Elastizitaet.nbconvert

December 30, 2023

1 Fakultät für Physik

1.1 Physikalisches Praktikum P1 für Studierende der Physik

Versuch P1-11, 12, 13 (Stand: Oktober 2023)

Raum F1-19

2 Elastizität

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: _____

[1]:
```
# 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
import uncertainties
from uncertainties import ufloat, unumpy
from uncertainties.unumpy import uarray as uarr
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 scipy
import matplotlib.image as mpimg
from io import StringIO

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)

def uplot(x,y,*args, **kwargs):
    utypes = (uncertainties.core.
Variable,uncertainties.core.AffineScalarFunc)
    xplt = x
    yplt = y
    if type(x[0]) in utypes:
        xplt = nomv(x)
        kwargs["xerr"] = stdv(x)
    if type(y[0]) in utypes:
        yplt = nomv(y)
        kwargs["yerr"] = stdv(y)
    plt.errorbar(xplt,yplt,*args,**kwargs)

def pltimg(path):
    try:
        img = mpimg.imread(path)
        imgplot = plt.imshow(img)
        plt.axis('off')
        plt.show()
    except Exception as e:
        print("Image Not Found")
        print(e)
```

3 Durchführung

3.1 Aufgabe 1: Elastizitätsmodul aus der Biegung flacher Stäbe

Bei diesem Versuchsteil bestimmen Sie für einige Stoffe den Elastizitätsmodul E aus der Biegung zweiseitig aufgelagerter und in der Mitte belasteter, flacher Stäbe. Hierzu stehen Ihnen fünf Materialien zur Verfügung:

- Kupfer,
- Aluminium,
- Edelstahl,
- Messing,
- PVC.

3.1.1 Aufgabe 1.1: Messuhr

Bestimmen Sie E aus der Biegung s des Stabs als Funktion der Kraft F_g angehängter Gewichte, mit Hilfe einer Messuhr. Für s gilt der Zusammenhang

$$s(F_g) = \frac{L^3 F_g}{4E\, b\, d^3},$$

worin L dem Abstand zwischen den beiden Auflagepunkten des jeweiligen Stabs, b der Breite und d der Dicke des Stabs entsprechen. Berücksichtigen Sie für Ihre Messung, dass die Messuhren eine zusätzliche Kraft von $3,6\,\mathrm{g/mm}$ auf den jeweiligen Stab ausüben.

---

[2]:
```
ΔUhr = 10e-6
ΔMassen = 0.5e-3

uL = ufloat(450e-3,2e-3)
ub = ufloat(25e-3, 0.1e-3)
ud = ufloat(6e-3, 0.1e-3)
ug = ufloat(9.809599, 0.00001)

%m In diesem Experiment geht es darum das Elastizitätsmodul aus der Biegung␣
 ↪eines Stabes bei verschiedenen Kräften zu bestimmen. Folgende Messwert␣
 ↪werden von dem Aufbau geliefert:

a11 = """
Masse in kg, PVC in mm, Aluminium in mm, Edelstahl in mm, Kupfer in mm, Messing␣
 ↪in mm
0,   8e-2,     0,     0,      0,     0
0.2, 78e-2,    12e-2, 4.5e-2, 6e-2,  8.7e-2
0.5, 186.5e-2, 30e-2, 11e-2,  16e-2, 21.2e-2
1,   364.5e-2, 62e-2, 22e-2,  32e-2, 42e-2
"""
```

```
m, PVC, Al, Fe, Cu, CuZn = np.genfromtxt(StringIO(a11), delimiter=',')[1:].
 ↪transpose()
PVC, Al, Fe, Cu, CuZn = PVC*1e-3, Al*1e-3, Fe*1e-3, Cu*1e-3, CuZn*1e-3

mpg = pd.read_csv(StringIO(a11))
#mpg.rename(columns={"l": 'l in m', "l'": "l' in m","s": "d in m", "ds":␣
 ↪"Unsicherheit in m"}, inplace=True)
display(mpg)

%m Die Messuhr hat eine Unsicherheit von [!ΔUhr*1e3!]mm. Die Massen der␣
 ↪Gewichte haben eine Unsicherheit von [!ΔMassen*1e3!]g. Die Balken haben alle␣
 ↪die gleichen Abmessungen $d = [!ud.n!]$m und $b = [!ub.n!]$m. Der Abstand␣
 ↪der Auflageflächen wurde mit $L = [!uL.n!]$m bestimmt. Für L wurde die␣
 ↪Unsicherheit auf [!uL.s*1e3!]mm geschätzt, für d und u auf [!ud.s*1e3!]mm.

%m Durch die Kraft, die die Messuhr auf den Stab ausübt setzt sich F aus zwei␣
 ↪Kräften $F_g$ und $F_m = \alpha\,g\,s$ zusammen. Damit ist die Formel für␣
 ↪die Auslenkung leicht verändert: $s(F) = \frac{L^{3}\,(F_{g} +␣
 ↪\alpha\,g\,s)}{4E\,b\,d^{3}}$. Aufgelöst nach s ergibt sich ein neuer␣
 ↪Zusamenhang von $F_g$ und $s$ wie folgt $s = \frac{L^3 F_g}{4 E b d^3 -␣
 ↪\alpha \,g L^3}$ mit $\alpha = 3.6\frac{Kg}{m}$. Diesen Zusammenhang␣
 ↪zusammen mit einem y-Achsenabschnit können wir im folgenden als Funktion für␣
 ↪einen Fit mit Kafe2 verwenden. Für die Erdbeschleunigung wurde hier ein␣
 ↪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 abgerufen am 01.12.2023)

def model(m, L, b, d, g, nullfehler, E=10):
    #print(4*E*b*d**3, 3.6*g*L**3)
    s = (L**3*m*g)/(4*E*1e9*b*d**3 - 3.6*g*L**3)
    #s = (L**3*m*g)/(4*E*b*d**3)
    return s+nullfehler

mats = ("PVC", "Aluminium", "Edelstahl", "Kupfer", "Messing")
data = (PVC, Al, Fe, Cu, CuZn)
for i in range(5):
    %m Anpassung für [!mats[i]!]
    xy_data = kafe2.XYContainer(x_data=m, y_data=data[i])
    xy_data.add_error(axis='x', err_val=ΔMassen)
    xy_data.add_error(axis='y', err_val=ΔUhr)

    lin_fit = kafe2.Fit(data=xy_data, model_function=model)
    lin_fit.add_parameter_constraint(name='L',value=uL.n,uncertainty=uL.s)
    lin_fit.add_parameter_constraint(name='b',value=ub.n,uncertainty=ub.s)
    lin_fit.add_parameter_constraint(name='d',value=ud.n,uncertainty=ud.s)
    lin_fit.add_parameter_constraint(name='g',value=ug.n,uncertainty=ug.s)
    lin_fit.do_fit()
```

```
    p = kafe2.Plot(lin_fit)
    p.x_label = 'Masse in Kg'
    p.y_label = 'Auslenkung s in m'
    p.plot()
    plt.title(mats[i])
    p.show()

    _,_,_,_,_,E = lin_fit.parameter_values
    _,_,_,_,_,ΔE = lin_fit.parameter_errors
    uE = ufloat(E,ΔE)*1e9

    %m Damit ergibt sich ein Wert von $E = [!uE.n*1e-9:.2f!]Gpa \pm [!uE.s*1e-9:
  ↪.2f!]Gpa$ für [!mats[i]!]

%m Die $ ^2$-Werte sind in einem annehmbaren Bereich und die Geraden passen gut␣
  ↪zu den Daten. Bei den Messungen für Messing, Kupfer und Edelstahl sind die␣
  ↪Messunsicherheiten zu groß geschätzt. Hier scheint eine relative␣
  ↪Unsicherheit in der Messuhr zu existieren, die aber leider nicht genau␣
  ↪bekannt ist. Insgesamt passen die bestimmten E-Module aber gut zu den␣
  ↪erwarteten Werten.
```

In diesem Experiment geht es darum das Elastizitätsmodul aus der Biegung eines Stabes bei verschiedenen Kräften zu bestimmen. Folgende Messwert werden von dem Aufbau geliefert:

	Masse in kg	PVC in mm	Aluminium in mm	Edelstahl in mm	Kupfer in mm	\
0	0.0	0.080	0.00	0.000	0.00	
1	0.2	0.780	0.12	0.045	0.06	
2	0.5	1.865	0.30	0.110	0.16	
3	1.0	3.645	0.62	0.220	0.32	

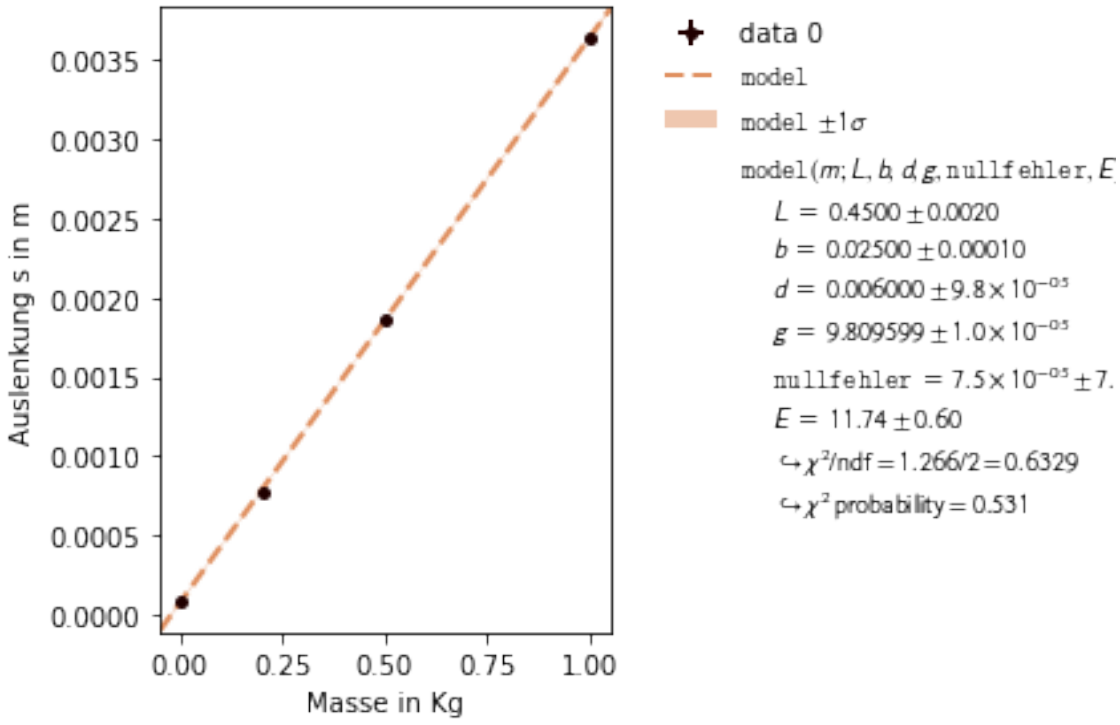
	Messing in mm
0	0.000
1	0.087
2	0.212
3	0.420

Die Messuhr hat eine Unsicherheit von 0.01mm. Die Massen der Gewichte haben eine Unsicherheit von 0.5g. Die Balken haben alle die gleichen Abmessungen $d = 0.006$m und $b = 0.025$m. Der Abstand der Auflageflächen wurde mit $L = 0.45$m bestimmt. Für L wurde die Unsicherheit auf 2.0mm geschätzt, für d und u auf 0.1mm.

Durch die Kraft, die die Messuhr auf den Stab ausübt setzt sich F aus zwei Kräften F_g und $F_m = \alpha\, g\, s$ zusammen. Damit ist die Formel für die Auslenkung leicht verändert: $s(F) = \frac{L^3\,(F_g+\alpha\, g\, s)}{4E\, b\, d^3}$. Aufgelöst nach s ergibt sich ein neuer Zusamenhang von F_g und s wie folgt $s = \frac{L^3 F_g}{4Ebd^3-\alpha\, gL^3}$ mit $\alpha = 3.6\frac{Kg}{m}$. Diesen Zusammenhang zusammen mit einem y-Achsenabschnit können wir im folgenden als Funktion für einen Fit mit Kafe2 verwenden. Für die Erdbeschleunigung wurde hier ein Literaturwert von 9.809599 $\frac{m}{s^2}$ (in Mannheim nach

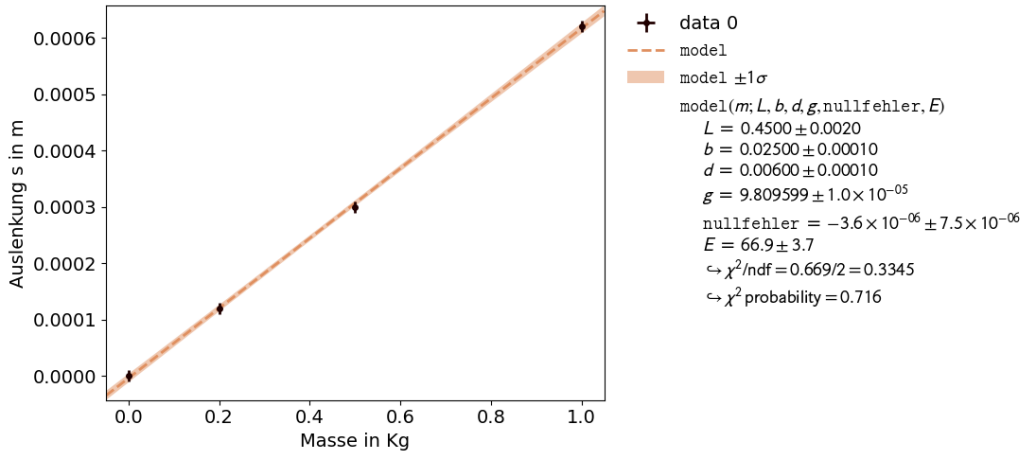
https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_1/1.1_masse/1.15/tabelle.pdf abgerufen am 01.12.2023)

Anpassung für PVC



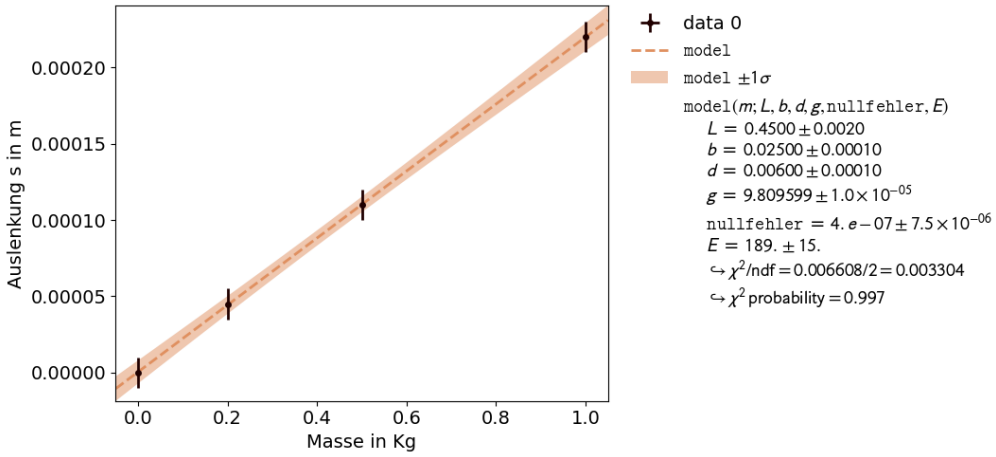
Damit ergibt sich ein Wert von $E = 11.74Gpa \pm 0.60Gpa$ für PVC

Anpassung für Aluminium



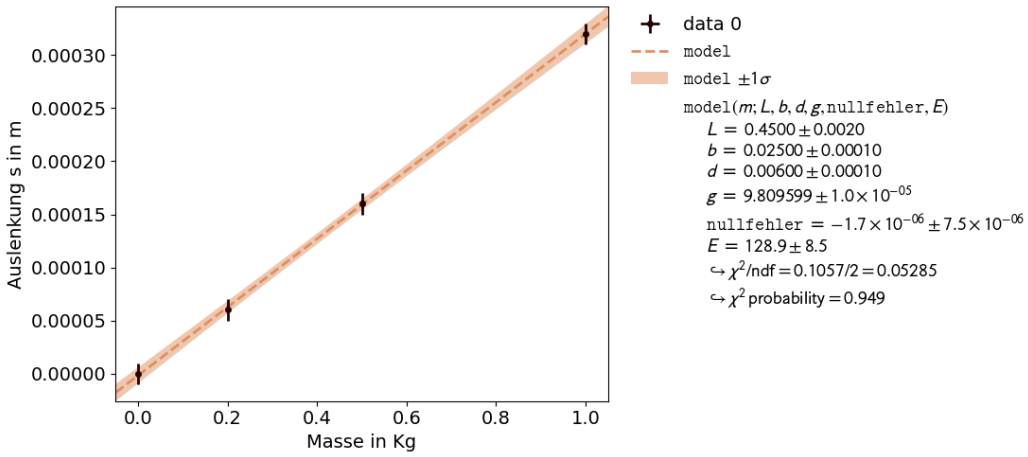
Damit ergibt sich ein Wert von $E = 66.87Gpa \pm 3.74Gpa$ für Aluminium

Anpassung für Edelstahl



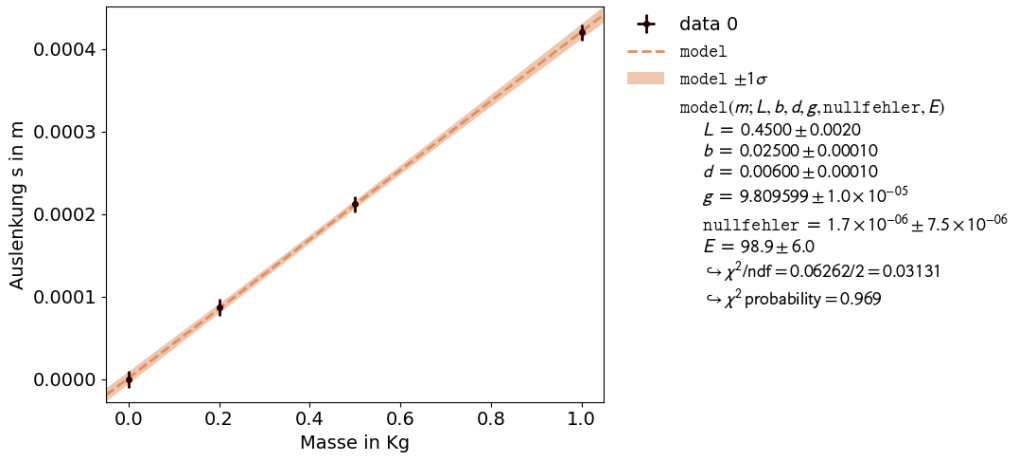
Damit ergibt sich ein Wert von $E = 188.61Gpa \pm 15.00Gpa$ für Edelstahl

Anpassung für Kupfer



Damit ergibt sich ein Wert von $E = 128.85Gpa \pm 8.53Gpa$ für Kupfer

Anpassung für Messing



Damit ergibt sich ein Wert von $E = 98.90Gpa \pm 6.00Gpa$ für Messing

Die χ^2-Werte sind in einem annehmbaren Bereich und die Geraden passen gut zu den Daten. Bei den Messungen für Messing, Kupfer und Edelstahl sind die Messunsicherheiten zu groß geschätzt. Hier scheint eine relative Unsicherheit in der Messuhr zu existieren, die aber leider nicht genau bekannt ist. Insgesamt passen die bestimmten E-Module aber gut zu den erwarteten Werten.

3.1.2 Aufgabe 1.2: Spiegelsystem

Für diesen Aufgabenteil bestimmen Sie die Biegung des Stabs aus der zusätzlichen Neigung α der Stabenden bei Belastung. Diese bestimmen Sie mit Hilfe zweier Spiegel, die nahe den Stabenden, leicht gegeneinander geneigt montiert sind. Über die Spiegel wird ein punktförmiger Laserstrahl auf einen entfernten Schirm gelenkt. Aus der Verschiebung δ der Strahlposition mit und ohne Belastung kann α nach der Formel

$$\alpha(F_g) = \frac{3L^2\, F_g}{4E\, b\, d^3}$$

berechnet werden.

---

[3]:
```
%m In diesem Versuch soll das Elastizitätsmodul eines Balkens mit Hilfe von␣
↪Spiegeln auf den Auflagepunktn bestimmt werden. Aus dem Winkel $\alpha$␣
↪ergibt sich die verschiebung auf dem Schirm direkt als $v = 2L\alpha +␣
↪4(l+L)\alpha$ unter Annahme der Kleinwinkelnäherung die hier anwenbar ist.␣
↪Die Spiegel wurden aus Stabilitätsgründen $\bar{l}=3\,mm$  innerhalb der␣
↪Stützstellen montiert. Dementsprechend muss die Formel folgendermaßen␣
↪modifiziert werden:  $v = 2L\alpha + 4(l-\bar{l}+L)\alpha$.

# Die Spiegel sitzen 3mm weiter innen als erhofft die .

ΔUhr = 0.5e-3
```

```
ΔUhr2 = 5e-3
ΔMassen = 0.5e-3

uL = ufloat(455e-3,2e-3)
ulfehler = ufloat(3e-3,0.3e-3)
ub = ufloat(25e-3, 0.5e-3)
ud = ufloat(6e-3, 0.5e-3)
ug = ufloat(9.809599, 0.00001)

ul = ufloat(4.877,2e-3)

%m Folgende Messwert werden für die Verschiebung v des Lasers von dem Aufbau␣
 ↪geliefert:
#alle PVC Messwerte relativ zum 1kg Wert
a11 = """
Masse in kg, PVC in mm, Aluminium in mm, Messing in mm
0          , -531     , 0               , 0
0.2        , -439     , 17.5            , 12.3
0.5        , -276     , 46              , 31
0.7        , -165     , 64.5            , 43.3
1          ,    0     , 92              , 62
"""

m, PVC, Al, CuZn = np.genfromtxt(StringIO(a11), delimiter=',')[1:].transpose()
PVC = PVC - PVC[0]
PVC, Al, CuZn = PVC*1e-3, Al*1e-3, CuZn*1e-3

mpg = pd.read_csv(StringIO(a11))
mpg[" PVC in mm"] = mpg[" PVC in mm"] - mpg[" PVC in mm"][0]
display(mpg)

%m Das Ablesen der Verschiebung für den Laserstrahl wird mit einem Geodreieck␣
 ↪gemacht und hat eine Unsicherheit von [!ΔUhr*1e3!]mm außer bei PVC, wo die␣
 ↪Verschiebungen zu groß sind für das Geofreieck, sodass ein weniger genaues␣
 ↪Maßband genutzt wird mit einer Unsicherheit von [!ΔUhr2*1e3!]mm. Die Massen␣
 ↪der Gewichte haben eine Unsicherheit von [!ΔMassen*1e3!]g. Die Balken haben␣
 ↪alle die gleichen Abmessungen $d = [!ud.n!]$m und $b = [!ub.n!]$m. Der␣
 ↪Abstand der Auflageflächen wurde mit $L = [!uL.n!]$m bestimmt. Für L wurde␣
 ↪die Unsicherheit auf [!uL.s*1e3!]mm geschätzt, für d und u auf [!ud.s*1e3!
 ↪]mm. Für die Erdbeschleunigung wurde hier ein 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 abgerufen am 01.12.
 ↪2023). Der Abstand $l$ zwischen Wand und dem der Wand am nächsten stehenden␣
 ↪Spiegel ist $l = [!ul.n!]$m und wurde mit einer Unsicherheit von $[!ul.s!]$m␣
 ↪bestimmt.

def model(m, L, b, d, g, l, lfehler, E=10):
```

```
      = (3*L**2*m*g)/(4*E*1e9*b*d**3)
    v = (4*(l-2*lfehler)+6*L)*
    return v

mats = ("PVC", "Aluminium", "Messing")
data = (PVC, Al, CuZn)
for i in range(3):
    %m Anpassung für [!mats[i]!]
    xy_data = kafe2.XYContainer(x_data=m, y_data=data[i])
    xy_data.add_error(axis='x', err_val=ΔMassen)
    xy_data.add_error(axis='y', err_val=ΔUhr2 if mats[i]=="PVC" else ΔUhr)

    lin_fit = kafe2.Fit(data=xy_data, model_function=model)
    lin_fit.add_parameter_constraint(name='L',value=uL.n,uncertainty=uL.s)
    lin_fit.add_parameter_constraint(name='b',value=ub.n,uncertainty=ub.s)
    lin_fit.add_parameter_constraint(name='d',value=ud.n,uncertainty=ud.s)
    lin_fit.add_parameter_constraint(name='g',value=ug.n,uncertainty=ug.s)
    lin_fit.add_parameter_constraint(name='l',value=ul.n,uncertainty=ul.s)
    lin_fit.add_parameter_constraint(name='lfehler',value=ulfehler.
 ↪n,uncertainty=ulfehler.s)
    lin_fit.do_fit()

    p = kafe2.Plot(lin_fit)
    p.x_label = 'Masse in Kg'
    p.y_label = 'Auslenkung v in m'
    p.plot()
    plt.title(mats[i])
    p.show()

    _,_,_,_,_,_,E = lin_fit.parameter_values
    _,_,_,_,_,_,ΔE = lin_fit.parameter_errors
    uE = ufloat(E,ΔE)*1e9

    %m Damit ergibt sich ein Wert von $E = [!uE.n*1e-9:.2f!]Gpa \pm [!uE.s*1e-9:
 ↪.2f!]Gpa$ für [!mats[i]!]

%m Die hier bestimmten Werte sind im Rahmen der bestimmten Messunsicherheiten␣
 ↪mit den im vorherigen Versuch bestimmten Werten kompatibel.
```

In diesem Versuch soll das Elastizitätsmodul eines Balkens mit Hilfe von Spiegeln auf den Auflagepunktn bestimmt werden. Aus dem Winkel α ergibt sich die verschiebung auf dem Schirm direkt als $v = 2L\alpha + 4(l + L)\alpha$ unter Annahme der Kleinwinkelnäherung die hier anwenbar ist. Die Spiegel wurden aus Stabilitätsgründen $\bar{l} = 3\,mm$ innerhalb der Stützstellen montiert. Dementsprechend muss die Formel folgendermaßen modifiziert werden: $v = 2L\alpha + 4(l - \bar{l} + L)\alpha$.

Folgende Messwert werden für die Verschiebung v des Lasers von dem Aufbau geliefert:

Masse in kg	PVC in mm	Aluminium in mm	Messing in mm

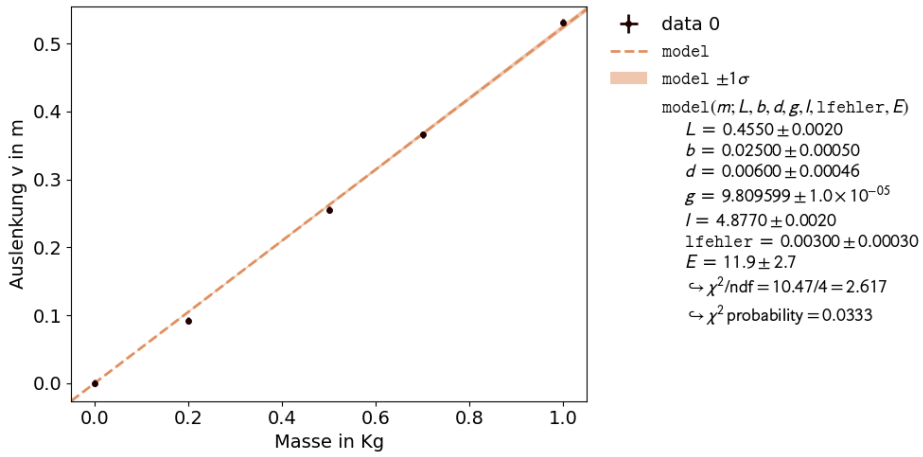
0	0.0	0	0.0	0.0
1	0.2	92	17.5	12.3
2	0.5	255	46.0	31.0
3	0.7	366	64.5	43.3
4	1.0	531	92.0	62.0

Das Ablesen der Verschiebung für den Laserstrahl wird mit einem Geodreieck gemacht und hat eine Unsicherheit von 0.5mm außer bei PVC, wo die Verschiebungen zu groß sind für das Geofreieck, sodass ein weniger genaues Maßband genutzt wird mit einer Unsicherheit von 5.0mm. Die Massen der Gewichte haben eine Unsicherheit von 0.5g. Die Balken haben alle die gleichen Abmessungen $d = 0.006$m und $b = 0.025$m. Der Abstand der Auflageflächen wurde mit $L = 0.455$m bestimmt. Für L wurde die Unsicherheit auf 2.0mm geschätzt, für d und u auf 0.5mm. Für die Erdbeschleunigung wurde hier ein 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 abgerufen am 01.12.2023). Der Abstand l zwischen Wand und dem der Wand am nächsten stehenden Spiegel ist $l = 4.877$m und wurde mit einer Unsicherheit von 0.002m bestimmt.

Anpassung für PVC

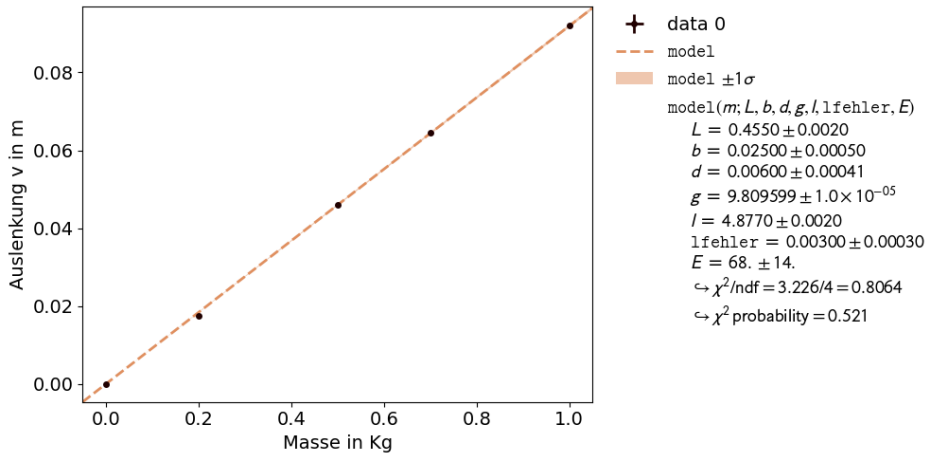
```
/tmp/ipykernel_43172/533294392.py:39: RuntimeWarning: divide by zero encountered
in divide
    = (3*L**2*m*g)/(4*E*1e9*b*d**3)
/tmp/ipykernel_43172/533294392.py:39: RuntimeWarning: invalid value encountered
in divide
    = (3*L**2*m*g)/(4*E*1e9*b*d**3)
```

```
Warning: the cost function has been evaluated as infinite. The fit might not
converge correctly.
```



Damit ergibt sich ein Wert von $E = 11.95Gpa \pm 2.73Gpa$ für PVC

Anpassung für Aluminium

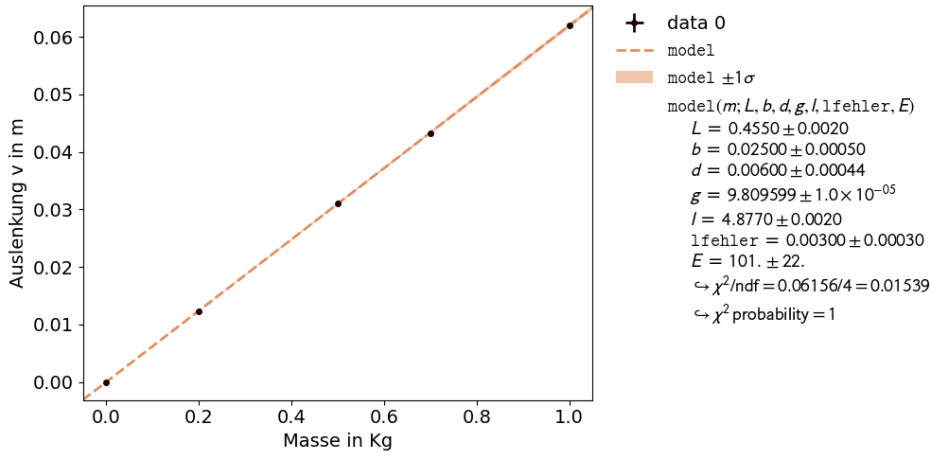


Damit ergibt sich ein Wert von $E = 68.16Gpa \pm 14.15Gpa$ für Aluminium

Anpassung für Messing

```
/tmp/ipykernel_43172/533294392.py:39: RuntimeWarning: divide by zero encountered
in divide
    = (3*L**2*m*g)/(4*E*1e9*b*d**3)
/tmp/ipykernel_43172/533294392.py:39: RuntimeWarning: invalid value encountered
in divide
    = (3*L**2*m*g)/(4*E*1e9*b*d**3)

Warning: the cost function has been evaluated as infinite. The fit might not
converge correctly.
```



Damit ergibt sich ein Wert von $E = 101.17Gpa \pm 22.10Gpa$ für Messing

Die hier bestimmten Werte sind im Rahmen der bestimmten Messunsicherheiten mit den im vorherigen Versuch bestimmten Werten kompatibel.

3.2 Aufgabe 2: Elastizitätsmodul aus der Schallgeschwindigkeit

In diesem Versuchsteil bestimmen Sie die Zeit Δt, die ein Verdichtungsstoß benötigt, um mehrfach im Stab der Länge L hin- und herzulaufen. Der Verdichtungsstoß wird verursacht durch das axiale Anschlagen einer Pendelkugel ans entsprechende Stabende erzeugt. Die entstehende Welle wird an den freien Stabenden jeweils reflektiert wird. Am erregerfernen Stabende werden die Schallsignale mit Hilfe eines Piezokristalls in Spannungspulse umgewandelt und mit einem Oszilloskop aufgenommen.

Bestimmen Sie die Zeitdifferenzen Δt zu möglichst vielen Reflektionen und ermitteln Sie die Schallgeschwindigkeit v_{ph} als Steigung der resultierenden Geraden. Beachten Sie, dass es bei jeder Messung nötig ist, in einem Vorversuch zunächst etwa die Mitte jenes Bereichs der Anschlagstärke zu suchen, in dem die Ergebnisse für Δt reproduzierbar sind. Störsignale, z.B. von unvermeidlichen andersartigen Anregungsmoden lassen sich auf dem Oszilloskop gut vom Hauptschwingungsmodus unterscheiden. Lediglich bei PVC wird das Signal so schnell gedämpft, dass nur wenige Reflektionen messbar sind.

Achtung: Der Piezodetektor verträgt wegen der leicht aufbrechenden Klebungen keine harten Stöße. Die Pendelkugel darf daher aus höchstens 8 cm Entfernung vom Stabende losgelassen werden.

---

[4]:
```
%m Es werden die zu vermessenden Stäbe einseitig mit einem Pendel angestoßen␣
 ↪und die Erschütterung am anderen Ende mit einem PicoScope aufgezeichnet. Es␣
 ↪werden mehrere Reflektionen gemessen und aus der Periodendauer mit der Länge␣
 ↪des Stabs die Schallgeschwindigkeit im Medium ermittelt.
L = [ufloat(146e-2,2e-3),ufloat(1458e-3,2e-3),ufloat(1461e-3,2e-3)] #eintragen
L = [i+13e-3 for i in L]
d = [ufloat(1e-2,0.1e-3),ufloat(1e-2,0.1e-3),ufloat(1.06e-2,0.1e-3)]
m = [ufloat(1032e-3,1e-3),ufloat(327e-3,1e-3),ufloat(167e-3,1e-3)]
 = [4*m[i]/(np.pi*d[i]**2*L[i]) for i in np.arange(0,len(L))]#aus Durchmesser␣
 ↪und Länge mit Masse errechnen
names =["Kupfer","Aluminium","PVC"]
Δt_all = [50e-6,80e-6,660e-6]

v = [0 for i in L]
Δv = [0 for i in L]
E = [0 for i in L]
ΔE = [0 for i in L]

def lineModel(i,v=300,L=nomv(L[0])):
    return i*2*L/v
autokorruse = True

if autokorruse:
```

```
  %m Die Daten wurden am PC durch Setzen zweier Cursor an der geschätzten␣
↪Maxima ermittelt. Für die Daten wurde jeweils die größte Breite eines Peaks␣
↪als Unsicherheit angenommen, da im verwischten Maximum keine größere␣
↪Genauigkeit garantiert werden dann und sich diese Ungenauigkeit fortsetzen␣
↪kann.
  for k in range(0,len(L)):
      file = "A2_" + names[k] + ".csv"
      mpg = pd.read_csv(file)
      mpg.rename(columns={'i':'Anzahl Durchläufe','t': 'Zeit in s'},␣
↪inplace=True)
      j,t = np.genfromtxt(file, delimiter=',')[1:].transpose()
      Δt = Δt_all[k] #Abschätzen
      %m Vermessung eines [!names[k]!]-Stabes der Länge $L=([!L[k].n:.4!]\pm[!
↪L[k].s!])m$, Dicke $d=([!d[k].n!]\pm[!d[k].s!])m$ und Masse $m=([!m[k].n!
↪]\pm[!m[k].s!])kg$:
      display(mpg)
      #Fitten
      %m An die Daten wird nun ein Modell der Form $t = i \frac{2 L}{v}$␣
↪angepasst.
      xy_data = kafe2.XYContainer(x_data=j, y_data=t)
      line_fit = kafe2.Fit(data=xy_data, model_function=lineModel)
      line_fit.add_error("y",Δt)
      line_fit.add_parameter_constraint(name='L',   value=nomv(L[k]),  ␣
↪uncertainty=stdv(L[k]))
      line_fit.do_fit()
      p = kafe2.Plot(line_fit)
      p.x_label = 'Anzahl Durchläufe'
      p.y_label = 'Zeit in s'
      p.plot()
      p.show()
      v[k] = ufloat(line_fit.parameter_name_value_dict['v'],line_fit.
↪parameter_errors[0])
  %m Wie zu erkennen ist, ist die $\chi^2-Wahrscheinlichkeit$ aller Fits␣
↪äußerst gut, allerdings ist $\frac{\chi^2}{ndof}$ weit von 1 entfernt. Dies␣
↪liegt zum Teil an den scheinbar recht groß abgeschätzten Fehlern - der␣
↪Einfluss der breiteren Maxima auf die früheren Maxima ist wohl kleiner als␣
↪erwartet. Aber auch die geringe Anzahl der Messpunkte ist problematisch, ist␣
↪allerdings durch das recht schnelle Verwischen der Maxima bedingt, die ein␣
↪Ablesen weiterer Werte dahingehend unmöglich machen, dass es zum Ratespiel␣
↪wird, welches Maxima das richtige nächste ist. Eine messliche Verbesserung␣
↪unsererseits wäre es dann gewesen für jedes Material mehrerere Messreihen␣
↪aufzunehmen und immer die geringe Anzahl an bestimmbaren Perioden zu messen␣
↪und diese dann zusammenzufügen. Diese Variante hätte auch definitiv erlaubt␣
↪die Auswirkung der größeren Ableseungenauigkeit zu reduzieren.
```

```
else:
    i = 0
    #possibility 2 (with raw files)
    #Ermittelung der Autokorrelation
    file = "A2_Rohdaten/Alu" + ".csv"
    mpg = pd.read_csv(file)
    mpg.rename(columns={'Kanal A':'Auslenkung','Zeit': 'Zeit in s'},␣
 ↪inplace=True)
    t,ay = np.genfromtxt(file, delimiter=',')[1:].transpose()
    Δt = 1E-3 #Abschätzen
    %m Vermessung von [!names[i]!]:
    display(mpg)
    %m Diese Daten werden nun autokorreliert, um die Peaks und damit die␣
 ↪Periodendauer der umlaufenden Erschütterung zu bestimmen.
    autocov  = np.correlate(ay,ay, 'full')[-len(ay):]
    autocorr = autocov/autocov[0]
    #Bestimmen der Minima und Maxima
    ay_max = sci.signal.find_peaks(autocorr)[0]
    ay_min = sci.signal.find_peaks(autocorr*(-1))[0]
    # Darstellung der Autokorrelation und der Extrema
    plt.plot(t,autocorr)
    plt.scatter(t[ay_max],autocorr[ay_max])
    plt.scatter(t[ay_min],autocorr[ay_min])
    plt.xlabel("time difference")
    plt.ylabel("autocorrelation")
    plt.show()
    t_diff = np.concatenate([np.diff(t[ay_max]),np.diff(t[ay_min])])
    T_mean = np.mean(t_diff)
    ΔT = np.sqrt(np.var(t_diff,ddof=1))
    T = ufloat(T_mean,ΔT)
    v[i] = L[i]/T
    %m Es ergibt sich eine Periodendauer von $T_{[!names[i]!]} = ([!T_mean:.3!]␣
 ↪\pm [!ΔT:.3!])\, s$ und damit eine Schallgeschwindigkeit von $v_{[!names[i]!
 ↪]} = ([!nomv(v[i]):.2!] \pm [!stdv(v[i]):.2!])\, \frac{m}{s}$ .




for i in range(0,len(v)):
    E[i] =  [i]*v[i]**2 *1e-9

%m Aus diesen Werten lassen sich nach $E=  v^2$ Werte für E folgern, wobei $v$␣
 ↪aus den Fits entnommen und $\rho$ aus Masse und Volumen der Stäbe bestimmt␣
 ↪wurde:
```

```
df = pd.DataFrame({
    "Material": names,
    r"E-Modul in Gpa": E
})
display(df)
%m Es gibt kein wirklichen Vergleichswerte für diese Werte, da die genauen␣
 ↪Zusammensetzungen unbekannt sind und andere Stäbe als in den anderen␣
 ↪Aufgaben verwendet werden, allerdings liegen die Werte in ähnlichen␣
 ↪Größenordnungen. Lediglich der Wert für PVC ist etwas niedrig, was aber␣
 ↪leicht durch die geringe Anzahl an Messpunkten und ihre große Unsicherheit␣
 ↪zu begründen ist.
```

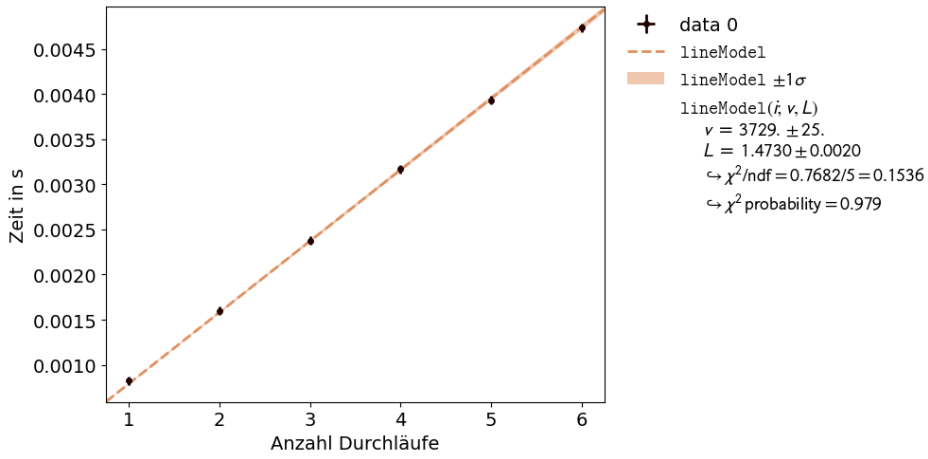
Es werden die zu vermessenden Stäbe einseitig mit einem Pendel angestoßen und die Erschütterung am anderen Ende mit einem PicoScope aufgezeichnet. Es werden mehrere Reflektionen gemessen und aus der Periodendauer mit der Länge des Stabs die Schallgeschwindigkeit im Medium ermittelt.

Die Daten wurden am PC durch Setzen zweier Cursor an der geschätzten Maxima ermittelt. Für die Daten wurde jeweils die größte Breite eines Peaks als Unsicherheit angenommen, da im verwischten Maximum keine größere Genauigkeit garantiert werden dann und sich diese Ungenauigkeit fortsetzen kann.

Vermessung eines Kupfer-Stabes der Länge $L = (1.473 \pm 0.002)m$, Dicke $d = (0.01 \pm 0.0001)m$ und Masse $m = (1.032 \pm 0.001)kg$:

	Anzahl Durchläufe	Zeit in s
0	1	0.000823
1	2	0.001600
2	3	0.002377
3	4	0.003167
4	5	0.003932
5	6	0.004734

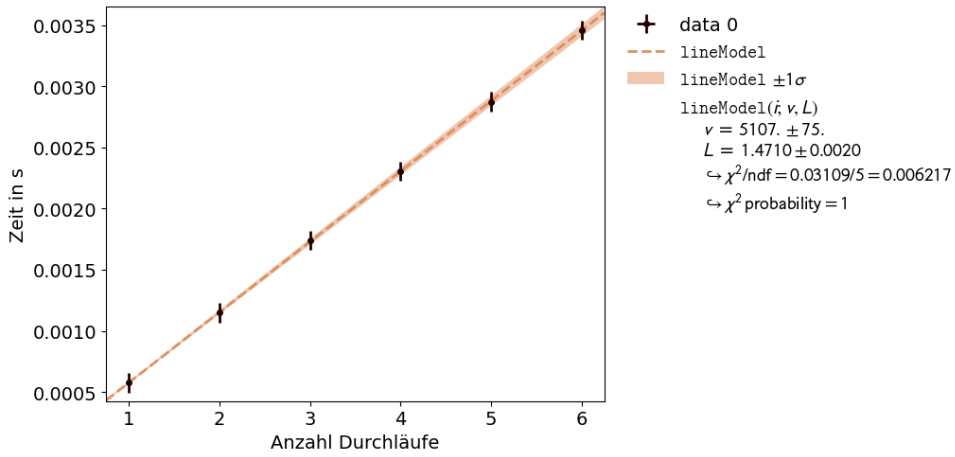
An die Daten wird nun ein Modell der Form $t = i\frac{2L}{v}$ angepasst.



Vermessung eines Aluminium-Stabes der Länge $L = (1.471 \pm 0.002)m$, Dicke $d = (0.01 \pm 0.0001)m$ und Masse $m = (0.327 \pm 0.001)kg$:

	Anzahl Durchläufe	Zeit in s
0	1	0.000576
1	2	0.001149
2	3	0.001739
3	4	0.002305
4	5	0.002872
5	6	0.003458

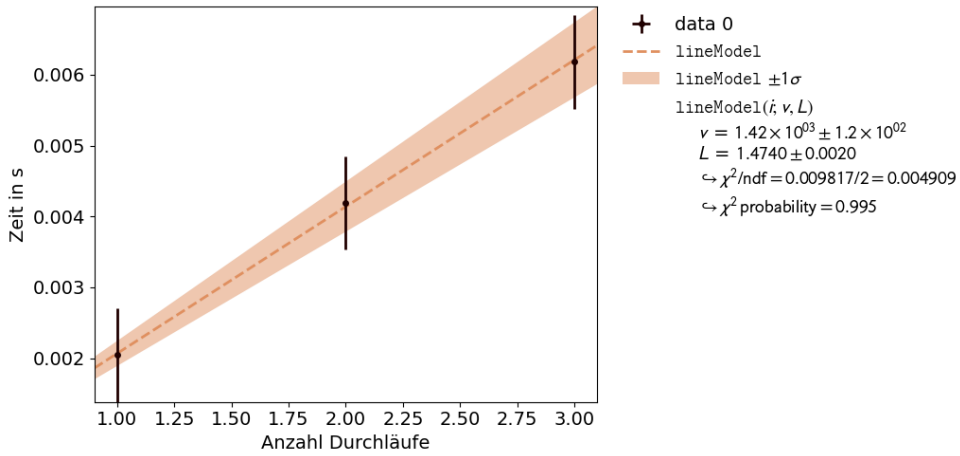
An die Daten wird nun ein Modell der Form $t = i\frac{2L}{v}$ angepasst.



Vermessung eines PVC-Stabes der Länge $L = (1.474 \pm 0.002)m$, Dicke $d = (0.0106 \pm 0.0001)m$ und Masse $m = (0.167 \pm 0.001)kg$:

	Anzahl Durchläufe	Zeit in s
0	1	0.002044
1	2	0.004191
2	3	0.006178

An die Daten wird nun ein Modell der Form $t = i\frac{2L}{v}$ angepasst.



Wie zu erkennen ist, ist die $\chi^2 - Wahrscheinlichkeit$ aller Fits äußerst gut, allerdings ist $\frac{\chi^2}{ndof}$ weit von 1 entfernt. Dies liegt zum Teil an den scheinbar recht groß abgeschätzten Fehlern - der Einfluss der breiteren Maxima auf die früheren Maxima ist wohl kleiner als erwartet. Aber auch die geringe Anzahl der Messpunkte ist problematisch, ist allerdings durch das recht schnelle Verwischen der Maxima bedingt, die ein Ablesen weiterer Werte dahingehend unmöglich machen, dass es zum Ratespiel wird, welches Maxima das richtige nächste ist. Eine messliche Verbesserung unsererseits wäre es dann gewesen für jedes Material mehrerere Messreihen aufzunehmen und immer die geringe Anzahl an bestimmbaren Perioden zu messen und diese dann zusammenzufügen. Diese Variante hätte auch definitiv erlaubt die Auswirkung der größeren Ableseungenauigkeit zu reduzieren.

Aus diesen Werten lassen sich nach $E = \rho v^2$ Werte für E folgern, wobei v aus den Fits entnommen und ρ aus Masse und Volumen der Stäbe bestimmt wurde:

	Material	E-Modul in Gpa
0	Kupfer	124.1+/-3.0
1	Aluminium	73.8+/-2.6
2	PVC	2.6+/-0.4

Es gibt kein wirklichen Vergleichswerte für diese Werte, da die genauen Zusammensetzungen unbekannt sind und andere Stäbe als in den anderen Aufgaben verwendet werden, allerdings liegen die Werte in ähnlichen Größenordnungen. Lediglich der Wert für PVC ist etwas niedrig, was aber leicht durch die geringe Anzahl an Messpunkten und ihre große Unsicherheit zu begründen ist.

3.3 Aufgabe 3: Torsionsmodul

Bestimmen Sie für wenigstens einen der zur Verfügung stehenden Stoffe den Schubmodul G aus der Schwingungsdauer T von geeignet angeregten Torsionsschwingungen.

Hierzu ist ein zylindrischer Stab aus dem zu untersuchenden Stoff jeweils am oberen Ende fest eingespannt und trägt am unteren Ende eine Drehscheibe. Die Drehscheibe wird um einen kleinen Winkel ausgelenkt und führt, nachdem Sie sie loslassen, Drehschwingungen aus. Verwenden Sie die Scheibe ohne, mit zwei und mit vier Zusatzmassen. Bestimmen Sie aus den zugehörigen Schwingungsdauern sowie den berechenbaren Zusatzträgheitsmomenten das Trägheitsmoment der Scheibe.

Führen Sie diese Messung für mindestens eines der zur Verfügung stehenden Materialien durch.

In dieser Aufgabe geht es daum das Schubmodul bzw. Torsionsmodul eines Materials zu untersuchen. Dabei wird die Frequenz der Schwingung eines Torsionspendels gemessen wobei das zu untersuchende Material die Rückstellene Federkraft aufbringt. Dabei wird eine Auslenkung von 10° nicht überschritten, da für größere Winkel die Kleinwinkelnäherungen zunehmend ungenau werden und nichtlineare Terme einen wesentlichen Einfluss auf das Ergebnis nehmen. Außerdem wird sonst der Torsionsstab beschädigt.

Aus dem Schubmodul kann die Torsionssteifigkeit S_t als Produkt mit dem Torsionsträgheitsmoment errechnet werden $S_t = GI_T = G\frac{r^4\pi}{2}$ und damit das Direktionsmoment $D = \frac{S_t}{l}$ mit der Länge l des Stabes. Die Schwingdauer eines Drehpendels ergibt sich aus dem Trägheitsmoment Θ sodass $T = 2\pi\sqrt{\frac{\Theta}{D}}$

Damit ergibt sich für einen Regression das Modell $T = \sqrt{8\frac{\Theta l\pi}{Gr^4}}$. Das Trägheitsmoment ergibt sich dann nach dem Satz von Steinert aus dem Trägheitsmoment der Platte und den Trägheitsmomenten und Abständen der Massen.

[5]:
```
%%m In diesem Versuch wird das Schubmodul von Aluminium über die Periodendauer␣
 ↪eines Torsionspendels bestimmt.
ul = ufloat(0.8, 1e-3)
ur = ufloat(3.97e-3, 0.1e-5)/2

um = ufloat(0.837, 1e-3)
uR2 = ufloat(0.0349, 0.5e-3) #Masseradius
uR = np.sqrt(2)*(ufloat(0.08, 0.5e-3) + 2*uR2)/2 #Masseradius

%%m Folgende Messwert werden für die Schwingdauer T der Drehscheibe für 100␣
 ↪Perioden von dem Aufbau geliefert:

a11 = """
Anzahl extra Massen, Schwingdauer T in s, Unsicherheit in s
0,56.44,1
2,112.18,1
4,146.75,1
"""

n, T, dT = np.genfromtxt(StringIO(a11), delimiter=',')[1:].transpose()
T, dT = T/100, dT/100
```

```
%m Die Länge des Stabes beträgt ($[!ul.n*1e3:.2f!]\pm[!ul.s*1e3:.2f!]$)mm. Der␣
 ↪radius des Stabs beträgt ($[!ur.n*1e3:.2f!]\pm[!ur.s*1e3:.2f!]$)mm. Die␣
 ↪Masse der Zusatzgewichte beträgt laut Datenblatt ($[!um.n:.2f!]\pm[!um.s:.3f!
 ↪]$)Kg mit einem Radius von ($[!uR2.n*1e3:.2f!]\pm[!uR2.s*1e3:.2f!]$)mm. Der␣
 ↪Abstand der Massenschwerpunkte von der Rotationsachse ist nicht einfach zu␣
 ↪messen, weswegen der Abstand zwischen zwei nebeneinanderliegenden Massen␣
 ↪bestimmt wurde und darüber die diagonale des entstehenden Vierecks bestinmmt␣
 ↪wurde unter der Annahme, dass der Aufbau in guter Näherung␣
 ↪rotationssymmetrisch ist. Damit ergibt sich ein Wert von ($[!uR.n*1e3:.2f!
 ↪]\pm[!uR.s*1e3:.2f!]$)mm

mpg = pd.read_csv(StringIO(a11))
display(mpg)

def model(n, θ_platte=0.007, R=uR.n, R2=uR2.n, r=ur.n, l=ul.n, m=um.n, G=1):
    θ = θ_platte + n*m*(R2**2/2 + R**2)
    T = np.sqrt((8*θ*l*pi)/(G*1e9*r**4))
    return T

xy_data = kafe2.XYContainer(x_data=n, y_data=T)
xy_data.add_error(axis='x', err_val=0)
xy_data.add_error(axis='y', err_val=dT)

lin_fit = kafe2.Fit(data=xy_data, model_function=model,minimizer="iminuit")
lin_fit.add_parameter_constraint(name='R',value=uR.n,uncertainty=uR.s)
lin_fit.add_parameter_constraint(name='R2',value=uR2.n,uncertainty=uR2.s)
lin_fit.add_parameter_constraint(name='r',value=ur.n,uncertainty=ur.s)
lin_fit.add_parameter_constraint(name='l',value=ul.n,uncertainty=ul.s)
lin_fit.add_parameter_constraint(name='m',value=um.n,uncertainty=um.s)
lin_fit.do_fit()

p = kafe2.Plot(lin_fit)
p.x_label = 'Anzahl Zusatzgewichte'
p.y_label = 'Periodendauer in s'
p.plot()
p.show()

_,_,_,_,_,_,G = lin_fit.parameter_values
_,_,_,_,_,_,ΔG = lin_fit.parameter_errors
uG = ufloat(G,ΔG)*1e9

%m Damit ergibt sich ein Wert von $G = [!uG.n*1e-9:.2f!]Gpa \pm [!uG.s*1e-9:.2f!
 ↪]Gpa$. Die Gerade passt gut zu den gemessenen Daten und die $\chi^2$-Werte␣
 ↪sind in einem guten Bereich.
```

In diesem Versuch wird das Schubmodul von Aluminium über die Periodendauer eines Torsionspendels bestimmt.

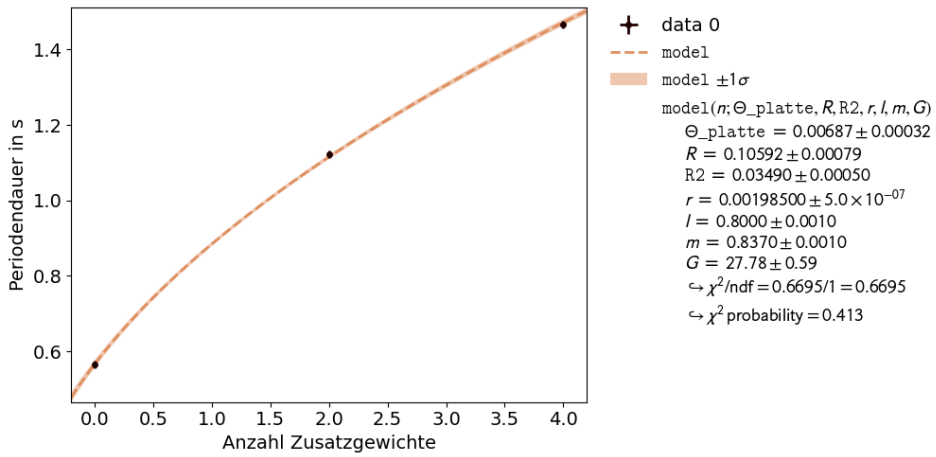
Folgende Messwert werden für die Schwingdauer T der Drehscheibe für 100 Perioden von dem Aufbau geliefert:

Die Länge des Stabes beträgt (800.00 ± 1.00)mm. Der radius des Stabs beträgt (1.98 ± 0.00)mm. Die Masse der Zusatzgewichte beträgt laut Datenblatt (0.84 ± 0.001)Kg mit einem Radius von (34.90 ± 0.50)mm. Der Abstand der Massenschwerpunkte von der Rotationsachse ist nicht einfach zu messen, weswegen der Abstand zwischen zwei nebeneinanderliegenden Massen bestimmt wurde und darüber die diagonale des entstehenden Vierecks bestinmmt wurde unter der Annahme, dass der Aufbau in guter Näherung rotationssymmetrisch ist. Damit ergibt sich ein Wert von (105.92 ± 0.79)mm

	Anzahl extra Massen	Schwingdauer T in s	Unsicherheit in s
0	0	56.44	1
1	2	112.18	1
2	4	146.75	1

```
/tmp/ipykernel_43172/3677204157.py:28: RuntimeWarning: invalid value encountered
in sqrt
  T = np.sqrt((8*0*l*pi)/(G*1e9*r**4))
```

```
Warning: the cost function has been evaluated as infinite. The fit might not
converge correctly.
```



Damit ergibt sich ein Wert von $G = 27.78Gpa \pm 0.59Gpa$. Die Gerade passt gut zu den gemessenen Daten und die χ^2-Werte sind in einem guten Bereich.

[ ]: