Methoden zur Bestimmung der Messunsicherheit nach GUM – Teil 2

Methods for the Determination of the Measurement Uncertainty Using GUM – Part 2

Willfried Schwarz

Dieser zweite Teil stellt eine Fortsetzung des Beitrages aus dem Heft 2/2020 dar. Nachdem im ersten Teil die klassische GUM-Analyse vorgestellt worden ist, wird in diesem Heft gezeigt, wie Messunsicherheitsbestimmungen auch nach der Monte-Carlo-Methode vorgenommen und wie Messunsicherheiten bei Ausgleichungsrechnungen bestimmt werden können. Es werden weiterhin die Vorteile des GUM dargelegt und die Kritikpunkte am GUM angesprochen.

Schlüsselwörter: Präzision, Genauigkeit, GUM, Messunsicherheit, erweiterte Messunsicherheit, Wahrscheinlichkeitsdichtefunktionen, systematische Messabweichungen, Messunsicherheitsanalyse, Monte-Carlo-Methode

This second part is a continuation of the article in issue 2/2020. After the classic GUM analysis was presented in the first part, this issue shows how measurement uncertainty determinations can also be made using the Monte Carlo Method and how measurement uncertainties can be determined in adjustment calculations. The advantages of the GUM as well as the criticisms of this GUM are presented.

Keywords: Precision, accuracy, GUM, measurement uncertainty, advanced uncertainty, probability density functions, systematic errors, uncertainty analysis, Monte Carlo method

3.4 Messunsicherheitsbestimmung nach der Monte-Carlo-Methode

Beim klassischen GUM-Verfahren treten Probleme auf, wenn z. B. die Eingangsgrößen nicht normalverteilt, sondern beliebig verteilt oder ihr Einfluss auf die Ausgangsgröße durch eine hochgradig nichtlineare Funktion beschrieben wird. In diesen Fällen kann der Einsatz der Monte-Carlo-Methode (MC) sinnvoll sein. Bei der MC-Methode werden für die Eingangsgrößen Verteilungen (z. B. Normal- oder Rechteckverteilung) angenommen bzw. es werden weitere vorhandene Informationen über deren Verteilung verwendet, um unter Beachtung dieser Informationen durch simuliertes, zufälliges Ziehen eine große Anzahl von konkreten Werten für die Eingangsgrößen zu erzeugen. Mit der MC-Methode wird also auf Grundlage einer großen Anzahl von durchgeführten Zufallsexperimenten, die am Rechner simuliert werden, die empirische Verteilungsfunktion der Zielgröße ermittelt, um dann daraus die Kenngrößen, wie z. B. den Wert der Zielgröße und deren Varianz bzw. Standardunsicherheit zu berechnen. Moderne Programmiersprachen, wie z. B. Python, bieten dafür entsprechende Funktionen an. Damit die nach der MC-Methode berechneten Ergebnisse zuverlässig sind, benötigt man eine sehr große Anzahl von zufälligen Ziehungen. In der Praxis liegt die Anzahl der (zufälligen) Realisierungen oftmals zwischen 1 000 und 1 000 000. Von Vorteil bei der MC-Methode ist, dass mit den simulierten Eingangsgrößen nur unter Verwendung der Berechnungsformeln die Zielgrößen einschließlich ihrer Standardunsicherheiten ermittelt werden, ohne erst Sensitivitätskoeffizienten durch mitunter aufwendige Ableitungen der Berechnungsformel nach den Eingangsgrößen bilden zu müssen. Die MC-Methode ist ein rechenintensives Verfahren. Die damit verbundenen langen Rechenzeiten, die als Nachteil der MC-Methode angesehen wurden, gehören der Vergangenheit an, wenigstens für die in diesem Artikel zu betrachtenden Zahlenbeispiele. Unter Einsatz moderner Rechnerarchitekturen und effizienter Programmiersprachen sind heutzutage die Rechenzeiten akzeptabel.

3.4.1 Ablauf einer Messunsicherheitsbestimmung

Eine Messunsicherheitsbestimmung nach der MC-Methode durchläuft folgende Schritte:

1. Wählen eines geeigneten Werts für die Anzahl m ($m > 1\,000 \ldots 1\,000\,000$) der zufälligen Realisierungen der Eingangsgrößen.
2. Entsprechend der für die Eingangsgrößen gewählten Verteilungen sind die m Realisierungen auf einem Rechner vorzunehmen.
3. Mit den Realisierungen sind die m Werte der Zielgröße $y_r = f(x_{1,r}, \ldots, x_{n,r})$, mit $r = 1, \ldots, m$ zu berechnen.
4. Werden die Werte der Zielgröße y_r nach aufsteigenden Werten sortiert, erhält man ihre empirische Verteilungsfunktion.
5. Der Schätzwert der Zielgröße $\overline{y}$ ergibt sich als Mittelwert aus den m Werten y_r auf Grundlage der Gl. (*2*) zu

$$\overline{y} = \frac{1}{m}\sum_{r=1}^{m} y_r \tag{29}$$

6. und die Standardunsicherheit $u(\overline{y})$ von $\overline{y}$, entwickelt aus Gl. (*4*), zu

$$u(\overline{y}) = \sqrt{\frac{1}{m-1}\sum_{r=1}^{m}(y_r - \overline{y})^2}. \tag{30}$$

Die einzelnen Schritte werden auf die folgenden Zahlenbeispiele angewendet.

3.4.2 Zahlenbeispiel 1

Als ein (einfaches) Zahlenbeispiel wird die Berechnung der Fläche F eines Rechtecks mit den Seitenlängen $a = 80$ m und $b = 20$ m gewählt. Die Standardunsicherheiten, die aus Mehrfachmessungen ($n_w > 20$) abgeleitet worden sind, betragen $u(a) = \pm 0{,}04$ m und $u(b) = \pm 0{,}06$ m. Die Messungen sind nicht korreliert. Gesucht ist die Fläche F des Rechtecks einschließlich ihrer Standardunsicherheit $u(F)$.

Bevor jedoch die Berechnung mit der MC-Methode demonstriert wird, wird die Standardunsicherheit zu Vergleichszwecken nach der klassischen GUM-Methode (vgl. in Teil 1 Abschnitt 3.3) bzw. nach der Fortpflanzung der Unsicherheiten nach GUM ermittelt.

Es ist

$$F = a \cdot b = 80\ \text{m} \cdot 20\ \text{m},$$

$$\boxed{F = 1600\ \text{m}^2}.$$

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import time
5
6 start_time = time.time()
7 m = 1000000 # Anzahl = Werte in der Stichprobe
8 a = 80 + 0.04*np.random.randn(m,1)# Normal-Verteilung, StdAbw = 0,04 m
9 b = 20 + 0.06*np.random.randn(m,1)# Normal-Verteilung, StdAbw = 0,06 m
10 yr = a * b  # Berechnung des Vektors mit den Flächen
11 F = yr.sum()/m # Mittelwert der Flächen
12 sF = np.sqrt(yr.var()) # StdAbw der Fläche
13 print F, sF # Ausgabe: Fläche und StdAbw
14 end_time = time.time()
15 print ("Berechnungszeit:  %.2f Sekunden" % (end_time - start_time))
```

Abb. 15 | Python-Programm für die Monte-Carlo-Methode zum Zahlenbeispiel 1

Die Sensitivitätskoeffizienten betragen

$$c_a = \frac{\partial F}{\partial a} = b = 20\ \text{m} \quad \text{und} \quad c_b = \frac{\partial F}{\partial b} = a = 80\ \text{m},$$

mit den Gewichtsfaktoren $G_a = G_b = 1$ (Normal-Verteilung).

Nach Gl. (*19*) in Teil 1 folgt für die Standardunsicherheit $u_c(F)$ der Fläche F (Erweiterungsfaktor $k = 1$):

$$u_c(F) = \sqrt{(c_a \cdot u(a))^2 + (c_b \cdot u(b))^2},$$

$$u_c(F) = \sqrt{(20\ \text{m} \cdot 0{,}04\ \text{m})^2 + (80\ \text{m} \cdot 0{,}06\ \text{m})^2},$$

$$u_c(F) = \sqrt{23{,}68\ \text{m}^4},$$

$$\boxed{u_c(F) = \pm 4{,}866\ \text{m}^2 \quad (k = 1)}.$$

Hinweis: Aus Gründen der Vergleichbarkeit wurden hier mehr Nachkommastellen angegeben als sonst üblich.

Zur Anwendung der MC-Methode können die im Abschnitt 3.4.1 aufgelisteten Schritte mit modernen Programmiersprachen, wie z. B. Python, leicht realisiert werden. Das in *Abb. 15* wiedergegebene Programm-Listing in Python verdeutlicht die Vorgehensweise. In den Zeilen 2 bis 4 werden die zusätzlich erforderlichen Pakete numpy, matplotlib und time eingebunden. In den Zeilen 6 und 14 werden die Zeitpunkte zur Bestimmung der Berechnungszeit ermittelt. Zeile 7 legt die Anzahl der Ziehungen fest. In den Zei-

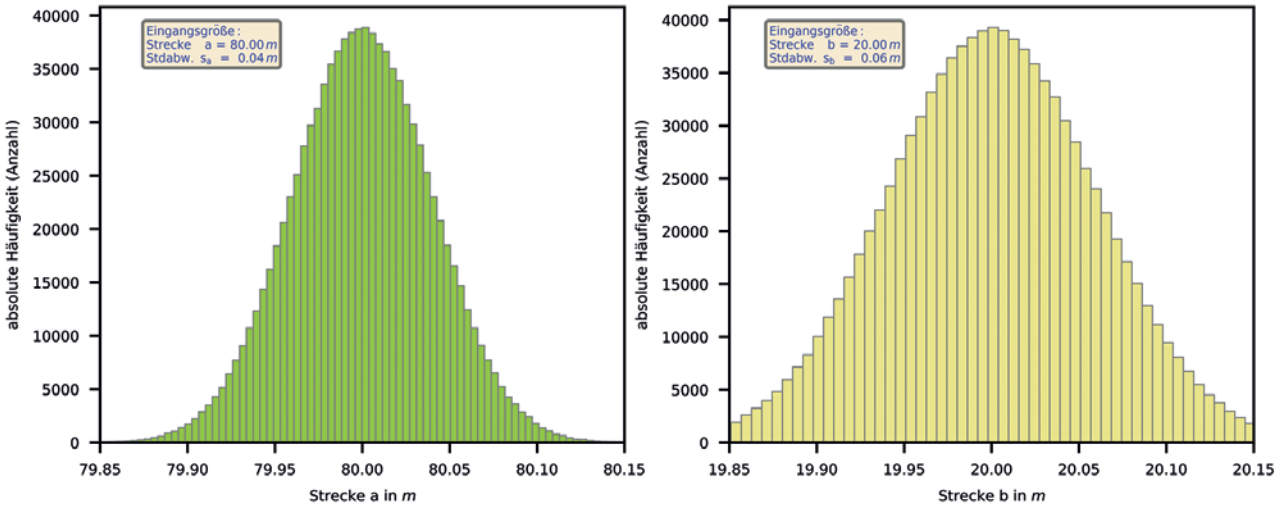

Abb. 16 | Absolute Häufigkeiten der Eingangsgrößen a und b (Maßstab der Achsen für die Strecken ist in beiden Abbildungen gleich)

len 8 und 9 werden die Vektoren für die Seitenlängen ***a*** und ***b*** berechnet. Dabei werden die jeweiligen Seitenlängen mit der Funktion *randn*(*m*, 1) aus dem numpy-Paket überlagert. Die Funktion *randn*() erzeugt *m* Werte, die normalverteilt sind. Um die mit der Standardabweichung = 1 normierten Werte auf die gewünschten Standardabweichungen anzupassen, werden die Funktionswerte mit den bekannten Standardunsicherheiten in den Zeilen 8 und 9 multipliziert. In der Zeile 10 wird mit der Berechnungsformel aus den Vektoren der Eingangsgrößen ***a*** und ***b*** der Vektor für die Einzelflächen ***yr*** gebildet. Die Berechnung des Mittelwerts aller Einzelflächen erfolgt entsprechend der Gl. (*29*) mit der sum()-Funktion von Python. In der Variablen *F* (Zeile 11) ist der Wert der Zielgröße, also der der Fläche, gespeichert. Die Standardunsicherheit *sF* wird aus der Varianz (Funktion var()) der Einzelflächen ***yr*** in Zeile 12 ermittelt. Python (Version 2.7) benötigt für die erforderlichen Berechnungen auf einem Rechner mit einer i7-6700-CPU (@ 3,40 GHz) unter Windows 7 Professional (x64) mit einem Arbeitsspeicher von 8 GB insgesamt 0,08 Sekunden.

Hinweis: Zufallszahlen können in Python für Gleichverteilungen mit der Funktion rand() und für Dreieckverteilungen mit der Funktion triangular() erzeugt werden.

In *Abb. 16* sind die mit dem Python-Programm berechneten absoluten Häufigkeiten der Eingangsgrößen ***a*** und ***b*** dargestellt. Die in den Abbildungen angegebenen Streckenlängen *a* und *b* mit ihren Standardunsicherheiten wurden aus den absoluten Häufigkeiten berechnet; sie sind mit den eingeführten Größen identisch und bestätigen damit die korrekte Bestimmung der absoluten Häufigkeiten entsprechend der gewählten Verteilungen mit dem Python-Programm.

In *Abb. 17* sind die nach der MC-Methode berechneten absoluten Häufigkeiten der Zielgröße „Fläche“ einschließlich der daraus abgeleiteten Größen für die Fläche und ihre Standardabweichung dargestellt. Ebenso wie die Eingangsgrößen ist auch die Zielgröße normalverteilt. Die Fläche wurde also zu 1 600,004 m² mit einer Standardunsicherheit von 4,871 m² für den Erweiterungsfaktor $k = 1$ bestimmt. Diese Werte stimmen mit den zu Beginn dieses Abschnitts nach dem klassischen GUM-Ansatz bestimmten Werten überein. Die gute Übereinstimmung ist auf die hohe Zahl von Stichproben ($m = 1\,000\,000$) und auch auf die nicht sehr ausgeprägte Nichtlinearität des funktionalen Zusammenhangs zurückzuführen.

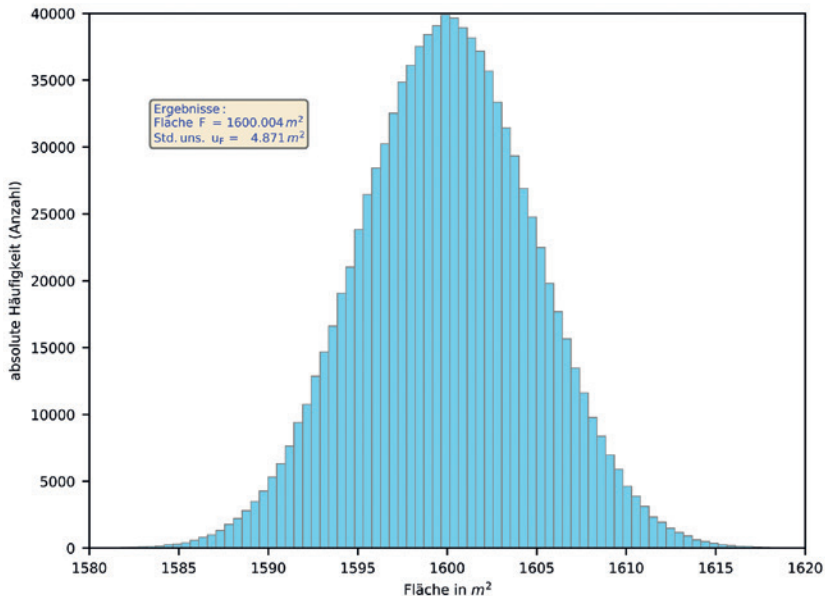

Abb. 17 | Nach der MC-Methode berechnete Wahrscheinlichkeitsdichte der Zielgröße „Fläche“

Hinweis: Auch hier wurden aus Gründen der Vergleichbarkeit mehr Nachkommastellen in den Ergebnissen angegeben als sonst üblich.

3.4.2 Zahlenbeispiel 2

Als weiteres Beispiel für die MC-Methode wird das im Abschnitt 3.3.4 (siehe Teil 1) zur Demonstration des klassischen GUM-Verfahrens benutzte Zahlenbeispiel verwendet. Die Verteilungen der Eingangsgrößen werden mit den Funktionen *randn*() für die Normalverteilung und *rand*() für die Rechteckverteilung aus dem *numpy*-Paket von Python simuliert. Der Programm-Code ist aus *Abb. 18* ersichtlich. In den Zeilen 8 bis 19 werden die Vektoren der Eingangsgrößen definiert, die jeweils mit den entsprechenden Verteilungen überlagert werden. Es werden wiederum für jede Eingangsgröße 1 000 000 Zahlenwerte generiert. In der Zeile 22 werden aus den

```
 1 # -*- coding: utf-8 -*-
 2 import numpy as np
 3 import matplotlib.pyplot as plt
 4 import time
 5
 6 rho = 200/np.pi
 7 start_time = time.time()
 8 m = 1000000                                    # Anzahl = Werte in der Stichprobe
 9 ha = 1061.32                                   # Höhe HA in m (DHHN2016-System)
10 r = 6380000.                                   # Erdradius in m
11 i = 1.45 + 0.0023*np.random.randn(m,1)         # Instrumentenhöhe in m, t- bzw. N-Vert.
12 t = 1.82 + 0.0023*np.random.randn(m,1)         # Reflektorhöhe in m, t- bzw. N-Vert.
13 d = 2545.139 + 0.002*np.random.randn(m,1)      # Schrägdistamz in m, N-Vert.
14 ka = 0.010 + 0.0005*np.random.randn(m,1)       # Additionskonstante in m, N-Vert.
15 km = 0.025 -0.0102/2 + 0.0102*np.random.rand(m,1)  # atmosphärische Korrektion in m, R-Vert.
16 kz = 0.002 -0.004/2  + 0.004*np.random.rand(m,1)   # zyklische Abweichung in m, R-Vert.
17 z = 64.7412/rho + 0.0002/rho*np.random.randn(m,1)  # Zenitwinkel in gon bzw. in rad, N-Vert.
18 kr = 0.13 -0.10/2 + 0.10*np.random.rand(m,1)    # Refraktionskoeffizient, R-Vert.
19 kg = 0.005 -0.010/2 + 0.010*np.random.rand(m,1)   # Geoidundulation in m, R-Vert.
20
21 # hb = Vektor mit den Höhen
22 hb = ha + i + (d+ka+km+kz)*np.cos(z) - t - kr*d**2/2/r + d**2*np.sin(z)**2/2/r + kg
23
24 B = hb.sum()/m # Mittelwert der Höhen
25 sB = np.sqrt(hb.var()) # StdAbw der Höhe
26 print B, sB # Ausgabe: Höhe und StdAbw
27 end_time = time.time()
28 print ("Berechnungszeit:  %.2f Sekunden" % (end_time - start_time))
```

Abb. 18 | Python-Code (Version 2.7) für die MC-Methode für das Zahlenbeispiel 2

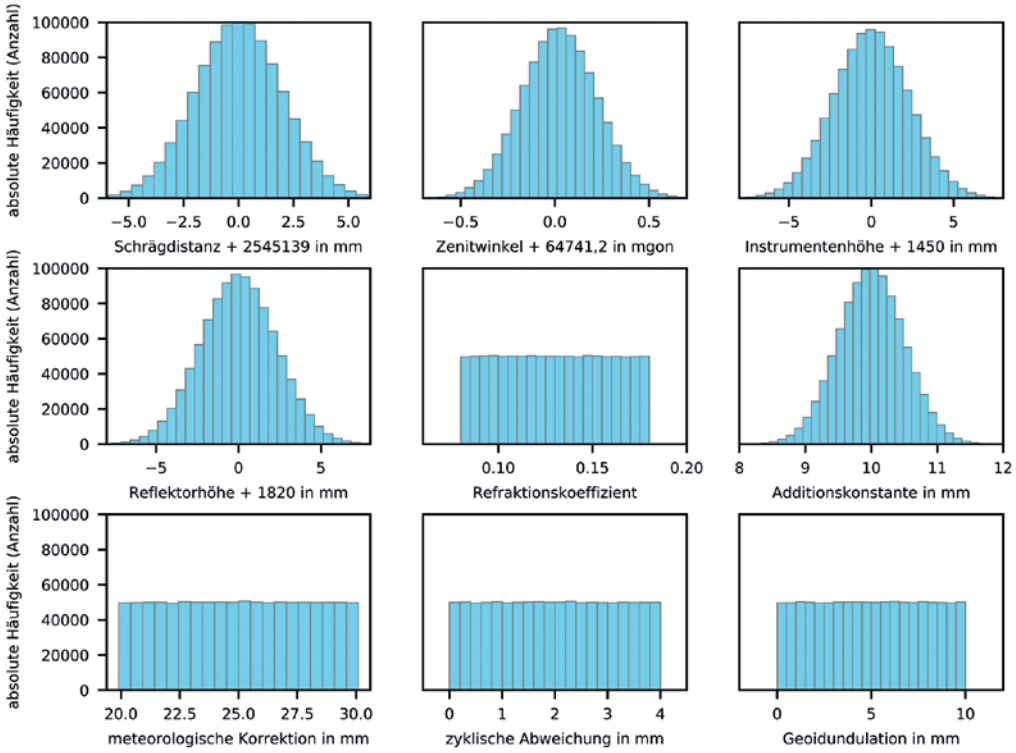

Abb. 19 | Wahrscheinlichkeitsdichten der Eingangsgrößen

erzeugten Eingangsgrößen der Vektor mit den Einzelhöhen des Punkts *B* sowie in den Zeilen 24 und 25 die Höhe des Punkts *B* als Mittelwert aller Einzelhöhen einschließlich seiner Standardabweichung berechnet.

In *Abb. 19* sind die Wahrscheinlichkeitsdichten aller Eingangsgrößen grafisch dargestellt.

Die Ergebnisse der Berechnungen einschließlich der Wahrscheinlichkeitsdichte für die Zielgröße „Höhe des Punkts *B*" sind aus *Abb. 20* ersichtlich. Die Höhe des Punkts *B* ergibt sich zu 2 399,9181 m mit einer Standardunsicherheit von ±0,0168 m für den Erweiterungsfaktor $k = 1$. Diese Ergebnisse stimmen mit den im Abschnitt 3.3.4 (siehe Teil 1) erhaltenen Werten sehr gut überein, wiederum wegen der hohen Strichprobenanzahl von $m = 1\,000\,000$. Visuell ist erkennbar, dass die Wahrscheinlichkeitsdichte (*Abb. 20*) nicht normalverteilt ist. Für die gesamten Simulationsberechnungen benötigt Python 0,38 Sekunden.

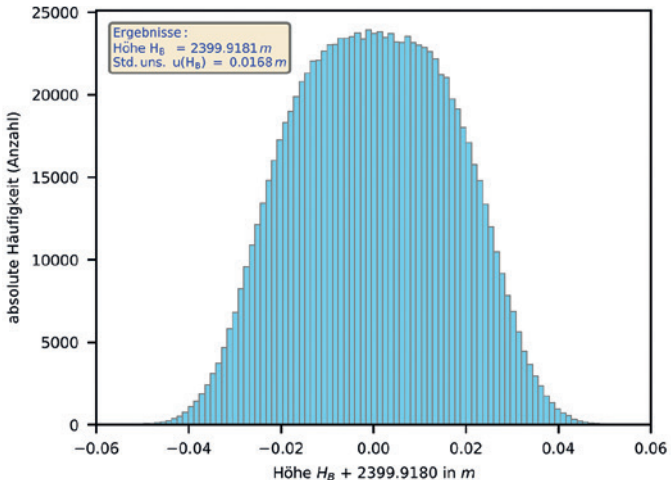

Abb. 20 | Nach der MC-Methode berechnete Wahrscheinlichkeitsdichte der Zielgröße „Höhe des Punkts *B*"

3.5 Messunsicherheitsbestimmung bei Ausgleichungsrechnungen

3.5.1 Vorgehensweise

Die MC-Methode (Abschnitt 3.4) ist auch geeignet (z. B. bei Ausgleichungen von geodätischen Netzen), die Informationen über die Verteilungen der Eingangsgrößen (Beobachtungen) auf die Zielgrößen (die Koordinaten der Neupunkte) zu übertragen. Dazu sind, einem Vorschlag von /Niemeier & Tengen 2017/ entsprechend, für jede Beobachtung, die in die Ausgleichung zumeist nach dem Gauß-Markov-Modell eingeführt werden, Teilbudgets aufzustellen. In jedem Teilbudget sind die Einflussgrößen, die sich auf die Qualität der Beobachtungen auswirken, den Typen A und B (vgl. Abschnitte 2 und 3.3 in Teil 1) zuzuordnen und zu quantifizieren. Die sich daraus ergebenden Standardunsicherheiten einer jeden Einflussgröße werden dann nach Abschnitt 3.3.2, Gl. (*19*) (siehe Teil 1) zur Standardunsicherheit der entsprechenden Beobachtung zusammengefasst, um damit die Gewichtsmatrix (für unkorrelierte Beobachtungen) zur Berechnung der Unbekannten nach dem Gauß-Markov-Modell aufzustellen.

Zur Übertragung der Wahrscheinlichkeitsdichten einer jeden Einflussgröße auf die Zielgrößen werden die einzelnen Beobachtungen mit den jeweiligen Verteilungen (in der Regel Normal- und Gleichverteilungen) durch rechnermäßig durchgeführte Zufallsexperimente überlagert. Die Anzahl der Zufallsexperimente *m* beträgt dabei, wie bereits ausgeführt, $m > 1\,000 \ldots 1\,000\,000$. Mit den Datensätzen der Zufallsexperimente werden nach der Gauß-Markov-Methode

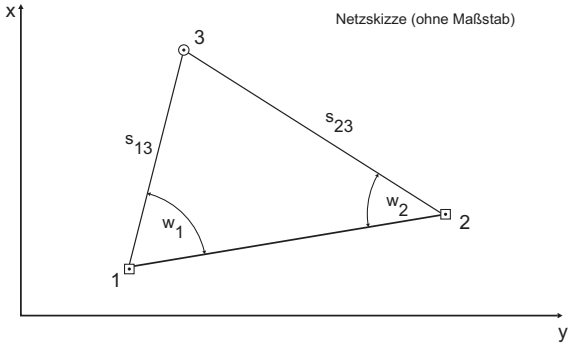

Abb. 21 | Netzskizze (ohne Maßstab)

die Zielgrößen, hier die Koordinatenunbekannten, berechnet. Aus den damit gewonnenen Wahrscheinlichkeitsdichten der Zielgrößen, den Koordinatenunbekannten, können ihre Standardunsicherheiten nach den Gln. (*29*) und (*30*) bestimmt werden.

Der hier aufgezeigte Ansatz ersetzt im Prinzip das stochastische Modell (Varianz-Kovarianz-Matrix) bei Ausgleichungsrechnungen. In den Standardunsicherheiten werden, der GUM-Methodik entsprechend, neben den zufälligen Messabweichungen auch durch den Messingenieur quantifizierte, nicht erfasste systematische Messabweichungen berücksichtigt und es werden für die Zielgrößen die Wahrscheinlichkeitsdichten bestimmt, um daraus die Standardunsicherheiten der Unbekannten als zutreffendere Genauigkeitsmaße im Vergleich zu den in der Ausgleichung berechneten Standardabweichungen abzuleiten.

3.5.2 Zahlenbeispiel

Das Zahlenbeispiel bezieht sich auf die Bestimmung der Koordinaten eines Neupunkts (Punkt 3 in *Abb. 21*) ausgehend von zwei gegebenen Festpunkten (Punkte 1 und 2 in *Abb. 21*) als Teil eines örtlichen, hochpräzisen Bauwerksüberwachungsnetzes. Gemessen werden auf den beiden Festpunkten jeweils die Horizontaldistanzen s_{13} und s_{23} zum Neupunkt und die Winkel w_1 und w_2.

Die Koordinaten der beiden Festpunkte und die Näherungskoordinaten des Neupunkts sind in *Tab. 6* zusammengestellt.

Die Beobachtungen (Strecken und Winkel) wurden direkt nacheinander über 20-mal ausgeführt, sodass die daraus berechneten Standardunsicherheiten der Mittelwerte als normalverteilt (Typ A) angesehen werden können.

Bei den Streckenmessungen sollen in diesem Demonstrationsbeispiel bis auf zyklische Abweichungen keine weiteren Einflussgrößen (wie z. B. Additionskonstante, Maßstabs- und meteorologische Korrekturen) wirksam bzw. zu bewerten sein. Die zyklischen Abweichungen wurden beim eingesetzten Tachymeter nicht durch eine Kalibrierung bestimmt. Der Hersteller versichert aber, dass die zyklischen Abweichungen bei diesem Instrumententyp unter normalen Bedingungen den Wert von ±2 mm nicht überschreiten. Da weitere Informationen zur Verteilung dieser Angaben fehlen, wird von einer Gleich-(Rechteck-)Verteilung ausgegangen. Die Halbbreite dieser Einflussgröße δ_{kz} ist also $a_{kz} = 2$ m mit dem Gewichtsfaktor $G_{kz} = \frac{1}{3}$ vom Typ B. Die Standardunsicherheit $u(k_z)$ beträgt damit nach Abschnitt 3.1.1.3

$$u(k_z) = \frac{2\ \text{mm}}{\sqrt{3}} = \pm 1{,}2\ \text{mm}.$$

Die Standardunsicherheiten der Mittelwerte der Distanzmessungen, berechnet aus den Verbesserungen zum Mittel einer jeden Beobachtung, sind $u(d) = \pm 0{,}8$ mm (Typ A) mit $G_d = 1$.

Bei den Winkelmessungen soll als weitere nicht durch eine Korrektion erfasste Einflussgröße eine Krümmung der Visurstrahlen bedingt durch Dichteunterschiede in der Atmosphäre (Refraktion) betrachtet werden. Es wird angenommen, dass aufgrund der örtlichen Gegebenheiten jeder Winkel um maximal ±0,4 mgon refraktionsmäßig beeinträchtigt sein kann. In Ermangelung weiterer Informationen wird wieder eine Gleichverteilung unterstellt. Die Halbbreite dieser Einflussgröße δ_r ist also $a_r = \pm 0{,}4$ mgon mit dem Gewichtsfaktor $G_r = \frac{1}{3}$ vom Typ B. Die Standardunsicherheit für den Refraktionseinfluss $u(r)$ beträgt somit

$$u(r) = \frac{0{,}4\ \text{mgon}}{\sqrt{3}} = \pm 0{,}2\ \text{mgon}.$$

Die Standardunsicherheiten der Mittelwerte der Winkelmessungen betragen $u(w) = \pm 0{,}2$ mgon (Typ A), $G_d = 1$.

Bei einer realistischen Genauigkeitsbewertung müssten noch einige andere, wesentliche Einflussgrößen mit einbezogen werden, worauf aber in diesem Zahlenbeispiel verzichtet wird. Es soll hier lediglich die Methode zur Berücksichtigung von Einflussgrößen des Typs B bei Ausgleichungen demonstriert werden.

In *Tab. 7* sind die Beobachtungen mit ihren Qualitätsangaben zusammengestellt.

Die Übertragung der Wahrscheinlichkeitsdichten der Einflussgrößen auf die Beobachtungen erfolgt mit dem aus *Abb. 22* ersichtlichen Python-Programm-Code. Die Vektoren ***msn***, ***msr***, ***msnr***, (Zeile 11 bis 13) sowie ***mwn***, ***mwr***, ***mwnr***, (Zeile 18 bis 20) enthalten die mit Zufallsoperatoren erzeugten Messwerte (m = 1 000 000) für die Normal- und die Gleichverteilungen sowie für die Summen beider Verteilungen.

In *Abb. 23* sind beispielhaft für die Strecke s_{13} und für den Winkel w_1 die Verhältnisse für die Wahrscheinlichkeitsdichten und deren Überlagerungen grafisch dargestellt.

Punkt	*y*	*x*
1	138,806 m	202,426 m
2	564,272 m	277,632 m
3	250,01 m	461,40 m

Tab. 6 | Koordinatenverzeichnis

Beobachtung	Mittelwert	Standard-unsicherheit des Mittels (Typ A)	Halbbreiten zykl. Abweichung bzw. Refraktion (Typ B)
s_{13}	281,8412 m	±0,8 mm	±2 mm
s_{23}	364,0518 m	±0,8 mm	±2 mm
w_1	63,0416 gon	±0,2 mgon	±0,4 mgon
w_2	44,8242 gon	±0,2 mgon	±0,4 mgon

Tab. 7 | Beobachtungen mit ihren Qualitätsangaben

```
# -*- coding: iso-8859-15 -*-
import numpy as np
import time
import winsound
import matplotlib.pyplot as plt

m = 1000000 # Anzahl der Zufallsexperimente
sd = 0.0008 # 0,8 mm, Standardunsicherheit des Mittelwertes der Distanzmessungen --> normal-verteilt
sz = 0.002 #  2 mm, Halbbreite zykl. Abweichungen --> rechteck-verteilt
s = 281.8412 # Strecke s12 in m
msn = s + sd * np.random.randn(m, 1) # Normal-Verteilung
msr = s - sz + sz*2*np.random.rand(m,1) # Gleich-Verteilung
msnr = s + sd * np.random.randn(m, 1) - sz + sz*2*np.random.rand(m,1)# Normal- und Gleich-Verteilung

sw = 0.0002 # 0,2 mgon, Standardunsicherheit des Mittelwertes der Winkelmessungen --> normal-verteilt
sr = 0.0004 # 0,4 mgon, Halbbreite der Refraktion --> rechteck-verteilt
w = 63.0416 # Winkel w1 in gon
mwn = w + sw * np.random.randn(m, 1) # Normal-Verteilung
mwr = w - sr + sr*2*np.random.rand(m,1) # Gleich-Verteilung
mwnr = w + sw * np.random.randn(m, 1) -sr + sr*2*np.random.rand(m,1)  # Normal- und Gleich-Verteilung
```

Abb. 22 | Programm-Code zur Übertragung der Wahrscheinlichkeitsdichten

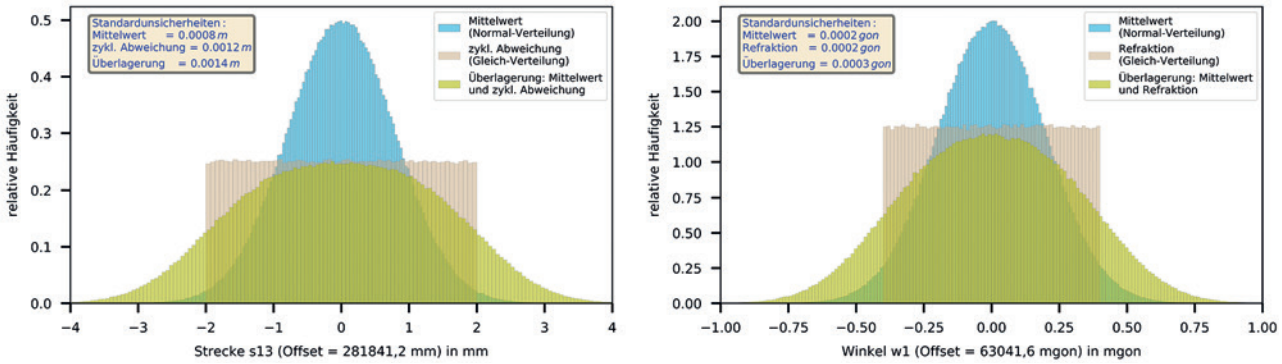

Abb. 23 | Wahrscheinlichkeitsdichten für die Strecke s_{13} und den Winkel w_1 (rechts)

Es werden jetzt für jeden der m Datensätze der Beobachtungen unter Anwendung des Gauß-Markov-Modells die Unbekannten $\boldsymbol{dy_3}$ und $\boldsymbol{dx_3}$ berechnet. *Abb. 24* zeigt einen Auszug des dafür verwendeten Python-Programms (Version 2.7). In den Zeilen 61 bis 64 werden die Datensätze mit den Beobachtungen erzeugt (Umfang der Stichproben $m = 10\,000$) und in den Zeilen 68 bis 74 erfolgt für jeden Datensatz die Berechnung der Unbekannten mit der vorher berechneten Inversen der Normalgleichungs- ($\boldsymbol{N}_{inv}$) und mit der Gewichtsmatrix $\boldsymbol{P}$. Alle berechneten Unbekannten werden im Vektor $\boldsymbol{u}$ (Zeile 74) gespeichert.

```
59 m = 10000 # Anzahl der Zufallsexperimente
60 # Vektoren der Beobachtungen: ms13, ms23, mw1 und mw2
61 ms13 = 281.8412 + 0.0008 * np.random.randn(m, 1) - 0.002 + 0.002*2*np.random.rand(m,1)
62 ms23 = 364.0518 + 0.0008 * np.random.randn(m, 1) - 0.002 + 0.002*2*np.random.rand(m,1)
63 mw1 = 63.0416 + 0.0002 * np.random.randn(m, 1) - 0.0004 + 0.0004*2*np.random.rand(m,1)
64 mw2 = 44.8242 + 0.0002 * np.random.randn(m, 1) - 0.0004 + 0.0004*2*np.random.rand(m,1)
65
66 u = np.array([]) # Definition einer Matrix für alle Unbekannten
67 f0 = np.array([s13, s23, w1, w2], float) # f(xo,yo, ...)
68 for i in range(0, m):
69     beob = np.array([ms13[i], ms23[i], mw1[i], mw2[i]], float) # Vektor mit den i_ten B
70     l = beob.transpose() - f0 # Absolutgliedvektor --> l = Beob. - f(xo,yo, ...)
71     x1 = np.dot(Ninv, A.transpose())
72     x2 = np.dot(x1, P)
73     x = np.dot(x2, l.transpose()) # Unbekannten des i_ten Datensatzes
74     u = np.append(u, x) # Vektor mit allen Unbekannten
75 u = u.reshape(m, 2) # neu formen
76 end_time = time.time()
77 winsound.Beep(2000, 80)
78 print ("Rechenzeit: %.2f Sekunden" % (end_time - start_time))
```

Abb. 24 | Python-Programm (Auszug) für die MC-Methode für das Zahlenbeispiel

In *Abb. 25* sind die Wahrscheinlichkeitsdichten der Unbekannten $\boldsymbol{dy_3}$ und $\boldsymbol{dx_3}$ (*Abb. 25* links und Mitte) sowie die Einzellösungen der beiden Unbekannten (*Abb. 25* rechts), berechnet aus dem Vektor $\boldsymbol{u}$, dargestellt. Das Python-Programm benötigt für die Berechnungen 0,13 Sekunden. Die aus den Wahrscheinlichkeitsdichten nach den Gln. (*29*) und (*30*) zu berechnenden Standardunsicherheiten der Unbekannten betragen für $u(dy_3) = \pm 1{,}0$ mm und für $u(dx_3) = \pm 1{,}1$ mm mit dem Erweiterungsfaktor $k = 1$. Der Korrelationskoeffizient $r(y_3, x_3)$ ergibt sich zu $r(y_3, x_3) = 0{,}1$ und drückt aus, dass zwischen den Unbekannten $\boldsymbol{dy_3}$ und $\boldsymbol{dx_3}$ nur eine sehr geringe Korrelation besteht. Die Unbekannten selbst betragen

$$y_3 = y_3^0 + dy_3 = (250{,}01 - 0{,}0017)\ \text{m} = 250{,}0083\ \text{m} \quad \text{und}$$

$$x_3 = x_3^0 + dx_3 = (461{,}40 + 0{,}0023)\ \text{m} = 461{,}4023\ \text{m}.$$

Eine klassische Berechnung der Unbekannten mit ihren Standardabweichungen in einer Ausgleichung nach dem Gauß-Markov-Modell ergibt die in *Tab. 8* zusammengestellten Angaben.

Danach sind die nach dem GUM-Ansatz und nach dem Gauß-Markov-Modell berechneten Koordinatenunbekannten identisch, während ihre nach dem GUM-Ansatz berechneten Standardunsicherheiten im Prinzip um den Faktor 2 größer sind als die nach dem Gauß-Markov-Modell berechneten Standardabweichungen.

Bezeichnung der Unbekannten	Ausgegl. Unbekannte	Standard-abweichung
y_3	250,0083 m	0,5 mm
x_3	461,4023 m	0,6 mm

Tab. 8 | Ergebnisse nach dem Gauß-Markov-Modell

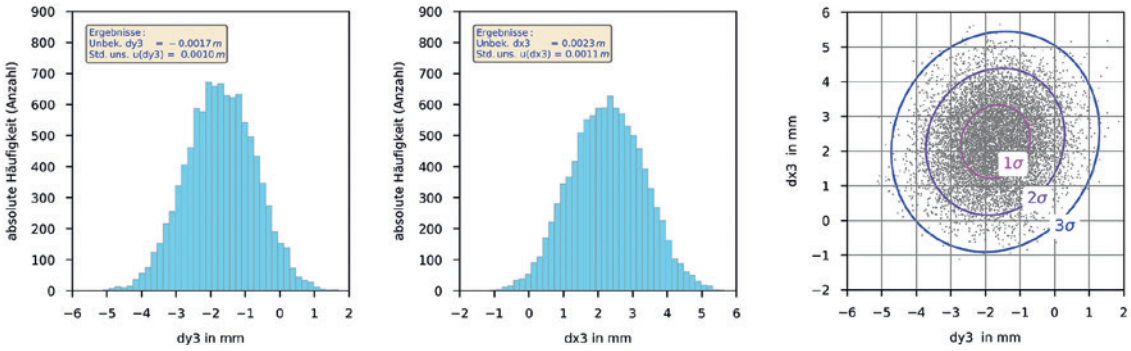

Abb. 25 | Wahrscheinlichkeitsdichten der Unbekannten dy_3 und dy_3 (links und Mitte); Einzellösungen der Unbekannten dy_3 und dy_3 mit den GUM-Kovarianz-Ellipsen 1σ, 2σ und 3σ (rechts)

```
85 # Überdeckungswahrscheinlichkeit (Verteilungsfunktion) für dy3
86 dy3=u[:,0] # in m
87 dy3= np.sort(dy3)
88 # Plott-Funktion aufrufen
89 anz, bins, patches = plt.hist(dy3, bins=300, color ='skyblue',linewidth=0.1, normed=True,cumulative=Tru
90 x=bins[0:len(bins)-1]
91 y=anz
92 plt.close() # Plott wird wieder gelöscht --> Vektoren m und bins werden gespeichert
93 # untere Grenze bestimmen
94 alpha=0.683
95 alpha1 = (1-alpha)/2
96 alpha2 = alpha+(1-alpha)/2
97 for i in range(0,len(y)):
98     if y[i]>(1-alpha)/2:
99         y3y_min = y[i]
100        y3x_min = x[i] # Quantile 1 in m
101        break
102 # obere Grenze bestimmen
103 for i in range(0, len(y)):
104     if y[i] > alpha+(1-alpha)/2:
105         y3y_max = y[i]
106         y3x_max = x[i] # Quantile 2 in m
107         break
108 print y3x_min # Quantile 1 in m
109 print y3x_max # Quantile 2 in m
```

Abb. 26 | Python-Code (Ausschnitt) zur Bestimmung der Quantil-Werte

Die Standardunsicherheiten beschreiben die Genauigkeitssituation zutreffender, weil u. a. Einflussgrößen berücksichtigt worden sind, die bei der normalen Ausgleichung gewöhnlich entfallen.

Es können auch Quantil-Werte für gewählte Überdeckungswahrscheinlichkeiten aus den empirisch bestimmten Wahrscheinlichkeitsdichten bestimmt werden. Anhand der Koordinatenunbekannten $\boldsymbol{dy_3}$, die hier als eindimensionale Zufallsvariable betrachtet wird, soll dies exemplarisch mit einem Python-Code (*Abb. 26*) demonstriert werden. In der Zeile 86 werden die zuvor berechneten dy_3-Unbekannten (abgespeichert in der Liste $\boldsymbol{u}$) dem Vektor $\boldsymbol{dy_3}$ zugeordnet. In der Zeile 87 werden die Unbekannten ihrer Größe nach sortiert. Mit dem Plott-Aufruf in Zeile 89 werden durch den Parameter „*bins* = 300“ 300 Klassen erzeugt und mit dem Parameter „*cumulative* = True“ wird eine fortwährende Aufsummierung der Anzahl der den einzelnen Klassen zugeordneten Einzelereignisse vorgenommen. Durch die fortlaufende Aufsummierung wird die Verteilungsfunktion für die Unbekannte $\boldsymbol{dy_3}$ empirisch ermittelt. Die zurückgegebene Variable *bins* enthält alle Klassengrenzen (in Zeile 90 in der Variablen $\boldsymbol{x}$ gespeichert) und die Variable $\boldsymbol{anz}$ die Anzahl der in jeder Klasse vorhandenen Einzelereignisse (in Zeile 91 in der Variablen $\boldsymbol{y}$ gespeichert). Der Wert *alpha* in Zeile 94 legt die gewählte Überdeckungswahrscheinlich fest (hier: 0,683 = 68,3 % für eine symmetrische Überdeckung), zu der der untere und der obere Quantil-Wert bestimmt werden sollen. In den Zeilen 97 bis 107 werden die Quantil-Werte durch entsprechende Abfragen ermittelt.

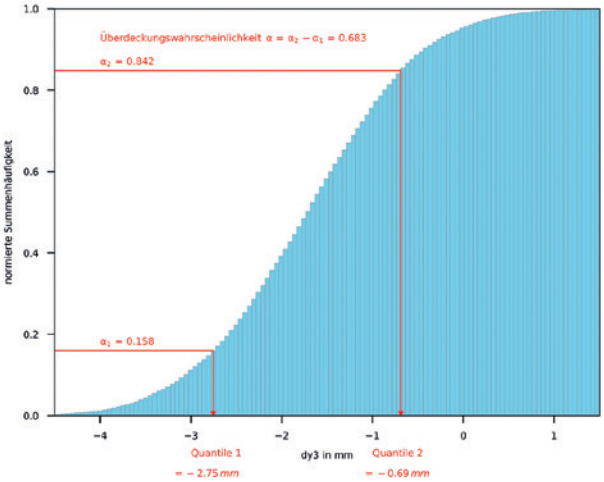

Abb. 27 | Quantil-Werte für die Koordinatenunbekannte dy_3 für eine symmetrische Überdeckungswahrscheinlichkeit von 68,3 %

In *Abb. 27* sind die Ergebnisse einschließlich der empirisch ermittelten Verteilungsfunktion für die Koordinatenunbekannte $\boldsymbol{dy_3}$ exemplarisch dargestellt.

Weiterhin lassen sich aus den berechneten Unbekannten $\boldsymbol{dy_3}$ und $\boldsymbol{dx_3}$ (abgespeichert im Vektor $\boldsymbol{u}$) als zweidimensionaler Zufallsvektor die Kovarianz-Matrix $\boldsymbol{Cov}(x_i, x_j)$, z. B. nach /Niemeier 2002, S. 20 f./, in einer angepassten Nomenklatur

$$\boldsymbol{Cov}\left(x_i, x_j\right) = E\left(\left(\boldsymbol{x}_i - \mu(x_i)\right)\left(\boldsymbol{x}_j - \mu(x_j)\right)^{\mathrm{T}}\right) \qquad (31)$$

mit den beiden Zufallsvariablen $\boldsymbol{x}_i$ und $\boldsymbol{x}_j$ sowie deren Erwartungswerte $\mu(x_i)$ und $\mu(x_j)$ berechnen. Die Kovarianz-Matrix ergibt sich zu

$$\boldsymbol{Cov}(x_i, x_j) = \begin{bmatrix} (u(x_i))^2 & u(x_i, x_j) \\ u(x_i, x_j) & (u(x_j))^2 \end{bmatrix} \quad (32)$$

mit den quadrierten Unsicherheiten $(u(x_i))^2$ und $(u(x_j))^2$ sowie der Kovarianz $u(x_i, x_i)$. Die Berechnung des (empirischen) Korrelationskoeffizienten $r(x_i, x_i)$ erfolgt nach Gl. (*20*) (siehe Teil 1).

Mit dem Python-Code nach *Abb. 28* ergibt sich die Kovarianz-Matrix der Unbekannten $\boldsymbol{dy_3}$ und $\boldsymbol{dx_3}$ für das Zahlenbeispiel zu

$$\boldsymbol{Cov}(dy_3, dx_3) = \begin{bmatrix} 1{,}007286 & 0{,}095669 \\ 0{,}095669 & 1{,}144441 \end{bmatrix}. \quad (33)$$

```
192 x = u[:,0]*1000 # x --> Vektor mit den dy3-Werten in mm
193 y = u[:,1]*1000 # y --> Vektor mit den dx3-Werten in mm
194 cov = np.cov(x, y) # Kovarianz-Matrix
```

Abb. 28 | Python-Code (Ausschnitt) zur Berechnung der Kovarianz-Matrix

Mit den Elementen der Kovarianz-Matrix können punktbezogene Kovarianz- oder auch Konfidenz-Ellipsen abgeleitet werden. Die Kovarianz-Ellipsen werden auch als Helmertsche Fehler-Ellipsen bezeichnet. In *Abb. 25* (rechts) sind exemplarisch die GUM-Kovarianz-Ellipsen ($1\,\sigma$, $2\,\sigma$ und $3\,\sigma$) für den Neupunkt 3 nach /Niemeier 2002, S. 256 f./ eingezeichnet.

In der *Tab. 9* (mittlere Spalte) ist in Prozenten angegeben, wie viele Punkte der *Abb. 25* (rechts) sich innerhalb der jeweiligen GUM-Kovarianz-Ellipse befinden. Vergleichsweise sind in dieser Tabelle (rechte Spalte) auch die theoretischen Wahrscheinlichkeiten für zweidimensionale, normalverteilte Zufallsvariable mit den Parametern $\sigma_1 = \sigma_2 = \sigma$ und mit dem Korrelationskoeffizienten $\rho = 0$ nach Gl. (*16*) (siehe Teil 1) eingetragen. So berechnet sich

Kovarianz-Ellipse	Empirische Punktmenge	Wahrscheinlichkeit für zweidimensionale Normalverteilung
$1\,\sigma$	38,1 %	39,3 %
$2\,\sigma$	86,5 %	86,5 %
$3\,\sigma$	99,2 %	98,9 %

Tab. 9 | Prozentuale Anteile der Punktmengen innerhalb der Kovarianz-Ellipsen

```
1  from scipy import integrate
2  import numpy as np
3
4  def f(x,y):
5      return 1/(2*np.pi*s**2)*np.exp(-1/2*((x**2+y**2)/s**2))
6  def bounds_x(y):
7      return [0,np.sqrt(r**2-y**2)]
8  def bounds_y():
9      return [0,r]
10
11 s=1.   # Standardabweichung s = 1 mm
12 r=1*s  # Integrationsradius 1 Sigma
13 erg = integrate.nquad(f, [bounds_x, bounds_y])
14 print(erg[0]*4)
15 r=2*s  # Integrationsradius 2 Sigma
16 erg = integrate.nquad(f, [bounds_x, bounds_y])
17 print(erg[0]*4)
18 r=3*s  # Integrationsradius 3 Sigma
19 erg = integrate.nquad(f, [bounds_x, bounds_y])
20 print(erg[0]*4)
```

Abb. 29 | Python-Code (Version 3.8) zur Berechnung der Wahrscheinlichkeiten zweidimensionaler Zufallsgrößen

für die 1σ-Ellipse nach /Brandt 2013, S. 94/ die theoretische Wahrscheinlichkeit zu $1 - 1/\sqrt{e} = 0{,}393 = 39{,}3$ % (auch nach /Niemeier 2002, S. 260/). Die theoretischen Wahrscheinlichkeiten für alle Kovarianz-Ellipsen können auch nach Gl. (*16*) (siehe Teil 1) mit dem in der *Abb. 29* wiedergegebenen Python-Programm mit den im scipy-Paket implementierten Approximationsalgorithmen für die zweifache Integration berechnet werden.

Die empirischen Werte nach *Tab. 9* stimmen allesamt gut mit den theoretischen Werten einer zweidimensionalen Normalverteilung überein.

4 GUM: PRO & CONTRA

In diesem Absatz wird diskutiert, welche Gründe für die Anwendung des GUM-Konzepts sprechen und welche dagegen.

Pro:

1. Mit dem GUM-Konzept werden die Messungen und die daraus abgeleiteten Größen nach einem einheitlichen Verfahren bewertet und dokumentiert.
2. Die Genauigkeitsbewertungen sind somit auch international vergleichbar, transparent und interpretierbar. Genauigkeitsbewertungen werden objektiver.
3. Die Bewertungen nach GUM sind realitätsnäher als bisherige Bewertungsmethoden, da nicht durch eine Korrektur erfasste systematische Messabweichungen im Genauigkeitsmaß berücksichtigt werden.
4. Beim klassischen GUM-Ansatz gehen Informationen über die Verteilungen der Eingangsgrößen und damit auch über die resultierenden Ausgangsgrößen verloren. Abhilfe hat hier aber die mit „GUM Supplement 1“ eingeführte numerische Simulation mithilfe der Monte-Carlo-Methode /DIN V ENV 13005 Beiblatt 1/ geschaffen.
5. Die GUM-Methode stellt eine Schnittstelle zwischen den Disziplinen her und reduziert die Gefahr von Fehlinterpretationen.

Contra:

1. Der wahre Wert wird durch die GUM-Methode nicht lokalisiert.
2. Die Grundprinzipien des GUM entsprechen nicht der Theorie der mathematischen Statistik (z. B. /Schmidt 2003/), vor allem nicht bei der Behandlung nicht erfasster systematischer Messabweichungen.
3. Der GUM trifft derzeit noch keine Festlegungen, in welcher Form empirisch bestimmte Wahrscheinlichkeitsdichten bzw. Verteilungsfunktionen übermittelt werden sollen.

5 ZUSAMMENFASSUNG

Der GUM eröffnet die Möglichkeit, Genauigkeitsmaße für Messungen und für daraus abgeleitete Größen nicht nur aus den zufälligen Messabweichungen zu berechnen, sondern es können auch *nicht erfasste systematische Messabweichungen* berücksichtigt werden.

Bevor auf die Bestimmung der Messunsicherheit nach GUM in diesem Beitrag eingegangen wurde, waren die Begriffe Auflösung, Präzision, Richtigkeit sowie Genauigkeit zu erläutern. Oftmals wird

der Begriff *Genauigkeit* nicht seiner Definition entsprechend verwendet. Wenn man in der Praxis von Genauigkeit spricht, handelt es sich eigentlich (nur) um die Wiederholgenauigkeit von Messungen, also um die Präzision, ohne dass systematische Messabweichungen berücksichtigt werden. Bei der *Genauigkeit* hingegen ist stets der Bezug zu einem Referenzwert, das ist in der Regel der *wahre Wert*, herzustellen, also auch unter Beachtung systematischer Messabweichungen.

Weiterhin werden in diesem Beitrag einige wichtige Grundlagen über Wahrscheinlichkeitsdichtefunktionen, wie z. B. Normal- bzw. Student-, Rechteck- und Dreieckverteilung und deren Parameter, diskutiert, weil sie beim GUM Anwendung finden. Da in das Genauigkeitsmaß des GUM systematische Messabweichungen einfließen, werden diese an einigen typischen Beispielen erläutert.

Der GUM kennt zwei Kategorien, um Messunsicherheiten zu bestimmen, und zwar Typ A: Berechnung der Messunsicherheit durch statistische Analyse der Messungen und Typ B: Berechnung der Messunsicherheit mit anderen Mitteln als der statistischen Analyse. Beim Typ A werden also die bekannten klassischen Verfahren der deskriptiven Statistik eingesetzt, wobei in der Regel die Messungen einer Normalverteilung folgen. Es lassen sich Standardabweichungen berechnen, die nach GUM als Standardunsicherheiten bezeichnet werden.

Beim Typ B hingegen – hierunter fallen die *nicht erfassten systematischen Messabweichungen* – wird durch Abschätzung versucht, die maximale Größe dieser Abweichungen zu bestimmen. Anschließend werden diese Abweichungen einer Verteilung zugeordnet, sodass dann aus den Parametern der Verteilungsfunktion letztlich die zu bestimmende Größe mit ihrer Standardunsicherheit abgeleitet werden kann. Zumeist liegen keine spezifischen Informationen über die Verteilungsfunktion vor, sodass man dann von einer Gleichverteilung (Rechteckverteilung) ausgeht.

Die kombinierte Messunsicherheit $u_c(y)$ ergibt sich schließlich als GUM-Festlegung durch eine quadratische Zusammenfassung aller Standardunsicherheiten $u_i(y)$ der Typen A und B. Um die Messunsicherheit mit einem gewünschten Vertrauensniveau zu verbinden, wird der Erweiterungsfaktor $k = 1, 2, 3$ oder 4 eingeführt, mit dem die (einfache) Messunsicherheit $u_c(y)$ multipliziert wird. Man erhält die *erweiterte Messunsicherheit* $u_{c(k=2)}$ oder $u_{c(k=3)}$ oder $u_{c(k=4)}$. An Zahlenbeispielen werden die einzelnen Schritte einer Messunsicherheitsanalyse eingehend erläutert.

Neben der Übertragung von Standardunsicherheiten nach der klassischen GUM-Methode (Typ A und Typ B) eröffnet das im Jahr 2008 eingeführte GUM-Supplement 1 /DIN V ENV 13005 Beiblatt 1 (Vornorm-Entwurf)/ auch die Fortpflanzung von Wahrscheinlichkeitsverteilungen unter Verwendung der Monte-Carlo-Methode auf die zu bestimmenden Größen. Damit wird die Bestimmung der Messunsicherheit wesentlich flexibler.

Die Monte-Carlo-Methode kann auch verwendet werden, um im Rahmen von Ausgleichungen (z. B. nach dem Gauß-Markov-Modell) nicht erfasste systematische Messabweichungen bzw. beliebige Verteilungen der Eingangsgrößen auf die Zielgrößen (die zu bestimmenden Koordinaten der Neupunkte) zu übertragen.

Mit dem GUM-Konzept werden die Messungen und die daraus abgeleiteten Größen nach einem einheitlichen Verfahren bewertet und dokumentiert. Die Genauigkeitsbewertungen nach GUM sind somit international vergleichbar, transparent und interpretierbar. Genauigkeitsbewertungen werden objektiver. Zudem ist die Messunsicherheit nach GUM realitätsnäher als nach bisherigen Bewertungsmethoden, da nicht durch eine Korrektur erfasste systematische Messabweichungen im Genauigkeitsmaß berücksichtigt werden. Weiterhin haben die GUM-Methoden eine Schnittstellenfunktion zwischen den an einem Projekt beteiligten Disziplinen. Die Genauigkeitsmaße werden nach einem festgelegten Verfahren bestimmt. Dadurch wird die Gefahr von Fehlinterpretationen reduziert.

Auf der anderen Seite werden die Methoden des GUM auch kritisiert. So wird der wahre Wert mit GUM nicht lokalisiert. Die Grundprinzipien des GUM entsprechen nicht der Theorie der mathematischen Statistik, vor allem nicht bei der Behandlung nicht erfasster systematischer Messabweichungen. Auch wenn aufgrund der Einsprüche von mathematischer Seite der DIN-Entwurf /DIN V ENV 13005:1999-06/ ersatzlos zurückgenommen worden ist und nicht als DIN-Norm erscheinen wird, haben die GUM-Methoden dennoch Eingang in das nationale und internationale Normenwerk – wie bereits in der Einführung angesprochen – gefunden. Weiterhin werden die Prinzipien des GUM im Rahmen der Bestimmungen des Deutschen Kalibrierdiensts (DKD) und der Eichgesetze in Deutschland für verbindlich erklärt, sodass der GUM auch darüber Eingang in die Praxis bekommt und an aktueller Bedeutung gewinnt. Aus diesen Gründen ist es unerlässlich, dass der GUM zukünftig in der Geodäsie und dabei speziell in der Ingenieurgeodäsie sowohl in der Lehre als auch in der Praxis als eine Methode zur Ableitung zutreffender Genauigkeitsmaße mehr Beachtung erfährt.

LITERATUR

Brandt, S. (2013): Datenanalyse für Naturwissenschaftler und Ingenieure. 5. Aufl. Springer, Berlin/Heidelberg.

DIN V ENV 13005:1999-06: Leitfaden zur Angabe der Unsicherheit beim Messen. ENV 13005. Beuth, Berlin (Vornorm wurde zurückgezogen).

DIN V ENV 13005 Beiblatt 1 (Vornorm-Entwurf): Leitfaden zur Angabe der Unsicherheit beim Messen – Beiblatt 1: Fortpflanzung von Verteilungen unter Verwendung einer Monte-Carlo-Methode, deutsche Übersetzung des GUM S1, 2010. Beuth, Berlin.

Niemeier, W. (2002): Ausgleichungsrechnung. W. de Gruyter, Berlin/New York.

Niemeier, W.; Tengen, D. (2017): Uncertainty assessment in geodetic network adjustment by combining GUM and Monte-Carlo-simulations. In: Journal of Applied Geodesy 11(2017)2, 67–76.

Schmidt, H. (2003): Warum GUM? Kritische Anmerkungen zur Normdefinition der „Messunsicherheit“ und zu verzerrten „Elementarfehlermodellen“. In: Zeitschrift für Vermessungswesen (zfv) 128(2003)5, 303–312.

Prof. Dr.-Ing. Willfried Schwarz

Malerstieg 12 | 99425 Weimar
willfried.schwarz@gmx.net