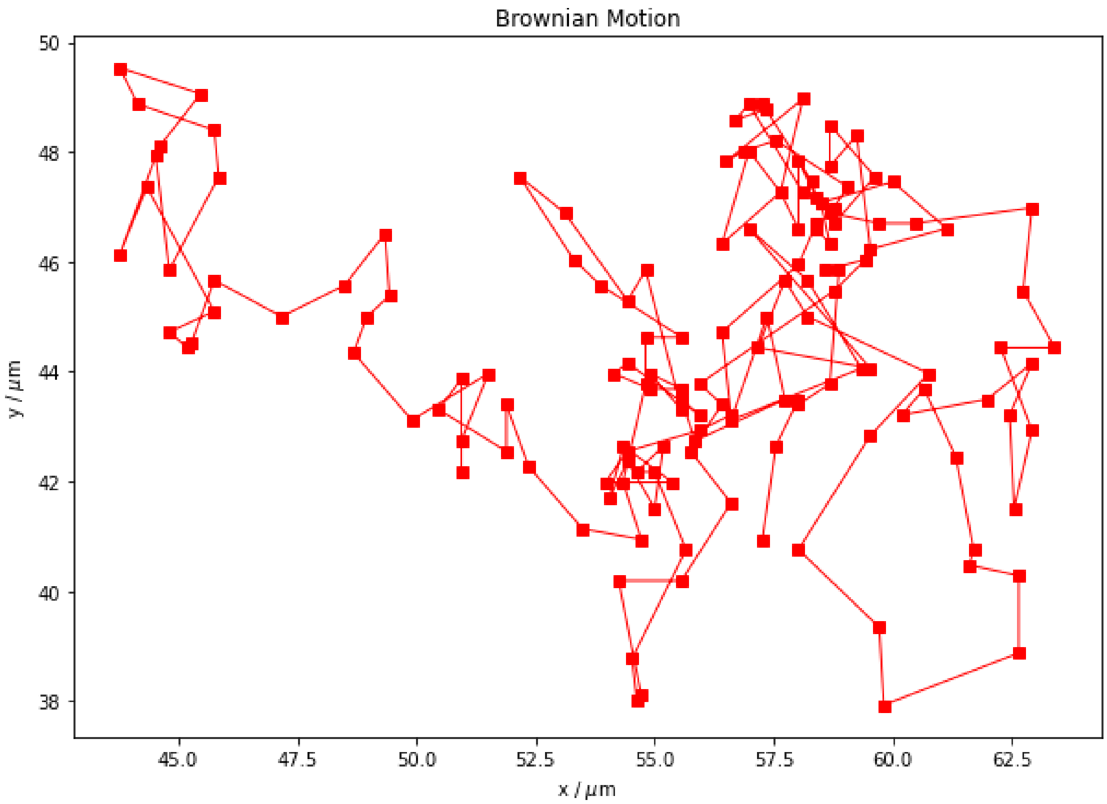
auswertung223

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')
```



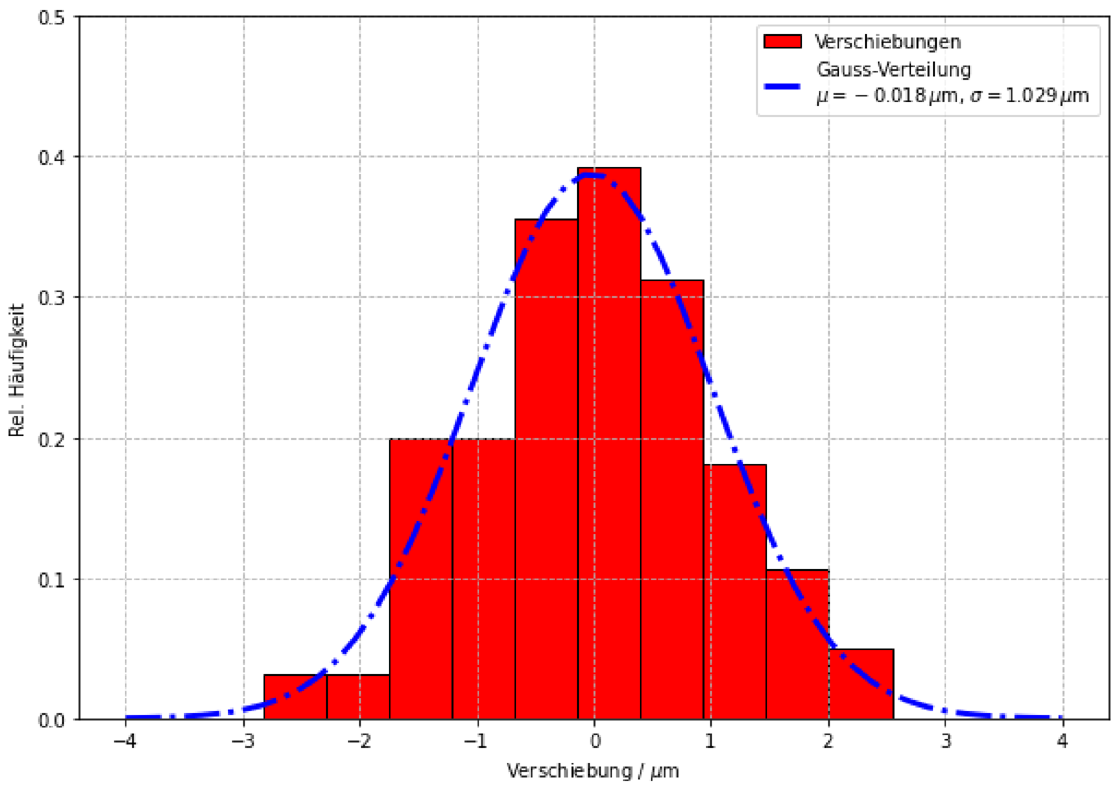
[4]:
```
# calculate squared mean and error

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_boltz_exp.
 ↪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), gauss, 'b-.', linewidth=3,␣
 ↪label=f'Gauss-Verteilung\n$\mu={np.round(mu, 3)}\,\mu\mathrm{{m}}$,␣
 ↪$\sigma={np.round(sigma, 3)}\,\mu\mathrm{{m}}$')
plt.ylabel(r'Rel. Häufigkeit')
```

```
plt.xlabel('Verschiebung / $\mu$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')
```

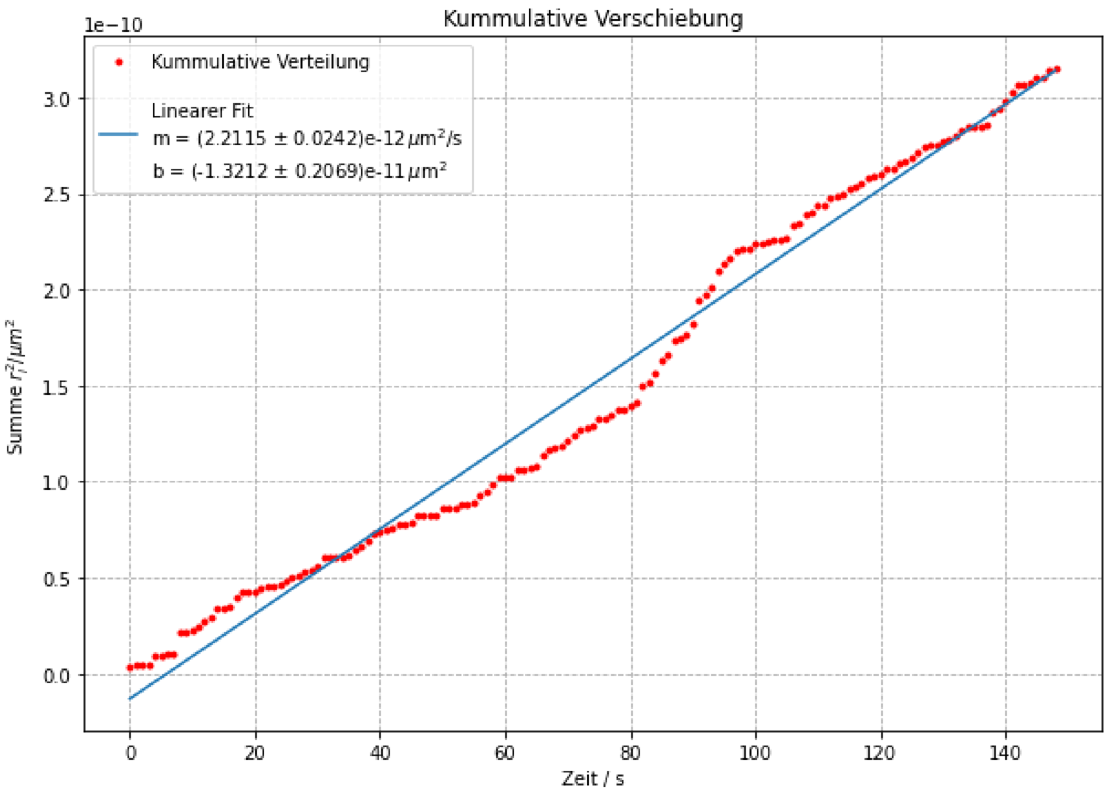


[6]:

```
#  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,␣
 ↪label='Kummulative Verteilung')
```

```
plt.plot(t[:-1], fit_func_linear(t[:-1], *popt), label=f'\nLinearer␣
↪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('Kummulative 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
```


Kummulative Verschiebung
1e−10
Kummulative Verteilung
Linearer Fit
m = (2.2115 ± 0.0242)e-12 μm²/s
b = (-1.3212 ± 0.2069)e-11 μm²
Summe $r_i^2/\mu m^2$
Zeit / s
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100
120
140

```
[ ]:
```

```
[ ]:
```