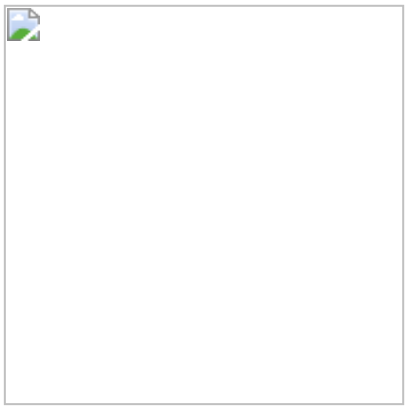
Kernel: Python 3 (Ubuntu Linux)



Fakultät für Physik

Physikalisches Praktikum P1 für Studierende der Physik

Versuch P1-71, 72, 73 (Stand: Dezember 2022)

Raum F1-14

Spezifische Ladung des Elektrons

Name: Aichert Vorname: Julius E-Mail: uhoeb@student.kit.edu

Name: Achtner Vorname: Martin E-Mail: urrvl@student.kit.edu

Gruppennummer: Mo01

Betreuer: _____

Versuch durchgeführt am: 8.11.2023

Beanstandungen:

Testiert am: _____ Testat: _____

Durchführung

Aufgabe 1: Fadenstrahlrohr

Hinweise zu allen hier durchzuführenden Messungen finden Sie in der Datei Hinweise-Aufgabe-1.md.

Aufgabe 1.1: Magnetfeld im Fadenstrahlrohr

- Schätzen Sie das Magnetfeld B, entlang der Mittelebene zwischen H_1 und H_2 im Inneren der Spulen, mit Hilfe einer baugleichen, weiteren Helmholtzspule H_3, einer, an verschiedenen Stellen mit Bohrungen versehenen, Holzplatte M, und einer Hall-Sonde ab. Dabei bezeichnet r im Folgenden den Abstand von der Symmetrieachse des Spulenpaares.
- Kalibrieren Sie die Hall-Sonde mit Hilfe einer weiteren, langen Spule, deren Magnetfeld Sie über das Ampèresche Gesetz bestimmen können.
- Diskutieren Sie die Homogenität von $B(r)$ für die vorliegende Spulenanordnung.
- Vergleichen Sie Ihre Messung für $r = 0$ mit Ihrer Erwartung.

In [4]:
```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import kafe2
#import PhyPraKit as ppk
import matplotlib as mpl
import matplotlib.pyplot as plt
import os
import scipy
mpl.rcParams['figure.dpi']=200
mu = scipy.constants.mu_0
```

Lösung:

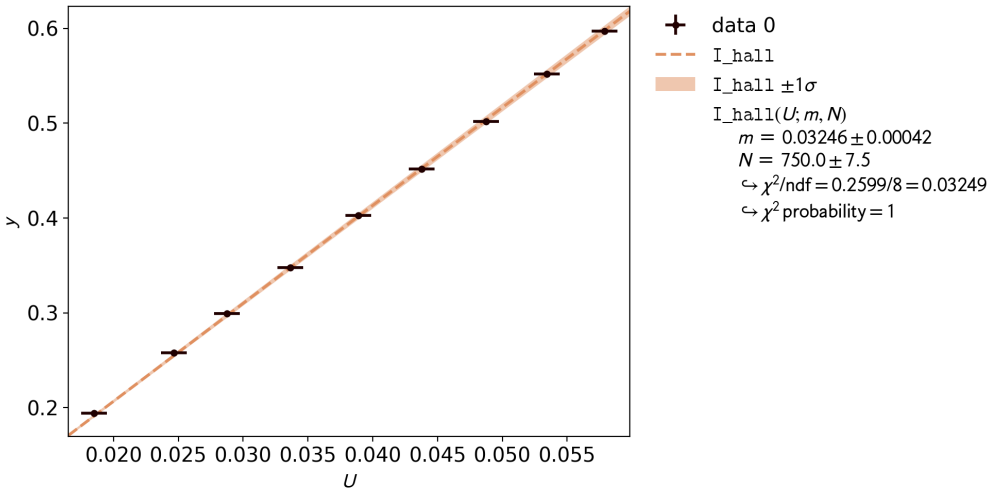
In [6]:
```
hall_helmh = np.genfromtxt( "hall_spannung_helmholtz.csv",
delimiter=",", skip_header=1 ) # Einlesen der Daten, speichern in
Variablen
pos = hall_helmh[:,0]
I_helmh = hall_helmh[:,1]
U_helmh = hall_helmh[:,2]
#print(pos, I_helmh, U_helmh)
hall_cal = np.genfromtxt( "hall_spannung_kalibrierung.csv",
delimiter=",", skip_header=1 ) # Einlesen der Daten, speichern in
Variablen
I_cal = hall_cal[:,0]
U_cal = hall_cal[:,1]
```

Bei einer Stromstärke von 2.002 A varriiert der Wert vor und nach der Messung, aufgrund von Erhitzung um 52.20 mV und 51.49mV Bei der Kallibrierung wurde keine nennenswerten Unterschiede zwischen der ersten und der letzten Messung bei der Hallspannung festgestellt.

In [29]:
```
def I_hall(U, m, N):
    l = 0.3
    return m * U / (mu * N) * l
I_error = 0.001
U_error = 0.001
N_error = 0.01 # relative
fit = kafe2.XYFit(xy_data=[-U_cal * 10**(-3), I_cal],
model_function=I_hall)
fit.add_error(axis='y', err_val=I_error)
fit.add_error(axis='x', err_val=U_error)

fit.add_parameter_constraint(name='N',   value=750,
uncertainty=N_error, relative=True)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.plot()
plot.show()

par_vals = fit.parameter_values
par_err = fit.parameter_errors
m = par_vals[0]
```

```
m_err = par_err[0]
print(m)
```

Out[29]:

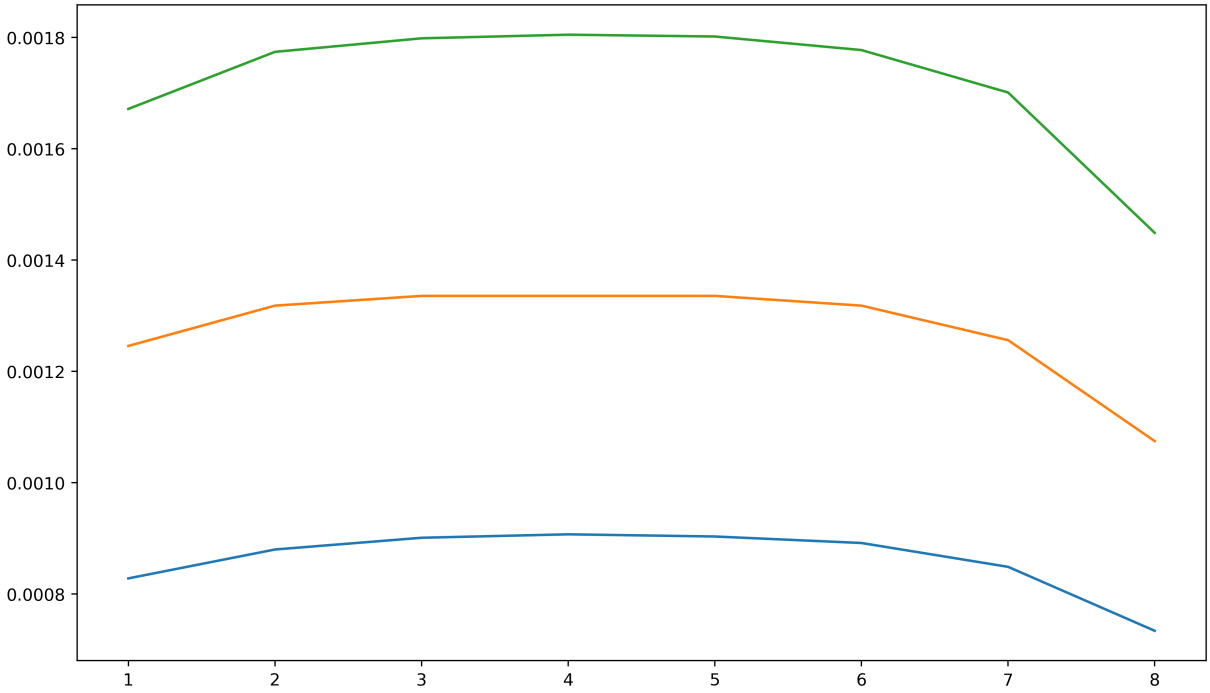



```
0.032459119926100 37
```

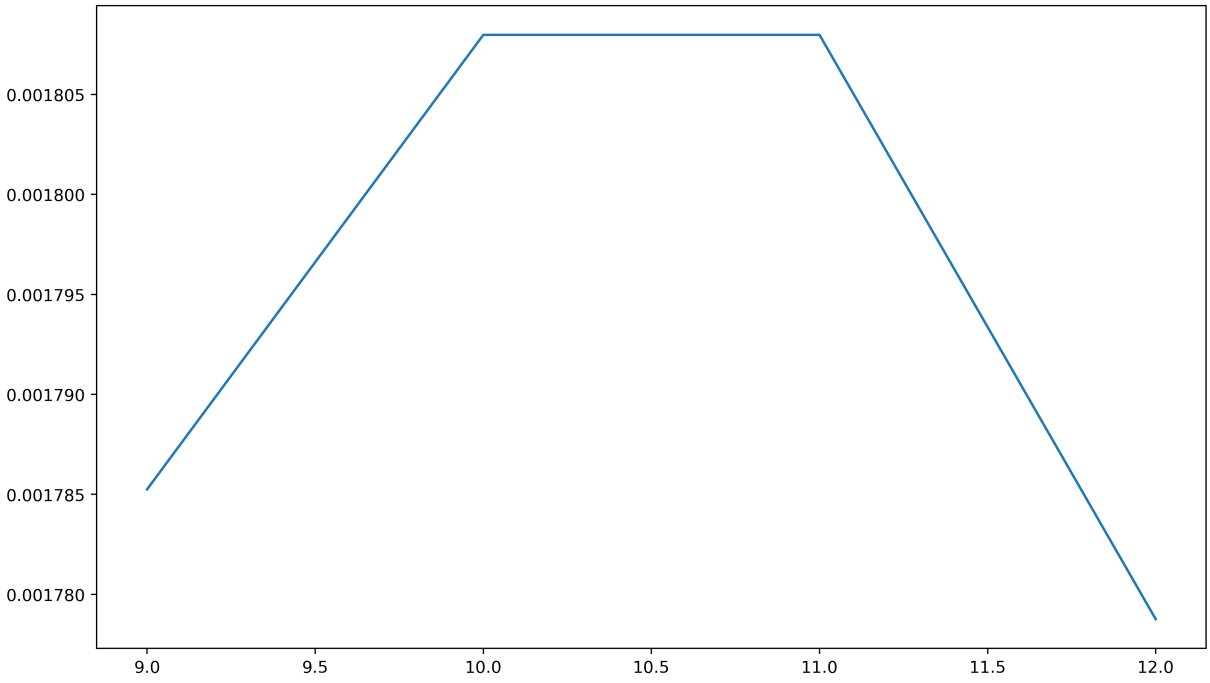
In [92]:

```
def B_helmh(U):
    return (m * U, np.sqrt((U * m_err)**2 + (m * U_error)**2)) # add
error on B with gauss law

B, B_err = B_helmh(U_helmh * 10**(-3))
print(B)

plt.plot(pos[0:24:3], B[0:24:3])
plt.plot(pos[1:25:3], B[1:25:3])
plt.plot(pos[2:26:3], B[2:26:3])
plt.show()

plt.plot(pos[26:36:3], B[26:36:3])
plt.plot(pos[26:36:3], B[26:36:3])
plt.plot(pos[26:36:3], B[26:36:3])
```

Out[92]:
```
[0.00082771 0.00124546 0.00167132 0.00087964 0.00131784 0.00177389
 0.00090074 0.00133537 0.00179824 0.00090691 0.00133537 0.00180473
 0.00090301 0.00133537 0.00180148 0.00089133 0.00131784 0.00177714
 0.00084848 0.00125584 0.00170086 0.00073358 0.0010744  0.00144865
 0.00089457 0.00132076 0.00178525 0.00090691 0.00133894 0.00180797
 0.00090691 0.00134121 0.00180797 0.00090074 0.00132109 0.00177876]
```




```
[<matplotlib.lines.Line2D at 0x7f3eac664070>]
```



Aufgabe 1.2: Elektronenkreisbahn

Bestimmen Sie den Durchmesser d der Elektronenbahn im Fadenstrahlrohr in zwei Messreihen:

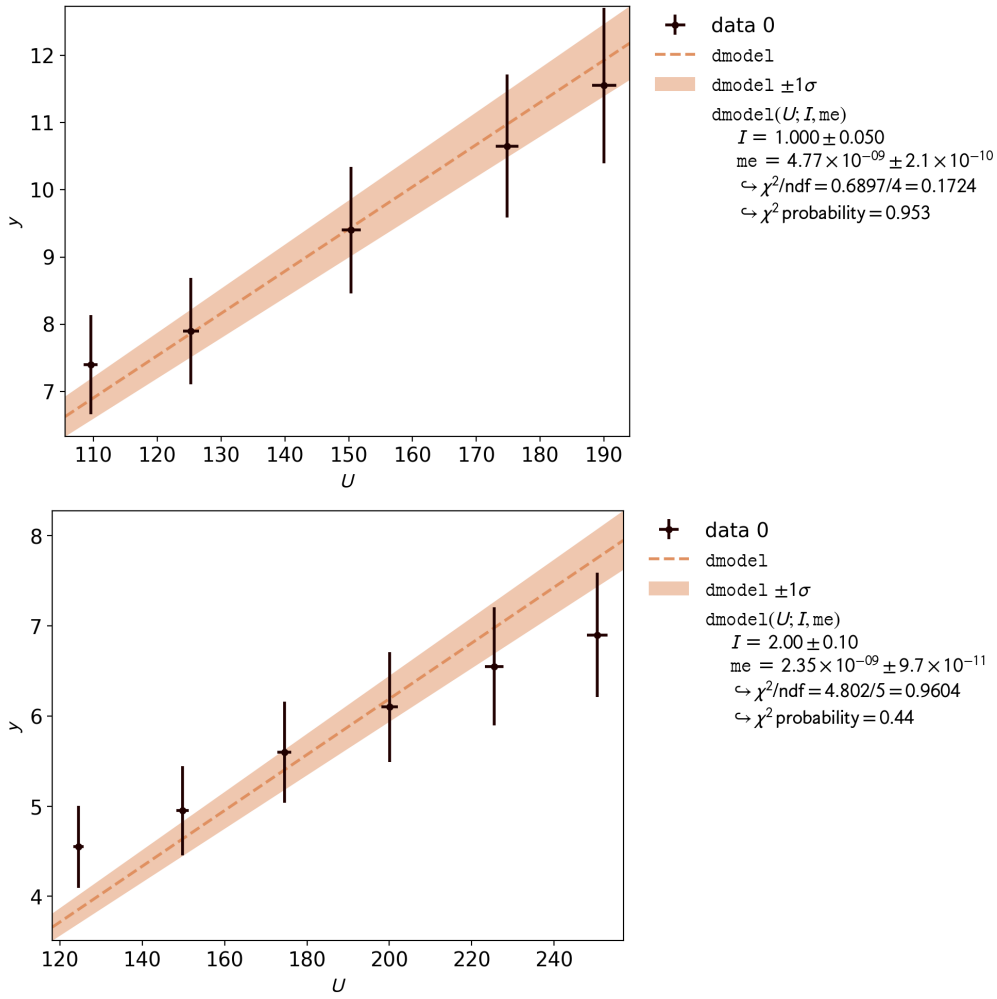
- Als als Funktion der Anodenspannung U (z.B. mit sechs Messpunkten 100; 125; $\ldots 250\ \mathrm{V}$) für zwei Spulenströme (z.B. $1\ \mathrm{A}$ und $2\ \mathrm{A}$).

- Als Funktion des Spulenstroms I (z.B. mit zehn Messpunkten $1,0; 1,2; \ldots 2,0\,\mathrm{A}$) für zwei Anodenspannungen (z.B. $125\,\mathrm{V}$ und $250\,\mathrm{V}$).

In [88]:

```
d_von_U = np.genfromtxt( "d_von_U.csv", delimiter=",", skip_header=1 )
# Einlesen der Daten, speichern in Variablen
U = d_von_U[:,0]
I = d_von_U[:,1]
d = d_von_U[:,2]

def dmodel(U, I, me): # x = U/I**2
    N = 130
    R = 0.15
    x = U
    return x * me * (125 * R**2) / (8 * mu**2 * N**2)

I_error = 0.05 # relative; nochmal beim Netzgerät nachschauen
U_error = 0.01 # relative; nochmal beim Netzgerät nachschauen, hier
nicht die Hall Spannung
d_error = 0.1 # relative

#first with data for which I = 1A
fit = kafe2.XYFit(xy_data=[U[:5], d[:5]], model_function=dmodel)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name='I',   value=1,
uncertainty=I_error, relative=True)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.plot()
plot.show()

#now with data for which I = 2A
fit = kafe2.XYFit(xy_data=[U[5:], d[5:]], model_function=dmodel)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name='I', value=2, uncertainty=I_error,
relative=True)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.plot()
plot.show()
```

Out[88]:



Wert bei 250A und 1 V-1,2V konnte nicht bestimmt werden.

In [93]:

```
"""d_von_U = np.genfromtxt( "d_von_U.csv", delimiter=",",
skip_header=1 ) # Einlesen der Daten, speichern in Variablen
U = d_von_U[:,0]
I = d_von_U[:,1]
d = d_von_U[:,2]

def dmodel(U, I, me): # x = U/I**2
    N = 130
    R = 0.15
    x = U
    return x * me * (125 * R**2) / (8 * mu**2 * N**2)


d_error = 0.1 # relative


#first with data for which I = 1A
fit = kafe2.XYFit(xy_data=[U[:7], d[:7]], model_function=dmodel)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name=I, value=1, uncertainty=I_error,
relative=True)

fit.do_fit()
```

```
plot = kafe2.Plot(fit)
plot.plot()
plot.show()

#now with data for which I = 2A
fit = kafe2.XYFit(xy_data=[U[7:13], d[7:13]], model_function=d)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name=I, value=2, uncertainty=I_error,
relative=True)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.show()"""
```

Out[93]:
```
'd_von_U = np.genfromtxt( "d_von_U.csv", delimiter=",", skip_header=1 )
# Einlesen der Daten, speichern in Variablen\nU = d_von_U[:,0]\nI =
d_von_U[:,1]\nd = d_von_U[:,2]\n\ndef dmodel(U, I, me): # x = U/I**2\n
N = 130\n    R = 0.15\n    x = U \n    return x * me * (125 * R**2) / (8
* mu**2 * N**2)\n\n\nd_error = 0.1 # relative\n\n\n#first with data for
which I = 1A\nfit = kafe2.XYFit(xy_data=[U[:7], d[:7]],
model_function=dmodel) \nfit.add_error(axis=\'y\', err_val=d_error,
relative=True)\nfit.add_error(axis=\'x\', err_val=U_error,
relative=True)\n\nfit.add_parameter_constraint(name=I, value=1,
uncertainty=I_error, relative=True)\n\nfit.do_fit()\nplot =
kafe2.Plot(fit)\nplot.plot()\nplot.show()\n\n#now with data for which I
= 2A\nfit = kafe2.XYFit(xy_data=[U[7:13], d[7:13]], model_function=d)
\nfit.add_error(axis=\'y\', err_val=d_error,
relative=True)\nfit.add_error(axis=\'x\', err_val=U_error,
relative=True)\n\nfit.add_parameter_constraint(name=I, value=2,
uncertainty=I_error, relative=True)\n\nfit.do_fit()\nplot =
kafe2.Plot(fit)\nplot.show()'
```

In [0]:
```
d_von_U = np.genfromtxt( "d_von_U.csv", delimiter=",", skip_header=1 )
# Einlesen der Daten, speichern in Variablen
U = d_von_U[:,0]
I = d_von_U[:,1]
d = d_von_U[:,2]

def dmodel(U, I, me): # x = U/I**2
    N = 130
    R = 0.15
    x = U
    return x * me * (125 * R**2) / (8 * mu**2 * N**2)

I_error = 0.05 # relative; nochmal beim Netzgerät nachschauen
U_error = 0.01 # relative; nochmal beim Netzgerät nachschauen, hier
nicht die Hall Spannung
d_error = 0.1 # relative

#first with data for which I = 1A
fit = kafe2.XYFit(xy_data=[U[:5], d[:5]], model_function=dmodel)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name='I',   value=1,
uncertainty=I_error, relative=True)

fit.do_fit()
```

```
plot = kafe2.Plot(fit)
plot.plot()
plot.show()

#now with data for which I = 2A
fit = kafe2.XYFit(xy_data=[U[5:], d[5:]], model_function=dmodel)
fit.add_error(axis='y', err_val=d_error, relative=True)
fit.add_error(axis='x', err_val=U_error, relative=True)

fit.add_parameter_constraint(name='I', value=2, uncertainty=I_error,
relative=True)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.plot()
plot.show()
```

Aufgabe 2: Methode von Busch

Hinweise zu allen hier durchzuführenden Messungen finden Sie in der Datei Hinweise-Aufgabe-2.md.

Aufgabe 2.1: Vorbereitung der Messung

Machen Sie sich mit der Methode zur Bestimmung der von e/m_e nach der Methode von Busch vertraut. Verändern Sie hierzu bei vorgegebener Beschleunigungsspannung U_z den Spulenstrom, und erklären Sie Ihre Beobachtungen mit eigenen Worten.

Lösung:

Sie können Ihr Protokoll direkt in dieses Dokument einfügen. Wenn Sie dieses Dokument als Grundlage für ein Jupyter notebook verwenden wollen können Sie die Auswertung, Skripte und ggf. bildliche Darstellungen mit Hilfe von python ebenfalls hier einfügen. Löschen Sie hierzu diesen kursiv gestellten Text aus dem Dokument.

Aufgabe 2.2: Bestimmung von e/m_e

Messen Sie für Beschleunigungsspannungen von $U = 200 \ldots 700\,\mathrm{V}$ (in Schritten von $50\,\mathrm{V}$) den nötigen Spulenstrom I, um auf dem Schirm einen Signalpunkt zu erzeugen. Gehen Sie dabei, für jeden Messpunkt so, wie in Aufgabe 2.1 vor. Tragen Sie U geeignet über I^2 auf und ermitteln Sie daraus e/m_e.

Lösung:

Sie können Ihr Protokoll direkt in dieses Dokument einfügen. Wenn Sie dieses Dokument als Grundlage für ein Jupyter notebook verwenden wollen können Sie die Auswertung, Skripte und

ggf. bildliche Darstellungen mit Hilfe von python ebenfalls hier einfügen. Löschen Sie hierzu diesen kursiv gestellten Text aus dem Dokument.

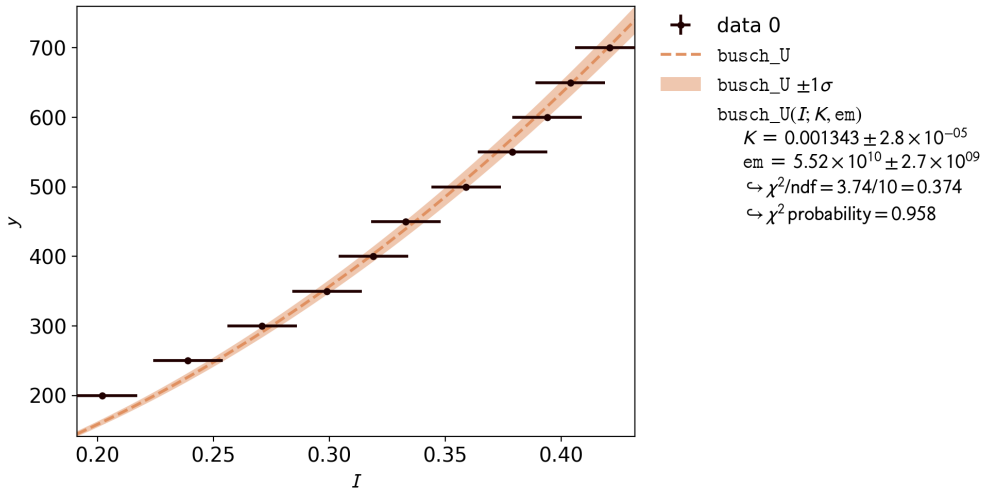
In [37]:
```
# bestimme zuerst konstante K * (ds - d2) für den Mittelwert des
Magnetfeldes
l = 0.2  # +- 0.0005
R = 0.045 # +-0.0005
N = 3000
d2 = 0.08 # +- 0.001
ds = 0.15
def B(a):
    return mu * N / l * 0.567 * (a/np.sqrt(R**2 + a**2) + (l -
a)/np.sqrt(R**2 + (l - a)**2))
k =  scipy.integrate.quad(B, d2, ds)
```

In [53]:
```
k0 = k[0]#0.0013435370205851793

k1 = 0.0013412079923252028 # + in l
k2 = 0.0013458624795267087 # - in l

k3 = 0.0013407491868634852 # + in R
k4 = 0.0013463147265669803 # - in R

k5 = 0.0013242059973940048 # + in d2
k6 = 0.0013628503364845855 # - in d2

dk = np.sqrt((k1-k0)**2 + (k2-k0)**2 + (k3-k0)**2 + (k4-k0)**2 + (k5-
k0)**2 + (k6-k0)**2 + (k[1])**2) # systematic and numerical error on k
print(dk)
```

Out[53]:
```
2.7803101049229007e-05
```

In [72]:
```
busch_data = np.genfromtxt("busch.csv", delimiter=",", skip_header=1 )
U_b = busch_data[:,0]
I_b = busch_data[:,1]
print(I_b)

def busch_U(I, K, em):
    return 1/(np.pi * 8) * K**2 * em * I**2

I_error = 0.015
U_error = 1

fit = kafe2.XYFit(xy_data=[I_b, U_b], model_function=busch_U)
fit.add_error(axis='y', err_val=U_error)
fit.add_error(axis='x', err_val=I_error)

fit.add_parameter_constraint(name="K", value=k0, uncertainty=dk)

fit.do_fit()
plot = kafe2.Plot(fit)
plot.plot()
plot.show()
```

Out[72]: [0.202 0.239 0.271 0.299 0.319 0.333 0.359 0.379 0.394 0.404 0.421]



In [0]: