blatt03_Guth_Venker_Jaekel

May 18, 2021

1 Abgabe SMD Blatt 03

1.0.1 von Nico Guth, David Venker, Jan Jäkel

[1]:
```
class PDF(object):
    '''
    Displays PDF files in Jupyter Notebooks
    '''
    def __init__(self, pdf, size=(200,200)):
        self.pdf = pdf
        self.size = size

    def _repr_html_(self):
        return '<iframe src={0} width={1[0]} height={1[1]}></iframe>'.
↪format(self.pdf, self.size)

    def _repr_latex_(self):
        return r'\includegraphics[width=1.0\textwidth]{{{0}}}'.format(self.pdf)
```

2 Aufgabe 5

2.0.1 b)

[2]:
```
import numpy as np
from project_b4.random import LCG
```

[3]:
```
lcg_default = LCG()
print(lcg_default.random_raw(10))
```

```
[1013904223 1196435762 3519870697 2868466484 1649599747 2670642822
 1476291629 2748932008 2180890343 2498801434]
```

[4]:
```
a = range(1,10)
for a in a:
    lcg = LCG(seed=0,a=a,c=3,m=1024)
    numbers = lcg.random_raw(size=10000)
    period_length = np.diff(np.where(numbers == numbers[0])[0])
    print(f'a = {a}')
```

```python
    print(f'numbers = {numbers}')
    print(f'period length = {period_length}')
    print()
```

```
a = 1
numbers = [  3   6   9 … 298 301 304]
period length = [1024 1024 1024 1024 1024 1024 1024 1024 1024]

a = 2
numbers = [   3    9   21 … 1021 1021 1021]
period length = []

a = 3
numbers = [  3  12  39 … 820 415 224]
period length = [512 512 512 512 512 512 512 512 512 512 512 512 512 512 512 512
512 512
 512]

a = 4
numbers = [   3   15   63 … 1023 1023 1023]
period length = []

a = 5
numbers = [  3  18  93 … 110 553 720]
period length = [1024 1024 1024 1024 1024 1024 1024 1024 1024]

a = 6
numbers = [  3  21 129 … 409 409 409]
period length = []

a = 7
numbers = [  3  24 171 … 808 539 704]
period length = [256 256 256 256 256 256 256 256 256 256 256 256 256 256 256 256
256 256
 256 256 256 256 256 256 256 256 256 256 256 256 256 256 256 256 256
 256 256 256]

a = 8
numbers = [  3  27 219 … 731 731 731]
period length = []

a = 9
numbers = [  3  30 273 … 178 581 112]
period length = [1024 1024 1024 1024 1024 1024 1024 1024 1024]
```

[5]:
```python
a2 = range(1,1000)
a_max_period_length = []
for a in a2:
    lcg = LCG(seed=0,a=a,c=3,m=1024)
    numbers = lcg.random_raw(size=10000)
    period_length = np.diff(np.where(numbers == numbers[0])[0])
    if 1024 in period_length:
        a_max_period_length.append(a)

print(f'maximal period length for a in \n {a_max_period_length}')
```

```
maximal period length for a in
 [1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 73, 77,
81, 85, 89, 93, 97, 101, 105, 109, 113, 117, 121, 125, 129, 133, 137, 141, 145,
149, 153, 157, 161, 165, 169, 173, 177, 181, 185, 189, 193, 197, 201, 205, 209,
213, 217, 221, 225, 229, 233, 237, 241, 245, 249, 253, 257, 261, 265, 269, 273,
277, 281, 285, 289, 293, 297, 301, 305, 309, 313, 317, 321, 325, 329, 333, 337,
341, 345, 349, 353, 357, 361, 365, 369, 373, 377, 381, 385, 389, 393, 397, 401,
405, 409, 413, 417, 421, 425, 429, 433, 437, 441, 445, 449, 453, 457, 461, 465,
469, 473, 477, 481, 485, 489, 493, 497, 501, 505, 509, 513, 517, 521, 525, 529,
533, 537, 541, 545, 549, 553, 557, 561, 565, 569, 573, 577, 581, 585, 589, 593,
597, 601, 605, 609, 613, 617, 621, 625, 629, 633, 637, 641, 645, 649, 653, 657,
661, 665, 669, 673, 677, 681, 685, 689, 693, 697, 701, 705, 709, 713, 717, 721,
725, 729, 733, 737, 741, 745, 749, 753, 757, 761, 765, 769, 773, 777, 781, 785,
789, 793, 797, 801, 805, 809, 813, 817, 821, 825, 829, 833, 837, 841, 845, 849,
853, 857, 861, 865, 869, 873, 877, 881, 885, 889, 893, 897, 901, 905, 909, 913,
917, 921, 925, 929, 933, 937, 941, 945, 949, 953, 957, 961, 965, 969, 973, 977,
981, 985, 989, 993, 997]
```

- Bei LCG ist die maximale Periodenlänge m.
- Aus dem Code: die maximale Periodenlänge ist erreicht bei z.B. $a \in 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, ...$

- Allgemein: die maximale Periodenlänge wird erreicht wenn (aus der Vorlesung):
  - $c \neq 0$
  - c und m teilerfremd
  - Jeder Primfaktor von m teilt $(a - 1)$
  - Wenn $m \bmod 4 = 0$ ist auch $(a - 1) \bmod 4 = 0$
- Die genannten Bedingungen stimmen bei dem hier verwendeten Beispiel immer dann wenn $(a - 1) \bmod 4 = 0$, da 2 der einzige Primfaktor von $m = 1024$ ist.

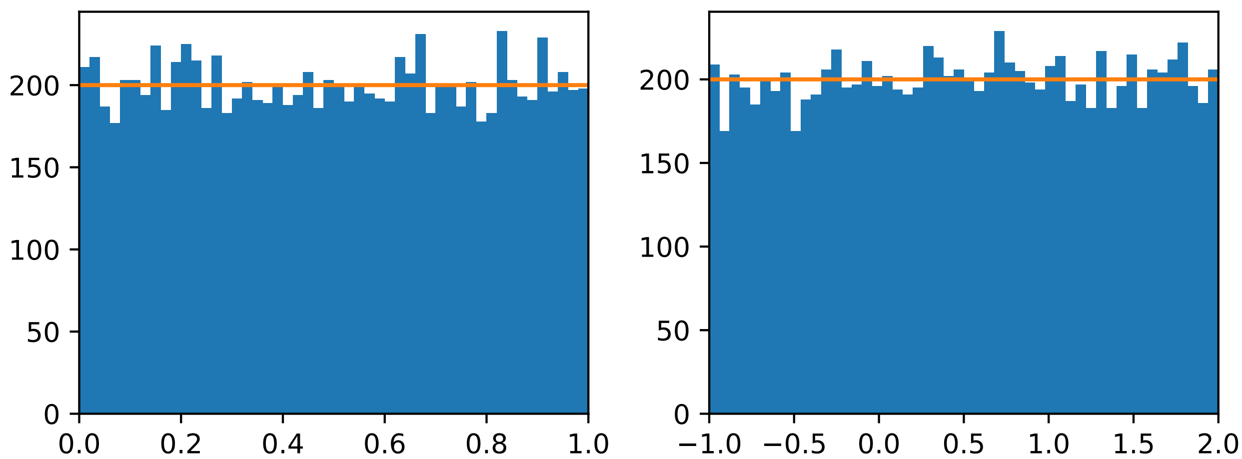
2.0.2 c)

[6]:
```python
PDF('lcg.pdf',size=(650,550))
```

[6]:

```
Information:
  Exercise: Linear-Kongruent
  Group name: project_b4

Tests:
  m=16:              True
  std. uniform:      True
  uniform [-1, 2]: True
```



2.0.3 d)

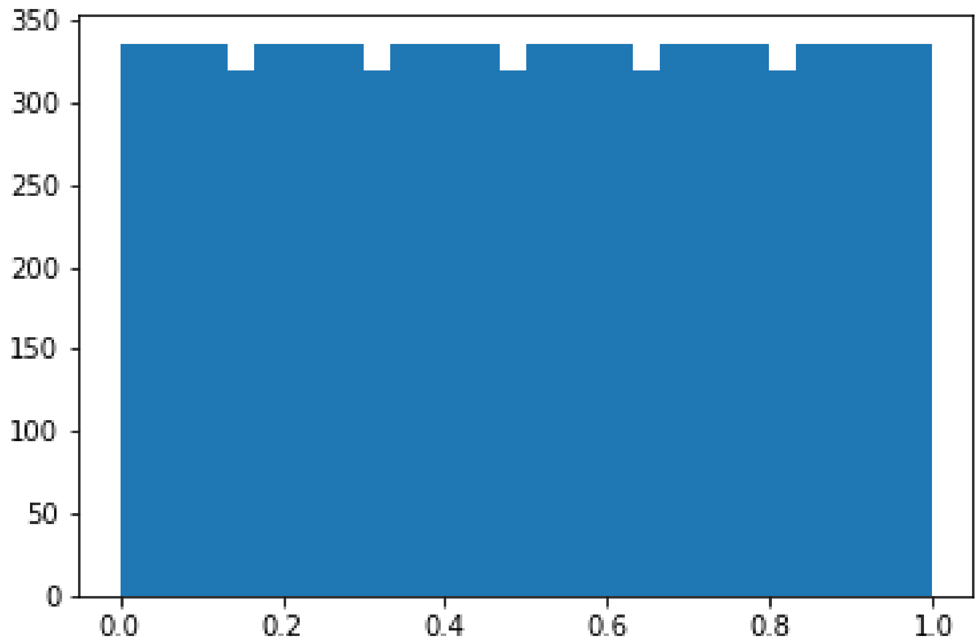
[7]:
```python
lcg = LCG(seed=0,a=1601,c=3456,m=10000)
```

[8]:
```python
numbers_lcg = lcg.uniform(low=0,high=1,size=10000)
numbers_lcg
```

[8]:
```
array([0.3456, 0.6512, 0.9168, …, 0.1888, 0.6144, 0.    ])
```

[9]:
```python
import matplotlib.pyplot as plt
%matplotlib inline
```

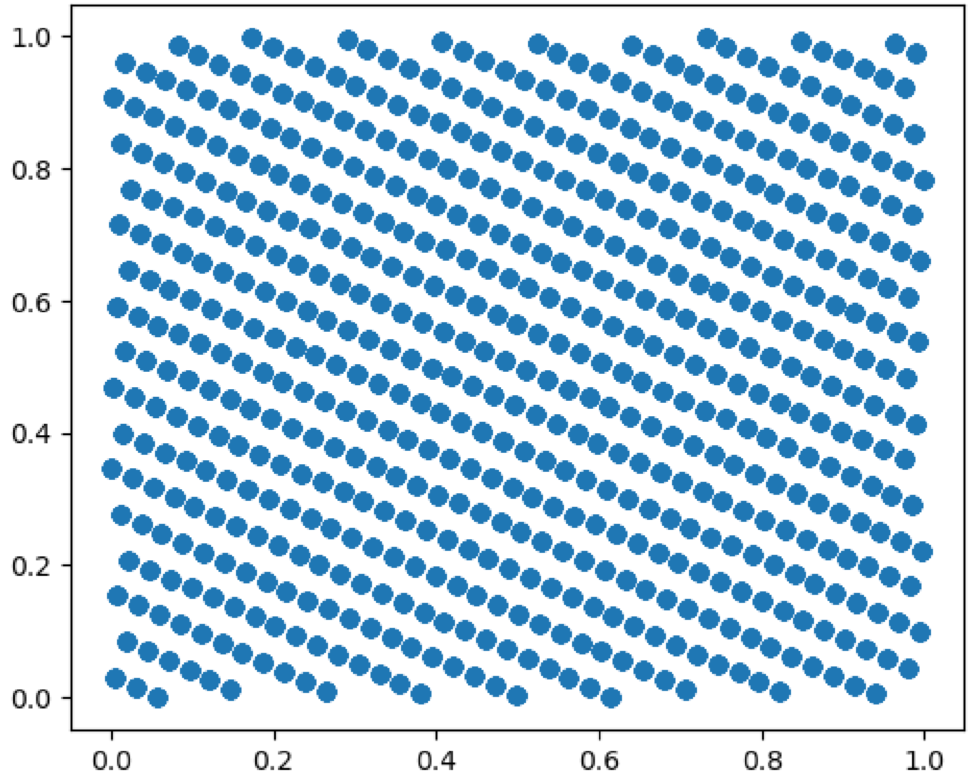
[10]:
```python
plt.hist(numbers_lcg,bins=30);
```



Im Histogramm sieht man, dass die Zahlen gut gleichverteilt sind. Aber man erkennt auch schon eine Regelmäßigkeit. Für alle Seeds sollte es gleich funktionieren. x_0 bestimmt nur den Startpunkt in der Periode, aber das Verhalten bleibt im groben gleich.

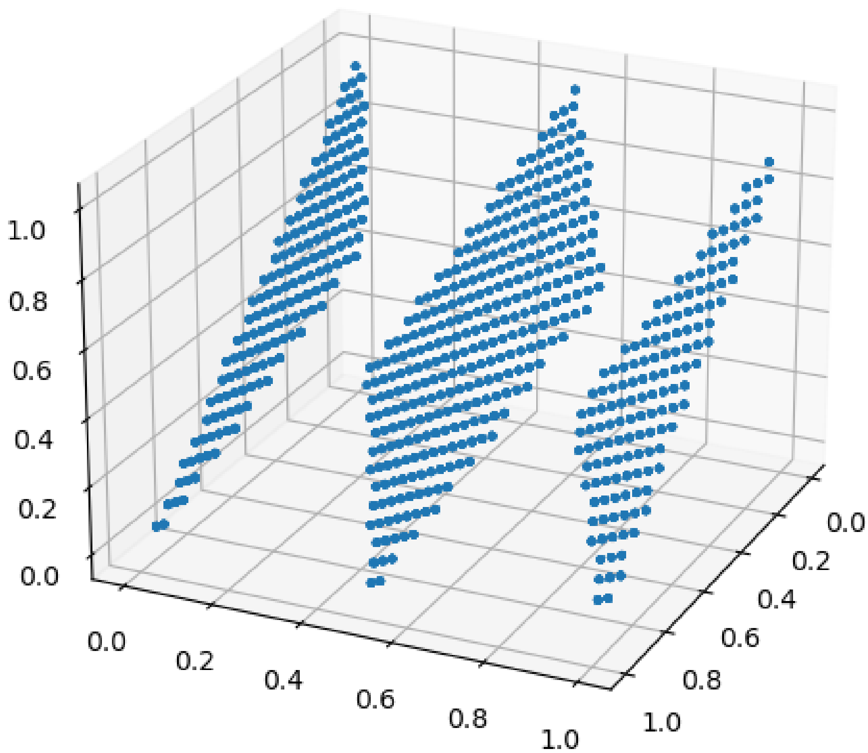
2.0.4 e)

[11]:
```
plt.figure(figsize=(6,5),dpi=100)
plt.scatter(numbers_lcg[:-1],numbers_lcg[1:]);
# (x1,x2),(x2,x3)...(xn-1,xn)
```



Hier sieht man eine regelmäßige und diskrete Verteilung. Das ist bei einem Zufallsgenerator nicht erwünscht.

[12]:
```
fig = plt.figure(figsize=(8,6),dpi=100)
ax = fig.add_subplot(projection='3d')
ax.scatter(numbers_lcg[:-2],numbers_lcg[1:-1],numbers_lcg[2:],s=5)
ax.view_init(elev=23, azim=25)
plt.show()
```

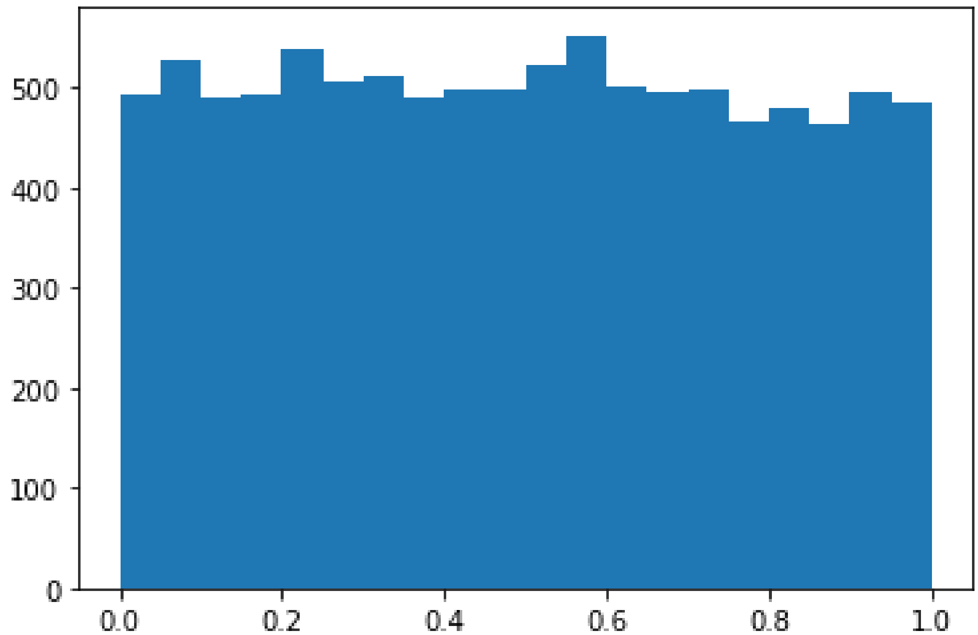


Im 3d Spektraltest sieht man, dass viele Triplets nicht erzeugt werden. Und deswegen der Generator doch nicht so zufällig ist.

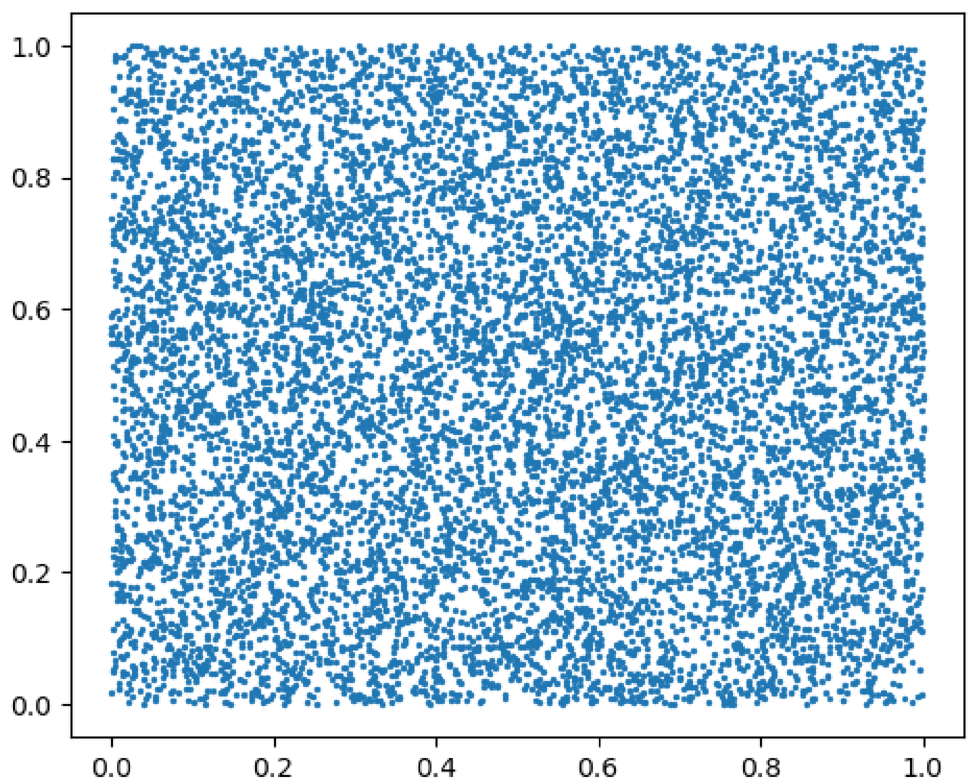
2.0.5 f)

[13]:
```
rng = np.random.default_rng()
numbers_np = rng.uniform(0,1,size=10000)
```

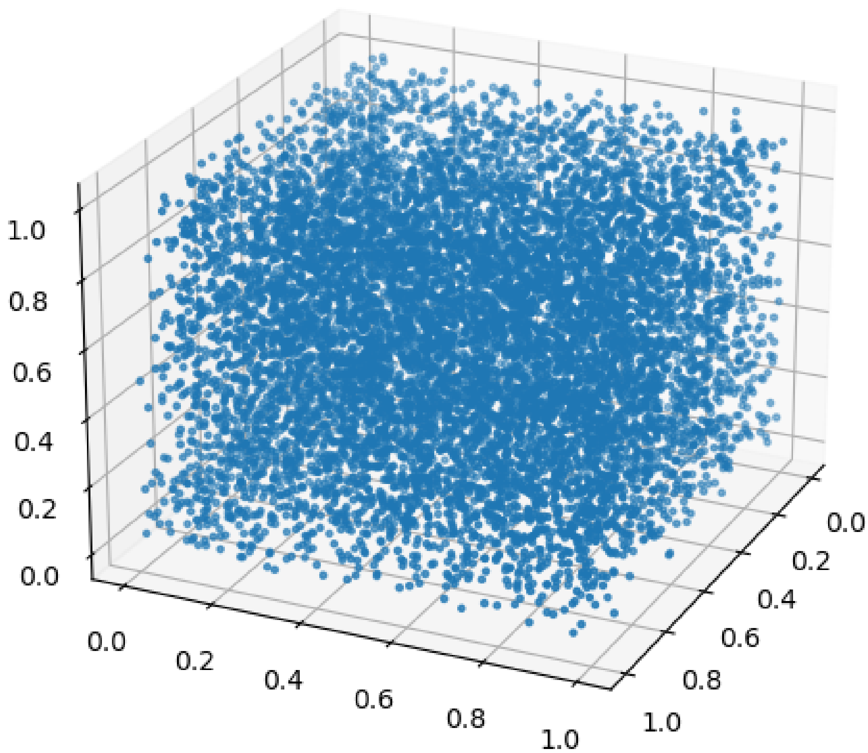
[14]:
```
plt.hist(numbers_np,bins=20);
```



[15]:
```
plt.figure(figsize=(6,5),dpi=100)
plt.scatter(numbers_np[:-1],numbers_np[1:],s=2);
```



[16]:
```
fig = plt.figure(figsize=(8,6),dpi=100)
ax = fig.add_subplot(projection='3d')
ax.scatter(numbers_np[:-2],numbers_np[1:-1],numbers_np[2:],s=5)
ax.view_init(elev=23, azim=25)
plt.show()
```



Der Numpy Random Number Generator ist deutlich besser verteilt.
Allerdings ist beim Histogram die Anzahl der Werte nicht ganz so gleichmäßig verteilt.

3 Aufgabe 6

3.0.1 Vorgehensweise:

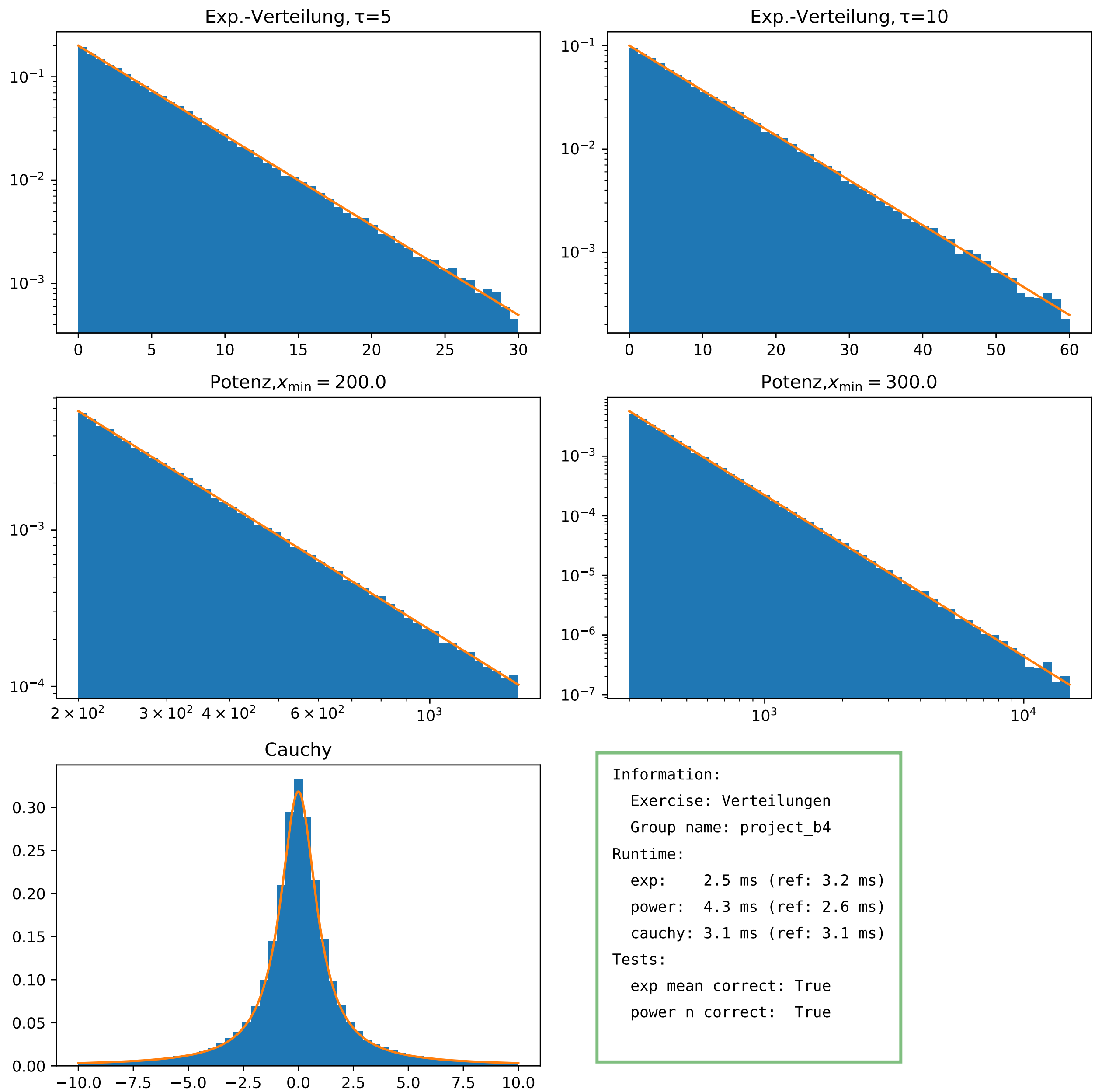
- Bestimme die Normierungskonstante N der Wahrscheinlichkeitsdichte $f(x)$
- Bestimme die Verteilungsfunktion $F(x)$ der gegebenen Wahrscheinlichkeitsdichte $f(x)$

- Bestimme die Umkehrfunktion $F^{-1}(y)$ der Verteilungsfunktion $F(x)$
- Wenn y nun gleichverteilte Zufallszahlen zwischen 0 und 1 sind, dann erzeugt $F^{-1}(y)$ Zufallszahlen die der Wahrscheinlichkeitsdichte $f(x)$ entsprechen

Die Rechnungen sind dem PDF angehangen.

[17]:
```
PDF('distributions.pdf',size=(650,650))
```

[17]:



[ ]:
```
```

Aufgabe 6

a) $f(x) = N \cdot \exp(-x/\tau) \qquad x \in [0, \infty)$

Normieren:

$$\int_0^\infty f(x)\,dx \overset{!}{=} 1$$

$$= \int_0^\infty N \exp(-x/\tau)\,dx$$

$$= N \left[(-\tau)\exp(-x/\tau)\right]_0^\infty$$

$$= N(-\tau)[0-1]$$

$$= N\tau \overset{!}{=} 1$$

$$\Leftrightarrow \quad N = \frac{1}{\tau}$$

Verteilungsfunktion:

$$F(x) = \int_0^x f(x)\,dx$$

$$= \int_0^x \frac{1}{\tau} \exp\left(-\frac{x}{\tau}\right)dx$$

$$= \frac{1}{\tau}\left[(-\tau)\exp\left(-\frac{x}{\tau}\right)\right]_0^x$$

$$= (-1)\left[\exp\left(-\frac{x}{\tau}\right) - 1\right]$$

$$= 1 - \exp\left(-\frac{x}{\tau}\right)$$

Invertieren:

$$F(x) = 1 - \exp\left(-\frac{x}{\tau}\right) = y(x)$$

$$\Leftrightarrow \exp\left(-\frac{x}{\tau}\right) = 1 - y$$

$$\Leftrightarrow -\frac{x}{\tau} = \ln(1-y)$$

$$\Leftrightarrow x = (-\tau)\ln(1-y)$$

$$\Rightarrow F^{-1}(y) = (-\tau)\ln(1-y) \qquad y \in [0,1)$$

b)

$$f(x) = N \cdot x^{-n} \qquad x \in [x_{min}, x_{max}] \qquad n \geq 2$$

Normierung:

$$\int_{x_{min}}^{x_{max}} f(x)\,dx \overset{!}{=} 1$$

$$= \int_{x_{min}}^{x_{max}} N \cdot x^{-n}\,dx$$

$$= N \cdot \left[\frac{1}{(-n+1)} x^{-n+1}\right]_{x_{min}}^{x_{max}}$$

$$= N \frac{1}{(1-n)} \cdot \left[x_{max}^{1-n} - x_{min}^{1-n}\right] \overset{!}{=} 1$$

$$\Leftrightarrow N = (1-n)\left(x_{max}^{1-n} - x_{min}^{1-n}\right)^{-1}$$

Verteilungsfunktion:

$$F(x) = \int_{x_{min}}^{x} f(x)\,dx$$

$$= \int_{x_{min}}^{x} N x^{-n}\,dx$$

$$= N \frac{1}{(-n+1)} \left[x^{-n+1} \right]_{x_{min}}^{x}$$

$$= N \frac{1}{(1-n)} \left[x^{-n+1} - x_{min}^{-n+1} \right]$$

Invertierung:

$$F(x) = N \frac{1}{(1-n)} \left(x^{1-n} - x_{min}^{1-n} \right) = y(x)$$

$$\Leftrightarrow \quad x^{1-n} - x_{min}^{1-n} = N^{-1} (1-n)\, y$$

$$\Leftrightarrow \quad x^{1-n} = N^{-1} (1-n)\, y + x_{min}^{1-n}$$

$$\Leftrightarrow \quad x = \left(N^{-1} (1-n)\, y + x_{min}^{1-n} \right)^{\frac{1}{1-n}}$$

$$\Rightarrow \quad F^{-1}(y) = \left(N^{-1} (1-n)\, y + x_{min}^{1-n} \right)^{\frac{1}{1-n}} \qquad y \in [0,1)$$

c)

$$f(x) = \frac{1}{\pi} \frac{1}{1+x^2} \qquad x \in \mathbb{R}$$

schon normiert

Verteilungsfunktion:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\pi} \frac{1}{1+x^2}\, dx$$

$$= \frac{1}{\pi} \int_{-\infty}^{x} \frac{1}{1+x^2}\, dx$$

$$= \frac{1}{\pi} \left[\arctan(x)\right]_{-\infty}^{x}$$

$$= \frac{1}{\pi}\left[\arctan(x) + \frac{\pi}{2}\right]$$

$$= \frac{1}{\pi} \arctan(x) + \frac{1}{2}$$

Invertieren:

$$F(x) = \frac{1}{\pi} \arctan(x) + \frac{1}{2} = y(x)$$

$$\Leftrightarrow \quad \arctan(x) = \pi\left(y - \frac{1}{2}\right)$$

$$x = \tan\left(\pi\left(y - \frac{1}{2}\right)\right)$$

$$\Rightarrow F^{-1}(y) = \tan\left(\pi\left(y - \frac{1}{2}\right)\right) \qquad y \in [0, 1)$$