ESRF user meeting tutorial 2026 T10A
Advanced session on pyFAI

ESRF | The European Synchrotron

Parallax:

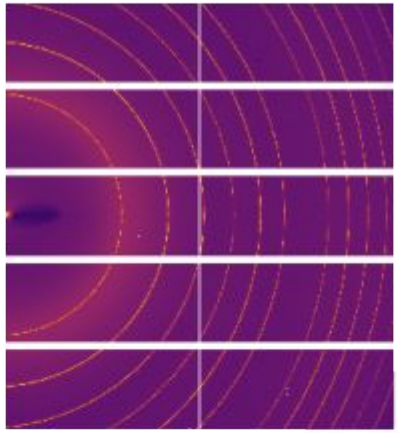
A journey in sensor physics

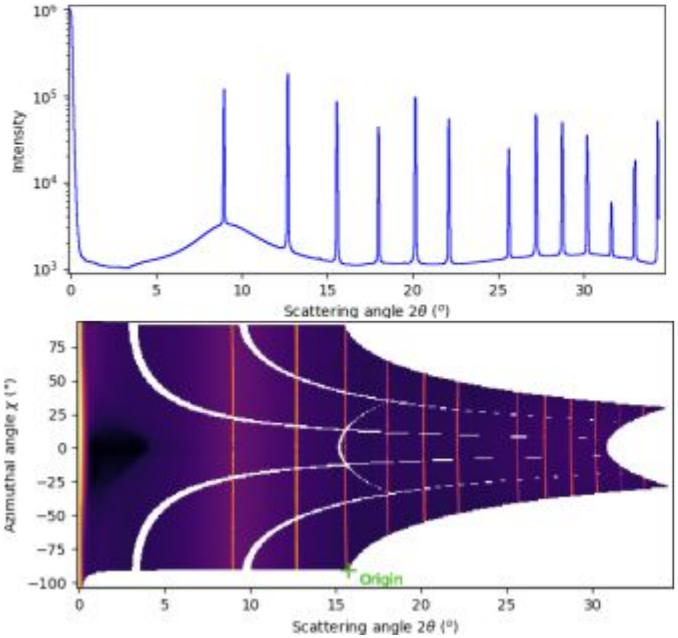
J. Kieffer, with the input from:
G. Lotze, V. Poline, M. Ruat, P-A Douissard, M. Di Michiel,
V. Diadkin, D. Chernyshov

Fast Azimuthal Integration using Python

PyFAI

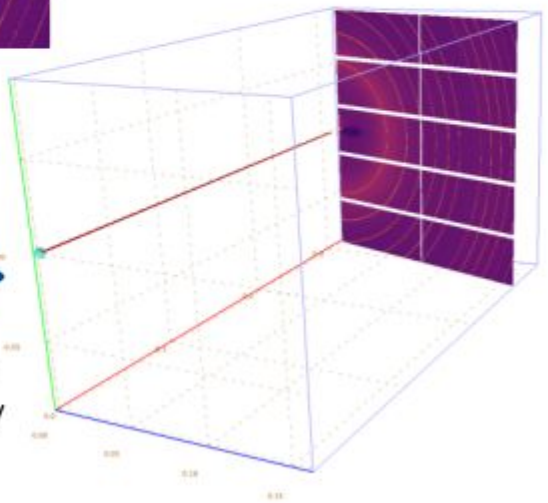
Fast Azimuthal Integration





Average
1d and 2d

Calibrate
3d geometry

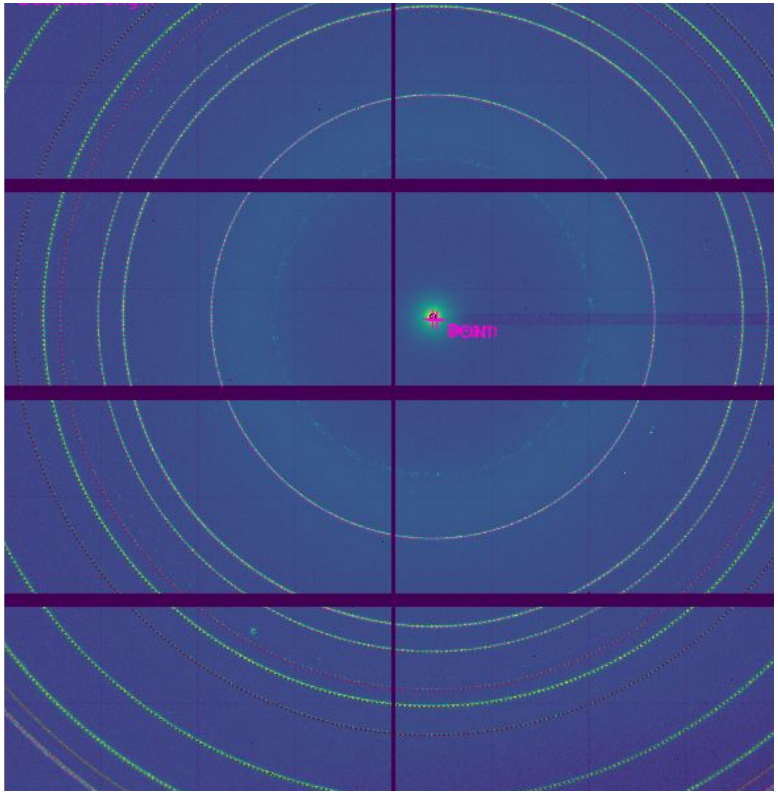


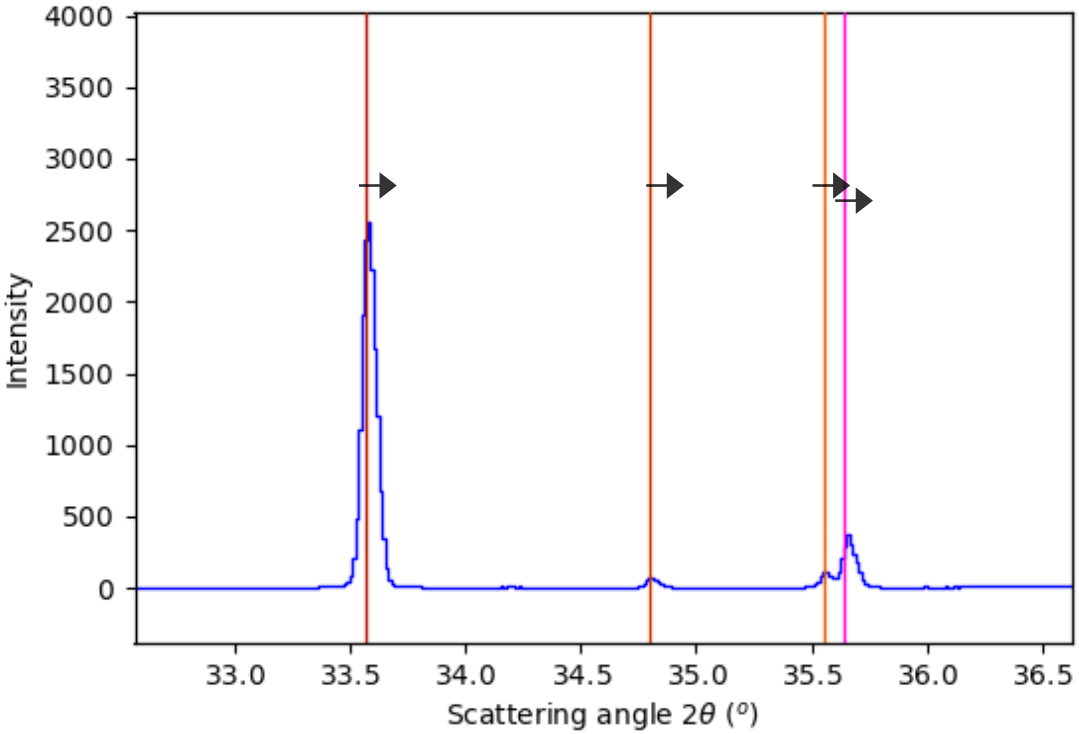
Kieffer, Jerome; Valls, Valentin; Gutierrez-Fernandez; Edgar, Ashiotis; Giannis, Karkoulis, Dimitris; Nawaz, Zubair; Deschildre, Aurore; Vincent, Thomas; Picca, Frederic Emmanuel; Massahud, Emily; Payno, Henri; Huder, Loïc; Wright, Johnathan Paul; Pandolfi, Ronald; Jankowski, Maciej; Paleo, Pierre; Faure, Bertrand; Storm, Malte; De Nolf, Wout; Wright, Christopher J.; Hopkins, Jesse B,; Pascal, Elena; Weninger, Clemens; Detlefs, Carsten; Plaswig, Florian; Lavanchy, Aurelien; gbenecke; zxs-un; iltommi11; dodogerstlin; harshal301002; Lotze, Gudrun

https://zenodo.org/records/17984195

Parallax effect observed at ESRF-ID13

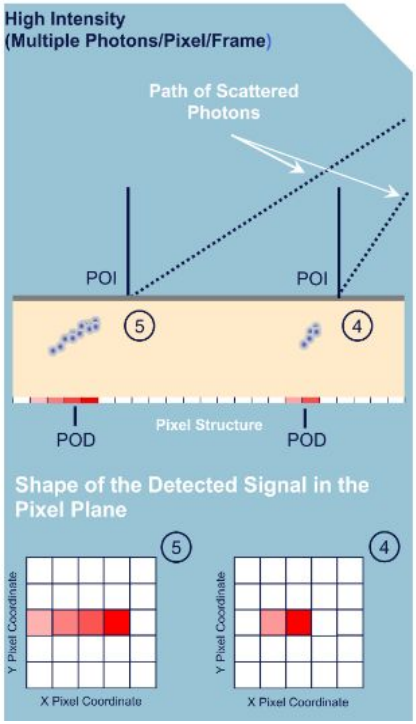
- **Eiger 4M detector 75µm pixel size**
- **Silicon sensor 450µm thickness**
- **Sample: α-Al_2O_3 @13,45 keV**

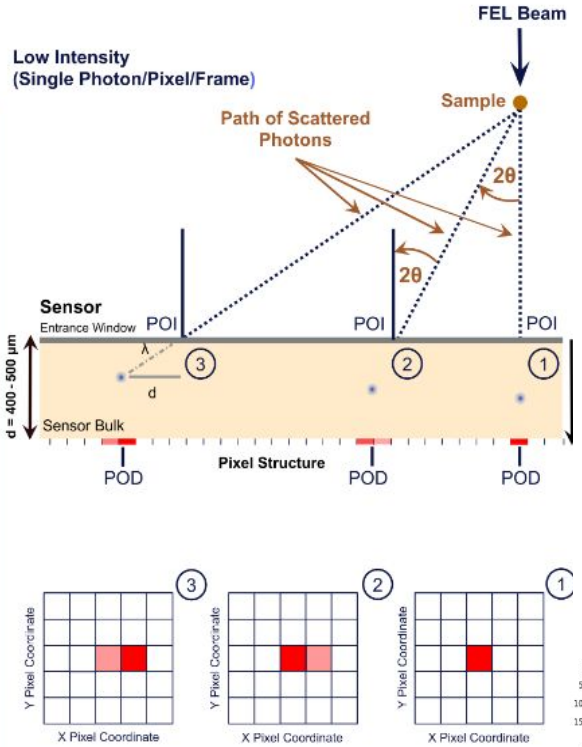




At large incident angle (>30°), photons get detected at even larger scattering angles

Parallax in photon detection

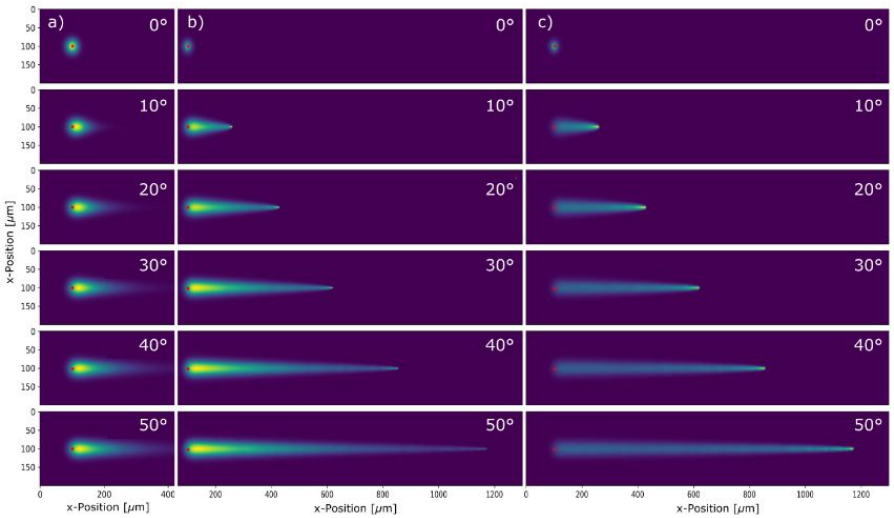




Beer–Lambert law:

$$-d\,I = \mu \cdot I \cdot d\,x$$

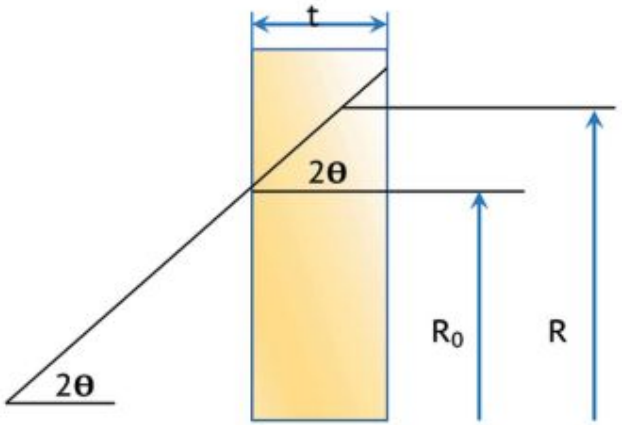
$$I = I_0 \cdot e^{-\mu \cdot r}$$

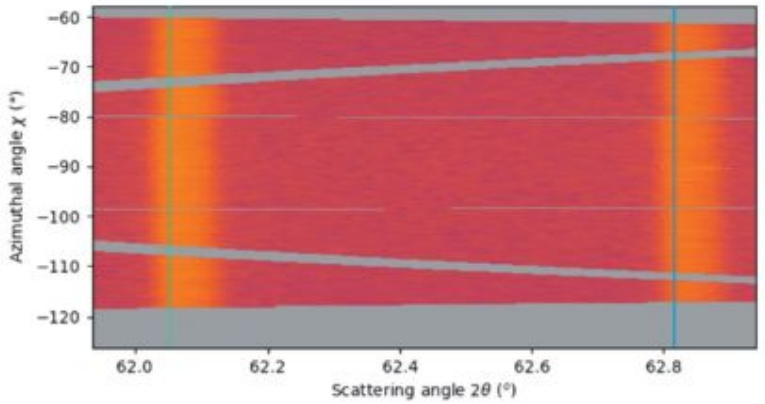


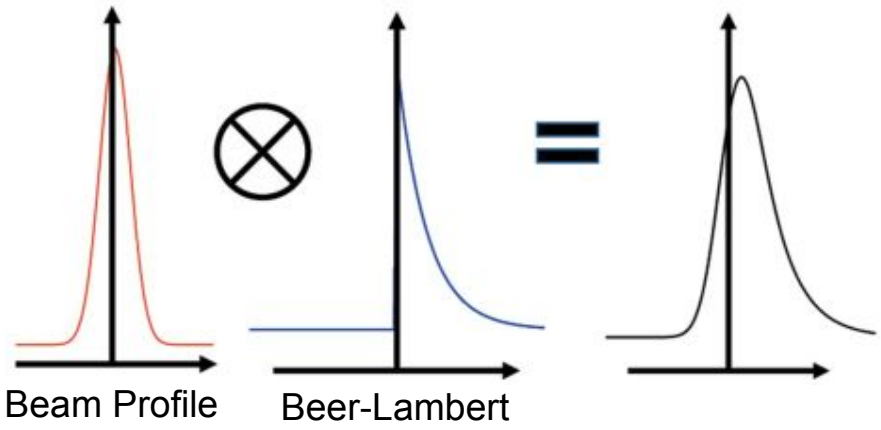
On the Influence of Parallax Effects in Thick Silicon Sensors in Coherent Diffraction Imaging

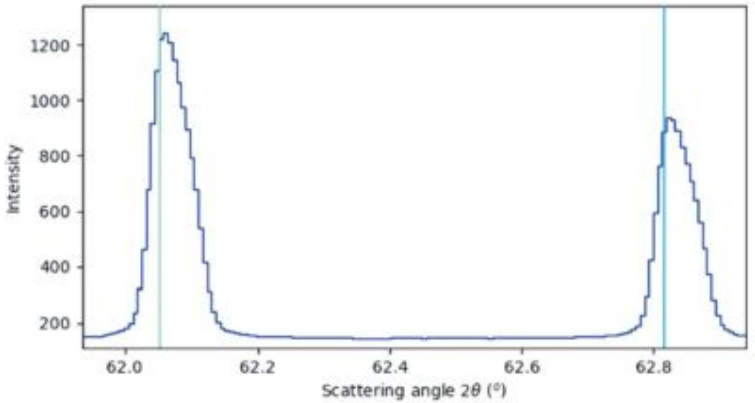
doi:10.1088/1742-6596/3010/1/012135

Sensor absorption convoluted with beam profile (1d)







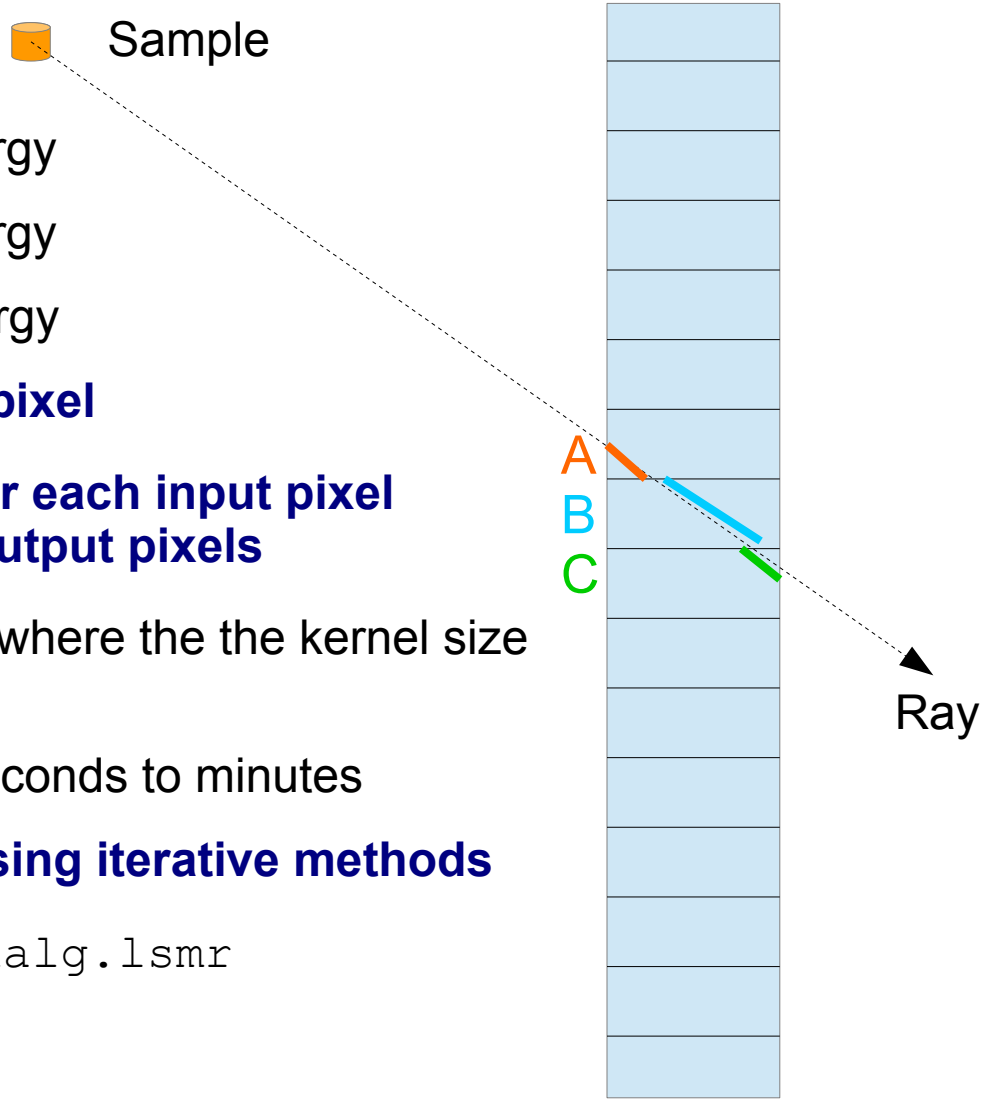


On the resolution function for powder diffraction with area detectors
https://doi.org/10.1107/S2053273321007506

https://github.com/silx-kit/pyFAI/blob/main/doc/source/usage/tutorial/ThickDetector/Parallax_model.ipynb

Ray-tracing of absorption in pixel depth

- **Pixels:**
  - A→Length→Deposited energy
  - B→Length→Deposited energy
  - C→Length→Deposited energy
- **Throw thousands of rays per pixel**
- **Build a (sparse) matrix with for each input pixel provides the contribution to output pixels**
  - Similar to a convolution but where the the kernel size changes from pixel to pixel
  - Matrix construction takes seconds to minutes
- **Possible to invert the effect using iterative methods**
  - Using `scipy.sparse.linalg.lsmr`
  - MLEM for poissonian data.
- **Not taking into account the integrating/counting nature of the detector**

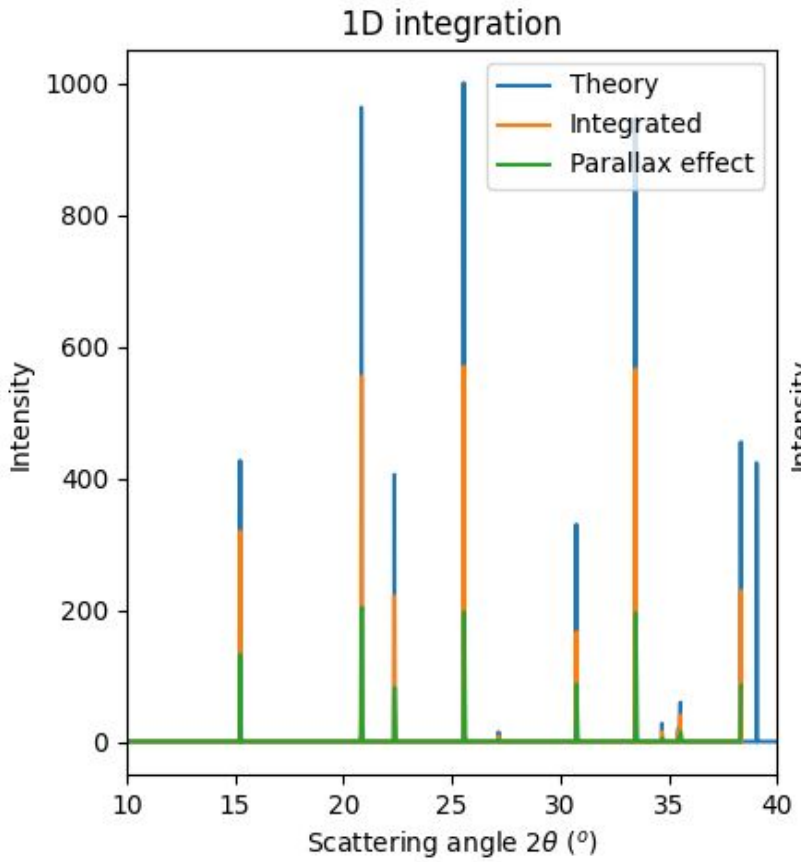


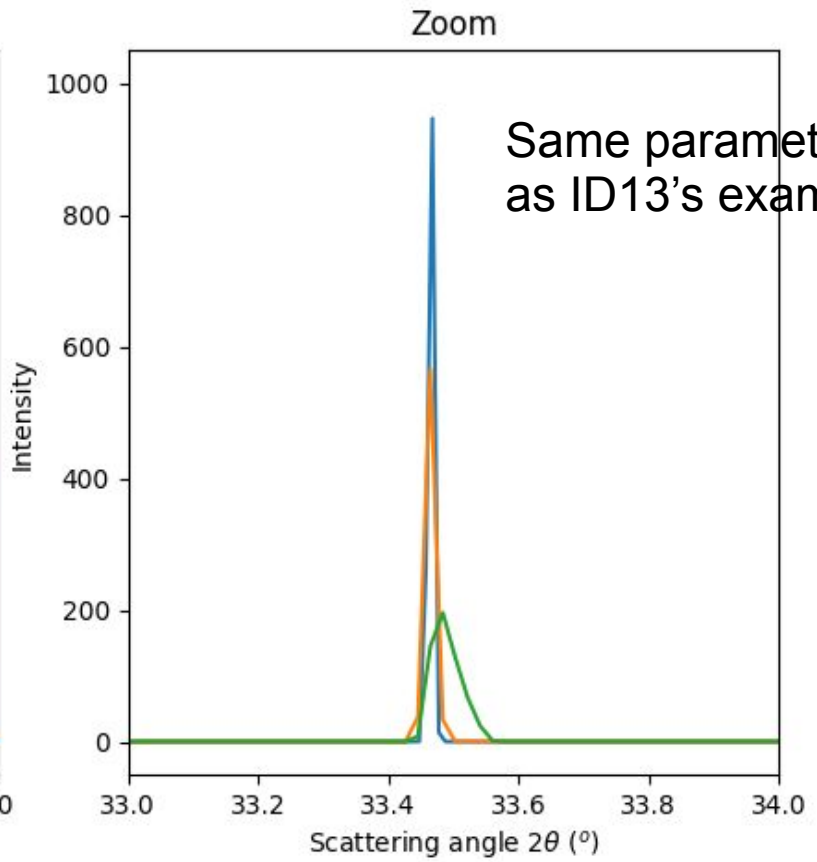
Simulation of the parallax effect in the sensor with pyFAI:

- **Based on ray-tracing:**

https://github.com/silx-kit/pyFAI/blob/main/doc/source/usage/tutorial/ThickDetector/raytracing.ipynb

  - Build a blurring operator by throwing thousands of rays per pixel
  - Can be used to de-convolute with iterative methods (i.e. MLEM, …)

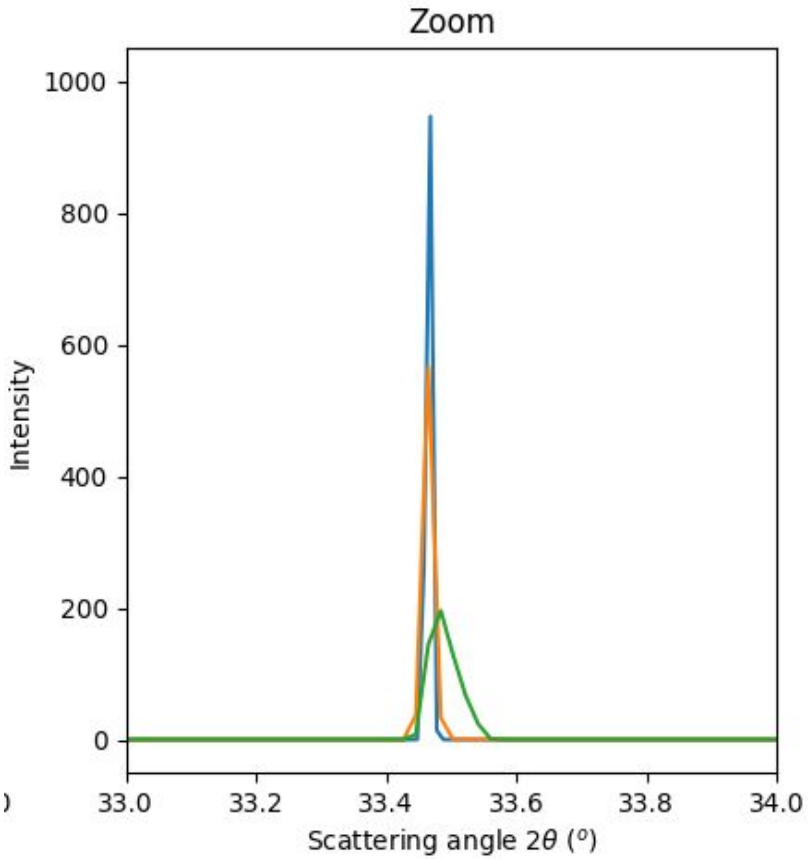




Same parameters as ID13's example

Exercises: Apply parallax effect on synthetic data

- **Build a synthetic geometry**
- **Create a calibrant**
- **Simulate a powder diffraction pattern and the image**
- **Perform the raytracing for the detector**
- **Blur the image**
- **Integrate the images and observe the effect of the parallax**



https://github.com/silx-kit/pyFAI/blob/main/doc/source/usage/tutorial/ThickDetector/Parallax_simple.ipynb

Correction of the parralax effect ?

Different strategies to correct the effect:

- Pseudo-inversion (LSMR, MLEM, …)
- Take into account the effect in peak-position

Thick sensor approximation: ⚠ math ⚠

Average value of an observable: $\langle x \rangle = \int_0^\infty \mathrm{x}(z) \cdot \mathrm{P}(z)\, dz$

Geometrically, one has: $\mathrm{x}(z) = z \cdot \tan(\alpha)$

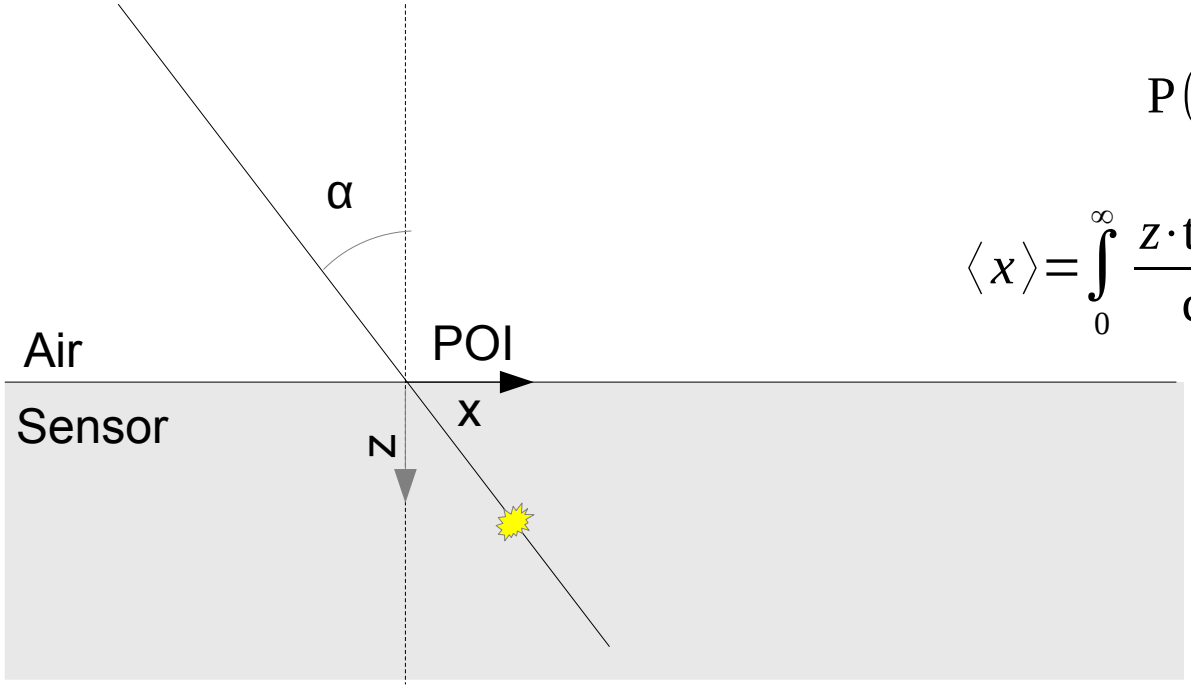
P(*z*) follows Beer-Lambert law: $\mathrm{P}(z) = A\, e^{\frac{-\mu z}{\cos(\alpha)}}$

Normalization: $1 = \int_0^\infty \mathrm{P}(z)\, dz$

Which leads to: $A = \frac{\mu}{\cos(\alpha)}$

$$\mathrm{P}(z) = \frac{\mu}{\cos(\alpha)} \cdot e^{\frac{-\mu z}{\cos(\alpha)}}$$

$$\langle x \rangle = \int_0^\infty \frac{z \cdot \tan(\alpha) \cdot \mu}{\cos(\alpha)} e^{\frac{-\mu z}{\cos(\alpha)}} dz = \frac{\sin(\alpha)}{\mu}$$

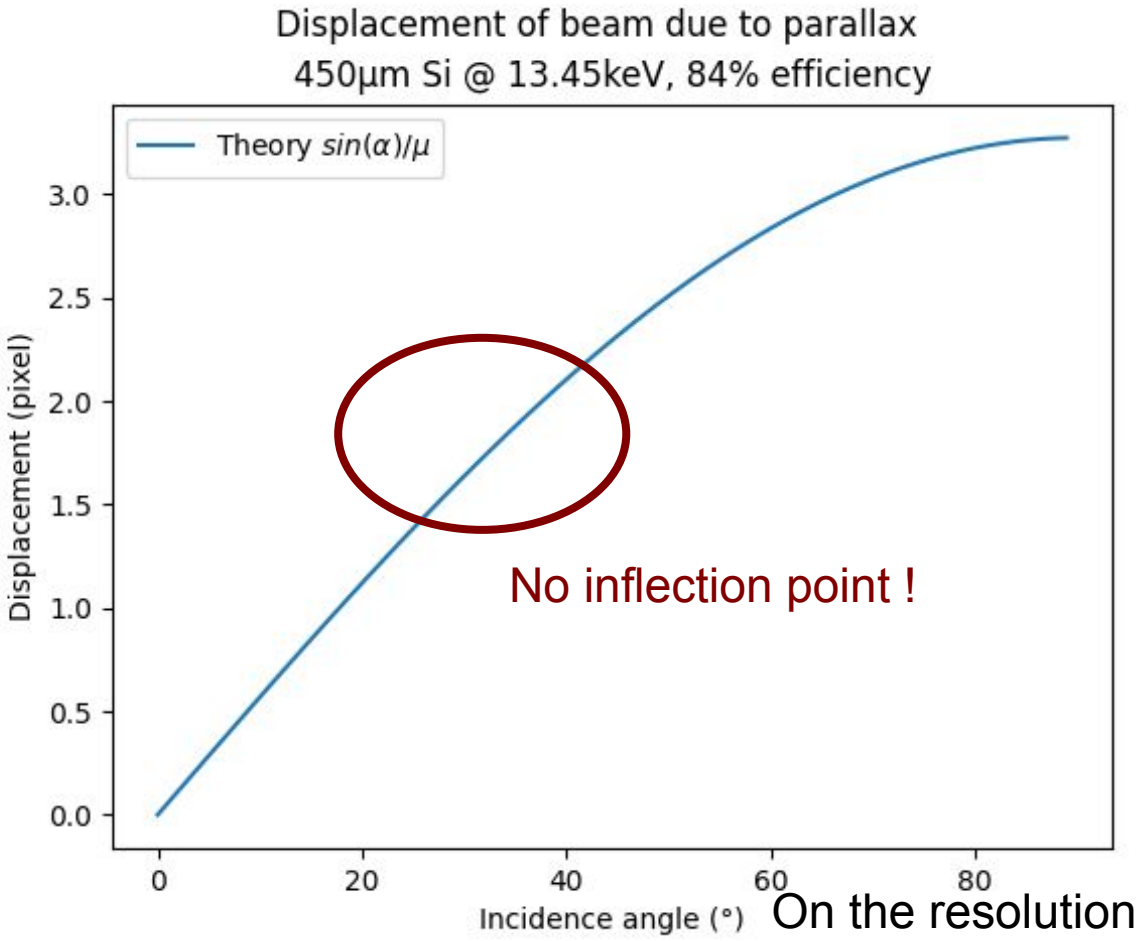


$$\Delta x = \frac{\sin(\alpha)}{\mu}$$

Thick-sensor model: quantification

- **All photons are caught**
- **Does not depend on the sensor thickness !**

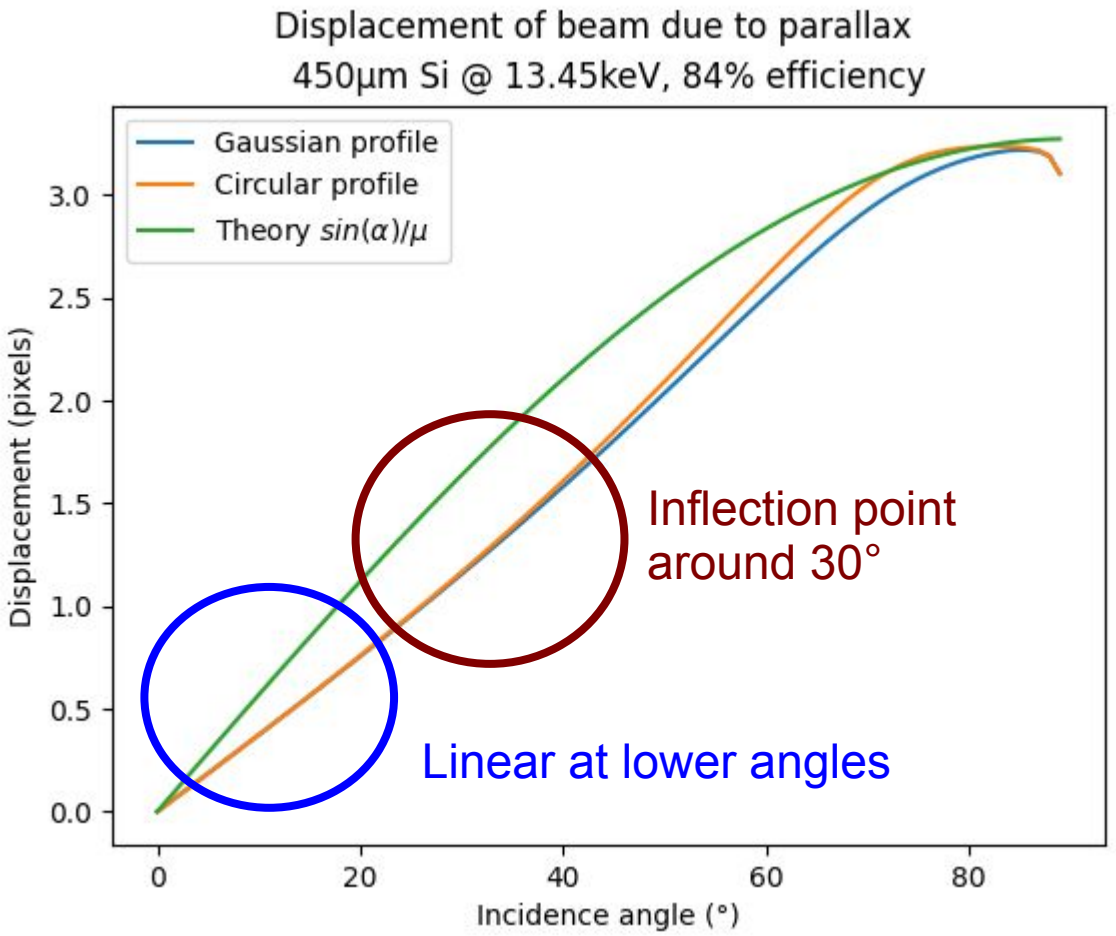
$$\Delta x = \frac{\sin(\alpha)}{\mu}$$



Does not match observation !
→ Displacement decreases with incident angle

On the resolution function for powder diffraction wit
https://doi.org/10.1107/S2053273321007506

Thin-sensor convoluted with beam profile (1d), quantification



Issues:
* Requires the beam profile
* Numerical convolution

⇒ Search for a solution independent of the profile

http://www.silx.org/doc/pyFAI/dev/usage/tutorial/ThickDetector/Parallax_model.html

Thin-sensor without beam profile: ⚠ math ⚠

Average value of an observable: $\langle x \rangle = \int_0^e \mathrm{x}(z) \cdot \mathrm{P}(z)\, dz$

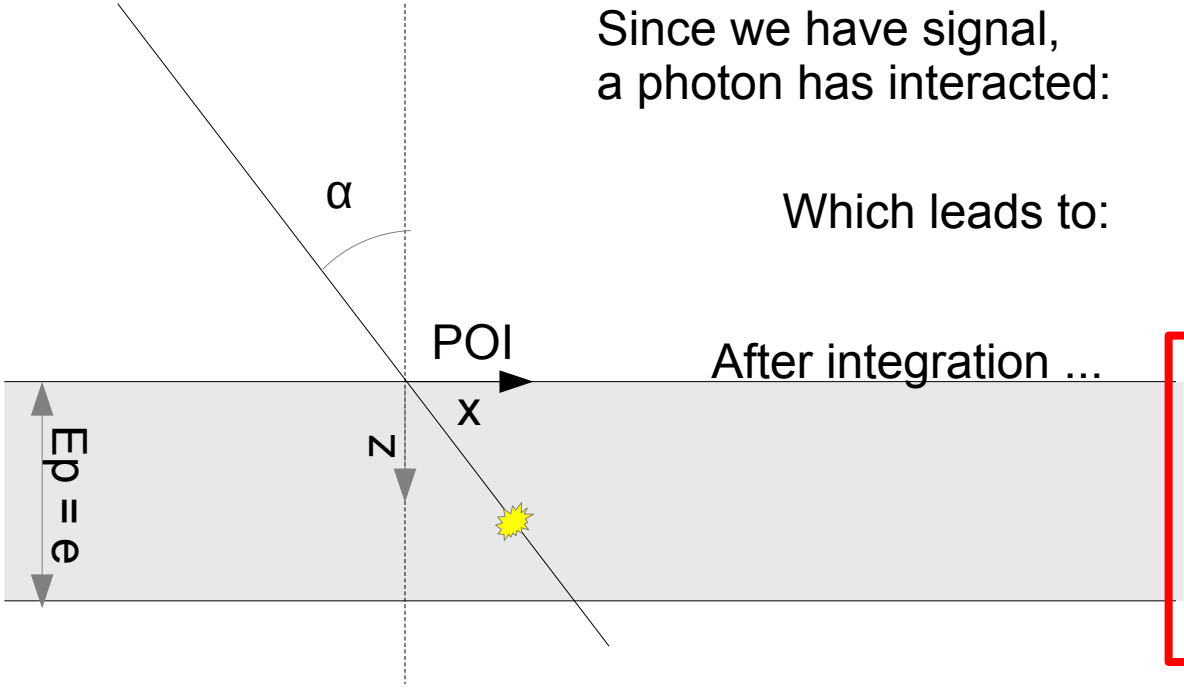
Geometrically, one has: $\mathrm{x}(z) = z \cdot \tan(\alpha)$

P(*z*) follows Beer-Lambert law: $\mathrm{P}(z) = A\, e^{\frac{-\mu z}{\cos(\alpha)}} = A\, e^{-\mu' z}$

With: $\mu' = \frac{\mu}{\cos(\alpha)}$, *A* constant

Since we have signal, a photon has interacted: $1 = \int_0^e \mathrm{P}(z)\, dz$

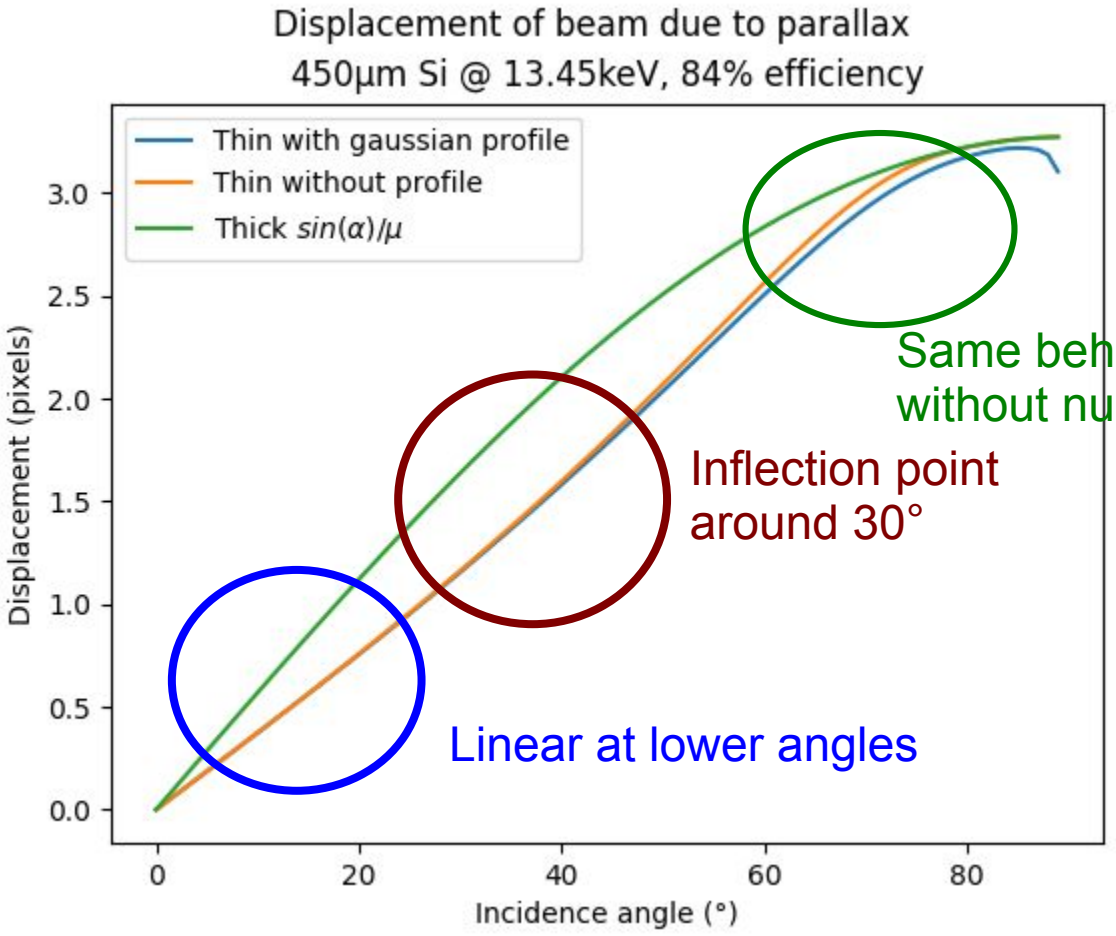
Which leads to: $\mathrm{P}(z) = \mu' \cdot \frac{e^{-\mu' z}}{1 - e^{-\mu' e}}$



After integration ...

$$\langle x \rangle = \frac{\sin(\alpha)}{\mu} \cdot \frac{1 - (1 + \mu' e) e^{-\mu' e}}{1 - e^{-\mu' e}}$$

Comparison of models:



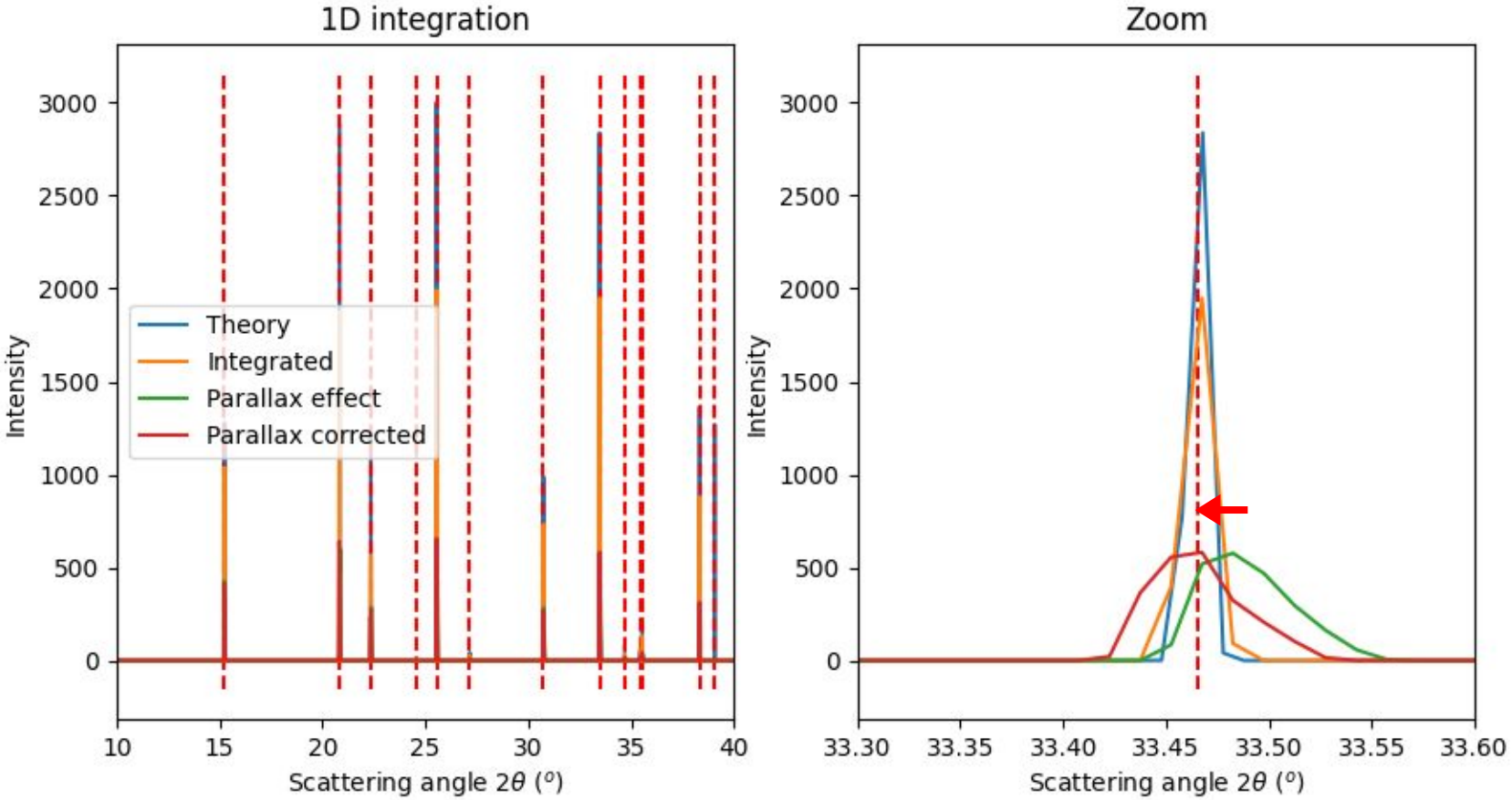
Same behavior at large angle without numerical noise

Inflection point around 30°

Linear at lower angles

Demo time (bis)

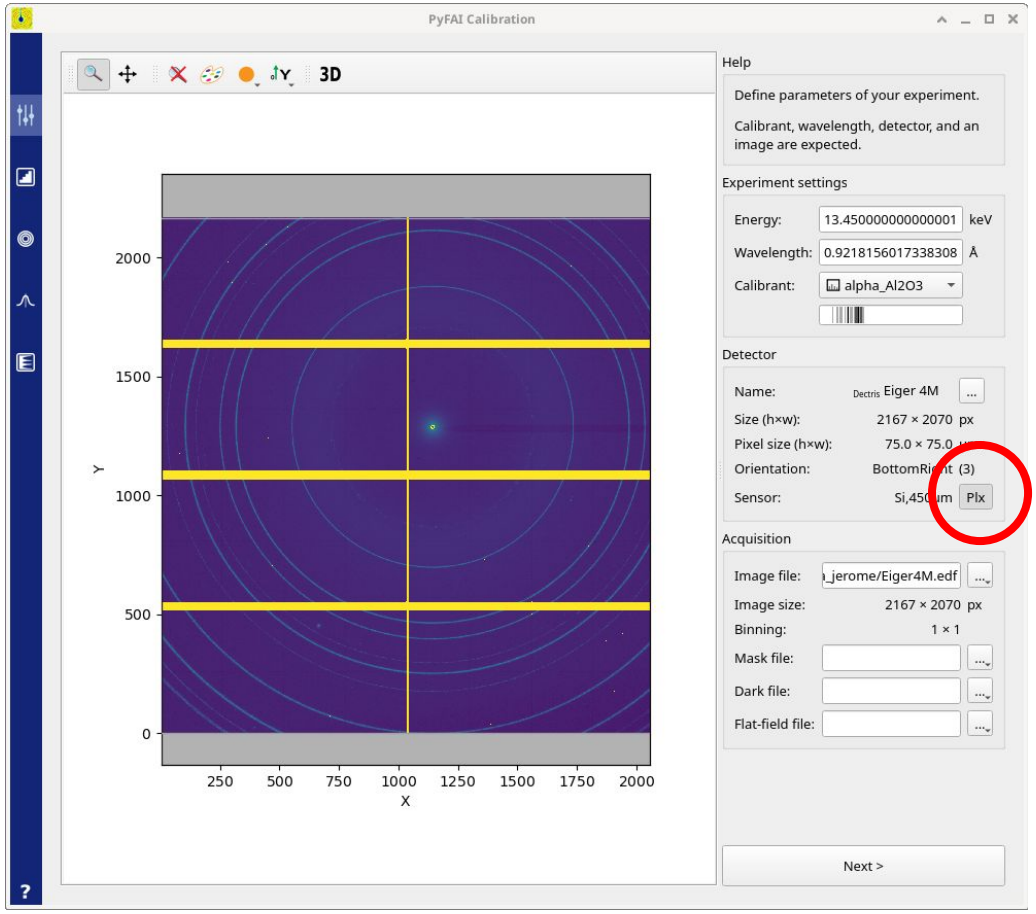
- **Implementation in geometry engine of pyFAI**



Makes the geometry engine of pyFAI much slower for now …
→ rough edges still present & bugs probable !

Change of PONI-file format to cope for parallax corrections

- **With pyFAI v2025.12+ there is a new PONI-file version:**
  - Version 2.1 (when the parallax is disabled, backwards compatible)
  - Version 3.0 (when the parallax is enabled, backwards incompatible)



New button to activate parallax correction

Some limitations:

- Refinement does not always converge better.
- Calculation are slower for now.

Conclusions

- **PyFAI offers several ways to deal with parallax correction:**
  - Calculate the blurring effect via raytracing then de-smear with iterative algorithm (slow)
  - Integrate the effect in the geometry engine
- **For this the sensor composition and thickness has been tabulated.**
  - For all 300 detectors tabulated
  - Thanks to:
    - **Help of all detector manufacturers,**
    - **Detector group at ESRF**
    - **Integration work by Gudrun Lotze.**
- **All this tutorial was on the peak position …**
  … not on their intensity or profile of the peaks.
  - Absorption efficiency correction is already available in pyFAI
    - **but neither addresses the profile modification**

Questions ?