245_auswertung

December 9, 2024

[43]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
import scipy.integrate as integrate
import scipy.constants as constants

%run ../lib.ipynb

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

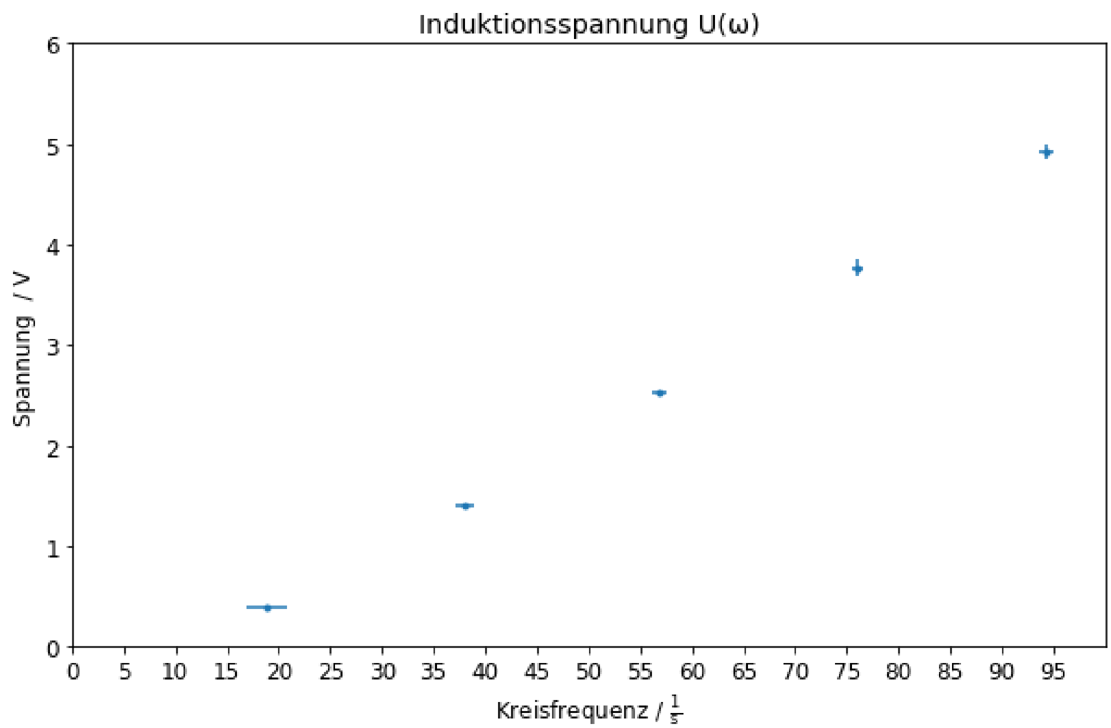
[44]:
```
def linear_origin(x, m):
    return x * m

def linear_translated(x, m, b):
    return x * m + b
```

0.0.1 Aufgabe 2

[45]:
```
data_2a = np.loadtxt('data2a.txt', skiprows=1, usecols=(0,1,2,3), unpack=True)

# Umrechnung f ->
data_2a_angf = data_2a[0] * (2 * np.pi)
data_2a_angf_err = data_2a[1] * (2 * np.pi)

# Umrechnung Peak-to-Peak -> Peak
data_2a_peaku = data_2a[2] * 0.5
data_2a_peaku_err = data_2a[3] * 0.5

data_2b = np.loadtxt('data2b.txt', skiprows=1, usecols=(0,1,2,3), unpack=True)

data_2b_i = data_2b[0]
data_2b_i_err = data_2b[1]

# Umrechnung Peak-to-Peak -> Peak
data_2b_peaku = data_2b[2] * 0.5
data_2b_peaku_err = data_2b[3] * 0.5
```

Induktionsspannung abhängig von Drehfrequenz der Spule

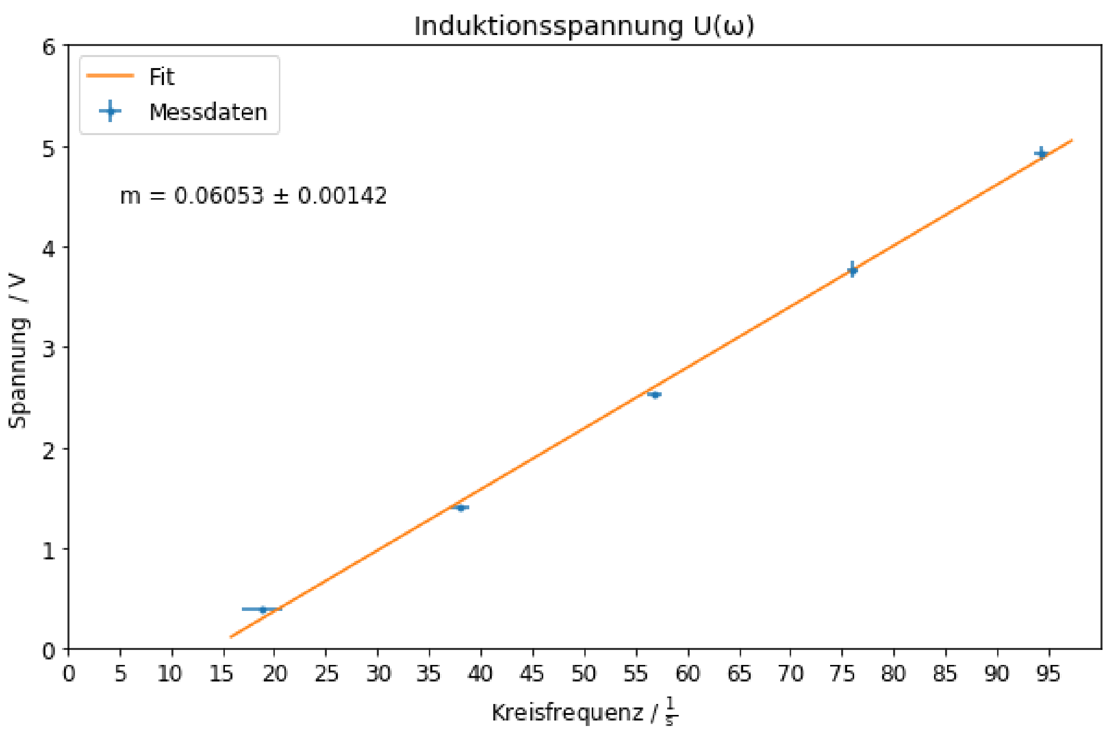
[46]:
```
plt.figure(figsize=(10,6))
plt.errorbar(data_2a_angf, data_2a_peaku, xerr=data_2a_angf_err,␣
↪yerr=data_2a_peaku_err, fmt='.')
plt.axis([0,100,0,6])
plt.xticks(np.arange(0,100,5))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Spannung  / V')
plt.title('Induktionsspannung U( )')
plt.savefig('ui_by_omega.png')
```



[47]:
```
fu_lin_popt, fu_lin_pcov = curve_fit(linear_translated, data_2a_angf,␣
↪data_2a_peaku, [0, 0])

fu_lin_m = ValErr.fromFit(fu_lin_popt, fu_lin_pcov, 0)
fu_lin_b = ValErr.fromFit(fu_lin_popt, fu_lin_pcov, 1)

plt.figure(figsize=(10,6))
plt.errorbar(data_2a_angf, data_2a_peaku, xerr=data_2a_angf_err,␣
↪yerr=data_2a_peaku_err, fmt='.', label='Messdaten')
plt.plot(spacearound(data_2a_angf, 3),␣
↪linear_translated(spacearound(data_2a_angf, 3), *fu_lin_popt), label='Fit')
plt.axis([0,100,0,6])
```

```
plt.xticks(np.arange(0,100,5))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Spannung  / V')
plt.title('Induktionsspannung U( )')
plt.legend(loc='upper left')
plt.text(5, 4.5, rf'm = {fu_lin_m.strfmtf(5, 0)}', horizontalalignment='left',
     verticalalignment='center')
plt.savefig('ui_by_omega_fit.png')

fu_lin_m
```

[47]: ValErr(0.06052529937399517, 0.0014159332063130817)



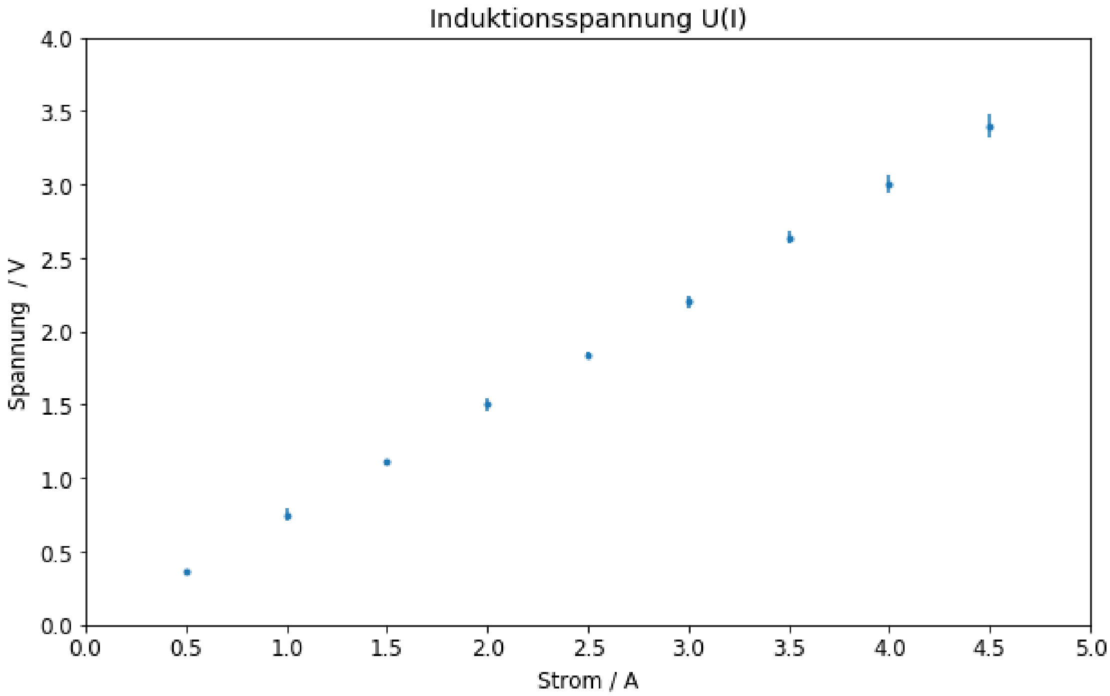
[48]:
```
# Skript: U_peak( ) = B * A * N *
# m = B * A * N <=> B = m / (A * N)

A_spule = 41.7 * 1e-2 * 1e-2
N_spule = 4000

B_val = fu_lin_m.val / (A_spule * N_spule)
B_err = fu_lin_m.err / (A_spule * N_spule)

B_exp = ValErr(B_val, B_err)
```

```
#Helmholtz B-Feld im Zentrum
# B(0) = 8/sqrt(125) * (mu_0 * N * I) / R

N_hh_spule = 124
r_hh_spule = 295 * 0.5 * 1e-3
I_hh_spule = 4

print(r_hh_spule)

B_HH_val = (8 / np.sqrt(125)) * ((constants.mu_0 * N_hh_spule * I_hh_spule) /␣
 ↪(r_hh_spule))
B_HH_err = (8 / np.sqrt(125)) * ((constants.mu_0 * N_hh_spule * 0.1) /␣
 ↪(r_hh_spule))
B_theo = ValErr(B_HH_val, B_HH_err)

print(f"B_exp = ({B_exp.strfmtf(2, -3)}) T\nB_theo = ({B_theo.strfmtf(2, -3)})␣
 ↪T")

sig_exp_theo = np.abs(B_exp.val - B_theo.val)/np.sqrt(B_exp.err ** 2 + B_theo.
 ↪val ** 2)

print(f"Sigma-Abweichung: {sig_exp_theo}")
```

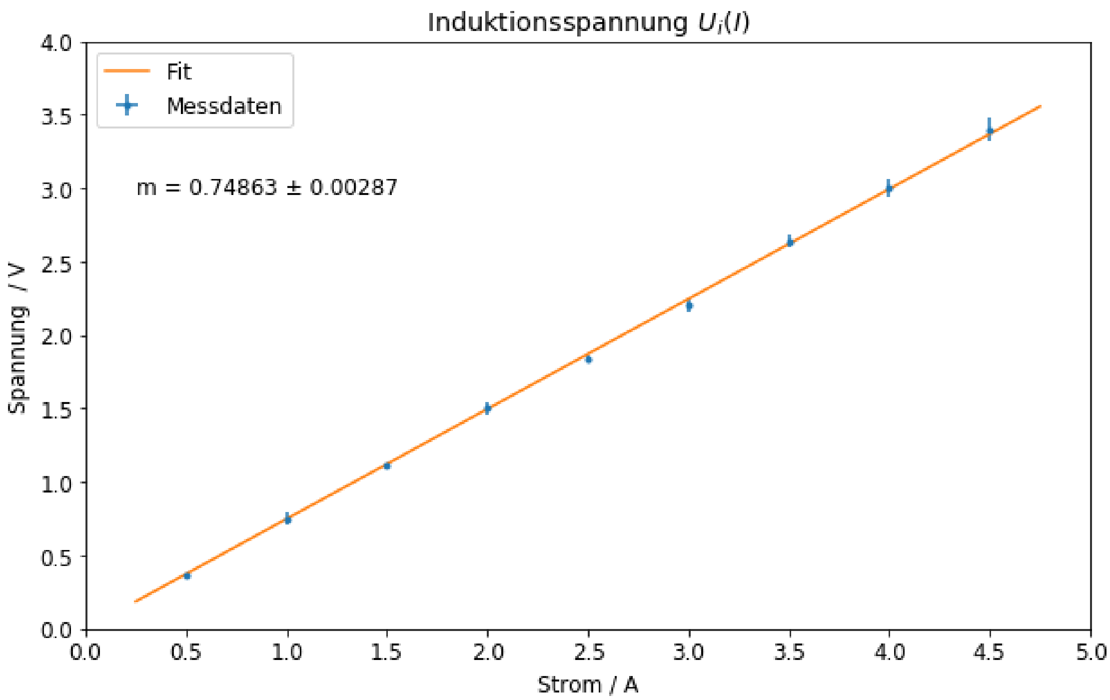
```
0.1475
B_exp = (3.63e-3 ± 0.08e-3) T
B_theo = (3.02e-3 ± 0.08e-3) T
Sigma-Abweichung: 0.1999907036329482
```

Induktionsspannung abhängig vom Strom durch die Spule

[49]:
```
plt.figure(figsize=(10,6))
plt.errorbar(data_2b_i, data_2b_peaku, xerr=data_2b_i_err,␣
 ↪yerr=data_2b_peaku_err, fmt='.')
plt.axis([0,5,0,4])
plt.xticks(np.arange(0,5.5,0.5))
plt.xlabel('Strom / A')
plt.ylabel('Spannung  / V')
plt.title('Induktionsspannung U(I)')
plt.savefig('ui_by_i.png')
```



[50]:
```
iu_lin_popt, iu_lin_pcov = curve_fit(linear_origin, data_2b_i, data_2b_peaku,␣
 ↪[0])

iu_lin_m = ValErr.fromFit(iu_lin_popt, iu_lin_pcov, 0)

plt.figure(figsize=(10,6))
plt.errorbar(data_2b_i, data_2b_peaku, xerr=data_2b_i_err,␣
 ↪yerr=data_2b_peaku_err, fmt='.', label='Messdaten')
plt.plot(spacearound(data_2b_i, 0.25), linear_origin(spacearound(data_2b_i, 0.
 ↪25), *iu_lin_popt), label='Fit')
plt.axis([0,5,0,4])
plt.xticks(np.arange(0,5.5,0.5))
plt.xlabel('Strom / A')
plt.ylabel('Spannung  / V')
plt.title('Induktionsspannung $U_i(I)$')
plt.text(0.25, 3, rf'm = {iu_lin_m.strfmtf(5, 0)}', horizontalalignment='left',
     verticalalignment='center')
plt.legend(loc='upper left')
plt.savefig('ui_by_i_fit.png')
```



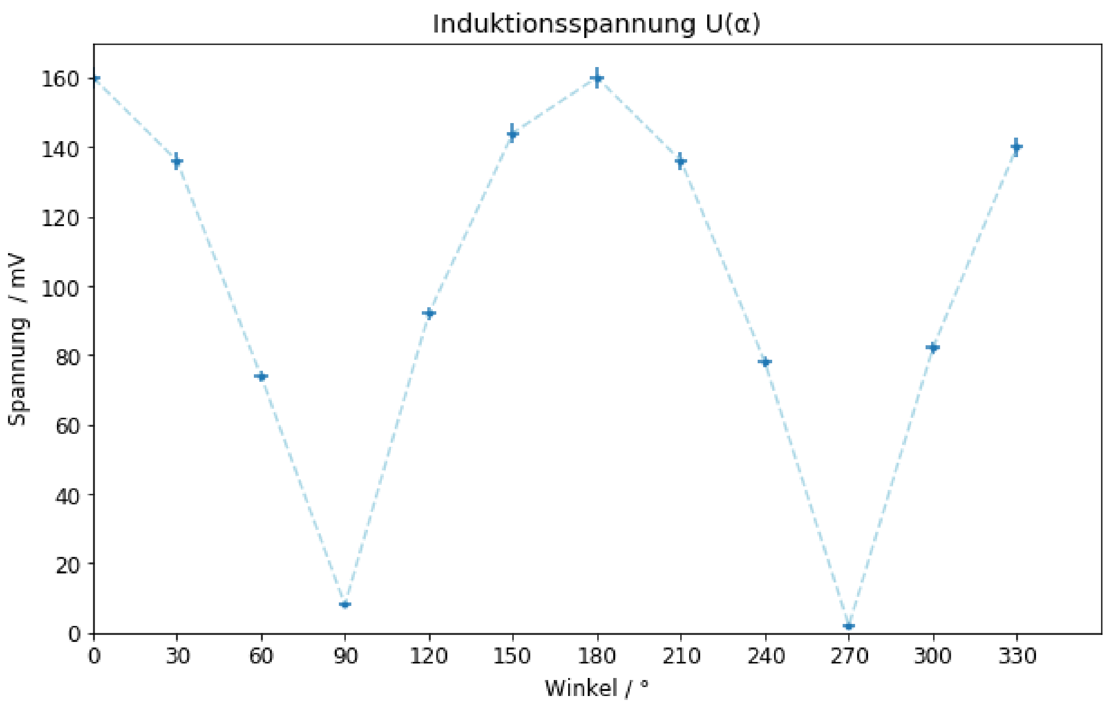
0.0.2 Aufgabe 3

[51]:
```
data_3a = np.loadtxt('data3a.txt', skiprows=1, usecols=(0,1,2), unpack=True)

# Winkel
data_3a_ang = data_3a[0]
data_3a_ang_err = data_3a[1]

# Umrechnung Peak-to-Peak -> Peak
data_3a_peaku = data_3a[2] * 0.5


data_3b = np.loadtxt('data3b.txt', skiprows=1, usecols=(0,1,2,3,4,5,6,7),␣
↪unpack=True)

# Umrechnung f ->
data_3b_angf = data_3b[0] * (2 * np.pi)
data_3b_angf_err = data_3b[1] * (2 * np.pi)

# Umrechnung Peak-to-Peak -> Peak
data_3b_peakuind = data_3b[2] * 0.5
data_3b_peakuind_err = data_3b[3] * 0.5

# Umrechnung Peak-to-Peak -> Peak
```

```
data_3b_peakuhh = data_3b[4] * 0.5
data_3b_peakuhh_err = data_3b[5] * 0.5

data_3b_ihh = data_3b[6] * 0.5
data_3b_ihh_err = data_3b[7] * 0.5
```

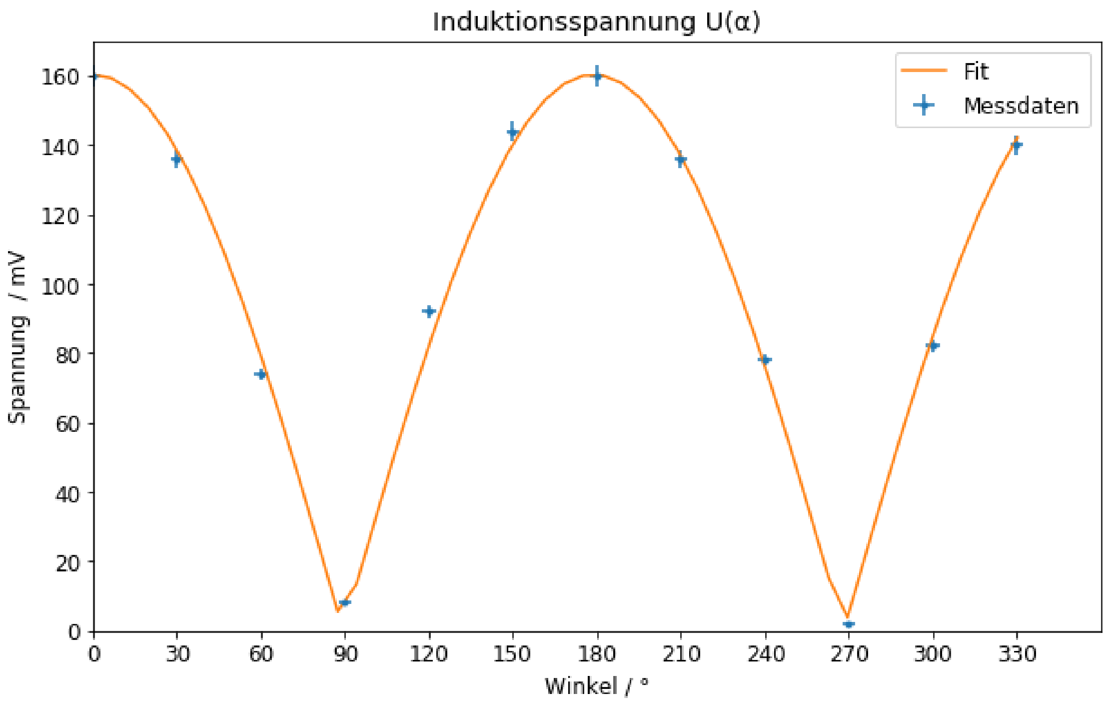
Induktionsspannung abhängig vom Winkel der Spule

[52]:
```
plt.figure(figsize=(10,6))
plt.errorbar(data_3a_ang, data_3a_peaku, xerr=data_3a_ang_err,␣
 ↪yerr=data_3a_peaku * 0.02, fmt='.')
plt.plot(data_3a_ang, data_3a_peaku, color='lightblue', linestyle='--')
plt.axis([0,360,0,170])
plt.xticks(np.arange(0,340,30))
plt.xlabel('Winkel / °')
plt.ylabel('Spannung  / mV')
plt.title('Induktionsspannung U( )')
plt.savefig('ui_by_alpha.png')
```



[53]:
```
def fit_cos(x, a, b):
    return a * np.abs(np.cos(x * b))

au_cos_popt, au_cos_pcov = curve_fit(fit_cos, data_3a_ang, data_3a_peaku, [160.
 ↪, 1./59.])
```

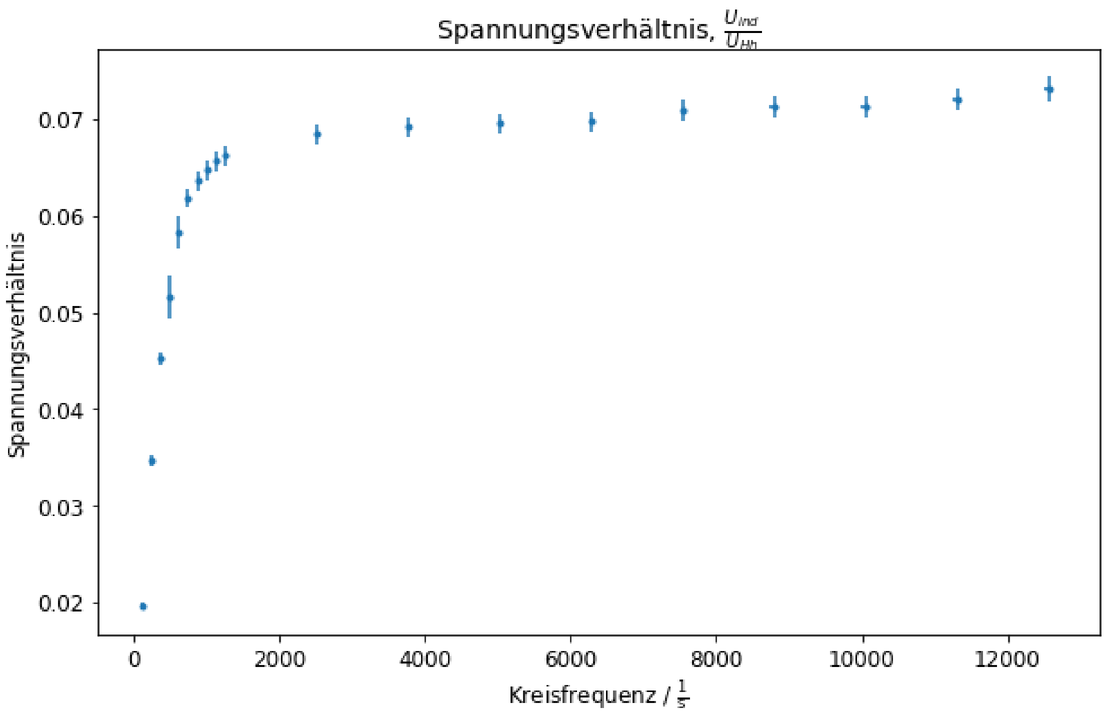
```
plt.figure(figsize=(10,6))
plt.errorbar(data_3a_ang, data_3a_peaku, xerr=data_3a_ang_err,␣
 ↪yerr=data_3a_peaku * 0.02, fmt='.', label='Messdaten')
plt.plot(spacearound(data_3a_ang, 0.25), fit_cos(spacearound(data_3a_ang, 0.
 ↪25), *au_cos_popt), label='Fit')
plt.axis([0,360,0,170])
plt.xticks(np.arange(0,340,30))
plt.xlabel('Winkel / °')
plt.ylabel('Spannung  / mV')
plt.title('Induktionsspannung U( )')
plt.legend(loc='upper right')
plt.savefig('ui_by_alpha_fit.png')
```



Verhältnis von induzierter und angelegter Spannung, abhängig von der Frequenz der Induktionsspule

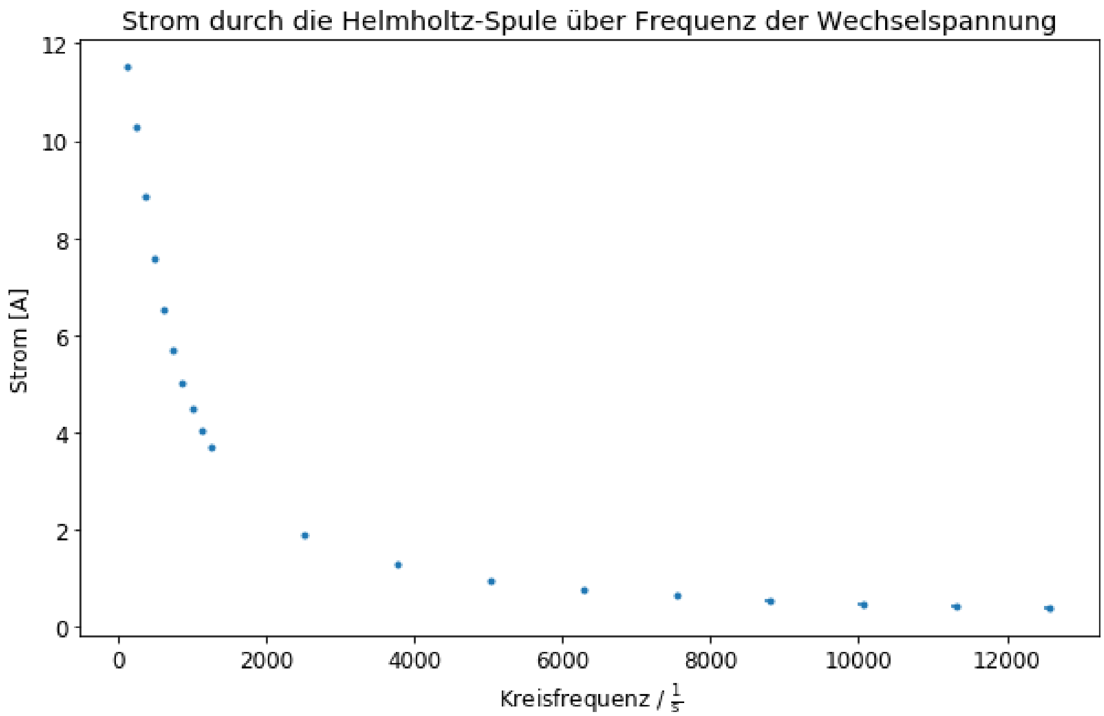
[54]:
```
plt.figure(figsize=(10,6))

#rel_U = data_2a_peakuind / data_2a_peakuhh
#rel_U_err = (1 / data_2a_peakuhh) * np.sqrt(data_2a_peakuind_err**2 +␣
 ↪(data_2a_peakuind * data_2a_peakuhh_err / data_2a_peakuhh)**2)
```

```
rel_U, rel_U_err = div_with_err(data_3b_peakuind * 1e-3, data_3b_peakuind_err *␣
↪1e-3, data_3b_peakuhh, data_3b_peakuhh_err)

plt.errorbar(data_3b_angf, rel_U, xerr=data_3b_angf_err, yerr=rel_U_err, fmt='.
↪')
#plt.axis([0,13000,0,0.065])
#plt.xticks(np.arange(0,14000,1000))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Spannungsverhältnis')
plt.title(r'Spannungsverhältnis, $\frac{U_{ind}}{U_{Hh}}$')
plt.savefig('ui_to_uhh_by_f.png')
```



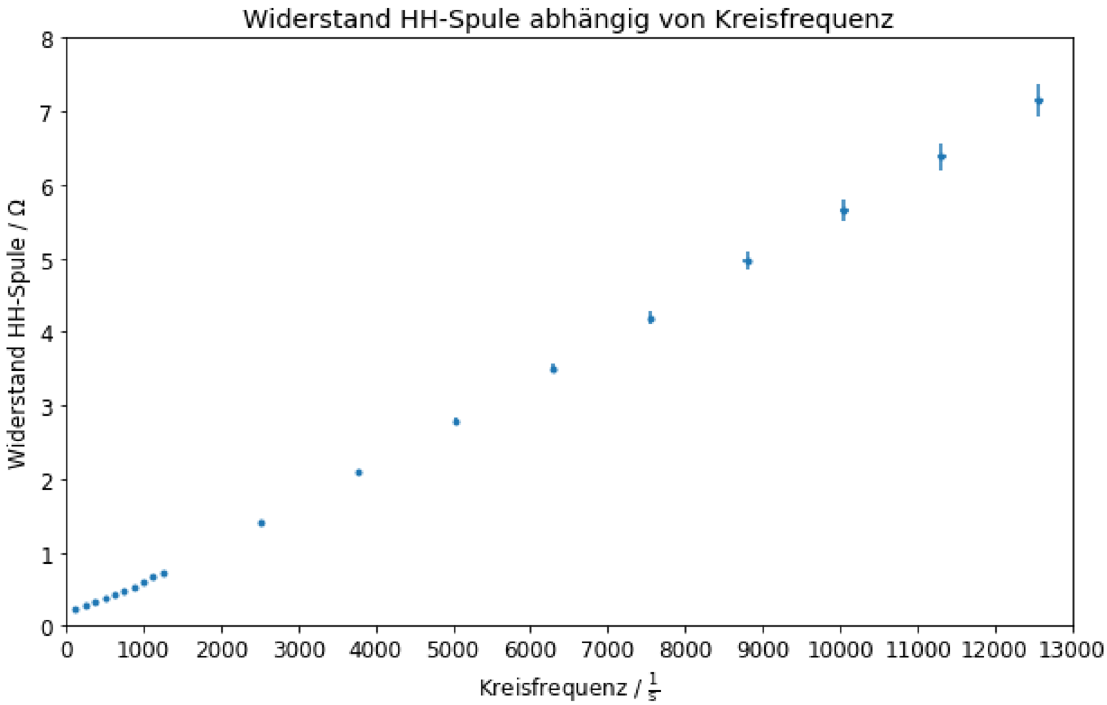
[55]:
```
plt.figure(figsize=(10,6))

#rel_U = data_2a_peakuind / data_2a_peakuhh
#rel_U_err = (1 / data_2a_peakuhh) * np.sqrt(data_2a_peakuind_err**2 +␣
↪(data_2a_peakuind * data_2a_peakuhh_err / data_2a_peakuhh)**2)

#rel_U, rel_U_err = div_with_err(data_3b_peakuind * 1e-3, data_3b_peakuind_err␣
↪* 1e-3, data_3b_peakuhh, data_3b_peakuhh_err)

plt.errorbar(data_3b_angf, data_3b_ihh, xerr=data_3b_angf_err,␣
↪yerr=data_3b_ihh_err, fmt='.')
```

```python
#plt.axis([0,13000,0,0.065])
#plt.xticks(np.arange(0,14000,1000))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Strom [A]')
plt.title(r'Strom durch die Helmholtz-Spule über Frequenz der Wechselspannung')
plt.savefig('i_by_acf.png')
```

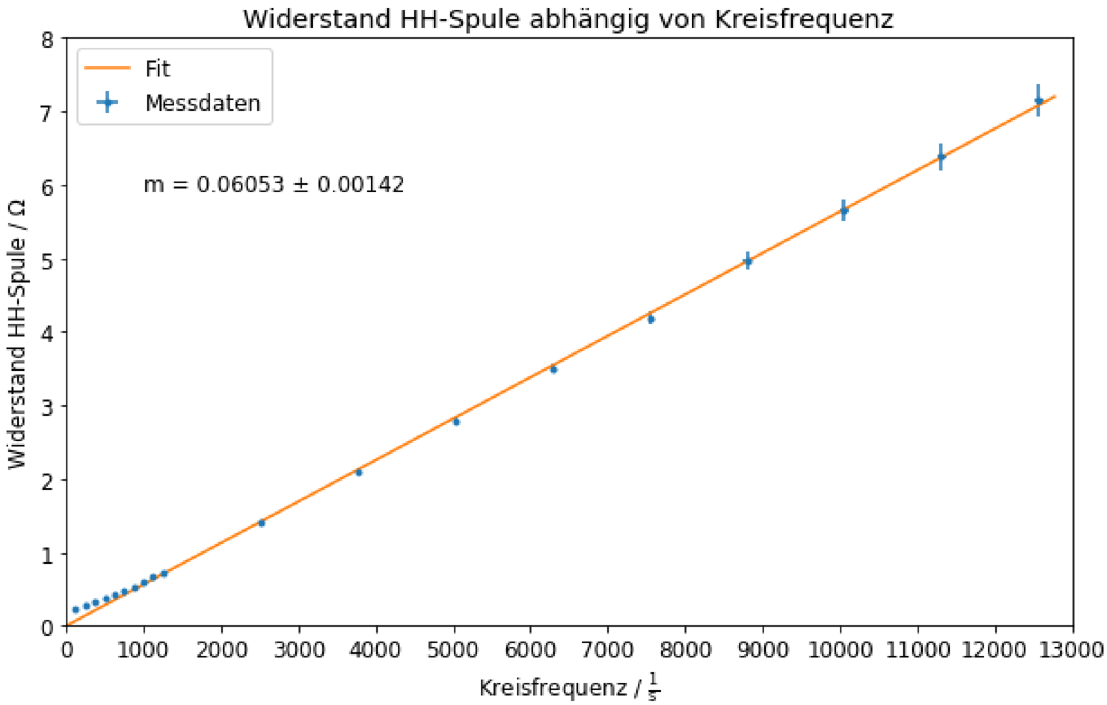


Widerstand der Helmholtz-Spule abhängig von der Frequenz der Induktionsspule

[56]:
```python
plt.figure(figsize=(10,6))

R_HH, R_HH_err = div_with_err(data_3b_peakuhh, data_3b_peakuhh_err,␣
 ↪data_3b_ihh, data_3b_ihh_err)

plt.errorbar(data_3b_angf, R_HH, xerr=data_3b_angf_err, yerr=R_HH_err, fmt='.')
plt.axis([0,13000,0,8])
plt.xticks(np.arange(0,14000,1000))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Widerstand HH-Spule / Ω')
plt.title(r'Widerstand HH-Spule abhängig von Kreisfrequenz')
plt.savefig('rhh_by_f.png')
```



[57]:
```
frhh_lin_popt, frhh_lin_pcov = curve_fit(linear_origin, data_3b_angf, R_HH, [0])

frhh_lin_m = ValErr.fromFit(fu_lin_popt, fu_lin_pcov, 0)
#frhh_lin_b = ValErr.fromFit(fu_lin_popt, fu_lin_pcov, 1)

plt.figure(figsize=(10,6))
plt.errorbar(data_3b_angf, R_HH, xerr=data_3b_angf_err, yerr=R_HH_err, fmt='.',␣
 ↪label='Messdaten')
plt.plot(spacearound(data_3b_angf, 200),␣
 ↪linear_origin(spacearound(data_3b_angf, 200), *frhh_lin_popt), label='Fit')
plt.axis([0,13000,0,8])
plt.xticks(np.arange(0,14000,1000))
plt.xlabel(r'Kreisfrequenz / $\frac{1}{\mathrm{s}}$')
plt.ylabel('Widerstand HH-Spule / Ω')
plt.title(r'Widerstand HH-Spule abhängig von Kreisfrequenz')
plt.legend(loc='upper left')
plt.text(1000, 6, rf'm = {frhh_lin_m.strfmtf(5, 0)}',␣
 ↪horizontalalignment='left',
     verticalalignment='center')
plt.savefig('rhh_by_f_fit.png')
```



[58]:
```
frhh_lin_m
```

[58]:
```
ValErr(0.06052529937399517, 0.0014159332063130817)
```

0.0.3 Aufgabe 4

Ohne Kompensation

[59]:
```
A_spule = 41.7 * 1e-2 * 1e-2
N_spule = 4000

freq_ok = ValErr(14.8, 0.2) * 2 * np.pi
u_ind_ok = ValErr(0.148, 0.001) * 0.5

print(freq_ok, u_ind_ok)

# U = B A N w <=> B = U / A N w

B_erd_ok = u_ind_ok.val / (A_spule * freq_ok.val * N_spule)
B_erd_ok_err = np.sqrt((freq_ok.err/freq_ok.val)**2 + (u_ind_ok.err/u_ind_ok.
 ↪val)**2) * B_erd_ok

print(f"({(ValErr(B_erd_ok, B_erd_ok_err) * 1e6).strfmtf(1, 0)}) µT")
```

```
ValErr(92.99114254625788, 1.2566370614359172) ValErr(0.074, 0.0005)
```

```
(47.7 ± 0.7) µT
```

Mit Kompensation

[60]:
```
freq_mk = ValErr(14.7, 0.3) * 2 * np.pi
i_mk = ValErr(0.0598, 0.0002)
u_ind_mk = ValErr(0.045, 0.002) * 0.5

print(freq_mk, u_ind_mk)

B_erd_hor = u_ind_mk.val / (A_spule * freq_mk.val * N_spule)
B_erd_hor_err = np.sqrt((freq_mk.err/freq_mk.val)**2 + (u_ind_mk.err/u_ind_mk.
 ↪val)**2) * B_erd_ok

print(f"B_hor = ({(ValErr(B_erd_hor, B_erd_hor_err) * 1e6).strfmtf(1, 0)}) µT")

N_hh_spule = 124
r_hh_spule = 295 * 0.5 * 1e-3

B_vert = (8 / np.sqrt(125)) * ((constants.mu_0 * N_hh_spule * i_mk.val) /␣
 ↪(r_hh_spule))
B_vert_err = (8 / np.sqrt(125)) * ((constants.mu_0 * N_hh_spule * i_mk.err) /␣
 ↪(r_hh_spule))

print(f"B_vert = ({(ValErr(B_vert, B_vert_err) * 1e6).strfmtf(2, 0)}) µT")

alpha_rad = np.arctan(B_vert / B_erd_hor)
alpha_deg = (360 / (2 * np.pi)) * alpha_rad

alpha_rad_err = np.sqrt((B_erd_hor * B_vert_err / (B_vert**2 +␣
 ↪B_erd_hor**2))**2 + (B_vert * B_erd_hor_err / (B_vert**2 + B_erd_hor**2))**2)
alpha_deg_err = (360 / (2 * np.pi)) *alpha_rad_err

print(alpha_deg, "+-", alpha_deg_err)
```

```
ValErr(92.36282401553991, 1.8849555921538759) ValErr(0.0225, 0.001)
B_hor = (14.6 ± 2.3) µT
B_vert = (45.20 ± 0.15) µT
72.09525774376331 +- 2.67842382397209
```