

AVANCES DE TESIS

SEMANA 28/JUN/2024

Espectros Experimentales y de la Simulación

En el cluster del ICN se tienen alrededor de 2000 imágenes con las siguientes características: NSAMP 1, 700x700 px, y EXPOSURE 360-1200 s (con un mayor número de imágenes con EXPOSURE 1200 s). De estas imágenes se obtiene el espectro de energía utilizando un algoritmo que filtra y cuenta la carga de un evento. En concreto este algoritmo se centra en **filtrar las trazas de muones** (que suelen ser bastante rectas) y se espera así obtener una **distribución de Landau**.

```
## Datos de la CCD
CCD_depth = 725 #micras
px_to_cm = 0.0015
px_to_micras = 15
micra_to_cm = 1 / 10000

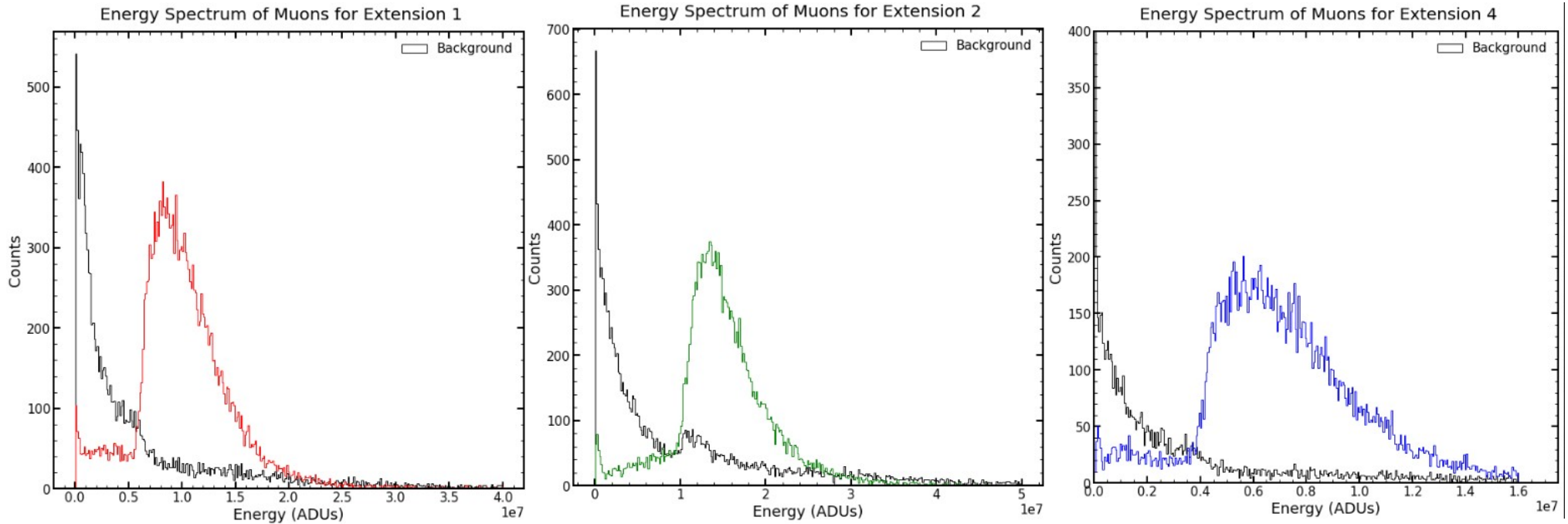
## Datos del filtro de muones
Solidit = 0.7
Elip = 3.5
DeltaEL_range_min, DeltaEL_range_max = 0.9, 3.55

ratio_keV = 0.0037
DeltaEL_range = 85

## Unidades, número de sigmas y número de bins (en las unidades 0 = ADUs, 1 = e-, 2 = KeV)
units = 1
n_sigmas = 5
numero_bins = 500
```

[Datos y parámetros que se utilizan]

ESPECTROS DE ENERGÍA (POR EXTENSIÓN).

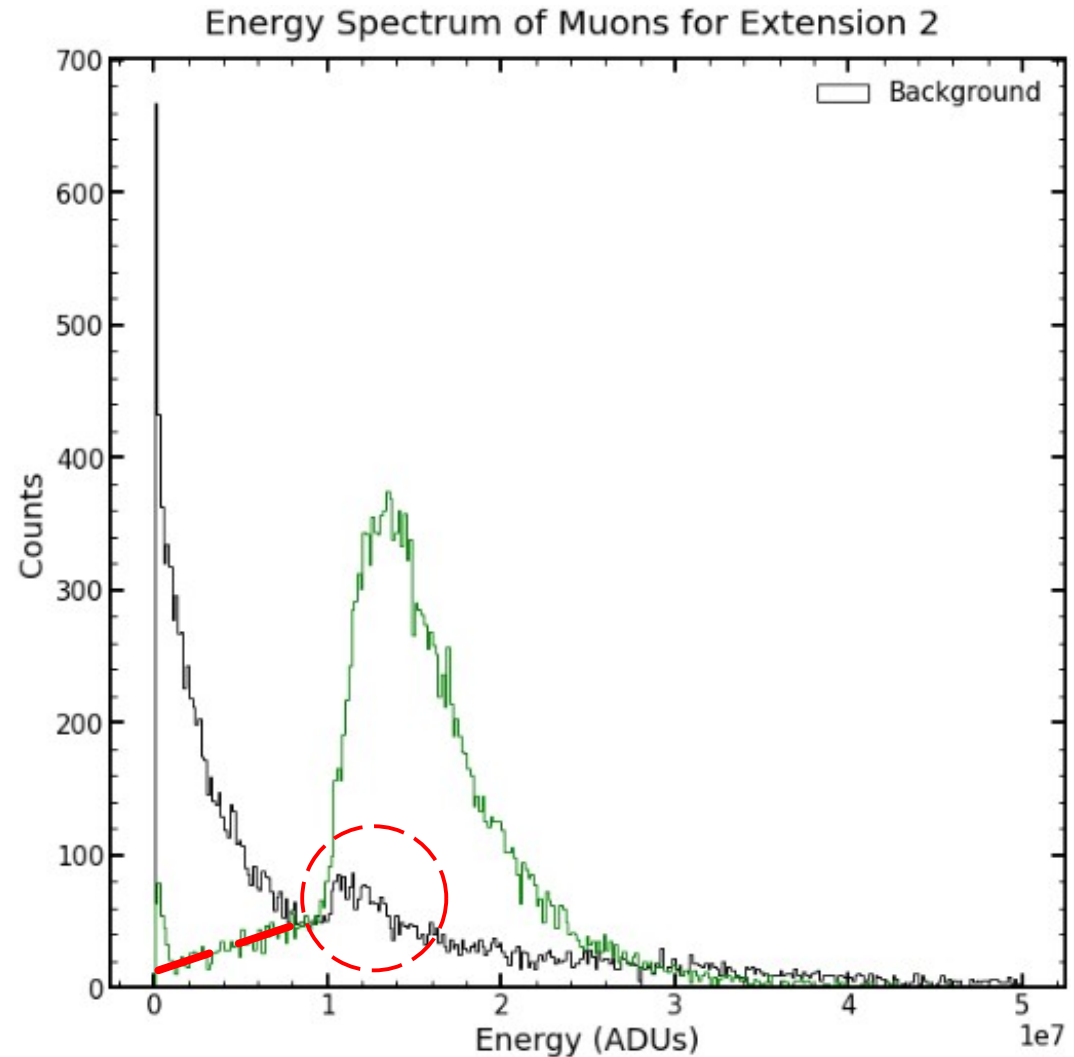


Valor de los
máximos de cada
espectro.

Pico de Extensión 1: 7748516.0 ADUs
Pico de Extensión 2: 12059729.0 ADUs
Pico de Extensión 4: 6139996.5 ADUs

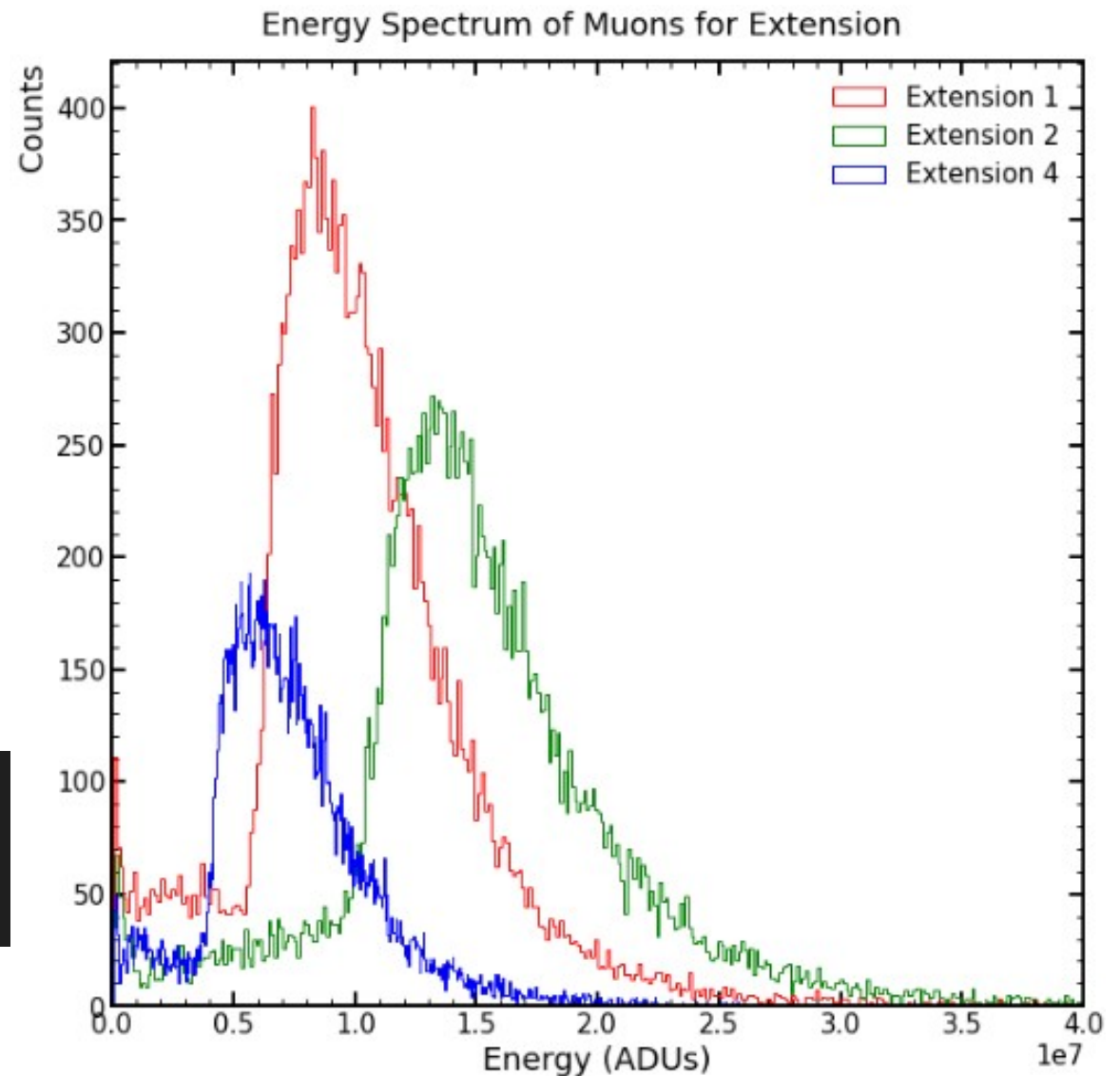
Parece que el espectro de las Extensiones 1 y 4 tienen una distribución y background correctos. Sin embargo, el background de la Extensión 2 es diferente y parece que aquí se están “escapando” muones del filtro además de una pendiente creciente al principio del espectro.

Se está trabajando en un script para modificar los parámetros del filtro de muones **por extensión**.



Aquí se muestran los espectros de las tres extensiones. Se realizó la calibración considerando que el valor del valor mas probable en la distribución de Landau, para muones con momentos de 600 MeV, es de 152.545 KeV (valor obtenido con el script de C++).

```
La calibración para cada extensión es:  
50794.95230915468 ADUs/KeV  
79056.86190960045 ADUs/KeV  
40250.39496542004 ADUs/KeV
```

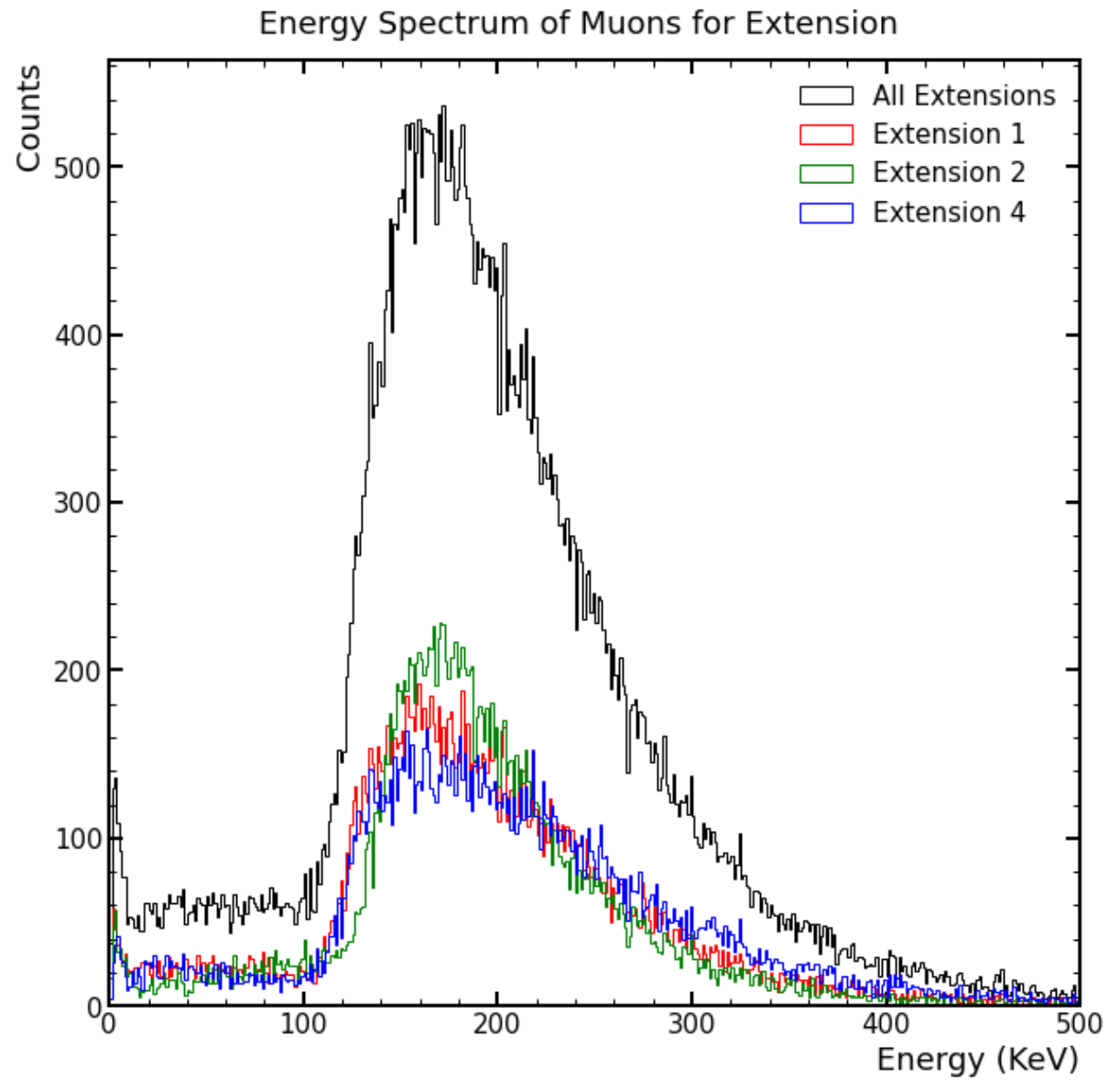


Si se considera que cada electron es equivalente a 3.7 eV (0.0037 KeV) entonces se puede obtener la calibración en ADUs por electrón para cada extensión, las cuales se muestran abajo.

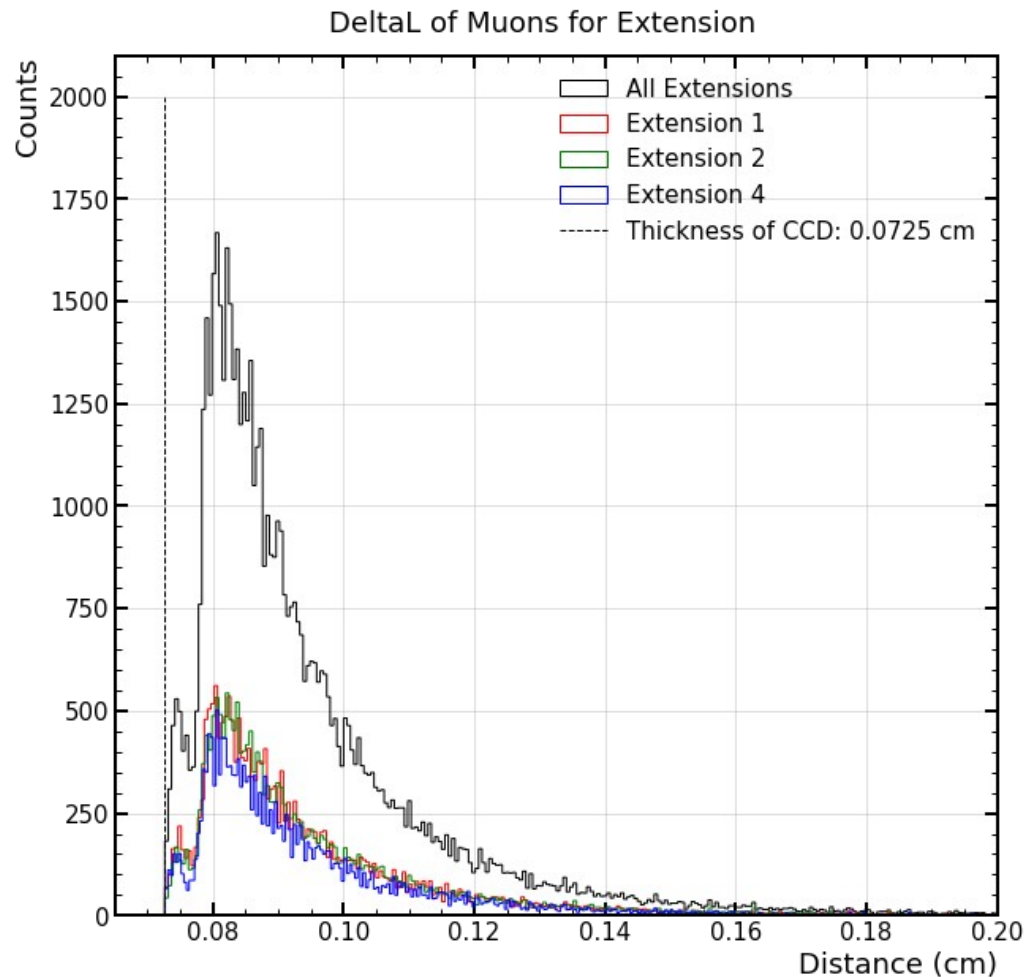
La calibración para cada extensión es:
187.94132354387233 ADUs/e-
292.5103890655217 ADUs/e-
148.92646137205415 ADUs/e-

La calibración para las imágenes de 324
227 ADUs/e-
220.4 ADUs/e-
197.7 ADUs/e-

Se muestran los espectros con la calibración ADUs \rightarrow KeV aplicada. Tal vez el número de bins se pueda aumentar (aquí se hizo con 300 bins) para obtener un valor mas fino del pico al sumar todas las extensiones.



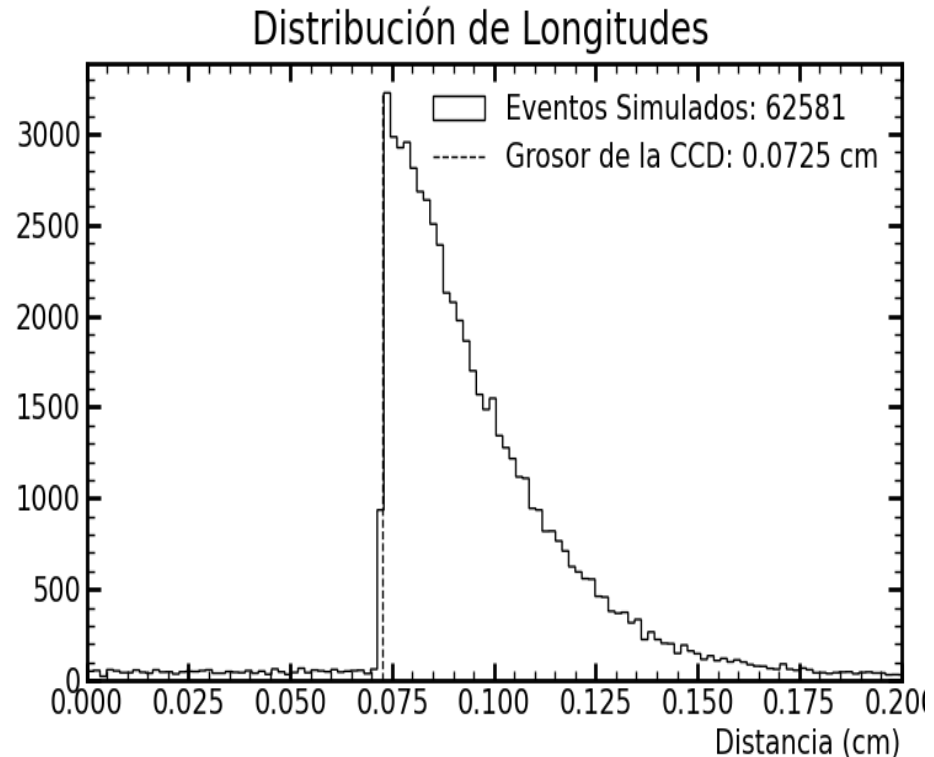
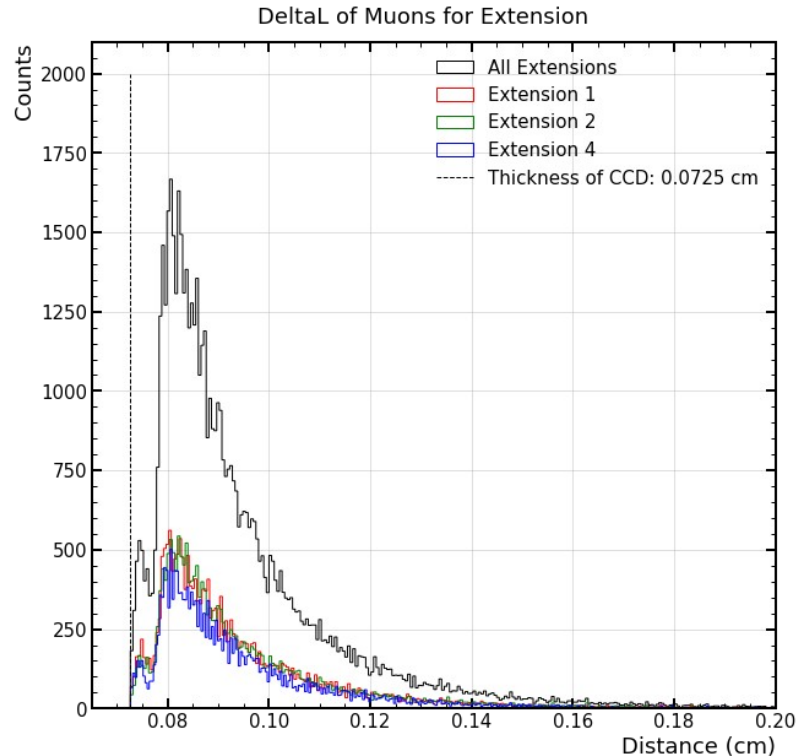
ESPECTROS DE LONGITUDES



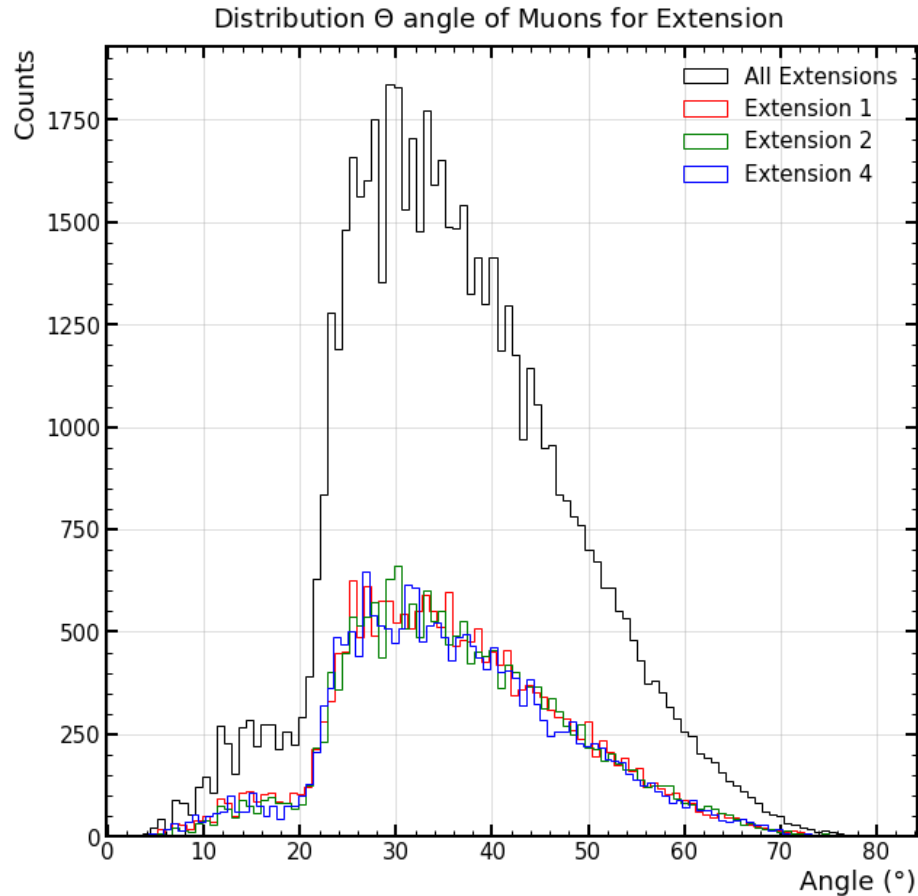
Aquí se muestra el espectro de longitudes de los muones que atraviezan la CCD.

Se observa la línea del grosor de la CCD (0.0725 cm) que debería de ser el máximo de la gráfica pero por la dificultad de identificar los muones verticales no se logra obtener.

Aquí se muestra la distribución de longitudes obtenida con la simulación de primeros principios (simulando, aproximadamente, el mismo número de muones que se obtuvieron experimentalmente los cuales fueron 61,120). Claramente al espectro experimental le “hacen falta datos” para completar el de la simulación. Para saber cuales son las longitudes que falta se tiene que analizar el espectro de ángulos θ (que está mas adelante).

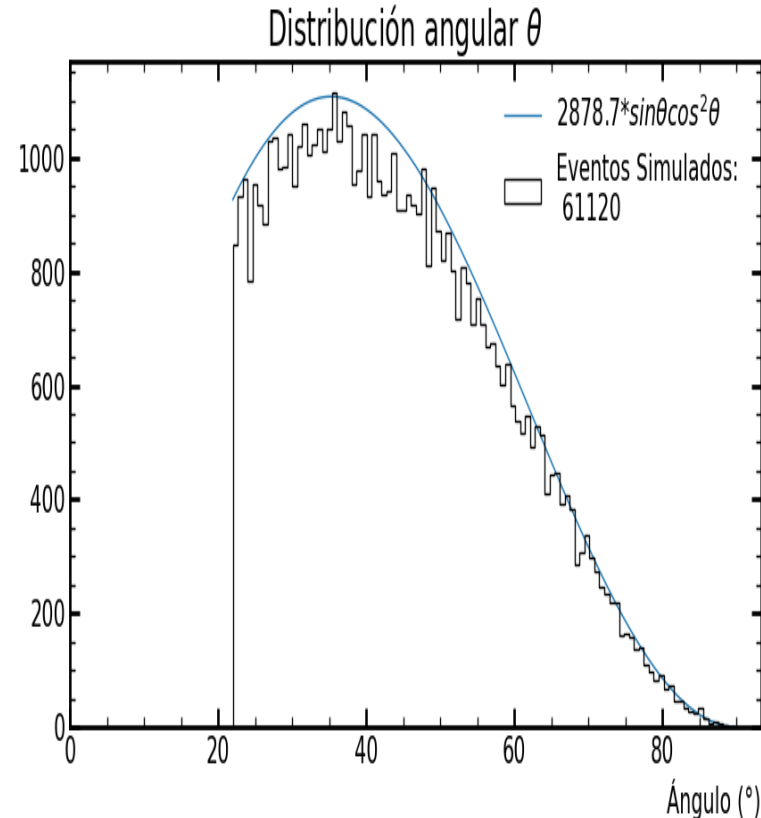
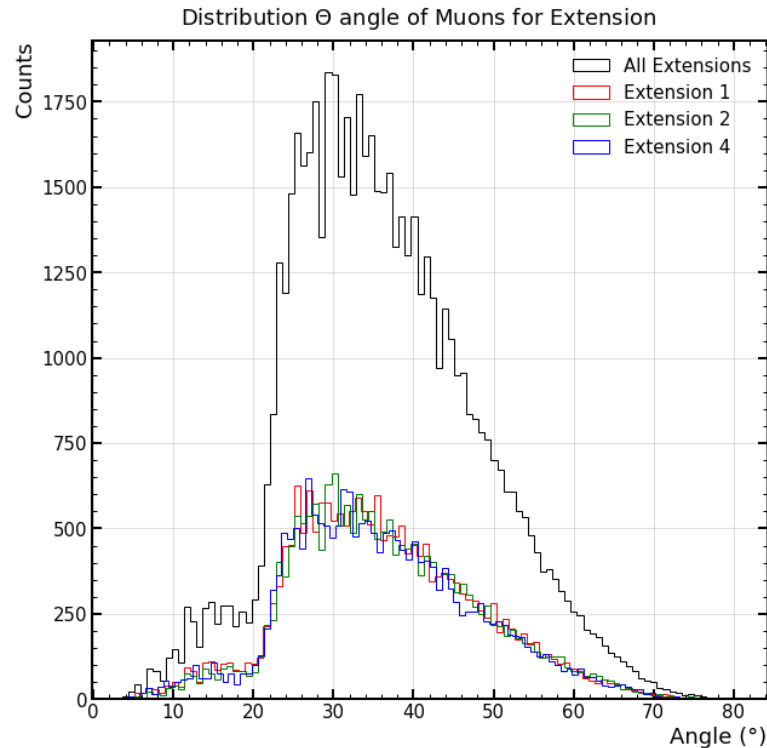


ESPECTROS DE ÁNGULOS THETA



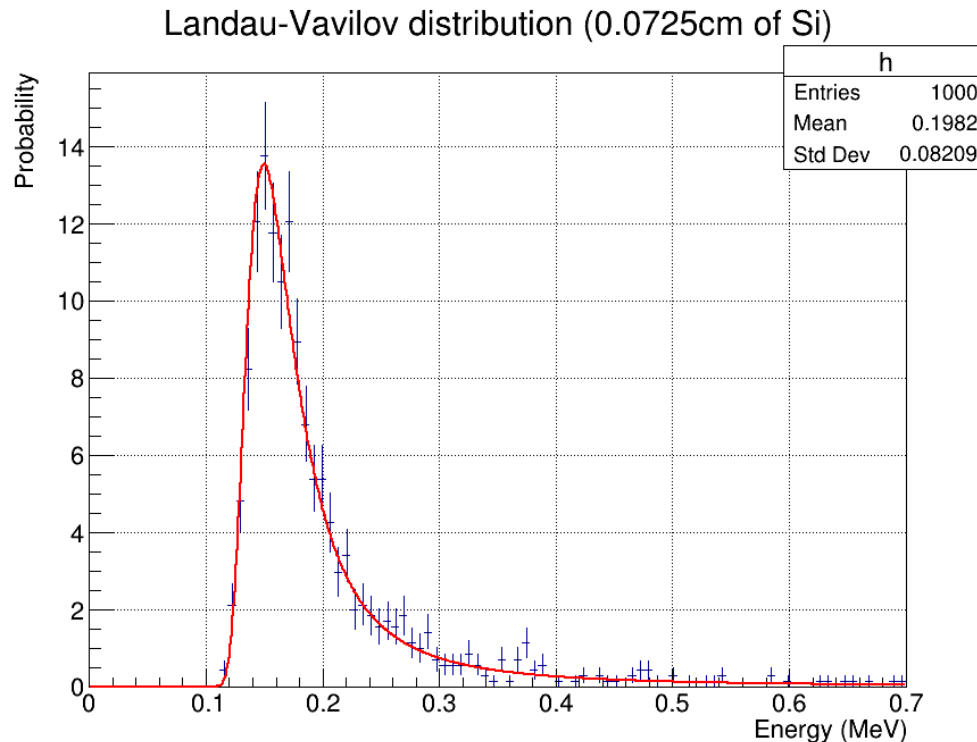
Aquí se muestra el espectro de ángulos θ de los muones que se detectan. Esta distribución debería de coincidir con la expresión $\sin\theta\cos^2\theta$, sin embargo filtro no logra identificar muones con ángulos θ pequeños (al parecer menores a $\approx 22^\circ$).

Se muestra el espectro que se obtiene de la simulación si se le sustraen los datos menores de 22° (se simuló el mismo número de muones que se detectaron experimentalmente). Tal vez a la simulación faltaría agregarle una función de ruido para esos ángulos pequeños.



Distribución de Landau en la simulación

Ya se desarrolló el script en ROOT que nos proporciona la distribución de Landau para muones con un momento específico, y considerando el grosor de la CCD que es de 725 μm . Además ya se pueden obtener valores aleatorios que siguen dicha distribución.



Valores
aleatorios de
la distribución

```
root [0]
Processing LandauVav
Edep = 695.09 KeV
root [1] .x LandauVa
Warning in <TCanvas:
Warning in <TROOT::A
Edep = 183.893 KeV
root [2] .x LandauVa
Warning in <TCanvas:
Warning in <TROOT::A
Edep = 154.972 KeV
root [3] .x LandauVa
Warning in <TCanvas:
Warning in <TROOT::A
Edep = 141.143 KeV
```

Y una sugerencia para utilizar esto en la simulación fué realizar el script en Python que ejecute este programa en ROOT para así obtener el dicho valor aleatorio así que en eso se trabajó pero hubo algunas complicaciones.

La primera de ellas, que ya se solucionó, es que los valores “aleatorios” que se escogen de la simulación son los mismos cada que se ejecuta el programa por primera vez.

```
bruce@bruce-Latitude-E6320:~/Documents/Programas/Simulacion_ab_initio$ root -l -b -n LandauVavilov_Mau.C
root [0]
Processing LandauVavilov_Mau.C...
Edep = 695.09 KeV
root [1] .q
bruce@bruce-Latitude-E6320:~/Documents/Programas/Simulacion_ab_initio$ root -l -b -n LandauVavilov_Mau.C
root [0]
Processing LandauVavilov_Mau.C...
Edep = 695.09 KeV
root [1] █
```

Es el mismo valor!!

Para solucionarlo solo se agregó una línea de código que cambia la semilla aleatoria del programa cada que se ejecuta.

```
void LandauVavilov_Mau() {  
    gRandom->SetSeed(0); // Cambia la semilla aleatoria para el GetRandom  
}
```


```
bruce@bruce-Latitude-E6320:~/Documents/Programas/Simulacion_ab_initio$ root -l -b -n LandauVavilov_Mau.C  
root [0]  
Processing LandauVavilov_Mau.C...  
Edep = 193.266 KeV  
root [1] .q  
bruce@bruce-Latitude-E6320:~/Documents/Programas/Simulacion_ab_initio$ root -l -b -n LandauVavilov_Mau.C  
root [0]  
Processing LandauVavilov_Mau.C...  
Edep = 162.136 KeV  
root [1]
```

Ya no son el mismo

Con esto se garantiza que aunque el valor del momento del muon sea el mismo el valor de la energía si será aleatorio.

Otra cuestión es que las variables que se guardan en el entorno **NO** se quedan guardadas de manera permanente sino que se eliminan cuando el script termina, por esta razón no es posible obtener, de esta manera, el valor de la variable “Edep”.

Sin embargo, una manera de obtenerlo es utilizando el texto que imprime el script en ROOT como se muestra a continuación.

```
new_env = subprocess.run(["root", "-l", "-b", "/home/bruce/Documents/Programas/Simulacion_ab_initio/LandauVavilov_Mau.C", "-q"],  
                          capture_output=True)   
  
# print(os.getenv('PATH'))  
# subprocess.run()  
Random_energy_Landau = float(new_env.stdout.decode('ascii').split('=')[-1].split(' ')[1])  
# print(float(new_env.stdout.decode('ascii').split('=')[-1].split(' ')[1]))
```

Esta opción permite obtener el texto que imprime el script de ROOT

Y lo que regresa es la siguiente cadena str en binario, que se puede decodificar y obtener el valor de Edep.

```
b'\nProcessing /home/bruce/Documents/Programas/Simulacion_ab_initio/LandauVavilov_Mau.C...\nEdep = 357.328 KeV\n'  
b'\nProcessing /home/bruce/Documents/Programas/Simulacion_ab_initio/LandauVavilov_Mau.C...\nEdep = 204.466 KeV\n'
```

Esta podría ser una manera de tener acceso al valor aleatorio de la distribución de Landau. Sin embargo se tendría que corroborar si esto es práctico o no. Otro problema con el que se está trabajando es en darle como argumento la energía de Smith-Duller. Tal vez aquí si se pueda aplicar la opción de guardar una variable (a partir de Python) para leerla en el script de ROOT.

```
Random_energy = rand.choices(Energy, list_dis_Energy) ## Escoje una energía segun la distribución de Smith-Duller en
list_random_energy.append(Random_energy[0])

os.environ["EN_SMITH"] = str(Random_energy[0])
```

Se añade la variable de energía al entorno desde Python

Se lee el valor de la energía en el script de ROOT y se transforma en tipo float, despues el programa continua su proceso con esta variable.

```
// double En_Smith;
// char En_Smith_char[100] = getenv("EN_SMITH");
float p = atof(getenv("EN_SMITH")); // Momentum parameter (in MeV)
```

Parece que esta opción funciona de manera correcta ya que si regresa el número aleatorio para cada una de sus distribuciones de Landau. Ahora solo se tiene que corregir el hecho de que el script de ROOT recibe como variable el **MOMENTO** del muon y no su energía, pero eso solo es realizar una operación antes.

```
Hora de inicio del cálculo: 2024-06-27 23:20:33.228239
Se simularán 2 muones.
Energía de SMith-Duller: 32.622220097116696
b'\nProcessing /home/bruce/Documents/Programas/Simulacion_ab_initio/LandauVavilov_Mau.C...\nEdep = 699.92 KeV\n'
b''
Energía de SMith-Duller: 16835.508029612007
b'\nProcessing /home/bruce/Documents/Programas/Simulacion_ab_initio/LandauVavilov_Mau.C...\nEdep = 695.232 KeV\n'
b''
Hora final de cálculo: 2024-06-27 23:20:36.743908
Tiempo de cálculo: 0:00:03.515669
Se guardó la información de los muones simulados en el archivo GenMuon.root
```