# 1 Analisi dei dati sperimentali: Determinazione di g

In questo esperimento, determineremo il valore dell'accelerazione di gravità $g$ utilizzando un pendolo semplice. La teoria ci dice che il periodo $T$ di un pendolo semplice è legato alla sua lunghezza $L$ dalla seguente equazione:

$$T = 2\pi\sqrt{\frac{L}{g}}$$

Elevando al quadrato entrambi i membri, otteniamo:

$$T^2 = \frac{4\pi^2}{g}L$$

o, ancora,

$$L = \frac{gT^2}{4\pi^2}$$

Questa è l'equazione di una retta del tipo $y = mx + q$, dove $y = L$, $x = T^2$, $m = \frac{g}{4\pi^2}$ e $q = 0$. Quindi, se plottiamo $L$ in funzione di $T^2$, dovremmo ottenere una retta la cui pendenza $m$ è legata a $g$ dalla relazione:

$$g = 4\pi^2 m$$

Utilizzeremo Python per analizzare i dati sperimentali e determinare $g$. Ecco il codice commentato:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Dati sperimentali
L = np.array([0.15, 0.2, 0.25, 0.3, 0.34, 0.4, 0.45, 0.54, 0.6])
# t_dati sono le misure ripetute della durata di 20 oscillazioni
    per ogni lunghezza
t_dati = np.array ([[16.8,16.7,  16.7,  16.3 , 16.9],[18.6 , 18.5 ,
     18.6 , 18.5, 18.7],[20.2 ,20.3 , 20.3 , 20.2, 20.4], [22.6,
    22.6,  22.7 , 22.5,  22.8],[23.4 , 23.5 , 23.4 , 23.1 ,
    23.6],[25.8 ,25.7, 25.8,  25.9, 25.7],[27.9, 28.0, 28.0, 27.8,
    27.7],[29.7, 29.7, 29.6,  29.8, 29.7,],[31.4,  31.5 , 31.5,
    31.4, 31.3]])
t =  np.mean(t_dati, axis=1) # tempi medi di 20 oscillazioni
T = t/20.0 #periodo calcolato
T2 = T**2 #quadrati dei periodi calcolati
erroriL=np.array
    ([0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001]) #
    errori sulle lunghezze (in metri)
massimi_colonne = np.max(t_dati, axis=1)
minimi_colonne = np.min(t_dati, axis=1)
errorit = np.array((massimi_colonne - minimi_colonne) / 2)  #
    calcoliamo l'array di errori ottenuti con la semidispersione
erroriT=errorit/20  #l'errore su T propagato
erroriT2=2*T*erroriT #l'errore su T^2 propagato
```

```
18
19 # Definizione della funzione di regressione lineare
20 def linear_fit(x, a, pendenza):
21     return a + pendenza * x
22
23 # Fitting dei dati sperimentali
24 params, covariance = curve_fit(linear_fit, T2, L)
25
26 # Estrazione dei parametri
27 a, pendenza = params
28 err_a, err_pendenza = np.sqrt(np.diag(covariance))
29 err_g =  4 * np.pi**2*err_pendenza
30
31 # Calcolo di g
32 g = 4 * np.pi**2 * pendenza
33
34 # Creazione del modello lineare per il grafico
35 T2_fit = np.linspace(min(T2), max(T2), 100)
36 L_fit = linear_fit(T2_fit, a, pendenza)
37
38 # Creiamo il grafico
39 larghezza_originale, altezza_originale = plt.gcf().get_size_inches
       () # dimensioni originali del grafico (piccole)
40 fattore_di_scala = 2.6 # il fattore di ingrandimento (occorre fare
       tentativi per trovare quello giusto)
41 nuova_larghezza = larghezza_originale * fattore_di_scala #nuove
       dimensioni riscalate
42 nuova_altezza = altezza_originale * fattore_di_scala
43 plt.figure(figsize=(nuova_larghezza, nuova_altezza)) #grafico
       ingrandito
44 plt.errorbar(T2, L, xerr=erroriT2, yerr=erroriL, fmt='o',
       markersize=1, label='Dati con errori su x e y', color='black',
       ecolor='blue')  # barre di errore
45 plt.plot(T2, pendenza * T2 + a, color='red', linewidth=0.5, label='
       Retta di regressione')  # retta di regressione
46 plt.xlabel('T^2 (m^2)') #etichetta asse x
47 plt.ylabel('L (m)') #etichetta asse y
48 plt.title('Determinazione sperimentale di g') #titolo del grafico
49 txt = f'g = {g:.3f} m/s^2\nErrore su g = {err_g:.3f} m/s^2'
50 plt.text(0.1, 0.85, txt, transform=plt.gca().transAxes, bbox=dict(
       facecolor='white', alpha=0.5)) #risultato per g in un box
       informativo
51 plt.grid(True) #mostriamo la griglia
52
53 # Stampa dei risultati
54 print(f'a = ({a:.4f} \pm {err_a:.4f}) m')
55 print(f'pendenza = ({pendenza:.4f} \pm {err_pendenza:.4f}) m/s^2')
56 print(f'g = ({g:.1f} \pm {err_g:.1f}) m/s^2')
57
58 # Mostra il grafico
59 plt.show()
```

## 2 Spiegazione del codice

1. Importiamo le librerie necessarie: numpy per calcoli numerici, matplotlib per la creazione di grafici e scipy.optimize per il fitting dei dati.
2. Definiamo i dati sperimentali: lunghezze del pendolo e tempi per 20 oscillazioni.
3. Calcoliamo i periodi medi e i loro quadrati.
4. Calcoliamo gli errori sulle misure utilizzando la semidispersione.
5. Definiamo la funzione di regressione lineare.
6. Utilizziamo curve_fit per trovare i parametri della retta di best fit.
7. Estraiamo i parametri e calcoliamo $g$ e il suo errore.
8. Creiamo il grafico con i dati sperimentali, le barre di errore e la retta di regressione.
9. Stampiamo i risultati.

Questo codice permette di analizzare i dati sperimentali in modo efficiente, fornendo una stima di $g$ con il relativo errore e una rappresentazione grafica dei risultati.