Status of collimation tracking code development for FCC-ee

A. Abramov, R. Bruce, F. Carlier, M. Hofer, L. Nevay, S. Redaelli

FCCIS WP2 Workshop - 01/12/2021

Many thanks for discussions and input to:

M. Boscolo, H. Burkhard, T. Charles, M. Hofer, G. Iadarola, M. Migliorati, J. Molson, M. Moudgalya, T. Persson, T. Pieloni, G. Roy, L. Van Riesen-Haupt, S. White, F. Zimmermann

Simulation tools

- **Tracking studies are essential for designing a collimation system**
- **Effects such as synchrotron radiation and optics tapering make the tracking studies more challenging for the FCC-ee**
- **The requirements for collimation simulation tools are:**
  - **Tracking of beam electrons (and positrons) in the magnetic lattice**
  - **Particle-matter interactions inside the collimators**
  - **Accurate and efficient tracking over many turns**
  - **Radiation damping and optics tapering**
  - **Beam-beam effects**
- **Studied several different simulation tools:**
  - MAD-X, SixTrack-FLUKA coupling, BDSIM, Merlin++, pyAT, Xtrack
  - Alternatives not studied so far: SAD, BMAD, ELEGANT, MDISim, others…

Simulation tools

CHART
Swiss Accelerator Research and Technology

- **No framework that currently has the requirements for collimation**
- **There is an EPFL-CERN collaboration (F. Carlier, T. Pieloni) to develop a comprehensive beam dynamics simulation framework for the FCC-ee (talk F. Carlier, talk T. Pieloni)**
- **Collimation developments focus on a connection between a tracker and a physics engine**

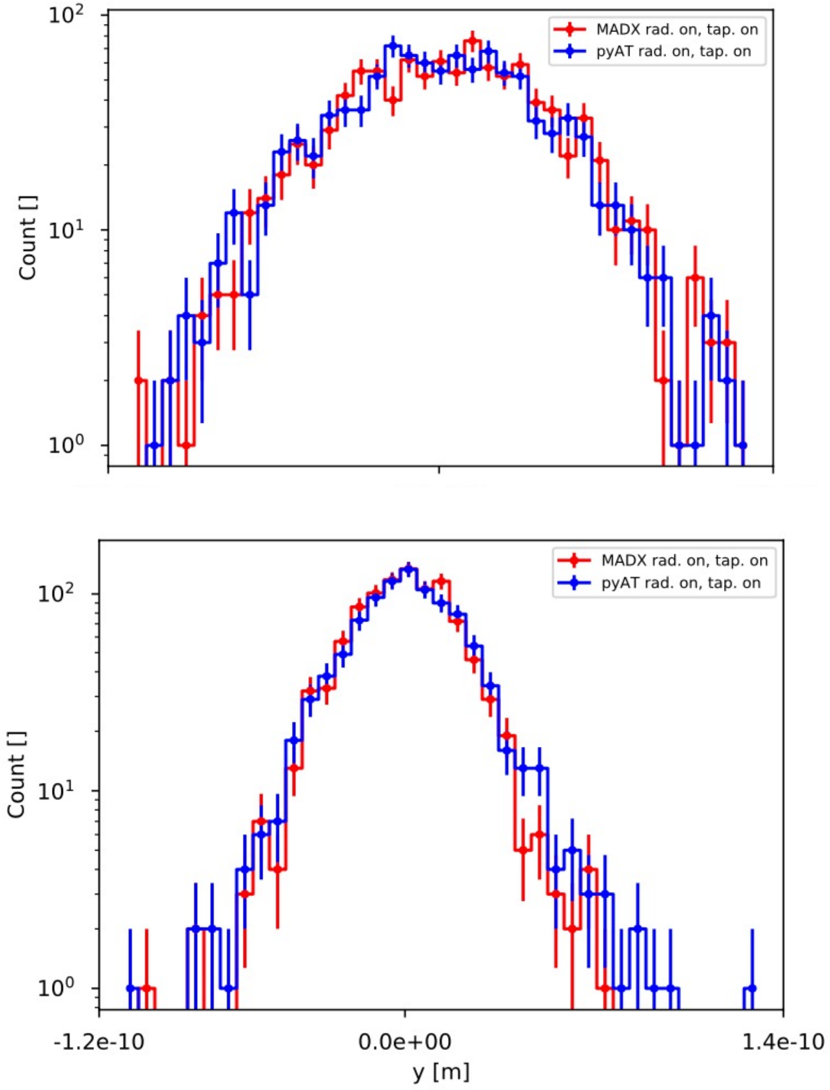
Code	6D symplectic tracking	Synchrotron radiation	Tapering (lattice)	Geant4 integration	FLUKA integration	Speed	Aperture modelling
MAD-X							
SixTrack							
BDSIM							
Merlin++							
pyAT				★			
xTrack		★		★			

Not available Partial availability Available ★ = in development

pyAT

https://github.com/atcollab/at

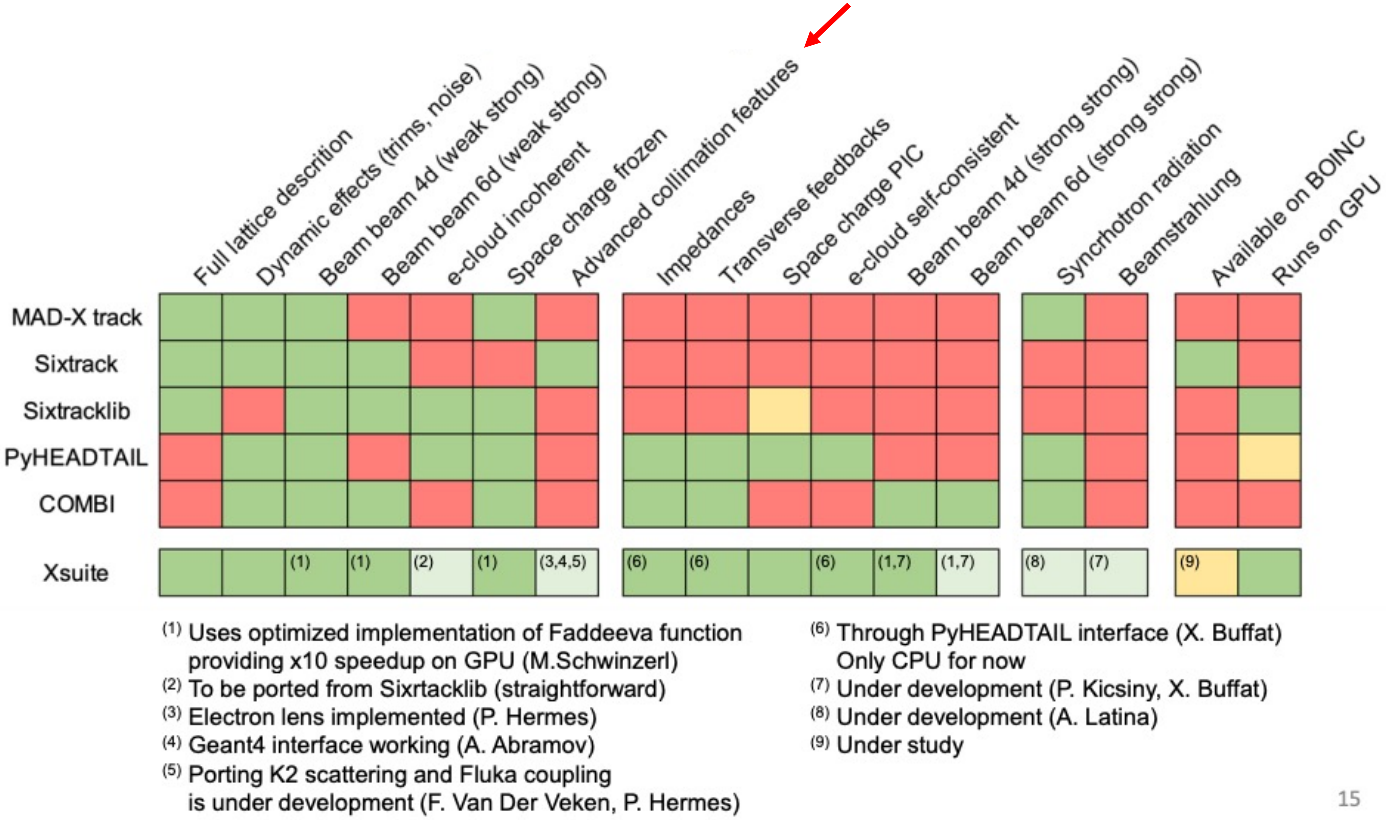
- **PyAT is a promising tracking tool for the FCC-ee**
  - First used for FCC-ee by M. Rakic at EPFL
  - Demonstrated tracking with radiation and tapering and an acceptable computational efficiency
  - Straightforward to work with
- **Latest developments**
  - An option for per-element quantum excitation (S. White)
  - Loss map features enabled – apertures and loss recording
  - New features in **MAD-X** -> **pyAT** converter (F. Carlier)
    - Apertures translated
    - Collimator lattice element introduced
- **Developments for collimation**
  - Need to include scattering in the collimators
  - Currently working on a connection to Geant4 for collimators



Distribution of 10^3 particles after 10^3 turns, ideal Higgs lattice (120 GeV)

Xtrack

- **New tracking tool, part of the Xsuite project (talk by G. Iadarola)** https://xsuite.readthedocs.io/en/latest/
  - Developed by R. De Maria and G. Iadarola
  - Designed to consolidate multiple beam dynamics studies at CERN
  - **Python** on the top level, auto-generated C and extensions for performance-critical tasks
  - Multi-platform, compatible with **CPU** and **GPU**
  - Rapid development
  - Documentation, tests
  - Working with G. Iadarola on an interface to Geant4 for collimation
  - Planned to also interface with K2 and FLUKA



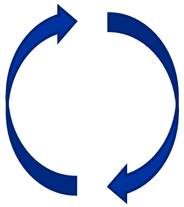
G. Iadarola https://indico.cern.ch/event/1070294/contributions/4500628/attachments/2305233/3921762/021_Xsuite.pdf

Collimation in Geant4 – general principle

example lattice section

- **Only collimators in the Geant4 world, with particle transfer mechanisms (like the SixTrack-FLUKA coupling)**

- **Same general workflow, regardless of the tracking code connected:**
  1. Define all collimators
  2. Select a collimator and add particles
  3. Run the physical interaction simulation
  4. Return eligible particles

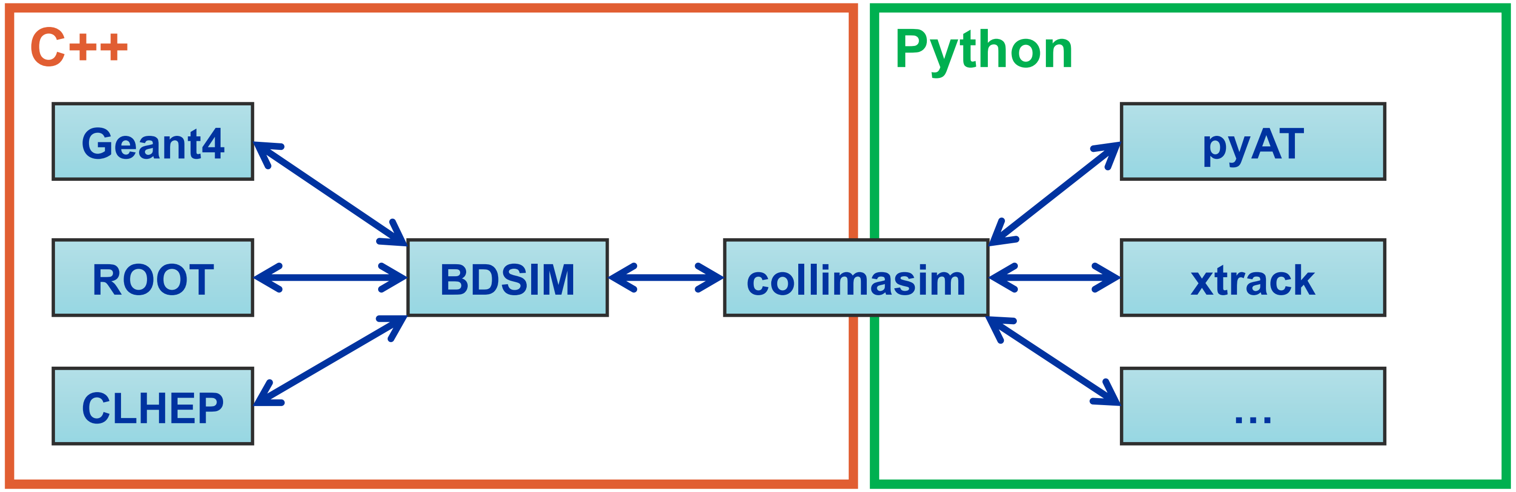


Steps 2 – 4 are repeated for every collimator pass

- **Need to define interfaces for communication between the codes involved**

Python bindings for Geant4

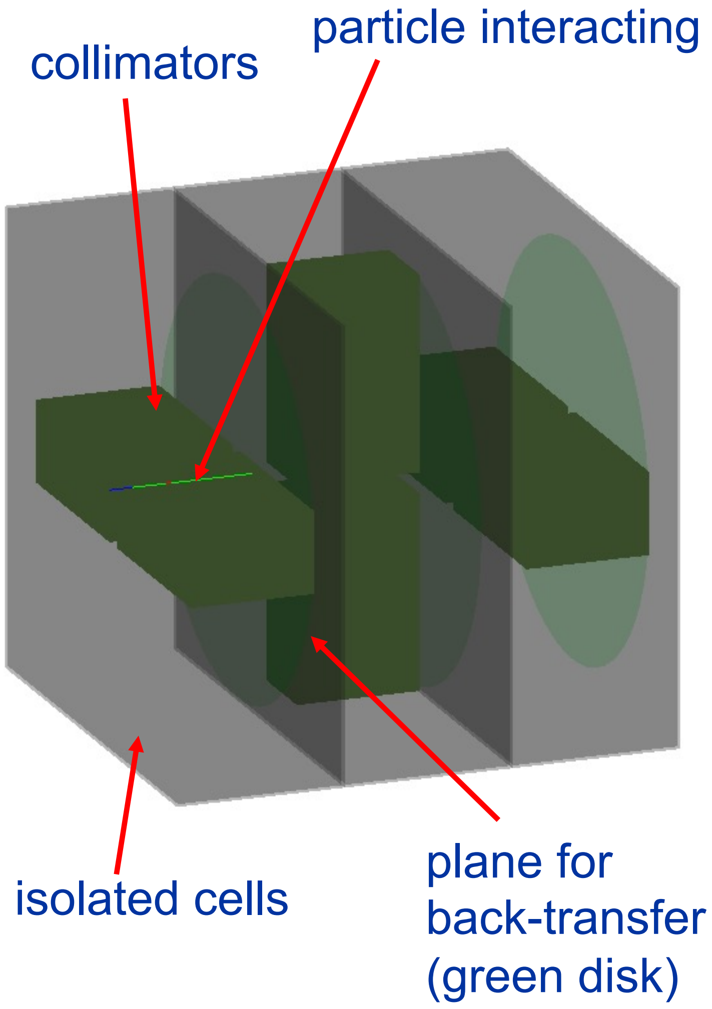
- **Existing general interface to Geant4 (BDSIM) for collimation (L. Nevay).**
- **Build upon the existing code and add a Python wrapper**
  - Dedicated C++ classes for interfacing with tracking codes
  - Python bindings via pybind11
  - Direct and dynamic communication at runtime
- **The result is collimasim – a Python package running Geant4.**
  https://gitlab.cern.ch/anabramo/collimasim



Integration with pyAT

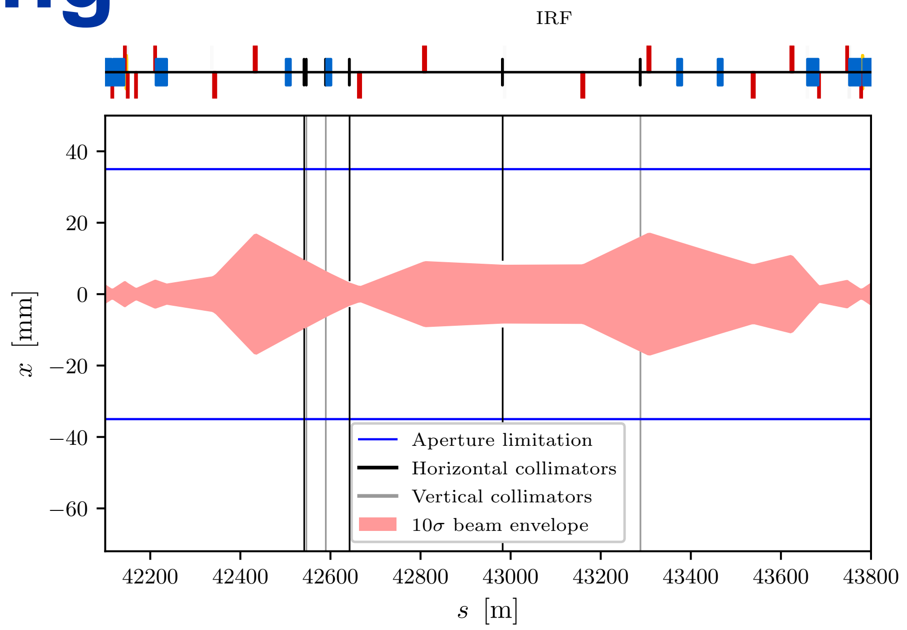
- **Implemented a connection with pyAT**
  - Custom integrator (pass method) prepared for pyAT
  - Workflow developed to prepare collimation studies
- **Current limitations**
  - Secondary particles are not tracked
  - Only rectangular block jaw collimators supported

```python

# Load the pyAT lattice as normal, the name is 'ring'

# Initialise the BDSIM interface
pyCollimatorPass.initialise(bdsim_config_file="settings.gmad",
                            collimator_file="collgaps.dat",
                            reference_pdg_id=-11,
                            reference_Ek=182.4994890018054,
                            relative_energy_cut=0.3,
                            seed=1991)

# Change the pass method of loaded collimators
coll_idc = at.get_refpts(ring, at.elements.Collimator)

for ci in coll_idc:
    ring[ci].PassMethod='pyCollimatorPass'
```



Testing the pyAT-Geant4 coupling

- **Use the collimation setup from FCC week 2021:**
  - ttbar mode (182.5 GeV), beam 1 (positron)
  - Two-stage collimation system in IRF
  - 1 µm impact parameter pencil beam
  - 1 sigma beam in the vertical, nominal vertical emittance
  - 30 GeV energy cut
  - **No radiation and tapering**
  - **Not the latest optics and aperture model**
- **Compare against the SixTrack-FLUKA coupling**
  - The same beam distribution is used for both cases
  - 5e6 particles in SixTrack and 1e6 particles in pyAT



Halo collimation system in IRF

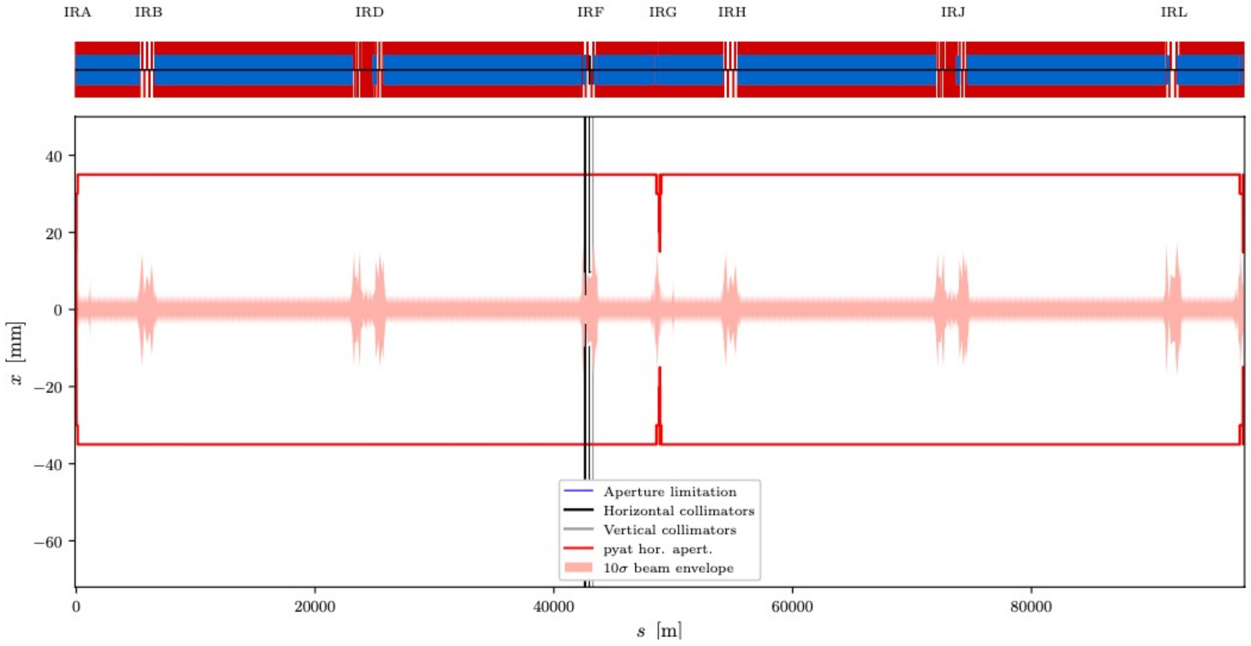
Collimator	Type	Plane	Opening [σ]
TCP.A.B1	Prim.	H	10
TCP.B.B1	Prim.	V	80
TCS.B1.B1	Sec.	V	89.5
TCS.A1.B1	Sec.	H	11.5
TCS.A2.B1	Sec.	H	11.5
TCS.B2.B1	Sec.	V	89.5

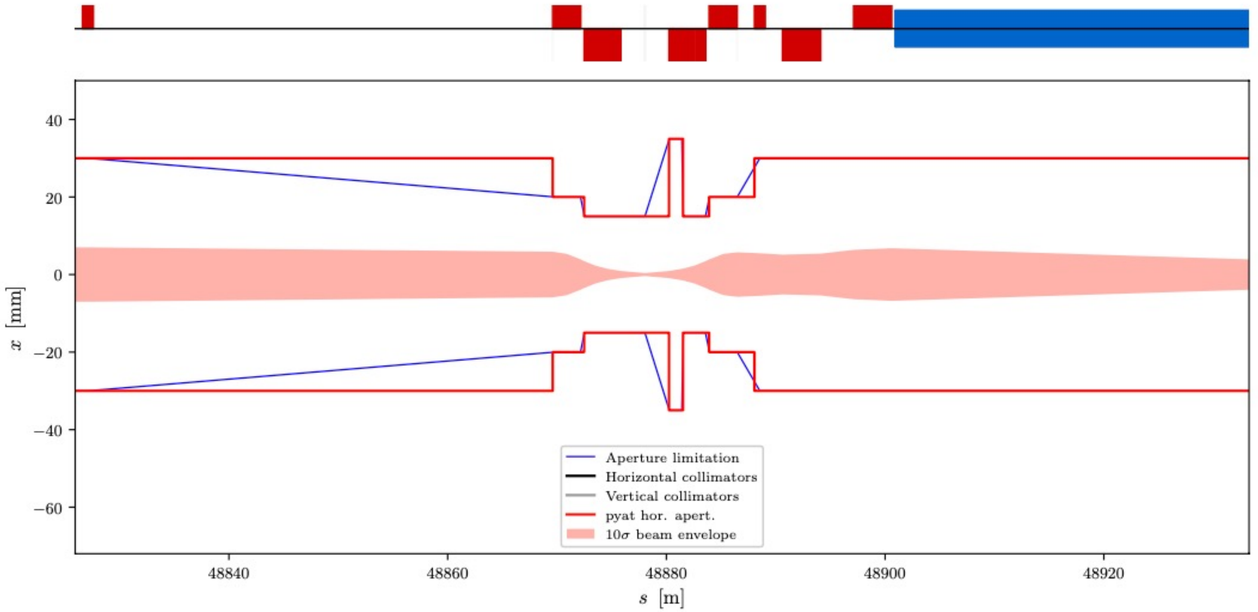
Collimator settings

FCC week talk:
https://indico.cern.ch/event/995850/contributions/4410113/

Aperture definition

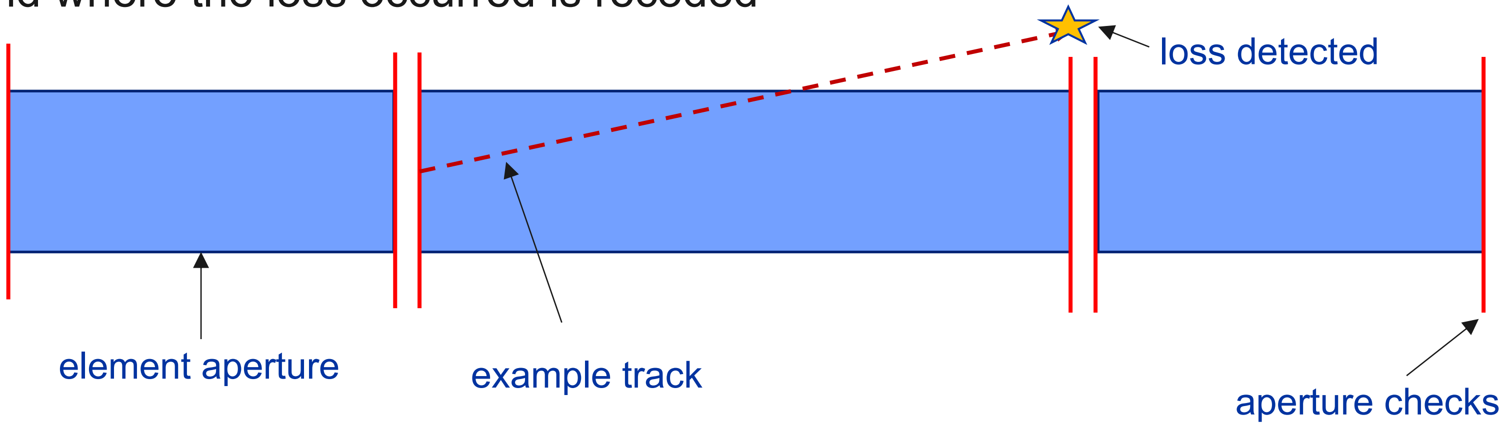
- **Aperture model:**
  - FCC week (older) aperture model (M. Moudgalya)
  - 35 mm circular aperture in the arcs
  - 15 mm experimental beampipe, no SR masks
- **pyAT aperture**
  - Translated from MAD-X by F. Carlier’s converter.
  - Nearest neighbour interpolation for all elements.
- **SixTrack aperture:**
  - Exported from MAD-X.
  - Linear interpolation in 10 cm intervals.



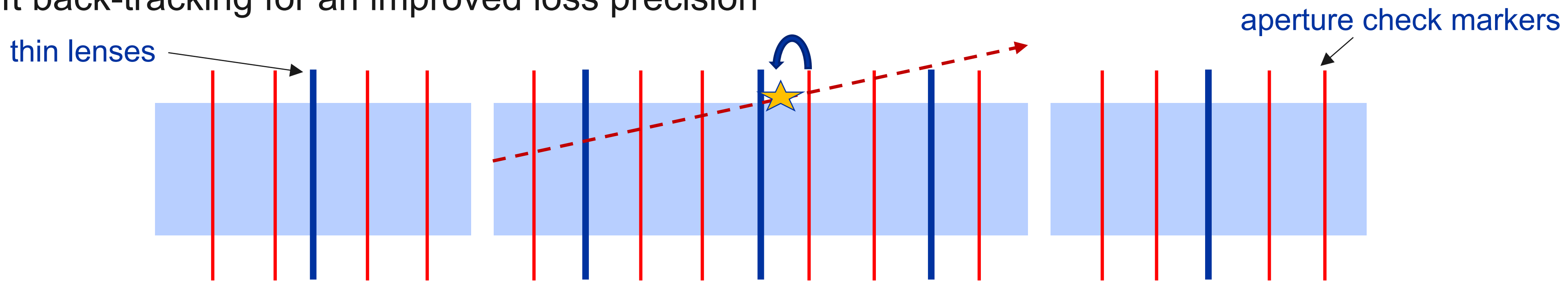


Aperture definition caveats

- **pyAT**
  - Thick elements in lattice – splitting only during integration
  - The aperture is defined as an element attribute and checked at entrance and exit
  - Only element id where the loss occurred is recoded

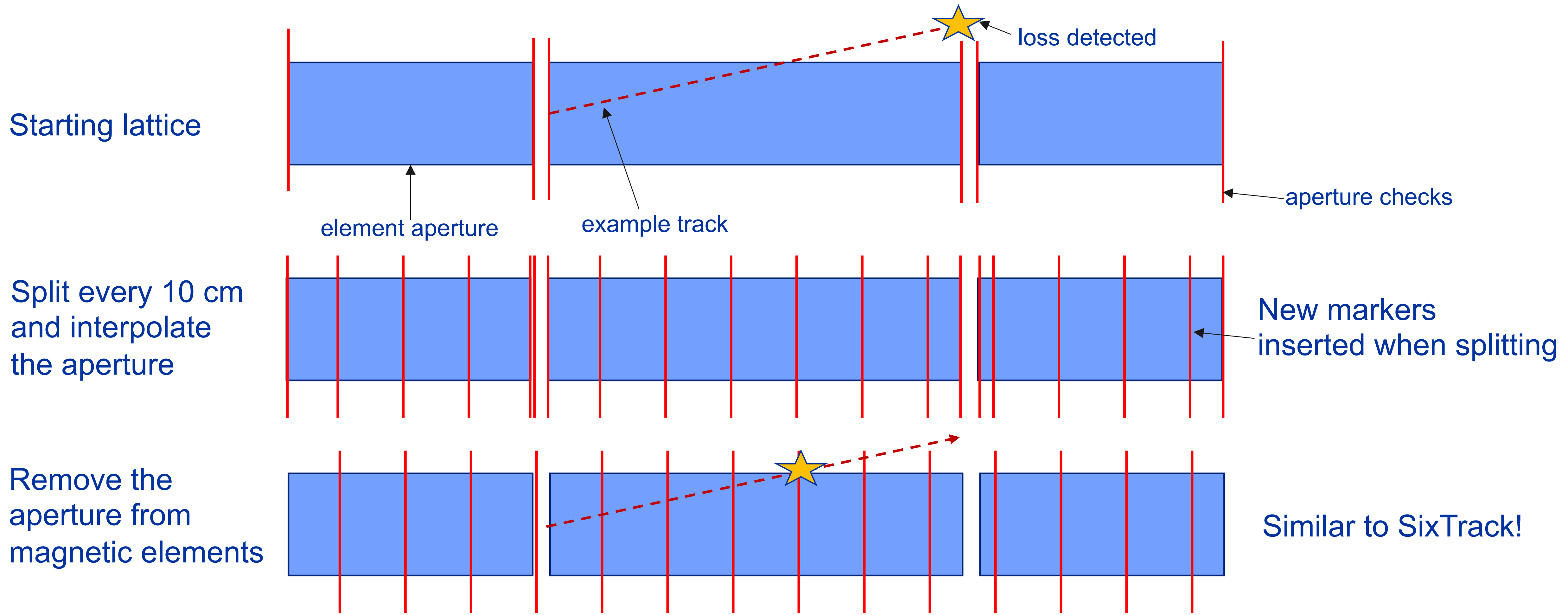


- **SixTrack**
  - Thin lenses and dedicated aperture markers in the lattice
  - Aperture can be interpolated – additional markers (usually every 10 cm)
  - Drift back-tracking for an improved loss precision



New pyAT aperture definition

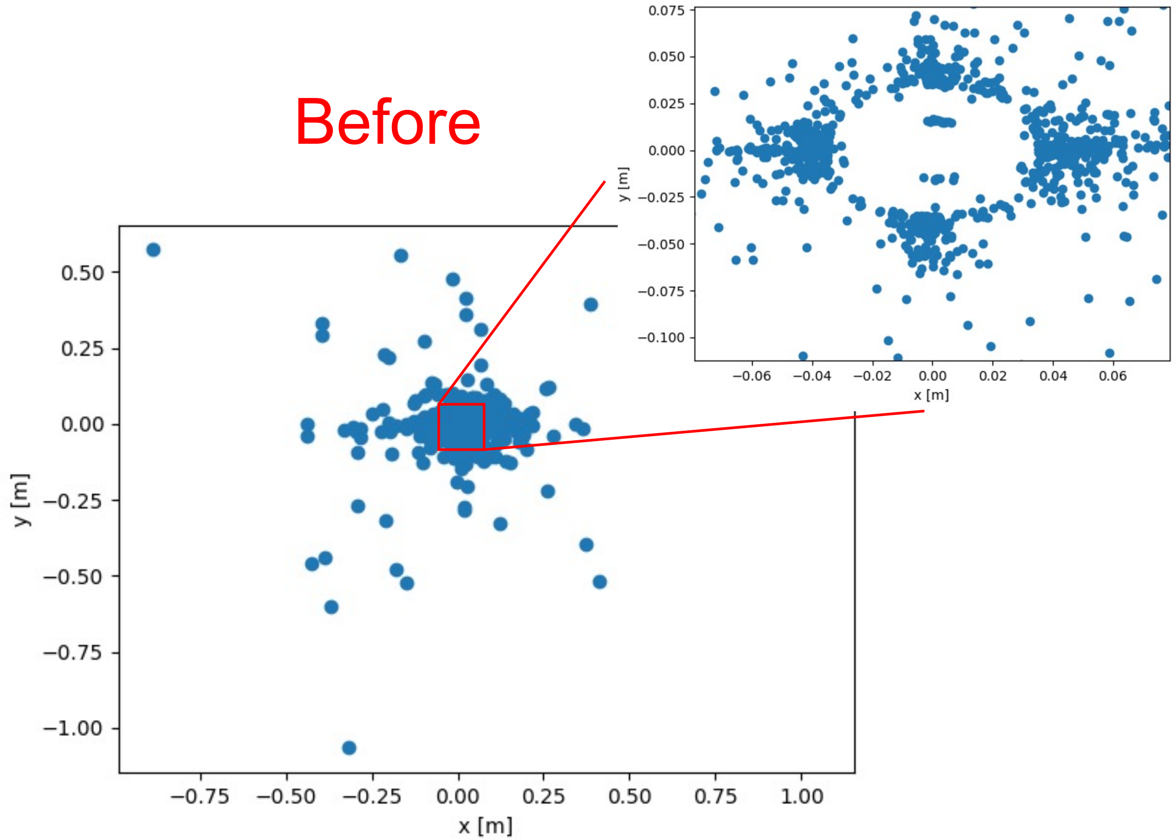
- New method for splitting the lattice elements proposed by S. White
- After a few manipulations, the aperture model is closer to SixTrack
- Only nearest neighbour interpolation and no backtracking



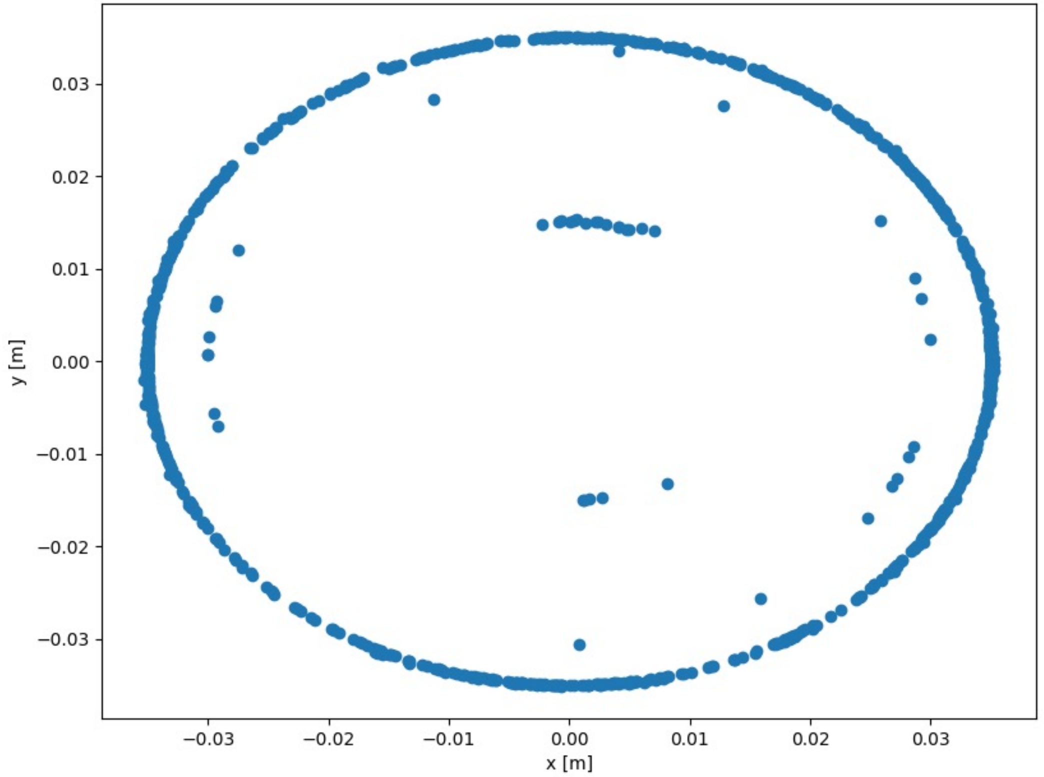
New pyAT aperture definition

- The new aperture definition gives a much better resolution of the losses
- The loss maps are directly comparable with ones from SixTrack
- The tracking speed is severly impacted (factor 10 slower), but it is not a showstopper

Before



Now

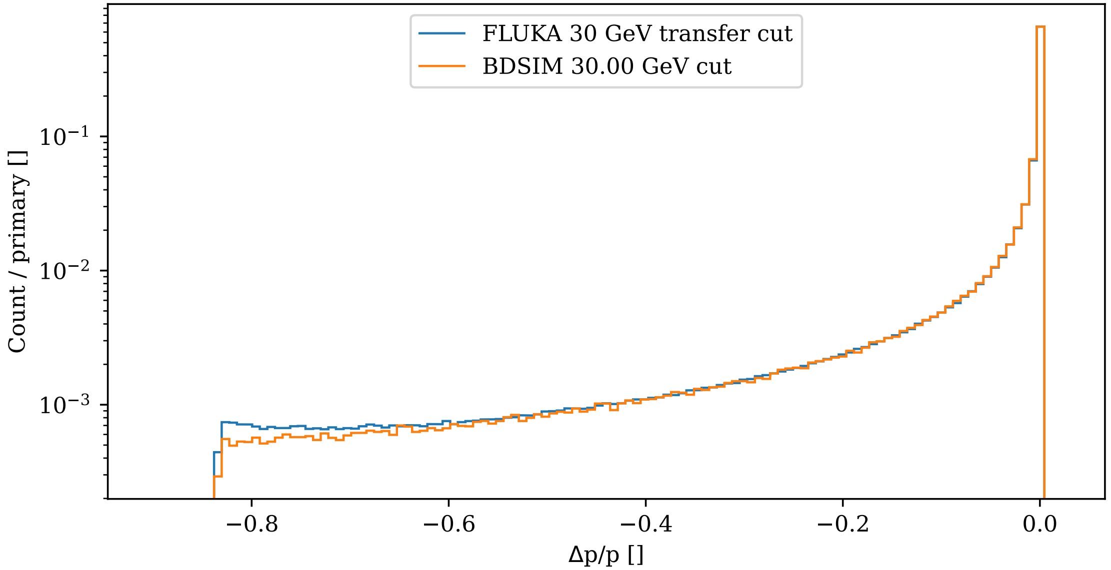


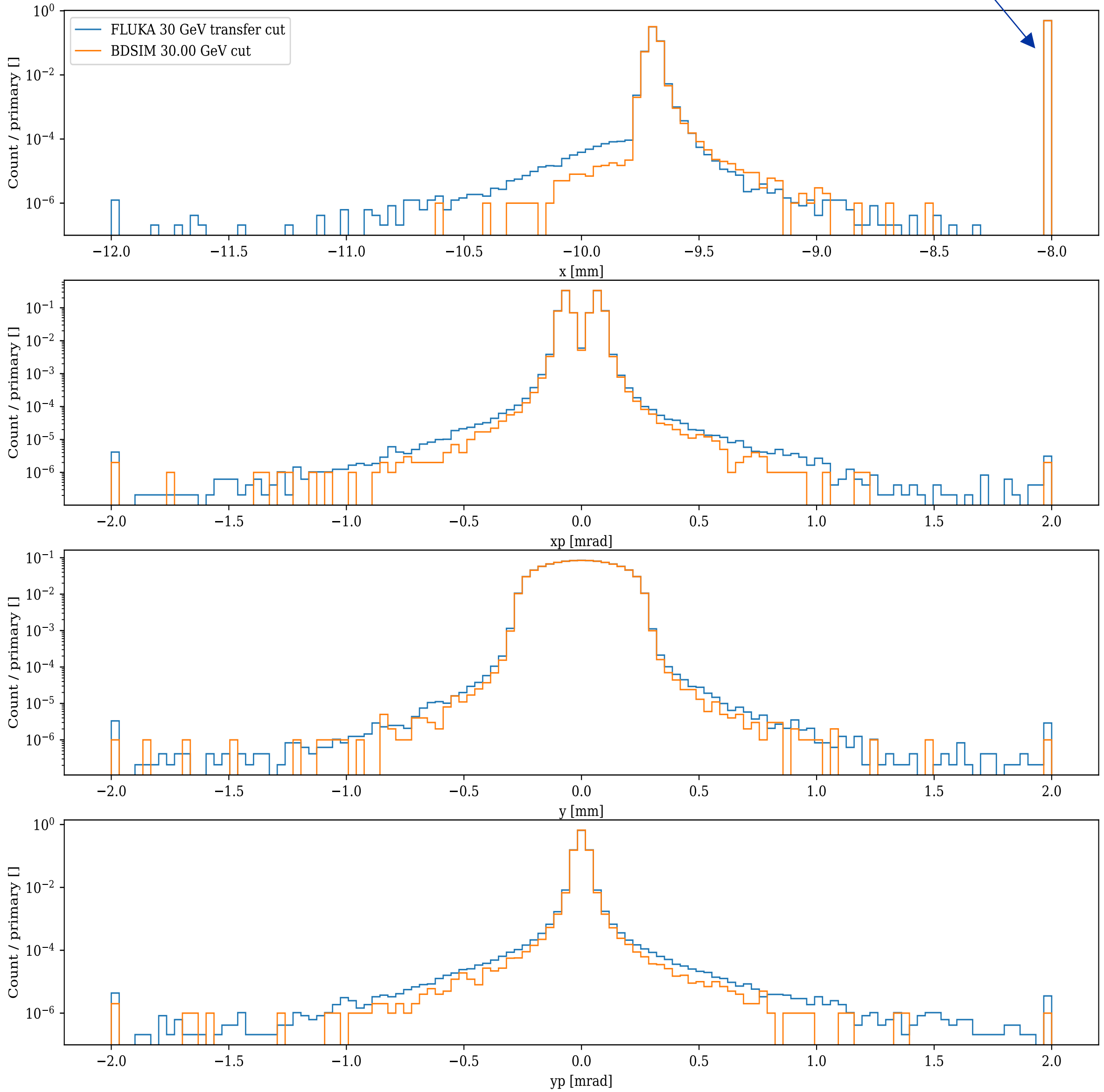
Transverse loss coordinates along the whole ring for 10,000 particles

Collimator leakage comparison

*second jaw in overflow bin to improve visibility

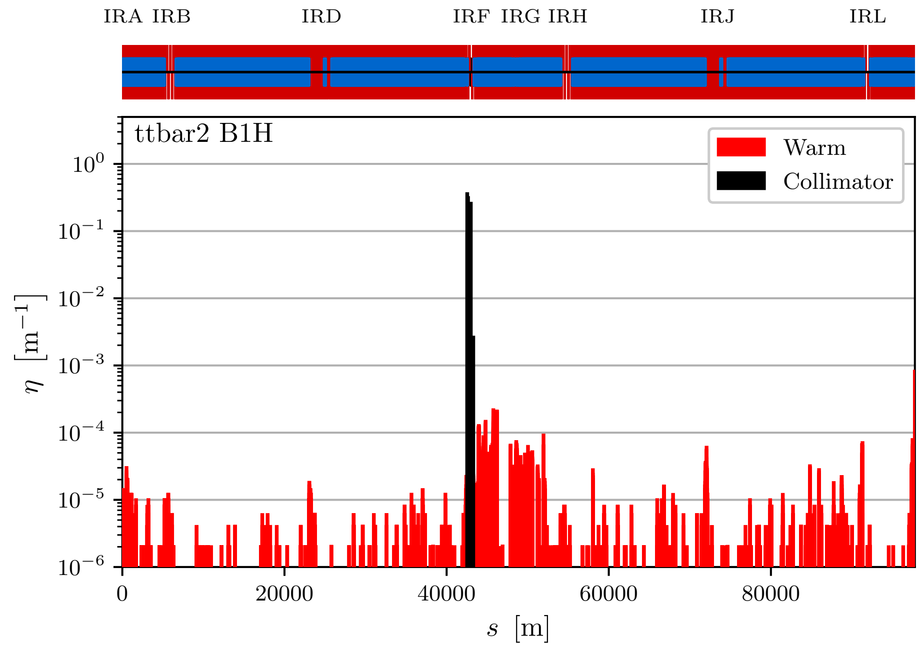
- Spectra of primary e+ and secondary e+/e- compared (<u>no other secondary particles</u>)
- The leakage from BDSIM has no secondary e+/e- as pyAT can't track them
- Found good agreement between FLUKA and BDSIM for the 30 GeV cut used

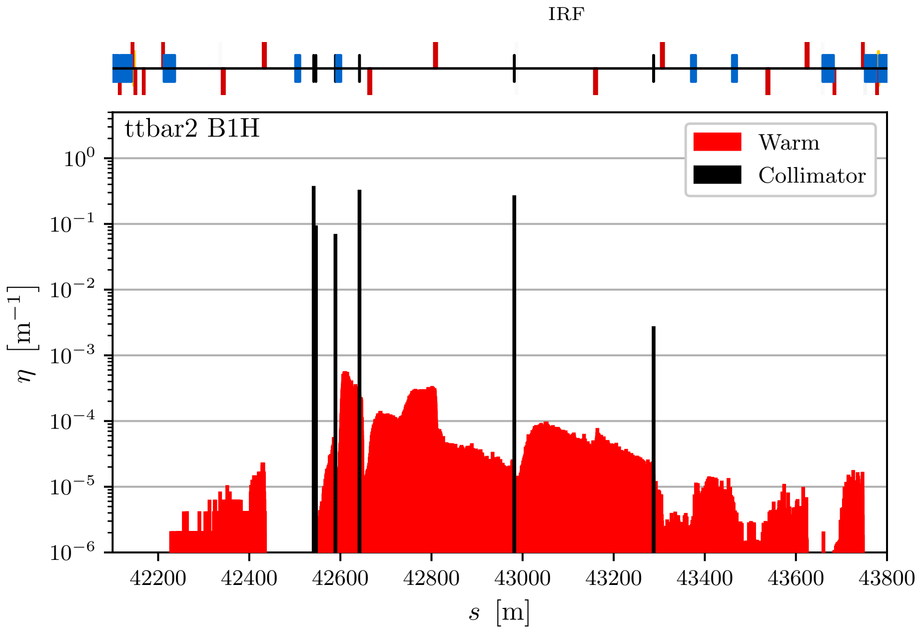




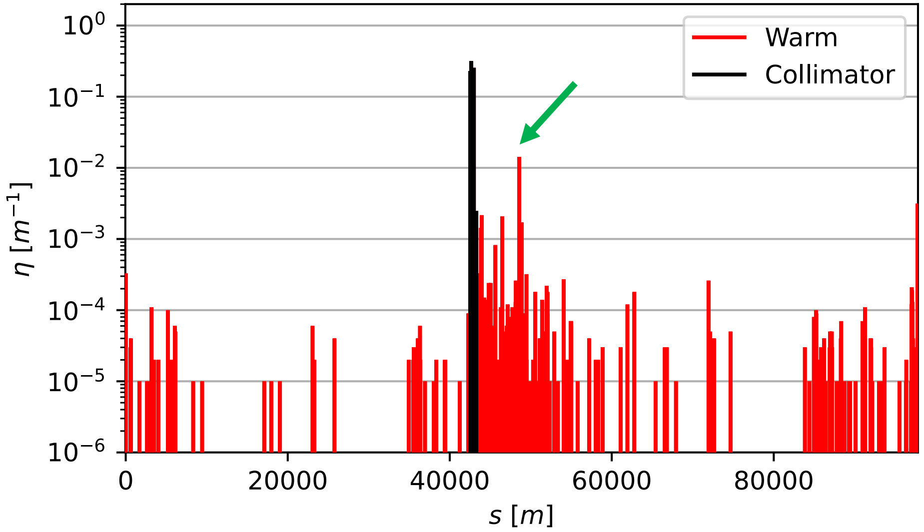
Loss map comparison

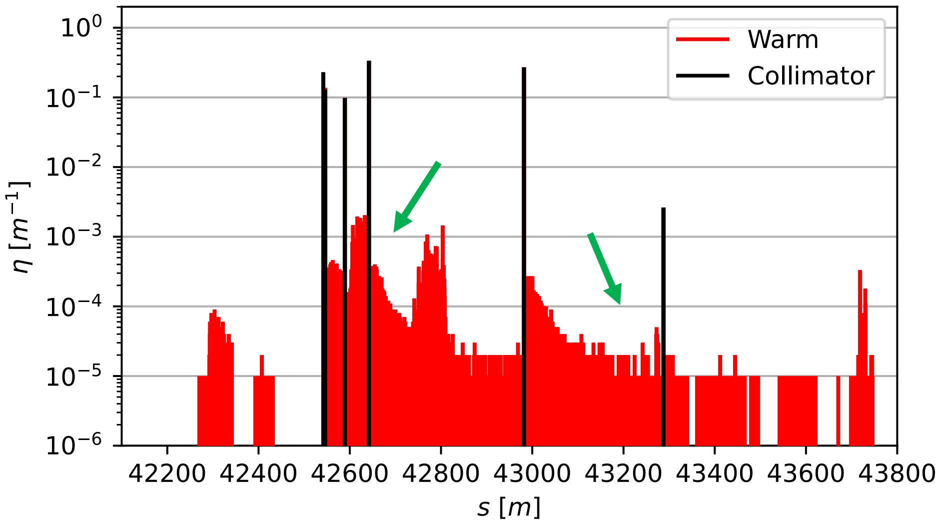
SixTrack-FLUKA





pyAT-Geant4





Loss map comparison

- **The agreement between pyAT-Geant4 and SixTrack-FLUKA is significantly improved after the latest developments**
- **Discrepancies still remain**
  - Large losses are observed in pyAT around IPG, which are not seen in SixTrack
  - The losses around the ring are slightly higher in pyAT
- **To check:**
  - Investigate the effects of the different aperture interpolation
  - Compare tracking and optical functions for the current model
  - Check the loss map normalization in pyAT
- **Tracking studies with radiation and tapering to follow**

Summary

- **Development of collimation simulations**
  - pyAT and Xtrack were selected as the most promising tracking codes
  - Status of pyAT collimation developments
    - A Python interface to BDSIM (Geant4) for particle-matter interactions inside the collimators was developed, to enable multi-turn tracking studies
    - Testing and validation with comparisons to SixTrack-FLUKA
  - Status of Xtrack collimation developments:
    - Interface to Geant4 implemented, similar to pyAT
    - Benchmarks being set up now
    - Other developments in ABP, such as synchrotron radiation

- **Future work for the simulations**
  - Discussions between ABP collimation team and FLUKA team to develop a FLUKA integration with Xtrack and pyAT for collimation
  - Integrate the collimation models with other studies, such as MDI-related activities
  - Include errors, imperfections, and misalignments into the collimation models
  - Investigate the options for including beam-beam effects, including Beamstrahlung (discussions ABP, EPFL, MDI team)

Thank you!

Collimator leakage comparison

*second jaw in overflow bin to improve visibility

- Spectra of primary e+ and secondary e+/e- compared (no other particles)
- Secondary e+/e- now included in BDSIM for comparison, although not tracked in pyAT