

AVANCES DE TESIS

SEMANA 14/JUN/2024

Simulación de Muones

Se sigue trabajando en obtener la **distribución de Landau** para el grosor específico de la CCD (725 micras) y para muones.

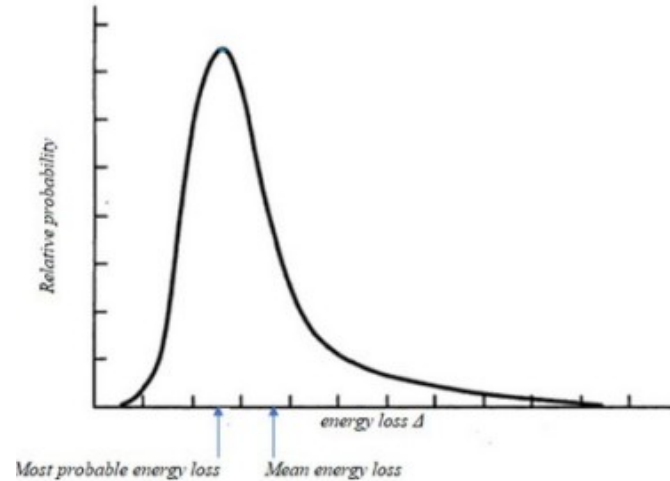
$$f(x, \Delta) = \phi(\lambda)/\xi, \quad \text{where}$$

$$\phi(\lambda) = \frac{1}{\pi} \int_0^{\infty} \exp(-u \ln u - u \lambda) \sin \pi u \, du$$

$$\lambda = \frac{1}{\xi} [\Delta - \xi (\ln \xi - \ln \varepsilon + 1 - C)]$$

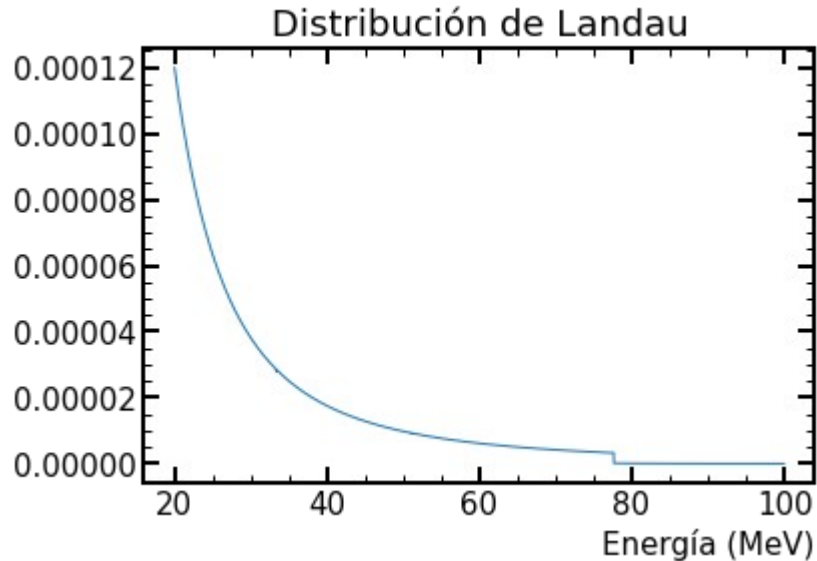
$C = \text{Euler's Const} = 0.577 \dots$ and

$$\xi = 2 \pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A} \left(\frac{z}{\beta} \right)^2 x.$$



$$\ln \varepsilon = \ln \frac{(1 - \beta^2) I^2}{2 m c^2 \beta^2} + \beta^2.$$

- En la gráfica de abajo se muestra lo que se supone es la distribución de Landau obtenida con el código implementado en Python.



Claramente debe haber algún error ya que no se forma la estructura que se espera. Además de haber un error en la integración.

El doctor Alexis proporcionó un código de ROOT hecho por su alumno Bryan Olmos que ya forma la distribución deseada. Los parámetros se ajustaron a las características de la CCD.

```
double LV (double *lx, double *lpar) {  
    double Delta = lx[0]; // Energy loss in absorber  
    double L = lpar[0]; // Thickness of absorber (Distance crossed by the particle)  
  
    double p = lpar[1]; // Momentum (in MeV/c)  
    double K = 0.307075; // K coefficient = 4*pi*N*r^2*m*c^2 (in MeV mol^-1 cm^2)  
    int z = -1; // Charge number of incident particle  
    double ZA = 0.498487; // Atomic number over Atomic mass of absorber (for Si)  
    double c = TMath::C(); // Speed of light  
    double me = 0.510998928; // Electron mass in MeV/c^2  
    double M = 105.65839; // Muon mass in MeV/c^2  
    double I = 0.000000174; // Mean excitation energy (for Si)  
    double bg = p/M;  
    double beta = bg/sqrt(1+(pow(bg,2))); // Beta factor  
    double gamma = 1/sqrt(1-(pow(beta,2))); // Gamma factor  
    double pi = TMath::Pi();  
    double rho = 2.33; // Density of material (for Si)  
  
    double d; // Variable for the Density effect
```

Constantes que se utilizan y datos del Si

```
double a = 0.1492; // Parameters (taken from W.R. Leo for SI)  
double k = 3.25; //  
double X0 = 0.2014; //  
double X1 = 2.87; //  
double C = -4.44; //  
// double d0 = 0.0; //  
double X = log10(bg);  
    if (X>=X1) {  
        d = 2*log(10.0)*X-C;  
    }  
    else if (X0<=X && X<X1) {  
        d = 2*log(10.0)*X-C+a*(pow((X1-X),k));  
    }  
    else if (X<X0) {  
        // d = d0*(pow(10,(2*(X-X0))));  
        d = 0;  
    }
```

Más parámetros del Si

Cada una de las variables se muestran a continuación (el programa tambien cuenta con la expresión de la energía promedio de Bethe-Bloch)

```
double xi = (K/2)*rho*ZA*L*pow((z/beta),2); // Xi variable
```

$$\xi = 2 \pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A} \left(\frac{z}{\beta} \right)^2 x.$$

```
double loge = log((1-(pow(beta,2)))*(pow(I,2))/(2*me*(pow(beta,2)))+(pow(beta,2))); // log epsilon variable
```

$$\ln \varepsilon = \ln \frac{(1 - \beta^2) I^2}{2 m c^2 \beta^2} + \beta^2.$$

```
double EC = 0.577; // Euler's constant
```


$C = \text{Euler's Const} = 0.577 \dots$ and

```
double lambda = (Delta-xi*(log(xi)-loge+1-EC))/xi; // Lambda parameter
```

$$\lambda = \frac{1}{\xi} [\Delta - \xi (\ln \xi - \ln \varepsilon + 1 - C)]$$

Es esta parte la distribución de Landau ya está directamente implementada por ROOT con la función Landau(), tambien se puede implementar la distribución Vavilov()).

```
if (kappa<=0.01) {  
    double phi = TMath::Landau(lambda, lambdamp, 1.0);  
    return phi/xi;  
}  
else if (0.01<kappa && kappa<10) {  
    double vav = TMath::Vavilov(Delta-Deltamp, kappa, beta2);  
    return vav;  
}  
else {  
    double gauss = exp(((Delta-DeltaAv)**2)/(2*sigma2));  
    double gauss = TMath::Gaus(Delta, DeltaAv, sqrt(sigma2));  
    return gauss;  
}
```


$$f(x, \Delta) = \phi(\lambda)/\xi,$$

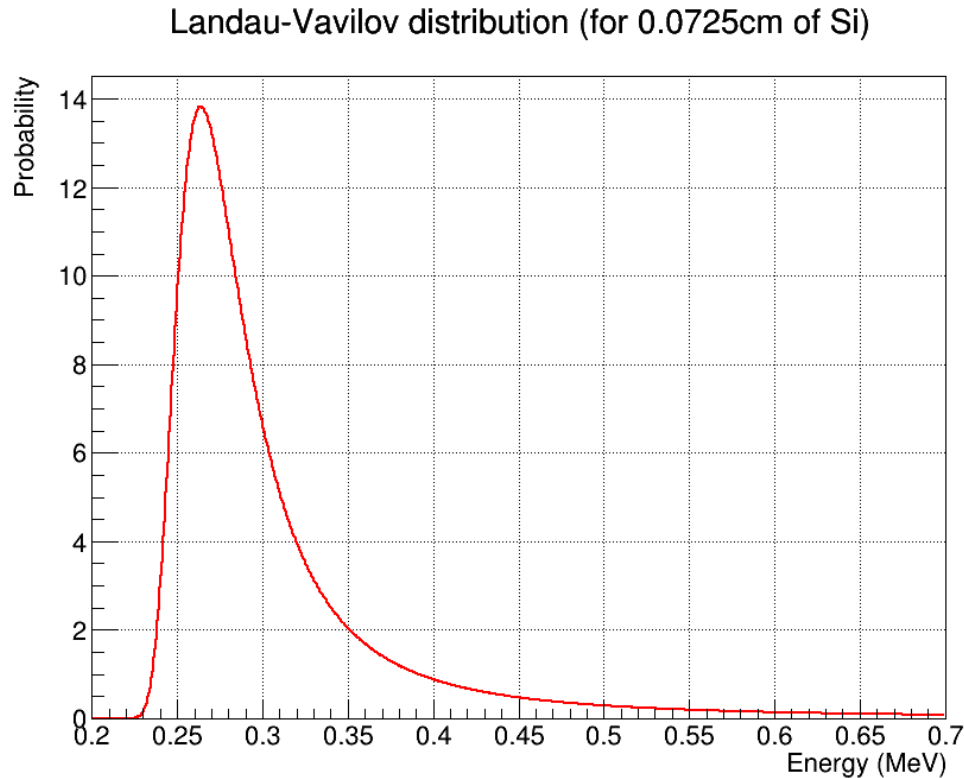


Otras distribuciones

Con esta función se manda a llamar la anterior y se llena un histograma. Aquí se coloca el grosor de la CCD (0.0725 cm).

```
void LandauVavilov_Mau() {  
    TCanvas *cnv = new TCanvas("cnv", "", 900, 700);  
    cnv->SetGrid();  
  
    TLatex lat;  
  
    TF1 *f = new TF1("f", LV, 0, 100, 2);  
    f->SetNpx(1000);  
  
    double s = 0.0725; // Distance parameter (in cm)  
    double p = 1000;   // Momentum parameter (in MeV)  
  
    f->SetParameter(0, s);  
    f->SetParameter(1, p);  
    f->SetRange(0.2, 0.7);  
    //f->SetTitle("Landau-Vavilov distribution (for 0.0725cm of Si);#font[12]{Energy} (MeV);Probability");  
    f->SetTitle("Landau-Vavilov distribution (for 0.0725cm of Si);Energy (MeV);Probability");  
    f->Draw();  
    // std::cout <<"Hello, World! \n"<< std::endl;  
  
    // std::cout << "Most Probably Energy" << std::endl;  
    // std::cout << Deltamp * 1000 << std::endl;  
}
```


Esta es la distribución de Landau que se obtiene. El valor del pico (valor mas probable) se encuentra en 266.603 KeV.



Valor del pico con ROOT

Most Probable Energy in KeV
266.603

Valor del pico con Python

266.0812893754627 KeV

Tal vez esa variación entre los valores se deba a las constantes que se usan en los dos programas, (el que ocupa ROOT es mas preciso).

Ya que se corrobore los detalles que faltan bastará con trabajar en **implementar en la simulación con PyROOT** el código que se mostró antes para obtener la distribución de Landau y así poder elegir un valor aleatorio siguiendo dicha distribución.

Espectros Experimentales