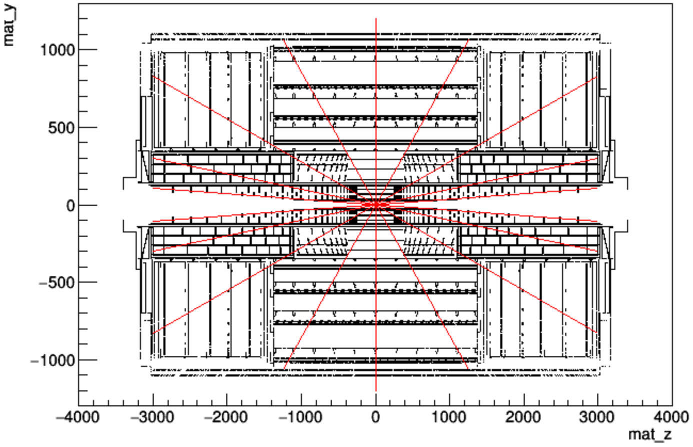
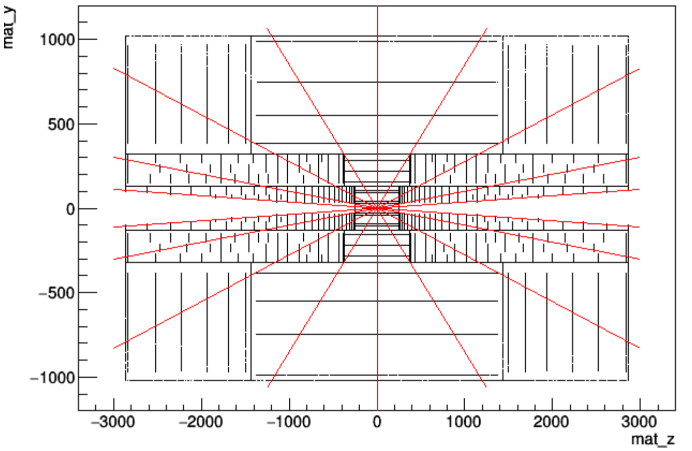
Auto-tuning of the Acts material mapping with Orion

Corentin Allaire

Material Mapping

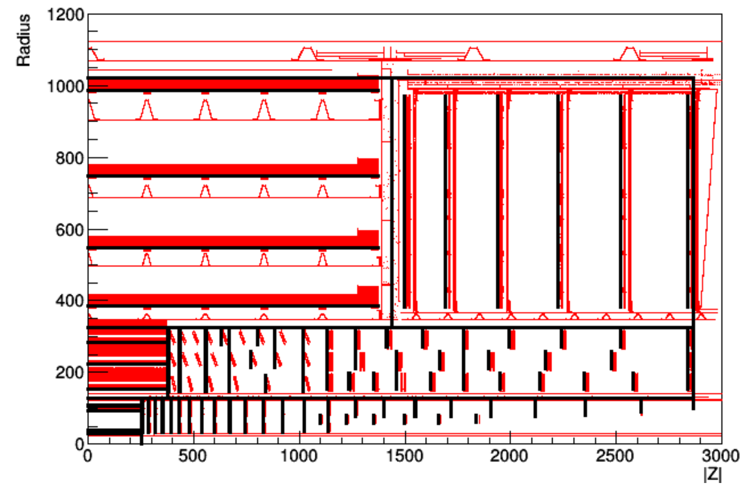
- Particle detector are usually simulated using a Geant 4 model -> Extremely precise but also extremely heavy to use
- When reconstructing trajectories we use a simplify geometry : Tracking Geometry -> Position of the sub-detector, active surfaces (sensor), other large structures (magnets,…)
- Only contain geometric information (where are my different surface) -> Used to navigate through the detector
- Information on which material a particle following a specific path encountered is still needed to properly account for the effect of particle/material interaction
- A simplify model of the material in the detector need to be built -> Material map

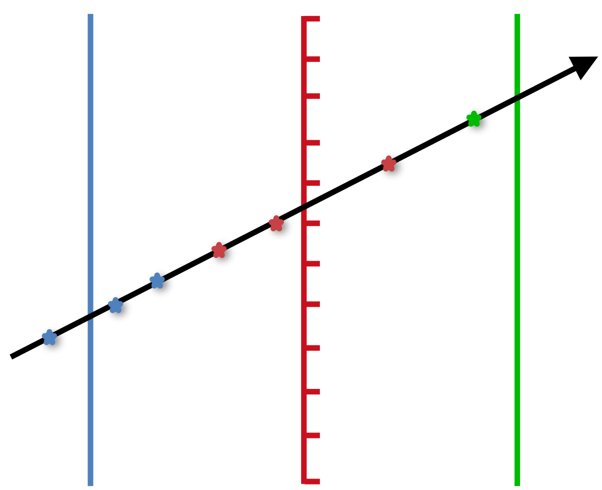




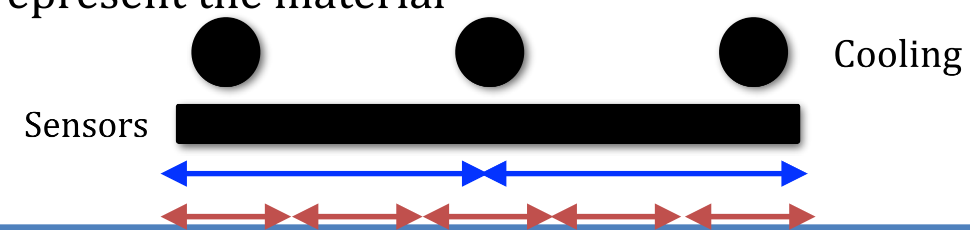
Material Mapping

- To create the map we select a set of surface in our tracking geometry
- We then collect all the materials in our detector with a G4 simulation (using geantino)
- We can then associate each material to the closest surface





- Each of our mapping surfaces is bin along 2 direction (R, Phi), (Z, phi), …
- The material is thus accumulated in each bin, then average for our geantino to form our map
- The correct binning need to be found to properly represent the material



Problematic

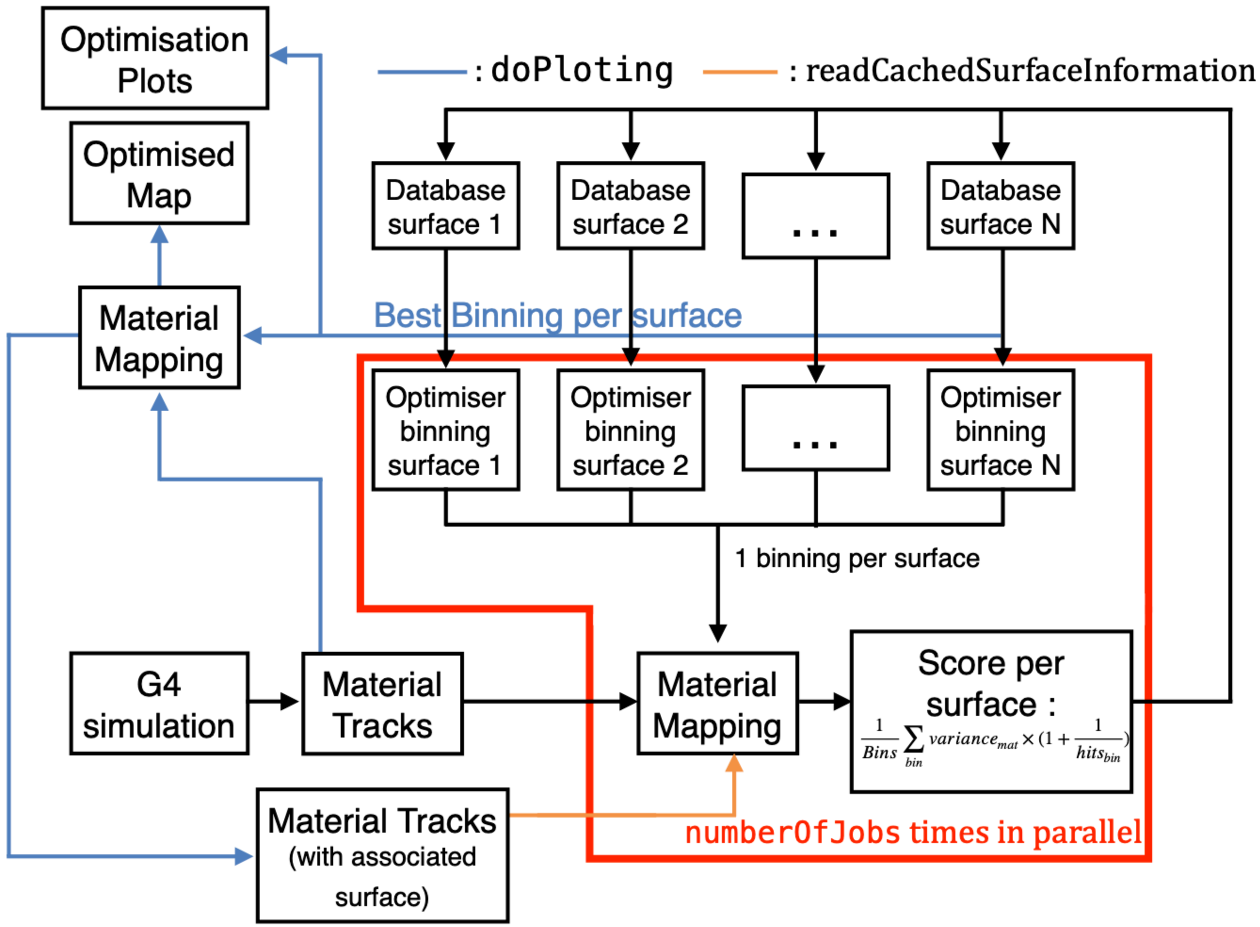
- This material mapping depends on some expertise in both tracking and detector design to best understand how should material
- Time-consuming : each iteration of the mapping optimisation takes a while. Obtaining a good map can take weeks.
- Needs to be redone after each change of detector geometry
- Solution : let an algorithm perform the optimisation -> Auto-tuning of the material mapping
- This should be :
  - Faster (trading CPU for person power)
  - More precise (large fraction f the phase space probed)
  - More reliable/reproducible (lower risk of human error)

Oríon

- Oríon : open-source python asynchronous framework for black-box function optimisation.
- Originally developed to optimise ML algorithm hyper-parameters
- Link : https://orion.readthedocs.io/en/stable/index.html
- Easy to integrate with any python code (Acts has a python based job system)
- Works in parallel thanks to its database system
- Implement many optimisation algorithms :
  - **Random Search**
  - **Grid Search**
  - **Hyperband**
  - **ASHA:** Asynchronous Successive Halving Algorithm
  - **TPE:** Tree-structured Parzen Estimator
  - **Evolution-ES**
  - **Scikit-Optimize:** providing a wrapper for *skopt* optimizers, e.g Scikit Bayesian Optimizer
  - **Robust Bayesian Optimization:** providing a wrapper for *RoBO* optimizers, e.g: Gaussian Process, Gaussian Process with MCMC, Random Forest, DNGO and BOHAMIANN.

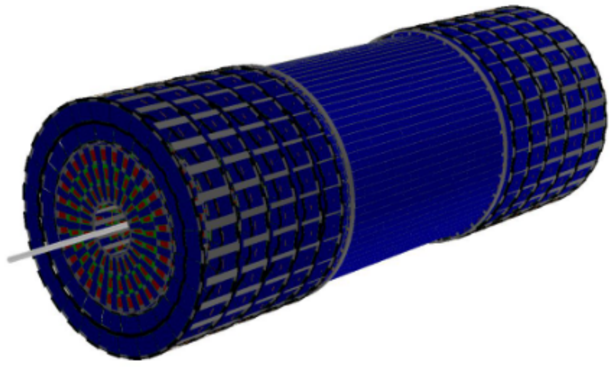
Auto-tuning of the material mapping

- One optimiser instance per surface we want to map onto
- Combined together to parametrise one mapping job
- N mapping running in parallel
- After the N finishes update the optimiser : score for each binning (variance in each bin)
- After enough iteration : get the best binning per surface -> run one last mapping



Optimisation for the Open Data Detector (ODD)

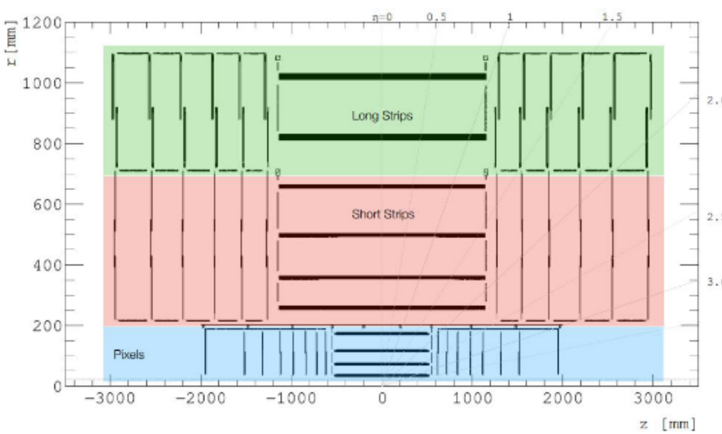
- Input :
  - 10 000 000 tracks
  - Bin range for each surface between 1 and 240
  - ~6000 trials
  - ~ 3-4 days of running on 40 cores



- Binning : [Phi-R] for disk (end-caps)
  or [Phi-Z] for cylinder (barrel)

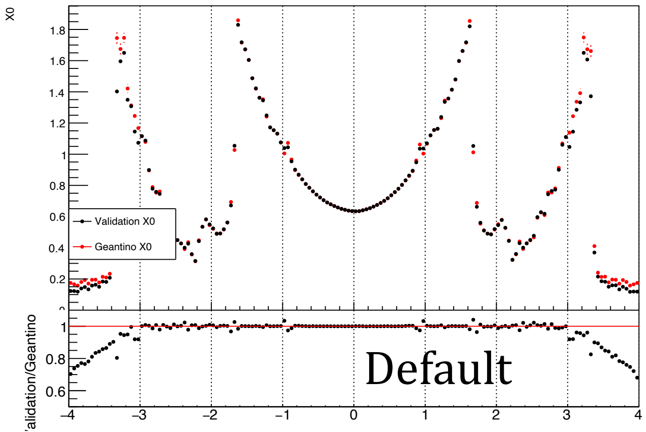
- We use the same surfaces for the mapping as the default map for the ODD

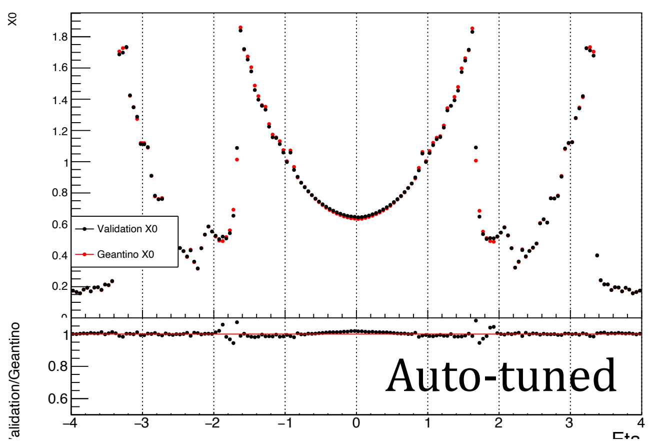
- Can we achieve map of the same quality as the optimised one ?



Result of this optimisation

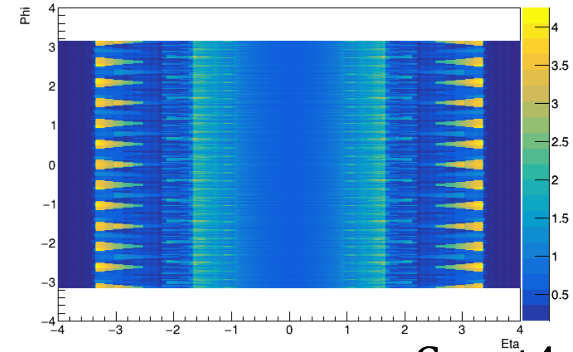
- Comparison of the amount of material encountered by a track as function of eta for the navigation with map and Geant4 scan
- In the default case we have a good agreement except in the endcap (this could easily be fixed)
- The auto-tuning perform similarly well over the entire eta range
- A small issue in the barrel region : the ODD ShortStrips::Barrel seem to favour 1 Z bin -> not sure why yet
- The optimisation doesn't help you select which surface to map onto but find the best binning



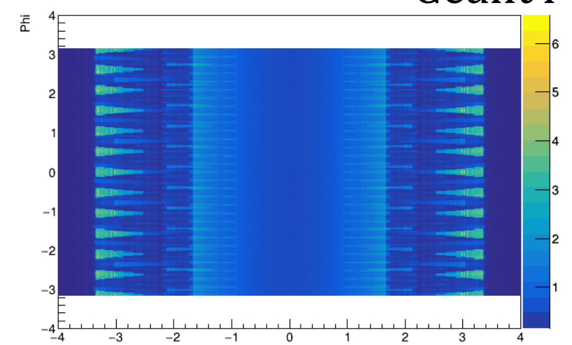


Result of this optimisation

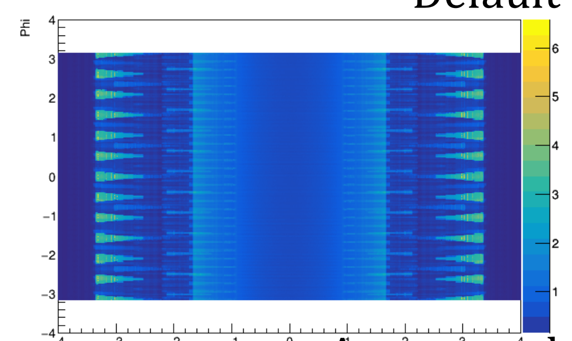
- Amount of material in the detector as function of eta and phi
- Same pattern in the 3 cases -> good binning
- The algorithm tend to select large amount of bin due to the definition of the score -> some refining might be needed
- Looking into new scoring function that would perform better



Geant4



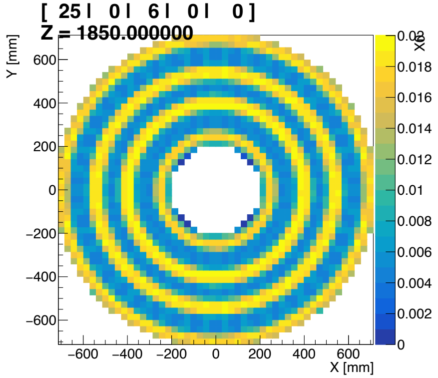
Default

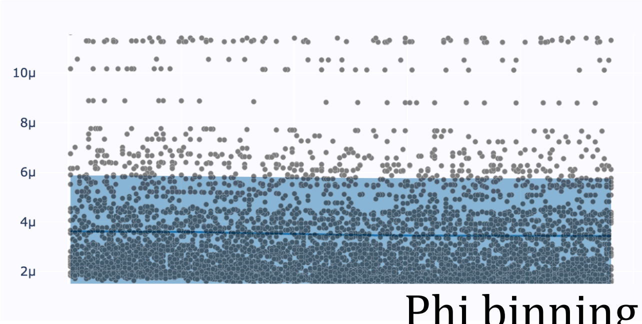


Auto-tuned

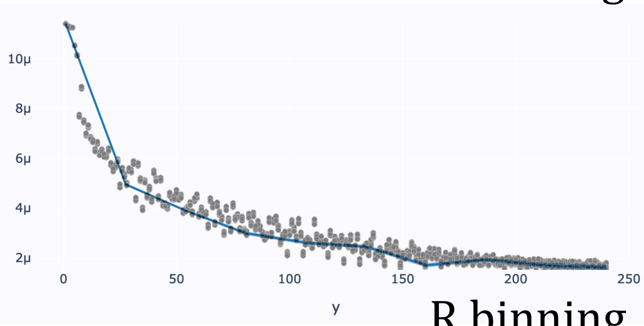
Result for a specific disk surface

- Orion provides many performance plot that show
- Here for example looking at one of the endcap disk the material is only R dependent
- Score as function of the binning choice in both dimension
- No dependency in Phi -> Score might need to be change to favor small number of bin
- Strong dependency is R -> more bin better precision



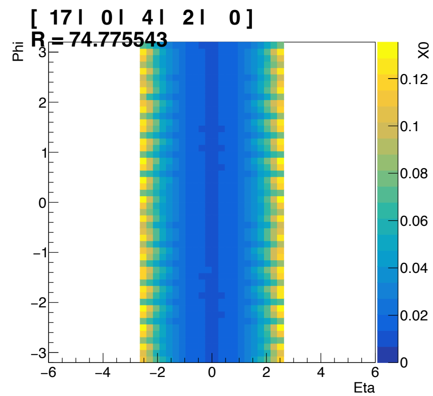


Phi binning



R binning

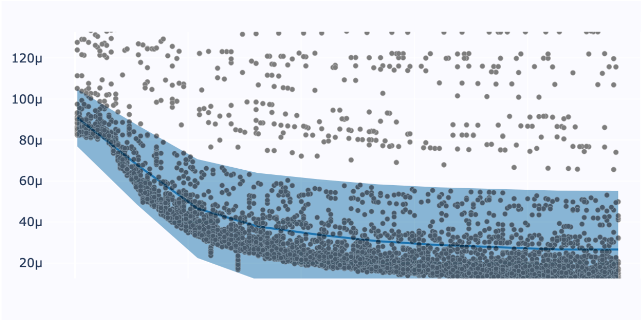
Result for a specific cylinder surface



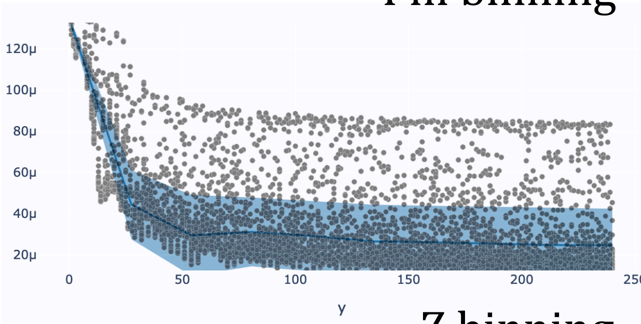
- Same for one of the barrel

- Here we have a dependency in both Phi and Z

- Binning plateau reached for Phi and Z followed by a slow decrease -> Change of the scoring should favour the binning reaching the plateau



Phi binning



Z binning

Conclusion and next steps

- We have implemented an automatic tuning algorithm for the material mapping
- Require user input to choose the surfaces used in the mapping but optimise the binning for those surfaces
- Tomorrow, tutorial on how to use this : https://acts.readthedocs.io/en/latest/howto/run_material_mapping_auto_tuning.html#material-validation
- Perform relatively well but more work is still needed :
- New scoring function should be explored
- Currently, just using random search with large amount of trial to check the scoring and the approach but more advance search algorithm will need to be tested

BACKUP