

Name: Marius Pfeiffer

Matrikel-Nr.: 4188573

E-Mail: marius.pfeiffer@stud.uni-heidelberg.de

Betreut durch: Dimitrij Bathauer

10.12.2024

---

## **Versuch 223: Bestimmung der Boltzmannkonstante mit Hilfe der Brownschen Bewegung**



Abbildung 1: Versuchsaufbau

## **Inhaltsverzeichnis**

<b>1 Einleitung</b>	<b>2</b>
1.1 Physikalische Grundlagen . . . . .	2
1.2 Versuchsdurchführung . . . . .	5
<b>2 Messprotokoll</b>	<b>7</b>
<b>3 Auswertung</b>	<b>8</b>
3.1 Berechnung des mittleren Verschiebungskadrates und dessen Fehler . . . . .	8
3.2 Kontrollverteilung . . . . .	10
3.3 Kumulative Verteilung der Verschiebungskadrates . . . . .	10
<b>4 Zusammenfassung und Diskussion</b>	<b>12</b>

# 1 Einleitung

Bereits im Jahr 1827 war dem schottischen Botaniker Robert Brown aufgefallen, dass Blütenpollen in einem Glas Wasser eigenartige Zickzackbewegungen ausführen. Erst nahezu 100 Jahre später erkannte Albert Einstein, dass die Bewegungen auf fortwährende Stöße der Wassermoleküle zurückzuführen sind. Während Einstein damit ein weiteres Argument für die Existenz von Atomen und Molekülen lieferte, war seine Beobachtung auch eine Evidenz für die molekulare Theorie der Wärme. Die mittlere Geschwindigkeit der Wassermoleküle hängt demnach von der Temperatur des Wassers ab. Erst im Jahre 1926 konnte der französische Physiker Jean-Baptiste Perrin die Brown'sche Molekularbewegung experimentell mit hoher Genauigkeit bestätigen, wofür er auch den Physik-Nobelpreis erhielt.

In Versuch 223 beobachten wir mit einem Mikroskop die Brown'sche Bewegung von in Wasser suspendierten Latexpartikeln. Wir werden die statistische Bewegung untersuchen und anhand der Bahn eines einzelnen Teilchens die Boltzmannkonstante bestimmen.

## 1.1 Physikalische Grundlagen

Als vereinfachtes Modell der Brown'schen Bewegung betrachten wir einen eindimensionalen Random-Walk. Wir stellen uns also ein einzelnes, freies Teilchen vor, welches sich auf einer Linie bewegen kann. Alls  $\tau$  Sekunden erfährt das Teilchen einen Stoß, also  $n = \frac{t}{\tau}$  Stöße in einer Zeit  $t$ . Bei jedem Stoß besteht die gleiche Wahrscheinlichkeit  $p = \frac{1}{2}$ , dass das Teilchen um die Distanz  $\delta$  entweder nach links ( $-\delta$ ) oder nach rechts ( $+\delta$ ) verschoben wird. Um nach  $n$  Stößen an der Position  $m\delta$  zu sein, muss sich das Teilchen somit  $\frac{n+m}{2}$ -mal nach rechts und  $\frac{n-m}{2}$ -mal nach links bewegt haben. Um die Position  $x = m\delta$  zu erreichen gibt es

$$\binom{n}{\frac{1}{2}(n+m)} = \frac{n!}{(\frac{1}{2}(n+m))!(\frac{1}{2}(n-m))!} \quad (1)$$

Wege, die das Teilchen gelaufen sein könnte. Die Wahrscheinlichkeit, dass sich das Teilchen nach  $n$  Stößen an der Position  $x = m\delta$  befindet ist binomialverteilt nach

$$P(m; n) = \binom{n}{\frac{1}{2}(n+m)} p^{\frac{n+m}{2}} (1-p)^{\frac{n-m}{2}}. \quad (2)$$

Die Wahrscheinlichkeit  $p$  für einen Sprung nach links bzw. nach rechts ist für beide Richtungen  $p = \frac{1}{2}$ . Somit können wir die Wahrscheinlichkeitsfunktion vereinfachen zu

$$P(m; n) = \frac{n!}{(\frac{1}{2}(n+m))!(\frac{1}{2}(n-m))!} \left(\frac{1}{2}\right)^n. \quad (3)$$

Da das Zeitintervall  $\tau$  zwischen den Stößen sehr klein ist, wird  $n = \frac{t}{\tau}$  sehr groß. Somit können  $n!$  und  $m!$  mit der Stirling'schen Formel

$$n! = (2\pi n)^{\frac{1}{2}} n^n e^{-n} \quad (4)$$

angenähert und die Wahrscheinlichkeitsfunktion zu

$$P(m; n) = \sqrt{\frac{2}{\pi n}} e^{-\frac{m^2}{2n}} \quad (5)$$

umgeformt werden

Wir möchten nun von der diskreten Verteilung der Variablen  $m$  und  $n$  zu einer kontinuierlichen Verteilung für die leichter messbaren Variablen  $x$  und  $t$  übergehen.  $m$  ist entweder gerade oder ungerade, somit ist  $\Delta m = \pm 2$ . Betrachtet auf ein Intervall  $[x, x + \Delta x]$ , kann sich ein Teilchen an  $\frac{\Delta x}{\Delta m \delta} = \frac{\Delta x}{2\delta}$  verschiedenen Positionen befinden. Wir können die Variable  $m$  also entsprechend

$$P(m; n) \frac{\Delta x}{2\delta} = P(x; n) \Delta x \quad (6)$$

substituieren. Weiter setzen wir  $n = t/\tau$ ,  $m = x/\delta$  und führen den Diffusionskoeffizienten

$$D = \frac{\delta^2}{2\tau} \quad (7)$$

ein. Damit erhalten wir die Funktion

$$P(x; t) \Delta x = \frac{\Delta x}{\sqrt{4\pi D t}} e^{-\frac{x^2}{4Dt}} \quad (8)$$

für die Wahrscheinlichkeit, dass sich ein Teilchen nach der Zeit  $t$  im Intervall  $[x, x + \Delta x]$  befindet. Setzen wir

$$\mu = \langle x \rangle = \int_{-\infty}^{\infty} x P(x; t) dx = 0, \quad (9)$$

$$\sigma^2 = \langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 P(x; t) dx = 2Dt, \quad (10)$$

so lässt sich diese mit einer Gaussverteilung

$$G(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (11)$$

um  $x = 0$  identifizieren. Aus der Definition der Varianz  $\sigma$  ergibt sich sogleich die Einstein-Smoluchowski-Gleichung

$$\sqrt{\langle x^2 \rangle} = \sqrt{2Dt}. \quad (12)$$

Welche besagt, dass der mittlere Abstand eines Teilchens von dessen Ursprungsort mit  $\sqrt{t}$  zunimmt.

Diese Gesetzmäßigkeiten lassen sich leicht von der bisherigen eindimensionalen Betrachtung auf höhere Dimensionen erweitern. Es gilt dann für das mittlere Verschiebungskquadrat

$$\langle r^2 \rangle = \langle x^2 \rangle + \langle y^2 \rangle. \quad (13)$$

Durch die Isotropie der Brown'schen Bewegung gilt im zweidimensionalen Fall  $\sqrt{\langle r^2 \rangle} = \sqrt{4Dt}$  und im dreidimensionalen Fall  $\sqrt{\langle r^2 \rangle} = \sqrt{6Dt}$ . Nach Einstein gilt für den zuvor eingeführten Diffusionskoeffizienten

$$D = \frac{kT}{f}. \quad (14)$$

Dieser gibt die Beweglichkeit eines Teilchens, abhängig von der Boltzmannkonstante  $k$ , der Temperatur  $T$  und dem Reibungskoeffizienten  $f$  an. Letzterer lässt sich für kugelförmige

Partikel mit dem Radius  $a$ , suspendiert in einer Flüssigkeit mit Viskosität  $\eta$  nach dem Stokes'schen Gesetz

$$f = 6\pi\eta a \quad (15)$$

berechnen. Es gilt also

$$D = \frac{kT}{6\pi\eta a}. \quad (16)$$

Setzen wir dies in die Definition des mittleren Verschiebungsquadrats in zwei Dimensionen ein, so erhalten wir

$$\langle r^2 \rangle = \frac{4kT}{6\pi\eta a} t \quad (17)$$

und schließlich

$$k = \frac{6\pi\eta a}{4Tt} \langle r^2 \rangle \quad (18)$$

zur experimentellen Bestimmung der Boltzmannkonstante  $k$ .

## 1.2 Versuchsdurchführung

Wie eingangs bereits erwähnt, betrachten wir in diesem Versuch in Wasser suspendierte Latexpartikel Mikroskop. Der erste Versuchsteil umfasst die Herstellung der Probe, während es in den nachfolgenden Teilen um die Erfassung der Bewegung, sowie die Eichung der Software geht.

**Herstellen der Probe suspendierter Latexpartikel.** Bevor wir mit der eigentlichen Messung beginnen können, müssen wir eine Probe auf einem Objektivträger präparieren. Dieser widmen wir uns im ersten Versuchsteil. Hierzu kleben wir ein Stück doppelseitiges Klebeband, aus dessen Mitte wir ein Loch ausgestanzt hatten, auf den Objektivträger. In den ausgestanzten Bereich pipettieren wir ca.  $150\mu\text{L}$  der Probeflüssigkeit und kleben dann ein weiteres Deckglas darauf, um die Flüssigkeit einzuschließen. Dabei achten wir auf die Vermeidung von Luftblasen in der Flüssigkeit. Wir spannen nun den Objektivträger in das Mikroskop ein. Nach dem Einstellen des Objektivs *100/1.25 oil* verwenden wir den groben und feinen Fokusantrieb des Mikroskops, um die suspendierten Latexpartikel in der Probe zu fokussieren.

Für alle folgenden Schritte verwenden wir die Funktionen, darunter eine Video- und Bildübertragung, Aufzeichnung und Maßstab-Kalibrierung, der Mikroskopsoftware am PC.

**Aufnahme einer Bildfolge.** Aus den sichtbaren Partikeln wählen wir einen eher mittig gelegenen aus, um möglichst zu vermeiden, dass dieser während der Messung aus dem Bild wandert. Abbildung (2) zeigt einen Schnappschuss aus dem Zeitraum der Messung, das beobachtete Partikel ist hier mit einem Kreis markiert. Nun nehmen wir eine Folge von 156 Bildern auf und notieren im Nachhinein noch die Raumtemperatur für die späteren Berechnungen.

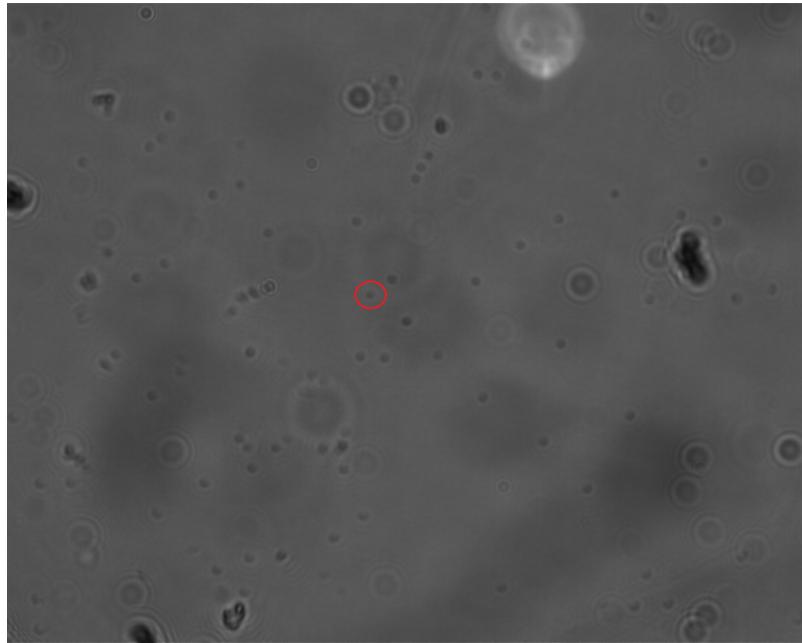


Abbildung 2: Auszug aus der aufgezeichneten Bildreihe mit dem beobachteten Partikel mittig, rot markiert.

**Eichen des abgebildeten Ausschnitts mit einem Mikrometermaßstab.** Um in der nachfolgenden Auswertung mit den korrekten Längeneinheiten rechnen zu können, nutzen wir einen Mikrometermaßstab, um den Abbildungsmaßstab zu ermitteln. Abbildung (3)

zeigt, wie der Mikrometermaßstab unter dem Mikroskop aussieht. Je zwei Teilstriche auf der Skala sind  $10\mu\text{m}$  voneinander entfernt. In der Abbildung sind außerdem die beiden Messpunkte zu sehen, welche von der Software zur Berechnung des Abbildungsmaßstabs benutzt wird.

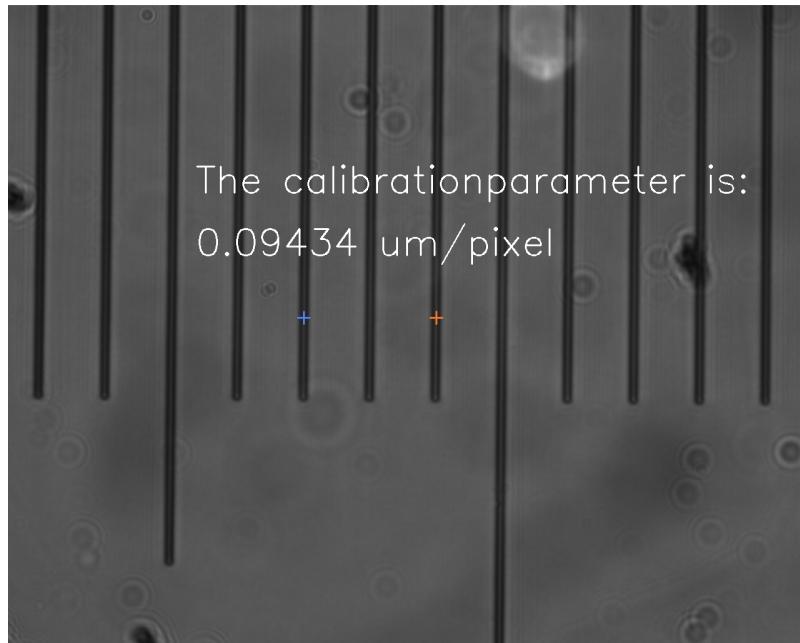


Abbildung 3: Objektmikrometer unter dem Mikroskop zur Bestimmung des Abbildungsmaßstabs.

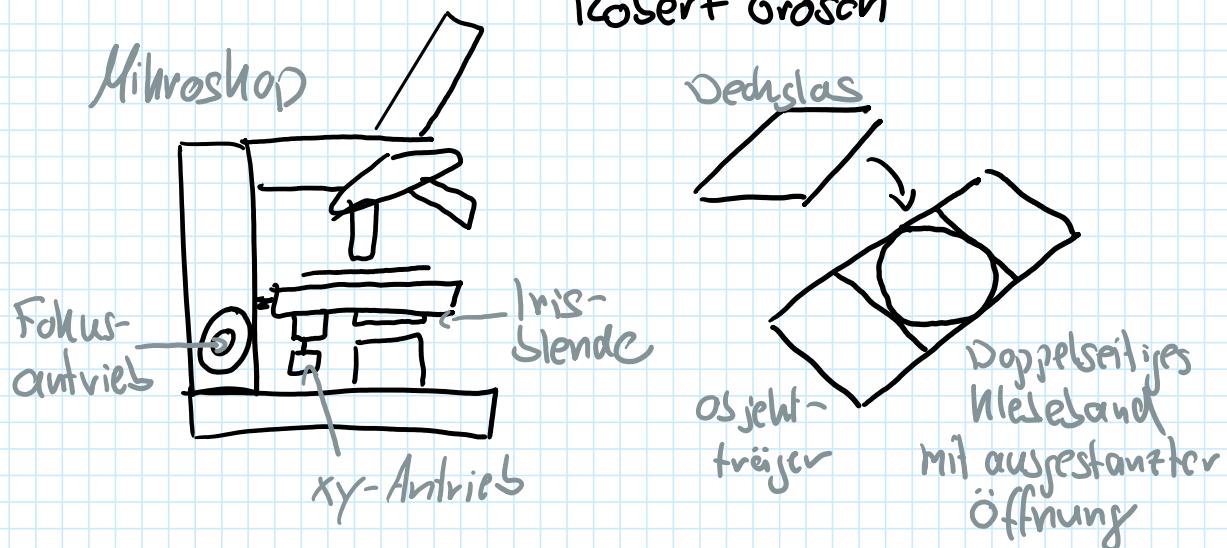
**Vermessen der Partikelpositionen für jedes Bild der auf genommenen Folge.** Abschließend müssen wir das von uns zuvor ausgewählte Partikel auf jedem der aufgenommenen Fotos markieren. Hierbei hilft uns zu Teilen wieder die Mikroskopsoftware. Da diese jedoch das ausgewählte Partikel nicht automatisch erkennt, müssen wir diesen für jedes Bild einzeln manuell suchen und auswählen. Die ausgewählten Positionen werden von der Software automatisch in einer Textdatei zur Auswertung gespeichert.

## 2 Messprotokoll

Messprotokoll 223

Marius Pfeiffer 10.12.2024

Robert Grosch



Durchmesser Teilchen:  $(755 \pm 30) \text{ nm}$  Fehlerangabe auf Flasche

Zimmertemperatur:  $(22.2 \pm 0.1)^\circ\text{C}$  Skalenfehler

Kalibrationsmaßstab:  $0.09434 \mu\text{m}/\text{px}$

*Berdhauer*

### 3 Auswertung

#### 3.1 Berechnung des mittleren Verschiebungskadrates und dessen Fehler

Wir betrachten zunächst den Weg unseres ausgewählten Partikels in der zweidimensionalen Ebene, zu sehen in Abbildung (4). Es ist zu beachten, dass das Partikel durch auch Bewegungen in der vertikalen  $z$ -Ebene ausgeführt hat. Dies haben wir hier vernachlässigt, was zu einem späteren Zeitpunkt auch bei der Fehlerdiskussion relevant ist.

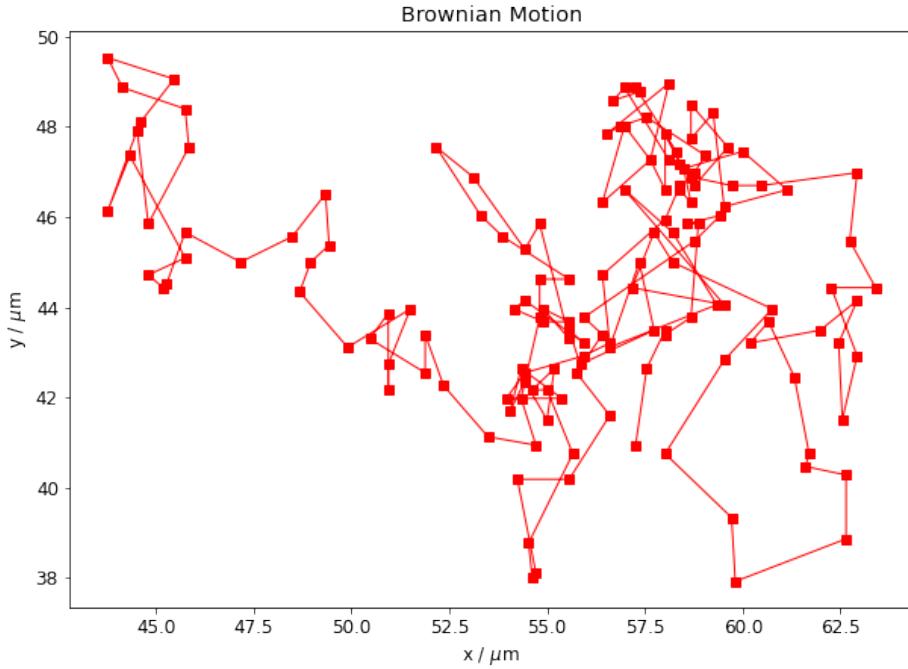


Abbildung 4: Aufgezeichneter Weg des Partikels in der zweidimensionalen Ebene.

Aus der Änderung zweier nebeneinander liegender  $x$ - und  $y$ -Koordinaten

$$\Delta x_i = x_{i+1} - x_i \quad (19)$$

$$\Delta y_i = y_{i+1} - y_i \quad (20)$$

berechnen wir nun die mittleren Verschiebungskadrates  $\langle x^2 \rangle$  und  $\langle y^2 \rangle$  in  $x$ - bzw.  $y$ -Richtung. Sowie daraus das gesamte mittlere Verschiebungskadrat  $\langle r^2 \rangle$  nach Gleichung (13). Als Mittelwert über alle Verschiebungskadrates zwischen je zwei nebeneinander liegenden aufgezeichneten Koordinaten kommen wir damit auf

$$\overline{\langle r^2 \rangle} = (2.12 \pm 0.18) \cdot 10^{-12} \text{m.} \quad (21)$$

Der mittlere zeitliche Abstand zwischen zwei Koordinatenaufzeichnungen beträgt jeweils  $\bar{t} = 1 \text{s}$ .

Nach Gleichung (18) können wir nun einen Wert für die Boltzmannkonstante berechnen. Hierzu ziehen wir noch die Daten aus dem Messprotokoll hinzu Teilchendurchmesser  $(755 \pm 30) \text{nm}$ , diesen rechnen wir noch in den Radius  $a$  um, sowie die Zimmertemperatur  $T = (22.2 + 274.15 \pm 0.1) \text{K}$ . Die Viskosität des Wassers bei Raumtemperatur bestimmen wir aus dem in der Praktikumsanleitung gegebenen Plot, zu sehen in Abbildung (5).

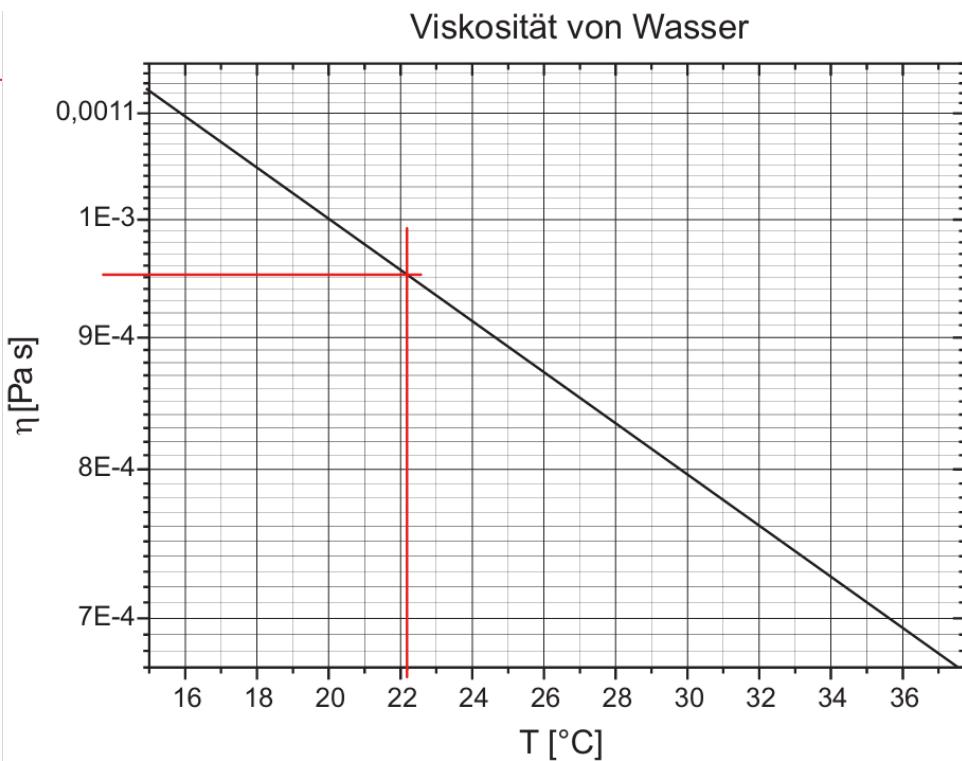


Abbildung 5: Temperaturabhängigkeit der Viskosität von Wasser.

Wie in der Abbildung zu sehen, können wir am Grafen bei  $22.2^\circ\text{C}$  eine Viskosität von in etwa  $(9.50 \pm 0.01) \cdot 10^{-4} \frac{\text{Pa}}{\text{s}}$  ablesen.

Damit erhalten wir für die Boltzmannkonstante einen Wert von

$$k_1 = (1.21 \pm 0.12) \cdot 10^{-23} \frac{\text{J}}{\text{K}}. \quad (22)$$

Aufgrund der großen Anzahl an Variablen in dieser Formel verwenden wir hier für die Fehlerberechnung die relativen Fehler:

$$\Delta k_1 = k_1 \cdot \sqrt{\left(\frac{\Delta \overline{\langle r^2 \rangle}}{\overline{\langle r^2 \rangle}}\right)^2 + \left(\frac{\Delta a}{a}\right)^2 + \left(\frac{\Delta \eta_{H_2O}}{\eta_{H_2O}}\right)^2 + \left(\frac{\Delta T}{T}\right)^2 + \left(\frac{\Delta t}{t}\right)^2}. \quad (23)$$

Um den Diffusionskoeffizienten  $D$  zu berechnen stellen wir die Gleichung  $\sqrt{\langle r^2 \rangle} = \sqrt{4Dt}$  um, zu

$$D = \frac{1}{4} \frac{\overline{\langle r^2 \rangle}}{\bar{t}} \quad (24)$$

und erhalten

$$D = (5.3 \pm 0.5) \cdot 10^{-13} \frac{\text{m}^2}{\text{s}} \quad (25)$$

mit Fehler nach der klassischen Gauß'schen Fehlerfortpflanzung.

### 3.2 Kontrollverteilung

Wir möchten nun die mathematische Aussage prüfen, dass die Wahrscheinlichkeit, ein Partikel nach der Zeit  $t$  im Intervall  $[x, x + \Delta x]$  befindet tatsächlich einer Gaussverteilung folgt. Hierzu fügen wir alle berechneten mittleren Verschiebungsquadrate in  $x$ - und  $y$ -Richtung in eine gemeinsame Liste und stellen die Werte in einem Histogramm, siehe Abbildung (6), dar.

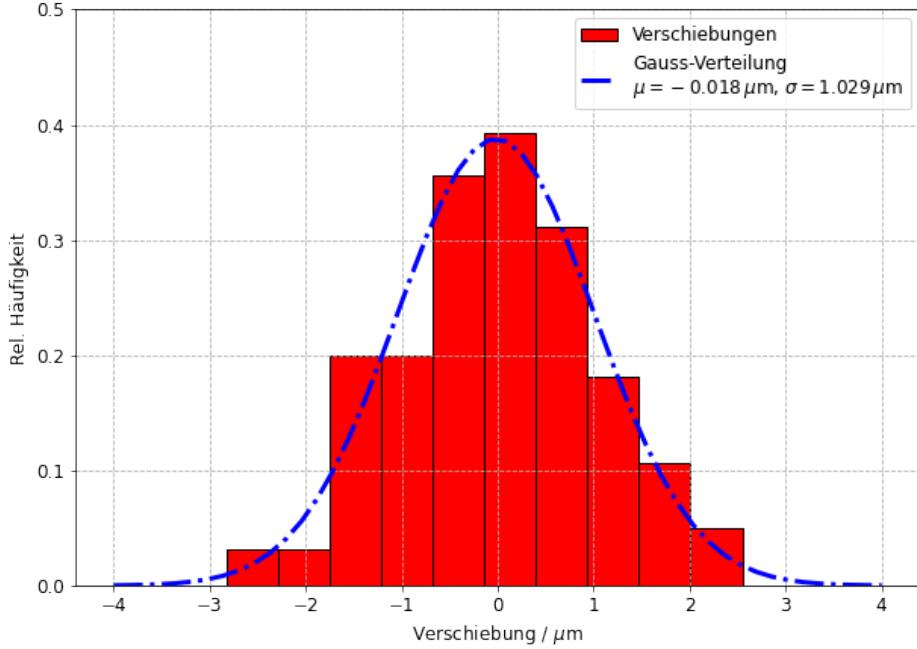


Abbildung 6: Histogramm der mittleren Verschiebungsquadrate in in  $x$ - und  $y$ -Richtung.

$\mu$  und  $\sigma^2$  der im Plot in blau dargestellten Gaußkurve berechnen wir aus den Daten anhand der standardmäßigen Formeln für Mittelwert und Standardabweichung

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i, \quad (26)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N |x_i - \bar{x}|^2}. \quad (27)$$

Es ist gut zu erkennen, dass die Kurve die Form der Daten bereits sehr gut beschreibt.

### 3.3 Kumulative Verteilung der Verschiebungsquadrate

Für einen weiteren Ansatz zur Berechnung der Boltzmannkonstante betrachten wir nun zunächst die kumulative Verschiebung als Funktion der Zeit, also deren kumulative Verteilung, dargestellt in Abbildung (7).

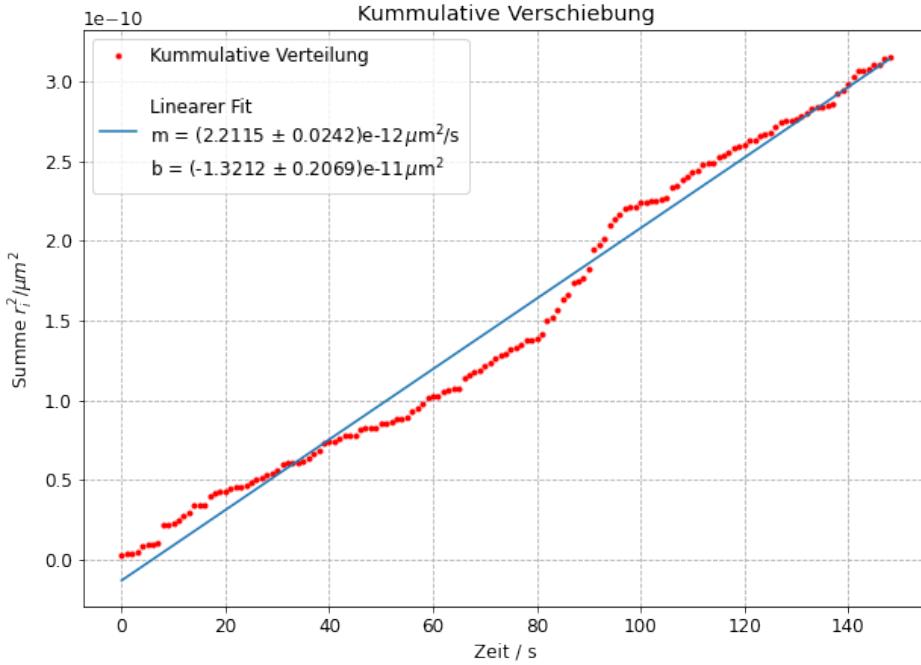


Abbildung 7: Kumulative Verschiebung eines Partikels als Funktion der Zeit.

An die Verteilung fitten wir eine standardmäßige lineare Funktion der Form  $mx + b$ , im Plot dargestellt in Blau. Die optimierten Werte für  $m$  und  $b$ , sowie deren Fehler, sind in der Legende des Plots angegeben. Mit der Steigung  $m$  der Geraden können wir nun das Verhältnis  $\frac{\langle r^2 \rangle}{t}$  in der ursprünglichen Formel für die Boltzmannkonstante ersetzen. Es gilt also

$$k_2 = \frac{6\pi\eta am}{4T} \quad (28)$$

und

$$\Delta k_2 = k_2 \cdot \sqrt{\left(\frac{\Delta m}{m}\right)^2 + \left(\frac{\Delta a}{a}\right)^2 + \left(\frac{\Delta \eta_{H_2O}}{\eta_{H_2O}}\right)^2 + \left(\frac{\Delta T}{T}\right)^2}. \quad (29)$$

Wir berechnen damit einen Wert von

$$k_2 = (1.26 \pm 0.06) \cdot 10^{-23} \frac{J}{K}. \quad (30)$$

Die Formel für den Diffusionskoeffizienten reduziert sich unter Verwendung der Steigung  $m$  auf

$$D = \frac{1}{4}m = (5.53 \pm 0.07) \cdot 10^{-13} \frac{m^2}{s}. \quad (31)$$

## 4 Zusammenfassung und Diskussion

Die Brown'schen Bewegung beschreibt die zufälligen Zickzackbewegungen kleiner Teilchen in einer Flüssigkeit oder einem Gas, die bei einer Temperatur über 0K durch Stöße mit den umgebenden Molekülen verursacht werden. Sie wurde 1827 von Robert Brown beobachtet und später durch die kinetische Gastheorie und Albert Einsteins mathematische Beschreibung erklärt. Bei Betrachtung des mittleren Verschiebungsquadrats, welches ein Partikel durch seine Brown'sche Bewegung zurücklegt ergibt sich ein direkter mathematischer Zusammenhang zwischen diesem und der Boltzmannkonstante, nach der Formel

$$k = \frac{6\pi\eta a}{4Tt} \langle r^2 \rangle.$$

In dieser Formel lässt sich der Faktor

$$\frac{kT}{6\pi\eta a}$$

mit dem Diffusionskoeffizienten  $D$  identifizieren, welcher die Beweglichkeit eines Teilchens angibt.

Ziel von Versuch 223 war es, die Boltzmannkonstante anhand der Brown'schen Bewegung von in Flüssigkeit suspendierten Latexpartikeln zu bestimmen. Hierzu haben wir die Bewegung in der horizontalen Ebene eines einzelnen Latexpartikels mit einem Mikroskop über einen Zeitraum von 150s beobachtet und aufgezeichnet. Aus den zurückgelegten Wegstrecken pro Sekunde haben wir das mittlere Verschiebungskquadrat des Partikels zu

$$\overline{\langle r^2 \rangle} = (2.12 \pm 0.18) \cdot 10^{-12} \text{m}^2$$

berechnet. Mit der oben genannten Formel konnten wir daraus für die Boltzmannkonstante einen Wert von

$$k_1 = (1.21 \pm 0.12) \cdot 10^{-23} \frac{\text{J}}{\text{K}}$$

berechnen. Außerdem haben wir für den Diffusionskoeffizienten eines Latexpartikels einen Wert von

$$D = (5.3 \pm 0.5) \cdot 10^{-13} \frac{\text{m}^2}{\text{s}}$$

bestimmt.

Der von uns hier berechnete Wert für die Boltzmannkonstante weicht um etwa  $1.6\sigma$  vom Literaturwert<sup>1</sup> ab. Die Abweichung ist damit zwar als nicht signifikant einzustufen, dennoch bietet es sich an, auf einige mögliche Fehlerquellen einzugehen. Zum einen ist es wichtig zu bemerken, dass wir, wie bereits erwähnt, die Bewegung des Partikels nur in zwei Dimensionen, anstatt der möglichen drei betrachtet haben. Dies führt zu einer Unterschätzung des mittleren Verschiebungskquadrates und somit auch der Boltzmannkonstante. Weiter spielen Umgebungseffekte eine Rolle. So können eine schwankende Temperatur um die Probe, beispielsweise verursacht durch das Mikroskop, oder auch eine imperfekte Abdichtung der Suspension zu Bewegungen der Flüssigkeit und der Partikel, unabhängig von der eigentlichen Brown'schen Bewegung führen.

Die mathematische Theorie hinter der Brown'schen Bewegung sagt voraus, dass die Wahrscheinlichkeit, ein Partikel nach der Zeit  $t$  im Intervall  $[x, x + \Delta x]$  zu finden, Gaußverteilt

---

<sup>1</sup> $k = 1.380\,649 \cdot 10^{-23} \frac{\text{J}}{\text{K}}$ , 2022 CODATA recommended values

ist. Um dies zu bestätigen haben wir alle mittleren Verschiebungsquadrate gemeinsam in einem Histogramm geplottet, siehe Abbildung (6). Aus den Daten haben wir außerdem den Mittelwert  $\mu = -0.018\mu\text{m}$ , sowie die Standardabweichung  $\sigma = 1.029\mu\text{m}$  bestimmt, um damit eine Gaußkurve mit in das Histogramm zu plotten. Trotz der vorliegenden vergleichsweise geringen Menge an Daten konnte durch diese ide Form des Histogramms bereits sehr gut beschrieben werden.

Mit der Betrachtung der kumulativen Verteilung der Verschiebungsquadrate als Funktion der Zeit haben wir im letzten Teil der Auswertung die Boltzmannkonstante noch auf einem weiteren Weg berechnet. Anhand der Anpassung einer linearen Funktion haben wir hierbei die Steigung der kumulativen Verteilung ermittelt, welche in der Berechnung der Boltzmannkonstante nun den Faktor  $\langle r^2 \rangle / t$  ersetzt. Der berechnete Wert beläuft sich auf

$$k_2 = (1.26 \pm 0.06) \cdot 10^{-23},$$

was in etwa  $0.44\sigma$  vom zuvor berechneten Wert abweicht. Für den Diffusionskoeffizienten haben wir in diesem Verfahren einen Wert von

$$D = (5.53 \pm 0.07) \cdot 10^{-13}$$

ermittelt, welcher um ca.  $0.54\sigma$  vorherigen Wert abweicht. Die Abweichungen sind nicht von den zuvor Berechneten werden sind nicht signifikant und sollten vermutlich hauptsächlich auf den Fit an die kumulative Verteilung zurückzuführen sein. Betrachten wir Abbildung (7) der Auswertung, so ist zu sehen, dass die Datenpunkte sehr stark um die optimierte Gerade schwanken. Diese Ungenauigkeit zeigt sich in den berechneten Werten wieder.

# Python Code, Hauptprogramm

auswertung223

February 13, 2025

```
[1]: import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import numpy as np
from scipy.stats import norm
from scipy.optimize import curve_fit

plt.rcParams.update({'font.size': 12})

%run ../lib.ipynb
```

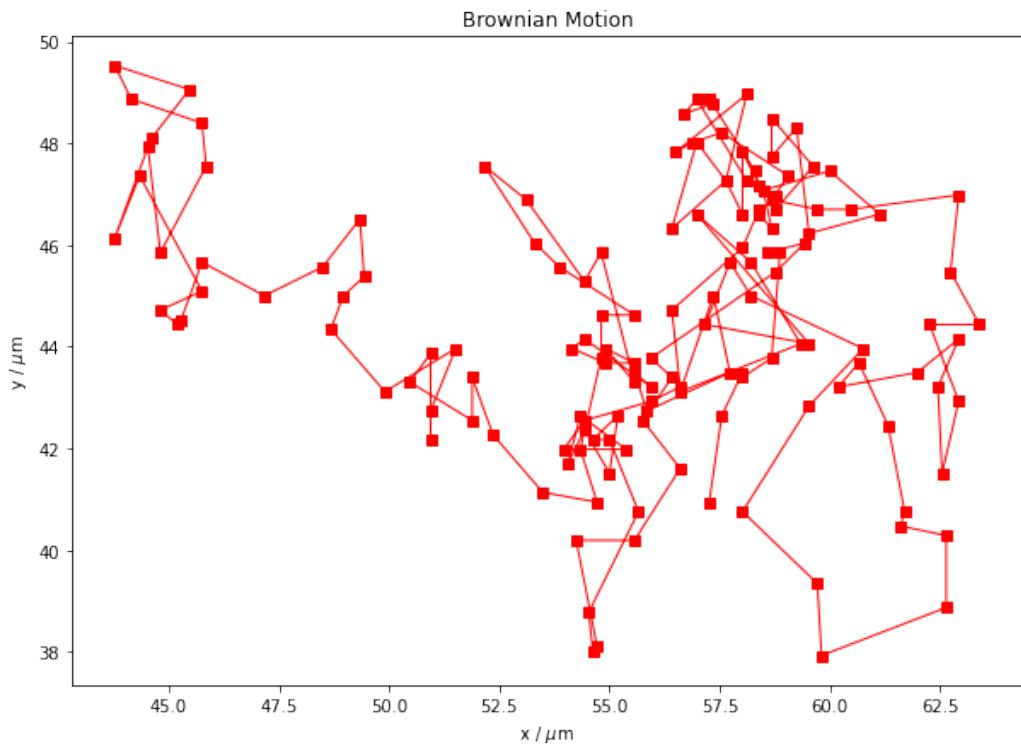
```
[2]: # import position data
t,x,y=np.loadtxt('position_data.txt', delimiter=",",
                  dtype="float", unpack=True)

class daten:
    durchmesser_teilchen = ValErr(755, 30) * 0.5 * 10**(-9) # m
    zimmertemp = ValErr(22.2, 0.1) + 274.15 # K
    kalibmassstab = 0.09434 # mum / px

k_boltzmann_lit = ValErr(1.380649 * 10**(-23), 0) # J/K
```

```
[3]: # plot brownian motion in 2d

plt.figure(figsize=(10,7))
plt.plot(x, y, marker='s', color='red', linewidth=1)
plt.xlabel('x / $\mu$m')
plt.ylabel('y / $\mu$m')
plt.title('Brownian Motion')
plt.savefig('brown1.png', format='png')
```



```
[4]: # calculate squared mean and error

dt = np.array([])
dx = np.array([])
dy = np.array([])
for i in range(0, len(t) - 1):
    dt = np.append(dt, t[i+1] - t[i])
    dx = np.append(dx, x[i+1] - x[i])
    dy = np.append(dy, y[i+1] - y[i])

r_squared = (dx**2 + dy**2) * 10**(-12)

r_squared_mean_val = np.mean(r_squared)
r_squared_mean_std = np.std(r_squared) / np.sqrt(len(r_squared))

r_squared_mean = ValErr(r_squared_mean_val, r_squared_mean_std)

print(r_squared_mean.strfmtf2(4, -12))

dt_mean_val = np.mean(dt)
dt_mean_std = np.std(dt) / np.sqrt(len(dt))
```

```

dt_mean = ValErr(dt_mean_val, dt_mean_std)

print('dt:', dt_mean_val, '+-', dt_mean_std)

eta_h2o = ValErr(9.50, 0.01) * 10**(-4) # Pa*s = kg/(m*s)

k_boltz_exp_val = r_squared_mean.val * (6 * np.pi * eta_h2o.val * daten.
                                         durchmesser_teilchen.val) / (4 * daten.zimmertemp.val * dt_mean.val)
k_boltz_exp_err = k_boltz_exp_val * np.sqrt(np.sum([r_squared_mean.relerr()**2, □
                                         daten.durchmesser_teilchen.relerr()**2, eta_h2o.relerr()**2, daten.
                                         zimmertemp.relerr()**2, dt_mean.relerr()**2])))

k_boltz_exp = ValErr(k_boltz_exp_val, k_boltz_exp_err)

# sqrt(<r^2>) = sqrt(4Dt) <=> D = <r^2> / 4t

D = (1/4) * (r_squared_mean / dt_mean)

print_all(
    k_boltz_exp.strfmtf2(4, -23, 'k_B'),
    'abw von literatur: ' + str(np.round(k_boltzmann_lit.
                                         sigmadiff(k_boltzmann_lit), 4)) + ' ',
    '-----',
    D.strfmtf2(4, -13, 'D'))

```

```

(2.1176 ± 0.1751)e-12
dt: 1.0 +- 0.0
k_B = (1.2076 ± 0.1108)e-23
abw von literatur: 1.5622
-----
D = (5.2940 ± 0.4377)e-13

```

[5]: # Kontrollverteilung

```

all_data=np.append(dx, dy)

mu = np.mean(all_data)
sigma = np.std(all_data)
gauss = norm.pdf(np.linspace(-4,4), mu, sigma)

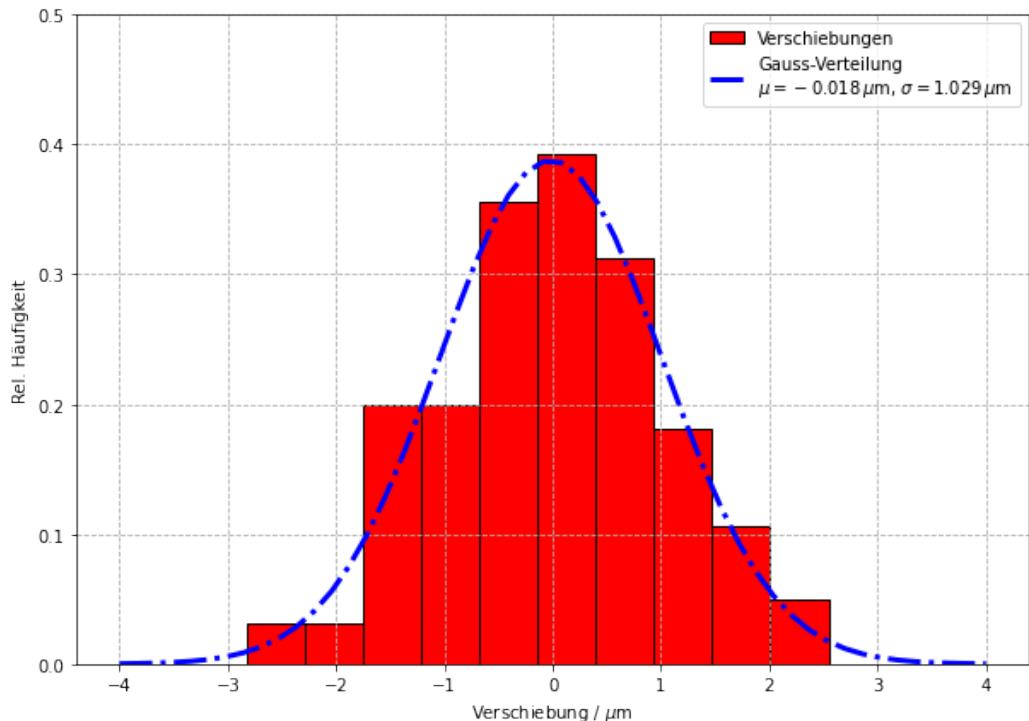
plt.figure(figsize=(10,7))
plt.hist(all_data, histtype='bar', edgecolor='black', color='red', □
        density=True, label='Verschiebungen')
plt.plot(np.linspace(-4,4), gaus, 'b-.', linewidth=3, □
        label=f'Gauss-Verteilung\n$\mu={np.round(mu, 3)}$, $\mu\mathrm{m}$', □
        '$\sigma={np.round(sigma, 3)}$, $\sigma\mathrm{m}$')
plt.ylabel(r'Rel. Häufigkeit')

```

```

plt.xlabel('Verschiebung / $\mu\text{m}$')
plt.yticks(np.arange(0, 0.6, 0.1))
plt.xticks(np.arange(-4, 5, 1))
plt.grid(linestyle='--')
plt.legend()
plt.savefig('brown2.png', format='png')

```



```

[6]: # Kumulative Verteilung der Verschiebungsquadrate

r_kumm = np.cumsum(r_squared)

def fit_func_linear(x, m, b):
    return m * x + b

popt, pcov = curve_fit(fit_func_linear, t[:-1], r_kumm)
m_fitted = ValErr.fromFit(popt, pcov, 0)
b_fitted = ValErr.fromFit(popt, pcov, 1)

plt.figure(figsize=(10,7))
plt.plot(t[:-1], r_kumm, marker='.', color='red', linewidth=0, u
         ↪label='Kumulative Verteilung')

```

```

plt.plot(t[:-1], fit_func_linear(t[:-1], *popt), label=f'\nLinearer\u20ac
˓→Fit\n{m_fitted.strfmtf2(4,-12, "m")}\$, \mu\mathrm{{m}}^{\{2\}} /_
˓→\mathrm{{s}}\$ \n{b_fitted.strfmtf2(4,-11, "b")}\$, \mu\mathrm{{m}}^{\{2\}}\$')
plt.xlabel('Zeit / s')
plt.ylabel('Summe $r_i^2 / \mu m^2$')
plt.title('Kumulative Verschiebung')
plt.grid(linestyle='--')
plt.legend()
plt.savefig('brown3.png', format='png')

k_boltz_exp_cumm_val = m_fitted.val * (6 * np.pi * eta_h2o.val * daten.
˓→durchmesser_teilchen.val) / (4 * daten.zimmertemp.val)
k_boltz_exp_cumm_err = k_boltz_exp_cumm_val * np.sqrt(np.sum([m_fitted.
˓→relerr()**2, daten.durchmesser_teilchen.relerr()**2, eta_h2o.relerr()**2, daten.zimmertemp.relerr()**2]))

k_boltz_exp_cumm = ValErr(k_boltz_exp_cumm_val, k_boltz_exp_cumm_err)

D_cumm = (1/4) * m_fitted

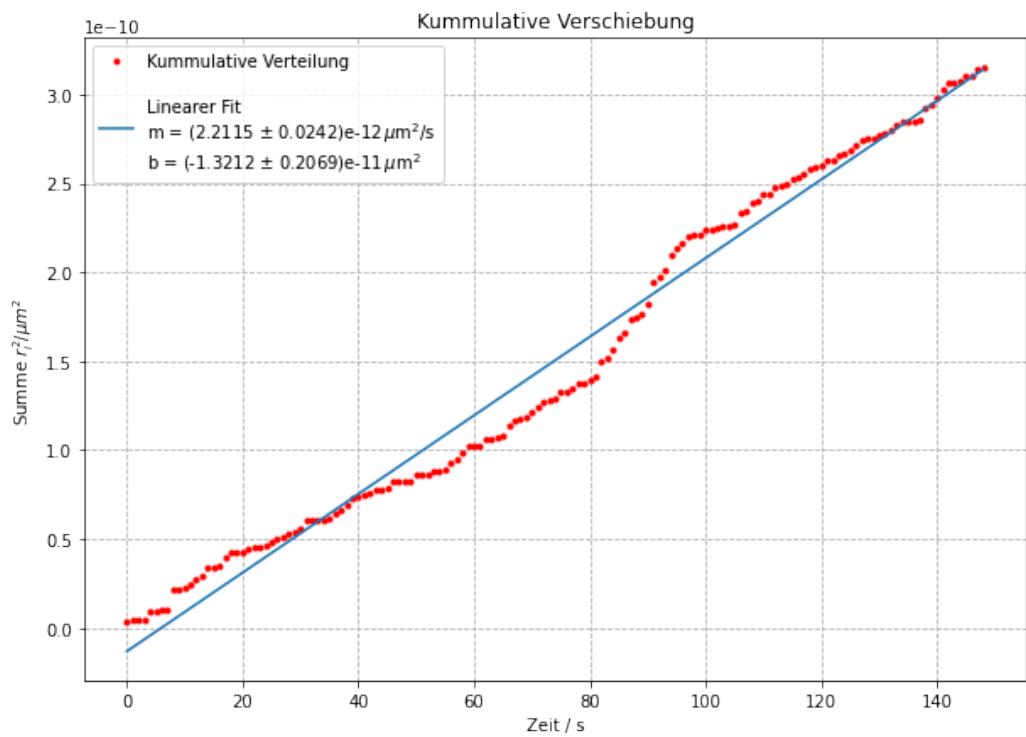
print_all(
    k_boltz_exp_cumm.strfmtf(4, -23, 'k_B_cumm'),
    f'abw von vorherigem wert: {k_boltz_exp_cumm.sigmadiff(k_boltz_exp)}',
    f'abw von literaturwert: {k_boltz_exp_cumm.sigmadiff(k_boltzmann_lit)}',
    '-----',
    D_cumm.strfmtf(4, -13, 'D_cumm'),
    f'abw von vorherigem wert: {D_cumm.sigmadiff(D)}')

```

```

k_B_cumm = 1.2612e-23 ± 0.0520e-23
abw von vorherigem wert: 0.43773622790818695
abw von literaturwert: 2.2981161756840587
-----
D_cumm = 5.5289e-13 ± 0.0604e-13
abw von vorherigem wert: 0.5315058684910744

```



[ ]:

[ ]:

# Python Code, Bibliothek

lib

February 13, 2025

```
[3]: def floatfmt(v, prec, exp):
       return f"{v/10**exp}:0={prec}f}{f'e{exp}' if exp != 0 else ''}"
```

```
[1]: import math
      import numpy as np

      class ValErr:
          val: float = 0
          err: float = 0
          err_set = False

          def __init__(self, val, err=0):
              self.val = val
              if err != 0:
                  self.err_set = True
                  self.err = err

          def getTuple(self):
              return (self.val, self.err)

          def setErr(self, err_value):
              self.err_set = True
              self.err = err_value

          @classmethod
          def fromMeasurements(self, measurements):
              return ValErr(np.mean(measurements), (1 / math.sqrt(len(measurements))) ↴
                           * np.std(measurements, ddof=1))

          @classmethod
          def fromTuple(self, tup):
              return ValErr(tup[0], tup[1])

          @classmethod
          def fromFit(self, popt, pcov, i):
              return ValErr(popt[i], np.sqrt(pcov[i][i]))
```

```

@classmethod
def fromFitAll(self, popt, pcov):
    for i in range(0, len(popt)):
        yield ValErr(popt[i], np.sqrt(pcov[i][i]))

@classmethod
def fromValPerc(self, v, perc):
    return ValErr(v, v * perc/100)

def strfmt(self, prec=2):
    if self.err != 0:
        return fr"{{self.val:.{prec}e} ± {self.err:.{prec}e}}"
    else:
        return f"{{self.val:.{prec}e}}"

def strfmtf(self, prec, exp, name = ""):
    prefix = ""
    if name != "":
        prefix = f"{{name}} = "

    if self.err != 0:
        return prefix + fr"{{floatfmt(self.val, prec, exp)} ± {{floatfmt(self.
→err, prec, exp)}}}"
    else:
        return prefix + f"{{floatfmt(self.val, prec, exp)}}"

def strfmtf2(self, prec, exp, name = ""):
    prefix = ""
    if name != "":
        prefix = f"{{name}} = "

    if self.err != 0:
        return prefix + fr"{{f'(' if exp != 0 else ''){{self.val/10**exp}:.
→0=1.{prec}f} ± {{self.err/10**exp}:0=1.{prec}f}{f')e{exp}}' if exp != 0 else_
→''))"
    else:
        return prefix + f"{{floatfmt(self.val, prec, exp)}}"

def strltx(self, prec=2):
    if self.err != 0:
        return fr"{{self.val:.{prec}e} \pm {self.err:.{prec}e}}"
    else:
        return f"{{self.val}}"

def relerr(self):
    return self.err / self.val

```

```

def sigmadiff(self, other):
    return np.abs(self.val - other.val) / np.sqrt(self.err**2 + other.
→err**2)

def __repr__(self):
    return f"ValErr({self.val}, {self.err})"

def __radd__(self, other):
    return self.__add__(other)

def __add__(self, other):
    if isinstance(other, self.__class__):
        return ValErr(self.val + other.val, math.sqrt(self.err**2 + other.
→err**2))
    elif isinstance(other, float) or isinstance(other, int):
        return ValErr(self.val + other, self.err)
    else:
        raise TypeError(f"unsupported operand type(s) for +: '{self.
→__class__}' and '{type(other)}'")

def __rsub__(self, other):
    return self.__sub__(other)

def __sub__(self, other):
    if isinstance(other, self.__class__):
        return ValErr(self.val - other.val, math.sqrt(self.err**2 + other.
→err**2))
    elif isinstance(other, float) or isinstance(other, int):
        return ValErr(self.val - other, self.err)
    else:
        raise TypeError(f"unsupported operand type(s) for +: '{self.
→__class__}' and '{type(other)}'")

def __rmul__(self, other):
    return self.__mul__(other)

def __mul__(self, other):
    if isinstance(other, self.__class__):
        return ValErr(self.val * other.val, math.sqrt((other.val * self.
→err)**2 + (self.val * other.err)**2))
    elif isinstance(other, float) or isinstance(other, int):
        return ValErr(self.val * other, self.err * other)
    else:
        raise TypeError(f"unsupported operand type(s) for +: '{self.
→__class__}' and '{type(other)}'")

```

```

def __rtruediv__(self, other):
    if isinstance(other, self.__class__):
        return ValErr(other.val / self.val, math.sqrt((other.err / self.
val)**2 + (other.val * self.err / self.val**2)**2))
    elif isinstance(other, float) or isinstance(other, int):
        return ValErr(other / self.val, np.abs(other / self.val**2) * self.
err)
    else:
        raise TypeError(f"unsupported operand type(s) for +: '{self.
__class__}' and '{type(other)}'")

def __truediv__(self, other):
    if isinstance(other, self.__class__):
        return ValErr(self.val / other.val, math.sqrt((self.err / other.
val)**2 + (self.val * other.err / other.val**2)**2))
    elif isinstance(other, float) or isinstance(other, int):
        return ValErr(self.val / other, self.err / other)
    else:
        raise TypeError(f"unsupported operand type(s) for +: '{self.
__class__}' and '{type(other)}'")

```

```
[5]: def spacearound(dat, add):
       return np.linspace(dat[0] - add, dat[len(dat)-1] + add)
```

```
[6]: def div_with_err(a, a_err, b, b_err):
       err = (1 / b) * np.sqrt(a_err**2 + (a * b_err / b)**2)
       return (a / b, err)
```

```
[7]: def print_all(*args):
       for e in args:
           print(e)
```

```
[ ]:
```