Laboraratorio di Calcolo. Prova d'esame 14/02/2019

La prova ci incentra su alcuni studi e simulazioni al computer della funzione

$$f(x) = x^2\, e^{-x} + e^{-\frac{(x-1)^2}{2}} \qquad (x \geq 0)\,. \tag{1}$$

della quale ci interessiamo ai soli non negativi della x.

1. Cominciamo con uno script in R per avere un'idea del comportamento della funzione nell'intervallo di interesse.

   (a) Si definiscano le funzioni f1(), f2() e f(), ove l'ultima è l'Eq. (1), mentre le prime due sono i due termini da cui essa è costituita; nel codice R f() deve richiamare le altre due.

   (b) Si plotti f() nell'intervallo $[0, 10]$ e, sullo stesso plot ma con altri colori, f1() e f2().
   (Nota: si presti attenzione che il plot contenga lo zero delle ordinate.)

   (c) Usando l'apposita funzione di R si clicchi sul plot nel punto dove $f(x)$ mostra il massimo in modo che R 'legga' il valore delle coordinate; riportare quindi, mediante copia/incolla, i valori delle coordinate ottenute in un opportuno comando di stampa in modo che compaiano al docente che correggerà lo script senza che questi debba cliccare sul plot.

   (d) Plottare nuovamente $f(x)$ (soltanto) facendo sempre attenzione che lo zero delle ordinate compaia nel plot. Quindi,

   - si estraggano 'a caso' 100000 punti nel rettangolo che ha per base l'intervallo $[0, 10]$ e un'altezza *leggermente maggiore* del massimo di $f(x)$ in tale intervallo, e li aggiungiamo al plot.
   - si aggiungano ancora, con altro colore, i punti 'sotto' la curva $f(x)$.

   (e) Si faccia infine l'istogramma con 100 bin ('intervallini') dei soli punti 'sotto' la curva $f(x)$.

   (Nota: Fra un plot e l'altro si dia un comando opportuno in modo che eseguendo lo script si abbia il tempo di esaminare un plot alla volta.)

   ⇒ `esplorazione_f.R`

2. In un altro script R si calcoli quindi l'integrale $\int_0^{10} f(x)\, dx$ in due modi

   (a) mediante l'apposita funzione di R;

   (b) mediante campionamento, estraendo 100000 punti in un rettangolo opportuno (metodo 'hit/miss').

   ⇒ `integrale.R`

3. Implementiamo quindi la funzione dell'Eq. (1) in C mediante una function che chiameremo banalmente `f()`, a variabili `double` (non c'è nessun bisogno di definire separatamente `f1()` e `f2()`, le quali ci servivano solo per avere un'idea, mediante la grafica, dei due contributi).

   Per testarla calcoliamo e stampiamo i valori che essa assume per x che va da 0 a 10 a passi di 1.

   $\Rightarrow$ `check_f.c`

4. A questo punto possiamo passare a calcolare l'integrale per campionamento in C.

   (a) Si scriva innanzitutto la funzione `runif()` la quale a ogni chiamata 'ritorni' un *numero random uniforme* 'continuo' in un intervallo i cui estremi sono passati come parametro.

   (b) Si scriva quindi la funzione `intHM()`, per calcolare l'integrale di $f(x)$ fra a e b per campionamento ('Hit/Miss'). Alla funzione vanno passati anche l'altezza h del rettangolo di riferimento (vedi quanto imparato negli script R) e il numero n di punti da estrarre in tale rettagolo. La funzione estrae i punti nel rettangolo mediante opportune chiamate a `runif()` e conta il numero di punti 'sotto $f(x)$'. Alla fine 'ritorna' il valore dell'integrale stimato con questo metodo.

   (c) Nel main si diano, *preferibilmente* 'a riga di comando' (oppure su richiesta del programma stesso), a, b, h e n, controllando che i valori siano 'ragionevoli' ($0 \leq a < b$; $h > 0$; e ovviamente $n > 0$).
   Quindi l'integrale viene calcolato e stampato. In particolare, si calcoli l'integrale fra 0 e 10, per controllare che tutto funzioni, avendolo già calcolato in R.

   Nota: per il *seed* del generatore random uniforme della libreria C si usi il proprio numero di matricola (idem nel seguito).

   $\Rightarrow$ `integrale.c`

5. Siamo ora pronti a generare, sempre mediante hit/miss, dei numeri casuali la cui probabilità è proporzionale a $f(x)$, ovvero tali che, se ne facciamo l'istogramma delle frequenze, questo verrà simile a quello visto in `esplorazione_f.R`.

   (a) Sulla falsariga di `intHM()` si scriva la funzione `rf()` la quale esegua il campionamento nel solito rettangolo e 'ritorni' nel vettore `rx[]`, definito nel main, soltanto i valori delle ascisse dei punti 'sotto $f(x)$'. Va anche 'ritornato' in maniera opportuna il numero di valori validi

contenuti in `rx[]` (come sappiamo il metodo hit/miss è intrinsecamente inefficiente: quindi `rx[]` va dimensionato nel main con il massimo numero di valori che potrebbe contenere).

(b) Come nel punto precedente, vengono dati al main i parametri necessari, quindi viene chiamata `rf()`.
Dopo la chiamata il main ha a disposizione il vettore `rx[]` e il numero di valori validi, che indichiamo con `nrx`. Da `nrx` e `n`, insieme all'area del rettangolo in cui è avvenuta l'estrazione ci possiamo ricalcolare l'integrale fra 0 e 10 di $f(x)$, che possiamo confrontare con quanto già sappiamo.

⇒ `generazione_rx.c`

6. Proseguiamo quindi il lavoro scrivendo un altro programma che è la continuazione di quello precedente
⇒ `binning_rx.c`
Per prima cosa copiamo `generazione_rx.c` in `binning_rx.c`, in modo da conservare il vecchio programma. Quindi:

   (a) aggiungere la funzione `binning()` alla quale si passano i risultati ottenuti da `rf()` (ovvero `rx[]` e `nrx`) e i limiti a e b del campionamento. A essa andrà passato anche il vettore `bins[100]`, che conterrà i risultati del 'binning', ovvero il numero di volte che i valori di `rx` apparterranno a un particolare intervallino ('bin'). Quindi
      - `bins[0]` conterrà il numero di valori `rx` compresi in $[a, a + \Delta x]$, con $\Delta x = (b - a)/100$;
      - `bins[1]` conterrà il numero di valori di `rx` compresi in $[a + \Delta x, a + 2\,\Delta x]$;
      - e così via ...

      In realtà, fare per ogni elemento di `rx[]` un *loop* sui 100 possibili bin consuma molto tempo di calcolo. È molto più efficiente calcolare direttamente l'indice del bin come rapporto (arrotondato all'intero più basso) fra la distanza $x - a$ e l'ampiezza Δx dei bin.

   (b) Dopo aver chiamato `rf()` il main chiama `binning()`. Dopo la chiamata il main ha a disposizione anche `bins[]` riempito da `binning()`.

   (c) I valori risultanti di `bins[]` vengono visualizzati sul monitor, insieme all'indice del bin e valore centrale della x nel bin (nell'ordine: indice del bin, centro bin, conteggi nel bin).

   (d) Essi sono anche scritti, nello stesso ordine, ma senza didascalie, in formato testo nel file `bins.txt` (tre valori per riga).

7. L'ultimo punto consiste nel rileggere `bins.txt` da uno script R al fine di effettuarne una rappresentazione grafica mediante un 'plot a barre', usando come 'nomi' delle barre i valori dei centri dei bin.
⇒ `mostra_bins.R`