Laboratorio di Calcolo per Fisici, Ottava esercitazione

Canale Lp-P, Docente: Cristiano De Michele

Lo scopo dell'ottava esercitazione di laboratorio è di fare pratica con le istruzioni di input/output da file e la scrittura modulare di programmi tramite funzioni.

La distribuzione binomiale, o distribuzione di Bernoulli, è una distribuzione di probabilità discreta che descrive la probabilità che in n esperimenti si verifichi k volte un evento con probabilità individuale p. La distribuzione è data da:

$$P(k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}, \tag{1}$$

dove in questa formula $n!$ indica il *fattoriale* del numero n, che è definito come segue:

$$n! = 1 \times 2 \times 3 ... \cdot (n-1) \cdot n$$

Essendo la distribuzione di Bernoulli definita per un insieme discreto e finito di valori, si ha:

$$\sum_{k=0}^{n} P(k) = 1 \tag{2}$$

detta *condizione di normalizzazione*.
Esempi di processi descritti dalla distribuzione binomiale sono il lancio di una moneta ($p_{testa} = p_{croce} = \frac{1}{2}$), il lancio di n dadi a 6 facce, l'estrazione di palline o bossoli da un'urna, l'estrazione (ripetuta) di una carta da un mazzo di carte, etc. Consideriamo per esempio il lancio di n dadi a 6 facce. La probabilità p che esca il numero 3 su un dado sarà $p = \frac{1}{6}$ e la distribuzione binomiale descriverà la probabilità che in k dadi, degli n lanciati, sia uscita la faccia con il numero 3. Ad esempio, la probabilità che lanciando 3 dadi non si ottenga il numero 3 ($k = 0$) è pari a

$$P(k=0) = \frac{3!}{0!\,3!} \left(\frac{1}{6}\right)^0 \left(\frac{5}{6}\right)^3 = \frac{125}{216},$$

quella di ottenerlo su un solo dado ($k = 1$) è pari a

$$P(k=1) = \frac{3!}{1!\,2!} \left(\frac{1}{6}\right) \left(\frac{5}{6}\right)^2 = \frac{25}{72},$$

quella di ottenerlo sue due dadi ($k = 2$) è pari a

$$P(k=2) = \frac{3!}{2!\,1!} \left(\frac{1}{6}\right)^2 \left(\frac{5}{6}\right) = \frac{5}{72},$$

ed infine quella di ottenerlo su 3 dadi ($k = 3$) è pari a

$$P(k=3) = \frac{3!}{3!\,0!} \left(\frac{1}{6}\right)^3 \left(\frac{5}{6}\right)^0 = \frac{1}{216}.$$

Come si può verificare, la distribuzione appena scritta soddisfa la condizione di normalizzazione, si ha cioè: $\sum_{k=0}^{3} P(k) = 1$. L'istogramma della distribuzione, per questo caso, è mostrato in figura 1.
Il calcolo del fattoriale di un numero n in C si può implementare definendo un'opportuna funzione, ovvero:

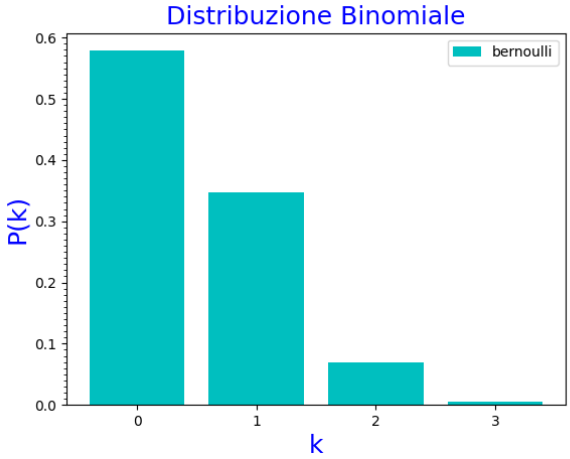

Figura 1: Distribuzione di Bernoulli per $p = 1/6$, $n = 3$.

```c
unsigned long int fattoriale(int n)
{
  // fattoriale
  unsigned long int i;
  unsigned long int p = 1;
  for (i = 1; i <= n; i++)
    {
      p *= i;
    }
  return p;
}
```

Si noti che abbiamo definito la variabile `fattoriale` come `unsigned long int` (intero senza segno a 8 bytes), perché il valore del fattoriale ($n!$) è sempre positivo e cresce molto rapidamente al crescere di n.
Come si può vedere dalla sua dichiarazione, la funzione `fattoriale`, restituisce un `unsigned long int` (ossia un intero senza segno a 8 bytes), e richiede coma argomento un intero. Ad esempio, per utilizzare tale funzione e assegnare il fattoriale del numero 4 alla variabile a di tipo `unsigned long int`, si dovrà usare la seguente istruzione C:

```c
  a = fattoriale(4);
```

► **Prima parte:**

1. Creare una cartella EX8 che conterrà il materiale dell'esercitazione.
2. Creare un programma `bernoulli.c` che
   (a) richieda l'immissione del numero totale di eventi n e la probabilità di un singolo evento p.

(b) calcoli la distribuzione di Bernoulli per valori arbitrari di n e p. Nel programma si dovranno definire sia la funzione `fattoriale()` (vedi sopra) sia una funziona chiamata `binomiale()` che calcolerà la probabilità di ottenere k volte un evento con probabilità p (si veda l'Eq. (1)). La funzione `binomiale()` dovrà prendere come argomenti i numeri interi k (numero di volte che si è verificato l'evento con probabilità p) e n (numero di eventi) ed il numero in virgola mobile p (probabilità del singolo evento). Si ricorda che in `C` l'elevamento a potenza si può calcolare utilizzando la funzione `pow( )` della libreria `math.h`. Ad es. per calcolare x^y si usa `pow(x,y)`.

(c) attraverso un ciclo su k (con $k \in [0, n]$) scriva su un file `bernoulli.dat` i valori di k e $P(k)$, su due colonne:

$$\begin{array}{ll} 0 & P(0) \\ 1 & P(1) \\ 2 & P(2) \\ \vdots & \quad \vdots \\ n & P(n) \end{array}$$

Suggerimento: Per la stampa della binomiale su di un file il cui nome deve dipendere dal numero n immesso in input, potete usare le seguenti istruzioni C per aprirlo in scrittura:

```
sprintf(fn,"bernoulli_%i.dat", n);
fp=fopen(fn,"w");
```

dove la funzione `sprintf()` (definita in `stdio.h`) *scrive* sulla stringa (array di caratteri) `fn` il nome del file in cui comparirà anche il valore della variabile intera fornita come argomento (in questo caso n). Successivamente, alla funzione `fopen` viene passata, invece che la solita stringa di testo con il nome del file, l'array `fn`. Nel caso ad esempio in cui $n = 4$, la funzione `fopen` aprirà un file chiamato `bernoulli_4.dat` in modalità scrittura. La lunghezza dell'array `fn` deve essere tale da poter contenere tutti i caratteri che compongono il nome del file compreso il carattere di fine stringa (80 in questo caso è una lunghezza più che sufficiente).

(d) verifichi che la $P(k)$ soddisfi la condizione di normalizzazione (vedi Eq. (2)).

3. Fissare $p = \frac{1}{6}$ e graficare con `python` la funzione $P(k)$ per i seguenti valori di n: $n = 2$; $n = 4$; $n = 10$; $n = 20$.
Suggerimento: ricordatevi che per graficare un istogramma potete usare la seguente istruzione python:

```
plt.bar(x, y, color='c',label='bernoulli')
```

4. Che cosa si nota all'aumentare di n? Perché? Salvate i dati usati per le figure su diversi file `bernoulli_n.dat` e i grafici su file immagine `bernoulli_n.png`.

► **Seconda parte:**

Creare un secondo programma `lanci.c` che simuli l'evento descritto dalla distribuzione di Bernoulli. In particolare, il programma deve simulare una serie di M lanci di n dadi a 6 facce, e registrare, in un

opportuno array di lunghezza pari a n, il numero di volte in cui la faccia numero 3 è comparsa su k dadi. Quindi tale array, che chiameremo `isto`, non è altro che un istogramma delle occorrenze degli eventi. Esso dovrà essere inizializzato a zero all'inizio del programma e dovrà essere opportunamente normalizzato a 1 alla fine dei lanci, dividendo ogni suo elemento per M. Per crearlo basta fare quanto segue: ad ogni lancio si contano il numero di dadi per i quali è uscito il numero 3, se tali dadi sono k, allora si dovrà incrementare di 1 l'elemento k-esimo dell'array (cioè `isto[k]`). Una volta eseguiti tutti i lanci, `isto[0]` conterrà un valore pari al numero di volte in cui non è uscito su alcun dado il numero 3, `isto[1]` conterrà un valore pari al numero di volte che è uscito il numero 3 su un solo dado e così via. Notare infine che questo istogramma tenderà alla distribuzione di Bernoulli, se il numero dei lanci M è sufficientemente grande. Nello specifico, il programma `lanci.c` deve:

1. Chiedere all'utente di inserire il numero n di dadi e il numero di lanci M;
2. Chiamare n volte, per ognuno degli M lanci, una funzione `lancia_dado()` che, tramite l'utilizzo di un generatore di numeri casuali, simuli il lancio di un dado a 6 facce (facendo in questo modo sarà come se avessimo tirato n dadi ad ogni lancio).
3. Costruire l'istogramma delle occorrenze `isto[]`.
4. Normalizzare l'istogramma come discusso in precedenza.
5. Salvare su di un file chiamato `istogramma.dat` l'istogramma normalizzato e graficare poi tale istogramma con `python` salvando il grafico nel file `istrogramma.png`.

► **Terza parte** Confrontare (graficamente) i dati generati dai due programmi per gli stessi valori di n e k. Qual è il numero minimo di lanci da usare nel secondo programma per ottenere un campionamento ragionevole della distribuzione di Bernoulli? Se si esegue il secondo programma più volte per uno stesso numero di lanci M, come cambia la distribuzione? Che cosa si può fare per diminuire le *fluttuazioni* nel risultato? Scrivete le vostre osservazioni e risposte su un file `risposte.txt`.