RELAZIONE sul PROGETTO
per l'ESAME di
TECNICHE di ANALISI NUMERICA e SIMULAZIONE

# Simulazione di un rivelatore di vertice

---

Anno Accademico 2018/2019

Professore: MASERA MASSIMO

Candidate: RACCA ELEONORA
SAUDA CRISTINA

# 1 Introduzione

Il progetto consiste nello sviluppo di un programma ROOT per la simulazione di un rivelatore di vertice di un esperimento a collider. Il rivelatore è costituito da una struttura cilindrica concentrica formata da una beam pipe di berillio e due layer di rivelatori al silicio.
Il programma consta di tre parti: una fase di generazione degli eventi, una di ricostruzione e infine una di analisi.

# 2 Struttura del programma

Il programma si articola su tre macro principali più due macro per l'esecuzione ed una serie di classi per l'implementazione delle macro. Le classi utilizzate sono le seguenti:

- `Rivelatore`, contenente le informazioni geometriche e fisiche dell'apparato.
- `Punto`, contenente le coordinate nei vari sistemi di riferimento (cartesiane, cilindriche e sferiche).
- `Urto` eredita da `Punto` e implementa diversi metodi per generare urti sulla beam pipe e sui layer, effettuare lo smearing e generare i punti di rumore.
- `Vertice`, eredita da `Punto` e memorizza le coordinate del vertice e ha un metodo per ricostruire la coordinata del vertice.
- `Trasporto` contiene la direzione di propagazione dei singoli urti e permette la rotazione angolare in caso di scattering multiplo.

Per compilare ed eseguire il programma con le configurazioni di default è necessario seguire questa procedura:

1. da terminale lanciare i seguenti comandi:

```
cd Provette
root
```

2. all'interno di ROOT eseguire i seguenti comandi

```
.x compila.C
EsecuzioneEsame()
```

La funzione `EsecuzioneEsame` permette di specificare, tramite l'aggiunta di un parametro booleano, l'avvio di simulazioni multiple a differenti livelli di rumore. La singola simulazione è formata da tre stadi: generazione, ricostruzione e analisi.
Essa viene eseguita andando a leggere da due file di tipo testo le informazioni necessarie. Nello specifico, *Generazione.txt* contiene le informazioni riguardanti il numero di eventi da generare, la distribuzione della molteplicità, la presenza o meno dello scattering multiplo e la distribuzione di pseudorapidità $\eta$, mentre il file *Ricostruzione.txt* contiene le informazioni necessarie alla macro di ricostruzione per abilitare il rumore e specificarne il tipo di distribuzione.
Nel caso di esecuzioni multiple, la fase di generazione è singola, mentre ricostruzione e analisi sono differenti per via di diversi livelli di rumore, riportati nei file *RicostruzioneRumoreN.txt*. Una volta eseguite tutte le simulazioni richieste, viene invocata una ulteriore funzione `PostAnalisi` che permette di comparare tramite dei `MultiGraph` i grafici di efficienza e risoluzione.
Il compito della macro *EsecuzioneEsame.C* è quindi quello di richiamare le macro di generazione *Albero.C*, ricostruzione *Ricostruzione.C* e analisi *Analisi.C*.

La macro *Albero.C* è quella che si occupa della generazione degli eventi. Viene creato il `Tree` "gaggia" con 4 `Branch`: Vertice, UrtiBeamPipe, UrtiRivelatore1 e UrtiRivelatore2. Il primo branch contiene oggetti di classe `Vertice`, mentre gli altri 3 branch sono dei `TClonesArray` di classe `Urto`. All'interno della macro viene letto il file di generazione che contiene i parametri per la simulazione. Per la molteplicità

dell'evento sono supportate le distribuzioni gaussiana, uniforme e fissa, oppure quella contenuta nel file *kinem.root*. Si specifica all'interno del file testuale se utilizzare una distribuzione di pseudorapidità uniforme o la distribuzione contenuta in *kinem.root*.
Per ogni evento, viene scelto il numero di particelle mediante la funzione `DecisioneMolteplicita` e viene generato il vertice tramite l'apposito costruttore.
Per ogni particella da generare, viene calcolata la direzione di propagazione con la funzione `EtaTheta` per l'angolo $\theta$ e campionando da una distribuzione uniforme tra $[0, 2\pi]$ per l'angolo $\phi$. Successivamente, viene generato l'urto sulla beam pipe tramite il metodo `UrtoDaVertice` e gli urti sui due layer tramite il metodo `UrtoDaUrto`. Il tutto viene salvato nel tree ed esportato nel file *Output/Simulazione.root*.

La macro *Ricostruzione.C* legge il tree "gaggia" e i suoi 3 branch. Crea il tree della ricostruzione "rovere", nel quale sono salvati gli urti ricostruiti dopo l'applicazione dello smearing gaussiano e dello scattering multiplo nei branch dei due layer del rivelatore e nel terzo salva il numero di punti di rumore generato. Si cicla su tutti gli eventi della simulazione e per ognuno viene applicato lo smearing gaussiano con la funzione `Smearing`, che, iterando sugli urti per ogni layer, applica lo smearing tramite la funzione `SmearingGaussiano`, metodo di `Urto`. In seguito, viene aggiunto il rumore tramite due possibili funzioni a seconda della distribuzione: `RumoreGaussiano` e `RumoreFissa`. I punti di rumore si aggiungono in fondo ai TClonesArray dei due layer. Il tree viene salvato nel file *Output/Ricostruzione.root* nel caso di singola simulazione oppure nei file *Output/RicostruzioneRumoreN.root* per simulazioni multiple.

La macro *Analisi.C* legge i tree "gaggia" e "rovere" e i relativi branch. Per ogni evento si valutano tutte le possibili coppie di punti su layer 1 e layer 2. Per ogni coppia di punti si valuta la differenza $\Delta\phi$: se questa è minore di una soglia impostata dall'utente, allora viene invocato il metodo `TrovaVertice` di `Vertice` per calcolare tutti i possibili vertici ricostruiti dell'evento con le tracce ipotizzate. Si valuta allora quale sia il vertice più probabile tramite la funzione `Moda`. Essa cerca il primo bin dell'istogramma in cui è presente il valore massimo di conteggi e assegna alla $z_{ricostruita}$ il valore centrale del bin (esempi in Figura 2 e Figura 3). In seguito, controlla che non siano presenti altri massimi a destra: in caso negativo, viene restituito il valore precedentemente trovato. In caso contrario, viene valutata la distanza tra i due bin: se piccola, viene fatta la media tra i due, altrimenti viene scelto il valore centrale minore. In Figura 4 è riportato un esempio di tale situazione.
Infine, si valutano efficienza e risoluzione della simulazione tramite la costruzione di grafici. Tutto viene salvato nel file *Output/Analisi.root* in caso di simulazione singola oppure *Output/Analisi_RumoreN.root* per simulazioni multiple.
Inoltre, in caso di simulazioni multiple, viene invocata la funzione `PostAnalisi`, che apre i vari file di analisi e combina insieme i grafici di efficienza e risoluzione per effettuare un confronto al crescere del rumore. Per stampare i grafici prodotti e salvati nei file *.root* durante l'esecuzione del programma è stata implementata la macro *generaGrafici.C*.

# 3 Risultati

Di seguito vengono riportati i grafici prodotti durante l'analisi.
In Figura 1 viene presentata la distribuzione degli scarti $(z_{ricostruita} - z_{vero})$ lungo l'asse z.

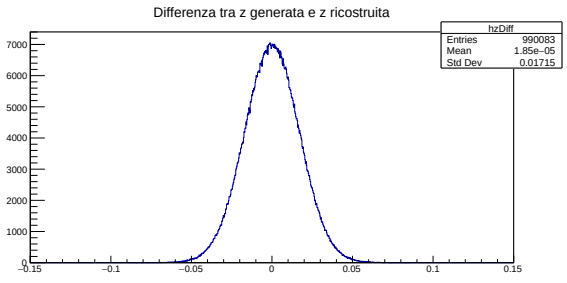


***Figura 1:*** *Distribuzione degli scarti sulla variabile z*

Come spiegato precedentemente, per valutare la coordinata z del vertice ricostruito per ogni evento, è stato costruito un istogramma con tutti i possibili vertici dell'evento. Viene poi valutata la moda della distribuzione e presa come coordinata del vertice. La Figura 2 e Figura 3 sono due esempi di distribuzione in cui la moda è ben definita. Nel primo caso tutti i possibili vertici sono raggruppati attorno al valore medio; nel secondo le possibili coppie di punti ricostruiscono il vertice sia attorno al picco, sia in posizioni più distanti.

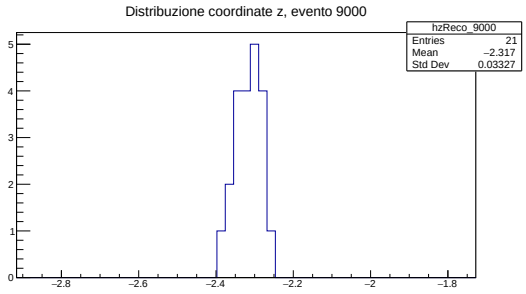


***Figura 2:*** *Istogramma per valutare la moda della distribuzione dei vertici ricostruiti*

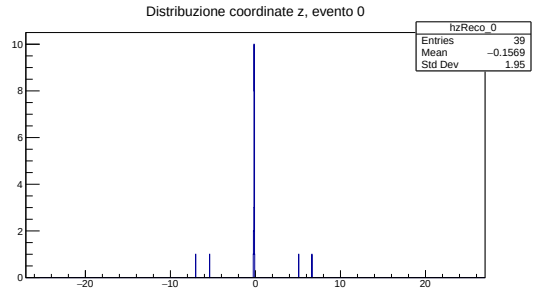


***Figura 3:*** *Istogramma per valutare la moda della distribuzione dei vertici ricostruiti*

Invece, in Figura 4 si può vedere che la distribuzione è bimodale. In questo caso la funzione `Moda` effettua una media dei due massimi se essi sono sufficientemente vicini, altrimenti prende il valore del massimo più vicino allo 0.

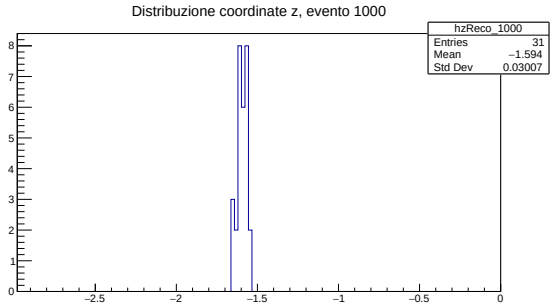


***Figura 4:*** *Istogramma per valutare la moda della distribuzione dei vertici ricostruiti*

Alcune ricostruzioni però possono non dare risultati. In Figura 5 viene riportato il caso in cui non è presente una moda della distribuzione, perché tutte le possibili coppie portano a ricostruire un vertice diverso. In questo caso, il vertice non viene ricostruito.
In Figura 6 viene riportato un grafico vuoto: questo è dovuto al fatto che nessuna delle coppie possibili riesce a ricostruire un possibile vertice.

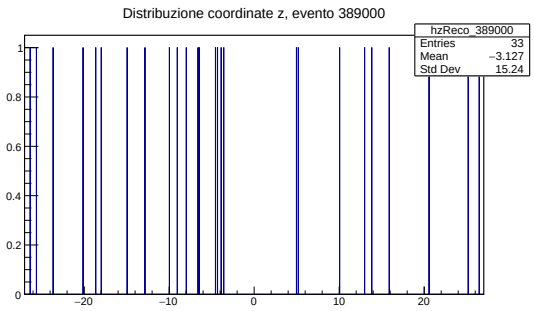


***Figura 5:*** *Istogramma per valutare la moda della distribuzione dei vertici ricostruiti*

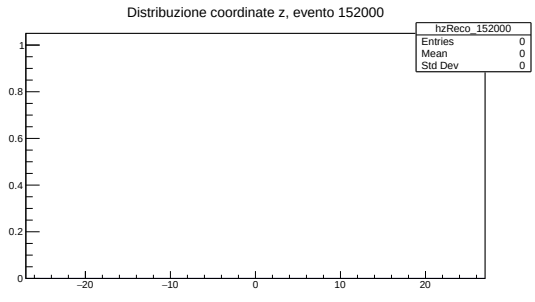


***Figura 6:*** *Istogramma per valutare la moda della distribuzione dei vertici ricostruiti*

Infine vengono proposti dei confronti di ricostruzione in presenza di diverso livello di rumore. Sono state lanciate quattro analisi con livelli di rumore crescente: 0 punti di rumore presenti oppure

10/50/100 punti di rumore per layer. In Figura 8 si può osservare come varia la risoluzione dell'apparato in funzione della molteplicità all'aumentare del rumore sui due layer, mentre in Figura 7 viene riportato il confronto della risoluzione dell'apparato all'aumentare del rumore in funzione della posizione del vertice dell'evento.

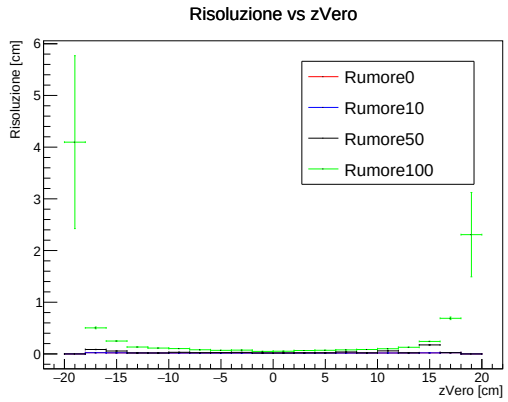


***Figura 7:*** *Confronto della risoluzione dell'apparato in funzione della posizione del vertice*

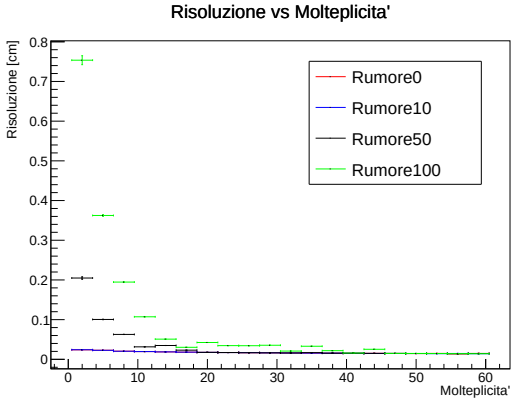


***Figura 8:*** *Confronto della risoluzione dell'apparato in funzione della molteplicità*

In Figura 9 e Figura 10 vengono riportati anche i grafici di confronto in assenza dell'analisi con 100 punti di rumore per layer, per poter osservare meglio le curve che altrimenti vengono schiacciate verso l'asse delle ascisse a causa della curva a rumore maggiore.

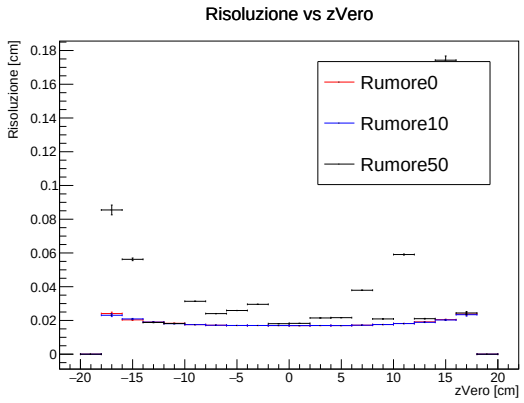


***Figura 9:*** *Confronto della risoluzione dell'apparato in funzione della posizione del vertice escludendo la curva per 100 punti di rumore per layer*

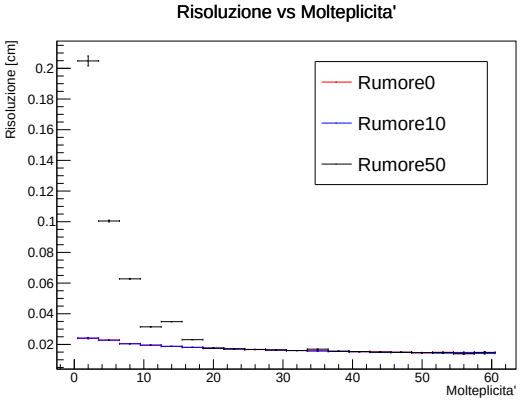


***Figura 10:*** *Confronto della risoluzione dell'apparato in funzione della molteplicità escludendo la curva per 100 punti di rumore per layer*

Analogamente, vengono riportati anche i confronti al variare del rumore per l'efficienza di ricostruzione dell'apparato in funzione della molteplicità dell'evento. In Figura 11 viene valutata l'efficienza di ricostruzione su tutti gli eventi della simulazione, mentre in Figura 12 il confronto escludendo la ricostruzione con 100 punti di rumore. In Figura 13 e Figura 14 si valuta l'efficienza selezionando solo gli eventi che hanno una generazione del vertice entro $1\sigma$ da $z = 0$ e in Figura 15 e Figura 16 l'efficienza selezionando gli eventi entro $3\sigma$ da $z = 0$.

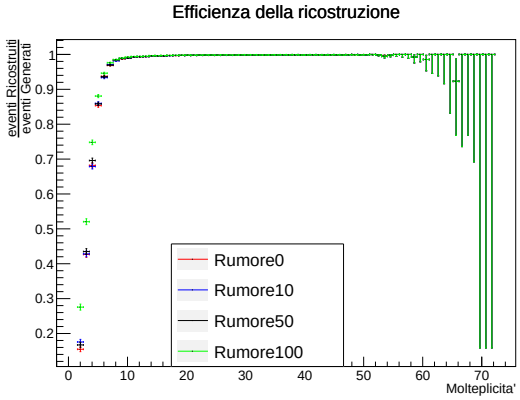


***Figura 11:*** *Efficienza di ricostruzione in funzione della molteplicità valutando tutti gli eventi*

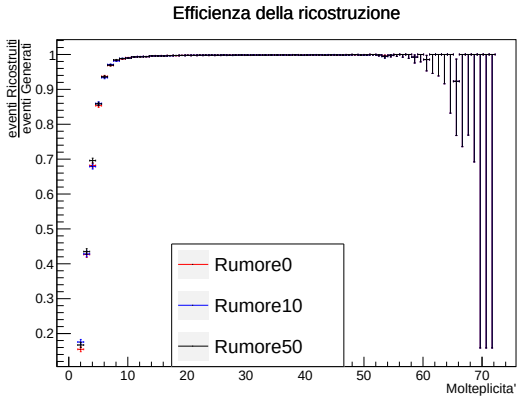


***Figura 12:*** *Efficienza di ricostruzione in funzione della molteplicità valutando tutti gli eventi escludendo l'analisi con 100 punti di rumore*

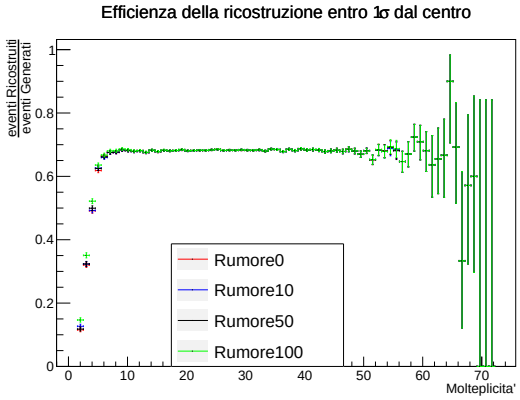


***Figura 13:*** *Efficienza di ricostruzione in funzione della molteplicità valutando gli eventi entro 1σ*

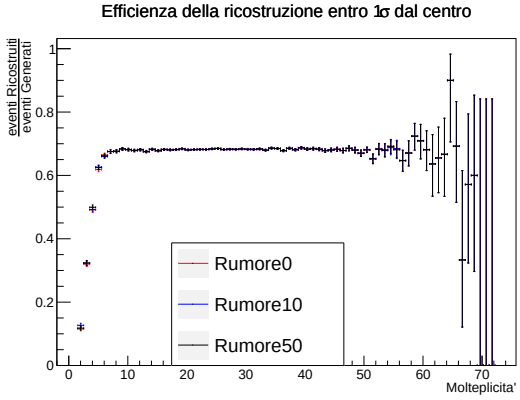


***Figura 14:*** *Efficienza di ricostruzione in funzione della molteplicità valutando gli eventi entro 1σ escludendo l'analisi con 100 punti di rumore*

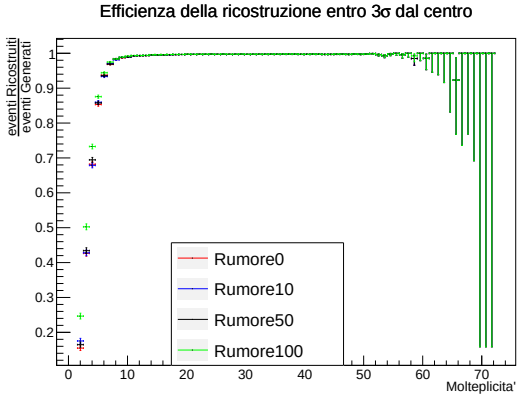


***Figura 15:*** *Efficienza di ricostruzione in funzione della molteplicità valutando gli eventi entro 3σ*

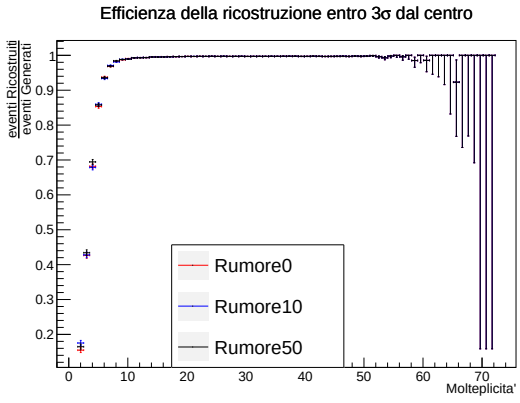


***Figura 16:*** *Efficienza di ricostruzione in funzione della molteplicità valutando gli eventi entro 3σ escludendo l'analisi con 100 punti di rumore*