Análisis de Imágenes

Imágenes con NSAMP1

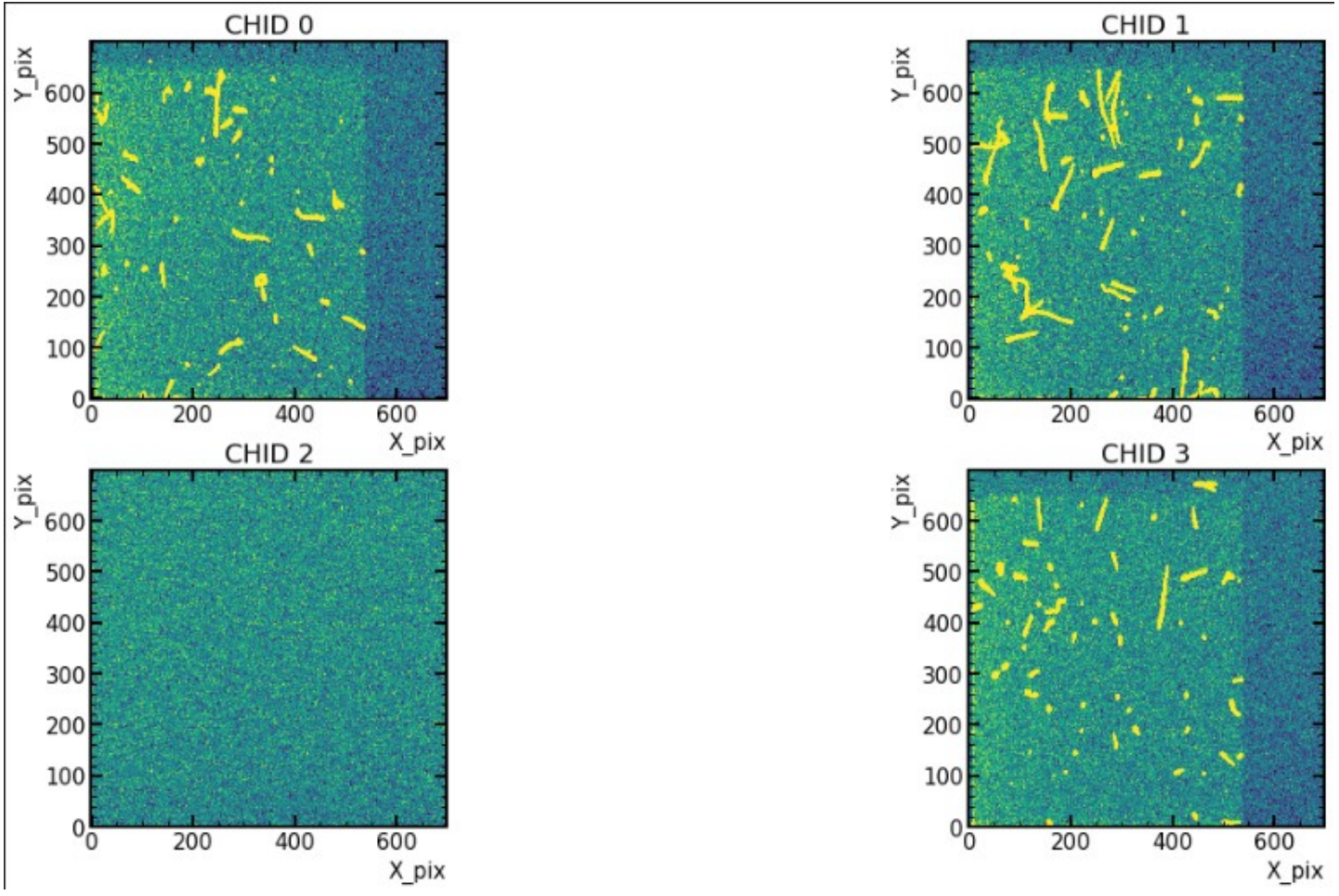
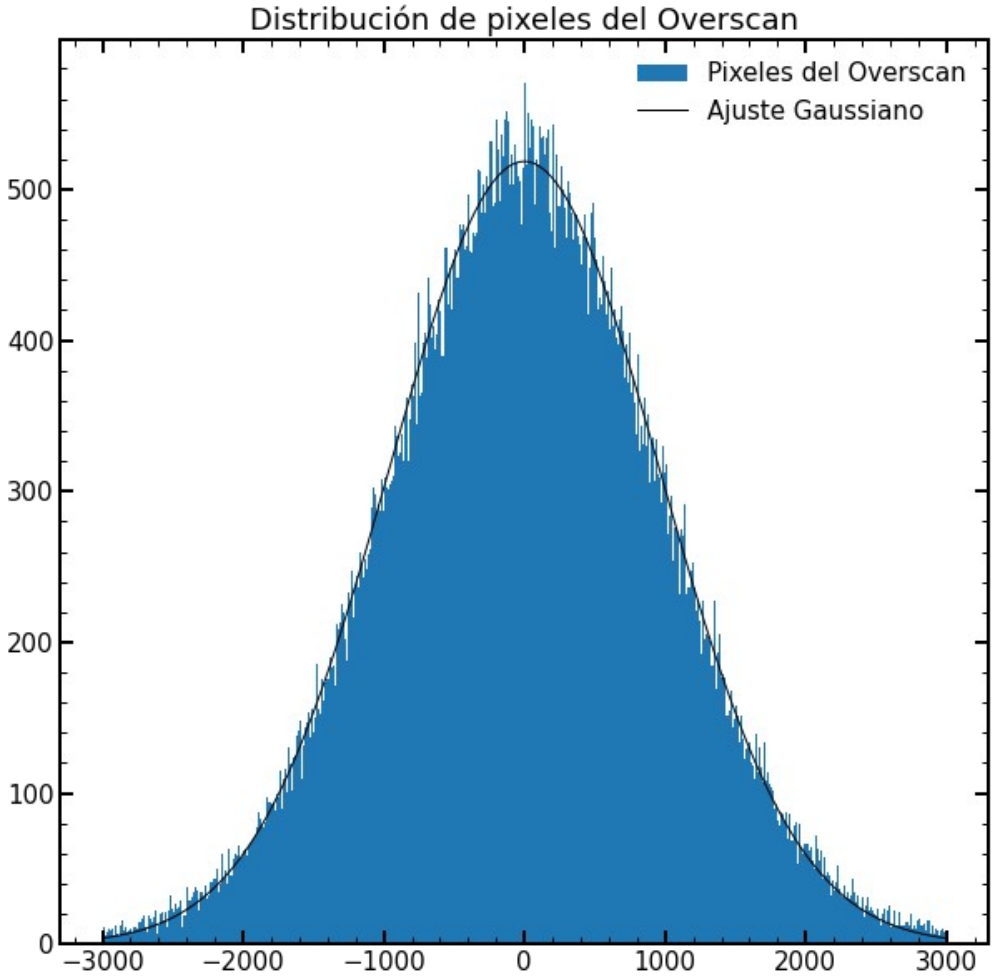


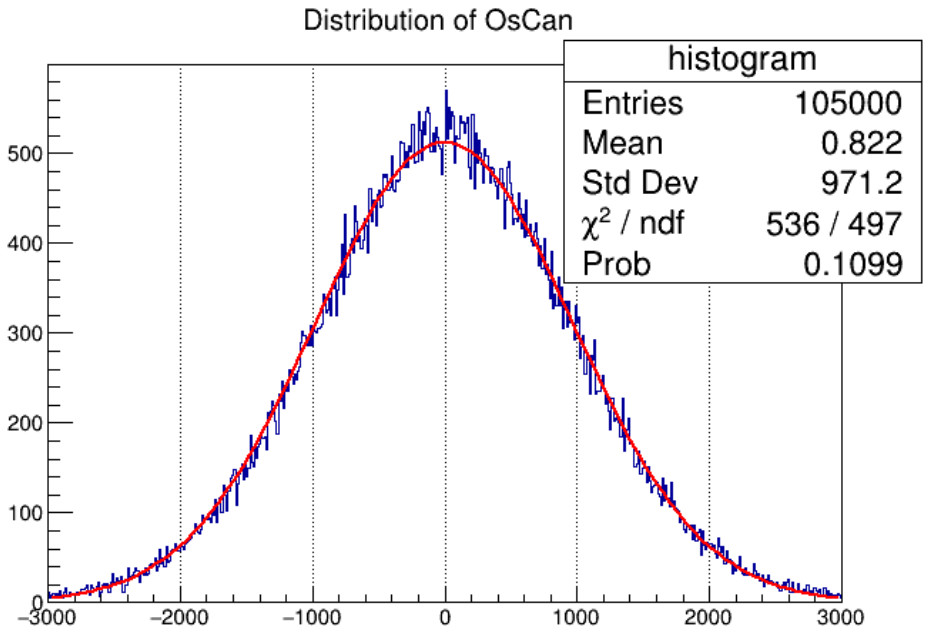
Imagen de típica de NSAMP1, EXPOSURE_1200, NROW_700, NCOL_700 que se han tomado (se cuentan con **≈2200 imágenes**).



A este tipo de imágenes se les ajusta una distribución gaussiana simple. En este caso, con 500 BINS, los parámetros del ajuste, en ADUs, son:

- Media: -0.972
- Amplitud: 518.541
- Sigma: 961.113

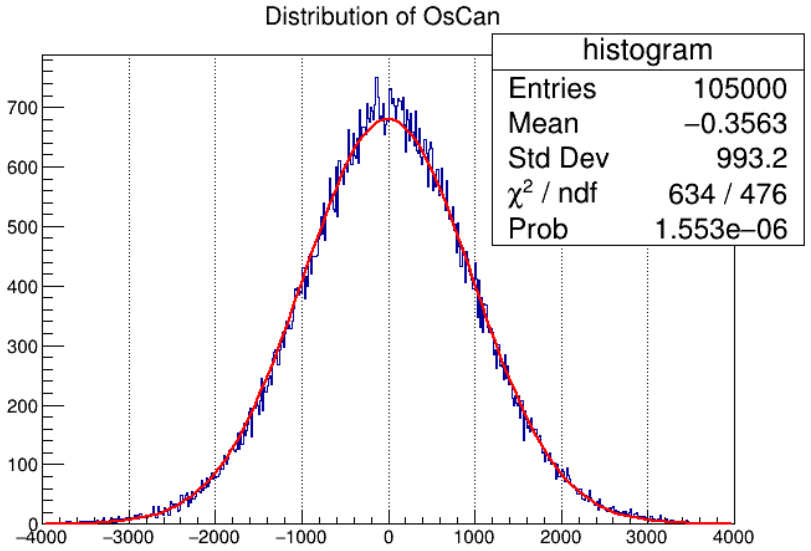
Este ajuste se realiza con la función *curve_fit()* de la librería scipy. Esta paquetería **NO** devuelve directamente los **errores en los ajustes** ni tampoco la **calidad del ajuste** (chi squared, ndf, probabilidad).

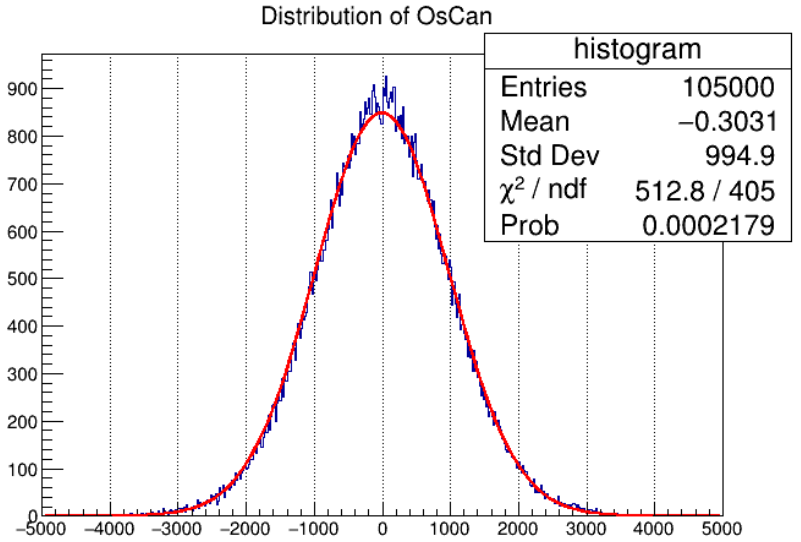


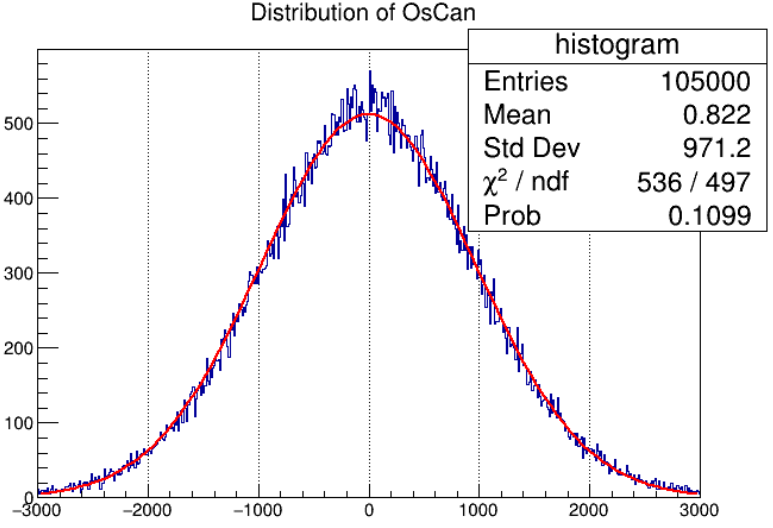
Este otro ajuste es realizado con **pyRoot**. Aquí si es posible obtener tanto los errores como el chi squared, el *ndf* y la probabilidad.

Para este ajuste los parámetros, en ADUs, son:

```
Parameters of the Gaussian Fit
Hight:  511.62385640961145  +-  2.0009642530113547
Mean:  1.0480375650039562  +-  3.0572521497910845
Sigma:  975.0578632190452  +-  2.374885836152714
```







El que tiene una relación chi-squared/ndf mas cercana a uno es cuando se toma el intervalo (-3000, 3000).

En los otros casos el ndf no es el mismo (Checar esto con el asesor).

Imágenes con NSAMP324

16/MAYO/2024

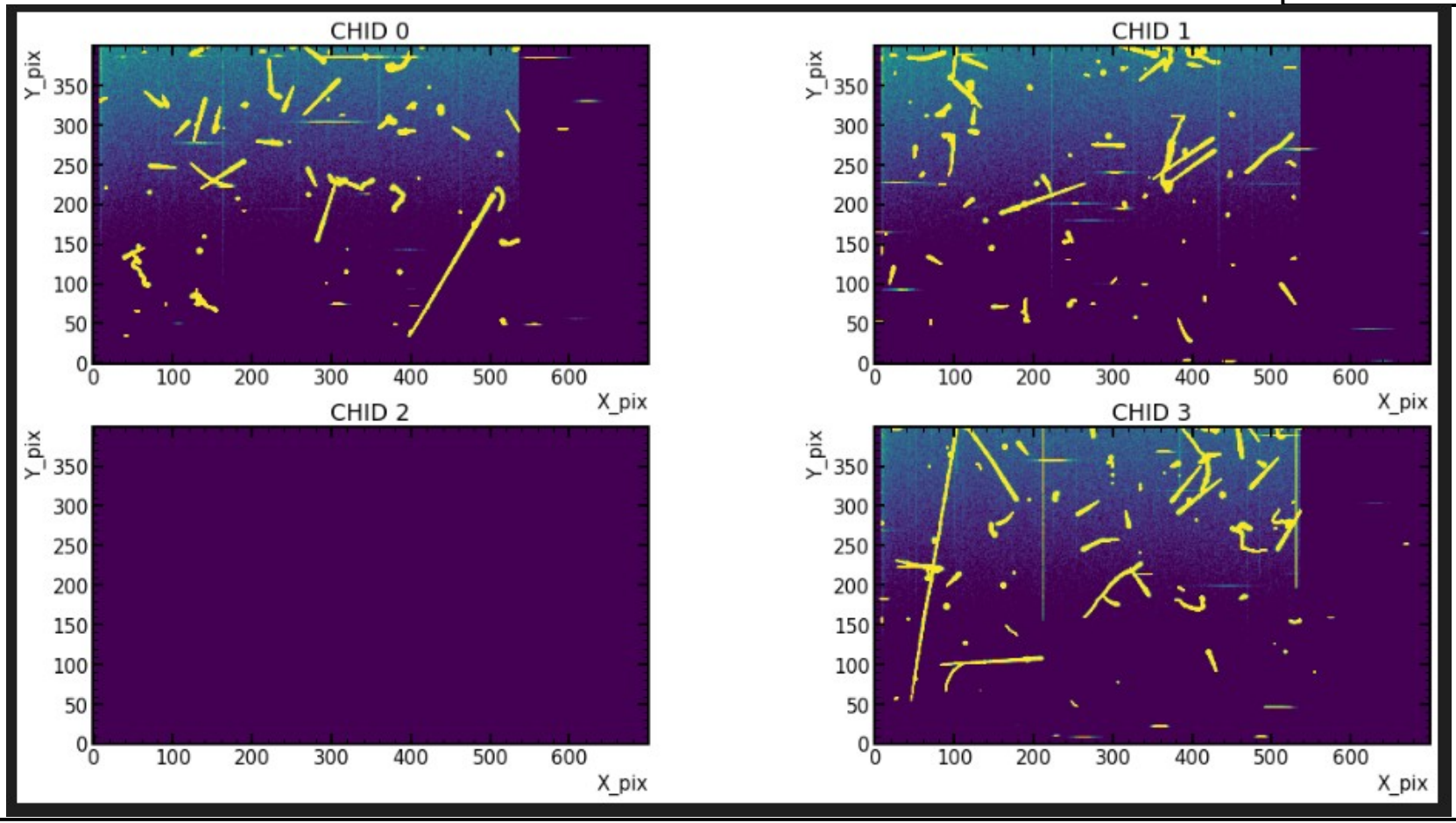
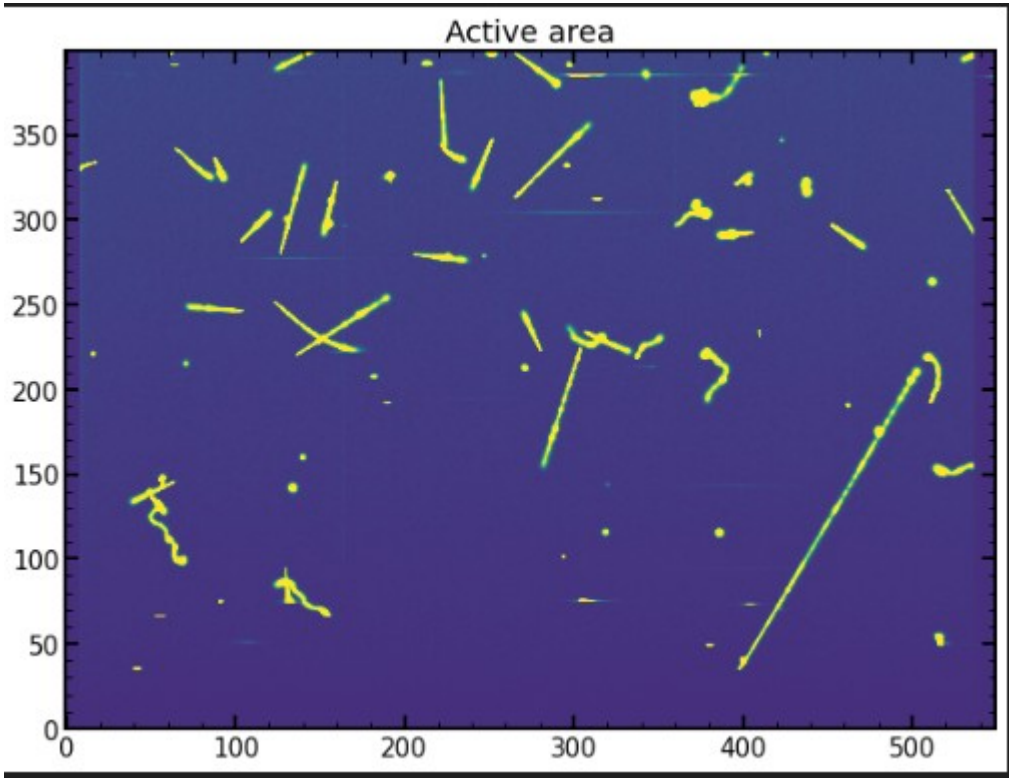
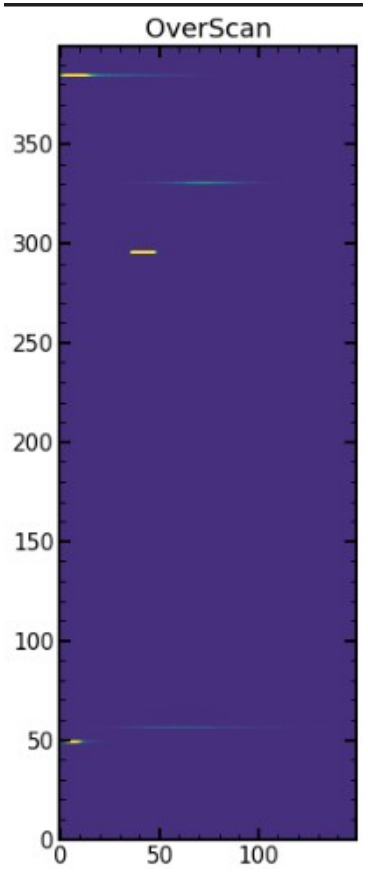


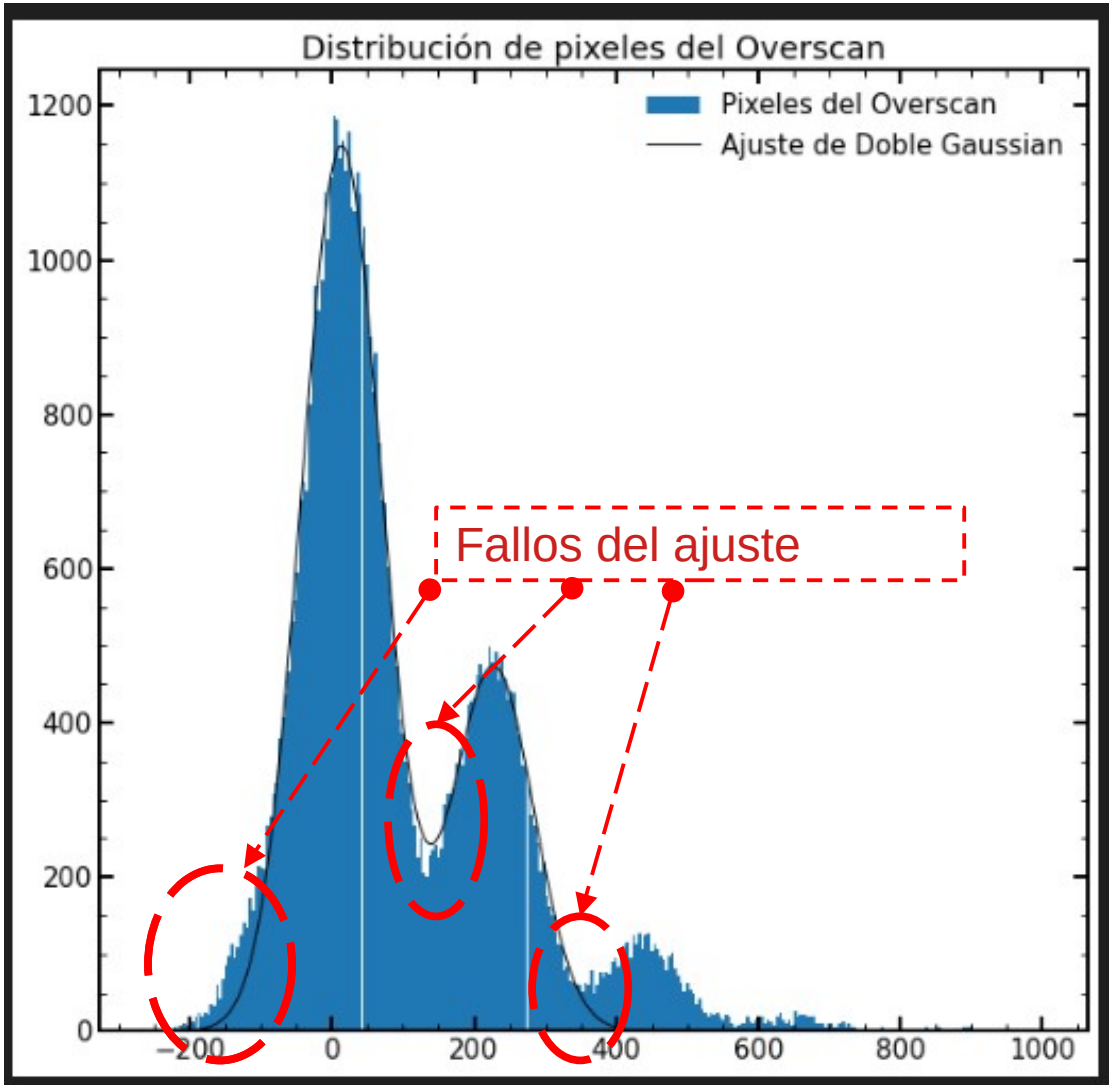
Imagen de típica de NSAMP324, EXPOSURE_0, NROW_400, NCOL_700 que se han tomado

EXTENSIÓN 1





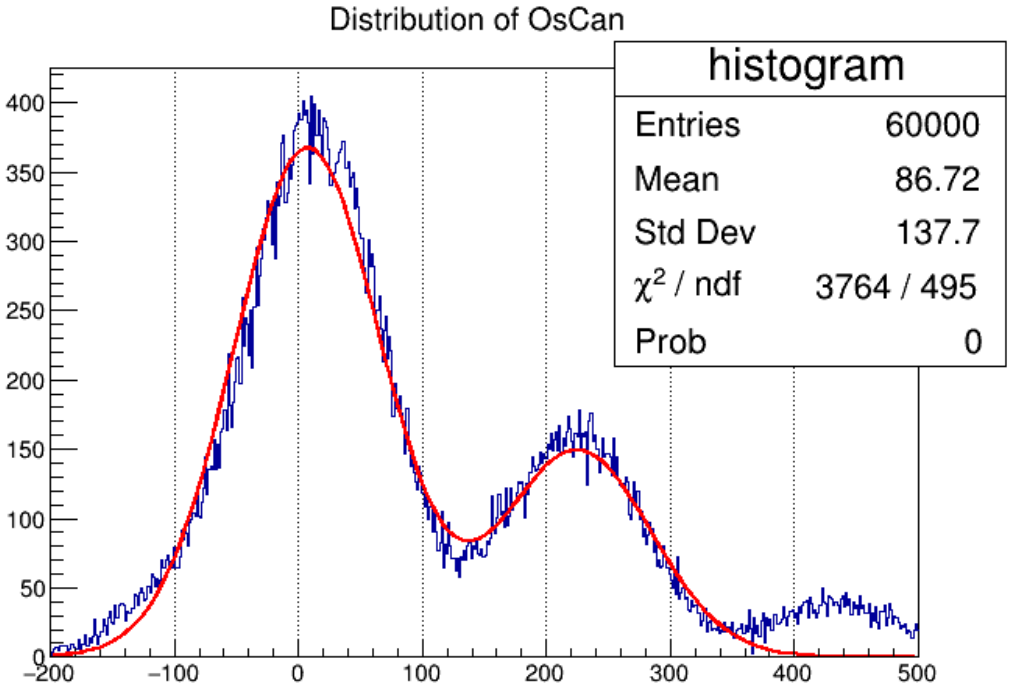
A esta extensión ya se le sustrajo el promedio por renglon.



Para este ajuste, hecho con *curve_fit(),* los parámetros, en ADUs, son:

- Media: 13.446
- Sigma: 56.934
- Ganancia: 214.0283

Nuevamente recordemos que esta función **NO** devuelve directamente los **errores en los ajustes** ni tampoco la **calidad del ajuste.** Además de que tiene muchos errores el ajuste.

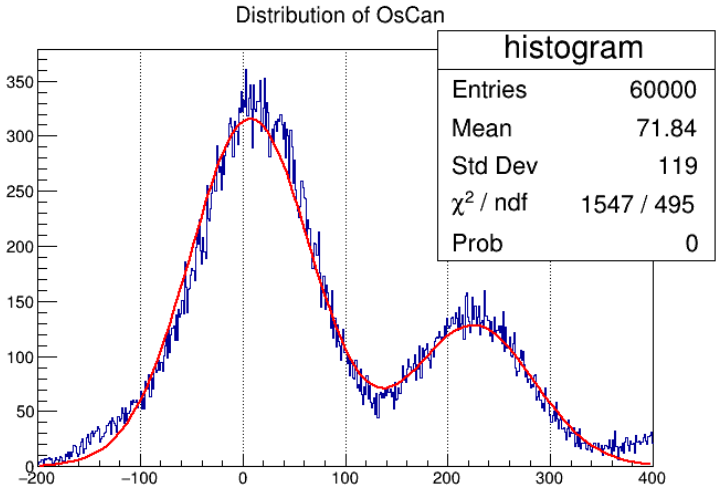


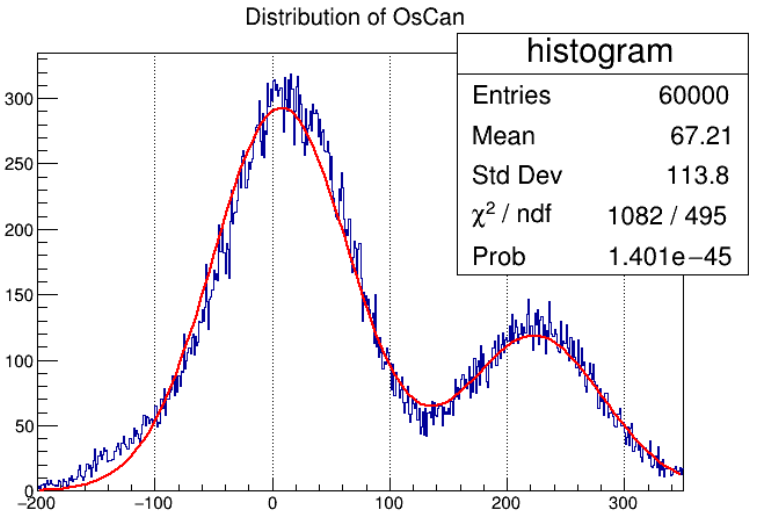
Este otro ajuste es realizado con **pyRoot**. Aquí si es posible obtener tanto los errores como el chi squared, el *ndf* y la probabilidad.

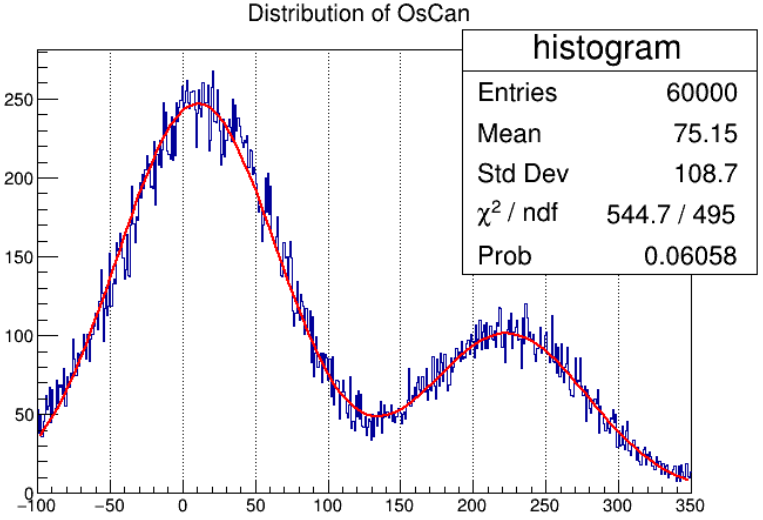
Para este ajuste los parámetros, en ADUs, son:

```
Parameters of the Doble-Gaussian Fit
Mean:  8.058791807742981  +-  0.3436594354170305
Sigma:  59.36355774226314  +-  0.2574608064263725
Gain:  217.6488366821839  +-  0.5962708208571232
```

Pero notemos que los errores en el ajuste siguen presentes pr lo que debemos de ajustar mas el intervalo donde se realiza.







Para este ajuste los parámetros, en ADUs, son:

```
Parameters of the Doble-Gaussian Fit
Mean:  10.709666073347556  +-  0.3472715059669186
Sigma:  55.48398451972508  +-  0.2403648349121254
Gain:  212.7151552911119  +-  0.5887832627459921
```

PyROOT si puede obtener la calidad del ajuste y los errores de los parámetros de manera directa por lo tanto sería buena idea poder utilizarla en los algoritmos.

Se podría implementar en los algoritmos, sin embargo será necesario corroborar **si el cluster del ICN cuenta con PyROOT**, de lo contrario se tendría que ver la posibilidad de instalarlo.

Simulación de Muones

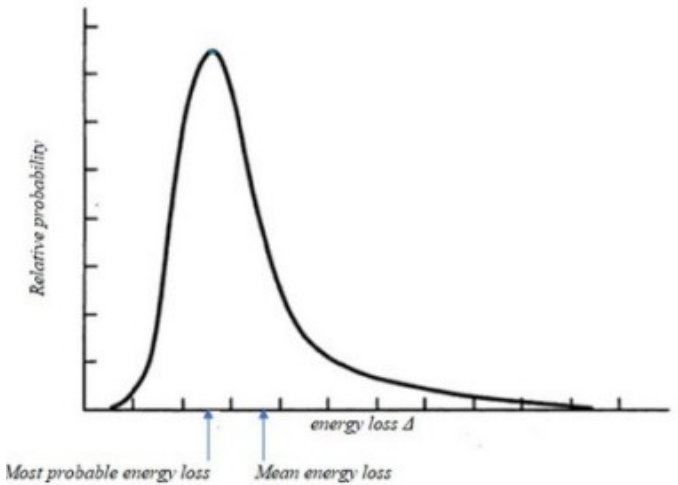
- Se sigue trabajando en obtener la **distribución de Landau** para el grosor específico de la CCD (725 micras) y para muones. El inconveniente siguen siendo los valores que arroja el proceso de integración.

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

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

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

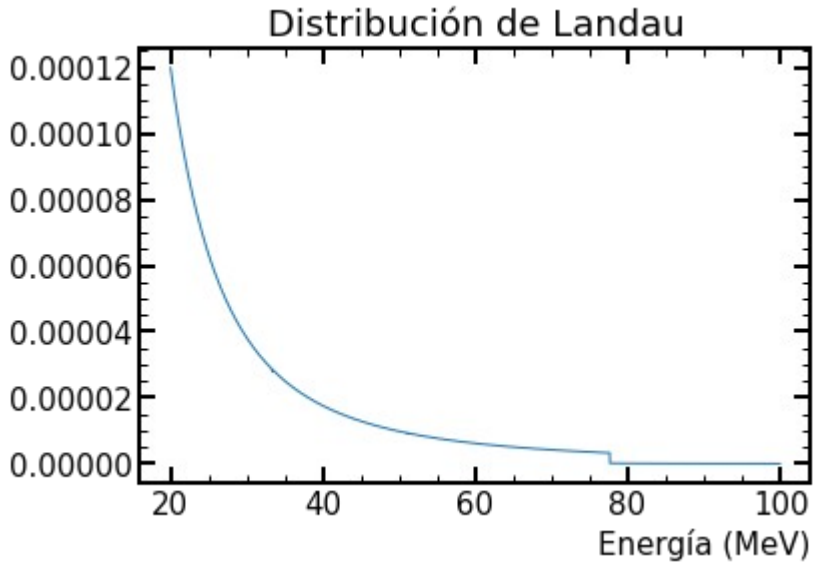
C = Euler's Const = 0.577 ... and



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

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

- En la gráfica de abajo se muestra lo que se supone es la distribución de Landau obtenida de implementar las ecuaciones de la abajo a la izquierda.



Claramente debe haber algun error ya que no se forma la estructura que se espera.

```python
def Landau_dis(thickness, energy):
    # Costantes
    K_Bethe = 0.1535 #MeVcm²/g
    # K_Bethe = 153.5 #KeVcm²/g

    Z = 14
    A = 28.085
    I = 0.000173 # MeV
    # I = 0.173 # KeV

    a = 0.1492
    C0 = -4.4
    X1 = 2.87
    X0 = 0.2014
    m = 3.25

    density = 2.33  # g/cm^3 del Silicio

    muon_mass = 105.66 #MeV/c²
    electron_mass = 0.510998950 # MeV/c²

    # muon_mass = 105660 #KeV/c²
    # electron_mass = 510998950 # KeV/c²
```

Aquí se muestran todas las constantes (y sus unidades) que se utilizan.

Aunque claramente algun valor incorrecto de estos valores no puede generar un error en la forma de la gŕafica

```
# Variables
gamma = (energy + muon_mass) / muon_mass
Beta = np.sqrt(1 - 1/(gamma**2))

xi = K_Bethe * density * Z * thickness/ (A * (Beta**2))  ## MeV
# xi = K_Bethe * density * Z * thickness/ (A * (Beta**2))/1000  ## KeV

ln_eps = np.log(((1 - Beta**2) * I**2)/(2 * electron_mass * Beta**2)) + Beta**2

## Factor lamda
lamda =(1 / xi) * (energy - xi * (np.log(xi) - ln_eps + 1 - 0.577))

## Factor phi(lambda)
# phi_lambda = scy.integrate.quad(phi_factor, 0, np.inf, args=(lamda))[0]/ np.pi
phi_lambda = scy.integrate.quad(phi_factor, 0, np.inf, args=(lamda))[0]/ np.pi

# phi_lambda = scy.integrate.quad(phi_factor, 0, np.inf, args=(lamda))

return phi_lambda/xi
```

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

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

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

C = Euler's Const = 0.577 . . . and

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

```
def phi_factor(x, alfa):
    return np.exp(- x * np.log(x) - x * alfa) * np.sin(x * np.pi)
```

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

Se ha corroborado varias veces el código pero se debe seguir checando para confirmar que aquí no se encuentra el error.

Tambien se tiene que corroborar si la masa de esta ecuación es la masa del electrón o la masa de la partícula incidente sobre todo porque en la definición de Xi si se especifica. Claramente esto afecta los valores de la distribución.

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

¿Masa del electron o masa del muon?

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

Masa del Electron

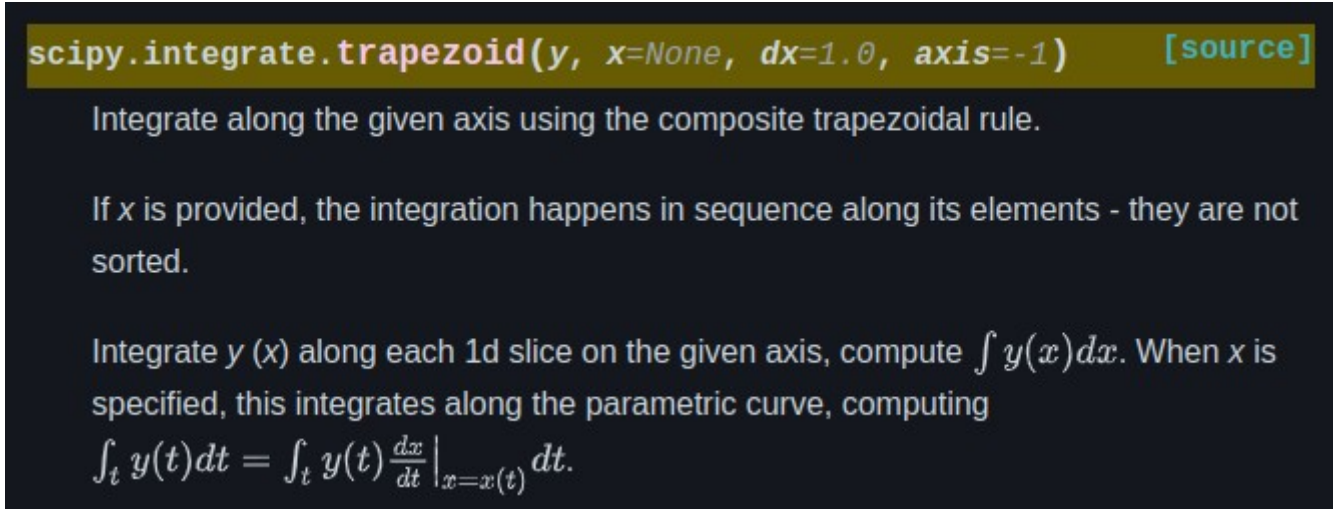
Si no se encuentra ningun error en el código al momento de implementar las ecuaciones entonces el error se deberá a como se realiza la integración.

```
## Factor phi(lambda)
# phi_lambda = scy.integrate.quad(phi_factor, 0, np.inf, args=(lamda))[0]/ np.pi
phi_lambda = scy.integrate.quad(phi_factor, 0, np.inf, args=(lamda))[0]/ np.pi
```

Para realizarlo se una la función **quad()** de la librería **scipy.integrate.** Antes ya se había tenido errores en cuanto a que no se realizaba la integración sin embargo eso se corrigió.

```
Landau_dis(0.0725, 106)
✓ 0.0s
1.41066077123826 92e-09
```

Si se sospecha que aquí se encuentra el error entonces se podría optar por realizar otro tipo de integración. En la misma librería se encuentra una función llamada **trapezoid().**



scipy.integrate.trapezoid(*y*, *x*=*None*, *dx*=*1.0*, *axis*=*-1*) [source]

Integrate along the given axis using the composite trapezoidal rule.

If *x* is provided, the integration happens in sequence along its elements - they are not sorted.

Integrate *y* (*x*) along each 1d slice on the given axis, compute $\int y(x)dx$. When *x* is specified, this integrates along the parametric curve, computing $\int_t y(t)dt = \int_t y(t)\frac{dx}{dt}\Big|_{x=x(t)} dt$.

O si no se puede buscar otra librería útil o intentar realizar la integral *“a mano”***.**