Laboratorio di Fisica 1
R4: Misura di variabili aleatorie

Gruppo 17: Bergamaschi Riccardo, Graiani Elia, Moglia Simone

8/11/2023 – 15/11/2023

Sommario

Il gruppo di lavoro ha misurato due variabili aleatorie, osservando come queste rispecchino le rispettive distribuzioni teoriche (di Bernoulli e di Poisson).

1 Processo di Bernoulli

1.1 Materiali e strumenti di misura utilizzati

Materiali	**Descrizione/Note**
Sei dadi	Di colori diversi, per poterli identificare facilmente
Calcolatore	Computer portatile[4]

1.2 Dati sperimentali

Eseguiamo 400 lanci simultanei dei sei dadi, riportando in una tabella il risultato di ciascuno di essi, ordinati secondo una successione cromatica prestabilita.

1.3 Simulazione

Successivamente, simuliamo 10^{12} lanci di 5 dadi su un calcolatore tramite un programma da noi compilato[5]. Calcolato il valore medio e la deviazione standard del numero di "1" usciti per ogni lancio dei dadi, riusciremo ad osservare se i lanci da noi effettuati rispecchiano la distribuzione teorica bernoulliana.

[4] Intel core i5 gen 7 dual-core 1.6 GHz
[5] *Vedi* Appendice 1

2 Processo di Poisson

2.1 Materiali e strumenti di misura utilizzati

Strumento di misura	**Soglia**	**Portata**	**Sensibilità**
Contatore Geiger	1 conteggi/s	???	1 conteggi/s
Metro a nastro	0.1 cm	300.0 cm	0.1 cm

Altro	**Descrizione/Note**
Sei dadi	Usati per riprodurre un processo bernoulliano.
Campione di Torio-232	Usato per riprodurre un processo poissoniano.

2.2 Esperienza e procedimento di misura

1. Acceso e impostato adeguatamente il contatore Geiger, misuriamo per 5 diverse distanze tra contatore e campione il numero di raggi γ_{ij} emessi da quest'ultimo in 3600 intervalli da un secondo.
2. Direzionato il contatore dal verso opposto rispetto al campione, misuriamo il numero di raggi γ_{amb} rilevabili che definiscono il fondo dei nostri dati.

Fissato un sistema di riferimento solidale all'apparato di misura, con origine nella posizione di partenza delle sferette e $\hat{x} = \hat{g}$, possiamo scrivere la seguente legge del moto:

$$x(t) = \frac{1}{2} g t^2$$

Per ogni sferetta i, posto $x\left(\overline{t_i}\right) = d_0 - \frac{1}{2}\varnothing_i$, la norma di $\vec{g}$ è ricavabile da:

$$g = \frac{2d_0 - \varnothing_i}{\left(\overline{t_i}\right)^2}$$

dove determiniamo l'errore su g propagando gli errori su $d_0, \varnothing_i$ e $\overline{t_i}$, avendo posto

$$\delta\overline{t_i} = \sigma_{\overline{t_i}} = \frac{\sigma_{t_i}}{\sqrt{100}} = \frac{\sigma_{t_i}}{10}.$$

Di seguito riportiamo gli istogrammi dei tempi e i valori di g così ottenuti.

Figura 1: Istogrammi dei dati t_1 e t_2 raccolti

i	$\varnothing_i$ (mm)	$\overline{t_i}$ (ms)	g (m/s^2)	ε
A	24.63 ± 0.01	503.62 ± 0.03	9.83 ± 0.02	1.00
B	22.23 ± 0.01	503.91 ± 0.03	9.82 ± 0.02	0.93

2.3 Esperienza sulla distribuzione di Poisson

1. Consideriamo la distanza tra i due fototraguardi e impostiamo i fotodiodi del contatore su A+B.
2. Usando solo
   (a) Appeso il campione alla molla, allineiamo i due fototraguardi aiutandoci con la livella, in modo tale che possano rilevare le oscillazioni nel modo più accurato possibile;
   (b) Tiriamo leggermente il campione verso il basso e poi lo rilasciamo, in modo che il sistema molla inizi a oscillare con direzione il più possibile parallela a $\vec{g}$;
   (c) Attesa la stabilizzazione dell'oscillazione, avviamo l'acquisizione della misura di un tempo (20 periodi) $20T_i$.
   (d) Ripetiamo molte volte (in tutto N_{20T_i}) i punti (b) e (c). In particolare, $N_{20T_A} = N_{20T_B} = 25$ e $N_{20T_C} = N_{20T_{A+B}} = 30$.
3. Infine, misuriamo con la bilancia, separatamente, la massa della molla m_m e la massa del gancio m_g.

Infatti, nel caso dinamico, il contributo di queste masse *non* si annulla; in particolare, la massa del gancio contribuisce appieno (in quanto è solidale col grave), mentre la massa della molla contribuisce per circa $\frac{1}{3}$. La massa effettiva da considerare per ogni grave sarà allora:

$$((m_{\text{eff}})_i)_{\text{best}} = (m_i)_{\text{best}} + (m_g)_{\text{best}} + \frac{1}{3}(m_m)_{\text{best}}$$

$$\delta\,(m_{\text{eff}})_i = \delta m_i + \delta m_g + \frac{1}{3}\delta m_m$$

Di seguito sono riportate le distribuzioni dei dati raccolti:

Figura 2: Istogrammi dei periodi delle oscillazioni di A e B

Figura 3: Istogrammi dei periodi delle oscillazioni di C e $A + B$

Poiché i nostri dati hanno assunto distribuzioni grossolanamente approssimabili a gaussiane, possiamo procedere al calcolo di k, utilizzando, per ogni grave i, i seguenti valori:

$$(20T_i)_{\text{best}} = \overline{20T_i} \qquad \wedge \qquad \delta\,(20T_i) = \sigma_{\overline{20T_i}} = \frac{\sigma_{20T_i}}{\sqrt{N_{20T_i}}}$$

dove $\overline{20T_i}$ e σ_{20T_i} indicano rispettivamente media e deviazione standard dei tempi.

Per determinare la costante elastica della molla, abbiamo effettuato una regressione lineare (stavolta pesata) sui quadrati dei valori medi dei tempi (T_i^2, con $\delta T_i^2 = 5 \cdot 10^{-3} (20T_i)_{\text{best}} \delta(20T_i)\ )^2$ rispetto alla massa $(m_{\text{eff}})_i$, facendo riferimento alla relazione $T_i^2 = \frac{4\pi^2}{k} (m_{\text{eff}})_i$. Allora, detto b il coefficiente angolare della retta di regressione, varrà:

$$k_{\text{best}} = \frac{4\pi^2}{b_{\text{best}}} \qquad \wedge \qquad \frac{\delta k}{k_{\text{best}}} = \frac{\delta b}{b_{\text{best}}}$$

Si noti che, anche in questo caso, l'intercetta a della retta dev'essere compatibile con 0.

Di seguito è riportata la retta di regressione, assieme ai risultati ottenuti:

Figura 4: La retta di regressione (in rosso) e la sua regione di incertezza (in rosa).

- $a = (0.02 \pm 0.19)\,\text{cm}$ (compatibile con 0)
- $b = (4.604 \pm 0.002) \cdot 10^{-4}\ \text{s}^2/\text{g} = (46.04 \pm 0.02) \cdot 10^{-2}\ \text{s}^2/\text{kg}$
- $k = (85.74 \pm 0.04)\,\text{N/m}$

3 Conclusioni

Per valutare numericamente la consistenza tra i due valori di k ottenuti, abbiamo calcolato il seguente valore (numero puro):

$$\varepsilon = \frac{|(k_{\text{statica}})_{\text{best}} - (k_{\text{dinamica}})_{\text{best}}|}{\delta k_{\text{statica}} + \delta k_{\text{dinamica}}}$$

Allora k_{statica} e k_{dinamica} sono consistenti se e solo se $\varepsilon \leq 1$.

Nel nostro caso, $\varepsilon = 1.33$. Il gruppo di lavoro ha ipotizzato che questa inconsistenza (comunque contenuta, seppur non trascurabile) fra le due misure possa essere ragionevolmente giustificata dalla difficoltà incontrata nel ridurre al minimo le oscillazioni in direzione perpendicolare a $\vec{g}$; considerato inoltre che la posizione dei fototraguardi non era ottimale, ciò potrebbe avere ulteriormente influenzato la distribuzione dei tempi. È in effetti possibile osservare che le distribuzioni da noi ottenute non sono, il più delle volte, del tutto simmetriche: la moda sembra essersi spostata leggermente a sinistra – un possibile sintomo dell'influenza di un errore sistematico sulle misure.

[2] La formula per l'errore su T_i^2 segue direttamente dalla propagazione degli errori:

$$\frac{\delta T_i^2}{(T_i^2)_{\text{best}}} = 2\frac{\delta T_i}{(T_i)_{\text{best}}} \qquad \delta T_i^2 = 2\,(T_i)_{\text{best}}\,\delta T_i \qquad \delta T_i^2 = \frac{(20T_i)_{\text{best}}\,(\delta 20T_i)}{200}$$

da cui quanto riportato sopra. Si osservi che δT_i^2 dipende da $(20T_i)_{\text{best}}$: proprio questo è il motivo dietro alla scelta del metodo pesato per la regressione lineare.

Appendices

A Codice Rust per $5 \cdot 10^{12?}$ lanci di dadi

Qui riportiamo il codice Rust, da noi scritto, che ci ha permesso di lanciare virtualmente $5 \cdot 10^{12?}$ dadi in maniera estremamente efficiente.

```rust
/**** Architettura del programma ****
 * Il lavoro viene spezzato fra i vari
*/

/* Costanti durante la compilazione */
/// Parametri della simulazione
const N_THREADS:    u8  = 50;         /// Numero di thread
const N_MILESTONES: u32 = 10_000;     /// Numero di
const N_THROWS:     u32 = 2_000_000;
const N_ROLLS:      u8  = 5;
/// Altre costanti
const N_BINS: usize = (N_ROLLS + 1) as usize;
const N_COUNT_TOTAL: u128 = (N_MILESTONES as u128) * (N_THREADS as u128);

/*** Imports ***/
use std::{sync::mpsc::{self, Sender}, thread, io::{self, Write}};
use rand::prelude::Distribution;


fn work(tx: Sender<[u128; N_BINS]>) {
    /*** Setup ***/
    let mut rng = rand::thread_rng();
    let die = rand::distributions::Uniform::from(1..7u8);
    /*** Main program ***/
    let mut counter: u8;
    let mut bins: [u128; N_BINS];
    for _milestone in 0..N_MILESTONES {
        bins = [0; N_BINS];
        for _throw in 0..N_THROWS {
            counter = 0;
            for _roll in 0..N_ROLLS {
                // Roll a die
                if die.sample(&mut rng) == 1 {
                    counter += 1;   // Success!
                }
            }
            bins[counter as usize] += 1;
        }
```

```
39          tx.send(bins).unwrap();
40      }
41  }
42
43
44  fn print_info() {
45      println!("#dadi:   {N_ROLLS}");
46      println!("#lanci:  {N_THROWS}");
47      println!("#volte:  {N_MILESTONES}");
48      println!("#thread: {N_THREADS}");
49      let total: u128 = (N_THROWS as u128)*N_COUNT_TOTAL;
50      println!("#totale: {total}");
51  }
52
53  fn print_bins(count: u128, bins: [u128; N_BINS]) {
54      let percent: u128 = (100*count)/N_COUNT_TOTAL;
55      print!("[{count}/{N_COUNT_TOTAL}|{percent}%]");
56      for i in 0..N_BINS {
57          let bin = bins[i];
58          print!("\t{bin}");
59      }
60      print!("\n");
61      io::stdout().flush().unwrap();
62  }
63
64
65  fn main() {
66      print_info();
67      /*** Spawn threads ***/
68      let (tx, rx) = mpsc::channel::<[u128;N_BINS]>();
69      for _i in 0..N_THREADS {
70          let txi = tx.clone();
71          thread::spawn(move ||{ work(txi); });
72      }
73      /*** Combine all outputs ***/
74      let mut count: u128 = 0;
75      let mut bins: [u128; N_BINS] = [0; N_BINS];
76      for bin in rx {
77          for i in 0..N_BINS {
78              bins[i] += bin[i];
79          }
80          count += 1;  // another thread has finished!
81          if (count % (N_THREADS as u128)) == 0 {
82              print_bins(count, bins);
83          }
84          if count == N_COUNT_TOTAL {
```

```
85                 break;
86             }
87         }
88     }
```