auswertung

December 7, 2024

[89]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
import scipy.integrate as integrate

%run ../lib.ipynb

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

[90]:
```
f_a3 = np.loadtxt('aufgabe3_frequenzgang.txt', skiprows=1, usecols=(0,1),␣
↪unpack=True)
```

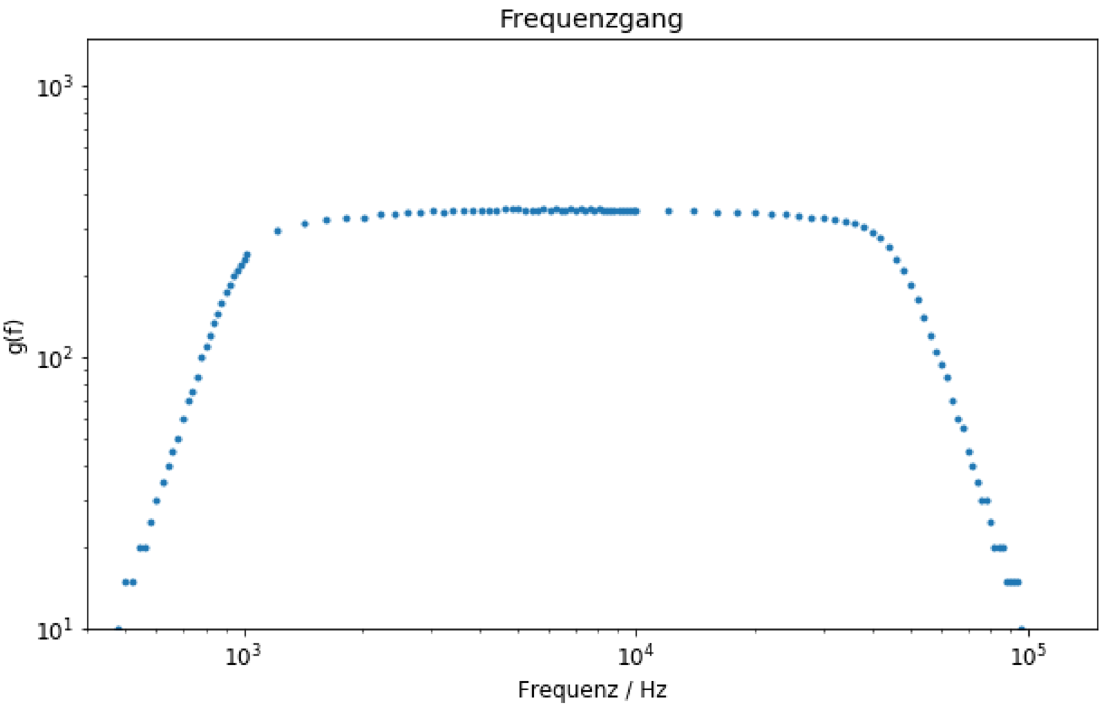
0.0.1 Konstanten

[91]:
```
D = 1e-3
U_ein = 0.2
lb = 19
ub = 13
U_aus = f_a3[1][lb:-ub]
g = U_aus / (U_ein * D)

np.max(g)
```

[91]:
```
354.99999999999994
```

0.0.2 Plot des Frequenzgangs

[92]:
```
plt.figure(figsize=(10,6))
plt.loglog(f_a3[0][lb:-ub], g, linestyle='None', marker='.')
plt.axis([4E2, 1.5e5, 10, 1.5E3])
plt.xlabel('Frequenz / Hz')
plt.ylabel('g(f)')
plt.title('Frequenzgang')
plt.savefig('freq_data.png', dpi=100)
```



0.0.3 Definition Fitfunktion

[93]:
```
def fit_func(f, V, W1, W2, n1, n2):
    hp_filter = np.sqrt(1 + 1 / (f / W1)**(2*n1))
    tp_filter = np.sqrt(1 + (f/W2)**(2*n2))
    return V / (hp_filter * tp_filter)
```

0.0.4 Startparameter Fitting

[94]:
```
verst = 1000
untere_grenzfreq = 1000
obere_grenzfreq = 50000
filterord_n1 = filterord_n2 = 5
p0 = [verst, untere_grenzfreq, obere_grenzfreq, filterord_n1, filterord_n2]
```

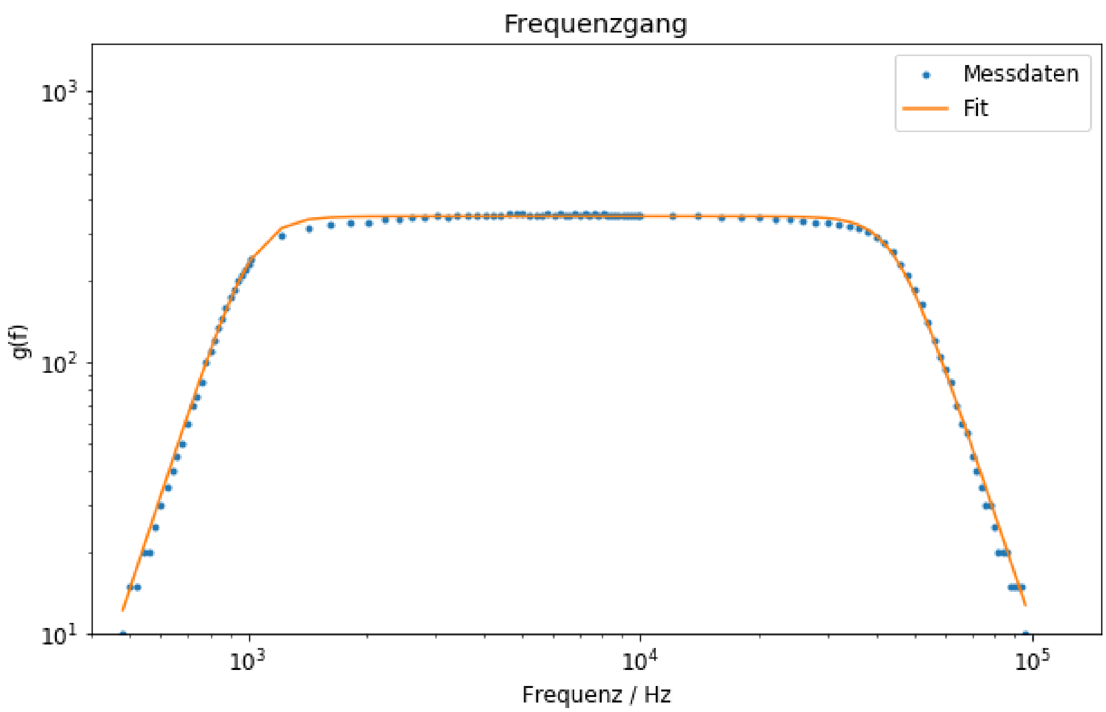
0.0.5 Fitting

[95]:
```
popt_freq, pcov_freq = curve_fit(fit_func, f_a3[0][lb:-ub], g, p0)

V_ve = ValErr.fromFitAll(popt_freq, pcov_freq)

list(V_ve)
```

```
[95]: [ValErr(346.5518430884586, 0.8216377544429044),
 ValErr(1027.9483181651547, 4.764504463568716),
 ValErr(44598.41662432931, 203.70317002946283),
 ValErr(4.387039307736234, 0.09704251588395478),
 ValErr(4.301916086387025, 0.08225781876207226)]
```

0.0.6 Funktion + Fit

```
[96]: plt.figure(figsize=(10,6))
plt.loglog(f_a3[0][lb:-ub], g, linestyle='None', marker='.', label='Messdaten')
plt.loglog(f_a3[0][lb:-ub], fit_func(f_a3[0][lb:-ub], *popt_freq), label='Fit')
plt.axis([4E2, 1.5e5, 10, 1.5E3])
plt.xlabel('Frequenz / Hz')
plt.ylabel('g(f)')
plt.title('Frequenzgang')
plt.legend(loc='best')
plt.savefig('freq_data_fit.png', dpi=100)
```



0.0.7 Numerische Integration

```
[97]: def fit_func_square(f, V, W1, W2, n1, n2):
    return fit_func(f, V, W1, W2, n1, n2)**2
```

[98]:
```
B_int_res = integrate.quad(fit_func_square, f_a3[0][lb], f_a3[0][-ub],␣
 ↪args=tuple(popt_freq))

B = ValErr(B_int_res[0], B_int_res[0] * 0.02)
print(fr"Wert des Integrals: B = {B.strfmtf(3, 9)}")
```

```
Wert des Integrals: B = 5.349e9 ± 0.107e9
```

0.1 Bestimmung der Boltzmannkonstante

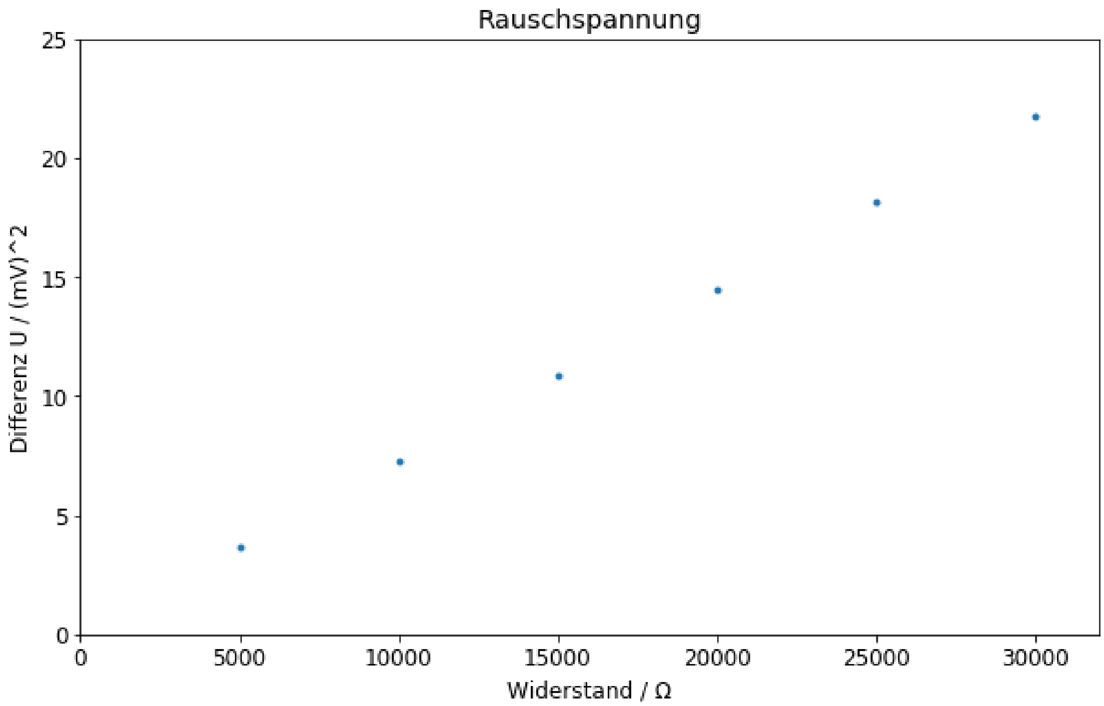
[99]:
```
f_a2 = np.loadtxt('aufgabe2_rauschspannung.txt', skiprows=1, usecols=(0,1,2),␣
 ↪unpack=True)
R = f_a2[0][:-1]
U = f_a2[1][:-1]
U_err = f_a2[2][:-1]

U_0 = ValErr(f_a2[1][-1], f_a2[2][-1])
```

0.1.1 Differez und Fehler der Differenz

[100]:
```
def U_diff(u, u_0):
    return u**2 - u_0**2

def U_diff_err(u, u_err, u_0, u_0_err):
    return np.sqrt((u_err * 2 * u) ** 2 + (u_0_err * 2 * u_0) ** 2)

fehler_D=np.sqrt((U_err * 2 * U) ** 2 + (U_0.err * 2 * U_0.val) ** 2)

Diff = U_diff(U, U_0.val)
Diff_Err = U_diff_err(U, U_err, U_0.val, U_0.err)
```

0.1.2 Plot der Daten

[101]:
```
plt.figure(figsize=(10,6))
plt.errorbar(R, Diff, yerr=Diff_Err, fmt='.')
plt.axis([0,3.2e4,0,25])
plt.xlabel('Widerstand / Ω')
plt.ylabel('Differenz U / (mV)^2')
plt.title('Rauschspannung')
plt.savefig('ur_data.png', dpi=100)
```



0.1.3 Linearer Fit an Daten

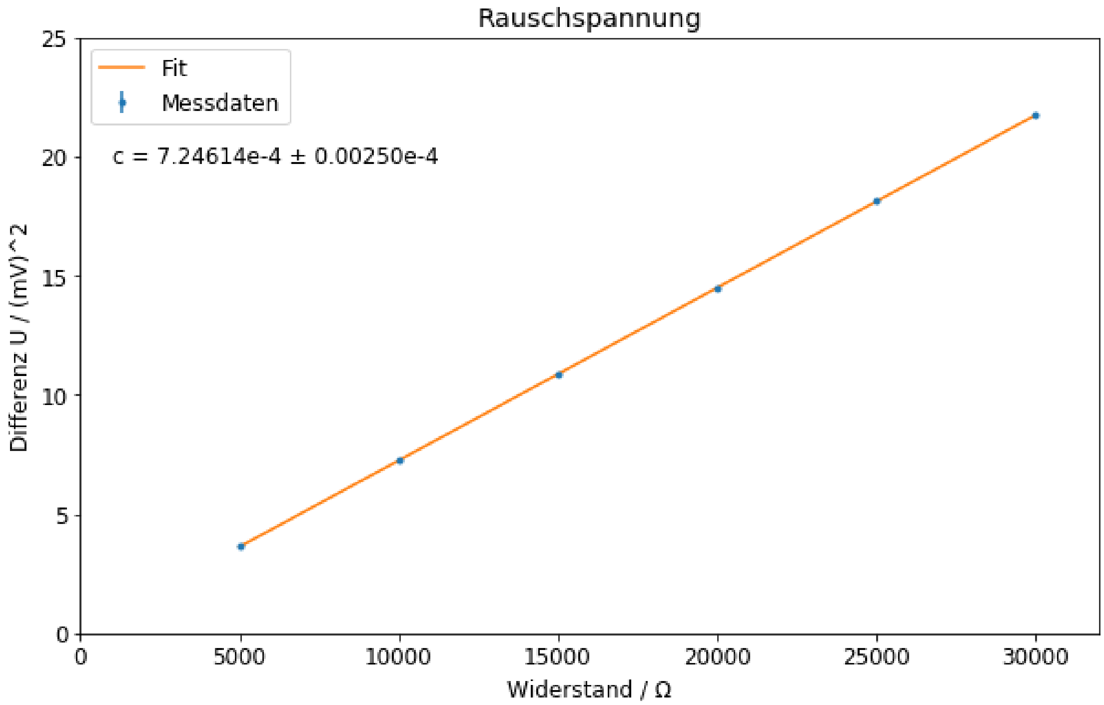
[102]:
```
def lin_fit_func(x, c):
    return x * c
```

[103]:
```
popt_lin, pcov_lin = curve_fit(lin_fit_func, R, Diff, sigma=Diff_Err,␣
 ↪absolute_sigma=True)

c = ValErr.fromFit(popt_lin, pcov_lin, 0)

print(f"Geradensteigung: c = {c.strfmtf(6, -4)}")
```

```
Geradensteigung: c = 7.246144e-4 ± 0.002505e-4
```

[104]:
```
plt.figure(figsize=(10,6))
plt.errorbar(R, Diff, yerr=Diff_Err, fmt='.', label='Messdaten')
plt.plot(R, lin_fit_func(R, *popt_lin), label='Fit')
plt.text(1000, 20, rf'c = {c.strfmtf(5, -4)}', horizontalalignment='left',
     verticalalignment='center')
plt.axis([0,3.2e4,0,25])
plt.xlabel('Widerstand / Ω')
plt.ylabel('Differenz U / (mV)^2')
plt.title('Rauschspannung')
plt.legend(loc='upper left')
```

```
plt.savefig('ur_data_fit.png', dpi=100)
```



0.1.4 Bestimmung der Güte des Fits

[105]:
```
chisquare=np.sum(((lin_fit_func(R,*popt_lin)-Diff)**2/Diff_Err**2))
dof=5 #degrees of freedom
chisquare_red=chisquare/dof
print(f'chi^2 = {chisquare}')
print(f'chi^2 red = {chisquare_red}')
```

```
chi^2 = 83.0708880081837
chi^2 red = 16.61417760163674
```

[106]:
```
from scipy.stats import chi2
prob=round(1-chi2.cdf(chisquare,dof),2)*100
print(f'Wahrscheinlichkeit = {prob}%')
```

```
Wahrscheinlichkeit = 0.0%
```

0.1.5 Ausrechnen der Boltzmannkonstante

[110]:
```
# c = 4kTB <=> k = c/(4TB),
T = 23. + 273.15
T_err = 0.1

k = c.val * (10**-6) / (4. * T * B.val)
k_stat = (c.err / c.val) * k
k_sys = np.sqrt((T_err / T)**2 + (B.err/B.val) ** 2) * k


f"k = {floatfmt(k, 5, -23)} ± {floatfmt(k_stat, 5, -23)} ± {floatfmt(k_sys, 5,␣
↪-23)}"
```

[110]:
```
'k = 11.43531e-23 ± 0.00395e-23 ± 0.22874e-23'
```