5 | 6 | 2

3,5/5 | 5/5 | 8,5/10

Sheet03

May 10, 2022

Programming Exercises:

The modified project code can be found in the branch: 'imsodone' in the project git lab.

Exercise 5

a)

[1]:
```
def advance(self):
        '''
        advance the state of this random generator by one step

        Blatt 3, Aufgabe 1
        '''

        # Fügen Sie hier Ihren Code ein, um den LCG korrekt zu implementieren

        self.state = (self.a * self.state + self.c) % self.m
```

b)

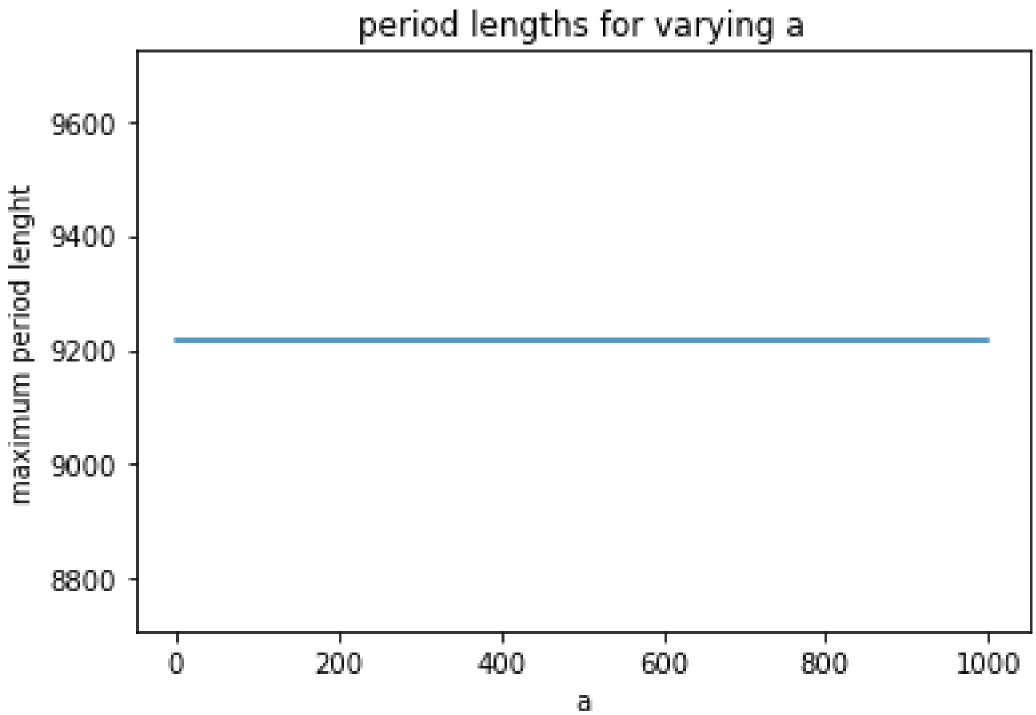
[2]:
```
from project_c3 import random
import numpy as np
import matplotlib.pyplot as plt


a_array = np.linspace(0, 1000, 100, dtype = int)

c = 3
m = 1024
amount_random_numbers = 10000


random_numbers = [random.LCG(a, c=c, m=m).random_raw(size =␣
 ↪amount_random_numbers) for a in a_array]


def max_len(arr):

    mp = {}
```

you miss important values for a here, use arange(1, 1024, 1)

```python
    maxDict = 0

    #creates a dictionary where the key is the element and the value is the␣
↪index of the element
    #if a double element occurs and its distance to the previous occurance is␣
↪the biggest yet,
    #we update the max distance

    for i in range(len(arr)):

        if arr[i] not in mp.keys():
            mp[arr[i]] = i

        else:
            maxDict = max(maxDict, i-mp[arr[i]])

    return maxDict

max_lengths = [max_len(random_numbers[s]) for s in range(len(random_numbers))]

plt.figure()
plt.plot(a_array, max_lengths)
plt.xlabel('a')
plt.ylabel('maximum period lenght')
plt.title('period lengths for varying a')

None
```

you have to find a min here. As soon as a random number repeats itself you have found the period-length.

See lecture for max. period lenght.



−1P

In this specific example, a does not have an influence on the period lenght

c)

[3]:
```
def uniform(self, low=0, high=1, size=None):
        '''
        Draw uniforn random numbers by converting them from the raw numbers'''
        raw = self.random_raw(size=size)

        # Fügen Sie hier ihren Code ein, um die rohen Zufallszahlen
        # zu kontinuierlich gleichverteilten Zufallszahlen zwischen
        # low und high zu transformieren

        u = np.divide(raw, self.m)

        result = u * (high - low)
        return result
```

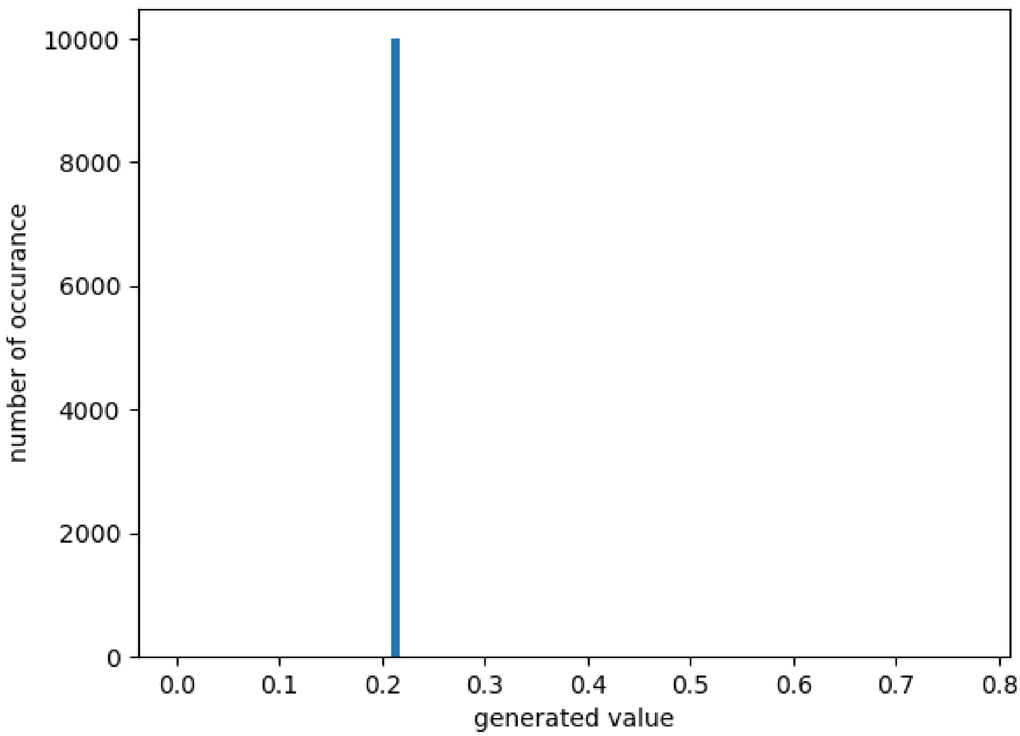
+ low

−0.5P

d)

[4]:
```
a = 1601
c = 3456
m = 10000
```

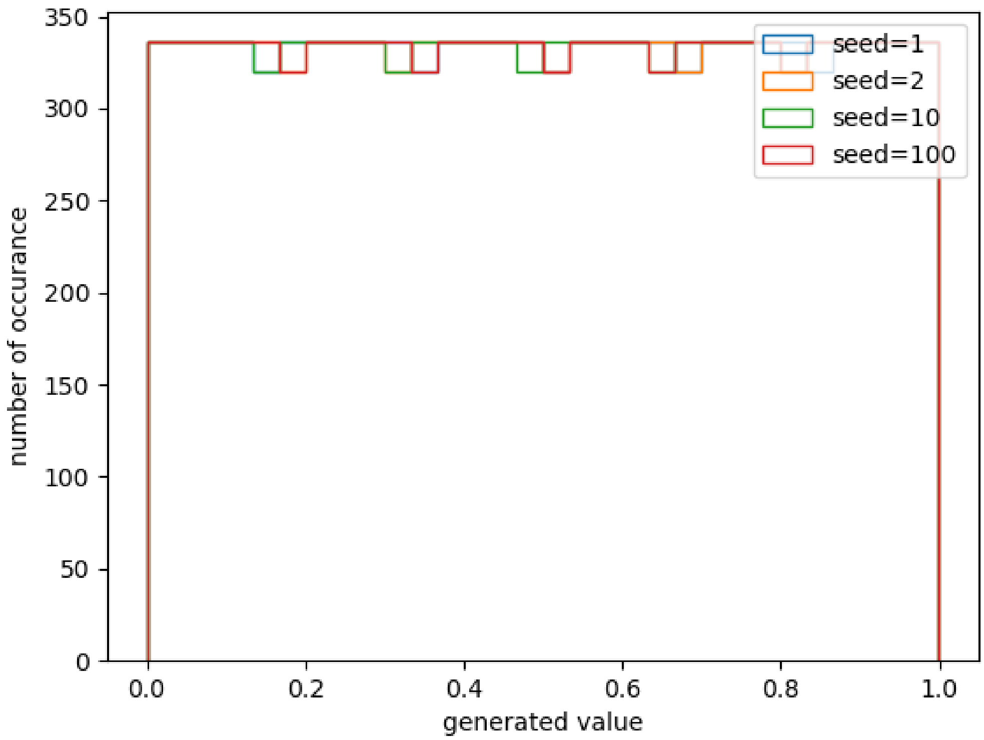
```
LCG_instance = random.LCG(a,c,m)

random_number_uniform = LCG_instance.uniform(low = 0, high = 1, size = 10000)

%matplotlib widget

plt.figure()
plt.hist(random_number_uniform, bins =100)
plt.xlabel('generated value')
plt.ylabel('number of occurance')

None
```



It does not meet the requirements, as the number 0.21562952 seems to generated unproportianaly often compared to the other numbers.

[5]:
```
seed = np.array([1, 2, 10, 100])

plt.figure()
for s in seed:
    LCG_instance = random.LCG(seed = s, a = a,c = c,m= m)
```

```
    random_number_uniform = LCG_instance.uniform(low = 0, high = 1, size =␣
↪10000)

    plt.hist(random_number_uniform, bins =30, label ='seed='+str(s),␣
↪histtype=u'step')
    plt.xlabel('generated value')
    plt.ylabel('number of occurance')
plt.legend()

None
```



The seed value has a big influence on the maximum period lenght, with different seeds the distrubitions become almost uniform. However one can still see repeating patterns. (Drops at regular intervals)

c)

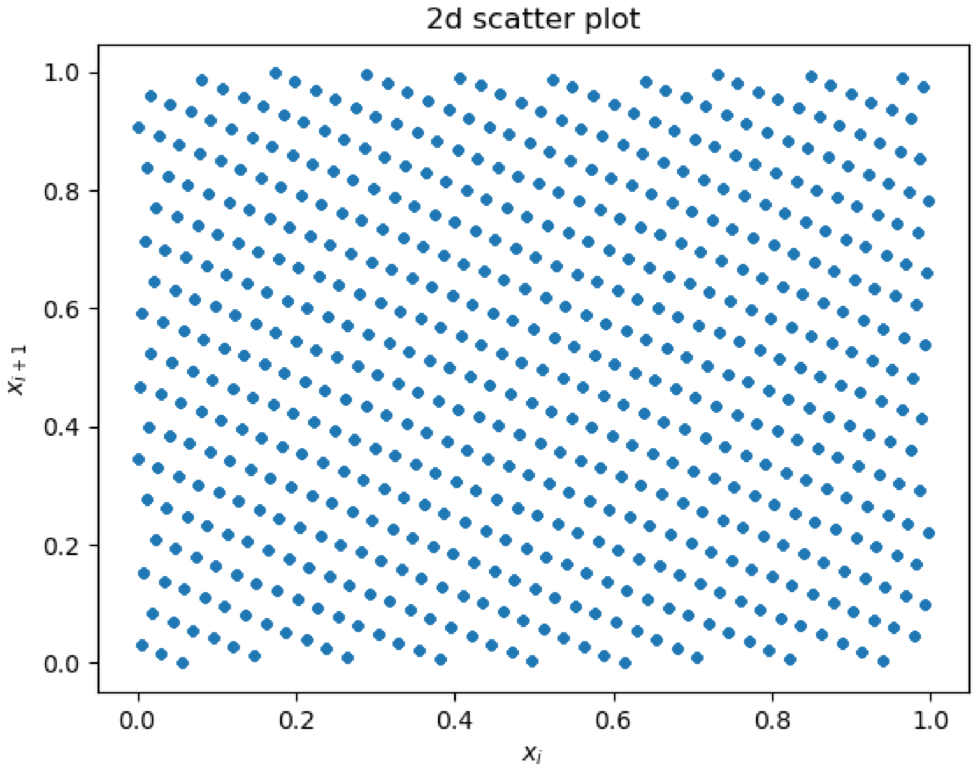
The lcg pdf shows that the implemented methods work (all Tests True), however one of the limits becomes apparent: As the lcg cannot generate negative numbers, the produced distributions will always have x >0.

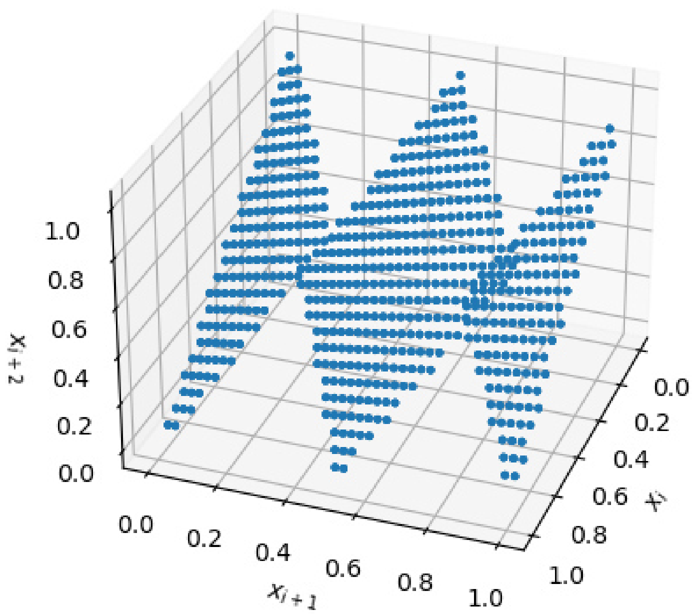
This is due to the small mistake in uniform().

e)

[6]:
```
LCG_instance = random.LCG(a = a,c = c,m= m)

random_number_uniform = LCG_instance.uniform(low = 0, high = 1, size = 10000)

plt.figure()
plt.plot(random_number_uniform[:-1], random_number_uniform[1:], linestyle =␣
 ↪'None', marker='.')
plt.xlabel(r'$x_i$')
plt.ylabel(r'$x_{i+1}$')
plt.title('2d scatter plot')

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection = '3d')

ax.scatter(
    random_number_uniform[:-2], random_number_uniform[1:-1],␣
 ↪random_number_uniform[2:],
    s=5,
    alpha=0.3,
    )

ax.view_init(elev=30, azim=20)
ax.set_xlabel(r'$x_i$')
ax.set_ylabel(r'$x_{i+1}$')
ax.set_zlabel(r'$x_{i+2}$')
```

```
None
```


2d scatter plot
x_{i+1}
x_i
0.0
0.2
0.4
0.6
0.8
1.0



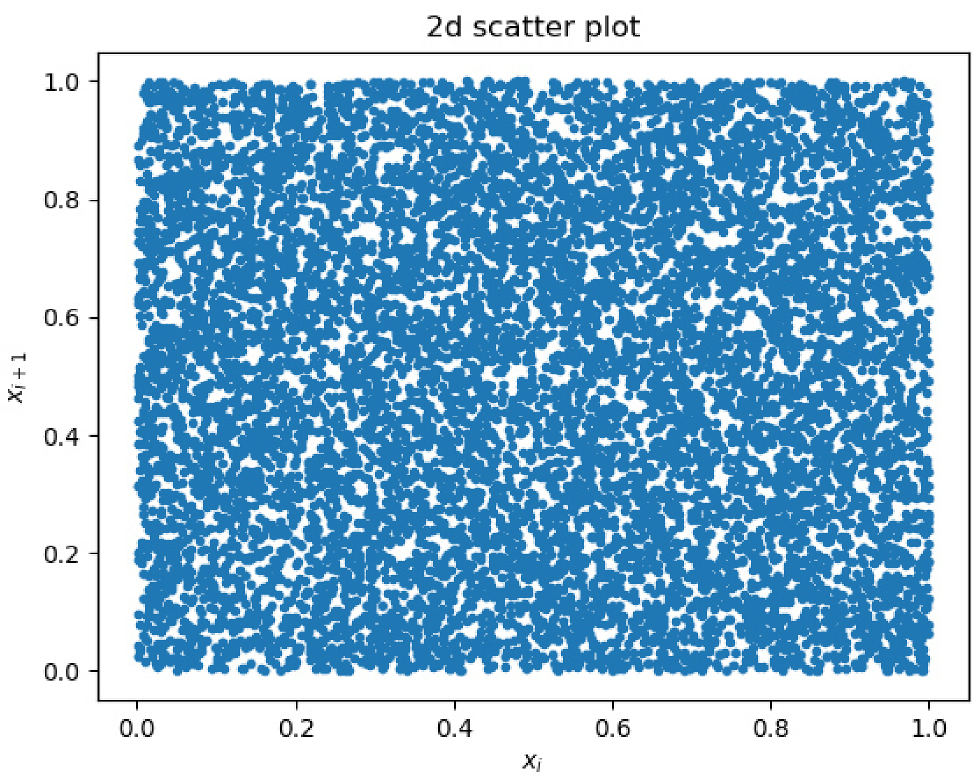
The result does not meet the requirements. In the 2D Plot you can clearly see many parallel lines, meaning many value-pairs occur

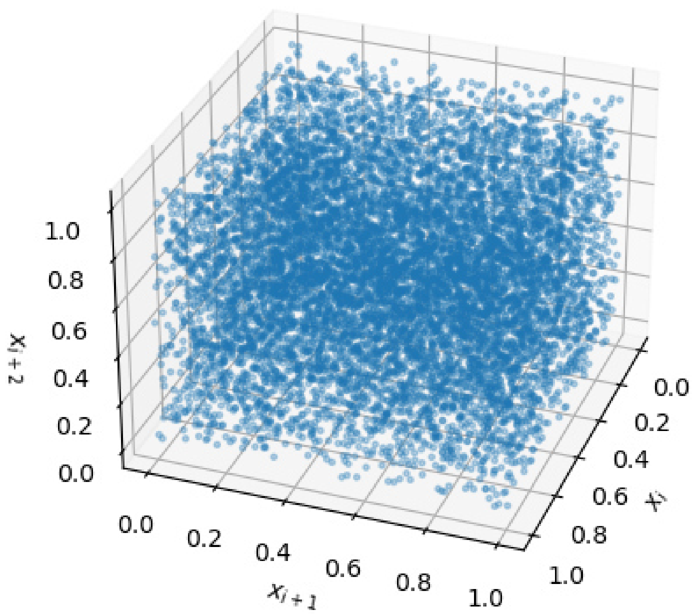
The 3d plots shows a similar result, as we can clearly see parallel planes, meaning also many triplets occur

f)

[7]:
```
rng = np.random.default_rng(0)

random_number_uniform = rng.random(10000)

plt.figure()
plt.plot(random_number_uniform[:-1], random_number_uniform[1:], linestyle =␣
  ↪'None', marker='.')
plt.xlabel(r'$x_i$')
plt.ylabel(r'$x_{i+1}$')
plt.title('2d scatter plot')

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection = '3d')
```

```
ax.scatter(
    random_number_uniform[:-2], random_number_uniform[1:-1],␣
 ↪random_number_uniform[2:],
    s=5,
    alpha=0.3,
    )

ax.view_init(elev=30, azim=20)
ax.set_xlabel(r'$x_i$')
ax.set_ylabel(r'$x_{i+1}$')
ax.set_zlabel(r'$x_{i+2}$')

None
```





The 2d and 3d plots show no apparent patterns and thus the generator meets the requirements for a good random number generator

3,5/5

Exercise 6:

Interpretation of the overview pdf:

The implemented distributions seem to work, as all the test show true. Also the fitted functions match the underlying distribution , hinting at a correct implementation. The plots of the exponential and power distribution appear linear due to chosen scale for the axis'

Code for Exercise 6:

[10]:
```
def exponential(self, tau, size=None):
        '''
        Draw exponentially distributed random numbers.

        Blatt 3, Aufgabe 2a)
        '''
        # So können Sie ein array mit shape=size
        # mit standard gleichverteilten Zufallszahlen erzeugen
        u = self.uniform(size=size)
```

```
        # Fügen Sie hier den Code ein um Zufallszahlen aus der
        # angegebenen Verteilung zu erzeugen

        # dummy, so the code works. Can be removed / replaced
        values = -tau*np.log(1-u)

        return values

def power(self, n, x_min, x_max, size=None):
    '''
    Draw random numbers from a power law distribution
    with index n between x_min and x_max

    Blatt 3, Aufgabe 2b)
    '''
    # Fügen Sie hier den Code ein um Zufallszahlen aus der
    # angegebenen Verteilung zu erzeugen

    # dummy, so the code works. Can be removed / replaced

    u = self.uniform(size=size)



    values = np.power(u*(x_max**(-n+1)-x_min**(-n+1))+x_min**(-n+1),1/(-n+1))


    return values

def cauchy(self, size=None):
    '''
    Draw random numbers from a power law distribution
    with index n between x_min and x_max

    Blatt 3, Aufgabe 2c)
    '''
    # Fügen Sie hier den Code ein um Zufallszahlen aus der
    # angegebenen Verteilung zu erzeugen

    # dummy, so the code works. Can be removed / replaced
    u = self.uniform(size=size)

    values = np.tan(np.pi*u-np.pi/2)


    return values
```

Aufgabe 6

Die invertierten Funktionen werden dabei alle so auch im gitlab-Repo implementiert.

Exponentialverteilung

DIe Exponentilafunktion lässt sich normieren auf:

$$\int_0^\infty N e^{\frac{-x}{\tau}} \mathrm{d}x \overset{!}{=} 1$$
$$= N\left[-\tau e^{\frac{-x}{\tau}}\right]_0^\infty$$
$$= \tau$$

$$\implies N = \tau^{-1}$$

Durch die Integration und Invertierung erhalten wir:

$$\int_0^{x'} \tau^{-1} e^{\frac{-x'}{\tau}} \mathrm{d}x' = \left[-e^{\frac{-x'}{\tau}}\right]_0^{x'} = -e^{\frac{-x}{\tau}} + 1$$

$$\implies x(u) = -\tau \ln(1-u)$$

Potenzverteilung

Wie zuvor lässt sich die Fuktion normieren zu:

$$\int_{x_{min}}^{x_{max}} N x^{-n} \mathrm{d}x \overset{!}{=} 1$$
$$= N\left[\frac{1}{1-n} \cdot x^{-n+1}\right]_{x_{min}}^{x_{max}}$$
$$= N \frac{x_{max}^{1-n} - x_{min}^{1-n}}{1-n}$$

$$\implies N = \frac{1-n}{x_{max}^{1-n} - x_{min}^{1-n}}$$

und ebenfalls durch Integration und Invertierung erhalten wir:

$$\int_{x_{min}}^{x'} N x'^{-n} \mathrm{d}x' = \frac{1}{(x_{max}^{1-n} - x_{min}^{1-n})}(x^{-n+1} - x_{min}^{-n+1})$$

$$\implies x(u) = \left(u \cdot (x_{max}^{1-n} - x_{min}^{1-n}) + x_{min}^{-n+1}\right)^{\frac{1}{1-n}}$$

Cauchy-Verteilung

Die Cauchy-Verteilung muss nicht normiertwerden, daher erhalten wir:

$$\int_{-\infty}^{x} \frac{1}{\pi(1+x'^2)} \mathrm{d}x' = \frac{1}{\pi}(\arctan x + \frac{\pi}{2})$$

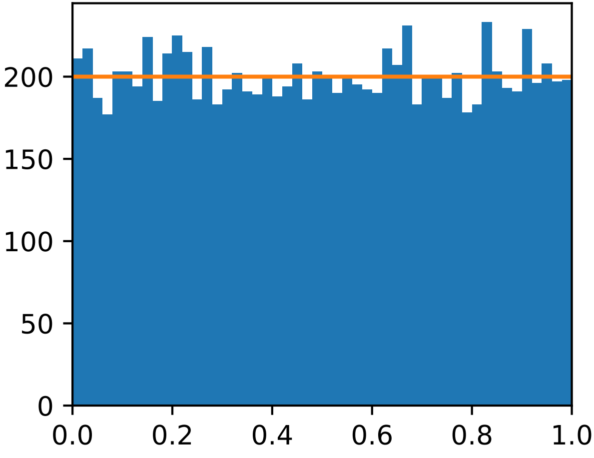
$$\implies x(u) = \tan(\pi u - \frac{\pi}{2})$$

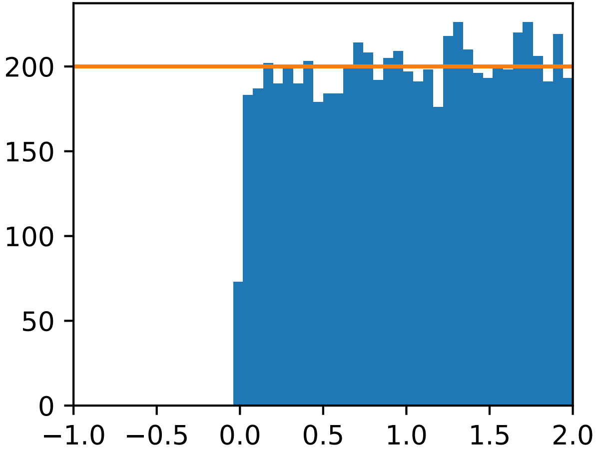
In [ ]:

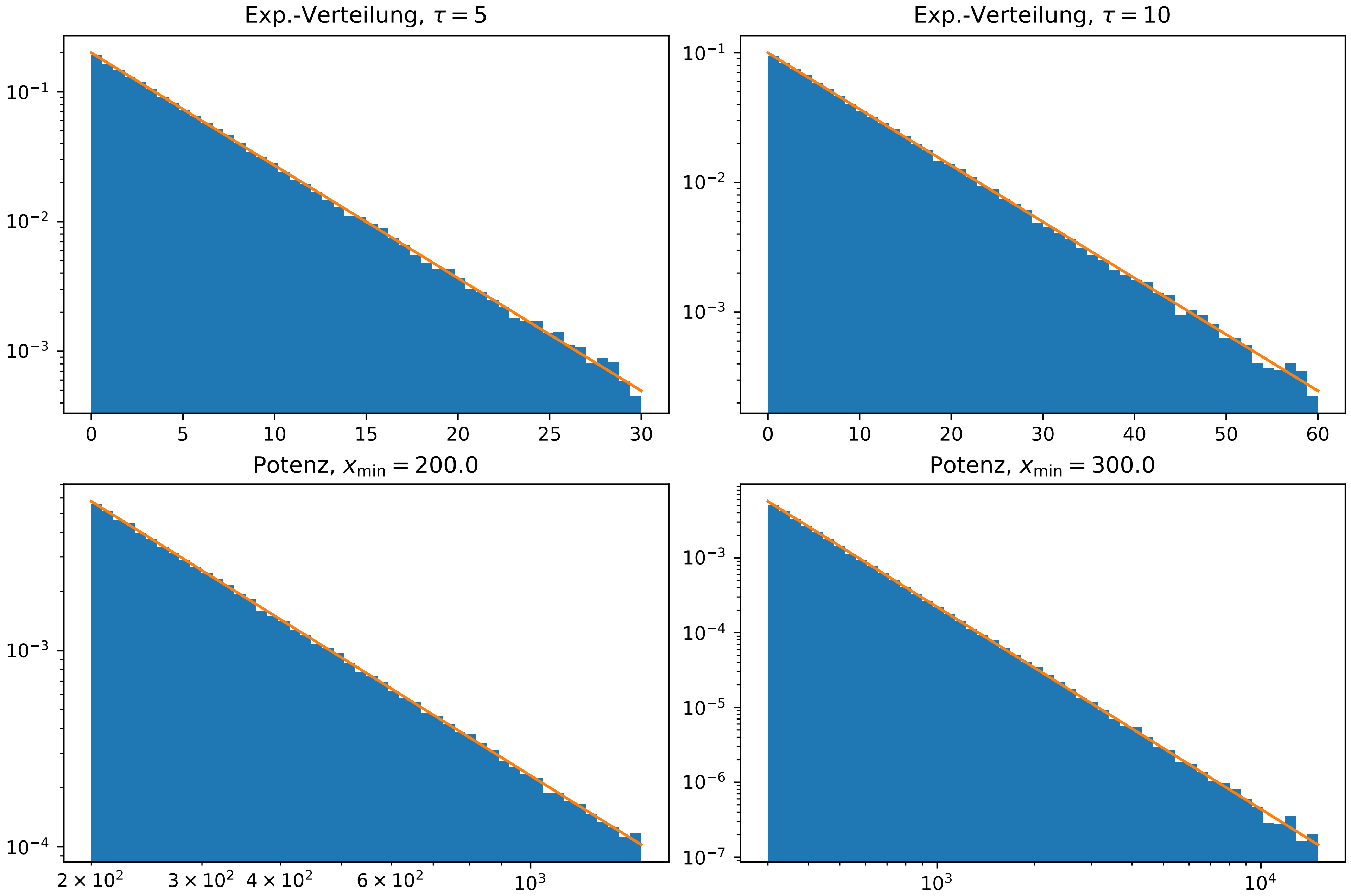
5/5, nice!

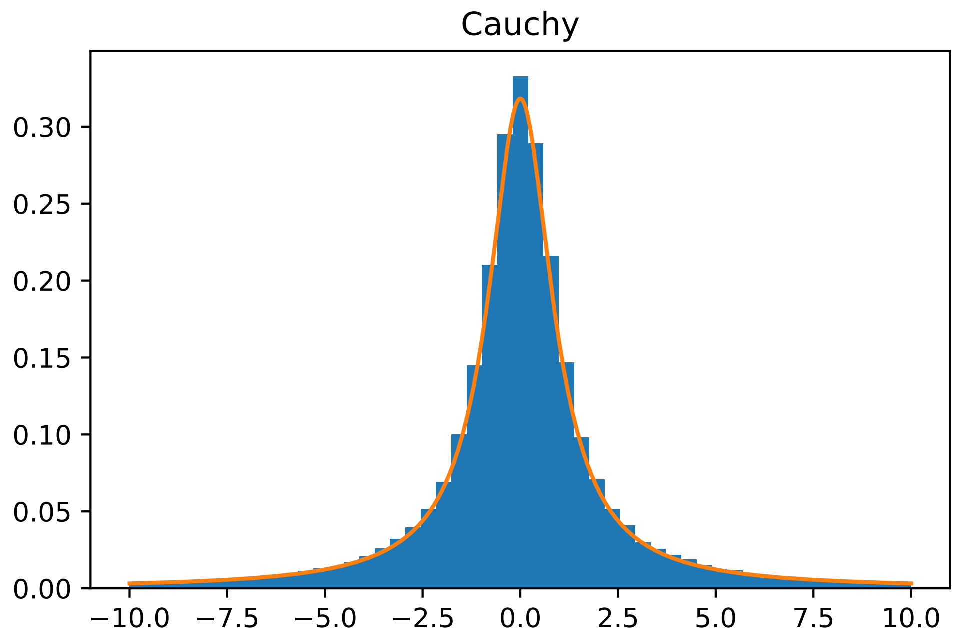
you can consider presenting this

```
Information:
  Exercise: Linear-Kongruent
  Group name: project_c3


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










```
Information:
  Exercise: Verteilungen
  Group name: project_c3
Runtime:
  exp:    2.5 ms (ref: 3.2 ms)
  power:  4.0 ms (ref: 2.6 ms)
  cauchy: 3.3 ms (ref: 3.1 ms)
Tests:
  exp mean correct: True
  power n correct:  True
```