

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

Praktikanten:  
Gerasimov, V. & Reiter, L.

Betreuer:  
Neitzel, C.

Versuchsdurchführung:  
22. Oktober, 2019

## Inhaltsverzeichnis

	Seite
<b>1 Einführung</b>	<b>2</b>
<b>2 Versuchsaufbau, Literaturwerte &amp; Vorbereitung</b>	<b>3</b>
<b>3 Vorversuch: qualitative Beobachtung des Verhaltens eines Kreisels</b>	<b>3</b>
3.1 Durchführung	3
3.2 Auswertung	4
<b>4 Messung der Reibungsverluste &amp; Bestimmung der Dämpfungskonstante und Halbwertszeit</b>	<b>4</b>
4.1 Durchführung	4
4.2 Messergebnisse	4
4.3 Kurvenanpassung mit Python	5
4.3.1 Source Code & Input	5
4.3.2 Output	7
4.4 Auswertung	8
<b>5 Bestimmung von <math>I_z</math> aus der Präzessionsfrequenz</b>	<b>8</b>
5.1 Durchführung	8
5.2 Messergebnisse	9
5.3 Kurvenanpassung mit Python, Schritt 1	9
5.3.1 Source Code & Input	9
5.3.2 Output	13
5.4 Kurvenanpassung mit Python, Schritt 2	13
5.4.1 Source Code & Input	13
5.4.2 Output	16
5.5 Auswertung	16
<b>6 Bestimmung von <math>I_x</math> aus der Größe und der Richtung der Umlaufgeschwindigkeit</b>	<b>17</b>
6.1 Durchführung	17
6.2 Messergebnisse	17
6.3 Kurvenanpassung mit Python	18
6.3.1 Source Code & Input	18
6.3.2 Output	20
6.4 Auswertung	21
<b>7 Bestimmung von <math>I_x</math> aus der Nutationsfrequenz</b>	<b>21</b>
7.1 Durchführung	21
7.2 Messergebnisse	21
7.3 Kurvenanpassung mit Python	22
7.3.1 Source Code & Input	22
7.3.2 Output	24
7.4 Auswertung	24
<b>8 Fazit</b>	<b>25</b>

## 1 Einführung<sup>1</sup>

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  $k$  und Halbwertszeit  $T_{1/2}$ . Wir bestimmen aus der Präzessionsfrequenz eines schweren Kreisels sein Trägheitsmoment  $I_z$  um die Figurenachse  $\vec{Z}$ . Das Trägheitsmoment  $I_x$  senkrecht zur Figurenachse bestimmt wir aus der Größe und der Richtung der Umlaufgeschwindigkeit  $\vec{\Omega}$  der momentanen Drehachse um die Figurenachse des Kreisels und zusätzlich ein zweites Mal aus der gemessenen Nutationsfrequenz  $f_N$ .

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

<sup>1</sup>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 führt die Figuren- achse eine Eigendrehung mit  $\omega_F$  durch und rotiert mit der Nutationsfrequenz  $\omega_N$  gleichzeitig um die raumfeste Drehimpulsachse. Die Winkelgeschwindigkeit ist nicht konstant sondern bewegt sich mit  $\Omega$  um die Figuren- achse  $\vec{Z}$ . Diese Bewegung kann mit Hilfe einer farbigen Sektorscheibe beobachtet werden. Die Bewegungen der cha- rakteristischen 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  $\omega_P$  auf einem Kegelmantel um die Richtung der Gewichtskraft.

## 2 Versuchsaufbau<sup>2</sup>, Literaturwerte & Vorbereitung

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

## 3 Vorversuch: qualitative Beobachtung des Verhaltens eines Kreisels

### 3.1 Durchführung<sup>2</sup>

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

- 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.
- 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 Figuren- achse  $\vec{Z}$  wandernden Ort der momentanen Drehachse  $\vec{\omega}$  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 Farb- bringe verläuft, ändert sich die Farbe am Ort der momentanen Drehachse nicht, d.h.  $\vec{\omega}$  läuft auf einem Kreis um  $\vec{Z}$ .

- 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 Aufnahme- achse verschoben ist. Liegt keine Nutation vor ( $\vec{\omega} || \vec{Z}$ ), 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

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.

- 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 wird die Dreh- richtung des Kreisels geändert und der Versuch wiederholt.

### 3.2 Auswertung

- 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 Figuren- achse, weil auf den Kreisel ein äußeres Drehmoment  $\vec{M}$  wirkt.

Nachdem die Farbscheibe mit den Sektoren angebracht und ausbalanciert wurde, verschwindet das äußere Dreh- moment und der nun kräftefreie Kreisel behält seinen Drehimpuls bei (im Falle von vernachlässigbarer Reibung). Die Figuren- achse  $\vec{Z}$  ändert sich jetzt 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.

- 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.

- 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 unge- fähr in die ursprüngliche Richtung der Drehachse (bevor die Nutation eingeleitet wurde) und die Figuren- achse  $\vec{Z}$  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 Figuren- achse jetzt eine Kombination aus Nutations- und Präzessionsbewegung vollführt.

- Ohne Farbscheibe und Zusatzgewichte sind die Richtungen der Eigendrehung und Präzessionsdrehung entge- gengesetzt 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 Dämpfungskonstan- te und Halbwertszeit

### 4.1 Durchführung<sup>4</sup>

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  $f_0 \approx 600 - 700 \text{ min}^{-1}$  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.

<sup>4</sup>Dr. J. Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

<sup>2</sup>Dr. J. Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

<sup>3</sup>Dr. J. Wagner - Physikalisches Anfängerpraktikum - V. 1.2 Stand 06/2016, Versuch 14

Tabelle 1: Messung der Drehfrequenz

Zeit $t$ [s]	Drehfrequenz $f$ [min <sup>-1</sup> ]
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

<sup>1</sup> sowohl  $\Delta t$  als auch  $\Delta f$   
sind grob abgeschätzt

### 4.3 Kurvenanpassung mit Python

#### 4.3.1 Source Code & Input

Wir drücken zuerst  $f$  über die entsprechende Kreisfrequenz  $\omega$  aus:

$$\omega = 2\pi f \quad (1)$$

$$\Delta\omega = 2\pi(\Delta f) \quad (2)$$

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

$$\omega = \omega_0 e^{-kt} \quad (3)$$

So sieht unsere Python-Implementierung aus:

Header:

```
1 %matplotlib inline
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from scipy.stats import chi2
5 from decimal import Decimal
6
7 def format_e(n):
8     a = '%e' % Decimal(n)
9     return a.split('e')[0].rstrip('0').rstrip('.')+'e'+a.split('e')[1]
```

Messwerte aus Tabelle 1 in SI Einheiten:

```
1 t = np.array([0,120,240,360,480,600,720])
2 Fehler_t = np.array([2,2,2,2,2,2,2])
3
4 f = np.array([648,600,552,515,470,437,403]) / 60
5 Fehler_f = np.array([10,10,10,10,10,10,10]) / 60
```

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

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

Fitfunktion (3) wird deklariert:

```
1 from scipy import odr
2
3 def fit_func(p, x):
4     (A, k) = p
5     return A*np.exp(-k*x)
6
7 model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
1 x = t
2 y = omega
3 delta_x = Fehler_t
4 delta_y = Fehler_omega
```

Startparameter für Ausgleichsrechnung werden gesetzt, sodass Lösung konvergiert:

```
1 para0 = [1, 0]
2
3 data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
4 odr = odr.ODR(data, model, beta0=para0)
5 out = odr.run()
```

Endgültige Ausgleichsparameter und ihre Kovarianzmatrix werden ausgelesen:

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

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

```
1 nstd = 10
2
3 popt_top = popt+nstd*perr
4 popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

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

Diagramm (Abb.2) wird erstellt:

```
1 fig, ax = plt.subplots(1)
2 plt.ticklabel_format(axis='both', style='sci', scilimits=(0,3), useMathText=True)
3 plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
Messdaten')
4 plt.title('Kreisfrequenz als Funktion der Zeit')
5 plt.grid(True)
6 plt.yscale('log')
7 plt.xlabel('Zeit ' + r'$t$' + ' ' + r'$[s]$')
8 plt.ylabel('Kreisfrequenz ' + r'$\omega$' + ' ' + r'$[Hz]$')
9 plt.plot(x_fit, fit, 'r', lw=1, label='Fit')
10 ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$' + '
Umgebung')
11 plt.legend(loc='best')
12
13 plt.savefig('figures/213_Fig1.pdf', format='pdf', bbox_inches='tight')
```

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

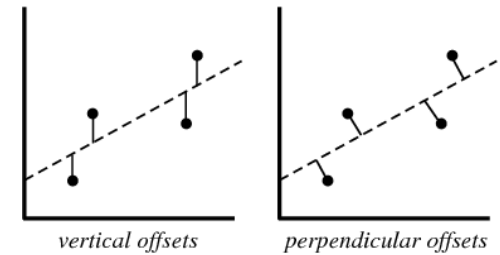


Abbildung 1: Abstände von Messdaten zur Fitfunktion<sup>6</sup>

<sup>5</sup> 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.

<sup>6</sup> <http://mathworld.wolfram.com/LeastSquaresFitting.html>

```

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

```

Auslesen der Messergebnisse:

```

1 omega_0 = popt[0]
2 Fehler_omega_0 = perr[0]
3
4 k = popt[1]
5 Fehler_k = perr[1]

```

Die Halbwertszeit können wir nun sofort aus der Dämpfungskonstante  $k$  berechnen:

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

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

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

```

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

```

Ausgabe der Messergebnisse wird erstellt:

```

1 print('Startwert der Kreisfrequenz: ')
2 print('omega_0 [Hz] =', format_e(omega_0), ' +/- ', format_e(Fehler_omega_0))
3 print('Chi-Quadrat =', chisquare)
4 print('Freiheitsgrade =', dof)
5 print('Chi-Quadrat reduziert =', chisquare_red)
6 print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob, '%')
7 print('\n')
8 print('Dämpfungskonstante: ')
9 print('k [Hz] =', format_e(k), ' +/- ', format_e(Fehler_k))
10 print('\n')
11 print('Halbwertszeit: ')
12 print('T_halb [s] =', format_e(T_halb), ' +/- ', format_e(Fehler_T_halb))

```

### 4.3.2 Output

```

1 Startwert der Kreisfrequenz:
2 omega_0 [Hz] = 6.791995e+01 +/- 1.653924e-01
3 Chi-Quadrat = 0.2262760874189914
4 Freiheitsgrade = 5
5 Chi-Quadrat reduziert = 0.04525521748379828
6 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 100.0 %
7
8 Dämpfungskonstante:
9 k [Hz] = 6.600142e-04 +/- 6.605926e-06
10
11 Halbwertszeit:
12 T_halb [s] = 1.0502e+03 +/- 1.051121e+01

```

Wir erfahren also sofort, dass

$$k = 6.600(66) \times 10^{-4} \text{ Hz}$$

$$T_{1/2} = 1.050(11) \times 10^3 \text{ s}$$

und als Ergebnis auf unseren Anpassungstest:

$$\chi_{red}^2 = 4.5 \times 10^{-2}$$

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

$$P \approx 100.0\%$$

und wir erhalten das Diagramm in Abb.2.

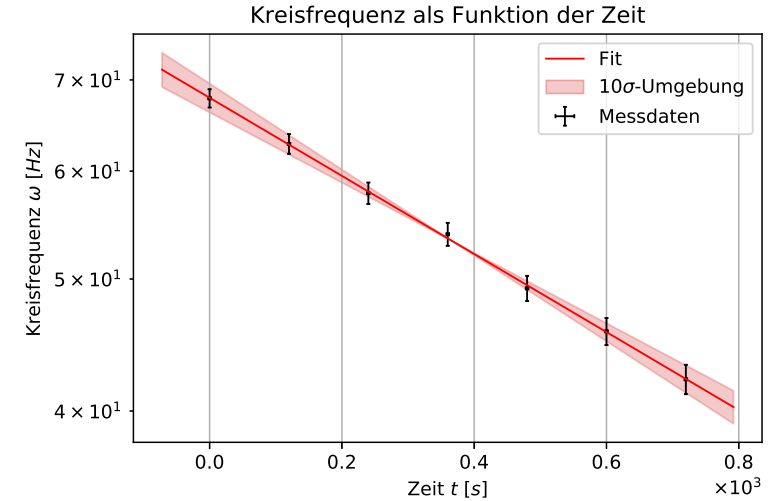


Abbildung 2

## 4.4 Auswertung

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

$$k = 6.600(66) \times 10^{-4} \text{ Hz}$$

$$T_{1/2} = 1.050(11) \times 10^3 \text{ s}$$

Der jedoch geringe Wert  $\chi_{red}^2 = 4.5 \times 10^{-2}$  sagt uns, dass die bei der Messung geschätzten Messunsicherheiten zu groß abgeschätzt worden sind. D.h.  $\Delta k$  und  $\Delta T_{1/2}$  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  $\Delta k$  und  $\Delta T_{1/2}$  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<sup>7</sup>

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.

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

- Wir belasten den kräftefreien Kreisel mit folgenden Zusatzmassen:

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

<sup>7</sup>Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

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

Gemessen wird mit der Stoppuhr die Präzessionsdauer  $T_P$  für jede Gewichtseinstellung bei jeweils vier verschiedene Frequenzen  $f_F$  im Bereich  $250 \text{ min}^{-1} < f_F < 700 \text{ min}^{-1}$ . Zur ersten Messreihe (\*) können wir auch die Messdaten aus der vorherigen Aufgabe hinzufügen, da es genau die gleichen Messkriterien ( $n = 1, l = 20 \text{ cm}$ ) erfüllt. Bei 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  $f_F$  und die Präzessionsdauer  $T_P$ .

## 5.2 Messergebnisse

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

Tabelle 2: Messung der Präzessionsdauer

Messreihe <i>Aufgabe<sub>Nr.</sub></i>	Eigenfrequenz $f_F$ [ $\text{min}^{-1}$ ]	Präzessionsdauer $T_P$ [s]	Abstand $l$ [cm]	Anzahl der Gewichte $n$ [1]
(a)	$500 \pm 10$	$74.80 \pm 0.30$	$20.0 \pm 0.1$	1
"	$500 \pm 10$	$75.00 \pm 0.30$	"	"
"	$500 \pm 10$	$74.80 \pm 0.30$	"	"
(b <sub>1</sub> )(*)	$298 \pm 10$	$47.90 \pm 0.30$	$20.0 \pm 0.1$	1
"	$404 \pm 10$	$64.00 \pm 0.30$	"	"
"	$501 \pm 10$	$78.50 \pm 0.30$	"	"
"	$602 \pm 10$	$92.51 \pm 0.30$	"	"
(b <sub>2</sub> )	$297 \pm 10$	$61.46 \pm 0.30$	$15.0 \pm 0.1$	1
"	$407 \pm 10$	$85.31 \pm 0.30$	"	"
"	$495 \pm 10$	$101.56 \pm 0.30$	"	"
"	$601 \pm 10$	$122.67 \pm 0.30$	"	"
(b <sub>3</sub> )	$599 \pm 10$	$44.76 \pm 0.30$	$20.0 \pm 0.1$	2
"	$502 \pm 10$	$38.28 \pm 0.30$	"	"
"	$403 \pm 10$	$30.72 \pm 0.30$	"	"
"	$301 \pm 10$	$23.04 \pm 0.30$	"	"
(b <sub>4</sub> )	$603 \pm 10$	$61.06 \pm 0.30$	$15.0 \pm 0.1$	2
"	$499 \pm 10$	$50.67 \pm 0.30$	"	"
"	$300 \pm 10$	$32.23 \pm 0.30$	"	"
"	$298 \pm 10$	$31.06 \pm 0.30$	"	"

<sup>1</sup>  $\Delta f_F$  grob abgeschätzt

<sup>2</sup>  $\Delta T_P$  grob abgeschätzt als durchschnittliche menschliche Reaktionszeit von 200 ms und Faktor  $\sqrt{2}$ , weil zum Messen die Stoppuhr doppelt betätigt werden muss:

$$0.2 \text{ s} \times \sqrt{2} \approx 0.3 \text{ s} = \Delta T_P$$

(\*) Zu dieser Messreihe wird im weiteren Verlauf auch Messreihe (a) gezählt

eines einzelnen Gewichts beträgt

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

und die Erdbeschleunigung nähern wir mit

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

## 5.3 Kurvenanpassung mit Python, Schritt 1

### 5.3.1 Source Code & Input

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

$$\omega_F = 2\pi f_F \quad (6)$$

$$\Delta\omega_F = 2\pi(\Delta f_F) \quad (7)$$

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

$$T_P = S\omega_F \quad (8)$$

So sieht die Fortführung unserer Python-Implementierung aus:

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

```
1 f_F = np.array([500,500,500,298,404,501,602,297,407,495, 601,599,502,403,301,603,499,300,298]) /60 ]
2 Fehler_f_F = np.full(f_F.size, 10) /60
3
4 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,
5                38.28,30.72,23.04,61.06,50.67,32.23,31.06])
6 Fehler_T_P = np.full(T_P.size, 0.30)
```

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

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

Fitfunktion (8) wird deklariert:

```
1 from scipy import odr
2
3 def fit_func(p, x):
4     (s) = p
5     return s*x
6
7 model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben (aufgeteilt auf die Messreihen (a) + (b<sub>1</sub>), (b<sub>2</sub>), (b<sub>3</sub>) und (b<sub>4</sub>)):

```
1 x_1 = omega_F[0:7]
2 y_1 = T_P[0:7]
3 delta_x_1 = Fehler_omega_F[0:7]
4 delta_y_1 = Fehler_T_P[0:7]
5
6 x_2 = omega_F[7:11]
7 y_2 = T_P[7:11]
8 delta_x_2 = Fehler_omega_F[7:11]
9 delta_y_2 = Fehler_T_P[7:11]
10
11 x_3 = omega_F[11:15]
12 y_3 = T_P[11:15]
13 delta_x_3 = Fehler_omega_F[11:15]
14 delta_y_3 = Fehler_T_P[11:15]
15
16 x_4 = omega_F[15:19]
17 y_4 = T_P[15:19]
18 delta_x_4 = Fehler_omega_F[15:19]
19 delta_y_4 = Fehler_T_P[15:19]
```

Startparameter für Ausgleichsrechnung werden gesetzt, sodass Lösung konvergiert:

```
1 para0 = [1]
2
3 data_1 = odr.RealData(x_1, y_1, sx=delta_x_1, sy=delta_y_1)
4 odr_1 = odr.ODR(data_1, model, beta0=para0)
5 out_1 = odr_1.run()
6
7 data_2 = odr.RealData(x_2, y_2, sx=delta_x_2, sy=delta_y_2)
8 odr_2 = odr.ODR(data_2, model, beta0=para0)
9 out_2 = odr_2.run()
10
11 data_3 = odr.RealData(x_3, y_3, sx=delta_x_3, sy=delta_y_3)
12 odr_3 = odr.ODR(data_3, model, beta0=para0)
13 out_3 = odr_3.run()
14
15 data_4 = odr.RealData(x_4, y_4, sx=delta_x_4, sy=delta_y_4)
16 odr_4 = odr.ODR(data_4, model, beta0=para0)
17 out_4 = odr_4.run()
```

Endgültige Ausgleichsparameter und ihre Kovarianzmatrix werden ausgelesen:

```

1  popt_1 = out_1.beta
2  perr_1 = out_1.sd_beta
3
4  popt_2 = out_2.beta
5  perr_2 = out_2.sd_beta
6
7  popt_3 = out_3.beta
8  perr_3 = out_3.sd_beta
9
10 popt_4 = out_4.beta
11 perr_4 = out_4.sd_beta

```

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

```

1  nstd = 1
2
3  popt_top_1 = popt_1+nstd*perr_1
4  popt_bot_1 = popt_1-nstd*perr_1
5
6  popt_top_2 = popt_2+nstd*perr_2
7  popt_bot_2 = popt_2-nstd*perr_2
8
9  popt_top_3 = popt_3+nstd*perr_3
10 popt_bot_3 = popt_3-nstd*perr_3
11
12 popt_top_4 = popt_4+nstd*perr_4
13 popt_bot_4 = popt_4-nstd*perr_4

```

Plot-Umgebung wird angegeben:

```

1  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)
2  fit_1 = fit_func(popt_1, x_fit_1)
3  fit_top_1 = fit_func(popt_top_1, x_fit_1)
4  fit_bot_1 = fit_func(popt_bot_1, x_fit_1)
5
6  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)
7  fit_2 = fit_func(popt_2, x_fit_2)
8  fit_top_2 = fit_func(popt_top_2, x_fit_2)
9  fit_bot_2 = fit_func(popt_bot_2, x_fit_2)
10
11 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)
12 fit_3 = fit_func(popt_3, x_fit_3)
13 fit_top_3 = fit_func(popt_top_3, x_fit_3)
14 fit_bot_3 = fit_func(popt_bot_3, x_fit_3)
15
16 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)
17 fit_4 = fit_func(popt_4, x_fit_4)
18 fit_top_4 = fit_func(popt_top_4, x_fit_4)
19 fit_bot_4 = fit_func(popt_bot_4, x_fit_4)

```

Diagramm (Abb.3) wird erstellt:

```

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

```

```

18 ax.fill_between(x_fit_4, fit_top_4, fit_bot_4, color='C1', alpha=.25, label=str(nstd)+r'\sigma$'+
19               'Umgebung 4')
20 box = ax.get_position()
21 ax.set_position([box.x0, box.y0, box.width * 0.8, box.height])
22 ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
23 plt.savefig('figures/213_Fig2.pdf', format='pdf', bbox_inches='tight')

```

Auslesen der Messergebnisse:

```

1  S_1 = popt_1[0]
2  Fehler_S_1 = perr_1[0]
3  S_2 = popt_2[0]
4  Fehler_S_2 = perr_2[0]
5  S_3 = popt_3[0]
6  Fehler_S_3 = perr_3[0]
7  S_4 = popt_4[0]
8  Fehler_S_4 = perr_4[0]

```

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

```

1  dof_1 = x_1.size-popt.size
2  chisquare_1 = out_1.sum_square
3  chisquare_red_1 = chisquare_1/dof_1
4  prob_1 = round(1-chi2.cdf(chisquare_1,dof_1),2)*100
5
6  dof_2 = x_2.size-popt.size
7  chisquare_2 = out_2.sum_square
8  chisquare_red_2 = chisquare_2/dof_2
9  prob_2 = round(1-chi2.cdf(chisquare_2,dof_2),2)*100
10
11 dof_3 = x_3.size-popt.size
12 chisquare_3 = out_3.sum_square
13 chisquare_red_3 = chisquare_3/dof_3
14 prob_3 = round(1-chi2.cdf(chisquare_3,dof_3),2)*100
15
16 dof_4 = x_4.size-popt.size
17 chisquare_4 = out_4.sum_square
18 chisquare_red_4 = chisquare_4/dof_4
19 prob_4 = round(1-chi2.cdf(chisquare_4,dof_4),2)*100

```

Ausgabe der Messergebnisse wird erstellt:

```

1  print('Steigungen: ')
2  print('S_1 [s^2] =', format_e(S_1), ' +- ', format_e(Fehler_S_1))
3  print('Chi-Quadrat =', chisquare_1)
4  print('Freiheitsgrade =', dof_1)
5  print('Chi-Quadrat reduziert =', chisquare_red_1)
6  print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_1, '%')
7  print('\n')
8  print('S_2 [s^2] =', format_e(S_2), ' +- ', format_e(Fehler_S_2))
9  print('Chi-Quadrat =', chisquare_2)
10 print('Freiheitsgrade =', dof_2)
11 print('Chi-Quadrat reduziert =', chisquare_red_2)
12 print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_2, '%')
13 print('\n')
14 print('S_3 [s^2] =', format_e(S_3), ' +- ', format_e(Fehler_S_3))
15 print('Chi-Quadrat =', chisquare_3)
16 print('Freiheitsgrade =', dof_3)
17 print('Chi-Quadrat reduziert =', chisquare_red_3)
18 print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_3, '%')
19 print('\n')
20 print('S_4 [s^2] =', format_e(S_4), ' +- ', format_e(Fehler_S_4))
21 print('Chi-Quadrat =', chisquare_4)
22 print('Freiheitsgrade =', dof_4)
23 print('Chi-Quadrat reduziert =', chisquare_red_4)
24 print('Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten =', prob_4, '%')
25 print('\n')

```

<sup>8</sup>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

```

1 Steigungen:
2 S_1 [s^2] = 1.465235e+00 +- 1.366893e-02
3 Chi-Quadrat = 8.120624052429571
4 Freiheitsgrade = 5
5 Chi-Quadrat reduziert = 1.6241248104859143
6 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 15.0 %
7
8 S_2 [s^2] = 1.965064e+00 +- 1.134583e-02
9 Chi-Quadrat = 0.8421813756024409
10 Freiheitsgrade = 2
11 Chi-Quadrat reduziert = 0.42109068780122044
12 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 66.0 %
13
14 S_3 [s^2] = 7.224199e-01 +- 4.304101e-03
15 Chi-Quadrat = 0.7948108139557275
16 Freiheitsgrade = 2
17 Chi-Quadrat reduziert = 0.39740540697786375
18 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 67.0 %
19
20 S_4 [s^2] = 9.780491e-01 +- 1.114486e-02
21 Chi-Quadrat = 2.8383188936144617
22 Freiheitsgrade = 2
23 Chi-Quadrat reduziert = 1.4191594468072308
24 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 24.0 %

```

Wir erfahren also sofort, dass

$$S_1 = 1.465(14) \text{ s}^2$$

$$S_2 = 1.965(11) \text{ s}^2$$

$$S_3 = 7.224(43) \times 10^{-1} \text{ s}^2$$

$$S_4 = 9.78(11) \times 10^{-1} \text{ s}^2$$

und als Ergebnis auf unseren Anpassungstests:

$$\chi_{red,1}^2 = 1.6$$

$$\chi_{red,2}^2 = 4.2 \times 10^{-1}$$

$$\chi_{red,3}^2 = 4.0 \times 10^{-1}$$

$$\chi_{red,4}^2 = 1.4$$

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

$$P_1 \approx 15.0\%$$

$$P_2 \approx 66.0\%$$

$$P_3 \approx 67.0\%$$

$$P_4 \approx 24.0\%$$

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 Ausgleichsrechnung wie folgt:

$$I_z = \text{konst.} \quad (9)$$

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

$$g = 9.80984(2) \text{ m s}^{-2}$$

$$m_G = 9,85(1) \text{ g}$$

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

$$m = m_G \times n \quad (10)$$

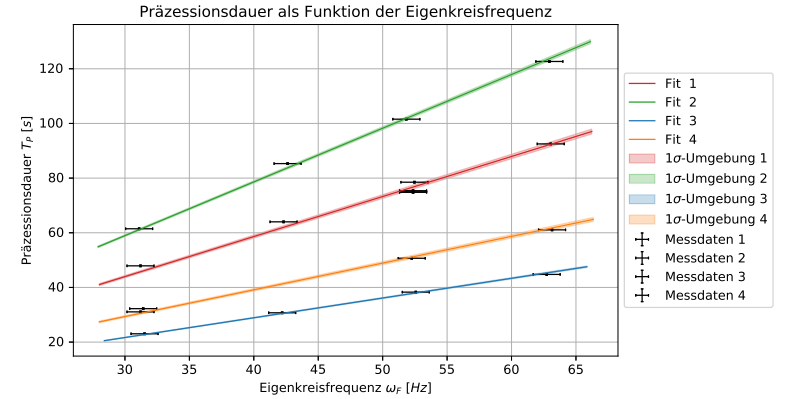


Abbildung 3

$$\Delta m = \Delta m_G \times \sqrt{n} \quad (11)$$

$$I_z = \frac{mglS}{2\pi} \quad (12)$$

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

So sieht die Fortführung unserer Python-Implementierung aus:

Messwerte der Steigung S aus Schritt 1 in SI Einheiten:

```

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

```

Messwerte aus Tabelle 2 in SI Einheiten:

```

1 l = np.array([20,15,20,15]) /1e2
2 Fehler_l = np.full(l.size, 0.1) /1e2
3
4 n = np.array([1,1,2,2])

```

Angaben aus "Versuchsaufbau, Literaturwerte & Vorbereitung" in SI Einheiten:

```

1 m_G = 9.85 /1e3
2 Fehler_m_G = 0.01 /1e3
3
4 g = 9.80984
5 Fehler_g = 0.00002

```

Berechnung der Masse  $m$  und  $\Delta m$  nach (10) bzw. (11):

```

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

```

Berechnung des Trägheitsmoments  $I_z$  und  $\Delta I_z$  nach (12) bzw. (13):

```

1 I_z = m*g*l*S/(2*np.pi)
2 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:

```

1 from scipy import odr
2
3 def fit_func(p, x):
4     (c) = p
5     return x*0+c
6
7 model = odr.Model(fit_func)

```

darzustellende Daten werden übergeben:

```
1 x = S
2 y = I_z
3 delta_x = Fehler_S
4 delta_y = Fehler_I_z
```

Startparameter für Ausgleichsrechnung werden gesetzt, sodass Lösung konvergiert:

```
1 para0 = [0]
2
3 data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
4 odr = odr.ODR(data, model, beta0=para0)
5 out = odr.run()
```

Endgültige Ausgleichsparameter und ihre Kovarianzmatrix werden ausgelesen:

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

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

```
1 nstd = 1
2
3 popt_top = popt+nstd*perr
4 popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

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

Diagramm (Abb.4) wird erstellt:

```
1 fig, ax = plt.subplots(1)
2 plt.ticklabel_format(axis='both', style='sci', scilimits=(0,3), useMathText=True)
3 plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
Messdaten')
4 plt.title('Trägheitsmoment')
5 plt.grid(True)
6 plt.xlabel('Steigung ' + r'$S$' + ' ' + r'$[s^2]$')
7 plt.ylabel('Trägheitsmoment ' + r'$I_z$' + ' ' + r'$[kg \cdot m^2]$')
8 plt.plot(x_fit, fit, color='r', lw=1, label='Fit')
9 ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$' + '-
Umgebung')
10 plt.legend(loc='best')
11
12 plt.savefig('figures/213_Fig3.pdf', format='pdf', bbox_inches='tight')
```

Auslesen der Messergebnisse:

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

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

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

Ausgabe der Messergebnisse wird erstellt:

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

<sup>9</sup>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

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

Wir erfahren also sofort, dass

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

und als Ergebnis auf unseren Anpassungstest:

$$\chi_{red}^2 = 1.1$$

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

$$P \approx 37.0\%$$

und wir erhalten das Diagramm in Abb.4.

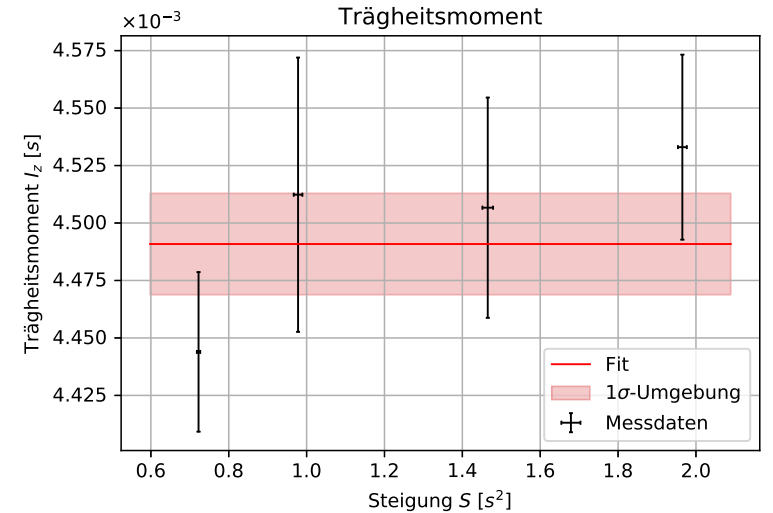


Abbildung 4

## 5.5 Auswertung

- 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  $f_F$  und Drehmoment  $M$  bzw. Gewichtskonfiguration  $\{n, l\}$ , 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.
- Die gemessenen Werte für die 4 Steigungen  $S_i$  und  $I_z$  machen Sinn und passen zu unserem theoretischen Modellen



Tabelle 3: Messung der Präzessionsdauer

Messreihe <i>Aufgabe<sub>Nr.</sub></i>	Eigenfrequenz $f_F$ [ $\text{min}^{-1}$ ]	Präzessionsdauer $T_P$ [s]	Abstand $l$ [cm]	Anzahl der Gewichte $n$ [1]
(a)	$500 \pm 10$	$74.80 \pm 0.30$	$20.0 \pm 0.1$	1
"	$500 \pm 10$	$75.00 \pm 0.30$	"	"
"	$500 \pm 10$	$74.80 \pm 0.30$	"	"

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

(8) und (9) :

$$S_1 = 1.465(14) \text{ s}^2$$

$$S_2 = 1.965(11) \text{ s}^2$$

$$S_3 = 7.224(43) \times 10^{-1} \text{ s}^2$$

$$S_4 = 9.78(11) \times 10^{-1} \text{ s}^2$$

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

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

$$P_1 \approx 15.0\%$$

$$P_2 \approx 66.0\%$$

$$P_3 \approx 67.0\%$$

$$P_4 \approx 24.0\%$$

$$P \approx 37.0\%$$

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<sup>10</sup>

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.

- Die Umlaufrichtung der momentanen Drehachse wird mit Hilfe der Farbscheibe bestimmt. Dafür beachten wir die Reihenfolge der Farben im einfarbigen Punkt.
- Mit der Stoppuhr messen wir für 10 Frequenzen im Bereich  $300 \text{ min}^{-1} < f_F < 600 \text{ min}^{-1}$  jeweils die Zeit  $T_{10}$  für 10 Umläufe der momentanen Drehachse um die Figurenachse. Die Frequenz des Farbwechsels  $\Omega$  der Sektorscheib entspricht genau der Umlaufgeschwindigkeit  $\Omega$  der momentanen Drehachse um die Figurenachse des Kreisels.

### 6.2 Messergebnisse

- 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

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

$$\Omega = \frac{I_x - I_z}{I_x} \omega_F \quad (14)$$

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

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

Tabelle 4: Messung der Periode vom Farbwechsel

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

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

## 6.3 Kurvenanpassung mit Python

### 6.3.1 Source Code & Input

Wir können sofort folgende Zusammenhänge schließen:

$$T_{Farbe} = T_{10}/10 \quad (15)$$

$$\Delta T_{Farbe} = (\Delta T_{10})/10 \quad (16)$$

$$|\Omega| = \frac{2\pi}{T_{Farbe}} \quad (17)$$

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

$$\Omega = \frac{2\pi}{T_{Farbe}} \quad (18)$$

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

Wir gehen davon aus, dass die Umlaufgeschwindigkeit  $\Omega$  proportional zur Eigenkreisfrequenz zunimmt. Daher ist unser funktionales Modell für die Ausgleichsrechnung wie folgt:

$$\boxed{\Omega = S_{Farbe} \omega_F} \quad (20)$$

Aus (14) und (20) folgt außerdem:

$$I_{x,1} = \frac{I_z}{1 - S_{Farbe}} \quad (21)$$

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

<sup>10</sup>Dr. J. Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

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

So sieht die Fortführung unserer Python-Implementierung aus:

Messwerte aus Tabelle 4 in SI Einheiten:

```
1 f_F = np.array([485,360,511,397,460,415,560,510,460,400]) /60
2 Fehler_f_F = np.full(f_F.size, 10) /60
3
4 T_10 = np.array([22.37,29.90,21.37,28.01,23.29,26.48,19.89,20.53,23.17,26.42])
5 Fehler_T_10 = np.full(T_Farbe.size, 0.30)
```

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

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

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

```
1 omega_F = 2*np.pi*f_F
2 Fehler_omega_F = 2*np.pi*Fehler_f_F
3
4 Omega = 2*np.pi/T_Farbe
5 Fehler_Omega = 2*np.pi*Omega*Fehler_T_Farbe/T_Farbe
```

Fitfunktion (20) wird deklariert:

```
1 from scipy import odr
2
3 def fit_func(p, x):
4     (s) = p
5     return x*s
6
7 model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
1 x = omega_F
2 y = Omega
3 delta_x = Fehler_omega_F
4 delta_y = Fehler_Omega
```

Startparameter für Ausgleichsrechnung werden gesetzt, sodass Lösung konvergiert:

```
1 para0 = [1]
2
3 data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
4 odr = odr.ODR(data, model, beta0=para0 )
5 out = odr.run()
```

Endgültige Ausgleichsparameter und ihre Kovarianzmatrix werden ausgelesen:

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

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

```
1 nstd = 1
2
3 popt_top = popt+nstd*perr
4 popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

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

Diagramm (Abb.5) wird erstellt:

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

```
4 plt.grid(True)
5 plt.xlabel('Eigenkreisfrequenz ' + r'${\omega_F}$' + ' ' + r'${[Hz]}$')
6 plt.ylabel('Winkelgeschwindigkeit ' + r'${\Omega}$' + ' ' + r'${[Hz]}$')
7 plt.plot(x_fit, fit, color='C3', lw=1, label='Fit')
8 ax.fill_between(x_fit, fit_top, fit_bot, color='r', alpha=.25, label=str(nstd)+r'${\sigma}$' + '-
9 Umgebung')
10 plt.legend(loc='best')
11 plt.savefig('figures/213_Fig4.pdf', format='pdf', bbox_inches='tight')
```

Auslesen der Messergebnisse:

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

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

```
1 I_x_1 = I_z / (1 - S_Farbe)
2 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  $\Delta\omega_F$  und  $\Delta\Omega$ . D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert<sup>11</sup>.

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

Ausgabe der Messergebnisse wird erstellt:

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

### 6.3.2 Output

```
1 Steigung:
2 S_Farbe = 5.54362e-02 +- 3.58525e-04
3 Chi-Quadrat = 1.8131534915537362
4 Freiheitsgrade = 9
5 Chi-Quadrat reduziert = 0.20146149906152624
6 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 99.0 %
7
8 1.Messmethode:
9 I_x_1 [kg * m^2] = 4.754426e-03 +- 2.343641e-05
```

Wir erfahren also sofort, dass

$$S_{Farbe} = 5.544(36) \times 10^{-2}$$

$$I_{x,1} = 4.754(23) \times 10^{-3} \text{ kg m}^2$$

und als Ergebnis auf unseren Anpassungstest:

$$\chi_{red}^2 = 2.0 \times 10^{-1}$$

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

$$P \approx 99.0\%$$

und wir erhalten das Diagramm in Abb.5.

<sup>11</sup>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.

Winkelgeschwindigkeit der Drehachse als Funktion der Eigenkreisfrequenz

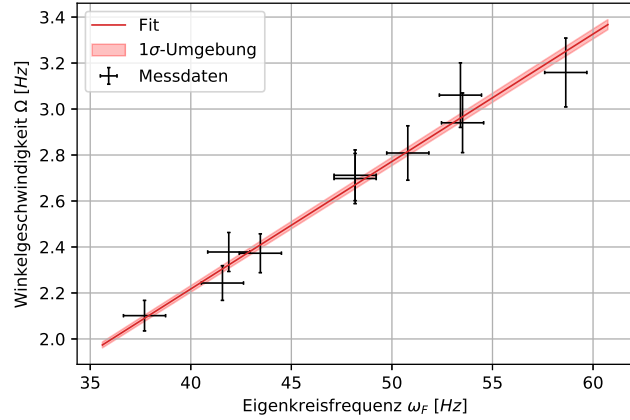


Abbildung 5

## 6.4 Auswertung

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

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

Das wird sich ein zweites Mal bestätigen lassen können, wenn wir  $I_z$  aus dem vorherigen Versuchsteil mit dem Wert für  $I_{x,2}$  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  $P \approx 99.0\%$  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.

$$S_{Farbe} = 5.544(36) \times 10^{-2}$$

$$I_{x,1} = 4.754(23) \times 10^{-3} \text{ kg m}^2$$

## 7 Bestimmung von $I_x$ aus der Nutationsfrequenz

### 7.1 Durchführung<sup>12</sup>

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

$$\omega_N = \frac{I_z}{I_x} \omega_F \quad (23)$$

gilt, sollte die Öffnung des Nutationskegels an der Spitze des Stabes nur  $1 - 2 \text{ cm}$  betragen. Bei unserer Durchführung war dieser Wert eher größer  $\approx 2 - 3 \text{ cm}$ , 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  $f_N$  und  $f_F$ . Wir sind dabei folgendermaßen vorgegangen: eher langsamere Messung der Nutationsfrequenz mit dem Stroboskop zu Beginn und erst bei der Festlegung von  $f_N$  wird  $f_F$  mit Hilfe des optischen Drehzahlmessers.

### 7.2 Messergebnisse

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

Tabelle 5: Messung der Nutationsfrequenz

Nutationsfrequenz $f_N$ [min <sup>-1</sup> ]	Drehfrequenz $f_F$ [min <sup>-1</sup> ]
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

<sup>1</sup>  $\Delta f_F$  grob abgeschätzt

<sup>2</sup>  $\Delta f_N$  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  $\omega_N$  aus:

$$\omega_N = 2\pi f_N \quad (24)$$

$$\Delta\omega_N = 2\pi(\Delta f_N) \quad (25)$$

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

$$\omega_N = S_N \omega_F \quad (26)$$

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

```
1 f_F = np.array([693,601,444,574,488,837,737,722,394,629]) / 6
2 Fehler_f_F = np.full(f_F.size, 10) / 60
3
4 f_N = np.array([655,585,435,525,455,780,695,685,395,600]) / 60
5 Fehler_f_N = np.full(f_N.size, 5) / 60
```

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

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

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

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

Fitfunktion (26) wird deklariert:

```
1 from scipy import odr
2
3 def fit_func(p, x):
4     (s) = p
5     return x*s
6
7 model = odr.Model(fit_func)
```

darzustellende Daten werden übergeben:

```
1 x = omega_F
2 y = omega_N
3 delta_x = Fehler_omega_F
4 delta_y = Fehler_omega_N
```

<sup>12</sup> Dr. J. Wagner - Physikalisches Anfängerpraktikum - V. 1.1 Stand 1/2018, Versuch 213

Startparameter für Ausgleichsrechnung werden gesetzt, sodass Lösung konvergiert:

```
1 para0 = [1]
2
3 data = odr.RealData(x, y, sx=delta_x, sy=delta_y)
4 odr = odr.ODR(data, model, beta0=para0 )
5 out = odr.run()
```

Auslesen der Messergebnisse:

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

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

```
1 nstd = 2
2
3 popt_top = popt+nstd*perr
4 popt_bot = popt-nstd*perr
```

Plot-Umgebung wird angegeben:

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

Diagramm (Abb.6) wird erstellt:

```
1 fig, ax = plt.subplots(1)
2 plt.errorbar(x, y, yerr=delta_y, xerr=delta_x, lw=1, ecolor='k', fmt='none', capsize=1, label='
  Messdaten')
3 plt.title('Nutationsfrequenz als Funktion der Eigenfrequenz')
4 plt.grid(True)
5 plt.xlabel('Eigenkreisfrequenz ' + r'\omega_F$' + ' ' + r'$[Hz]$')
6 plt.ylabel('Nutationskreisfrequenz ' + r'\omega_N$' + ' ' + r'$[Hz]$')
7 plt.plot(x_fit, fit, color='C3', lw=1, label='Fit')
8 ax.fill_between(x_fit, fit_top, fit_bot, color='C3', alpha=.25, label=str(nstd)+r'$\sigma$' + '-
  Umgebung')
9 plt.legend(loc='best')
10
11 plt.savefig('figures/213_Fig5.pdf', format='pdf', bbox_inches='tight')
```

Endgültige Ausgleichungsparameter und ihre Kovarianzmatrix werden ausgelesen:

```
1 S_N = popt[0]
2 Fehler_S_N = perr[0]
3
4 I_x_2 = I_z/S_N
5 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  $\Delta\omega_F$  und  $\Delta\omega_N$ . D.h. es wird jeweils der senkrechte/orthogonale Abstand der Messwerte zur Fitfunktion (Abb.1) berechnet und normiert<sup>13</sup>.

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

Ausgabe der Messergebnisse wird erstellt:

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

<sup>13</sup>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

```
1 Steigung:
2 S_N = 9.474419e-01 +- 6.580497e-03
3 Chi-Quadrat = 13.315446247417999
4 Freiheitsgrade = 9
5 Chi-Quadrat reduziert = 1.4794940274908888
6 Wahrscheinlichkeit ein größeres oder gleiches Chi-Quadrat zu erhalten = 15.0 %
7
8 2.Messmethode:
9 I_x [kg * m^2] = 4.739983e-03 +- 4.033036e-05
```

Wir erfahren also sofort, dass

$$S_N = 9.474(66) \times 10^{-1}$$

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

und als Ergebnis auf unseren Anpassungstest:

$$\chi_{red}^2 = 1.5$$

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

$$P \approx 15.0\%$$

und wir erhalten das Diagramm in Abb.5.

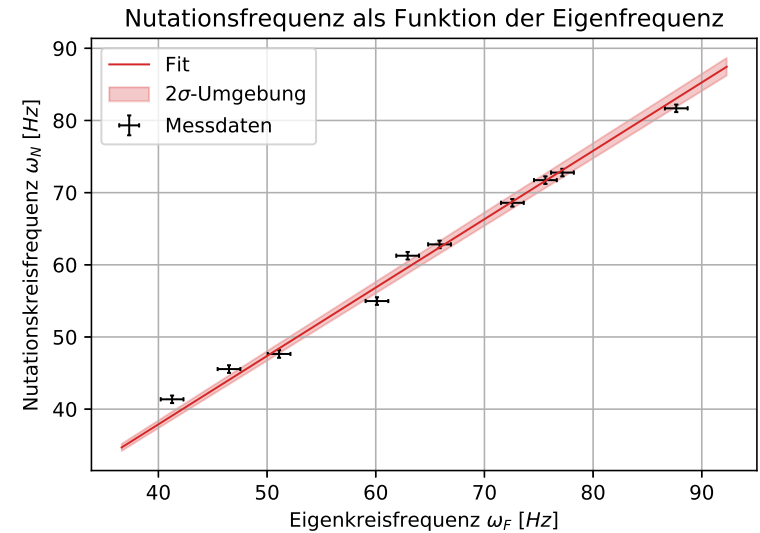


Abbildung 6

## 7.4 Auswertung

$$S_N = 9.474(66) \times 10^{-1}$$

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

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:

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

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

## 8 Fazit

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

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

und für die jeweiligen Trägheitsmomente:

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

Wenn wir uns diese Werte anschauen. Merken wir als erstes, dass  $I_{x,1}$  und  $I_{x,2}$  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  $I_{x,2}$ .
- ( $I_{x,2} > I_z$ ) mit einer statistischen Signifikanz von

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

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