## Pseudorandom Numbers

## Tomas Almlöf

Supervisor: Martin Lind
Examiner: Sorina Barza
Date: June 16, 2022


#### Abstract

In this thesis our goal is to study pseudorandom numbers. We will investigate how to produce pseudorandom samples from the uniform distribution with a method called the linear congruential method. Another method we will look at is the inverse sampling method which gives us the possibility to generate samples from other distributions that are not the uniform distribution. When generating pseudorandom samples quality is an important aspect, therefore we are going to take a look at a discrepancy which is a tool to determine quality of uniformly distributed samples. We implement the methods in Python and perform numerical experiments to test some quality aspects of the output.


## Sammanfattning

Målet med den här avhandlingen är att undersöka pseudoslumptal. Vi kommer att undersöka hur man kan producera pseudoslumptal som är likformigt fördelad med hjälp av den linjära kongruensmetoden. En annan metod vi ska undersöka är den så kallade inverse sampling method som används för att producera pseudoslumptal från andra fördelningar än den likformiga. När det kommer till att producera pseudoslumptal så är det viktigt att undersöka kvalitén. För att undersöka detta kommer vi att titta på diskrepans som är ett sätt att undersöka kvalitén för likformigt fördelade pseudoslumptal. Vi implementerar metoderna i Python och utför numeriska experiment för att testa några kvalitetsaspekter hos resultaten av implementeringen.

## Innehåll

1 Introduction ..... 5
2 Theoretical tools ..... 6
2.1 Probabilistic background ..... 6
2.1.1 Basic definitions ..... 6
2.1.2 Some usual continuous distributions ..... 6
2.1.3 The quantile function ..... 6
2.2 Discrepancy ..... 7
3 Pseudorandom number generation ..... 10
3.1 Pseudorandom samples, Inverse transform sampling ..... 10
3.2 Pseudorandom numbers, general framework ..... 10
3.3 The Linear Congruential Generator ..... 10
3.4 Discrepancy estimates for the LCG ..... 11
3.4.1 Discrepancy as quality measure ..... 11
3.4.2 Non-asymptotic behaviour of discrepancy ..... 12
4 Numerical experiments ..... 12
4.1 Implementation of LCG ..... 12
4.2 Computation of discrepancy of sets generated by LCM ..... 13
4.3 Inversive transform sampling ..... 15
5 Appendix: Python Code ..... 16

## 1 Introduction

Historically, randomness and random numbers have been used for a variety of purposes, from games to encryption and is today widely used in many fields. One particularly notable example is Monte Carlo simulations. Monte Carlo methods are a class of computational methods used in a wide range of STEM fields. Some examples of this would be to simulate fluids with a certain coupled degrees of freedom, calculating the probability of failure of a certain part on some machine, or numerical integration. The implementation of any Monte Carlo method requires random numbers.

However, the generation of truly random numbers is problematic in practice. In fact, generating the numbers requires a rule to implement it. Therefore, the obtained sequence is not truly random. Instead, one has to settle for the next best thing:pseudorandom numbers. These are sequences of numbers that 'appear' to be random. Typically, such sequences are constructed by number theoretic means.

In Chapter 2, we discuss some useful theoretical tools from probabilty. We also introduce the concept of discrepancy, which will be used to analyse methods of generating pseudorandom numbers. Roughly speaking, the discrepancy of a set (or sequence) of real numbers measure the deviation of the points of the set from being uniformly distributed.

In Chapter 3, we discuss pseudorandom numbers. For convenience, we will use the terminology that a set of pseudorandom numbers is drawn from a probability distribution if the set mimicks the properties of a random sample from the distribution.

We first discuss the linear congruential generator (LCG) which is an arithmetic method of generating pseudorandom numbers drawn from the uniform distribution $U([0,1])$ on $[0,1]$. It turns out that pseudorandom numbers from another distribution (say normal distribution) can be obtained by using a method called the inverse transform sampling to transform the uniform sample. The inverse sampling method supplies the user with a great deal of options in choosing a target probability distribution (see below).

We conclude the chapter with a discussion of how discrepancy can be used to capture certain quality aspects of pseudorandom samples.

In Chapter 4, we perform numerical experiments. We implement LCG and inverse transform method together with a discrepancy calculator in Python. Using these implementations, we can investigate the quality of the scheme of generating pseudorandom numbers.

In summary, the contribution of this thesis are

- an implementation of a scheme to generate pseudorandom numbers from any probability distribution,
- an analysis of the quality of the obtained pseudorandom numbers.


## 2 Theoretical tools

### 2.1 Probabilistic background

### 2.1.1 Basic definitions

Let $\Omega$ be a sample space with a probability function $P$. A subset of $A \subseteq \Omega$ is called an event. A random variable on $\Omega$ is a function $X: \Omega \rightarrow \mathbb{R}$. It is common practice in probability theory to use the following shorthand for events:

$$
\{X \leq x\}=\{u \in \Omega: X(u) \leq x\} .
$$

Let $X$ be any random variable. The distribution function of $X$ is given by

$$
F_{X}(x)=P(X \leq x) .
$$

Note that $\lim _{x \rightarrow-\infty} F_{X}(x)=0$ and $\lim _{x \rightarrow \infty} F_{X}(x)=1$, for any random variable $X$.
The variable $X$ is called a continuous random variable if there exists a function $f_{X}(x)$ such that

$$
F_{X}(x)=\int_{-\infty}^{x} f_{X}(t) d t .
$$

The function $f_{X}$ is called the density function of $X$. Two random variables $X, Y$ are called identically distributed if $F_{X}=F_{Y}$.

### 2.1.2 Some usual continuous distributions

The most useful continuous distribution is the normal distribution, when the density function is a scaled translate of the standard gaussian $\phi(x)=\frac{1}{\sqrt{2 \pi}} e^{-x^{2} / 2}$. However, we shall be interested in some other continuous distributions.

Let $I=[a, b]$ be an interval. The random variable $X$ is called uniformly distributed on $I$ (written $X \sim U(I)$ ) if $f_{X}(x)=1 /(b-a)$ for $x \in I$ and $f_{X}(x)=0$ for $x \notin I$. In particular, one can take $I=[0,1]$. In this case, it is useful to note that if $X \sim U([0,1])$ then

$$
F_{X}(x)= \begin{cases}0 & x \leq 0  \tag{1}\\ x & 0 \leq x \leq 1 \\ 1 & x \geq 1\end{cases}
$$

Another example is the exponential distribution. The continuous random variable $X$ has exponential distribution with intensity $\lambda>0$ (written $X \sim \operatorname{Exp}(\lambda)$ ) if its density function is $f_{X}(x)=\lambda e^{-\lambda x}$ for $x \geq 0$ and $f_{X}(x)=0$ for $x<0$. It is a simple calculation to see that

$$
F_{X}(x)= \begin{cases}1-e^{-\lambda x} & x \geq 0  \tag{2}\\ 0 & x \leq 0\end{cases}
$$

### 2.1.3 The quantile function

For any random variable $X$, the distribution function $F_{X}$ is increasing but not always strictly increasing. Thus, $F_{X}^{-1}$ is not always well-defined. Instead, one defines the generalised inverse or quantile function $Q_{X}:(0,1) \rightarrow \mathbb{R}$ by

$$
Q_{X}(x)=\inf \left\{t \in \mathbb{R}: x \leq F_{X}(t)\right\} .
$$

Remark 1. Some authors define $Q_{X}$ also for $\{0,1\}$, but then $Q_{X}$ takes values in the extended real numbers $\overline{\mathbb{R}}=\mathbb{R} \cup\{ \pm \infty\}$. For instance, it is immediately clear that for any $X$,

$$
Q_{X}(0)=\inf \left\{t \in \mathbb{R}: 0 \leq F_{X}(t)\right\}=\inf \mathbb{R}=-\infty .
$$

If $F_{X}\left(x_{0}\right)=1$ for some finite $x_{0}$, then $Q_{X}(1)<\infty$. If not,

$$
Q_{X}(1)=\inf \left\{t \in \mathbb{R}: 1 \leq F_{X}(t)\right\}=\inf \emptyset=\infty .
$$

The quantile function $Q_{X}$ satisfies a so-called Galois connection:

$$
\begin{equation*}
Q_{X}(x) \leq t \quad \Leftrightarrow \quad x \leq F_{X}(t) \tag{3}
\end{equation*}
$$

From (3) it follows that if $Y$ is another random variable and we set $Z=Q_{X}(Y)$, then the following two events coincide

$$
\begin{equation*}
\{Z \leq t\}=\left\{Y \leq F_{X}(t)\right\} \tag{4}
\end{equation*}
$$

Indeed, by using (3) we have

$$
\begin{aligned}
\{Z \leq t\}=\{u \in \Omega: Z(u) \leq t\} & =\left\{u \in \Omega: Q_{X}(Y(u)) \leq t\right\} \\
& =\left\{u \in \Omega: Y(u) \leq F_{X}(t)\right\}=\left\{Y \leq F_{X}(t)\right\} .
\end{aligned}
$$

Example 2. If $X \sim \operatorname{Exp}(\lambda)$, then $Q_{X}(x)=\frac{-\ln (1-x)}{\lambda}$.
Bevis. Since $X \sim \operatorname{Exp}(\lambda)$ we have that $F_{X}(x ; \lambda)=1-\operatorname{Exp}^{-\lambda x}$ for $x \geq 0$ and 0 otherwise. Let $1-\operatorname{Exp}^{-\lambda Q}=x$ for some $Q \in \mathbb{R} \Leftrightarrow-\lambda Q=\ln (1-x)$ divide by $-\lambda$ and we get $Q(x)=\frac{-\ln (1-x)}{\lambda}$.

Remark 3. It is not possible to find closed expressions for the quantile function of every random variable. Indeed, if $X$ has normal distribution then it is impossible to express $Q_{X}$ in terms of elementary functions. This is the main reason why statistics is filled with tables!

Proposition 4. Assume that $X$ is any random variable with quantile function $Q_{X}$ and that $Y \sim U([0,1])$. Set $Z=Q_{X}(Y)$, then $X$ and $Z$ are identically distributed: $F_{X}=F_{Z}$.

Bevis. By (4), $F_{Z}(t)=P(Z \leq t)=P\left(Y \leq F_{X}(t)\right)$. Furthermore, since $Y \sim U([0,1])$ and $F_{X}(t) \in[0,1]$, we have by (1) that $P\left(Y \leq F_{X}(t)\right)=F_{X}(t)$. I.e., $F_{Z}(t)=F_{X}(t)$.

### 2.2 Discrepancy

For any set $E \subseteq \mathbb{R}$, denote by $\chi_{E}(x)$ the characteristic function of $E$, i.e.

$$
\chi_{E}(x)= \begin{cases}1 & x \in E \\ 0 & x \notin E\end{cases}
$$

Let $P=\left\{x_{1}, x_{2}, \ldots, x_{n}, \ldots\right\} \subset[0,1]$ be countable (i.e. either finite or countable infinite). For any interval $J \subseteq[0,1]$, denote

$$
A_{k}(J, P)=\sum_{j=1}^{k} \chi_{J}\left(x_{j}\right)
$$

Note that $A_{k}(J, P)$ counts how many of the first $k$ numbers from $P$ belong to the interval $J$.

Definition 5. Fix $k \in \mathbb{N}$. The discrepancy $D_{k}(P)$ is defined by

$$
\begin{equation*}
D_{k}(P)=\sup _{0<u \leq 1}\left|\frac{A_{k}([0, u] ; P)}{k}-u\right| . \tag{5}
\end{equation*}
$$

In a sense, $D_{k}(P)$ measures how much the elements of $P$ deviate from being uniformly distributed.

Proposition 6. Let $P_{j}(1 \leq j \leq k)$ be finite sequences of $[0,1]$ with $\left|P_{j}\right|=N_{j}$. Set $P=\bigcup_{j=1}^{k} P_{j}$, where the first $N_{1}$ elements are the elements from $P_{1}$, the next $N_{2}$ elements are the elements of $P_{2}$ etc. Let $N=|P|=N_{1}+N_{2}+\cdots+N_{k}$. Then

$$
D_{N}(P) \leq \sum_{j=1}^{k} \frac{N_{j}}{N} D_{N_{j}}\left(P_{j}\right)
$$

Bevis. Let $J \subseteq[0,1)$ be any interval as in the definition of discrepancy. This implies that $A(J ; P)=\sum_{j=1}^{\bar{k}} A\left(J ; P_{j}\right)$ because of the definition of $P$. Hence we have that

$$
\left|A\left(J ; P_{j}\right)-N u\right|=\left|\sum_{j=1}^{k} A\left(J ; P_{j}\right)-N_{j} u(j)\right| \leq \sum_{j=1}^{k}\left|A\left(J ; P_{j}\right)-N_{j} u(j)\right| \leq \sum_{j=1}^{k} N_{j} D_{N_{j}}\left(P_{j}\right)
$$

Now divide the last inequality on the right hand side with $N$ and taking the supremum as in the definition of discrepancy we obtain

$$
\left|\frac{A\left(J ; P_{j}\right)}{N}-u\right| \leq \sum_{j=1}^{k} \frac{N_{j} D_{N_{j}}\left(P_{j}\right)}{N}
$$

Theorem 7. Let

$$
S=\left\{1, \frac{1}{2}, \frac{1}{3}, \frac{2}{3}, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \ldots\right\} .
$$

Then we have

$$
\begin{equation*}
D_{k}(S)=\mathcal{O}\left(\frac{1}{\sqrt{k}}\right) \tag{6}
\end{equation*}
$$

Bevis. Let

$$
S=\bigcup_{j=1}^{m} S_{j}+R(S)
$$

where $R(S)$ denotes the remainder. Now with $m$ being the number of åhole blocks". $S_{1}=1$, $S_{2}=\frac{1}{2}, S_{3}=\left\{\frac{1}{3}, \frac{2}{3}\right\}, \ldots, S_{m}=\frac{m-1}{m}$.
Now from our results from lemma 1 we have that

$$
D_{N}(S) \leq \sum_{j=1}^{j=m} \frac{\left|S_{j}\right|}{N} D_{\left|S_{j}\right|}\left(S_{j}\right)
$$

If we now apply theorem 1 we have that

$$
\begin{equation*}
\left|S_{j}\right| D_{\left|S_{j}\right|}\left(S_{j}\right) \leq 1 \Longrightarrow D_{N}(S) \leq \sum_{j=1}^{j=m} \frac{\left|S_{j}\right|}{N} D_{\left|S_{j}\right|}\left(S_{j}\right) \leq \sum_{j=1}^{j=m} \frac{1}{N}=\frac{m}{N} \tag{7}
\end{equation*}
$$

Note that the order of the blocks are $\left|S_{1}\right|=1, S_{2}|1|, S_{3}=|2|, S_{4}|3|, \ldots, S_{m}=|m-1|+R(S)$ which means that

$$
1+2+3+, \cdots+m-1 \leq N
$$

using the arithmetic sum of powers we have that

$$
1+2+3+, \cdots+m-1=\sum_{j=1}^{j=m-1} j=\frac{m(m-1)}{2}=\frac{m^{2}-m}{2}
$$

Using the definition of ordo we can get a relation between $N$ and $m$ as

$$
\begin{equation*}
c m^{2} \leq N \Longrightarrow \sqrt{\frac{N}{c}} \geq m \tag{8}
\end{equation*}
$$

Where $c$ is constant. Combining (1) and (2)

$$
D_{N}(S)=\frac{c \sqrt{N}}{N}=\frac{c}{\sqrt{N}}
$$

which is our desired result.
In Section 4 below, we shall perform a number of numerical experiments. Among other things, we shall compute discrepancy of finite sequences. Actually, we shall only compute discrepancy of the type

$$
D_{N}(P),
$$

where $N=|P|$. It turns out that when the discrepancy index" $k$ of $D_{k}(P)$ is the same as the size of $P$, i.e. $k=N$, then the ordering of the elements of $P$ does not matter. Indeed, this is obvious since

$$
A_{N}(J, P)=A_{N}\left(J, P^{\prime}\right)
$$

if $P^{\prime}$ is any reordering of $P$. This allows us to formulate the following lemma to compute $D_{N}(P)$.
Lemma 8. Let $P=\left\{x_{1}, x_{2}, \ldots, x_{N}\right\} \subset[0,1]$ and suppose that $x_{1} \leq x_{2} \leq \cdots \leq x_{N}$. Then

$$
\begin{equation*}
D_{N}(P)=\frac{1}{2 N}+\max _{1 \leq n \leq N}\left|x_{n}-\frac{2 n-1}{2 N}\right| . \tag{9}
\end{equation*}
$$

For a proof, see [1]. To create a program for calculating discrepancy, Python was used as language. The written program only needs a sequence of numbers as entry to calculate the discrepancy. The code for this program can be found in Appendix 1.

## 3 Pseudorandom number generation

### 3.1 Pseudorandom samples, Inverse transform sampling

As mentioned in the Introduction, generating pseudorandom numbers from a given distribution is an important task in several areas of applied mathematics and computer science.

We shall use the following terminology. Let $X$ be a random variable, we say that a set of numbers $x_{1}, x_{2}, \ldots, x_{n}$ is a pseudorandom sample from the distribution of $X$ if $x_{1}, x_{2}, \ldots, x_{n}$ behaves like a random sample from the distribution of $X$. We shall return briefly to what behaves likemeans later.

Of course, there are many distributions from which one might want pseudorandom samples. However, it turns out to be more or less sufficient to generate pseudorandom samples from $U([0,1])$. A pseudorandom sample from another distribution can then, in principle, be obtained by using inverse transform sampling.

The method works as follows. Assume that $X$ is an arbitrary random variable with quantile function $Q_{X}$ and $Y \sim U([0,1])$. By Proposition 4 above, $Q_{X}(Y)$ and $X$ are identically distributed. Hence, if $y_{1}, y_{2}, \ldots, y_{n}$ is a pseudorandom sample from $U([0,1])$, then we expect that

$$
x_{j}=Q_{X}\left(y_{j}\right) \quad(1 \leq j \leq n)
$$

is pseudorandom sample from the distribution of $X$. We will implement this scheme below.

### 3.2 Pseudorandom numbers, general framework

An algorithm that produces a pseudorandom sample from $U([0,1])$ is called a pseudorandom number generator (PRNG).

Below we shall give an abstract description of a pseudorandom number generator. Let $\mathcal{S}=\left\{z_{0}, z_{1}, \ldots z_{K}\right\}$ be a finite set of states and a state function $f: \mathcal{S} \rightarrow \mathcal{S}$ that progresses one state to the next, i.e. $z_{n}=f\left(z_{n-1}\right)$. The pseudorandom sample is generated by an output function $g: \mathcal{S} \rightarrow[0,1]$ in the following way

$$
\text { pseudorandom sample }=\left\{g\left(z_{n}\right)\right\} .
$$

The idea is that the function $f$ is somehow "unpredictableänd thus $\left\{g\left(z_{n}\right)\right\}$ appears to be a random sample from $U([0,1])$.
The initial state $z_{0}$, which is called the seed of the PRNG, is often provided by the user. Observe also that since $\mathcal{S}$ is a finite set, there is a smallest natural number $T \leq K$ such that $z_{T}=z_{0}$. The number $T$ is called the period of the PRNG. Since $g\left(z_{n+T}\right)=g\left(z_{n}\right)$, one typically only considers the pseudorandom sample $\left\{g\left(z_{n}\right): 1 \leq n \leq T\right\}$.
In the next subsection, we shall investigate a specific example of a PRNG.

### 3.3 The Linear Congruential Generator

A simple and yet rather powerful class of pseudorandom number generators is the linear congruential generators (LCG).

The method is based on modular arithmetic, i.e. congruences, hence its name.
We need two parameters for the method:

1. a large natural number $m$ (called the modulus),
2. a natural number $a$ (called the multiplier) such that $\operatorname{gcd}(m, a)=1$.

The set of states is $\mathcal{S}=\mathbb{Z}_{m}=\{0,1, \ldots, m-1\}$, that is, the residues $(\bmod m)$.
The state function $f: \mathbb{Z}_{m} \rightarrow \mathbb{Z}_{m}$ is defined as

$$
z_{n+1}=f\left(z_{n}\right)=a z_{n} \quad(\bmod m)
$$

and the output function $g$ is given by

$$
y_{n}=g\left(z_{n}\right)=\frac{z_{n}}{m} .
$$

The seed can be any $z_{0} \in \mathbb{Z}_{m}$ with $\operatorname{gcd}\left(z_{0}, m\right)=1$. The pseudorandom sample $\left\{y_{n}\right\}$ is determined by $m, a, z_{0}$. Hence, we denote by

$$
\operatorname{LCG}\left(m, a, z_{0}\right):=\left\{y_{n}\right\}
$$

Sometimes one takes $m$ to be prime, then any non-zero residue $\{1,2, \ldots, m-1\}$ can be used for multiplier and seed.
The size of the pseudorandom sample $\left\{y_{n}\right\}$ is simply the period of the LCG. Hence, it is interesting to calculate this quantity.

Proposition 9. The period of $\operatorname{LCG}\left(m, a, z_{0}\right)$ is the least natural number $h$ such that $a^{h} \equiv 1(\bmod m)$. (The number $h$ is called the multiplicative order of $a(\bmod m)$.)

Bevis. Iterating $\operatorname{LGC}\left(m, a, z_{0}\right)$ would yield

$$
\begin{array}{r}
a z_{0}=\beta \bmod m \\
a \beta=\gamma \bmod m \\
\vdots \\
a \eta=z_{0} \quad \bmod m .
\end{array}
$$

Since we are dealing with modulo we are dealing with periodicity, hence after $h$ iterations $a \eta=z_{0} \bmod m$. Since $\operatorname{gcd}(m, a)=1$ we have that $a^{h}=1 \bmod m$.

Remark 10. In particular, if $m=p$, where $p$ is a prime, then the period of $\operatorname{LCG}\left(p, a, z_{0}\right)$ is $\leq p-1$ for any $a \in \mathbb{Z}_{p} \backslash\{0\}$. If $a$ is such that the order $h$ of $a$ is $p-1$, then $a$ is called a primitive root modulo $p$.

### 3.4 Discrepancy estimates for the LCG

### 3.4.1 Discrepancy as quality measure

A fundamental question at this point is: what is the quality of the pseudorandom numbers generated by the LCG? Expressed differently, in which manner does $\operatorname{LCG}\left(m, a, z_{0}\right)=\left\{y_{n}\right\}$ behave like a random sample from $U([0,1])$ ? (This question was alluded to at the beginning of Section 3.)

It turns out that certain aspects of quality is captured by the discrepancy. The discrepancy can be thought of as a measure of how much a set of points deviate from the uniform
distribution. Hence, a small discrepancy would indicate that the point set in question is in some sense close to having uniform distribution.

In Section 4, we shall provide discrepancy charts for some particular implementations of LCG.

We mention here that discrepancy as a quality measure of pseudorandom numbers were previously also studied numerically in [2] before.

### 3.4.2 Non-asymptotic behaviour of discrepancy

It is important to stress the following point. For a sequence $S$, the asymptotic behaviour of $D_{K}(S)$ as $k \rightarrow \infty$ contains information of how uniformly distributed $S$ is. However, the set $\operatorname{LCG}\left(m, a, z_{0}\right)$ is always finite, with $\sharp\left(\operatorname{LCG}\left(m, a, z_{0}\right)\right)=T$, the period of the PRNG. For $k>T, D_{k}\left(\operatorname{LCG}\left(m, a, z_{0}\right)\right)$ provides nothing of interest and we cannot discuss the asymptotic behavior of the discrepancy of $\operatorname{LCG}\left(m, a, z_{0}\right)$

Nevertheless, discrepancy estimates contain some useful information. The next theorem is proved (in a more precise form) in [1].

Theorem 11. Let $S=\operatorname{LCG}\left(p, a, z_{0}\right)$ with $p \geq 3$ a prime number and arbitray $a, z_{0} \in$ $\mathbb{Z}_{p} \backslash\{0\}$. For $1 \leq k<T$,

$$
\begin{equation*}
D_{k}(S) \leq C \frac{\sqrt{p} \ln (p)^{2}}{k} \tag{10}
\end{equation*}
$$

We shall not give the proof of the above theorem, it is rather complicated. But we shall discuss the estimate (10). Say that we choose $a$ to be a primitive root modulo $p$, then $T$ is maximal, i.e. $T=p-1$. In applications, one typically does not use the full set $S$ but rather some $S^{\prime} \subset S$ whose size is a small fraction of $S$, say $\sharp\left(S^{\prime}\right)=(p-1) / 100$. Taking $k=\sharp\left(S^{\prime}\right)$, we have

$$
D_{k}\left(S^{\prime}\right)=D_{k}(S) \leq \frac{C \ln (k)^{2}}{k^{1 / 2}}
$$

for an absolute constant $C$. This agrees more or less with typical results for actual random sequences (see e.g. [1, Section 5.1.2]).
We shall illustrate these results numerically in Section 4 below.

## 4 Numerical experiments

### 4.1 Implementation of LCG

Example 12. We shall start with a small example of pseudorandom numbers generated with LCM. We take the modulus $m=31$ and multiplier $a=3$. It turns out that the multiplicative order of $3(\bmod 31)$ so $\sharp\left(\operatorname{LCM}\left(31,3, z_{0}\right)\right)=30$ for any seed $z_{0}$. Taking $z_{0}=2$ gives the result of Figure 1 .


Figur 1: Plot of pseudorandom numbers generated with LCG.

Note that the location of $y_{n} \in[0,1]$ is on the $y$-axis, on the $x$-axis are the integers $n=1,2, \ldots 30$.

The source code is located in Appendix.

### 4.2 Computation of discrepancy of sets generated by LCM

We shall investigate numerically the discrepancy of $\operatorname{LCG}\left(m, a, z_{0}\right)$. Observe that Lemma 8 allows us to calculate the discrepancy $D_{k}(S)$ : let $S_{k}$ be the first $k$ terms of $S$ and note that $D_{k}(S)=D_{k}\left(S_{k}\right)$. The discrepancy $D_{k}\left(S_{k}\right)$ can be obtained by ordering the elements of $S_{k}$ and calculate the expression (9).

Example 13. Consider first $S=\operatorname{LCG}(2203,3,1)$. The modulus $m=2203$ is a prime and the multiplier $a=3$ has is a primitive root modulo 2203. Using the implementation of Lemma 8, we calculate $D_{k}(S)$ for $1 \leq k \leq 2202$ and plot the result in Figure 2 below.


Figur 2: Plot of numbers generated by LGC with $\mathrm{m}=2203$, $\mathrm{a}=3$, seed $=1$.

Note that the shape of the graph is more or less in accordance with Theorem 10.
We consider another example with $S=\operatorname{LCG}(4253,5,2)$ (again prime modulus and a multiplier that is a primitive root modulo 4253). The result is shown in Figure 3.


Figur 3: Plot of numbers generated by LGC with $m=4253$, $a=5$, seed $=2$.

### 4.3 Inversive transform sampling

Example 14. We calculate the 100 pseudorandom numbers of $\operatorname{LCG}(103,2,1)$.


Figur 4: 100 samples $y_{n}$ generated from the uniform distribution.

Note that it seems quite likely that the mean value of the numbers are roughly 0.5 , which agrees with the fact that $E(Y)=0.5$ if $Y \sim U([0,1])$.

We shall show how to generate pseudorandom numbers from $\operatorname{Exp}(1)$ by using the fact that if $X \sim \operatorname{Exp}(1)$, then $Q_{X}(x)=-\ln (1-x)$. Hence, setting $x_{n}=-\ln \left(1-y_{n}\right)$ and plotting the result, we get the following figure.


Figur 5: Empirical distribution and cumulative distribution function of $\exp (1)$

It's easy to recognize the exponential distribution from Figure 5 and you can clearly see that the empirical distribution function follows the same pattern as the cumulative distribution function. Note however that it is not unlikely that the mean value of the sample $=1$, in accordance with the fact that $E(X)=1$ if $X \sim \operatorname{Exp}(1)$.

## 5 Appendix: Python Code

```
import math
from random import sample
from tkinter import Y
import numpy as np
from scipy.stats import expon
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from statsmodels.distributions.empirical_distribution import ECDF
print("-"*50)
print("Generates }\mp@subsup{\mp@code{a}}{\sqcup}{\prime}\mp@subsup{\mathrm{ sequence }}{\sqcup\mathrm{ of }}{\sqcup
print("-"*50)
print("\n")
#ecdf = ECDF([3, 3, 1, 4])
#print("ECDF", ecdf([3, 55, 0.5, 1.5]))
## LGC - Linear Congruential pseudo number generator /
#
# Description:
# LCG will generate a pseudo random numbers based on linear congruence me
#Choose a large m and an a such that gcd (m,a)=1.
def lgc(m,a,seed):
    x_n= []
    k=m-1
    z0 = (a*seed) % m
    z_n=[]
    z_n.append(z0)
    j=1
    while j < m:
        if (pow(a, j) % m) == 1:
                print("The}\sqcup\mathrm{ period }\sqcup\mathrm{ of }\llcorner\textrm{x}_\mp@subsup{n}{\sqcup}{\prime}\mathrm{ is : : {} ".format ( j ))
                break
                else:
            j+=1
    for i in range(1, j):
        z_n.append(a*z_n[i-1] %m)
        x_n.append (z_n[i-1]/m)
    return x_n
#Prints the generated list of chosen m, a and initial seed.
#print("The generated sequence x_n=", lgc(101,2,2))
```

\#Calls the function LCG with parameters modulus (m), multiplier (a) and s
$\# b=\operatorname{lgc}(1279,3,1)$
$\mathrm{b}=\lg \mathrm{c}(103,2,2)$
\#plt. plot(b, 'ro')
plt.grid()
plt.xlabel('n')
plt. ylabel ('Generated $\left\llcorner\right.$ number $_{\sqcup}$ value')

```
#plt.show()
```

```
#Inverse Sampling Exp.
def inv(LISTA,gamma):
    x = []
    for i in range(len(LISTA)):
            x.append ((-1/gamma)*np.log(1-LISTA[i]))
    return x
```

```
sample1 = inv(b,1)
#print("New list =", inv(b,1))
ecdf = ECDF(sample1)
print("x", ecdf.x)
plt.plot(ecdf.x, ecdf.y, label= 'ECDF&of
x = np.arange(0, 3.5, 0.1)
y = expon.cdf(x, 0, 1)
plt.plot(x, y, label= 'CDF\sqcupof 
#plt.plot(inv(b,1),"bo")
plt.grid()
plt.xlabel('x')
plt.ylabel('Cumulative}\sqcup\mathrm{ Probability')
plt.legend()
plt.grid()
plt.show()
```

```
\#—————— Discrepancy calculator ———
\#Calculates discrepancy with the help of proposition below.
\#Proposition 4.1.16
\#
\# Let \(P\) be a point set \(x 1, x 2, \ldots x n\) from \([0,1)\) and suppose \(x 1<=x 2<=x 3\)
\(\# D \_N(P)=1 / 2 N+\max \left(x \_n-(2 n-1) / 2 N\right)\). Here \(|P|=N\).
\#
```



```
def Discrepancy(list):
    \(\mathrm{N}=\operatorname{len}(\mathrm{list})\)
        list.sort()
        disc \(=[]\)
        for \(i\) in range(len(list)):
            disc.append (list[i] \(-(2 *(i+1)-1) /(2 * N))\)
        MAX \(=(1 /(2 * \mathrm{~N})+\max (\) disc \())\)
        \#print (MAX)
        return MAX
```

\# Prints the discrepancy of choosen list
\#print("The discprenacy of $x \_n="$, Discrepancy(b))
print ("-"*50)
\#print("Discrepancy $="$, disc)

```
\#Calls LGC and Discrepancy to calculate \(j\) discrepancies of the generated
\(\# L=\lg c(8191,3,2)\)
\(\mathrm{D}=[]\)
\(\mathrm{X}=[]\)
\#for \(j\) in range (2,8191,20):
    \# L2=L[0:j]
    \# X.append (j)
    \# D.append (Discrepancy(L2))
\#plt. plot \((X, D)\)
\#plt.text (5,0.001,'prime used = 1279')
\#plt.plot(n_value, disc, 'bo')
    plt. xlabel ('Index \(\sqcup\) of discrepancy \(^{\prime}\) )
    plt.ylabel ('Discrepancy \(\lrcorner v a l u e ')\)
    plt.grid ()
\#plt.show()
```


## Referenser

[1] Niederreiter, H. and Winterhof, A. Applied Number Theory. 185-363. Springer (2015).
[2] V. C. Bhavsar, U. G. Uday, J. D. Horton, L. Lambreau. Evaluation of the discrepancy of the linear congruental pseudo-random number sequence.
BIT 30 (1990), no. 2, 258-267.
[3] Kuipers, L. and Niederreiter, H. Uniform Distribution of Sequences. 88-170. John Wiley \& Sons, Inc. (1974).

