TANS 2025 - Zanusso

Emanuele Zanusso

2024-2025

Indice

1 Introduzione e macrostruttura

Il presente progetto inscena i processi di simulazione e ricostruzione FAST2, che avvengono in un esperimento a collider, nell'ipotesi che le particelle siano ultra relativistiche. Quest'ultima permette di trascurare l'effetto di un ipotetico campo magnetico, implicando solo traiettorie rettilinee.
Esso si articola in 4 directories principali:

- **work_TANS**: custodisce il cuore della consegna, ossia le macro per avviare la simulazione, la rivelazione, la ricostruzione e per realizzare un event display degli eventi simulati
- **classes_TANS**: vi sono le classi impiegate durante lo svolgimento della consegna
- **dat_TANS**: raccoglie le distribuzioni utili sulla cinematica delle particelle, oltre ai risultati (organizzati in TTree) delle suddette fasi, conservati all'interno di file .root
- **results_TANS**: raccoglie le macro per la produzione dei risultati relativi alle performance

2 Classi

2.1 points_TANS

- **cPoint**: implementa il concetto di punto (x, y, z) in coordinate cartesiane
- **cilPoint**: classe che implementa il concetto di punto (R, ϕ, Z) in coordinate cilindriche. Utile per lo sparpagliamento dovuto allo smearing
- **pAng**: classe di supporto per trattare gli angoli, soprattutto per collegare la grandezza pseudorapidità η all'angolo θ

2.2 detectors_TANS

- **dLayer**: classe che implementa il concetto di detector di raggio r, lunghezza l e spessore w. Costruisce anche l'ambiente per l'event display, sfruttando la classe TGeo
- **smLayer**: classe di supporto che implementa lo smearing dovuto al processo di rivelazione e di generazione di rumore nei rivelatori

2.3 phy_TANS

- **pKinematic**: classe che si incarica della cinematica legata alle particelle in relazione ai rivelatori (controllare che stiano dentro ai layer e determinare le intersezioni)

- **rTklet**: classe che, in fase di ricostruzione, permette di accoppiare i ϕ di diversi layer, determinando potenziali z_{rec}. Quest'ultimi vengono scremati e raccolti in un unico z_{rec}, attraverso un processo di running window (dimensionabile) su un vector
- **multS**: classe di supporto che implementa il multiple scattering dovuto al processo fisico di scattering, ruotando un vettore di componenti angolari (ϕ, θ)
- **lHit**: classe che connubia un cPoint al raggio del Layer. Serve per descrivere un hit, fornendone coordinate e layer su cui avviene

2.4 infrast_TANS

- **sConj**: classe di infrastruttura utile ad alcune stampe

3 Processo FAST2

Si intende ora fornire un esempio operativo FAST2 riguardante la sequenza:

$$simulazione \rightarrow rivelazione \rightarrow ricostruzione \rightarrow residui \rightarrow event\ display$$

I risultati mostrati in questa sede sono stati ottenuti attivando le seguenti opzioni:

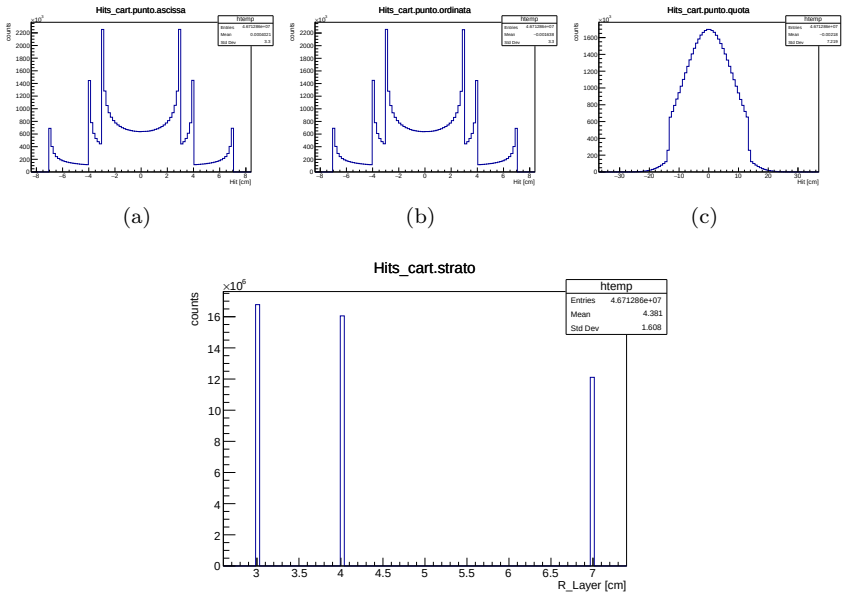
- **Eventi** = 1M , **Multiple Scattering**: Enabled, **Molteplicità**: "hm"
- **Noise**: Enabled (0.5%)
- **Running Window**: 1σ

Preservando la gerarchia delle directories, ci si colloca in ../work_TANS, per poi aprire una sessione ROOT.
In primo luogo, si carica tutto il necessario; si interpreta la macro .x compiler_TANS.C, che serve a caricare tutte le classi e le macro operative. Tempo impiegato: circa 2 minuti.

3.1 Simulazione

Si inizia lanciando una simulazione attraverso la funzione simulaz_TANS (int n_eve, bool MS, string mul_sel), che ha diversi argomenti, come il numero di eventi da simulare (collisioni p-p), la presenza di multiple scattering o meno e la scelta della distribuzione di molteplicità, che può essere quella assegnata ("*hm*"), quella uniforme ("*um*") o un valore fisso ("*numero*"). Il programma è creato in modo tale per cui verrebbe stampata una frase di controllo e di guida, qualora si sbagliasse input.
A seguito del completamento della simulazione, viene salvato un file, che di default si chiama "MC_truth.root", ovvero la realtà fisica estranea ai processi di

rivelazione, all'interno della directory dat_TANS. Questo file contiene un TTree che conserva variabili come la posizione del vertice di ogni evento, la molteplicità, la posizione degli hit sulla beam pipe e sui rivelatori e gli angoli.
Si riportano la distribuzioni, sugli assi (x,y,z,R), degli hit su tutti e tre gli strati:

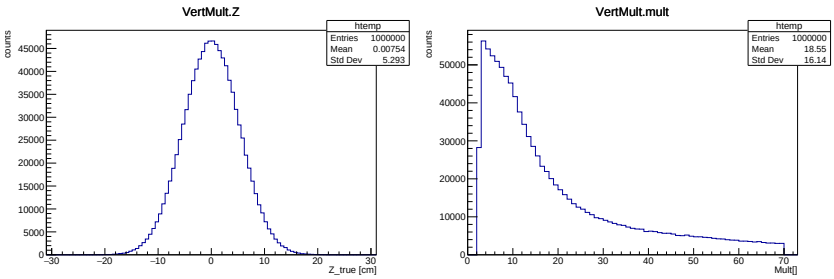


(a) (b) (c)

(d)

Figure 1: (a) asse x (b) asse y (c) asse z (d) hit sul raggio

Si può notare che anche la posizione sulla beam pipe viene contemplata, in quanto la realtà Monte Carlo rappresenta il puro evento fisico, estraneo al processo di rivelazione.
Si riportano di seguito le caratteristiche del vertice d'interazione, in particolare la distribuzione della quota Z_{true} e la distribuzione della molteplicità, che in questo esempio è quella assegnata ("hm"):

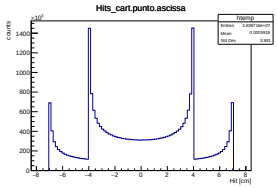


(a) (b)

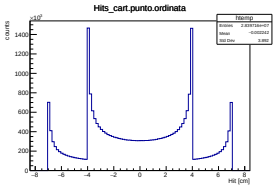
Figure 2: (a) Z_{true} (b) Molteplicità

3.2 Rivelazione

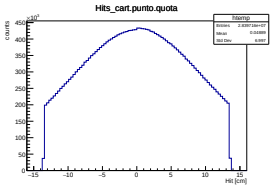
Dopo aver creato il processo fisico puro, si avvia il processo di rivelazione. Pertanto, si lancia la funzione dataReal_TANS (bool spur, double noise). Questa ricrea il comportamento dei due layer in silicio, implementando anche la risposta in termini di risoluzione del rivelatore, attraverso un processo di smearing gaussiano.
Gli argomenti sono opzionabili affinché venga anche inscenata la presenza di rumore (spur = 1) o meno (spur = 0) e, nel primo caso, si può anche decidere, tunando opportunamente l'argomento noise, la quantità di rumore da iniettare. Se noise = 0, non si ha rumore. Se noise = 1, si è sommersi totalmente dal rumore. Si riportano di seguito le precedenti e medesime grandezze :



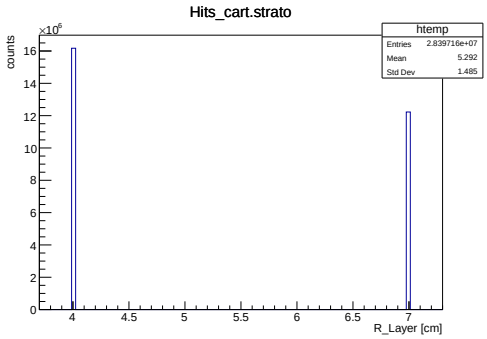
(a)



(b)



(c)



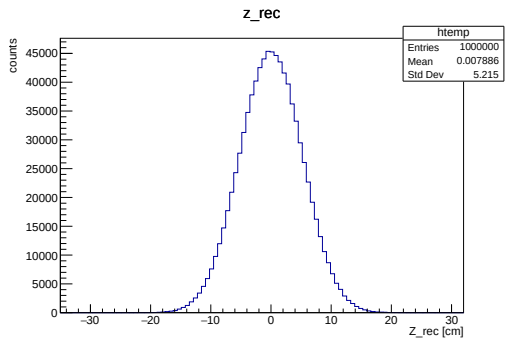
(d)

Figure 3: (a) asse x (b) asse y (c) asse z (d) hit sul raggio

Diversamente da prima, adesso viene evidenziato il comportamento attivo dei layers in silicio rispetto alla beam pipe, come si può notare.
Viene creato il file "data.root", che rappresenta il pool di dati simulato a seguito della rivelazione. Al suo interno vi sono gli hit sui layer 2 e 3, che corrispondono ai rivelatori. Questo è il dataset reale, ovvero quello che ci si aspetta di ottenere a seguito di una presa dati.

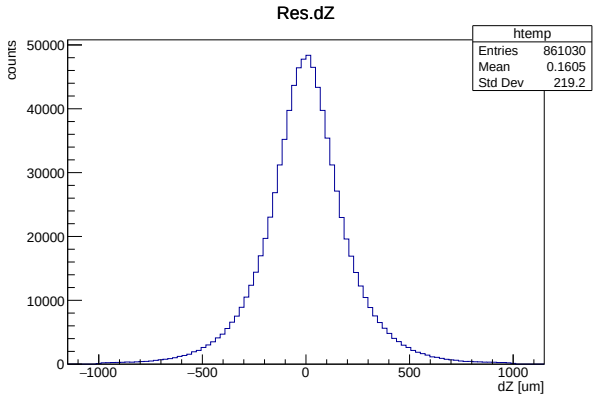
3.3 Ricostruzione

Si avvia un processo di ricostruzione del vertice di questi dati simulati presenti sul file "data.root", chiamando la funzione ricostruz_TANS(double window), che si incarica di prendere gli oggetti creati dalla rivelazione e di determinare la posizione del vertice evento per evento, salvando tutto nel file "ricostruz.root". Si può scegliere quanto essere "elastici" nella restrizione del dataset z_rec attraverso la scelta di window, che modula la rispettiva finestra di taglio in unità di σ_z. Di seguito si riporta la distribuzione della quota ricostruita relativa al vertice d'interazione Z_{rec}:



3.4 Residui tra Z_{true} e Z_{rec}

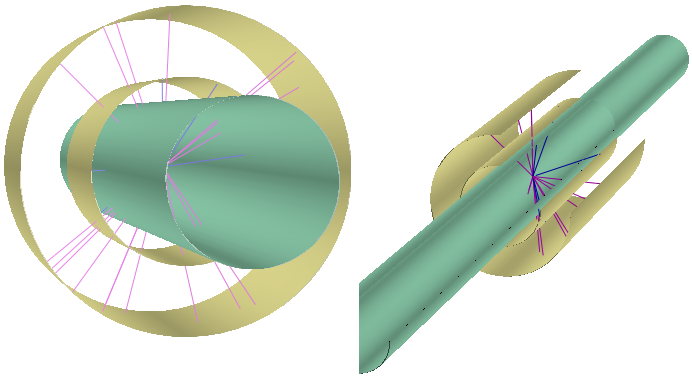
Si lancia la funzione " residues_TANS () " che preleva Z_true e Z_rec dai rispettivi TTree e ne produce il residuo. Si evidenzia il fatto che quest'ultimo è relativo alla distribuzione scelta in partenza, quindi comprende tutti i contributi in termini di molteplicità e restituisce una distribuzione che è, in generale, non gaussiana:



Le distribuzioni gaussiane relative ai vari tagli di molteplicità vengono presentate nel sezione 4.

3.5 Event display

Si lancia la funzione "TGeo_personal(int ev)", che si incarica di svolgere un event display (molto rudimentale) della realtà Monte Carlo e che come argomento ha il numero dell'evento che si vuole rappresentare. Viene ristampato il risultato finale, ossia il Z_true e Z_rec. Per una miglior interfaccia, si consiglia di utilizzare un 3D Viewer quale OpenGL, selezionabile direttamente dal TCanvas in uscita. L'interfaccia che si presenta è la seguente:



(a) (b)

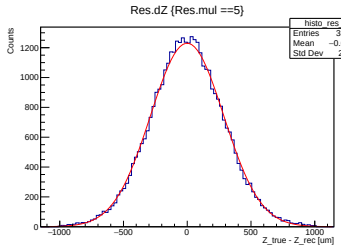
Figure 4: (a) evento d'esempio (b) sezione dell'evento d'esempio

4 Performance

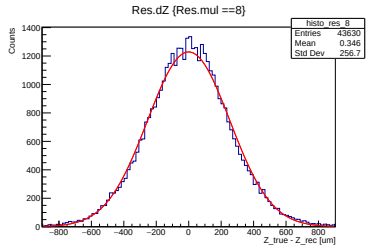
Si intende adesso fornire un andamento della risoluzione e dell'efficienza, in relazione alla molteplicità e a Z_{true}. A tal fine, ci si colloca all'interno della directory "results_TANS" e si interpreta la macro comp_TANS.C.

4.1 Residui e molteplicità

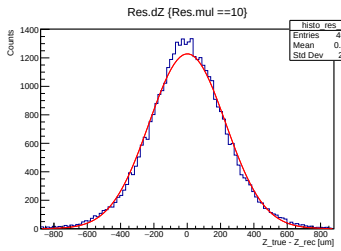
Si lancia la funzione "all_residues()". Essa preleva l'istogramma dei residui e si incarica di rappresentare l'andamento del residuo "dZ", al variare della molteplicità selezionata attraverso dei tagli sulla variabile "mul" applicati direttamente al sovrastante istogramma dei residui. I risultati vengono salvati nel TTree presente in "residues.root".
In questo contesto, dZ riceve un unico contributo in termini di molteplicità e, pertanto, si distribuisce secondo una gaussiana. Si riportano di seguiti i risultati ottenuti, per un sottoinsieme di molteplicità selezionate:



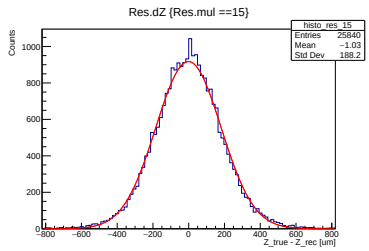
(a)



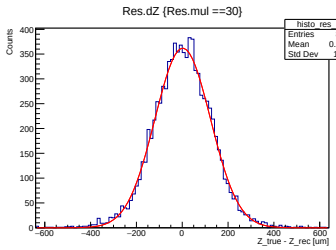
(b)



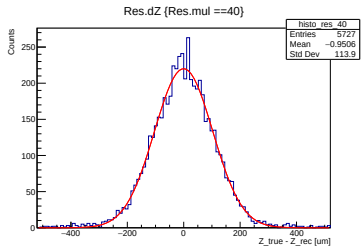
(c)



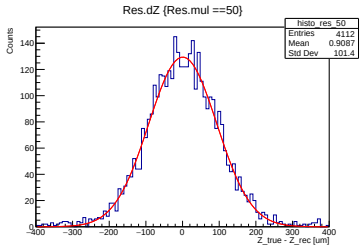
(d)



(e)



(f)

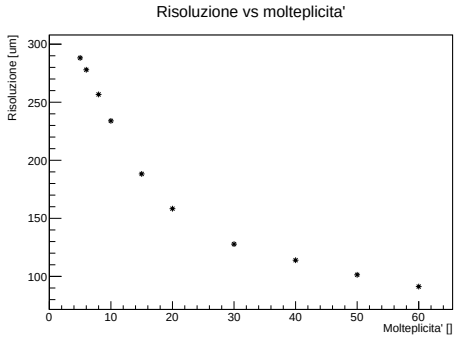


(g)

Si evidenzia come la risoluzione, identificabile nella dispersione di ciascuna gaussiana, diminuisca all'aumentare del valore di molteplicità selezionato. Questo sarà il tema della prossima sottosezione.

4.2 Risoluzione vs molteplicità

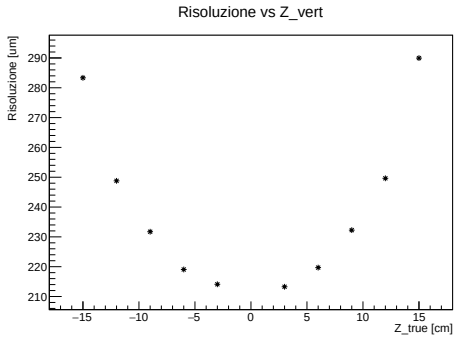
Si vuole adesso descrivere l'andamento della risoluzione rispetto al numero di particelle cariche prodotte nella collisione. Come già detto, ci si attende che la prima migliori al crescere della seconda:



Si può concludere che la risoluzione si assesti a valori inferiori a 100 μm, per valori di molteplicità superiori a 50.

4.3 Risoluzione vs Z_{rec}

Si vuole adesso descrivere l'andamento della risoluzione rispetto al taglio sulla quota Z_{true} del vertice. Si ottiene il seguente andamento:



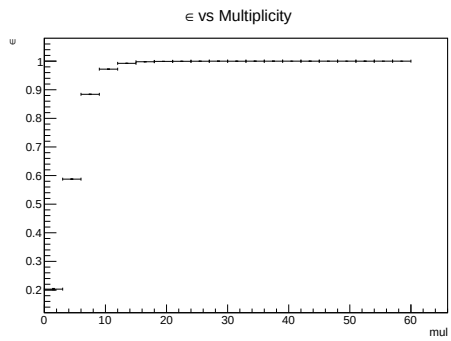
Si può osservare come la risoluzione sia migliore per punti centrali al rivelatore e come vada a degradarsi ai bordi.

4.4 Efficienza vs molteplicità

Si può definire l'efficienza ϵ come:

$$\epsilon = \frac{number\ of\ Z_{rec}}{number\ of\ Z_{true}}$$

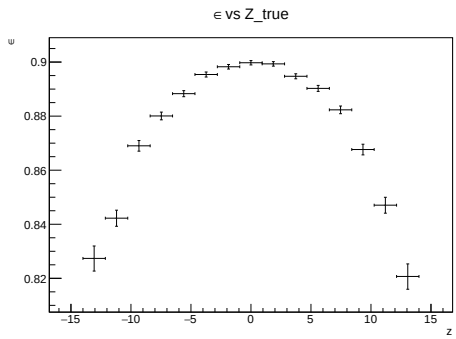
Di seguito si riporta l'andamento di ϵ, al variare della molteplicità:



Si evidenzia come l'efficienza migliori al crescere della molteplicità.

4.5 Efficienza vs Z_{true}

Si conclude mostrando l'andamento di ϵ, al variare di Z_{true}:



Si può notare come, nel presente caso, l'efficienza massima è $\simeq 90\%$ per regioni centrali del rivelatore.