Pendel

November 25, 2023

1 Fakultät für Physik

1.1 Physikalisches Praktikum P1 für Studierende der Physik

Versuch P1-20, 21, 22 (Stand: Oktober 2023)

Raum F1-11

2 Pendel

Name: *surname1* Vorname: *name1* E-Mail: *mail1*

Name: *surname2* Vorname: *name2* E-Mail: *mail2*

Gruppennummer: *groupnumber*

Betreuer:

Versuch durchgeführt am: *date*

Beanstandungen:

Testiert am: _____ Testat: _____

[3]:
```
# Hilfsfunktionen. Nicht weiter beachten
from IPython.core.display import Markdown
from IPython.display import display
def M_helper_func(txt):
    #print(txt)
    #print('fr"'+txt.replace("§","$").replace("{","{{").replace("}","}}").
 ↪replace("[!","{").replace("!]","}")+'"')
    execreturnvalue = ""
    loc = {}
    exec('execreturnvalue = fr"<div style=\'margin-left: 8px;\'>'+txt.
 ↪replace("{","{{").replace("}","}}").replace("[!","{").replace("!]","}")+'</
 ↪div>"', globals(), loc)
    display(Markdown(loc["execreturnvalue"]))
    return loc["execreturnvalue"]

from IPython.core.magic import (register_line_magic,
                                register_cell_magic)

from IPython.core import magic

def register_line_magic(f):
    setattr(f, magic.MAGIC_NO_VAR_EXPAND_ATTR, True)
    return magic.register_line_magic(f)

@register_line_magic
def m(line):
    M_helper_func(line)
```

```
import os
import numpy as np
from numpy import pi
import matplotlib.pyplot as plt
import matplotlib as mpl
import scipy as sci
import kafe2
import pandas as pd
from uncertainties import ufloat, unumpy
from uncertainties.unumpy import nominal_values, nominal_values as nomv,␣
 ↪std_devs, std_devs as stdv
from uncertainties.umath import sqrt as usqrt, sin as usin,exp as uexp
import warnings
warnings.filterwarnings('ignore')
import scipy
mpl.rcParams['figure.dpi'] = 100



class ng:
    def __init__(self,x=None,stat=None,syst=None):
        if x == None:
            self.syst = None
            self.stat = None
        else:
            self.syst = ufloat(x,0)
            self.stat = ufloat(x,0)
            if syst != None:
                self.syst = ufloat(x,syst)
            if stat != None:
                self.stat = ufloat(x,stat)
    def SetSyst(self,syst):
        self.syst = ufloat(nomv(self.syst),syst)
    def SetStat(self,stat):
        self.stat = ufloat(nomv(self.stat),stat)
    def __str__(self):
        return f'{nomv(self.stat):.2}±  {stdv(self.stat):.2}±  {stdv(self.syst):
 ↪.2}'
    def __add__(self,o):
        res = ng()
        if type(o) == ng:
            res.syst = self.syst + o.syst
            res.stat = self.stat + o.stat
        else:
            res.syst = self.syst + o
```

```
            res.stat = self.stat + o
        return res
    def __iadd__(self,o):
        return self.__add__(o)
    def __sub__(self,o):
        res = ng()
        if type(o) == ng:
            res.syst = self.syst - o.syst
            res.stat = self.stat - o.stat
        else:
            res.syst = self.syst - o
            res.stat = self.stat - o
        return res
    def __isub__(self,o):
        return self.__sub__(o)
    def __mul__(self,o):
        res = ng()
        if type(o) == ng:
            res.syst = self.syst * o.syst
            res.stat = self.stat * o.stat
        else:
            res.syst = self.syst * o
            res.stat = self.stat * o
        return res
    def __imul__(self,o):
        return self.__mul__(o)
    def __truediv__(self,o):
        res = ng()
        if type(o) == ng:
            res.syst = self.syst / o.syst
            res.stat = self.stat / o.stat
        else:
            res.syst = self.syst / o
            res.stat = self.stat / o
        return res
    def __itruediv__(self,o):
        return self.__truediv__(o)
    def __pow__(self,o):
        res = ng()
        if type(o) == ng:
            res.syst = self.syst**o.syst
            res.stat = self.stat**o.stat
        else:
            res.syst = self.syst**o
            res.stat = self.stat**o
        return res
    def __ipow__(self,o):
```

```
        return self.__pow__(o)
def sin(o):
    if type(o) == ng:
        res = ng()
        res.syst = usin(o.syst)
        res.stat = usin(o.stat)
        return res
    else:
        return np.sin(o)
def exp(o):
    if type(o) == ng:
        res = ng()
        res.syst = uexp(o.syst)
        res.stat = uexp(o.stat)
        return res
    else:
        return np.exp(o)
def sqrt(o):
    if type(o) == ng:
        res = ng()
        res.syst = usqrt(o.syst)
        res.stat = usqrt(o.stat)
        return res
    else:
        return np.sqrt(o)
```

Wenn Sie die Kugel in den Draht der Aufhängung "hineinfallen" lassen, gehen Sie das Risiko ein den Draht zu zerstören.# Durchführung

2.1 Aufgabe 1: Fadenpendel

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

Nehmen Sie mit dem Fadenpendel die folgenden Untersuchungen vor:

- Untersuchen Sie die Periode T_0 als Funktion der Anfangsauslenkung φ_0 der Kugel bei *großer* Auslenkung aus der Ruhelage. (Sie können die Kugel bis zu 60° auslenken.)
- Bestimmen Sie die Fallbeschleunigung g. Achten Sie in diesem Fall darauf, die Kugel *nicht zu weit* auszulenken. Welchen Effekt erwarten Sie für Ihre Wahl von φ_0, Ihrer ersten Untersuchung zufolge, durch die Annahme der Kleinwinkelnäherung?

---

[4]:
```
###Grundlegende Messungen der Pendelmaße
L = 2.355
ΔL_syst = 0.003

m = 0.860
```

```
Δm_syst = 0.5 * 10**(-3)

r = 1/2*61.75E-3
Δr_syst =2E-3 #stat
Δr_stat = 1E-3
s = L +r/2#Gewicht des Fadens wird vernachlässigt
Δs_syst = np.sqrt(ΔL_syst**2+Δr_syst**2)
Δs_stat = Δr_stat

ΔSchranke = 0.002

### Erste Messungen kleiner Auslenkungen
%m <h5> Bestimmung von g unter gültiger Kleinwinkelnäherung </h5>
%m Zunächst wurde das Pendel nur 5° ausegelenkt, um innerhalb der␣
 ↪Kleinwinkelnäherung zu bleiben und es wurden die Dauern verschiedene␣
 ↪Anzahlen an Perioden mit einer Lichtschranke gemessen. Der aus diesen Werten␣
 ↪gebildeten Wert wurde aus folgendem Fit bestimmt, während die Unsicherheit␣
 ↪als Unsicherheit einer Stichprobe und als rel. Unsicherheit auf den␣
 ↪erhaltenen Wert bestimmt wurde.

T_mess = [3.051,9.204,15.356,21.509,27.665,33.833,39.978,46.128,52.266,58.436]␣
 ↪#hier Periodendauern eintragen
i = np.arange(0,len(T_mess))
i = [(1+k*2) for k in i]
def lin_model(i,T=3):
    return i*T

xy_data = kafe2.XYContainer(x_data=i, y_data=T_mess)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)
lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = 't in s'
p.plot()
p.show()

T_perP = []
for i in range(0,len(T_mess)):
    T_perP +=[T_mess[i]/(1+i*2)]
T = lin_fit.parameter_values[0]
ΔT_stat = np.var([T_perP],ddof=1)
ΔT_syst = ΔSchranke*T

%m Er ergab sich eine Periodendauer $T = ([!T:.2!]\pm_{stat}[!ΔT_stat*1E5:.2f!
 ↪]\cdot 10^{-5}\pm_{syst}[!ΔT_syst:.2!])s$.
```

```
%m Betrachtet wurden dabei als statistischer Fehler der statistische Fehler in␣
 ↪Form der aus den Messungen errechneten Varianz und als systematischer Fehler␣
 ↪der relative Messfehler der Lichtschranke von [!ΔSchranke*100.:.2!]% auf den␣
 ↪Mittelwert.

###Bestimmung von g aus den Periodendauern
%m Mit dieser Periodendauer wurde in der Kleinwinkelnäherung ein Wert für die␣
 ↪Erdbeschleunigung g berechnet, wobei die Fehler mit Gausscher␣
 ↪Fehlerfortpflanzung bestimmt wurden. Hinzu kamen dabei der statistische␣
 ↪Fehler in Form der Varianz auf mehrere Messungen des Radius der Kugel und␣
 ↪damit auch die Schwerpunktsdistanz. Systematische Fehler wurden bei der␣
 ↪Messung der Länge $\Delta L_{syst} = [!ΔL_syst:.2!]\,m$ und der Masse␣
 ↪$\Delta m_{syst} = [!Δm_syst:.2!]\,kg$ in Form einer Angabe, auf den Radius␣
 ↪der Kugel durch Annahme von $\Delta r_{syst} = [!Δr_syst:.2!]\,m$␣
 ↪Genauigkeit des Messinstruments berücksichtigt.
g = 4*np.pi**2/T**2 *(s+2/5*r**2/s)
#Δg = sqrt((dg/dT)**2*ΔT**2+(dg/ds)**2*Δs**2+(dg/dr)**2*Δr**2) = 2*np.pi*np.
 ↪sqrt((T**(-2)*(s+2/5*r**2/s))**2*ΔT**2+(T**(-1)*(1-2/
 ↪5*r**2*s**(-2)))**2*Δs**2+(T**(-1)*(2/5*2*r/s))**2*Δr**2)
Δg_syst = 2*np.pi*np.sqrt((T**(-2)*(s+2/5*r**2/s))**2*ΔT_syst**2+(T**(-1)*(1-2/
 ↪5*r**2*s**(-2)))**2*Δs_syst**2+(T**(-1)*(2/5*2*r/s))**2*Δr_syst**2)
Δg_stat = 2*np.pi*np.sqrt((T**(-2)*(s+2/5*r**2/s))**2*ΔT_stat**2+(T**(-1)*(1-2/
 ↪5*r**2*s**(-2)))**2*Δs_stat**2+(T**(-1)*(2/5*2*r/s))**2*Δr_syst**2)
%m Dabei ergab sich eine Erdbeschleunigung $g= ([!g:.2!]\pm_{stat}[!Δg_stat:.2!
 ↪]\pm_{syst}[!Δg_syst:.2!])\frac{m}{s^2}$. Dieser Wert ist mit dem␣
 ↪Literaturwert von $9,809599\frac{m}{s^2}$ (in Mannheim nach https://www.ptb.
 ↪de/cms/fileadmin/internet/fachabteilungen/abteilung_1/1.1_masse/1.15/tabelle.
 ↪pdf) im Rahmen der angenommenen Unsicherheiten nicht kompatibel. Dies liegt␣
 ↪vermutlich vor allem an zu klein geschätzten systematischen Fehlern. Diese␣
 ↪könnten entweder in veralteten Angaben oder an zu kleinen Ungenauigkeiten␣
 ↪der Messinstrumente liegen.

%m <h5> Überprüfung der Gültigkeit der Kleinwinkelnäherung </h5>
%m Daraufhin wurde die Kleinwinkelnäherung für größere Winkel überprüft, indem␣
 ↪das Pendel bis zu 60° Grad ausgelenkt und die Dauer von 5 Perioden mit␣
 ↪selbiger Lichtschranke gemessen wurde. Die daraus berechneten Werte für g␣
 ↪weichen gut erkennbar mit steigendem Winkel immer mehr vom in der␣
 ↪Kleinwinkelnäherung bestimmten Wert ab. Gründe dafür sind zum einen, dass␣
 ↪der Sinus für steigende Winkel immer weniger linear ist, aber auch andere␣
 ↪Faktoren beeinträchtigen die Genauigkeit der Messungen bei höheren Winkel.␣
 ↪Dazu zählen unter anderem, dass keine perfekte Spannung des Fadens mehr␣
 ↪garantiert werden kann und die zunehmende Geschwindigkeit, die zu größeren␣
 ↪Reibungseffekten führt.

###Messung Periodendauern großer Winkel
 _0 = [60,55,50,45,40,35,30,25,20,15,10,5] #in Grad
```

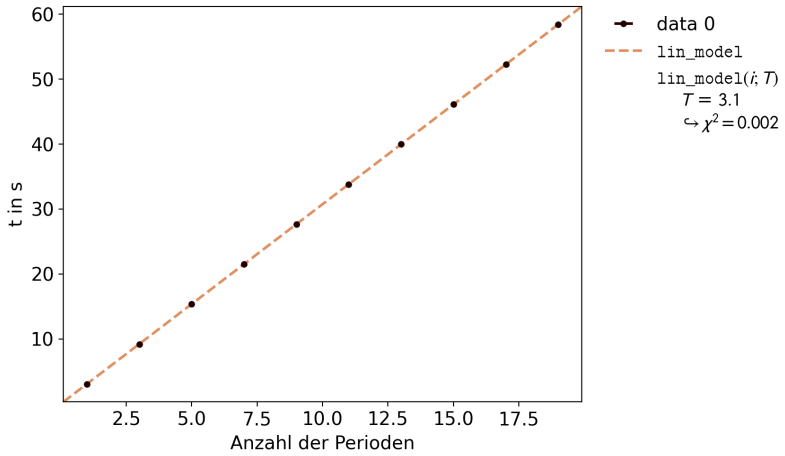
```
 _0 = [i *np.pi/180 for i in  _0] #Umrechnung in Rad
Δ _0_syst = 2 * np.pi/180
T_0_mess = [16.457,16.274,16.071,15.945,15.829,15.725,15.619,15.531,15.475,15.
 ↪421,15.379,15.356]
T_0_mess = [1/5*i for i in T_0_mess] #um Periodendauer einer Periode zu erhalten
#Anlegen eines kleinen Fits ohne Hinzufügen von Fehlern, um die statistische␣
 ↪Varianz zu erhalten
def sinsquare_model( _0,T=3):
    return T*(1+1/4*np.sin(1/2* _0)*np.sin(1/2* _0))

xy_data = kafe2.XYContainer(x_data= _0, y_data=T_0_mess)
sinsquare_fit = kafe2.Fit(data=xy_data, model_function=sinsquare_model)

sinsquare_fit.do_fit()
ΔT_0_stat = np.sqrt(np.var(T_0_mess))
#print(ΔT_0_stat)
ΔT_0_stat = np.sqrt(sinsquare_fit.parameter_cov_mat[0][0])
#print(ΔT_0_stat)
ΔT_0_syst = 0.002 #rel Fehler, wird mit jeweiligem Wert mult




#Berechnen von g aus den Werten großer Winkel in Kleinwinkelnäherung
g_big = [4*np.pi**2/i**2 *(s+2/5*r**2/s) for i in T_0_mess]
Δg_big_syst = [2*np.pi*np.sqrt((T**(-2)*(s+2/5*r**2/
 ↪s))**2*(T*ΔT_0_syst)**2+(T**(-1)*(1-2/
 ↪5*r**2*s**(-2)))**2*Δs_syst**2+(T**(-1)*(2/5*2*r/s))**2*Δr_syst**2) for T in␣
 ↪T_0_mess]
Δg_big_stat = [2*np.pi*np.sqrt((T**(-2)*(s+2/5*r**2/
 ↪s))**2*ΔT_stat**2+(T**(-1)*(1-2/5*r**2*s**(-2)))**2*Δs_stat**2+(T**(-1)*(2/
 ↪5*2*r/s))**2*Δr_syst**2) for T in T_0_mess]
Δg_big_full =[]
for i in range (0,len(g_big)):
    Δg_big_full += [np.sqrt(Δg_big_syst[i]**2+Δg_big_stat[i]**2)]
#Plotten der bestimmet Erdbeschleunigung aus kleinen Winkeln im Vergleich mit␣
 ↪den neuen Werten
plt.scatter( _0,g_big,label="g für große  mit Kleinwinkelnäherung")#Fehler
plt.errorbar( _0,g_big,yerr=Δg_big_full,xerr=[Δ _0_syst for i in  _0],ls="none")
x = np.linspace(np.min( _0),np.max( _0),100)
y = np.ones(100)*g
plt.plot(x,y,'r',label="Bestimmtes g")
plt.title("Vergleich bestimmtes g mit g aus Vernachlässigung großer Winkel")
plt.xlabel(" _0 in Rad")
plt.ylabel(r'Beschleunigung in $\frac{m}{s^2}$')
plt.legend()
plt.show()
```

```
%m Auch hier zeigt sich ein recht klein angenommener Fehler von g, der dazu␣
 ↪führt, dass die Werte nicht kompatibel mit den vorherigen sind.


%m Dies zeigt, dass für größere Auslenkungen eine andere Näherung benutzt␣
 ↪werden muss, um mit den anderen Messungen kompatibel zu bleiben, in diesem␣
 ↪Fall die Mittelwinkelnäherung.
#Fitten und Plotten der Mittelwinkelnäherung
%m <h5> Anwenden der Mittelwinkelnäherung </h5>
def sinsquare_model( _0,T=3):
    return T*(1+1/4*np.sin(1/2* _0)*np.sin(1/2* _0))

xy_data = kafe2.XYContainer(x_data= _0, y_data=T_0_mess)
xy_data.add_error(axis='x', err_val=Δ _0_syst)
xy_data.add_error(axis='y', err_val=ΔT_0_syst,relative=True)
sinsquare_fit = kafe2.Fit(data=xy_data, model_function=sinsquare_model)

sinsquare_fit.do_fit()
#sinsquare_fit.report(asymmetric_parameter_errors=True)
p = kafe2.Plot(sinsquare_fit)
p.x_label = ' _0 in Rad'
p.y_label = 'T_0 in s'
p.plot(asymmetric_parameter_errors=True)
p.show()
%m Auch in dieser zeigt sich allerdings bei großen Winkeln eine größere␣
 ↪Abweichung, sodass man zur Verwendung dieser Messpunkte in einer kompatiblen␣
 ↪Messung einen weiteren Term der Näherung nutzen müsste. Trotzdem zeigt sich,␣
 ↪dass der in der Näherung bestimmte Wert von g im Rahmen der Genauigkeit mit␣
 ↪der für bei kleinen Auslenkungen bestimmten Periodendauer kompatibel ist.
```

[4]: Bestimmung von g unter gültiger Kleinwinkelnäherung

[4]: Zunächst wurde das Pendel nur 5° ausegelenkt, um innerhalb der Kleinwinkelnäherung zu bleiben und es wurden die Dauern verschiedene Anzahlen an Perioden mit einer Lichtschranke gemessen. Der aus diesen Werten gebildeten Wert wurde aus folgendem Fit bestimmt, während die Unsicherheit als Unsicherheit einer Stichprobe und als rel. Unsicherheit auf den erhaltenen Wert bestimmt wurde.

[4]:



[4]: Er ergab sich eine Periodendauer $T = (3.1 \pm_{stat} 5.67 \cdot 10^{-5} \pm_{syst} 0.0061)s$.

[4]: Betrachtet wurden dabei als statistischer Fehler der statistische Fehler in Form der aus den Messungen errechneten Varianz und als systematischer Fehler der relative Messfehler der Lichtschranke von 0.2% auf den Mittelwert.

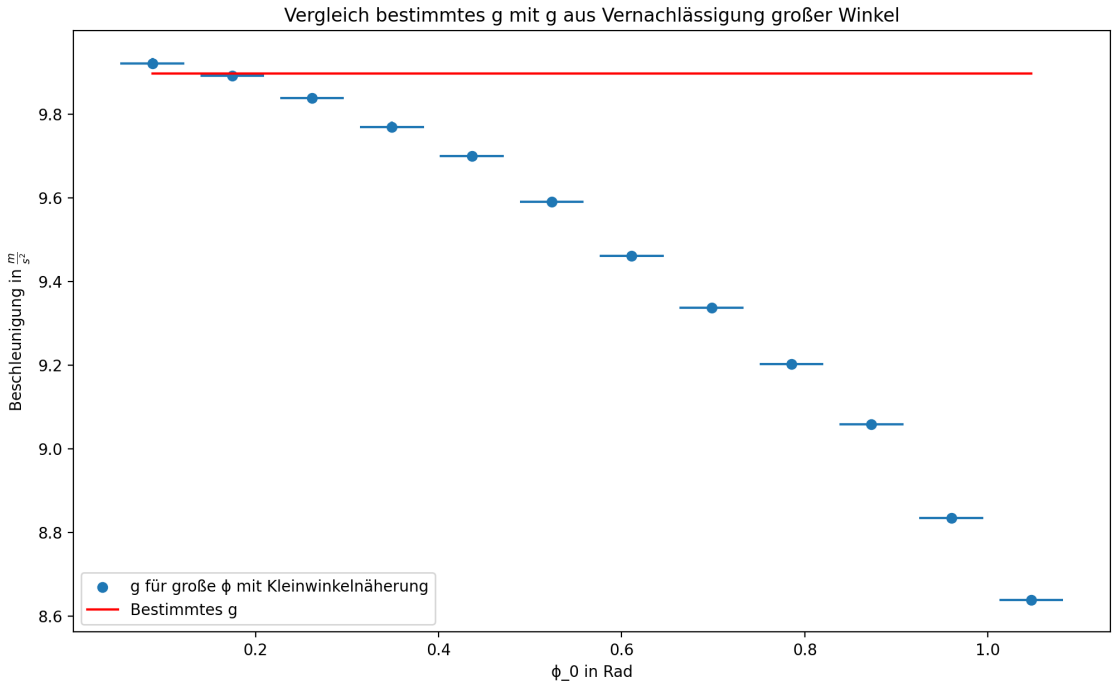
[4]: Mit dieser Periodendauer wurde in der Kleinwinkelnäherung ein Wert für die Erdbeschleunigung g berechnet, wobei die Fehler mit Gausscher Fehlerfortpflanzung bestimmt wurden. Hinzu kamen dabei der statistische Fehler in Form der Varianz auf mehrere Messungen des Radius der Kugel und damit auch die Schwerpunktsdistanz. Systematische Fehler wurden bei der Messung der Länge $\Delta L_{syst} = 0.003\,m$ und der Masse $\Delta m_{syst} = 0.0005\,kg$ in Form einer Angabe, auf den Radius der Kugel durch Annahme von $\Delta r_{syst} = 0.002\,m$ Genauigkeit des Messinstruments berücksichtigt.

[4]: Dabei ergab sich eine Erdbeschleunigung $g = (9.9 \pm_{stat} 0.002 \pm_{syst} 0.012)\frac{m}{s^2}$. Dieser Wert ist mit dem Literaturwert von $9,809599\frac{m}{s^2}$ (in Mannheim nach https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_1/1.1_masse/1.15/tabelle.pdf) im Rahmen der angenommenen Unsicherheiten nicht kompatibel. Dies liegt vermutlich vor allem an zu klein geschätzten systematischen Fehlern. Diese könnten entweder in veralteten Angaben oder an zu kleinen Ungenauigkeiten der Messinstrumente liegen.

[4]: Überprüfung der Gültigkeit der Kleinwinkelnäherung

[4]: Daraufhin wurde die Kleinwinkelnäherung für größere Winkel überprüft, indem das Pendel bis zu 60° Grad ausgelenkt und die Dauer von 5 Perioden mit selbiger Lichtschranke gemessen wurde. Die daraus berechneten Werte für g weichen gut erkennbar mit steigendem Winkel immer mehr vom in der Kleinwinkelnäherung bestimmten Wert ab. Gründe dafür sind zum einen, dass der Sinus für steigende Winkel immer weniger linear ist, aber auch andere Faktoren beeinträchtigen die Genauigkeit der Messungen bei höheren Winkel. Dazu zählen unter anderem, dass keine perfekte Spannung des Fadens mehr garantiert werden kann und die zunehmende Geschwindigkeit, die zu größeren Reibungseffekten führt.

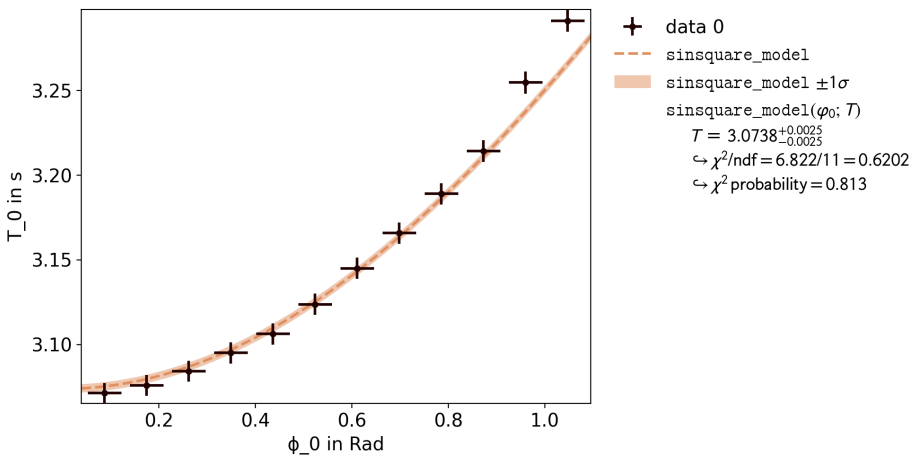
[4]:



[4]: Auch hier zeigt sich ein recht klein angenommener Fehler von g, der dazu führt, dass die Werte nicht kompatibel mit den vorherigen sind.

[4]: Dies zeigt, dass für größere Auslenkungen eine andere Näherung benutzt werden muss, um mit den anderen Messungen kompatibel zu bleiben, in diesem Fall die Mittelwinkelnäherung.

[4]: Anwenden der Mittelwinkelnäherung

[4]:


[4]: Auch in dieser zeigt sich allerdings bei großen Winkeln eine größere Abweichung, sodass man zur Verwendung dieser Messpunkte in einer kompatiblen Messung einen weiteren Term der Näherung

nutzen müsste. Trotzdem zeigt sich, dass der in der Näherung bestimmte Wert von g im Rahmen der Genauigkeit mit der für bei kleinen Auslenkungen bestimmten Periodendauer kompatibel ist.

2.2 Aufgabe 2: Reversionspendel

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

2.2.1 Aufgabe 2.1: Funktionsweise

Beschreiben Sie die Eigenschaften und Funktionsweise des Reversionspendles zur Bestimmung von g in eigenen Worten.

Das Reversionspendel ist ein langer Stab mit zwei Halterungen, auf denen der Stab kippbar in einer starren Halterung gelagert werden kann. Eine der Halterungen ist auf dem Stab verschiebbar. Wenn sich die Halterungen im korrekten Abstand zueinander befinden ist die Periodendauer der Schwingungen auf den jeweiligen Halterungen gleich. Die richtige Position kann leicht durch Vergleich der Periodendauern gefunden werden.

g kann dann einfach aus dem Abstand und der Periodendauer bestimmt werden ohne, dass das Trägheitsmoment bestimmt werden muss.

[5]:
```
L = 0.962
ΔL = 0.00002
  = 4940
Δ  = 0
D = 0.0015
ΔD = 0.00002

m_K = 0.083
Δm_K = 1E-3
m_K = 0.083
Δm_K = 1E-3

%m Ohne die verschiebbare Masse ergibt sich für das Pendel ein Trägheismoment␣
 ↪von $\Theta = \frac{m l^2}{3} = \frac{\pi L  d^2 l^2}{3} = [!
 ↪pi*L* *D**2*L**2/3:.4f!]\,\mathrm{Kg\,m^2}$

%m Der Schwerpunkt $s$ liegt bei $s = \frac{L}{2} = [!L/2:.4f!]\,\mathrm{m}$

%m Die reduzierte Länge $l_r$ liegt bei $s = \frac{2 L}{3} = [!2*L/3:.4f!
 ↪]\,\mathrm{m}$
```

[5]: Ohne die verschiebbare Masse ergibt sich für das Pendel ein Trägheismoment von $\Theta = \frac{ml^2}{3} = \frac{\pi L \rho d^2 l^2}{3} = 0.0104\,\mathrm{Kg\,m^2}$

[5]: Der Schwerpunkt s liegt bei $s = \frac{L}{2} = 0.4810\,\mathrm{m}$

[5]: Die reduzierte Länge l_r liegt bei $s = \frac{2L}{3} = 0.6413\,\mathrm{m}$

Die Halterungen um K und K' zu fixieren befinden sich am Aufhängungspunkt und bei der reduzierten Länge. Dementsprechend ist der Einfluss der Halterung bei K minimal, da sie nah am Rotationszentrum sitzt und so kaum zum Trägheitsmoment beiträgt. Der Einfluss der Halterung bei K' auf die Periodendauer ist auch vernachlässigbar, da es sich bei der reduzierten Länge befindet und nur eine sehr kleine Ausdehnung und damit vernachlässigbares eigenes Trägheitsmoment im Vergleich zum Trägheitsmoment des Pendels hat. Auch der Durchmesser des Stabes ist klein im Vergleich zur Länge des Pendels weswegen die Annäherung als dünner Stab gut ist.

2.2.2 Aufgabe 2.2: Bestimmung von g

Suchen Sie, indem Sie den beweglichen Keil K' des Reversionspendels geeignet verschieben, den Abstand d, der ℓ_r entspricht. Bestimmen Sie für diese Konfiguration des Pendels die folgenden Größen mit entsprechenden Unsicherheiten:

- Den Wert von ℓ_r und $\Delta\ell_r$;
- Die Periode T_0 und ΔT_0;
- Die Erbeschleunigung g und Δg.

Kalibrierung der Lichtschranke

[6]:
```
%m Im folgenden werden Korrekturdaten und statistische Unsicherheiten für die␣
 ↪Lichtschranke bestimmt. Es wird jeweils die Zeit bestimmt, die das Pendel␣
 ↪für 1, 2, 3, 4 und 5 Perioden braucht wobei jedes mal eine neue Messung␣
 ↪begonnen wird. Die Messwerte sind in folgender Tabelle zu sehen.

mpg = pd.read_csv('Periodendauern_calibrierug.csv')
mpg.rename(columns={'#': 'Anzahl Perioden','T_0': 'Periodendauer T_0 in s'},␣
 ↪inplace=True)
display(mpg)

%m Die Daten sollten einem linearen zusammenhang folgen, weswegen ein linearer␣
 ↪Fit durchgeführt wird.

nums, T_0s = np.genfromtxt('Periodendauern_calibrierug.csv', delimiter=',')[1:].
 ↪transpose()

def line_model(x, s, offset):
    return x*s+offset

(slope, offset), convM = scipy.optimize.curve_fit(line_model, nums, T_0s)
_, Δoffset = np.sqrt(np.diag(convM))
uoffset_stat = ufloat(offset, Δoffset)
uoffset_sys = ufloat(offset, abs(offset*2E-3))

plt.plot(nums, T_0s, ".")
lin = np.linspace(1.5,5.5,2)
plt.plot(lin, lin*slope+uoffset_sys.n)
```

```
plt.show()

ΔT_0 = np.std(T_0s - (nums*slope+uoffset_sys.n))

%m Es ergibt sich aus dem linearen Fit ein statistischer Fehler auf den␣
 ↪y-Achsenabschnitt von $[!uoffset_stat.s:.4f!]\,\mathrm{s}$. Weiter gibt es␣
 ↪einen Systematischen Fehler von 0.2%. Der y-Achsenabschnitt ist also $T_0 =␣
 ↪[!uoffset_stat.n:.3f!]\,\mathrm{s} \pm_{stat} [!uoffset_stat.s*1E3:.3f!
 ↪]\,\mathrm{ms} \pm_{sys} [!uoffset_sys.s*1E3:.3f!]\,\mathrm{ms}$ und wird im␣
 ↪folgenden (Gemeint ist hier der Rest der Aufgabe 2.2) zur Korrektur immer␣
 ↪von den Messerten abgezogen.

%m Aus der Standardabweichung der einzelnen Abständen von den Messwerten zur␣
 ↪Fitgeraden kann eine Unsicherheit statistische für die Zeitmessung angegeben␣
 ↪werden die $ΔT_0 = [!ΔT_0*1E3:.4f!]\,\mathrm{ms}$ beträgt.
```

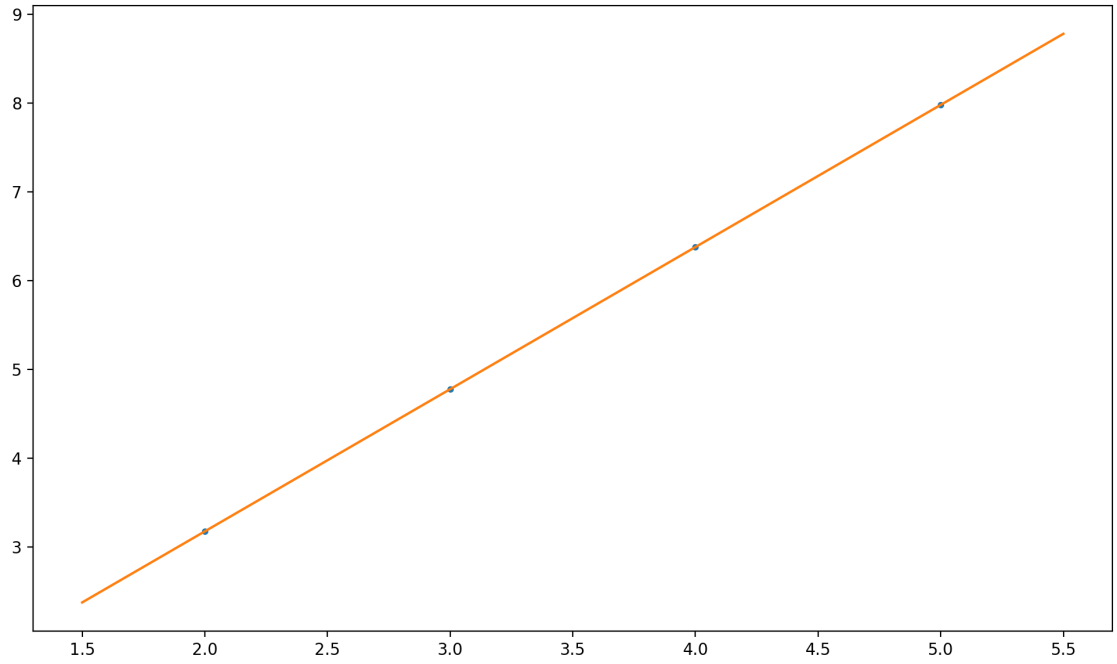
[6]: Im folgenden werden Korrekturdaten und statistische Unsicherheiten für die Lichtschranke bestimmt. Es wird jeweils die Zeit bestimmt, die das Pendel für 1, 2, 3, 4 und 5 Perioden braucht wobei jedes mal eine neue Messung begonnen wird. Die Messwerte sind in folgender Tabelle zu sehen.

[6]:

	Anzahl Perioden	Periodendauer T_0 in s
0	1	1.574
1	2	3.175
2	3	4.775
3	4	6.378
4	5	7.978

[6]: Die Daten sollten einem linearen zusammenhang folgen, weswegen ein linearer Fit durchgeführt wird.

[6]:



[6]: Es ergibt sich aus dem linearen Fit ein statistischer Fehler auf den y-Achsenabschnitt von 0.0016 s. Weiter gibt es einen Systematischen Fehler von 0.2%. Der y-Achsenabschnitt ist also $T_0 = -0.028\,\mathrm{s} \pm_{stat} 1.559\,\mathrm{ms} \pm_{sys} 0.055\,\mathrm{ms}$ und wird im folgenden (Gemeint ist hier der Rest der Aufgabe 2.2) zur Korrektur immer von den Messerten abgezogen.

[6]: Aus der Standardabweichung der einzelnen Abständen von den Messwerten zur Fitgeraden kann eine Unsicherheit statistische für die Zeitmessung angegeben werden die $\Delta T_0 = 0.6708\,\mathrm{ms}$ beträgt.

Messung

[7]:
```
%m In der Folgenden Tabelle sind unsere Messdaten zu finden. Es wurde jeweils␣
 ↪der Abstand zwischen den Schneiden der Klemmen, die Periodendauer für die␣
 ↪normale Schwingung und die Periodendauer für die inverse Schwingung an der␣
 ↪verschiebbaren Schneide gemessen. Es wurden jeweils 5 Perioden gemessen,␣
 ↪also 10 Durchschwünge der Lichtschranke und die Gesamtzeit in die Tabelle␣
 ↪eingetragen.

mpg = pd.read_csv('Periodendauern_revisionspendel.csv')
mpg.rename(columns={'d': 'l in m','T_0': 'T_0 in s', 'T_0reduziert': 'T_0\'  in␣
 ↪s'}, inplace=True)

display(mpg)

d, T_0, T_0reduziert = np.genfromtxt('Periodendauern_revisionspendel.csv',␣
 ↪delimiter=',')[1:].transpose()
T_0 /= 5
```

```
T_0reduziert /= 5

Δd_sys = 0.001
Δd_stat = 0.001

%m Für die Längenmessung wird eine systematische Unsicherheit von 1mm für die␣
 ↪Skala des Linals angesetzt und eine statistische Unsicherheit von 1mm beim␣
 ↪Ablesen des Wertes. Für die Zeitmessung wird die in der vorherigen Aufgabe␣
 ↪bestimmte Messunsicherheit verwendet.

%m Für die Zeitwerte wir die zuvor bestimmte statistische Unsicherheit␣
 ↪verwendet. Die Systematische Unsicherheit ist wie vorher 0.2%.

ud_sys = unumpy.uarray(d, Δd_sys)
ud_stat = unumpy.uarray(d, Δd_stat)

uT_0_sys = unumpy.uarray(T_0, T_0*2E-3) - uoffset_sys / 5
uT_0_stat = unumpy.uarray(T_0, ΔT_0) - uoffset_stat / 5
uT_0reduziert_sys = unumpy.uarray(T_0reduziert, T_0reduziert*2E-3/5) -␣
 ↪uoffset_sys/5
uT_0reduziert_stat = unumpy.uarray(T_0reduziert, ΔT_0/5) - uoffset_stat/5

plt.scatter(d,nomv(uT_0_stat), label="$K$", color="blue", )
#plt.errorbar(d,nomv(uT_0), color="blue", yerr=stdv(uT_0), capsize=3)
plt.scatter(d,nomv(uT_0reduziert_stat), label="$K'$", color="green")
#plt.errorbar(d,nomv(uT_0reduziert), color="green", yerr=stdv(uT_0reduziert),␣
 ↪capsize=3)
plt.plot(np.linspace(np.min(d), np.max(d)), np.linspace(np.min(d), np.max(d)) *␣
 ↪np.polyfit(d, nomv(uT_0_stat), 1)[0] + np.polyfit(d, nomv(uT_0_stat), 1)[1],␣
 ↪color="blue", linestyle="dotted")
plt.plot(np.linspace(np.min(d), np.max(d)), np.linspace(np.min(d), np.max(d)) *␣
 ↪np.polyfit(d, nomv(uT_0reduziert_stat), 1)[0] + np.polyfit(d,␣
 ↪nomv(uT_0reduziert_stat), 1)[1], color="green", linestyle="dotted")
plt.legend()

plt.ylabel("$T_0$ in s")
plt.xlabel("$d$ in m")
plt.show()

%m Wie zu sehen ist, eignet sich ein Linearer Fit nicht mehr, da die Kurve von␣
 ↪K' sichtbar gekrümmt ist. Deswegen wird im folgenden auf die Differenz der␣
 ↪beiden Werte eine Parabel gefittet.

uDeltaT_0_stat = uT_0_stat - uT_0reduziert_stat
uDeltaT_0_sys = uT_0_sys - uT_0reduziert_sys
```

```
def parabola_model(d,  =-7E-4,  =0.64,  =0.80):
    return  *( -d)*( -d)

%m Fit für statistische Unsicherheiten:
#Statistischer Unsicherheiten
xy_data = kafe2.XYContainer(x_data=d, y_data=nomv(uDeltaT_0_stat))
xy_data.add_error(axis='x', err_val=Δd_stat)
xy_data.add_error(axis='y', err_val=stdv(uDeltaT_0_stat))
lin_fit = kafe2.Fit(data=xy_data, model_function=parabola_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'd in m'
p.y_label = '$\Delta T_0 in m$'
p.plot()
p.show()

_, l_r, _ = lin_fit.parameter_values
_, Δl_r, _ = lin_fit.parameter_errors
ul_r_stat = ufloat(l_r, Δl_r)

%m Fit für systematische Unsicherheiten:
#Systematischer Unsicherheiten
xy_data = kafe2.XYContainer(x_data=d, y_data=nomv(uDeltaT_0_sys))
xy_data.add_error(axis='x', err_val=Δd_sys)
xy_data.add_error(axis='y', err_val=stdv(uDeltaT_0_sys))
lin_fit = kafe2.Fit(data=xy_data, model_function=parabola_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'd in m'
p.y_label = '$\Delta T_0 in m$'
p.plot()
p.show()

_, l_r, _ = lin_fit.parameter_values
_, Δl_r, _ = lin_fit.parameter_errors
ul_r_sys = ufloat(l_r, Δl_r)

%m Es wurde zur bestimmtung von der reduzierten Länge $l_r$ mit Kafe2 jeweils␣
 ↪ein Fit für statistische und systematische Unsicherheiten gemacht. Dabei␣
 ↪wurde das Modell $\alpha(\beta - l)(\gamma - l)$ gefittet, also eine␣
 ↪Linearfaktorzerlegung der Parabel mit Startparametern so gewählt, dass der␣
 ↪Parameter $\beta$ die Nullstelle der Kurve enthält. Kafe2 liefert dazu␣
 ↪gleich die Propagierten fehler für statistsche bzw. systematische Fehler.
```

```
%m Damit ergibt sich $l_r = [!ul_r_sys.n:.3f!]\,\mathrm{m} \pm_{stat} [!
 ↪ul_r_stat.s*1E3:.3f!]\,\mathrm{mm} \pm_{sys} [!ul_r_sys.s*1E3:.3f!
 ↪]\,\mathrm{mm}$ Die Eingangsfehler sind 1mm für statistsichen und␣
 ↪systematischen Fehler wie oben schon erwähnt. Daraus wurden mit der␣
 ↪Likelyhoodmethode die propagierten Fehler im Fit bestimmt. (Man kann sie␣
 ↪auch auf den beiden oberen Plots finden.)

%m Um jetzt noch den dazugehörigen Wert von T_0 mit Unsicherheit zu finden muss␣
 ↪ein linearer Fit in die ursprünglichen Messreihen von K gemacht werden und␣
 ↪damit dann den T_0 - Wert zu bestimmen (Da nur K' eine sichtbare Krümmung␣
 ↪auffweist und nicht K, reicht hier ein linearer Fit.)

%m Um eine Korrelation der Steigung und des y-Achsenabschnitts zu vermeiden␣
 ↪werden die Daten vor dem Fit um ihren Mittelwert verschoben, sodass beide␣
 ↪Parameter unkorreliert sind.

# Statistischer Unsicherheiten
ud_stat_shifted = ud_stat - np.average(nomv(ud_stat))
uT_0_stat_shifted = uT_0_stat - np.average(nomv(uT_0_stat))

%m Fit für statistische Unsicherheiten
%m Hier gehen die vorher festgelegten Unsicherheiten auf den Längenwerten und␣
 ↪auf der Periodendauer ein.
def lin_model(x, m, offset2):
    return x*m+offset2

xy_data = kafe2.XYContainer(x_data=nomv(ud_stat_shifted),␣
 ↪y_data=nomv(uT_0_stat_shifted))
xy_data.add_error(axis='x', err_val=Δd_stat)
xy_data.add_error(axis='y', err_val=stdv(uT_0_stat))
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'l in m'
p.y_label = 'T_0 in s'
p.plot()
p.show()

m, offset2 = lin_fit.parameter_values
Δm, Δoffset2 = lin_fit.parameter_errors

uT_0fin_stat = (ul_r_stat- np.average(nomv(ud_stat)))*ufloat(m,Δm) +␣
 ↪ufloat(offset2, Δoffset2) + np.average(nomv(uT_0_stat))

# Systematische Unsicherheiten
```

```
ud_sys_shifted = ud_stat - np.average(nomv(ud_stat))
uT_0_sys_shifted = uT_0_stat - np.average(nomv(uT_0_stat))

%m Fit für systematische Unsicherheiten
%m Hier gehen die vorher festgelegten Unsicherheiten auf den Längenwerten und␣
 ↪auf der Periodendauer ein.
def lin_model(x, m, offset2):
    return x*m+offset2

xy_data = kafe2.XYContainer(x_data=nomv(ud_sys_shifted),␣
 ↪y_data=nomv(uT_0_sys_shifted))
xy_data.add_error(axis='x', err_val=Δd_sys)
xy_data.add_error(axis='y', err_val=stdv(uT_0_sys))
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'l in m'
p.y_label = 'T_0 in s'
p.plot()
p.show()

m, offset2 = lin_fit.parameter_values
Δm, Δoffset2 = lin_fit.parameter_errors

uT_0fin_sys = (ul_r_sys- np.average(nomv(ud_sys)))*ufloat(m,Δm) +␣
 ↪ufloat(offset2, Δoffset2) + np.average(nomv(uT_0_sys))

%m Damit ergibt sich $T_0 = [!uT_0fin_sys.n:.3f!]\,\mathrm{s} \pm_{stat} [!
 ↪uT_0fin_stat.s*1E3:.3f!]\,\mathrm{ms} \pm_{sys} [!uT_0fin_sys.s*1E3:.3f!
 ↪]\,\mathrm{ms}$ direkt aus dem Kafe2 fit mit der Likelyhood-Methode wie im␣
 ↪Plot abzulesen ist.

ug_sys = 4*pi**2*ul_r_sys/uT_0fin_sys**2
ug_stat = 4*pi**2*ul_r_stat/uT_0fin_stat**2

%m Damit ergibt sich $g = [!ug_stat.n:.3f!]\,\mathrm{\frac{m}{s^2}} \pm_{stat}␣
 ↪[!ug_stat.s*1E3:.3f!]\,\mathrm{\frac{mm}{s^2}} \pm_{sys} [!ug_sys.s*1E3:.3f!
 ↪]\,\mathrm{\frac{mm}{s^2}}$ nach gausscher Messunsicherheitsfortpflanzung␣
 ↪aus $T_0$ und $l_r$ mit den oben bestimmten Unsicherheiten.
```

```
%m Dieser Wert stimmt nicht innerhalb der Messunsicherheit mit dem␣
↪Literaturwert überein. Das liegt daran, dass die Messung von l zwischen den␣
↪beiden Schneiden vorgenommen wurde und die Schneiden natürlich nicht mit dem␣
↪Schweerpunkt der Klemmen fluchten. Wenn man die halbe dicke der Klemmen auf␣
↪die Abstandsmesswerte addiert kommt man auf Werte wesentlich näher am␣
↪Literaturwert. Also ist die Ausdehnung der Klemmen hier nicht mehr␣
↪vernachlässigbar und müsste genauer berücksichtigt werden.
```

[7]: In der Folgenden Tabelle sind unsere Messdaten zu finden. Es wurde jeweils der Abstand zwischen den Schneiden der Klemmen, die Periodendauer für die normale Schwingung und die Periodendauer für die inverse Schwingung an der verschiebbaren Schneide gemessen. Es wurden jeweils 5 Perioden gemessen, also 10 Durchschwünge der Lichtschranke und die Gesamtzeit in die Tabelle eingetragen.

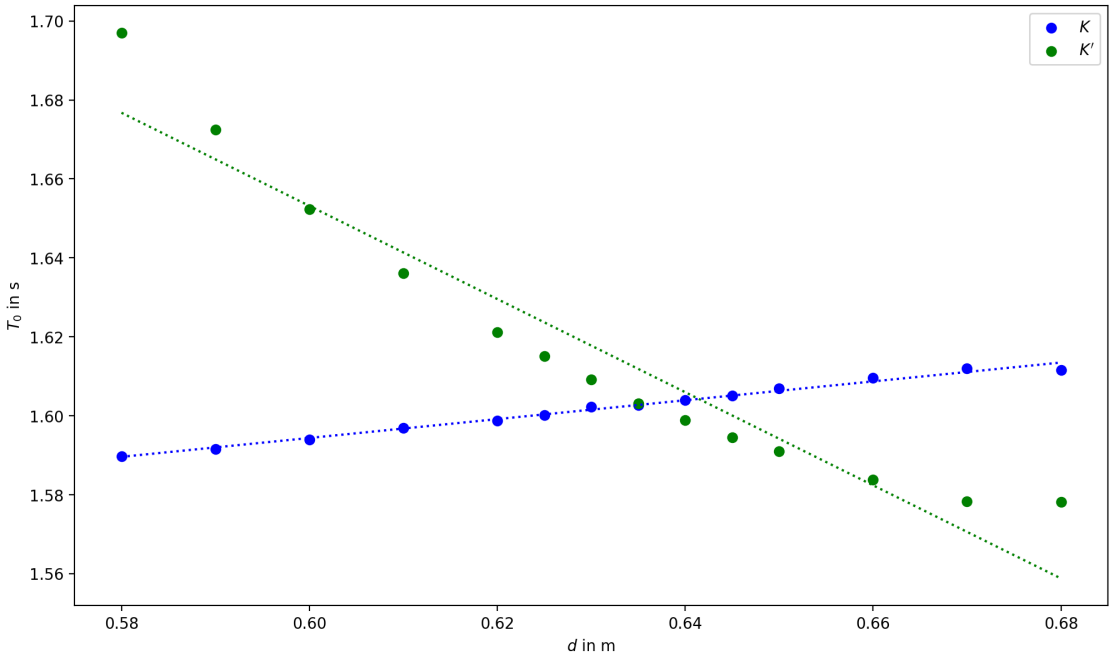
[7]:

	l in m	T_0 in s	T_0' in s
0	0.680	8.030	7.863
1	0.670	8.032	7.864
2	0.660	8.020	7.891
3	0.650	8.007	7.927
4	0.645	7.998	7.945
5	0.640	7.992	7.967
6	0.635	7.986	7.988
7	0.630	7.984	8.018
8	0.625	7.973	8.048
9	0.620	7.966	8.078
10	0.610	7.957	8.153
11	0.600	7.942	8.234
12	0.590	7.930	8.335
13	0.580	7.921	8.457

[7]: Für die Längenmessung wird eine systematische Unsicherheit von 1mm für die Skala des Linals angesetzt und eine statistische Unsicherheit von 1mm beim Ablesen des Wertes. Für die Zeitmessung wird die in der vorherigen Aufgabe bestimmte Messunsicherheit verwendet.

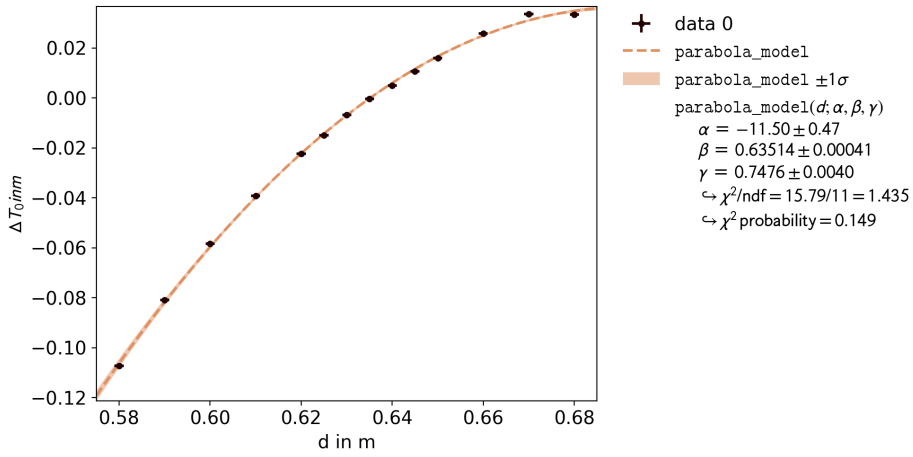
[7]: Für die Zeitwerte wir die zuvor bestimmte statistische Unsicherheit verwendet. Die Systematische Unsicherheit ist wie vorher 0.2%.

[7]:



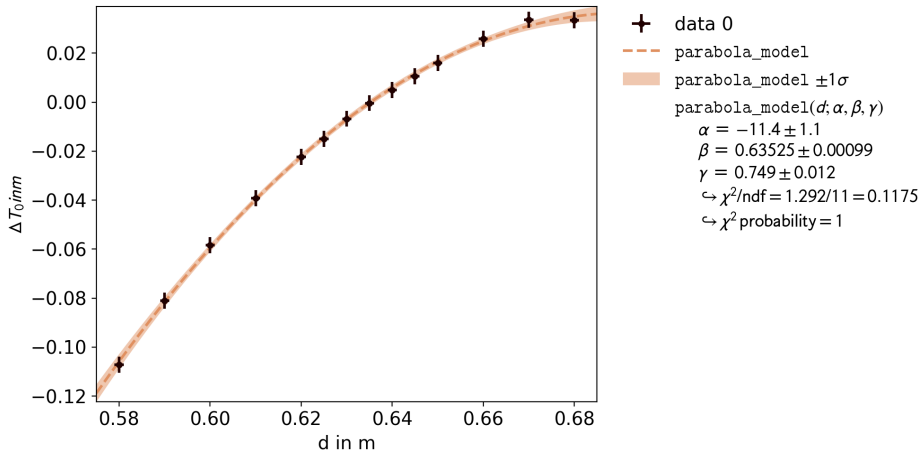
[7]: Wie zu sehen ist, eignet sich ein Linearer Fit nicht mehr, da die Kurve von K' sichtbar gekrümmt ist. Deswegen wird im folgenden auf die Differenz der beiden Werte eine Parabel gefittet.

[7]: Fit für statistische Unsicherheiten:

[7]:


[7]: Fit für systematische Unsicherheiten:

[7]:



[7]: Es wurde zur bestimmtung von der reduzierten Länge l_r mit Kafe2 jeweils ein Fit für statistische und systematische Unsicherheiten gemacht. Dabei wurde das Modell $\alpha(\beta - l)(\gamma - l)$ gefittet, also eine Linearfaktorzerlegung der Parabel mit Startparametern so gewählt, dass der Parameter β die Nullstelle der Kurve enthält. Kafe2 liefert dazu gleich die Propagierten fehler für statistsche bzw. systematische Fehler.

[7]: Damit ergibt sich $l_r = 0.635\,\mathrm{m} \pm_{stat} 0.407\,\mathrm{mm} \pm_{sys} 0.988\,\mathrm{mm}$ Die Eingangsfehler sind 1mm für statistsichen und systematischen Fehler wie oben schon erwähnt. Daraus wurden mit der Likelyhoodmethode die propagierten Fehler im Fit bestimmt. (Man kann sie auch auf den beiden oberen Plots finden.)

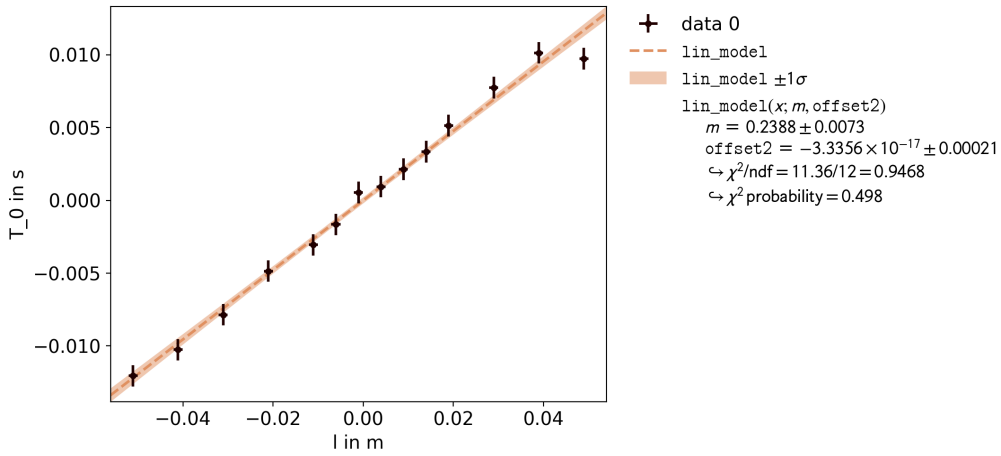
[7]: Um jetzt noch den dazugehörigen Wert von T_0 mit Unsicherheit zu finden muss ein linearer Fit in die ursprünglichen Messreihen von K gemacht werden und damit dann den T_0 - Wert zu bestimmen (Da nur K' eine sichtbare Krümmung auffweist und nicht K, reicht hier ein linearer Fit.)

[7]: Um eine Korrelation der Steigung und des y-Achsenabschnitts zu vermeiden werden die Daten vor dem Fit um ihren Mittelwert verschoben, sodass beide Parameter unkorreliert sind.

[7]: Fit für statistische Unsicherheiten

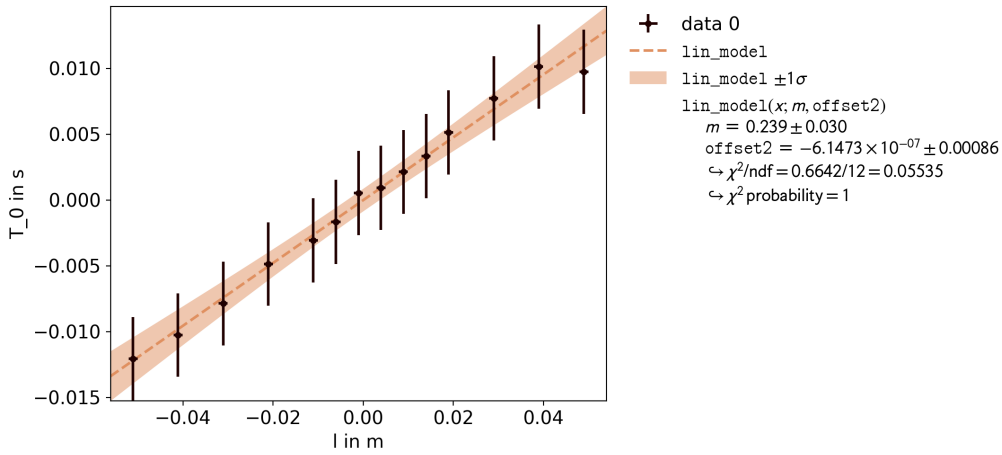
[7]: Hier gehen die vorher festgelegten Unsicherheiten auf den Längenwerten und auf der Periodendauer ein.

[7]:



[7]: Fit für systematische Unsicherheiten

[7]: Hier gehen die vorher festgelegten Unsicherheiten auf den Längenwerten und auf der Periodendauer ein.

[7]:


[7]: Damit ergibt sich $T_0 = 1.603\,\mathrm{s} \pm_{stat} 0.231\,\mathrm{ms} \pm_{sys} 0.896\,\mathrm{ms}$ direkt aus dem Kafe2 fit mit der Likelyhood-Methode wie im Plot abzulesen ist.

[7]: Damit ergibt sich $g = 9.761\,\frac{\mathrm{m}}{\mathrm{s}^2} \pm_{stat} 5.678\,\frac{\mathrm{mm}}{\mathrm{s}^2} \pm_{sys} 16.208\,\frac{\mathrm{mm}}{\mathrm{s}^2}$ nach gausscher Messunsicherheitsfortpflanzung aus T_0 und l_r mit den oben bestimmten Unsicherheiten.

[7]: Dieser Wert stimmt nicht innerhalb der Messunsicherheit mit dem Literaturwert überein. Das liegt daran, dass die Messung von l zwischen den beiden Schneiden vorgenommen wurde und die Schneiden natürlich nicht mit dem Schweerpunkt der Klemmen fluchten. Wenn man die halbe dicke der Klemmen auf die Abstandsmesswerte addiert kommt man auf Werte wesentlich näher am Literaturwert. Also ist die Ausdehnung der Klemmen hier nicht mehr vernachlässigbar und müsste genauer berücksichtigt werden.

2.3 Aufgabe 3: Gekoppelte Pendel

Hinweise zu allen hier durchzuführenden Messungen finden Sie in der Datei Hinweise-Aufgabe-3.md.

2.3.1 Aufgabe 3.1: Justierung der einzelnen Pendel

Stellen Sie bei den zwei baugleichen Pendeln durch Verschieben einer der Pendelscheiben gleiche Schwingungsdauern T_0 ein.

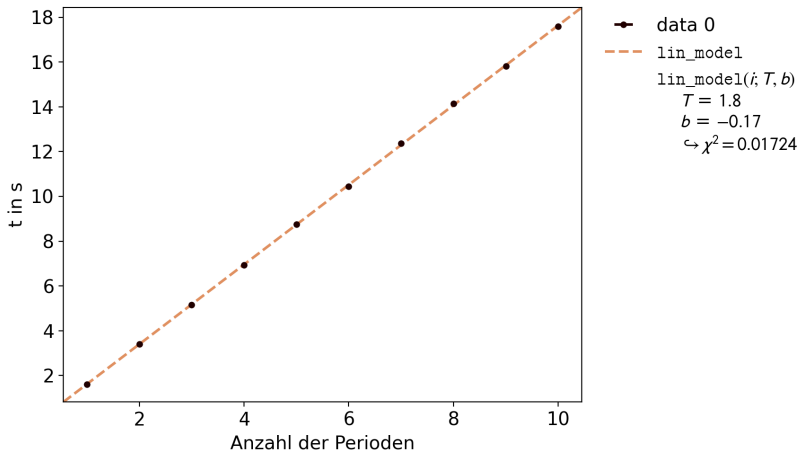
---

[8]:
```
L = ufloat(106.75E-2,2.5E-2) #Gesamtlänge Stab
#Das schwerere, linke Pendel sei im folgenden Pendel L und das rechte,␣
 ↪leichtere, Pendel R.
L_L = ufloat(81.5E-2,2E-3)#zu messender Abstand der Pendelscheibe von der␣
 ↪Aufhängung
L_R = ufloat(81.2E-2,2E-3)#zu messender Abstand der Pendelscheibe von der␣
 ↪Aufhängung
T_0L = [1.61,3.40/2,5.16/3,6.94/4,8.75/5,10.45/6,12.37/7,14.14/8,15.82/9,17.6/
 ↪10]#1.779
T_0L = [1.61,3.40,5.16,6.94,8.75,10.45,12.37,14.14,15.82,17.6]#1.779
T_0R = [1.65,2.31/2,4.94/3,6.92/4,8.72/5,10.38/6,12.17/7,13.92/8,15.71/9,17.57/
 ↪10]#1.775
T_0R = [1.65,2.31,4.94,6.92,8.72,10.38,12.17,13.92,15.71,17.57]#1.775
T_0 = ufloat(1/2*(np.mean(T_0R)+np.mean(T_0L)),1.5)#zu messende, gleiche␣
 ↪Schwingdauer
#Wir nehmen das Trägheitsmoment des linken Pendels
d = ufloat(17E-3,2E-3)
b = ufloat(7E-3,2E-3)
A = d*b#zu messender Querschnitt der Aufängung
  = 7.44E3 #7,44g/cm^3
m_k = ufloat(1.4,0.005)
m_s = ufloat(1.4,0.005)
r = ufloat(50E-3,2E-3) #Radius der Scheiben
m = m_k + m_s #alles dazu

%m Es wurde die Schwingdauer beider Pendel alleine, ohne Kopplung, über mehrere␣
 ↪Perioden bestimmt.
#Einfügen Fits Einzelschwing. bzw Werte
i = np.arange(1,len(T_0L)+1)
def lin_model(i,T=3,b=.3):
    return i*T+b

xy_data = kafe2.XYContainer(x_data=i, y_data=T_0L)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)
lin_fit.limit_parameter("b", lower=-0.3, upper=1.)
```

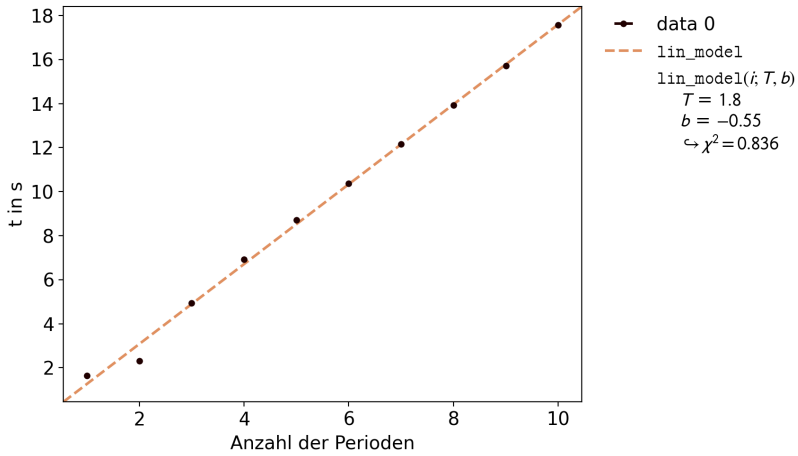
```
lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = 't in s'
p.plot(asymmetric_parameter_errors= True)
p.show()
T_0L = lin_fit.parameter_values[0]

xy_data = kafe2.XYContainer(x_data=i, y_data=T_0R)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = 't in s'
p.plot()
p.show()
T_0R = lin_fit.parameter_values[0]
T_0 = 1/2*(T_0L + T_0R)
%m Es ergaben sich aus den Fits Schwingdauern von $T_{0L} = [!T_0L:.4!]\,s$ und␣
 ↪$T_{0R} = [!T_0R:.4!]\,s$.
%m Die Werte liegen im Rahmen der Messunsicherheiten sehr nah beeinander,␣
 ↪sodass im weiteren Verlauf ihr Mittelwert $T=[!T_0:.4!]\,s$ als ungekoppelte␣
 ↪Schwingdauer beider Pendel behandelt wird.
```

[8]: Es wurde die Schwingdauer beider Pendel alleine, ohne Kopplung, über mehrere Perioden bestimmt.

[8]:



[8]:



[8]: Es ergaben sich aus den Fits Schwingdauern von $T_{0L} = 1.78\,s$ und $T_{0R} = 1.814\,s$.

[8]: Die Werte liegen im Rahmen der Messunsicherheiten sehr nah beeinander, sodass im weiteren Verlauf ihr Mittelwert $T = 1.797\,s$ als ungekoppelte Schwingdauer beider Pendel behandelt wird.

2.3.2 Aufgabe 3.2: Gekoppelte Pendel

Koppeln Sie die Pendel mittels einer Schraubenfeder (mit der Federkonstanten D) in jeweils gleichem Abstand ℓ von den Aufhängepunkten der Pendel. Hierbei, wie im Folgenden, sollte die koppelnde Feder die Pendel nur mit schwacher Spannung verbinden. Führen Sie dann das folgende Messprogramm durch:

- Bestimmen Sie die Perioden T_1 und T_2 der beiden Fundamentalschwingungen, bei denen keine Schwebung auftritt.
- Wiederholen Sie die Messungen bei verändertem Abstand ℓ'.
- Diskutieren Sie Ihre Ergebnisse für T_0, $T_1(\ell)$, $T_2(\ell)$, $T_1(\ell')$ und $T_2(\ell')$.
- Bestimmen Sie für ℓ und ℓ' aus $T_{1/2}$ die Werte von $\omega_{1/2}$.
- Berechnen Sie auf der Grundlage des Modells eines physikalischen Pendels aus ω_1 das Trägheitsmoment Θ und vergleichen Sie das Ergebnis mit Ihrer Erwartung aus den geometrischen Abmessungen und Massen der Pendel.
- Berechnen Sie aus ω_2 das Direktionsmoment D der koppelnden Feder. Führen Sie zwei alternative Bestimmung von D durch und vergleichen Sie Ihre Ergebnisse:
  - Mit Hilfe des Hook'schen Gesetzes;
  - Unter Verwendung der koppelnden Feder als Federpendel.

---

[14]:
```
%%m <h5> Messung der Periodendauern der Eigenschwingungen</h5>
#Messung der beiden Periodenlängen durch Mitteln über mehrere
l_0 = 41.5E-2
Δl_0= 5E-3
```

```
%m Die beiden Pendel wurden nun in einem Abstand $l = [!l_0:.3!]\,m$ mit einer␣
 ↪Feder gekoppelt und es wurden die Schwingdauer $T_{10}$ der gleichphasigen␣
 ↪Eigenschwingung ohne Einfluss der Feder und $T_{20}$ der gegenphasigen␣
 ↪Schwingung gemessen.
#Messung der Schwingung,bei der beide Pendel in Phase schwingen
T_10 = [ufloat(5.22/3,1),ufloat(8.77/5,1),ufloat(12.21/7,1),ufloat(15.96/9,1)]
T_10 = [5.22,8.77,12.21,15.96]
i = np.arange(3,9+1,2)
def lin_model(i,T=3,b=.3):
    return i*T+b

xy_data = kafe2.XYContainer(x_data=i, y_data=T_10)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$T_{10}$ in s'
p.plot()
p.show()
T_10 = lin_fit.parameter_values[0]

#Messung der Schwingung, bei der beide Pendel gegenphasig schwingen
T_20 = [ufloat(4.33/3,1),ufloat(6.87/5,1),ufloat(9.86/7,1),ufloat(12.78/9,1)]
T_20 = [4.33,6.87,9.86,12.78]
xy_data = kafe2.XYContainer(x_data=i, y_data=T_20)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$T_{20}$ in s'
p.plot()
p.show()
T_20 = lin_fit.parameter_values[0]

%m Dabei ergaben sich Werte von $T_{10} = [!T_10:.3!]\,s$ und $T_{20} = [!T_20:.
 ↪3!]\,s$. Man erwartet wegen der bei der gegenphasigen Schwingung␣
 ↪hinzukommenden Kopplung $\omega_1^2 < \omega_2^2$, dann aber wegen $\omega =␣
 ↪\frac{2\pi}{T}$ folglich $T_{10} > T_{20}$, die gemessenen Werte entsprechen␣
 ↪also in ihrem Verhältnis der Erwartung.

#Wiederholen der vorherigen Messungen mit verändertem l (Abstand der Kopplung)
l_1 = 30E-2
Δl_1 =5E-3
```

```
%m Anschließend wurde eine Kopplungshöhe $l_1 = [!l_1:.2!]\,m$ eingestellt und␣
 ↪es wurden die Schwingdauer $T_{11}$ der gleichphasigen Eigenschwingung ohne␣
 ↪Einfluss der Feder und $T_{21}$ der gegenphasigen Schwingung gemessen.

#Messung der Schwingung,bei der beide Pendel in Phase schwingen
T_11 = [ufloat(5.24/3,1),ufloat(8.73/5,1),ufloat(12.52/7,1),ufloat(15.56/9,1)]
T_11 = [5.24,8.73,12.52,15.56]
xy_data = kafe2.XYContainer(x_data=i, y_data=T_11)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$T_{11}$ in s'
p.plot()
p.show()
T_11 = lin_fit.parameter_values[0]


#Messung der Schwingung, bei der beide Pendel gegenphasig schwingen
T_21 = [ufloat(4.57/3,1),ufloat(7.7/5,1),ufloat(10.83/7,1),ufloat(13.72/9,1)]
T_21 = [4.57,7.7,10.83,13.72]
xy_data = kafe2.XYContainer(x_data=i, y_data=T_21)
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$T_{21}$'
p.plot()
p.show()
T_21 = lin_fit.parameter_values[0]

%m Dabei ergaben sich Werte von $T_{11} = [!T_11:.2!]\,s$ und $T_{21} = [!T_21:.
 ↪2!]\,s$. Man erwartet wegen der bei der gegenphasigen Schwingung␣
 ↪hinzukommenden Kopplung $\omega_1^2 < \omega_2^2$, dann aber wegen $\omega =␣
 ↪\frac{2\pi}{T}$ folglich $T_{11} > T_{21}$, die gemessenen Werte entsprechen␣
 ↪also in ihrem Verhältnis der Erwartung.
table = pd.DataFrame({
    "l [m]": [l_0, l_1],
    r"$T_{1i}$ [s] (gleichphasig)": [T_10, T_11],
    r"$T_{2i}$ [s] (gegenphasig)": [T_20, T_21],
})
display(Markdown(table.to_markdown()))
```

```
%m Im Rahmen der Messungenauigkeiten sind die Werte sehr kompatibel␣
 ↪miteinander, da die Verhältnisse gleich bleiben ($T_1 > T_2$). Die␣
 ↪Unterschiede der einzelnen Periodendauern sind im Rahmen der Genauigkeit␣
 ↪primär auf Unsicherheiten zurückzuführen, sodass ein Einfluss der␣
 ↪Kopplungshöhe nicht zu vermuten ist, da sonst stärkere Unterschiede zu␣
 ↪erwarten wären.

#w_i = 2*pi/T_i
#Δw = 2*pi*np.sqrt(T**(-2)*ΔT**2)
_10 = 2*np.pi/T_10
#Δ_10 = 2*np.pi*np.sqrt(T_10**(-2)*ΔT_10**2)
_11 = 2*np.pi/T_11
#Δ_11 = 2*np.pi*np.sqrt(T_11**(-2)*ΔT_11**2)
_20 = 2*np.pi/T_20
#Δ_20 = 2*np.pi*np.sqrt(T_20**(-2)*ΔT_20**2)
_21 = 2*np.pi/T_21
#Δ_21 = 2*np.pi*np.sqrt(T_21**(-2)*ΔT_21**2)
#Mitteln der Werte
_1  = 1/2 * ( _10+  _11)
_2  = 1/2 * ( _20+  _21)

%m Aus diesen Werten folgen Werte für  :
df = pd.DataFrame({
    "l [m]": [l_0, l_1],
    r"$\omega_{1i}$ [s] (gleichphasig)": [ _10,  _11],
    r"$\omega_{2i}$ [s] (gegenphasig)": [ _20,  _21]
})
'''
df = pd.DataFrame({
    "l [m]": [l_0, l_1],
    r"$\omega_{1i}$ [s]": [ _10,  _11],
    r"$\omega_{2i}$ [s]": [ _20,  _21],
    r"$\omega_{i+1}$ [s]": [ _1, _2]
})'''
display(Markdown(df.to_markdown()))
%m        Mittelwert: $ _1 = [! _1:.2!]\,s$␣
 ↪(gleichphasig) und $ _2 = [! _2:.2!]\,s$ (gegenphasig)
%m Diese Werte sind aus der Diskussion über die entsprechenden Periodendauern␣
 ↪folgend plausibel und den Erwartungen entsprechend.
%m <h5>Berechnung von $\Theta$</h5>
#Berechnung Θ aus w1 durch Modell als physikalisches Pendel
g = 9.809599#Literaturwert  einfügen
s = (m_k*l_0 + m_s * L_L +  /2*L**2*A)/(m_k + m_s +  *L_L*A)
Θ_Modell = m*g*s/ _1**2
%m Aus $ _1$ und den Maßen des Pendels lässt sich $Θ_{Modell}= [!nomv(Θ_Modell):
 ↪.2f!] \,kg\,m^2$ bestimmen.
#Berechnung Θ aus Werten von Pendeln etc.
```

```
Θ_Stab = 1/3 *  *A*L**3
Θ_s = 1/2 * m_s * r**2
Θ_Theorie = Θ_Stab + Θ_s + m_s * L_L**2+m_k *l_0**2
%m Es lässt sich aus den Abmessungen und Gewichten das Trägheitsmoment␣
 ↪theoretisch zu $Θ_{Theorie}= [!nomv(Θ_Theorie):.2f!] \,kg\,m^2$ bestimmen.
%m Diese Werte weichen durchaus voneinander ab, was im Rahmen der Zeitmessung␣
 ↪allerdings wenig überraschend ist. In der weiteren Rechnung wird der␣
 ↪Mittelwert der beiden Werte verwendet.
%m
%m Im folgenden soll das Direktionsmoment D, welches die Kopplung der beiden␣
 ↪Pendel beschreibt, durch verschiedene Methoden bestimmt werden:
%m Zunächst durch Verwendung des Trägheitsmoments und der beiden␣
 ↪Eigenfrequenzen.
%m <h5>Berechnung von $D_{\omega_2}$</h5>
#Berechnung D aus w2
D_ _2 = np.mean([Θ_Modell,Θ_Theorie])*( _2**2- _1**2)
%m Es ergibt sich das Direktionsmoment $D_{ _2} = Θ*( _2^2- _1^2) = [!nomv(D_ _2):
 ↪.2!]\,\frac{kg\,m^2}{s^2}$.

#Messung von D
%m <h5>Berechnung von $D_{hook}$</h5>
%m Nun soll es durch Anhängen verschiedener Massen und Messen der Auslenkungen␣
 ↪unter Annahme des linearen Bereichs des Hookschen Gesetzes bestimmt werden.
l_hook = 62.6E-2 -17.6E-2#Länge Feder messen
Δl_hook = 2E-3
#%m Bestimmung nach dem Hookschen Gesetz nach $\begin{equation*}\mathrm{d}F =␣
 ↪k\,\mathrm{d}z.\end{equation*}$
m_i = [0.5,0.2,0.1] #angehängte Gewichte
Δm_i = 0.005 #anpassen
dz_i = [75.6E-2,67.8E-2,65.3E-2] #Auslenkung aus Ruhelage
Δdz_i =1E-3 #Ableseganauigkeit
def line_model(m_i,k=1,b=.3):
    return m_i*g/k+b
xy_data = kafe2.XYContainer(x_data=m_i, y_data=dz_i)
xy_data.add_error(axis='x', err_val=Δm_i)
xy_data.add_error(axis='y', err_val=Δdz_i)
line_fit = kafe2.Fit(data=xy_data, model_function=line_model)

line_fit.do_fit()
p = kafe2.Plot(line_fit)
p.x_label = r'$m_i$ in kg'
p.y_label = r'$dz_i$ in m'
p.plot()
p.show()

k = line_fit.parameter_values[0]
```

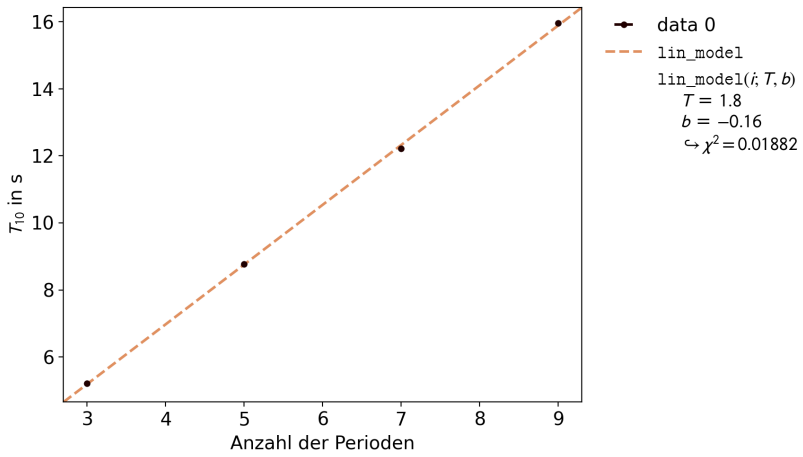
```
Δk = line_fit.parameter_errors[0]

D_hook = k*l_0**2
ΔD_hook = np.sqrt(l_0**4*Δk**2+(2*k*l_0)**2*Δl_0**2)
%m <h5>Berechnung von $D_{Pendel}$</h5>
%m Bei der Bestimmung über die Periodendauer des Federpendels wird die␣
 ↪benötigte Zeit einer angehängten Masse für mehrere Schwingungen gemessen, um␣
 ↪die Periodendauer und daraus über den Ansatz eines harmonischen Oszillators␣
 ↪$D$ zu bestimmen.
m_i = [0.2,0.2,0.2] #Angehängte Massen
m_i = 0.2
Δm_i = 0.005
T_0D = [4.67,9.5,14.27]#gemessene Periodendauern
times = [10,20,30]
ΔT_0D = 0.5
def square_model(i,k=0.3):
    return i*np.sqrt(4*np.pi*np.pi*m_i*k**(-1))
xy_data = kafe2.XYContainer(x_data=times, y_data=T_0D)
xy_data.add_error(axis='x', err_val=Δm_i)
xy_data.add_error(axis='y', err_val=ΔT_0D)
square_fit = kafe2.Fit(data=xy_data, model_function=square_model)

square_fit.do_fit()
p = kafe2.Plot(square_fit)
p.x_label = r'$m_i$ in kg'
p.y_label = r'$T_0$ in s'
p.plot()
p.show()

k = square_fit.parameter_values[0]
Δk = square_fit.parameter_errors[0]
%m <h5>Vergleich der D Werte</h5>
D_pendulum = k*l_0**2
ΔD_pendulum = np.sqrt(l_0**4*Δk**2+(2*k*l_0)**2*Δl_0**2)
%m $D_{ _2}= [!nomv(D_ _2):.2f!] \,\frac{kg\,m^2}{s^2}$
#%m $\Delta D_{ _2} = [!stdv(D_ _2):.2f!] \,\mathrm{Nm}$
%m $D_{Hooksch}= [!D_hook:.2f!]\,\frac{kg\,m^2}{s^2}$
#%m $\Delta D_{Hooksch} = [!ΔD_hook:.2f!] \,\mathrm{Nm}$
%m $D_{Pendel}= [!D_pendulum:.2f!] \,\frac{kg\,m^2}{s^2}$
#%m $\Delta D_{Pendel} = [!ΔD_pendulum:.2f!] \,\mathrm{Nm}$
```

```
%m Da $D_{ _2}$ von den meisten gemessenen Werten abhängt und im Vergleich␣
↪stärker von den anderen beiden Werten abweicht, als diese untereinander, ist␣
↪davon auszugehen, dass die anderen beiden Messverfahren genauer sind. Dies␣
↪könnte konkret daran liegen, dass die Feder bei der Kopplung durchhing und␣
↪konstant gespannt war, während sie bei den anderen beiden Experimenten mehr␣
↪der Theorie entsprechend gerade hing und in Ruhe eher entspannt war.␣
↪Relevant zur Bewertung wären allerdings auch die hier nicht bestimmten␣
↪Fehler, um einzuschätzen, wie kompatibel die Messungen sind.
```

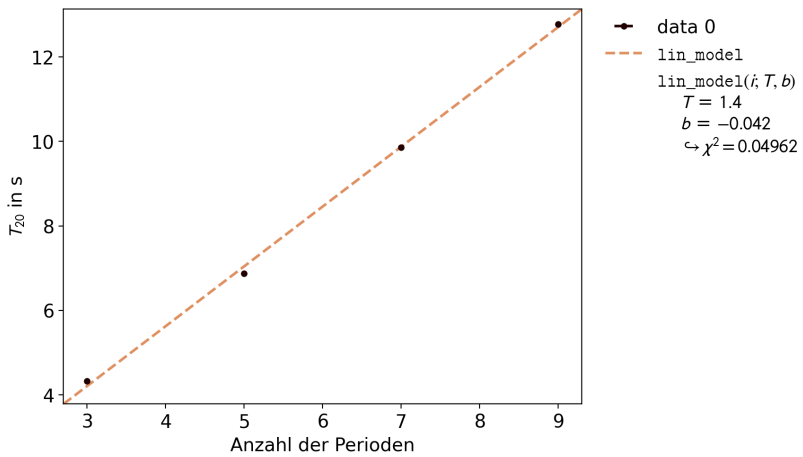
[14]: Messung der Periodendauern der Eigenschwingungen

[14]: Die beiden Pendel wurden nun in einem Abstand $l = 0.415\,m$ mit einer Feder gekoppelt und es wurden die Schwingdauer T_{10} der gleichphasigen Eigenschwingung ohne Einfluss der Feder und T_{20} der gegenphasigen Schwingung gemessen.

[14]:



[14]:



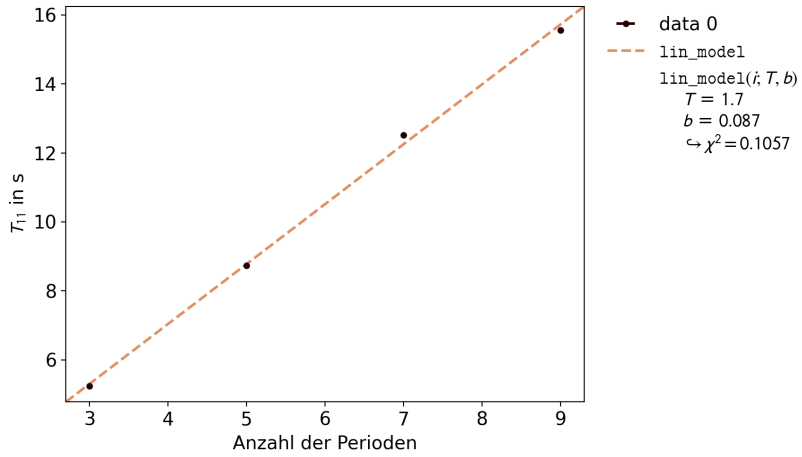
[14]: Dabei ergaben sich Werte von $T_{10} = 1.78\,s$ und $T_{20} = 1.42\,s$. Man erwartet wegen der bei der

gegenphasigen Schwingung hinzukommenden Kopplung $\omega_1^2 < \omega_2^2$, dann aber wegen $\omega = \frac{2\pi}{T}$ folglich $T_{10} > T_{20}$, die gemessenen Werte entsprechen also in ihrem Verhältnis der Erwartung.

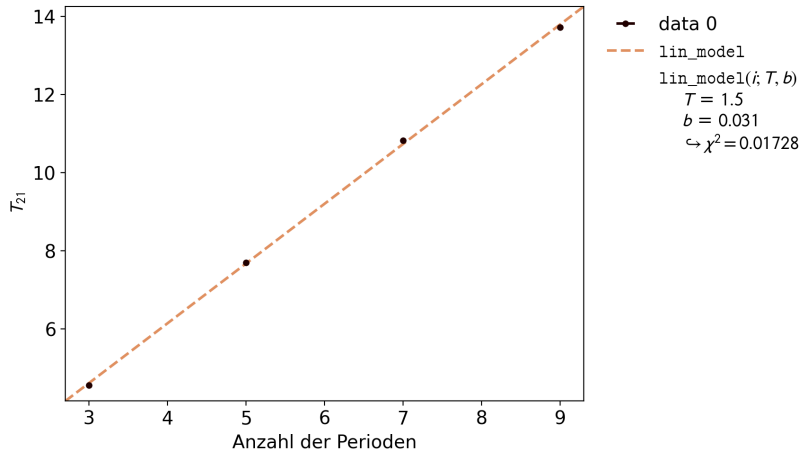
[14]:

Anschließend wurde eine Kopplungshöhe $l_1 = 0.3\,m$ eingestellt und es wurden die Schwingdauer T_{11} der gleichphasigen Eigenschwingung ohne Einfluss der Feder und T_{21} der gegenphasigen Schwingung gemessen.

[14]:



[14]:



[14]:

Dabei ergaben sich Werte von $T_{11} = 1.7\,s$ und $T_{21} = 1.5\,s$. Man erwartet wegen der bei der gegenphasigen Schwingung hinzukommenden Kopplung $\omega_1^2 < \omega_2^2$, dann aber wegen $\omega = \frac{2\pi}{T}$ folglich $T_{11} > T_{21}$, die gemessenen Werte entsprechen also in ihrem Verhältnis der Erwartung.

[14]:

	l [m]	T_{1i} [s] (gleichphasig)	T_{2i} [s] (gegenphasig)
0	0.415	1.783	1.417
1	0.3	1.7375	1.529

[14]:

Im Rahmen der Messungenauigkeiten sind die Werte sehr kompatibel miteinander, da die Verhältnisse gleich bleiben ($T_1 > T_2$). Die Unterschiede der einzelnen Periodendauern sind im Rahmen der Genauigkeit primär auf Unsicherheiten zurückzuführen, sodass ein Einfluss der Kopplungshöhe nicht zu vermuten ist, da sonst stärkere Unterschiede zu erwarten wären.

[14]: Aus diesen Werten folgen Werte für :

[14]:

	l [m]	ω_{1i} [s] (gleichphasig)	ω_{2i} [s] (gegenphasig)
0	0.415	3.52394	4.43415
1	0.3	3.61622	4.10934

[14]: Mittelwert: $\omega_1 = 3.6\,s$ (gleichphasig) und $\omega_2 = 4.3\,s$ (gegenphasig)

[14]: Diese Werte sind aus der Diskussion über die entsprechenden Periodendauern folgend plausibel und den Erwartungen entsprechend.

[14]: Berechnung von Θ

[14]: Aus ω_1 und den Maßen des Pendels lässt sich $\Theta_{Modell} = 1.36\,kg\,m^2$ bestimmen.

[14]: Es lässt sich aus den Abmessungen und Gewichten das Trägheitsmoment theoretisch zu $\Theta_{Theorie} = 1.53\,kg\,m^2$ bestimmen.

[14]: Diese Werte weichen durchaus voneinander ab, was im Rahmen der Zeitmessung allerdings wenig überraschend ist. In der weiteren Rechnung wird der Mittelwert der beiden Werte verwendet.

[14]: Im folgenden soll das Direktionsmoment D, welches die Kopplung der beiden Pendel beschreibt, durch verschiedene Methoden bestimmt werden:

[14]: Zunächst durch Verwendung des Trägheitsmoments und der beiden Eigenfrequenzen.

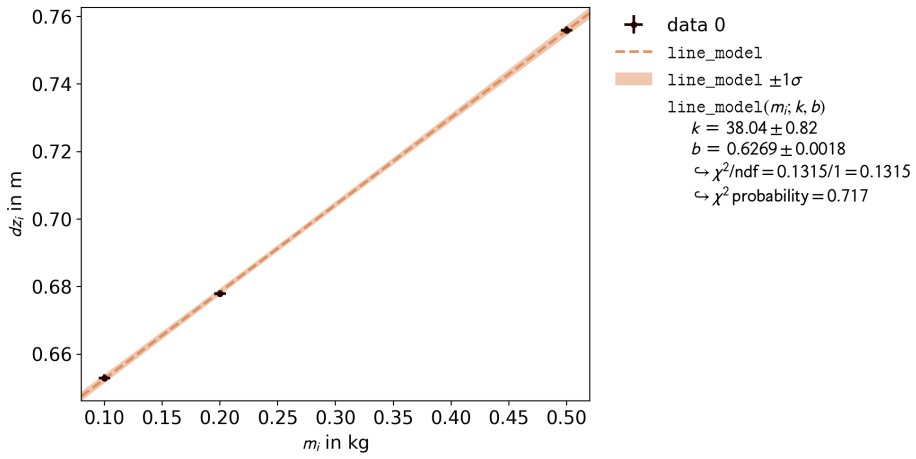
[14]: Berechnung von D_{ω_2}

[14]: Es ergibt sich das Direktionsmoment $D_{\omega_2} = \Theta * (\omega_2^2 - \omega_1^2) = 8.0\,\frac{kg\,m^2}{s^2}$.

[14]: Berechnung von D_{hook}

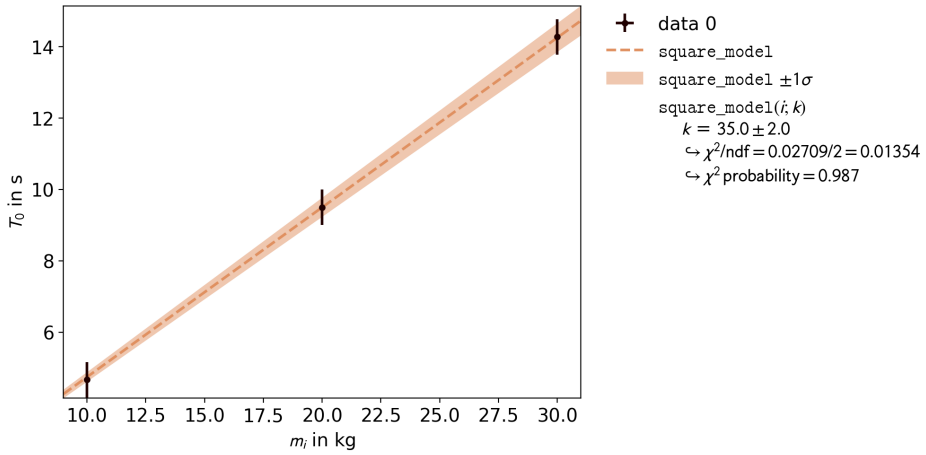
[14]: Nun soll es durch Anhängen verschiedener Massen und Messen der Auslenkungen unter Annahme des linearen Bereichs des Hookschen Gesetzes bestimmt werden.

[14]:



[14]:
Berechnung von D_{Pendel}

[14]:
Bei der Bestimmung über die Periodendauer des Federpendels wird die benötigte Zeit einer angehängten Masse für mehrere Schwingungen gemessen, um die Periodendauer und daraus über den Ansatz eines harmonischen Oszillators D zu bestimmen.

[14]:


[14]:
Vergleich der D Werte

[14]:
$D_{\omega_2} = 7.96 \frac{kg\,m^2}{s^2}$

[14]:
$D_{Hooksch} = 6.55 \frac{kg\,m^2}{s^2}$

[14]:
$D_{Pendel} = 6.03 \frac{kg\,m^2}{s^2}$

[14]:
Da D_{ω_2} von den meisten gemessenen Werten abhängt und im Vergleich stärker von den anderen beiden Werten abweicht, als diese untereinander, ist davon auszugehen, dass die anderen beiden Messverfahren genauer sind. Dies könnte konkret daran liegen, dass die Feder bei der Kopplung durchhing und konstant gespannt war, während sie bei den anderen beiden Experimenten mehr der Theorie entsprechend gerade hing und in Ruhe eher entspannt war. Relevant zur Bewertung wären allerdings auch die hier nicht bestimmten Fehler, um einzuschätzen, wie kompatibel die Messungen sind.

2.3.3 Aufgabe 3.3: Schwebung

- Messen Sie die Schwingungsperiode $\overline{T}$ zur Kreisfrequenz $\overline{\omega} = \frac{1}{2}(\omega_1 + \omega_2)$ und die Schwebungsperiode $\tilde{T}$ zur Kreisfrequenz $\tilde{\omega} = \frac{1}{2}|\omega_1 - \omega_2|$, bei Anregung der gekoppelten Pendel zu Schwebungen. Nutzen Sie dazu einen der Abstände ℓ, die Sie auch in Aufgabe 3.2 benutzt haben.
- Prüfen Sie den erwarteten Zusammenhang zwischen $\overline{T}$ und $\tilde{T}$ mit T_1 und T_2.

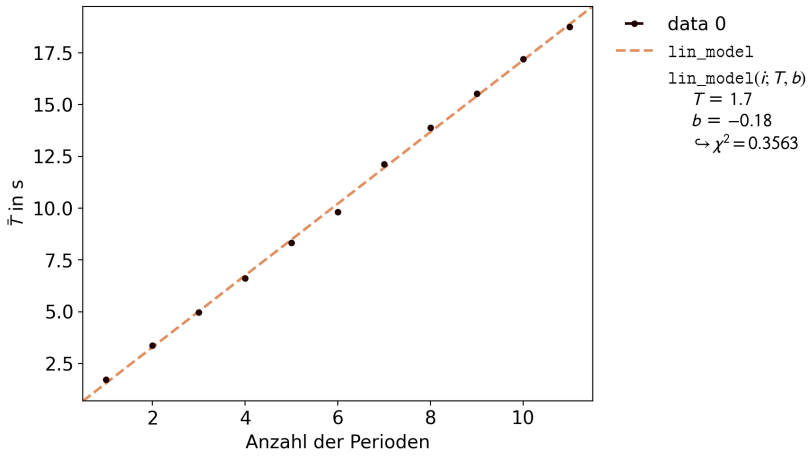
[10]:

```
%m Es wurde zu Beginn lediglich eines der Pendel so ausgelenkt und die␣
 ↪Periodendauer $\overline{T}$ der Schwingung eines Pendels und die␣
 ↪Periodendauer $\tilde{T}$ der Einhüllenden bestimmt.
T_bar = [ufloat(1.73,1),
                 ufloat(3.37,1),
                 ufloat(4.97,1),
                 ufloat(6.62,1),
                 ufloat(8.34,1),
                 ufloat(9.81,1), #weg?!
                 ufloat(12.13,1),
                 ufloat(13.89,1),
                 ufloat(15.54,1),
                 ufloat(17.21,1),
                 ufloat(18.75,1)] #Dauer Eigenschwingung (kürzer)


i = np.arange(1,len(T_bar)+1)
def lin_model(i,T=3,b=.3):
    return i*T+b

xy_data = kafe2.XYContainer(x_data=i, y_data=nomv(T_bar))
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$\bar{T}$ in s'
p.plot()
p.show()
T_bar = lin_fit.parameter_values[0]

#Dauer Einhüllende (Energieübertragung, länger)
T_tilde = [ufloat(11.06,1),
                  ufloat(21.3,1),
                  ufloat(31.29,1),
                  ufloat(41.74,1),
                  ufloat(52.41,1),
                  ufloat(62.53,1),
                  ufloat(73.07,1)]
i = np.arange(1,len(T_tilde)+1)
def lin_model(i,T=3,b=.3):
    return i*T+b

xy_data = kafe2.XYContainer(x_data=i, y_data=nomv(T_tilde))
lin_fit = kafe2.Fit(data=xy_data, model_function=lin_model)

lin_fit.do_fit()
```

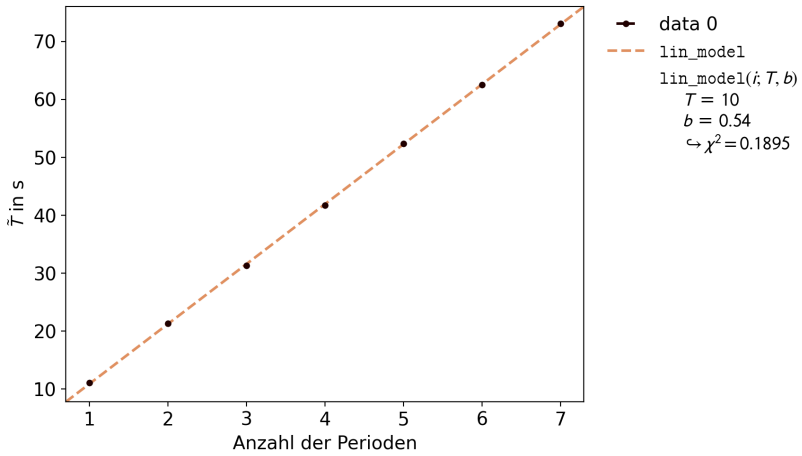
```
p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl der Perioden'
p.y_label = r'$\tilde{T}$ in s'
p.plot()
p.show()
T_tilde = lin_fit.parameter_values[0]


 _bar = 2*np.pi/T_bar
 _tilde = 2*np.pi/T_tilde
%m Die aus diesen gemessenen Werten bestimmten Frequenzen $\overline{ } = [!
  ↪ _bar:.2!]\, \frac{1}{s}$ und $\tilde{ } = [! _tilde:.2!]\, \frac{1}{s}$␣
  ↪passen im Rahmen der Messunsicherheiten einer Zeitmessung mit einer Stoppuhr␣
  ↪und dem ungenau einzeln auszulenkenden Pendel, gut zu den aus␣
  ↪Eigenfrequenzen zusammengesetzten Werten von $\overline{ }_{Eigen} =␣
  ↪\frac{1}{2}*( _2+ _1) = [!1/2*( _1+ _2):.2!]\, \frac{1}{s}$ und␣
  ↪$\tilde{ }_{eigen} = \frac{1}{2}*( _2- _1) = [!1/2*( _2- _1):.2!]\,␣
  ↪\frac{1}{s}$.
```

[10]: Es wurde zu Beginn lediglich eines der Pendel so ausgelenkt und die Periodendauer $\overline{T}$ der Schwingung eines Pendels und die Periodendauer $\tilde{T}$ der Einhüllenden bestimmt.

[10]:



[10]:



[10]: Die aus diesen gemessenen Werten bestimmten Frequenzen $\overline{\omega} = 3.6\,\frac{1}{s}$ und $\tilde{\omega} = 0.61\,\frac{1}{s}$ passen im Rahmen der Messunsicherheiten einer Zeitmessung mit einer Stoppuhr und dem ungenau einzeln auszulenkenden Pendel, gut zu den aus Eigenfrequenzen zusammengesetzten Werten von $\overline{\omega}_{Eigen} = \frac{1}{2} * (\omega_2 + \omega_1) = 3.9\,\frac{1}{s}$ und $\tilde{\omega}_{eigen} = \frac{1}{2} * (\omega_2 - \omega_1) = 0.35\,\frac{1}{s}$.

[0]:

[0]: