

Experiment 213 - Kreisel

Felix Fleischle

25.10.2021

Einleitung



Figure 1: Bild des Versuchsaufbaus aus der Praktikumsanleitung

Bei diesem Versuch ist es das Ziel, die Dämpfungskonstante und Halbwertszeit des Kreisels zu bestimmen, aus der Präzessionsfrequenz des schweren Kreisels das Trägheitsmoment I_z um die Figurennachse zu bestimmen, aus Größe und Richtung der Umlaufgeschwindigkeit der momentanen Drehachse um die Figurennachse das Trägheitsmoment senkrecht zur Figurennachse zu bestimmen, sowie das selbe Trägheitsmoment aus der Nutationsfrequenz zu bestimmen.

Unser Kreisel besteht aus einer Metallkugel mit Aluminiumstab, welche in

einer Luftpistole gelagert ist. Ohne zusätzliches Gewicht die Stabseite der Kugel etwas leichter, sodass der Kreisel nicht im Schwerpunkt gelagert ist. Wenn wir nun eine Scheibe auf die Stange schieben, ist der Kreisel im Schwerpunkt unterstützt und damit ein kräftefreier Kreisel. Die allgemeine Bewegung eines solchen kräftefreien Kreisels ist eine Nutationsbewegung. Diese kann man erreichen, indem man der Aluminiumstange einen leichten Schlag verpasst. Bei der Nutationsbewegung sind die drei charakteristischen Achsen des Kreisels, die Figurennachse, momentane Drehrichtung und Drehimpuls, nicht mehr parallel, sondern haben unterschiedliche Ausrichtungen. Der Drehimpuls ist dabei räumlich und zeitlich konstant, aber die Figurennachse rotiert auf dem Nutationskegel mit der Nutationsfrequenz ω_N .

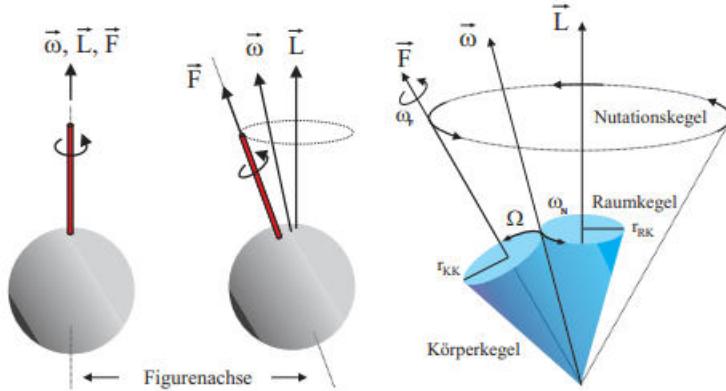


Figure 2: Nutationsbewegung

Währenddessen führt die Figurennachse eine Eigenrotation mit der Frequenz ω_F durch. Die momentane Drehrichtung ω ist daher nicht konstant. Die Figurennachse, der Drehimpuls und die momentane Winkelgeschwindigkeit liegen immer in einer Ebene, weshalb man die momentane Drehrichtung geometrisch zerlegen kann:

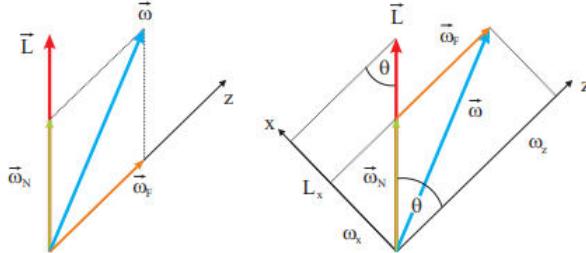


Figure 3: Geometrische Zerlegung von ω

Somit ergibt sich mit $\omega = \omega_F + \omega_N$ und $\omega_x = \omega_N \sin \theta$ ergibt sich

$$\omega_N = \frac{L}{I_x} \quad (1)$$

Bei kleinem θ kann man nähern dass $L = I_z \omega = I_z \omega_F$ und damit

$$\omega_N = \frac{I_z}{I_x} \omega_F \quad (2)$$

Man kann die Bewegung der momentanen Drehachse visualisieren. Dazu steckt man eine Platte mit Farbsektoren auf die Figurennachse. Bei der Nutation erscheinen die Farben vermischt, bis auf in einem Punkt, in dem die momentane Drehachse die Farbscheibe durchstößt.

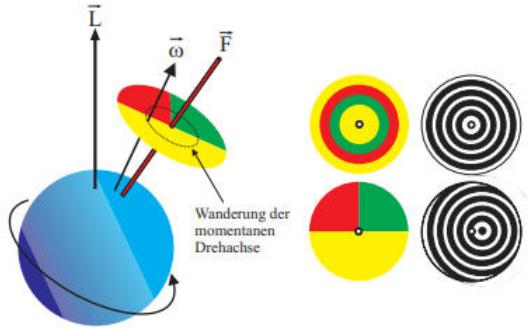


Figure 4: Visualisierung mit der Drehscheibe

Der Farbwechsel erfolgt mit der Winkelgeschwindigkeit Ω . Diese kann aus

folgender Formel bestimmt werden

$$\Omega = \frac{I_x - I_z}{I_x} \omega_F \quad (3)$$

bzw.

$$I_x - I_z = \frac{I_z}{\frac{\omega_F}{\Omega} - 1} \quad (4)$$

Wenn man eine Zusatzmasse auf den Kreisel montiert, ist dieser nicht mehr im Schwerpunkt unterstützt, sodass dieser zum schweren Kreisel wird. Dabei fallen $\vec{\omega}$ und \vec{L} zusammen:

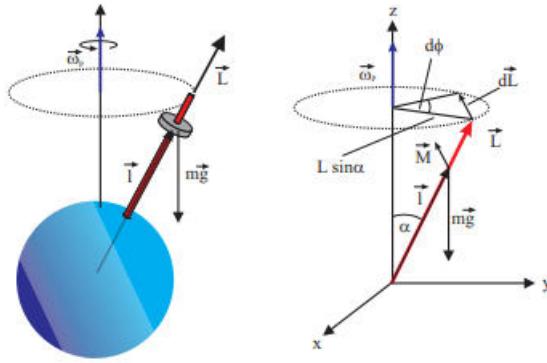


Figure 5: Schwerer Kreisel

Das externe Drehmoment, das auf den Kreisel wirkt ist

$$\vec{M} = \vec{l} \times m\vec{g} \quad (5)$$

Dadurch ändert sich der Drehimpuls zeitlich. Wegen $\vec{M} \perp \vec{L}$ ist $d\vec{L} \perp \vec{L}$. Der Drehimpuls weicht also seitlich der Gewichtskraft aus. Diese Bewegung heißt Präzession mit ω_P der Präzessionsfrequenz. Es gilt

$$\omega_P = \frac{mgl}{I_z \omega_F} \quad (6)$$

oder

$$\vec{M} = \vec{\omega}_p \times \vec{L} \quad (7)$$

Die Präzessionsfrequenz hängt also nicht von der räumlichen Orientierung des Kreisels ab.

Im ersten Versuchsteil lassen wir den Kreisel senkrecht, und beschleunigen den Kreisel auf ca. $600 - 700 \text{ min}^{-1}$. Dann messen wir alle 2 Minuten die Drehfrequenz über einen Zeitraum von 12 Minuten und bestimmen daraus Dämpfungskonstante und Halbwertszeit.

Danach arbeiten wir mit einem schweren Kreisel. Zuerst bringen wir im Abstand von 20cm ein Gewicht an und messen für drei verschiedene Winkel die Präzessionsdauer T_P . Daraus schließen wir den Einfluss der räumlichen Auswirkung auf die Präzessionsfrequenz. Anschließend messen wir für vier unterschiedliche Massenkonfigurationen jeweils für vier Frequenzen ebenfalls die Präzessionsdauer. Daraus bestimmen wir letztendlich das Trägheitsmoment I_z .

In den restlichen Versuchsteilen arbeiten wir mit einem freien Kreisel. Wir geben dem Kreisel einen Stoß und bestimmen die Umlaufrichtung der momentanen Drehachse mit der Farbplatte. Danach messen wir für 10 Frequenzen jeweils die Zeit für 10 Umläufe der momentanen Drehachse bzw. die Frequenz des Farbwechsels Ω . Daraus können wir letztendlich I_x bestimmen.

Im letzten Teil des Versuchs bestimmen wir bei Nutation jeweils 10 Wertepaare von ω_F und ω_N , woraus wir nochmals I_x bestimmen.

Experiment 2.13 Kreisel - Messprotokoll

Felix
Fleischke

Messaufbau

- Stahlkugel mit Aluminiumstab ($m = 4,164 \text{ kg}$) als Kreisel gelagert in Luftkissenwanne
- 2 Gewichte ($r_a = 0,725 \text{ cm}$, $r_i = 0,325 \text{ cm}$, $h = 1,40 \text{ m}$, $m = 9,85 \text{ g}$)
- Farbscheibe, Scheibe mit konzentrischen Ringen
- Stroboskop
- Stoppuhr
- Motor mit Netzgerät
- Gyroskop zur Demonstration von Kreiseln -> Chatton

Skizze



Durchführung

a) Vorversuch

a) Zur Seite drücken des Kugellagers beim kräftefreien Kreisel.

Beobachtung: Der Kreisel weicht dem externen Drehmoment seitlich aus, und "zieht sich um Finger hoch" (-Furkan)

b) Nutationsbewegung

Beobachtung: Es ~~zeigt~~ zeigt sich ein Punkt mit unvermischter Farbe

c) Beobachtung: Durch das Anbringen eines gewichtes beginnt eine Präzession, die vorher nicht vorhanden war.

d) Drehrichtung der Präzessions

Im Uhzeigersinn

2. Dämpfung

Wir messen über zwei Minuten die Prechfrequenz des Kreisels

Tabelle 1: Frequenz in Abhängigkeit der Zeit

Zeit t [min]	Frequenz w [$\frac{1}{min}$]
0	560 ± 10
2	510 ± 10
4	460 ± 10
6	430 ± 10
8	390 ± 10
10	350 ± 10
12	310 ± 10 20

3. Präzession

Wir messen die Präzessionsdauer T_p für drei verschiedene Winkel des Stabs gegen die Vertikale ($f \approx 500 \frac{1}{min}$)

Tabelle 2: Präzessionsdauer als Fkt. des Winkels

Winkel [$^{\circ}$]	Präzessionsdauer $T_p [s]$
20 ± 5	$125,42 \pm 5$
40 ± 5	$124,58 \pm 5$
90 ± 5	$127,98 \pm 5$

Nun messen wir Für 4 verschiedene Massen bei jeweils 4 Frequenzen die Präzessionsdauer:

Tabelle 3: Präzessionsdauer als Fkt. von m und f

Massen	Frequenz [Hz]	Präzessionsdauer T_p [s]
1·m bei 15cm	700 ± 5	$230,20 \pm 3$
"	560 ± 5	$210,96 \pm 3$
"	470 ± 5	$155,75 \pm 3$
"	410 ± 5	$143,45 \pm 3$
1m bei 20cm	700 ± 5	$201,46 \pm 3$
"	600 ± 5	$145,29 \pm 3$
"	530 ± 5	$132,89 \pm 3$
"	460 ± 5	$122,72 \pm 3$
2m bei 15cm	350 700 ± 5 315	$118,92 \pm 3$
"	600 ± 5	$111,24 \pm 3$
"	280 ± 5	$103,82 \pm 3$
"	2585 ± 5	$58,32 \pm 3$
2m bei 20cm	350 ± 5	$148,20 \pm 3$
"	300 ± 5	$131,73 \pm 3$
"	250 ± 5	$117,48 \pm 3$
"	430 ± 5	$124,02 \pm 3$

* Könnte auch $2 \cdot f$ sein, in der Auswertung untersuchen wir ob es passt.

4t. Umlauf der mom. Drehachse um die Figurenachse

Richtung der momentanen Drehachse:

gelb + grün \rightarrow blau \rightarrow also nach rechts! gibt es? ja/Nein/noch?

Tabelle 4: Zeit für 10 Umläufe der mom. Drehachse um die Figurenachse als Fkt. der Frequenz.

Frequenz f [$\frac{1}{min}$]	Zeit t [s]
400 ± 5	$8,48 \pm 0,6$
430 ± 5	$8,90 \pm 0,6$
530 ± 5 *	$11,07 \pm 0,6$
500 ± 5 *	$11,92 \pm 0,6$
470 ± 5 *	$12,59 \pm 0,6$
485 ± 5 *	$13,24 \pm 0,6$
430 ± 5 *	$13,84 \pm 0,6$
420 ± 5 *	$14,18 \pm 0,6$
400 ± 5	$14,87 \pm 0,6$
385 ± 5	$15,36 \pm 0,6$

* Könnte 2-f sein

5. Nutation

Tabelle 5: Wertepaare von ω_N und ω_F

Eigenrotation der Figurachse	ω_F [$\frac{1}{\text{min}}$]	Nutations- frequenz	ω_N [$\frac{1}{\text{min}}$]
610 ± 5		570 ± 5	
575 ± 5		540 ± 5	
540 ± 5		510 ± 5	
510 ± 5		480 ± 5	
480 ± 5		440 ± 5	
450 ± 5		410 ± 5	
410 ± 5		380 ± 5	
380 ± 5		350 ± 5	
340 ± 5		320 ± 5	
305 ± 5		280 ± 5	

25.10.21

J.B.

Experiment 213 Kreisel - Auswertung

Felix Fleischle - 25.10.2021

```
In [1]: import numpy as np  
import matplotlib.pyplot as plt  
from scipy.optimize import curve_fit
```

```
In [2]: # Zuerst definieren wir eine Funktion, die uns den Bruch vom Fehler eines Wertes durch den Wert selbst zurückgibt,  
# welche wir für die Berechnung des relativen Fehlers brauchen werden  
def errorFrac(x, x_err, p):  
    return (x_err * p)/x
```

1) Diskussion der Beobachtungen des Vorversuches

Zuerst haben wir den kräftefreien Kreisel in der senkrechten Lage beschleunigt und am Kugellager ein Drehmoment auf den Kreisel wirken. Wir haben beobachtet, dass der Kreisel sich "am Finger hochzieht", also dem Drehmoment seitlich "ausweicht", was wir auch schon beim einem Drehmoment wie bei der Gewichtskraft erwartet haben (siehe Einleitung).

Als nächstes haben wir dem Kreisel einen Stoß verpasst, sodass sich eine Nutationsbewegung eingestellt hat. Auf der Farbscheibe hat sich dabei tatsächlich ein Punkt gezeigt mit unvermischter Farbe. Dieser ist der erwartete Punkt, bei dem die momentane Drehachse die Farbscheibe durchstößt. Insgesamt war der Vorversuch gut um die Rotationsbewegungen des Kreisels zu verstehen, und alle unserer Beobachtungen entsprechen den Erwartungen.

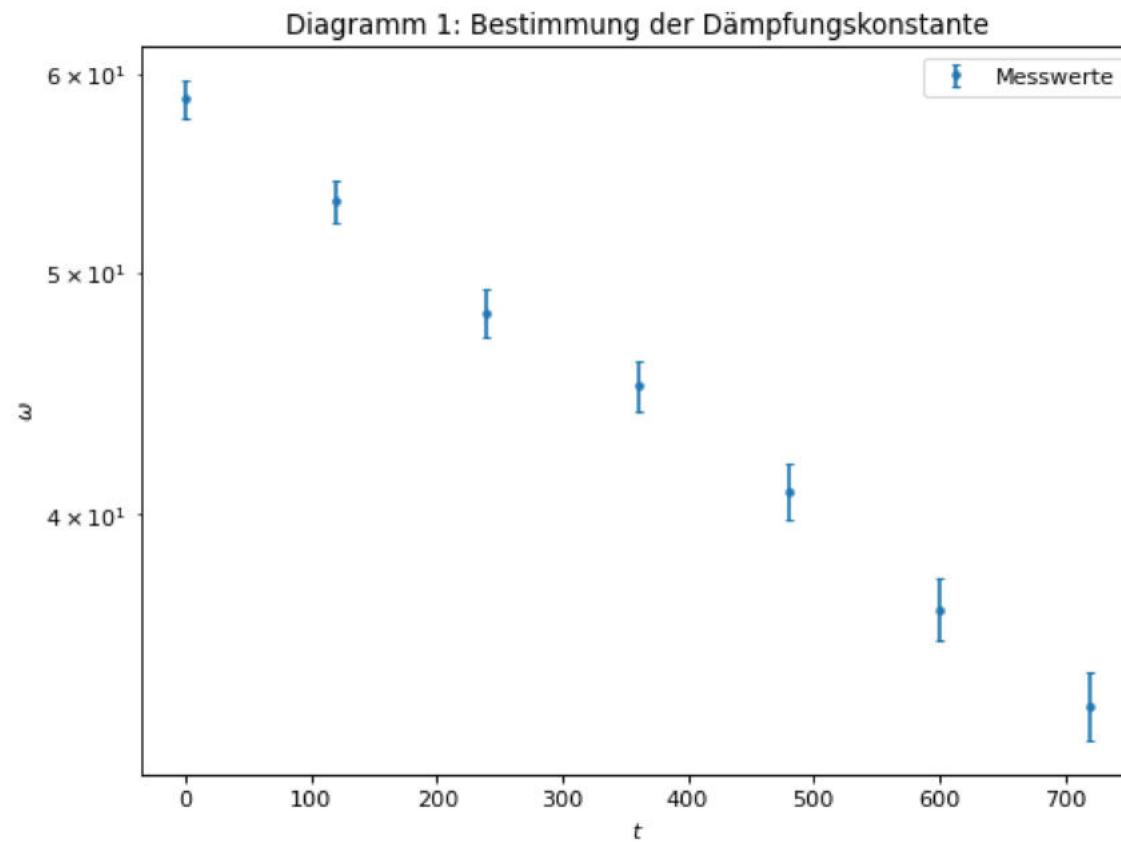
2) Dämpfung des Kreisels

```
In [3]: # Wir schreiben unsere gemessenen Daten in Arrays:  
  
t_daempfung = np.array([0 , 2 , 4 , 6 , 8 , 10 , 12])*60 # Sekunden  
omega_daempfung = 2 * np.pi * np.array([560 , 510 , 460 , 430 , 390 , 350 , 320]) / 60 # Hertz  
omega_daempfung_err = 2 * np.pi * np.ones(7) * 10 / 60  
  
print("Zeitwerte [s]: " , t_daempfung)  
print("Kreisfrequenzen [1/s]: " , omega_daempfung)  
print("Fehler Kreisfrequenzen [1/s]: " , omega_daempfung_err)
```

```
Zeitwerte [s]: [ 0 120 240 360 480 600 720]  
Kreisfrequenzen [1/s]: [58.64306287 53.40707511 48.17108736 45.0294947 40.8407045 36.65191429
```

```
33.51032164]  
Fehler Kreisfrequenzen [1/s]: [1.04719755 1.04719755 1.04719755 1.04719755 1.04719755 1.04719755  
1.04719755]
```

```
In [4]: # Wir plotten unsere Messwerte:  
plt.figure(figsize=(8, 6), dpi=80)  
plt.errorbar(t_daempfung, omega_daempfung, linestyle="None", marker = ".", yerr = omega_daempfung_err, capsize = 2, label="Messwer  
plt.title("Diagramm 1: Bestimmung der Dämpfungskonstante")  
plt.legend()  
plt.xlabel("$t$")  
plt.ylabel("$\omega$")  
plt.yscale("log")
```



```
In [5]: # Nun definieren wir eine Dämpfungsfunktion um den fit durchzuführen  
def daempfung(t_daempfung, delta, A):  
    return A * np.exp((- delta) * t_daempfung)  
values_0 = [0, 58.64306287]
```

```
# Wir führen den fit durch
popt, pcov = curve_fit(daempfung, t_daempfung, omega_daempfung,sigma = omega_daempfung_err, p0 = values_0)

delta = popt[0]
A = popt[1]
delta_err = pcov[0,0]
A_err = pcov[1,1]

print("Delta: ",delta, "1/s")
print("Fehler von Delta: ",delta_err, "1/s")
print("A: ", A, "1/s")
print("Fehler von A: ", A_err, "1/s")
```

```
Delta: 0.0007698051599880399 1/s
Fehler von Delta: 2.2737270927282008e-10 1/s
A: 58.60299643142167 1/s
Fehler von A: 0.09989577894309454 1/s
```

In [6]:

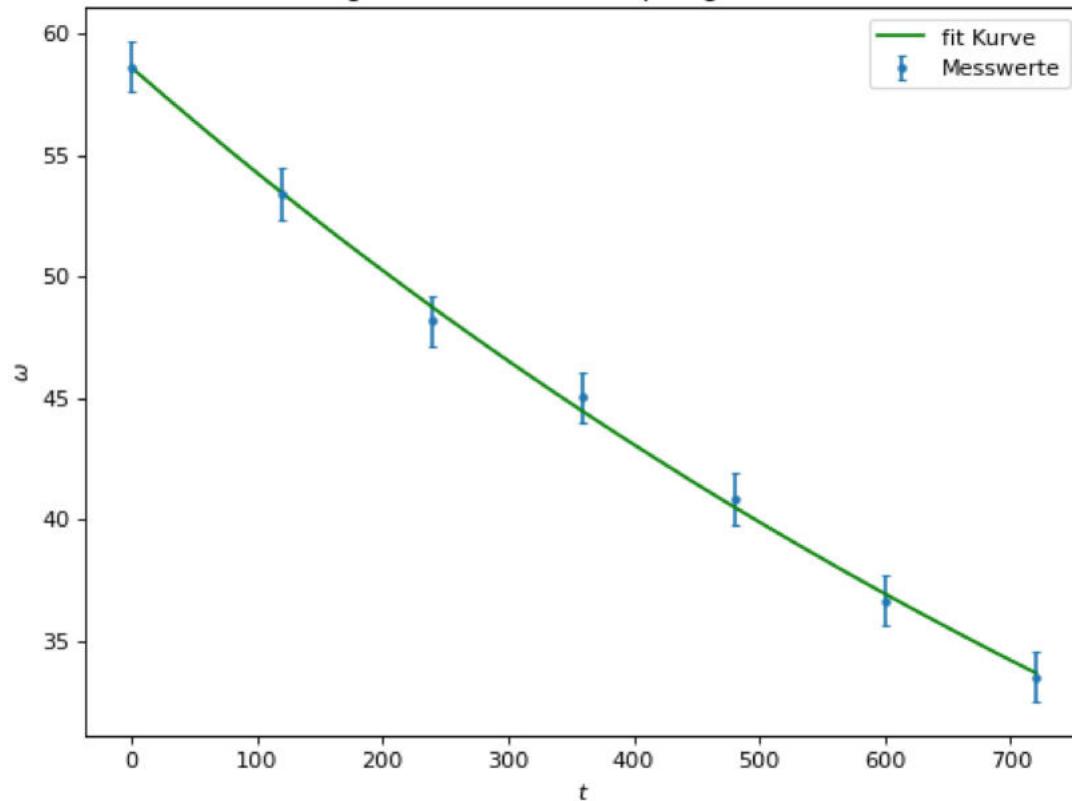
```
# Wir fügen den fit in den Plot ein
t_daempfung_res = np.linspace(0,720, 5000)

plt.figure(figsize=(8, 6), dpi=80)
plt.errorbar(t_daempfung, omega_daempfung, linestyle="None", marker = ".", yerr = omega_daempfung_err, capsize = 2, label="Messwerte")
plt.plot(t_daempfung_res, daempfung(t_daempfung_res, delta, A), color="green", label="fit Kurve")
plt.title("Diagramm 2: Fit der Dämpfungskonstante")
plt.legend()
plt.xlabel("$t$")
plt.ylabel("$\omega$")
```

Out[6]:

```
Text(0, 0.5, '$\omega$')
```

Diagramm 2: Fit der Dämpfungskonstante



```
In [7]: # Nun können wir die Halbwertszeit bestimmen  
t_halb = np.log(2)/delta  
t_halb_err = np.log(2) * delta_err / (delta**2)  
  
print("Halbwertszeit: ", t_halb, "s")  
print("Fehler der Halbwertszeit: ", t_halb_err, "s")
```

Halbwertszeit: 900.4189846827143 s
Fehler der Halbwertszeit: 0.00026595132725684955 s

Ergebnisse:

$$\delta = 0.00076980516 \pm 0.00000000023 \frac{1}{s}$$

$$t_{1/2} = 900.41898 \pm 0.00027 s$$

3a) Präzessionsdauer in Abhängigkeit des Winkels

Unsere Messergebnisse waren:

```
In [8]: winkel = np.array([20, 40, 90])*(np.pi/180) # Winkel in Radian
winkel_err = np.ones(3) * 5 *(np.pi/180) # Fehler des Winkels

t_p_winkel = np.array([125.42, 124.58, 127.98]) # Präzessionsdauer in Sekunden
t_p_winkel_err = np.ones(3)*5

print("Winkel: ", winkel, "rad")
print("Fehler des Winkels: ", winkel_err, "rad")
print("Präzessionsdauer: ", t_p_winkel, "s")
print("Fehler der Präzessionsdauer: ", t_p_winkel_err, "s")
```

Winkel: [0.34906585 0.6981317 1.57079633] rad
Fehler des Winkels: [0.08726646 0.08726646 0.08726646] rad
Präzessionsdauer: [125.42 124.58 127.98] s
Fehler der Präzessionsdauer: [5. 5. 5.] s

Wie wir sehen sind die Präzessionsdauern innerhalb der Fehlergrenzen gleich. Die Präzessionsdauer hängt also nicht von der räumlichen Ausrichtung des Kreisels ab, was wir auch schon in der Einleitung gesehen haben.

3b) Bestimmung des Trägheitsmoments I_z aus der Präzessionsdauer

```
In [9]: # Wir tragen unsere Messdaten für jedes Drehmoment in Arrays ein:

# 1 Masse bei 15cm Abstand
t_p_1m15cm = np.array([250.20, 219.96, 158.75, 143.45])
omega_1m15cm = 2*np.pi*(1/60)*np.array([700, 560, 470, 410]) / 2

# 1 Masse bei 20cm Abstand
t_p_1m20cm = np.array([201.46, 145.29, 132.89, 122.72])
omega_1m20cm = 2*np.pi*(1/60)*np.array([700, 600, 530, 460]) / 2

# 2 Massen bei 15cm Abstand
t_p_2m15cm = np.array([118.92, 111.24, 103.82, 58.32])
omega_2m15cm = 2*np.pi*(1/60)*np.array([350, 315, 280, 255])

# 2 Massen in 20cm Abstand
t_p_2m20cm = np.array([124.02, 148.20, 131.73, 117.48])
omega_2m20cm = 2*np.pi*(1/60)*np.array([430, 350, 300, 250])

# Fehler von t_p
t_p_err = 3
```

```

t_p_err_array = np.ones(4)*3
#Fehler von omega
omega_err = 2*np.pi*(1/60)*5

print("Kreisfrequenzen; 1 Masse 15cm Abstand: ", omega_1m15cm, "[1/s]")
print("Kreisfrequenzen; 1 Masse 20cm Abstand: ", omega_1m20cm, "[1/s]")
print("Kreisfrequenzen; 2 Massen 15cm Abstand: ", omega_2m15cm, "[1/s]")
print("Kreisfrequenzen; 2 Massen 20cm Abstand: ", omega_2m20cm, "[1/s]")

```

```

Kreisfrequenzen; 1 Masse 15cm Abstand: [36.65191429 29.32153143 24.60914245 21.4675498] [1/s]
Kreisfrequenzen; 1 Masse 20cm Abstand: [36.65191429 31.41592654 27.75073511 24.08554368] [1/s]
Kreisfrequenzen; 2 Massen 15cm Abstand: [36.65191429 32.98672286 29.32153143 26.70353756] [1/s]
Kreisfrequenzen; 2 Massen 20cm Abstand: [45.0294947 36.65191429 31.41592654 26.17993878] [1/s]

```

Wir bestimmen zuerst den Endwert der Frequenz nach jeder Messung mithilfe der Dämpfungskonstante

```

In [10]: def omegamean(omega, delta, t_p):
    return (omega/2)*(1+np.exp(-delta * t_p))

def omegameanerr(omega_err, omega, delta, t_p, delta_err, t_p_err):
    return np.sqrt( (omega_err / 2)**2 + ( (omega / 2) * np.exp(-delta*t_p)*t_p*delta_err )**2 + ( (omega / 2)*np.exp(-delta*t_p)*

omega_1m15cm_mean = omegamean(omega_1m15cm , delta , t_p_1m15cm)
omega_1m20cm_mean = omegamean(omega_1m20cm , delta , t_p_1m20cm)
omega_2m15cm_mean = omegamean(omega_2m15cm , delta , t_p_2m15cm)
omega_2m20cm_mean = omegamean(omega_2m20cm , delta , t_p_2m20cm)

omega_1m15cm_mean_err = omegameanerr(omega_err , omega_1m15cm , delta , t_p_1m15cm , delta_err , t_p_err)
omega_1m20cm_mean_err = omegameanerr(omega_err , omega_1m20cm , delta , t_p_1m20cm , delta_err , t_p_err)
omega_2m15cm_mean_err = omegameanerr(omega_err , omega_2m15cm , delta , t_p_2m15cm , delta_err , t_p_err)
omega_2m20cm_mean_err = omegameanerr(omega_err , omega_2m20cm , delta , t_p_2m20cm , delta_err , t_p_err)

print("Durchschnittskreisfrequenzen 1 Masse 15cm Abstand: ", omega_1m15cm_mean, "[1/s]")
print("Durchschnittskreisfrequenzen 1 Masse 20cm Abstand: ", omega_1m20cm_mean, "[1/s]")
print("Durchschnittskreisfrequenzen 2 Massen 15cm Abstand: ", omega_2m15cm_mean, "[1/s]")
print("Durchschnittskreisfrequenzen 2 Massen 20cm Abstand: ", omega_2m20cm_mean, "[1/s]")

```

```

Durchschnittskreisfrequenzen 1 Masse 15cm Abstand: [33.44134408 27.03787305 23.19369278 20.34533694] [1/s]
Durchschnittskreisfrequenzen 1 Masse 20cm Abstand: [34.01925182 29.75375467 26.40148352 22.99994301] [1/s]
Durchschnittskreisfrequenzen 2 Massen 15cm Abstand: [35.04876034 31.6331281 28.19542507 26.11736595] [1/s]
Durchschnittskreisfrequenzen 2 Massen 20cm Abstand: [42.97940326 34.67604635 29.90114039 25.04807726] [1/s]

```

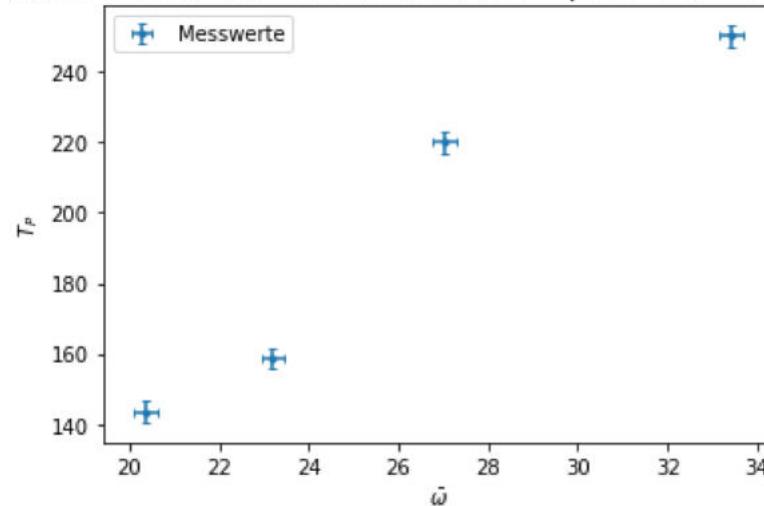
Jetzt können wir T_P gegen $\bar{\omega}$ auftragen und die Steigung bestimmen.

Wir tun dies zuerst für 1 Masse bei 15cm Abstand:

```
In [11]: plt.errorbar(omega_1m15cm_mean, t_p_1m15cm ,linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_1m15cm_mean_err, capsize=5)
plt.legend()
plt.title("Diagramm 3: Präzessionsdauer als Funktion der Frequenz - 1 Masse 15cm Abstand")
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")
```

Out[11]: Text(0, 0.5, '\$T_P\$')

Diagramm 3: Präzessionsdauer als Funktion der Frequenz - 1 Masse 15cm Abstand



```
In [12]: def linear(omega, s):
    return s*omega
def curvefitOmega(omega, t_p):
    return curve_fit(linear, omega, t_p, sigma = t_p_err_array)

popt115, pcov115 = curvefitOmega(omega_1m15cm_mean, t_p_1m15cm)

s_115 = popt115
s_115_err = np.sqrt(pcov115[0])

print("Steigung s, bei 1 Masse 15cm: ", s_115, "[s^2]")
print("Fehler von s: ", s_115_err, "[s^2]")

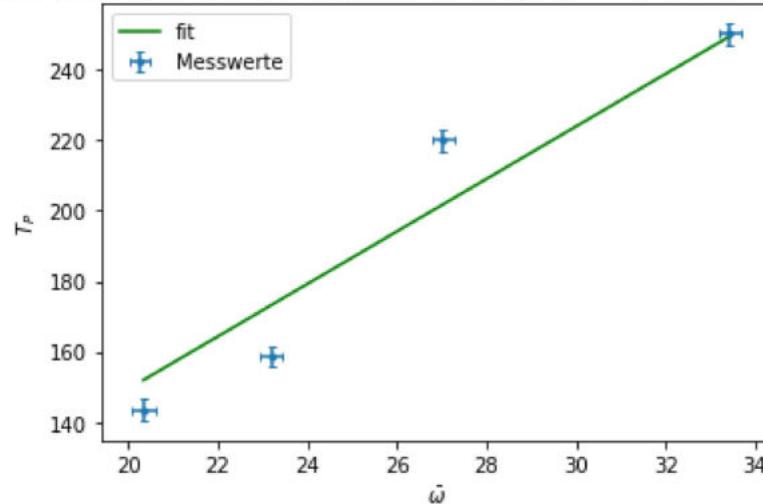
plt.errorbar(omega_1m15cm_mean, t_p_1m15cm ,linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_1m15cm_mean_err, capsize=5)
plt.plot(omega_1m15cm_mean, linear(omega_1m15cm_mean, s_115), color="green", label = "fit")
plt.legend()
plt.title("Diagramm 4: Präzessionsdauer als Funktion der Frequenz - 1 Masse 15cm Abstand - mit fit")
```

```
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")
```

Steigung s, bei 1 Masse 15cm: [7.46624211] [s^2]
Fehler von s: [0.26872371] [s^2]

Out[12]: Text(0, 0.5, '\$T_P\$')

Diagramm 4: Präzessionsdauer als Funktion der Frequenz - 1 Masse 15cm Abstand - mit fit

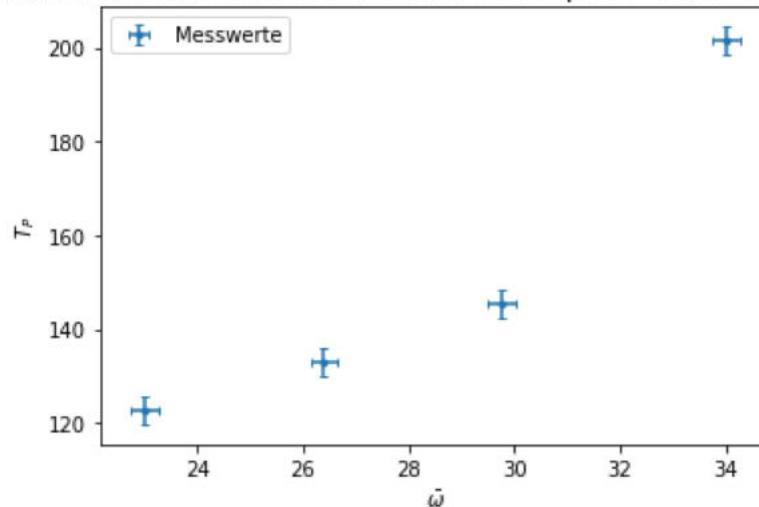


Für 1 Masse bei 20cm:

```
In [13]: plt.errorbar(omega_1m20cm_mean,t_p_1m20cm ,linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_1m20cm_mean_err, capsize=5)
plt.legend()
plt.title("Diagramm 5: Präzessionsdauer als Funktion der Frequenz - 1 Masse 20cm Abstand")
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")
```

Out[13]: Text(0, 0.5, '\$T_P\$')

Diagramm 5: Präzessionsdauer als Funktion der Frequenz - 1 Masse 20cm Abstand



In [14]: `popt120, pcov120 = curvefitOmega(omega_1m20cm_mean, t_p_1m20cm)`

```
s_120 = popt120
s_120_err = np.sqrt(pcov120[0])

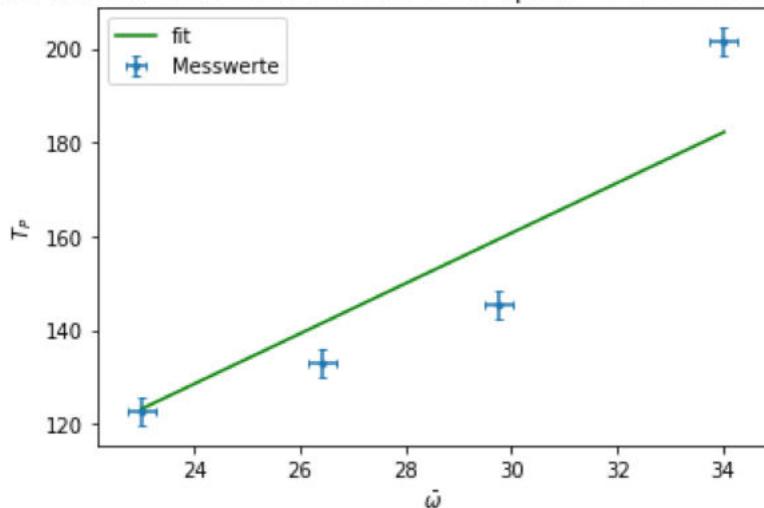
print("Steigung s, bei 1 Masse 20cm: ", s_120, "[s^2]")
print("Fehler von s: ", s_120_err, "[s^2"])

plt.errorbar(omega_1m20cm_mean, t_p_1m20cm, linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_1m20cm_mean_err, capsize=5)
plt.plot(omega_1m20cm_mean, linear(omega_1m20cm_mean, s_120), color="green", label = "fit")
plt.legend()
plt.title("Diagramm 6: Präzessionsdauer als Funktion der Frequenz - 1 Masse 20cm Abstand - mit fit")
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")
```

Steigung s, bei 1 Masse 20cm: [5.35621396] [s²]
Fehler von s: [0.25575639] [s²]

Out[14]: `Text(0, 0.5, 'T_P')`

Diagramm 6: Präzessionsdauer als Funktion der Frequenz - 1 Masse 20cm Abstand - mit fit



Nun für 2 Massen bei 15cm:

```
In [15]: popt215, pcov215 = curvefitOmega(omega_2m15cm_mean, t_p_2m15cm)

s_215 = popt215
s_215_err = np.sqrt(pcov215[0])

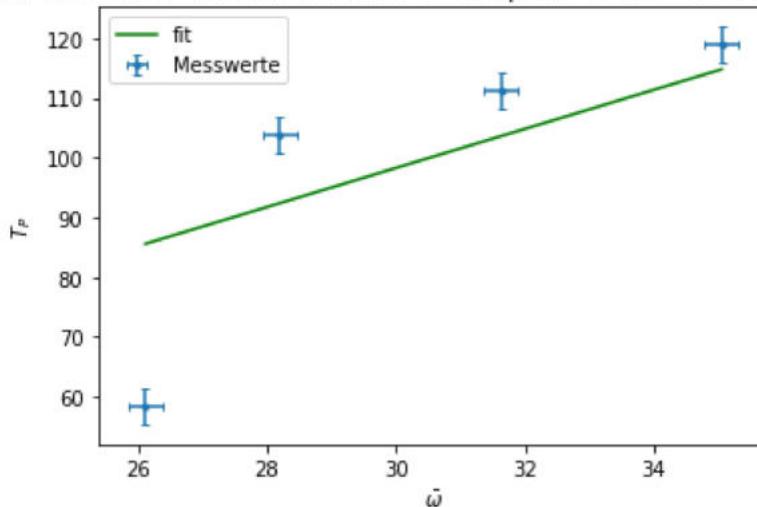
print("Steigung s, bei 2 Massen 15cm: ", s_215, "[s^2]")
print("Fehler von s: ", s_215_err, "[s^2"])

plt.errorbar(omega_2m15cm_mean,t_p_2m15cm ,linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_2m15cm_mean_err, capsize=5)
plt.plot(omega_2m15cm_mean, linear(omega_2m15cm_mean, s_215), color="green", label = "fit")
plt.legend()
plt.title("Diagramm 7: Präzessionsdauer als Funktion der Frequenz - 2 Massen 15cm Abstand - mit fit")
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")
```

Steigung s, bei 2 Massen 15cm: [3.27488598] [s²]
Fehler von s: [0.29198414] [s²]

Out[15]: Text(0, 0.5, '\$T_P\$')

Diagramm 7: Präzessionsdauer als Funktion der Frequenz - 2 Massen 15cm Abstand - mit fit



Und für 2 Massen bei 20cm Abstand:

```
In [16]: index = [0]
popt220, pcov220 = curve_fit(linear, omega_2m20cm_mean, t_p_2m20cm, sigma = t_p_err_array)

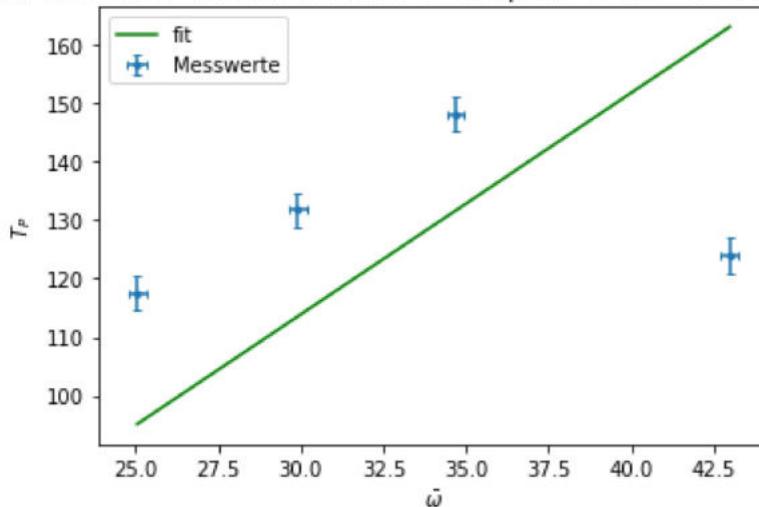
s_220 = popt220
s_220_err = np.sqrt(pcov220[0])

print("Steigung s, bei 2 Massen 20cm: ", s_220, "[s^2]")
print("Fehler von s: ", s_220_err, "[s^2]")

plt.errorbar(omega_2m20cm_mean,t_p_2m20cm ,linestyle="None", marker = ".", yerr = t_p_err_array, xerr = omega_2m20cm_mean_err, capsize=5)
plt.plot(omega_2m20cm_mean, linear(omega_2m20cm_mean, s_220), color="green", label = "fit")
plt.legend()
plt.title("Diagramm 7: Präzessionsdauer als Funktion der Frequenz - 2 Massen 15cm Abstand - mit fit")
plt.xlabel("$\bar{\omega}$")
plt.ylabel("$T_P$")

Steigung s, bei 2 Massen 20cm: [3.79573031] [s^2]
Fehler von s: [0.43872097] [s^2]
Out[16]: Text(0, 0.5, '$T_P$')
```

Diagramm 7: Präzessionsdauer als Funktion der Frequenz - 2 Massen 15cm Abstand - mit fit



Nun können wir das Trägheitsmoment I_z nach Gleichung (6) berechnen (mit $T = \frac{2\pi}{\omega}$ und $s = \frac{2\pi I}{mgl}$).

```
In [17]: m_masses = np.array([0.00985, 0.00985, 2*0.00985 , 2*0.00985]) #kg
l = np.array([0.15 , 0.20 , 0.15 , 0.20])
g = 9.81 # m/s^2

s_array = np.concatenate((s_115, s_120, s_215 , s_220))
s_array_err = np.concatenate((s_115_err, s_120_err, s_215_err , s_220_err))

I_z = (s_array * m_masses * g * l)/(2 * np.pi)
I_z_err = (s_array_err / s_array)
print("Trägheitsmomente: ",I_z, "kg m^2")
print("Fehler Trägheitsmomente: ", I_z_err, "kg m^2")
```

Trägheitsmomente: [0.01722339 0.01647454 0.01510925 0.02334967] kg m²
 Fehler Trägheitsmomente: [0.03599183 0.04774947 0.08915857 0.11558276] kg m²

Wir sehen, dass bei der letzten Messung irgendetwas schiefgelaufen ist. Daher berücksichtigen wir nur die ersten drei Messreihen:

```
In [18]: index = [3]
I_z_new = np.delete(I_z, index)
I_z_err_new = np.delete(I_z_err, index)

I_z_mean = np.mean(I_z_new)
```

```
# Wir fügen zum Fehler von I_z außer der Standardabweichung noch einen systematischen Fehler von 0.005 kgm^2 hinzu,
# um den Fehler nach oben abzuschätzen, da unsere Messwerte sehr stark verteilt waren

I_z_mean_err = np.sqrt( (np.std(I_z_new) / np.sqrt(3))**2 + (0.005)**2 )

print("Trägheitsmoment I_z: ", I_z_mean, "kg m^2")
print("Fehler I_z: ", I_z_mean_err, "kg m^2")
```

Trägheitsmoment I_z: 0.016269059483850917 kg m²
Fehler I_z: 0.0050254700101280865 kg m²

Ergebnisse:

$$I_z = 0.016 \pm 0.005 \text{ kgm}^2$$

4a) Bestimmung des größeren Trägheitsmoment von I_z und I_x

Wir haben gesehen, dass aufgrund der Farbreihenfolge sich der Kreisel um die momentane Drehachse gegen den Uhrzeigersinn dreht. Ω und ω_F haben somit das gleiche Vorzeichen und mit Gleichung (4) folgt $I_x - I_z > 0 \Rightarrow I_x > I_z$

4b) Bestimmung von I_x

Wir bestimmen die Umlauffrequenz der momentanen Drehachse aus unseren Zeiten für 10 Umläufe, und tragen dann die Winkelgeschwindigkeit gegen die Umlauffrequenz auf.

In [19]:

```
# Unsere Messwerte:
omega_f = 2 * np.pi * np.array([450, 430, 265, 250, 235, 227.5, 215, 210, 200, 192.5]) / 60 # 1/s
omega_f_err = 2 * np.pi * np.ones(10)*5 / 60
time_mom = np.array([8.48, 8.9, 11.07, 11.92, 12.59, 13.24, 13.80, 14.15, 14.87, 15.36]) # s
time_mom_err = np.ones(10)*0.6

periodendauer_mom = time_mom / 10
periodendauer_mom_err = time_mom_err / 10
omega_mom = (2 * np.pi)/(periodendauer_mom)
omega_mom_err = (2*np.pi*periodendauer_mom_err)/(periodendauer_mom**2)

print("Eigenrotationsfrequenzen: ", omega_f, "[1/s]")
print("Fehler Eigenenrotationsfrequenzen: ", omega_f_err, "[1/s]")
print("Frequenz der momentanen Drehachse: ", omega_mom, "[1/s]")
print("Fehler Frequenz der momentanen Drehachse: ", omega_mom_err, "[1/s]")
```

Eigenrotationsfrequenzen: [47.1238898 45.0294947 27.75073511 26.17993878 24.60914245 23.82374429
22.51474735 21.99114858 20.94395102 20.15855286] [1/s]

```

Fehler Eigenenrotationsfrequenzen: [0.52359878 0.52359878 0.52359878 0.52359878 0.52359878 0.52359878
0.52359878 0.52359878 0.52359878] [1/s]
Frequenz der momentanen Drehachse: [7.40941664 7.05975877 5.67586749 5.27112861 4.99061581 4.74560824
4.55303283 4.44041364 4.22541043 4.09061543] [1/s]
Fehler Frequenz der momentanen Drehachse: [0.52425118 0.47593879 0.30763509 0.26532527 0.23783713 0.21505778
0.19795795 0.18828609 0.17049403 0.15978967] [1/s]

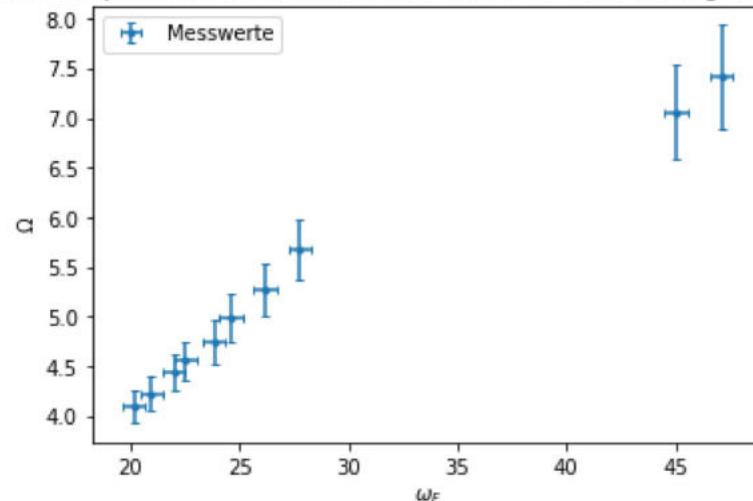
```

Wir nutzen Gleichung (3) und tragen Ω gegen ω_F auf und bestimmen aus der Steigung I_x .

```
In [20]: plt.errorbar(omega_f, omega_mom ,linestyle="None", marker = ".", yerr = omega_mom_err, xerr = omega_f_err, capsize = 2, label="Me
plt.legend()
plt.title("Diagramm 8: Frequenz der momentanen Drehachse als Fkt. der Eigenrotationsfrequenz")
plt.xlabel("$\omega_F$")
plt.ylabel("$\Omega$")
```

```
Out[20]: Text(0, 0.5, '$\Omega$')
```

Diagramm 8: Frequenz der momentanen Drehachse als Fkt. der Eigenrotationsfrequenz



```
In [21]: popt, pcov = curve_fit(linear, omega_f, omega_mom, sigma = omega_mom_err)

slope = popt
slope_err = np.sqrt(pcov[0])

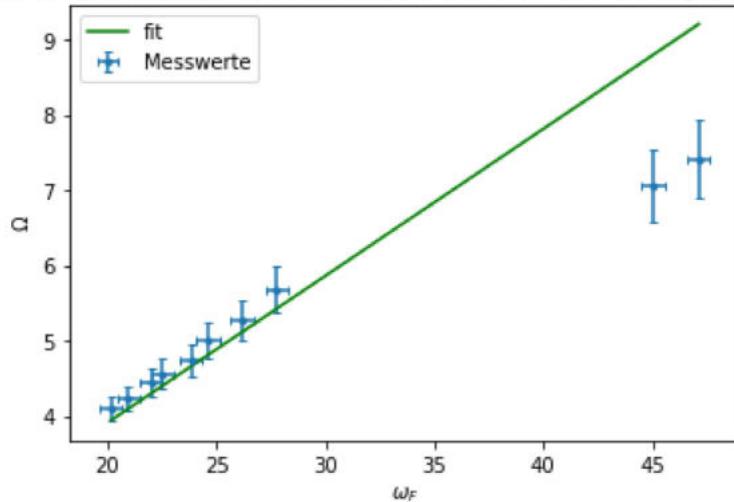
print("Steigung: ", slope)
print("Fehler Steigung: ", slope_err)
```

```
Steigung: [0.1953701]
Fehler Steigung: [0.00533666]
```

```
In [22]: plt.errorbar(omega_f, omega_mom ,linestyle="None", marker = ".", yerr = omega_mom_err, xerr = omega_f_err, capsize = 2, label="Me")
plt.plot(omega_f, linear(omega_f, slope), color="green", label="fit")
plt.legend()
plt.title("Diagramm 8: Frequenz der momentanten Drehachse als Fkt. der Eigenrotationsfrequenz")
plt.xlabel("$\omega_F$")
plt.ylabel("$\Omega$")
```

Out[22]: Text(0, 0.5, '\$\Omega\$')

Diagramm 8: Frequenz der momentanten Drehachse als Fkt. der Eigenrotationsfrequenz



In [23]: # Wir bestimmen nun das Trägheitsmoment in x-Richtung aus der Steigung:

```
I_x = I_z_mean / (1-slope)
I_x_err = np.sqrt( (I_z_mean_err/(1-slope))**2 + (I_z_mean * slope_err / ((1-slope)**2))**2 )

print("Trägheitsmoment I_x: ", I_x, "kgm^2")
print("Fehler I_x: ", I_x_err, "kgm^2")
```

Trägheitsmoment I_x: [0.02021931] kgm²

Fehler I_x: [0.00624713] kgm²

Ergebnis:

$$I_x = 0.020 \pm 0.006 \text{ kgm}^2$$

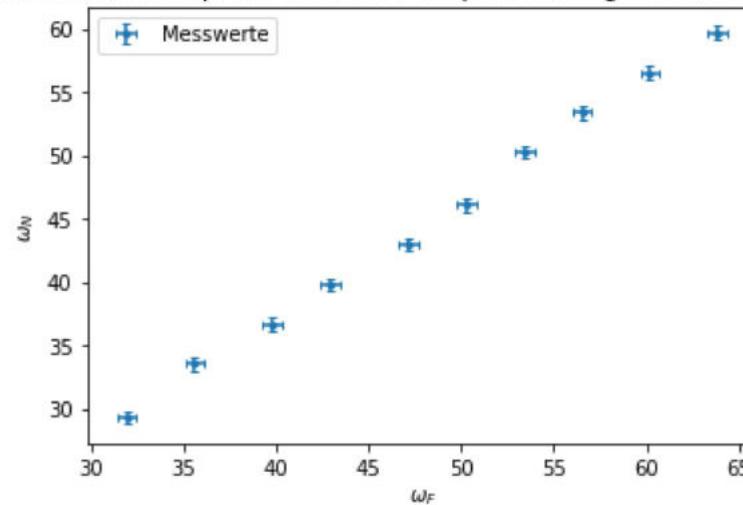
5) Bestimmung von I_x über die Nutationsfrequenz

```
In [24]: # Wir tragen unsere Werte für die Eigenrotationfrequenz und die Nutationsfrequenz in arrays ein
omega_f_nu = 2 * np.pi * np.array([610 , 575 , 540 , 510 , 480 , 450 , 410 , 380 , 340 , 305]) / 60
omega_n_nu = 2 * np.pi * np.array([570 , 540 , 510 , 480 , 440 , 410 , 380 , 350 , 320 , 280]) / 60
omega_f_nu_err = 2 * np.pi * np.ones(10)*5/60
omega_n_nu_err = 2 * np.pi * np.ones(10)*5/60

# Wir plotten unsere Messwerte
plt.errorbar(omega_f_nu, omega_n_nu ,linestyle="None", marker = ".", yerr = omega_n_nu_err, xerr = omega_f_nu_err, capsized = 2, 1
plt.legend()
plt.title("Diagramm 9 Nutationsfrequenz als Fkt. der Frequenz der Eigenrotation der Figurennachse")
plt.xlabel("$\omega_F$")
plt.ylabel("$\omega_N$")
```

Out[24]: Text(0, 0.5, '\$\omega_N\$')

Diagramm 9 Nutationsfrequenz als Fkt. der Frequenz der Eigenrotation der Figurennachse



```
In [25]: # Nun führen wir erneut einen fit durch um die Steigung zu bestimmen. Nach Gleichung (2) entspricht die Steigung genau I_z / I_x
popt_nu , pcov_nu = curve_fit(linear, omega_f_nu, omega_n_nu, sigma = omega_n_nu_err)

slope_nu = popt_nu
slope_nu_err = np.sqrt(pcov_nu[0])

print("Steigung: ",slope_nu)
print("Fehler Steigung: ",slope_nu_err)
```

Steigung: [0.93146777]

Fehler Steigung: [0.0037375]

In [26]:

```
# Wir bestimmen nun I_x aus Gleichung (2)
I_x_nu = I_z_mean / slope_nu
I_x_nu_err = np.sqrt((I_z_mean_err / slope_nu)**2 + (I_z_mean * slope_nu_err / (slope_nu)**2)**2)

print("Trägheitsmoment I_x: " , I_x_nu, "[kgm^2]")
print("Fehler Trägheitsmoment I_x: " , I_x_nu_err, "[kgm^2]")
```

Trägheitsmoment I_x: [0.01746605] [kgm^2]
Fehler Trägheitsmoment I_x: [0.00539567] [kgm^2]

Unser Ergebnis lautet also:

$$I_x = 0.0175 \pm 0.005 \text{ kgm}^2$$

In [27]:

```
# Wir berechnen die Abweichung zwischen unseren Werten für I_x:
sigma_i_x = (I_x - I_x_nu)/(np.sqrt(I_x_err**2 + I_x_nu_err**2))
print("Abweichung I_x: ", sigma_i_x)
```

Abweichung I_x: [0.33353898]

In []:

In []:

In []:

Zusammenfassung und Diskussion

Unser Ziel war es, die Dämpfungskonstante und Halbwertszeit des Kreisels zu bestimmen, aus der Präzessionsfrequenz des schweren Kreisels das Trägheitsmoment I_z um die Figurennachse zu bestimmen, aus Größe und Richtung der Umlaufgeschwindigkeit der momentanen Drehachse um die Figurennachse das Trägheitsmoment senkrecht zur Figurennachse zu bestimmen, sowie das selbe Trägheitsmoment aus der Nutationsfrequenz zu bestimmen.

Als erstes haben wir aus der Rotationsfrequenzabnahme die Dämpfungskonstante bestimmt. Unser Ergebnis war

$$\delta = 0.00076980516 \pm 0.00000000023 \frac{1}{s}$$

Daraus haben wir dann die Halbwertszeit bestimmt:

$$t_{1/2} = 900.41898 \pm 0.00027 s$$

Danach haben wir für den schweren Kreisel die Abhängigkeit der Präzessionsdauer vom Winkel untersucht, und wie erwartet gefunden, dass die Präzessionsdauer nicht vom Winkel abhängt.

Für unterschiedliche Drehmomente verändert sich allerdings die Präzessionsdauer. Aus den verschiedenen gemessenen Präzessiondauern haben wir das Trägheitsmoment I_z bestimmt. Unser Ergebnis für das Trägheitsmoment war

$$I_z = 0.016 \pm 0.005 kgm^2$$

Dabei haben wir die letzte Messreihe ignoriert, da diese sehr große Abweichungen zeigte. Die Erklärung ist sehr wahrscheinlich, dass wir hier nicht von der höchsten zu niedrigeren Frequenzen gemessen haben, sondern am Ende die Frequenz wieder erhöht haben, da wir sonst unter die angegebenen 250 Umdrehungen pro Minute gekommen wären. Die letzte Messung stimmte dann aber nicht mit den anderen überein, was diese sehr große Abweichung erzeugt hat.

Wir haben außerdem noch einen zusätzlichen Fehler für I_z eingefügt, da die Standardabweichung alleine ein sehr kleiner Fehler war, unsere Messwerte jedoch bei jeder Messreihe mindestens einen Punkt hatten, der überhaupt nicht auf der Gerade lag.

Im nächsten Teil des Versuchs haben wir gesehen, dass $I_x > I_z$, sowie das

$$I_x = 0.020 \pm 0.006 \text{ kgm}^2$$

Hier haben wir I_x aus der Umlauffrequenz der momentanen Drehachse bestimmt.

Dasselbe Trägheitsmoment haben wir dann nochmal aus der Nutationsfrequenz bestimmt. Dabei war unser Ergebnis

$$I_{x,n} = 0.0175 \pm 0.005 \text{ kgm}^2$$

Die Abweichung zwischen beiden Werten entspricht

$$\sigma = 0.33$$

Die Werte stimmen also innerhalb der Fehlergrenzen überein.

Wie auch schon vorhin erwähnt, waren unsere Messwerte teilweise nicht sehr übereinstimmend. Bei den Messreihen für die verschiedenen Drehmomente gibt es bei den letzten beiden Messreihen einen Messwert des nicht richtig mit den anderen übereinstimmt. Bei der letzten Messreihe ist die Erklärung relativ einfach, da wir wie auch schon vorhin erwähnt die Frequenz für den letzten Messwert noch einmal erhöht haben, anstatt von der höchsten Frequenz zur niedrigsten zu messen. Deshalb konnten wir diese Messreihe auch guten Gewissens aus dem Mittelwert auslassen. Bei der zweitletzten Messreihe ist die Erklärung nicht so einfach. Es ist gut möglich dass nach dem Messen der Frequenz durch das erneute Ausrichten des Kreisels die Frequenz verfälscht wurde, sodass der Zusammenhang nicht linear erscheint.

Bei der Bestimmung von I_x durch die Frequenz der momentanen Drehachse hatten wir ebenfalls zwei Messwerte die stark abweichen. Auch hier ist die Erklärung, dass wir nach den ersten beiden Messwerten den Kreisel nochmal beschleunigt haben, weshalb die ersten beiden Messwerte wieder nicht "von oben nach unten" gemessen wurden. Hier ist die Abweichung jedoch nicht so gravierend, dass wir die Messwerte ignorieren mussten.