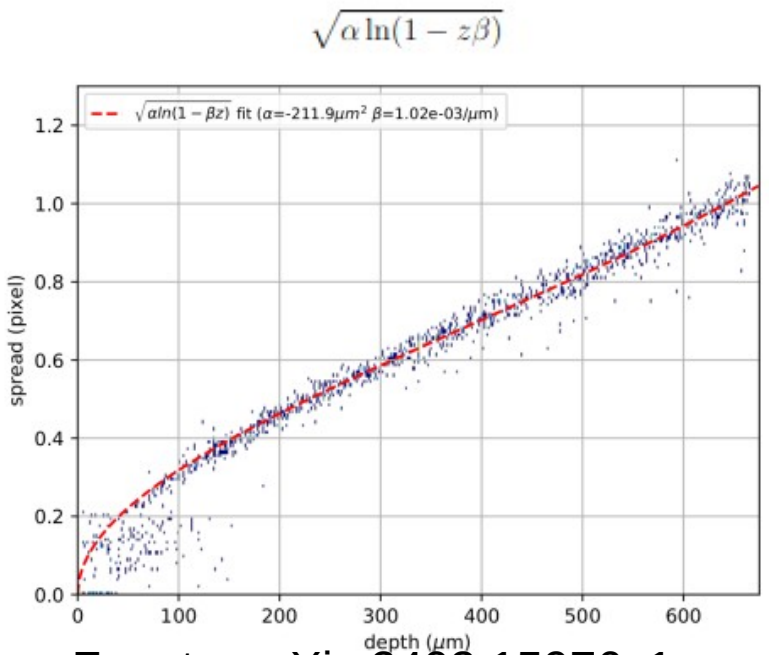
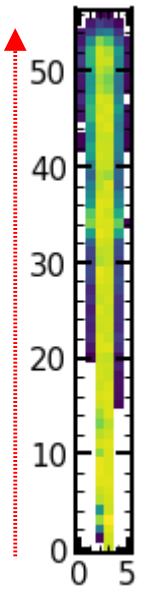
AVANCES DE TESIS
SEMANA 02/SEPT/2024

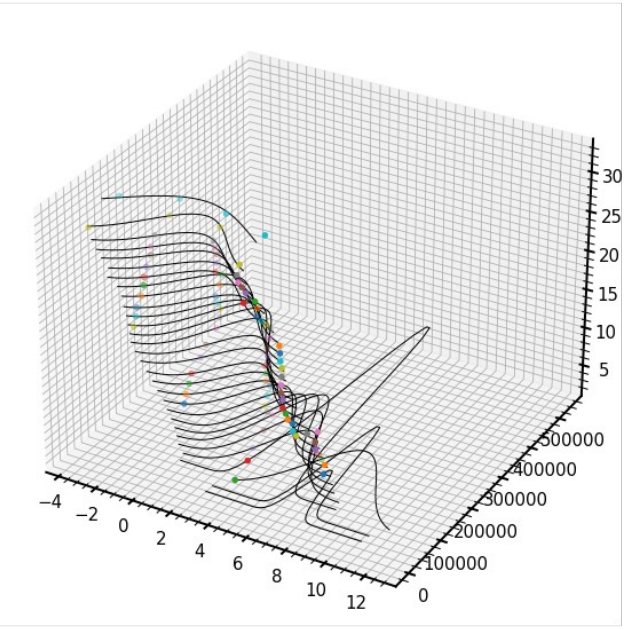
Modelo de Difusión

Se busca encontrar el valor de los parámetros a la ecuación mostrada abajo. Para ello se debe obtener el valor de la σ de una gaussiana que se ajusta a las “rebanadas” de trazas de muones.



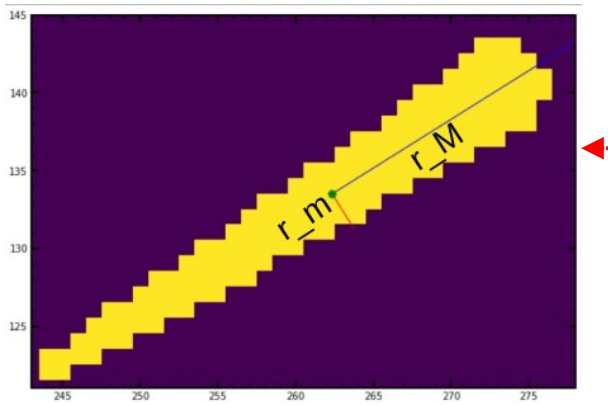
Fuente: arXiv:2403.15976v1





Se analiza en la dirección de la flecha

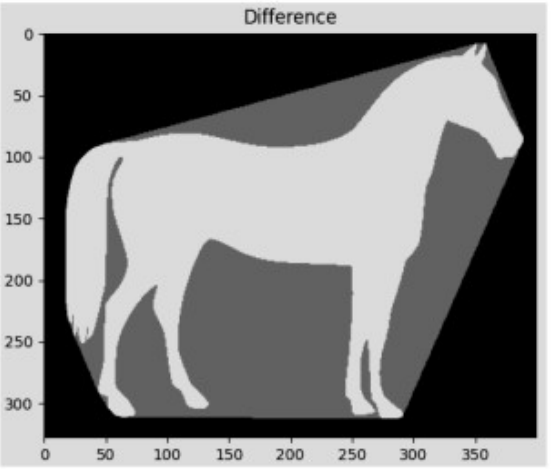
Para facilitad la obtención de los datos se buscan **muones rectos,** completamente verticales u horizontales. Se usaron las imágenes de 1 skip del cluster, y distintos valores en los parámetros del filtro



Elipticidad: relación entre radio menor y mayor de una elipce

Principales Parámetros

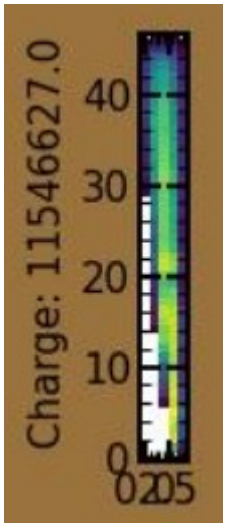
Carga del cluster (para este caso)

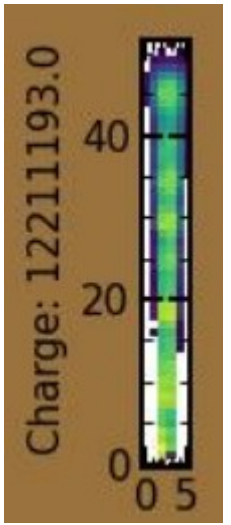


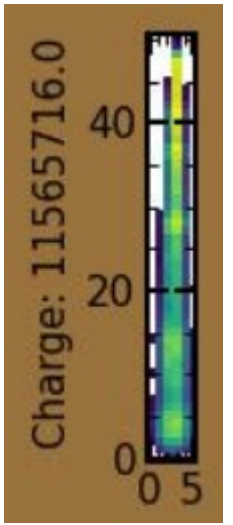
Solidity: relación entre píxeles vaciós al realizar una poligonalización

Parámetros con el mejor resultado

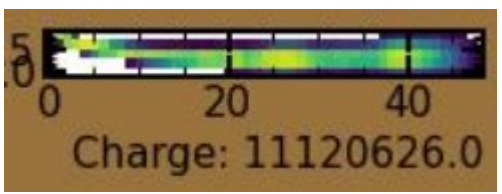
```
Muones rectos y verticales detectados en total:  429

    Solidit = 0.7
    Elipticity = 0.9
    min_Charge =  3 * 10**6 # ADUs
```

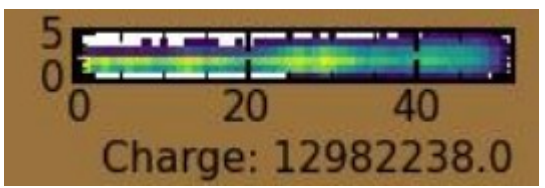


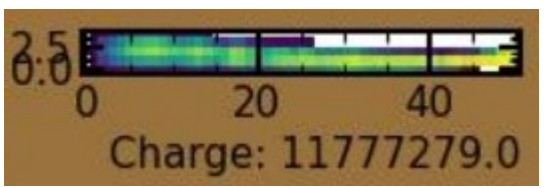




Muones Verticales

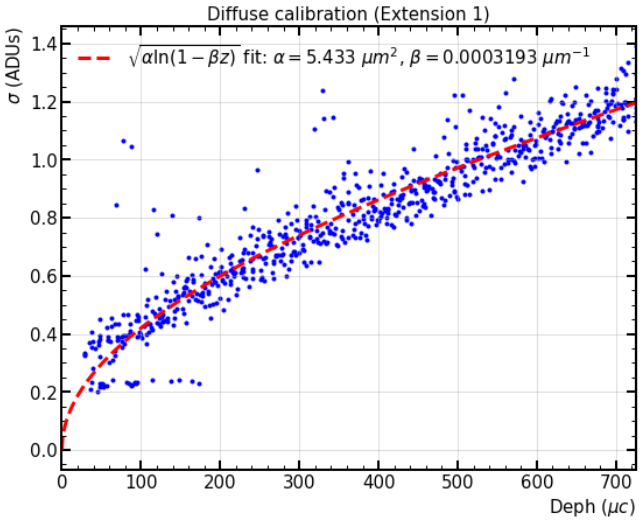


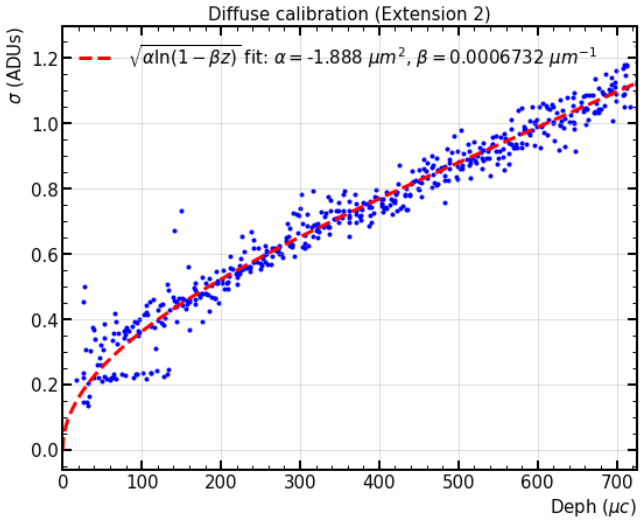


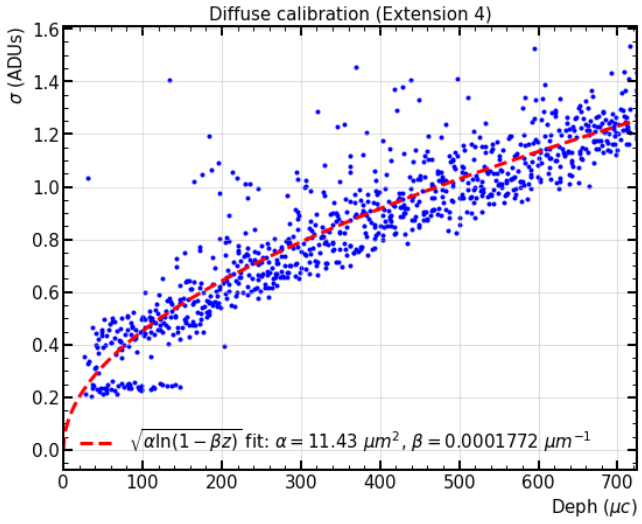


Muones Horizontales

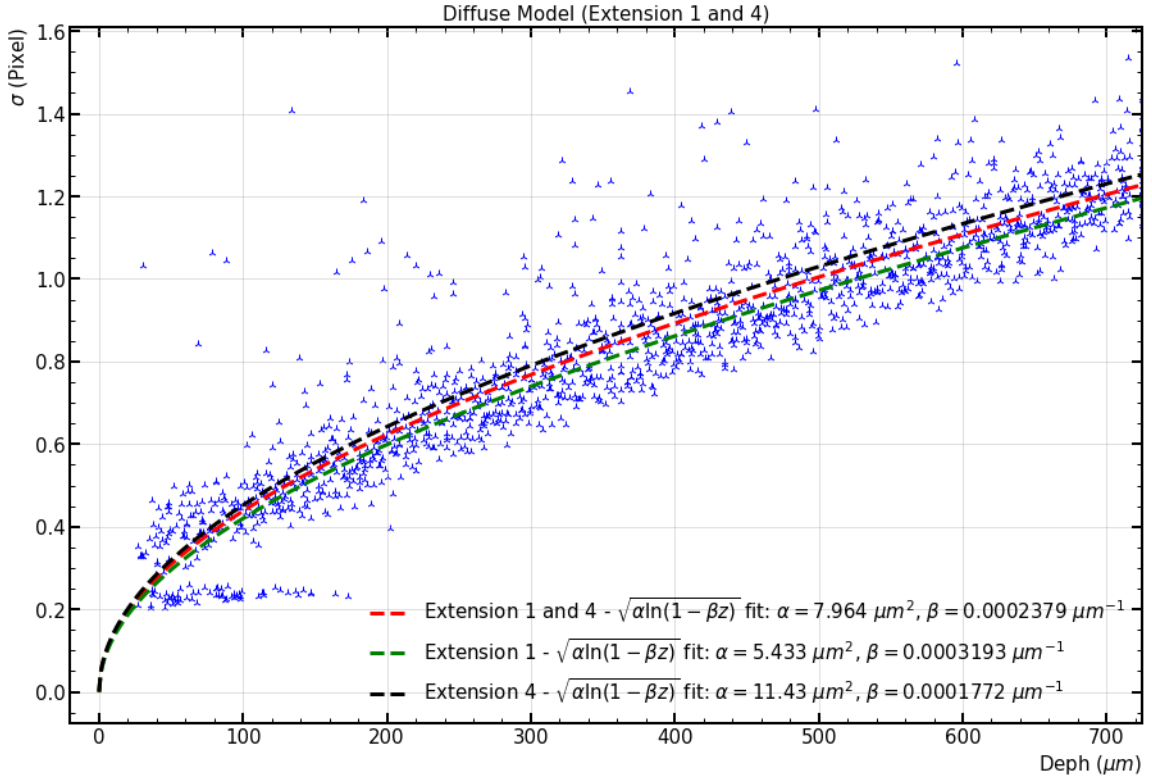
Se pudieron seleccionar 13 muones verticales para la extensión 1, 10 para la extensión 2, y 16 para la extensión para realizar los ajustes mostrados.



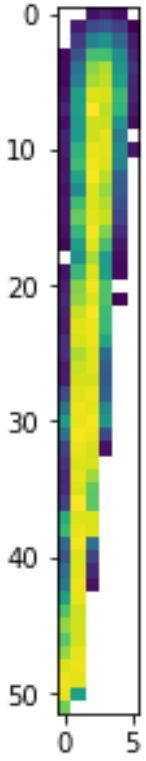


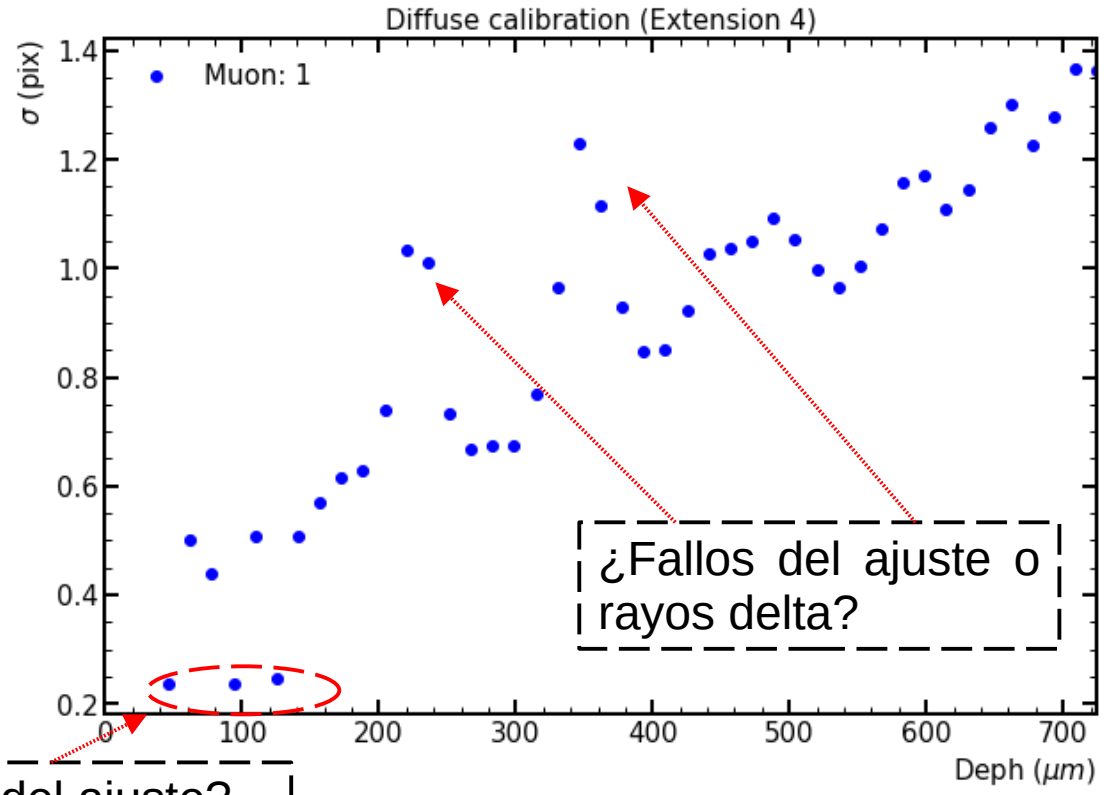


Se utilizaron las extensiones 1 y 4 para realizar un ajuste promedio, el cual se muestra abajo. Este ajuste es el que se utilizará, por ahora, para la simulación y para los espectros experimentales, aunque deberá ir mejorando.

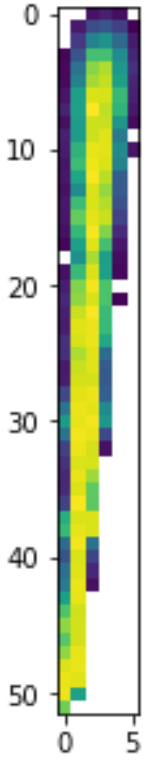


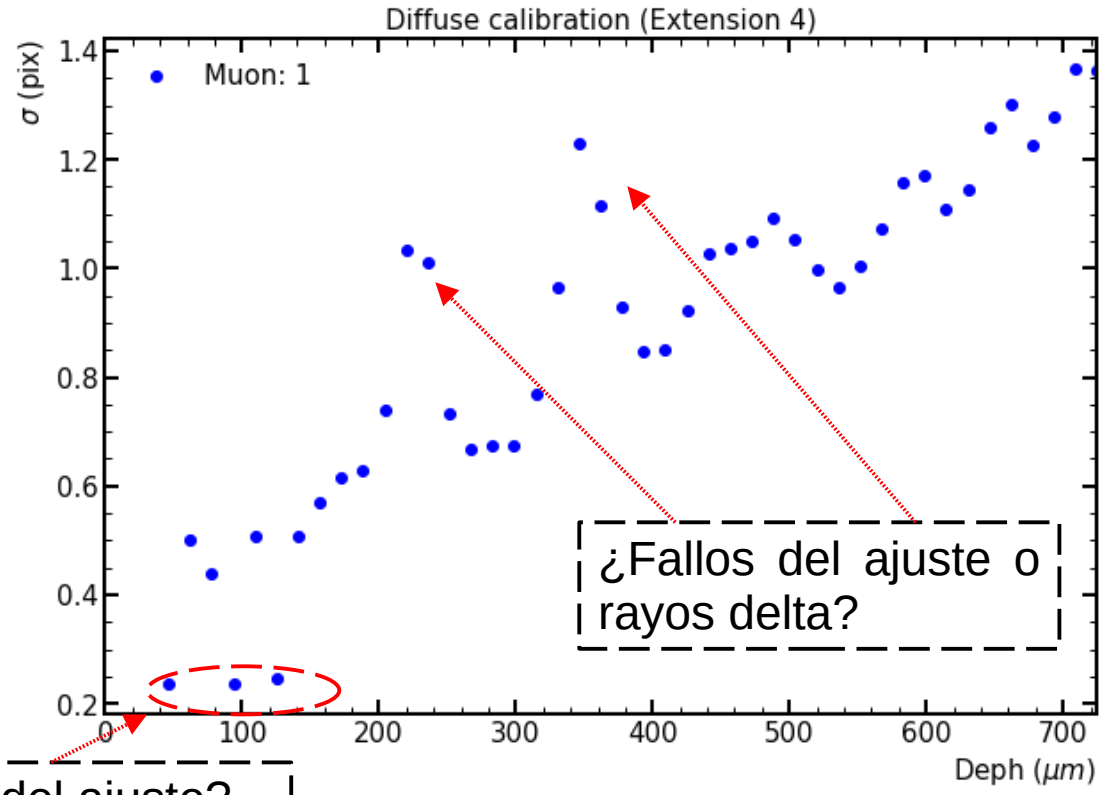
Para mejorarlo se debe verificar que los ajustes gaussianos sean correctos además de que el muon seleccionado no tenga rayos delta y en caso de tenerlos eliminarlos.



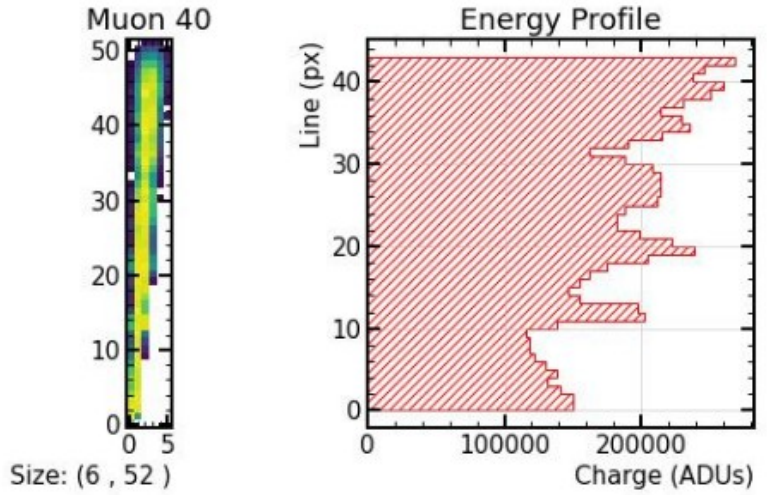


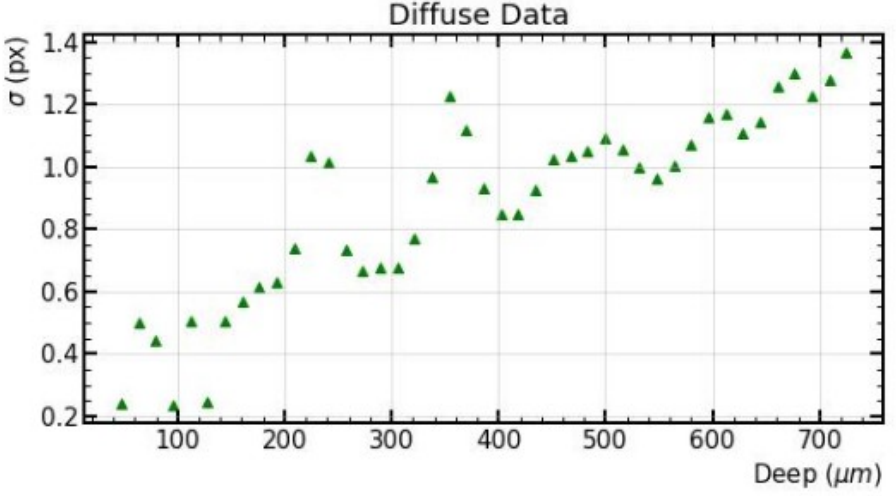
Para mejorarlo se debe verificar que los ajustes gaussianos sean correctos además de que el muon seleccionado no tenga rayos delta y en caso de tenerlos eliminarlos.

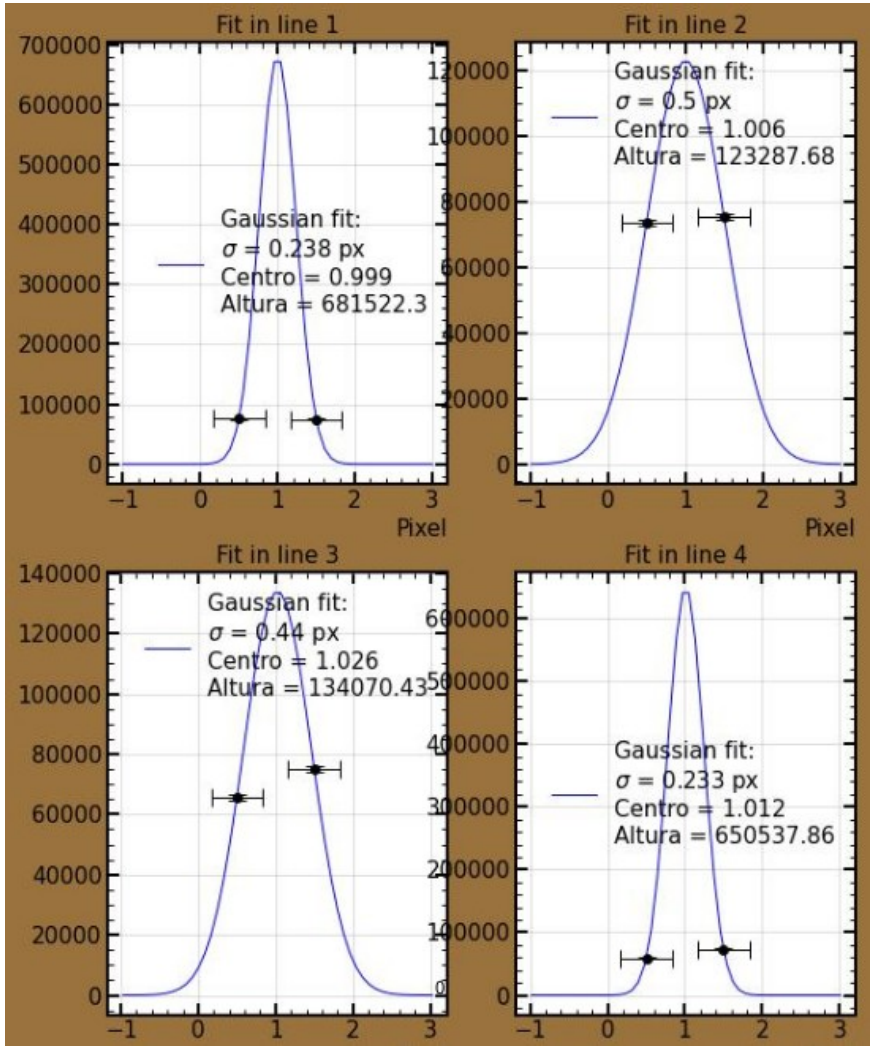


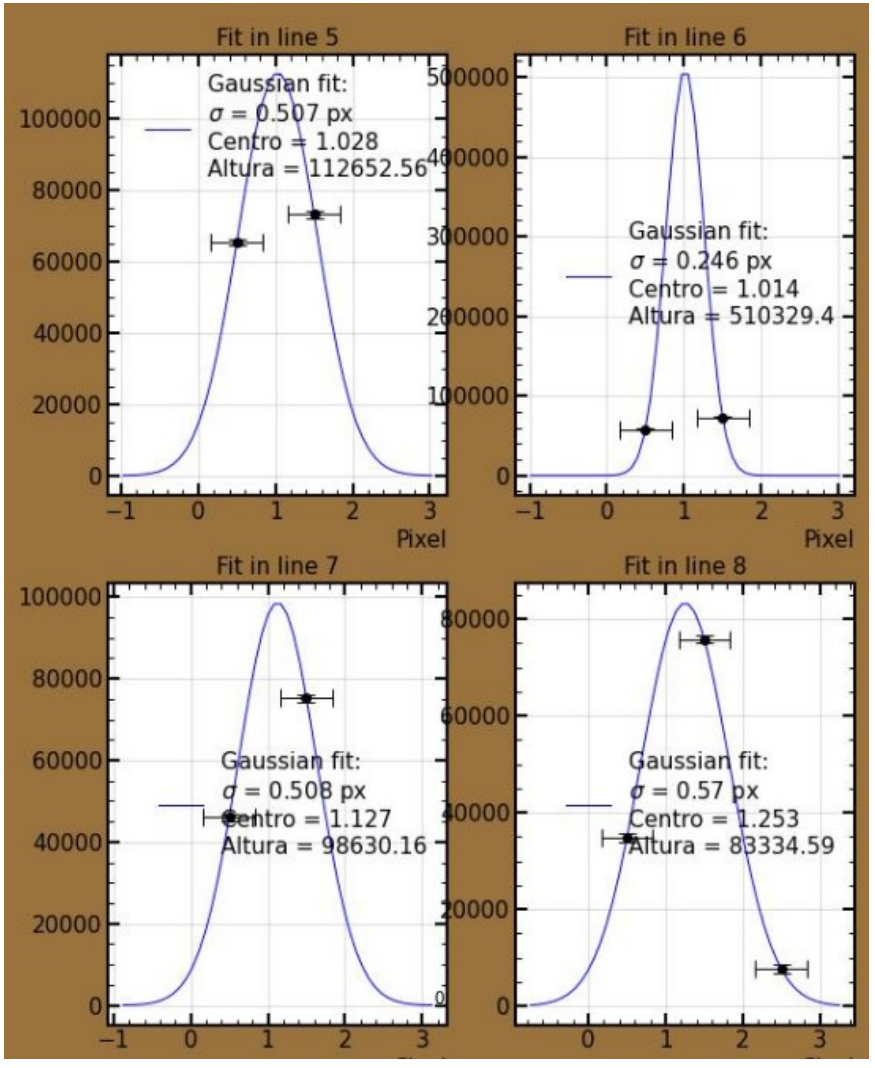


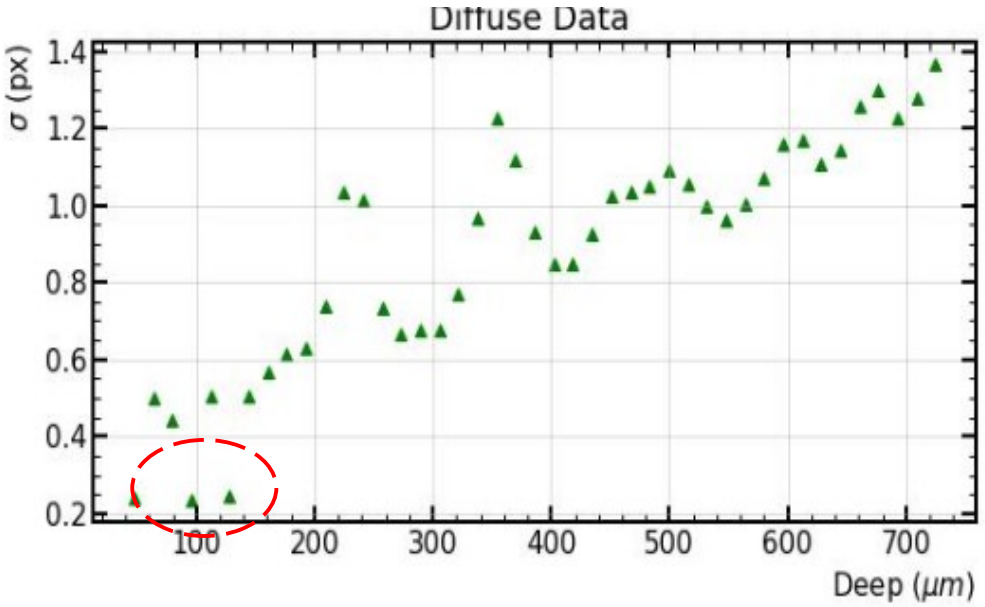
Se realizó un PDF para visualizar el perfil de energías del muon, su modelo de difusión y todas las gaussianas de ajustadas por renglon (con barras de errores).

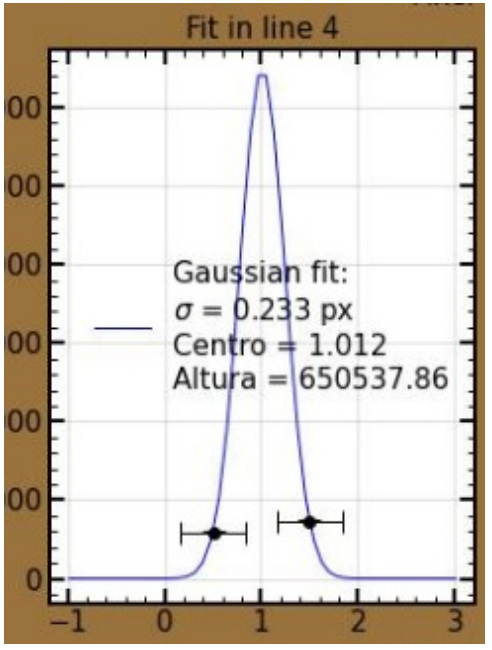


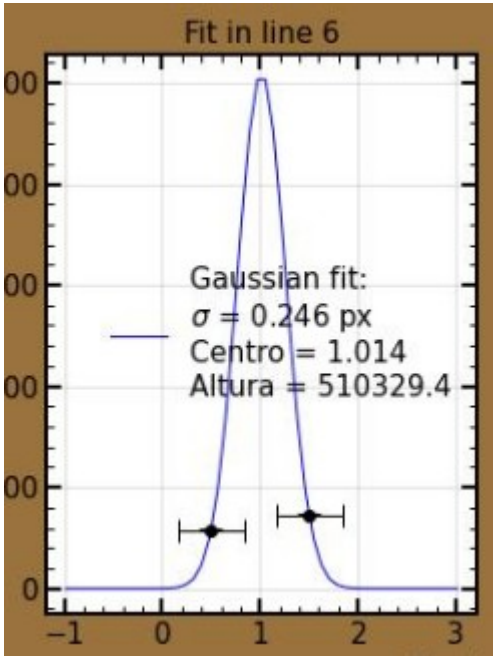



Fit in line 1
Gaussian fit:
σ = 0.238 px
Centro = 0.999
Altura = 681522.3
Pixel
Fit in line 2
Gaussian fit:
σ = 0.5 px
Centro = 1.006
Altura = 123287.68
Pixel
Fit in line 3
Gaussian fit:
σ = 0.44 px
Centro = 1.026
Altura = 134070.43
Fit in line 4
Gaussian fit:
σ = 0.233 px
Centro = 1.012
Altura = 650537.86

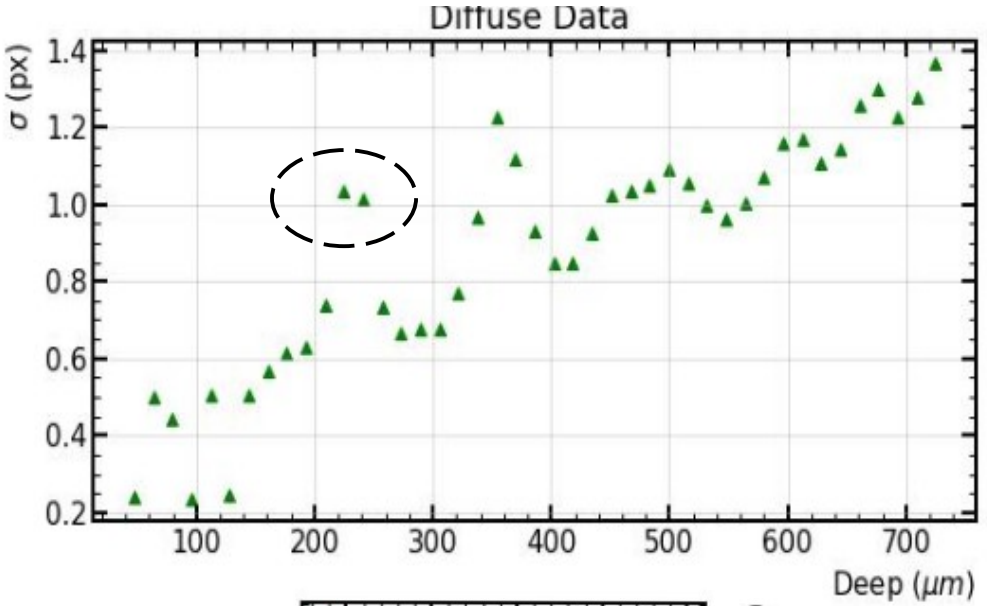

Fit in line 5
Gaussian fit:
σ = 0.507 px
Centro = 1.028
Altura = 112652.56
Pixel
Fit in line 6
Gaussian fit:
σ = 0.246 px
Centro = 1.014
Altura = 510329.4
Pixel
Fit in line 7
Gaussian fit:
σ = 0.508 px
Centro = 1.127
Altura = 98630.16
Fit in line 8
Gaussian fit:
σ = 0.57 px
Centro = 1.253
Altura = 83334.59

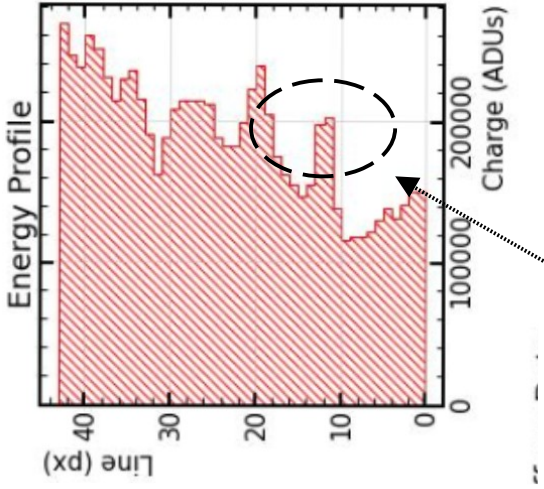


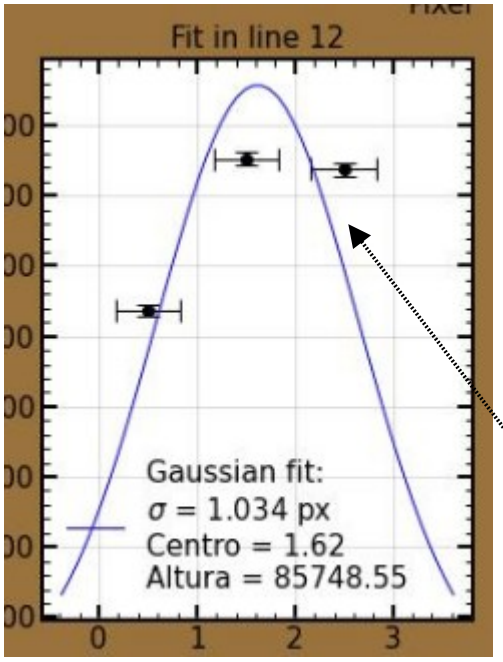


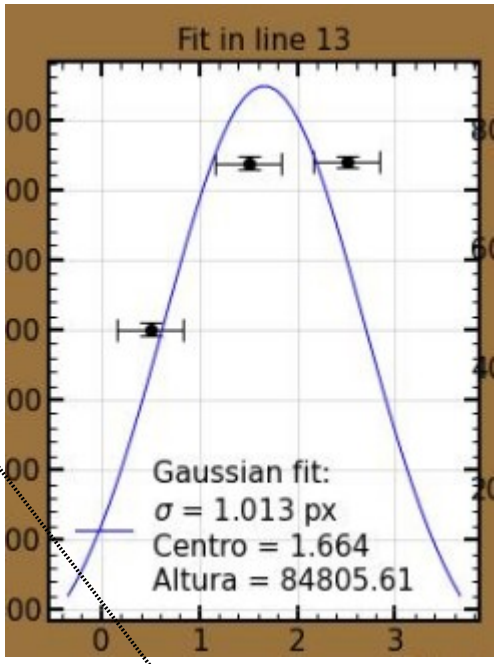


El error en el eje *y* es muy pequeño en comparación con el pico que no se aprecian las barras de error.

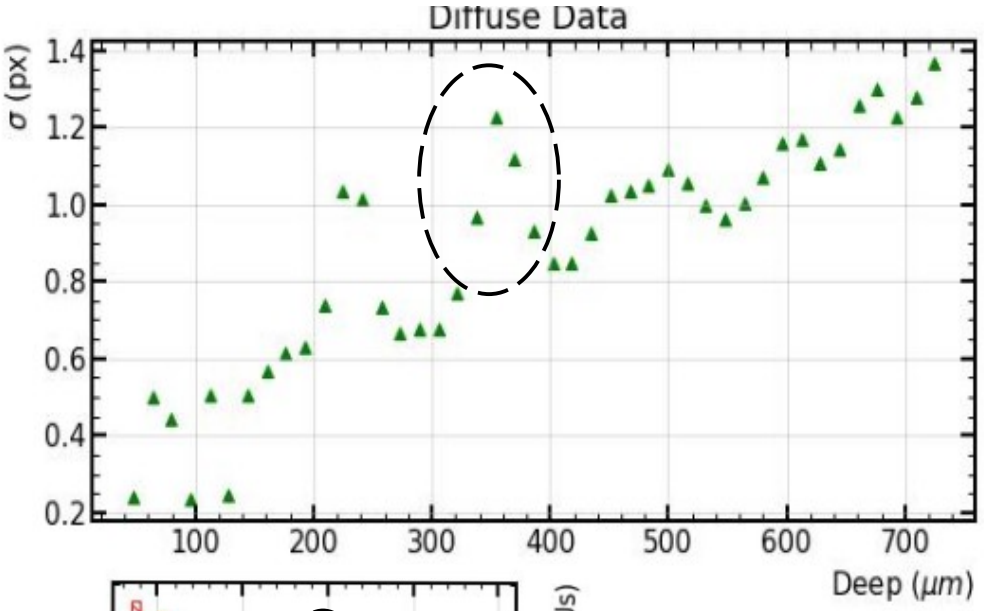


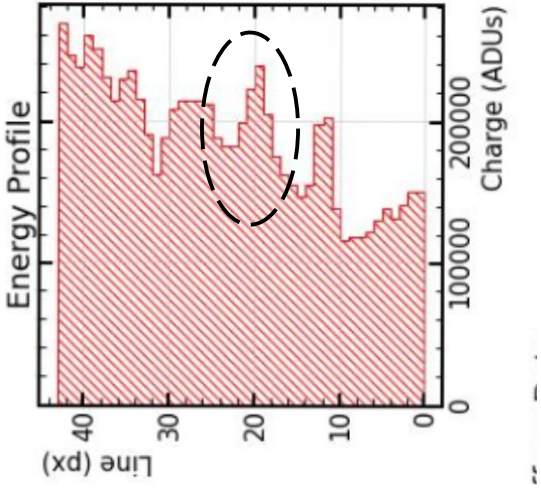




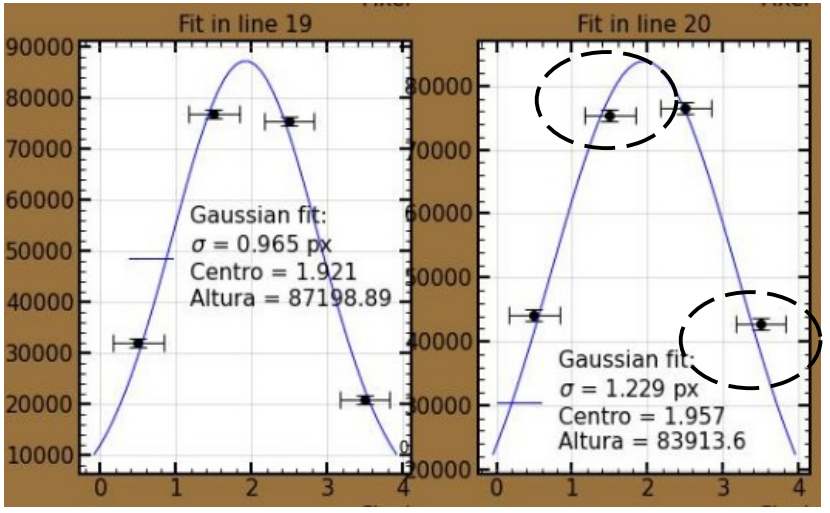


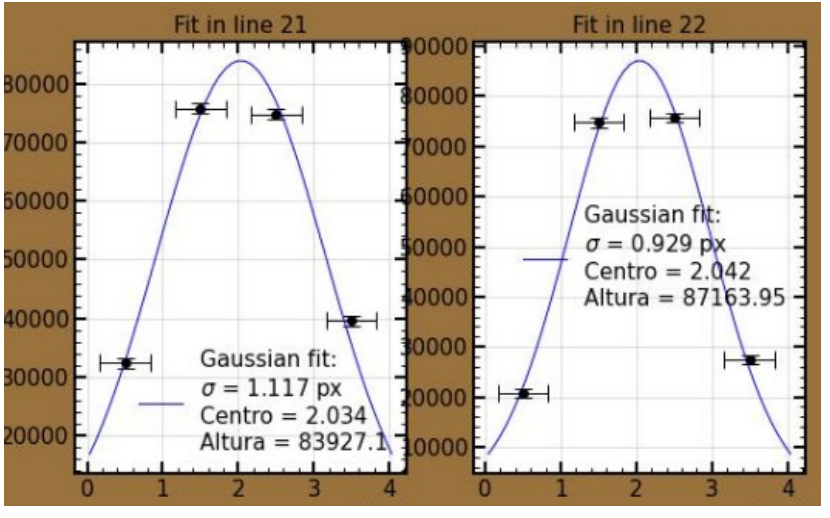
Malos ajustes pero tambien hay un amuento de energía en esos renglones.





Malos ajustes y posibles rayos delta tambien.





Simulación de Primeros Principios

Se está implementando el modelo de difusión para dibujar las trazas de los muones. Comenzamos suponiendo un muon que atraviesa toda la CCD (725 μm), que incide a un ángulo de 45° y deposita una energía de 695 MeV la cual se transforma a electrones con la equivalencia de 1 e- = 3.7 eV.

```
CCD_deep = 725 # micras
muon_deep = 725 # micras
muon_deep_px = muon_deep/15

CCD_deep_px = CCD_deep / 15 # px
ratio_eV_electron = 1 / 3.7 # e-/eV

print('Profundidad de la CCD: ', CCD_deep, ' micras, o ', np.around(CCD_deep_px, 3), ' píxeles')
print('Profundidad del muon: ', muon_deep, ' micras, o ', muon_deep_px, ' píxeles')

Energy_DP = 695 #MeV
Energy_DP_eV = Energy_DP * 10**(6) # eV
electrons = Energy_DP_eV * ratio_eV_electron
ang_theta = 45 # Grados
ang_theta_rad = np.radians(ang_theta)
delta_L = 1000 # micras

delta_XY = np.sqrt(delta_L**2 - muon_deep )
delta_XY_px = delta_XY / 15
# print(np.around(delta_XY_px, 0))
```

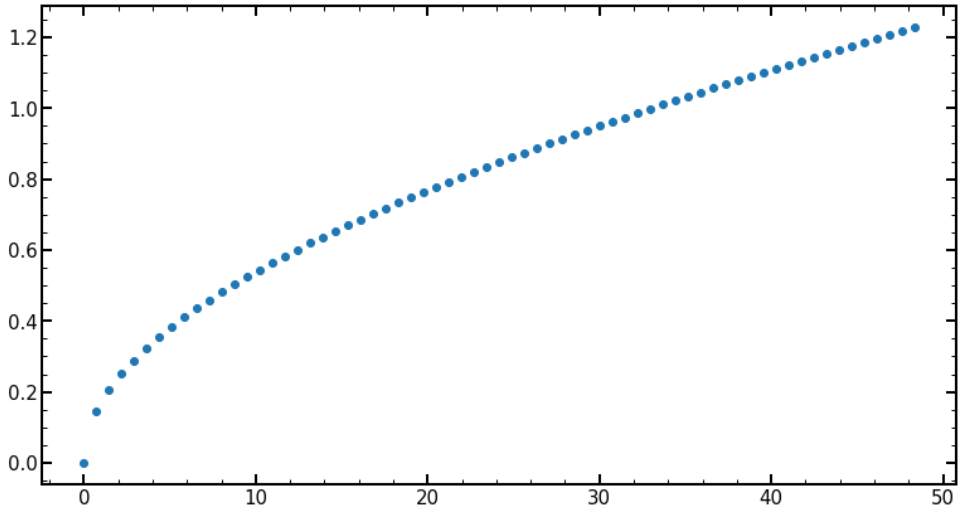
```
Profundidad de la CCD:  725  micras, o  48.333  píxeles
Profundidad del muon:  725  micras, o  48.333333333333336  píxeles
La energía depositada es:  187837837.838  e-
```

Con el dL se obtiene la longitud de la traza sobre el plano XY y se obtienen los valores de las sigmas. Se supone tambien que la carga depositada se distribuye de manera equitativa.

```
La longitud XY de la traza es:  999.637  micras, o  66.642  píxeles
La lista que relaciona XY con profundiad tiene longitud:  67
Longitud de la lista de profundidad:  96
```

```
sigma_values = diffution_curve(list_XY_deep * 15, Alpha, Beta) # micras
# print(len(sigma_values))
# print(len(sigma_values))
```

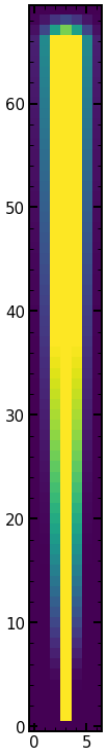
```
A cada linea de la traza le corresponde de energía  2818589.492  e-
```



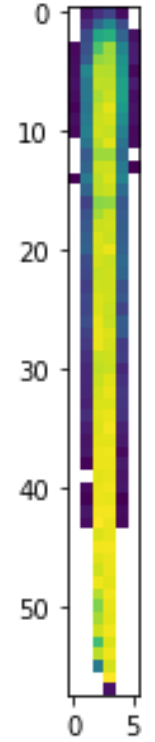
Estoy seguro que debe haber algun error en cuanto al mapeo que relaciona la longitud XY con las sigmas de la profundidad ya que no se toma necesariamente el valor promedio del tamaño del pixel pero de haberlo se rectificará.

Se realizan arreglos de 7x5 para cada uno de los renglones y se van concatenando con un algoritmo realizado iniciando desde la parte superior de la CCD. Tambien se consideran las interacciones a “primeros vecinos”.

```
# type(Gaussian_2
list_image_muon = [ ]

initial_row = [[0, 0,  0,  0,  0, 0, 0]]
list_image_muon.append(initial_row)

for index in np.arange(0, len(list_gaussians)):
    rows_gauss = np.vsplit(list_gaussians[index],5)
    # rows_gauss_2 = np.vsplit(list_gaussians[1],5)
    # rows_gauss_3 = np.vsplit(list_gaussians[2],5)
    # rows_gauss_4 = np.vsplit(list_gaussians[3],5)
    # print(rows_gauss[2])
    if index == 0:
        rows_gauss_next = np.vsplit(list_gaussians[index + 1],5)

        list_image_muon.append(rows_gauss[2] + rows_gauss_next[1])
        contribution_1 = rows_gauss[3]
        contribution_2 = rows_gauss[4]

    elif index == 1:
        list_image_muon.append(rows_gauss[2] + contribution_1)
        contribution_1 = rows_gauss[3] + contribution_2
        contribution_2 = rows_gauss[4]

    elif 1 < index < len(list_gaussians) - 1:
        list_image_muon.append(rows_gauss[2] + contribution_1 + contribution_2)
        contribution_1 = rows_gauss[3] + contribution_2
        contribution_2 = rows_gauss[4]

    elif index == len(list_gaussians) - 1 :
        list_image_muon.append(rows_gauss[2] + contribution_1 + contribution_2)
        list_image_muon.append(rows_gauss[3] + contribution_2)
        list_image_muon.append(rows_gauss[4])
        # list_image_muon.append(rows_gauss[5])

final_row = [[0, 0,  0,  0,  0, 0, 0]]
list_image_muon.append(final_row)
```



Muon simulado



Muon real