Laboratorio di Calcolo, Esercitazione 6, 21-22 novembre 2024

Canale Pet-Z, Docenti: Shahram Rahatlou, Sibilla Di Pace

Stima di π con l'ago di Buffon: Lo scopo di questa esercitazione è di stimare il valore di π con un metodo iterativo noto dal XVIII secolo che utilizza l'estrazione dei numeri casuali tramite le funzioni `srand48()` es `lrand48()` .

Supponiamo di avere un piano percorso da linee parallele distanti d tra di loro e un ago di lunghezza L con $L < d$. Lanciando l'ago sul piano, essa ha una probabilità $2L/\pi d$ di incrociare una linea del piano. Sia x la distanza tra il centro dell'ago e la linea piú vicina allago e θ l'angolo acuto tra l'ago e le linee. L'ago incrocerà una delle linee se è verificata la condizione $x < (L/2)\sin\theta$. Effettuando N_{lanci} lanci e indicando con S il numero di volte che l'ago incrocia una linea si ha che $S/N_{\text{lanci}} = 2L/(\pi d)$ da cui possiamo ottenere la stima $\pi = 2LN_{\text{lanci}}/(Sd)$.
In questa prova vogliamo ripetere la stima $N_{\text{exp}} = 1000$ di volte, ripetendo ogni volta procedura descritta sopra.

► Cartella di lavoro

Fare login sulla postazione utilizzando le credenziali user-id `studente` e password `informatica`. Creare una cartella `LCSR6` nella *home directory* con il comando `mkdir` in cui scriverete i programmi di oggi. Tutti i file di codice sorgente in C e in python dovranno trovarsi in questa cartella per essere visualizzati. **Le cartelle create sulla scrivania (Desktop) o in altre sotto-cartelle non verranno copiate né valutate.**

► Stima di π

Creare un programma `buffon-NNN.c`, dove `NNN` è il vostro numero di gruppo, ad esempio `098`, nella cartella `LCSR6` utilizzando l'editor di testo `emacs`, per eseguire le seguenti operazioni:

1. acquisire dall'utente il valore delle variabili L, d, N_{lanci} e verificarne individualmente la validità. Informare l'utente che i valori di L e d devono essere in cm e minori di 5 cm, oltre che $L < d$, e $N_{\text{lanci}} < 10000$;

2. creare un array `pigreco` di lunghezza N_{exp} per immagazzinare i valori di π stimati in ciascuna prova;

3. simulare N_{lanci} lanci dell'ago dove ciascun lancio consiste in

   - generare un valore casuale di θ compreso tra 0 e $\pi/2$ (usare `M_PI` della libreria matematica);
   - generare un valore casuale di x compreso tra 0 e $d/2$.

4. per ciascun lancio determinare se l'ago incrocia o meno una linea;

5. contare il numero S di lanci in cui l'ago ha incrociato una linea e calcolare il valore di π utilizzando l'espressione fornita ed immagazzinarlo nell'array `pigreco`;

6. ripetere i passi 3-7 per N_{exp} volte;

7. determinare il minimo π_{min}, il massimo π_{max}, e il valore medio $\langle\pi\rangle$ dei valori stimati, e stamparli sullo schermo con 12 cifre decimali; si ricorda che il valore medio di N misure $\{x_1, ..., x_N\}$ è definito come $\langle x\rangle = \sum_{j=1}^{N} x_j/N$;

8. scrivere in un file `pigreco.txt` il numero dell'esperimento e la relativa stima di π (due valori per riga) con il formato `%-4d \t %.12lf\n`

9. scrivere i valori immagazzinati nell'array `pigreco` in un file chiamato `pigreco.txt`;

10. creare un istogramma dei valori scritti nel file utilizzando il codice in python.

Si ricorda che per creare l'eseguibile utilizzando la libreria matematica dovete usare il comando `gcc -Wall -o app.exe programma.c -lm` dalla riga di comando nella shell.

Si consiglia di scrivere il programma in modo incrementale, verificando la corretta compilazione e l'esecuzione almeno dopo ciascuno dei passi indicati nel testo.

► Opzionale
Oltre alla media $\langle\pi\rangle$ potete calcolare e stampare sullo schermo l'incertezza $\delta\pi$ su questa media come $\delta\pi = \sigma/\sqrt{N_{\rm exp}}$, dove σ è la deviazione standard dei valori ottenuti.

Per N misure $\{x_1, ..., x_N\}$ la deviazione standard è definita come $\sigma^2 = \sum_{i=1}^{N}(x_i - \langle x\rangle)^2/(N-1) = \langle x^2\rangle - \langle x\rangle$, con $\langle x\rangle = \sum_{j=1}^{N} x_j/N$ e $\langle x^2\rangle = \sum_{j=1}^{N} x_j^2/N$

► Scrittura su file
Vi ricordo brevemente le istruzioni in C per la scrittura di dati su un file in formato testo.

Prima di tutto va aperto un file in scrittura sul disco specificando il nome del file

```
1  FILE* fp;
2  fp = fopen("nome_del_file", "w");
```

Listato 1: Apertura del file in scrittura

Poi potete scrivere i dati d'interesse, ad esempio x e y specificando i descrittori opportuni. La funzione `fprintf()` usa gli stessi formattatori della funzione `printf()`

```
1      fprintf(fp, "%lf %lf\n", x, y);
```

Listato 2: Scrittura di dati sul file

Infine, ricordatevi di chiudere il file prima di terminare il programma

```
1  fclose(fp);
```

Listato 3: Chiusura del file

▶ **Istogramma con python**

Vogliamo graficare l'andamento delle stime in funzione del numero di esperimenti e la distribuzione dei valori ottenuti con il seguente codice che deve essere chiamato `istogramma-NNN.py` e salvato nella cartella `LCSR6`.

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

# carica dati dal file
# exp sono numero di esperimenti nella prima colonna
# pi sono le stime nella seconda colonna
exp, pi = np.loadtxt('pigreco.dat', unpack=True)

# crea grafica di pi stimato in funzione del numero di esperimento
plt.title("Andamento di $\pi$ in funzione di esperimento")
plt.plot( exp, pi, '.', label='stima di $\pi$ con ago di buffon')

# specifica limite inferiore superiore per asse x e y con numeri
   opportuni in base ai vostri dati
plt.xlim(0, 1000)
plt.ylim(2., 4.)
# aggiungi legenda per gli assi
plt.xlabel('#esperimento')
plt.ylabel('$\pi$ stimato')
# linea orizzontale rossa al valore di pi greco usando la libreria
   matematica
plt.axhline( y=m.pi, color = 'red', linestyle='--')
# nome del file in cui salvare il grafico
plt.savefig("pi.png")
# mostra grafico
plt.show()

####  istogramma della distribuzione di pi greco
plt.title("Distribuzione delle stime di $\pi$ greco")

plt.hist(pi, color='green')
plt.xlabel("$\pi$ stimato")
plt.ylabel("# esperimenti")

plt.axvline( x=m.pi, color = 'blue', linestyle='-')

plt.show()
```

Listato 4: Programma `istogramma.py`