WS19/20, PAP2.1, Versuch 213: Kreisel

Praktikanten:
Gerasimov, V. & Reiter, L.

Betreuer:
Neitzel, C.

Versuchsdurchführung:
22. Oktober, 2019

Inhaltsverzeichnis

1 Einführung[1]

Dieser Versuch befasst sich mit den grundlegenden physikalischen Eigenschaften von Kreiseln und ihrer Vermessung. Zuerst wollen wir in einem Vorversuch qualitativ das Verhalten eines Kreisels untersuchen. Danach messen wir die Reibungsverluste des Kreisels und bestimmen die Dämpfungskonstante $\boldsymbol{k}$ und Halbwertszeit $\boldsymbol{T_{1/2}}$. Wir bestimmen aus der Präzessionsfrequenz eines schweren Kreisels sein Trägheitsmoment $\boldsymbol{I_z}$ um die Figurenachse $\vec{\boldsymbol{Z}}$. Das Trägheitsmoment $\boldsymbol{I_x}$ senkrecht zur Figurenachse bestimmt wir aus der Größe und der Richtung der Umlaufsgeschwindigkeit $\boldsymbol{\Omega}$ der momentanen Drehachse um die Figurenachse des Kreisels und zusätlich ein zweites Mal aus der gemessenen Nutationsfrequenz $\boldsymbol{f_N}$.

- Ein starrer Körper, der um einen festen Punkt rotiert, stellt einen Kreisel dar. Sind genau zwei Hauptträgheitsmomente identisch $(\boldsymbol{I_x = I_y})$, so wird der Kreisel als symmetrisch bezeichnet. Wird der Kreisel im Schwerpunkt

[1] Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

unterstützt so heißt der Kreisel kräftefrei. In diesem Fall wirken keine äußeren Drehmomente und der Drehimpuls ist räumlich und zeitlich konstant.

- Die allgemeine Bewegung eines kräftefreien Kreisels stellt eine Nutationsbewegung dar. Dabei fuhrt die Figurenachse eine Eigendrehung mit $\boldsymbol{\omega_F}$ durch und rotiert mit der Nutationsfrequenz $\boldsymbol{\omega_N}$ gleichzeitig um die raumfeste Drehimpulsachse. Die Winkelgeschwindigkeit ist nicht konstant sondern bewegt sich mit $\boldsymbol{\Omega}$ um die Figurenachse $\vec{\boldsymbol{Z}}$. Diese Bewegung kann mit Hilfe einer farbigen Sektorscheibe beobachtet werden. Die Bewegungen der charakteristischen Kreiselachsen kann man sich durch ein Abrollen eines körperfesten Kegels auf einen raumfesten Kegels veranschaulichen.
- Liegt der Auflagepunkt des Kreisels nicht im Schwerpunkt, so heißt der Kreisel schwerer Kreisel. In diesem Fall übt die Gewichtskraft ein Drehmoment aus, das zu einer Präzession führt. Dabei bewegt sich der Drehimpuls mit der Frequenz $\boldsymbol{\omega_P}$ auf einem Kegelmantel um die Richtung der Gewichtskraft.

2 Versuchsaufbau[2], Literaturwerte & Vorbereitung

- Stahlkugel mit Aluminiumstab (Masse $\boldsymbol{m_K = 4.164(1)\ kg}$ incl. Stab, Kugelradius $\boldsymbol{r_K = 5.08(1)\ cm}$) als Kreisel gelagert in einer Luftkissenpfanne
- 2 Gewichte ($\boldsymbol{r_a = 0,725(1)\ cm}$, $\boldsymbol{r_i = 0,325(1)\ cm}$, $\boldsymbol{h_G = 1,10(1)\ cm}$, $\boldsymbol{m_G = 9,85(1)\ g}$)
- Farbscheibe, Scheibe mit konzentrischen Ringen
- Stroboskop, optischer Drehzahlmesser
- Stoppuhr
- Motor mit Netzgerät
- Gyroskop zur Demonstration der Kreiseleigenschaften
- Literaturwert für Erdbeschleunigung[3]in Heidelberg, Baden Württemberg, Deutschland: $\boldsymbol{g = 9.80984(2)\ m\ s^2}$

3 Vorversuch: qualitative Beobachtung des Verhaltens eines Kreisels

3.1 Durchführung[2]

Der Vorversuch soll uns mit dem Kreisel vertraut machen und uns die später genauer zu untersuchenden Erscheinungen qualitativ demonstrieren.

(a) Wir öffnen das Druckluftventil an der Wand, stecken die Scheibe mit den Farbsektoren nach oben auf den Stab und balancieren die Scheibe wie zuvor beschrieben aus, sodass der Kreisel kräftefrei wird. Der Kreisel wird auf einige Umdrehungen pro Sekunde beschleunigt und wir beobachten die Reaktion des Kreisels, wenn der Metallring des Kugellagers am Stabende mit einem Finger zur Seite gedrückt wird. Alle Beobachtungen werden erläutert.

(b) Als nächstes stellen wir eine Nutationsbewegung ein, indem dem Stab ein leichter, seitlicher Stoß erteilt wird. Die Farbscheibe ist zu beobachten: In der mischfarbigen Fläche der sich drehenden Scheibe soll ein Punkt zu erkennen sein, an dem eine "reine, unvermischte"Farbe erscheint. An dieser Stelle ändert sich die Farbe gemäß der Farbanordnung auf der Sektorscheibe. Dieser Punkt stellt den um die Figurenachse $\vec{\boldsymbol{Z}}$ wandernden Ort der momentanen Drehachse $\vec{\boldsymbol{\omega}}$ dar.

Wir drehen die Scheibe um, sodass die Seite mit den farbigen Ringen nach oben zeigt und wiederholen den Versuch. Wenn beim Anschlagen des Kreisels einen Nutationskegel erreicht wird, der gerade in einem der Farbringe verläuft, ändert sich die Farbe am Ort der momentanen Drehachse nicht, d.h. $\vec{\boldsymbol{\omega}}$ läuft auf einem Kreis um $\vec{\boldsymbol{Z}}$.

(c) Wir legen zusätzlich die Scheibe mit den konzentrischen Kreisen auf die Farbscheibe und wählen zunächst die Seite der Scheibe, bei der der Mittelpunkt der Kreise seitlich gegen die Aufnahmeachse verschoben ist. Liegt keine Nutation vor ($\vec{\boldsymbol{\omega}} || \vec{\boldsymbol{Z}}$), so erkennt man ein System konzentrischer, verwaschener Kreise um den Stab. D.h. der Mittelpunkt des Kreissystems zeigt die Drehachse an. Danach drehen wir die Scheibe um und versetzen den Kreisel in Rotation. Durch einen seitlichen Stoß werden wieder die drei Kreiselachsen getrennt. Warum markiert

[2]Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213
[3]Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.2 Stand 06/2016, Versuch 14

jetzt der Mittelpunkt der verwaschenen Kreise die Drehimpulsachse, die räumlich stehen bleibt?

Wir Bringen ein Zusatzgewicht am Stab an und wiederholen den Versuch. Die Drehimpulsachse sollte nun ein Präzessionsbewegung durchführen.

(d) Ohne zusätzliche Farbscheibe richtet sich der Stab immer auf, d.h. der Schwerpunkt liegt unterhalb der Kugelmitte. Mit einem Zusatzgewicht am Ende des Stabes fällt dagegen der Kreisel um. In diesem Fall liegt der Schwerpunkt oberhalb der Kugelmitte. Wir versetzen in beiden Fällen den Kreisel in Drehung, lassen den Stab aus einer nichtvertikalen Stellung los und beobachten die Drehrichtung der Präzession. Danach wir die Drehrichtung des Kreisels geändert und der Versuch wiederholt.

3.2 Auswertung

(a) Ohne angebrachte Farbscheibe ist der Kreisel in der Luftkissenpfanne zuerst nicht kräftefrei. D.h. es kommt zur Präzession, einer zeitlichen Änderung der Richtung des Drehimpulses und der Figurenachse, weil auf den Kreisel ein äußeres Drehmoment $\vec{M}$ wirkt.

Nachdem die Farbscheibe mit den Sektoren angebracht und ausbalanciert wurde, verschwindet das äußere Drehmoment und der nun kräftefreie Kreisel behält seinen Drehimpuls bei (im Falle von vernachlässigbarer Reibung). Die Figurenachse $\vec{Z}$ änder sichjetzt nur noch leicht aufgrund der leichten restlichen Nutation. Der Kreisel behält nach dem Drücken seine Drehachse bei und die Farbsektoren verschwimmen zu einer Mischfarbe. Beim Drücken des Kugellagers spürt man einen deutlichen Widerstand, der mit zunehmender Drehfrequenz des Kreisels auch zunimmt.

(b) Nachdem sich die Nutationsbewegung eingestellt hat, lässt sich auf dem Farbsektorenring ein Kreis beobachten, der abwechselnd jeweils eine der 3 ursprünglichen Farben annimmt.

Wenn die Scheibe mit den Farbringen angebracht wird, so entsteht ein einfarbiger Punkt in der Mitte, der seine Farbe mit der Zeit nicht ändert, aber die Farbe ändert sich für verschiedene Nutationswinkel.

(c) Beide Scheiben mit den konzentrischen Kreisen erzeugen unterschiedlich stark verschwommene Ringsysteme die sich zeitlich nicht ändern, aber vom Nutationswinkel abhängen. Ein einfarbiger Punkt in der Mitte zeigt ungefähr in die ursprüngliche Richtung der Drehachse (bevor die Nutation eingeleitet wurde) und die Figurenachse $\vec{Z}$ kreist um ihn (Nutation). Diese Punkt zeigt uns die Richtung der momentanen Drehachse $\vec{\omega}$ an.

Sobald ein Zusatzgewicht am Stab angebracht wird fängt der zuvor räumlich feste einfarbige Punkt an zu wandern. Seine Bewegung entspricht nun der Präzessionsbewegung des Kreisels, da die Figurenachse jetzt eine Kombination aus Nutations- und Präzessionsbewegung vollführt.

(d) Ohne Farbscheibe und Zusatzgewichte sind die Richtungen der Eigendrehung und Präzessionsdrehung entgegengesetzt und mit Zusatzgewicht zeigen beide in die gleiche Richtung. Bei Änderung der Eigendrehung in die andere Richtung ändern sich auch Nutations- und Präzessionsdrehrichtung. Wie erwartet bleiben die Beziehungen zwischen den betrachteten Drehrichtungsvektoren erhalten, wenn alle Vektoren das Vorzeichen ändern.

4 Messung der Reibungsverluste & Bestimmung der Dämpfungskonstante und Halbwertszeit

4.1 Durchführung[4]

Wir bringen wieder die Sektorenscheibe an und überprüfen ob der Kreisel kräftefrei ist. Zusätzlich montieren wir beide Gewichte am Stabende. Der Kreisel wird mit Hilfe des Motors bei senkrechter Achse auf eine Anfangsfrequenz $f_0 \approx 600 - 700\, min^{-1}$ beschleunigt. Alle 2 Minuten wird die Drehfrequenz des Kreisels über einen Zeitraum von 12 Minuten gemessen. Die Drehfrequenz f zur jeweiligen Zeit t wird notiert.

4.2 Messergebnisse

Messdaten wurden dem Versuchsprotokoll (22. Oktober, 2019) entnommen und in Tabelle 1 übertragen.

[4] Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

Tabelle 1: Messung der Drehfrequenz

Zeit t $[s]$	Drehfrequenz f $[min^{-1}]$
0 ± 2	648 ± 10
120 ± 2	600 ± 10
240 ± 2	552 ± 10
360 ± 2	515 ± 10
480 ± 2	470 ± 10
600 ± 2	437 ± 10
720 ± 2	403 ± 10

[1] sowohl Δt als auch Δf sind grob abgeschätzt

4.3 Kurvenanpassung mit Python

4.3.1 Source Code & Input

Wir drücken zuerst f über die entsprechende Kreisfrequenz ω aus:

$$\omega = 2\pi f \tag{1}$$

$$\Delta\omega = 2\pi(\Delta f) \tag{2}$$

Wir gehen davon aus, dass die die Kreisfrequenz exponentiell mit der Zeit abnimmt. Daher ist unser funktionales Modell für die Ausgleichungsrechnung wie folgt:

$$\boxed{\omega = \omega_0 e^{-kt}} \tag{3}$$

So sieht unsere Python-Implementierung aus:

Header:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import chi2
from decimal import Decimal

def format_e(n):
    a = '%e' % Decimal(n)
    return a.split('e')[0].rstrip('0').rstrip('.')+'e'+a.split('e')[1]
```

Messwerte aus Tabelle 1 in SI Einheiten:

```
t = np.array([0,120,240,360,480,600,720]
Fehler_t = np.array([2,2,2,2,2,2,2])

f = np.array([648,600,552,515,470,437,403]) /60
Fehler_f = np.array([10,10,10,10,10,10,10]) /60
```

Berechnung von ω und $\Delta\omega$ nach (1) bzw. (2):

```
omega = 2*np.pi*f
Fehler_omega = 2*np.pi*Fehler_f
```

Fitfunktion (3) wird deklariert:

```
from scipy import odr

def fit_func(p, x):
    (A, k) = p
    return A*np.exp(-k*x)

model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
x = t
y = omega
delta_x = Fehler_t
delta_y = Fehler_omega
```

Startparameter für Ausgleichungsrechnung werden gesetzt, sodass Lösung konvergiert:

```
para0 = [1, 0]

data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
odr = odr.ODR(data, model, beta0=para0 )
out = odr.run()
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
popt = out.beta
perr = out.sd_beta
```

Angabe welche Sigma-Umgebung der Fitfunktion im Diagramm dargestellt werden soll:

```
nstd = 10

popt_top = popt+nstd*perr
popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

```
x_fit = np.linspace(min(x)-(max(x)-min(x))/10, max(x)+(max(x)-min(x))/10, 1000)
fit = fit_func(popt, x_fit)
fit_top = fit_func(popt_top, x_fit)
fit_bot = fit_func(popt_bot, x_fit)
```

Diagramm (Abb.2) wird erstellt:

```
fig, ax = plt.subplots(1)
plt.ticklabel_format(axis='both', style='sci', scilimits=(0,3), useMathText=True)
plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
    Messdaten')
plt.title('Kreisfrequenz als Funktion der Zeit')
plt.grid(True)
plt.yscale('log')
plt.xlabel('Zeit '+r'${t}$'+' '+r'${[s]}$')
plt.ylabel('Kreisfrequenz '+r'${\omega}$'+' '+r'${[Hz]}$')
plt.plot(x_fit, fit, 'r', lw=1, label='Fit')
ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung')
plt.legend(loc='best')

plt.savefig('figures/213_Fig1.pdf', format='pdf', bbox_inches='tight')
```

Der Chi-Quadrat-Test wird durchgeführt unter Berücksichtigung von $\boldsymbol{\Delta t}$ und $\boldsymbol{\Delta \omega}$. D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert[5]. Die Summe der normierten Abstandsquadrate, der χ^2-Wert, wird reduziert.

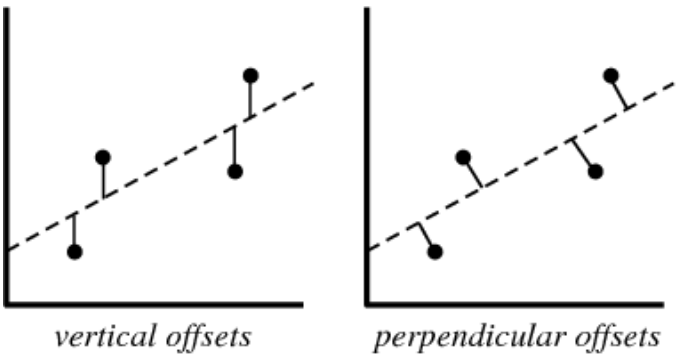

Abbildung 1: Abstände von Messdaten zur Fitfunktion[6]

[5] P. T. Boggs and J. E. Rogers, "Orthogonal Distance Regression," in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989," Contemporary Mathematics, vol. 112, pg. 186, 1990.

[6] http://mathworld.wolfram.com/LeastSquaresFitting.html

```
dof = x.size-popt.size
chisquare = out.sum_square
chisquare_red = chisquare/dof
prob = round(1-chi2.cdf(chisquare,dof),2)*100
```

Auslesen der Messergebnisse:

```
omega_0 = popt[0]
Fehler_omega_0 = perr[0]

k = popt[1]
Fehler_k = perr[1]
```

Die Halbwertszeit können wir nun sofort aus der Dämpfungskonstante $\boldsymbol{k}$ berechnen:

$$\boldsymbol{T_{1/2} = ln(2)/k} \tag{4}$$

$$\boldsymbol{\Delta T_{1/2} = |T_{1/2}\left(\frac{\Delta k}{k}\right)|} \tag{5}$$

Berechnung von $\boldsymbol{T_{1/2}}$ und $\boldsymbol{\Delta T_{1/2}}$ nach (4) bzw. (5):

```
T_halb = np.log(2)/k
Fehler_T_halb = T_halb*Fehler_k/k
```

Ausgabe der Messergebnisse wird erstellt:

```
print('Startwert der Kreisfrequenz: ')
print('omega_0 [Hz] =', format_e(omega_0), ' +- ', format_e(Fehler_omega_0))
print('Chi-Quadrat =', chisquare)
print('Freiheitsgrade =', dof)
print('Chi-Quadrat reduziert =', chisquare_red)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob, '%')
print('\n')
print('Dämpfungskonstante: ')
print('k [Hz] =',  format_e(k), ' +- ',  format_e(Fehler_k))
print('\n')
print('Halbwertszeit: ')
print('T_halb [s] =',  format_e(T_halb), ' +- ',  format_e(Fehler_T_halb))
```

4.3.2 Output

```
Startwert der Kreisfrequenz:
omega_0 [Hz] = 6.791995e+01  +-  1.653924e-01
Chi-Quadrat = 0.2262760874189914
Freiheitsgrade = 5
Chi-Quadrat reduziert = 0.045255217483798 28
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 100.0 %

Dämpfungskonstante:
k [Hz] = 6.600142e-04  +-  6.605926e-06

Halbwertszeit:
T_halb [s] = 1.0502e+03  +-  1.051121e+01
```

Wir erfahren also sofort, dass

$$\boldsymbol{k = 6.600(66) \times 10^{-4}\, Hz}$$
$$\boldsymbol{T_{1/2} = 1.050(11) \times 10^{3}\, s}$$

und als Ergebnis auf unseren Anpassungstest:

$$\boldsymbol{\chi^2_{red} = 4.5 \times 10^{-2}}$$

Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten ist

$$\boldsymbol{P \approx 100.0\%}$$

und wir erhalten das Diagramm in Abb.2.

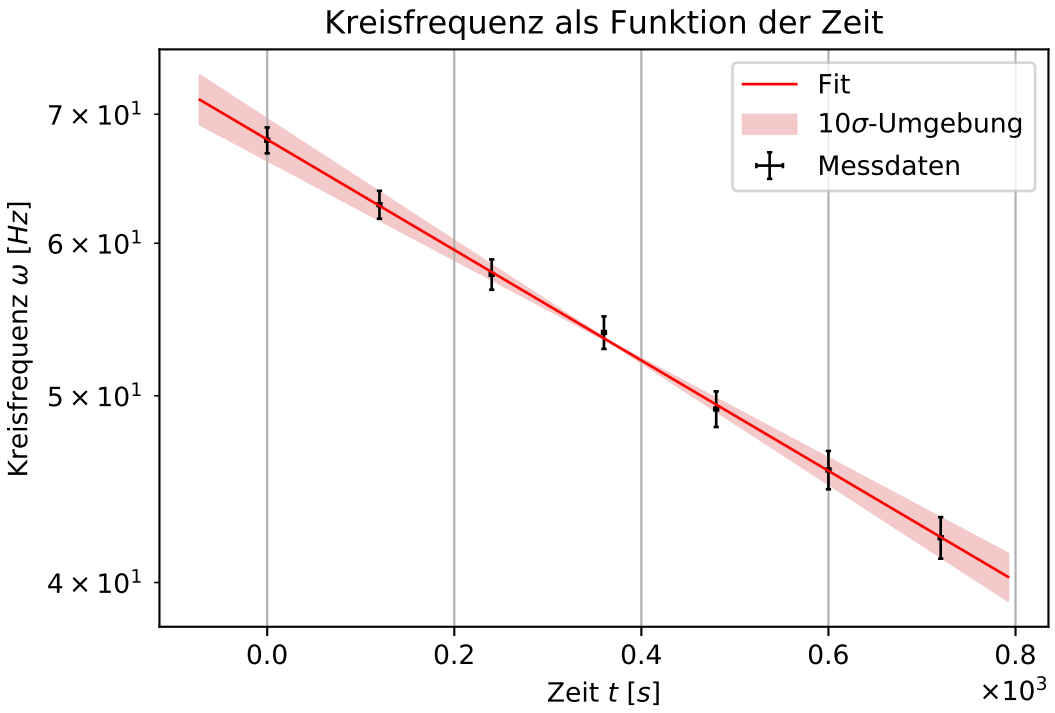

Abbildung 2

4.4 Auswertung

Die gemessenen Wert für die Dämpfungskonstante $\boldsymbol{k}$ und Halbwertszeit $\boldsymbol{T_{1/2}}$ machen Sinn und passen zu unserem theoretischem Modell (3):

$$\boldsymbol{k = 6.600(66) \times 10^{-4}\ Hz}$$
$$\boldsymbol{T_{1/2} = 1.050(11) \times 10^{3}\ s}$$

Der jedoch geringe Wert $\boldsymbol{\chi^2_{red} = 4.5 \times 10^{-2}}$ sagt uns, dass die bei der Messung geschätzten Messunsicherheiten zu groß abgeschätzt worden sind. D.h. $\boldsymbol{\Delta k}$ und $\boldsymbol{\Delta T_{1/2}}$ sind größtenteils aus statistischen Unsicherheiten hervorgegangen, existierende systematische Fehler waren Größenordnungen kleiner und beeinträchtigten die Messung und das Ergebnis nicht signifikant. Nur durch genaueres bestimmen der statischen Messunsicherheiten oder mehr Messungen, können $\boldsymbol{\Delta k}$ und $\boldsymbol{\Delta T_{1/2}}$ verringert werden und gegebenenfalls systematische Fehler gefunden, analysiert und korrigiert werden. Die Kreisfrequenz als Funktion der Zeit ist in Abbildung 2 dargestellt.

5 Bestimmung von I_z aus der Präzessionsfrequenz

5.1 Durchführung[7]

Bei allen Messungen dieser Aufgabe wird der Kreisel zunächst bei senkrechter Achse auf die gewünschte Geschwindigkeit gebracht. Anschließend wird die Achse durch Angreifen am Kugellager schräg gestellt und kurz vor der gewählten Ablesemarke möglichst nutationsfrei losgelassen. Als Ablesemarke dient ein Messingstab in der Kreiselbasis.

(a) Zuerst montieren wir die Farbscheibe auf den Stab und vergewissern uns, dass der Kreisel kräftefrei ist. Im Abstand von $\boldsymbol{l = 20\,cm}$ zur Kugelmitte wird ein Zusatzgewicht auf den Stab montiert. Die Drehgeschwindigkeit wird zuerst auf $\boldsymbol{f_F \approx 500\ min^{-1}}$ eingestellt. Den Stab wird bei gleicher Drehgeschwindigkeit des Kreisels möglichst nutationsfrei unter drei verschiedenen Winkeln des Stabs gegen die Vertikale losgelassen und die Zeit $\boldsymbol{T_P}$, in die Rotationsachse des Kreisels einen ganzen Umlauf vollzieht, bestimmt.

(b) Wir belasten den kräftefreien Kreisel mit folgenden Zusatzmassen:

- Ein Gewichtsstück bei $\boldsymbol{l = 20\,cm}$ (*).

[7] Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

- Ein Gewichtsstück bei $\boldsymbol{l = 15\ cm}$.
- Zwei Gewichtsstücke bei $\boldsymbol{l = 20\ cm}$.
- Zwei Gewichtsstücke bei $\boldsymbol{l = 15\ cm}$.

Gemessen wird mit der Stoppuhr die Präzessionsdauer $\boldsymbol{T_P}$ für jede Gewichtseinstellung bei jeweils vier verschiedene Frequenzen $\boldsymbol{f_F}$ im Bereich $\boldsymbol{250\,min^{-1} < f_F < 700\,min^{-1}}$. Zur ersten Messreihe (*) können wir auch die Messdaten aus der vorherigen Aufgabe hinzufügen, da es genau die gleichen Messkriterien ($\boldsymbol{n = 1}$,$\boldsymbol{l = 20\ cm}$) erfüllt. Bbei jeder Masseneinstellung beginnen wir bei einer hohen Frequenz und bremsen dann für die folgenden Messungen den Kreisel etwas ab. Für jede Messung notieren wir die Frequenz $\boldsymbol{f_F}$ und die Präzessionsdauer $\boldsymbol{T_P}$.

5.2 Messergebnisse

Messdaten wurden dem Versuchsprotokoll (22. Oktober, 2019) entnommen und in Tabelle 2 übertragen. Die Masse

Tabelle 2: Messung der Präzessionsdauer

Messreihe $Aufgabe_{Nr.}$	Eigenfrequenz $\boldsymbol{f_F}$ $[min^{-1}]$	Präzessionsdauer $\boldsymbol{T_P}$ $[s]$	Abstand $\boldsymbol{l}$ $[cm]$	Anzahl der Gewichte $\boldsymbol{n}$ [1]
(a)	500 ± 10	74.80 ± 0.30	20.0 ± 0.1	1
〃	500 ± 10	75.00 ± 0.30	〃	〃
〃	500 ± 10	74.80 ± 0.30	〃	〃
$(b_1)(*)$	298 ± 10	47.90 ± 0.30	20.0 ± 0.1	1
〃	404 ± 10	64.00 ± 0.30	〃	〃
〃	501 ± 10	78.50 ± 0.30	〃	〃
〃	602 ± 10	92.51 ± 0.30	〃	〃
(b_2)	297 ± 10	61.46 ± 0.30	15.0 ± 0.1	1
〃	407 ± 10	85.31 ± 0.30	〃	〃
〃	495 ± 10	101.56 ± 0.30	〃	〃
〃	601 ± 10	122.67 ± 0.30	〃	〃
(b_3)	599 ± 10	44.76 ± 0.30	20.0 ± 0.1	2
〃	502 ± 10	38.28 ± 0.30	〃	〃
〃	403 ± 10	30.72 ± 0.30	〃	〃
〃	301 ± 10	23.04 ± 0.30	〃	〃
(b_4)	603 ± 10	61.06 ± 0.30	15.0 ± 0.1	2
〃	499 ± 10	50.67 ± 0.30	〃	〃
〃	300 ± 10	32.23 ± 0.30	〃	〃
〃	298 ± 10	31.06 ± 0.30	〃	〃

[1] $\boldsymbol{\Delta f_F}$ grob abgeschätzt
[2] $\boldsymbol{\Delta T_P}$ grob abgeschätzt als durchschnittliche menschliche Reaktionszeit von 200 ms und Faktor $\boldsymbol{\sqrt{2}}$, weil zum Messen die Stoppuhr doppelt betätigt werden muss:
$\boldsymbol{0.2\ s \times \sqrt{2} \approx 0.3\ s = \Delta T_P}$
(*) Zu dieser Messreihe wird im weiteren Verlauf auch Messreihe (a) gezählt

eines einzelnen Gewichts beträgt

$$\boldsymbol{m_G = 9.85(1)\ g}$$

und die Erdbeschleunigung nähern wir mit

$$\boldsymbol{g = 9.851(1)\ m\ s^{-2}}$$

5.3 Kurvenanpassung mit Python, Schritt 1

5.3.1 Source Code & Input

Wir drücken $\boldsymbol{f_F}$ über die entsprechende Kreisfrequenz $\boldsymbol{\omega_F}$ aus:

$$\boldsymbol{\omega_F = 2\pi f_F} \tag{6}$$

$$\boldsymbol{\Delta\omega_F = 2\pi(\Delta f_F)} \tag{7}$$

Wir gehen davon aus, dass die Präzessionsdauer proportional zur Eigenkreisfrequenz zunimmt. Daher ist unser funktionales Modell für die Ausgleichungsrechnung wie folgt:

$$\boxed{\boldsymbol{T_P = S\omega_F}} \tag{8}$$

So sieht die Fortführung unserer Python-Implementierung aus:

Messwerte aus Tabelle 2 in SI Einheiten (vorerst nicht nach Messreihen aufgeteilt):

```
f_F = np.array([500,500,500,298,404,501,602,297,407,495, 601,599,502,403,301,603,499,300,298]) /60 ]
Fehler_f_F = np.full(f_F.size, 10) /60

T_P = np.array([74.80,75.00,75.39,47.90,64.00,78.50,92.51,61.46,85.31,101.56, 122.67,44.76,
                38.28,30.72,23.04,61.06,50.67,32.23,31.06])
Fehler_T_P = np.full(T_P.size, 0.30)
```

Berechnung der Eigenkreisfrequenz $\boldsymbol{\omega_F}$ und $\boldsymbol{\Delta\omega_F}$ nach (6) bzw. (7):

```
omega_F = 2*np.pi*f_F
Fehler_omega_F = 2*np.pi*Fehler_f_F
```

Fitfunktion (8) wird deklariert:

```
from scipy import odr

def fit_func(p, x):
    (s) = p
    return s*x

model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben (aufgeteilt auf die Messreihen $\boldsymbol{(a) + (b_1)}$, $\boldsymbol{(b_2)}$, $\boldsymbol{(b_3)}$ und $\boldsymbol{(b_4)}$):

```
x_1 = omega_F[0:7]
y_1 = T_P[0:7]
delta_x_1 = Fehler_omega_F[0:7]
delta_y_1 = Fehler_T_P[0:7]

x_2 = omega_F[7:11]
y_2 = T_P[7:11]
delta_x_2 = Fehler_omega_F[7:11]
delta_y_2 = Fehler_T_P[7:11]

x_3 = omega_F[11:15]
y_3 = T_P[11:15]
delta_x_3 = Fehler_omega_F[11:15]
delta_y_3 = Fehler_T_P[11:15]

x_4 = omega_F[15:19]
y_4 = T_P[15:19]
delta_x_4 = Fehler_omega_F[15:19]
delta_y_4 = Fehler_T_P[15:19]
```

Startparameter für Ausgleichungsrechnung werden gesetzt, sodass Lösung konvergiert:

```
para0 = [1]

data_1 = odr.RealData(x_1, y_1, sx=delta_x_1, sy=delta_y_1)
odr_1 = odr.ODR(data_1, model, beta0=para0 )
out_1 = odr_1.run()

data_2 = odr.RealData(x_2, y_2, sx=delta_x_2, sy=delta_y_2)
odr_2 = odr.ODR(data_2, model, beta0=para0 )
out_2 = odr_2.run()

data_3 = odr.RealData(x_3, y_3, sx=delta_x_3, sy=delta_y_3)
odr_3 = odr.ODR(data_3, model, beta0=para0 )
out_3 = odr_3.run()

data_4 = odr.RealData(x_4, y_4, sx=delta_x_4, sy=delta_y_4)
odr_4 = odr.ODR(data_4, model, beta0=para0 )
out_4 = odr_4.run()
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
popt_1 = out_1.beta
perr_1 = out_1.sd_beta

popt_2 = out_2.beta
perr_2 = out_2.sd_beta

popt_3 = out_3.beta
perr_3 = out_3.sd_beta

popt_4 = out_4.beta
perr_4 = out_4.sd_beta
```

Angabe welche Sigma-Umgebung der Fitfunktion im Diagramm dargestellt werden soll:

```
nstd = 1

popt_top_1 = popt_1+nstd*perr_1
popt_bot_1 = popt_1-nstd*perr_1

popt_top_2 = popt_2+nstd*perr_2
popt_bot_2 = popt_2-nstd*perr_2

popt_top_3 = popt_3+nstd*perr_3
popt_bot_3 = popt_3-nstd*perr_3

popt_top_4 = popt_4+nstd*perr_4
popt_bot_4 = popt_4-nstd*perr_4
```

Plot-Umgebung wird angegeben:

```
x_fit_1 = np.linspace(min(x_1)-(max(x_1)-min(x_1))/10, max(x_1)+(max(x_1)-min(x_1))/10, 1000)
fit_1 = fit_func(popt_1, x_fit_1)
fit_top_1 = fit_func(popt_top_1, x_fit_1)
fit_bot_1 = fit_func(popt_bot_1, x_fit_1)

x_fit_2 = np.linspace(min(x_2)-(max(x_2)-min(x_2))/10, max(x_2)+(max(x_2)-min(x_2))/10, 1000)
fit_2 = fit_func(popt_2, x_fit_2)
fit_top_2 = fit_func(popt_top_2, x_fit_2)
fit_bot_2 = fit_func(popt_bot_2, x_fit_2)

x_fit_3 = np.linspace(min(x_3)-(max(x_3)-min(x_3))/10, max(x_3)+(max(x_3)-min(x_3))/10, 1000)
fit_3 = fit_func(popt_3, x_fit_3)
fit_top_3 = fit_func(popt_top_3, x_fit_3)
fit_bot_3 = fit_func(popt_bot_3, x_fit_3)

x_fit_4 = np.linspace(min(x_4)-(max(x_4)-min(x_4))/10, max(x_4)+(max(x_4)-min(x_4))/10, 1000)
fit_4 = fit_func(popt_4, x_fit_4)
fit_top_4 = fit_func(popt_top_4, x_fit_4)
fit_bot_4 = fit_func(popt_bot_4, x_fit_4)
```

Diagramm (Abb.3) wird erstellt:

```
fig, ax = plt.subplots(1, figsize=[6.4 *1.5, 4.8])
plt.ticklabel_format(axis='both', style='sci', scilimits=(0,3), useMathText=True)
plt.errorbar(x_1, y_1, yerr=delta_y_1, xerr=delta_x_1, lw=1, ecolor='k', fmt='none', capsize=1,
    label='Messdaten 1')
plt.errorbar(x_2, y_2, yerr=delta_y_2, xerr=delta_x_2, lw=1, ecolor='k', fmt='none', capsize=1,
    label='Messdaten 2')
plt.errorbar(x_3, y_3, yerr=delta_y_3, xerr=delta_x_3, lw=1, ecolor='k', fmt='none', capsize=1,
    label='Messdaten 3')
plt.errorbar(x_4, y_4, yerr=delta_y_4, xerr=delta_x_4, lw=1, ecolor='k', fmt='none', capsize=1,
    label='Messdaten 4')
plt.title('Präzessionsdauer als Funktion der Eigenkreisfrequenz')
plt.grid(True)
plt.xlabel('Eigenkreisfrequenz '+r'${\omega_F}$'+' '+r'${[Hz]}$')
plt.ylabel('Präzessionsdauer '+r'${T_P}$'+' '+r'${[s]}$')
plt.plot(x_fit_1, fit_1, color='C3', lw=1, label='Fit  1')
plt.plot(x_fit_2, fit_2, color='C2', lw=1, label='Fit  2')
plt.plot(x_fit_3, fit_3, color='C0', lw=1, label='Fit  3')
plt.plot(x_fit_4, fit_4, color='C1', lw=1, label='Fit  4')
ax.fill_between(x_fit_1, fit_top_1, fit_bot_1, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung 1')
ax.fill_between(x_fit_2, fit_top_2, fit_bot_2, color='C2', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung 2')
ax.fill_between(x_fit_3, fit_top_3, fit_bot_3, color='C0', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung 3')
```

```
ax.fill_between(x_fit_4, fit_top_4, fit_bot_4, color='C1', alpha=.25, label=str(nstd)+r'$\sigma$'+'-Umgebung 4')
box = ax.get_position()
ax.set_position([box.x0, box.y0, box.width * 0.8, box.height])
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.savefig('figures/213_Fig2.pdf', format='pdf', bbox_inches='tight')
```

Auslesen der Messergebnisse:

```
S_1 = popt_1[0]
Fehler_S_1 = perr_1[0]
S_2 = popt_2[0]
Fehler_S_2 = perr_2[0]
S_3 = popt_3[0]
Fehler_S_3 = perr_3[0]
S_4 = popt_4[0]
Fehler_S_4 = perr_4[0]
```

Der Chi-Quadrat-Test wird durchgeführt unter Berücksichtigung von $\boldsymbol{\Delta\omega_F}$ und $\boldsymbol{\Delta T_P}$. D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert[8].

```
dof_1 = x_1.size-popt.size
chisquare_1 = out_1.sum_square
chisquare_red_1 = chisquare_1/dof_1
prob_1 = round(1-chi2.cdf(chisquare_1,dof_1),2)*100

dof_2 = x_2.size-popt.size
chisquare_2 = out_2.sum_square
chisquare_red_2 = chisquare_2/dof_2
prob_2 = round(1-chi2.cdf(chisquare_2,dof_2),2)*100

dof_3 = x_3.size-popt.size
chisquare_3 = out_3.sum_square
chisquare_red_3 = chisquare_3/dof_3
prob_3 = round(1-chi2.cdf(chisquare_3,dof_3),2)*100

dof_4 = x_4.size-popt.size
chisquare_4 = out_4.sum_square
chisquare_red_4 = chisquare_4/dof_4
prob_4 = round(1-chi2.cdf(chisquare_4,dof_4),2)*100
```

Ausgabe der Messergebnisse wird erstellt:

```
print('Steigungen: ')
print('S_1 [s^2] =', format_e(S_1), ' +- ', format_e(Fehler_S_1))
print('Chi-Quadrat =', chisquare_1)
print('Freiheitsgrade =', dof_1)
print('Chi-Quadrat reduziert =', chisquare_red_1)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_1, '%')
print('\n')
print('S_2 [s^2] =', format_e(S_2), ' +- ', format_e(Fehler_S_2))
print('Chi-Quadrat =', chisquare_2)
print('Freiheitsgrade =', dof_2)
print('Chi-Quadrat reduziert =', chisquare_red_2)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_2, '%')
print('\n')
print('S_3 [s^2] =', format_e(S_3), ' +- ', format_e(Fehler_S_3))
print('Chi-Quadrat =', chisquare_3)
print('Freiheitsgrade =', dof_3)
print('Chi-Quadrat reduziert =', chisquare_red_3)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_3, '%')
print('\n')
print('S_4 [s^2] =', format_e(S_4), ' +- ', format_e(Fehler_S_4))
print('Chi-Quadrat =', chisquare_4)
print('Freiheitsgrade =', dof_4)
print('Chi-Quadrat reduziert =', chisquare_red_4)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_4, '%')
print('\n')
```

[8] P. T. Boggs and J. E. Rogers, "Orthogonal Distance Regression," in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989," Contemporary Mathematics, vol. 112, pg. 186, 1990.

5.3.2 Output

```
Steigungen:
S_1 [s^2] = 1.465235e+00  +-  1.366893e-02
Chi-Quadrat = 8.120624052429571
Freiheitsgrade = 5
Chi-Quadrat reduziert = 1.6241248104859143
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 15.0 %

S_2 [s^2] = 1.965064e+00  +-  1.134583e-02
Chi-Quadrat = 0.8421813756024409
Freiheitsgrade = 2
Chi-Quadrat reduziert = 0.42109068780122044
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 66.0 %

S_3 [s^2] = 7.224199e-01  +-  4.304101e-03
Chi-Quadrat = 0.7948108139557275
Freiheitsgrade = 2
Chi-Quadrat reduziert = 0.39740540697786375
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 67.0 %

S_4 [s^2] = 9.780491e-01  +-  1.114486e-02
Chi-Quadrat = 2.8383188936144617
Freiheitsgrade = 2
Chi-Quadrat reduziert = 1.4191594468072308
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 24.0 %
```

Wir erfahren also sofort, dass

$$\boldsymbol{S_1 = 1.465(14)\ s^2}$$
$$\boldsymbol{S_2 = 1.965(11)\ s^2}$$
$$\boldsymbol{S_3 = 7.224(43) \times 10^{-1}\ s^2}$$
$$\boldsymbol{S_4 = 9.78(11) \times 10^{-1}\ s^2}$$

und als Ergebnis auf unseren Anpassungstests:

$$\boldsymbol{\chi^2_{red,1} = 1.6}$$
$$\boldsymbol{\chi^2_{red,2} = 4.2 \times 10^{-1}}$$
$$\boldsymbol{\chi^2_{red,3} = 4.0 \times 10^{-1}}$$
$$\boldsymbol{\chi^2_{red,4} = 1.4}$$

Die Wahrscheinlichkeiten ein größeres oder gleiches Chi-Quadrat zu erhalten sind jeweils

$$\boldsymbol{P_1 \approx 15.0\%}$$
$$\boldsymbol{P_2 \approx 66.0\%}$$
$$\boldsymbol{P_3 \approx 67.0\%}$$
$$\boldsymbol{P_4 \approx 24.0\%}$$

und wir erhalten das Diagramm in Abb.3.

5.4 Kurvenanpassung mit Python, Schritt 2

5.4.1 Source Code & Input

Wir gehen davon aus, dass das Trägheitsmoment konstant ist. Daher ist unser funktionales Modell für die Ausgleichungsrechnung wie folgt:

$$\boxed{\boldsymbol{I_z = konst.}} \tag{9}$$

Für diesen Schritt, benutzen wir die Angaben aus "Versuchsaufbau, Literaturwerte & Vorbereitung"

$$\boldsymbol{g = 9.80984(2)\ m\ s^2}$$
$$\boldsymbol{m_G = 9,85(1)\ g}$$

und die Messergebnisse aus Schritt 1 um das Trägheitsmoment um die Figurenachse zu berechnen:

$$\boldsymbol{m = m_G \times n} \tag{10}$$

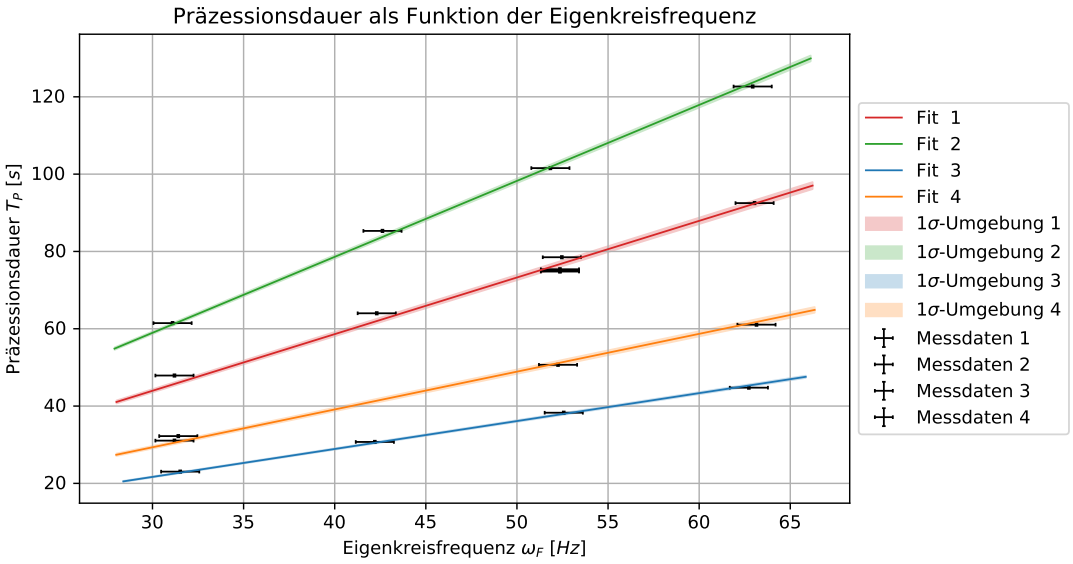

Abbildung 3

$$\Delta m = \Delta m_G \times \sqrt{n} \tag{11}$$

$$I_z = \frac{mglS}{2\pi} \tag{12}$$

$$\Delta I_z = |I_z| \sqrt{\left(\frac{\Delta m}{m}\right)^2 + \left(\frac{\Delta g}{g}\right)^2 + \left(\frac{\Delta l}{l}\right)^2 + \left(\frac{\Delta S}{S}\right)^2} \tag{13}$$

So sieht die Fortführung unserer Python-Implementierung aus:

Messwerte der Steigung S aus Schritt 1 in SI Einheiten:

```
S = np.array([S_1,S_2,S_3,S_4])
Fehler_S = np.array([Fehler_S_1,Fehler_S_2,Fehler_S_3,Fehler_S_4])
```

Messwerte aus Tabelle 2 in SI Einheiten:

```
l = np.array([20,15,20,15]) /1e2
Fehler_l = np.full(l.size, 0.1) /1e2

n =  np.array([1,1,2,2])
```

Angaben aus "Versuchsaufbau, Literaturwerte & Vorbereitungïn SI Einheiten:

```
m_G = 9.85 /1e3
Fehler_m_G = 0.01 /1e3

g = 9.80984
Fehler_g = 0.00002
```

Berechnung der Masse m und Δm nach (10) bzw. (11):

```
m = n*m_G
Fehler_m = np.sqrt(n)* Fehler_m_G
```

Berechnung des Trägheitsmoments I_z und ΔI_z nach (12) bzw. (13):

```
I_z = m*g*l*S/(2*np.pi)
Fehler_I_z = abs(I_z)*np.sqrt((Fehler_m/m)**2+(Fehler_g/g)**2+(Fehler_l/l)**2+(Fehler_S/S)**2)
```

Fitfunktion (9) wird deklariert:

```
from scipy import odr

def fit_func(p, x):
    (c) = p
    return x*0+c

model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
x = S
y = I_z
delta_x = Fehler_S
delta_y = Fehler_I_z
```

Startparameter für Ausgleichungsrechnung werden gesetzt, sodass Lösung konvergiert:

```
para0 = [0]

data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
odr = odr.ODR(data, model, beta0=para0 )
out = odr.run()
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
popt = out.beta
perr = out.sd_beta
```

Angabe welche Sigma-Umgebung der Fitfunktion im Diagramm dargestellt werden soll:

```
nstd = 1

popt_top = popt+nstd*perr
popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

```
x_fit = np.linspace(min(x)-(max(x)-min(x))/10, max(x)+(max(x)-min(x))/10, 1000)
fit = fit_func(popt, x_fit)
fit_top = fit_func(popt_top, x_fit)
fit_bot = fit_func(popt_bot, x_fit)
```

Diagramm (Abb.4) wird erstellt:

```
fig, ax = plt.subplots(1)
plt.ticklabel_format(axis='both', style='sci', scilimits=(0,3), useMathText=True)
plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
    Messdaten')
plt.title('Trägheitsmoment')
plt.grid(True)
plt.xlabel('Steigung '+r'${S}$'+' '+r'${[s^2]}$')
plt.ylabel('Trägheitsmoment '+r'${I_z}$' + ' '+r'${[s]}$')
plt.plot(x_fit, fit, color='r', lw=1, label='Fit')
ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung')
plt.legend(loc='best')

plt.savefig('figures/213_Fig3.pdf', format='pdf', bbox_inches='tight')
```

Auslesen der Messergebnisse:

```
I_z = popt[0]
Fehler_I_z = perr[0]
```

Der Chi-Quadrat-Test wird durchgeführt unter Berücksichtigung von $\boldsymbol{\Delta S}$ und $\boldsymbol{\Delta I_z}$. D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert[9]. Die Summe der normierten Abstandsquadrate, der χ^2-Wert, wird reduziert.

```
dof = x.size-popt.size
chisquare = out.sum_square
chisquare_red = chisquare/dof
prob = round(1-chi2.cdf(chisquare,dof),2)*100
```

Ausgabe der Messergebnisse wird erstellt:

```
print('Trägheitsmoment: ')
print('I_z [kg * m^2] =', format_e(I_z), ' +- ', format_e(Fehler_I_z))
print('Chi-Quadrat =', chisquare)
print('Freiheitsgrade =', dof)
print('Chi-Quadrat reduziert =', chisquare_red)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob, '%')
```

[9] P. T. Boggs and J. E. Rogers, "Orthogonal Distance Regression," in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989," Contemporary Mathematics, vol. 112, pg. 186, 1990.

5.4.2 Output

```
Trägheitsmoment:
I_z [kg * m^2] = 4.490859e-03  +-  2.207146e-05
Chi-Quadrat = 3.162438468024628
Freiheitsgrade = 3
Chi-Quadrat reduziert = 1.0541461560082093
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 37.0 %
```

Wir erfahren also sofort, dass

$$\boldsymbol{I_z = 4.491(22) \times 10^{-3}\ kg\ m^2}$$

und als Ergebnis auf unseren Anpassungstest:

$$\boldsymbol{\chi^2_{red} = 1.1}$$

Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten ist

$$\boldsymbol{P \approx 37.0\%}$$

und wir erhalten das Diagramm in Abb.4.

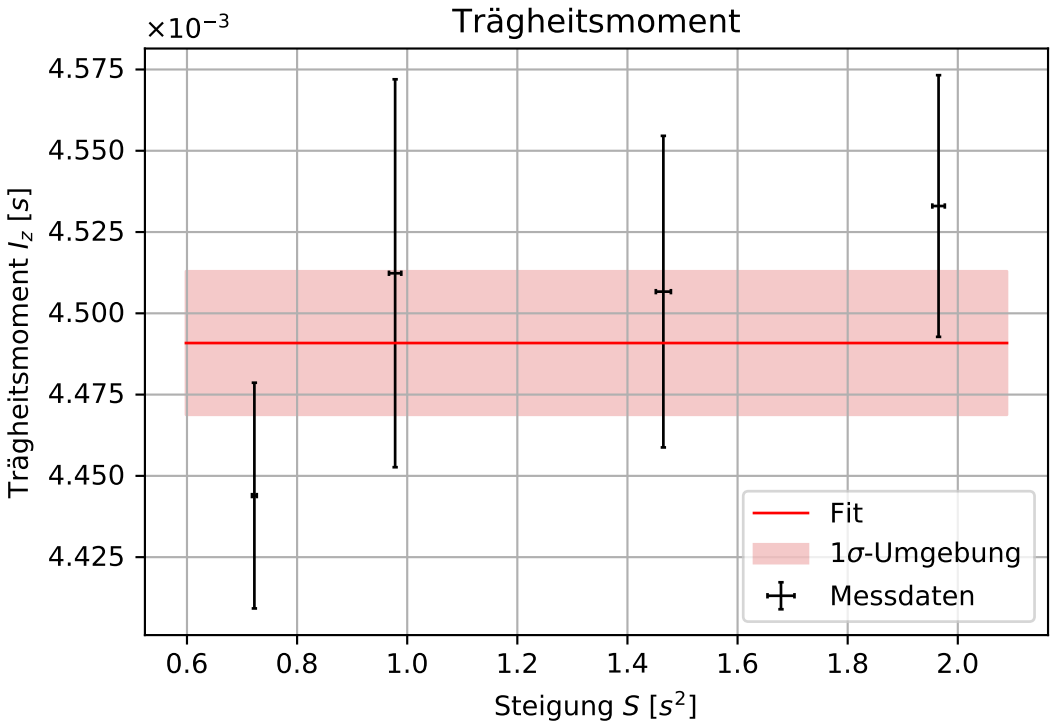

Abbildung 4

5.5 Auswertung

(a) Wie erwartet hat die Messreihe (a) Zeiten geliefert die nicht signifikant von einander abweichen (siehe Tab. 3). Alle 3 Werte liegen paarweise jeweils in der 1-Sigma-Umgebung von einander. Das stimmt mit unserem theoretischen Modell überein, dass die Präzessionsdauer, bei konstanter Eigenfrequenz $\boldsymbol{f_F}$ und Drehmoment $\boldsymbol{M}$ bzw. Gewichtskonfiguration $\boldsymbol{\{n, l\}}$, sich nicht ändert. Auch in diesem Versuch überwiegen die statistischen Messunsicherheiten jegliche systematischen Fehler. Nur durch viel genaueres bestimmen der statischen Messunsicherheiten oder mehr Messungen, können dieses systematische Fehler gefunden, analysiert und gegebenenfalls korrigiert werden.

(b) Die gemessenen Werte für die 4 Steigungen $\boldsymbol{S_i}$ und $\boldsymbol{I_z}$ machen Sinn und passen zu unserem theoretischen Modellen

Tabelle 3: Messung der Präzessionsdauer

Messreihe $Aufgabe_{Nr.}$	Eigenfrequenz $\boldsymbol{f_F}$ $[min^{-1}]$	Präzessionsdauer $\boldsymbol{T_P}$ $[s]$	Abstand $\boldsymbol{l}$ $[cm]$	Anzahl der Gewichte $\boldsymbol{n}$ [1]
(a)	500 ± 10	74.80 ± 0.30	20.0 ± 0.1	1
$''$	500 ± 10	75.00 ± 0.30	$''$	$''$
$''$	500 ± 10	74.80 ± 0.30	$''$	$''$

[1] $\boldsymbol{\Delta f_F}$ grob abgeschätzt
[2] $\boldsymbol{\Delta T_P}$ grob abgeschätzt als durchschnittliche menschliche Reaktionszeit von 200 ms und Faktor $\sqrt{2}$, weil zum Messen die Stoppuhr doppelt betätigt werden muss:
$\boldsymbol{0.2\ s \times \sqrt{2} \approx 0.3\ s = \Delta T_P}$

(8) und (9) :

$$\boldsymbol{S_1 = 1.465(14)\ s^2}$$
$$\boldsymbol{S_2 = 1.965(11)\ s^2}$$
$$\boldsymbol{S_3 = 7.224(43) \times 10^{-1}\ s^2}$$
$$\boldsymbol{S_4 = 9.78(11) \times 10^{-1}\ s^2}$$
$$\boldsymbol{I_z = 4.491(22) \times 10^{-3}\ kg\ m^2}$$

Die Präzessionsdauern als Funktionen der Eigenkreisfrequenz aus jeder Messreihe sind in Abbildung 3 dargestellt. Das resultierenden Messwerte für das Trägheitsmomente aus den einzelnen Messreihen und ihr Mittelwert sind in Abbildung 4 dargestellt.
Die Wahrscheinlichkeiten ein größeres oder gleiches Chi-Quadrat zu erhalten

$$\boldsymbol{P_1 \approx 15.0\%}$$
$$\boldsymbol{P_2 \approx 66.0\%}$$
$$\boldsymbol{P_3 \approx 67.0\%}$$
$$\boldsymbol{P_4 \approx 24.0\%}$$
$$\boldsymbol{P \approx 37.0\%}$$

sind auch alle in Ordnung. Weder waren unsere Abschätzungen für die Messunsicherheiten signifikant zu groß, weder sind wir auf signifikante systematische Fehler getroffen, die eine Korrektur von unseren theoretischen Modellen verlangen würden. Letzteres kann sich aber wie immer bei zunehmender Beseitigung statistischer Fehler und weitere Analyse ändern.

6 Bestimmung von I_x aus der Größe und der Richtung der Umlaufschwindigkeit

6.1 Durchführung[10]

Wir überprüfen, ob der Kreisel kräftefrei ist und versetzen anschließend den Kreisel bei senkrechter Achse mit Hilfe des Motors in Rotation. Nach dem Anwerfen wird durch einen leichten seitlichen Stoß auf die Achse, der Kreisel in Nutation versetzt.

(a) Die Umlaufrichtung der momentanen Drehachse wird mit Hilfe der Farbscheibe bestimmt. Dafür beachten wir die Reihenfolge der Farben im einfarbigen Punkt.

(b) Mit der Stoppuhr messen wir für 10 Frequenzen im Bereich $\boldsymbol{300\ min^{-1} < f_F < 600\ min^{-1}}$ jeweils die Zeit $\boldsymbol{T_{10}}$ für 10 Umläufe der momentanen Drehachse um die Figurenachse. Die Frequenz des Farbwechsels $\boldsymbol{\Omega}$ der Sektorscheib entspricht genau der Umlaufsgeschwindigkeit $\boldsymbol{\Omega}$ der momentanen Drehachse um die Figurenachse des Kreisels.

6.2 Messergebnisse

(a) In unserem Versuch rotierte der Kreisel von oben betrachtet gegen den Uhrzeigersinn. Die Farben im einfarbigem Punkt folgten der zeitlichen Reihenfolge "Gelb, Rot, Grün, Gelb...", was auf unserer Scheibe mit den Sektoren

[10] Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

auch der Reihenfolge gegen den Uhrzeigersinn entspricht. D.h. bei unserem Kreisel zeigen $\vec{\boldsymbol{\Omega}}$ und $\vec{\boldsymbol{\omega}}_{\boldsymbol{F}}$ in ungefähr die gleiche Richtung $(\vec{\boldsymbol{\Omega}}\vec{\boldsymbol{\omega}}_{\boldsymbol{F}} > \mathbf{0})$. Sie haben also das gleiche Vorzeichen in der Gleichung

$$\boldsymbol{\Omega} = \frac{\boldsymbol{I_x} - \boldsymbol{I_z}}{\boldsymbol{I_x}} \boldsymbol{\omega_F} \tag{14}$$

die, die Geometrie der Nutationsbewegung für symmetrische, kräftefreie Kreisel beschreibt.

(b) Messdaten wurden dem Versuchsprotokoll (22. Oktober, 2019) entnommen und in Tabelle 4 übertragen.

Tabelle 4: Messung der Periode vom Farbwechsel

Eigenfrequenz $\boldsymbol{f_F}$ $[min^{-1}]$	Dauer von 10 Perioden $\boldsymbol{T_{10}}$ $[s]$
485 ± 10	22.37 ± 0.30
360 ± 10	29.90 ± 0.30
511 ± 10	21.37 ± 0.30
397 ± 10	28.01 ± 0.30
460 ± 10	23.29 ± 0.30
415 ± 10	26.48 ± 0.30
560 ± 10	19.89 ± 0.30
510 ± 10	20.53 ± 0.30
460 ± 10	23.17 ± 0.30
400 ± 10	26.42 ± 0.30

[1] $\boldsymbol{\Delta f_F}$ grob abgeschätzt
[2] $\boldsymbol{\Delta T_{10}}$ grob abgeschätzt als durchschnittliche menschliche Reaktionszeit von 200 ms und Faktor $\sqrt{\mathbf{2}}$, weil zum Messen die Stoppuhr doppelt betätigt werden muss:
$\mathbf{0.2}\ \boldsymbol{s} \times \sqrt{\mathbf{2}} \approx \mathbf{0.3}\ \boldsymbol{s} = \boldsymbol{\Delta T_P}$

6.3 Kurvenanpassung mit Python

6.3.1 Source Code & Input

Wir können sofort folgende Zusammenhänge schließen:

$$\boldsymbol{T_{Farbe}} = \boldsymbol{T_{10}}/\mathbf{10} \tag{15}$$

$$\boldsymbol{\Delta T_{Farbe}} = (\boldsymbol{\Delta T_{10}})/\mathbf{10} \tag{16}$$

$$|\boldsymbol{\Omega}| = \frac{\mathbf{2\pi}}{\boldsymbol{T_{Farbe}}} \tag{17}$$

Wie sich in (14) herausstellt, zeigen $\vec{\boldsymbol{\Omega}}$ und $\vec{\boldsymbol{\omega}}_{\boldsymbol{F}}$ in die gleiche Richtung, deswegen gilt:

$$\boldsymbol{\Omega} = \frac{\mathbf{2\pi}}{\boldsymbol{T_{Farbe}}} \tag{18}$$

$$\boldsymbol{\Delta\Omega} = |\boldsymbol{\Omega}|\left(\frac{\boldsymbol{\Delta T_{Farbe}}}{\boldsymbol{T_{Farbe}}}\right) \tag{19}$$

Wir gehen davon aus, dass die Umlaufsgeschwindigkeit $\boldsymbol{\Omega}$ proportional zur Eigenkreisfrequenz zunimmt. Daher ist unser funktionales Modell für die Ausgleichungsrechnung wie folgt:

$$\boxed{\boldsymbol{\Omega} = \boldsymbol{S_{Farbe}}\boldsymbol{\omega_F}} \tag{20}$$

Aus (14) und (20) folgt außerdem:

$$\boldsymbol{I_{x,1}} = \frac{\boldsymbol{I_z}}{\mathbf{1} - \boldsymbol{S_{Farbe}}} \tag{21}$$

$$\boldsymbol{\Delta I_{x,1}} = |\boldsymbol{I_{x,1}}|\sqrt{\left(\frac{\boldsymbol{\Delta I_z}}{\boldsymbol{I_z}}\right)^{\mathbf{2}} + \left(\frac{\boldsymbol{\Delta}(\mathbf{1} - \boldsymbol{S_{Farbe}})}{\mathbf{1} - \boldsymbol{S_{Farbe}}}\right)^{\mathbf{2}}}$$

$$= |I_{x,1}| \sqrt{\left(\frac{\Delta I_z}{I_z}\right)^2 + \left(\frac{\Delta S_{Farbe}}{1 - S_{Farbe}}\right)^2} \tag{22}$$

So sieht die Fortführung unserer Python-Implementierung aus:

Messwerte aus Tabelle 4 in SI Einheiten:

```
f_F = np.array([485,360,511,397,460,415,560,510,460,400]) /60
Fehler_f_F = np.full(f_F.size, 10) /60

T_10 = np.array([22.37,29.90,21.37,28.01,23.29,26.48,19.89,20.53,23.17,26.42])
Fehler_T_10 = np.full(T_Farbe.size, 0.30)
```

Berechnung der Periode vom Farbwechsel T_{Farbe} und ΔT_{Farbe} nach (15) bzw. (16):

```
T_Farbe = T_10/10
Fehler_T_Farbe = Fehler_T_10 /10
```

Berechnung der Eigenkreisfrequenz ω_F und $\Delta\omega_F$ nach (6) bzw. (7):

```
omega_F = 2*np.pi*f_F
Fehler_omega_F = 2*np.pi*Fehler_f_F

Omega = 2*np.pi/T_Farbe
Fehler_Omega = 2*np.pi*Omega*Fehler_T_Farbe/T_Farbe
```

Fitfunktion (20) wird deklariert:

```
from scipy import odr

def fit_func(p, x):
    (s) = p
    return x*s

model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
x = omega_F
y = Omega
delta_x = Fehler_omega_F
delta_y = Fehler_Omega
```

Startparameter für Ausgleichungsrechnung werden gesetzt, sodass Lösung konvergiert:

```
para0 = [1]

data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
odr = odr.ODR(data, model, beta0=para0 )
out = odr.run()
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
popt = out.beta
perr = out.sd_beta
```

Angabe welche Sigma-Umgebung der Fitfunktion im Diagramm dargestellt werden soll:

```
nstd = 1

popt_top = popt+nstd*perr
popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

```
x_fit = np.linspace(min(x)-(max(x)-min(x))/10, max(x)+(max(x)-min(x))/10, 1000)
fit = fit_func(popt, x_fit)
fit_top = fit_func(popt_top, x_fit)
fit_bot = fit_func(popt_bot, x_fit)
```

Diagramm (Abb.5) wird erstellt:

```
fig, ax = plt.subplots(1)
plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
    Messdaten')
plt.title('Winkelgeschwindigkeit der Drehachse als Funktion der Eigenkreisfrequenz')
```

```
plt.grid(True)
plt.xlabel('Eigenkreisfrequenz '+r'${\omega_F}$'+' '+r'${[Hz]}$')
plt.ylabel('Winkelgeschwindigkeit '+r'${\Omega}$'+' '+r'${[Hz]}$')
plt.plot(x_fit, fit, color='C3', lw=1, label='Fit')
ax.fill_between(x_fit, fit_top, fit_bot, color='r', alpha=.25, label=str(nstd)+r'$\sigma$'+'-Umgebung')
plt.legend(loc='best')

plt.savefig('figures/213_Fig4.pdf', format='pdf', bbox_inches='tight')
```

Auslesen der Messergebnisse:

```
S_Farbe = popt[0]
Fehler_S_Farbe = perr[0]
```

Berechnung des Trägheitsmoments $\boldsymbol{I_{x,1}}$ und $\boldsymbol{\Delta I_{x,1}}$ nach (21) bzw. (22):

```
I_x_1 = I_z/(1-S_Farbe)
Fehler_I_x_1 = abs(I_x_1)*np.sqrt((Fehler_I_z/I_z)**2+(Fehler_S_Farbe/(1-S_Farbe))**2)
```

Der Chi-Quadrat- Test wird durchgeführt unter Berücksichtigung von $\boldsymbol{\Delta\omega_F}$ und $\boldsymbol{\Delta\Omega}$. D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert[11].

```
dof = x.size-popt.size
chisquare = out.sum_square
chisquare_red = chisquare/dof
prob = round(1-chi2.cdf(chisquare,dof),2)*100
```

Ausgabe der Messergebnisse wird erstellt:

```
print('Steigung:')
print('S_Farbe =', format_e(S_Farbe), ' +- ', format_e(Fehler_S_Farbe))
print('Chi-Quadrat =', chisquare)
print('Freiheitsgrade =', dof)
print('Chi-Quadrat reduziert =', chisquare_red)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob, '%')
print('\n')
print('1.Messmethode:')
print('I_x_1 [kg * m^2] =', format_e(I_x_1), ' +- ', format_e(Fehler_I_x_1))
```

6.3.2 Output

```
Steigung:
S_Farbe = 5.54362e-02  +-  3.58525e-04
Chi-Quadrat = 1.8131534915537362
Freiheitsgrade = 9
Chi-Quadrat reduziert = 0.20146149906152624
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 99.0 %

1.Messmethode:
I_x_1 [kg * m^2] = 4.754426e-03  +-  2.343641e-05
```

Wir erfahren also sofort, dass

$$\boldsymbol{S_{Farbe} = 5.544(36) \times 10^{-2}}$$
$$\boldsymbol{I_{x,1} = 4.754(23) \times 10^{-3}\ kg\ m^2}$$

und als Ergebnis auf unseren Anpassungstest:

$$\boldsymbol{\chi^2_{red} = 2.0 \times 10^{-1}}$$

Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten ist

$$\boldsymbol{P \approx 99.0\%}$$

und wir erhalten das Diagramm in Abb.5.

[11] P. T. Boggs and J. E. Rogers, "Orthogonal Distance Regression," in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989," Contemporary Mathematics, vol. 112, pg. 186, 1990.

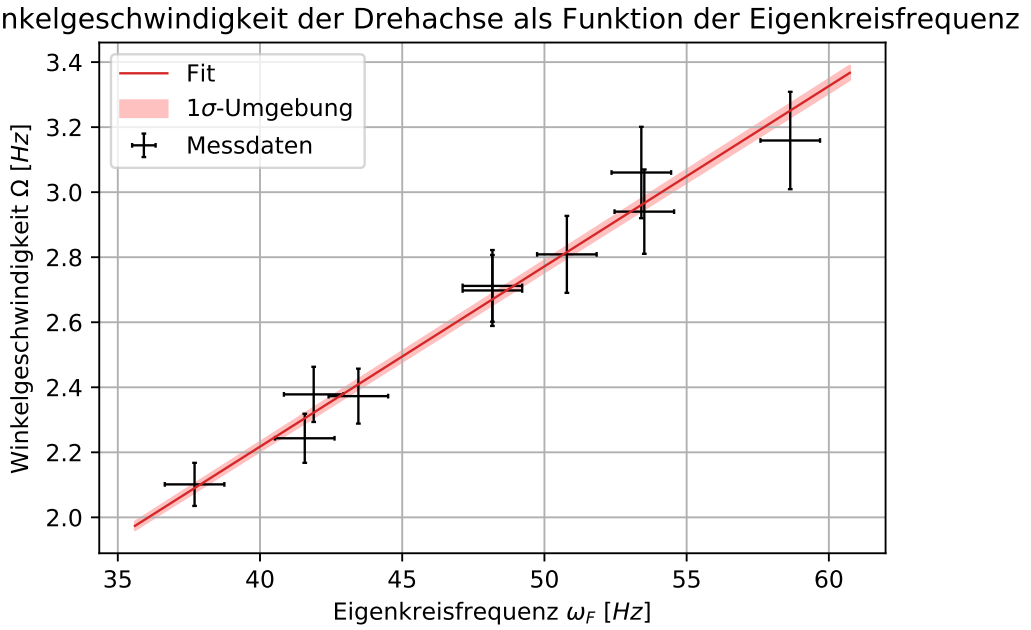

Abbildung 5

6.4 Auswertung

(a) Unsere Beobachtungen lassen eindeutig darauf schließen, dass in unserem theoretischem Modell (14) $\boldsymbol{\Omega}$ und $\boldsymbol{\omega_F}$ das gleiche Vorzeichen haben. Deswegen folgt auch

$$(\boldsymbol{\Delta I = I_x - I_z > 0}) \iff (\boldsymbol{I_x > I_z})$$

Das wird sich ein zweites Mal bestätigen lassen können, wenn wir $\boldsymbol{I_z}$ aus dem vorherigen Versuchsteil mit dem Wert für $\boldsymbol{I_{x,2}}$ aus der 2. Messmethode im nächsten Versuchsteil vergleichen.

(b) Die Winkelgeschwindigkeit der Drehachse als Funktion der Eigenkreisfrequenz ist in Abbildung 5 dargestellt. Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten $\boldsymbol{P \approx 99.0\%}$ ist ein Zeichen dafür, dass wahrscheinlich wieder Messunsicherheiten größer abgeschätzt wurden als sie es sind. Deswegen können wir wieder nur schlecht systematische Fehler in unserem theoretischem Modell feststellen ohne mehr Messdaten zu erheben oder mit anderen Messungen zu vergleichen.

$$\boldsymbol{S_{Farbe} = 5.544(36) \times 10^{-2}}$$
$$\boldsymbol{I_{x,1} = 4.754(23)) \times 10^{-3}\ kg\ m^2}$$

7 Bestimmung von I_x aus der Nutationsfrequenz

7.1 Durchführung[12]

Wir versetzenden kräftefreien Kreisel durch vorsichtiges Anschlagen an die Achse in Nutation. Damit die Näherung

$$\boldsymbol{\omega_N = \frac{I_z}{I_x}\omega_F} \tag{23}$$

gilt, sollte die Öffnung des Nutationskegels an der Spitze des Stabes nur $\boldsymbol{1-2\,cm}$ betragen. Bei unserer Durchführung war dieser Wert eher größer $\boldsymbol{\approx 2-3\,cm}$, weil sich bei Versuchen einen kleineren Nutationskegel zu erhalten, diese so klein ausfielen,dass die Messung mit dem Stroboskop sich als unpraktisch erwies. Nun bestimmt man 10 Wertepaare von $\boldsymbol{f_N}$ und $\boldsymbol{f_F}$. Wir sind dabei folgendermaßen vogegangen: eher langsamere Messung der Nutationsfrequenz mit dem Stroboskop zu Beginn und erst bei der Festlegung von $\boldsymbol{f_N}$ wird $\boldsymbol{f_F}$ mit Hilfe des optischen Drehzahlmessers.

7.2 Messergebnisse

Messdaten wurden dem Versuchsprotokoll (22. Oktober, 2019) entnommen und in Tabelle 5 übertragen.

[12] Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

Tabelle 5: Messung der Nutationsfrequenz

Nutationsfrequenz f_N $[min^{-1}]$	Drehfrequenz f_F $[min^{-1}]$
655 ± 5	693 ± 10
585 ± 5	601 ± 10
435 ± 5	444 ± 10
525 ± 5	574 ± 10
255 ± 5	488 ± 10
780 ± 5	637 ± 10
695 ± 5	737 ± 10
685 ± 5	722 ± 10
395 ± 5	394 ± 10
600 ± 5	629 ± 10

[1] Δf_F grob abgeschätzt
[2] Δf_N abgeschätzt als halbe Skaleneinteilung am Stroboskop

7.3 Kurvenanpassung mit Python

7.3.1 Source Code & Input

Wir drücken f_N über die entsprechende Kreisfrequenz ω_N aus:

$$\omega_N = 2\pi f_N \tag{24}$$

$$\Delta\omega_N = 2\pi(\Delta f_N) \tag{25}$$

Wir gehen davon aus, dass die Präzessionsdauer proportional zur Eigenkreisfrequenz zunimmt. Daher ist unser funktionales Modell für die Ausgleichungsrechnung wie folgt:

$$\boxed{\omega_N = S_N \omega_F} \tag{26}$$

So sieht die Fortführung unserer Python-Implementierung aus:
Messwerte aus Tabelle 5 in SI Einheiten:

```
f_F = np.array([693,601,444,574,488,837,737,722,394,629]) /6
Fehler_f_F = np.full(f_F.size, 10) /60

f_N = np.array([655,585,435,525,455,780,695,685,395,600]) /60
Fehler_f_N = np.full(T_Farbe.size, 5) /60
```

Berechnung der Eigenkreisfrequenz ω_F und $\Delta\omega_F$ nach (6) bzw. (7):

```
omega_F = 2*np.pi*f_F
Fehler_omega_F = 2*np.pi*Fehler_f_F
```

Berechnung der Nutationskreisfrequenz ω_N und $\Delta\omega_N$ nach (24) bzw. (25):

```
omega_N = 2*np.pi*f_N
Fehler_omega_N = 2*np.pi*Fehler_f_N
```

Fitfunktion (26) wird deklariert:

```
from scipy import odr

def fit_func(p, x):
    (s) = p
    return x*s

model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
x = omega_F
y = omega_N
delta_x = Fehler_omega_F
delta_y = Fehler_omega_N
```

Startparameter für Ausgleichungsrechnung werden gesetzt, sodass Lösung konvergiert:

```
para0 = [1]

data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
odr = odr.ODR(data, model, beta0=para0 )
out = odr.run()
```

Auslesen der Messergebnisse:

```
popt = out.beta
perr = out.sd_beta
```

Angabe welche Sigma-Umgebung der Fitfunktion im Diagramm dargestellt werden soll:

```
nstd = 2

popt_top = popt+nstd*perr
popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

```
x_fit = np.linspace(min(x)-(max(x)-min(x))/10, max(x)+(max(x)-min(x))/10, 1000)
fit = fit_func(popt, x_fit)
fit_top = fit_func(popt_top, x_fit)
fit_bot = fit_func(popt_bot, x_fit)
```

Diagramm (Abb.6) wird erstellt:

```
fig, ax = plt.subplots(1)
plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
    Messdaten')
plt.title('Nutationsfrequenz als Funktion der Eigenfrequenz')
plt.grid(True)
plt.xlabel('Eigenkreisfrequenz '+r'${\omega_F}$'+' '+r'${[Hz]}$')
plt.ylabel('Nutationskreisfrequenz '+r'${\omega_N}$'+' '+r'${[Hz]}$')
plt.plot(x_fit, fit, color='C3', lw=1, label='Fit')
ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$'+'-
    Umgebung')
plt.legend(loc='best')

plt.savefig('figures/213_Fig5.pdf', format='pdf', bbox_inches='tight')
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
S_N = popt[0]
Fehler_S_N = perr[0]

I_x_2 = I_z/S_N
Fehler_I_x_2 = abs(I_x_2)*np.sqrt((Fehler_I_z/I_z)**2+(Fehler_S_N/S_N)**2)
```

Der Chi-Quadrat-Test wird durchgeführt unter Berücksichtigung von $\boldsymbol{\Delta\omega_F}$ und $\boldsymbol{\Delta\omega_N}$. D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert[13].

```
dof = x.size-popt.size
chisquare = out.sum_square
chisquare_red = chisquare/dof
prob = round(1-chi2.cdf(chisquare,dof),2)*100
```

Ausgabe der Messergebnisse wird erstellt:

```
print('Steigung: ')
print('S_N =', format_e(S_N), ' +- ', format_e(Fehler_S_N))
print('Chi-Quadrat =', chisquare)
print('Freiheitsgrade =', dof)
print('Chi-Quadrat reduziert =', chisquare_red)
print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob, '%')
print('\n')
print('2.Messmethode: ')
print('I_x [kg * m^2] =', format_e(I_x_2), ' +- ', format_e(Fehler_I_x_2))
```

[13] P. T. Boggs and J. E. Rogers, "Orthogonal Distance Regression," in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989," Contemporary Mathematics, vol. 112, pg. 186, 1990.

7.3.2 Output

```
Steigung:
S_N = 9.474419e-01  +-  6.580497e-03
Chi-Quadrat = 13.315446247417999
Freiheitsgrade = 9
Chi-Quadrat reduziert = 1.4794940274908888
Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 15.0 %

2.Messmethode:
I_x [kg * m^2] = 4.739983e-03  +-  4.033036e-05
```

Wir erfahren also sofort, dass

$$S_N = 9.474(66) \times 10^{-1}$$
$$I_{x,2} = 4.739(40) \times 10^{-3}\ kg\ m^2$$

und als Ergebnis auf unseren Anpassungstest:

$$\chi^2_{red} = 1.5$$

Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten ist

$$P \approx 15.0\%$$

und wir erhalten das Diagramm in Abb.5.

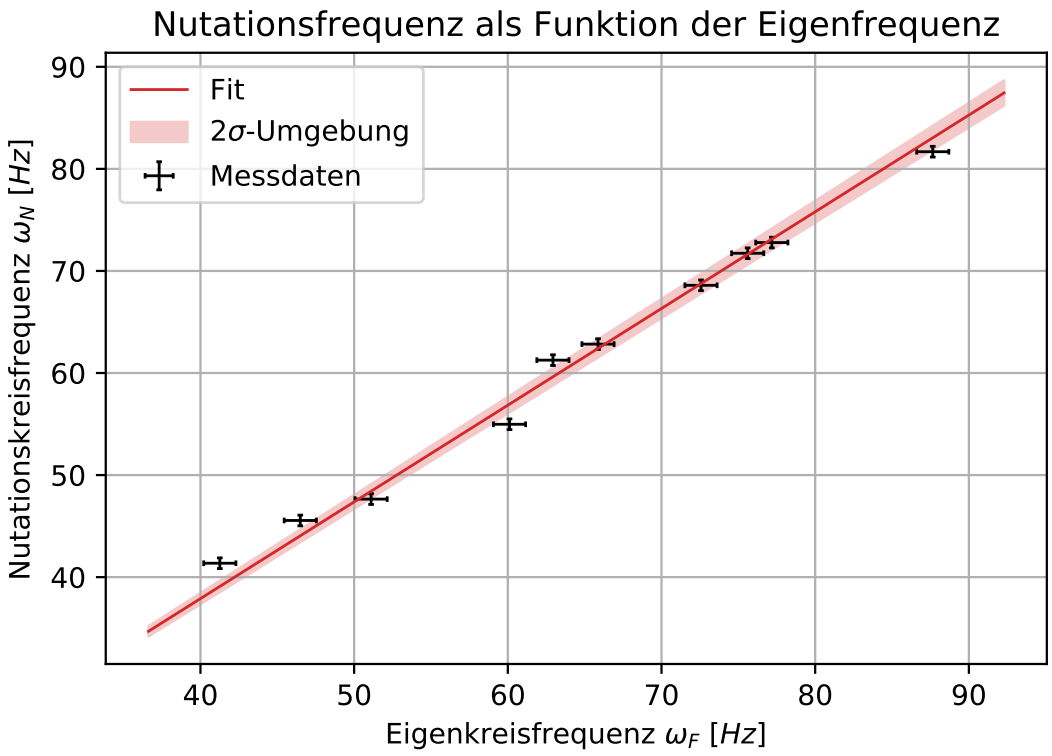

Abbildung 6

7.4 Auswertung

$$S_N = 9.474(66) \times 10^{-1}$$
$$I_{x,2} = 4.739(40)) \times 10^{-3}\ kg\ m^2$$

Die 'Nutationsfrequenz als Funktion der Eigenfrequenz ist in Abbildung 6 dargestellt. Die Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat $P \approx 15.0\%$ ist in Ordnung. Unsere Abschätzungen für die Messunsicherheiten waren also nicht signifikant zu groß. Weil wir in (23) eine Näherung durchgeführt habe, haben wir garantiert systematische

Fehler, aber für die relativ kleinen Nutationswinkel sollten diese im Größenordnungen von einigen Prozent liegen. In diesem Größenbereich halten sich auch unsere statistischen Messfehler auf:

$$\frac{\boldsymbol{\Delta I_{x,2}}}{\boldsymbol{I_{x,2}}} = \mathbf{0.84\%}$$

D.h. es ist eher unwahrscheinlich das der systematische und somit gesammte Fehler von $\boldsymbol{I_{x,2}}$ 5% überschreitet. Wenn aber der systematische Fehler signifikanter ist als der statistische, müsste unser Messwert signifikant von $\boldsymbol{I_{x,1}}$ aus der 1. Messmethode abweichen.

8 Fazit

Die gemessenen Wert für die Dämpfungskonstante $\boldsymbol{k}$ und Halbwertszeit $\boldsymbol{T_{1/2}}$ sind:

$$\boldsymbol{k = 6.600(66) \times 10^{-4}\ Hz}$$
$$\boldsymbol{T_{1/2} = 1.050(11) \times 10^{3}\ s}$$

und für die jeweiligen Trägheitsmomente:

$$\boldsymbol{I_z = 4.491(22) \times 10^{-3}\ kg\ m^2}$$
$$\boldsymbol{I_{x,1} = 4.754(23) \times 10^{-3}\ kg\ m^2}$$
$$\boldsymbol{I_{x,2} = 4.739(40) \times 10^{-3}\ kg\ m^2}$$

Wenn wir uns diese Werte anschauen. Merken wir als erstes, dass $\boldsymbol{I_{x,1}}$ und $\boldsymbol{I_{x,2}}$ jeweils in der 1-Sigma-Umgebung von einander liegen und somit sich nicht signifikant unterscheiden. Daraus können wir folgende Aussagen schließen:

- Die Näherung (23) war gut genug und innerhalb unserer statistischen Messfehler von $\boldsymbol{I_{x,2}}$.
- $\boldsymbol{(I_{x,2} > I_z)}$ mit einer statistischen Signifikanz von

  $$\frac{\boldsymbol{I_{x,2} - I_z}}{\boldsymbol{\sqrt{(\Delta I_{x,2})^2 + (\Delta I_z)^2}}} = \mathbf{5.4}$$

  also mehr als 5-Sigma Gewissheit $\boldsymbol{\approx 1 - \left(3 \times 10^{-8}\right) = 99,999997\%}$. Das bestätigt also nochmal unsere Bestimmung der Richtung von $\boldsymbol{\vec{\Omega}}$ gegenüber $\boldsymbol{\vec{\omega}_F}$.