Metodologie informatiche avanzate per la Fisica Nucleare e Subnucleare

Modulo su Metodo Monte Carlo e suo utilizzo in HEP

Maximiliano Sioli

A.A. 2007/08

Cosa faremo in questa parte del corso

1) Parte teorica

   Richiameremo i concetti fondamentali di probabilità e statistica e forniremo un approccio “teorico” al metodo Monte Carlo.

2) Esempio di un HEP MC: FLUKA

   Vedremo come alcuni dei concetti espressi nella prima parte del corso sono implementati in un particolare codice MC per la fisica delle alte energie.

3) Analisi dati con l’uso del metodo MC

   Mostreremo come e dove il metodo MC interviene durante l’analisi dei dati nella fisica delle alte energie.

N.B. Non impareremo ad usare FLUKA, Geant4 o ROOT, ma li utilizzeremo come esempi di strumenti di calcolo per implementare le nozioni che via via apprenderemo

Definizione di metodo MC

- Definizione generale: il metodo MC è una qualunque tecnica numerica che utilizza *generatori di numeri casuali* per risolvere un problema.
- Definizione più "sottile" (J.H. Haltom, 1970):
  supponiamo di avere un problema in cui vi sia da stimare un parametro *F* caratteristico di una certa popolazione (un numero reale, un array o una variabile booleana). Il metodo MC è una tecnica numerica che permette di costruire un campione di tale popolazione e di estrarre da esso una stima statistica di *F*.
- In generale, l'utilizzo della tecnica MC può essere diviso in due categorie:
  - Tecnica di *integrazione* numerica, particolarmente utile nel caso di spazi multi-dimensionali
  - Tecnica di *simulazione* della realtà fisica, che è stocastica per natura. In questo caso la tecnica MC tenta di riprodurre in dettaglio il processo fisico in esame, simulando:
    - I valori medi delle grandezze in gioco
    - Le fluttuazioni intorno al valor medio

  Come? "discretizzando" la variabile temporale → Catene di Markov

Osservazione

Sebbene tutti i processi fisici siano stocastici nel loro aspetto più fondamentale (la natura è “quantistica” dopo tutto), l’utilizzo del metodo MC dipende dalla modellizzazione del sistema fisico che vogliamo studiare:

- Ad esempio, l’equazione di trasferimento del calore o l’equazione di trasporto di Boltzmann richiedono l’uso del MC come “integratore” numerico
- Lo sviluppo di un parton shower o lo sviluppo di uno sciame di raggi cosmici richiede l’uso del MC come “simulatore” stocastico

Storia del metodo MC

- Inventato da Ulam e von Neumann negli anni ‘40 mentre lavoravano al progetto Manhattan e studiavano le dinamiche delle esplosioni nucleari (il nome è invece legato alle case da gioco del Principato di Monte Carlo)
- Al tempo, l’interesse era puramente accademico: non vi erano calcolatori...
- Con l’avvento dei calcolatori l’interesse è rinato.
- Storicamente, i primi calcoli su larga scala basati sul metodo Monte Carlo vennero eseguiti per lo studio di **scattering e assorbimento dei neutroni**. Questi processi hanno **natura casuale**, e dunque si può facilmente far corrispondere un campione ipotetico, costruito con numeri casuali, al campione reale.
- Applicazioni in molte aree: scienze matematiche, fisiche, statistica, matematica finanziaria...

Utilizzo del metodo MC in HEP - cont

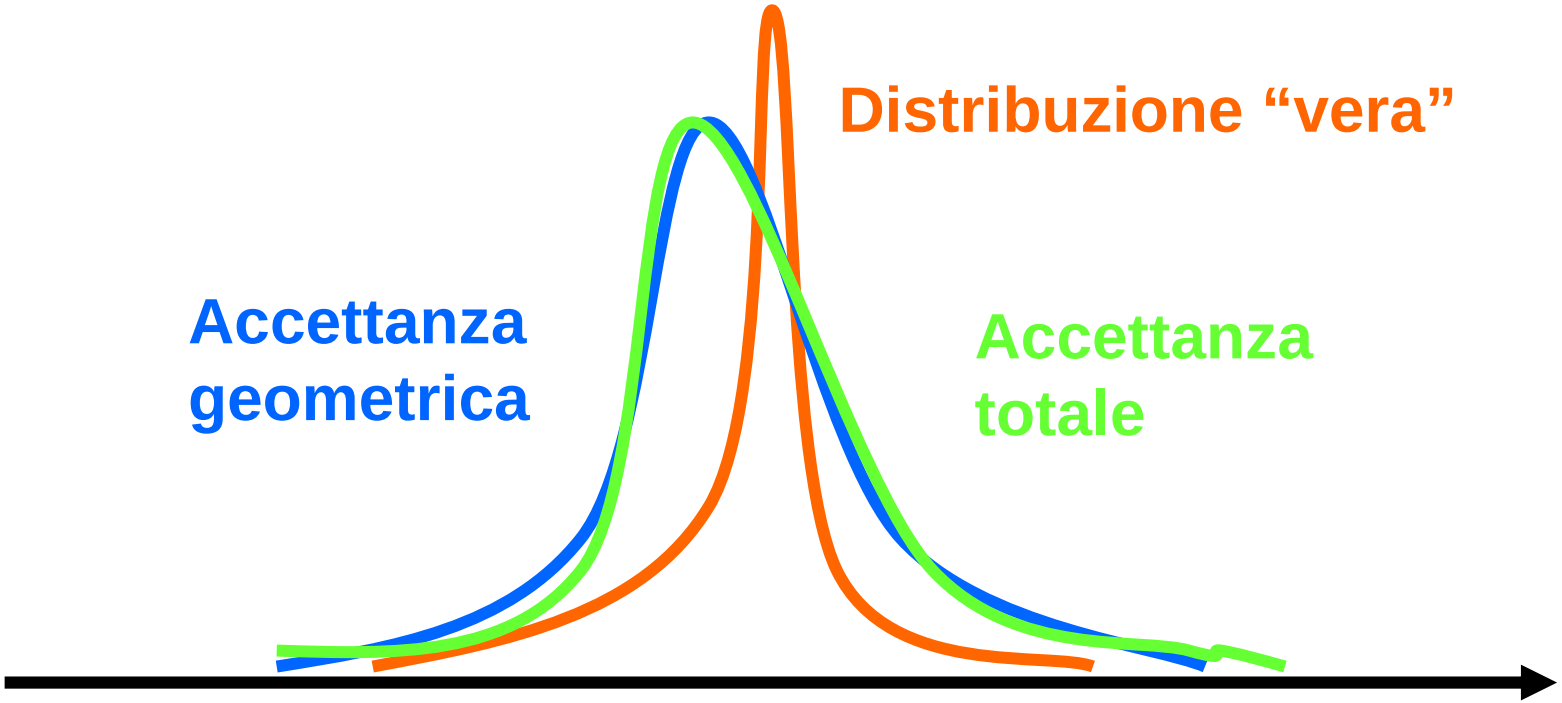
- Nella fisica delle alte energie, il metodo MC viene utilizzato in fasi diverse:
  - simulazione del processo fisico (la “cinematica”):
    → *Event Generators*. Possono essere:
    - Parametrici (o parametrizzati) → Risposta veloce
    - Completi (calcolo di sezioni d’urto, 4-vettori etc...)
  - Simulazione del rivelatore, a tre livelli:
    - geometria
    - simulazione degli “hits”
    - digitizzazione (risoluzioni, efficienze...)
  - A questo livello i dati simulati sono nello stesso formato dei dati sperimentali
  - Definizione del volume fiduciale (boundary conditions)
  - Definizione dei tagli di analisi

Utilizzo del metodo MC in HEP

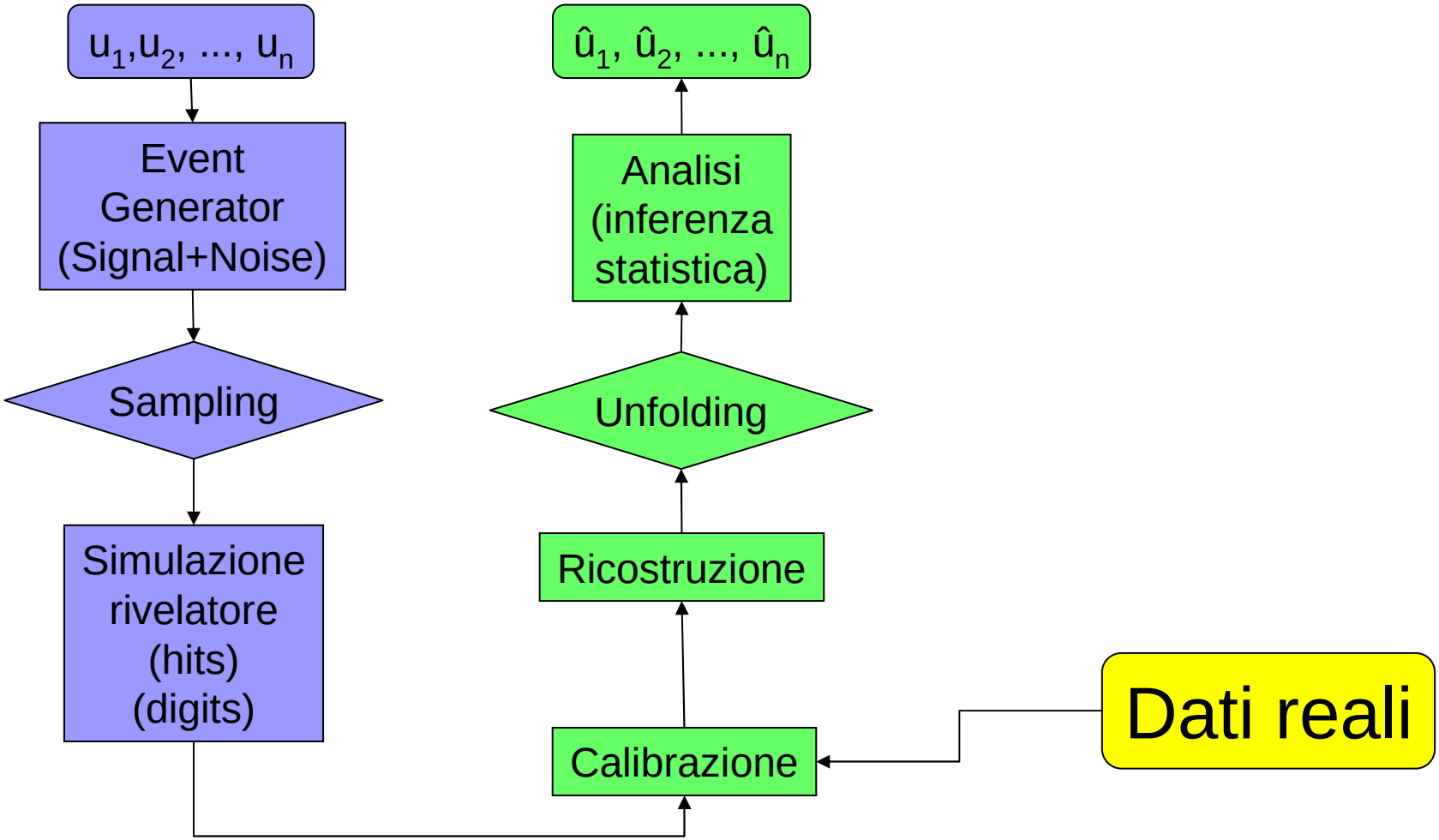
- L'uso del MC è fondamentale per scoprire a priori la sensibilità di un esperimento (e quindi capire se vale la pena costruirlo e chiedere finanziamenti!)
- Il MC mi permette di valutare segnale e fondo
- Esempio: voglio costruire un esperimento per verificare l'ipotesi di oscillazione di neutrini $\nu_\mu \rightarrow \nu_\tau$
  Cosa devo fare? Usare il MC per:
  - Simulare i processi fisici che generano il segnale (interazioni ν_τ CC e decadimento del tau)
  - Simulare i processi fisici che possono "mimare" il segnale (e.g. decadimento di particelle "charmate")
  - Simulare la risposta del rivelatore
  - Ottimizzare i tagli → studio della sensibilità

Simulazione del rivelatore

- Quali sono i processi fisici che avvengono all'interno dell'apparato?
- Con quale probabilita' avvengono?
- Come modificano l'evento fisico iniziale? Con quali fluttuazioni?
- La geometria di un rivelatore e' spesso complessa: come si calcola l'accettanza dell'apparato?
- Come le varie (in)efficienze dei rivelatori modificheranno la risposta dell'apparato?
- Simulazione della risposta del rivelatore alle particelle
- Ottimizzazione del design del rivelatore
- Calcolo di risoluzioni, efficienze, accettanze...



Flow chart di un'analisi sperimentale in HEP



Visione schematica di una sequenza di codici MC come black-box per fare inferenza sui parametri

Esempio: oscillazioni dei neutrini

In questa parte del corso mostreremo che peso ha la simulazione Monte Carlo all'interno della "black-box"

Richiami di probabilità e statistica

Materiale utile per introdurre i concetti fondamentali del metodo

Variabili aleatorie e distribuzioni

- Variabile aleatoria (**RV**): è una variabile che può assumere un certo valore (in un dominio discreto o continuo) che è però sconosciuto a priori.
  N.B. La sua distribuzione può essere invece nota a priori.

- *Cos'è la distribuzione di una RV?*
  La funzione densità di probabilità (**PDF**) definisce la probabilità che una RV *u'* assuma un valore in un particolare intorno *du* del dominio:

$$\rho(u)du = \mathcal{P}[u < u' < u + du],$$

$\rho(u)$= pdf

- La funzione cumulativa di probabilità (**CDF**) è definita come:

$$R(u) = \int_{-\infty}^{u} \rho(x)dx, \qquad \rho(u) = \frac{dR(u)}{du}$$

$R(u)$ = funzione monotona non-decrescente e $0 \leq R(u) \leq 1$

Valore di aspettazione, varianza, covarianza

- Valore di aspettazione di una funzione $f(u')$:

$$E(f) = \int f(u) dR(u) = \int f(u)\rho(u)du\,.$$

Se $u' \in \mathcal{U}(0,1)$, ovvero ha una PDF uniforme tra 0 e 1, allora $E(f) = \int_0^1 f(u)du$

- Varianza di una funzione $f(u')$:

$$V(f) = E[f - E(f)]^2 = \int [f - E(f)]^2 dR = E(f^2) - E^2(f)\,.$$

→ Deviazione standard: $\sigma(f) = \sqrt{V(f)}\,.$

- Se *x* e *y* sono due RV e *c* una costante. si ha:

$$E(cx + y) = cE(x) + E(y),$$

$$V(cx + y) = c^2 V(x) + V(y) + 2c\,\mathrm{Cov}(x, y),$$

dove $\mathrm{Cov}(x, y) = E([x - E(x)][y - E(y)])$ è la covarianza tra *x* e *y*

→ Se $\mathrm{Cov}(x, y) = 0$ allora *x* e *y* sono scorrelate ma non necessariamente indipendenti

La legge dei grandi numeri

Scegliamo *n* numeri casuali u_i con PDF uniforme nell'intervallo (a,b) e per ciascun u_i valutiamo la funzione $f(u_i)$.
Quando *n* diventa grande:

$$\frac{1}{n}\sum_{i=1}^{n} f(u_i) \underset{n\to\infty}{\longrightarrow} E(f) = \frac{1}{b-a}\int_a^b f(u)du$$

Nel linguaggio statistico, il termine a sinistra è un **estimatore consistente** dell'integrale poiché, sotto certe condizioni, converge al valore esatto dell'integrale quando *n* tende ad infinito.

Le condizioni riguardano la funzione *f*, che deve essere:

1. Integrabile
2. Continua a tratti (può avere un numero finito di discontinuità)
3. Finita ovunque

La legge dei grandi numeri può essere espressa con la seguente affermazione:
"L'estimatore MC di un integrale converge al valor vero quando il campionamento diviene molto grande".

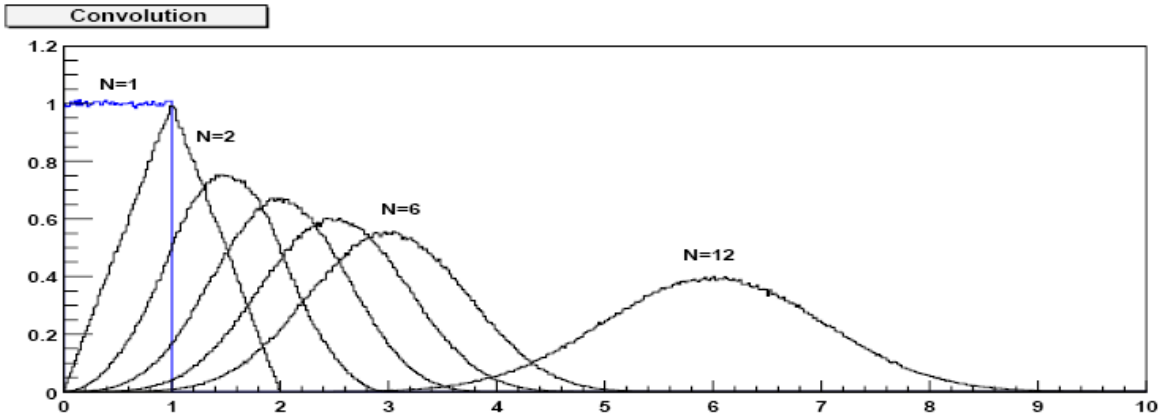
Il teorema del limite centrale

La somma di un grande numero di RV indipendenti è sempre distribuita normalmente (distribuzione Gaussiana), indipendentemente da quali siano le PDF delle singole variabili, purchè esse abbiano media e varianza finite e che *n* sia un numero grande.

→ Indichiamo con $N(\mu, \sigma^2)$ la PDF normale con media μ e varianza σ^2 :

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

→ Illustrazione del teorema del limite centrale per $x_i \in \mathcal{U}(0,1), i = 1, \ldots, 12$:



Generatore di numeri random normalmente distribuiti basato sul teorema del limite centrale

Sia $x_i \in \mathcal{U}(0,1), i = 1, \ldots, n,$ consideriamo $R_n = \sum_{i=1}^{n} x_i,$

Si ha che:

$$\left.\begin{array}{l} E(x_i) = \frac{1}{2} \\ V(x_i) = \frac{1}{12} \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} E(R_n) = \frac{n}{2} \\ V(R_n) = \frac{n}{12} \end{array}\right.$$

Da cui otteniamo che:

$$\frac{R_n - n/2}{\sqrt{n/12}} \underset{n\to\infty}{\longrightarrow} N(0,1),$$

→Otteniamo un generatore di numeri random distribuiti normalmente.

Operativamente, una scelta conveniente risulta essere:

$$n = 12 \quad \longrightarrow \quad R_{12} - 6\,.$$

Attenzione: questo tipo di generatore non riproduce correttamente le code della distribuzione.

Proprietà matematiche del metodo Monte Carlo

Consideriamo $u_i \in \mathcal{U}(a,b)$, per la legge dei grandi numeri:

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n} f(u_i)}_{\text{Estimatore MC dell'integrale}} \quad \underset{n\to\infty}{\longrightarrow} \quad \frac{1}{b-a}\int_a^b f(u)du$$

Proprietà matematiche dell'estimatore MC:

1. Se $V(f) < \infty$, l'estimatore MC è *consistente*, cioè converge al valor vero dell'integrale per *n* molto grande.
2. L'estimatore MC è *unbiased* per ogni *n*: il valore di aspettazione dell'estimatore MC è il valore vero dell'integrale (facile da verificare dalla linearità dell'operatore E).
3. L'estimatore MC è asintoticamente distribuito normalmente
4. La deviazione standard dello stimatore MC è: $\sigma = \frac{1}{\sqrt{n}}\sqrt{V(f)}$

→ Lo stimatore MC della deviazione standard è: $\hat{\sigma} = \frac{1}{\sqrt{n}}\sqrt{\hat{V}(f)}$

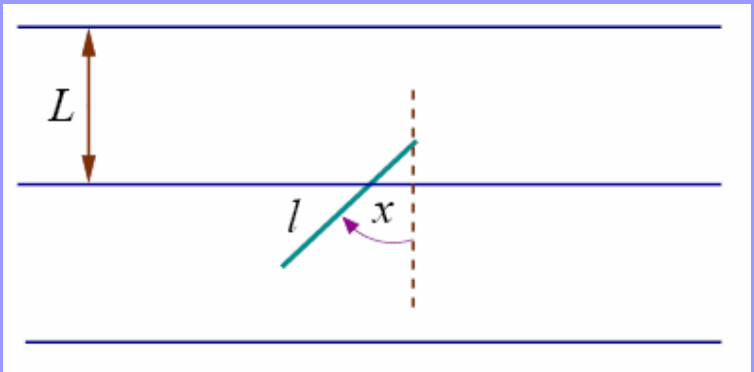
dove: $\hat{V}(f) = \frac{1}{n-1}\sum_{i=1}^{n}\left[f(u_i) - \frac{1}{n}\sum_{i=1}^{n} f(u_i)\right]^2$

Il problema dell'ago di Buffon ed i metodi Monte Carlo

(Buffon 1777, Laplace 1886)

Un ago di lunghezza *l* viene lanciato random su un piano orizzontale in cui sono tracciate linee rette distanti tra loro L≥*l*. Se l'ago cade su una linea viene registrato un "hit", altrimenti un "miss". Contando gli "hit" e "miss", calcolare il valore di π.

Esperimento:



n – numero di "hit"

N – numero di tentativi ("hit+miss")

Teoria:

Sia x l'angolo tra l'ago e la perpendicolare alle linee, $x \in \mathcal{U}(0,\pi)$

→ La funzione di densità di probabilità di x:

$$\rho(x) = \frac{1}{\pi}$$

Sia $p(x)$ la probabilità di un "hit" per un dato angolo $p(x) = (l/L)|\cos x|$

→ La probabilità totale dell' "hit":

$$P = E[p(x)] = \int_0^{\pi} p(x)\rho(x)dx = \frac{2l}{\pi L}$$

→ Legge dei grandi numeri:

$$\frac{n}{N} \underset{N\to\infty}{\longrightarrow} P = \frac{2l}{\pi L} \quad \Longrightarrow \quad \boxed{\frac{2Nl}{nL} \underset{N\to\infty}{\longrightarrow} \pi}$$

Numeri casuali e generatori di numeri pseudo-casuali

Numeri casuali

- Il metodo MC, per sua natura, si basa su sequenze di numeri casuali
- Cos'è un numero casuale? Semplicemente il valore assunto da una variabile aleatoria → non predicibile sulla base della storia passata
- Come ottenere tali sequenze?
  - Da processi fisici: tempi di decadimento di sostanze radioattive, tempi di arrivo dei raggi cosmici, rumore termico nei dispositivi elettronici... o lanciando monete: e.g. lanciando una moneta per 32 volte ottengo un numero casuale a 32 bit...
    Drawbacks dei generatori fisici:
    - sono lenti...
    - non sono stabili: piccoli cambiamenti al contorno possono cambiare le proprietà probabilistiche del fenomeno
    - A tale scopo in passato esistevano *tabelle* di numeri casuali. Non pratiche, ma con l'avvento degli *storage devices* si sta tornando al loro utilizzo (CDs, DVD etc)
  - Da sequenze *pseudo-casuali*, ovvero sequenze generate da formule matematiche... quindi sequenze in teoria perfettamente predicibili!

Numeri pseudo-casuali

$$x_{n+1} = f(x_n)$$

sono sequenze perfettamente deterministiche!

- La definizione formale di numeri pseudo-casuali urta contro la definizione stessa di sequenza casuale, ovvero di sequenza di numeri “algoritmicamente incomprimibile”... infatti è proprio l’esatto contrario!
- Comunque, sebbene siano sequenze perfettamente deterministiche, le loro proprietà statistiche sono molto simili a quelle di sequenze di veri numeri casuali.
- Sono veloci, semplici ed affidabili: hanno di fatto sostituito i generatori basati su processi fisici

Come è possibile generare numeri random da sequenze deterministiche?

- Vediamo un esempio di transizione al caos studiando questa sequenza:

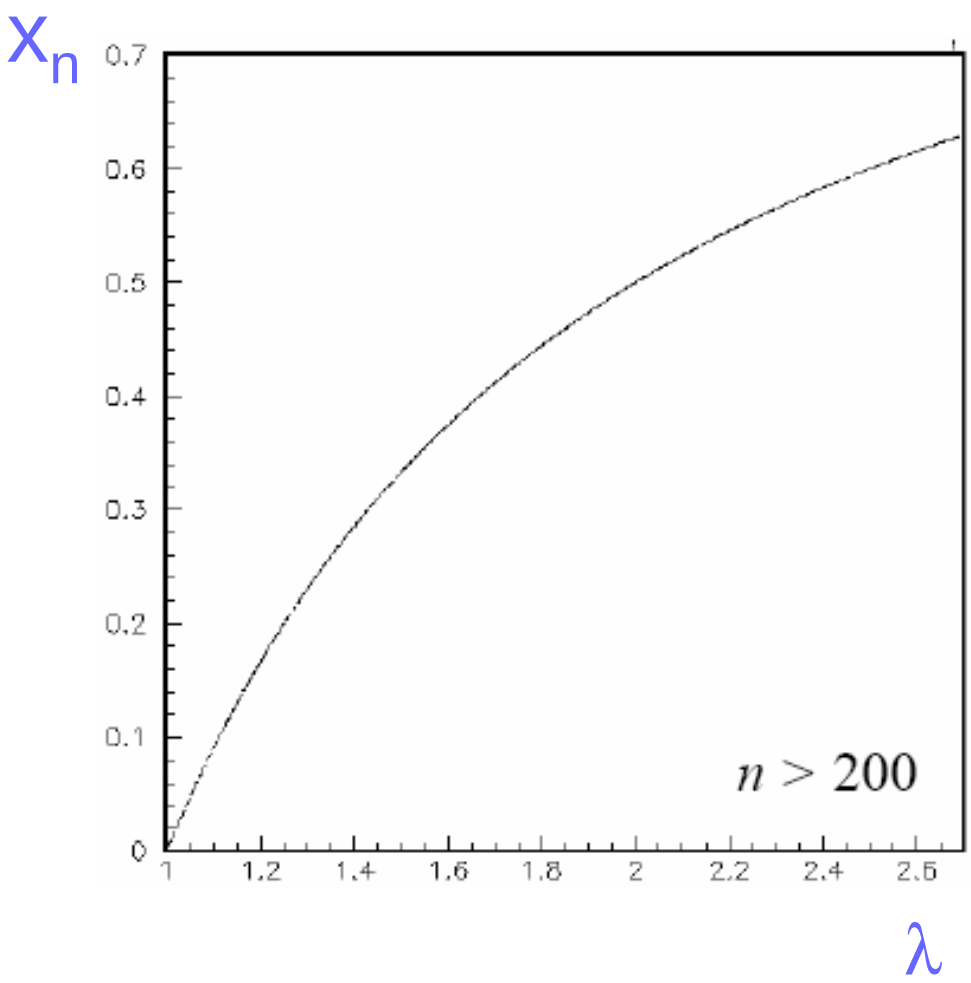
$$x_{n+1} = \lambda x_n (1 - x_n)$$

  - La successione assume comportamenti diversi a seconda del parametro λ: per alcuni valori di questo parametro la sequenza converge ad un valore stabile per $n \to \infty$:

$$x_\infty = \lambda x_\infty (1 - x_\infty) \Rightarrow x_\infty = \frac{\lambda - 1}{\lambda}$$

Stabilità

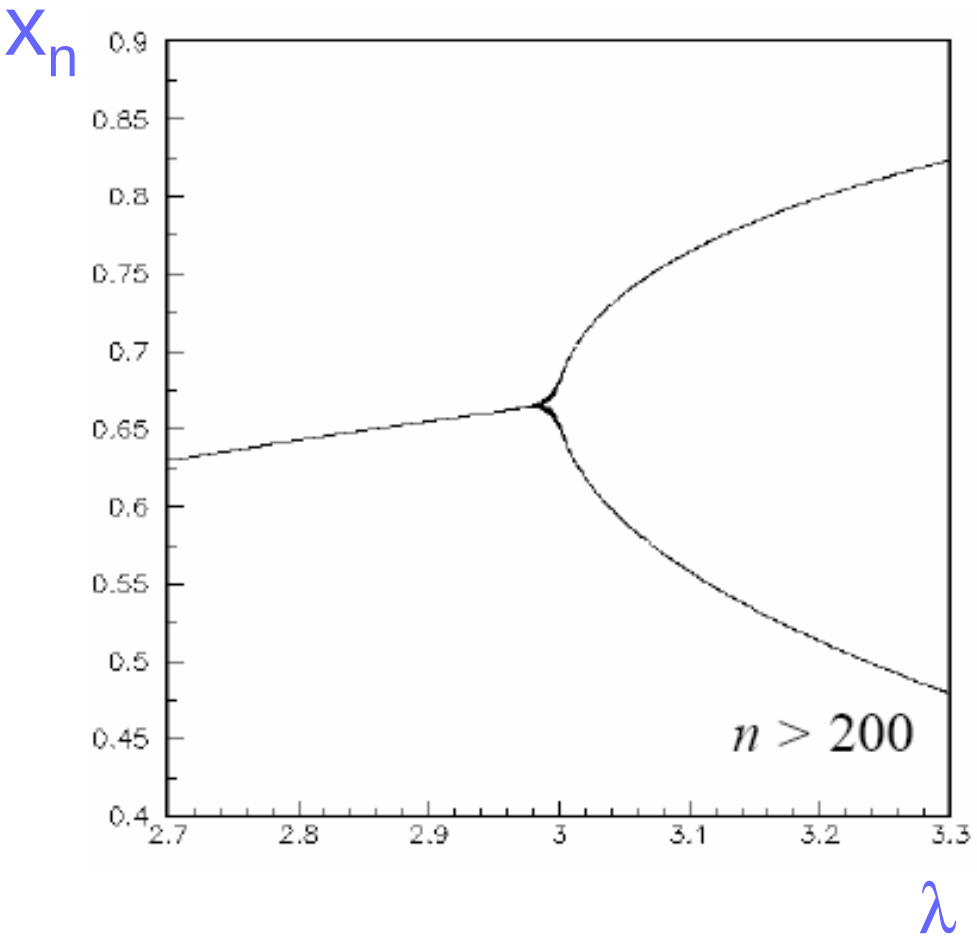
- Per piccoli valori di λ, la successione converge.
- Nell'esempio è stato posto x_0=0.5 e sono mostrate le "storie" delle sequenze per n>200



Biforcazione

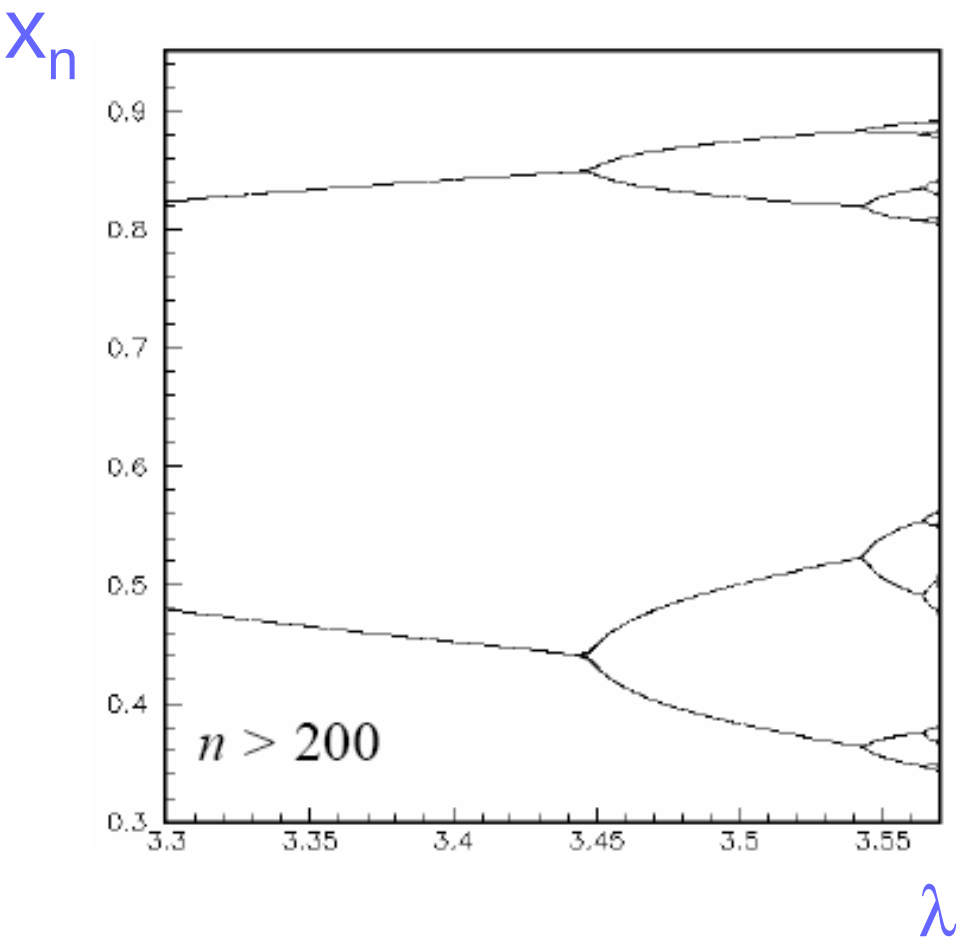
- Per λ>3, la successione non converge, ma oscilla tra due valori:

$$x_a = \lambda\, x_b\, (1 - x_b)$$
$$x_b = \lambda\, x_a\, (1 - x_a)$$



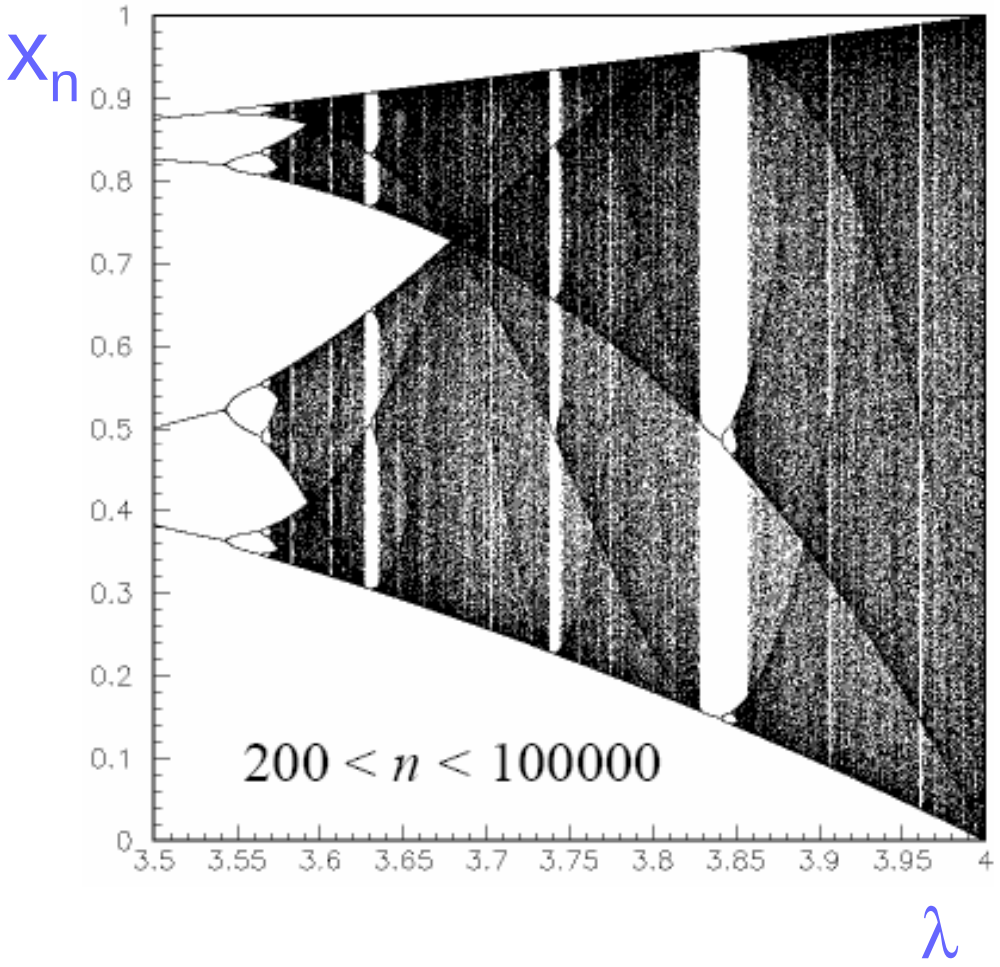
Biforcazione

- Al crescere di λ le biforcazioni aumentano, ovvero il valore assunto dalla variabile oscilla tra 2, 4, 8,16 ... valori



Comportamento caotico

- Per λ ancora maggiori il comportamento diviene completamente caotico. E.g., per λ =4 i valori riempiono densamente l'intervallo [0,1]



Generatori di numeri pseudo-casuali (PRNG)

Tipico schema di un PRNG:

- si fissano k costanti X_0, X_1,..., X_k
- una volta generati *(n-1)* numeri casuali, l'*n*-esimo numeri si genera

$$X_n = f(X_{n-1}, X_{n-2}, \ldots, X_{n-k}),\ n \geq k.$$

Spesso si generano numeri interi oppure bits, successivamente vengono convertiti in numeri floating-point con distribuzione uniforme tra [0,1) → $\mathcal{U}(0,1)$.

Periodo di un RNG. Formalmente:

P è il periodo di un RNG $\Leftrightarrow \exists_{\nu,P} : X_i = X_{i+jP}\ (j = 1, 2, \ldots)\ \forall_{i \geq \nu}$.

dove ν e P sono numeri interi
Ovvero: *da un certo punto ν in poi* i numeri casuali iniziano a ripetersi

Quanto deve essere grande P? Dipende dall'uso: se per risolvere il nostro problema abbiamo bisogno di ***N*** numeri, di solito si richiede $\boldsymbol{P \geq N^2}$

Record: Mersenne Twister (Matsumoto & Nishimura) ha un P ≈ 10^{6000}

M. Matsumoto and T. Nishimura, *Mersenne Twister: A 623-dimensionall y equidistributed uniform pseudo-random number generator*, ACM Transactions on Modeling and Computer Simulation, 8 (1998) 3.

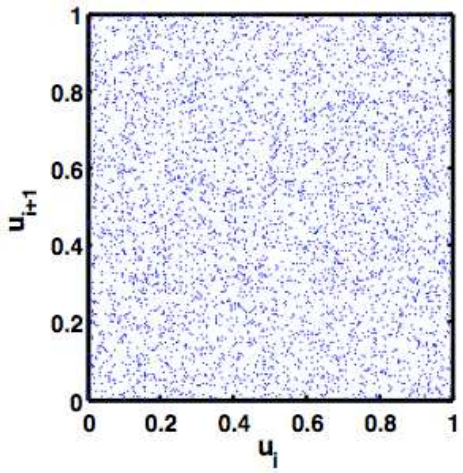
Generatori di numeri pseudo-casuali - cont

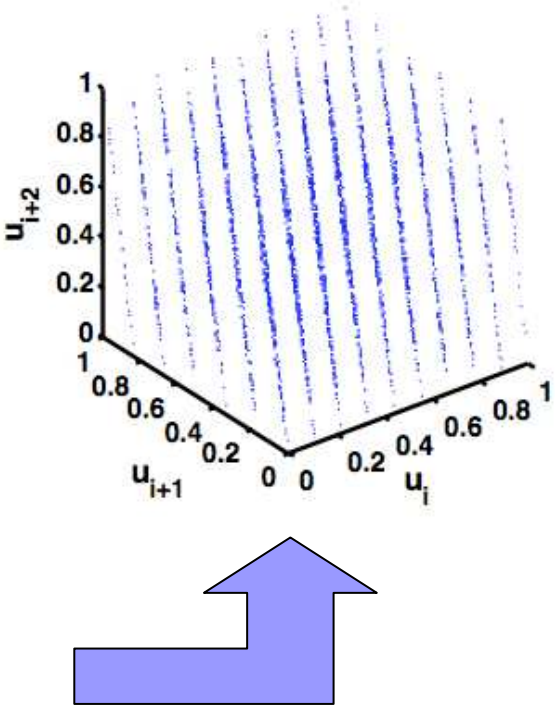
Il primo generatore di numeri pseudo-casuali è stato
il generatore "mid-square" di John von Neumann:

$$X_n = \lfloor X_{n-1}^2 \cdot 10^{-m} \rfloor - \lfloor X_{n-1}^2 \cdot 10^{-3m} \rfloor \cdot 10^{2m}$$

Dove:
- X_i ed m sono interi posivi
- X_0 è una costante
- [...] è l'operatore di troncamento all'intero

Genera sequenze di *2m* cifre di numeri interi
→ sequenze molto corte!





Generatore Lineare Congruenziale (LCG)

Definizione:

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \ldots + a_k X_{n-k} + c) \bmod m,$$

dove a_1, a_2, ..., a_k, c, m sono interi positivi fissati

→ Periodo di LCG: $P \leq m^k - 1$

→ Versione comune (e.g. Pascal o C++)

$$k = 1: \quad X_n = (a X_{n-1} + c) \bmod m$$

→ Drawback: *effetto Marsaglia*, cioè i punti tendono a giacere su iperpiani... vedi ad esempio il generatore *rand()* della libreria *stdlib* del C++

Generatori con Registri a Scorrimento (SRG)

Si applica ai cifre binarie (*bits*) → facile poi passare ad interi

Definizione:

$$b_n = (a_1 b_{n-1} + \ldots + a_k b_{n-k}) \bmod 2,$$

dove $a_1, a_2, \ldots, a_k$ sono costanti binarie (0/1)

→ Facile da implementare: *(a+b) mod 2 = a XOR b*
→ Versione comune (e.g. Pascal o C++)
→ Periodo di un SRG: $P \leq 2^k - 1$
→ Gode della proprietà esaustiva: prima della fine di un ciclo, produce tutte le possibili sequenze di *k* bits
→ Drawback: fallisce con i più moderni test statistici
→ Generatore di Tezuka (1995): combinazione di 3 SR, ha $P \approx 2^{26}$
→ Mersenne Twister: basato su generatori SR, ha $P = 2^{19937} - 1$

Generatori di Fibonacci Ritardati (LFG)

Definizione:

$$X_n = (X_{n-r} \odot X_{n-s}) \bmod m, \quad n \geq r,\ r > s \geq 1,$$

dove $\odot \in \{+, -, \times, \text{xor}\}$.

→ Periodo di un SRG: $P \leq (2^r\text{-}1)m/2$

→ Generatore di Marsaglia, molto popolare (RANMAR):
combinazione di 2 generatori, ha $P \approx 2^{43}$, ottime proprietà statistiche

1° fase $r_i = (r_{i-97} - r_{i-33}) \bmod 2^{24}.$

2° fase $$t_i = \begin{cases} t_{i-1} - 7654321, & \text{if } t_{i-1} - 7654321 \geq 0, \\ t_{i-1} - 7654321 + 2^{24} - 3 & \text{otherwise.} \end{cases}$$

3° fase $s_i = (r_i - t_i) \bmod 2^{24}.$

Generatori SWB (subtract-with-borrow)

- Marsaglia-Zaman (1991)

Scheme: $X_n = (X_{n-r} \ominus X_{n-s}) \bmod m, \quad n \geq r,\ r > s \geq 1,$

where: $x \ominus y \bmod m = \begin{cases} x - y - c + m & \text{and } c = 1 \text{ when } x - y - c < 0, \\ x - y - c & \text{and } c = 0 \text{ when } x - y - c \geq 0, \end{cases}$

initially: $c = 0.$

▶ Drawbacks: Do not satisfy some recent statistical tests!

▷ Generator RCARRY (Marsaglia & Zaman, 1991):
$P \approx 10^{171}$, simple and fast, but does not satisfy some recent tests.

Generatori non-lineari

- Eichenauer & Lehn:

$$X_n = (aX_{n-1}^{-1} + b) \bmod m$$

→ m = numero primo

- Eichenauer-Hermann:

$$X_n = [a(n + n_0) + b]^{-1} \bmod m$$

→ Ogni numero è indipendente dai precedenti

- L. Blum, M. Blum, Shub:

$$X_n = X_{n-1}^2 \bmod m;$$

→ m = prodotto di primi (usato in crittografia)

Test statistici sui PRNG

- Come si fa a verificare la bontà di un certo PRNG?
  → Si formula una proprietà della distribuzione $\mathcal{U}(0,1)$ e si verifica che le sequenze generate dal PRNG possieda questa proprietà
- Problema euristico:
  Si può immaginare un numero infinito di test,
  ma se ne possono testare in numero finito.
  Se il nostro PRNG soddisfa i primi *n* test, non è detto che soddisfi l'*n+1*
- Selezione in negativo: se il nostro generatore "passa" tutti i test, aumenta la nostra confidenza sulla sua bontà
- → ideazione di BATTERIE di test
  - La più famosa batteria è la DIEHARD di Marsaglia

    `http://www.stat.fsu.edu/pub/diehard/`

Ha permesso di eliminare molti PRNG scadenti... persino alcuni generatori fisici!

Test statistici sui PRNG - cont

- Test di ipotesi sulla distribuzione
  - L'intervallo [0,1) viene diviso in k bins. Si generano in totale N numeri casuali e si contano le entries N_i

$$\chi^2 = \sum_{i=1}^{k} \frac{(N_i - N/k)^2}{N/k}$$

Deve seguire una distribuzione chi^2 con un dof. Valida per N/k>10

- Test sulle correlazioni
  - Si studiano in sostanza le distanze tra gli iper-piani

- Gap-test:
  - Si scelgano due numeri α e β con $0 < \alpha < \beta < 1$.
    Si generino poi *r+1* numeri uniformi in [0,1). La probabilità che i primi *r* numeri cadano fuori (α,β) e il numero *r+1* cada dentro (α,β) è:

$$P_r = p(1-p)^r \quad \text{dove} \quad p = \beta - \alpha$$

N.B. è la stessa probabilità di un "random-walk test"

Stato dell'arte

- Non è possibile definire il miglior PRNG, ma esiste un lavoro (hep-lat/9309020) che definisce i criteri che deve possedere un buon PRNG per essere usato in HEP e meccanica statistica
- **Generatore RANLUX:** si basa sul generatore SWB RCARRY of Marsaglia & Zaman, con l'aggiunta di un algoritmo per scartare alcune sequenze di numeri
  - Periodo: $P \approx 10^{171}$
  - Finora non sono state osservati scostamenti dal comporamento casuale!

Generazione di numeri casuali non uniformi

Numeri casuali di distribuzioni non uniformi sono spesso ottenuti a partire da numeri casuali uniformi applicando alcuni metodi di trasformazioni.

I.Metodi generali

1. ***Metodo di trasformazione inversa:***

Sia *U* un numero casuale distribuito uniformemente tra 0 e1 (→ $U \in \mathcal{U}(0,1)$) e *F* una funzione di distribuzione cumulativa continua crescente. Allora la variabile $X = F^{-1}(U)$ è distribuita secondo la funzione di distribuzione cumulativa F(x).

Dimostrazione: $\mathcal{P}[X \leq x] = \mathcal{P}[F^{-1}(U) \leq x] = \mathcal{P}[U \leq F(x)] = F(x).$

Generalizzazione: Se *F* è una qualunque funzione non decrescente, bisogna considerare:

$$X = \inf\{x : U \leq F(x)\}$$

Esempio 1: Distribuzione esponenziale E(0,1) $\boxed{\rho(x) = e^{-x},\ x > 0}$

$\Rightarrow$ cdf: $F(x) = \int_0^x e^{-x'}\,dx' = 1 - e^{-x}.$

sia $r \in \mathcal{U}(0,1)$: $\quad r = F(x) = 1 - e^{-x} \quad \Rightarrow \quad x = -\ln(1-r),$

se $r \in \mathcal{U}(0,1)$ allora $1 - r \in \mathcal{U}(0,1) \quad \Rightarrow \quad \boxed{x = -\ln r}$

Esempio 2: Distribuzione discreta:

$$\mathcal{P}[X = k] = p_k,\ k = 0, 1, \ldots;\ \sum_k p_k = 1.$$

se $r \in \mathcal{U}(0,1)$ allora $\boxed{X = \min\{k : r \leq \sum_{i=0}^{k} p_i\}}$.

Algoritmo in C++:

```
int DiscreteGen(double rn, double* p) {
// Generation of discrete distribution P{X = k} = p[k],
// rn - random number of uniform distribution over (0,1).
  int k = 1;
  double sum = p[0];
  while (sum < rn) sum += p[k++];
  return k - 1;
}
```

Limitazioni del metodo di trasformazione inversa:

- Solitamente richiede che la funzione di distribuzione cumulativa sia conosciuta analiticamente e che sia invertibile analiticamente. Solo un ristretto numero di funzioni soddisfano queste richieste.

- In teoria è possibile integrare numericamente la funzione densità di probabilità e tabulare (con un istogramma) la funzione di distribuzione cumulativa. In questo modo è possibile invertire la funzione di distribuzione cumulativa numericamente (anziché analiticamente). Questo procedimento è spesso molto lento e poco accurato, perciò non molto spesso adottato.

2. ***Metodo del rigetto (hit-or-miss)*** (von Neumann, 1951)***:***

Sia *f(**x**)* la funzione di densità di probabilità, ***x*** può essere una variabile n-dimensionale.

a) determinare una PDF *g(x)* per la quale la generazione di punti casuali è semplice e veloce (nel caso più semplice g(x) = cost) e scegliere una costante *c>0* in modo tale che:

$$f(x) \leq c\, g(x),\ \forall\, x$$

b) Generare punti X secondo la g(x) e un numero casuale $U \in \mathcal{U}(0,1)$

c) Se $c\,U g(X) \leq f(X)$ → *X* viene accettato, altrimenti viene rigettato e si torna al punto b).

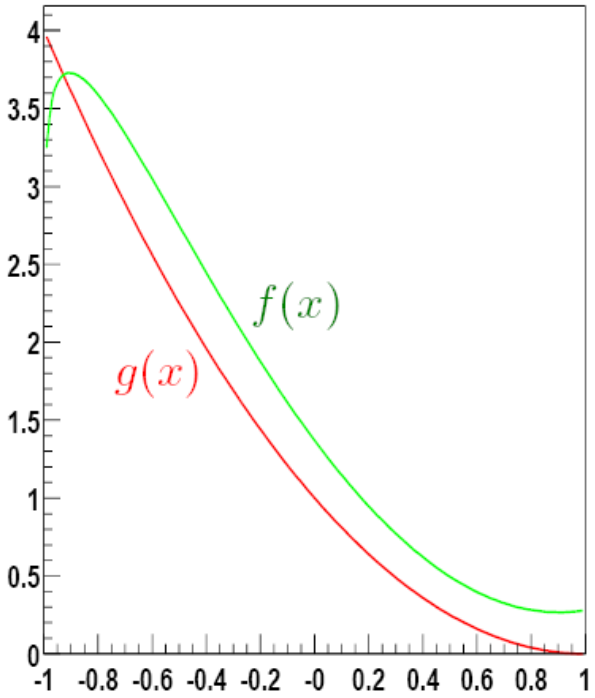
Strada alternativa:

c) Calcolare il peso dell'evento, $w(X) = \frac{f(X)}{g(X)}$ e trovare il massimo peso w_{max} .

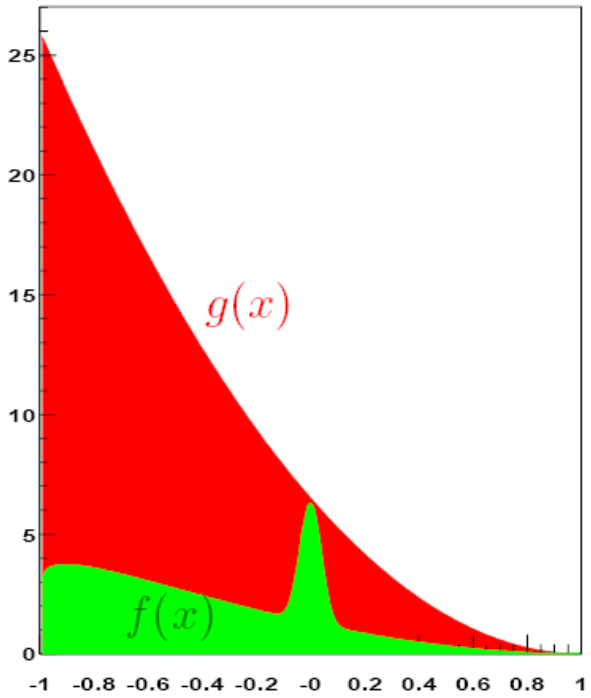
Se $w(X) \geq U w_{max}$ allora l'evento viene accettato, altrimenti viene rigettato e si torna la punto b).

Limitazioni del metodo del rigetto:

- Gli zeri della funzione *g(x)* sono pericolosi se al contempo si ha *f(x)≠0*.
- I picchi della funzione *f(x)* possono causare perdite di efficienza del metodo se non sono “ben approssimati” dalla funzione *g(x).*



zeri della funzione g(x)
→ code infinite nel peso!



Presenza di un picco nella f(x)
→ alto rate di rigetto

Distribuzione normale *N(0,1)*

pdf: $\rho(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

a) The method based on the Central Limit Theorem, see Lecture 1.

→ **Drawback:** Lack of inifinite tails of Gaussian distribution!

b) Inverse transform method:

▷ The cumulative function of one-dimensional Gaussian distribution cannot be expressed in terms of elementary functions!

▶ Go to $\mathbf{2}$ dimensions:

Box-Müller method

pdf: $\varrho(x) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$.

and make transformation to polar coordinates, then invert the cumulative function:

$$x = \sqrt{-2\ln r_1}\,\cos(2\pi r_2), \qquad y = \sqrt{-2\ln r_1}\,\sin(2\pi r_2)$$

where $r_1, r_2 \in \mathcal{U}(0,1)$.

⇒ General Gaussian distribution $N(\mu,\sigma)$: $x' = \mu + \sigma x$.

Distribuzione esponenziale

pdf: $\rho(x) = \frac{1}{\lambda} e^{-\frac{x-\theta}{\lambda}}, \qquad x \geq \theta\,.$

▶ Transformation: $x \to x' = \frac{x-\theta}{\lambda} \Rightarrow E(\theta,\lambda) \to E(0,1)$: $\rho(x') = e^{-x'}$, $x' \geq 0$.

▷ Inverse transform method: $x' = -\ln r, \;\; r \in \mathcal{U}(0,1)$.

Distribuzione di Breit-Wigner (Cauchy)

pdf: $\rho(x) = \frac{\lambda}{\pi} \frac{1}{(x-\theta)^2 + \lambda^2}, \quad -\infty < x < +\infty\,.$

▶ Transformation: $x \to x' = \frac{x-\theta}{\lambda} \Rightarrow C(\theta,\lambda) \to C(0,1)$: $\rho(x') = \frac{1}{\pi} \frac{1}{1+x'^2}$.

▷ Inverse transform method: $x' = \tan(\pi[r - \frac{1}{2}]), \;\; r \in \mathcal{U}(0,1)$.

Distribuzione legge di potenza

pdfs: $\rho_1(x) = nx^{n-1}, \;\; \rho_2(x) = n(1-x)^{n-1}, \;\; 0 \leq x \leq 1, \; n = 1, 2, \ldots$.

a) Inverse transform method: $x = r^{1/n}, \;\; r \in \mathcal{U}(0,1)$.

b) Let: $r_1, r_2, \ldots, r_n \in \mathcal{U}(0,1)$ – independent random numbers.

$x = \max\{r_1, r_2, \ldots, r_n\}$ – is distributed according to $\rho_1(x)$,

$x = \min\{r_1, r_2, \ldots, r_n\}$ – is distributed according to $\rho_2(x)$.

Distribuzione binomiale *b(n,p)*

pdf: $\mathcal{P}[X = m] = \binom{n}{m} p^m (1-p)^{n-m}, \quad 0 < p < 1, \; m = 0, 1, \ldots, n.$

▶ Algoritm 1: Rejection method

```
long BinomialGen1(long n, double p) {
  double r; long m = 0;
  for (long i = 0; i < n; i++){
    r = RNG();          // random number generation
    if (r <= p) m++; }
  return m;
} //
```

Drawback: Many random numbers needed!

▶ Algoritm 2: Only one random number needed!

```
long BinomialGen2(long n, double p, double r) {
// r - random number of uniform distribution over (0,1)
  long m = 0;
  for (long i = 0; i < n; i++)
    if (r <= p) { m++; r /= p;}
    else r = (1 - r)/(1 - p);
  return m;
} //
```

Drawback: More floating-point operations!

Distribuzione di Poisson $P(\mu)$

pdf: $\mathcal{P}[X=k] = \frac{\mu^k}{k!}\, e^{-\mu}, \quad k = 0, 1, \ldots; \qquad E(k) = V(k) = \mu\,.$

- Algoritm 1: Popular in HEP applications

```
int PoissonGen1(double mu) {
  int k = -1;
  double r, s = 1.0, q = exp(-mu);
  while (s > q) { r = RNG(); s *= r; k++; }
  return k;
} //Many random numbers needed, but can be used e.g. to conctruct particles 4-momenta
```

- Algoritm 2: Inverse transform method

```
int PoissonGen2(double mu, double r) {
// r - random number of uniform distribution over (0,1)
  int k = 0;
  double q = exp(-mu), s = q, p = q;
  while (r > s) { k++; p *= mu/k; s += p; }
  return k;
}// Only one random number needed, but more floating-point operations!
```