Metodi Computazionali della Fisica
Secondo Modulo: C++

Seconda Lezione

La lezione di oggi

Obiettivo:

- implementare un generatore di numeri casuali;
- implementare una classe `Data`;
- verificare l'equivalenza tra propagazione dell'errore *standard* e quella con tecniche Monte Carlo.

Contenuti:

- Linear Congruential Generators;
- teoremi della probabilità.

Strumenti:

- la routine `RANO`.

Numeri casuali

- I numeri casuali *fisici* sono generati da processi fisici *realmente* casuali, quali, ad esempio, decadimenti radioattivi, rumore termico, la roulette, ...
- La necessità di un generatore *realmente* casuale (ovvero non *algoritmico*) è fondamentale nel caso nel campo informatico (si veda, ad esempio, l'uso della crittografia per le connessioni sicure tipo `ssh`).

Numeri pseudo-casuali

Caratteristiche:

- Ripetibilità: lo stesso valore iniziale (*seed*) deve produrre lo stessa sequenza di numeri casuali (fondamentale per il debugging).
- Casualità: i numeri casuali devono essere:
  - uniformemente distribuiti - ad esempio, in $[0, 1)$;
  - non correlati (per quanto possibile).
- Lunga periodicità: qualunque generatore dopo un certo numero di chiamate produce le stesse sequenze.
- Indipendenza: periodo e casualità non devono dipendere dal seed iniziale.
- Velocità.
- Portabilità: gli stessi risultati devono essere prodotti su diverse macchine.

Linear Congruential Generators (LCG)

Sequenze di interi tra 0 ed $m-1$ possono essere generate con la seguente relazione:

$$I_{i+1} = (aI_i + c) \mathrm{mod} m$$

dove a, c ed m sono costanti intere ed m è la periodicità del generatore.
Numeri reali $[0, 1)$ si ottengono dividendo per m: $F_i = I_i/m$.
Le costanti a, c ed m non sono scelte a caso. Possibili valori sono:

a	c	m
106	1283	6075
9301	49297	233280
16807	0	2147483647
1103515245	12345	2147483648

Esercizio 1: Implementare l'algoritmo dell'LCG

Teoremi fondamentali della probabilità

- Legge grandi numeri: data una variabile casuale X con valore di aspettazione finito μ, al crescere del numero delle osservazioni x_i la loro media aritmetica tende al valore di aspettazione

$$\lim_{N\to\infty} \frac{x_1 + \cdots + x_N}{N} = \mu$$

- Teorema del Limite Centrale: data una variabile casuale X con valore di aspettazione finito μ e varianza σ^2, al crescere del numero delle osservazioni la loro distribuzione tende ad una gaussiana con media μ e varianza σ^2/N.

Somma di due misure

Nel caso in cui si abbiano due misure indipendenti x_1 e x_2 distribuite secondo una guassiana:

$$P(x_1) \propto \exp\left(\frac{-x_1^2}{2\sigma_1^2}\right) \qquad P(x_2) \propto \exp\left(\frac{-x_2^2}{2\sigma_2^2}\right)$$

la distribuzione di probabilità della loro somma è data da

$$\begin{aligned} P(x_1, x_2) = P(x_1)P(x_2) &\propto \exp\left[-\frac{1}{2}\left(\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2}\right)\right] \\ &\propto \exp\left[-\frac{1}{2}\frac{(x_1+x_2)^2}{\sigma_1^2+\sigma_2^2}\right], \end{aligned}$$

da cui la regola di somma in quadratura degli errori

$$\sigma_{1+2}^2 = \sigma_1^2 + \sigma_2^2 \,.$$

Esercizio 2

Implementare la classe definita dal seguente header file

```
#ifndef _DATA_HPP
#define _DATA_HPP

#include <iostream>
#include <cmath>
#include <vector>
#include <cstdlib>

enum {NODIST,FLAT,GAUSS};

using namespace std;

class Data {

public:

  Data();
  Data(double,vector<double>);
  Data(const Data&);
  Data(const Data&,size_t);

  ~Data(){};

  double get_mean(){return _mean;};
  double get_sigmatot(){
    double tmp=0.;
    for(size_t i=0;i<_sigma.size();i++)tmp+=_sigma[i]*_sigma[i];
    return sqrt(tmp);
  };

private:

  double _mean;
  vector<double> _sigma;

};

#endif
```

Esercizio 3

- Utilizzare la classe `Data` per generare coppie di dati artificiali.
- Verificare che nel limite di grande N:
  - media ed errore della somma di N coppie di dati equivalgono a quelle che si ottengono dalla somma in quadratura degli errori;
  - il risultato diventa indipendente dal seed iniziale.
- Visuallizare con `gnuplot` la distribuzione gaussiana e confrontarla con quelle ottenute con $N = 10$, 100, 1000 e 10000.