

Karol Peszek – Opracowanie pomiarów z zadania M8

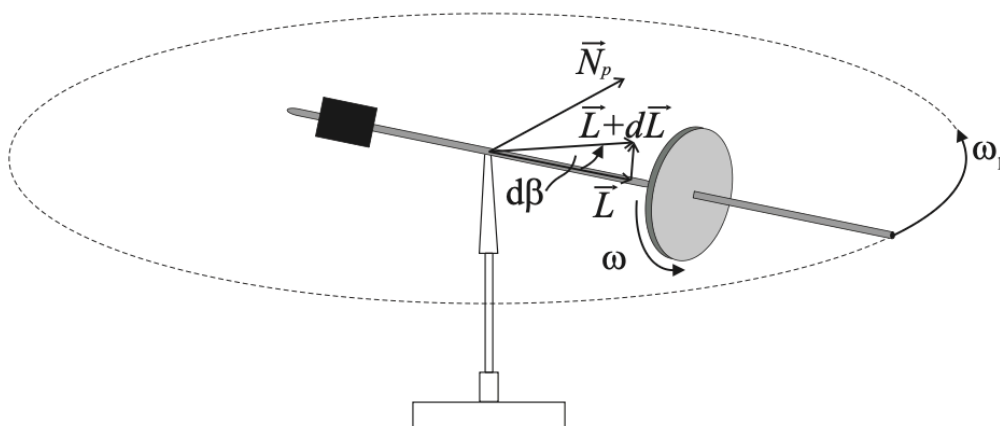
Zestaw pomiarowy 24

1. Cel ćwiczenia

Celem doświadczenia jest zbadanie dynamiki ruchu precesyjnego bąka symetrycznego. W szczególności analizie podlegają:

- Zależność okresu precesji T_p od okresu obrotu tarczy T .
- Wpływ masy dodatkowego obciążenia m na częstość precesji
- Wyznaczenie bezwładności obrotowej tarczy J dwiema metodami dynamicznymi i jedną geometryczną

2. Podstawy teoretyczne



Żyroskop (bąk symetryczny) to bryła sztywna o symetrii obrotowej. Gdy tarcza obraca się z prędkością kątową ω , jej moment pędu wynosi:

$$\vec{L} = J\vec{\omega}$$

Jeżeli na żyroskop działa wewnętrzny moment siły N (pochodzący od ciężarka o masie m na ramieniu r), to powoduje on zmianę kierunku wektora momentu pędu:

$$\vec{N} = \frac{d\vec{L}}{dt}$$

Moment siły jest prostopadły do wektora \vec{L} , zatem żyroskop zaczyna wykonywać ruch precesyjny wokół osi pionowej z prędkością kątową ω_p .

Zależność opisuje wzór:

$$N_p = J \omega_p \omega$$

3. Model matematyczny

W doświadczeniu wykorzystujemy mierzalne parametry, takie jak okres obrotu tarczy T oraz okres precesji T_p . Wykorzystując zależność $\omega = \frac{2\pi}{T}$ oraz $\omega_p = \frac{2\pi}{T_p}$, otrzymamy równanie robocze:

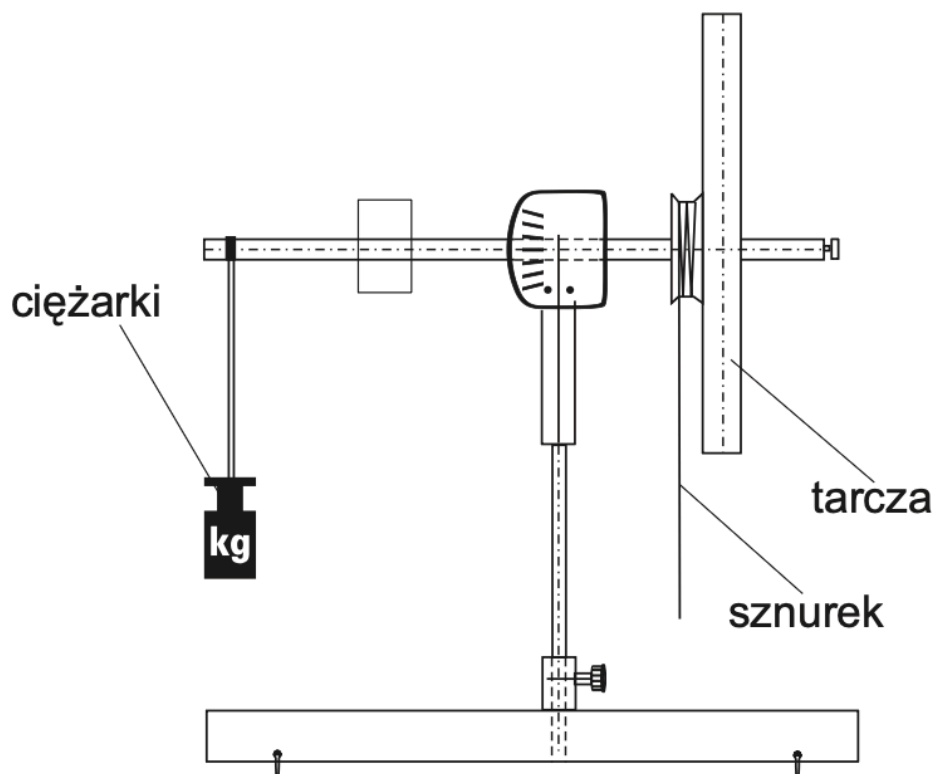
$$m_{gr} = \frac{J \cdot 4\pi^2}{T \cdot T_p}$$

Z powyższego wzoru można wyznaczyć moment bezwładności J poprzez analizę regresji liniowej dla dwóch przypadków:

1. Zmiennej masy m : badając zależność $\frac{1}{T T_p} = f(m)$
2. Zmiennej prędkości obrotowej: badając zależność $\frac{1}{T_p} = g(T)$

Dodatkowo moment bezwładności tarczy (traktowanej jako krążek o masie M_t i promieniu R) obliczany jest ze wzoru geometrycznego:

$$J = \frac{1}{2} M_t R^2$$



Rys. 1.7.2: Żyroskop używany do pomiaru czasu precesji.

4. Ładowanie danych

```
In [1]: #import odpowiednich bibliotek
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [2]: #ładowanie pomiarów średnicy dysku, promienia siły oraz czasu precesji
disk_diameter = pd.read_csv('https://raw.githubusercontent.com/karol...')
force_radius = pd.read_csv('https://raw.githubusercontent.com/karol...')
precession_time = pd.read_csv('https://raw.githubusercontent.com/karol...')

#konwersja tak, aby średnica dysku i promień siły były w metrach
disk_diameter['D'] = disk_diameter['D'] / 100
force_radius['R'] = force_radius['R'] / 100

#konwersja tak, aby masa ciężarka była w kg
precession_time['m'] = precession_time['m'] / 1000

#tworzymy listę mas ciężarków
masses = np.array(precession_time['m'].unique())

#pomiar precesji został dokonany jako połowa okresu więc aby uzyskać
precession_time['T_p'] *= 2
```

5. Obliczenia pośrednie

```
In [3]: #obliczamy średnią średnicę dysku i jej niepewność
D = disk_diameter['D'].mean()
u_D = disk_diameter['D'].std(ddof=1) #użycie ddof=1 dla niepewności

#stosujemy współczynnik studenta, aby uwzględnić małą liczbę pomiarów
u_D *= 2.776 #dla 5 pomiarów

print(f"Średnia średnica dysku: {D:.3f} m ± {u_D:.3f} m")
```

Średnia średnica dysku: 0.260 m ± 0.027 m

```
In [4]: #obliczamy średni promień siły i jego niepewność
R = force_radius['R'].mean()
u_R = force_radius['R'].std(ddof=1) #użycie ddof=1 dla niepewności

#stosujemy współczynnik studenta, aby uwzględnić małą liczbę pomiarów
u_R *= 2.776 #dla 5 pomiarów

print(f"Średni promień siły: {R:.3f} m ± {u_R:.3f} m")
```

Średni promień siły: 0.271 m ± 0.010 m

```
In [5]: #dla każdego pomiaru czasu precesji obliczamy średni okres obrotu t.
precession_time['T'] = (precession_time['T_1'] + precession_time['T_2'])
```

```
In [6]: #dla każdej masy obliczamy 1/T*T_p -> dla każdej masy wartość powinna być stała
precession_time['inv_T_Tp'] = 1 / (precession_time['T'] * precession_time['T_p'])
```

```
In [7]: #dla każdej masy obliczamy średnią wartość 1/(T*T_p) oraz jej niepewność
```

```
T_T_p_inv = np.array([None for _ in masses])
u_T_T_p_inv = np.array([None for _ in masses])

for i, m in enumerate(masses):
    subset = precession_time[precession_time['m'] == m]
    T_T_p_inv[i] = subset['inv_T_Tp'].mean()
    u_T_T_p_inv[i] = subset['inv_T_Tp'].std(ddof=1)

#stosujemy współczynnik studenta, aby uwzględnić małą liczbę pomiarów
u_T_T_p_inv[i] *= 2.252 #dla 10 pomiarów
for i, m in enumerate(masses):
    print(f"Dla masy {m:.3f} kg: 1/(T*T_p) = {T_T_p_inv[i]:.6f} 1/s^2")
```

Dla masy 0.010 kg: $1/(T \cdot T_p) = 0.074541 \text{ 1/s}^2 \pm 0.102413 \text{ 1/s}^2$
Dla masy 0.020 kg: $1/(T \cdot T_p) = 0.159938 \text{ 1/s}^2 \pm 0.140086 \text{ 1/s}^2$
Dla masy 0.030 kg: $1/(T \cdot T_p) = 0.227281 \text{ 1/s}^2 \pm 0.323740 \text{ 1/s}^2$
Dla masy 0.040 kg: $1/(T \cdot T_p) = 0.431432 \text{ 1/s}^2 \pm 0.337438 \text{ 1/s}^2$
Dla masy 0.060 kg: $1/(T \cdot T_p) = 0.524299 \text{ 1/s}^2 \pm 0.369291 \text{ 1/s}^2$
Dla masy 0.070 kg: $1/(T \cdot T_p) = 0.554994 \text{ 1/s}^2 \pm 0.212062 \text{ 1/s}^2$
Dla masy 0.080 kg: $1/(T \cdot T_p) = 0.722577 \text{ 1/s}^2 \pm 0.819681 \text{ 1/s}^2$
Dla masy 0.090 kg: $1/(T \cdot T_p) = 0.715545 \text{ 1/s}^2 \pm 0.328125 \text{ 1/s}^2$
Dla masy 0.100 kg: $1/(T \cdot T_p) = 0.747053 \text{ 1/s}^2 \pm 0.694370 \text{ 1/s}^2$

```
In [8]: #możemy również policzyć błędy względne i sprawdzić, czy są zgodne z oczekiwaniami
relative_errors = u_T_T_p_inv / T_T_p_inv
for i, m in enumerate(masses):
```

```
print(f"Dla masy {m:.3f} kg: błąd względny  $1/(T \cdot T_p) = \{100 * r$ 
```

```
Dla masy 0.010 kg: błąd względny  $1/(T \cdot T_p) = 137.39\%$ 
Dla masy 0.020 kg: błąd względny  $1/(T \cdot T_p) = 87.59\%$ 
Dla masy 0.030 kg: błąd względny  $1/(T \cdot T_p) = 142.44\%$ 
Dla masy 0.040 kg: błąd względny  $1/(T \cdot T_p) = 78.21\%$ 
Dla masy 0.060 kg: błąd względny  $1/(T \cdot T_p) = 70.44\%$ 
Dla masy 0.070 kg: błąd względny  $1/(T \cdot T_p) = 38.21\%$ 
Dla masy 0.080 kg: błąd względny  $1/(T \cdot T_p) = 113.44\%$ 
Dla masy 0.090 kg: błąd względny  $1/(T \cdot T_p) = 45.86\%$ 
Dla masy 0.100 kg: błąd względny  $1/(T \cdot T_p) = 92.95\%$ 
```

```
In [9]: #wykonujemy wykres i regresję liniową dla zależności  $1/(T \cdot T_p)$  od m
a, b = np.polyfit(masses.astype(float), T_T_p_inv.astype(float), 1)
print(f"Współczynniki regresji: a = {a:.2f}, b = {b:.2f}")

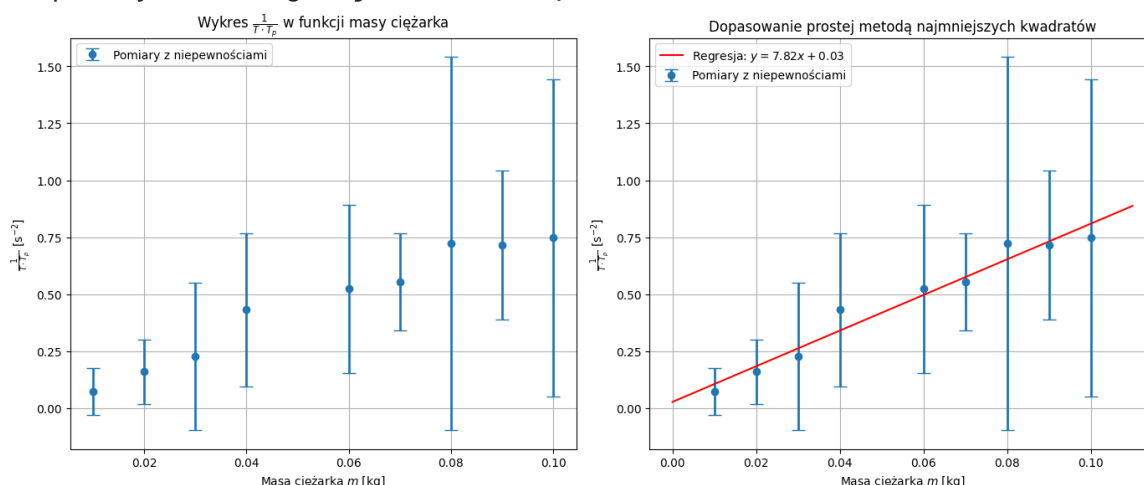
#tworzymy wykresy obok siebie
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

#wykres 1: surowe dane z niepewnościami
ax1.errorbar(masses, T_T_p_inv, yerr=u_T_T_p_inv, fmt='o', label='P')
ax1.set_xlabel(r'Masa ciężarka $m$ [kg]')
ax1.set_ylabel(r'$\frac{1}{T \cdot T_p}$ [s$^{-2}$]')
ax1.set_title(r'Wykres $\frac{1}{T \cdot T_p}$ w funkcji masy ciężar')
ax1.legend(loc='upper left')
ax1.grid()

#wykres 2: dane z regresją liniową
x_plot = np.linspace(0, max(masses) * 1.1, 100)
ax2.errorbar(masses, T_T_p_inv, yerr=u_T_T_p_inv, fmt='o', label='P')
ax2.plot(x_plot, a * x_plot + b, 'r-', label=f'Regresja: $y = \{a:.2$
ax2.set_xlabel(r'Masa ciężarka $m$ [kg]')
ax2.set_ylabel(r'$\frac{1}{T \cdot T_p}$ [s$^{-2}$]')
ax2.set_title(r'Dopasowanie prostej metodą najmniejszych kwadratów')
ax2.legend(loc='upper left')
ax2.grid()

plt.tight_layout()
plt.show()
```

Współczynniki regresji: a = 7.82, b = 0.03



```
In [10]: #analogiczną regresję można wykonać przy założeniu, że prosta przec
```

```

#model y = ax. Metoda najmniejszych kwadratów: a = sum(x*y) / sum(x**2)
x = masses.astype(float)
y = T_T_p_inv.astype(float)
a_fixed_all_masses = np.sum(x * y) / np.sum(x**2)

print(f"Współczynnik regresji (przechodzącej przez początek): a = {a_fixed_all_masses}")

#tworzymy wykresy obok siebie
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

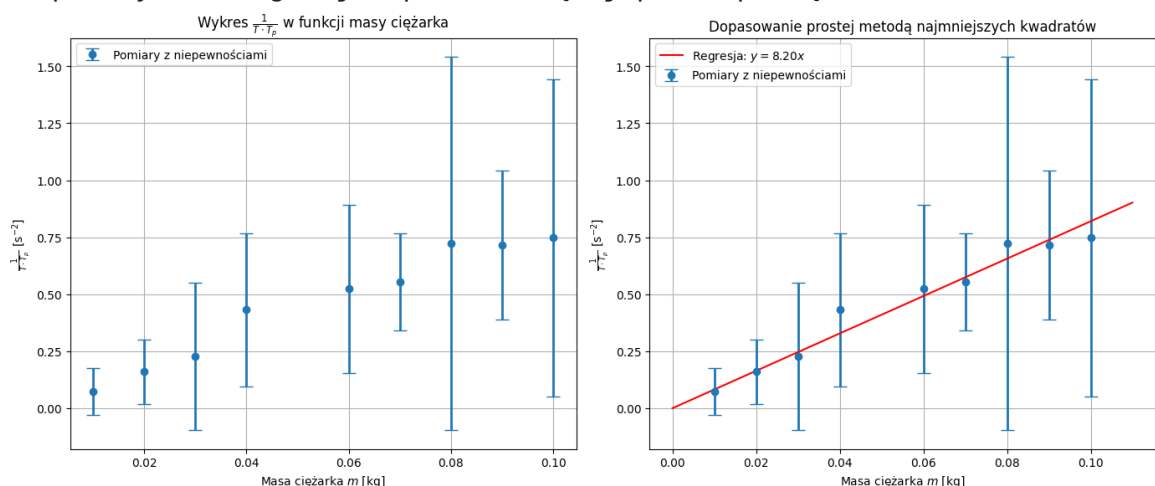
#wykres 1: surowe dane z niepewnościami
ax1.errorbar(masses, T_T_p_inv, yerr=u_T_T_p_inv, fmt='o', label='Pomiary z niepewnościami')
ax1.set_xlabel(r'Masa ciężarka $m$ [kg]')
ax1.set_ylabel(r'$\frac{1}{T \cdot T_p}$ [s$^{-2}$]')
ax1.set_title(r'Wykres $\frac{1}{T \cdot T_p}$ w funkcji masy ciężarka')
ax1.legend(loc='upper left')
ax1.grid()

#wykres 2: dane z regresją liniową
x_plot = np.linspace(0, max(masses) * 1.1, 100)
ax2.errorbar(masses, T_T_p_inv, yerr=u_T_T_p_inv, fmt='o', label='Pomiary z niepewnościami')
ax2.plot(x_plot, a_fixed_all_masses * x_plot, 'r-', label=f'Regresja: y = {a_fixed_all_masses}x')
ax2.set_xlabel(r'Masa ciężarka $m$ [kg]')
ax2.set_ylabel(r'$\frac{1}{T \cdot T_p}$ [s$^{-2}$]')
ax2.set_title(r'Dopasowanie prostej metodą najmniejszych kwadratów')
ax2.legend(loc='upper left')
ax2.grid()

plt.tight_layout()
plt.show()

```

Współczynnik regresji (przechodzącej przez początek): $a = 8.20$



In [11]: #teraz wykonamy wykres i regresję dla zależności $1/T_p$ od T grupując

```

#obliczamy odwrotność okresu precesji
precession_time['inv_Tp'] = 1 / precession_time['Tp']

#przygotowanie kolorów dla różnych mas
colors = plt.cm.viridis(np.linspace(0, 1, len(masses)))
max_T = precession_time['T'].max()

```

```
#wykres dla regresji  $y = ax + b$ 
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 16))

for m, color in zip(masses, colors):
    subset = precession_time[precession_time['m'] == m]
    x = subset['T']
    y = subset['inv_Tp']

    #wykres 1: Punkty pomiarowe
    ax1.plot(x, y, 'o', color=color, label=f'm={m:.3f} kg')

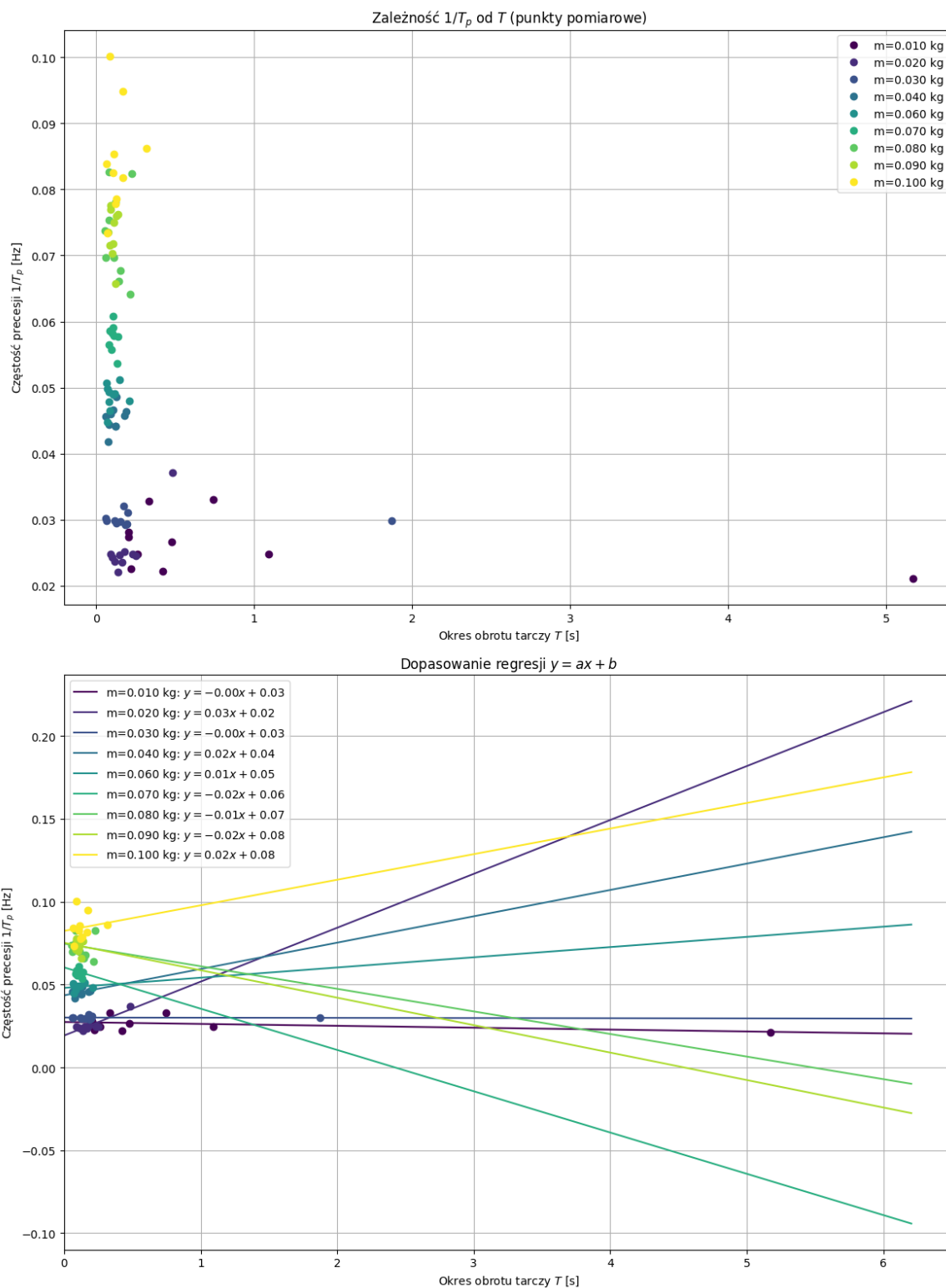
    #regresja  $y = ax + b$ 
    a, b = np.polyfit(x, y, 1)

    #wykres 2: Punkty + Regresja
    ax2.plot(x, y, 'o', color=color)
    x_plot = np.linspace(0, max_T * 1.2, 100)
    ax2.plot(x_plot, a * x_plot + b, '-', color=color, label=f'm={m} kg')

ax1.set_xlabel(r'Okres obrotu tarczy  $T$  [s]')
ax1.set_ylabel(r'Częstość precesji  $1/T_p$  [Hz]')
ax1.set_title(r'Zależność  $1/T_p$  od  $T$  (punkty pomiarowe)')
ax1.legend()
ax1.grid()

ax2.set_xlabel(r'Okres obrotu tarczy  $T$  [s]')
ax2.set_ylabel(r'Częstość precesji  $1/T_p$  [Hz]')
ax2.set_title(r'Dopasowanie regresji  $y=ax+b$ ')
ax2.legend()
ax2.grid()
ax2.set_xlim(left=0)

plt.tight_layout()
plt.show()
```



```
In [12]: #wykres dla regresji  $y = ax$  (przechodzącej przez 0)
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 16))
max_T = precession_time['T'].max()

a_fixed_by_mass = {}

for m, color in zip(masses, colors):
    subset = precession_time[precession_time['m'] == m]
    x = subset['T']
    y = subset['inv_Tp']

    #wykres 1: Punkty pomiarowe
```



```
ax1.plot(x, y, 'o', color=color, label=f'm={m:.3f} kg')

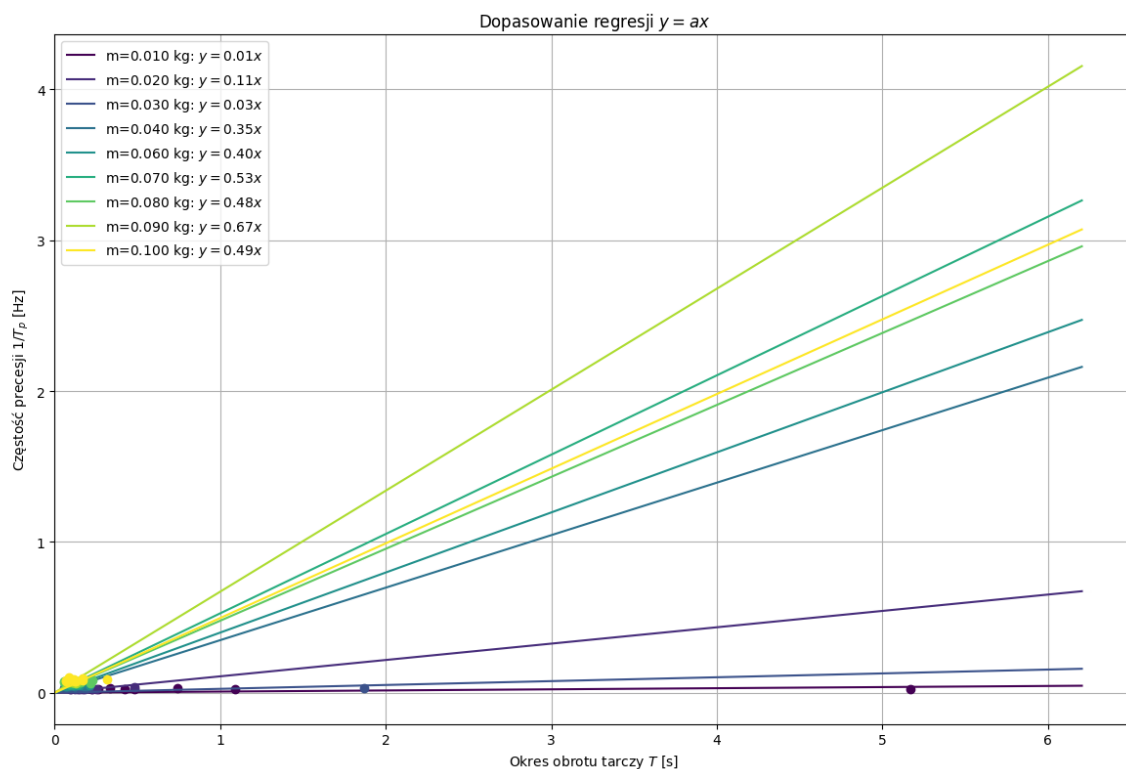
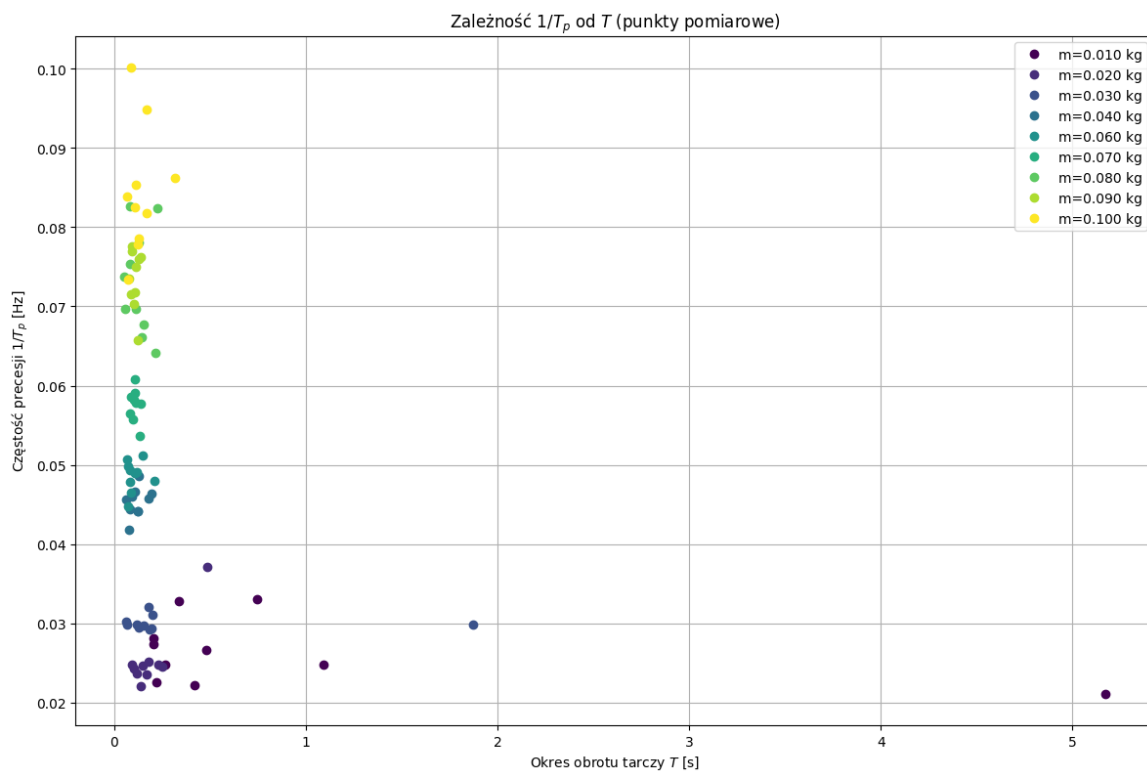
#regresja y = ax
#metoda najmniejszych kwadratów dla y=ax: a = sum(x*y) / sum(x^2)
a_fixed_by_mass[m] = np.sum(x * y) / np.sum(x**2)

#wykres 2: Punkty + Regresja
ax2.plot(x, y, 'o', color=color)
x_plot = np.linspace(0, max_T * 1.2, 100)
ax2.plot(x_plot, a_fixed_by_mass[m] * x_plot, '-', color=color,

ax1.set_xlabel(r'Okres obrotu tarczy $T$ [s]')
ax1.set_ylabel(r'Częstość precesji $1/T_p$ [Hz]')
ax1.set_title(r'Zależność $1/T_p$ od $T$ (punkty pomiarowe)')
ax1.legend()
ax1.grid()

ax2.set_xlabel(r'Okres obrotu tarczy $T$ [s]')
ax2.set_ylabel(r'Częstość precesji $1/T_p$ [Hz]')
ax2.set_title(r'Dopasowanie regresji $y=ax$')
ax2.legend()
ax2.grid()
ax2.set_xlim(left=0)

plt.tight_layout()
plt.show()
```



6. Obliczanie momentów bezwładności różnymi sposobami

```
In [13]: #aby policzyć numeryczny moment bezwładności tą metodą musimy ustalić
g = 9.81
pi = 3.41

#ustawianie masy tarczy w kg
M_t = 1.185

#obliczamy promień tarczy
```

```

r = D / 2
u_r = u_D / 2
print(f"Promień tarczy: {r:.3f} m ± {u_r:.3f} m")

```

Promień tarczy: 0.130 m ± 0.014 m

Obliczanie momentu bezwładności przy założeniu, że tarcza jest idealnym dyskiem matematycznym

```

In [14]: #wyznaczenie momentu bezwładności tarczy metodą geometryczną
J_geom = (1/2) * M_t * r**2

#obliczanie niepewności metodą różniczki zupełnej
#zakładamy, że M_t jest stałą, więc liczymy pochodną cząstkową tylko
#J = (1/2) * M_t * D^2
#|dJ/dD| = M_t * D
u_J_geom = ((M_t * r * u_r) ** 2) ** 0.5

print(f"Moment bezwładności (geometryczny): {J_geom:.6f} kg*m^2")
print(f"Niepewność (geometryczna): {u_J_geom:.6f} kg*m^2")

```

Moment bezwładności (geometryczny): 0.009982 kg*m²

Niepewność (geometryczna): 0.002106 kg*m²

Obliczanie momentu bezwładności korzystając z policzonej regresji $\frac{1}{T T_p} = a \cdot m$, gdzie a jest współczynnikiem regresji przechowywanej w zmiennej `a_fixed_all_masses`

Przekształcamy tutaj wzór podany wyżej:

$$mgr = \frac{J \cdot 4\pi^2}{T \cdot T_p} \Big|_{\cdot T T_p}$$

$$mgr T T_p = J \cdot 4\pi^2 \Big|_{\cdot \frac{1}{4\pi^2}}$$

$$J = \frac{mgr T T_p}{4\pi^2}$$

$$J = \frac{mgr}{4\pi^2 \cdot \frac{1}{T T_p}}$$

Korzystamy tutaj z regresji liniowej $\frac{1}{T T_p} = a \cdot m$

$$J = \frac{\cancel{mgr}}{4\pi^2 \cdot a \cdot \cancel{m}}$$

$$J = \frac{gr}{4\pi^2 \cdot a}$$

```

In [15]: J_dyn_all_masses = (g * r) / (4 * (pi ** 2) * a_fixed_all_masses)
#do policzenia niepewności skorzystamy z tego, że uznajemy g, pi oraz

```

```
u_J_dyn_all_masses = (g * u_r) / (4 * (pi ** 2) * a_fixed_all_masse

print(f"Moment bezwładności (dynamiczny): {J_dyn_all_masses:.6f} kg
print(f"Niepewność (dynamiczna): {u_J_geom:.6f} kg*m^2")
```

Moment bezwładności (dynamiczny): 0.003338 kg*m²

Niepewność (dynamiczna): 0.002106 kg*m²

Obliczanie momentu bezwładności korzystając z policzonych regresji $\frac{1}{T_p} = a \cdot T$ dla ustalonej masy m , gdzie a jest współczynnikiem regresji przechowywanym wraz z masami w słowniku `a_fixed_by_mass`

Przekształcamy tutaj wzór podany wyżej:

$$mgr = \frac{J \cdot 4\pi^2}{T \cdot T_p} \Big|_{\cdot T T_p}$$

$$mgr T T_p = J \cdot 4\pi^2 \Big|_{\cdot \frac{1}{4\pi^2}}$$

$$J = \frac{mgr T T_p}{4\pi^2}$$

$$J = \frac{mgr}{4\pi^2 \cdot \frac{1}{T T_p}}$$

Korzystamy tutaj z regresji liniowej $\frac{1}{T_p} = a \cdot T$

$$J = \frac{mgr}{4\pi^2 \cdot \frac{a \cdot \cancel{T}}{\cancel{T}}}$$

$$J = \frac{mgr}{4\pi^2 \cdot a}$$

```
In [16]: #obliczanie momentu bezwładności metodą dynamiczną dla każdej masy
results = {}
for m in masses:
    a_fixed = a_fixed_by_mass[m]
    J_dyn = (m * g * r) / (4 * (pi ** 2) * a_fixed)
    #do policzenia niepewności skorzystamy z tego, że uznajemy g, p
    u_J_dyn = (m * g * u_r) / (4 * (pi ** 2) * a_fixed)
    results[m] = (J_dyn, u_J_dyn)
    print(f"Dla masy {m:.3f} kg: Moment bezwładności (dynamiczny):
```

Dla masy 0.010 kg: Moment bezwładności (dynamiczny): 0.036856 kg*m²
 \pm 0.003888 kg*m²
 Dla masy 0.020 kg: Moment bezwładności (dynamiczny): 0.005050 kg*m²
 \pm 0.000533 kg*m²
 Dla masy 0.030 kg: Moment bezwładności (dynamiczny): 0.032134 kg*m²
 \pm 0.003389 kg*m²
 Dla masy 0.040 kg: Moment bezwładności (dynamiczny): 0.003146 kg*m²
 \pm 0.000332 kg*m²
 Dla masy 0.060 kg: Moment bezwładności (dynamiczny): 0.004124 kg*m²
 \pm 0.000435 kg*m²
 Dla masy 0.070 kg: Moment bezwładności (dynamiczny): 0.003644 kg*m²
 \pm 0.000384 kg*m²
 Dla masy 0.080 kg: Moment bezwładności (dynamiczny): 0.004592 kg*m²
 \pm 0.000484 kg*m²
 Dla masy 0.090 kg: Moment bezwładności (dynamiczny): 0.003680 kg*m²
 \pm 0.000388 kg*m²
 Dla masy 0.100 kg: Moment bezwładności (dynamiczny): 0.005531 kg*m²
 \pm 0.000583 kg*m²

```
In [17]: #finalnie możemy przedstawić podsumowanie wyników (momenty bezwładności)
summary_data = {
    'Metoda': [],
    'Masa ciężarka (kg)': [],
    'Moment bezwładności (kg*m^2)': [],
    'Niepewność (kg*m^2)': [],
    'Niepewność względna (%)': []
}

#dodajemy wynik geometryczny
summary_data['Metoda'].append('Geom.')
summary_data['Masa ciężarka (kg)'].append('N/A')
summary_data['Moment bezwładności (kg*m^2)'].append(J_geom)
summary_data['Niepewność (kg*m^2)'].append(u_J_geom)
summary_data['Niepewność względna (%)'].append((u_J_geom / J_geom) * 100)

#dodajemy wynik dynamiczny dla całego zbioru danych
summary_data['Metoda'].append('Dyn. (wszystkie masy)')
summary_data['Masa ciężarka (kg)'].append('N/A')
summary_data['Moment bezwładności (kg*m^2)'].append(J_dyn_all_masses)
summary_data['Niepewność (kg*m^2)'].append(u_J_dyn_all_masses)
summary_data['Niepewność względna (%)'].append((u_J_dyn_all_masses / J_dyn_all_masses) * 100)

#dodajemy wyniki dynamiczne dla każdej masy
for m in masses:
    J_dyn, u_J_dyn = results[m]
    summary_data['Metoda'].append('Dyn.')
    summary_data['Masa ciężarka (kg)'].append(m)
    summary_data['Moment bezwładności (kg*m^2)'].append(J_dyn)
    summary_data['Niepewność (kg*m^2)'].append(u_J_dyn)
    summary_data['Niepewność względna (%)'].append((u_J_dyn / J_dyn) * 100)

summary_df = pd.DataFrame(summary_data)
print("\nPodsumowanie wyników momentu bezwładności:")

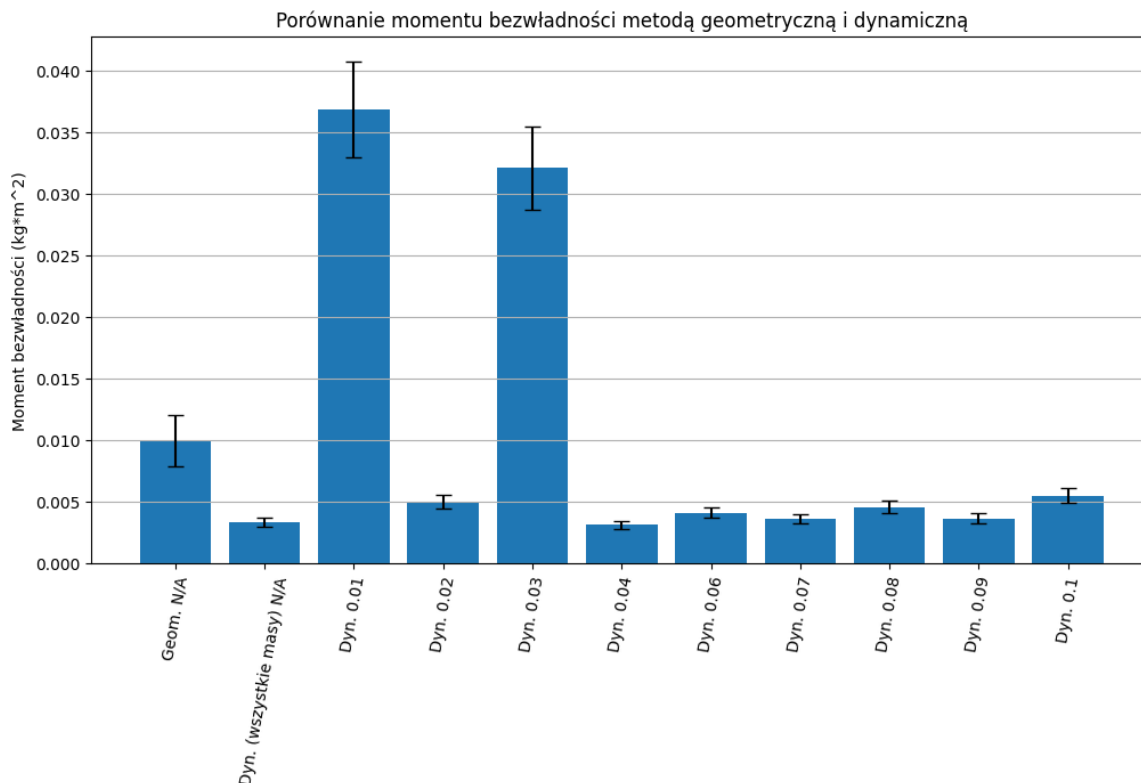
summary_df.head(len(summary_df))
```

Podsumowanie wyników momentu bezwładności:

Out[17]:

	Metoda	Masa ciężarka (kg)	Moment bezwładności (kg*m ²)	Niepewność (kg*m ²)	Niepewność względna (%)
0	Geom.	N/A	0.009982	0.002106	21.096052
1	Dyn. (wszystkie masy)	N/A	0.003338	0.000352	10.548026
2	Dyn.	0.01	0.036856	0.003888	10.548026
3	Dyn.	0.02	0.005050	0.000533	10.548026
4	Dyn.	0.03	0.032134	0.003389	10.548026
5	Dyn.	0.04	0.003146	0.000332	10.548026
6	Dyn.	0.06	0.004124	0.000435	10.548026
7	Dyn.	0.07	0.003644	0.000384	10.548026
8	Dyn.	0.08	0.004592	0.000484	10.548026
9	Dyn.	0.09	0.003680	0.000388	10.548026
10	Dyn.	0.1	0.005531	0.000583	10.548026

```
In [18]: #przedstawiamy również te dane w formie wykresu słupkowego z niepewnościami
plt.figure(figsize=(12, 6))
bars = plt.bar(summary_df.index, summary_df['Moment bezwładności (kg*m^2)'])
plt.xticks(summary_df.index, summary_df['Metoda'] + ' ' + summary_df['Masa ciężarka (kg)'])
plt.ylabel('Moment bezwładności (kg*m^2)')
plt.title('Porównanie momentu bezwładności metodą geometryczną i dynamiczną')
plt.grid(axis='y')
plt.show()
```



7. Interpretacja otrzymanych wyników

Jak można zauważyć na wykresie słupkowym model geometryczny oraz modele dynamiczne dla mas 10g i 30g stanowczo odbiegają od reszty zarówno w kwestii wartości jak i dużo większego błędu. Co do modeli dynamicznych dla mas 10g i 30g powodem takich rozbieżności może być dużo większy błąd względny wyrażenia $\frac{1}{TT_p}$, które dla mas 10g i 30g wynoszą około 60%, a dla pozostałych mas wynoszą około 30-40%. Oprócz tego wyniki te pokazują, że model geometryczny mógł nie być odpowiedni dla tego dysku, na przykład ze względu na inny kształt który większą część masy skupiał dalej od osi obrotu.

8. Obliczanie ostatecznego wyniku wraz z niepewnością

W tym celu obliczymy wynik odrzucając pomiary dla 10g i 30g i korzystając z obliczonych wartości dla pozostałych mas ciężarków.

```
In [19]: #usuwanie mas 10g i 30g
results_filtered = {m: results[m] for m in results if m not in [0.01, 0.03]}
print("\nWyniki po usunięciu mas 10g i 30g:")
for m in results_filtered:
    J_dyn, u_J_dyn = results_filtered[m]
    print(f"Dla masy {m:.3f} kg: Moment bezwładności (dynamiczny):")
```

Wyniki po usunięciu mas 10g i 30g:

Dla masy 0.020 kg: Moment bezwładności (dynamiczny): 0.005050 kg*m²
 ± 0.000533 kg*m²
 Dla masy 0.040 kg: Moment bezwładności (dynamiczny): 0.003146 kg*m²
 ± 0.000332 kg*m²
 Dla masy 0.060 kg: Moment bezwładności (dynamiczny): 0.004124 kg*m²
 ± 0.000435 kg*m²
 Dla masy 0.070 kg: Moment bezwładności (dynamiczny): 0.003644 kg*m²
 ± 0.000384 kg*m²
 Dla masy 0.080 kg: Moment bezwładności (dynamiczny): 0.004592 kg*m²
 ± 0.000484 kg*m²
 Dla masy 0.090 kg: Moment bezwładności (dynamiczny): 0.003680 kg*m²
 ± 0.000388 kg*m²
 Dla masy 0.100 kg: Moment bezwładności (dynamiczny): 0.005531 kg*m²
 ± 0.000583 kg*m²

```
In [20]: #obliczanie średniej ważonej J po usunięciu mas 10g i 30g
final_J = np.sum([m[0] / (m[1] ** 2) for m in results_filtered.values()])
final_u_J = (1 / np.sum([1 / (m[1] ** 2) for m in results_filtered.values()]))

print(f"Średnia ważona J po usunięciu mas 10g i 30g: {final_J:.6f} kg*m^2")
print(f"Błąd względny: {100 * final_u_J / final_J:.2f}%")
```

Średnia ważona J po usunięciu mas 10g i 30g: 0.003976 kg*m² ± 0.000161 kg*m²
 Błąd względny: 4.05%

Zatem ostatecznie wyliczony moment bezwładności tarczy to

$$J = 0.003976 \text{ kg} \cdot \text{m}^2 \pm 0.000161 \text{ kg} \cdot \text{m}^2$$