Kreisel

December 8, 2023

1 Fakultät für Physik

1.1 Physikalisches Praktikum P1 für Studierende der Physik

Versuch P1-26, 27, 28 (Stand: Oktober 2023)

Raum F1-15

2 Kreisel

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
from kafe2 import MultiFit
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 warnings
warnings.filterwarnings('ignore')
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)
    plt.show()
```

3 Durchführung

3.1 Aufgabe 1: Physik starrer Körper

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

Bei dieser Aufgabe handelt es sich um **Demonstrationsversuche**, die Sie sich gemeinsam mit Ihrem:r Tutor:in während der Praktikumsvorbesprechung überlegen und im Anschluß durchführen können, um sich durch eigene Erfahrung mit der Physik starrer Körper vertraut zu machen. Wir

empfehlen Ihnen, die nötige Zeit für die entsprechenden Versuche einzuräumen und jedem:r Teilnehmer:in die Versuche, bei denen Sie selbst das Studienobjekt sind, selbst durchzuführen.

3.1.1 Aufgabe 1.1: Drehimpulserhaltung

Um persönliche Erfahrungen mit der Drehimpulserhaltung zu sammeln, stehen Ihnen ein Drehschemel und ein Rad mit zwei Griffen und Zugband zur Verfügung. Notieren Sie Ihre Erfahrungen und diskutieren Sie die zugrundeliegenden physikalischen Effekte.

- Nehmen Sie auf dem Drehschemel Platz; halten Sie die Radachse vertikal; werfen Sie den Fahrradkreisel von Hand an, oder lassen Sie sich dabei helfen; Drehen Sie dann den Fahrradkreisel in die Horizontale und halten Sie ihn schließlich an.

  Durch das Anwerfen des Kreisels beginnt die sitzende Person sich in die entgegengesetzte Richtung zu drehen. Wird der Fahrradkreisel in die horizontale Richtung gebracht, stoppt die Drehung der Sitzenden. Dieser Effekt kann über die Präzession oder auch über die Drehimpulserhaltung in der vertikalen Achse erklärt werden.

- Gehen Sie wie oben vor, aber beginnen Sie mit dem Fahrradkreisel in der Horizontalen und drehen Sie ihn dann in die Vertikale.

  Wenn man den Fahrradkreisel in der horizonralen Richtung andreht und dann in die vertikale Position bringt fängt man an sich in die entgegengesetzte Richtung zu drehen wie der Kreisel. Auch das kann wieder über die Drehimpulserhaltung erklärt werden. Wenn man den Kreisel anhält hört man auf selbst auf sich zu drehen, was auch wieder durch Drehimpulserhaltung zu erklären ist.

- Gehen Sie wie oben vor, aber lassen Sie den Fahrradkreisel von einer anderen Person anwerfen, die diesen dann an Sie übergibt.

  Wird der Fahrradkreisel in die horizontale Richtung gebracht, startet eine Drehung der Person in die gleiche Richtung wie der Kreisel vorher. Dieser Effekt kann über wieder über die Drehimpulserhaltung in der vertikalen Achse erklärt werden. Wird der Kreisel gestoppt, endet nicht die Drehung der Person.

- Gehen Sie wie oben vor, aber drehen Sie den Fahrradkreisel langsam von der Vertikalen um 180 Grad erneut in die Vertikale. Dabei zeigt ein Griff des Kreisels mal nach oben und mal nach unten.

  Die Person dreht sich immer in die entgegengesetzte Richtung wie der Kreisel. Je vertikaler die Achse des Kreises ist, desto schneller die Drehung um die eigene Achse.

- Versetzen Sie den Drehschemel (mit oder ohne Fahrradkreisel in Drehung); halten Sie dabei die Arme ausgestreckt; führen Sie dann die Arme an den Körper. Der Effekt verstärkt sich, wenn Sie Gewichte in den Händen halten.,

  Je näher man die Arme an den Körper bringt, desto schneller dreht man sich. Auch dieser Effekt folgt aus der Drehimpulserhaltung.

3.1.2 Aufgabe 1.2: Trägheitstensor

Um Erfahrungen mit dem Trägheitstensor zu sammeln steht Ihnen ein Aufbau zur Verfügung, mit dem Sie entsprechend präparierte Holzkisten in ihren Schwerpunkten aufhängen und in Rotation versetzen können. Notieren Sie Ihre Erfahrungen und diskutieren Sie die zugrundeliegenden physikalischen Effekte.

Es wird ein Holzkästchen mit Ösen an den Seitenflächen über einen Draht mit einem Motor verbunden und so frei im Raum aufgehängt. Anschließend wird die Ausrichtung des Körpers bei Rotation nach Befestigung an den verschiedenen Seiten beobachtet.

Es ist zu erwarten, dass nach dem Dschinbekow-Effekt, lediglich die Rotation um die Hauptachsen extremaler Hauptträgheitsmomente stabil ist und eine Rotation um das mittlere Trägheitsmoment instabil ist, sich also in eine Rotation um eine der anderen Achsen wandelt.

Diese sind erwartet die längste und kürzeste Achse durch den Körper, die normal zu Oberfläche stehen.

Beobachtet wird, dass nur die Rotation um die Achse maximalen Trägheitsmoments stabil ist - also die durch die Seite größter Fläche und kleinsten Durchmesser des dahinter liegenden Körpers - und auch die Rotation um das minimale Trägheitsmoment sich für größere Drehzahlen in eine um das größte Moment wandelt. Dies liegt daran, dass die Drehung um das minimale Moment lediglich semistabil, also nur für kleinere Drehzahlen stabil ist.

Eine weitere Erklärungsidee unsererseits war ein hörbares Klackern innerhalb des Kästchens, welches zu einer Verfälschung der homogen angenommenen Masserverteilung führt und damit die Größe der Trägheitsmomente beeinflusst.

3.1.3 Aufgabe 1.3: Kreiselkompass

Für diesen Versuch steht Ihnen ein kardanisch gelagerter Kreisel auf einer drehbaren Grundfläche zur Verfügung, der sich gegen die Grundfläche kippen lässt. Der innere Kardanrahmen ist durch Arretierfedern fixiert. Beschreiben und diskutieren Sie das Verhalten des Kreisel nachdem Sie zunächst den Kreisel und dann seine Standfläche in Rotation versetzt haben.

Der Versuch Kreiselkompass soll die theoretische Möglichkeit verdeutlichen, einen Kompass auf der Rotation der Erde zu basieren.

Es wird ein Kreisel auf eine sich drehende Scheibe gestellt, um eine Rotation der Erde in einer Geschwindigkeit zu simulieren, bei der trotz Reibungseffekten und in kleineren Zeitspannen, die gleichen Effekte beobachtet werden.

Diese sind erwartet primär eine Ausrichtung der Rotationsachse des Kreisels parallele zu der der “Erde” (der Scheibe, die im weiteren als Erde bezeichnet wird, da sie diese simulieren soll). Auch eine Verkippung des Kreisels vor seiner Rotation relativ zur Achse, um einen anderen Breitengrad zu simulieren, sollte diesen Effekt nicht beeinträchtigen.

Bei der Ausführung wurde dieser Effekt gut beobachtet. Die Kreiselachse strebte deutlich der Erdachse entgegen und erreichte Sie auch wenn ein Breitengerad in der Einstellung nahe >70°

gewählt wurde. Bei Positionen in der Nähe des symbolisierten Äquators wurde die Ausrichtung durch die Federn im Aufbau daran gehindert sich ganz auszurichten.

Der Effekt funktioniert dabei so, dass eine Rotation der Erde eine Kraft in tangentialer Richtung auf den Kreisel ausübt. Diese Kraft erzeugt nach Rechter-Hand-Regel ein Drehmoment in Richtung der Rotationsachse. Dieses wirkt so lange, bis die Rotationsachse des Kreisels ausgerichtet ist.

3.2 Aufgabe 2: Kardanisch gelagerter Kreisel

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

Im Verlauf dieses Versuchs untersuchen Sie den kardanisch gelagerten Kreisel quantitativ. Sie lernen dabei mehrere charakteristische Eigenschaften symmetrischer Kreisel kennen. Sie bestimmen die Trägheitsmomente entlang der Hauptträgheitsachsen des Kreisels und schätzen die Masse des Kreiselrotors ab.

3.2.1 Aufgabe 2.1: Dämpfung

Bestimmen Sie die Dämpfung des Kreisels aus einer Messreihe von mindestens 30 Punkten. Stellen Sie die Kreisfrequenz ω des Kreisels geeignet als Funktion der Zeit t dar.

---

[2]:
```
### Messung der Datenpunkte
%m Es wurde die Dämpfung des Kreisels gemessen, indem er beschleunigt wurde und␣
 ↪dann das Absinken seiner Frequenz mit der Zeit beobachtetet wurde.
mpg = pd.read_csv('A21.csv')
mpg.rename(columns={'t': 'Zeit in s','f': 'Frequenz in Hz'}, inplace=True)
t,f = np.genfromtxt('A21.csv', delimiter=',')[1:].transpose()
Δt = 1E-3 #als kleinste Zeiteinheit des Arduinos
Δf = 1E-3 #Abschätzen
display(mpg)
%m Als Unsicherheiten wurden dabei $[!Δt:.3f!]\,s$ für die Zeitmessung (als␣
 ↪kleinste Zeiteinheit des Arduinos) und $[!Δf:.3f!]\,Hz$ für die␣
 ↪Frequenzmessung (als angenommen größerer Fehler als die Stellen, um die␣
 ↪Mittelung über eine Sekunde zu berücksichtigen) angenommen.
ut = unumpy.uarray(t, Δt)
uw = 2*np.pi*unumpy.uarray(f, Δf)

plt.errorbar(nomv(ut),nomv(uw),xerr=stdv(ut),yerr=stdv(uw))
plt.title("Darstellung der gemessenen Kreisfrequenzen")
plt.xlabel("t in s")
plt.ylabel(r'$\omega$ in Hz')
plt.show()
%m Es ist gut erkennbar, dass einer der Werte um 1000s ein Ausreißer war, der␣
 ↪auftrat, als das Messgerät verrutschte, sodass dieser nun verworfen wird.
index = np.argmin(uw[:1200])
ut = np.delete(ut,index)
uw = np.delete(uw,index)
```

```
### Fitten eines Modells
%m An die Daten wird nun ein phänomenologisches Modell der Form $\omega(t,␣
 ↪\omega_{0}, a, b,c) = \omega_{0}\,e^{-a\,t} + b\,t^{c}$ angepasst.
def slowDownModel(t, _0=200,a=1,b=0,c=2):
    return  _0*np.exp(-1*a*t)+b*t**c

xy_data = kafe2.XYContainer(x_data=nomv(ut), y_data=nomv(uw))
slow_fit = kafe2.Fit(data=xy_data, model_function=slowDownModel)
slow_fit.add_error("x",stdv(ut))
slow_fit.add_error("y",stdv(uw))
slow_fit.do_fit()
p = kafe2.Plot(slow_fit)
p.x_label = 't in s'
p.y_label = 'Frequenz in Hz'
p.plot()
p.show()

### Bewerten des Fits
%m Wie erkennenbar ist, folgen die Daten dem Modell scheinbar sehr gut, die␣
 ↪Linie ist hinter den Punkten nicht mehr wirklich sichtbar, allerdings ist␣
 ↪die $\chi^2$ Wahrscheinlichkeit 0, also sehr schlecht. Dies könnte unter␣
 ↪anderem daran liegen, dass Kafe2 hier sehr viele Daten zu verarbeiten hat,␣
 ↪also viel Potential für unerkannte systematische Fehler bleibt. Es könnte␣
 ↪aber auch daran liegen, dass die Unsicherheiten der Messung zu konservativ␣
 ↪abgeschätzt wurden und die Messungen des Arduinos eigentlich ungenauer waren␣
 ↪(die Mittelung über eine Sekunde also zu möglicherweise größeren␣
 ↪Abweichungen führte, als angenommen).
```

[2]: Es wurde die Dämpfung des Kreisels gemessen, indem er beschleunigt wurde und dann das Absinken seiner Frequenz mit der Zeit beobachtetet wurde.

[2]:

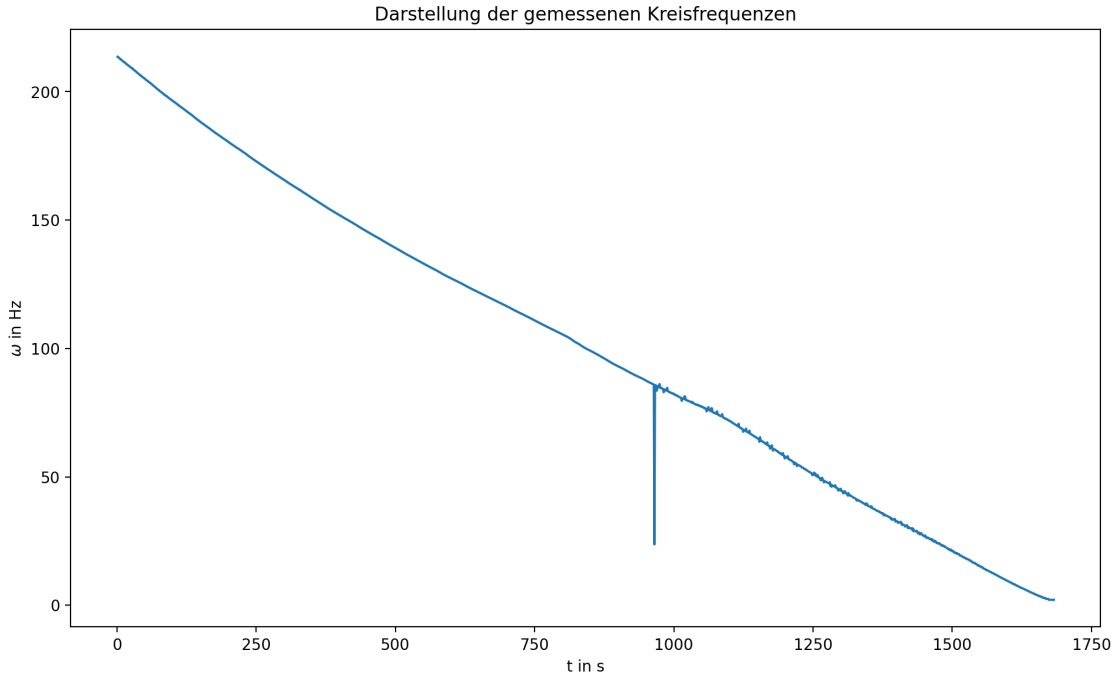
	Zeit in s	Frequenz in Hz
0	1.023632	33.99146
1	2.031244	33.96758
2	3.054725	33.94070
3	4.062286	33.91580
4	5.070041	33.87076
…	…	…
1592	1678.461271	0.33700
1593	1679.464402	0.33700
1594	1680.467536	0.33700
1595	1681.470867	0.33700
1596	1682.474093	0.33700

[1597 rows x 2 columns]

[2]: Als Unsicherheiten wurden dabei 0.001 s für die Zeitmessung (als kleinste Zeiteinheit des Arduinos)

und $0.001\,Hz$ für die Frequenzmessung (als angenommen größerer Fehler als die Stellen, um die Mittelung über eine Sekunde zu berücksichtigen) angenommen.

[2]:



[2]:

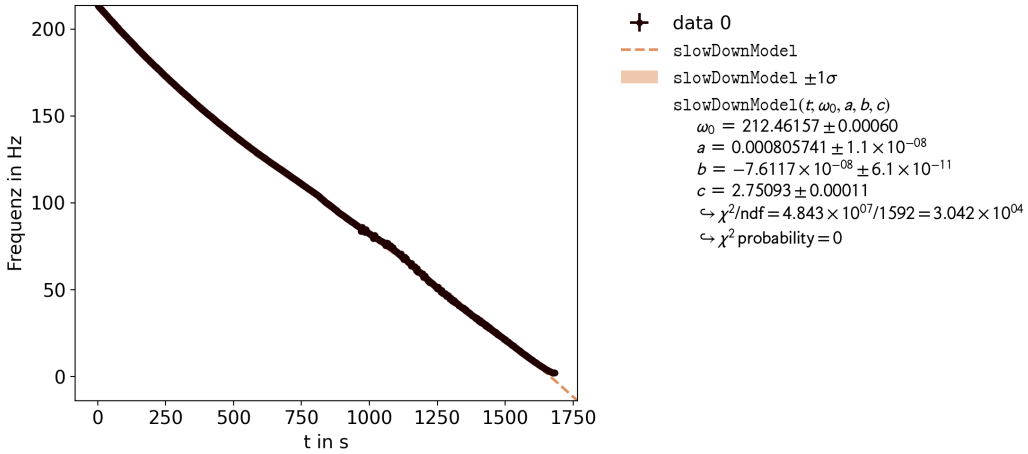
Es ist gut erkennbar, dass einer der Werte um 1000s ein Ausreißer war, der auftrat, als das Messgerät verrutschte, sodass dieser nun verworfen wird.

[2]:

An die Daten wird nun ein phänomenologisches Modell der Form $\omega(t, \omega_0, a, b, c) = \omega_0\, e^{-a\,t} + b\, t^c$ angepasst.

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

[2]:



[2]:

Wie erkennenbar ist, folgen die Daten dem Modell scheinbar sehr gut, die Linie ist hinter den Punkten nicht mehr wirklich sichtbar, allerdings ist die χ^2 Wahrscheinlichkeit 0, also sehr schlecht. Dies könnte unter anderem daran liegen, dass Kafe2 hier sehr viele Daten zu verarbeiten hat, also viel Potential für unerkannte systematische Fehler bleibt. Es könnte aber auch daran liegen, dass die Unsicherheiten der Messung zu konservativ abgeschätzt wurden und die Messungen des Arduinos eigentlich ungenauer waren (die Mittelung über eine Sekunde also zu möglicherweise größeren Abweichungen führte, als angenommen).

3.2.2 Aufgabe 2.2: Nutation

- Bestimmen Sie die Nutationsfrequenz ω_N des Kreisels als Funktion von ω. Nehmen Sie eine Messreihe mit mindestens 30 Messpunkten auf und stellen Sie sie geeignet dar.
- Wiederholen Sie die Messreihe für eine Konfiguration in der Sie zwei Zylindergewichte zusätzlich an den äußeren Kardanrahmen der Kreiselaufhängung geschraubt haben.

---

[3]:

```
### Messung der Datenpunkte
%m <h5> Messung der Nutation ohne Zusatzgewichte </h5>
%m Es wurde die Nutationsfrequenz des Kreisels beobachtet, indem er␣
 ↪beschleunigt und die Nutationsfrequenz nach Schlägen auf den Rahmen mithilfe␣
 ↪der zur Verfügung gestellten Schwanenhälse bei verschiedenen Frequenzen␣
 ↪gemessen wurde.
mpg = pd.read_csv('A22_noMass.csv')
mpg.rename(columns={'f': 'Frequenz in Hz', 'fn': 'Nutationsfrequenz in Hz'},␣
 ↪inplace=True)
Δf = 1E-2 #Abschätzen #aus Anzahl Stellen?
Δf_N = 2E-1 #Abschätzen #aus Anzahl Stellen?
Δf1 = Δf
Δf_N1 = Δf_N
display(mpg)

%m Als Unsicherheiten wurden dabei $[!Δf1:.3f!]\,Hz$ für die Frequenz des␣
 ↪Kreisels (Ungenauigkeit des Ablesens der wandelnden Zahlen und␣
 ↪Beeinträchtigung durch die Schläge) und $[!Δf_N1:.3f!]\,Hz$ für die␣
 ↪Frequenzmessung der Nutation (Schnell wandelnde Zahlen, die teilweise␣
 ↪doppelt so groß waren und dann spontan halbiert und mit den anderen␣
 ↪gemittelt werden mussten) angenommen.
f1, f_N1 = np.genfromtxt('A22_noMass.csv', delimiter=',')[1:].transpose()
uw1 = unumpy.uarray(f1, Δf1) * 2*np.pi
uw_N1 = unumpy.uarray(f_N1, Δf_N1) * 2*np.pi


### Zweite Messreihe mit Gewichten
%m <h5> Messung der Nutation mit Zusatzgewichten </h5>
#Dieses Trägheitsmoment kommt zu Θ_x hinzu
m_N = ufloat(1,1E-3)# Eintragen
r_N = ufloat(10E-2,5E-3) #Eintragen
```

```
l_N = ufloat(14.9E-2,0.2E-2)
 _Z = 2*(1/2*m_N*r_N**2+m_N*l_N**2)
%%m Nun wurden Massen von $[!nomv(m_N):.2f!]\,kg$ angehängt und die gleiche␣
 ↪Messung erneut durchgeführt. Zu den vorher aufgeführten Fehlern kommen nun␣
 ↪eine Unsicherheit von $[!stdv(m_N):.3f!]\, kg$ auf die Massen (nach Angabe␣
 ↪Datenblatt), eine Unsicherheit von $[!stdv(r_N):.3f!]\,m$ auf den Radius von␣
 ↪$[!nomv(r_N):.2f!]\,m$ (Abschätzen der eigenen Genauigkeit beim Messen mit␣
 ↪Hilfe der gegebenen Messinstrumente) und eine Unsicherheit von  $[!stdv(l_N):
 ↪.3f!]\,m$ auf den Abstand vom Schwerpunkt der Gewichte zur Symmetrieachse␣
 ↪des Kreisels von $[!nomv(l_N):.2f!]\,m$ (nach Datenblatt).
mpg = pd.read_csv('A22_withMass.csv')
mpg.rename(columns={'f': 'Frequenz in Hz', 'fn': 'Nutationsfrequenz in Hz'},␣
 ↪inplace=True)
Δf2 = Δf
Δf_N2 = Δf_N
display(mpg)

f2,f_N2 = np.genfromtxt('A22_withMass.csv', delimiter=',')[1:].transpose()
uw2 = unumpy.uarray(f2, Δf2) * 2*np.pi
uw_N2 = unumpy.uarray(f_N2, Δf_N2) * 2*np.pi


### Fitten der Messreihen auf Beschreibungen
%%m <h5> Fitten der Messreihen </h5>
%%m Die Messreihen sollen nun auf Geraden der Form $\omega_N = \omega *m +b$␣
 ↪gefittet werden, wobei $m$ die Steigung beschreibt, die sich aus den␣
 ↪Trägheitsmomenten ergibt: $m =␣
 ↪\frac{\theta_{z}^{\prime}}{\sqrt{\theta_{x}^{\prime}\,\theta_{y}^{\prime}}}.
 ↪$  Die Steigungen der beiden Fits unterscheiden sich, da sich␣
 ↪$\theta_{x}^{\prime}$ durch Anfügen der zusätzlichen Masse ändert.

def nutModel(w,m=1,b=0):
    return w*m+b
# Reihe 1:
xy_data = kafe2.XYContainer(x_data=nomv(uw1), y_data=nomv(uw_N1))
xy_data.label = 'Messung ohne Zusatzgewichte'
line_fit1 = kafe2.Fit(data=xy_data, model_function=nutModel)
line_fit1.add_error("x",stdv(uw1))
line_fit1.add_error("y",stdv(uw_N1))
line_fit1.do_fit()
mN1,bN1 = line_fit1.parameter_values

# Reihe 2:
xy_data = kafe2.XYContainer(x_data=nomv(uw2), y_data=nomv(uw_N2))
xy_data.label = 'Messung mit Zusatzgewichten'
line_fit2 = kafe2.Fit(data=xy_data, model_function=nutModel)
line_fit2.add_error("x",stdv(uw2))
```

```
line_fit2.add_error("y",stdv(uw_N2))
line_fit2.do_fit()
mN2,bN2 = line_fit2.parameter_values

p = kafe2.Plot([line_fit1,line_fit2])
p.x_label = r'$\omega$ in Hz'
p.y_label = 'Nutationsfrequenz in Hz'
p.plot()
p.show()

### Auswertung
%m Wie zu erkennen ist, ist die $\chi^2$ Wahrscheinlichkeit sehr gut,␣
 ↪allerdings ist die $\frac{\chi^2}{ndof}$ besonders für den zweiten Fit recht␣
 ↪weit von 1 entfernt. Ein möglicher Grund hierfür wäre die geringe Anzahl an␣
 ↪Datenpunkten bei zu groß geschätzten Unsicherheiten. Die geringe Anzahl an␣
 ↪Messwerten entspringt vor allem dem Problem, dass bei zu geringen Frequenzen␣
 ↪des Kreisels andere Effekte schwer zu beobachten sind, während zu große␣
 ↪Frequenzen ein mögliches Sicherheitsrisiko darstellen. Eine mögliche Lösung␣
 ↪wäre es hierfür gewesen, den messbaren Bereich mehrmals zu durchmessen, um␣
 ↪mehr Werte aufzunehmen.
```

[3]: Messung der Nutation ohne Zusatzgewichte

[3]: Es wurde die Nutationsfrequenz des Kreisels beobachtet, indem er beschleunigt und die Nutationsfrequenz nach Schlägen auf den Rahmen mithilfe der zur Verfügung gestellten Schwanenhälse bei verschiedenen Frequenzen gemessen wurde.

[3]:

	Frequenz in Hz	Nutationsfrequenz in Hz
0	17.2	8.96
1	16.5	8.86
2	16.0	8.29
3	15.5	8.00
4	15.0	7.81
5	14.5	7.40
6	14.0	7.20
7	13.6	7.00
8	13.0	6.70
9	12.5	6.50
10	12.0	6.25
11	11.5	5.90
12	11.0	5.60
13	10.5	5.40
14	10.0	5.00

[3]: Als Unsicherheiten wurden dabei $0.010\,Hz$ für die Frequenz des Kreisels (Ungenauigkeit des Ablesens der wandelnden Zahlen und Beeinträchtigung durch die Schläge) und $0.200\,Hz$ für die Frequenzmessung der Nutation (Schnell wandelnde Zahlen, die teilweise doppelt so groß waren und dann spontan halbiert und mit den anderen gemittelt werden mussten) angenommen.

[3]:
Messung der Nutation mit Zusatzgewichten

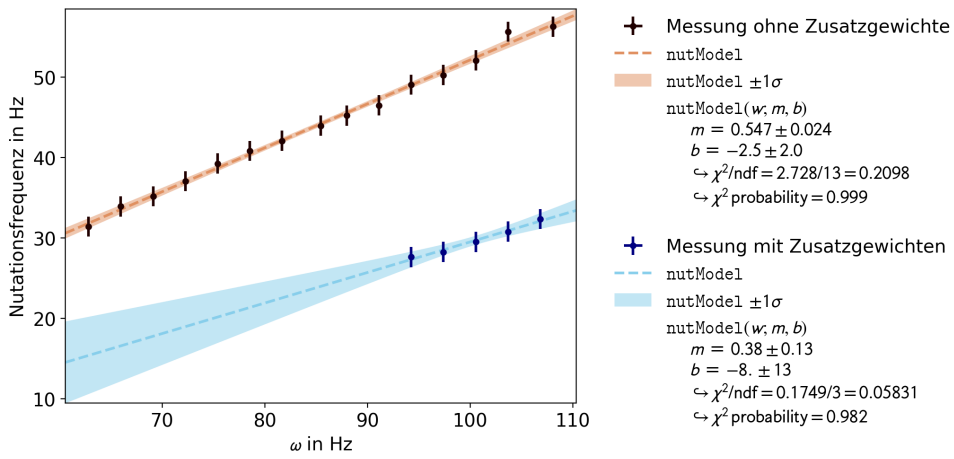
[3]:
Nun wurden Massen von $1.00\,kg$ angehängt und die gleiche Messung erneut durchgeführt. Zu den vorher aufgeführten Fehlern kommen nun eine Unsicherheit von $0.001\,kg$ auf die Massen (nach Angabe Datenblatt), eine Unsicherheit von $0.005\,m$ auf den Radius von $0.10\,m$ (Abschätzen der eigenen Genauigkeit beim Messen mit Hilfe der gegebenen Messinstrumente) und eine Unsicherheit von $0.002\,m$ auf den Abstand vom Schwerpunkt der Gewichte zur Symmetrieachse des Kreisels von $0.15\,m$ (nach Datenblatt).

[3]:

	Frequenz in Hz	Nutationsfrequenz in Hz
0	17.0	5.15
1	16.5	4.90
2	16.0	4.70
3	15.5	4.50
4	15.0	4.40

[3]:
Fitten der Messreihen

[3]:
Die Messreihen sollen nun auf Geraden der Form $\omega_N = \omega * m + b$ gefittet werden, wobei m die Steigung beschreibt, die sich aus den Trägheitsmomenten ergibt: $m = \frac{\theta'_z}{\sqrt{\theta'_x\,\theta'_y}}$. Die Steigungen der beiden Fits unterscheiden sich, da sich θ'_x durch Anfügen der zusätzlichen Masse ändert.

[3]:


[3]:
Wie zu erkennen ist, ist die χ^2 Wahrscheinlichkeit sehr gut, allerdings ist die $\frac{\chi^2}{ndof}$ besonders für den zweiten Fit recht weit von 1 entfernt. Ein möglicher Grund hierfür wäre die geringe Anzahl an Datenpunkten bei zu groß geschätzten Unsicherheiten. Die geringe Anzahl an Messwerten entspringt vor allem dem Problem, dass bei zu geringen Frequenzen des Kreisels andere Effekte schwer zu beobachten sind, während zu große Frequenzen ein mögliches Sicherheitsrisiko darstellen. Eine mögliche Lösung wäre es hierfür gewesen, den messbaren Bereich mehrmals zu durchmessen, um mehr Werte aufzunehmen.

3.2.3 Aufgabe 2.3: Präzession

Bestimmen Sie Präzessionsfrequenz Ω des nutationsfreien, symmetrischen Kreisels als Funktion von ω. Tun Sie dies, indem Sie einen Metallstab als zusätzliches Gewicht in den inneren Kardanrahmen der Kreiselaufhängung schrauben. Nehmen Sie eine Messreihe mit midestens 30 Messpunkten auf und stellen Sie sie geeignet dar.

---

[4]:
```
m_Stab = ufloat(330E-3,1E-3)
l_Stab = ufloat(41.8E-2-25.5E-2,3E-3) #Abstand Schwerpunkt #25.5E-2 41.8E-2
m_G = ufloat(375E-3,1E-3)
s = ufloat(10.91E-2,0.03E-2)

### Messung der Datenpunkte
%m <h5> Messung der Präzession mit verschiedenen Positionen des Gewichts </h5>
%m Es wurde die Präzession des Kreisels beobachtet, indem er beschleunigt und␣
 ↪mit einem mit einem Gewicht versehenen Stab erweitert wurde, der durch die␣
 ↪Gewichtskraft zu einer Kraft auf den Kreisel führte und so erlaubte die␣
 ↪Periodendauer einer Präzession bei verschiedenen Frequenzen und␣
 ↪verschiedenen Abständen des Gewichts zu messen.
mpg = pd.read_csv('A23_1.csv')
mpg.rename(columns={'fv': 'Frequenz in Hz vorher','fn': 'Frequenz in Hz␣
 ↪nachher','T': 'Periodendauer in s','m': 'Angehängte Masse in kg','r':␣
 ↪'Abstand der Masse in m'}, inplace=True)
display(mpg)

##Uncertainties
#Δf = 1E-4 #Abschätzen #aus Anzahl Stellen?
Δf_P = 1E-1 #Abschätzen #aus Anzahl Stellen?
ΔT_P = 4E-1 #Unsicherheit auf die Messung mit der Stoppuhr #abschätzen
Δr_P = 2E-3 #Abschätzen


r1 = l_Stab + s
m1 = m_Stab
d2 = ufloat(0.04335,1E-3)
r2 = (l_Stab*m_Stab+d2*m_G)/(m_Stab+m_G)+s
m2 = m_Stab + m_G
d3 = ufloat(15.5E-2 +2E-2,1E-3)
r3 = (l_Stab*m_Stab+d3*m_G)/(m_Stab+m_G) + s
m3 = m_Stab + m_G
#Sofern kein Stab da: Schwerpunkt Stab und zuzüglich Abstand Rahmen (s) als␣
 ↪Radius, Masse Stab als Masse
#später: Schwerpunkt Stab und Masse m_G daran zuzüglich Abstand Rahmen (s),␣
 ↪Masse Summe der Massen
```

```
%m Der Stab wog $m_{Stab} = ([!nomv(m_Stab):.3!]\pm [!stdv(m_Stab):.3!])\,kg$␣
 ↪und sein Schwerpunkt lag bei $l_{Stab} = ([!nomv(l_Stab):.3!]\pm [!
 ↪stdv(l_Stab):.3!])\,m$ (bestimmt durch Auflage an einer Kante und Abmessen␣
 ↪der Distanz zum Gleichgewichtspunkt). Das anhängbare Gewicht wog $m_{G} = ([!
 ↪nomv(m_G):.3!]\pm [!stdv(m_G):.3!])\,kg$ und seine Distanz zum Schwerpunkt␣
 ↪wurde den jeweiligen Radien zugerechnet, wobei eine homogene␣
 ↪Massenverteilung und so eine Unsicherheit von $[!stdv(d2):.3!]\,m$␣
 ↪angenommen wurde.
%m Als Unsicherheiten der Messung selbst wurden dabei $[!2*np.pi*Δf_P:.2!]\,␣
 ↪Hz$ auf die Drehfrequenz des Kreisels gewählt (da die Anzeige in der␣
 ↪Schnelle nicht genauer abgelesen werden konnte und sich die Frequenz während␣
 ↪der Messung ändert - um diesem Effekt entgegenzuwirken, wurde die Frequenz␣
 ↪vor und nach der Präzession bestimmt und gemittelt) und $[!ΔT_P:.2!]\, s$␣
 ↪für die Zeitmessung (Messung der Dauer einer Periode von Hand mit einer␣
 ↪Stoppuhr, wobei Beginn und Ende einer Periode nur bedingt präzise abgepasst␣
 ↪werden konnten).


f_Pv1, f_Pn1, m_P1, r_P1, T_P1 = np.genfromtxt('A23_1.csv', delimiter=',')[1:].
 ↪transpose()
uw_Pv1 = unumpy.uarray((f_Pv1+f_Pn1)/2, Δf_P) * 2*np.pi
uT_P1 = unumpy.uarray(T_P1, ΔT_P)

f_Pv2, f_Pn2, m_P2, r_P2, T_P2 = np.genfromtxt('A23_2.csv', delimiter=',')[1:].
 ↪transpose()
uw_Pv2 = unumpy.uarray((f_Pv2+f_Pn2)/2, Δf_P) * 2*np.pi
uT_P2= unumpy.uarray(T_P2, ΔT_P)

f_Pv3, f_Pn3, m_P3, r_P3, T_P3 = np.genfromtxt('A23_3.csv', delimiter=',')[1:].
 ↪transpose()
uw_Pv3 = unumpy.uarray((f_Pv3+f_Pn3)/2, Δf_P) * 2*np.pi
uT_P3 = unumpy.uarray(T_P3, ΔT_P)

### Darstellen
plt.
 ↪errorbar(nomv(uw_Pv1),nomv(uT_P1),xerr=stdv(uw_Pv1),yerr=stdv(uT_P1),label=str(np.
 ↪around(nomv(m1),3))+"kg im Abstand von "+str(np.
 ↪around(nomv(r1),3))+"m",fmt=".")
plt.
 ↪errorbar(nomv(uw_Pv2),nomv(uT_P2),xerr=stdv(uw_Pv2),yerr=stdv(uT_P2),label=str(np.
 ↪around(nomv(m2),3))+"kg im Abstand von "+str(np.
 ↪around(nomv(r2),3))+"m",fmt=".")
plt.
 ↪errorbar(nomv(uw_Pv3),nomv(uT_P3),xerr=stdv(uw_Pv3),yerr=stdv(uT_P3),label=str(np.
 ↪around(nomv(m3),3))+"kg im Abstand von "+str(np.
 ↪around(nomv(r3),3))+"m",fmt=".")
```

```
plt.xlabel("Kreisfrequenz in Hz")
plt.ylabel("Periodendauer Präzession in s")
plt.legend()
plt.show()
%m Die Punkte scheinen wie erwartet auf Geraden zu liegen. Es ist zu erkennen,␣
 ↪dass die Unsicherheit auf die Periodendauer recht groß ist, sodass die␣
 ↪Zeitmessung besser anders durchgeführt worden wäre. Ideen hierfür wären␣
 ↪beispielsweise auch hier die Verwendung der Schwanenhälse oder anderer␣
 ↪Lichtschrankensysteme.
```

[4]: Messung der Präzession mit verschiedenen Positionen des Gewichts

[4]: Es wurde die Präzession des Kreisels beobachtet, indem er beschleunigt und mit einem mit einem Gewicht versehenen Stab erweitert wurde, der durch die Gewichtskraft zu einer Kraft auf den Kreisel führte und so erlaubte die Periodendauer einer Präzession bei verschiedenen Frequenzen und verschiedenen Abständen des Gewichts zu messen.

[4]:

	Frequenz in Hz vorher	Frequenz in Hz nachher	Angehängte Masse in kg	\
0	22.60	22.10	0	
1	21.35	20.93	0	
2	20.40	19.90	0	
3	20.55	20.15	0	
4	19.80	19.40	0	
5	19.20	18.83	0	
6	18.60	18.30	0	
7	18.06	17.70	0	
8	17.45	17.10	0	
9	16.75	16.43	0	

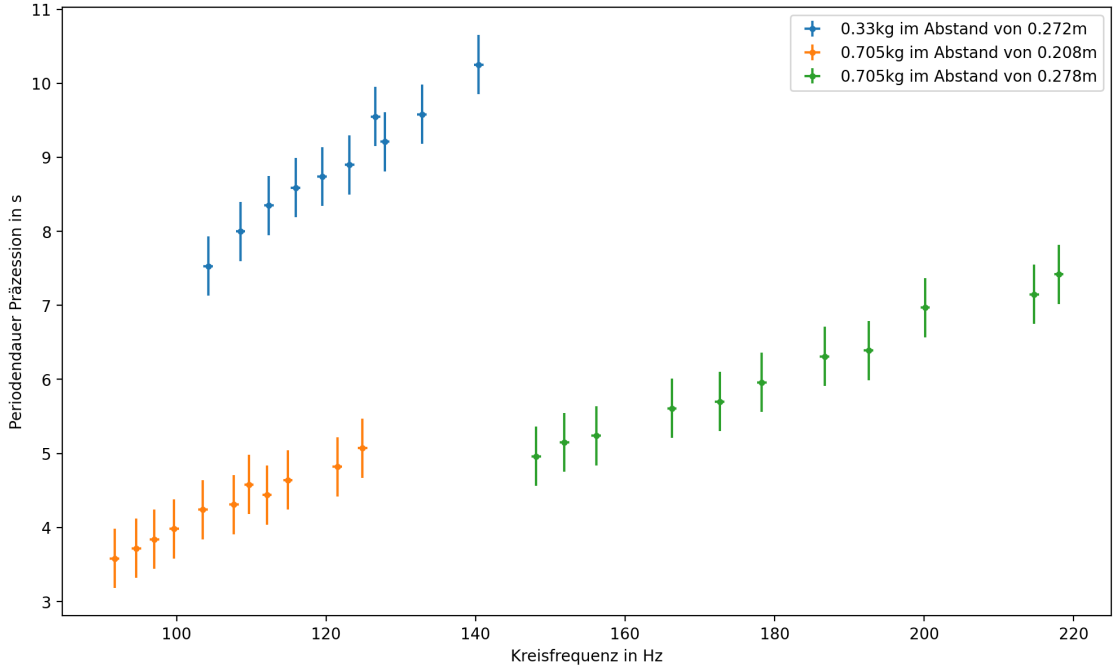
	Abstand der Masse in m	Periodendauer in s
0	0	10.25
1	0	9.58
2	0	9.55
3	0	9.21
4	0	8.90
5	0	8.74
6	0	8.59
7	0	8.35
8	0	8.00
9	0	7.53

[4]: Der Stab wog $m_{Stab} = (0.33 \pm 0.001)\,kg$ und sein Schwerpunkt lag bei $l_{Stab} = (0.163 \pm 0.003)\,m$ (bestimmt durch Auflage an einer Kante und Abmessen der Distanz zum Gleichgewichtspunkt). Das anhängbare Gewicht wog $m_G = (0.375 \pm 0.001)\,kg$ und seine Distanz zum Schwerpunkt wurde den jeweiligen Radien zugerechnet, wobei eine homogene Massenverteilung und so eine Unsicherheit von $0.001\,m$ angenommen wurde.

[4]: Als Unsicherheiten der Messung selbst wurden dabei $0.63\,Hz$ auf die Drehfrequenz des Kreisels

gewählt (da die Anzeige in der Schnelle nicht genauer abgelesen werden konnte und sich die Frequenz während der Messung ändert - um diesem Effekt entgegenzuwirken, wurde die Frequenz vor und nach der Präzession bestimmt und gemittelt) und $0.4\,s$ für die Zeitmessung (Messung der Dauer einer Periode von Hand mit einer Stoppuhr, wobei Beginn und Ende einer Periode nur bedingt präzise abgepasst werden konnten).

[4]:



[4]:

Die Punkte scheinen wie erwartet auf Geraden zu liegen. Es ist zu erkennen, dass die Unsicherheit auf die Periodendauer recht groß ist, sodass die Zeitmessung besser anders durchgeführt worden wäre. Ideen hierfür wären beispielsweise auch hier die Verwendung der Schwanenhälse oder anderer Lichtschrankensysteme.

3.2.4 Aufgabe 2.4: Trägheitsmomente entlang der Hauptträgheitsachsen

- Berechnen Sie aus den Messungen die Sie im Rahmen der **Aufgaben 2.2** und **2.3** durchgeführt haben, die Trägheitsmomente θ_x, θ_y und θ_z entlang der Hauptträgheitsachsen.
- Schätzen Sie aus Ihren Berechnungen die Masse M des Kreiselrotors ab.

---

[6]:

```
###Trägheitsmomentbestimmung durch Multifit
%m <h5> Bestimmung der einzelnen Trägheiten durch einen Multifit </h5>
%m Es werden die einzelnen Funktionen durch ihre entsprechenden Zusammenhänge␣
 ↪mit den Trägheitsmomenten ersetzt und gemeinsam für diese gefittet.

## Defining the Models
def praezModelm1(w,m1,r1,k, _z=1,b1=1):
    return 2*np.pi * _z * w /(m1*k*r1)+b1
```

```
def praezModelm2(w,m2,r2,k, _z=1,b2=1):
    return 2*np.pi * _z * w /(m2*k*r2) +b2

def praezModelm3(w,m3,r3,k, _z=1,b3=1):
    return 2*np.pi * _z * w /(m3*k*r3) +b3

def noMNutModel(w, _z=1,  _x=1,  _y=1):
    return w *  _z/np.sqrt( _x *  _y)

def MNutModel(w,  _Z,  _z=1,  _x=1,  _y=1):
    return w *  _z/np.sqrt(( _x+ _Z)*  _y)


## Fitting Procedure
fitlist = []
for i in range(0,5):
    fitlist.append([])

#Fitting the Präzession
xy_data = kafe2.XYContainer(x_data=nomv(uw_Pv1), y_data=nomv(uT_P1))
xy_data.add_error(axis='x', err_val=stdv(uT_P1))
xy_data.add_error(axis='y', err_val=stdv(uT_P1))
fitlist[0] = kafe2.Fit(data=xy_data, model_function=praezModelm1)
fitlist[0].add_parameter_constraint(name='m1',   value=nomv(m1),  ␣
 ↪uncertainty=stdv(m1))
fitlist[0].add_parameter_constraint(name='k',   value=9.81,   uncertainty=0.01)
fitlist[0].add_parameter_constraint(name='r1',   value=nomv(r1),  ␣
 ↪uncertainty=stdv(r1))
fitlist[0].data_container.axis_labels = ("Frequenz in Hz", "Periodendauer in s")
fitlist[0].data_container.label = "Präzession bei "+ str(np.
 ↪around(nomv(m1),3))+"kg im Abstand von "+str(np.around(nomv(r1),3))+"m"
fitlist[0].limit_parameter(" _z",1E-12,None)
fitlist[0].model_label = "Parametrization"


xy_data = kafe2.XYContainer(x_data=nomv(uw_Pv2), y_data=nomv(uT_P2))
xy_data.add_error(axis='x', err_val=stdv(uT_P2))
xy_data.add_error(axis='y', err_val=stdv(uT_P2))
fitlist[1] = kafe2.Fit(data=xy_data, model_function=praezModelm2)
fitlist[1].add_parameter_constraint(name='m2',   value=nomv(m2),  ␣
 ↪uncertainty=stdv(m2))
fitlist[1].add_parameter_constraint(name='k',   value=9.81,   uncertainty=0.01)
fitlist[1].add_parameter_constraint(name='r2',   value=nomv(r2),  ␣
 ↪uncertainty=stdv(r2))
fitlist[1].data_container.axis_labels = ("Frequenz in Hz", "Periodendauer in s")
```

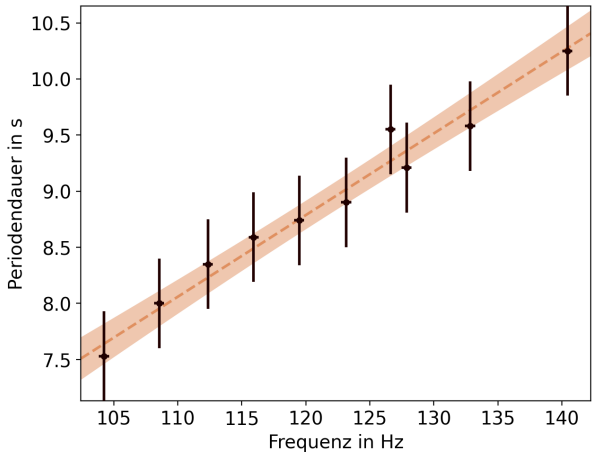
```
fitlist[1].data_container.label = "Präzession bei "+ str(np.
 ↪around(nomv(m2),3))+"kg im Abstand von "+str(np.around(nomv(r2),3))+"m"
fitlist[1].model_label = "Parametrization"

xy_data = kafe2.XYContainer(x_data=nomv(uw_Pv3), y_data=nomv(uT_P3))
xy_data.add_error(axis='x', err_val=stdv(uT_P3))
xy_data.add_error(axis='y', err_val=stdv(uT_P3))
fitlist[2] = kafe2.Fit(data=xy_data, model_function=praezModelm3)
fitlist[2].add_parameter_constraint(name='m3',   value=nomv(m3),  ␣
 ↪uncertainty=stdv(m3))
fitlist[2].add_parameter_constraint(name='k',   value=9.81,   uncertainty=0.01)
fitlist[2].add_parameter_constraint(name='r3',   value=nomv(r3),  ␣
 ↪uncertainty=stdv(r3))
fitlist[2].data_container.axis_labels = ("Frequenz in Hz", "Periodendauer in s")
fitlist[2].data_container.label = "Präzession bei "+ str(np.
 ↪around(nomv(m3),3))+"kg im Abstand von "+str(np.around(nomv(r3),3))+"m"
fitlist[2].model_label = "Parametrization"


# Nutation Fit without added
xy_data = kafe2.XYContainer(x_data=nomv(uw1), y_data=nomv(uw_N1)) #change␣
 ↪number of data
xy_data.label = 'Messung ohne Zusatzgewichte'
xy_data.add_error(axis='x', err_val=stdv(uw1))
xy_data.add_error(axis='y', err_val=stdv(uw_N1))
fitlist[3] = kafe2.Fit(data=xy_data, model_function=noMNutModel)
fitlist[3].data_container.axis_labels = ("Frequenz in Hz", "Nutationsfrequenz␣
 ↪in Hz")

# Nutation Fit with added
xy_data = kafe2.XYContainer(x_data=nomv(uw2), y_data=nomv(uw_N2)) #change␣
 ↪number of data
xy_data.label = 'Messung mit Zusatzgewichten'
xy_data.add_error(axis='x', err_val=stdv(uw2))
xy_data.add_error(axis='y', err_val=stdv(uw_N2))
fitlist[4] = kafe2.Fit(data=xy_data, model_function=MNutModel)
fitlist[4].data_container.axis_labels = ("Frequenz in Hz", "Nutationsfrequenz␣
 ↪in Hz")
fitlist[4].add_parameter_constraint(name=' _Z',   value=nomv( _Z),  ␣
 ↪uncertainty=stdv( _Z))


# Construct a MultiFit object
multi_fit = MultiFit(fit_list=fitlist, minimizer='iminuit')
multi_fit.limit_parameter(" _x",1E-12,None)
multi_fit.limit_parameter(" _y",1E-12,None)
```

```
multi_fit.limit_parameter(" _Z",1E-12,None)

# Do the fit
multi_fit.do_fit()

# Darstellung
plot = kafe2.Plot(multi_fit, separate_figures=True)
plot.plot()
plot.show()

nm1,nr1,nk,n _z,nb1,nm2,nr2,nb2,nm3,nr3,nb3,n _x,n _y,n _Z = multi_fit.
 ↪parameter_values
Δm1,Δr1,Δk,Δ _z,Δb1,Δm2,Δr2,Δb2,Δm3,Δr3,Δb3,Δ _x,Δ _y,Δ _Z = multi_fit.
 ↪parameter_errors

%m Die $\chi^2$ Wahrscheinlichkeit ist 1 und $\frac{\chi^2}{ndof}$ nicht sehr␣
 ↪weit von 1 entfernt, sodass der Fit als gut betrachtet wird. Auffällig sind␣
 ↪trotzdem die verhältnismäßig großen Fehler. Diese sind vermutlich besonders␣
 ↪in der Zeitmessung und den schlecht ablesbaren Nutationsfrequenzen begründet.

%m Es ergaben sich folgende Werte für die Trägheitsmomente: $\theta_x = ([!n _x:
 ↪.3f!] \pm [!Δ _x:.3f!] )\,kg\,m^2$,  $\theta_y = ([!n _y:.3f!] \pm [!Δ _y:.3f!
 ↪] )\,kg\,m^2$ und  $\theta_z = ([!n _z:.3f!] \pm [!Δ _z:.3f!] )\,kg\,m^2$.␣
 ↪Diese Werte sind durchaus realistisch und.

### Berechnung von M_Rotor
d_Rotor = ufloat(13.5E-2,0.01E-2)
n _z = ufloat(n _z,Δ _z)
M_Rotor = 8*n _z*d_Rotor**(-2)

%m <h5> Berechnung des Rotorgewichts aus den Trägheitsmomenten </h5>
%m Aus den im Fit bestimmten Trägheitsmomenten wird nun dan Gewicht des Rotors␣
 ↪zu $M_{Rotor} = ([!nomv(M_Rotor):.2f!]\pm [!stdv(M_Rotor):.2f!])\,kg$␣
 ↪bestimmt. Dabei geht auch noch der Durchmesser des Rotors von $d_{Rotor} =␣
 ↪([!nomv(d_Rotor):.2f!]\pm [!stdv(d_Rotor):.4f!])\,m$ mit ein.
%m Dieses Ergebnis klingt durchaus realistisch, würde bei diesem Gewicht weder␣
 ↪der Tisch kaputt gehen, noch eine zu kleine Masse Rotationsenergie aufnehmen.
 ↪ Die Größe des Fehlers ist für die gegebenen Messmittel erstaunlich klein.
```

[6]: Bestimmung der einzelnen Trägheiten durch einen Multifit

[6]: Es werden die einzelnen Funktionen durch ihre entsprechenden Zusammenhänge mit den Trägheitsmomenten ersetzt und gemeinsam für diese gefittet.

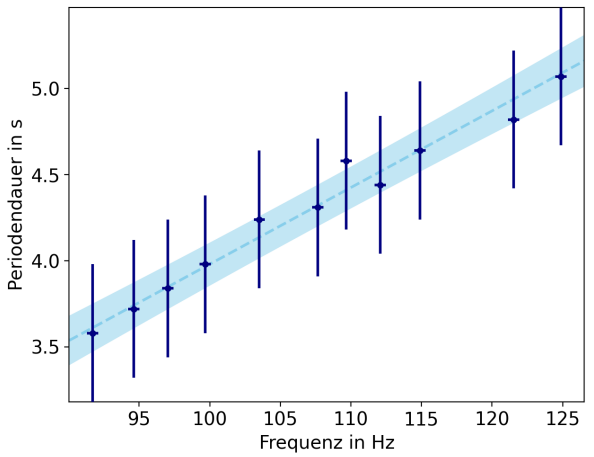
[6]:



Präzession bei 0.33kg im Abstand von 0.272m
Parametrization
Parametrization $\pm 1\sigma$

praezModelm1(w; m1, r1, k, θ_z, b1)
m1 = 0.3300 ± 0.0010
r1 = 0.2721 ± 0.0030
k = 9.8100 ± 0.0058
θ_z = 0.0102 ± 0.0011
b1 = 0.05 ± 0.92
↪ χ^2/ndf = $1.085/8 = 0.1356$
↪ χ^2 probability = 0.998
↪ global χ^2/ndf = $6.931/38 = 0.1824$
↪ global χ^2 probability = 1

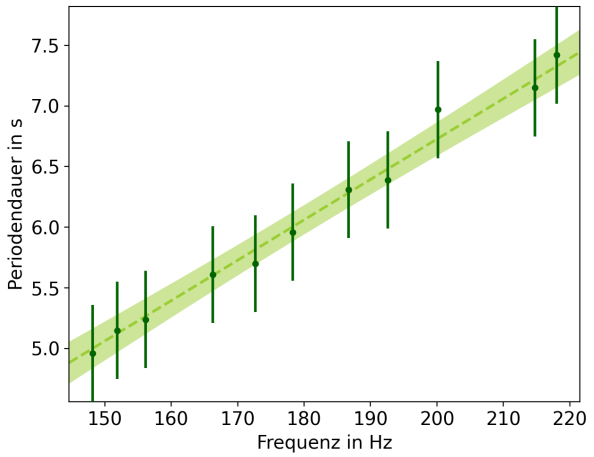
[6]:



Präzession bei 0.705kg im Abstand von 0.208m
Parametrization
Parametrization $\pm 1\sigma$

praezModelm2(w; m2, r2, k, θ_z, b2)
m2 = 0.7050 ± 0.0014
r2 = 0.2085 ± 0.0015
k = 9.8100 ± 0.0058
θ_z = 0.0102 ± 0.0011
b2 = -0.47 ± 0.51
↪ χ^2/ndf = $0.3828/9 = 0.04254$
↪ χ^2 probability = 1
↪ global χ^2/ndf = $6.931/38 = 0.1824$
↪ global χ^2 probability = 1

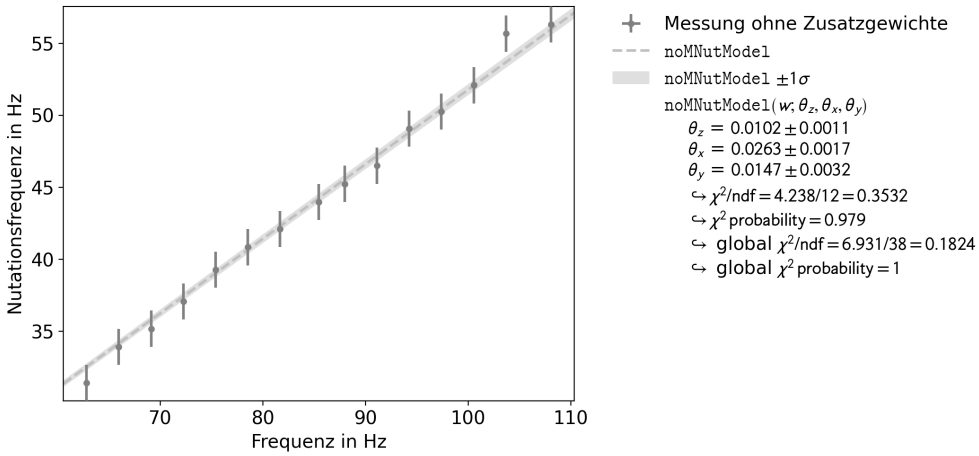
[6]:



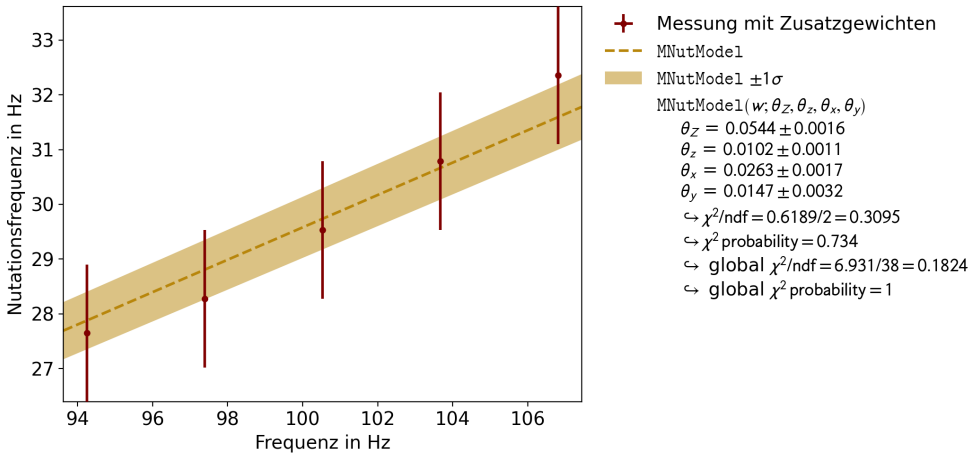
Präzession bei 0.705kg im Abstand von 0.278m
Parametrization
Parametrization $\pm 1\sigma$

praezModelm3(w; m3, r3, k, θ_z, b3)
m3 = 0.7050 ± 0.0014
r3 = 0.2785 ± 0.0015
k = 9.8100 ± 0.0058
θ_z = 0.0102 ± 0.0011
b3 = 0.07 ± 0.63
↪ χ^2/ndf = $0.6059/9 = 0.06732$
↪ χ^2 probability = 1
↪ global χ^2/ndf = $6.931/38 = 0.1824$
↪ global χ^2 probability = 1

[6]:



[6]:



[6]: Die χ^2 Wahrscheinlichkeit ist 1 und $\frac{\chi^2}{ndof}$ nicht sehr weit von 1 entfernt, sodass der Fit als gut betrachtet wird. Auffällig sind trotzdem die verhältnismäßig großen Fehler. Diese sind vermutlich besonders in der Zeitmessung und den schlecht ablesbaren Nutationsfrequenzen begründet.

[6]: Es ergaben sich folgende Werte für die Trägheitsmomente: $\theta_x = (0.026 \pm 0.002)\,kg\,m^2$, $\theta_y = (0.015 \pm 0.003)\,kg\,m^2$ und $\theta_z = (0.010 \pm 0.001)\,kg\,m^2$. Diese Werte sind #TODO

[6]: Berechnung des Rotorgewichts aus den Trägheitsmomenten

[6]: Aus den im Fit bestimmten Trägheitsmomenten wird nun dan Gewicht des Rotors zu $M_{Rotor} = (4.48 \pm 0.46)\,kg$ bestimmt. Dabei geht auch noch der Durchmesser des Rotors von $d_{Rotor} = (0.14 \pm 0.0001)\,m$ mit ein.

[6]: Dieses Ergebnis klingt durchaus realistisch, würde bei diesem Gewicht weder der Tisch kaputt gehen, noch eine zu kleine Masse Rotationsenergie aufnehmen. Die Größe des Fehlers ist für die gegebenen Messmittel erstaunlich klein.

[0]: