

# Monte Carlo

Laboratorio di Metodi Computazionali e Statistici (2023/2024)

8 Novembre 2023

# Metodo di Monte Carlo

Il termine **metodo di Monte Carlo** si riferisce a qualsiasi metodo numerico che faccia uso di numeri “casuali” (random) per risolvere probabilisticamente un problema.

Metodi di Monte Carlo sono normalmente utilizzati in ambito scientifico per:

- simulare processi stocastici
- simulare la risposta di apparati sperimentali
- calcoli numerici approssimati (integrali, etc..)

Ma è anche molto sfruttato in molti altri ambiti: previsioni meteorologiche, chimica molecolare, elettronica, dinamica dei fluidi, biologia, computer grafica, intelligenza artificiale, finanza, valutazione di progetti.

## Breve nota storica

Le origini del Metodo di Monte Carlo risalgono alla metà degli anni 40 nell'ambito del Progetto Manhattan.

I formalizzatori del metodo sono Enrico Fermi, John von Neumann e Stanislaw Marcin Ulam, il nome Monte Carlo fu inventato in seguito da Nicholas Constantine Metropolis in riferimento alla nota tradizione nei giochi d'azzardo dello stato omonimo nel sud della Francia (e in particolare alla roulette in cui i numeri sono estratti a caso).

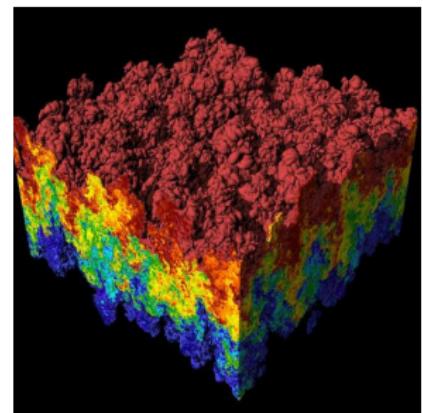
Pare che l'idea venne a Ulam mentre era convalescente da una malattia. Per passare il tempo giocava a un solitario con le carte e si chiese quale fosse la probabilità di completarlo correttamente. Le regole del solitario rendevano difficile il calcolo di questa probabilità. Tuttavia pensò che si sarebbero potute simulare con un computer molte disposizioni casuali di un mazzo di carte e controllare in quanti di questi casi si riusciva a completare il solitario. In questo modo si poteva calcolare la probabilità di vincere al solitario in modo empirico.

Ulam ne parlò con Von Neumann il quale comprese il potenziale di questa intuizione e sviluppò il primo algoritmo che permetteva di generare numeri casuali con un computer.

# Monte Carlo in Fisica

In fisica teorica della materia

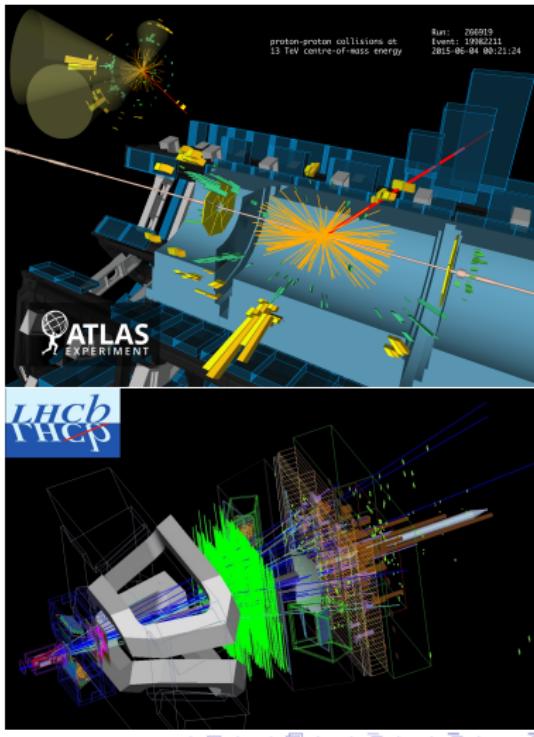
- Calcolo di integrali a N dimensioni (con N grande, meccanica statistica)
- Simulazione di processi stocastici a molti corpi (Random Walk, etc..)



# Monte Carlo in Fisica

In fisica nucleare e sub-nucleare.

- Simulazione del processo iniziale (fisica quantistica-relativistica)
- Simulazione dell'interazione con il rivelatore
- Produzione di dati simulati formalmente identici a quelli "veri" utili per progettare, ottimizzare, controllare le prestazioni rivelatore



# Numeri casuali

I metodi di MC richiedono la generazione dei valori (numeri casuali) di variabili aleatorie di cui sono note le distribuzioni di probabilità.

Un **sequenza di numeri casuali** è una sequenza di numeri che non hanno alcuna relazione di successione tra di loro (ma che seguono, tutti, una stessa distribuzione di probabilità).

Come “generare” sequenze di numeri casuali ?

# Numeri pseudo-casuali

I numeri casuali reali non possono essere generati:

- la rappresentazione floating-point fornisce, come sappiamo, un'approssimazione dei numeri reali      bit definiti per indicare l'numero
- una sequenza veramente casuale non può essere facilmente generata (richiederebbe una memoria infinita)

I generatori di numeri pseudo-casuali sono algoritmi che producono una sequenza deterministica di numeri che soddisfano alcune proprietà delle sequenze casuali. In particolare, si desidera generare una sequenza  $x_n$  che è:

- uniformemente distribuita in  $[0,1]$ ;
- gli elementi della sequenza non sono correlati (ad esempio l'assenza di correlazione significa che  $(x_n, x_{n+1})$  non presenta "pattern" particolari);
- periodo di ripetizione molto lungo;
- devono essere calcolati rapidamente.

# Generatori di numeri casuali

Esempio (Middle Square, Von Neumann, 1946):

Dato un numero intero di 10 cifre (seme della sequenza) lo si eleva al quadrato e si prendono le 10 cifre centrali come numero successivo:

$$4423177834 = 19564 \underbrace{5021511889}_{\text{elemento } n} \underbrace{31556}_{\text{elemento } n+1}$$

A volte stesso  
numero ripetuto  $\infty$   
A volte periodo molto  
breve

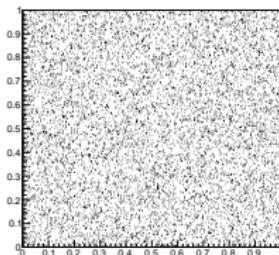
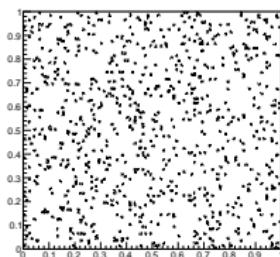
Linear Congruential Generator (LCG)

$$x_{n+1} = (ax_n + c) \bmod m; \quad \begin{array}{c} \text{resto} \\ \text{intero} \\ \text{divisione} \\ \text{per } m \end{array} \quad n \geq 0 \quad \text{con } a, c, m \in \mathbb{N} \quad \begin{array}{l} \text{m da il} \\ \text{periodo} \end{array}$$

$\bmod$  = resto intero della divisione

Siccome  $m$  è, al più il periodo della sequenza, deve essere grande ( $x_{n+1}$  assume valori interi tra 0 e  $m-1$ )

$$a=1101, c=0, m=2^{30}-1 \quad a=7^5, c=0, m=2^{31}-1$$



# Generatori di numeri casuali moderni

*Shift di bit e spostamento di registri*

Nel 1997 ci fu un notevole balzo in avanti negli algoritmi di generazione deterministica dei numeri.

Generatori moderni solitamente combinazioni di operazioni di spostamenti di registri e manipolazioni su bit Algoritmi veloci e con periodi estremamente lunghi

La proposta di un nuovo metodo detto Mersenne twister, ad opera di Makoto Matsumoto e Takuji Nishimura (utilizza i numeri primi di Mersenne  $M_p = 2^p - 1$ ):

- periodo di  $2^{19937} - 1$

# Generatori di numeri casuali moderni nei vari linguaggi

## C++

```
#include <random>
linear_congruential_engine
mersenne_twister_engine
subtract_with_carry_engine
```

## Python

random

(MT algo)

## Matlab

→ <sup>imposta</sup>  
seme e algoritmo randomizzazione  
rng  
rand

(MT default)

# Generatori di numeri casuali in ROOT

## Generatori in ROOT

- [TRandom1](#), metodo RANLUX (utilizza una serie modificata della sequenza di Fibonacci), periodo relativamente lungo:  $10^{171}$  (piuttosto lento 242 ns/chiamata)
- [TRandom2](#), metodo Tausworthe (che utilizza uno shift dei registri), periodo relativamente corto  $10^{26}$  ma molto veloce (adatto ad applicazioni veloci, che non necessitano periodi lunghi)
- [TRandom3](#), consigliato: metodo MT, periodo:  $10^{6000}$  (leggermente meno veloce di TRandom2 (45ns/chiamata) ma con periodo lunghissimo)
- Altre opzioni disponibili in [ROOT::Math::MixMaxEngine](#), [ROOT::Math::StdEngine](#)

# Generatori di numeri casuali moderni: modalità di utilizzo

In generale si divide l'utilizzo del generatore in due passi

- definizione del “seme” della sequenza in modo da poter generare sequenze sempre diverse o esattamente la stessa sequenza (per scopi di debugging): a seme uguale corrisponde sequenza uguale. *di numeri casuali*
- generazione del singolo valore

Ricordatevi che la definizione del “seme” va fatta una volta nell’algoritmo mentre la generazione deve essere fatta per N volte dove N è il numero di valori che vogliamo generare

# Esempi

Macro ROOT

C++

```
TRandom3 rnd;           // creo oggetto
rnd.SetSeed(121356);   // inizializzo la sequenza
// oppure rnd.SetSeed(time(0))
// per avere una sequenza sempre diversa
// (time(0) fornisce il numero di secondi
// trascorsi dal 1/1/1970)

rnd.Rndm();           // estrae numero tra 0 e 1
```

Python

```
from ROOT import *
import math

r = TRandom3()
r.SetSeed(121356)
r.Rndm() #estrae numero tra 0 e 1
```

## Esempi (II)

Python

```
import random as rnd #importo libreria random  
rnd.seed(121356) #inizializza la sequenza  
  
rnd.random() #estrae numero tra 0 e 1
```

Matlab

```
rng(0, 'twister') %analogo a rng('default')  
%setta il generatore al  
%seed di default (=0) e  
%algoritmo Mersenne Twister  
rand() %estrae numero tra 0 e 1
```

# Generatore “base”

Da questo momento in poi assumeremo di avere a disposizione un generatore “base” che produce valori distribuiti uniformemente tra 0 e 1

$$\eta \in [0, 1)$$

$$p(\eta) = 1$$

$$\int_0^1 p(\eta) d\eta = 1$$

# La distribuzione binomiale

Una interessante applicazione delle generazioni di distribuzioni uniformi è la generazione di processi binomiali.

Tale distribuzione si basa su eventi ([prove](#)) caratterizzati da due soli risultati possibili  $a$  (stato 1, successo) o  $\bar{a}$  (stato 2, insuccesso):

$$\begin{aligned} p(a) &= p \\ p(\bar{a}) &= q = 1 - p \end{aligned}$$

# La distribuzione binomiale

Una interessante applicazione delle generazioni di distribuzioni uniformi è la generazione di processi binomiali.

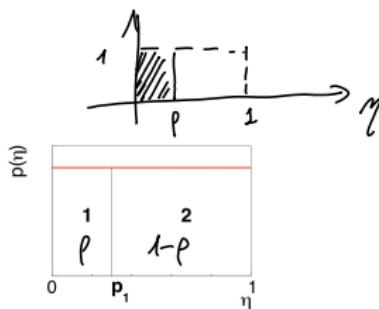
Tale distribuzione si basa su eventi (**prove**) caratterizzati da due soli risultati possibili  $a$  (stato 1, successo) o  $\bar{a}$  (stato 2, insuccesso):

$$\begin{aligned} p(a) &= p \\ p(\bar{a}) &= q = 1 - p \end{aligned}$$

Questo tipo di evento è facile da generare !

Estraggo  $\eta \in [0, 1]$  se  $\eta < p$  allora il risultato è  $a$  (stato 1, successo) altrimenti  $\bar{a}$  (stato 2, insuccesso).

Applicazione tipica: simulazione di un apparato con efficienza di selezione nota.



# La distribuzione multinomiale

La distribuzione multinomiale generalizza la distribuzione binomiale

Se la distribuzione binomiale si basa su eventi (prove) caratterizzati da due soli risultati possibili  $a$  o  $\bar{a}$ , la distribuzione multinomiale descrive il caso più generale in cui gli eventi sono caratterizzati da un numero finito di risultati, ognuno con la propria probabilità  $p_1, p_2, p_3, \dots, p_n$       n risultati con prob. associata

# La distribuzione multinomiale

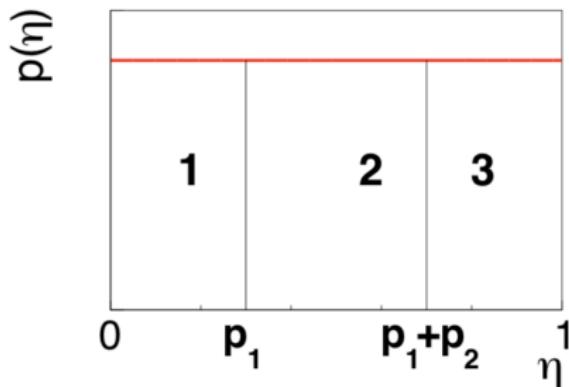
La distribuzione multinomiale generalizza la distribuzione binomiale

Se la distribuzione binomiale si basa su eventi (prove) caratterizzati da due soli risultati possibili  $a$  o  $\bar{a}$ , la distribuzione multinomiale descrive il caso più generale in cui gli eventi sono caratterizzati da un numero finito di risultati, ognuno con la propria probabilità  $p_1, p_2, p_3, \dots, p_n$

$n$  è casuale e rappresenta la probabilità

Estraggo  $\eta \in [0, 1]$ :

- se  $\eta < p_1$  si considera l'evento di tipo 1
- se  $p_1 < \eta < p_1 + p_2$  si considera l'evento di tipo 2
- se  $p_1 + p_2 < \eta < p_1 + p_2 + p_3$  si considera l'evento di tipo 3
- ...



# Esempio: Python

```
1 from ROOT import *
2 import time
3
4 rnd = TRandom3()
5 rnd.SetSeed(int(time.time()))
6
7 ntot = 100000
8 nacc1 = 0.0
9 nacc2 = 0.0
10 nacc3 = 0.0
11 p1 = 0.3
12 p2 = 0.2
13 for i in range(0,ntot):
14     xx = rnd.Rndm()
15     if xx<p1:
16         nacc1 = nacc1 + 1
17     elif xx>p1 and xx<=p1+p2:
18         nacc2 = nacc2 + 1
19     elif xx>p1+p2:
20         nacc3 = nacc3 + 1
21
22 print(nacc1/ntot)
23 print(nacc2/ntot)
24 print(nacc3/ntot)
```

# Esempio: C++

```
1 #include<time.h>
2 #include<iostream>
3 #include<cmath>
4 #include<TRandom3.h>
5
6 using namespace std;
7
8 int main(){
9     TRandom3 rnd;
10    rnd.SetSeed(int(time(0)));
11    int ntot = 100000;
12    int nacc1 = 0, nacc2 = 0, nacc3 = 0;
13    double p1 = 0.3, p2 = 0.2, xx;
14
15    for (int i=0;i<ntot;i++){
16        xx = rnd.Rndm();
17        if(xx<p1){
18            nacc1 = nacc1 + 1;
19        } else if(xx>p1 && xx<=p1+p2){
20            nacc2 = nacc2 + 1;
21        } else if(xx>p1+p2){
22            nacc3 = nacc3 + 1;
23        }
24    }
25
26    cout<<(double)nacc1/ntot<<" "<<(double)nacc2/ntot<<" ";
27    cout<<(double)nacc3/ntot<<endl;
```

# Generazione di distribuzione di probabilità arbitrarie

Abbiamo visto come generare eventi con distribuzione di probabilità uniforme, ed abbiamo anche visto in quale contesto tali eventi sono utili.

Tuttavia la maggior parte dei problemi di statistica applicati alla fisica richiedono eventi con **distribuzioni di probabilità non uniformi**.

È quindi importante imparare a **generare eventi distribuiti secondo una generica densità di probabilità** a partire da eventi distribuiti uniformemente.

Diversi metodi:

- Reiezione boxare la  $f(x)$  e generare casualmente punti nella scatola
- Inversione invertire l'integrale di  $f(x)$  e generare direttamente sul dom di  $f(x)$
- Reiezione con campionamento

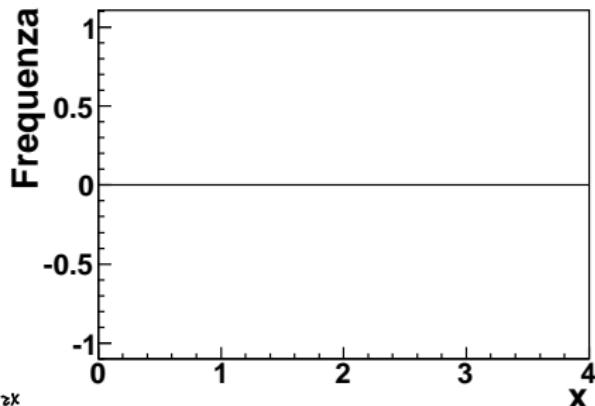
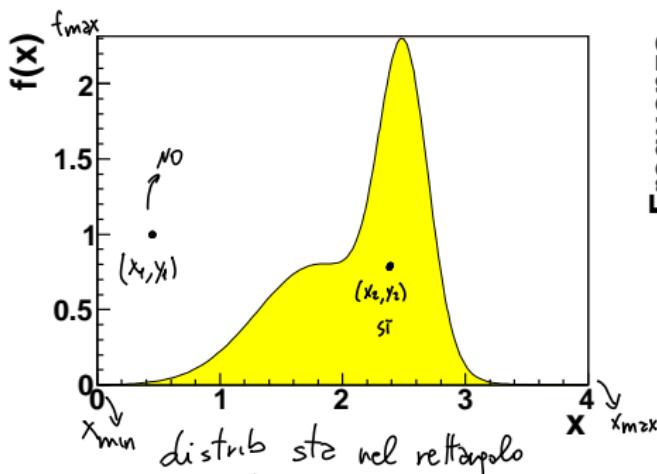
# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

dominio limitato  
immagine limitata

Se estraiamo un coppia di valori di  $(x, y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.



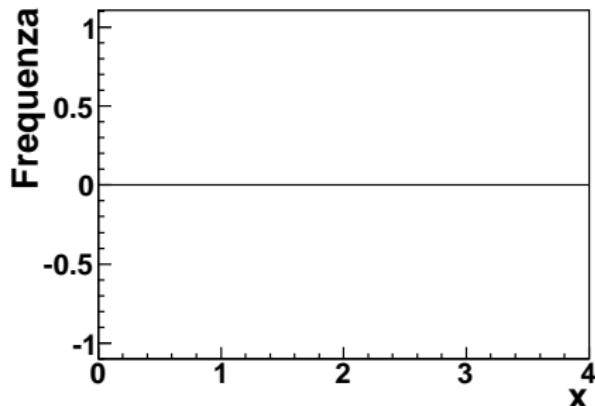
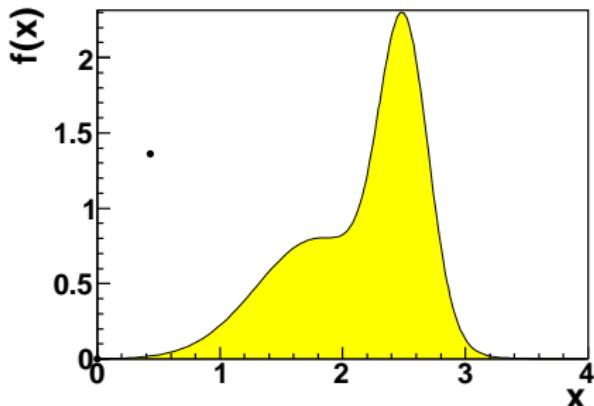
Genero casualmente  $(x_1, y_1)$  e accetto se sto nella distribuzione  
 $y_1 < f(x_1)$  se si rango istogramma (+1) in pos  $x_1$

# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.

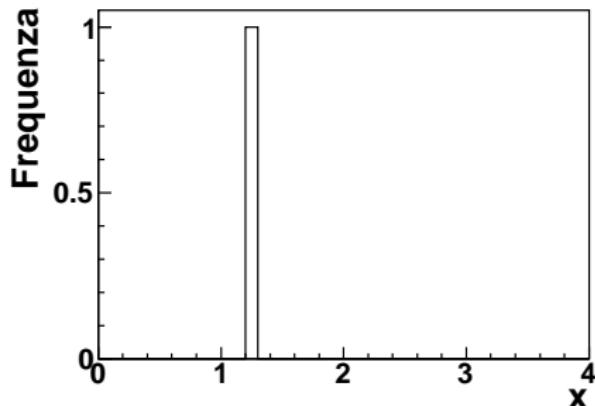
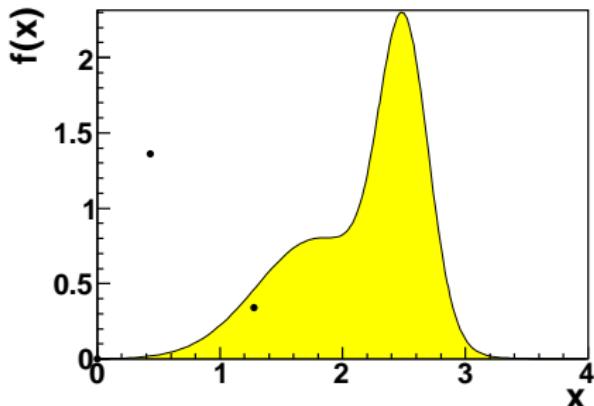


# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.

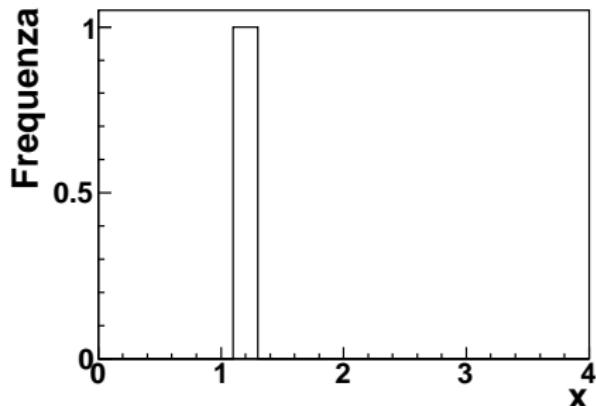
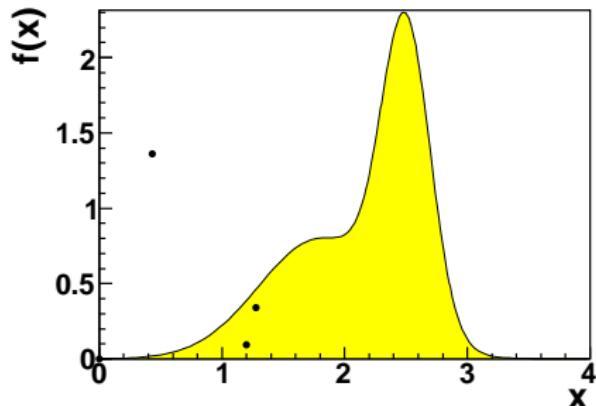


# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.

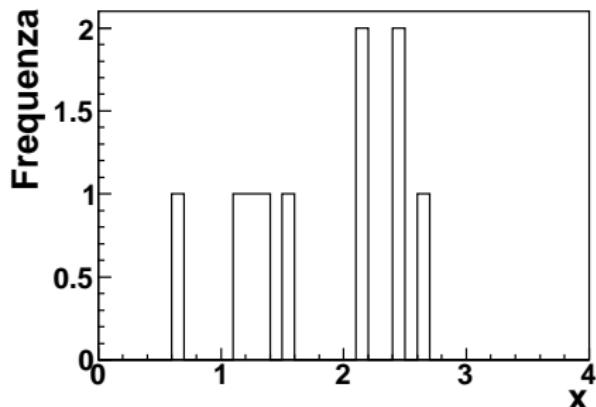
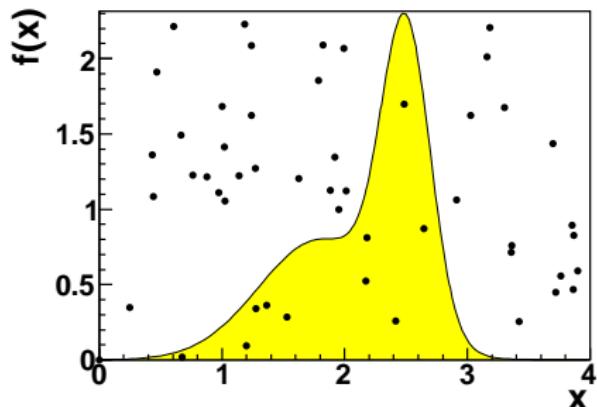


# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.

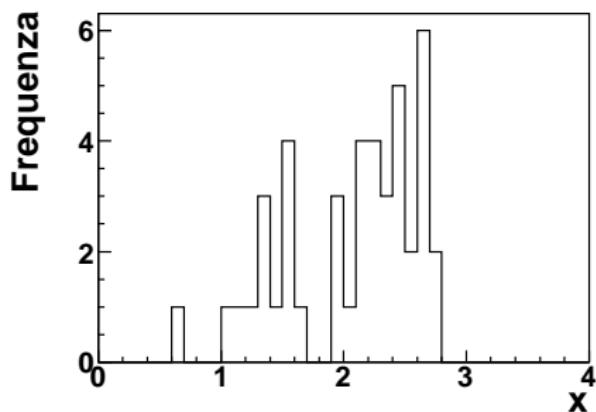
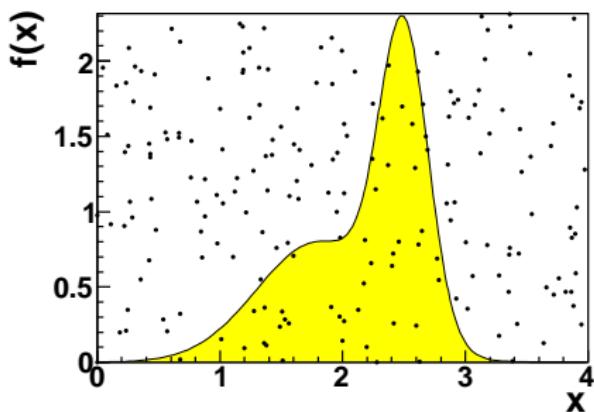


# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.

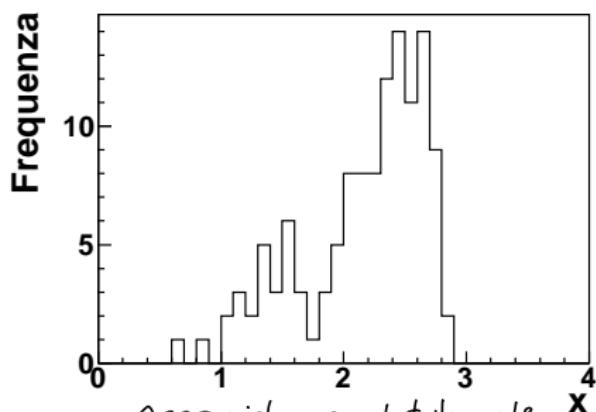
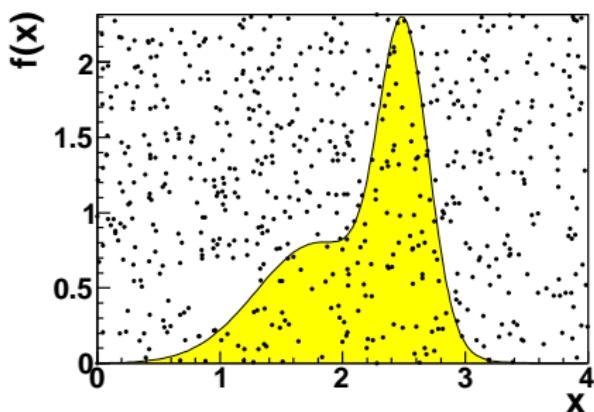


# Metodo di reiezione

Supponiamo che la distribuzione da generare  $f(x)$

- sia definita nell' intervallo  $[x_{min}, x_{max}]$
- sia, in tale intervallo, compresa tra 0 e  $f_{max}$

Se estraiamo un coppia di valori di  $(x,y)$  uniformemente distribuiti in  $[x_{min}, x_{max}] \times [0, f_{max}]$  accetteremo  $x$  se  $y$  è minore di  $f(x)$  altrimenti lo rigettiamo.



Assumiglio a distribuzione  
voglio generare

# Efficienza e limiti del metodo di reiezione

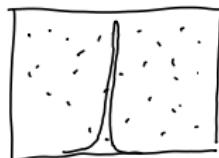
L'efficienza del metodo vale:

$$\varepsilon = \frac{\text{Valori accettati}}{\text{Valori estratti}} = \frac{\int_{x_{min}}^{x_{max}} f(x) dx}{(x_{max} - x_{min}) y_{max}}$$

Area funz  
rispetto area  
rettangolo

Limiti del metodo:

- si può applicare solo a funzioni limitate e con dominio limitato;
- risulta molto inefficiente nel caso si manipolino distribuzioni che si addensano in piccole regioni dell'intervallo (con picchi stretti).



pochi eventi validi rispetto ai generati  
perdo tempo

# Metodo di inversione

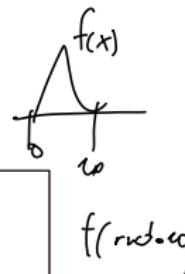
Data  $f(x)$ , definita in  $[x_{min}, x_{max}]$ , cerchiamo una funzione  $g$  tale che, se  $\eta$  è una variabile aleatoria distribuita uniformemente in  $[0, 1]$ , allora  $g(\eta)$  è distribuita secondo  $f$  in  $[x_{min}, x_{max}]$ .

Poniamo:

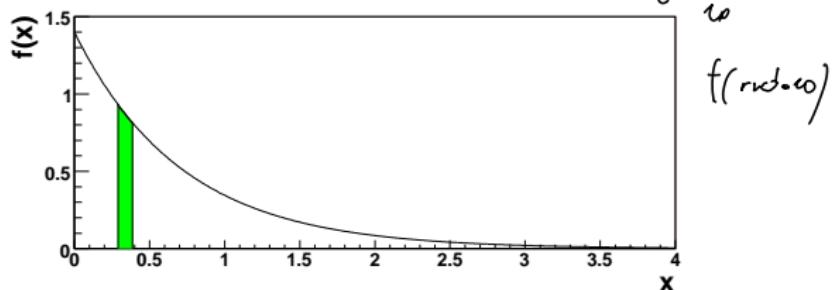
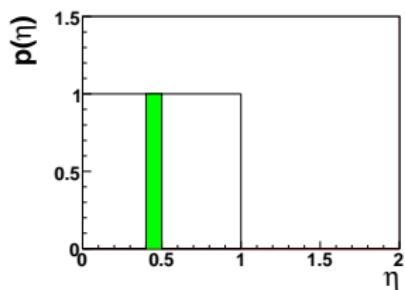
Devo trovare una funzione  
che mappi valori di  $\eta$   
in  $x$

$$g(0) = x_{min}, \quad g(1) = x_{max} \rightarrow \text{cond estremi}$$

$$p(\eta)d\eta = f(x)dx \rightarrow \text{prob conservata}$$



(la seconda condizione assicura la conservazione della probabilità)



In questo caso  $x_{min} = 0$ ,  $x_{max} = \infty$ .

# Metodo di inversione (II)

Integriamo la relazione precedente  $\rho(\eta) d\eta = f(x) dx$   $\rho(\eta) = 1$  <sup>distrib.</sup>  
uniforme

$$\int_0^\eta d\eta' = \int_{g(0)=x_{min}}^{g(\eta)=x} f(y) dy \Rightarrow \eta = \int_{x_{min}}^x f(y) dy$$

Risolvendo l'equazione ottenuta con la soluzione dell'integrale si ottiene

$$x = g(\eta)$$

se  $f(x)$  non è una densità di probabilità occorre normalizzarla

Dist. Rayleigh:

$$f(x) = \frac{2}{\pi} x e^{-\left(\frac{x}{b}\right)^2}$$

$$f(0) = 0$$

$$f(x) \rightarrow \frac{f(x)}{\int_{x_{min}}^{x_{max}} f(x) dx}$$

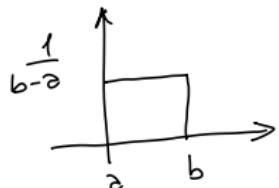
$$\begin{cases} \eta - \frac{b^2}{2} = -\frac{b^2}{2} e^{-\left(\frac{x}{b}\right)^2} \\ -\frac{b^2}{2} \eta + 1 = e^{-\left(\frac{x}{b}\right)^2} \\ \ln\left(1 - \frac{b^2}{2} \eta\right) = -\frac{x^2}{b^2} \\ x = \pm \sqrt{-b^2 \ln\left(1 - \frac{b^2}{2} \eta\right)} \end{cases}$$

# Generazione distribuzione uniforme in intervallo $[a, b]$

$$\eta \in (0, 1] \quad x \in (a, b)$$

$$\int_0^1 \rho(\eta) d\eta = \int_a^b f(x) dx$$

"normalize  
sempre"



$$\int_a^x \frac{1}{b-a} dx = \eta$$

$$\frac{1}{b-a} \int_a^x dx = \eta$$

$$\frac{x-a}{b-a} = \eta$$

$$\frac{1}{b-a} (x-a) = \eta$$

$$x = \eta(b-a) + a$$

## Metodo di inversione (III): esempio

normalizzo .  $\frac{1}{x}$  se  
non è normalizzata

Simulo decadimento

Problema generare numeri casuali secondo  $e^{-\lambda x}$  in  $[0, \infty)$ :  $\int_0^\infty e^{-\lambda x} dx = -\frac{1}{\lambda} e^{-\lambda x} \Big|_0^\infty = \frac{1}{\lambda}$

- ottengo la densità di probabilità  $p(x) = \lambda e^{-\lambda x}$

- integro:

$$\int p(u) du = \int p(x) dx$$

$$\eta = \int_0^x p(y) dy = \int_0^x \lambda e^{-\lambda y} dy$$

$$\eta = -e^{-\lambda y} \Big|_0^x = 1 - e^{-\lambda x}$$

$$1 - \eta = e^{-\lambda x}$$

$$\log(1 - \eta) = -\lambda x$$

- ... e risolvo rispetto a  $x$ :

$$x = -\frac{1}{\lambda} \ln(1 - \eta) \quad x \text{ è chiesto come un exp}$$

Si noti che il metodo di inversione non richiede che il dominio della funzione sia limitato ma solo che se ne conosca l'integrale e che l'integrale sia invertibile.

Metodo efficienza 1

# Esempio: Python

```
1 from ROOT import *
2 import time
3 import math as m
4
5 rnd = TRandom3()
6 rnd.SetSeed(int(time.time()))
7
8 h = TH1D("h","",20,0,10)
9 lam = 0.5
10
11 for i in range(0,1000):
12     x = -1./lam*m.log(1-rnd.Rndm())
13     h.Fill(x)
14
15 h.Draw()
16 gApplication.Run(True)
```

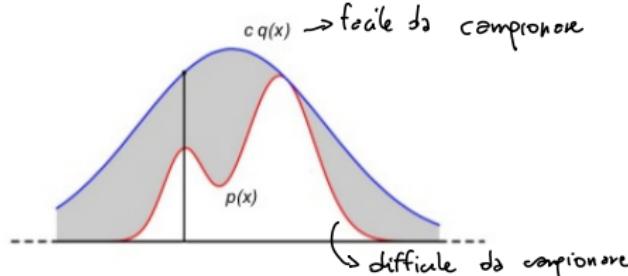
# Esempio: C++

```
1 #include<iostream>
2 #include<cmath>
3 #include<TRandom3.h>
4 #include<TMath.h>
5 #include<TH1D.h>
6 #include<TApplication.h>
7
8 using namespace std;
9 TApplication app("app", NULL, NULL);
10
11 int main(){
12     TRandom3 rnd;
13     rnd.SetSeed(int(time(0)));
14     TH1D h("h", "", 20, 0, 10);
15     double lam = 0.5;
16     int ntot = 1000;
17     double x;
18
19     for (int i=0;i<ntot;i++){
20         x = -1./lam*TMath::Log(1-rnd.Rndm());
21         h.Fill(x);
22     }
23
24     h.Draw();
25     app.Run(true);
26 }
```

# Metodo di reiezione con campionamento

eseguo due passaggi

- estraggo prima  $q(x)$
- estraggo  $p(x)$  da  $q(x)$



- $p(x)$  è difficile da campionare ma può essere valutata per ogni  $x$
- $q(x)$  invece:
  - è facile da campionare
  - si può definire una costante  $c$  tale che

$$p(x) \leq cq(x)$$

- Metodo di reiezione con campionamento:

- estraggo  $x$  secondo  $q(x)$  (utilizzando il metodo di inversione)
- estraggo  $u$  uniformemente in  $[0, cq(x)]$ 
  - se  $u \leq p(x)$  accetto  $x$
  - altrimenti estraggo un altro  $x$

migliore  
efficienza

# Generazione della gaussiana

La gaussiana non può essere generata con il metodo della reiezione (non ha dominio limitato) né con il metodo di inversione (non ha un semplice integrale analitico). *non posso perdere tempo facendo integrale numerico*

## Metodo della gaussiana 2D

Considero due gaussiane indipendenti (con  $\bar{x} = 0$  e  $\sigma = 1$ ) lungo  $x$  e  $y$

$$p(x, y) dx dy = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy = \frac{1}{2\pi} e^{-\frac{(x^2+y^2)}{2}} dx dy$$

passo in coordinate polari:

$$r^2 = x^2 + y^2 \quad \theta = \arctan\left(\frac{y}{x}\right)$$

$$dx dy = r dr d\theta$$

$$p(r, \theta) r dr d\theta = \underbrace{\frac{1}{2\pi} e^{-\frac{r^2}{2}}}_{\text{distrib per } r} r dr d\theta \quad \text{gaussiana 2D in polari}$$

## Generazione della gaussiana (II)

Effettuo il cambio di variabile  $u = r^2/2$ , segue  $du = rdr$  e quindi: genero in  $u$  e  $\theta$

$$p(x, y) dx dy = p(r, \theta) r dr d\theta = \underbrace{e^{-u}}_{\text{e}^{-u} du} \underbrace{\frac{d\theta}{2\pi}}$$

La variabile  $u$  potrà essere generata secondo  $e^{-u}$  tra  $[0, \infty)$  e  $\theta$  come uniformemente distribuita tra  $[0, 2\pi]$ ; per ricavare  $x$  e  $y$  basterà effettuare i cambi di variabile a ritroso

$$\begin{aligned} u &= -\ln(1 - \eta_1) & r &= \sqrt{2u} & \theta &= 2\pi\eta_2 \\ x &= r \cos \theta \\ y &= r \sin \theta \end{aligned}$$

$\eta_1$   
 $\eta_2$

Genero 2 variabili distr gaussiane

Abbiamo quindi definito un metodo per generare due numeri casuali indipendenti distribuiti secondo una gaussiana con media 0 e deviazione standard 1 partendo da due numeri uniformemente distribuiti da  $[0, 1]$ . Per gaussiana con media  $\mu$  e  $\sigma$ :

$$x' = \mu + \sigma \times x$$

# Esempio: Python

```
1 from ROOT import *
2 import time
3 import math as m
4
5 rnd = TRandom3()
6 rnd.SetSeed(int(time.time()))
7
8 h1 = TH1D("h1","",20,-5,5)
9 h2 = TH2D("h2","",20,-5,5,20,-5,5)
10
11 for i in range(0,100):
12     u = -m.log(1-rnd.Rndm())
13     th = rnd.Rndm()*2*m.pi
14     r = m.sqrt(2*u)
15     x = r*m.cos(th)
16     y = r*m.sin(th)
17     h1.Fill(x)
18     h1.Fill(y)
19     h2.Fill(x,y)
20
21 c1 = TCanvas()
22 h1.Draw()
23 c2 = TCanvas()
24 h2.Draw()
25
26 gApplication.Run(True)
```

# Esempio: C++

```
1 #include<iostream>
2 #include<cmath>
3 #include<TRandom3.h>
4 #include<TMath.h>
5 #include<TH1D.h>
6 #include<TH2D.h>
7 #include<TApplication.h>
8 #include<TCanvas.h>
9 using namespace std;
10 TApplication app("app", NULL, NULL);
11 int main(){
12     TRandom3 rnd;
13     rnd.SetSeed(int(time(0)));
14     TH1D h1("h1", "", 20, -5, 5);
15     TH2D h2("h2", "", 20, -5, 5, 20, -5, 5);
16     double u, th, r, x, y;
17     for (int i=0;i<100;i++){
18         u = -TMath::Log(1-rnd.Rndm());
19         th = rnd.Rndm()*2*M_PI;
20         r = TMath::Sqrt(2*u);
21         x = r*TMath::Cos(th); y = r*TMath::Sin(th);
22         h1.Fill(x); h1.Fill(y); h2.Fill(x,y);
23     }
24     TCanvas c1; h1.Draw();
25     TCanvas c2; h2.Draw();
26     app.Run(true);
27 }
```



# Generazione in ROOT

Generazione di variabili che seguono distribuzioni comuni:

```
TRandom3 rnd;
...
double x = rnd.Gaus(0,1); // numeri appartenenti ad una gaussiana
                          // centrata un 0 con larghezza 1

double x = rnd.Exp(tau); // numeri appartenenti ad un'esponenziale

...
```

Generazione di variabili che seguono una funzione qualsiasi

```
TRandom3 rnd;
...
TF1 f("f","...",-10,10);
```

double x = f.GetRandom(); // ritorna numeri distribuiti come f  
*→ usare solo in casi estremi xk prende TF1, normalizzate, dividete in bin e poi calcolate integrali usando i punti e i interpolando con paraboloida per 3 punti*

*NB* non usa metodo di inversione

# GetRandom

- `TF1::GetRandom` estraie il numero random utilizzando il metodo di inversione ma con un campionamento della funzione
  - Normalizza la funzione
  - Divide la funzione in bins
  - Calcola l'integrale in ogni bin utilizzando come funzione una parabola (interpolando parabolicamente)
  - Genera un numero random  $\eta \in [0, 1]$
  - Controlla a quale bin l'integrale  $\eta$  corrisponde
  - Calcola  $x$  utilizzando la curva parabolica nel bin selezionato
- Sconsigliato se non in casi particolari
  - è un'approssimazione della vostra funzione
  - Non è molto veloce
  - È una libreria di alto livello che ovviamente fa tutto per l'utilizzatore (con i suoi pro e contro)

Sto generando un secondo funzione ma con appox parabolico di  $f$

# Generazione di una pdf come somma pdf

$$\text{pdf} = f * \text{pdf1} + (1-f) * \text{pdf2}$$

Supponiamo di voler generare:

*se sommo pdf due determinate in quale  
evento sono*

$$\text{pdf}(x) = \sum_{i=1}^n f_i h_i(x)$$

$$\text{con } \sum_{i=1}^n f_i = 1 \text{ e } \int_{-\infty}^{+\infty} h_i(x) dx = 1$$

Estraggo  $x \in [0, 1]$ :

- se  $x < f_1$  si considera l'evento di tipo 1 distribuito secondo  $h_1$
- se  $f_1 < x < f_1 + f_2$  si considera l'evento di tipo 2 distribuito secondo  $h_2$
- se  $f_1 + f_2 < x < f_1 + f_2 + f_3$  si considera l'evento di tipo 3 distribuito secondo  $h_3$
- ...

# Riassunto Metodi Generazione

- Inversione:
  - funzione integrabile, cumulativa invertibile
  - massima efficienza (ad ogni valore estratto  $\eta$  avete un corrispondente valore di  $x$ )
- Reiezione:
  - qualsiasi funzione con dominio limitato
  - efficienza dipendente dalla funzione
- Reiezione con campionamento:
  - efficienza dipendente dalla funzione di envelope

# Calcolo di Integrali con il metodo MonteCarlo

- Metodi MonteCarlo sono spesso usati come tecniche numeriche per il calcolo degli integrali

# Calcolo di Integrali: metodo di reiezione (I)

Supponiamo di dover calcolare l'integrale di una funzione in un intervallo limitato  $[x_{min}, x_{max}]$ , e di conoscere il massimo ed il minimo della funzione in tale intervallo. Se generiamo  $N_{tot}$  punti uniformemente distribuiti nel rettangolo

$$[x_{min}, x_{max}] \times [f_{min}, f_{max}]$$

avremo che la frazione  $p$  di punti che cadono sotto la funzione è pari al rapporto tra l'integrale e l'area del rettangolo  $A_{tot}$ .

$$A_{tot} = (x_{max} - x_{min})(f_{max} - f_{min})$$

# Calcolo di Integrali: metodo di reiezione (I)

Supponiamo di dover calcolare l'integrale di una funzione in un intervallo limitato  $[x_{min}, x_{max}]$ , e di conoscere il massimo ed il minimo della funzione in tale intervallo. Se generiamo  $N_{tot}$  punti uniformemente distribuiti nel rettangolo

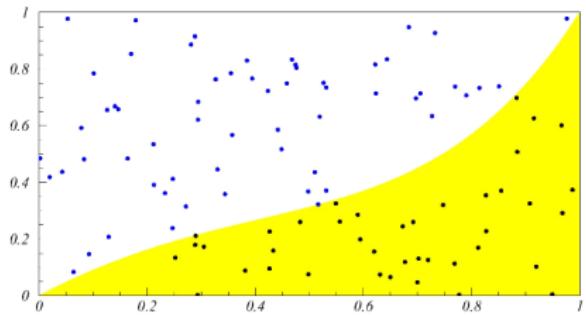
$$[x_{min}, x_{max}] \times [f_{min}, f_{max}]$$

avremo che la frazione  $p$  di punti che cadono sotto la funzione è pari al rapporto tra l'integrale e l'area del rettangolo  $A_{tot}$ .

$$A_{tot} = (x_{max} - x_{min})(f_{max} - f_{min})$$

Si ha:

$$I = \int_{x_{min}}^{x_{max}} f(x)dx = A_{tot}p = A_{tot} \frac{N_{acc}}{N_{tot}}$$



# Calcolo di Integrali: metodo di reiezione (II)

La distribuzione di successi è binomiale con parametro  $p$ . Per cui:

$$I = \int_{x_{min}}^{x_{max}} f(x)dx = A_{tot}p = A_{tot} \frac{N_{acc}}{N_{tot}}$$

$$\sigma(I) = A_{tot} \sqrt{\frac{p(1-p)}{N_{tot}}} \propto \frac{1}{\sqrt{N_{tot}}}$$

L'errore decresce con  $N_{tot}$  molto lentamente

# Calcolo di Integrali: approccio alternativo

Metodo di campionamento semplice: per calcolare l'integrale lo si può riscrivere come

$$I = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{g(x)} g(x)dx$$

Calcolo sul aspett  
 del rapporto  
 ↓ deve essere uniforme tra a e b

$$I = E\left[\frac{f(x)}{g(x)}\right]_g$$

dove  $x$  è una variabile casuale distribuita secondo  $g(x)$ .

Assumiamo  $g(x) = \frac{1}{(b-a)}$  per  $a < x < b$  (per cui la variabile aleatoria  $x$  è distribuita uniformemente in  $(a, b)$ )

$$I = E\left[\frac{f(x)}{\frac{1}{b-a}}\right]_g = (b-a)E[f(x)]$$

$$E[f(x)] = \frac{1}{N} \sum_{i=1}^N f(x_i)$$

# Calcolo di Integrali: approccio alternativo

$$\begin{aligned}
 I &= \int_a^b f(x) dx \\
 &= \int_a^b (b-a)f(x) \left( \frac{1}{b-a} \right) dx = (b-a)E[f(x)] \\
 &= (b-a) \frac{1}{N} \sum_{i=1}^N f(x_i) = (b-a)\langle f \rangle
 \end{aligned}$$

dove  $x_i$  sono distribuiti uniformemente nell'intervallo  $[a, b]$ . L'errore è dato da

$$\sigma_N = (b-a) \sqrt{\frac{(\langle f^2 \rangle - \langle f \rangle^2)N/(N-1)}{N}} = (b-a) \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N-1}}$$

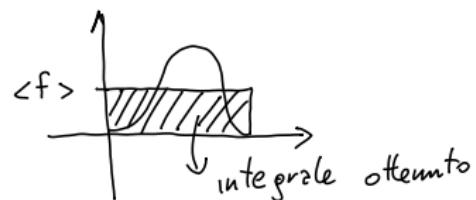
varianza su  $f$  e molt.  $\cdot (b-a)$

Se uso  $g(x)$  generico: cerca di ridurre errore sull'integrale, salvo  $g(x)$   
non uniforme

# Calcolo di Integrali: approccio alternativo

- Generare  $N$  numeri casuali  $\eta_i$  distribuiti uniformemente in  $(0,1)$
- Calcolare  $x_i = a + \eta_i(b - a)$
- Calcolare  $f(x_i)$
- Calcolare  $I = (b - a) \frac{1}{N} \sum_{i=1}^N f(x_i)$

Vantaggio : non butta via eventi  
 ogni  $x_i$  generato viene usato  
 poche e semplici istruzioni



# Calcolo di Integrali: importance sampling

- L'idea di base è quello di ricorrere a tecniche che riducono la varianza (senza modificare la media)  $\text{uso } g(x) \text{ non costante}$
- Provo con una densità di probabilità non uniforme  $g(x)$

$$\int_a^b \frac{f(x)}{g(x)} g(x) dx = E[f(x)/g(x)]_g \\ = \langle f/g \rangle_g$$

dove  $x_i$  sono distribuiti secondo  $g(x)$

- Il campionamento uniforme usa una  $g(x)$  costante, invece in questo caso, si campionano maggiormente le regioni in cui  $g(x)$  è grande

Vantaggio: uso uno sorts di peso che campiona maggiormente le regioni dove  $g(x)$  è grande

Come  $g(x)$ : Facilmente campionabile (per campionarla es. usare inversione)

# Calcolo di Integrali: importance sampling

- L'errore è dato da

$$\sigma_N = \sqrt{\frac{(\langle (f/g)^2 \rangle - \langle f/g \rangle^2)N/(N-1)}{N}} = \sqrt{\frac{\langle (f/g)^2 \rangle - \langle f/g \rangle^2}{N-1}}$$

per N fissato l'errore è minimo se

$$g(x) = \frac{f(x)}{\int f(x)dx}$$

$$g(x) \rightarrow f(x)$$

$$g(x) = f(x)$$

$$\sigma_n \rightarrow 0$$

campionabile  
 facilmente e sia  
 simile a  $f(x)$   
 il più possibile  
 $\frac{f(x)}{g(x)} \rightarrow 1$

usando una  $g(x)$  deve essere  
 normalizzata

# Calcolo di Integrali: Esempi

ESAME

Immaginiamo di voler integrare una funzione  $f(x, y, z)$  sul volume di un solido:

$$I = \int_{\Omega} f(x, y, z) d\Omega$$

- Volume di un solido  $f(x, y, z) = 1$
- Massa di un solido
- Momento di inerzia di un solido
- Carica elettrica
- Campo elettrico, gravitazionale
- ...

# Calcolo di Integrali: Esempi

Vogliamo calcolare il volume di un solido  $f(x, y, z) = 1$ :

$$I = \Omega = \int_{\Omega} d\Omega$$

Se generiamo  $N$  punti uniformemente distribuiti in un volume che contenga il nostro solido  $V_{tot}$  avremo che la frazione  $p$  di punti che cadono all'interno del solido è pari al rapporto tra il volume del solido di interesse  $\Omega$  e il volume che contiene il solido  $V_{tot}$

Per cui si ha:

$$\Omega = \int_{\Omega} d\Omega = V_{tot} p = V_{tot} \frac{N_{\Omega}}{N}$$

# Calcolo di Integrali: Esempi

Nel caso  $f(x, y, z)$  generica, possiamo riscrivere l'integrale come

$$I = \int_{\Omega} f(x, y, z) d\Omega = \int_{\Omega} \underbrace{\frac{f(x, y, z)}{g(x, y, z)} g(x, y, z)}_{\text{val' aspett del rapporto . cost}} d\Omega$$

Assumiamo  $g(x, y, z) = \frac{1}{\Omega}$  (per cui le variabili aleatorie  $(x, y, z)$  sono distribuite uniformemente in  $\Omega$ )

$$I = E\left[\frac{f(x, y, z)}{\frac{1}{\Omega}}\right]_g = \Omega E[f(x, y, z)]_g$$

$$E[f(x, y, z)]_g = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} f(x_i, y_i, z_i)$$

con  $N_{\Omega}$  = numero di punti all'interno del volume  $\Omega$

# Calcolo di Integrali: Esempi

Nel caso in cui non si può (non si sa) generare uniformemente in  $\Omega$ : genero N punti uniformemente in un volume che contenga il nostro solido  $V_{tot}$ .

$\Omega = \frac{N_\Omega}{N} V_{tot}$       f è una sorta di peso, non moltiplica · 1 ma · f( $x, y, z$ )

$$\begin{aligned}
 I &= \Omega \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} f(x_i, y_i, z_i) = \\
 &= V_{tot} \frac{N_\Omega}{N} \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} f(x_i, y_i, z_i) = \\
 &= \frac{V_{tot}}{N} \sum_{i=1}^{N_\Omega} f(x_i, y_i, z_i)
 \end{aligned}$$

# Calcolo di Integrali: Esempi

Possiamo anche vedere  $f(x, y, z)$  come peso per cui:

$$I = \int_{\Omega} f(x, y, z) d\Omega = V_{\text{tot}} \frac{\sum_i f_i(x, y, z)}{N}$$

Analogamente a:

$$I = \int_{\Omega} d\Omega = V_{\text{tot}} p = V_{\text{tot}} \frac{N_{\Omega}}{N}$$

# Calcolo di Integrali

Quando diventa conveniente valutare un integrale con il metodo Monte Carlo ? Ricordiamo che ogni estrazione implica il calcolo della funzione integranda (per stabilire se  $y < f(x)$  cioè se il punto è contenuto nell'area).

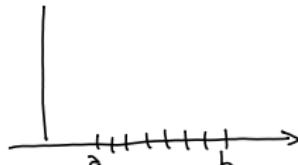
Consideriamo una funzione a  $m$  dimensioni e chiamiamo  $N$  il numero di valutazioni della funzione integrale:

**Simpson:** fissato a  $n$  il numero di campionamenti per dimensione  $m$  vale  $n^m$

**Monte Carlo:** la precisione migliora come  $1/\sqrt{N}$

Quando a parità di tempo di esecuzione i due metodi danno precisioni simili con stesso tempo di esecuzione ? La risposta non è semplice perché dipende da come è fatta la funzione.

In generale però per  $m > 4$  il metodo Monte Carlo comincia a diventare competitivo.



$n^m$  intervalli

con  $S = n$   
e  $10$  dimensioni  
 $S^{10} = 10^M$ , tenti  
passaggi

# Appendice: errore integrale

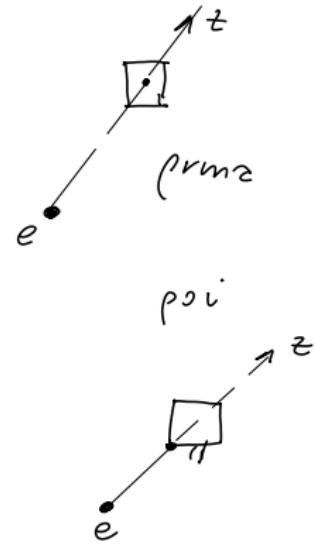
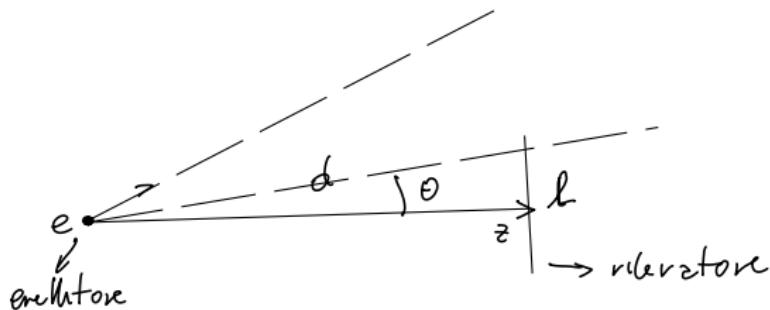
Lo stimatore della varianza:

$$\begin{aligned}\hat{V}^2 &= \frac{1}{N-1} \left( \sum_{i=1}^N f^2(x_i) - \left( \sum_{i=1}^N f(x_i) \right)^2 \right) \\ &= \frac{N}{N-1} (\langle f^2 \rangle - \langle f \rangle^2)\end{aligned}$$

L'errore sul valor medio:

$$\sigma_N = \sqrt{\frac{\hat{V}^2}{N}} = \sqrt{\frac{|(N/N-1)|(\langle f^2 \rangle - \langle f \rangle^2)}{N}} = \sqrt{\frac{(\langle f^2 \rangle - \langle f \rangle^2)}{N-1}}$$

H. on es 2)



$$\hat{u} = (\cos \phi \sin \theta, \sin \phi \sin \theta, \cos \theta)$$

$s\hat{u}$  incontra il piano

$$d = (s\hat{u})_z = s \cos \theta \quad s = \frac{d}{\cos \theta}$$

$$(s\hat{u})_x$$

$$(s\hat{u})_y \text{ e controlla se } y > x < \left| \frac{L}{2} \right|$$

- Estrare campione che segue  $f(x) = \frac{k}{\lambda^k} x^{k-1} e^{-(x/\lambda)^k}$  WEIBULL  $\lambda = 1$   $k = 1.5$

Metodo inversione:

$$\eta = \int_{x_{\min}}^{x=\gamma(y)} f(y) dy = \int_0^x dy \frac{k}{\lambda^k} y^{k-1} e^{-(y/\lambda)^k} = \frac{k}{\lambda^k} \int_0^{(x/\lambda)^k} dt \frac{\lambda^k}{ky^{k-1}} e^{-t} = \frac{k}{\lambda^k} \int_0^{(x/\lambda)^k} dt \lambda^k e^{-t} =$$

$$\eta = k \int_0^{(x/\lambda)^k} dt e^{-t} = k \left[ -e^{-t} \right]_0^{(x/\lambda)^k} = -k e^{-(x/\lambda)^k} + k$$

$$\eta - k = -k e^{-(x/\lambda)^k} \quad e^{-(x/\lambda)^k} = 1 - \frac{\eta}{k} \quad -(x/\lambda)^k = \ln \left( 1 - \frac{\eta}{k} \right) \quad x = \sqrt[k]{-\lambda \ln \left( 1 - \frac{\eta}{k} \right)} \quad \eta \text{ random}$$

- Estrare campione che segue Rayleigh  $f(x) = \frac{2}{\sigma^2} x e^{-(x/b)^2}$

Metodo inversione:

$$\eta = \int_{x_{\min}}^{x=\gamma(y)} f(y) dy = \int_0^x dy \frac{2}{\sigma^2} y e^{-(y/b)^2} = \frac{1}{\sigma^2} \int_0^x dt \frac{b^2}{2y} 2y e^{-t} = \frac{b^2}{\sigma^2} \int_0^x dt e^{-t} =$$

$$\eta = \frac{b^2}{\sigma^2} \left( -e^{-(y/b)^2} \right) \Big|_0^x = \frac{b^2}{\sigma^2} \left( -e^{-x^2/b^2} + 1 \right)$$

$$\eta - \frac{b^2}{\sigma^2} = -e^{-x^2/b^2} \left( \frac{b^2}{\sigma^2} \right) \rightarrow -e^{-x^2/b^2} = \eta \frac{\sigma^2}{b^2} - 1 \rightarrow e^{-x^2/b^2} = 1 - \frac{\sigma^2}{b^2} \eta$$

$$\rightarrow -\frac{x^2}{b^2} = \ln \left( 1 - \frac{\sigma^2}{b^2} \eta \right) \rightarrow x = \pm \sqrt{-b^2 \ln \left( 1 - \frac{\sigma^2}{b^2} \eta \right)}$$

$x$  segue Rayleigh

Random