



Metodi Monte Carlo

1) Generazione di distribuzioni

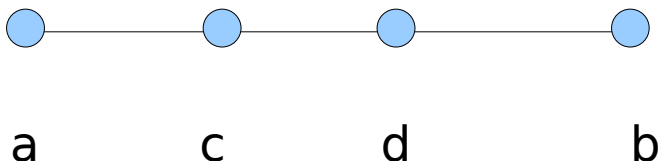
Probabilità e misura

Supponiamo di avere una variabile continua: x

Intendiamo per variabile continua una qualunque variabile il cui dominio sia in corrispondenza biunivoca con i numeri reali. (Variabili continue non banali: funzioni, operatori, distribuzioni)

Supponiamo $x \in [a, b]$ e supponiamo che la probabilità che x abbia un qualunque valore in $[a, b]$ sia indipendente da x . Poiché in $[a, b]$ ci sono infiniti numeri, la probabilità di trovare un particolare valore di x sarà $1/\text{infinito} = 0$.

Ci possiamo chiedere però qual è la probabilità di trovare x in un sotto intervallo $[c, d]$: $a < c < d < b$.



$$P(c < x < d) = \frac{m(c, d)}{m(a, b)}$$

$m(c, d)$

misura del segmento c, d

In spazi più complessi può essere non banale.

Area in 2 dimensioni, volume in 3, ipervolume in n dimensioni.

Definizione di p.d.f

$$P(x_0 < x < x_0 + dx) = F(x_0, x_0 + dx) = F(x_0, x_0) + \left. \frac{dF}{dx} \right|_{x_0} dx = \left. \frac{dF}{dx} \right|_{x_0} dx$$



0 Probabilità di trovare x_0

$$\mu(x) = \frac{dF}{dx}$$

Probability density function (p.d.f)

$$F(x)$$

Funzione cumulativa di probabilità

$$P(c < x < d) = \int_c^d \mu(x) dx$$

$$P(a < x < b) = 1 \Leftrightarrow \int_a^b \mu(x) dx = 1$$

Esempio: **Distribuzione piatta** $\mu(x) = h$

$$\int_a^b h dx = 1 \Leftrightarrow h(b-a) = 1 \Leftrightarrow h = \frac{1}{(b-a)}$$

In generale gli estremi a, b possono assumere anche i valori $\pm \infty$

Proprietà della pdf.

$\mu(x)$ è definita positiva $\mu(x) > 0$

$\mu(x)$ è integrabile su tutto l'intervallo $[a,b]$;

$\mu(x)$ può anche non essere una funzione, se è una funzione può essere discontinua, indefinita e divergente in un insieme di punti a misura nulla.

Esempio di funzione valida in un intervallo

[a,b]:
$$\frac{1}{\sqrt{|x-c|}}$$

Esempio di funzione valida in un intervallo

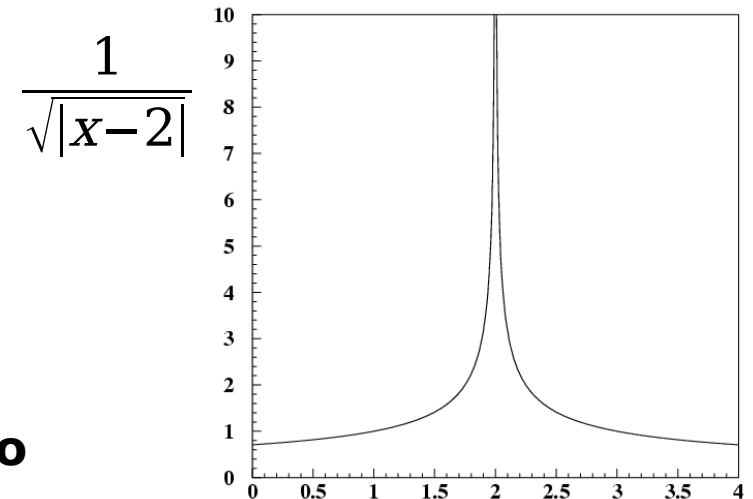
[a, +∞] (a > 0):

$$e^{-x}, \frac{1}{x^2}$$

Esempio di funzione non valida in un intervallo

[a, +∞] (a > 0):

h = costante



Esercizio: determinare la normalizzazione

Funzione gradino.

Consideriamo la funzione cumulativa di probabilità definita in $[a,b]$:

$$P(x): \begin{cases} 1 & x > c \\ 0 & x < c \end{cases}$$

Determiniamo $\mu(x)$, può essere una funzione?

$$\int_0^x \mu(x') dx' = 0 \text{ per } x < c \Rightarrow \mu(x) = 0$$

$$\int_0^c \mu(x) dx + \int_c^x \mu(x') dx' = \int_c^x \mu(x') dx' = 1 \text{ per } x > c$$

$$\int_c^x \mu(x') dx' = 1 \Rightarrow \frac{d}{dx} \int_c^x \mu(x') dx' = 0 \Rightarrow \mu(x) = 0 \text{ per } x > c$$

μ è nullo tranne in c che è un insieme di misura nulla per cui:

$$\left[\int_0^x \mu(x') dx' = 0 \quad \forall x \right] \neq P(x) \quad \mu(x) = \delta(x-c) \quad \text{distribuzione}$$

Funzioni a valori discreti.

La nostra definizione di p.d.f si applica strettamente a distribuzioni di una variabile continua.

Tuttavia moltissimi problemi di probabilità si risolvono con variabili discrete.
Es. Poisson, binomiali

$$P(n) = \frac{\mu^n}{n!} e^{-\mu}$$

E' una funzione di distribuzioni a valori interi, possiamo scriverla sotto forma di densità?

$$\mu_P(x) = \sum_{n=0}^{\infty} \frac{\mu^n}{n!} e^{-\mu} \delta(x-n)$$

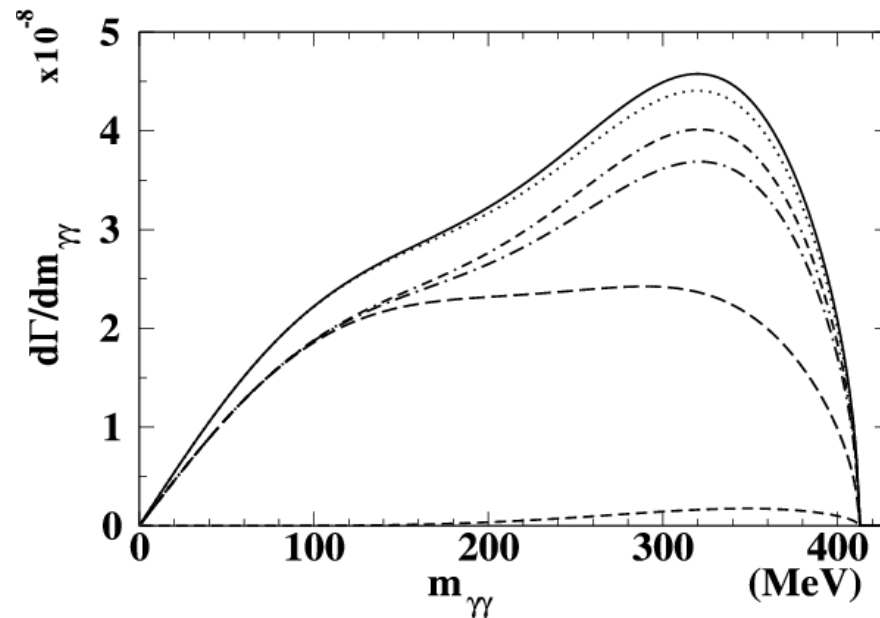
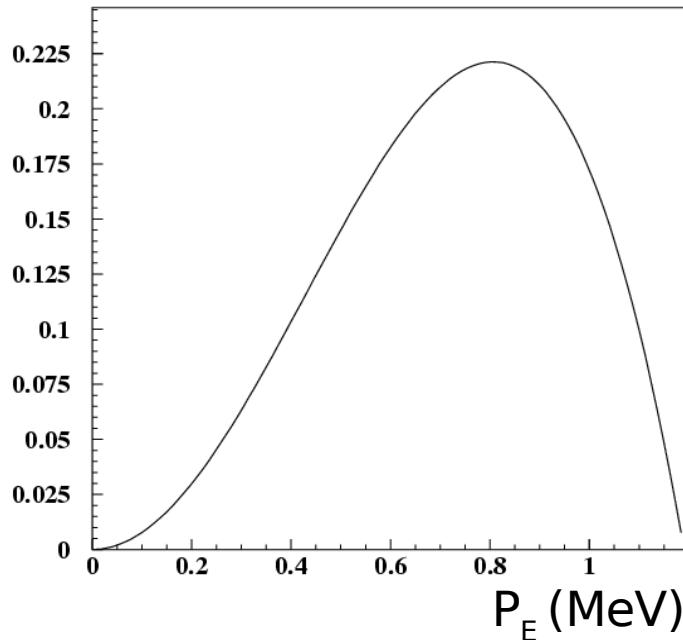
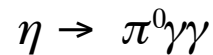
infatti:

$$P(N) = \int_{N-\Delta}^{N+\Delta} \mu_P(x) dx = \sum_{n=0}^{\infty} \frac{\mu^n}{n!} e^{-\mu} \int_{N-\Delta}^{N+\Delta} \delta(x-n) dx = \sum_{n=0}^{\infty} \frac{\mu^n}{n!} e^{-\mu} \delta_{Nn} = \frac{\mu^N}{N!} e^{-\mu}$$

Problema: generazione di distribuzioni.

La meccanica quantistica ci dice che le osservabili fisiche sono distribuzioni di probabilità. A tutti i processi fisici possiamo associare una data P.D.F.

Es. Decadimento a tre corpi:



Misure di apparato.

Il processo di misura fornisce anch'esso delle distribuzioni a causa degli errori statistici sulle misure.

Simulazione:

Processo fisico

Fisica delle particelle
(produzione e decadimento)

Interazione con il rivelatore
(Interazione radiazione- materia)

Risposta del rivelatore

Nella sua forma più elementare può ridursi alla simulazione della curva di risposta e risoluzione.

Il metodo di base è lo stesso:

SIMULAZIONE DI FUNZIONI DI DISTRIBUZIONE

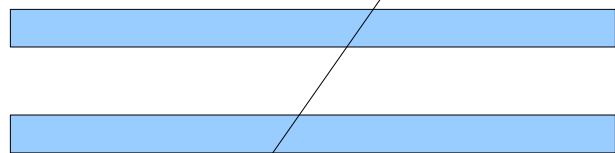
Generazioni di distribuzioni piatte.

Vedremo in seguito che possiamo generare una distribuzione arbitraria a partire da una distribuzione piatta in un intervallo $[0,1]$.

Vedremo anche che possiamo generare distribuzioni multidimensionali arbitrarie a partire da distribuzioni monodimensionali.

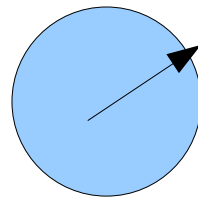
E' di importanza primaria la generazione di variabili uniformemente distribuite in un dato intervallo.

Come possiamo generarle?



cosmico

Scintillatori in coincidenza



Cilindro rotante a
velocità costante

quando passa un
cosmico leggo la
posizione della lancetta

Otengo una distribuzione piatta in $[0, 2\pi]$

Vantaggi e svantaggi

Vantaggi

- La distribuzione è piatta e realmente imprevedibile;

Svantaggi

- La generazione è lenta (Hz), rispetto alla velocità di calcolo (Ghz);
- Dobbiamo montare un apparato pesante e ingombrante ad ogni computer;
- Non possiamo riprodurre la stessa sequenza di numeri casuali (importante per debugging)

Curiosità

- random.org genera numeri casuali registrando rumore atmosferico nelle onde radio (registrato a 8 KHz).

Numeri pseudo-casuali

- Un computer non può produrre numeri realmente casuali:
 - Qualunque algoritmo noi usiamo utilizza un criterio, delle regole per generare i numeri casuali;
 - Se noi riavviamo la macchina nelle stesse condizioni della prima generazione l'algoritmo, qualunque esso sia, partirà dalla stessa condizione;
 - L'idea è quella di trovare un metodo che ci permette di generare una sequenza di numeri in un dato intervallo distribuiti in modo piatto;
- Parto dal numero a, determino b, quindi c e così' via:

Von Newman:

```
1111
fai il quadrato 1234321
completa a 8 01234321
prendo le 4 cifre centrali 2343
rifaccio il calcolo.
```

In questo modo ottengo una sequenza di numeri pseudo-casuali.

Caratteristiche di un generatore pseudo casuale.



- 1) Periodo: il numero minimo di step oltre il quale la sequenza si ripete uguale a se stessa;
- 2) Casualità: la distribuzione è realmente piatta?
- 3) correlazione: se genero due sequenze di numeri, e ne studio la correlazione, tale correlazione è nulla? E tre? E N?
- 4) Predicibilità: dati i primi N numeri, posso predire l' $N+1$?

Rand() Per l'implementazione GNU conosciamo solo il periodo 2147483647. Pochissimo (la simulazione dei processi di fisica in un detector richiede milioni di numeri casuali), se vogliamo generare milioni di eventi abbiamo rapidamente problemi (iniziamo a generare la stessa sequenza di eventi).

Esempio



$U_i = (aU_{i-1} + c) \bmod m$ mod m significa resto della divisione con m.

In questo modo generiamo numeri pseudo-casuali da 0 a m-1.

Non uniforme in n dimensioni

Generatori in root

- 1) Trandom periodo 10^9 problemi (correlazione tra i bit generati.), 34 ns/call
- 2) Trandom1(UInt_t seed, Int_t lux) (RANLUX delle CERNLIB)
lux (luxury level) livello di accuratezza nella generazione casuale
0 periodo lungo ma forti correlazioni, 3 qualunque correlazione può essere difficilmente osservata, 4 tutti e 24 bits sono completamente caotici periodo 10^{171} 242 ns/call
- 3) Trandom2 37 ns/call 10^{26} è il periodo
- 4) Trandom3 45 ns/call 10^{6000} è il periodo
Mersenne Twister: uniformemente distribuito in 623 dimensioni.
Default di root:
gRandom (punta a Trandom3)
gRandom->rndm(); genera un numero casuale tra 0 e 1 Double_t 8 bytes
gRandom->SetSeed(UInt_t) imposta il seed number

Esercizio: scriviamo una macro che genera 1000 numeri casuali e li inserisce in un istogramma.

Generazione di funzioni di distribuzione

La generazione di funzioni di distribuzioni arbitrarie può essere ottenuta a partire da funzioni di distribuzioni piatte in un intervallo finito. La maggior parte dei generatori di numeri casuali generano distribuzioni piatte nell'intervallo $[0,1]$.

Problema: Generazione di distribuzioni arbitrarie in intervalli arbitrari.

Supponiamo di avere una variabile x , generata piatta in $[a,b]$.

Consideriamo una funzione biunivoca $g(x)$ e indichiamo con $y = g(x)$.
Se x è una variabile casuale anche y sarà una variabile casuale.
Domanda: Qual è la distribuzione di y ?

Poiché g è biunivoca $P(y < y' < y + dy) = P(x < x' < x + dx)$

Indichiamo con $\mu_y(y)$ la funzione di distribuzione di y , e $\mu_x(x)$ la funzione di distribuzione di x . Otteniamo ($x = g^{-1}(y)$):

$$\mu_y(y) dy = \mu_x(x) dx \Leftrightarrow \mu_y(y) g'(x) dx = \mu_x(x) dx \Leftrightarrow \mu_y(y) = \frac{\mu_x(g^{-1}(y))}{g'(g^{-1}(y))}$$

Il dominio di y sarà ovviamente $[g(a),g(b)]$

Generazione di funzioni piatte in intervallo arbitrario.

Supponiamo di voler generare una distribuzione piatta in $[a,b]$ a partire da una piatta in $[0,1]$

Costruiamo la funzione $y = (b-a)*x+a$

$$g'(x) = b-a \quad \mu_x(x) = 1$$

$$\mu(y) = \frac{\mu_x(g^{-1}(y))}{g'(g^{-1}(y))} = \frac{1}{b-a}$$

Con questa trasformazione lineare otteniamo la distribuzione voluta.

Per risolvere il problema generale si ragiona in modo inverso. Se io voglio ottenere una certa distribuzione $\mu(x)$, quale variabile $y=g(x)$ devo generare piatta in modo che $x=g^{-1}(y)$ è distribuito secondo $\mu(x)$?

$$\mu_y(y) g'(x) = \mu_x(x) \quad \text{y lo voglio generare piatto in } [0,1]. \quad \text{Quindi } \mu_y(y) = 1$$

$$g'(x) = \mu_x(x) \Rightarrow g(x) = \int \mu_x(x) dx \quad \text{Generiamo y in } [0,1], \text{ calcoliamo } g^{-1}(y)$$

Esempio

Generare una distribuzione del tipo $e^{-(t/\tau)}$.

Distribuzione importante, distribuzione temporale dei decadimenti di particelle con vita media τ .

1) Normalizziamo nell'intervallo $[0, +\infty]$

$$A \int_0^{+\infty} e^{-\frac{t}{\tau}} dt = A\tau = 1 \Leftrightarrow A = \frac{1}{\tau} \quad \mu(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}}$$

2) Calcoliamo:

$$g'(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}} \Rightarrow g(t) = \frac{1}{\tau} \int e^{-\frac{t}{\tau}} dt = -e^{-\frac{t}{\tau}} + C$$

3) Determiniamo C: $g(0) = C-1$

$$g(+\infty) = C$$

Scegliendo $C=1$ mappiamo $0 \rightarrow 0 \quad +\infty \rightarrow 1$

$$y \in [0,1] \text{ OK.} \quad y = -e^{-\frac{t}{\tau}} + 1 \Leftrightarrow t = -\tau \log(1-y)$$

Esercizio: Generare una Breit and Wigner:
$$\mu(m) = \frac{1}{(m - m_0)^2 + \Gamma^2}$$

Metodo generale

Una particolare primitiva di $g'(x) = \mu_x(x)$ è $g(x) = \int_a^x \mu_x(x') dx'$

$g(a) = 0$ $g(b) = 1$ perché $\mu(x)$ è normalizzata.

1) Determiniamo numericamente $g(x)$ tramite tabulazione (es. cernlib 100 punti.)

2) Invertiamo numericamente $g(x)$

basta risolvere l'equazione $g(x) - y = 0$ con un metodo numerico qualunque.

Bisezione Newton

Se non sapete di cosa parlo ve lo spiego.

Pregi: Veloce, $g(x)$ possiamo valutarla una sola volta, per ogni estrazione di y otteniamo un valore di x .

Difetti: non è utilizzabile per $a, b \infty$ inadeguata per funzioni discontinue o divergenti. La discontinuità viene saltata dallo step di tabulazione.

Bisogna scegliere in modo adeguato lo step di tabulazione.

Migliorabile con step variabile $|f'(x)| * \text{step} < \text{delta}$

In tal modo usiamo step piccoli nei punti di grande variazione e step grandi in regioni smooth.

Distribuzioni multidimensionali.

$$P(x < x' < x + dx, y < y' < y + dy) = h(x,y)dxdy$$

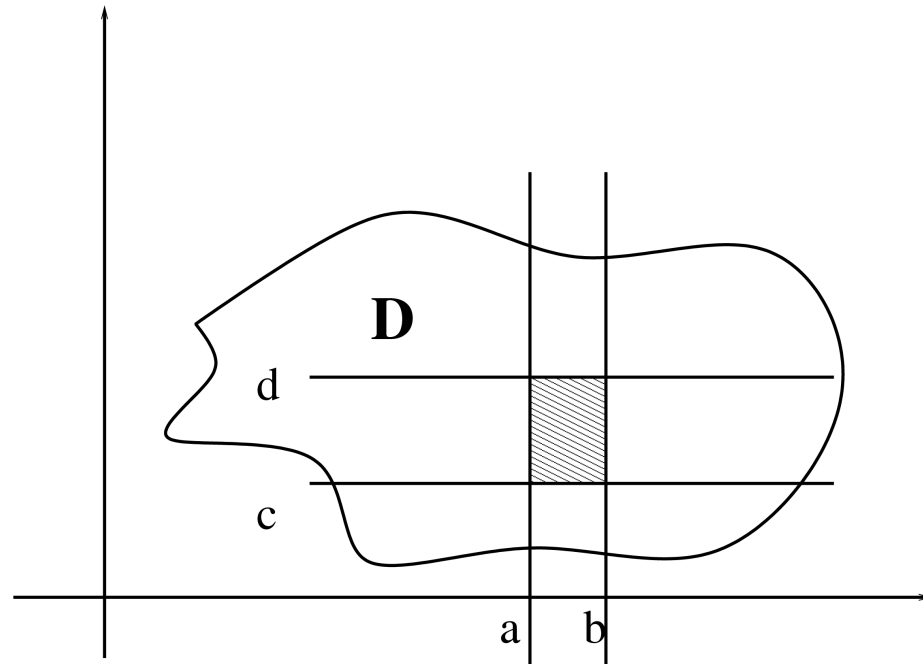
$$P(a < x < b, c < y < d) = \int_a^b \int_c^d h(x, y) dxdy$$

Data una distribuzione bidimensionale mi posso chiedere come si distribuisce una sola variabile indipendentemente dalla distribuzione dell'altra.

$$\mu(x) = \int_D h(x, y) dy$$

Supponiamo di generare x, y secondo $h(x, y)$. Se istogrammo x senza porre alcuna condizione su y ottengo $\mu(x)$.

Su questa osservazione è basato il metodo di hit or miss.



Il metodo hit or miss.

Supponiamo di voler generare secondo una distribuzione $\mu(x)$ definita in un intervallo $[a,b]$.

Consideriamo la seguente funzione definita in $[a,b] \times [0, \max]$.

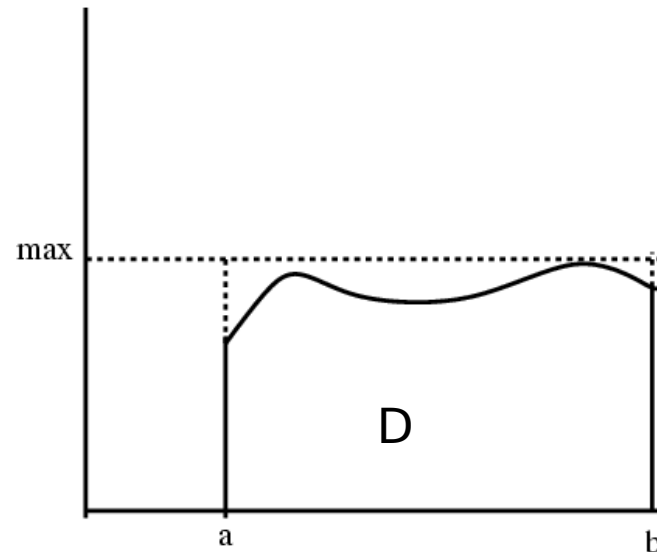
$$\begin{aligned} h(x,y) &= 1 & y < \phi(x) \\ h(x,y) &= 0 & y > \phi(x) \end{aligned}$$

x, y posso generarli secondo h in un modo molto semplice, genero x piatto in $[a,b]$ e y piatto in $[0, \max]$, se $y < \phi(x)$ conservo x, y altrimenti rigetto x, y .

Se ignoro y come si distribuisce x ?

$$\mu(x) = \int_0^{\max} h(x, y) dy = \int_0^{\phi(x)} dy = \phi(x)$$

Ossia $\mu(x)$ si distribuisce esattamente come la funzione $\phi(x)$.



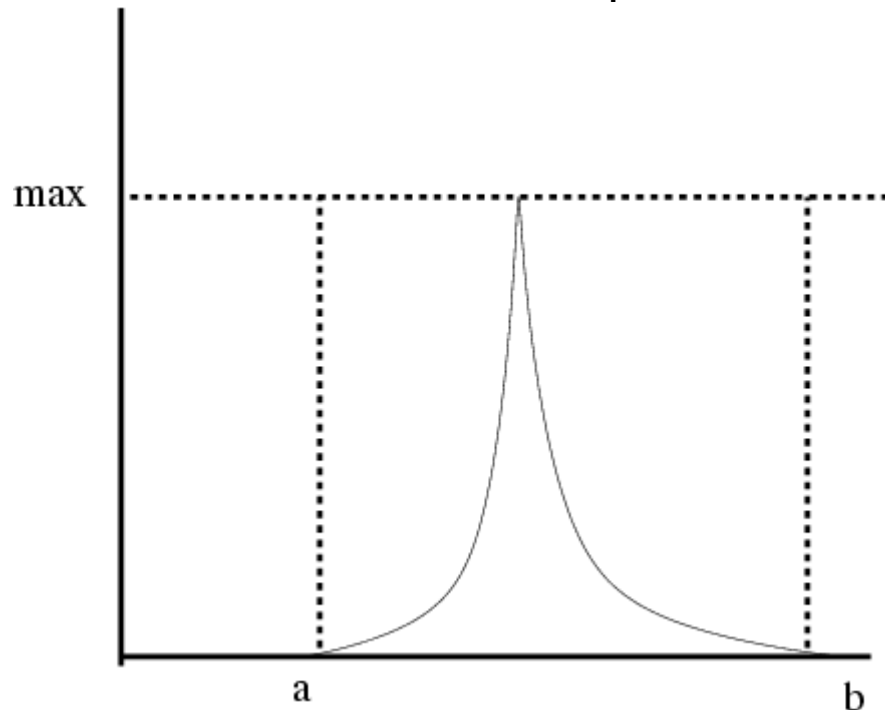
Il metodo hit or miss.

Vantaggi:

- Semplicissimo da implementare, non dobbiamo fare integrali né trovare zeri di una funzione, ci basta un $\text{if}(y < f(x))$.
- La $\mu(x)$ è determinata con la precisione della macchina, non ci sono passi di integrazione o di tabulazione (se $\mu(x)$ è nota analiticamente)

Svantaggi:

- Terribilmente lento per funzioni piccate $1/x^2$, Briet and Wigner e così via.



Il numero di tentativi che rigettiamo è proporzionale al rapporto tra le aree della figura e del quadrato che la contiene.

esercizio

Generare con il metodo hit or miss una distribuzione del tipo:

$\log(x) + \exp(-\ln^2 x)$ in un intervallo opportuno di x .

Trasformazione tra distribuzioni multidimensionali.

Consideriamo una trasformazione del piano:

$$G = \begin{cases} x = g(u, v) \\ y = f(u, v) \end{cases}$$

La trasformazione è non singolare se risulta:

$$\begin{vmatrix} \frac{\partial g}{\partial u} & \frac{\partial g}{\partial v} \\ \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} \end{vmatrix} = |J| \neq 0$$

Poiché G è biunivoca

$$P(x < x' < x + dx, y < y' < y' + dy) = P(u < u' < u + du, v < v' < v + dv)$$

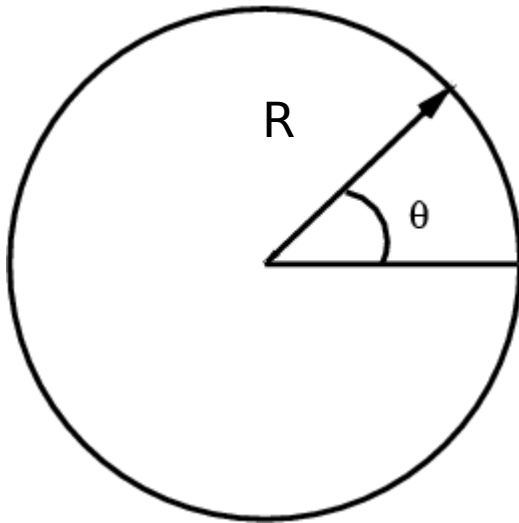
$$\mu(x, y) dx dy = \nu(u, v) du dv$$

$$\longrightarrow \mu(x, y) |J| du dv = \nu(u, v) du dv \Leftrightarrow \mu(x, y) |J| = \nu(u, v)$$

$$dx dy = |J| du dv$$

esercizio

Generare una distribuzione uniforme in un cerchio.



Si potrebbe pensare...

1) genero r piatto in $0 - R$, e θ in $0 - 2\pi$
provare

2)

$$\mu(x, y) |J| = v(u, v) \Leftrightarrow \frac{1}{\text{mis}_C} r = v(r, \theta)$$

$v(r, \theta)$ è fattorizzata in $g(r) * h(\theta)$

Generazione di una distribuzione gaussiana.

$$g(x) = \int \mu_x(x) = \frac{1}{\sqrt{2\pi\sigma}} \int e^{-\frac{x^2}{2\sigma^2}} dx$$

Non integrabile analiticamente....

Definiamo la funzione:

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Troviamo la funzione di distribuzione in coordinate polari: $(u, v) = (r, \theta)$

$$\mu(x, y) |J| = \nu(u, v) \Leftrightarrow \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} r = \nu(r, \theta) \Leftrightarrow \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} r = \nu(r, \theta)$$

$$g(r) = \frac{1}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} r \qquad f(\theta) = \frac{1}{2\pi}$$

Verificare che le distribuzioni sono gaussiane e la correlazione è nulla.

Generazione di distribuzioni multivariate

Se vogliamo generare un insieme di variabili distribuite gaussianamente e scorrelate possiamo generarle una indipendente dall'altra. $x_1, x_2, x_3, \dots, x_n$.

E se vogliamo generare N variabili con una fissata matrice di covarianza?

Se la distribuzione $h(x,y)$ è fattorizzabile in $g(x)*f(y)$ ed il dominio D è esprimibile come $D_x \times D_y$ allora le variabili x,y sono scorrelate. Infatti:

$$\begin{aligned} \text{cov}(x, y) &= \int_D (x - \bar{x})(y - \bar{y}) h(x, y) dx dy = \int_{D_x} (x - \bar{x}) g(x) dx \int_{D_y} (y - \bar{y}) f(y) dy \\ &= \left(\int_{D_x} xg(x) dx - \bar{x} \int_{D_x} g(x) dx \right) \left(\int_{D_y} yf(y) dy - \bar{y} \int_{D_y} f(y) dy \right) = (\bar{x} - \bar{x})(\bar{y} - \bar{y}) = 0 \end{aligned}$$

Consideriamo una matrice di covarianza NxN, diagonale:

$$V = \begin{pmatrix} \sigma_1^2 & 0 & \dots \\ 0 & \sigma_2^2 & \dots \\ 0 & 0 & \dots \end{pmatrix}$$

Se indichiamo con V_{ij} un generico elemento possiamo scrivere:

$$V_{ij} = \sigma_i^2 \delta_{ij} \quad \text{Indicando con } \chi^2 \text{ la quantità: } \chi^2 = \frac{\sum_i^N (x_i - \bar{x}_i)^2}{\sigma_i^2}$$

La distribuzione: $N e^{-\frac{\chi^2}{2}}$ Rappresenta N variabili scorrelate distribuite in modo gaussiano.

Dimostrazione

1) Il dominio è $R^N = R \times R \times R \dots \times R$ N volte

$$2) N e^{-\frac{\chi^2}{2}} = N e^{-\frac{\sum_i (x_i - \bar{x}_i)^2}{2\sigma_i^2}} = N \prod_i e^{-\frac{(x_i - \bar{x}_i)^2}{2\sigma_i^2}}$$

Quindi le variabili sono scorrelate. Inoltre:

$$\langle x_i \rangle = \int_{R^N} x_i N \prod_j e^{-\frac{(x_j - \bar{x}_j)^2}{2\sigma_j^2}} \prod_k dx_k = N \prod_{j \neq i} \int_R e^{-\frac{(x_j - \bar{x}_j)^2}{2\sigma_j^2}} dx_j \int_R x_i e^{-\frac{(x_i - \bar{x}_i)^2}{2\sigma_i^2}} dx_i = \bar{x}_i$$

Allo stesso modo si dimostra che:

$$\langle x_i^2 - \bar{x}_i^2 \rangle = \sigma_i^2$$

Come generalizziamo al caso di una matrice di covarianza completa?

Osserviamo che:

$$\chi^2 = \frac{\sum_i (x_i - \bar{x}_i)^2}{\sigma_i^2} = \sum_{i=1}^N (x_i - \bar{x}_i) \frac{1}{\sigma_i^2} (x_i - \bar{x}_i) = \sum_{i,j=1}^N (x_i - \bar{x}_i) (x_j - \bar{x}_j) \frac{1}{\sigma_i^2} \delta_{ij} = \sum_{i,j=1}^N (x_i - \bar{x}_i) (x_j - \bar{x}_j) V_{ij}^{-1}$$

Multivariata arbitraria

Ci possiamo aspettare che tale distribuzione sia data da: $Ne^{-\frac{x^2}{2}}$

$$\text{con } x^2 = \sum_{i,j=1}^N (x_i - \bar{x}_i)(x_j - \bar{x}_j) V_{ij}^{-1} = (\vec{x} - \vec{\bar{x}})^T V^{-1} (\vec{x} - \vec{\bar{x}})$$

Dobbiamo dimostrare:

$$1) \langle \mathbf{X}_i \rangle = \bar{\mathbf{X}}_i$$

$$2) \langle \mathbf{X}_i^2 - \langle \mathbf{X}_i \rangle^2 \rangle = V_{ii}$$

$$3) \langle (\mathbf{X}_i - \langle \mathbf{X}_i \rangle)(\mathbf{X}_j - \langle \mathbf{X}_j \rangle) \rangle = V_{ij}$$

Osserviamo che essendo V una matrice simmetrica reale definita positiva ammette un insieme di N autovalori reali positivi. E che la matrice di trasformazione $x' = Rx$, R è ortogonale e a determinante unitario. Sotto la trasformazione R abbiamo:

$$x^2 = (\vec{x} - \vec{\bar{x}})^T V^{-1} (\vec{x} - \vec{\bar{x}}) = \Delta \vec{x}^T V^{-1} \Delta \vec{x} = (R^T \Delta \vec{x}')^T V^{-1} (R^T \Delta \vec{x}') = (\Delta \vec{x}')^T R V^{-1} R^T \Delta \vec{x}'$$

$$(RV^{-1}R^T) = (RVR^T)^{-1} = \begin{pmatrix} \frac{1}{\lambda_1^2} & 0 & \dots \\ 0 & \frac{1}{\lambda_2^2} & \dots \\ 0 & 0 & \dots \end{pmatrix}$$

RVR^T è infatti la matrice diagonale degli autovalori.

Inoltre dato $x' = Rx$, si ha $x = R^T x'$ o in termini di componenti:
 $x_i = R_{ji} x'_j$, da cui:

$$J_{ij} = \frac{\partial x_i}{\partial x'_j} = R_{ki} \frac{\partial x'_k}{\partial x'_j} = R_{ki} \delta_{jk} = R_{ji} = R_{ij}^T \Leftrightarrow J = R^T \Rightarrow |J| = |R^T| = 1$$

$$e^{-\frac{x^2}{2}} d^N x = e^{-\frac{1}{2} \sum_i \frac{(x'_i - \bar{x}'_i)^2}{\lambda_i^2}} d^N x'$$

Questo significa che data un insieme di variabili correlate di matrice di covarianza V , possiamo sempre trovare una rotazione che trasformi tali variabili in variabili scorrelate con σ^2 pari agli autovalori di V .

Dimostriamo la 1:

$$\begin{aligned}\langle \mathbf{x}_i \rangle &= \int \mathbf{x}_i N e^{-\frac{\chi^2}{2}} d^N \mathbf{x} = N \int R_{ji} \mathbf{x}'_j e^{-\frac{1}{2} \frac{\sum_i (\mathbf{x}'_i - \bar{\mathbf{x}}'_i)^2}{\lambda_i^2}} d^N \mathbf{x}' = N R_{ji} \int \mathbf{x}'_j e^{-\frac{1}{2} \frac{\sum_i (\mathbf{x}'_i - \bar{\mathbf{x}}'_i)^2}{\lambda_i^2}} d^N \mathbf{x}' \\ &= R_{ji} \bar{\mathbf{x}}'_j = \bar{\mathbf{x}}_i\end{aligned}$$

Dimostriamo la 3:

$$\begin{aligned}\langle (\mathbf{x}_i - \langle \mathbf{x}_i \rangle) (\mathbf{x}_j - \langle \mathbf{x}_j \rangle) \rangle &= \int (\mathbf{x}_i - \bar{\mathbf{x}}_i) (\mathbf{x}_j - \bar{\mathbf{x}}_j) N e^{-\frac{\chi^2}{2}} d^N \mathbf{x} = \\ &= N R_{ki} R_{lj} \int (\mathbf{x}'_k - \bar{\mathbf{x}}'_k) (\mathbf{x}'_l - \bar{\mathbf{x}}'_l) e^{-\frac{1}{2} \frac{\sum_i (\mathbf{x}'_i - \bar{\mathbf{x}}'_i)^2}{\lambda_i^2}} d^N \mathbf{x}' = R_{ki} R_{lj} \lambda_l^2 \delta_{kl} = R_{ki} \lambda_l^2 \delta_{kl} R_{lj} = \\ &= \left[R^T \begin{pmatrix} \lambda_1^2 & 0 & \dots \\ 0 & \lambda_2^2 & \dots \\ 0 & 0 & \dots \end{pmatrix} R \right]_{ij} = V_{ij}\end{aligned}$$

Generazione di distribuzioni multivariate

Procediamo nel modo seguente:

- Determiniamo autovalori ed autovettori della matrice (la matrice di covarianza è simmetrica reale quindi ammette sempre autovalori reali);
- troviamo la rotazione che trasforma le variabili x_i nelle loro proiezioni sugli autovettori x'_j
- Generiamo x'_j gaussiane secondo la radice degli autovalori;
- Ruotiamo in senso inverso per riottenere x_i da x'_j che saranno dunque correlate secondo la matrice di covarianza data.



Simulazione di processi fisici: decadimenti

Simulazione di processi fisici: decadimenti

Il decadimento di una particella di massa M in n corpi è dato da:

$$d\Gamma = \frac{(2\pi)^4}{2M} |\mathbf{M}| d\Phi_n(P; p_1, p_2, \dots, p_n) \quad d\Phi_n(P; p_1, p_2, \dots, p_n) = \delta^4\left(P - \sum_{i=1}^n p_i\right) \frac{\prod_{i=1}^n d^3 p_i}{(2\pi)^3 2 E_i}$$

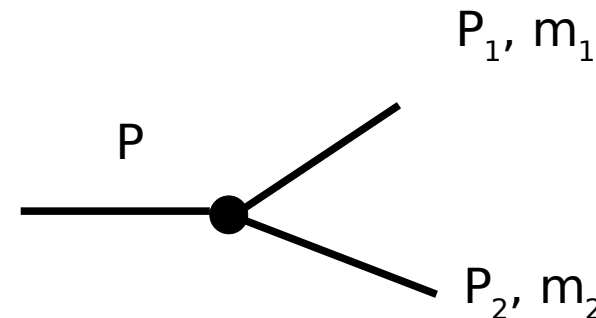
P quadrimpulso della particella madre
 p_1, p_2, \dots, p_n quadrimpulsi delle particelle figlie

Conservazione
del quadrimpulso
totale

Nel caso di decadimento a due corpi:

$$E_1 = \frac{M^2 - m_2^2 + m_1^2}{2M}$$

$$|p_1| = |p_2| = \frac{\left[\left(M^2 - (m_1 + m_2)^2 \right) \left(M^2 - (m_1 - m_2)^2 \right) \right]^{1/2}}{2M}$$



$$d\Gamma = \frac{1}{32\pi^2} |\mathbf{M}|^2 \frac{|\vec{p}_1|}{M^2} d\Omega$$

\mathbf{M} può essere al più una funzione di θ, ϕ

Osservazioni:

1) la conservazione del quadrimpulso totale è garantita dallo spazio delle fasi;

2) la conservazione del momento angolare, i fattori di spin compaiono in $|\mathbf{M}|$.

$$\text{Es: } \phi \rightarrow \eta \gamma$$

ϕ Vettore
spin = 1
Isospin 0
parità -
coniugazione di carica -
cotenuto in quarks s sbar

γ Vettore
spin = 1
Isospin 0
parità -
coniugazione di carica -
cotenuto in quarks 0

η Pseudoscalare
spin = 0
Isospin 0
parità -
coniugazione di carica +
cotenuto in quarks uubar
dubar sbar

Conservazione del momento angolare totale:

$$1 \text{ spin } \phi = 1 \text{ spin } \gamma + 0 \text{ spin } \eta + L \text{ momento orbitale} = 1 + L$$

Sono possibili tutti gli L per cui $1+L = 1$ secondo la somma dei momenti angolari:

$$\begin{aligned} \mathbf{L} = \mathbf{0} &\rightarrow \mathbf{1} + \mathbf{0} = \mathbf{1} & \mathbf{L} = \mathbf{2} &\rightarrow \mathbf{1} + \mathbf{2} = \mathbf{1,2,3} \\ \mathbf{L} = \mathbf{1} &\rightarrow \mathbf{1} + \mathbf{1} = \mathbf{0,1,2} & \mathbf{L} = \mathbf{3} &\rightarrow \mathbf{1} + \mathbf{3} = \mathbf{2,3,4} \end{aligned}$$

Conservazione della parità:

$$P(\phi) = P(\gamma) * P(\eta) * (-1)^L$$

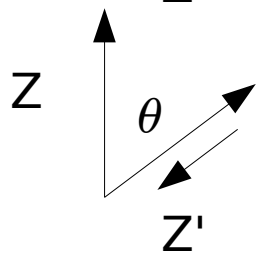
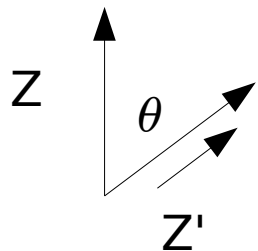
$$-1 = -1 \times -1 \times (-1)^L \Rightarrow L \text{ è dispari}$$

~~L = 0, 1, 2~~ Il momento angolare totale è 1.

Conservazione di L_z :

Scegliamo il sistema di riferimento in modo che l'asse z coincida con lo spin della ϕ .

Lo spin del fotone è lungo la direzione di volo. La componente lungo z



$$\text{spin } \mathbf{M} = |{}_z \langle \mathbf{1}, \mathbf{1} | \mathbf{H} | \mathbf{1}, \mathbf{1} \rangle_z|^2 = \left(\frac{1 + \cos \theta}{2} \right)^2$$

$$\mathbf{M} = |{}_z \langle \mathbf{1}, -\mathbf{1} | \mathbf{H} | \mathbf{1}, \mathbf{1} \rangle_z|^2 = \left(\frac{1 - \cos \theta}{2} \right)^2 \quad \mathbf{M} = \left(\frac{1 + \cos \theta}{2} \right)^2 + \left(\frac{1 - \cos \theta}{2} \right)^2 = \frac{1 + \cos^2 \theta}{2}$$

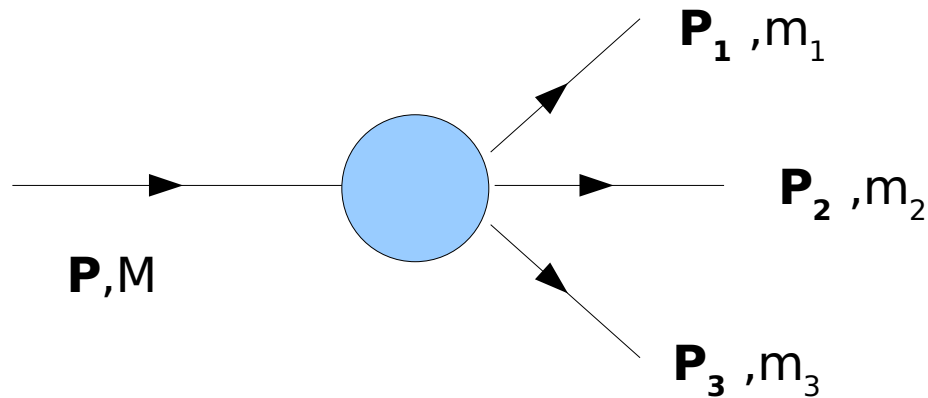
Sommiamo sugli stati di spin finali:

Generare decadimento

$$\phi \rightarrow \eta \gamma, \eta \rightarrow \gamma \gamma$$

$$\phi \rightarrow \pi^0 \gamma, \pi^0 \rightarrow \gamma \gamma$$

Decadimento a tre corpi

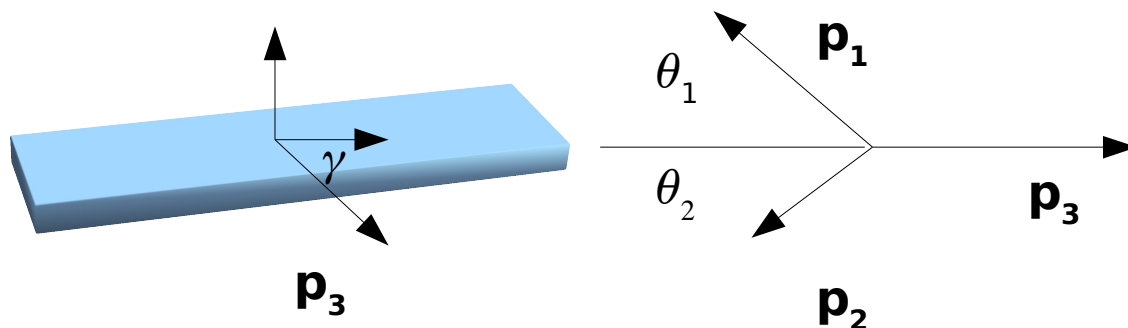


Ci mettiamo nel sistema di riferimento della particella madre $\mathbf{P} = \mathbf{0}$.

$$\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3 = \mathbf{0}$$

Questo significa che gli impulsi delle tre particelle sono in un piano, la posizione del piano nello spazio è definita da due angoli α, β , mentre la direzione della particella 3 rispetto ad un asse definito nel piano la indichiamo con γ .

In tal modo l'orientazione del sistema delle tre particelle è completamente definita.



$$p_1 \cos \theta_1 + p_2 \cos \theta_2 = p_3$$
$$p_1 \sin \theta_1 + p_2 \sin \theta_2 = 0$$

Dati i moduli dei tre impulsi anche la direzione è fissata.

Ampiezza di decadimento differenziale

$$d\Gamma = \frac{1}{(2\pi)^5} \frac{1}{16M} |\mathbf{M}|^2 dE_1 dE_2 d\alpha d(\cos\beta) d\gamma$$

Essendo $E_1 + E_2 + E_3 = E_{\text{madre}}$, possiamo esprimere E_3 in funzione di E_1 ed E_2

Se la particella madre è uno scalare, la fisica sarà indipendente dall'orientazione del piano di giacenza degli impulsi delle tre particelle. Lo stesso si può dire se mediamo sugli stati di spin della particella madre (fascio non polarizzato).

Integrando sugli angoli α, β e γ otteniamo.

$$d\Gamma = \frac{1}{(2\pi)^3} \frac{1}{8M} |\bar{\mathbf{M}}|^2 dE_1 dE_2$$

Definiamo: $m_{12}^2 = (p_1 + p_2)^2$ $m_{23}^2 = (p_2 + p_3)^2$ $m_{13}^2 = (p_1 + p_3)^2$
valgono le seguenti relazioni:

$$m_{12}^2 + m_{23}^2 + m_{13}^2 = M^2 + m_1^2 + m_2^2 + m_3^2$$

$$m_{12}^2 = M^2 + m_3^2 - 2ME_3 \quad m_{23}^2 = M^2 + m_1^2 - 2ME_1$$

$$m_{12}^2 = M^2 + m_3^2 - 2M(E - E_1 - E_2)$$

Applicando la trasformazione di variabili: $(E_1, E_2) \rightarrow (m_{12}^2, m_{23}^2)$

$$d\Gamma = \frac{1}{(2\pi)^3} \frac{1}{32M^3} |\bar{\mathbf{M}}|^2 dm_{12}^2 dm_{23}^2$$

La dinamica del decadimento può essere completamente descritta dalle variabili m_{12}^2, m_{23}^2 . La distribuzione bidimensionale nel piano (m_{12}^2, m_{23}^2) è chiamata Dalitz plot. Se $|\bar{\mathbf{M}}|^2$ è costante la distribuzione nel piano è uniforme. Tuttavia il range di variabilità delle due variabili è ristretto dalla conservazione dell'energia.

Fissato un valore di m_{12}^2 , m_{23}^2 può variare in un intervallo di valori $[(m_{23}^2)_{\min}, (m_{23}^2)_{\max}]$, tali valori corrispondono alle configurazioni in cui \mathbf{p}_2 è parallelo o antiparallelo a \mathbf{p}_3 rispettivamente.

$$(m_{23}^2)_{\max} = (E_2^* + E_3^*) - \left(\sqrt{E_2^{*2} - m_2^2} - \sqrt{E_3^{*2} - m_3^2} \right)^2$$

$$(m_{23}^2)_{\min} = (E_2^* + E_3^*) - \left(\sqrt{E_2^{*2} - m_2^2} + \sqrt{E_3^{*2} - m_3^2} \right)^2$$

E_2^* e E_3^* sono le energie di 2,3 nel sistema di riposo di 1,2

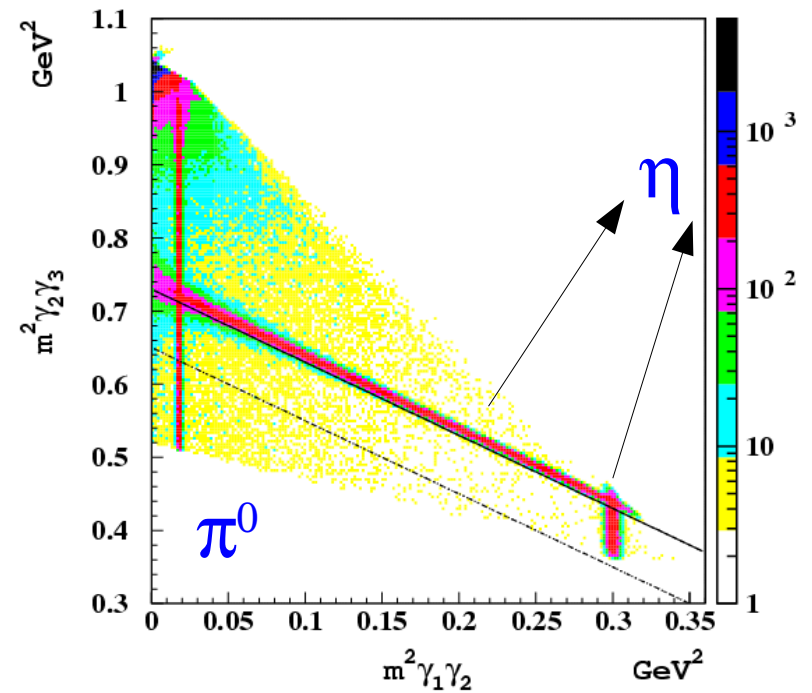
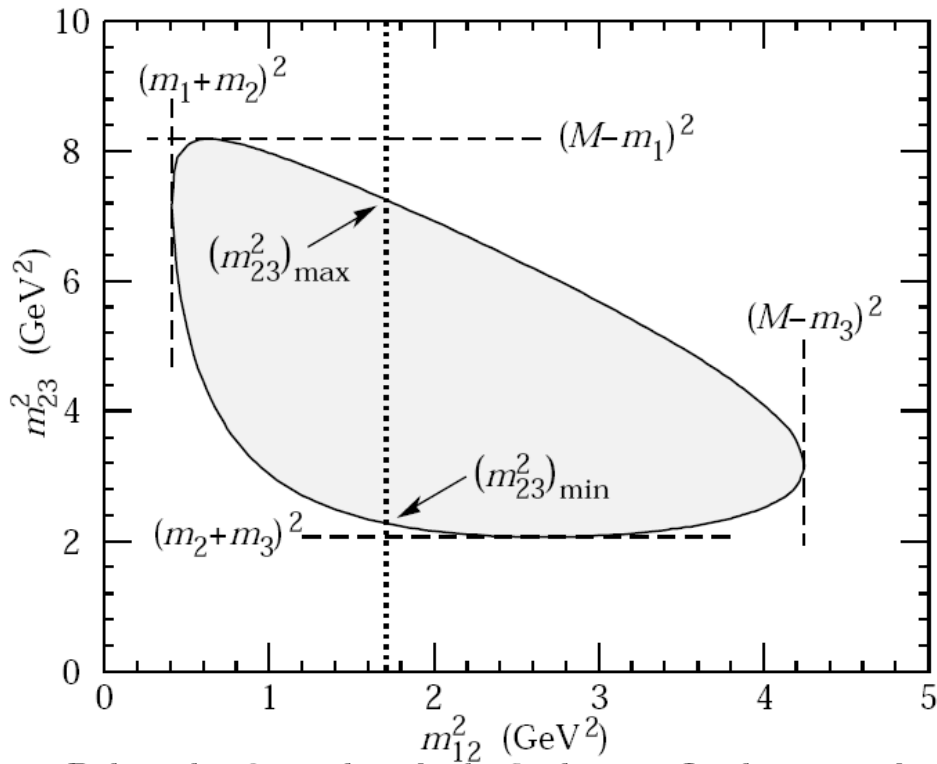
$$E_2^* = (m_{12}^2 - m_1^2 + m_2^2)/2m_{12}$$

$$E_3^* = (m_{12}^2 - m_1^2 + m_3^2)/2m_{12}$$

Campione a 3 fotoni

$$\phi \rightarrow \eta, \pi^0 \gamma \rightarrow \gamma\gamma\gamma$$

Dati di KLOE



Disuniformità nel Dalitz plot danno informazioni sulla fisica del decadimento.



Simulazione di decadimenti a tre corpi.

Nel caso di elemento di matrice piatto ci basta generare secondo lo spazio delle fasi. Estraiamo m_{12}^2 e m_{23}^2 nella regione ammessa, generiamo gli angoli α, β, γ uniformi. Potremmo costruire da noi il generatore (nella parte iniziale abbiamo imparato a generare praticamente tutte le distribuzioni necessarie), calcolare i vettori $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ e studiarne il comportamento.

Esistono librerie precompilate che possono essere utilizzate.

In root la classe **TgenPhaseSpace** ci permette di determinare la distribuzione dei momenti nel decadimento a tre corpi, in ipotesi di elemento di matrice costante.

Usualmente si dice che le particelle sono generate secondo lo spazio delle fasi.

public:

```
    TGenPhaseSpace ()  
    TGenPhaseSpace (const TGenPhaseSpace& gen)  
    virtual ~TGenPhaseSpace ()  
    static TClass* Class ()  
    Double_t Generate ()  
    TLorentzVector* GetDecay (Int_t n)  
    Int_t GetNt () const  
    Double_t GetWtMax () const  
    virtual TClass* IsA () const  
    TGenPhaseSpace& operator= (const TGenPhaseSpace&)  
    Bool_t SetDecay (TLorentzVector& P, Int_t nt, Double_t* mass, Option_t* opt = "")  
    virtual void ShowMembers (TMemberInspector& insp, char* parent)  
    virtual void Streamer (TBuffer& b)  
    void StreamerNVirtual (TBuffer& b)
```

Come possiamo generare secondo un Dalitz plot non piatto?




Supponiamo di conoscere il Dalitz plot di un processo a tre corpi. Ossia conosciamo: $M(m_{12}^2, m_{23}^2)$. Come possiamo generare velocemente secondo il processo fisico dato?

Generiamo gli eventi secondo lo spazio delle fasi, calcoliamo le quantità m_{12}^2 e m_{23}^2 , essendo invarianti di Lorentz possiamo valutarle in qualunque sistema di riferimento.

A questo punto possiamo scegliere due criteri:

- 1) Pesare l'evento: calcoliamo $|M(m_{12}^2, m_{23}^2)|^2$ e associamo a quell'evento un peso proporzionale a $|M(m_{12}^2, m_{23}^2)|^2$;
- 2) Determiniamo il massimo e il minimo di $|M(m_{12}^2, m_{23}^2)|^2$ sullo spazio delle fasi. Generiamo un numero casuale tra min e max, se tale numero è minore di $|M(m_{12}^2, m_{23}^2)|^2$ accettiamo l'evento, altrimenti lo rigettiamo (hit or miss bidimensionale).



1) conserviamo tutti gli eventi, ma nel costruire una qualunque distribuzione dobbiamo ricordarci di inserire il peso corretto nell'istogramma considerato. L'errore statistico in un bin non sarà dato dalla \sqrt{n} , a causa della presenza dei pesi. Dobbiamo tenerne conto quando effettuiamo fits etc..

2) In caso di $|M|^2$ molto disuniforme rigettiamo gran parte degli eventi (esempio KLOE), tuttavia possiamo trattare gli eventi esattamente come trattiamo i dati e anche il trattamento statistico è molto più semplice.

Esercizio:

Simulare il decadimento $\eta \rightarrow \pi^0 \gamma \gamma$ assumendo un elemento di matrice del tipo:

$$|M|^2 = \frac{1}{(m_{\gamma\gamma} - m_{a0})^2 + \Gamma_{a0}^2}$$

$$M_{a0} = 980 \text{ MeV}$$

$$\Gamma_{a0} = 50 \text{ MeV}$$

Simulare il successivo decadimento $\pi^0 \rightarrow \gamma \gamma$;

Assumendo che la risoluzione del rivelatore dei fotoni è:

$$\frac{\sigma_E}{E} = \frac{0.057}{\sqrt{E(\text{GeV})}}$$

Simulare la risposta del calorimetro e istogrammare $m_{12}^2 m_{34}^2$ dove 1,2,3,4 sono i fotoni ordinati secondo la loro energia.

Fare lo stesso nel caso di elemento di matrice piatto.



Simulazione della risposta di un rivelatore



I rivelatori di particelle sono degli apparati che rivelano la presenza di particelle di vario tipo tramite l'emissione di segnali di tipo elettronico.

La rivelazione delle particelle è possibile grazie ai processi fisici che si presentano nel momento in cui una particella attraversa un mezzo materiale. Tali processi sono noti come processi di interazioni radiazione materia.

I processi sono di due tipi:

Distruuttivi: la particella scompare al momento dell'interazione

Non distruuttivi: la particella mantiene il suo stato fisico ma varia la sua configurazione cinematica, quadrimpulso - spin.

Processi di interazione di particelle cariche pesanti ($m \gg m_e$)

Ionizzazione:

energia persa da particelle per interazione con elettroni orbitali nel materiale.

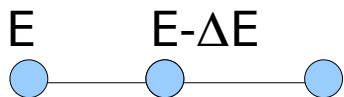
$$-\frac{dE}{dx} = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right]$$

La formula fornisce solo il valor medio, mentre le fluttuazioni statistiche intorno al valor medio sono date da una $f(\Delta, \beta\gamma, x)$ approssimativamente una distribuzione di Landau.

Δ energia persa per ionizzazione

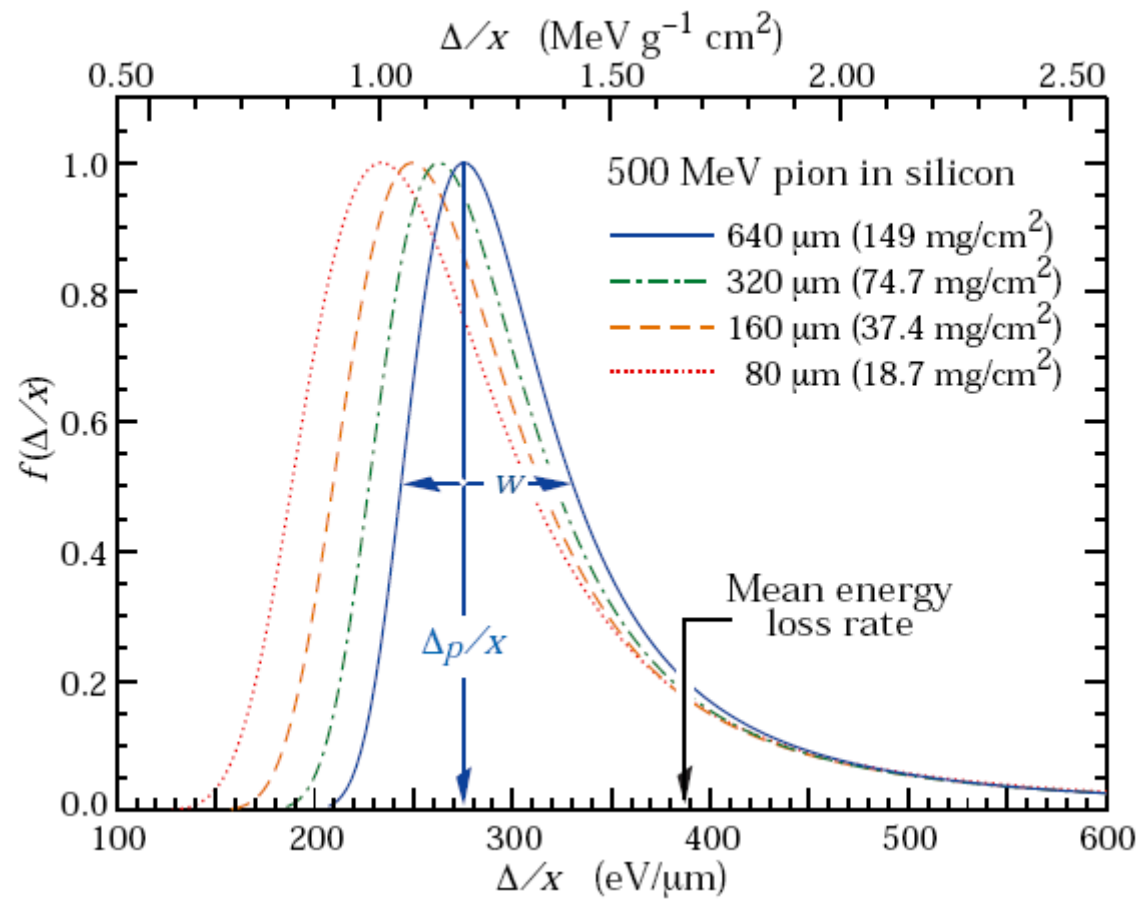
x spessore attraversato

Dato uno step di attraversamento δx , calcoliamo il dE utilizzando la formula di Bethe, ed estraiamo secondo la distribuzione di Landau l'energia persa dalla particella nel tratto dx .



Step di tracciamento

Symbol	Definition	Units or Value
α	Fine structure constant ($e^2/4\pi\epsilon_0\hbar c$)	1/137.035 999 11(46)
M	Incident particle mass	MeV/ c^2
E	Incident particle energy $\gamma M c^2$	MeV
T	Kinetic energy	MeV
$m_e c^2$	Electron mass $\times c^2$	0.510 998 918(44) MeV
r_e	Classical electron radius $e^2/4\pi\epsilon_0 m_e c^2$	2.817 940 325(28) fm
N_A	Avogadro's number	$6.022 1415(10) \times 10^{23} \text{ mol}^{-1}$
ze	Charge of incident particle	
Z	Atomic number of absorber	
A	Atomic mass of absorber	g mol^{-1}
K/A	$4\pi N_A r_e^2 m_e c^2 / A$	0.307 075 MeV $\text{g}^{-1} \text{ cm}^2$ for $A = 1 \text{ g mol}^{-1}$
I	Mean excitation energy	eV (<i>Nota bene!</i>)
$\delta(\beta\gamma)$	Density effect correction to ionization energy loss	
$\hbar\omega_p$	Plasma energy ($\sqrt{4\pi N_e r_e^3} m_e c^2 / \alpha$)	$28.816 \sqrt{\rho(Z/A)} \text{ eV}^{(a)}$
N_e	Electron density	(units of r_e) $^{-3}$
w_j	Weight fraction of the j th element in a compound or mixture	
n_j	\propto number of j th kind of atoms in a compound or mixture	
—	$4\alpha r_e^2 N_A / A$	($716.408 \text{ g cm}^{-2}$) $^{-1}$ for $A = 1 \text{ g mol}^{-1}$
X_0	Radiation length	g cm^{-2}
E_c	Critical energy for electrons	MeV
$E_{\mu c}$	Critical energy for muons	GeV
E_s	Scale energy $\sqrt{4\pi/\alpha} m_e c^2$	21.2052 MeV
R_M	Molière radius	g cm^{-2}



Elettroni di Knock on.

La distribuzione di probabilità per elettroni di alta energia (δ rays) è data da:

$$\frac{d^2 N}{dTdx} = \frac{1}{2} K Z^2 \frac{Z}{A} \frac{1}{\beta^2} \frac{F(T)}{T} \quad T_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma \frac{m_e}{M} + \left(\frac{m_e}{M}\right)^2}$$
$$F(T) = 1 - \beta^2 \frac{T}{T_{max}}$$

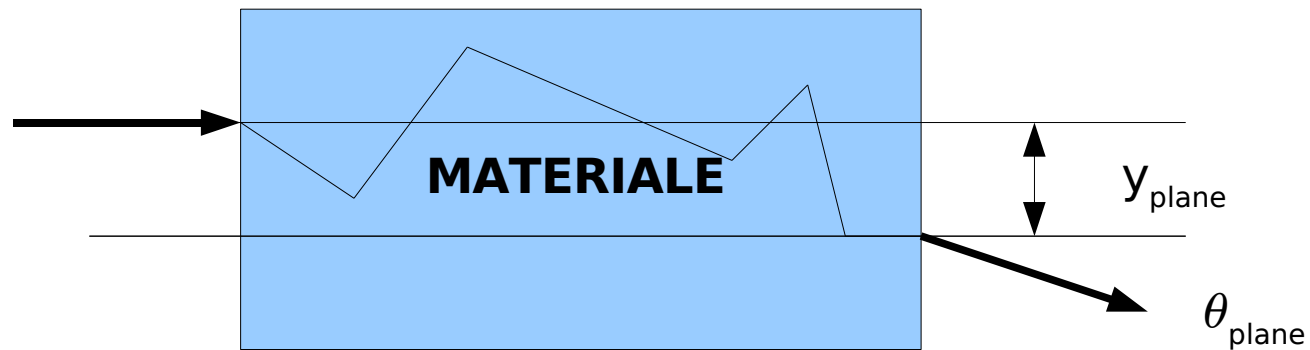
Allo step dx, calcoliamo la probabilità P di emissione di un elettrone (integrale della formula in T), estraiamo un numero tra 0 e 1, se il numero è minore di P, decidiamo di emettere un elettrone, l'energia di tale elettrone viene estratta secondo dN/dT.

L'angolo di emissione è fissato: $\cos \theta = (T_e / p_e)(p_{max} / T_{max})$

Scattering multiplo.



Scattering elastico coulombiano contro i nuclei.



$$\theta_0 = \frac{13.6 \text{ MeV}}{\beta c p} z \sqrt{x/X_0} [1 + 0.038 \ln(x/X_0)]$$

$$\sigma_y = \frac{1}{\sqrt{3}} x \theta_0 \quad \rho_{y\theta} = \sqrt{3}/2$$

$$\sigma_\theta = \theta_0$$



Altri processi:

- Sciami elettromagnetici (probabilità di creazione di coppia e probabilità di Brhemstralug da valutare ad ogni step);
- Effetto fotoelettrico;
- Effetto compton;
- Radiazione cherenkov;
- Interazione adronica, spallazione di neutroni

Tutti questi processi devono essere valutati al singolo step, inoltre nel caso di particelle instabili dobbiamo anche valutare la probabilità di decadimento (il tempo di decadimento può essere valutato già dalla creazione. Al dato step dobbiamo solo verificare se la particella è viva o morta).

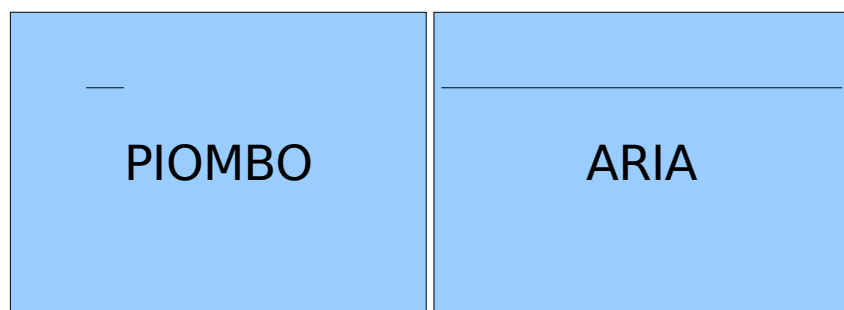
Cosa succede ad una particella stabile?

1) Può uscire dall'apparato simulato (di solito si pongono dei materiali fittizi, in FLUKA black hole – WORLD in GEANT4, in cui si arresta il tracciamento di ogni particella che lo raggiunge).

2) La particella si può arrestare nel materiale (la sua vita non è più seguita) l'energia della particella viene depositata localmente nel momento in cui decidiamo di arrestare la particella.

Uno dei parametri più importanti di un MC è la soglia di energia al di sotto della quale una particella non è più tracciata. Se allo step n mi accorgo che $E < E_{\text{cut}}$ la particella non è più tracciata e l'energia è depositata localmente.

Problema:



Una particella che nel piombo al di sotto di E_{cut} non si sposta potrebbe attraversare completamente un volume di un materiale più leggero.

Il problema può essere risolto impostando tagli per range invece che per energia.

Calcolo il range della particella nel dato materiale, se tale range è minore del taglio arresto il tracking della particella. Simula molto meglio la realtà. E' necessario calcolare il range per ogni materiale.



GEANT 4

Programma main()

1) Creiamo un oggetto della classe RunManager:

```
G4RunManager* runManager = new G4RunManager;
```

controlla il flusso del programma e gestisce la generazione di eventi in un singolo run

```
runManager->SetUserInitialization(new ExN01DetectorConstruction);
```


Creiamo un oggetto geometria e passiamo il suo puntatore al runManager (dovremo descrivere la geometria in questa classe, che deve essere una classe derivata di **G4VUserDetectorConstruction**)

In questa classe definiremo:

- Geometria dell'apparato
- I materiali
- La definizione delle regioni sensibili (la parte che misuriamo)
- Il criterio di lettura delle regioni sensibili

}

Non in questo corso



```
runManager->SetUserInitialization(new ExN01PhysicsList);
```

Derivata da: **G4VUserPhysicsList**

Creiamo un oggetto descrivente la fisica dei processi e passiamo il puntatore al runManager

(dovremo indicare i processi di fisica che vogliamo simulare in questa classe)

- Le particelle che saranno usate nella simulazione
- I tagli sul range di queste particelle;
- Tutti i processi di fisica che vogliamo siano simulati


```
runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
```

Derivata da: **G4VUserPrimaryGeneratorAction**

Definiamo l'evento primario (il generatore nel decadimento $\eta \rightarrow \pi^0 \gamma \gamma$ ad esempio). Nel nostro caso un fascio di particelle monoenergetiche.

```
runManager->initialize();
```

Finora abbiamo solamente definito e descritto ciò che vogliamo, adesso la geometria viene creata e valutate le sezioni d'urto per tutti i processi inclusi nella PhysicsList



```
int numberOfEvent = 3;
```

Qui generiamo gli eventi veri e propri.

```
runManager>beamOn(numberOfEvent);
```

Quando creiamo Run Manager vengono creati altri oggetti. Non possiamo ricrearli da zero ma dobbiamo accedere a questi oggetti primari, per ottenere un puntatore all'interfaccia utente ad esempio:

```
G4UImanager* UI = G4UImanager::getUIpointer();
```

Non c'è new perché l'oggetto è già presente. Dobbiamo solo accedere al suo indirizzo.

Se vogliamo mandare un comando al generatore:

```
UI->ApplyCommand("/run/verbose 1");
```

Definizione della geometria

Abbiamo un volume World (mondo) che contiene tutti gli altri volumi, ogni volume deve essere contenuto in volumi precedentemente creati, incluso il volume mondo.

Quando creiamo un volume, definiamo la forma e lo inseriamo in un altro volume che lo contiene. Quindi abbiamo un volume madre e un volume figlio.

Il volume figlio è descritto nel sistema di coordinate del volume madre.

Per descrivere la forma di un volume, si introduce il concetto di solido. Un solido è un oggetto geometrico completamente definito (cilindro, cubo, sfera...)

Un volume logico è dato da un solido che ne descrive la forma, e da alcune proprietà: il materiale, la regione sensibile, la presenza di un campo magnetico.

Un volume fisico è una copia del volume logico, che poi viene posizionato in un volume fisico madre.

Creazione della geometria

Dobbiamo creare una classe, figlia di **G4VUserDetectorConstruction**
class CorsoRivelatore : public G4VUserDetectorConstruction

Classe figlia significa che eredita tutti i metodi della classe madre

```
#ifndef CorsoRivelatore_H //istruzioni di precompilazione
#define CorsoRivelatore_H 1 // ci assicura che il file sarà include
                          // solo una volta

class G4LogicalVolume; // predefinizione delle classi, ci assicurano che
class G4VPhysicalVolume; // non avremo errori di compilazione

#include "G4VUserDetectorConstruction.hh"

class CorsoRivelatore : public G4VUserDetectorConstruction
{
public:

    CorsoRivelatore(); // costruttore
    ~CorsoRivelatore(); // distruttore

    G4VPhysicalVolume* Construct(); // metodo che restituisce un puntatore
                                   // ad un volume fisico (il mondo)

private:
    // membri privati

    // Logical volumes
    //
    G4LogicalVolume* pezzo_log;

    // Physical volumes
    //
    G4VPhysicalVolume* pezzo_phys; // definizione dei volumi fisici
};

#endif
```

CorsoRivelatore.hh

```

//-----
#include "CorsoRivelatore.hh" //possiamo inserirlo prima, perché abbiamo usato
                             // la pre-definizione

#include "G4Material.hh" // Materiali
#include "G4Box.hh" // Geometri Scatola
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh" // Volumi logici
#include "G4ThreeVector.hh"
#include "G4PVPlacement.hh"
#include "globals.hh"

CorsoRivelatore::CorsoRivelatore() // costruttore, inizializziamo tutti gli oggetti
: pezzo_log(0),
  pezzo_phys(0)
{;}

CorsoRivelatore::~CorsoRivelatore() // distruttore, non facciamo nulla
{
}

G4VPhysicalVolume* CorsoRivelatore::Construct() // Definizione del metodo Construct
{

//----- materials

G4double a; // atomic mass
G4double z; // atomic number
G4double density;

// G4Material* Ar =
// new G4Material("ArgonGas", z= 18., a= 39.95*g/mole, density= 1.782*mg/cm3);

// G4Material* Al =
// new G4Material("Aluminum", z= 13., a= 26.98*g/mole, density= 2.7*g/cm3);

// Definiamo il materiale
G4Material* Pb =
new G4Material("Lead", z= 82., a= 207.19*g/mole, density= 11.35*g/cm3);

//----- volumes

//----- experimental hall (world volume)

```

```
G4double pezzo_x = 54.0*cm; // possiamo inserire un unità di misura -54 +54
G4double pezzo_y = 10.0*cm; // -10 + 10
G4double pezzo_z = 10.0*cm;
G4Box* pezzo_box
    = new G4Box("pezzo_box",pezzo_x,pezzo_y,pezzo_z); // creo una scatola
pezzo_log = new G4LogicalVolume(pezzo_box,
                               Pb, "pezzo_log", 0, 0, 0); // creo un volume logico
// lo posiziono, 0 rotazione no rotazione, altrimenti matrice di rotazione, G4ThrrreVector(x,y,z) traslazione rispetto al volume madre,
// puntatore volume logico, nome,
// puntatore volume madre (0 solo per World volume) nel nostro caso l'unico volume che abbiamo, numero di copia
pezzo_phys = new G4PVPlacement(0, G4ThreeVector(),
                               pezzo_log, "pezzo", 0, false, 0); // posiziono il volume

return pezzo_phys; // ritorno il mondo, ossia il pezzo
}
```

Definizione delle particelle

Dobbiamo creare la classe che definisce le particelle che vogliamo propagare e i processi che vogliamo simulare.

```
#ifndef CorsoRivelatorePhysics_h
#define CorsoRivelatorePhysics_h 1

#include "G4VUserPhysicsList.hh"
#include "globals.hh"

class CorsoRivelatorePhysics: public G4VUserPhysicsList // classe figlia di G4VUserPhysicsList
{
public:
    CorsoRivelatorePhysics();
    ~CorsoRivelatorePhysics();

protected:
    // Construct particle and physics process
    void ConstructParticle(); // metodi virtuali, significa che ci sono come definizione, ma non esiste un codice
    void ConstructProcess(); // che ne determina il funzionamento, dobbiamo noi creare questo codice
    void SetCuts(); // dobbiamo costruire le particelle, definire i processi, impostare i tagli di range
    //protected: // metodi non modificabili
    void ConstructGeneral();
    void ConstructEM();
};

#endif
```

→ Di servizio

Costruttori di particelle
G4BosonConstructor
G4LeptonConstructor
G4MesonConstructor

G4BarionConstructor
G4IonConstructor
G4ShortlivedConstructor .

```
#include "globals.hh"
#include "CursoRivelatorePhysics.hh"
#include "G4ParticleTypes.hh" // definizione dei tipi di particella predefiniti
#include "G4LeptonConstructor.hh"
#include "G4IonConstructor.hh"
#include "G4BosonConstructor.hh"
#include "G4ProcessManager.hh"

CursoRivelatorePhysics::CursoRivelatorePhysics()
{;}

CursoRivelatorePhysics::~CursoRivelatorePhysics()
{;}

void CursoRivelatorePhysics::ConstructParticle() // specifichiamo il metodo virtuale
{
    // In this method, static member functions should be called
    // for all particles which you want to use.
    // This ensures that objects of these particle types will be
    // created in the program.

    // G4Geantino::GeantinoDefinition();
    // creo elettroni
    G4Electron::ElectronDefinition();
    // creo tutti i leptoni
    G4LeptonConstructor pConstructor;
    pConstructor.ConstructParticle();
    // creo tutti i bosoni (anche fotoni);
    G4BosonConstructor pBosons;
    pBosons.ConstructParticle();
    // creo tutti gli ioni
    G4IonConstructor pIon;
    pIon.ConstructParticle();

    G4Proton::ProtonDefinition();
    G4AntiProton::AntiProtonDefinition();

    G4Neutron::NeutronDefinition();
    G4AntiNeutron::AntiNeutronDefinition();

    G4PionPlus::PionPlusDefinition();
    G4PionMinus::PionMinusDefinition();
    G4PionZero::PionZeroDefinition();
    G4Eta::EtaDefinition();
    G4EtaPrime::EtaPrimeDefinition();
    G4KaonPlus::KaonPlusDefinition();
    G4KaonMinus::KaonMinusDefinition();
    G4KaonZero::KaonZeroDefinition();
    G4AntiKaonZero::AntiKaonZeroDefinition();
    G4KaonZeroLong::KaonZeroLongDefinition();
    G4KaonZeroShort::KaonZeroShortDefinition();
}
```

```
void CorsoRivelatorePhysics::ConstructProcess()
{
    // Define transportation process

    AddTransportation(); // processo di propagazione nello spazio di una particella
    ConstructEM(); // funzione interna che si occupa dell'attivazione dei processi elettromagnetici
    ConstructGeneral();
}

void CorsoRivelatorePhysics::SetCuts() // impostiamo i tagli per range, sono poi riconvertiti in energia a seconda del materiale
{
    // suppress error messages even in case e/gamma/proton do not exist
    G4int temp = GetVerboseLevel();
    SetVerboseLevel(0);
    // " G4VUserPhysicsList::SetCutsWithDefault" method sets
    // the default cut value for all particle types
    SetCutsWithDefault();

    // Retrieve verbose level
    SetVerboseLevel(temp);
}
```



```

#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"

#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"

#include "G4MuIonisation.hh"
#include "G4MuBremsstrahlung.hh"
#include "G4MuPairProduction.hh"

#include "G4hIonisation.hh"

#include "G4StepLimiter.hh"
#include "G4UserSpecialCuts.hh"

void CorsoRivelatorePhysics::ConstructEM()
{
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();

    if (particleName == "gamma") {
      // gamma
      pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
      pmanager->AddDiscreteProcess(new G4ComptonScattering);
      pmanager->AddDiscreteProcess(new G4GammaConversion);

    } else if (particleName == "e-") {
      //electron
      pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
      pmanager->AddProcess(new G4eIonisation, -1, 2,2);
      pmanager->AddProcess(new G4eBremsstrahlung, -1, 3,3);

    } else if (particleName == "e+") {
      //positron
      pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
      pmanager->AddProcess(new G4eIonisation, -1, 2,2);
      pmanager->AddProcess(new G4eBremsstrahlung, -1, 3,3);
      pmanager->AddProcess(new G4eplusAnnihilation, 0, -1,4);
    }
  }
}

```

```

} else if( particleName == "mu+" ||
           particleName == "mu-" ) {
  //muon
  pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
  pmanager->AddProcess(new G4MuIonisation, -1, 2,2);
  pmanager->AddProcess(new G4MuBremsstrahlung, -1, 3,3);
  pmanager->AddProcess(new G4MuPairProduction, -1, 4,4);

} else if ((!particle->IsShortLived()) &&
           (particle->GetPDGCharge() != 0.0) &&
           (particle->GetParticleName() != "chargedgeantino")) {
  //all others charged particles except geantino
  pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
  pmanager->AddProcess(new G4hIonisation, -1, 2,2);
  //step limit
  pmanager->AddProcess(new G4StepLimiter, -1, -1,3);
  //pmanager->AddProcess(new G4UserSpecialCuts, -1, -1,4);
}
}
}

```

```
#include "G4Decay.hh"
void CorsoRivelatorePhysics::ConstructGeneral()
{
    // Add Decay Process
    G4Decay* theDecayProcess = new G4Decay();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ){
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        if (theDecayProcess->IsApplicable(*particle)) {
            pmanager ->AddProcess(theDecayProcess);
            // set ordering for PostStepDoIt and AtRestDoIt
            pmanager ->SetProcessOrdering(theDecayProcess, idxPostStep);
            pmanager ->SetProcessOrdering(theDecayProcess, idxAtRest);
        }
    }
}
```

Definizione del fascio.



Vogliamo simulare un fascio di fotoni che incide sul nostro blocco di piombo.

```
#ifndef CorsoPrimaryGeneratorAction_h
#define CorsoPrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"

class G4ParticleGun;
class G4Event;

class CorsoPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
public:
    CorsoPrimaryGeneratorAction();
    ~CorsoPrimaryGeneratorAction();

public:
    void GeneratePrimaries(G4Event* anEvent);

private:
    G4ParticleGun* particleGun; // lo utilizziamo internamente per definire il fascio
};

#endif
```

```

#include "CorsoPrimaryGeneratorAction.hh"

#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "globals.hh"

CorsoPrimaryGeneratorAction::CorsoPrimaryGeneratorAction()
{
    G4int n_particle = 1;
    particleGun = new G4ParticleGun(n_particle); // definiamo il numero di particelle da generare per evento
                                                // 1 nel nostro caso, 3 nel caso p0 gg

    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
    G4String particleName;
    particleGun->SetParticleDefinition(particleTable->FindParticle(particleName="e-")); // definisco il tipo di particella
    particleGun->SetParticleEnergy(1.0*GeV); // l'energia della particella
    particleGun->SetParticlePosition(G4ThreeVector(0.0,-10.0*cm, 0.0)); // generiamo il fascio sotto il blocco World
    particleGun->SetParticleMomentumDirection(G4ThreeVector(0.,1.,0.)); // direzione del fascio verso l'alto
}

CorsoPrimaryGeneratorAction::~CorsoPrimaryGeneratorAction()
{
    delete particleGun; // necessario, nel distruttore dobbiamo eliminare l'oggetto
}

void CorsoPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    // questa funzione genera il vertice primario (spiegare il formalismo di vertice e traccia)
    particleGun->GeneratePrimaryVertex(anEvent);
}

```

```
void G4ParticleGun::GeneratePrimaryVertex(G4Event* evt)
```

```
{  
  if(particle_definition==0) return;
```

```
  // create a new vertex
```

```
  G4PrimaryVertex* vertex =
```

```
    new G4PrimaryVertex(particle_position,particle_time);
```

Definisco il vertice

```
  // create new primaries and set them to the vertex
```

```
  G4double mass = particle_definition->GetPDGMass();
```

```
  G4double energy = particle_energy + mass;
```

```
  G4double pmom = std::sqrt(energy*energy-mass*mass);
```

```
  G4double px = pmom*particle_momentum_direction.x();
```

```
  G4double py = pmom*particle_momentum_direction.y();
```

```
  G4double pz = pmom*particle_momentum_direction.z();
```

```
  for( G4int i=0; i<NumberOfParticlesToBeGenerated; i++ )
```

```
  {
```

```
    G4PrimaryParticle* particle =
```

```
      new G4PrimaryParticle(particle_definition,px,py,pz);
```

```
    particle->SetMass( mass );
```

```
    particle->SetCharge( particle_charge );
```

```
    particle->SetPolarization(particle_polarization.x(),
```

```
                             particle_polarization.y(),
```

```
                             particle_polarization.z());
```

```
    vertex->SetPrimary( particle );
```

```
  }
```

Definisco la particella

```
  evt->AddPrimaryVertex( vertex );
```

Inserisco il vertice
primario nell'evento

Associo la particella al
vertice

```
}
```

Visualizzazione

Per utilizzare la visualizzazione, dobbiamo inizializzare il driver grafico, utilizzare una sessione interattiva per interagire con il driver grafico

nella sessione interattiva dobbiamo

- Richiamare il driver grafico `/vis/open OGLSX`
- Disegnare il volume `/vis/drawVolume`
- Visualizziamo le traiettorie `/vis/scene/add/trajectories`
- Accendiamo il fascio `/run/beamOn 1`
- Per cambiare angolo di vista `/vis/viewer/set/viewpointThetaPhi 10 20`