $p = 7$ und $q = 11$. Dann ist $m = 77$. Aus Definition 2.12 folgt:

$$\lambda(\lambda(77)) = \lambda(6 \cdot 10) = \lambda(60) = \lambda(2^2 \cdot 3 \cdot 5) = 2 \cdot 2 \cdot 4 = 16$$

Die Periodenlängen $\pi(x_0)$ der quadratischen Reste von $\mathbb{Z}_{77}$ sind folgende:

$$\pi(1) = 1,\ \pi(4) = 4,\ \pi(9) = 4,\ \pi(16) = 4,\ \pi(25) = 4,\ \pi(36) = 2,\ \pi(64) = 2$$

Sie alle sind Teiler von 16.

3.2.6. Matrixkongruenzgenerator

Der Matrixkongruenzgenerator folgt denselben Prinzipien wie der lineare Kongruenzgenerator, nur, dass sie erzeugten Zufallszahlen diesmal nicht ein-, sondern mehrdimensional sind. Dies bringt den Vorteil, dass mehrere Zufallszahlen parallel generiert werden können. Die Berechnungsvorschrift für die Zufallsvektoren lautet daher:

$$\mathbf{x}_{n+1} \equiv (A \cdot \mathbf{x}_n + \mathbf{c}) \bmod m. \tag{3.8}$$

Hierbei sind $\mathbf{x}_i, \mathbf{c} \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times d}$ und $m \in \mathbb{N}$. Außerdem gilt, dass sowohl die Vektor- als auch die Matrixeinträge ganzzahlige Elemente zwischen 1 und $m - 1$ sind.
Die Wahl der Matrix sagt etwas über die Zufallszahlenfolge aus. Daher werden nun einige Matrizen vorgestellt, die für die Generatoren für $\mathbf{c} = 0$ verwendet werden [Gen02, S. 34 f.]:

- mit Matrizen der Form

$$A = \begin{pmatrix} a_1 & -1 & 0 & \dots & 0 \\ 0 & a_2 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & -1 \\ -1 & 0 & 0 & \dots & a_d \end{pmatrix}$$

  können schnell neue Zufallsvektoren berechnet werden

- Diagonalmatrizen erzeugen Zufallsvektoren, deren Einträge jeweils ein Vielfaches vom vorhergehenden Vektor sind

- für Matrizen der Form

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \dots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

  korrelieren die Vektoreinträge miteinander

- für $m = 2$ existieren Shift-Register-Generatoren [Edd90, S. 66] der Rekursion

$$\mathbf{x}_{n+1} = T\mathbf{x}_n.$$

  Dabei ist

$$T = (\mathbb{I} + L^s) \cdot (\mathbb{I} + R^t)$$

mit $\mathbb{I}$ als $d \times d$ Einheitsmatrix, $s, t < d$ und

$$L = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \text{ sowie } R = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}.$$

Für $s + t \geq d$ gilt für die daraus resultierende Matrix T und für die Berechnung der Zufallsbits folgendes:

- die Zufallsbits werden um t Bits nach rechts verschoben
- exklusives Oder
- die Zufallsbits werden um s Bit nach links verschoben
- für $L^s R^t = 0$ gilt ein exklusives Oder

3.2.7. Mersenne-Twister

Der Mersenne-Twister wurde von Makoto Matsumoto und Takuji Nishimura 1998 [MN98] entwickelt. Er heißt so, weil die Periodenlänge des Generators eine Mersenne-Primzahl ist. Doch was ist überhaupt eine Mersenne-Primzahl?

Definition 3.8. *Die Menge der Mersenne-Zahlen ist wie folgt definiert:*

$$M = \{2^n - 1 | n \in \mathbb{N}\}. \tag{3.9}$$

Die Primzahlen in dieser Menge heißen Mersenne-Primzahlen.

Der Mersenne-Twister arbeitet mit n Startvektoren $\mathbf{x}_k$ mit $k \in \{0, ..., n-1\}$ der Länge w. Die Einträge dieser Vektoren kommen aus dem Restklassenkörper $\mathbb{F}_2$. Somit stellen die Vektoren eine Zahl zwischen 0 und $2^w - 1$ in binärer Schreibweise dar. Demzufolge arbeitet der Generator mit Zufallsbits. Die Bildungsvorschrift der Zufallsvektoren lautet [MN98, S. 8]:

$$\mathbf{x}_{k+n} := \mathbf{x}_{k+m} \oplus (\mathbf{x}_k^u | \mathbf{x}_{k+1}^l) A \tag{3.10}$$

mit $k = 0, 1, ...$ Hierfür ist m eine ganze Zahl zwischen 1 und n. Des Weiteren existiert eine weitere ganze Zahl r zwischen 1 und $w - 1$ die folgenden Grund hat: $\mathbf{x}_k^u$ meint die Auswahl von den ersten $w - r$ Vektoreinträgen von $\mathbf{x}_k$, $\mathbf{x}_{k+1}^l$ hingegen die letzten r Einträge des Vektors $\mathbf{x}_{k+1}$. Aus dieser Auswahl folgt die Zusammensetzung eines neuen Vektors $\mathbf{x} = (\mathbf{x}_k^u | \mathbf{x}_{k+1}^l) = (x_{w-1}, x_{w_2}, ..., x_0)$. A ist eine $w \times w$-Matrix mit Einträgen aus $\mathbb{F}_2$. Matsumoto und Nishimura fanden eine Matrix für eine schnelle Vektor-Matrix-Multiplikation, welche die folgende Form hat [MN98, ebd.]:

$$A = \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ a_{w-1} & a_{w-2} & \cdots & \cdots & a_0 \end{pmatrix},$$

wobei die unterste Zeile als $\mathbf{a} = (a_{w-1}, a_{w-2}, ..., a_0)$ bezeichnet wird. Die Berechnung von $\mathbf{x}A$ kann somit abgekürzt werden [MN98, ebd.]:

$$\mathbf{x}A = \begin{cases} \mathbf{x} \text{ wird um 1 Bit nach rechts verschoben} & \text{für } x_0 = 0 \\ \mathbf{x} \text{ wird um 1 Bit nach rechts verschoben und bitweise mit } \mathbf{a} \text{ addiert} & \text{für } x_0 = 1 \end{cases}.$$

Nach der Vektor-Matrix-Multiplikation folgt die bitweise Addition mit $\mathbf{x}_{k+m}$, für die das $\oplus$ steht.

Die erzeugten Zufallszahlen sollen auch k-dimensional mit einer Genauigkeit von v Bit sein. Das heißt, die Verteilung von Punkten $(x_i, x_{i+1}, ..., x_{i+k-1}) \in [0,1]^k$ ist im Raum $[0,1]^k$ gleich dicht.

Um diese k-dimensionale Verteilung zu gewährleisten, wird jeder generierter Zufallsvektor $\mathbf{x}$ mit einer Matrix $T \in \mathrm{GL}(w, \mathbb{F}_2)$ von rechts multipliziert, so dass $\mathbf{z} = \mathbf{x}T$. Für diese Berechnungen werden folgende Transformationen benutzt [MN98, S. 9]:

$$\mathbf{y} := \mathbf{x} \oplus (\mathbf{x} \gg u) \tag{3.11}$$

$$\mathbf{y} := \mathbf{y} \oplus ((\mathbf{y} \ll s) \wedge \mathbf{b}) \tag{3.12}$$

$$\mathbf{y} := \mathbf{x} \oplus ((\mathbf{y} \ll t) \wedge \mathbf{c}) \tag{3.13}$$

$$\mathbf{z} := \mathbf{y} \oplus (\mathbf{y} \gg l) \tag{3.14}$$

Hierbei stehen $\wedge$ für die logische UND-Verknüpfung, $\ll$ für eine Verschiebung um j Bits nach links sowie $\gg$ für eine Verschiebung um j Bits nach rechts. Des Weiteren sind u, s, t, l ganzzahlige Werte sowie $\mathbf{b}, \mathbf{c} \in \mathbb{F}_2^w$ geeignete Bitmasken, um die Transformationen zu vollführen.

Der bekannteste Mersenne-Twister ist der MT19937 mit einer 623-dimensionalen Gleichverteilung auf einer Genauigkeit von 32 Bits. Die Parameter des MT19937 lauten wie folgt [MN98, S. 11]:

- $(w, n, m, r) = (32{,}624{,}397{,}31)$
- $a = 9908\mathrm{B0DF}_{16}$

- $u = 11$
- $s = 7$, $b = 9\text{D2C}5680_{16}$
- $t = 15$, $c = \text{EFC}60000_{16}$
- $l = 16$

Die Periodenlänge des MT19937 beträgt $2^{19937} - 1$, wobei es sich um eine Mersenne-Primzahl handelt. Er verarbeitet insgesamt 624 Startwerte mit einer Länge von 32 Bit und gibt auch genauso viele Zufallsvektoren wieder aus, weshalb er für Monte-Carlo-Simulationen überaus geeignet ist. Dieser Prozess dauert 10,28 Sekunden, wodurch er sich auch durch seine Schnelligkeit auszeichnet. Jedoch benötigt er für den Prozess einen großen Arbeitsspeicher von $624 \cdot 32\text{Bit} = 2496$ Bit. Des Weiteren ist er auch nicht für kryptographische Anwendungen geeignet, sondern eher dafür gedacht, um gleichverteilte Zufallszahlen im Intervall [0,1] zu erzeugen. Dafür müssen die Zahlen zwischen 0 und $2^{32} - 1$ einfach nur durch $2^{32} - 1$ geteilt werden [MN98, S. 5].

3.3. Zufallsgeneratoren basierend auf dem Satz von Weyl

Basierend auf dem Satz von Weyl gibt es unter anderem folgende Zahlenfolgen, die gleichverteilt modulo Eins sind [JRBT09, S. 114 f.]:

- $(\mathrm{n} \cdot \alpha)_{\mathrm{n} \geq 1}$ mit irrationalem α
- $(\mathrm{n}^{\alpha} \cdot \log_{\beta}(\mathrm{n}))_{\mathrm{n} \geq 1}$ mit $0 < \alpha < 1$ und β als beliebige reelle Zahl
- $(\mathrm{P}(\mathrm{n}))_{\mathrm{n} \geq 1}$ mit $\mathrm{P}(\mathrm{n})$ als Polynom mit Grad $g \geq 1$ und mindestens einem irrationalen Koeffizienten, wodurch das erste Beispiel mit in diesem eingeschlossen ist.

Mithilfe dieser Zahlenfolgen können Zufallsgeneratoren implementiert werden, dessen Pseudozufallszahlen gleichverteilt modulo Eins sind. Für die erzeugten Zufallszahlen gilt, dass diese dicht im Intervall $[0, 1)$ sind [JRBT09, ebd.]. Somit können unendlich viele Zufallszahlen generiert werden, die gleichverteilt modulo Eins sind.

Simulierte Verteilungen können in Kapitel 5 betrachtet werden. Auch da ist erkennbar, dass keine erzeugte Zufallszahl doppelt vorkommt, sondern wenn, dann sehr dicht aneinander gereiht sind.

3.4. Algorithmus K

Der Algorithmus K wurde 1959 von Donald Knuth [Knu97, S. 5 f.] entwickelt. Dafür braucht man als Eingabe eine zehnstellige Zahl X. Im Grundprinzip unterliegt die Zahl X einer Reihe von Transformationsalgorithmen, welche unterschiedlich funktionieren. Im Folgenden wird der Algorithmus beschrieben:

Algorithmus 3.9 (Algorithmus K). *Gib eine 10-stellige Zahl ein und befolge folgende Anweisungen:*

*K*1. *[Wähle Anzahl der Iterationen.] Wähle die milliardste Stelle von X als Y. Führe die Iteration $Y+1$-mal aus.*

*K*2. *[Wähle einen zufälligen Schritt aus.] Wähle die hundertmillionste Stelle von X als Z. Gehe zu Schritt $K(3+Z)$ und führe diese Schritte bis $K13$ aus.*

*K*3. *[Überprüfe, ob $X \geq 5 \cdot 10^9$.] Wenn $X < 5 \cdot 10^9$, setze $X = X + 5 \cdot 10^9$.*

*K*4. *[Quadrat-Mitten-Methode.] Quadriere X und wähle davon die mittleren zehn Stellen aus. Dies ist das neue X.*

*K*5. *[Multiplikation.] Multipliziere X mit* 1001001001 *und wähle davon die letzten zehn Stellen aus. Dies ist das neue X.*

*K*6. *[Pseudo-Komplement.] Wenn $X < 10^8$, setze $X = X + 9814055677$, sonst setze $X = 10^{10} - X$.*

*K*7. *[Hälften vertauschen] Vertausche die ersten fünf Stellen mit den letzten fünf Stellen von X.*

*K*8. *[Multiplikation.] Führe denselben Schritt wie in $K5$ aus.*

*K*9. *[Betrag verringern.] Verringere jeden Betrag jeder Stelle von X, die nicht null ist, mit eins.*

*K*10. *[Änderung mit 99999.] Wenn $X < 10^5$, setze $X = X^2 + 99999$, sonst setze $X = X - 99999$.*

*K*11. *[Normierung.] Wenn $X < 10^9$, setze $X = 10 \cdot X$ und wiederhole diesen Schritt.*

*K*12. *[Andere Quadrat-Mitten-Methode.] Multipliziere X mit $X-1$ und wähle davon die mittleren 10 Stellen. Das ist das neue X.*

*K*13. *[Wiederholung?] Wenn $Y > 0$, verringere Y um eins und beginne wieder bei $K2$. Wenn $Y = 0$, endet der Algorithmus mit der generierten Zufallszahl X.*

Sehr häufig kommt es vor, dass dieser Algorithmus gegen $X = 6065038420$ konvergiert, was nicht gerade für einen guten Zufallsgenerator spricht, zumal der Aufwand, um eine neue Zufallszahl zu finden, für eine kleine Periode recht groß ist. Selbst Knuth ist der Meinung, dass Zufallszahlen nicht mit einem zufällig gewählten Algorithmus wie diesem generiert werden sollten, sondern etwas mehr Theorie notwendig ist [ebd, S. 6], um einen guten Zufallsgenerator zu konstruieren.

4. Tests zur Überprüfung der Güte von Pseudozufallszahlen

4.1. Chi-Quadrat-Test

Der Chi-Quadrat-Test überprüft mittels einer Nullhypothese $H_0 : F_X = F_X^e$, ob die Verteilung der generierten Zufallszahlen F_X der erwarteten Verteilungsfunktion F_X^e entsprechen. Dahingehend wird diese gegen eine Alternativhypothese $H_1 : F_X \neq F_X^e$ getestet [Eck14, S. 314]. Bei den Kongruenzgeneratoren sowie bei den Generatoren, die nach dem Satz von Weyl aufgebaut sind, wird demzufolge auf Gleichverteilung getestet.

Beim Chi-Quadrat-Test geht man folgendermaßen vor (Notation nach [Knu97, S. 42 ff.]): Zunächst unterteilt man die Zufallszahlen in s Kategorien, wobei $s \in 1, ..., k$. Diese Kategorien treten jeweils mit einer Wahrscheinlichkeit p_s auf. Führt man ein Experiment mit $n \in \mathbb{N}$ unabhängig durchgeführten Wiederholungen durch, berechnet man daraus die zu erwartenden Häufigkeiten $n_s = n \cdot p_s$ Hierbei sei zu beachten, dass $n_s \geq 5$, wenn $k \leq 8$, ansonsten $n_s \geq 1$ gilt [Eck14, S. 314]. Dies setzt auch ein eine ausreichend große Anzahl an Wiederholungen voraus. Ihnen gegenüber stehen die beobachteten Häufigkeiten der jeweiligen Zufallszahlen $Y_s \in \mathbb{N}$. Mithilfe dieser Werte berechnet man die χ^2-Statistik mit $\nu = k - 1$ Freiheitsgraden:

$$\chi^2 = \sum_{s=1}^{l} \frac{(Y_s - n_s)^2}{n_s} \tag{4.1}$$

Bei einem bestimmten Signifikanzniveau α kann man nun das errechnete χ^2 mit einem Quantil $\chi^2_{\nu,p=(1-\alpha)}$ in der χ^2-Verteilungstabelle vergleichen. Die Nullhypothese wird verworfen, wenn $\chi^2 \geq \chi^2_{\nu,p}$.

Darüber hinaus lässt sich auch noch etwas über die Zufälligkeit des Experiments sagen: liegt χ^2 mit dem Freiheitsgrad ν unter einem Signifikanzniveau von $p = 0,01$ oder über eines von $p = 0,99$, ist die beobachtete Verteilung nicht zufällig genug. Bei $0,01 \leq p < 0,1$ und $0,9 \leq p < 0,99$ befindet sie sich in einem grauen Bereich. Das bedeutet: Die generierten Zufallszahlen sind erst dann Zufallszahlen, wenn sich χ^2 mit dem Freiheitsgrad ν in einem Signifikanzniveau von $0,1 \leq p < 0,9$ befindet [Knu97, S. 46 f.].

4.2. Kolmogorov-Smirnov-Anpassungstest

Der Kolmogorov-Smirnov-Test [Eck14, S. 328 f.] (im Folgenden KS-Test genannt) wird ebenfalls zur Überprüfung der Verteilungsfunktion verwendet. Vorteilhaft gegenüber dem Chi-Quadrat-Anpassungstest ist, dass der KS-Test für kleinere n und auf stetige Verteilungsfunktionen anwendbar ist. Zudem müssen die Zufallszahlen nicht in Kategorien unterteilt werden.

Im KS-Test wird ebenfalls eine Nullhypothese $H_0 : F_X(x_i) = F_X^e(x_i)$ gegen eine Alternativhypothese $H_1 : F_X(x_i) \neq F_X^e(x_i)$ getestet. Nach der Ausgabe von n Zufallszahlen

werden diese nach der Wertigkeit (von der kleinsten bis zur größten Zufallszahl) sortiert, um eine Verteilungsfunktion $F_X(x)$ nachzubilden. In der Regel stellt diese eine gleichverteilte Funktion dar. Zugleich werden die Funktionswerte von $F_X^e(x)$ mit den ermittelten Zufallswerten bestimmt. Danach werden die Differenzen $|F_X(x_i) - F_X^e(x_i)|$ und $|F_X(x_{i-1}) - F_X^e(x_i)|$ gebildet und das Supremum unter den $2n$ Differenzen gesucht. Mithilfe des Supremums und folgender Gleichung wird der Testvariablenwert bestimmt:

$$k_n = \sqrt{n} \cdot \sup|F_X - F_X^e|. \tag{4.2}$$

Nun wird das Quantil $k_{1-\alpha,n}$ für ein Signifikanzniveau α in einer Wertetabelle ermittelt. Wenn $k_n < k_{1-\alpha,n}$ gilt, wird die Nullhypothese behalten, wird sie verworfen.

4.3. Run-Test

Der Run-Test von Abraham Wald und Jacob Wolfowitz [WW40, S. 147 ff.] bezieht sich auf die Zufälligkeit zweier unabhängiger Zufallsvariablen X und Y kontinuierlicher Verteilungsfunktionen. Dabei sind $\{x_i|i \in \{1, ..., m\}\}$ die Menge der unabhängigen Beobachtungen von X und $\{y_j|i \in \{1, ..., n\}\}$ die Menge der unabhängigen Beobachtungen von Y. Die Gesamtheit der Beobachtungen wird von $Z := \{z_l|l \in \{1, ..., o = m + n\}\}$ mit $z_1 < ... < z_o$ dargestellt. Außerdem gibt es noch eine weitere Menge $V = \{v_i|i \in \{1, ..., o\}\}$, für die folgende Vorschrift gilt:

$$v_i = \begin{cases} 0 & z_i \in \{x_1, ..., x_m\} \\ 1 & z_i \in \{y_1, ..., y_n\} \end{cases}. \tag{4.3}$$

Nun ergibt sich aus der Reihenfolge der Zufallszahlen eine Zahl U, die die Anzahl der Runs angibt. Ein Run ist die Folge von mindestens einer identischen Zufallszahl. Für die Anzahl der Runs gibt es $\frac{(o)!}{m! \cdot n!}$ Möglichkeiten der Ausgabe mit der Wahrscheinlichkeit $\frac{m! \cdot n!}{(o)!}$. Die Anzahl der Runs von X wird mit e_0 und die Anzahl der Runs von Y mit e_1 bezeichnet. Demzufolge kann man die Anzahl der gesamten Runs auch als $U = e_0 + e_1$ umschreiben. Es gibt zwei Möglichkeiten, wie U unter der Nullhypothese H_0, dass die beobachtete Reihenfolge zufällig ist, aussehen kann [BT13, S. 105 f.]:

1. $U = 2k$: Daraus folgt, dass $e_0 = e_1 = k$. Die Wahrscheinlichkeit für $2k$ Runs lässt sich daher wie folgt berechnen:

$$P(U = 2k) = \frac{2\binom{m-1}{k-1}\binom{n-1}{k-1}}{\binom{o}{m}} \qquad k \in \{1, ..., m\} \tag{4.4}$$

2. $U = 2k - 1$: Das bedeutet, dass $e_0 = k$, $e_1 = k - 1$ oder $e_0 = k - 1$, $e_1 = k$. Die Wahrscheinlichkeit für $2k - 1$ Runs lässt sich so berechnen:

$$P(U = 2k - 1) = \frac{\binom{m-1}{k-1}\binom{n-1}{k-2} + \binom{m-1}{k-2}\binom{n-1}{k-1}}{\binom{o}{m}} \qquad k \in \{2, ..., m + 1\} \tag{4.5}$$

Die Nullhypothese H_0 wird unter einem Signifikanzniveau α abgelehnt, wenn $U < u_{\frac{\alpha}{2}}$ oder $U > u_{1-\frac{\alpha}{2}}$. Daraufhin kann man ergründen, ob H_0 abgelehnt wurde, weil sich die Beobachtungen von X, Y zu wenig abwechseln ($U < u_\alpha$) oder sich zu oft abwechseln ($U > u_{1-\alpha}$). Die kritischen Werte können in einer entsprechenden Tabelle abgelesen werden.

Beispiel 4.1. Betrachte den Blum-Blum-Shub-Generator

$$x_{n+1} = (x_n)^2 \bmod 209$$

mit Startwert $x_0 = 4$. Die Zufallsfolge sieht folgendermaßen aus:

$$4\ 16\ 47\ 119\ 158\ 93\ 80\ 130\ 180\ 5\ 25\ 207.$$

Daraus folgt für die Zufallsbits

$$0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1.$$

Es sind sowohl $m = 6$ gerade als auch $n = 6$ ungerade Zufallszahlen, also $o = 12$ Zufallszahlen insgesamt sowie $U = 6$ Runs mit $e_0 = e_1 = k = 3$. Die Nullhypothese ist für $\alpha = 0,05$ zulässig, das heißt, die Zahlenfolge unterliegt einer Zufälligkeit.

Für große Stichproben, also $m, n > 20$, wird die Verteilung von U durch eine Normalverteilung angenähert [BT13, S. 107]. Unter H_0 gilt für den Erwartungswert:

$$E(U) = \frac{2mn}{o} + 1 \tag{4.6}$$

und für die Varianz:

$$Var(U) = \frac{2mn(2mn - o)}{o^2(o - 1)}. \tag{4.7}$$

Wenn $m = \alpha o$ und $n = o(1 - \alpha)$ mit $\alpha \in (0, 1)$, gelten [BT13, ebd.]:

$$\lim_{o\to\infty} E(\frac{(U)}{o}) = 2\alpha(1 - \alpha) \tag{4.8}$$

und

$$\lim_{o\to\infty} Var(\frac{(U)}{\sqrt{o}}) = 4\alpha^2(1 - \alpha)^2. \tag{4.9}$$

Unter H_0 gilt nun, dass

$$W = \frac{U - 2\alpha o(1 - \alpha)}{2\sqrt{o}\alpha(1 - \alpha)} \tag{4.10}$$

ist asymptotisch standardnormalverteilt. Somit wird H_0 verworfen, wenn $|W| \geq w_{1-\frac{\alpha}{2}}$. $w_{1-\alpha}$ lässt sich in der Tabelle für die Quantile der Standardnormalverteilung entnehmen.

4.4. serielle Autokorrelation

Mithilfe der seriellen Autokorrelation kann man die Unabhängigkeit einer Folge von Zufallszahlen überprüfen. Der Autokorrelationskoeffizient zum Zeitabstand k ergibt sich aus [Guj04, S. 450]:

$$\rho_k = \frac{E([u_t - E(u_t)][u_{t+k} - E(u_{t+k})])}{\sqrt{Var(u_t)}\sqrt{Var(u_{t+k})}}. \tag{4.11}$$

Mit Gleichung (2.48) folgt:

$$\rho_k = \frac{\sum_{(t=1)}^{n-k}(u_t - \overline{u})(u_{t+k} - \overline{u})}{\sum_{t=1}^{n-k}(u_t - \overline{u})^2}. \tag{4.12}$$

Da man davon ausgeht, dass die Varianzen gleich bleiben, ist $Var(u_t) = Var(u_{t+k})$ und $E(u_t) = 0$ [Guj04, S. 450].

Die Autokorrelation bewegt sich in einem Bereich von $(-1, 1)$. Ist $\rho_k = 0$, so kann man wegen der Unkorreliertheit von Zufälligkeit sprechen, und je weiter sich der Korrelationskoeffizient von 0 entfernt, umso weniger zufällig sind die generierten Zahlen.

5. Simulation als praktische Anwendung

5.1. Simulation mit dem linearen Kongruenzgenerator

Für genügend große n und den Bedingungen aus Satz 3.5 kann man mit dem Lehmer-Generator Zufallszahlen und ein Muster der Punkte $(u_{n-1}; u_n)$ mit $u_i = \frac{x_i}{m}$ erzeugen. Um das deutlich zu machen, seien hierfür $n = 10, 100, 1000$. Der Quellcode befindet sich im Anhang.

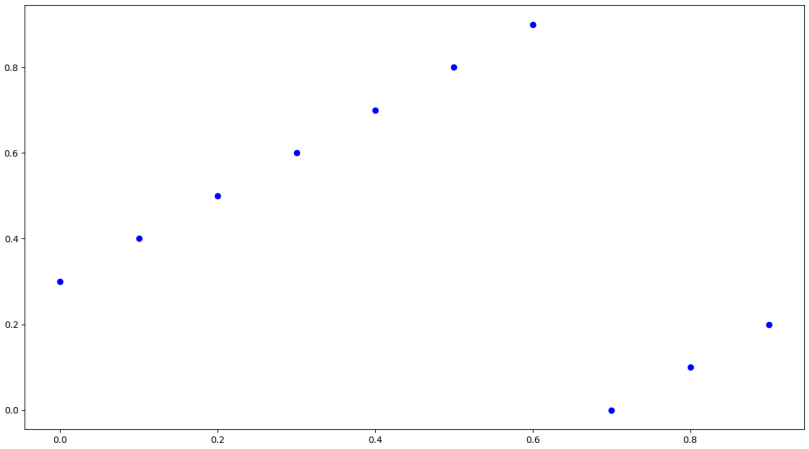

Abbildung 1: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 1$, $b = 3$, $n = 10$

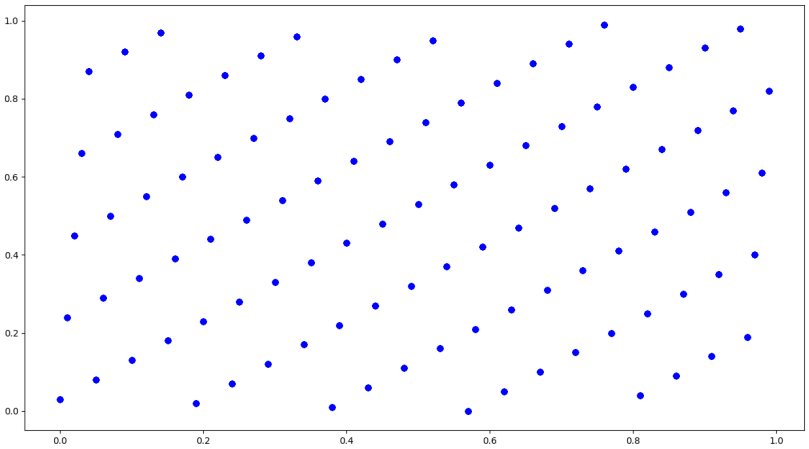

Abbildung 2: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 21$, $b = 3$, $n = 100$

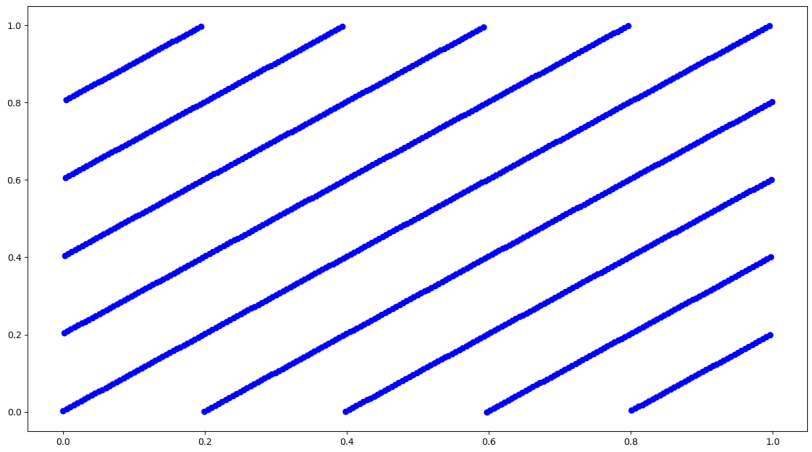

Abbildung 3: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 201$, $b = 3$, $n = 1000$

Wie man erkennen kann, nehmen die Zufallspunkte auf den einzelnen Abbildungen Geradenmuster an, deren Geraden zueinander parallel sind. Interessant zu sehen ist auch, wie sich die Muster verändern, wenn einzelne Bedingungen aus Satz 3.5 nicht mehr gelten. Gilt Bedingung 1 nicht mehr, sähe das erzeugte Muster nun so aus:

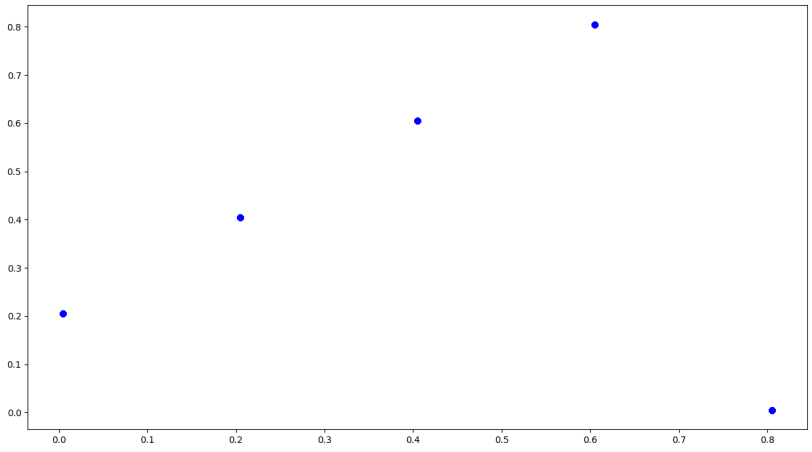

Abbildung 4: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 201$, $b = 200$, $n = 1000$

Das Geradenmuster ist immer noch erkennbar, dennoch ist die Punktdichte wegen der deutlich verminderten Periodenlänge 5 (ablesbar an der Anzahl der Punkte in diesem Graphen) sehr gering.

Wenn die Bedingung 3 nicht mehr gilt, sieht der Graph so aus:

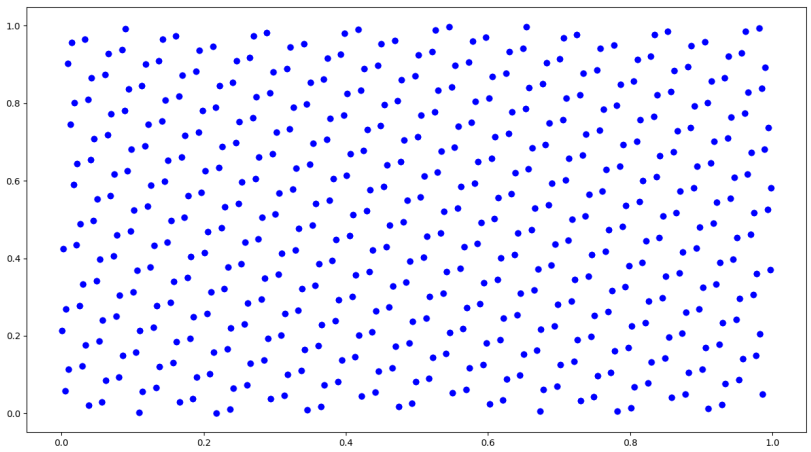

Abbildung 5: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 211$, $b = 3$, $n = 1000$

Wenn Bedingung 2 nicht mehr gilt, gibt es folgende Möglichkeiten:

I $(a - 1)$ ist nicht mehr durch 2 teilbar

II $(a - 1)$ ist nicht mehr durch 2 teilbar

III $(a - 1)$ ist weder durch 2 noch durch 5 teilbar

Für den Fall, dass I gilt, sieht der Graph so aus:

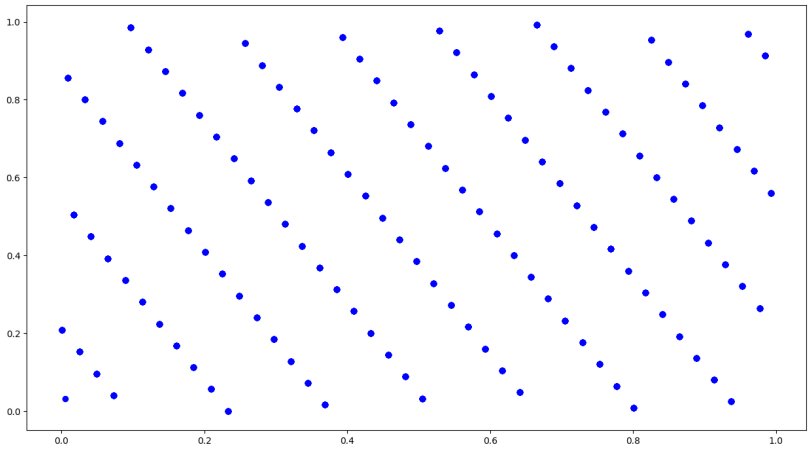

Abbildung 6: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 206$, $b = 3$, $n = 1000$

Wenn II gilt, sieht er so aus:

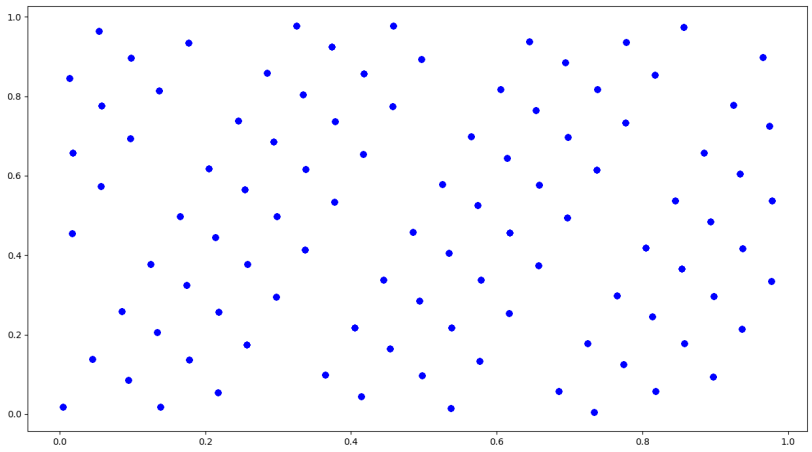

Abbildung 7: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 203$, $b = 3$, $n = 1000$

Wenn III gilt, sieht er so aus:

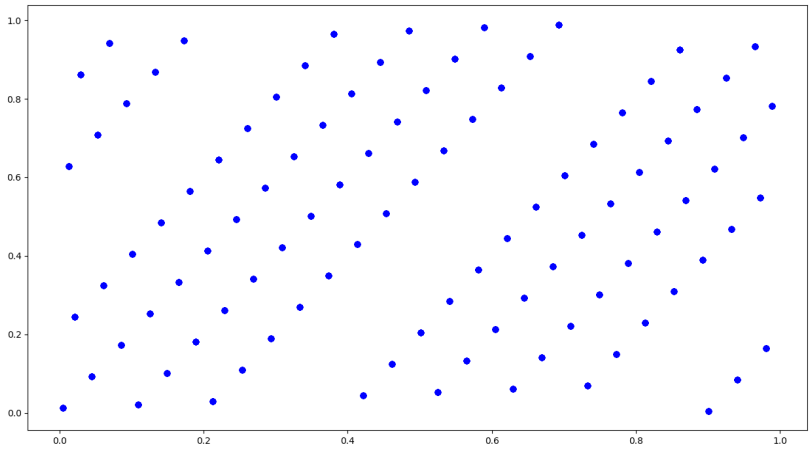

Abbildung 8: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 202$, $b = 3$, $n = 1000$

Für den Fall, dass keine der Bedingungen gilt, sind die Punkte folgendermaßen verteilt:

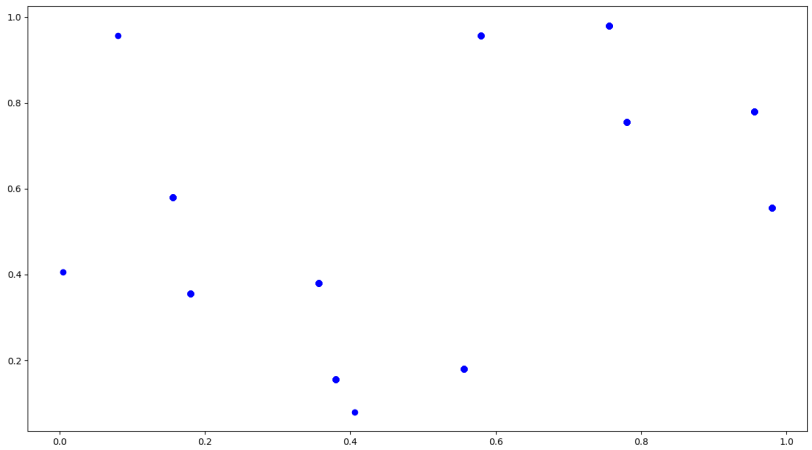

Abbildung 9: Simulation des LKG mit den Parametern $x_0 = 5$, $a = 74$, $b = 36$, $n = 1000$

Die Größe der Punkte zeigt, wie oft die Zufallszahlen generiert wurden. Die Punkte $(0,005; 0,406)$, $(0,406; 0,080)$ und $(0,080; 0,956)$ wurden nur einmal generiert und erscheinen daher kleiner.

5.2. Simulationen mit dem Weyl-Generator

Wie in Kapitel 3.3 erwähnt, gibt es drei wesentliche Möglichkeiten, einen Weyl-Generator zu implementieren. Diese werden im Folgenden vorgestellt. Auf dem Graphen werden die N Punkte $(u_n = \frac{n}{N}; y_n = x_n - [x_n])$ dargestellt.

5.2.1. Weyl-Generator nach der ersten Möglichkeit

Nach Kapitel 3.3, erster Punkt wurde als Beispiel der Generator mit der Bildungsvorschrift $x_n = \sqrt{7}n$ implementiert. Die Graphen der Pseudozufallszahlenverteilung sehen so aus:

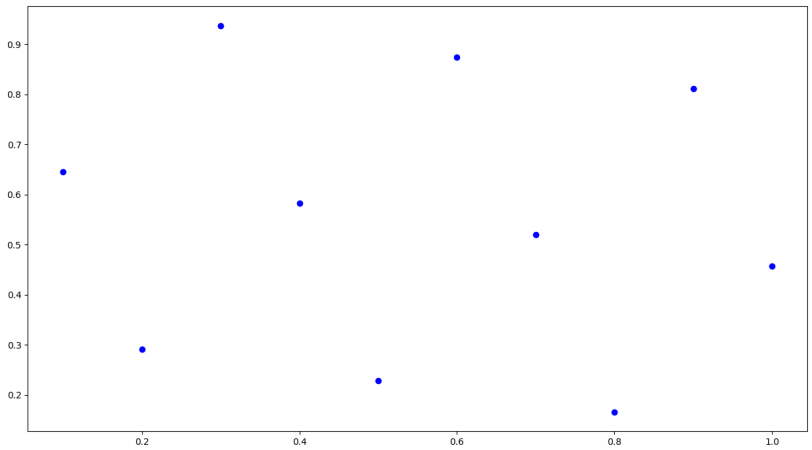

Abbildung 10: Simulation des Weyl-Generators I für $n \in [1, 10]$

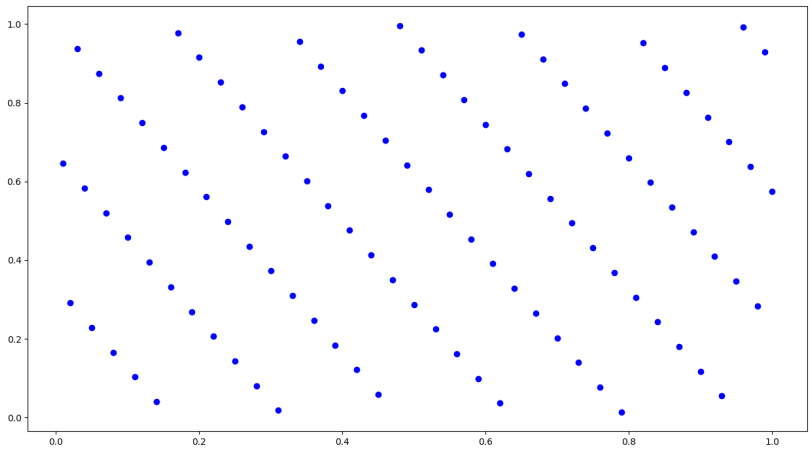

Abbildung 11: Simulation des Weyl-Generators I für $n \in [1, 100]$

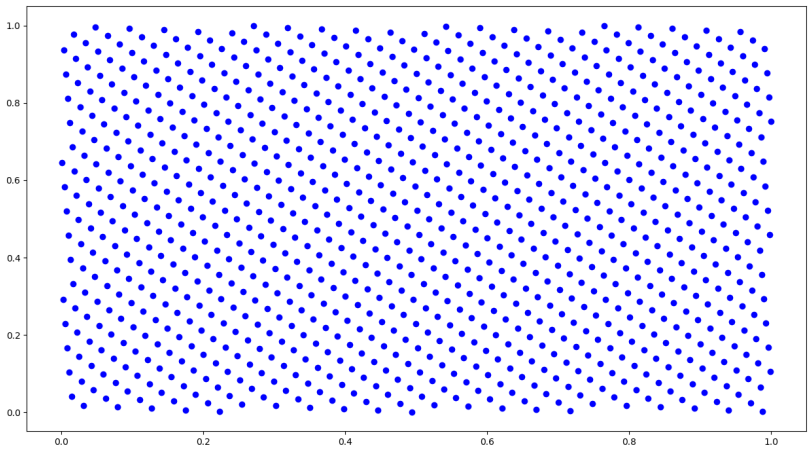

Abbildung 12: Simulation des Weyl-Generators I für $n \in [1, 1000]$

Die Verteilung dieser erzeugten Zufallszahlen hat eine große Ähnlichkeit mit der des Lehmer-Generators, wenn alle Bedingungen erfüllt sind. Jedoch sind die erzeugten Zahlen gleichmäßiger verteilt als in den Verteilungsgraphiken des selbigen, was man sehr gut an den 1000 erzeugten Zufallszahlen erkennen kann.

5.2.2. Weyl-Generator nach der zweiten Möglichkeit

Nach Kapitel 3.3 wurde hierfür der Generator mit der Bildungsvorschrift $x_n = n^{0,4} log_{3,4}(n)$ implementiert. Die Graphen der Pseudozufallszahlenverteilung sehen so aus:

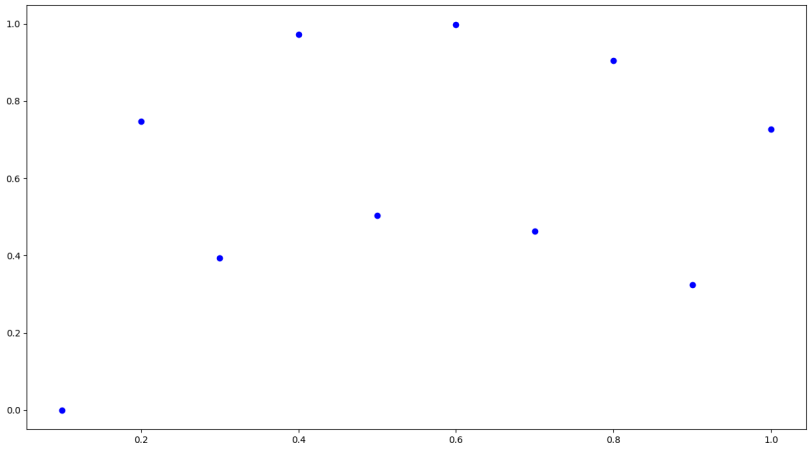

Abbildung 13: Simulation des Weyl-Generators II für $n \in [1, 10]$

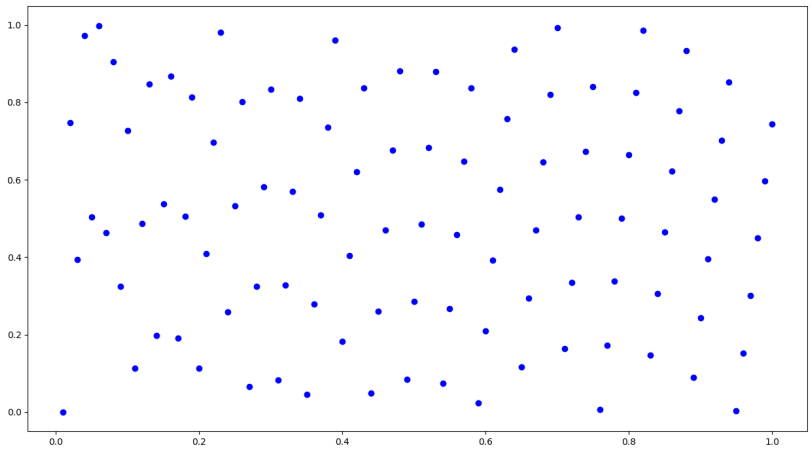

Abbildung 14: Simulation des Weyl-Generators II für $n \in [1, 100]$

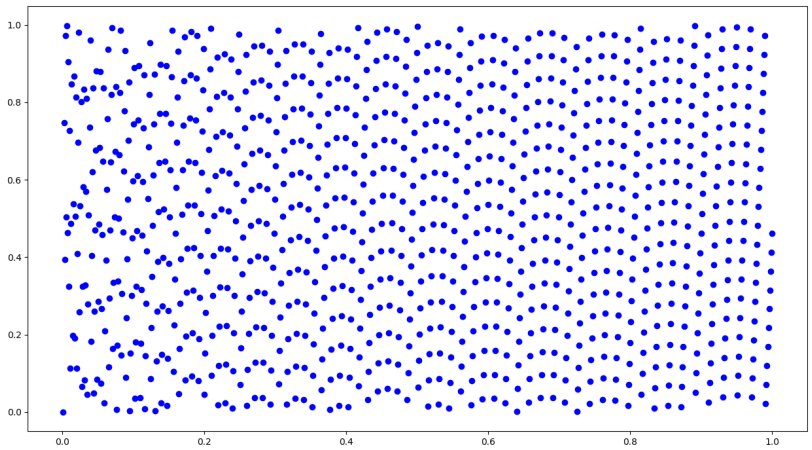

Abbildung 15: Simulation des Weyl-Generators II für $n \in [1, 1000]$

Bereits für die Erzeugung von zehn Pseudozufallszahlen ist ein Muster erkennbar, welches zwei Kurven zeigt, die bis $n = 5$ steigen und ab da wieder fallen. Für steigendes n wird das Verteilungsmuster erkennbar feiner. Ab $n \approx 20$ bzw. $n \approx 200$ wird dieses zunächst ungeordnete Muster auch geordneter.

5.2.3. Weyl-Generator nach der dritten Möglichkeit

Nach Kapitel 3.3 wurde dafür als Beispiel der Generator mit der Bildungsvorschrift $x_n = 9,81 + 2n + \sqrt{5}n^2$ implementiert. Die Graphen der Pseudozufallszahlenverteilung

sehen demnach so aus:

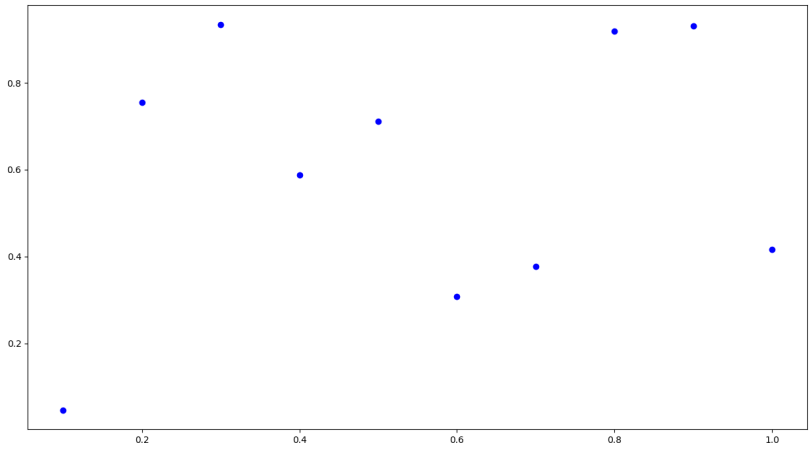

Abbildung 16: Simulation des Weyl-Generators III für $n \in [1, 10]$

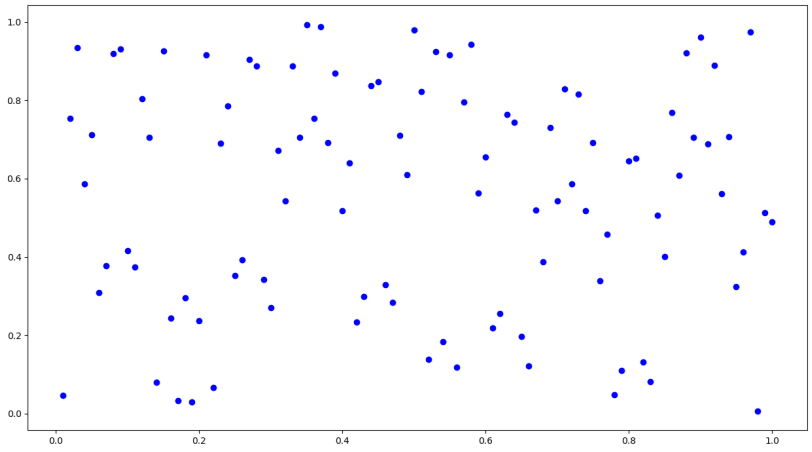

Abbildung 17: Simulation des Weyl-Generators III für $n \in [1, 100]$

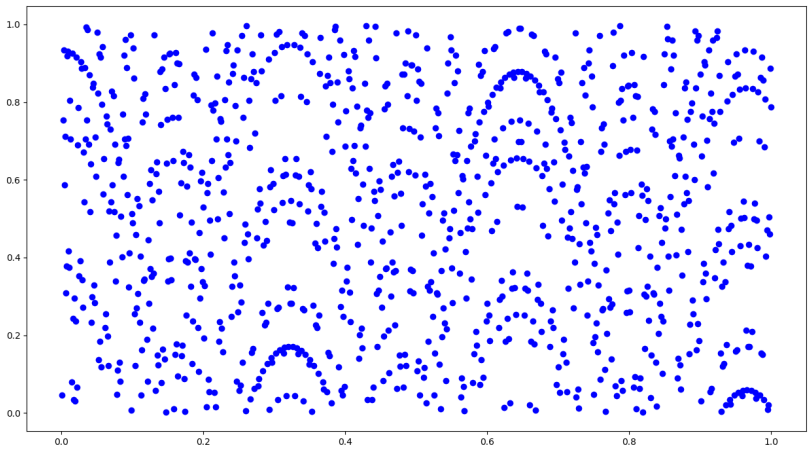

Abbildung 18: Simulation des Weyl-Generators III für $n \in [1, 1000]$

Es fällt auf, dass für zehn erzeugte Zufallszahlen kein festes Muster erkennbar ist. Dieses wird erst ab $n = 100$ und ab $n = 1000$ deutlich sichtbar, dass eine Reihe von nach unten geöffneten quadratischen Funktionen als Muster gezeigt wird.

5.3. Statische Vergleiche der beiden Zufallsgeneratoren

Im Folgenden werden der Lehmer-Generator und die drei Varianten des Weyl-Generators miteinander verglichen. Hierfür werden die folgenden Testverfahren genutzt: der Chi-Quadrat-Test und die serielle Autokorrelation.

5.3.1. Chi-Quadrat-Test

Für den Chi-Quadrat-Test wurde das Intervall [0;1) in zehn gleich große Kategorien, also Teilintervalle eingeteilt, so dass $p_s = 0,1$. Des Weiteren wurde pro Generator, der oben vorgestellt wurde, $n = 1000$ Zufallszahlen generiert. Daraus ergibt sich $n_s = 100$. Gemäß Gleichung (4.1) wurde das χ^2 pro Generator berechnet:

Linearer Kongruenzgenerator .

Kategorie	LKG	LKG ohne Bed. 1	LKG ohne Bed. 2a	LKG ohne Bed. 2b	LKG ohne Bed. 2c	LKG ohne Bed. 3	LKG ohne alles
[0;0,1)	100	200	104	120	100	100	1
[0,1;0,2)	100	0	96	80	100	100	199
[0,2;0,3)	100	200	104	120	100	100	0
[0,3;0,4)	100	0	96	80	100	100	200
[0,4;0,5)	100	200	104	120	100	100	1
[0,5;0,6)	100	0	96	80	100	100	199
[0,6;0,7)	100	200	104	120	100	100	0
[0,7;0,8)	100	0	96	80	100	100	200
[0,8;0,9)	100	200	104	120	100	100	0
[0,9;1)	100	0	96	80	100	100	200
χ^2	0	1000	16	40	0	0	992,04

Weyl-Generator .

Kategorie	Weyl 1	Weyl 2	Weyl 3
[0;0,1)	99	106	108
[0,1;0,2)	101	98	100
[0,2;0,3)	100	98	90
[0,3;0,4)	99	103	98
[0,4;0,5)	100	99	90
[0,5;0,6)	101	101	96
[0,6;0,7)	99	103	103
[0,7;0,8)	101	93	95
[0,8;0,9)	100	103	117
[0,9;1)	100	96	103
χ^2	0,06	1,38	6,76

χ^2_9 für $p \in [0,01; 0,99]$ befindet sich laut der Tabelle im Kapitel B.1 im Intervall $[2,088; 21,67]$. Nach den Kriterien in Kapitel 4.1 entsprechen die generierten Zahlen folgender Generatoren einer Zufälligkeit:

- der Lehmer-Generator, bei dem die Bedingung 2a verletzt wurde
- der Weyl-Generator, der nach der dritten Möglichkeit konzipiert wurde

Außerdem sind die Zahlen folgender Generatoren gleichverteilt für $\chi^2_{9;0,95}$:

- der Lehmer-Generator, dessen Bedingungen für eine maximale Periodenlänge gelten, bei dem die Bedingung 2b verletzt, bei dem die Bedingung 2c verletzt und bei dem die Bedingung 3 verletzt wurde
- der Weyl-Generator, der nach allen Möglichkeiten konzipiert wurde

Zusammenfassend lässt sich im Vergleich der Generatorenarten sagen, dass Zufälligkeit und Gleichverteilung nicht immer miteinander zusammenhängen, da nur von zwei von fünf Generatoren, die gleichverteilte Zufallszahlen ausgeben, auch gleichzeitig Zufälligkeit gewährleisten.

5.3.2. Serielle Autokorrelation

Hierfür werden alle drei Weyl-Generatoren und der Lehmer-Generator betrachtet. Da es sich um gleichverteilte Zufallszahlen im Intervall [0;1] handelt, gilt für den Mittelwert in Gleichung (4.12) $\overline{u} = 0,5$. Des Weiteren wird die serielle Autokorrelation zweier aufeinanderfolgender Zufallszahlen betrachtet, das heißt $k = 0$. Somit berechnet man für $n = 10$ Zufallszahlen:

$$\rho_1 = \frac{\sum_{t=1}^{9}(u_t - 0,5)(u_{t+1} - 0,5)}{\sum_{t=1}^{9}(u_t - 0,5)}$$

Iteration	LKG	Weyl 1	Weyl 2	Weyl 3
1	0,8	0,646	0	0,046
2	0,1	0,292	0,747	0,754
3	0,4	0,937	0,393	0,935
4	0,7	0,583	0,972	0,587
5	0	0,229	0,504	0,712
6	0,3	0,875	0,998	0,308
7	0,6	0,520	0,463	0,377
8	0,9	0,166	0,904	0,918
9	0,2	0,812	0,324	0,932
10	0,5	0,458	0,726	0,417
ρ_1	-0,24	$\approx$ -0,47	$\approx$ -0,58	$\approx$ 0,13

Wie sich herausstellt, korrelieren die aufeinanderfolgenden Zufallszahlen aller Generatoren miteinander, am stärksten die Zufallszahlen des Weyl-Generators, der nach der zweiten Möglichkeit konzipiert wurde. Das heißt auch, dass die Zahlen nicht zufällig erscheinen. Näher am Zufall dran sind die Zufallszahlen des Weyl-Generators, die nach der dritten Möglichkeit konzipiert wurden. Daraus kann man schließen, dass die Zufallszahlen in Teilen von der jeweils vorhergehenden Zufallszahl abhängig ist. Jedoch sind die Zufallszahlen, die nach dem Satz von Weyl generiert werden, nicht abhängig von der vorher generierten Zufallszahl, weshalb der Test weniger geeignet ist. Daher wird hier zu dem Runs-Test gegriffen.

5.3.3. Runs-Test

Es werden die Zufallszahlen des Weyl-Generators betrachtet. Diejenigen, die kleiner als 0,5 sind, werden mit 0, diejenigen, die größer oder gleich 0,5 sind, mit 1 bezeichnet:

Iteration	Weyl 1	Weyl 2	Weyl 3
1	1	0	0
2	0	1	1
3	1	0	1
4	1	1	1
5	0	1	1
6	1	1	0
7	1	0	0
8	0	1	1
9	1	0	1
10	0	1	0
m (Anzahl der 0)	4	4	4
n (Anzahl der 1)	6	6	6
Runs	8	8	5

Für ein Signifikanzniveau von $\alpha = 0,05$ ist das Intervall, für die generierte Pseudozufallszahlen noch zufällig sind, [2;9]. Es liegen alle Runs der drei Weyl-Generatoren im gültigen Bereich, womit nun ihre Zufälligkeit nachgewiesen ist.

5.4. Ausblick in die Monte-Carlo-Methoden

Eine weitere Anwendung finden Pseudozufallszahlen in den Monte-Carlo-Methoden (mehr Informationen finden sich in [Edd90], [Gen02], [Lem09], [MGNR12], [Nie92]). Hierbei handelt es sich um Algorithmen, die Pseudozufallszahlen wegen ihrer Wiederholbarkeit in der Art ihrer Erzeugung verwenden. Bevorzugt wird unter anderem der Mersenne-Twister für die Simulationen verwendet [MGNR12, S. 3]. Mit den Beobachtungen, die mithilfe der Pseudozufallszahlen gemacht werden, können Eigenschaften eines Systems mit sich zufällig verhaltenden Bestandteilen, gewonnen werden. Aus diesen Simulationen erhält man Voraussagen oder Schlussfolgerungen [Lem09, S. 1 f.].

Es gibt recht viele Monte-Carlo-Methoden. Das Standardbeispiel ist die Monte-Carlo-Integration. Hier wird ein Integral der reellwertigen Funktion $f : [0,1] \to \mathbb{R}$ mit den Zufallszahlen $x_1, ..., x_n \in [0,1]$

$$I[f] = \int_0^1 f(x)\mathrm{d}x \tag{5.1}$$

durch

$$I[f] = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{5.2}$$

approximiert. Bedeutender ist aber die Monte-Carlo-Integration für $f : [0,1]^d \to \mathbb{R}$ mehrdimensionale Zufallsvektoren $\mathbf{x}_1, ..., \mathbf{x}_n \in [0,1]^d$ mit $d \in \mathbb{N}$, welche analog zur eindimensionalen Monte-Carlo-Integration funktioniert [MGNR12, S. 6].

Bekannte Monte-Carlo-Simulationen sind unter anderem:

- der Wiener-Prozess, welcher mithilfe der Monte-Carlo-Integration einen Wechel von einem diskreten zu einem kontinierlichen Modell verspricht (nachzulesen in [BBPZ13, S. 124-136])
- der Markov-Prozess, welcher eine Kombination aus
  - einem diskreten oder kontinuierlichen Zeitübergang und
  - einem diskreten oder kontinuierlichen Zustandsübergang

  ist. Der Wiener-Prozess ist eine Spezialform des Markov-Prozesses. Mehr ist nachzulesen in [BBPZ13], [MGNR12], [WW16].

Des Weiteren gibt es auch Quasi-Monte-Carlo-Methoden [Lem09], [Nie92]. Diese erstellt Mengen von Zufallsvektoren $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, deren Elemente $\mathbf{x}_i \in [0,1]^d$ sind. Die Quasi-Monte-Carlo-Integration ist analog zur Monte-Carlo-Integration.

Monte-Carlo-Simulationen finden im Alltag viele Anwendungsbereiche, vor allem in der theoretischen Physik (Quantenmechanik) und im Finanzwesen (Entwicklung von Geldanlagen).

6. Fazit

Pseudozufallszahlen haben anwendungsbezogen eine große Bedeutung. Wenn man Zufallszahlen generieren möchte, sollte man dabei immer im Hinterkopf haben, was man damit erreichen möchte. Für Monte-Carlo-Methoden, die für Simulationen und Modellierungen verwendet werden, eignen sich Matrixkongruenzgeneratoren, die in der Lage sind, parallel mehrere Pseudozufallszahlen zu generieren, und es erspart dem/der Programmierenden, mehrere lineare Kongruenzgeneratoren nebeneinander laufen zu lassen. Möchte man hochdimensional gleichverteilte Pseudozufallszahlen, kann man auch den Mersenne-Twister nutzen, denn dieser generiert aus 624 Startwerten in einem Durchlauf 624 neue Pseudozufallszahlen mit einer Länge von 32 Bit. Daher sollte man sich für diesen Generator Arbeitsspeicher einräumen, da dieser ungefähr 2,5 Kilobyte benötigt. Auch für Verschlüsselungen eignet sich dieser nicht so gut. In diesem Fall ist dem/der Programmierenden geraten, den Blum-Blum-Shub-Generator zu nutzen, dessen Aufgabe es ist, Zufallsbits zu generieren. Dennoch braucht man für so eine möglichst große Periodenlänge auch recht große Primzahlen, die kongruent zu drei modulo vier sind.

Periodenlänge ist ein gutes Stichwort: um eine maximale Periodenlänge zu gewährleisten, hilft es, bei der Wahl der Parameter darauf zu achten, dass diese den Bedingungen entsprechen, wie sie beim linearen Kongruenzgenerator der Fall sind. Es gibt aber auch Kongruenzgeneratoren, deren Periodenlänge nicht der Modulus ist, sondern größer oder kleiner sind. Ein Beispiel für Periodenlängen, die größer als der Modulus sind, sind die Pseudozufallszahlen, die vom additiven Kongruenzgenerator erzeugt werden. Beispiele für die kleinere Periodenlängen als der Modulus sind, wären Pseudozufallszahlen, die vom multiplikativen Kongruenzgenerator oder vom Blum-Blum-Shub-Generator erzeugt werden. Zweiteres weist eine Periodenlänge auf, die wesentlich kleiner als der Modulus ist, da diese nur die zweifach verknüpfte Carmichael-Funktion des Modulus teilen. Es gibt aber auch Generatoren, deren Periodenlänge man auch wählen kann, da diese ein unendliches Repertoire an Pseudozufallszahlen aufweisen, wie die Generatoren, die nach dem Satz von Weyl konzipiert werden, da diese einzig vom n-ten Reihenglied abhängen. Bei den Matrixkongruenzgeneratoren schneidet der Mersenne-Twister MT19937 mit einer Periodenlänge von $2^{19937} - 1$ recht gut ab. Eine geringe Periodenlänge weist der Quadrat-Mitten-Generator auf, die generierten Pseudozufallszahlen recht schnell periodisch wird oder gegen Null konvergieren.

Wie sieht es mit der Gleichverteilung aus? Generell sind die Pseudozufallszahlen, die von Kongruenzgeneratoren erstellt werden, gleichverteilt im Intervall [0;1], da man diese nur durch den Modulus selbst teilen braucht. Entsprechend kann man das auch mit Hilfe des Chi-Quadrat-Tests überprüfen, in dem man die Zahlen in geeignete Intervalle unterteilt und dann herausfindet, dass die Differenz der ermittelten Häufigkeit und der erwarteten Häufigkeit gleich Null ist und ihr χ^2 immer unter den Quantilen bleibt, was auf Gleichverteilung hinweist. Jedoch sieht es anders aus, wenn die Bedingungen für eine maximale Periodenlänge verletzt werden. Dann ist keine ausreichende Gleichverteilung mehr gegeben. Im Einzelfall besteht sie weiterhin, aber aufgrund der verminderten Periodenlänge, die unter dem Maximum liegt, macht es ihn zu keinem guten Kongruenz-

generator mehr. Auch die Pseudozufallszahlen, die nach dem Satz von Weyl generiert wurden, sind gleichverteilt. Dies kann man aber erst feststellen, wenn wirklich genug Pseudozufallszahlen generiert werden, was beispielsweise bei $n = 1000$ der Fall ist.

Bei der Zufälligkeit von Pseudozufallszahlen kommt es auf das Testverfahren an. Beim Chi-Quadrat-Test schneidet der lineare Kongruenzgenerator, bei dem alle Bedingungen für eine maximale Periodenlänge eingehalten werden, schlecht ab, was die Zufälligkeit angeht, da $\chi^2 = 0$ darauf hindeutet, dass keine Zufälligkeit gegeben ist. Es liegt aber auch daran, dass die generierten Pseudozufallszahlen von der vorhergehenden Pseudozufallszahl abhängen. Diese Nichtzufälligkeit kann durch die serielle Autokorrelation bestätigt werden. Bei den Generatoren, die nach dem Satz von Weyl implementiert werden, kommt es auf die Möglichkeit an, nach denen man sie konzipiert. Dazu muss aber auch gesagt werden, dass sie an sich nicht voneinander abhängen, da der Laufindex des Folgengliedes eine wesentlichere Rolle bei der Erzeugung spielt. Somit muss man ein anderes Testverfahren anwenden, was nur geringfügig von der Reihenfolge der generierten Pseudozufallszahlen abhängt, wie den Runs-Test. Somit kommt man zu eindeutigeren Ergebnissen. Wie sieht es nun mit dem Algorithmus K aus? Daher, dass die durch den Algorithmus generierten Pseudozufallszahlen häufig gegen 6065038420 konvergieren, kann man hier kaum von Zufälligkeit sprechen. Auch beim Quadrat-Mitten-Generator stellt sich bei sehr vielen Zahlen eine periodische Folge ein, die nicht nur für die kurze Periodenlänge, sondern auch dafür sorgt, dass nur wenig Zufälligkeit gegeben ist.

Alles in allem bleibt zu sagen, dass die Generierung von Zufallszahlen ein recht breites Spektrum bietet, vor allem, was die Anwendung betrifft, wie zum Beispiel bei der Modellierung von Klimaprozessen. Daher bietet ein guter Ausblick hinsichtlich Pseudozufallszahlen die Monte-Carlo-Methoden, von denen es recht viele Prozesse gibt, so dass nur kurz auf die Wesentlichen eingegangen werden konnte. Weiterführend interessant wären auch die Anwendungsbereiche bezüglich dem Shuffle-Modus in Musikprogrammen und/oder MP3-Playern oder die *random*-Funktion in Python, vor allem, wie da die Algorithmen aufgebaut sind, da es sich vermutlich um andere handelt, als die, die hier dargestellt wurden. Jedoch sollte man sich vorher mit den Grundlagen des Programmierens, der Restklassenarithmetik und etwas Statistik auseinandersetzen, um einen guten Generator zu implementieren.

Literatur

[BBPZ13] Buchholz, Martin ; Bungartz, Hans-Joachim ; Pflüger, Dirk ; Zimmer, Stefan: *Modellbildung und Simulation. Eine anwendungsorientierte Einführung*. 2. Springer-Spektrum, 2013. – DOI 10.1007/978-3-642-37656-6

[BBS86] Blum, Lenore ; Blum, Manuel ; Shub, Michael: A Simple Unpredictable Pseudo-Random Number Generator. In: *SIAM Journal on Computing* 15 (1986), Nr. 2, S. 364–383

[BT13] Büning, Herbert ; Trenkler, Götz: *Nichtparametrische statistische Methoden*. 2. Walter de Gruyter, 2013

[Car12] Carmichael, Robert D.: On Composite Numbers P Which Satisfy the Fermat Congruence $a^{P-1} \equiv 1 \bmod P$. In: *The American Mathematical Monthly* 19 (1912), Februar, Nr. 2, S. 22–27. – URL: http://www.jstor.org/stable/2972687

[Eck14] Eckstein, Peter P.: *Repetitorium Statistik. Deskriptive Statistik - Stochastik - Induktive Statistik*. 8. Springer-Gabler-Verlag, 2014

[Edd90] Eddy, William F.: Random number generators for parallel processors. In: *Journal of Computational and Applied Mathematics* 31 (1990), S. 63–71

[EL86] Eichenauer, Jürgen ; Lehn, Jürgen: A non-linear congruential pseudo random number generator. In: *Statistical Papers* 27 (1986), S. 315–326

[Gen02] Gentle, James E.: *Random Number Generation and Monte Carlo Methods*. 2. George Mason University, 2002

[Guj04] Gujarati, Damodar N.: *Basic Econometrics*. 4. The McGraw-Hill Companies, 2004

[Hen17] Henze, Norbert: *Stochastik für Einsteiger. Eine Einführung in die faszinierende Welt des Zufalls*. 11. Springer-Spektrum-Verlag, 2017

[JRBT09] Jäger, Bernd P. ; Rudolph, Paul E. ; Biebler, Karl-Ernst ; Tuchscherer, Armin: Über die Erzeugung gleichverteilter Zufallszahlen. In: Spilke, Joachim (Hrsg.) ; Becker, Claudia (Hrsg.) ; Haerting, Johannes (Hrsg.) ; Schumacher, Erich (Hrsg.): *KSFE 2009. Proceedings der 13. Konferenz der SAS®-Anwender in Forschung und Entwicklung (KSFE)*, Shaker-Verlag, 2009, S. 107–117. – URL: http://saswiki.org/images/6/6b/13.KSFE-2009-Jaeger-Erzeugung_Pseudozufallszahlen.pdf

[JT05] Janke, Steven J. ; Tinsley, Frederick C.: *Introduction to Linear Models and Statistical Inference*. John Wiley and Sons, Inc., 2005. – DOI: 10.1002/047174011X

[Knu97] Knuth, Donald E.: *The Art of Computer Programming*. Bd. 2. Seminumerical Algorithms. 3. Addison-Wesley, 1997

[Kra08] Kramer, Jürg: *Zahlen für Einsteiger. Elemente der Algebra und Aufbau der Zahlbereiche*. Vieweg-Verlag, 2008

[Leh51] Lehmer, Derrick H.: Mathematical methods in large-scale computing units. In: *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery*, Harvard University Press, 1951, S. 141–146

[Lem09] Lemieux, Christiane: *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, 2009. – DOI: 10.1007/978-0-387-78165-5

[MGNR12] Müller-Gronbach, Thomas ; Novak, Erich ; Ritter, Klaus: *Monte-Carlo-Algorithmen*. Springer, 2012. – DOI 10.1007/978-3-540-89141-3

[MN98] Matsumoto, Makoto ; Nishimura, Takuji: Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. In: *ACM Transactions on Modeling and Computer Simulation* 8 (1998), Januar, Nr. 1, S. 3–30

[Nie92] Niederreiter, Harald: *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992

[Rib06] Ribenboim, Paulo: *Die Welt der Primzahlen. Geheimnisse und Rekorde*. Springer-Verlag, 2006

[Wey16] Weyl, Hermann: Über die Gleichverteilung von Zahlen mod. Eins. In: *Mathematische Annalen* 77 (1916), S. 313–352

[WW40] Wald, Abraham ; Wolfowitz, Jacob: On a Test Whether Two Samples are from the Same Population. In: *The Annals of Mathematical Statistics* 11 (1940), Juni, Nr. 2. – URL: http://www.jstor.org/stable/2235872

[WW16] Webel, Karsten ; Wied, Dominik: *Stochastische Prozesse. Eine Einführung für Statistiker und Datenwissenschaftler*. 2. Springer-Gabler, 2016. – DOI 10.1007/978-3-658-13885-1

A. Quelltexte

A.1. Lehmer-Generator

```
import matplotlib.pyplot as plt

x=input()                    #Eingabe des Startwertes
a=input()                    #Eingabe des Parameters (lineares Glied)
b=input()                    #Eingabe des Parameters (absolutes Glied)
m=input()                    #Eingabe des Restklassenparameters
n=0
o=0
p=0
q=0
r=0
s=0
t=0
c=0
d=0
w=0
i=1                          #Zaehlvariable
y=x                          #Hilfsvariable
z=0                          #Zufallszahl


for i in range(1,1001):      #fuer n Implikationen
    z=(a*y+b)%m              #Berechnungsvorschrift
    u=float(y)/m             #Umwandlung von y in eine Dezimalzahl
    v=float(z)/m             #Umwandlung von z in eine Dezimalzahl
    print y, z, u, v, i
    plt.plot(u, v, "bo")     #Diskrete Punkte (u,v) in ein
                             #Diagramm abbilden
    i=i+1                    #Erhoehung der Zaehlvariable
    if 0<=v<0.1:
        n=n+1
    elif 0.1<=v<0.2:
        o=o+1
    elif 0.2<=v<0.3:
        p=p+1
    elif 0.3<=v<0.4:
        q=q+1
    elif 0.4<=v<0.5:
        r=r+1
    elif 0.5<=v<0.6:
```

```
        s=s+1
    elif 0.6<=v<0.7:
        t=t+1
    elif 0.7<=v<0.8:
        c=c+1
    elif 0.8<=v<0.9:
        d=d+1
    elif 0.9<=v<1:
        w=w+1
    y=z                         #Hilfsvariable nimmt Zufallswert an

print "Anzahl aller Zufallszahlen zwischen 0 und 0.1:" ,n
print "Anzahl aller Zufallszahlen zwischen 0.1 und 0.2:" ,o
print "Anzahl aller Zufallszahlen zwischen 0.2 und 0.3:" ,p
print "Anzahl aller Zufallszahlen zwischen 0.3 und 0.4:" ,q
print "Anzahl aller Zufallszahlen zwischen 0.4 und 0.5:" ,r
print "Anzahl aller Zufallszahlen zwischen 0.5 und 0.6:" ,s
print "Anzahl aller Zufallszahlen zwischen 0.6 und 0.7:" ,t
print "Anzahl aller Zufallszahlen zwischen 0.7 und 0.8:" ,c
print "Anzahl aller Zufallszahlen zwischen 0.8 und 0.9:" ,d
print "Anzahl aller Zufallszahlen zwischen 0.9 und 1:" ,w
plt.show()                      #Ausgabe des Diagramms
```

A.2. Weyl-Generator für die Pseudozufallszahlenfolge $x_n = \sqrt{7}n$

```
import matplotlib.pyplot as plt
import math as m

n=input()                       #Eingabe des Startwertes
a=0
b=0
c=0
d=0
e=0
f=0
g=0
h=0
j=0
k=0
i=1                             #Zaehlvariable
z=0                             #Zufallszahl


for i in range(1,n):            #fuer n-1 Implikationen
```

```
    z=m.sqrt(7)*i           #Berechnungsvorschrift
    u=float(i)/(n-1)        #Umwandlung von y in ein
                            #Bruchteilfolgenglied
    v=float(z)-int(z)       #Umwandlung von z in ein
                            #Bruchteilfolgenglied
    print i, u, z, v
    plt.plot(u, v, "bo") #Diskrete Punkte (u,v) in ein
                            #Diagramm abbilden
    i=i+1                   #Erhoehung der Zaehlvariable
    if 0<=v<0.1:
        a=a+1
    elif 0.1<=v<0.2:
        b=b+1
    elif 0.2<=v<0.3:
        c=c+1
    elif 0.3<=v<0.4:
        d=d+1
    elif 0.4<=v<0.5:
        e=e+1
    elif 0.5<=v<0.6:
        f=f+1
    elif 0.6<=v<0.7:
        g=g+1
    elif 0.7<=v<0.8:
        h=h+1
    elif 0.8<=v<0.9:
        j=j+1
    elif 0.9<=v<1:
        k=k+1

print "Anzahl aller Zufallszahlen zwischen 0 und 0.1:" ,a
print "Anzahl aller Zufallszahlen zwischen 0.1 und 0.2:" ,b
print "Anzahl aller Zufallszahlen zwischen 0.2 und 0.3:" ,c
print "Anzahl aller Zufallszahlen zwischen 0.3 und 0.4:" ,d
print "Anzahl aller Zufallszahlen zwischen 0.4 und 0.5:" ,e
print "Anzahl aller Zufallszahlen zwischen 0.5 und 0.6:" ,f
print "Anzahl aller Zufallszahlen zwischen 0.6 und 0.7:" ,g
print "Anzahl aller Zufallszahlen zwischen 0.7 und 0.8:" ,h
print "Anzahl aller Zufallszahlen zwischen 0.8 und 0.9:" ,j
print "Anzahl aller Zufallszahlen zwischen 0.9 und 1:" ,k
plt.show()                  #Ausgabe des Diagramms
```

A.3. Weyl-Generator für die Pseudozufallszahlenfolge $x_n = n^{0,4} log_{3,4}(n)$

```
import matplotlib.pyplot as plt
import math as m

n=input()                       #Eingabe des Startwertes
a=0
b=0
c=0
d=0
e=0
f=0
g=0
h=0
j=0
k=0
i=1                             #Zaehlvariable
z=0                             #Zufallszahl


for i in range(1,n):            #fuer n-1 Implikationen
    z=(i**0.4)*m.log(i, 3.4)
                                #Berechnungsvorschrift
    u=float(i)/(n-1)            #Umwandlung von y in ein
                                #Bruchteilfolgenglied
    v=float(z)-int(z)           #Umwandlung von z in ein
                                #Bruchteilfolgenglied
    print i, u, z, v
    plt.plot(u, v, "bo") #Diskrete Punkte (u,v) in ein
                                #Diagramm abbilden
    i=i+1                       #Erhoehung der Zaehlvariable
    if 0<=v<0.1:
        a=a+1
    elif 0.1<=v<0.2:
        b=b+1
    elif 0.2<=v<0.3:
        c=c+1
    elif 0.3<=v<0.4:
        d=d+1
    elif 0.4<=v<0.5:
        e=e+1
    elif 0.5<=v<0.6:
```

```
        f=f+1
    elif 0.6<=v<0.7:
        g=g+1
    elif 0.7<=v<0.8:
        h=h+1
    elif 0.8<=v<0.9:
        j=j+1
    elif 0.9<=v<1:
        k=k+1

print "Anzahl aller Zufallszahlen zwischen 0 und 0.1:" ,a
print "Anzahl aller Zufallszahlen zwischen 0.1 und 0.2:" ,b
print "Anzahl aller Zufallszahlen zwischen 0.2 und 0.3:" ,c
print "Anzahl aller Zufallszahlen zwischen 0.3 und 0.4:" ,d
print "Anzahl aller Zufallszahlen zwischen 0.4 und 0.5:" ,e
print "Anzahl aller Zufallszahlen zwischen 0.5 und 0.6:" ,f
print "Anzahl aller Zufallszahlen zwischen 0.6 und 0.7:" ,g
print "Anzahl aller Zufallszahlen zwischen 0.7 und 0.8:" ,h
print "Anzahl aller Zufallszahlen zwischen 0.8 und 0.9:" ,j
print "Anzahl aller Zufallszahlen zwischen 0.9 und 1:" ,k
plt.show()                  #Ausgabe des Diagramms
```

A.4. Weyl-Generator für die Pseudozufallszahlenfolge $x_n = 9,81 + 2n + \sqrt{5}n^2$

```
import matplotlib.pyplot as plt
import math as m

n=input()                   #Eingabe des Startwertes
a=0
b=0
c=0
d=0
e=0
f=0
g=0
h=0
j=0
k=0
i=1                         #Zaehlvariable
z=0                         #Zufallszahl


for i in range(1,n):        #fuer n-1 Implikationen
```

```
    z=9.81+2*i+m.sqrt(5)*i**2
                            #Berechnungsvorschrift
    u=float(i)/(n-1)        #Umwandlung von y in ein
                            #Bruchteilfolgenglied
    v=float(z)-int(z)       #Umwandlung von z in ein
                            #Bruchteilfolgenglied
    print i, u, z, v
    plt.plot(u, v, "bo") #Diskrete Punkte (u,v) in ein
                            #Diagramm abbilden
    i=i+1                   #Erhoehung der Zaehlvariable
    if 0<=v<0.1:
        a=a+1
    elif 0.1<=v<0.2:
        b=b+1
    elif 0.2<=v<0.3:
        c=c+1
    elif 0.3<=v<0.4:
        d=d+1
    elif 0.4<=v<0.5:
        e=e+1
    elif 0.5<=v<0.6:
        f=f+1
    elif 0.6<=v<0.7:
        g=g+1
    elif 0.7<=v<0.8:
        h=h+1
    elif 0.8<=v<0.9:
        j=j+1
    elif 0.9<=v<1:
        k=k+1

print "Anzahl aller Zufallszahlen zwischen 0 und 0.1:" ,a
print "Anzahl aller Zufallszahlen zwischen 0.1 und 0.2:" ,b
print "Anzahl aller Zufallszahlen zwischen 0.2 und 0.3:" ,c
print "Anzahl aller Zufallszahlen zwischen 0.3 und 0.4:" ,d
print "Anzahl aller Zufallszahlen zwischen 0.4 und 0.5:" ,e
print "Anzahl aller Zufallszahlen zwischen 0.5 und 0.6:" ,f
print "Anzahl aller Zufallszahlen zwischen 0.6 und 0.7:" ,g
print "Anzahl aller Zufallszahlen zwischen 0.7 und 0.8:" ,h
print "Anzahl aller Zufallszahlen zwischen 0.8 und 0.9:" ,j
print "Anzahl aller Zufallszahlen zwischen 0.9 und 1:" ,k
plt.show()                  #Ausgabe des Diagramms
```

B. Tabellen

B.1. Chi-Quadrat-Test

Zusammengestellt aus den Informationen, die in [JT05, S. 570] und [Knu97, S. 44] erweitert zu finden sind.

ν	p=0,005	p=0,01	p=0,05	p=0,25	p=0,5	p=0,75	p=0,95	p=0,99	p=0,995
1	0,000	0,000	0,004	0,102	0,455	1,323	3,841	6,635	7,879
2	0,010	0,020	0,103	0,575	1,386	2,773	5,991	9,210	10,60
3	0,072	0,115	0,352	1,213	2,366	4,108	7,815	11,35	12,84
4	0,207	0,297	0,711	1,923	3,357	5,385	9,488	13,28	14,86
5	0,412	0,554	1,145	2,675	4,351	6,262	11,07	15,09	16,75
6	0,676	0,872	1,635	3,455	5,348	7,841	12,59	16,81	18,55
7	0,989	1,239	2,167	4,255	6,346	9,037	14,07	18,48	20,88
8	1,344	1,647	2,733	5,071	7,344	10,22	15,51	20,09	21,96
9	1,735	2,088	3,325	5,899	8,343	11,39	16,92	21,67	23,59
10	2,156	2,558	3,940	6,737	9,342	12,55	18,31	23,21	25,19
11	2,603	3,053	4,575	7,584	10,34	13,70	19,68	24,73	26,76
12	3,074	3,571	5,226	8,438	11,34	14,85	21,03	26,23	28,30

B.2. Kolmogorov-Smirnov-Test

n	p=0,01	p=0,05	p=0,25	p=0,5	p=0,75	p=0,95	p=0,99
1	0,010	0,050	0,250	0,500	0,750	0,950	0,990
2	0,014	0,067	0,293	0,518	0,707	1,098	1,273
3	0,017	0,079	0,311	0,515	0,754	1,102	1,359
4	0,019	0,088	0,320	0,511	0,764	1,130	1,378
5	0,022	0,095	0,325	0,525	0,767	1,139	1,402
6	0,023	0,100	0,327	0,532	0,770	1,146	1,414
7	0,025	0,104	0,328	0,536	0,776	1,154	1,425
8	0,027	0,107	0,328	0,539	0,780	1,159	1,433
9	0,028	0,112	0,327	0,541	0,783	1,162	1,439
10	0,029	0,115	0,330	0,543	0,785	1,166	1,444
11	0,030	0,117	0,333	0,544	0,786	1,169	1,448
12	0,031	0,119	0,336	0,545	0,788	1,171	1,452

Weitere Werte sind in [Knu97, S. 51] zu finden.

B.3. Runs-Test für α=0,05

Die obere Zahl in der Tabelle ist $u_{\frac{\alpha}{2}}$, die untere $u_{1-\frac{\alpha}{2}}$. Weitere Weite für α=0,025 sind in [JT05, S. 574] zu finden.

m	n=2	3	4	5	6	7	8	9	10	11	12
2	- -	- -	- -	- -	- -	- -	2 -	2 -	2 -	2 -	2 -
3	- -	- -	- 7	2 -	2 -	2 -	2 -	2 -	3 -	3 -	3 -
4	- -	- 7	2 8	2 9	3 9	3 9	3 -	3 -	3 -	3 -	4 -
5	- -	2 -	2 9	3 9	3 10	3 10	3 11	4 11	4 11	4 -	4 -
6	- -	2 -	3 9	3 10	3 11	4 11	4 12	4 12	5 12	5 13	5 13
7	- -	2 -	3 9	3 10	4 11	4 12	4 13	5 13	5 13	5 14	6 14
8	2 -	2 -	3 -	3 11	4 12	4 13	5 13	5 14	6 14	6 15	6 15
9	2 -	2 -	3 -	4 11	4 12	5 13	5 14	6 14	6 15	6 15	7 16
10	2 -	3 -	3 -	4 11	5 12	5 13	6 14	6 15	6 16	7 16	7 17
11	2 -	3 -	3 -	4 -	5 13	5 14	6 15	6 15	7 16	7 17	8 17
12	2 -	3 -	4 -	4 -	5 13	6 14	6 15	7 16	7 17	8 17	8 18

C. Erklärung

Ich versichere, dass ich die Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln (z. B. Nachschlagewerke oder Internet) angefertigt habe. Alle Stellen der Arbeit, die ich aus diesen Quellen und Hilfsmitteln dem Wortlaut oder dem Sinne nach entnommen habe, sind kenntlich gemacht und im Literaturverzeichnis aufgeführt. Weiterhin versichere, ich, dass weder ich noch andere diese Arbeit weder in der vorliegenden noch in einer mehr oder weniger abgewandelten Form als Leistungsnachweise in einer anderen Veranstaltung bereits verwendet haben oder noch verwenden werden.

Die „Richtlinie zur Sicherung guter wissenschaftlicher Praxis für Studierende an der Universität Potsdam (Plagiatsrichtlinie) - Vom 20. Oktober 2010“, im Internet unter `http://www.uni-potsdam.de/am-up/2011/ambek-2011-01-037-039.pdf`, ist mir bekannt.

Es handelt sich bei dieser Arbeit um meinen ersten Versuch.

Brandenburg an der Havel, den 14. Juni 2018 Unterschrift