SMD-Abgabe

8. Übungsblatt

Lars Kolk
lars.kolk@tu-dortmund.de

Julia Sobolewski
julia.sobolewski@tu-dortmund.de

Jannine Salewski
jannine.salewski@tu-dortmund.de

Abgabe: 13.12.2018

TU Dortmund – Fakultät Physik

22	23	24	Σ
6/6	~~4/6~~ 5/6	5,5/8	16,5/20

Aufgabe 23:

a) Punkte der Geraden: $\begin{pmatrix} x_1 \\ z_1 \end{pmatrix}$ und $\begin{pmatrix} x_2 \\ z_2 \end{pmatrix}$

$x_1 = a z_1 + b$ $\Rightarrow a = \frac{x_2 - x_1}{z_2 - z_1}$ ✓

$x_2 = a z_2 + b$ $\Rightarrow b = x_1 - \frac{x_2 - x_1}{z_2 - z_1} \cdot z_1$ ✓ f $= x_1 \frac{-z_2}{z_1 - z_2} + x_2 \frac{z_1}{z_1 - z_2}$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} z_1 & 1 \\ z_2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$ ✓

$$\begin{pmatrix} a \\ b \end{pmatrix} = \underbrace{\frac{1}{z_2 - z_1} \begin{pmatrix} 1 & -1 \\ -z_2 & z_1 \end{pmatrix}}_{B \hat{=} \text{Jacobimatrix}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$ (✓) 0/1 da die Geradengleichung fehlt bzw. b falsch ausgerechnet

$$V\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{(z_2 - z_1)^2} \begin{pmatrix} 1 & -1 \\ -z_2 & z_1 \end{pmatrix} \begin{pmatrix} \sigma_{x_1}^2 & 0 \\ 0 & \sigma_{x_2}^2 \end{pmatrix} \begin{pmatrix} 1 & -z_2 \\ -1 & z_1 \end{pmatrix}$$ $\rightarrow$ BVB-Formel

$$= \frac{1}{(z_2 - z_1)^2} \begin{pmatrix} \sigma_{x_1}^2 & -\sigma_{x_2}^2 \\ -\sigma_{x_1}^2 z_2 & \sigma_{x_2}^2 z_1 \end{pmatrix} \begin{pmatrix} 1 & -z_2 \\ -1 & z_1 \end{pmatrix}$$

$$= \frac{1}{(z_2 - z_1)^2} \begin{pmatrix} \sigma_{x_1}^2 + \sigma_{x_2}^2 & -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) \\ -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) & \sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2 \end{pmatrix}$$ Kovarianzmatrix (✓) 1/1P

$\sigma_a = \frac{1}{(z_2 - z_1)} \cdot \sqrt{\sigma_{x_1}^2 + \sigma_{x_2}^2}$

$\sigma_b = \frac{1}{z_2 - z_1} \cdot \sqrt{\sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2}$

Fehler

$$\rho(a,b) = \frac{1}{\sigma_a \sigma_b} \operatorname{Cov}(a,b) = -\frac{(z_2 \sigma_{x_1}^2 + z_1 \sigma_{x_2}^2)}{\sqrt{(\sigma_{x_1}^2 + \sigma_{x_2}^2)(\sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2)}}$$ Korellationskoeffizient ✓ 1P./1P.

b) $x_3 = a \cdot z_3 + b = \frac{x_2 - x_1}{z_2 - z_1} \cdot z_3 + x_1 - \frac{x_2 - x_1}{z_2 - z_1} \cdot z_1$ ✓ f s.o.

$$V[x_3] = \underbrace{\begin{pmatrix} \frac{\partial x_3}{\partial a} & \frac{\partial x_3}{\partial b} \end{pmatrix}}_{\text{Jacobi}} \cdot \frac{1}{(z_2 - z_1)^2} \begin{pmatrix} \sigma_{x_1}^2 + \sigma_{x_2}^2 & -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) \\ -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) & \sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2 \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial x_3}{\partial a} \\ \frac{\partial x_3}{\partial b} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{z_2 - z_3}{z_2 - z_1} & \frac{z_3 - z_1}{z_2 - z_1} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{x_1}^2 + \sigma_{x_2}^2 & -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) \\ -(\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1) & \sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2 \end{pmatrix} \cdot \frac{1}{(z_2 - z_1)^2} \cdot \begin{pmatrix} \frac{z_2 - z_3}{z_2 - z_1} \\ \frac{z_3 - z_1}{z_2 - z_1} \end{pmatrix}$$ ff ff ff

$$= \frac{1}{(z_2 - z_1)^4} \left[(\sigma_{x_1}^2 + \sigma_{x_2}^2)(z_2 - z_3)^2 - (\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1)(z_3 - z_1)^2 \right]$$ 1,5/2P.

$$\sigma_{x_3} = \sqrt{V[x_3]} = \frac{1}{(z_2 - z_1)^2} \cdot \sqrt{(\sigma_{x_1}^2 + \sigma_{x_2}^2)(z_2 - z_3)^2 - (\sigma_{x_1}^2 z_2 + \sigma_{x_2}^2 z_1)(z_3 - z_1)^2}$$ ✓

c) $$V[x_3] = \frac{1}{(z_2 - z_1)^4} \begin{pmatrix} z_2 - z_3 & z_3 - z_1 \end{pmatrix} \begin{pmatrix} \sigma_{x_1}^2 + \sigma_{x_2}^2 & 0 \\ 0 & \sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2 \end{pmatrix} \begin{pmatrix} z_2 - z_3 \\ z_3 - z_1 \end{pmatrix}$$

ohne Korellation $= \frac{1}{(z_2 - z_1)^4} \cdot \left[(\sigma_{x_1}^2 + \sigma_{x_2}^2)(z_2 - z_3)^2 + (\sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2)(z_3 - z_1)^2 \right]$

$$\sigma_{x_3 \text{ohne}} = \frac{1}{(z_2 - z_1)^2} \sqrt{(\sigma_{x_1}^2 + \sigma_{x_2}^2)(z_2 - z_3)^2 + (\sigma_{x_1}^2 z_2^2 + \sigma_{x_2}^2 z_1^2)(z_3 - z_1)^2}$$ (✓) FF

mit Korellation fehlt $\Rightarrow$ 0,5/1P.

$\sigma_{x_3}^2 = \sqrt{z_3^2 \sigma_a^2 + \sigma_b^2 + 2(z_3 \cdot \operatorname{cov}(a,b))}$

5/6P.

Aufgabe 22

a)

Gegeben:

$$y = a_o + a_1 \cdot x$$
$$a_0 = 1,0 \pm 0,2$$
$$a_1 = 1,0 \pm 0,2$$
$$\rho = -0,8$$

Ohne Korrelation :

$$\sigma_y^2 = \left(\frac{\partial y}{\partial a_0}\sigma_{a_0}\right)^2 + \left(\frac{\partial y}{\partial a_1}\sigma_{a_1}\right)^2$$
$$= \sigma_{a_0}^2 + \sigma_{a_1}^2 x^2 = 0,04 + 0,04 \cdot x^2$$

✓

Mit Korrelation:

$$\sigma_y^2 = \left(\frac{\partial y}{\partial a_0}\sigma_{a_0}\right)^2 + \left(\frac{\partial y}{\partial a_1}\sigma_{a_1}\right)^2 + 2\frac{\partial y}{\partial a_0}\frac{\partial y}{\partial a_1} \cdot \sigma_{a_0}\sigma_{a_1} \cdot \rho$$
$$= \sigma_{a_0}^2 + \sigma_{a_1}^2 x^2 + 2 \cdot \sigma_{a_0}\sigma_{a_1} \cdot \rho x$$
$$= 0,04 \cdot x^2 - 0,064 \cdot x + 0,04$$

✓ 2/2

b)

In [33]:
```
import numpy as np
import uncertainties as unc
import uncertainties.unumpy as unp
from uncertainties import correlated_values
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
np.random.seed(42)
```

In [34]:
```
def parabel(x, a, b, c):
    return a*x**2+b*x+c
```

In [35]:
```
a_0 = unc.ufloat(1.0,0.2) # = 1,0 +- 0,2
a_1 = unc.ufloat(1.0,0.2) # = 1,0 +- 0,2
rho = -0.8
cov = rho*a_0.s*a_1.s #.s gibt die Varianz
covmatrix = np.array([[a_0.s**2, 0], [0, a_1.s**2]])
covmatrix
```

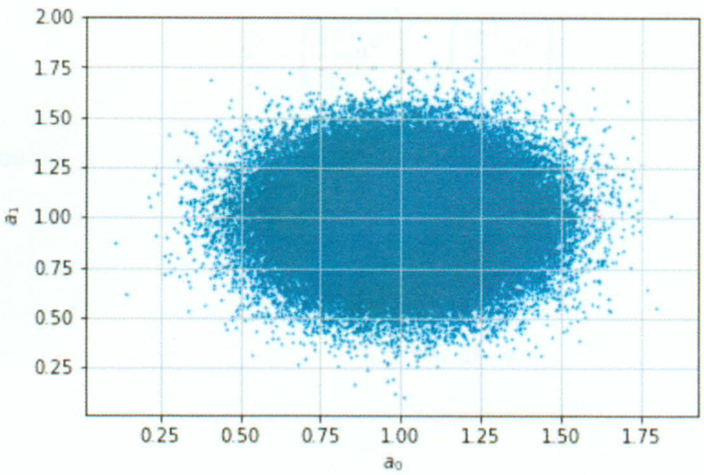
Out[35]:
```
array([[0.04, 0.  ],
       [0.  , 0.04]])
```

Ohne Korrelation

Scatterplot

In [36]:
```
Zufall=np.random.multivariate_normal(mean = [a_0.n, a_1.n], cov = co
vmatrix, size=10**5) # Einzig hübsche Funktion die ich gefunden habe
, die Zufallszahlen mit Mittelwert und Standard-Abweichungen generie
rt
a0_ok=Zufall[:,0]
a1_ok=Zufall[:,1]
plt.scatter(a0_ok, a1_ok, s=0.5)
plt.xlabel(r'$a_0$')
plt.ylabel(r'$a_1$')
plt.grid()
print(a_0.n, np.mean(a_0))
print(a_1.n, np.mean(a_1))
```

```
1.0 1.00+/-0.20
1.0 1.00+/-0.20
```



✓

Numerische Berechnung

In [37]:
```
xx = np.linspace(-10, 10, 100)
array=[]
for x in xx:
    y=a0_ok+a1_ok*x
    yy = np.var(y)
    array.append(yy)

params, cov0 = curve_fit(parabel, xx, array)
a, b, c = params
print(params)
print(a, b, c)
```

```
[0.03984436 0.00016159 0.04014075]
0.03984436065601598 0.00016159282630883538 0.04014074532424168
```

$f(x) = ax^2 + bx + c = 0.0398x^2 + 0.0002x + 0,0401$

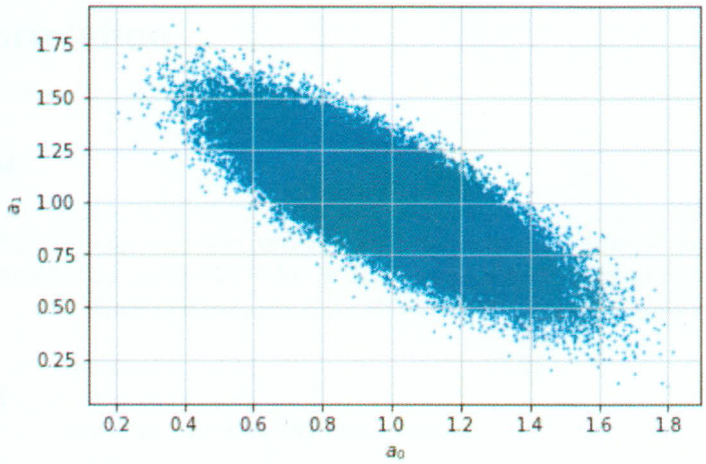
Mit Korrelation

In [38]:
```
a_0 = unc.ufloat(1.0,0.2) # = 1,0 +- 0,2
a_1 = unc.ufloat(1.0,0.2) # = 1,0 +- 0,2
rho = -0.8
cov = rho*a_0.s*a_1.s #.s gibt die Varianz
covmatrix = np.array([[a_0.s**2, cov], [cov, a_1.s**2]])
covmatrix
```

Out[38]:
```
array([[ 0.04 , -0.032],
       [-0.032,  0.04 ]])
```

Scatterplot

In [39]:
```
Zufall=np.random.multivariate_normal(mean = [a_0.n, a_1.n], cov = co
vmatrix, size=10**5) # Einzig hübsche Funktion die ich gefunden habe
, die Zufallszahlen mit Mittelwert und Standard-Abweichungen generie
rt
a0_mk=Zufall[:,0]
a1_mk=Zufall[:,1]
plt.scatter(a0_mk, a1_mk, s=0.5)
plt.xlabel(r'$a_0$')
plt.ylabel(r'$a_1$')
plt.grid()
```



✓

Numerische Berechnung

In [40]:
```
xxx = np.linspace(0, 5, 100)
array2=[]
for x in xxx:
    y=a0_mk+a1_mk*x
    yy = np.var(y, ddof = 1)
    array2.append(yy)

params, cov0 = curve_fit(parabel, xxx, array2)
a, b, c = params
print(params)
```

```
[ 0.03996244 -0.06405527  0.04012347]
```

$f(x) = 0,0400x^2 - 0,064x + 0,0401$ ✓ 2/2

c)

In [41]:
```
def varianz_ok(x, sigma_a0, sigma_a1):
    return sigma_a0**2 + sigma_a1**2*x**2

def varianz_mk(x, sigma_a0, sigma_a1, cov):
    return sigma_a0**2+sigma_a1**2*x**2 + 2*cov*x
```

Unkorreliert

In [42]:
```
xx={-3, 0, 3}
for x in xx:
    print('x =',x,':')
    meanAn = 1+x
    stdAn = np.sqrt(varianz_ok(x, 0.2, 0.2))
    y = a0_ok+a1_ok*x
    meanMC = np.mean(y)
    stdMC = np.std(y)
    print('Analytischer Mittelwert =', meanAn, ', Standardabweichung
=', stdAn)
    print('Monte-Carlo Mittelwert =', meanMC,', Standardabweichung =
', stdMC)
    print('Abweichung des Mittelwerts =', (meanMC-meanAn)/meanAn*100
, '%')
    print('Abweichung =', (stdMC-stdAn)/stdAn*100, '% \n')
```

```
x = 0 :
Analytischer Mittelwert = 1 , Standardabweichung = 0.2
Monte-Carlo Mittelwert = 1.0011420114028204 , Standardabweichung =
0.200351554334479
Abweichung des Mittelwerts = 0.1142011402820442 %
Abweichung = 0.1757771672394881 %

x = 3 :
Analytischer Mittelwert = 4 , Standardabweichung = 0.6324555320336
76
Monte-Carlo Mittelwert = 3.9988847185005585 , Standardabweichung =
0.6318423614378131
Abweichung des Mittelwerts = -0.027882037486037792 %
Abweichung = -0.09695078385846202 %

x = -3 :
Analytischer Mittelwert = -2 , Standardabweichung = 0.632455532033
676
Monte-Carlo Mittelwert = -1.9966006956949185 , Standardabweichung
= 0.6310746491101183
Abweichung des Mittelwerts = -0.16996521525407537 %
Abweichung = -0.21833676102371732 %
```

✓

Schön, aber nur korreliert war nötig

Korreliert

In [13]:
```python
xx={-3, 0, 3}
for x in xx:
    print('x =',x,':')
    meanAn = 1+x
    stdAn = np.sqrt(varianz_mk(x, 0.2, 0.2,  -0.032))
    y = a0_mk+a1_mk*x
    meanMC = np.mean(y)
    stdMC = np.std(y)
    print('Analytischer Mittelwert =', meanAn, ', Standardabweichung =', stdAn)
    print('Monte-Carlo Mittelwert =', meanMC,', Standardabweichung =', stdMC)
    print('Abweichung des Mittelwerts =', (meanMC-meanAn)/meanAn*100, '%')
    print('Abweichung=', (stdMC-stdAn)/stdAn*100, '% \n')
```

```
x = 0 :
Analytischer Mittelwert = 1 , Standardabweichung = 0.2
Monte-Carlo Mittelwert = 1.0002855463340872 , Standardabweichung = 0.2003074389650816
Abweichung des Mittelwerts = 0.028554633408717223 %
Abweichung= 0.1537194825407956 %

x = 3 :
Analytischer Mittelwert = 4 , Standardabweichung = 0.45607017003965533
Monte-Carlo Mittelwert = 3.999391221542233 , Standardabweichung = 0.4556507003969493
Abweichung des Mittelwerts = -0.015219461444171412 %
Abweichung= -0.091974803497789 %

x = -3 :
Analytischer Mittelwert = -2 , Standardabweichung = 0.7694153624668538
Monte-Carlo Mittelwert = -1.9988201288740588 , Standardabweichung = 0.7693798578621581
Abweichung des Mittelwerts = -0.058993556297060046 %
Abweichung= -0.004614491265403017 %
```

2/2

Aufgabe 24

a)

Designmatrix aufstellen:

$$A = \begin{pmatrix} \cos\left(\frac{0}{6}\pi\right) & \sin\left(\frac{0}{6}\pi\right) \\ \cos\left(\frac{1}{6}\pi\right) & \sin\left(\frac{1}{6}\pi\right) \\ \cos\left(\frac{2}{6}\pi\right) & \sin\left(\frac{2}{6}\pi\right) \\ \cos\left(\frac{3}{6}\pi\right) & \sin\left(\frac{3}{6}\pi\right) \\ \cos\left(\frac{4}{6}\pi\right) & \sin\left(\frac{4}{6}\pi\right) \\ \cos\left(\frac{5}{6}\pi\right) & \sin\left(\frac{5}{6}\pi\right) \\ \cos\left(\frac{6}{6}\pi\right) & \sin\left(\frac{6}{6}\pi\right) \\ \cos\left(\frac{7}{6}\pi\right) & \sin\left(\frac{7}{6}\pi\right) \\ \cos\left(\frac{8}{6}\pi\right) & \sin\left(\frac{8}{6}\pi\right) \\ \cos\left(\frac{9}{6}\pi\right) & \sin\left(\frac{9}{6}\pi\right) \\ \cos\left(\frac{10}{6}\pi\right) & \sin\left(\frac{10}{6}\pi\right) \\ \cos\left(\frac{11}{6}\pi\right) & \sin\left(\frac{11}{6}\pi\right) \end{pmatrix}$$ ✓

1,5/1,5

b)

Bei der Aufgabe ist kein python nötig, auch per Hand rechenbar. Übung für Klausur...

In [2]:
```
import numpy as np
```

$$\hat{a} = \left(A^T A\right)^{-1} A^T y$$

In [3]:
```
A=np.matrix([[np.cos(0), np.sin(0)],
             [np.cos(1/6*np.pi), np.sin(1/6*np.pi)],
             [np.cos(2/6*np.pi), np.sin(2/6*np.pi)],
             [np.cos(3/6*np.pi), np.sin(3/6*np.pi)],
             [np.cos(4/6*np.pi), np.sin(5/6*np.pi)],
             [np.cos(5/6*np.pi), np.sin(5/6*np.pi)],
             [np.cos(6/6*np.pi), np.sin(6/6*np.pi)],
             [np.cos(7/6*np.pi), np.sin(7/6*np.pi)],
             [np.cos(8/6*np.pi), np.sin(8/6*np.pi)],
             [np.cos(9/6*np.pi), np.sin(9/6*np.pi)],
             [np.cos(10/6*np.pi), np.sin(10/6*np.pi)],
             [np.cos(11/6*np.pi), np.sin(11/6*np.pi)]])

y=np.matrix([-0.032, 0.010, 0.057, 0.068, 0.076, 0.080, 0.031, 0.005, -0.041, -0
.041 -0.090, -0.088, -0.074])
a=(A.T*A)**(-1)*A.T*y.T
a
```

← hier kein Komma deswegen keine Dimensionsprobleme

Wert doppelt

Out[3]:
```
matrix([[-0.04019568],
        [ 0.08817037]])
```

(✓)

1/1,5

$$a_0 = -0.0401956 \quad (f)$$
$$a_1 = 0.08817037 \quad (f)$$

c)

$$V[\hat{a}] = \sigma^2 \left(A^T A\right)^{-1}$$

In [4]:
```
sigma = 0.011
V = sigma**2 * (A.T * A).I
V
```

Out[4]:
```
matrix([[ 2.01871558e-05, -6.71728350e-07],
        [-6.71728350e-07,  2.20223518e-05]])
```

=0

=0

In [5]:
```
np.sqrt(V[0,0]), np.sqrt(V[1,1])
```

Out[5]:
```
(0.004493011885516234, 0.004692797863270538)
```

Also ergibt sich:

$$\sigma_{\alpha_1} = 4,49 \cdot 10^{-3}$$
$$\sigma_{\alpha_2} = 4,69 \cdot 10^{-3}$$

In [6]:
```
rho=V[0, 1]/(np.sqrt(V[0,0]*V[1, 1]))
rho
```

Out[6]:
```
-0.0318584221555165
```

womit

$$\rho = \frac{\overset{Cov(a_1,a_2)}{\cancel{\sigma_{12}}}}{\sigma_1 \sigma_2} = \frac{-6,717 \cdot 10^{-7}}{\sqrt{2,019 \cdot 10^{-5} \cdot 2,202 \cdot 10^{-5}}} = -0,032$$

folgt.

Sollte mit richtigen Werten noch kleiner sein $\rho \approx 0$

1,5/1,5

d)

Es gilt:

$$\cos(\Psi + \delta) = \cos(\Psi)\cos(\delta) - \sin(\Psi)\sin(\delta)$$

Daraus folgt:

$$a_1 = A_0 \cdot \cos\delta$$
$$a_2 = -A_0 \cdot \sin\delta$$

und daraus wiederum

$$a_1{}^2 + a_2{}^2 = A_0{}^2 \rightarrow A_0 = \sqrt{a_1{}^2 + a_2{}^2}$$
$$\delta = -\arctan\frac{a_2}{a_1} \Rightarrow \delta = \arctan\left(-\frac{a_2}{a_1}\right)$$

Für die Fehler folgt dann:

$$\sigma_{A_0} = \sqrt{\frac{{a_0}^2}{{a_1}^2 + {a_0}^2}\sigma_{a_0}^2 + \frac{{a_1}^2}{{a_1}^2 + {a_0}^2}\sigma_{a_1}^2 + 2\frac{a_0 a_1}{({a_1}^2 + {a_0}^2)} \cdot \mathrm{Cov}(a_0, a_1)} \approx 0,0045$$

$$\sigma_\delta = \sqrt{\frac{{a_0}^2}{({a_1}^2 + {a_0}^2)^2}\sigma_{a_0}^2 + \frac{{a_1}^2}{({a_1}^2 + {a_0}^2)^2}\sigma_{a_1}^2 - 2\frac{a_0 a_1}{({a_1}^2 + {a_0}^2)^2} \cdot \mathrm{Cov}(a_0, a_1)} \approx 0,05$$

$$A_0 = 0,0958 \pm 0,0045 \quad (\checkmark)$$
$$\delta = 1.1450 \pm 0,0467 \quad f$$

Kovarianzmatrix mit BVB^T Formel ausrechnen, dann kann auch die Korrelation berechnet werden

$\frac{1,5}{3,5}$