Analisi Statistica dei Dati Sperimentali

Laboratorio di Fisica

Indice

1 Introduzione

In questo capitolo esploreremo l'analisi statistica dei dati sperimentali utilizzando Python. Gli argomenti trattati includeranno la gestione delle misure come variabili casuali, la funzione di distribuzione normale, l'uso di matplotlib per la visualizzazione dei dati e la regressione lineare. Eseguiremo anche un esempio completo per il calcolo dell'accelerazione di gravitá.

2 Misure come Variabili Casuali

Le misure sperimentali sono trattate come variabili casuali, il che implica che i valori misurati sono soggetti a variazioni casuali dovute a errori di misurazione e condizioni sperimentali.

3 Distribuzione Normale e Stima dei Parametri

La distribuzione normale è spesso utilizzata per modellare errori e incertezze nelle misure. La funzione di distribuzione normale è definita da due parametri principali: la media e la deviazione standard.

3.1 Esempio di Stima dei Parametri

Consideriamo i seguenti dati sperimentali:

Misura	Valore
1	4.900
2	19.600
3	44.100
4	78.400
5	122.500

Il codice Python per calcolare la media e la deviazione standard è:

```python
import numpy as np

# Dati sperimentali
dati = np.array([4.900, 19.600, 44.100, 78.400, 122.500])

# Calcolo della media e della deviazione standard
media = np.mean(dati)
deviazione_standard = np.std(dati, ddof=1)

print(f"Media: {media:.3f} m")
print(f"Deviazione standard: {deviazione_standard:.3f} m")
```

Listing 1: Calcolo della media e deviazione standard

4 Uso di Matplotlib per la Visualizzazione dei Dati

Matplotlib è una libreria potente per la creazione di grafici in Python. Di seguito è riportato un esempio di come realizzare un grafico delle misure sperimentali con barre di errore.

4.1 Esempio di Grafico

Consideriamo l'esempio dell'accelerazione di gravità con i seguenti dati:

Tempo (s)	Distanza (m)
1.0	4.900
2.0	19.600
3.0	44.100
4.0	78.400
5.0	122.500

Per mostrare le barre di errore, inseriamo nel codice errori volutamente molto grandi. Si noti inoltre che abbiamo modificato la grandezza dei punti col parametro **markersize**, la larghezza dei segmenti per le barre con il parametro **capsize**, e la dimensione del grafico con l'istruzione **plt.figure(figsize=(12, 9))**.

```
import numpy as np
import numpy as np
import matplotlib.pyplot as plt

# Dati sperimentali
tempo = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
distanza = np.array([4.900, 19.600, 44.100, 78.400, 122.500])
error_distanza = np.array([1, 2, 3, 1, 4])  # Errori aumentati
error_t2 = np.array([1, 1, 1, 1, 1])

# Configurazione per usare LaTeX in Matplotlib


# Creazione del grafico con dimensioni maggiori e punti più piccoli
plt.figure(figsize=(12, 9))  # Dimensioni della figura
plt.errorbar(tempo**2, distanza, yerr=error_distanza, xerr=error_t2, fmt='o', label='Dati sperimentali', capsize=5, elinewidth=2, markersize=6)
plt.title(r'Misurazione dell\'Accelerazione di gravità', fontsize=16)
plt.xlabel(r'Tempo al quadrato $(s^2)$', fontsize=14)
plt.ylabel(r'Distanza (m)', fontsize=14)
plt.legend()
plt.grid(True)
plt.savefig('grafico_misure.png')  # Salvataggio dell'immagine
plt.show()
```

Listing 2: Grafico delle misure sperimentali con barre di errore

Nella figura 1 è mostrato il grafico risultante.

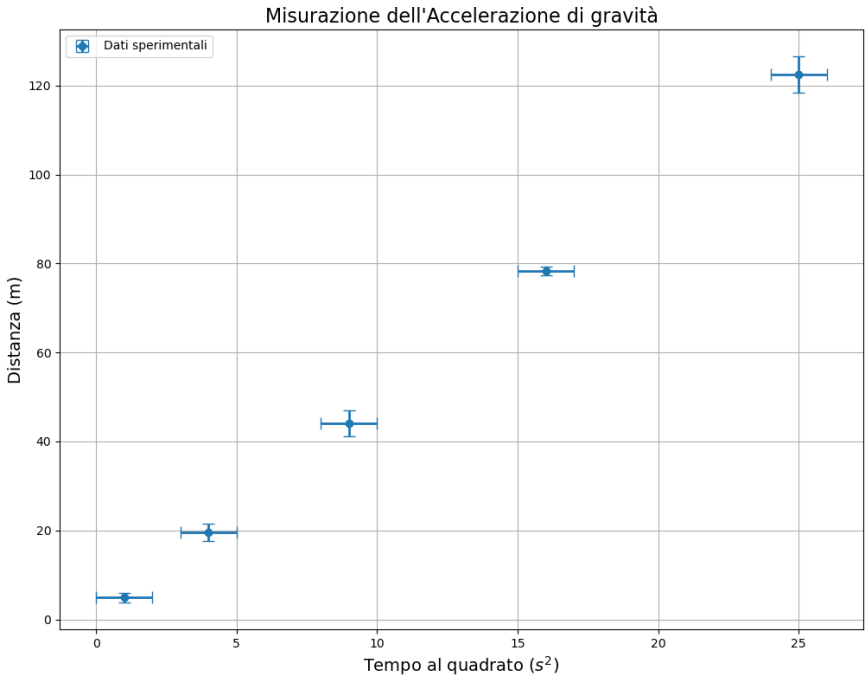

Figura 1: Grafico delle misure sperimentali con barre di errore.

5 Regressione Lineare

In fisica, il moto uniforme è un tipo di moto in cui un oggetto si sposta con velocità costante. Questo significa che la distanza percorsa dall'oggetto è proporzionale al tempo impiegato. La relazione tra spazio y e tempo x in un moto uniforme può essere espressa tramite la seguente equazione lineare:

$$y = v \cdot x + s_0$$

dove:

- v è la velocità costante dell'oggetto (pendenza della retta).
- s_0 è la posizione iniziale dell'oggetto (intercetta della retta).

L'obiettivo è trovare i parametri v e s_0 che meglio rappresentano i dati sperimentali. Utilizzando una regressione lineare, possiamo ottenere questi parametri adattando una retta ai dati.

5.1 Tabella dei Dati

Per l'analisi, consideriamo i seguenti dati sperimentali raccolti per tempo e spazio:

Tempo (s)	**Spazio** (m)
1.0	2.0
2.0	4.0
3.0	6.0
4.0	8.0
5.0	10.0

Tabella 1: Dati sperimentali di tempo e spazio.

5.2 Regressione Lineare

La regressione lineare cerca di adattare una retta ai dati sperimentali, trovando i parametri a e b che minimizzano la somma dei quadrati delle differenze tra i valori osservati e quelli previsti dalla retta. In questo caso, la retta di regressione è data da:

$$y = a \cdot x + b$$

dove:

- a è la pendenza della retta, che rappresenta la velocità v.
- b è l'intercetta, che rappresenta la posizione iniziale s_0.

5.3 Uso di Python per la Regressione Lineare

Per calcolare i parametri della retta di regressione e il loro errore, utilizziamo il modulo `scipy.optimize.curve_fit` di Python, che permette di adattare una funzione ai dati sperimentali. La funzione `curve_fit` ritorna i parametri ottimizzati e le loro deviazioni standard. Utilizziamo anche `matplotlib` per visualizzare i dati e la retta di regressione.

Ecco il codice Python utilizzato:

```python
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
# Dati sperimentali
tempo = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
spazio = np.array([2.0, 4.0, 6.0, 8.0, 10.0])

# Definizione della funzione di modello lineare
def linear_model(x, a, b):
    return a * x + b

# Fitting dei dati
params, params_covariance = curve_fit(linear_model, tempo, spazio)

# Estrazione dei parametri
pendenza, intercetta = params
pendenza_error, intercetta_error = np.sqrt(np.diag(params_covariance))

print(f"Pendenza: {pendenza:.3f} ś {pendenza_error:.3f}")
print(f"Intercetta: {intercetta:.3f} ś {intercetta_error:.3f}")

# Creazione del grafico
plt.figure(figsize=(8, 6))
plt.scatter(tempo, spazio, label='Dati sperimentali')
plt.plot(tempo, linear_model(tempo, *params), 'r-', label='Retta di regressione')
plt.xlabel('Tempo (\si{\second})')
plt.ylabel('Spazio (\si{\meter})')
plt.title('Fitting Lineare')
plt.legend()
plt.grid(True)
plt.savefig('regressione_lineare.png')  # Salvataggio dell'immagine
plt.show()
```

5.4 Output del Codice Python

L'output del codice Python è il seguente:

Pendenza: $2.020 \pm 0.083\,\mathrm{m\,s^{-1}}$
Intercetta: $0.200 \pm 0.276\,\mathrm{m}$

Pendenza: La pendenza della retta di regressione è $2.020 \pm 0.083\,\mathrm{m\,s^{-1}}$. Questo valore rappresenta la velocità media del moto. L'errore associato indi-

ca l'incertezza nella determinazione della pendenza, che riflette la variabilità dei dati sperimentali rispetto alla retta di regressione.

Intercetta: L'intercetta della retta di regressione è 0.200 ± 0.276 m. Questo valore rappresenta la posizione iniziale, cioè il valore di y quando x è zero. L'errore associato all'intercetta indica l'incertezza nella misura del punto in cui la retta di regressione interseca l'asse delle ordinate. Questo valore può dare indicazioni sul punto di partenza del movimento descritto dai dati.

Di seguito , in figura 2 è mostrato il grafico della regressione lineare che visualizza i dati sperimentali insieme alla retta di regressione ottenuta.

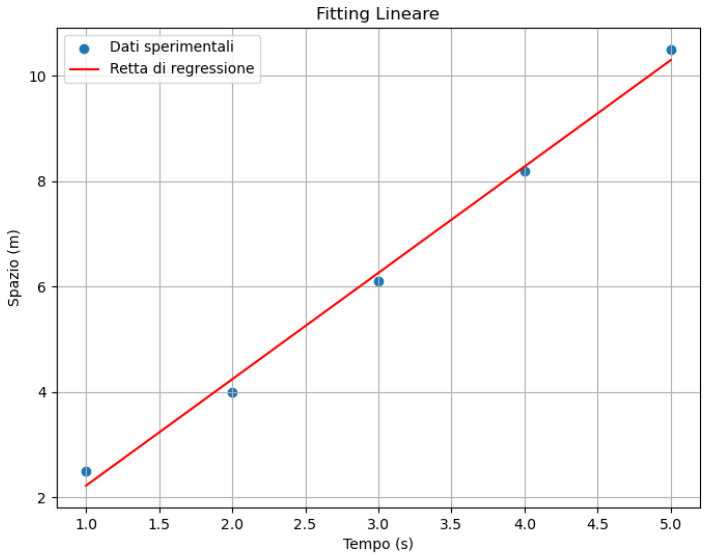

Figura 2: Grafico della regressione lineare: i dati sperimentali sono mostrati come punti, e la retta di regressione è mostrata in rosso.