

9.25/20P.

SMD-Abgabe

2. Übungsblatt

Lars Kolk

lars.kolk@tu-dortmund.de

Julia Sobolewski

julia.sobolewski@tu-dortmund.de

Jannine Salewski

jannine.salewski@tu-dortmund.de

Abgabe: 1.11.2018

TU Dortmund – Fakultät Physik

1 Aufgabe 5

Im folgenden wird U immer als die vorrausgesetzte Gleichverteilung angesehen. Da die Algorithmen effizient sein sollen, wird im folgenden die Methode der Transformierung der Gleichverteilung genutzt.

1.1 a)

Normierung:

$$\int_{x_{\min}}^{x_{\max}} A dx = A(x_{\max} - x_{\min}) = 1$$
$$\Rightarrow A = \frac{1}{x_{\max} - x_{\min}}$$

Transformation:

$$U = \int_{x_{\min}}^x \frac{1}{x_{\max} - x_{\min}} dx'$$
$$= \left[\frac{1}{x_{\max} - x_{\min}} \cdot x' \right]_{x_{\min}}^x$$
$$= \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Inverse bilden:

$$x(U) = U \cdot (x_{\max} - x_{\min}) + x_{\min}$$

In Abbildung 1 ist die resultierende, normierte Verteilung mit den Parametern

$$x_{\min} = -100$$
$$x_{\max} = 100$$

zu sehen. Für das Histogramm wurden 1 Millionen Zufallswerte genommen.

10.

1.2 b)

Normierung:

$$\int_0^\infty N \exp(-t/\tau) dt = \frac{-N}{\tau} [\exp(-t/\tau)]_0^\infty = \frac{N}{\tau} = 1$$
$$\Rightarrow N = \tau$$

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

Transformation:

$$U = \int_0^t \frac{1}{\tau} \cdot \exp(-t'/\tau) dt' = -\exp(-t/\tau) + 1$$

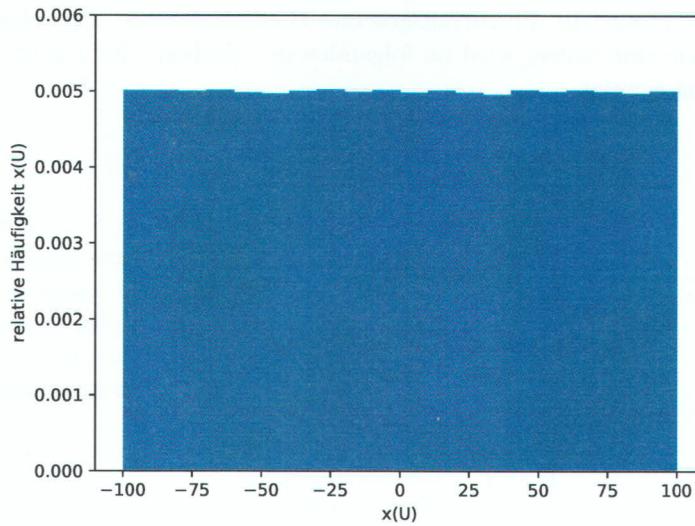


Abbildung 1: Transformierte Verteilung Aufgabenteil a).

Inverse bilden:

$$t(U) = -\tau \ln(1-U) \quad \checkmark$$

Das zugehörige Histogramm ist in Abbildung 2 zu sehen, dafür wurde für der Parameter als $\tau = 10$ angenommen und es wurden 10000 Zufallswerte verwendet.

10.

1.3 c)

Normierung:

$$\begin{aligned} \int_{x_{\min}}^{x_{\max}} N \cdot x^{-n} dx &= \frac{N}{1-n} [x^{1-n}]_{x_{\min}}^{x_{\max}} = \frac{N}{1-n} (x_{\max}^{1-n} - x_{\min}^{1-n}) = 1 \\ \Rightarrow N &= \frac{1-n}{x_{\max}^{1-n} - x_{\min}^{1-n}} \quad \checkmark \end{aligned}$$

Transformation:

$$\begin{aligned} U &= \int_{x_{\min}}^x \frac{1-n}{x_{\max}^{1-n} - x_{\min}^{1-n}} \cdot x'^{-n} dx' \\ &= \frac{1-n}{x_{\max}^{1-n} - x_{\min}^{1-n}} \left[\frac{1}{1-n} x'^{1-n} \right]_{x_{\min}}^x \\ &= \frac{1}{x_{\max}^{1-n} - x_{\min}^{1-n}} (x^{1-n} - x_{\min}^{1-n}) \quad \checkmark \end{aligned}$$

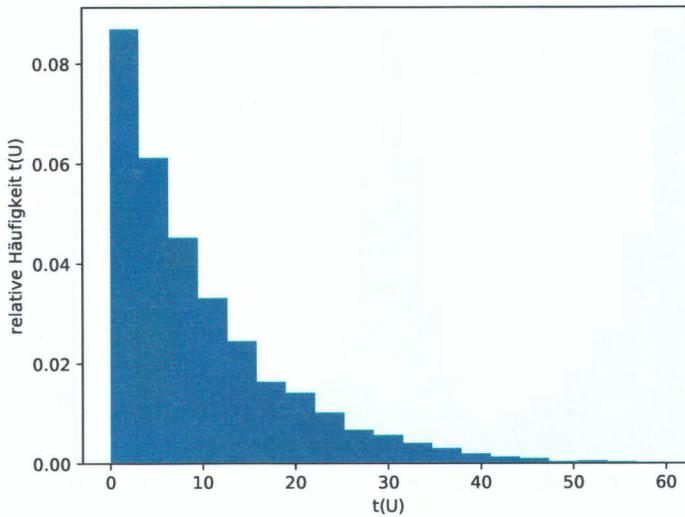


Abbildung 2: Transformierte Verteilung Aufgabenteil b).

Inverse bilden:

$$x(U) = [U \cdot (x_{\max}^{1-n} - x_{\min}^{1-n}) + x_{\min}^{1-n}]^{\frac{1}{1-n}} \quad \checkmark$$

Das zugehörige, normierte Histogramm ist in Abbildung 3 zu sehen. Hierfür wurden folgende Werte angenommen:

$$\begin{aligned} x_{\min} &= 1 \\ x_{\max} &= 10 \\ n &= 2 \end{aligned}$$

10.

1.4 d)

Normierung ist hier nicht mehr notwendig, da die Funktion schon normiert ist. Transformation:

$$\begin{aligned} U &= \int_{-\infty}^x \frac{1}{\pi} \frac{1}{1+x'^2} dx' \quad \checkmark \\ &= \frac{1}{\pi} [\arctan(x')]_{-\infty}^x \\ &= \frac{1}{\pi} \left(\arctan(x) + \frac{\pi}{2} \right) \quad \checkmark \end{aligned}$$

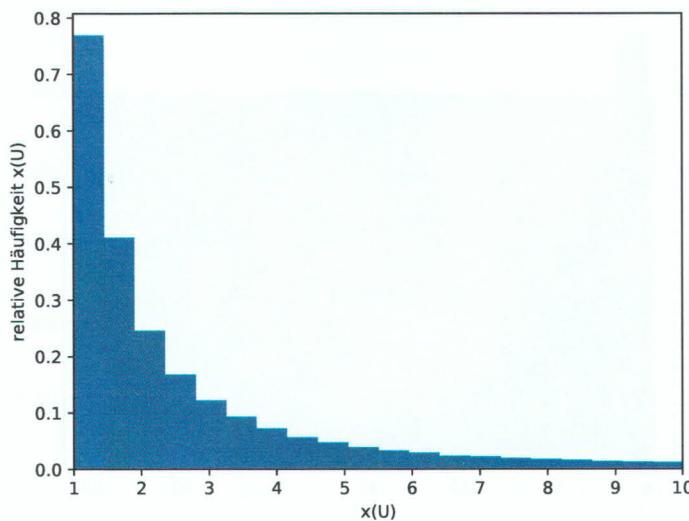


Abbildung 3: Transformierte Verteilung Aufgabenteil c).

Inverse bilden:

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

1P.

Das dazugehörige Histogramm ist in Abbildung 4 zu sehen. Hierbei wurden erneut 1 Millionen Zufallszahlen verwendet.

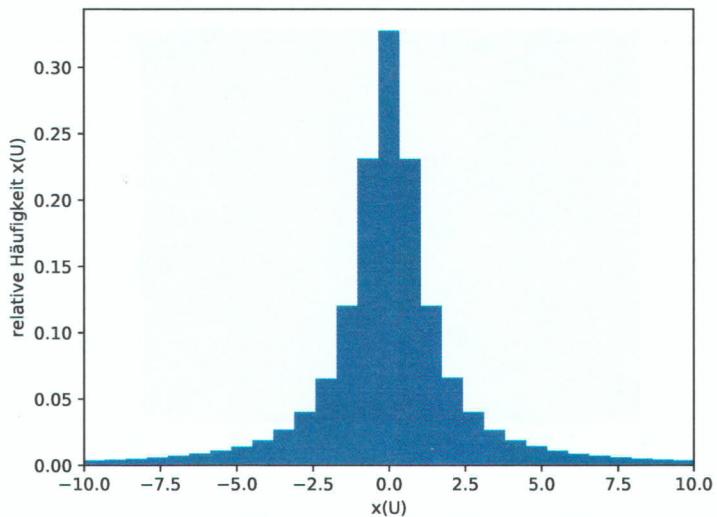


Abbildung 4: Transformierte Verteilung Aufgabenteil d).

✓ 40.

2 Aufgabe 6

In dieser Aufgabe wird der Zufallsgenerator

$$x_n = (a \cdot x_{n-1} + b) \% m \rightarrow v_n = \frac{x_n}{m} \quad (1)$$

untersucht.

Für die Teilaufgaben 2.2 und 2.3 werden die Werte $a = 1601$, $b = 3456$ und $m = 10^4$ verwendet.

2.1 Teilaufgabe a)

Für den Zufallszahlengenerator (1) ergeben sich für ein variales, ganzzahliges a , $b = 3$, $m = 1024$ und dem Startwert $x_0 = 0$ die in 5 zu sehenden Periodenlängen.

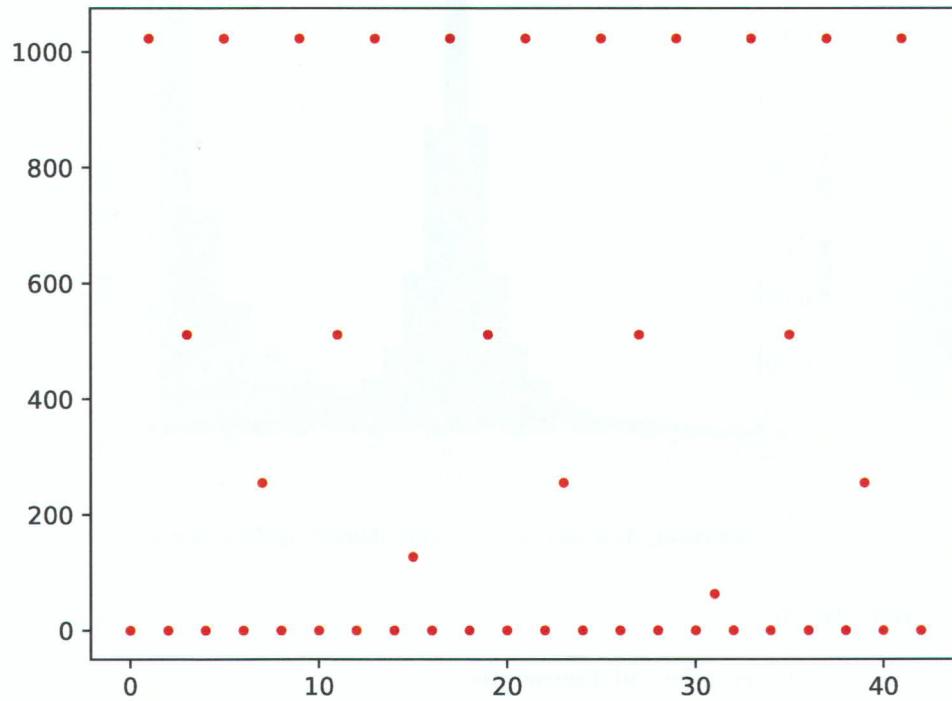


Abbildung 5: Periodenlängen in Abhängigkeit von a

wenn du die Werte
zeigst, ist es einf.
zu erkennen --

Es ist zu erkennen, dass bei $a = 4 \cdot n + 1$ die maximale Periodenlänge auftritt. Das Ergebnis lässt sich mit den Regeln für gute linear-kongruente Generatoren erklären:

- $b = 3 \neq 0$
- b und m sind Teilerfremd, da die Quersumme von 1024 gleich 7 ist. Daher ist m nicht durch b teilbar.
- Jeder Primfaktor von m teilt $a - 1$, da $1024 = 2^{10}$ und $a - 1 = 4n$ mit $n \in \mathbb{N}$. **Klausurklausur?**
- $1024 = 2^{10} = 4^5$, $a - 1 = 4n$ $n \in \mathbb{N}$ → beide sind durch 4 teilbar. ✓

2.2 Teilaufgabe b)

1P.

Für die gegebenen Startwerte ergibt sich folgendes Diagramm:

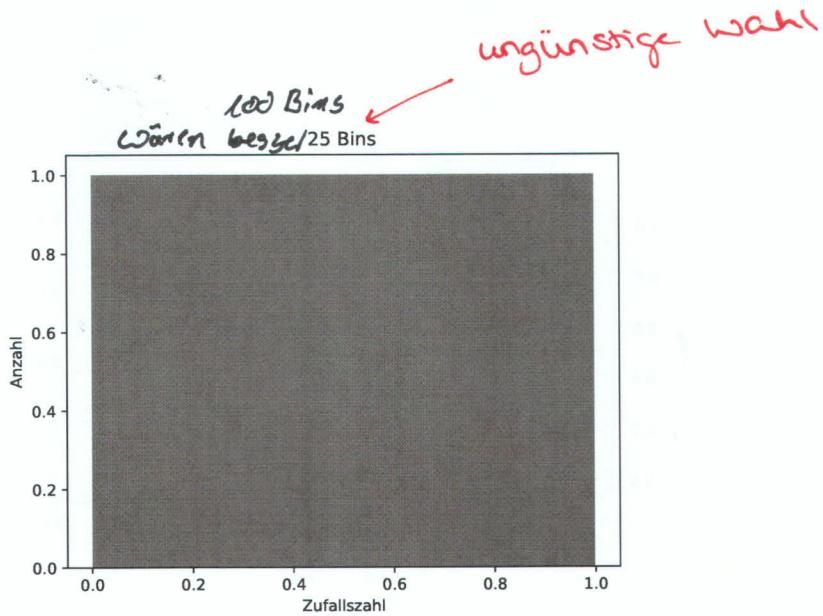


Abbildung 6: Histogramm der Zufallszahlen

innenhalb
der Bins!

Das in 6 zu sehende Ergebnis ist für Zufallszahlengeneratoren gut, da keine Zahl (viel)
oft gezogen wird, als eine andere.

(✓)
0.5P.

2.3 Teilaufgabe c)

Für die gegebenen Startwerte ergeben sich folgende Scatter-Plots:

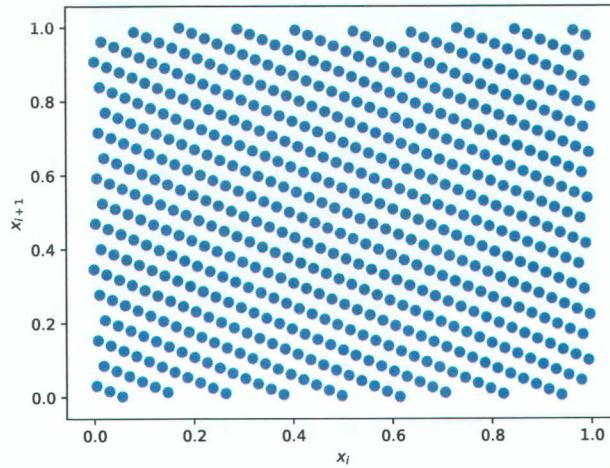


Abbildung 7: 2D-Scatter

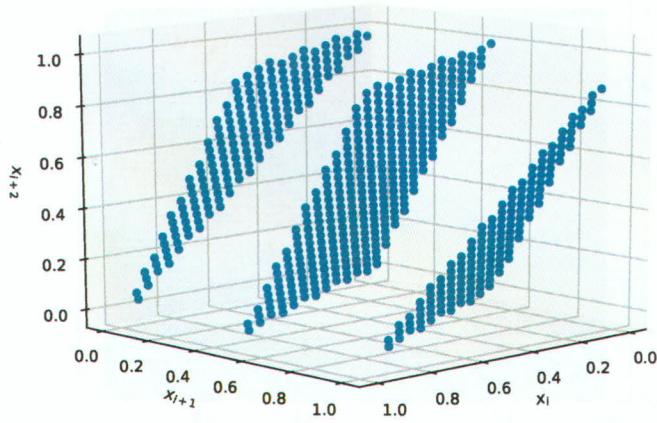


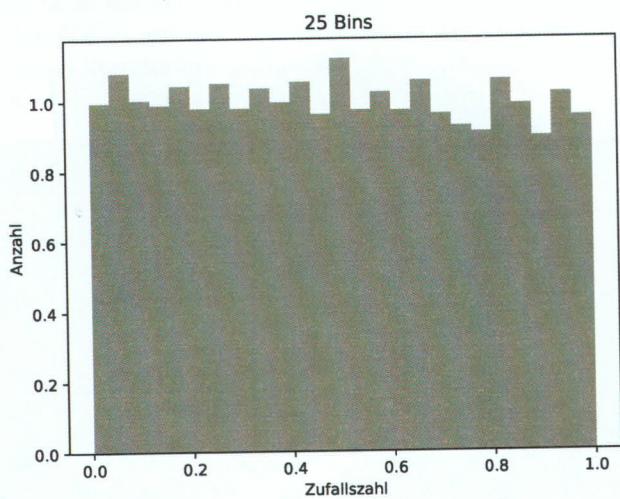
Abbildung 8: 3D-Scatter

Da sich in beiden Plots Regelmäßigkeiten erkennen lassen, handelt es sich um keinen guten Zufallszahlengenerator.

10.

2.4 Teilaufgabe d)

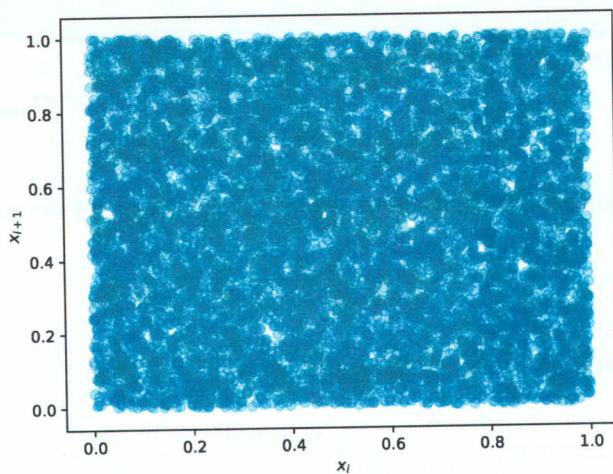
In dieser Teilaufgabe werden die Ergebnisse aus 2.2 und 2.3 mit `numpy.random.uniform(0, 1)` verglichen. Damit die Ergebnisse rekonstruierbar sind, wurde als Seed `np.random.seed(42)` gewählt. Für die Funktion `numpy.random.uniform(0, 1)` ergibt sich folgendes Histogramm:



✓

Abbildung 9: Histogramm für `numpy.random.uniform(0, 1)`

Im Vergleich mit 6 fällt auf, dass die Zufallszahlen weniger gleichverteilt sind, was den Zufallsgenerator in dieser Hinsicht etwas schlechter macht. Es ergeben sich folgende Scatter-Plots:



alpha kleiner wähler

✓

Abbildung 10: 2D-Scatter

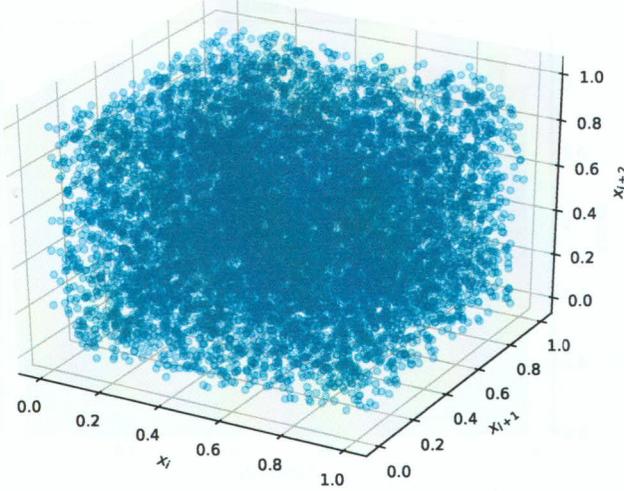


Abbildung 11: 3D-Scatter

Im Vergleich zu 10 und 11 fällt hier jedoch auch, dass die Zufallszahlen hier "zufälliger" verteilt sind und sich kein Gitter erkennen lässt, womit `numpy.random.uniform(0, 1)` in dieser Hinsicht besser ist.

1P.

jeder "

2.5 Teilaufgabe e)

Für ganzzahlige x_0 liefert der Zufallsgenerator den Wert $\frac{1}{2}$ genau 1 mal (pro Periode).
Für nicht-ganzzahlige x_0 tritt dieser Wert nicht auf.

16 oder 0 mal

OP
3.5P

3 Aufgabe 7

- a) Der Korrelationskoeffizient berechnet sich aus Formel (2) und beträgt $\rho = 0,8$. ✓

$$\rho = \frac{\text{Cov}(x, y)}{\sigma_x \cdot \sigma_y} \quad \checkmark \quad (2)$$

0.5P

- b) Die 2D-Gaußverteilung sieht wie folgt aus:

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right)\right) \stackrel{!}{=} \text{const.}$$

Durch Umformung erhält man Gleichung 3. Diese ist eine Ellipsengleichung.

$$\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} = \text{const.} \quad (3)$$

NP.

- c) In Abbildung 12 ist die 2D-Gaußverteilung als Heatmap dargestellt. Zusätzlich sind die Ellipse, bei der $f(x, y)$ auf das $1/\sqrt{e}$ -fache seines Maximums abgefallen ist, und die Mittelwerte mit ihren Standardabweichungen eingezeichnet.

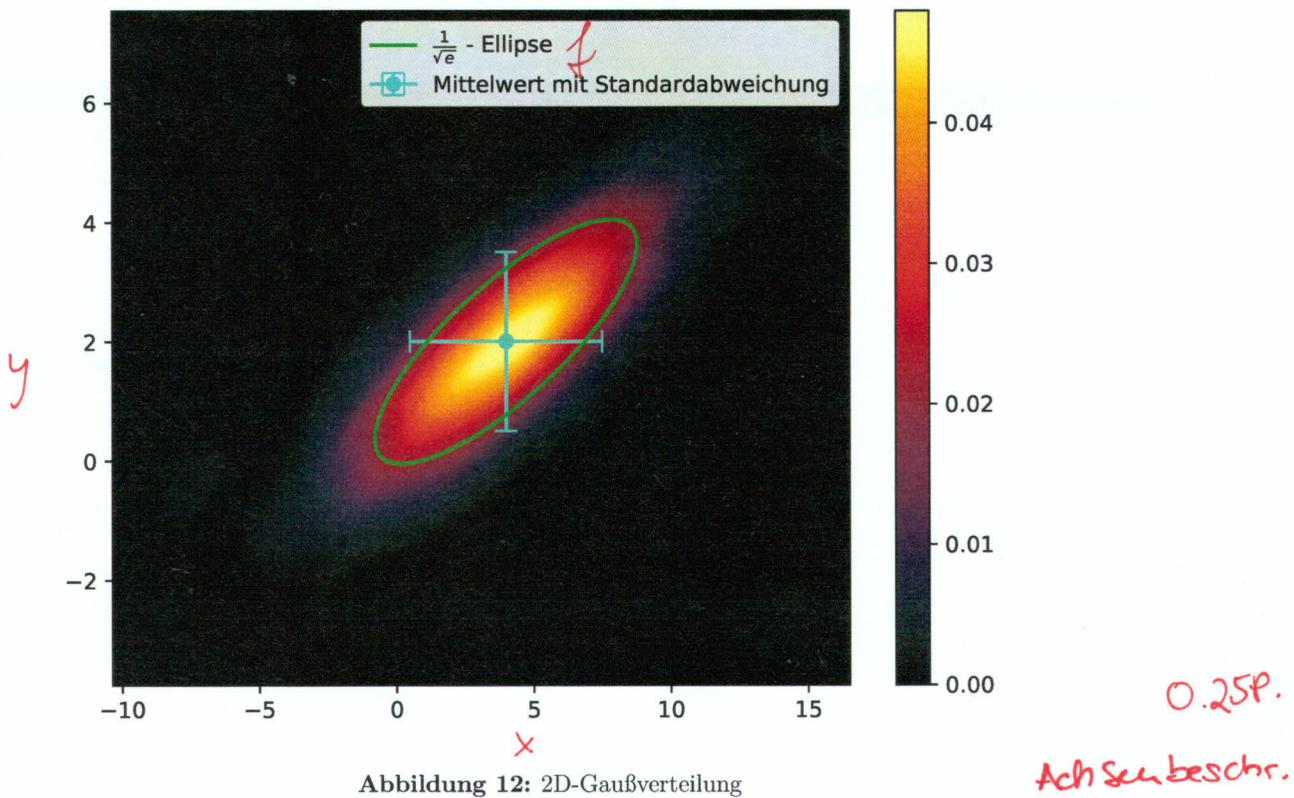


Abbildung 12: 2D-Gaußverteilung

$$n = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix}$$

$$\zeta(x'y') =$$

$$f(x|y) = \frac{f(x,y)}{f(y)}$$

$$e) f(y) = \int_{-\infty}^{\infty} f(x,y) dx$$

$$f(y|x) = \frac{f(x,y)}{f(x)}$$

$$f(x) = \int_{-\infty}^{\infty} f(x,y) dy$$

$$f) E(x|y) = \int_{-\infty}^{\infty} x f(x|y) dx$$

13

1.75P.

Code fuer Blatt02

Kolk, Sobolewski, Salewski

2. November 2018

```
.../B/7/Blatt02_Kolk_Sobolewski_Salewski/Aufgabe06.py

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 np.random.seed(42)
5
6
7 #Erzeugt Zufallszahlen nach der Definition der Aufgabe. fac gibt an, wie oft durchgegangen
    werden soll (fac*m mal)
8 def random(a, b, m, x_0, fac):
9     x_a=x_0
10    List=[None]
11    for i in range(fac*m):
12        x_n=((a*x_a+b) % m)
13        List[i]=x_n/m
14        List+=[None]
15        x_a=x_n
16    List=List[:-1]
17    return(List)
18
19 #Funktion zum finden von Periodenlängen
20 def findperiod(list):
21     #für ein x...
22     for x in range(len(list)):
23         #—suche ein xx mit List[x]==List[xx+x]
24         for xx in range(len(list)-x):
25             if list[x] == list[xx+x] and xx != 0:
26                 #wenn gefunden, dann überprüfe, ob List[x+i]==List[xx+x+i]
27                 for i in range(len(list)-x-xx):
28                     #Tritt Ungleichheit auf => brich ab
29                     if list[x+i] != list[xx+x+i]:
30                         break
31                 #keine Ungleichheiten bis zum Ende der Liste: Periode gefunden!
32                 elif i==(len(list)-xx-x-1):
33                     return xx
34
35 #Funktion zum shiften von Arrayeinträgen
36 def shift(list, n):
37     n = n % len(list)
38     return list[n:] + list[:n]
39
40 #Teilaufgabe a)
41 #stelle Periodenlängen von 0 bis 42 in einem Plot da.
42 x0=0
43 for a in range(43):
44     plt.plot(a, findperiod(random(a, 3, 1024, x0, 2)), 'r.')
45 plt.savefig("Teilaufgabe_a).pdf")
46 plt.clf()
47 plt.cla()
48 #Teilaufgabe b)
49 #Erstelle histogramm für m=10000, b=3456 und a=1601
50 Data=random(1601, 3456, 10000, 0, 1)
51 plt.hist(Data, density=True, color='grey', bins=25)
52 plt.xlabel("Zufallszahl")
53 plt.ylabel("Anzahl")
54 plt.title("25 Bins")
55 plt.savefig("Teilaufgabe_b)_Histogramm.pdf")
56 plt.clf()
```

```

57
58 #Teilaufgabe c)
59 #2D-Scatter
60 x=random(1601, 3456, 10000, 0, 1)
61 y = shift(x, 1)
62 z=shift(Data, 2)
63 plt.scatter(x, y)
64 plt.xlabel(r'$x_{(i)}$')
65 plt.ylabel(r'$x_{(i+1)}$')
66 plt.savefig("Teilaufgabe_c)_2D-Scatter.pdf")
67 plt.clf()
68
69 #3D-Scatter
70 fig = plt.figure()
71 ax=Axes3D(fig)
72
73 ax.scatter(x, y, z, alpha=0.3)
74 ax.set_xlabel(r'$x_{(i)}$')
75 ax.set_ylabel(r'$x_{(i+1)}$')
76 ax.set_zlabel(r'$x_{(i+2)}$')
77 plt.show()
78 #plt.savefig("Test.pdf")
79 plt.clf()
80
81 #Teilaufgabe d)
82 newdata=[]
83 #Erstelle 10000 Zufallszahlen
84 for x in range(10000):
85     newdata.append(np.random.uniform(0, 1))
86
87 #Erstelle Histogramm
88 plt.hist(newdata, density=True, color='grey', bins=25)
89 plt.xlabel("Zufallszahl")
90 plt.ylabel("Anzahl")
91 plt.title("25 Bins")
92 plt.savefig("Teilaufgabe_d)_Histogramm.pdf")
93 plt.clf()
94
95 #Erstelle Scatter
96 x=newdata
97 y=shift(newdata, 1)
98 z=shift(newdata, 2)
99
100 #Erstelle 2D Scatter
101 plt.scatter(x, y, alpha=0.3)
102 plt.xlabel(r'$x_{(i)}$')
103 plt.ylabel(r'$x_{(i+1)}$')
104 plt.savefig("Teilaufgabe_d)_2D-Scatter.pdf" )
105 plt.clf()
106 plt.cla()
107
108 #3D Scatter
109 #ich weiß nicht wieso, aber ich muss das anscheinend neu definieren, sonst bleiben meine Plots
110 # leer .
110 fig = plt.figure()
111 ax=Axes3D(fig)
112 #z=newdata
113 #y=shift(newdata, 1)
114 #z=shift(newdata, 2)
115 ax.scatter(x, y, z, alpha=0.3)
116 ax.set_xlabel(r'$x_{(i)}$')
117 ax.set_ylabel(r'$x_{(i+1)}$')
118 ax.set_zlabel(r'$x_{(i+2)}$')
119 plt.show()
120 plt.clf()
121 #print(z)
122
123 #Teilaufgabe e)
124 #Überprüfe für 1^4 Startwerte, wie oft 1/2 vorkommt und schreibe Ergebnisse in txt
125 datacount=open("datalog.txt", "w")
126 counter=[]

```

```

127 for x in range(10000):
128     counter.append(random(1, 3, 1024, x/1000, 1).count(0.5))
129     string="Fuer x_0 = " + str(x/1000) +" kommt der Wert 0.5 " + str(counter[x]) + " mal vor \n"
130     datacount.write(string)

    .../B/7/Blatt02_Kolk_Sobolewski_Salewski/aufgabe07.py

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from numpy import linalg as LA
4 from scipy.stats import kde
5
6 mu_x = 4
7 mu_y = 2
8 sigma_x = 3.5
9 sigma_y = 1.5
10 covariance = 4.2
11
12
13 # a) Korrelationskoeffizient
14 rho = covariance/(sigma_x*sigma_y)
15 print(rho)
16
17
18 # c) Ellipse
19 np.random.seed(100)
20 mean = [mu_x, mu_y]
21 cov = [[sigma_x**2, covariance], [covariance, sigma_y**2]]
22
23 x, y = np.random.multivariate_normal(mean, cov, 10000).T          # Ziehen der Normalverteilung
24
25 w, v = LA.eig(cov)          # Bestimmung der Eigenwerte und -vektoren der Covarianz-Matrix
26 ind_w_max = w.argmax()
27 v_max = v[:, ind_w_max]      # Eigenvektor in Richtung der großen Hauptachse
28 ind_w_min = w.argmin()
29 v_min = v[:, ind_w_min]      # Eigenvektor in Richtung der kleinen Hauptachse
30
31 theta = np.arctan(v_max[1]/v_max[0])          # Bestimmung des Winkels zwischen der großen
32                                     # Hauptachse und der x-Achse
33
34 chisquare_val = 1.87          # Die Wahrscheinlich 1/e nach der Chi^2-Verteilung
35 theta_grid = np.linspace(0, 2*np.pi)
36 phi = -theta
37 X0 = mean[0]
38 Y0 = mean[1]
39 a = np.sqrt(chisquare_val*max(w))          # Länge der großen Hauptachse
40 b = np.sqrt(chisquare_val*min(w))          # Länge der kleinen Hauptachse
41
42 ellipse_x_r = a * np.cos(theta_grid)        # Umrechnung in kartesische Koordinaten
43 ellipse_y_r = b * np.sin(theta_grid)        #
44
45
46 R = np.array([[np.cos(phi), -np.sin(phi)], [np.sin(phi), np.cos(phi)]])          # Berechnung der
47                                     # Rotationsmatrix
48
49 r_ellipse = np.dot(np.array((ellipse_x_r, ellipse_y_r)).T, R)          # Drehung der Ellipse
50                                     # mithilfe der Rotationsmatrix
51
52 plt.plot(r_ellipse[:,0] + X0, r_ellipse[:,1] + Y0, color='lime', linestyle='--',
53           label=r'$\frac{1}{\sqrt{e}}$ - Ellipse')          # Plotten der Ellipse
54 plt.errorbar(mu_x, mu_y, xerr=sigma_x, yerr=sigma_y, capsizes=5, color='cyan', marker='o',
55           label=r'Mittelwert mit Standardabweichung')          # Plotten der Abweichungen
56
57 nbins=100
58                                     # Plotten einer Heatmap für die gezogene Gaußverteilung
59 k = kde.gaussian_kde([x,y])
60 xi, yi = np.mgrid[x.min():x.max():nbins*lj, y.min():y.max():nbins*lj]

```

```

59 zi = k(np.vstack([xi.flatten(), yi.flatten()]))
60
61 plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='gouraud', cmap='inferno')
62 plt.colorbar()
63
64 plt.legend()
65 plt.tight_layout()
66 #plt.show()
67 plt.savefig('ellipse.pdf')

    .../B/7/Blatt02_Kolk_Sobolewski_Salewski/Aufgabe5.py

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from scipy.optimize import curve_fit
4 import uncertainties.unumpy as unp
5 from uncertainties import ufloat
6 from scipy.stats import stats
7
8 #Aufgabe 5 a)
9 np.random.seed(12)
10 uniform1 = np.random.uniform(0, 1, 1000000)
11 x_min1 = -100
12 x_max1 = 100
13 uniform1_transformiert = uniform1*(x_max1- x_min1) +x_min1
14 plt.ylim(0, 0.006) ← besser an y-max anpassen
15 plt.hist(uniform1_transformiert, bins=20, density=True)
16 plt.xlabel('x(U)')
17 plt.ylabel('relative Häufigkeit x(U)')
18 plt.savefig('Transformiert1.pdf')
19 plt.clf()
20
21 #b)
22 uniform2 = np.random.uniform(0, 1, 10000)
23 tau = 10
24 uniform2_transformiert = -tau * np.log(1-uniform2)
25 plt.hist(uniform2_transformiert, bins=np.linspace(0,60,20), density=True)
26 plt.xlabel('t(U)')
27 plt.ylabel('relative Häufigkeit t(U)')
28 plt.savefig('Transformierte2.pdf')
29 plt.clf()
30
31 #c)
32 uniform3 = np.random.uniform(0, 1, 100000)
33 xmin3= 1
34 xmax3= 10
35 n=2
36 uniform3_transformiert = ( uniform3*(xmax3***(1-n)-xmin3***(1-n))+xmin3***(1-n) )***(1/(1-n))
37 plt.xlim(xmin3, xmax3)
38 plt.hist(uniform3_transformiert, bins=20, density=True)
39 plt.xlabel('x(U)')
40 plt.ylabel('relative Häufigkeit x(U)')
41 plt.savefig('Transformierte3.pdf')
42 plt.clf()
43
44 #d)
45 uniform4 = np.random.uniform(0,1, 1000000)
46 uniform4_transformiert = np.tan(np.pi*(uniform4-1/2))
47 plt.xlim(-10,10)
48 plt.hist(uniform4_transformiert, bins=np.linspace (-10 , 10, 30), density=True)
49 plt.xlabel('x(U)')
50 plt.ylabel('relative Häufigkeit x(U)')
51 plt.savefig('Transformierte4.pdf')
52 plt.clf()
53
54 #e)
55 data = np.load("empirisches histogramm.npy")
56 plt.hist(data['bin_mid'], bins=np.linspace(0, 1, 51) ,weights =data['hist'])
57 plt.show ()

```