(Toy) MonteCarlo

Matteo Duranti

matteo.duranti@pg.infn.it

(cfr. https://space.umd.edu/dch/p405s06/sullivan_geometry_factor.pdf)

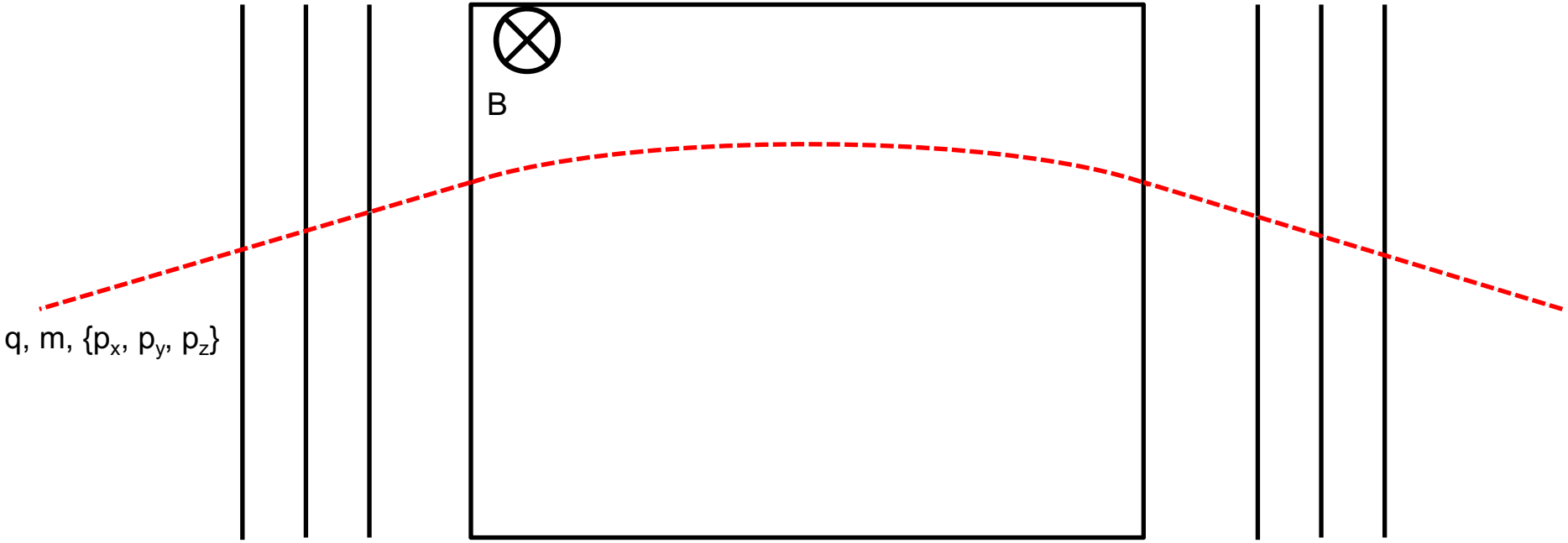
Esercitazione finale

L'esercitazione finale, tipicamente è un (Toy-)MonteCarlo:

- https://www.fisgeo.unipg.it/~duranti/Sito//MetodiComputazionali-2018_2019-files_files/EsercitazioneFinale.pdf
- https://www.fisgeo.unipg.it/~duranti/Sito//MetodiComputazionali-2017_2018-files_files/Esercitazione_finale.pdf
- https://www.fisgeo.unipg.it/~duranti/Sito//MetodiComputazionali-2019_2020-files_files/EsercitazioneFinale.pdf

Esempio realistico

Vogliamo simulare uno "spettrometro magnetico" sottoposto a particelle cariche



magnete:
il campo (uniforme) curva le particelle cariche

rivelatori al silicio:
tracciano le particelle incidenti

rivelatori al silicio:
tracciano le particelle incidenti

Generatore

Quanti eventi generiamo? Cosa utilizziamo per generare? …

Generatore

Quanti eventi generiamo? Cosa utilizziamo per generare? …

- generator random:

```
// --- random object ---
static TRandom3* tran =NULL;
if(!tran){
  tran= new TRandom3();
  tran->SetSeed(76);//this gives a "good" first event
}
```

Generatore

Quanti eventi generiamo? Cosa utilizziamo per generare? …

- generator random:

```
// --- random object ---
static TRandom3* tran =NULL;
if(!tran){
  tran= new TRandom3();
  tran->SetSeed(76);//this gives a "good" first event
}
```

- definiamo le variabili che ci servono:

```
double iDir[3];

unsigned long long int NTRIG = TMath::Power(2, 27) - 1;//permitted up to 2^32 -1 //BUT YOU CAN HAVE A PROBLEM WITH
    THE MAXIMUM NUMBER OF ENTRIES FILLABLE IN A HISTO BIN!!!!
//unsigned long long int NTRIG = 1;
//unsigned long long int NTRIG = 3;

// ******* START THE SIMULATION ***********
float pperc=0.;
float perc;
unsigned int nentries= NTRIG;
printf("-------------------------------------------------------------------------\n");
printf("Simulating %u events\n", nentries);
printf("-------------------------------------------------------------------------\n");
```

Generatore

Quanti eventi generiamo? Cosa utilizziamo per generare? …

- generator random:

```
// --- random object ---
static TRandom3* tran =NULL;
if(!tran){
  tran= new TRandom3();
  tran->SetSeed(76);//this gives a "good" first event
}
```

- definiamo le variabili che ci servono:

```
double iDir[3];

unsigned long long int NTRIG = TMath::Power(2, 27) - 1;//permitted up to 2^32 -1 //BUT YOU CAN HAVE A PROBLEM WITH
    THE MAXIMUM NUMBER OF ENTRIES FILLABLE IN A HISTO BIN!!!!
//unsigned long long int NTRIG = 1;
//unsigned long long int NTRIG = 3;

// ******* START THE SIMULATION ***********
float pperc=0.;
float perc;
unsigned int nentries= NTRIG;
printf("-------------------------------------------------------------------------\n");
printf("Simulating %u events\n", nentries);
printf("-------------------------------------------------------------------------\n");
```

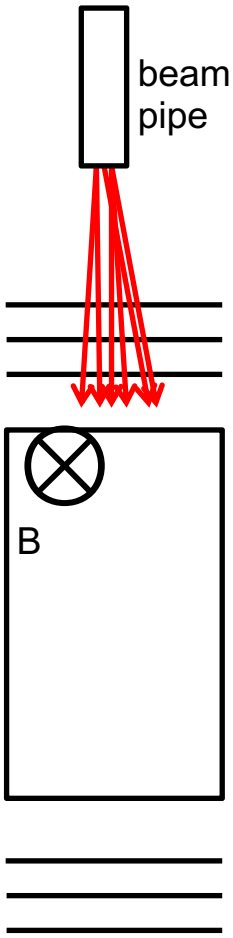
- loop sugli eventi

```
for(unsigned long long int ii=0;ii<nentries;ii++){
  perc=ii/(nentries*1.);
  if (perc>=pperc){
    printf("Processed %5.0f%%\n", pperc*100);
    pperc+=0.01;
  }
```

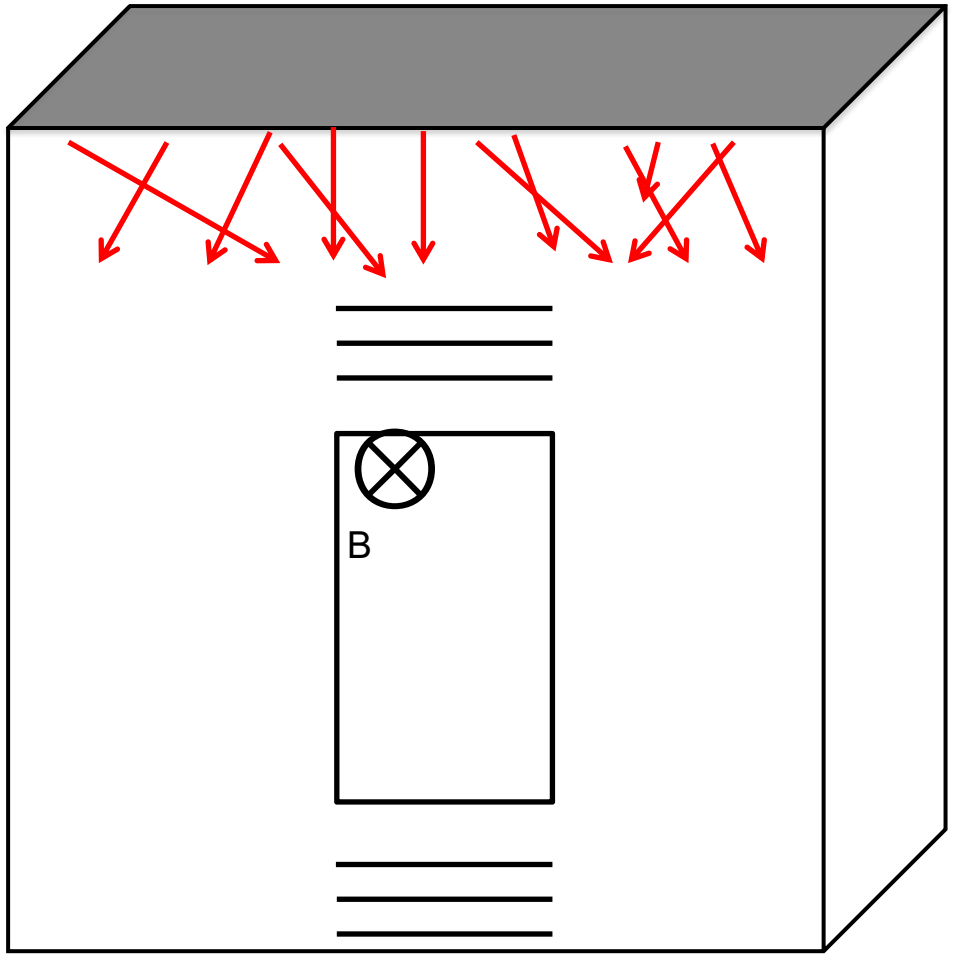
Generatore

Come arrivano le particelle incidenti lo spettrometro?

Fascio di particelle:

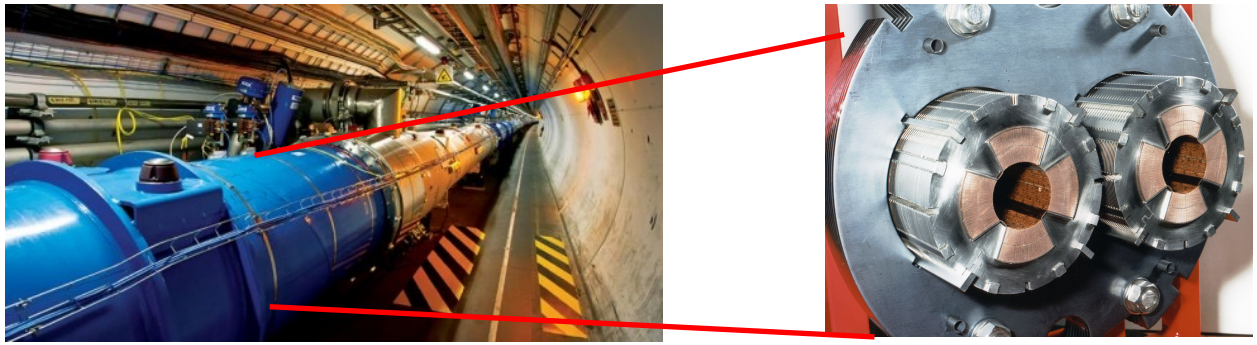


Isotropicamente:



Fascio di particelle

Beam pipe:



"spot size":

- ~ 1 mm di larghezza (esempio)
- idealmente gaussiano

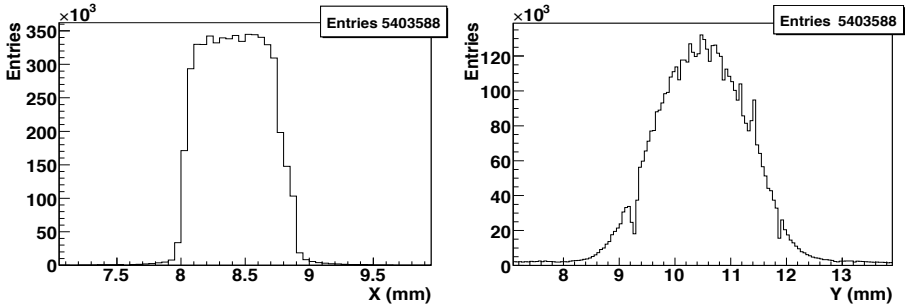


Figura 3.15: Profili orizzontale (sinistra) e verticale (destra) del fascio.

"divergenza":

- ~ 10 μrad
- idealmente gaussiana

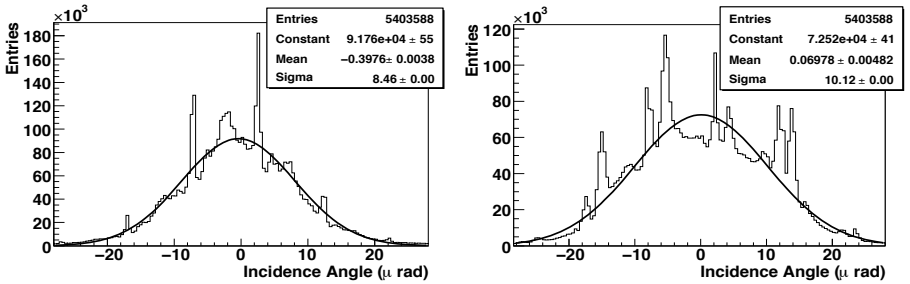


Figura 3.16: Distribuzioni dell'angolo di incidenza del fascio al cristallo nel piano orizzontale (sinistra) e verticale (destra). La divergenza del fascio viene definita dalla larghezza della funzione gaussiana utilizzata per descrivere tali distribuzioni. I picchi nelle distribuzioni sono dovuti alla risoluzione dei rivelatori.

Generatore

Come arrivano le particelle incidenti lo spettrometro?

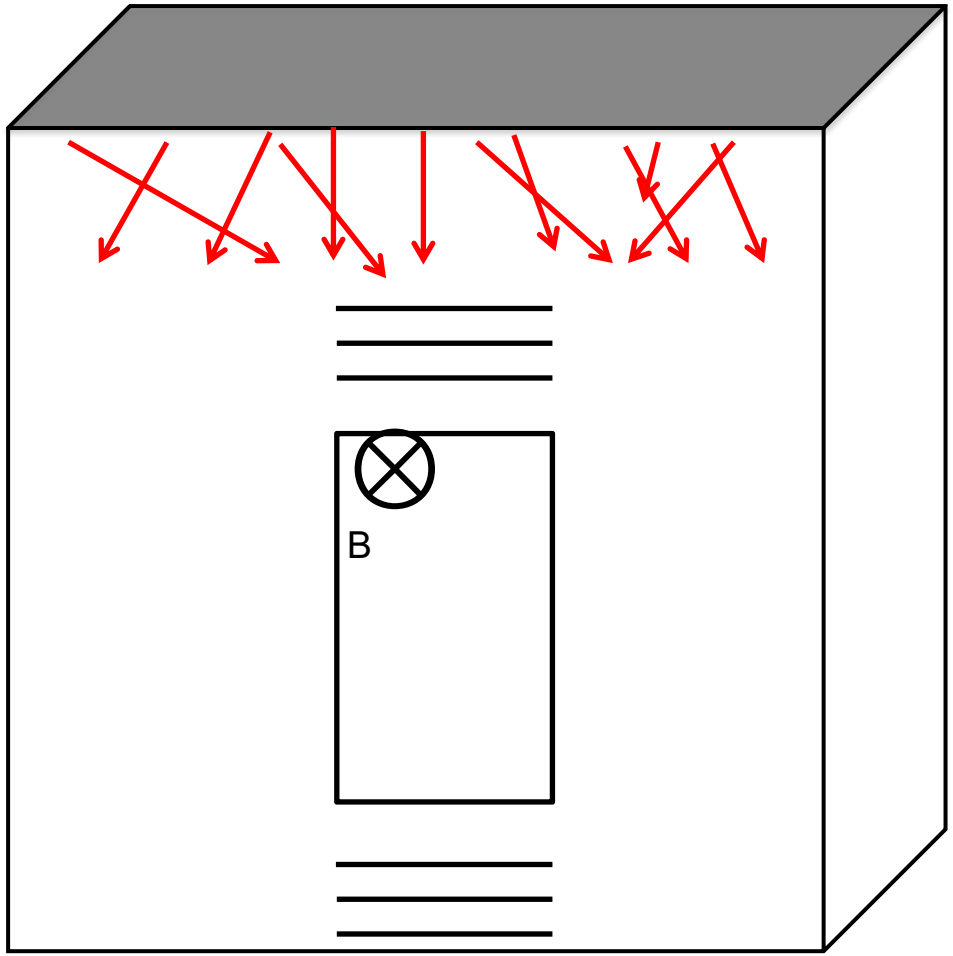
Fascio di particelle:

```
const double PI= TMath::Pi();

// beam pipe position
Double_t BeamPos[3] = {0, 0, -10000}; //mm

// beam spot dimension (RMS)
Double_t Xwidth = 1.0 //mm
Double_t Ywidth = 1.0 //mm

// beam divergence (RMS)
Double_t BeamDivergence = 1.0 //murad
```

```
// gen point on beam pipe
Double_t Xg = 0.0;
Xg = tran->Gaus(BeamPos[0], Xwidth);
Double_t Yg = 0.0;
Yg = tran->Gaus(BeamPos[1], Ywidth);
Double_t Zg = 0.0;
Zg = BEamPos[2];

// gen direction
Double_t theta = 0.0;
theta = tran->Gaus(0.0, BeamDivergence);
Double_t phi = 0.0;
phi = tran->Uniform(0, 2.*PI);
```

*la gaussiana su θ_x e θ_y in realtà non si fa così

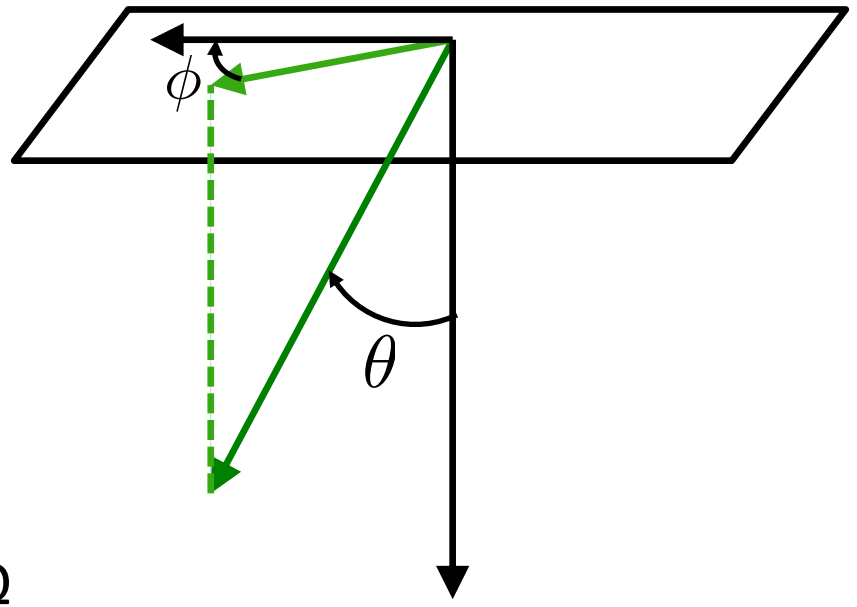
Isotropicamente:



Generazione isotropa

Flusso isotropo, Φ:

$$\vec{p} = \{p_x, p_y, p_z\} \rightarrow \{|p|, \theta, \phi\}$$



Isotropo significa che

$$\frac{d\Phi}{d\Omega} = \frac{d\Phi}{d\phi\, d\cos\theta} = k$$

Il numero di particelle che attraversano un elemento di area, Ω

$$N \propto \int_{\Sigma} \vec{\Phi} \cdot d\vec{\sigma} \implies \frac{dN}{d\phi\, d\cos\theta} \propto \cos\theta$$

$$\frac{dN}{d\phi\, \cos\theta\, d\cos\theta} \sim k \sim \frac{dN}{d\phi\, d\cos^2\theta}$$

ref. Sullivan, NIM 95 (1971) 5-11

Generatore

Come arrivano le particelle incidenti lo spettrometro?

Fascio di particelle:

```
const double PI= TMath::Pi();

// beam pipe position
Double_t BeamPos[3] = {0, 0, -10000}; //mm

// beam spot dimension (RMS)
Double_t Xwidth = 1.0 //mm
Double_t Ywidth = 1.0 //mm

// beam divergence (RMS)
Double_t BeamDivergence = 1.0 //murad
```

```
// gen point on beam pipe
Double_t Xg = 0.0;
Xg = tran->Gaus(BeamPos[0], Xwidth);
Double_t Yg = 0.0;
Yg = tran->Gaus(BeamPos[1], Ywidth);
Double_t Zg = 0.0;
Zg = BEamPos[2];

// gen direction
Double_t theta = 0.0;
theta = tran->Gaus(0.0, BeamDivergence);
Double_t phi = 0.0;
phi = tran->Uniform(0, 2.*PI);
```

Isotropicamente:

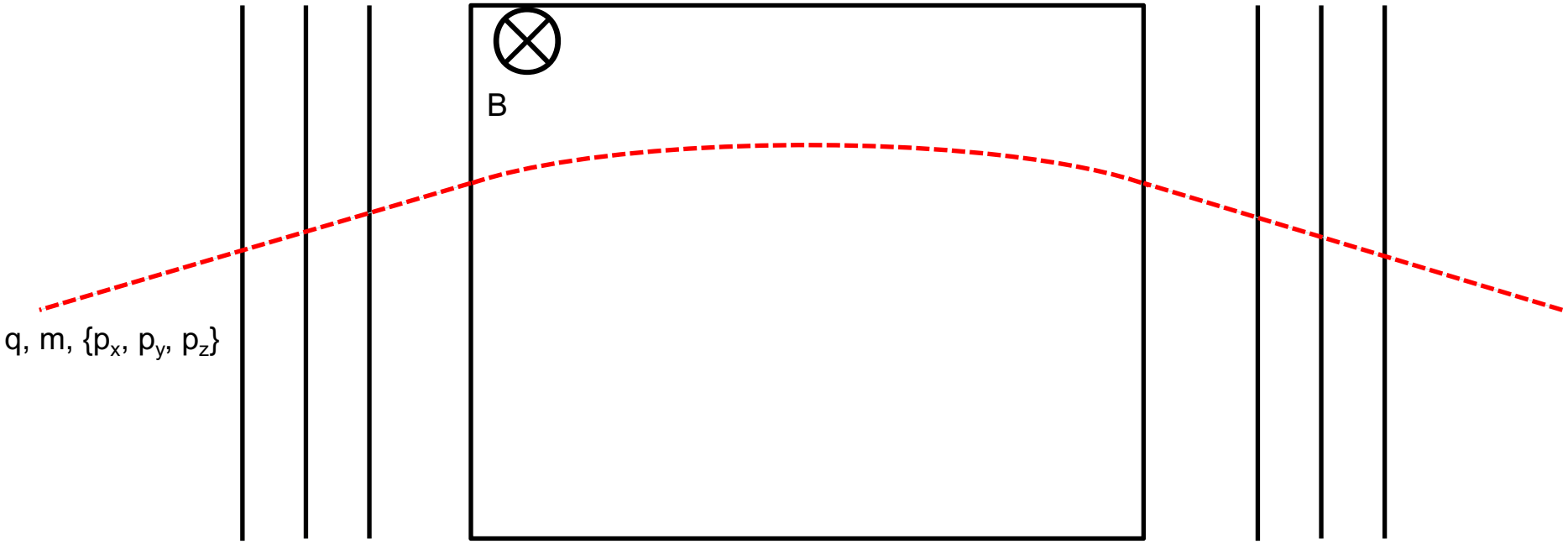
```
// top plane features
Double_t Lg = 3.900; // m
Double_t Zg = (3.900/2.0); // m
```

```
// generation features | theta is wrt positive z-axis
const double PI= TMath::Pi();
Double_t THETA1= PI/2.0;
Double_t THETA2= PI;
Double_t PHI1= 0.;
Double_t PHI2= 2.*PI;
```

```
// --- gen point on top plane surface ---
double Xg = 0.0;
Xg = tran->Uniform( -Lg/2., Lg/2.);
double Yg = 0.0;
Yg = tran->Uniform(-Lg/2., Lg/2.);

// --- gen particle direction ---
double ctheta2 = 0.0;
ctheta2 = tran->Uniform( cos(THETA1)*cos(THETA1), cos(THETA2)*cos(THETA2) );
double phi = 0.0;
phi = tran->Uniform(PHI1, PHI2);

double theta = 0.0;
theta = PI - acos( sqrt(ctheta2) );
```

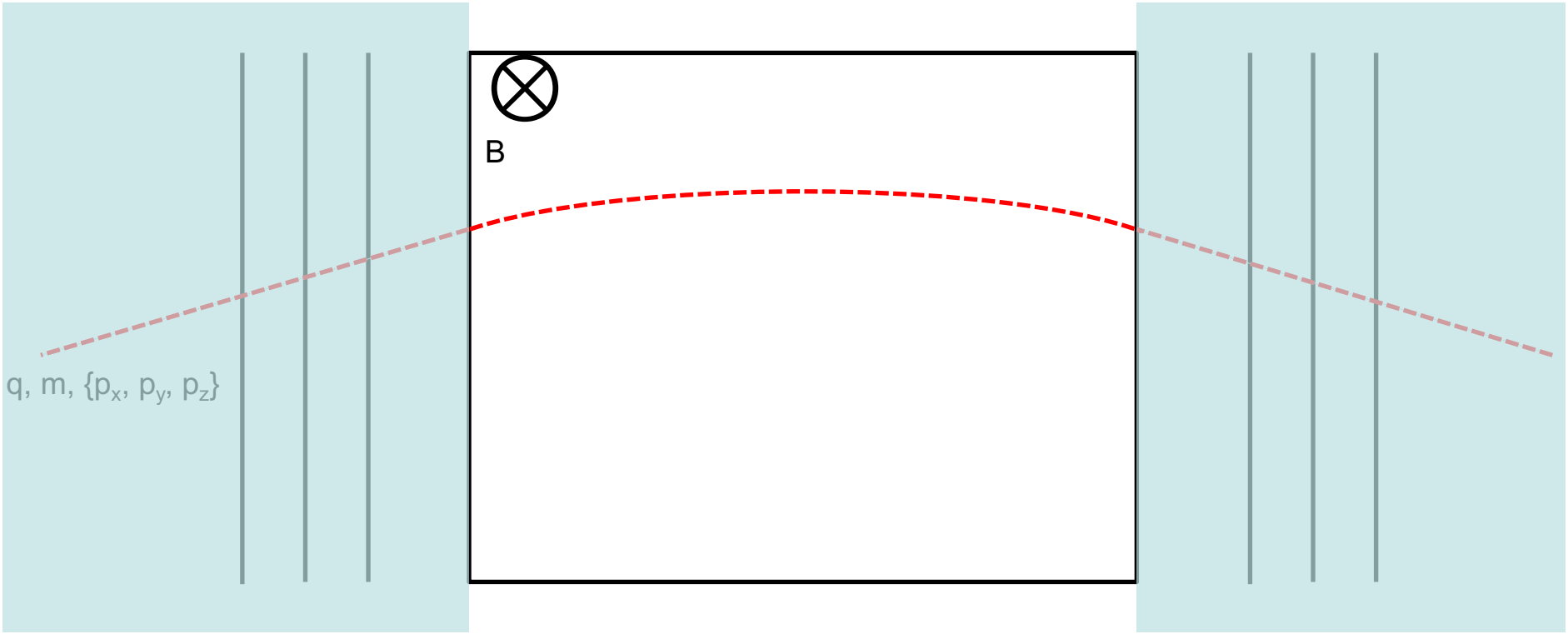
Propagazione

Una volta generate le particelle le dobbiamo far propagare:



Propagazione

Una volta generate le particelle le dobbiamo far propagare:

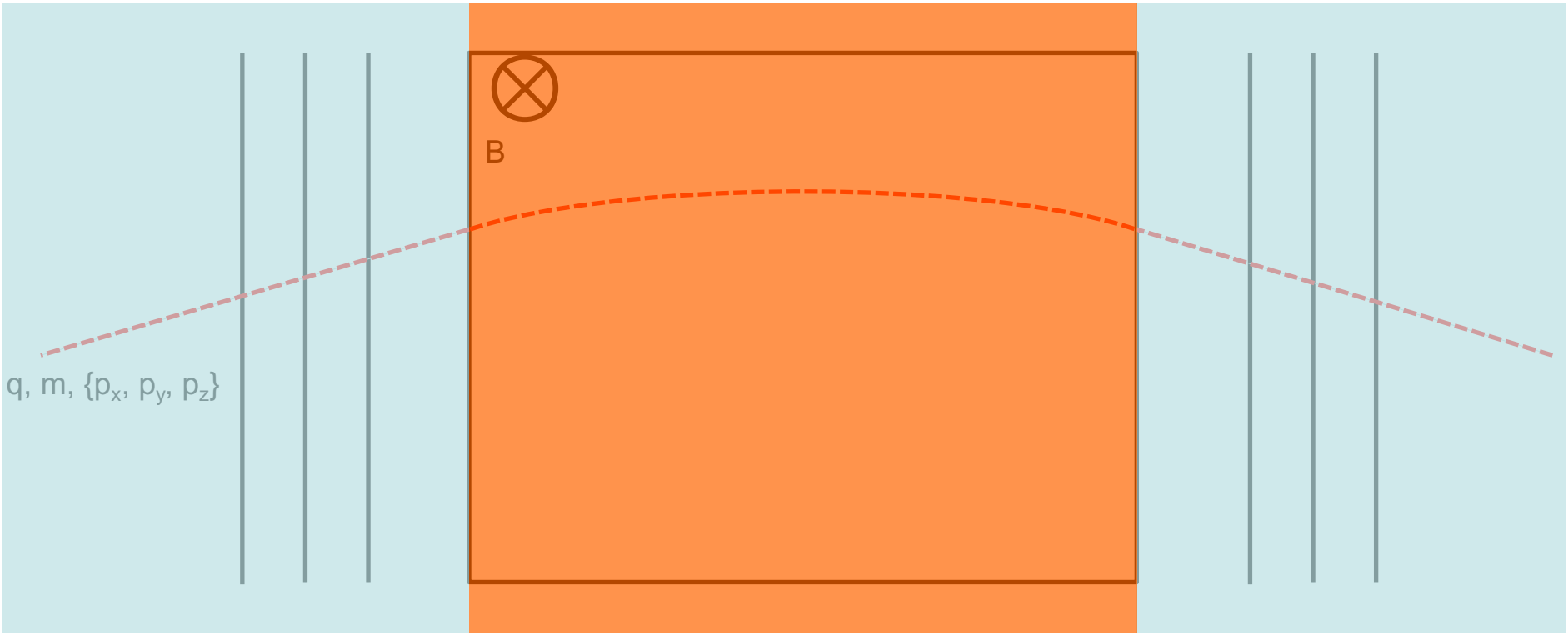


Fuori dal campo magnetico

Fuori dal campo magnetico

Propagazione

Una volta generate le particelle le dobbiamo far propagare:



Fuori dal campo magnetico

Nel campo magnetico

Fuori dal campo magnetico

Propagazione

Come si propagano le particelle?

Fuori dal campo magnetico:

```
// **** step 2 / propagation ****

// --- directions ---
iDir[0] = sin(theta)*cos(phi);
iDir[1] = sin(theta)*sin(phi);
iDir[2] = cos(theta);

Ngen++;
```

```
  //at the detector position 1
  double tt1 = 0.0;
  tt1 = (Zd1-Zg)/iDir[2];

  double Xd1= 999999999.9;
  Xd1 = Xg + tt1*iDir[0];
  double Yd1 = 999999999.9;
  Yd1 = Yg + tt1*iDir[1];
```

Propagazione

Come si propagano le particelle?

Fuori dal campo magnetico:

```
// **** step 2 / propagation ****

// --- directions ---
iDir[0] = sin(theta)*cos(phi);
iDir[1] = sin(theta)*sin(phi);
iDir[2] = cos(theta);

Ngen++;
```

```
  //at the detector position 1
  double tt1 = 0.0;
  tt1 = (Zd1-Zg)/iDir[2];

  double Xd1= 999999999.9;
  Xd1 = Xg + tt1*iDir[0];
  double Yd1 = 999999999.9;
  Yd1 = Yg + tt1*iDir[1];
```

Nel campo magnetico:

```
printf("Initial:\n");
printf("r[3]={%f, %f, %f};\n", r[0], r[1], r[2]);
printf("p[3]={%f, %f, %f};\n", p[0], p[1], p[2]);

double boost = InitializePars(q, m, p, r);
double beta[3];
for (int ii=0; ii<3; ii++){
  beta[ii]=p[ii]/(m*boost);
}
printf("Beta: vx=%f c , vy=%f c, vz=%f c\n", beta[0], beta[1], beta[2]);
```

```
printf("Starting tracing!\n");
errors+=IntegrateMotion(start, start, stop, step, errperc, p, r, sp, sB,
```

Propagazione

Come si propagano le particelle?

Fuori dal campo magnetico:

```
// **** step 2 / propagation ****

// --- directions ---
iDir[0] = sin(theta)*cos(phi);
iDir[1] = sin(theta)*sin(phi);
iDir[2] = cos(theta);

Ngen++;
```

```
//at the detector position 1
double tt1 = 0.0;
tt1 = (Zd1-Zg)/iDir[2];

double Xd1= 999999999.9;
Xd1 = Xg + tt1*iDir[0];
double Yd1 = 999999999.9;
Yd1 = Yg + tt1*iDir[1];
```

Nel campo magnetico:

```
printf("Initial:\n");
printf("r[3]={%f, %f, %f};\n", r[0], r[1], r[2]);
printf("p[3]={%f, %f, %f};\n", p[0], p[1], p[2]);

double boost = InitializePars(q, m, p, r);
double beta[3];
for (int ii=0; ii<3; ii++){
  beta[ii]=p[ii]/(m*boost);
}
printf("Beta: vx=%f c , vy=%f c, vz=%f c\n", beta[0], beta[1], beta[2]);
```

```
printf("Starting tracing!\n");
errors+=IntegrateMotion(start, start, stop, step, errperc, p, r, sp, sB,
```

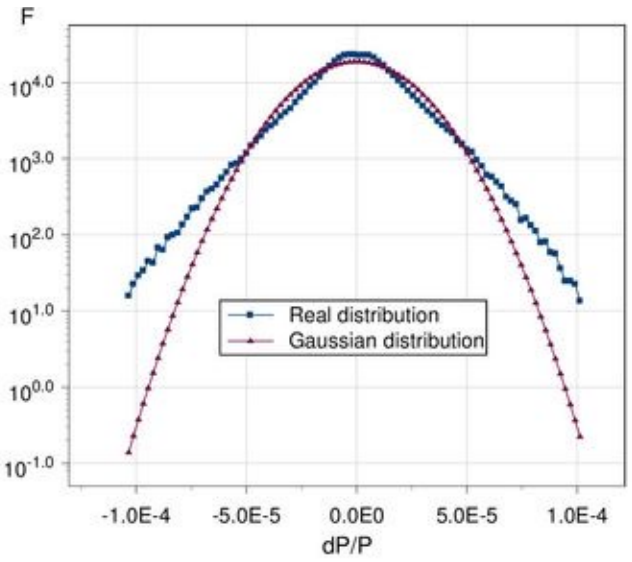
Necessaria la risoluzione di un sistema di equazioni differenziali

→ durante il corso vedremo sistemi numerici per la risoluzione di ODE (Ordinary Differential Equations)

Generatore (energia/momento)

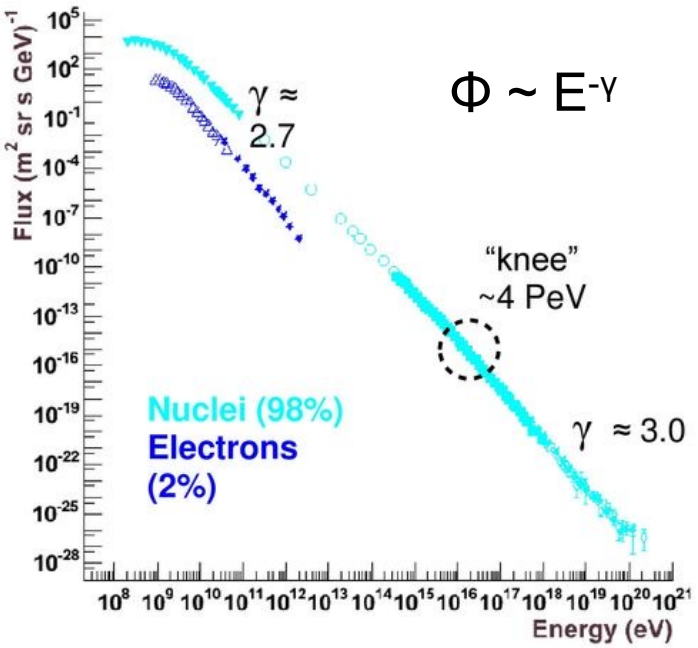
Che energia hanno le particelle?

Monocromatico:




```
Double_t MOMMEAN = 100; //GeV/c
DOuble_t DeltaPoP = 0.00005;

double mom = tran->Gaus(MOMMEAN, MOMMEAN*DeltaPoP);
double w=1;
```

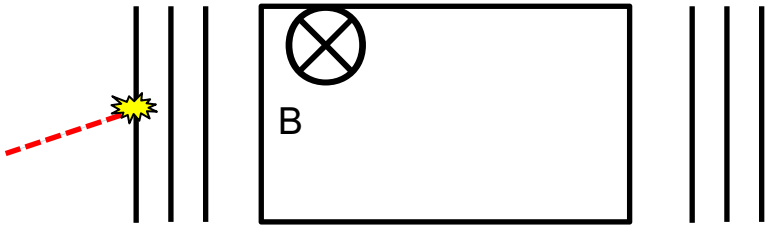
Spettro:

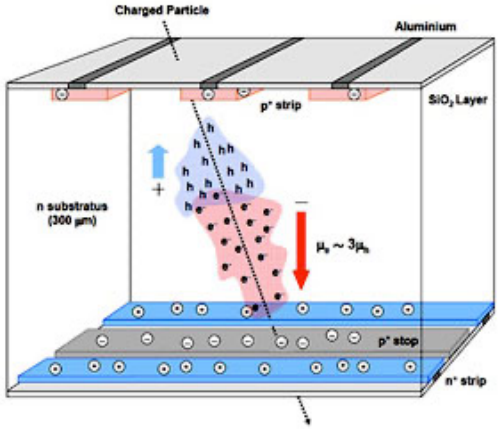



```
Double_t MOM1 =  0.1;//GeV/c
Double_t MOM2 = 1000;//GeV/c
Double_t INDEX = -3;

// --- gen momentum ---
double logmom = tran->Uniform(TMath::Log10(MOM1), TMath::Log10(MOM2));
double mom=pow(10.0, logmom);//GeV/c
double w = pow(mom, INDEX)*mom;//p^{index}/p^{-1}
```

Interazioni

Come interagiscono le particelle con il rivelatore?





Nel caso di un rivelatore al silicio (abbastanza sottile da non disintegrare la particelle) la perdita di energia per unità di materiale attraversato è data dalla formula di Bethe-Bloch

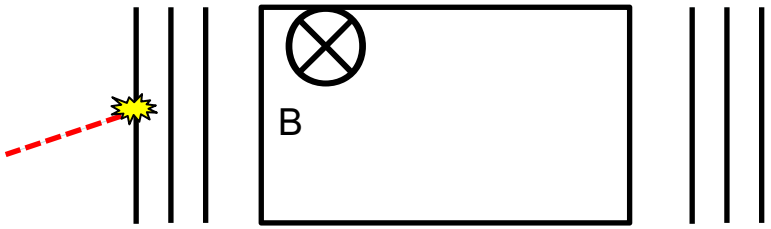
The mean rate of energy loss by moderately relativistic charged heavy particles is well-described by the "Bethe equation,"

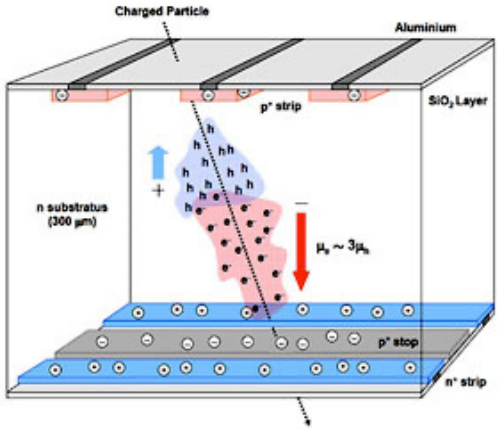
$$\left\langle -\frac{dE}{dx} \right\rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2 m_e c^2 \beta^2 \gamma^2 W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] . \quad (33.5)$$

It describes the mean rate of energy loss in the region $0.1 \lesssim \beta\gamma \lesssim 1000$

Interazioni

Come interagiscono le particelle con il rivelatore?

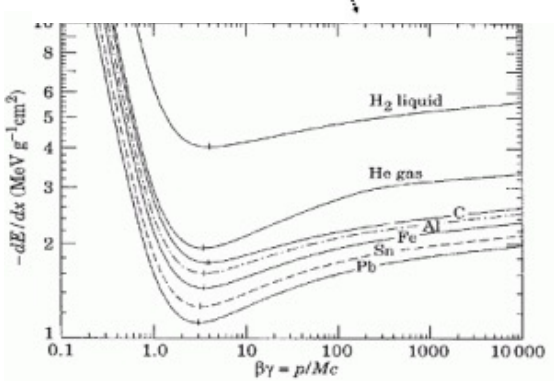




Nel caso di un rivelatore al silicio (abbastanza sottile da non disintegrare la particelle) la perdita di energia per unità di materiale attraversato è data dalla formula di Bethe-Bloch

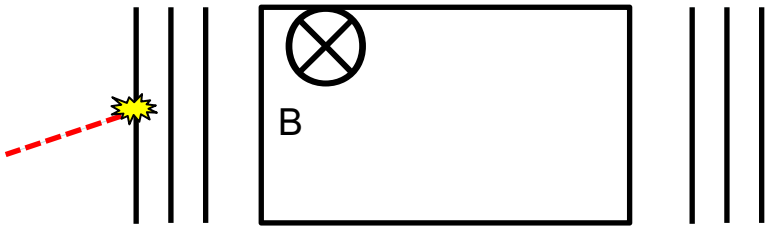
In generale questo valore dipende da:

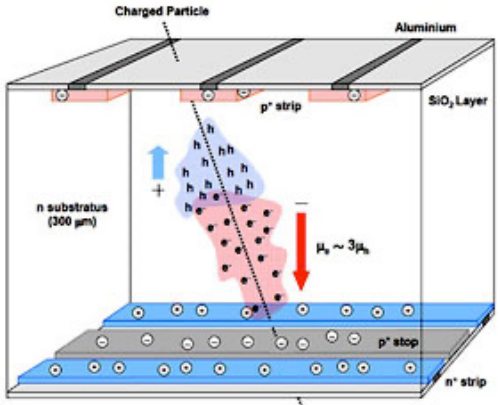
- tipo particella incidente
- energia particella incidente
- materiale atttraversato



Interazioni

Come interagiscono le particelle con il rivelatore?

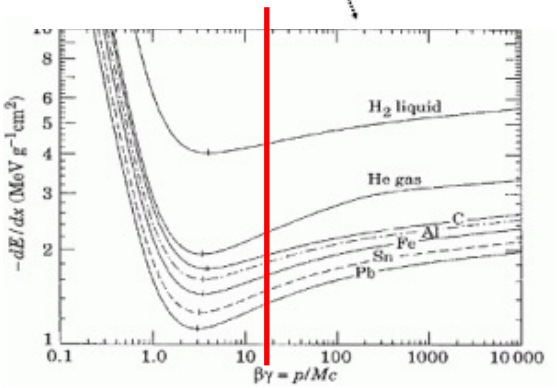




Nel caso di un rivelatore al silicio (abbastanza sottile da non disintegrare la particelle) la perdita di energia per unità di materiale attraversato è data dalla formula di Bethe-Bloch

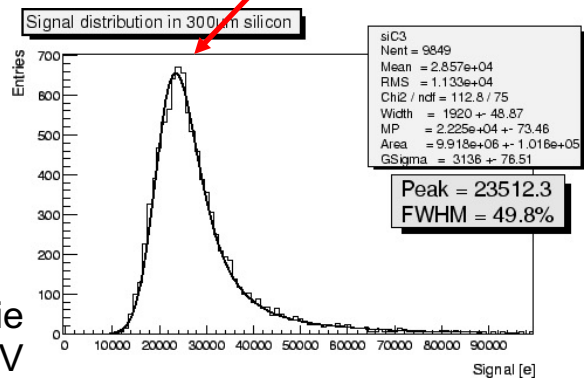
In generale questo valore dipende da:

- tipo particella incidente
- energia particella incidente
- materiale atttraversato



Se il rivelatore è molto sottile (< mm) il valore dato dalla Bethe-Bloch ci fornisce solamente il valore medio (*) di una variabile casuale distribuita secondo una Landau

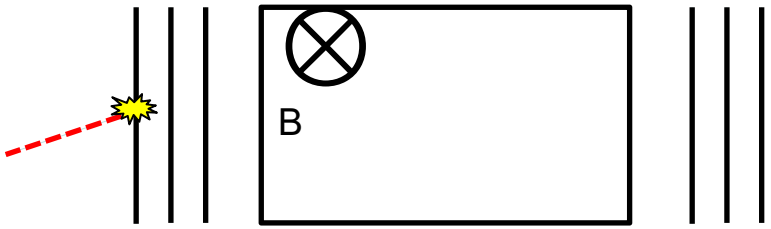
```
Double_t TRandom::Landau(Double_t mean = 0,
                         Double_t sigma = 1)
```



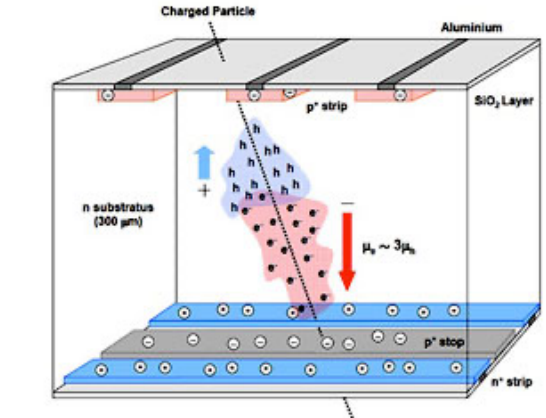
* la media (in realtà valore più probabile) è ~ 80 coppie elettrone-lacuna/μm e l'energia per creare una coppia è 3.6 eV

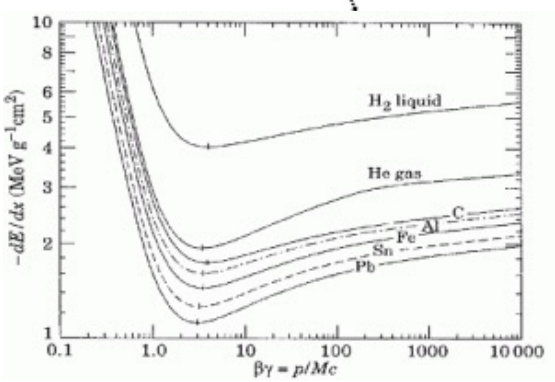
Interazioni

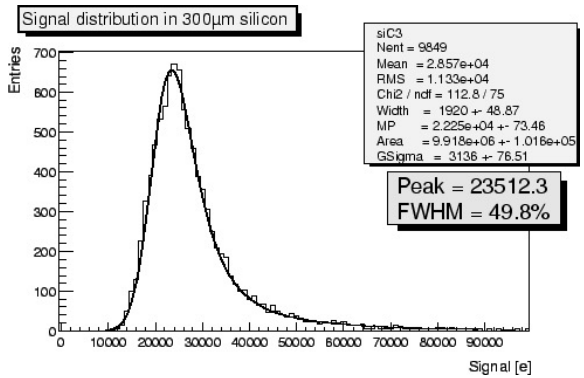
Come interagiscono le particelle con il rivelatore?




```
TF1* f = new TF1("edep_single", "[0]*TMath::Landau(x, [1], [2])",
    Edep_mean-5.0*Edep_sigma, Edep_mean+15.0*Edep_sigma);
f->SetNpx(10000);

f->SetParameter(0, 1.0);
f->SetParameter(1, Edep_mean);
f->SetParameter(2, Edep_sigma);
//  f->Draw();
//  return;

int nlayers=12;

int ntrials=1000000;
for (int nn=0; nn<ntrials; nn++) {

  double sum=0.0;
  double trunc_sum=0.0;
  double higher=0.0;

  for (int ii=0; ii<nlayers; ii++) {
    double Edep = f->GetRandom();
    sum+=Edep;
    trunc_sum+=Edep;
    if (Edep>higher) {
      higher=Edep;
    }
  }
```

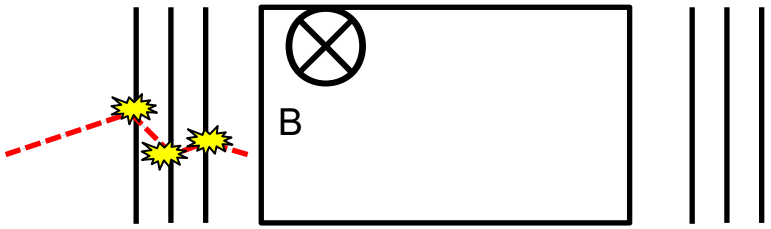




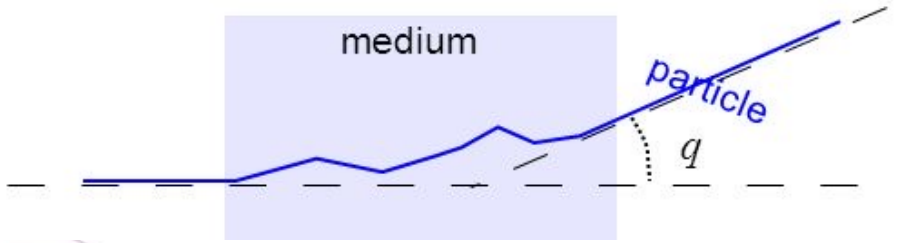


Interazioni

Come interagiscono le particelle con il rivelatore?



La traiettoria delle particelle è deflessa dalla collisioni Coulombiane con gli atomi del materiale attraversato



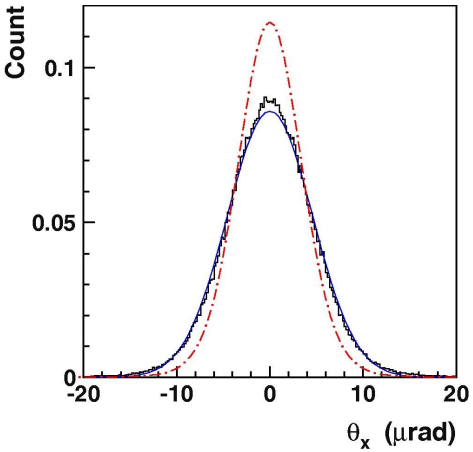
E la deviazione standard della distribuzione degli angoli di deflessione è:

$$\theta_0 = \frac{13.6\text{MeV}}{\beta cp} z \sqrt{x/X_0}\,(1 + 0.038 \ln(x/X_0))$$

della particella incidente

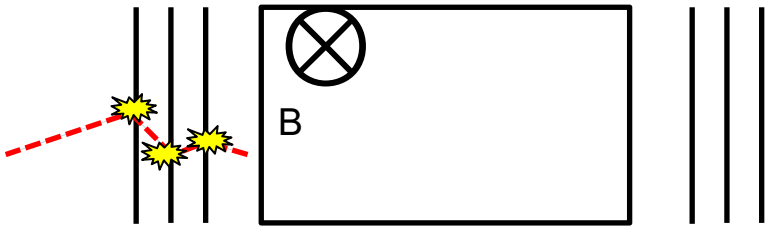
frazione (in unità di lunghezza di radiazione) di materiale attraversato

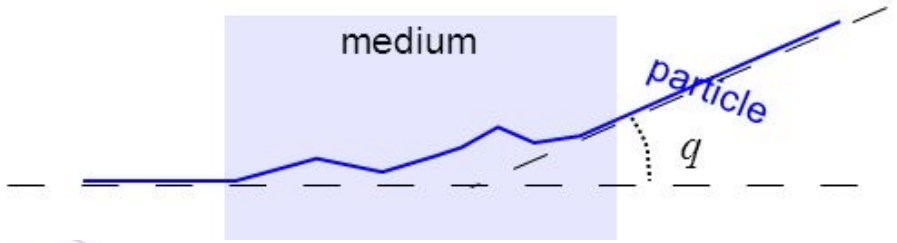
Ad esempio in 2mm di silicio la distribuzione degli angoli:



Interazioni

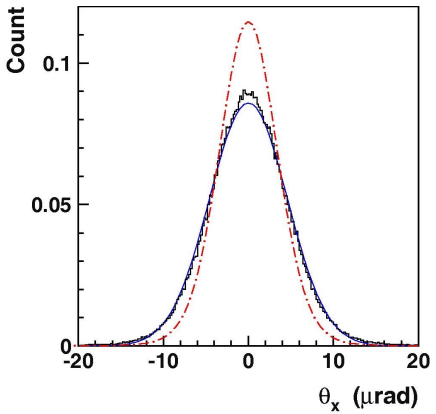
Come interagiscono le particelle con il rivelatore?






```
TF1* fspace = new TF1("fspace",
    "TMath::Exp(-(x*x)/(2.0*[0]*[0]))*TMath::Abs(TMath::Sin(x))",
    -100.0*theta0*1000.0, 100.0*theta0*1000.0);
fspace->SetNpx(10000);
fspace->SetParameter(0, theta0*1000.0);//mrad

int ntrials=1000000;
for (int nn=0; nn<ntrials; nn++) {
  double theta=gRandom->Gaus(0.0, theta0*1000.0);//mrad
  h->Fill(theta);
//    double theta_full=gRandom->Gaus(0.0, theta0_space*1000.0);//mrad //è
    sbagliato!!!
  double theta_full=fspace->GetRandom();
  h2->Fill(theta_full);
  double phi=gRandom->Uniform(0, 2.0*TMath::Pi());
  h3->Fill(TMath::ATan(TMath::Tan(theta_full)*TMath::Sin(phi)));
  h4->Fill(TMath::ATan(TMath::Tan(theta_full)*TMath::Cos(phi)));
}
```

$$\theta_0 = \frac{13.6\text{MeV}}{\beta cp} z \sqrt{x/X_0} \left(1 + 0.038 \ln(x/X_0)\right)$$



E ogni altro effetto abbia senso simulare…

- le particelle emettono radiazione (sincrotrone, etc…)?
- le particelle decadono durante il moto?
- …