Introducing Xcoll:

A Streamlined Approach to Collimation and Beam Loss Simulations Using Xsuite

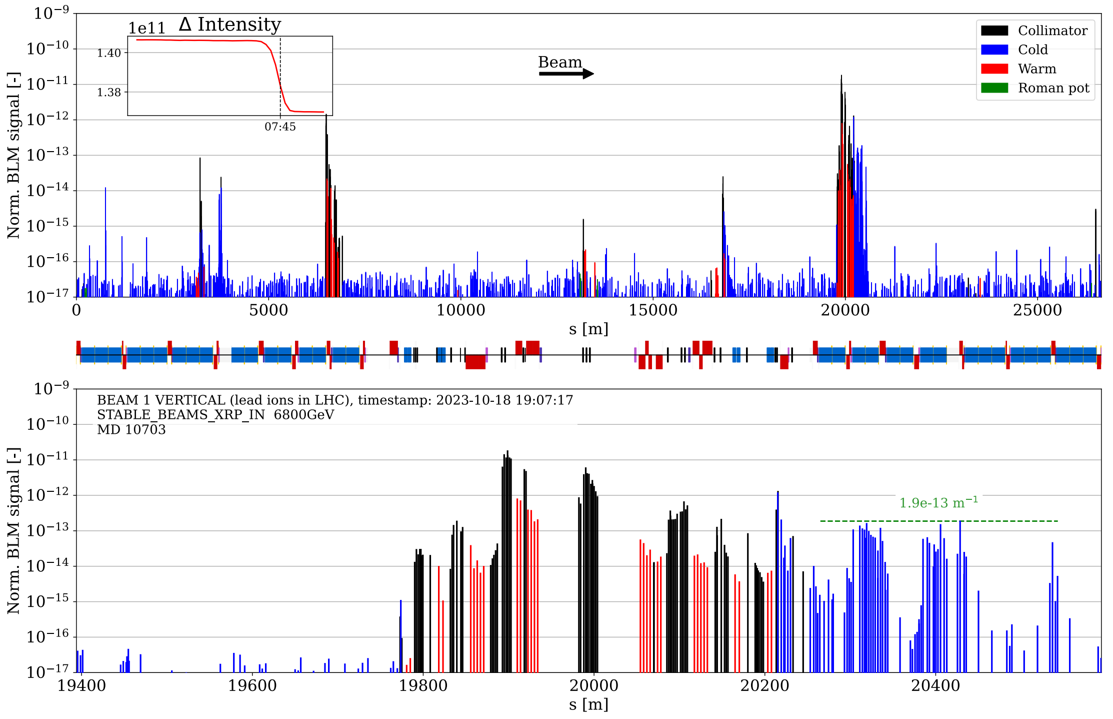
***Frederik F. Van der Veken**, G. Broggi, B. Lindström, S. Solstrand, A. Donadon Servelle, S. Redaelli, D. Veres*

ICAP'24, 03/10/2024

Outline

Code Design Philosophy - à la Xsuite

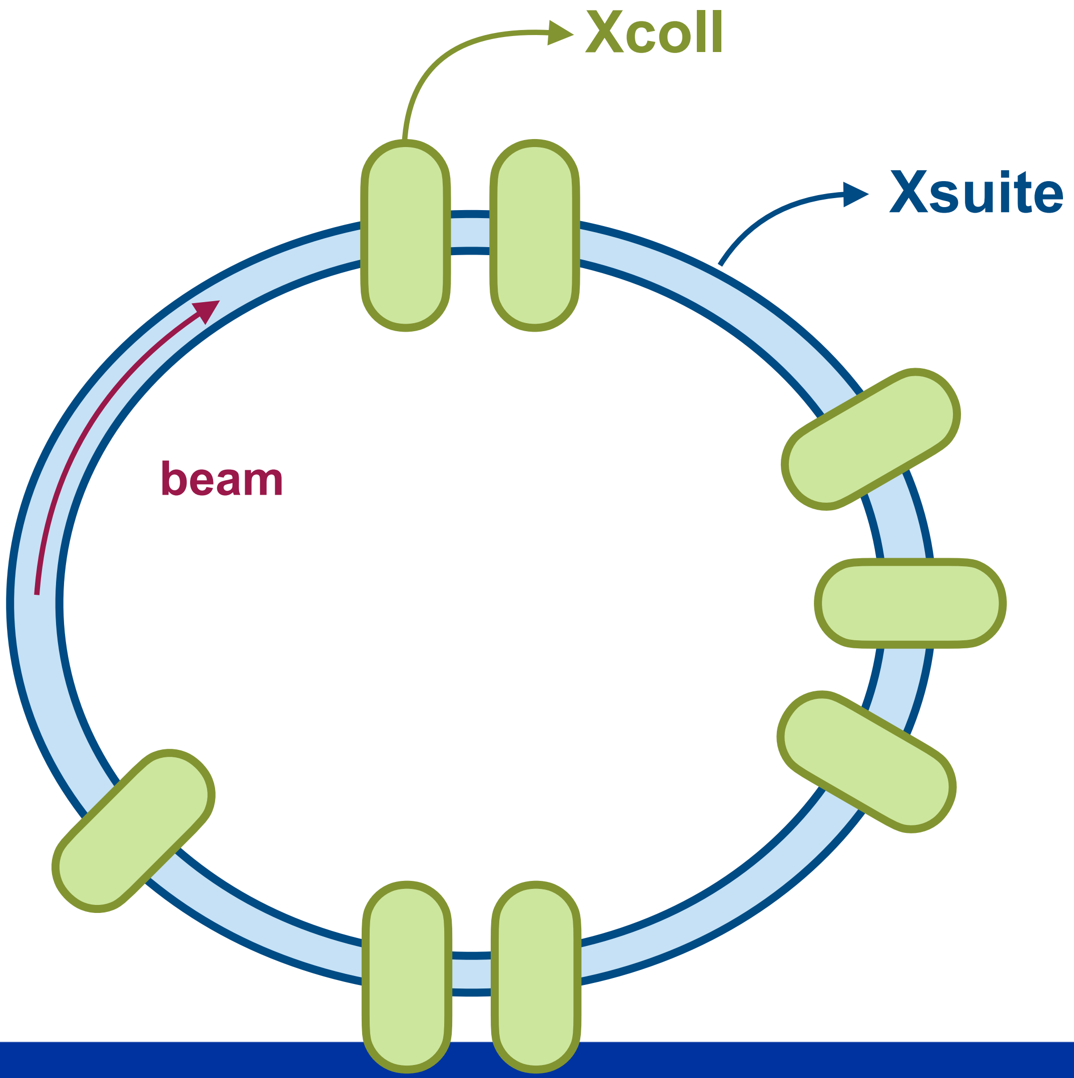
- **Standardisation:**
  - common approach for easy comparison between different simulation setups and measurement
  - e.g. direct integration with recent lossmaps tool
- **Flexibility:**
  - user-friendly modularity stimulates autonomy (not dependent on developers for small changes)
  - while guaranteeing robustness and reliability
- **Maintability:**
  - code readability and documentation is a must to ensure future-proofing code development
  - robust and encompassing test suite



Collimation Simulations

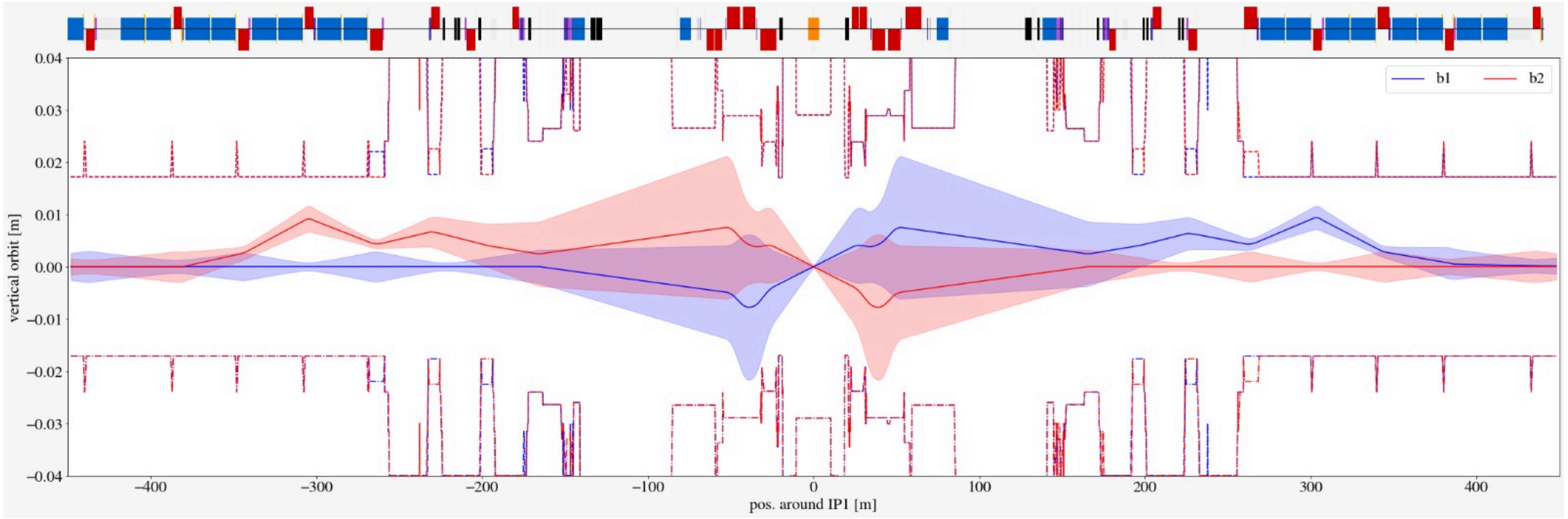
- Collimation simulations have two types of physics:
  - **accelerator physics** to track high-energy particles through a lattice
  - **material interactions** to simulate the behaviour when a particle hits a collimator

- Original code was SixTrack (FORTRAN)
- New code is **Xtrack** + **Xcoll** (Python/C)



Beam Loss Simulations

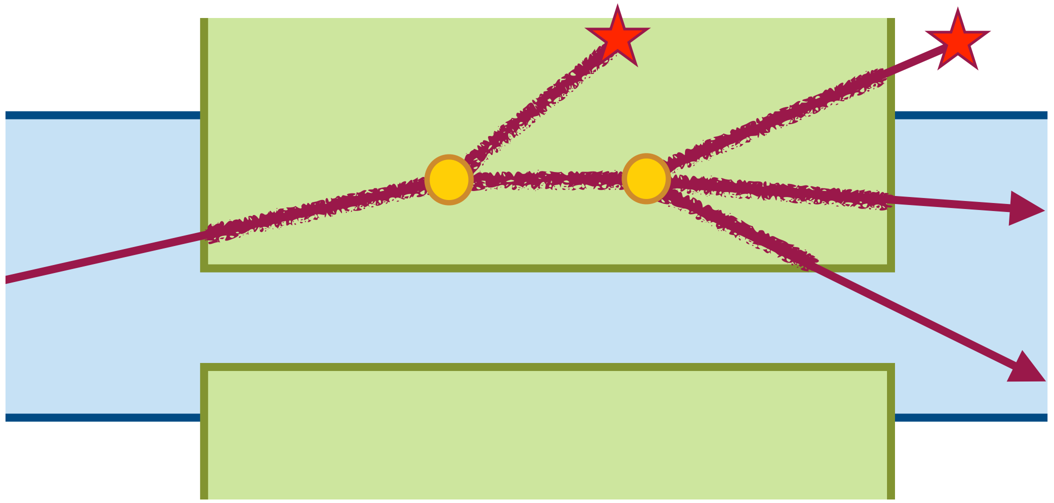
- Simulations that investigate realistic beam losses need **aperture model**
- Xtrack provides aperture elements and advanced **interpolation** to refine losses location



B. Lindström

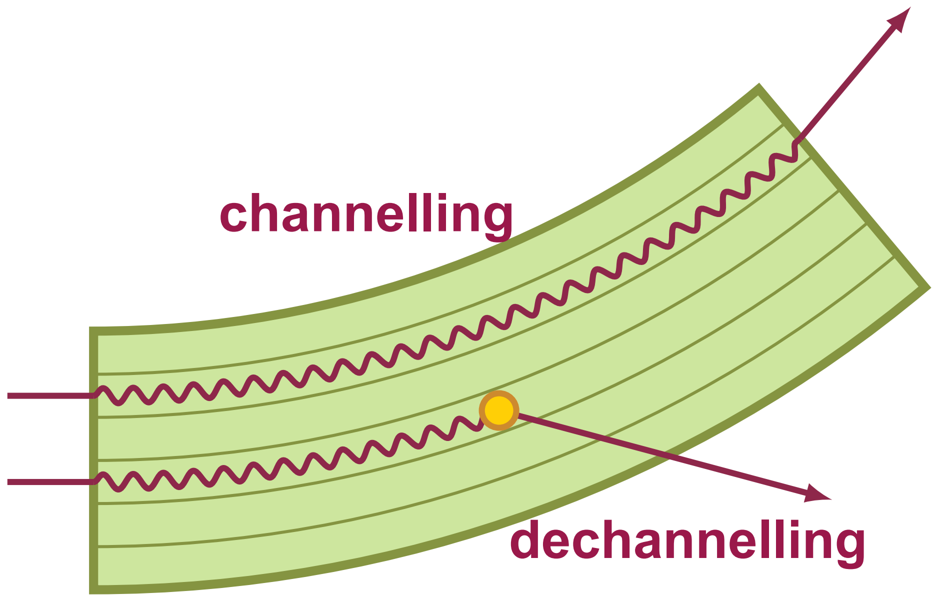
Material Interactions

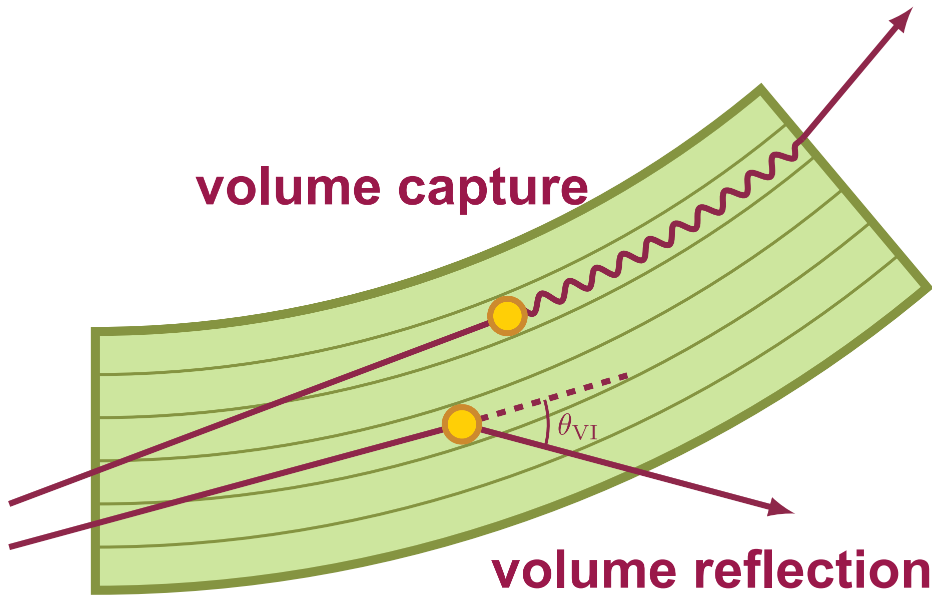
- When particle hits collimator: send to **scattering engine**
- **Hard interactions**:
  - Absorption, nuclear, coulomb, single-diffractive, ...
  - Can create multiple **children**
- Multiple **soft interactions** are accumulated as:
  - Multiple Coulomb Scattering
  - Bethe-Bloch ionisation loss
- When particle exits collimator, send **back to tracking**
- If exit beyond aperture: log as absorbed



Material Interactions: Crystals

- Very different physics in crystals: **channelling**
- Particles can lose channeling due to: nuclear interaction or **dechanneling**
- Particles can be **captured** and channelled, or **reflected** on the crystal planes
- Regular **amorphous** interactions when not parallel to crystal lattice



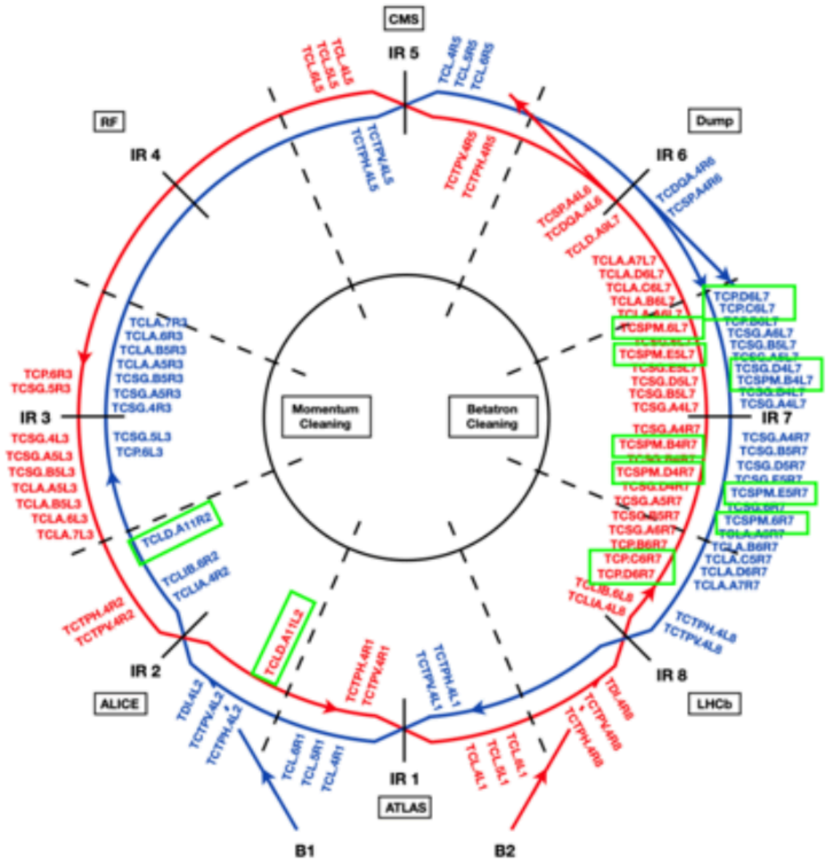


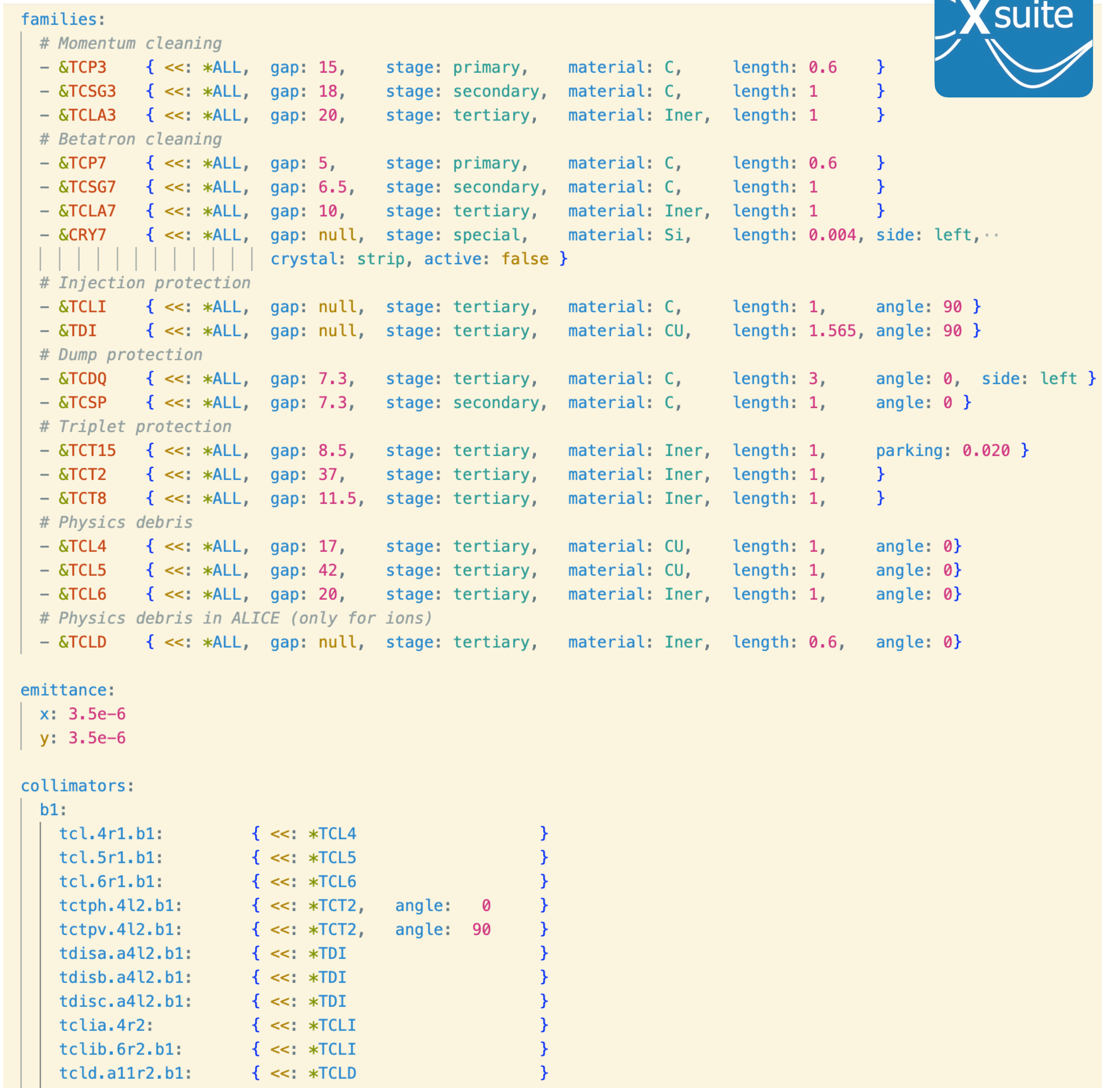
Outline

Collimator Database

- Collimation often need multitude of collimators
- Xcoll provides **database** API to install and manage collimator families

LHC
101 collimators

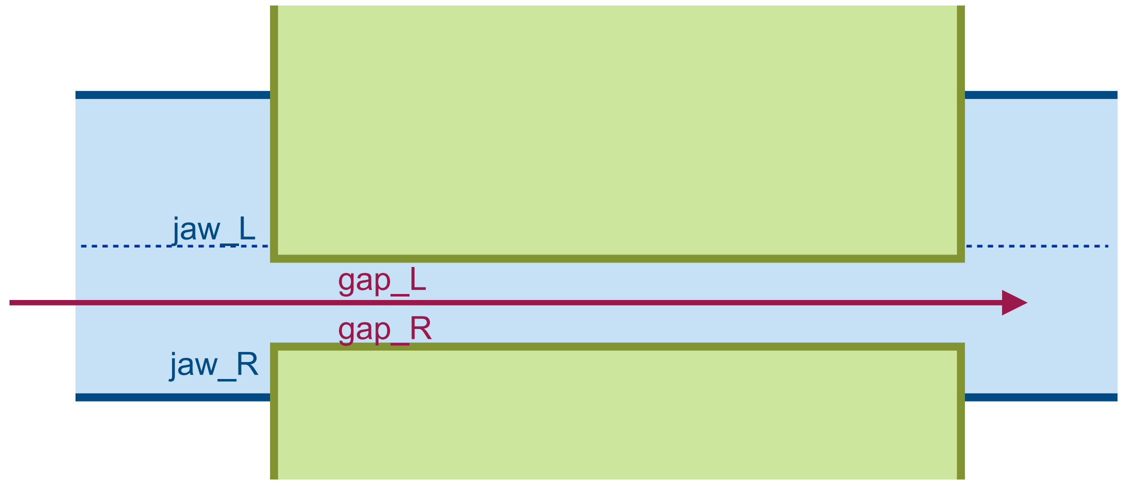


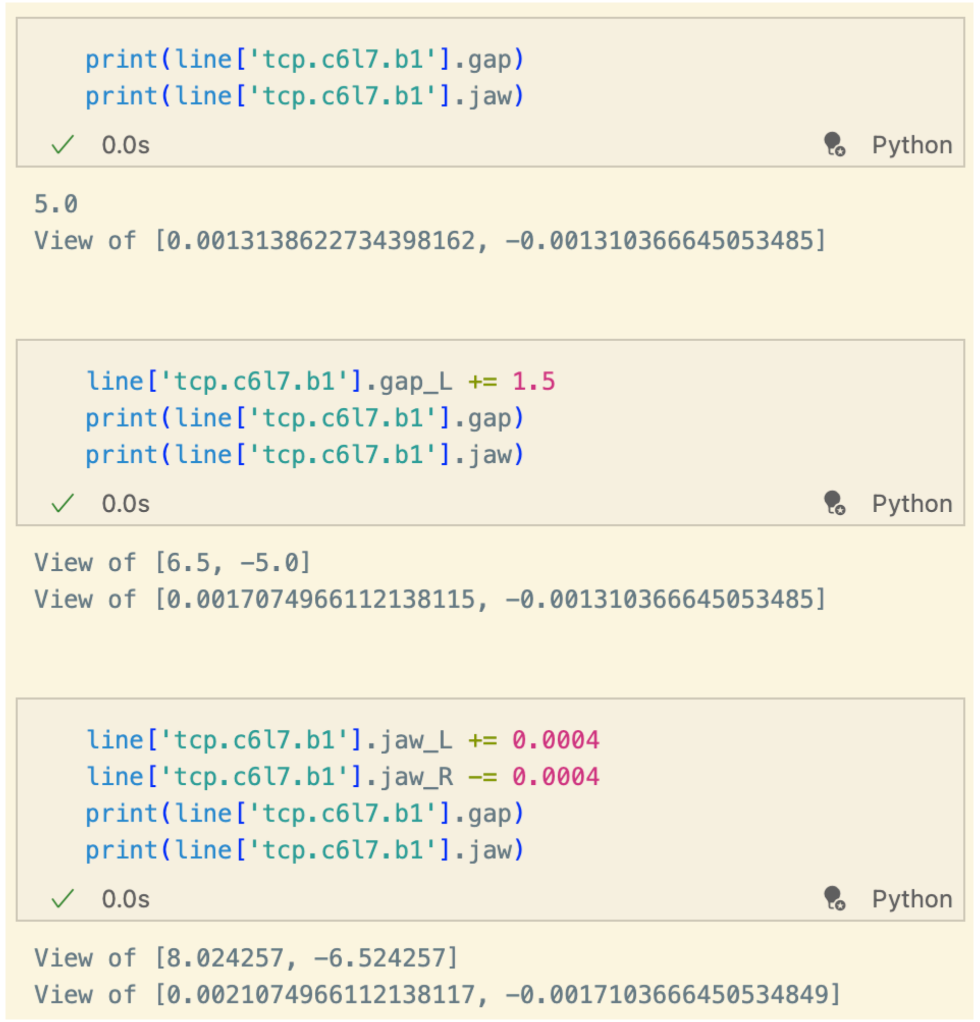


```
families:
  # Momentum cleaning
  - &TCP3    { <<: *ALL,  gap: 15,    stage: primary,    material: C,    length: 0.6    }
  - &TCSG3   { <<: *ALL,  gap: 18,    stage: secondary,  material: C,    length: 1      }
  - &TCLA3   { <<: *ALL,  gap: 20,    stage: tertiary,   material: Iner, length: 1      }
  # Betatron cleaning
  - &TCP7    { <<: *ALL,  gap: 5,     stage: primary,    material: C,    length: 0.6    }
  - &TCSG7   { <<: *ALL,  gap: 6.5,   stage: secondary,  material: C,    length: 1      }
  - &TCLA7   { <<: *ALL,  gap: 10,    stage: tertiary,   material: Iner, length: 1      }
  - &CRY7    { <<: *ALL,  gap: null,  stage: special,    material: Si,   length: 0.004, side: left,··
                          crystal: strip, active: false }
  # Injection protection
  - &TCLI    { <<: *ALL,  gap: null,  stage: tertiary,   material: C,    length: 1,     angle: 90 }
  - &TDI     { <<: *ALL,  gap: null,  stage: tertiary,   material: CU,   length: 1.565, angle: 90 }
  # Dump protection
  - &TCDQ    { <<: *ALL,  gap: 7.3,   stage: tertiary,   material: C,    length: 3,     angle: 0,  side: left }
  - &TCSP    { <<: *ALL,  gap: 7.3,   stage: secondary,  material: C,    length: 1,     angle: 0 }
  # Triplet protection
  - &TCT15   { <<: *ALL,  gap: 8.5,   stage: tertiary,   material: Iner, length: 1,     parking: 0.020 }
  - &TCT2    { <<: *ALL,  gap: 37,    stage: tertiary,   material: Iner, length: 1,     }
  - &TCT8    { <<: *ALL,  gap: 11.5,  stage: tertiary,   material: Iner, length: 1,     }
  # Physics debris
  - &TCL4    { <<: *ALL,  gap: 17,    stage: tertiary,   material: CU,   length: 1,     angle: 0}
  - &TCL5    { <<: *ALL,  gap: 42,    stage: tertiary,   material: CU,   length: 1,     angle: 0}
  - &TCL6    { <<: *ALL,  gap: 20,    stage: tertiary,   material: Iner, length: 1,     angle: 0}
  # Physics debris in ALICE (only for ions)
  - &TCLD    { <<: *ALL,  gap: null,  stage: tertiary,   material: Iner, length: 0.6,   angle: 0}

emittance:
  x: 3.5e-6
  y: 3.5e-6

collimators:
  b1:
    tcl.4r1.b1:        { <<: *TCL4                      }
    tcl.5r1.b1:        { <<: *TCL5                      }
    tcl.6r1.b1:        { <<: *TCL6                      }
    tctph.4l2.b1:      { <<: *TCT2,   angle:   0        }
    tctpv.4l2.b1:      { <<: *TCT2,   angle:  90        }
    tdisa.a4l2.b1:     { <<: *TDI                       }
    tdisb.a4l2.b1:     { <<: *TDI                       }
    tdisc.a4l2.b1:     { <<: *TDI                       }
    tclia.4r2:         { <<: *TCLI                      }
    tclib.6r2.b1:      { <<: *TCLI                      }
    tcld.a11r2.b1:     { <<: *TCLD                      }
```

Collimator Gap and Jaw

- Collimator openings typically specified in **beam size** around the **closed orbit**
- Tracking uses openings in **absolute units** around the **survey centre** (mostly equal to the beam pipe centre)
- Xcoll leverages on Python's class property system to provide **flexible interplay** between both





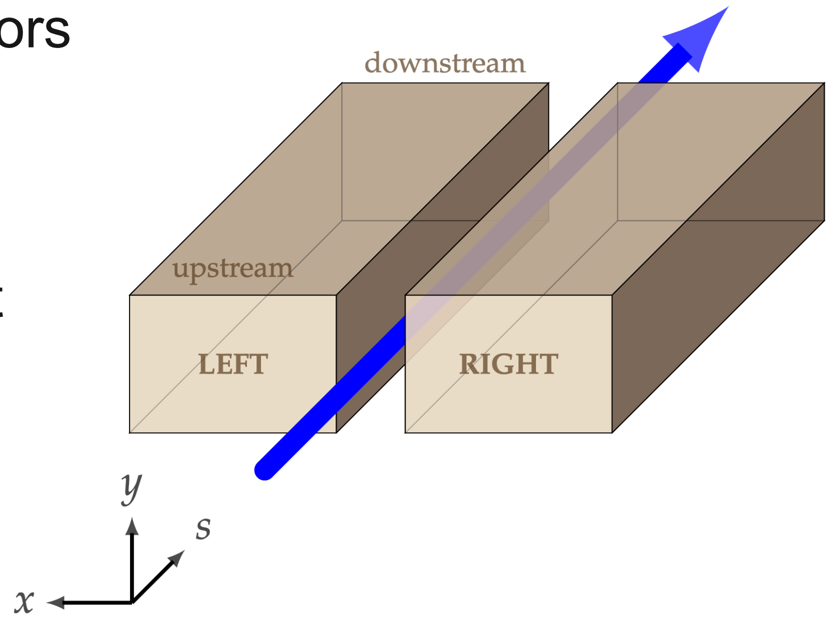
```
print(line['tcp.c6l7.b1'].gap)
print(line['tcp.c6l7.b1'].jaw)
```
✓ 0.0s Python
```
5.0
View of [0.0013138622734398162, -0.0013103666450534 85]
```

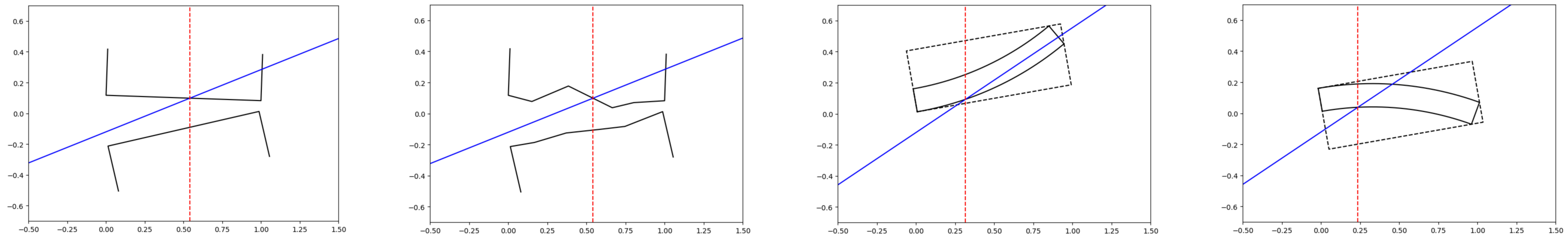
```
line['tcp.c6l7.b1'].gap_L += 1.5
print(line['tcp.c6l7.b1'].gap)
print(line['tcp.c6l7.b1'].jaw)
```
✓ 0.0s Python
```
View of [6.5, -5.0]
View of [0.0017074966112138115, -0.0013103666450534 85]
```

```
line['tcp.c6l7.b1'].jaw_L += 0.0004
line['tcp.c6l7.b1'].jaw_R -= 0.0004
print(line['tcp.c6l7.b1'].gap)
print(line['tcp.c6l7.b1'].jaw)
```
✓ 0.0s Python
```
View of [8.024257, -6.524257]
View of [0.0021074966112138117, -0.0017103666450534849]
```

Collimator Geometry

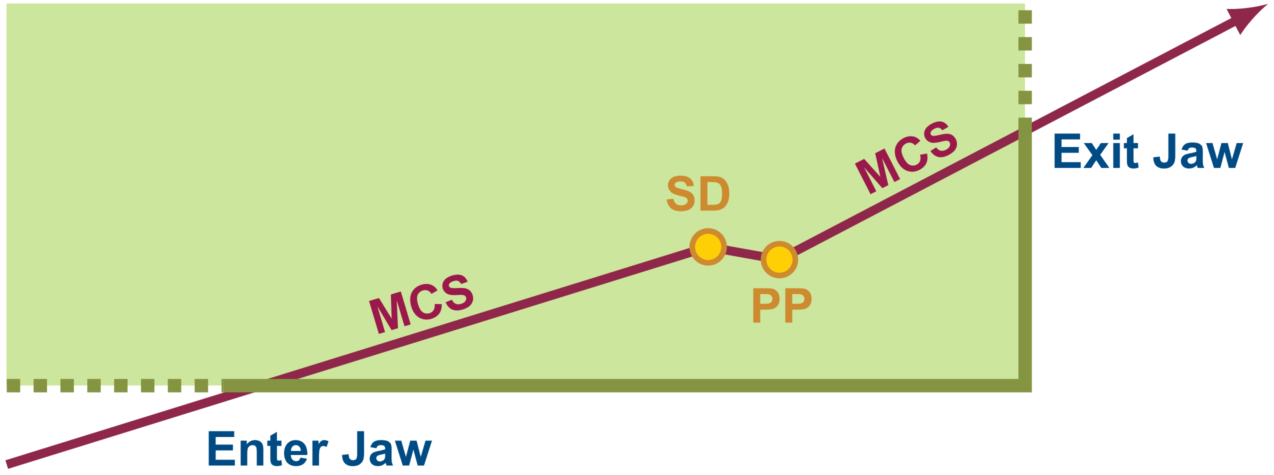
- Important emphasis on **geometry** when working with collimators
- Jaw position, angle around the beam axis, tilt around the transverse axis, non-standard jaw structure, ...
- Routines in C to define arbitrary shapes, and calculate impact and exit points
- Full **separation** between geometry and scattering





Impact Table

- Table to log impacts, interactions, and exits
- **Full table** available in Everest
- **Impacts** available in Geant4 coupling
- WIP for a FLUKA table

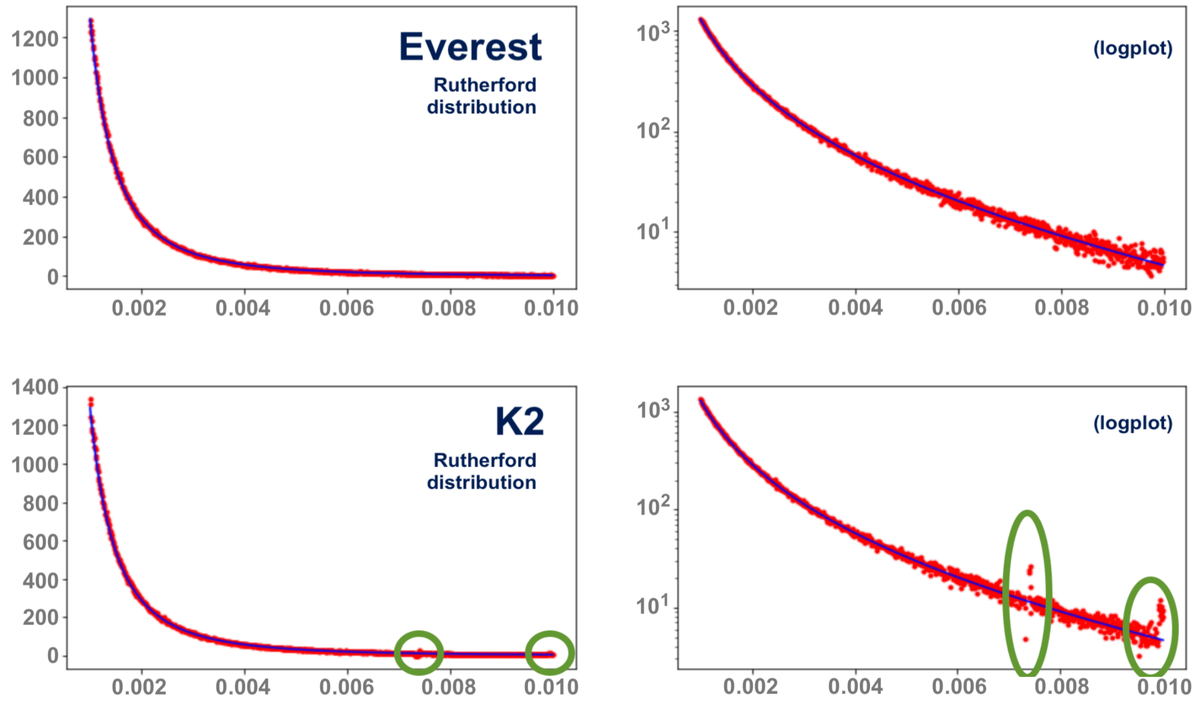


	interaction_type	id_before	s_before	x_before	px_before	id_after	s_after	x_after	px_after
6847	Enter Jaw L	2227	0.538346	0.000000e+00	0.000015	-1	-1.000000	-1.000000e+00	-1.000000
6848	Multiple Coulomb Scattering	2227	0.538346	0.000000e+00	0.000015	2227	0.574394	5.094311e-07	0.000013
6849	Single Diffractive	2227	0.574394	5.094311e-07	0.000013	2227	0.574394	5.094311e-07	-0.000007
6850	Multiple Coulomb Scattering	2227	0.574394	5.094311e-07	-0.000007	2227	0.580198	4.678767e-07	-0.000007
6851	PP Elastic	2227	0.580198	4.678767e-07	-0.000007	2227	0.580198	4.678767e-07	0.000024
6852	Multiple Coulomb Scattering	2227	0.580198	4.678767e-07	0.000024	2227	0.600000	9.381799e-07	0.000024
6853	Exit Jaw	2227	0.600000	9.381799e-07	0.000024	-1	-1.000000	-1.000000e+00	-1.000000

Outline

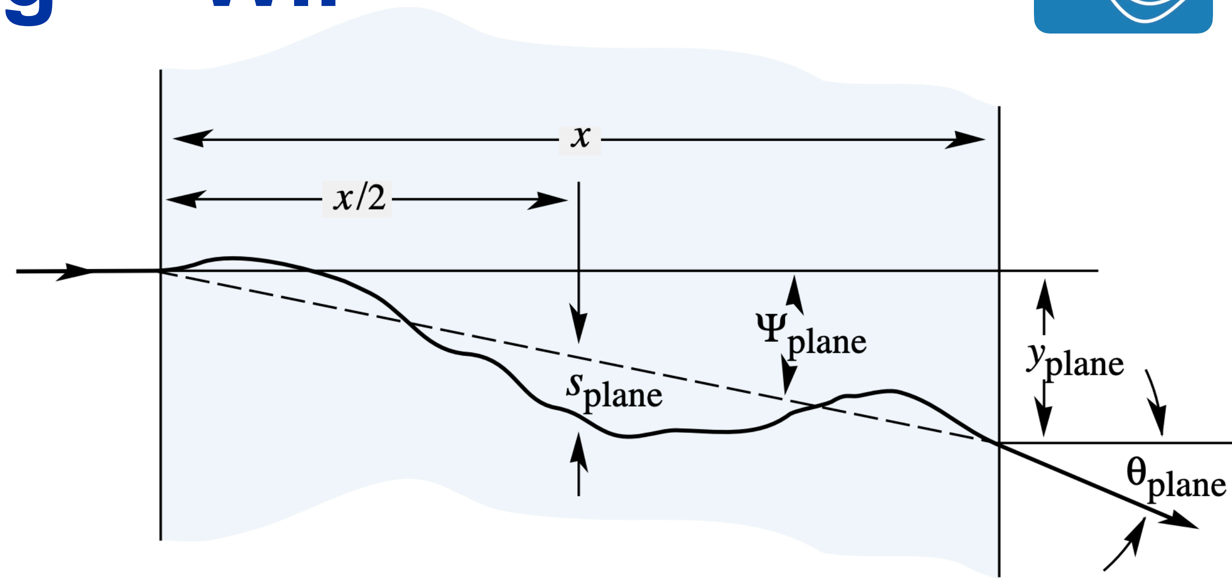
Everest as a Successor to K2

- **Native C** implementation (based on a translation from K2 in FORTRAN, and expanded further)
- **Speed gain** of factor ~6 compared to original implementation (single CPU)
- Strong emphasis on **code readability** and logic flow
- **OpenMP**-compatible
- Working on GPU implementation
- Removed artefacts from Rutherford random generator

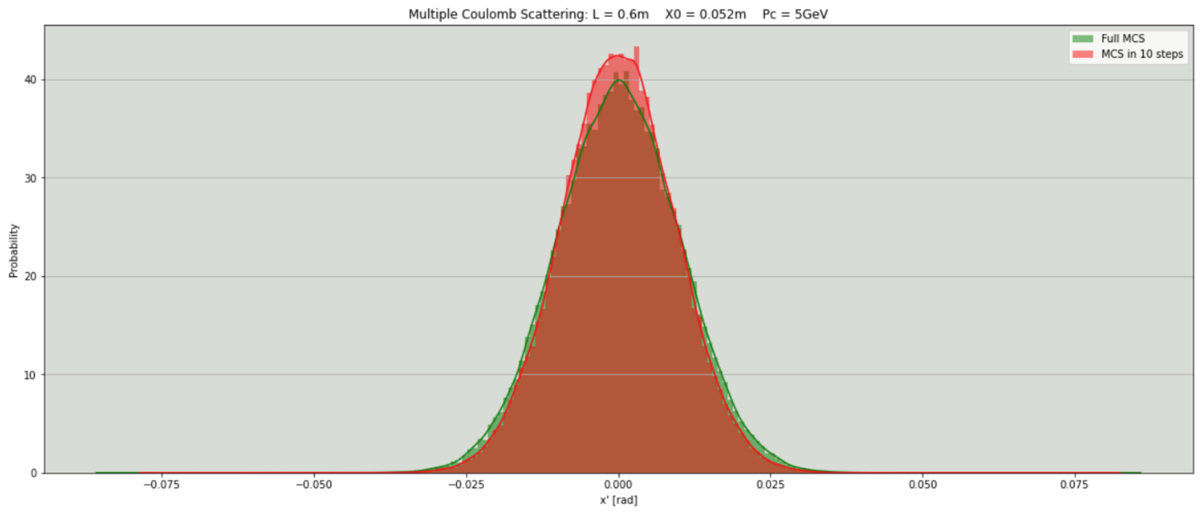


Multiple Coulomb Scattering - WIP

- Resummation of many small angle scatters
- Acts as **kick** and **displacement**
- Depends on length of traversed material
- Stepwise approach to account for **edge effects**

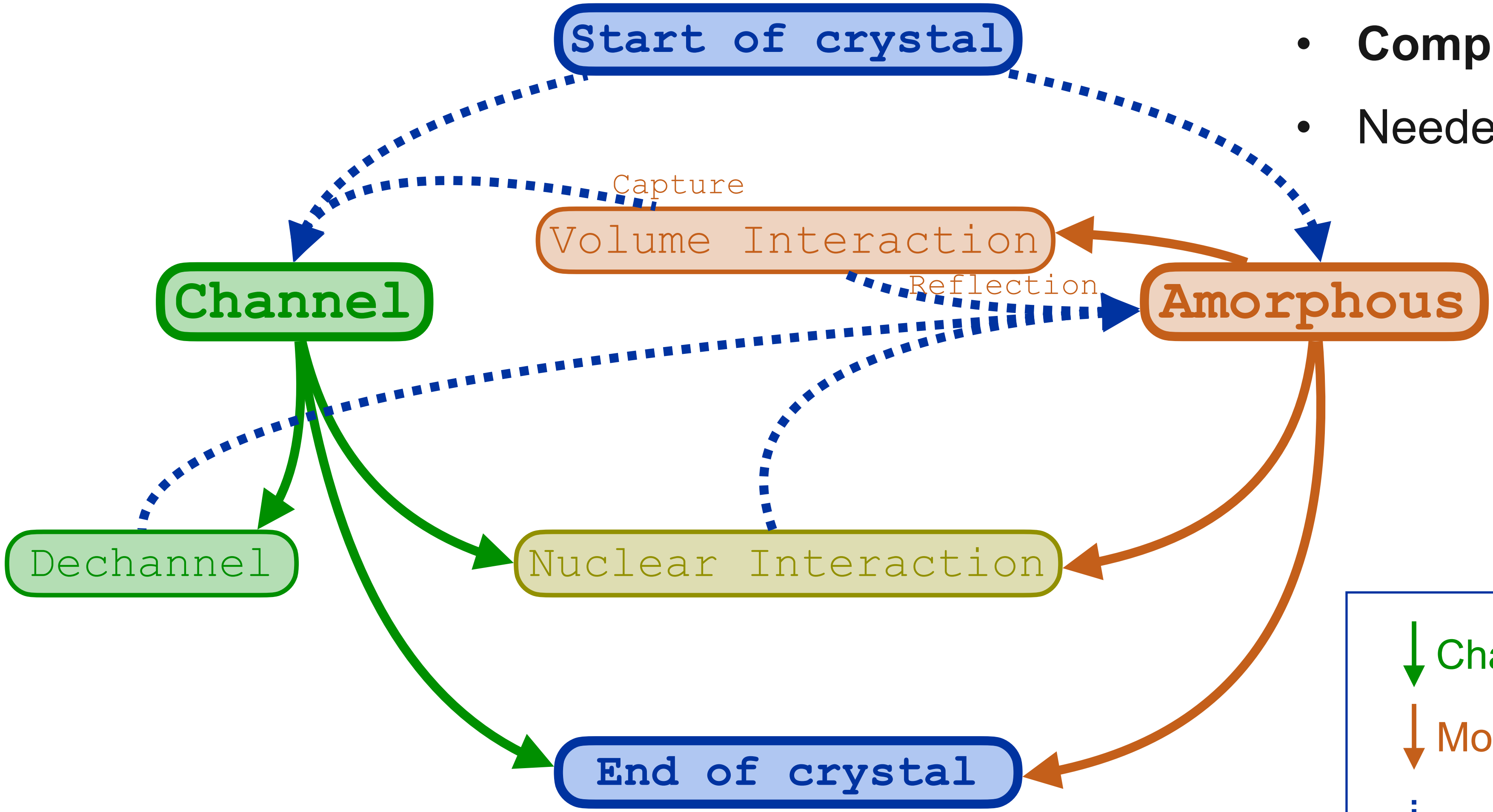


source: PDG 2022

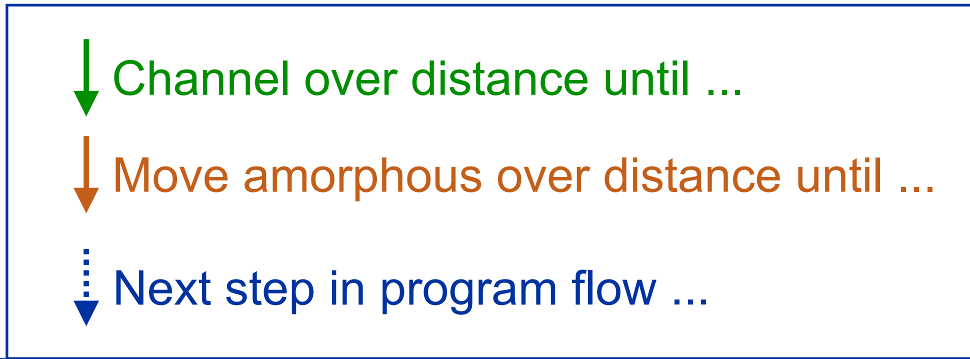


- But MCS not meant to be used in steps, as this **depletes the tails**
- Working on implementation to predict exit or interaction point, to know **exact traversed length**

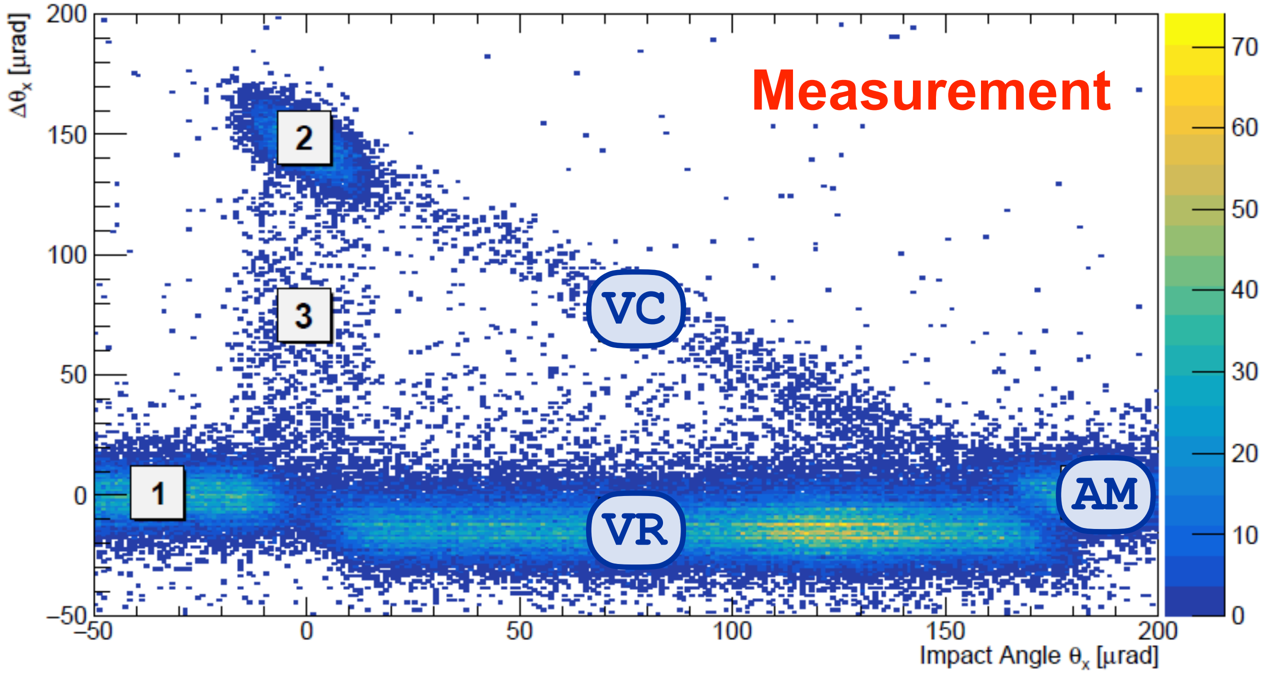
Everest Crystals: New Logic Flow

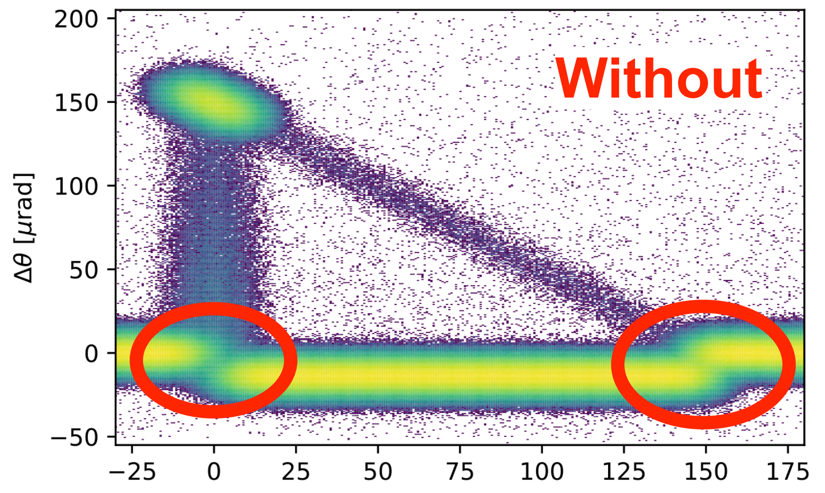


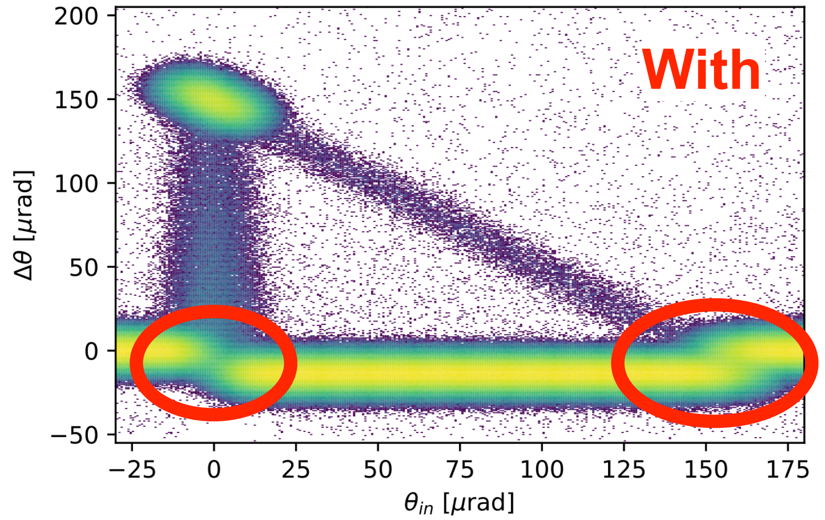
- **Complete revision** of crystal routine
- Needed to simulate **long crystals**


Channel over distance until ...
Move amorphous over distance until ...
Next step in program flow ...

Crystals Transition Regions - WIP



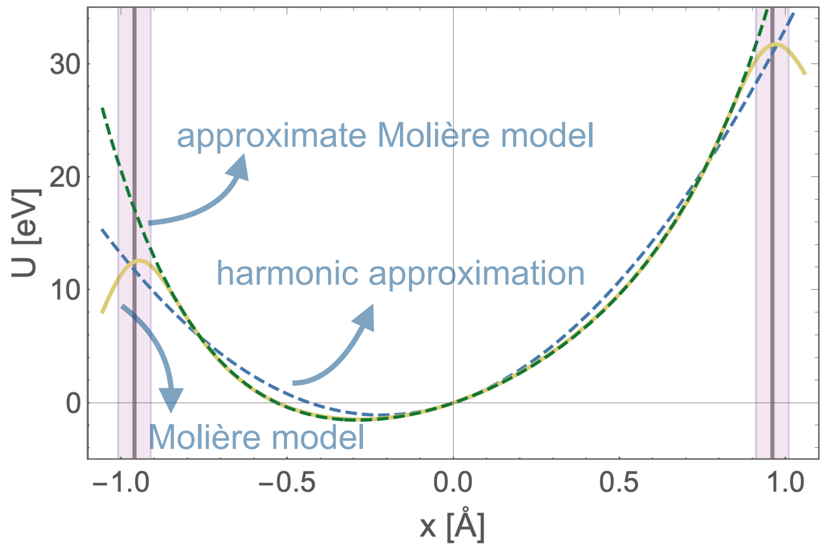


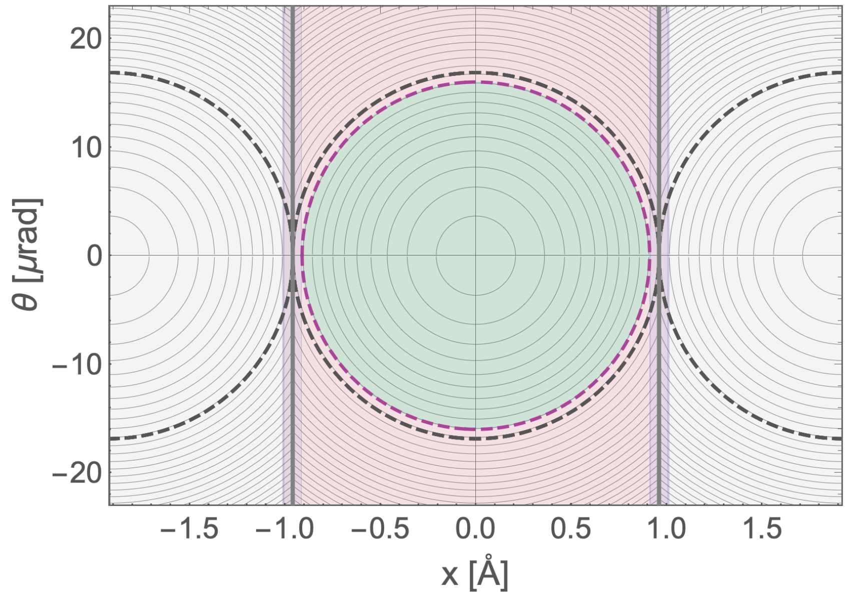


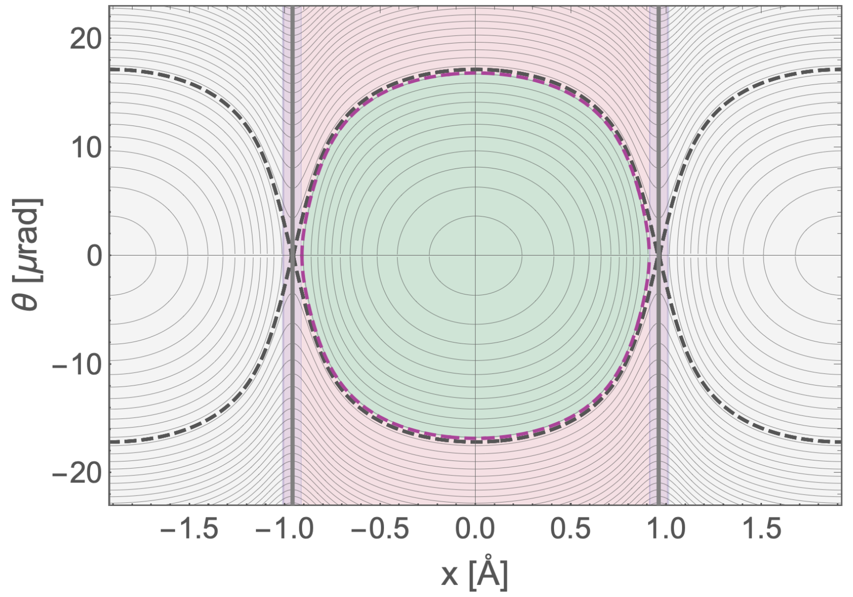
- Transition regions taken over from K2
- Naturally arise (at least partially) from new code logic flow
- Investigations are on-going

Exact Crystal Channeling - WIP

- Original implementation applies **statistical approach**
- Evolving towards more **analytical** approach:
  - to better understand output angle distribution
  - to allow **sliced crystals**
- Solved for **straight crystal**

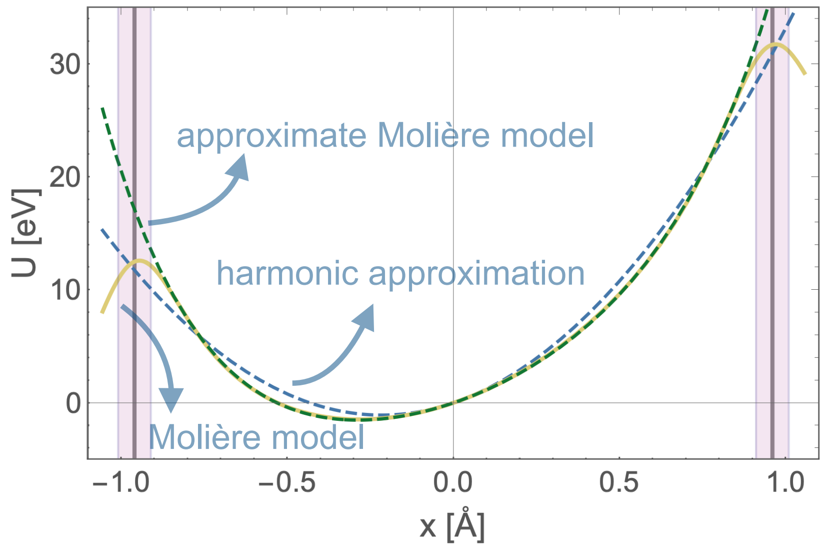


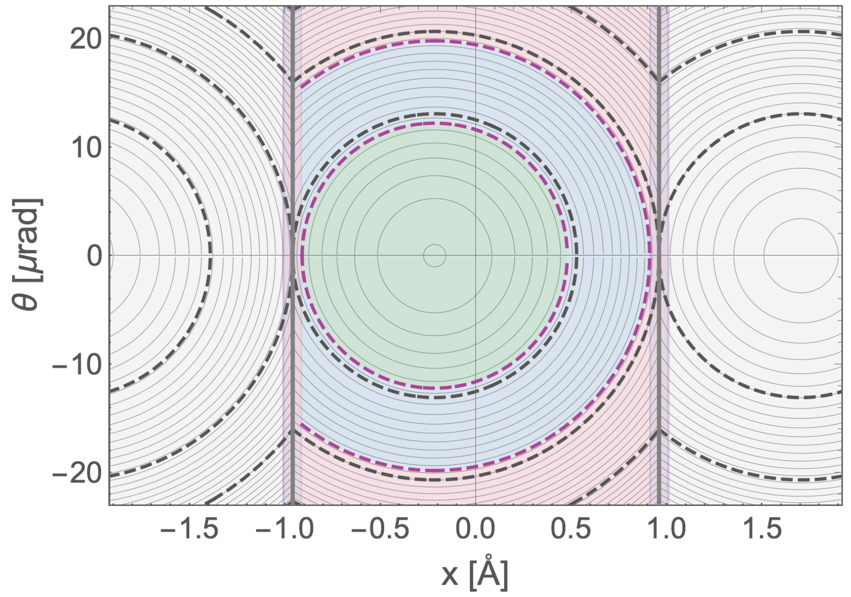


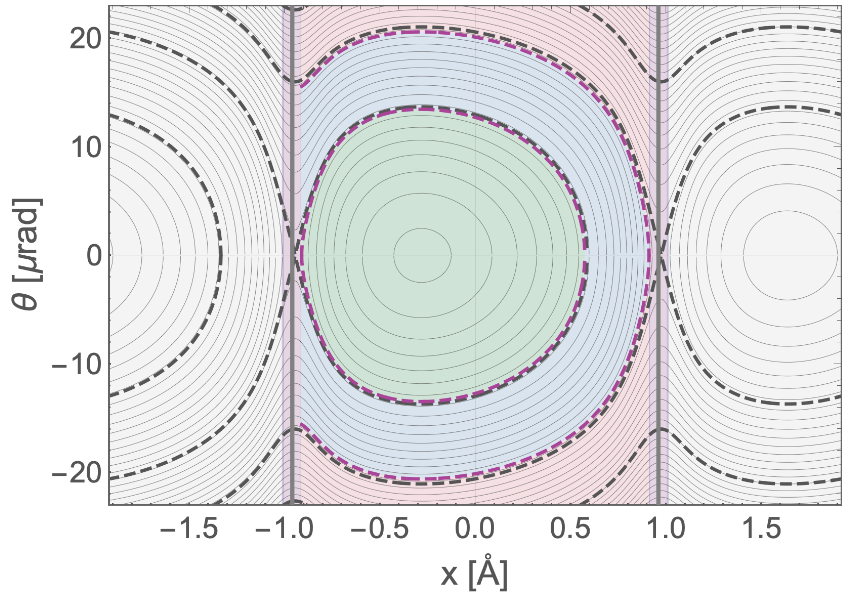


Exact Crystal Channeling - WIP

- Original implementation applies **statistical approach**
- Evolving towards more **analytical** approach:
  - to better understand output angle distribution
  - to allow **sliced crystals**
- Working on a solution for bent crystals

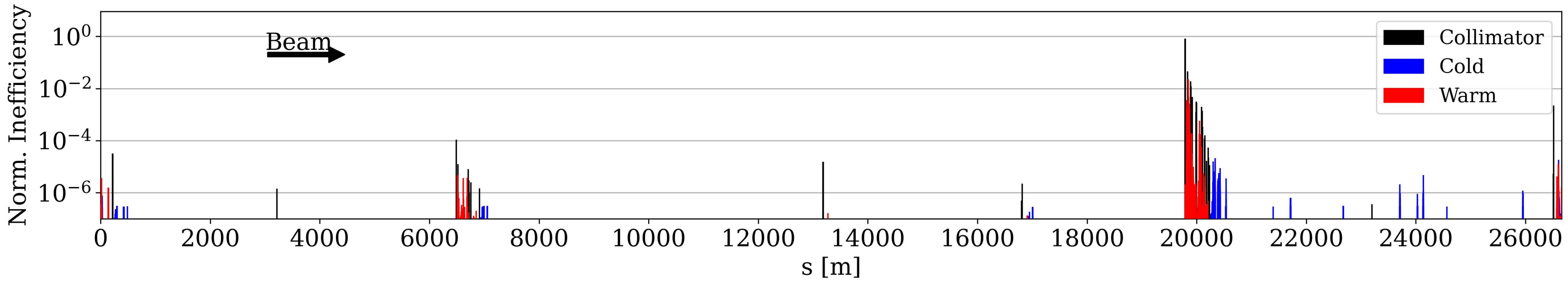






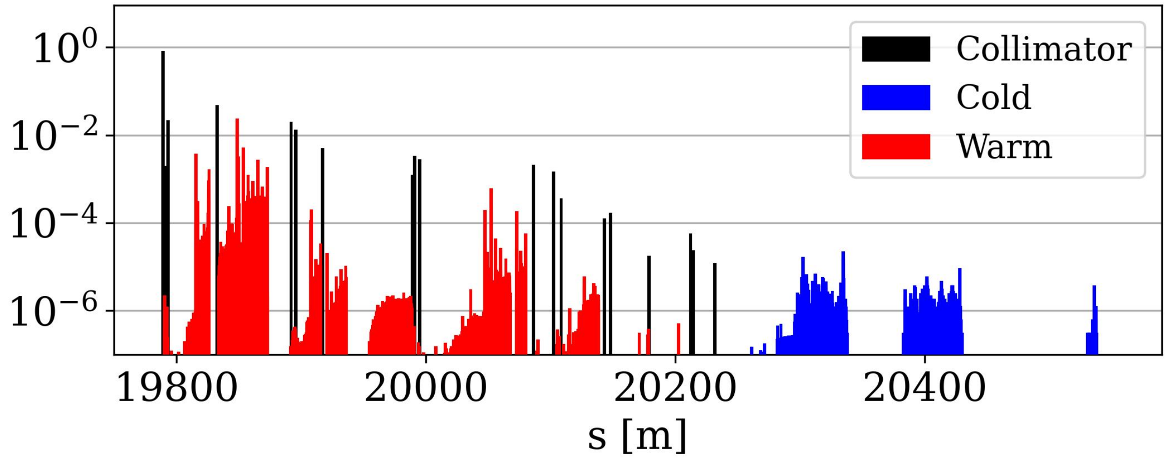
Outline

Coupling to FLUKA



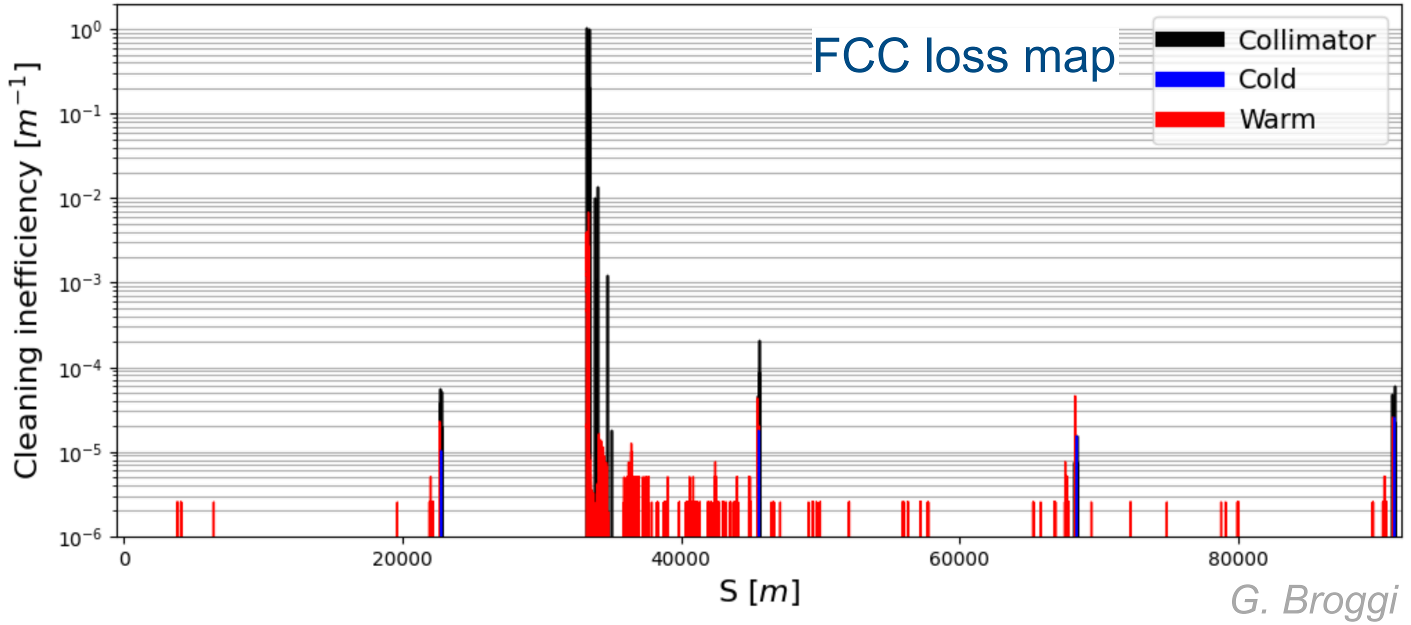
- Not trivial to set up communication to FLUKA (fluka and flukaserver executable, network protocol)
- Xcoll manages: file generation, processes, and communication, for **maximal user-friendliness**
- Modularity and flexibility still lacking
  - need to **freeze collimators** after connection
- Many layers:

Xcoll → *SixTrack protocol (FORTRAN)*
→ *FlukaIO (FORTRAN)*
→ *FlukaIO (C)*
→ *FLUKA*



Coupling to Geant4

- Xcoll manages communication to BDSIM via collimasim
- Not trivial to set up $\longrightarrow$ WIP
- Modularity and flexibility still lacking
  - need to **freeze collimators** after connection
- Used in full production for FCC studies
- Several layers:
  Xcoll $\longrightarrow$ *collimasim (Python)*
  $\longrightarrow$ *BDSIM (C++)*
  $\longrightarrow$ *Geant4*
- Need better integration



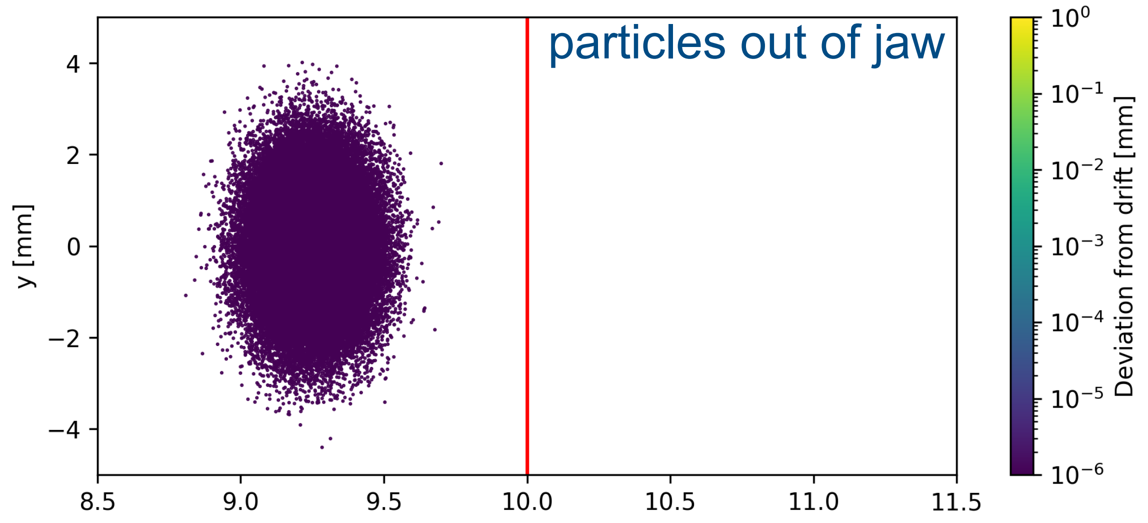
G. Broggi

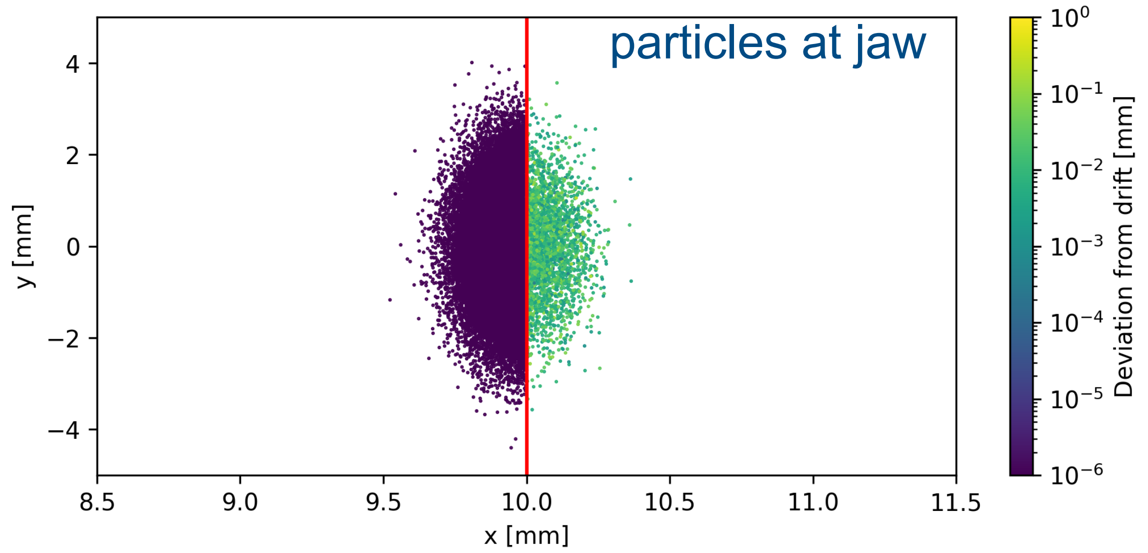
Timing Comparison

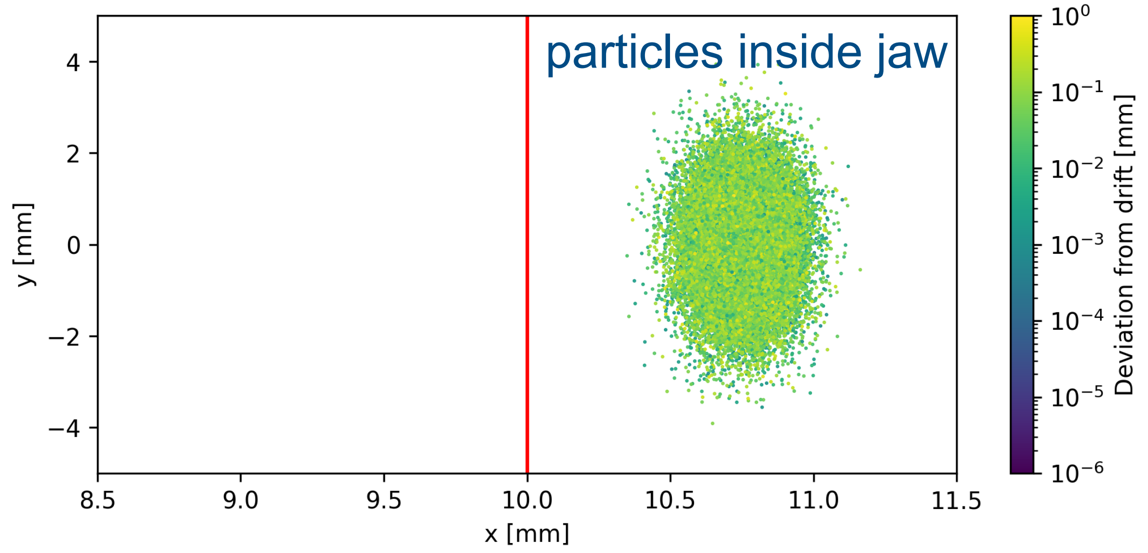
- Very clear **speed gain** between Everest and external material scattering codes
- FLUKA still x10 slower than Geant4:
  - realistic geometry vs box
  - communication to FLUKA can still be optimised

100k particles	Outside Jaw		At Jaw		Inside jaw	
	survived	CPU time	survived	CPU time	survived	CPU time
Everest	100'000	8.2ms	52'897	29.7ms	5'239	46.6ms
Geant4	100'000	2.5	58'454	25.3s	12'577	52.0s
FLUKA	100'000	100.2s	68'852	565.7s	35'293	1146.4s

Benchmark: Everest

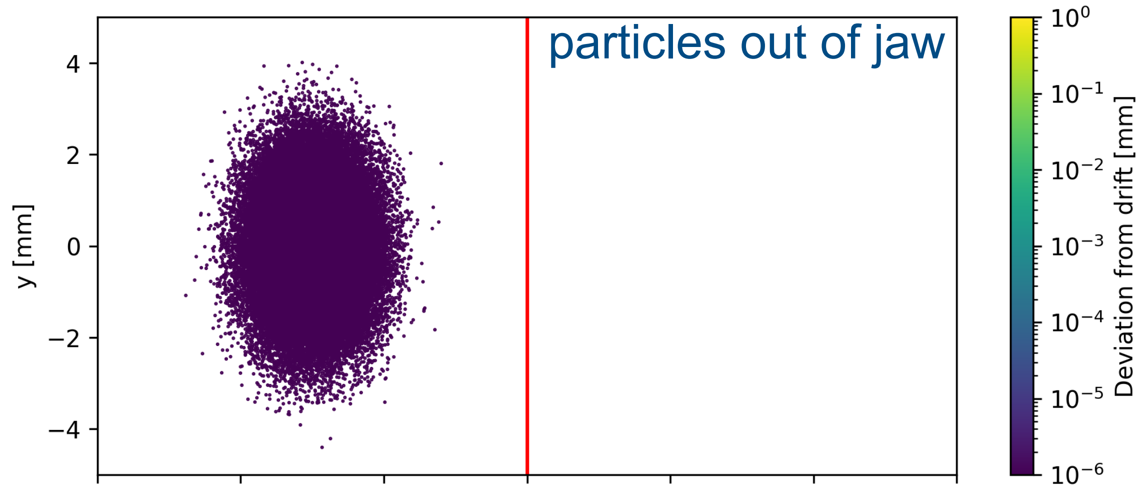


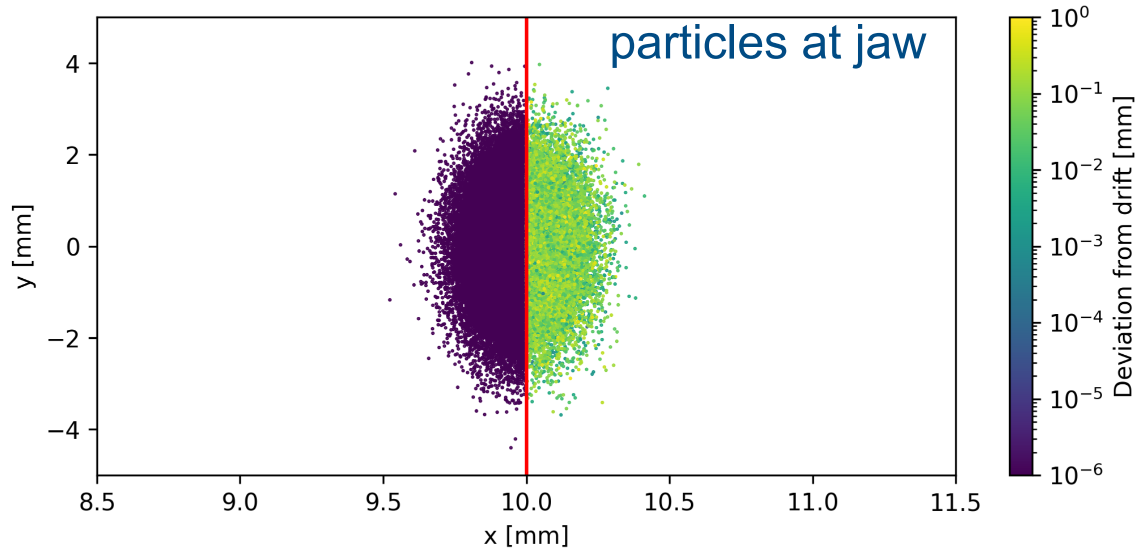


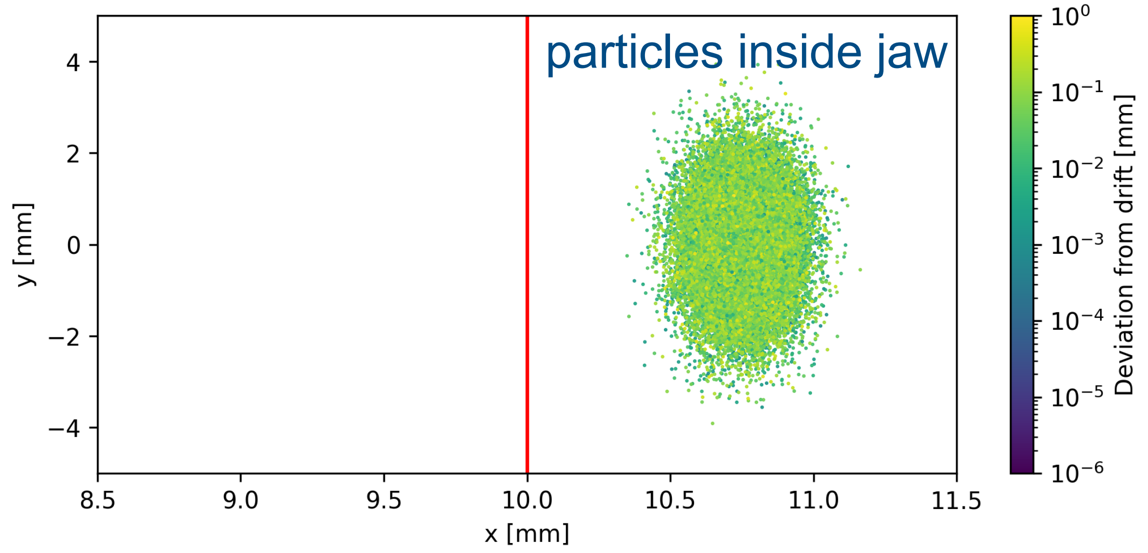


- Benchmark under active analysis

Benchmark: FLUKA

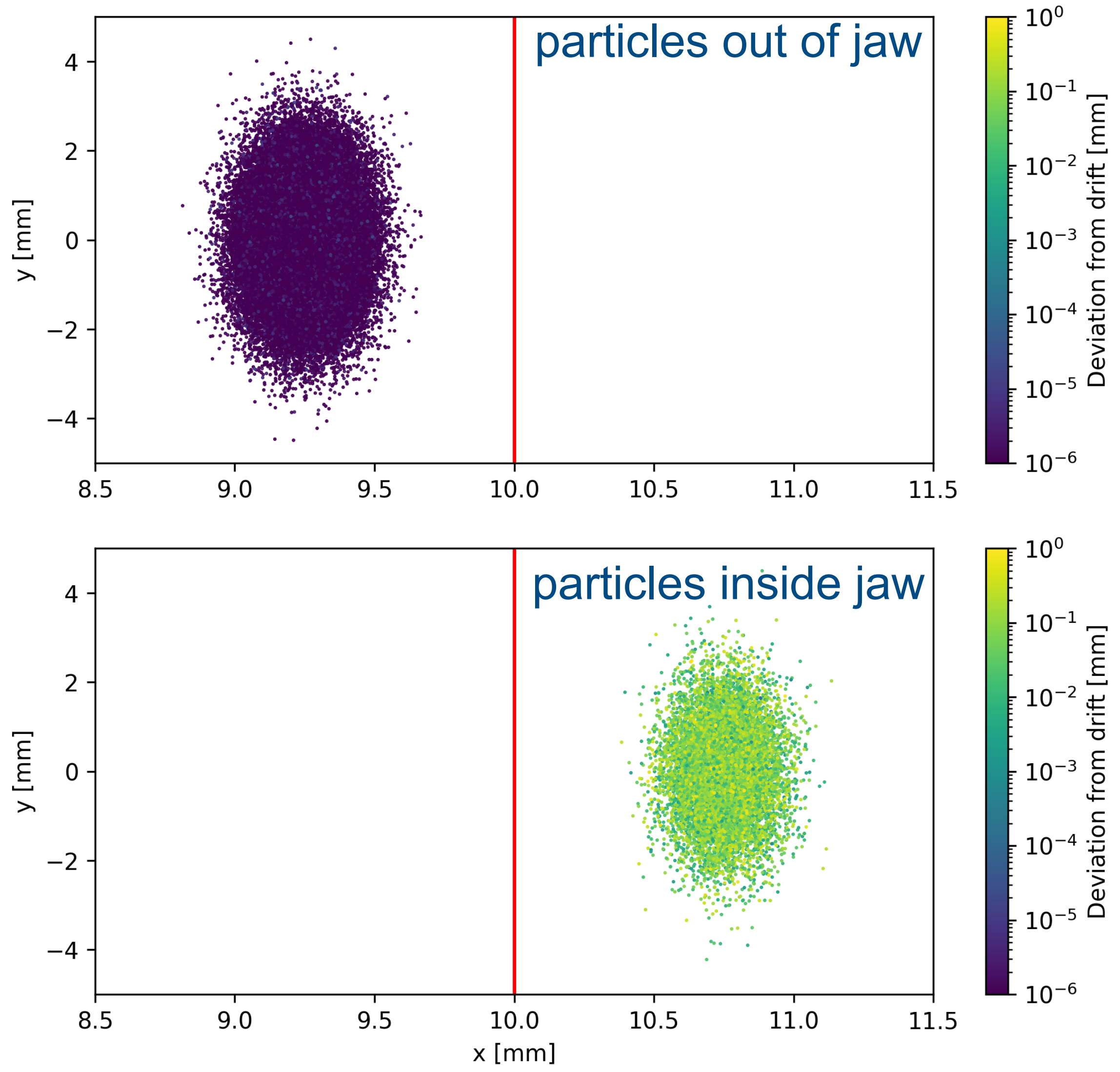


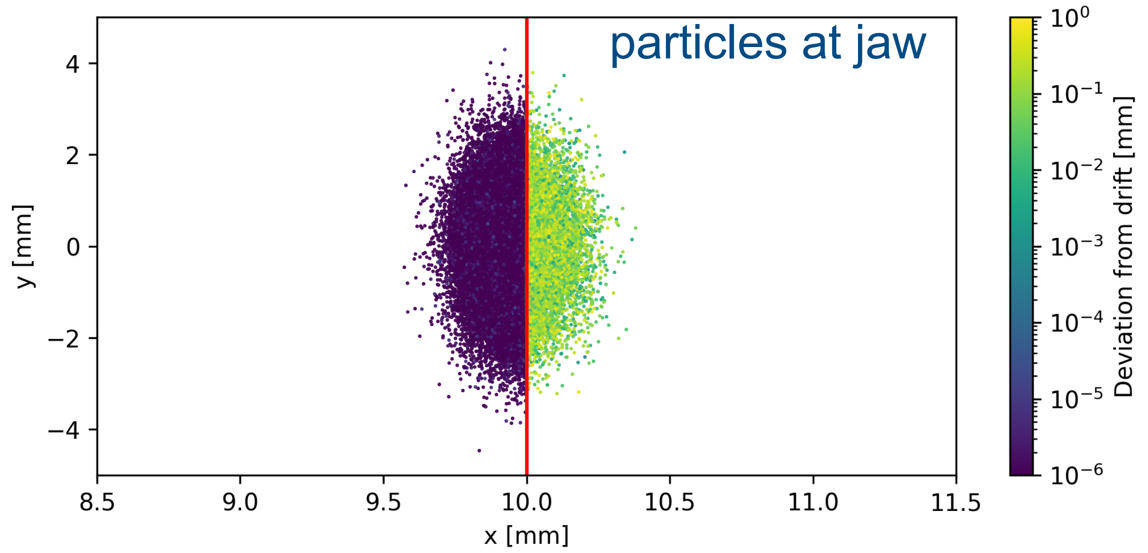




- Benchmark under active analysis

Benchmark: Geant4





- Benchmark under active analysis

Outline

LHC Example: Betatron Cleaning During Ramp

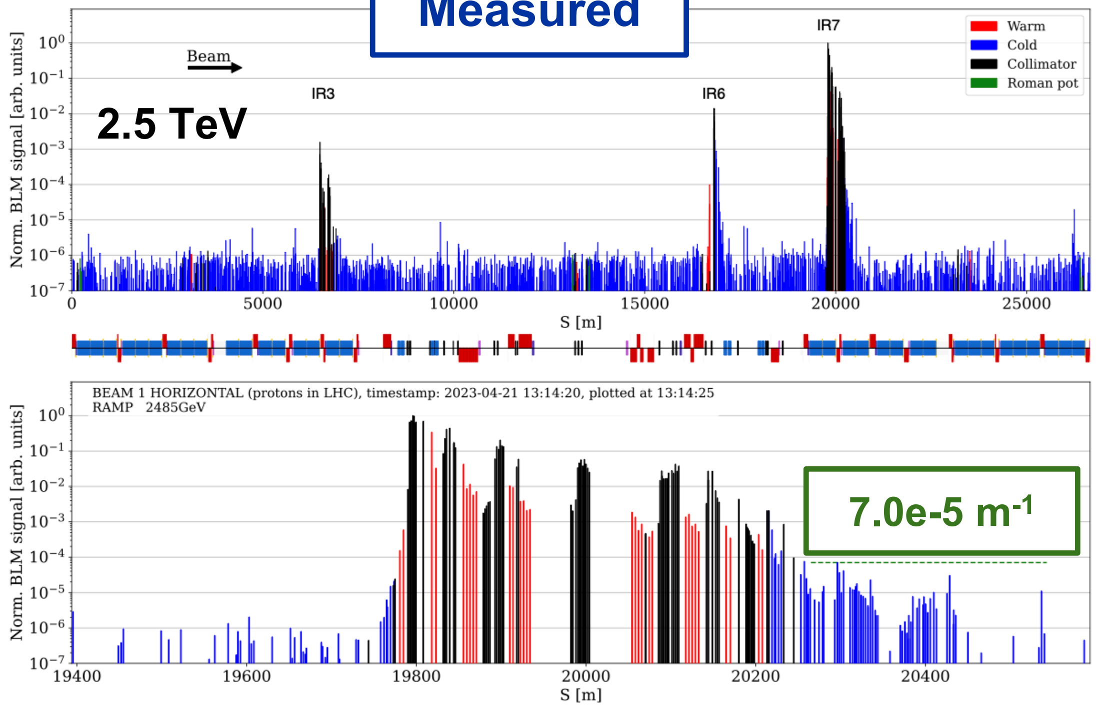
- **Good qualitative agreement** between measurement and simulation
  - Caveat: comparing losses signal to losses simulations not trivial
  - Similar losses patterns in collimator insertion regions

Cleaning inefficiency simulations

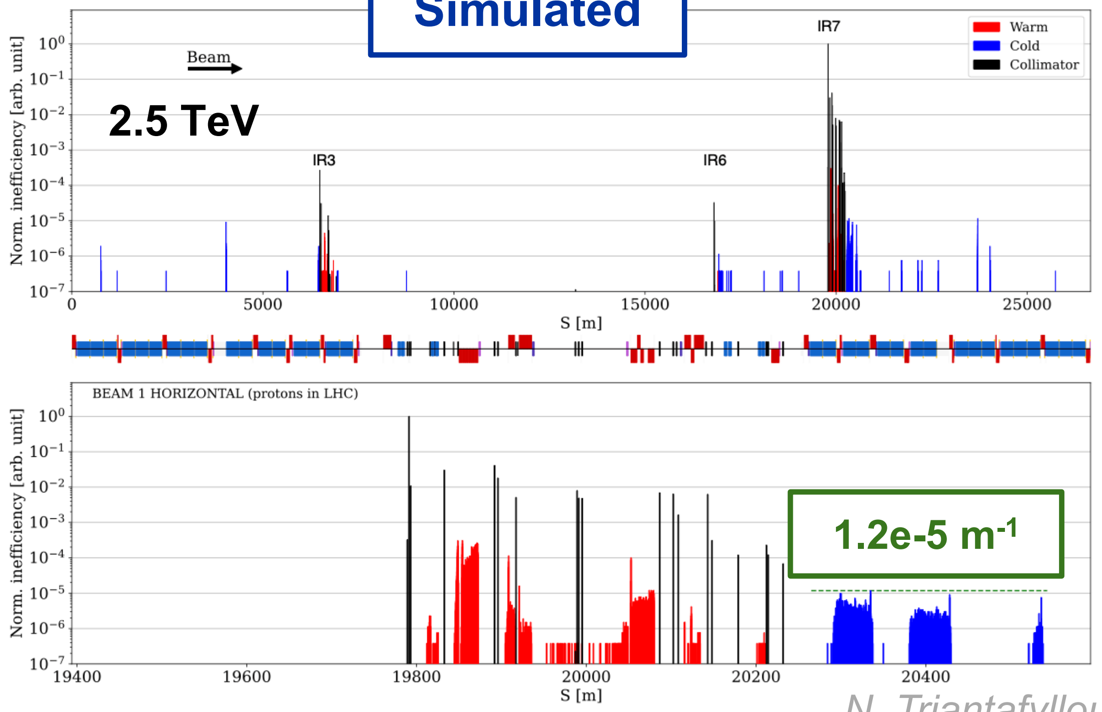
$$\eta = \frac{N_{\text{loc}}}{N_{\text{tot}}\Delta s}$$

where N_{loc} the local losses over distance Δs and N_{tot} is the total number of losses in the collimation system

Measured



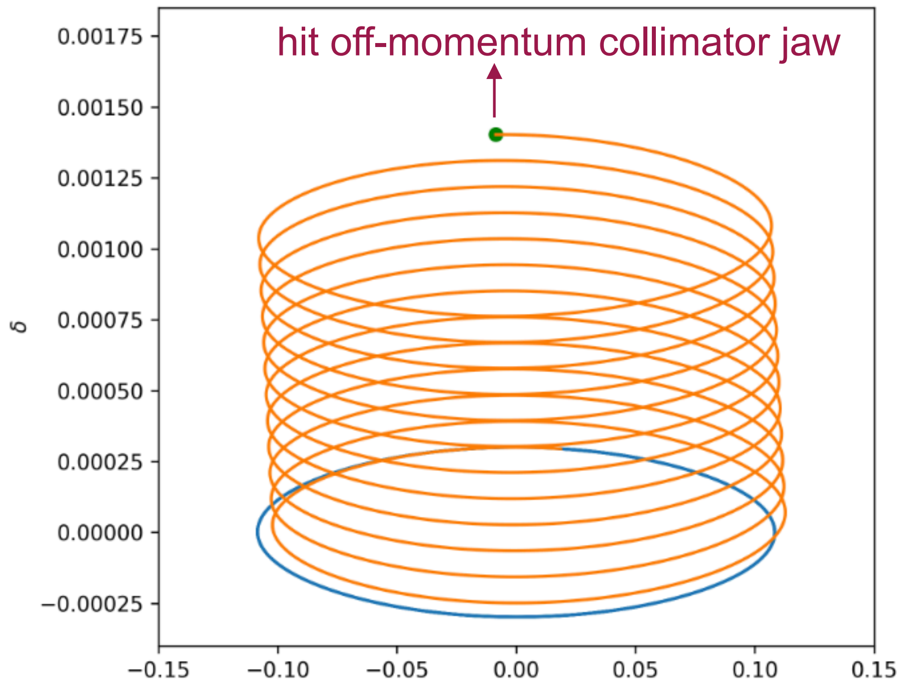
Simulated

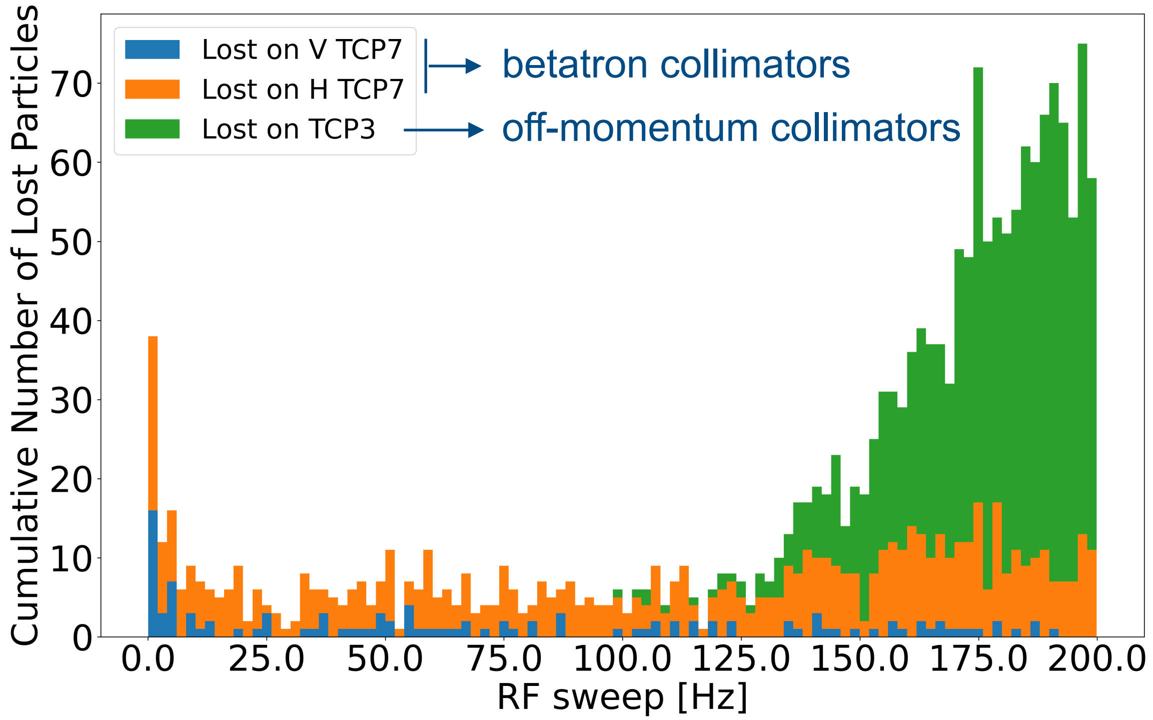


N. Triantafyllou, HB'23

RF Sweep to Simulate Off-Momentum Loss Maps

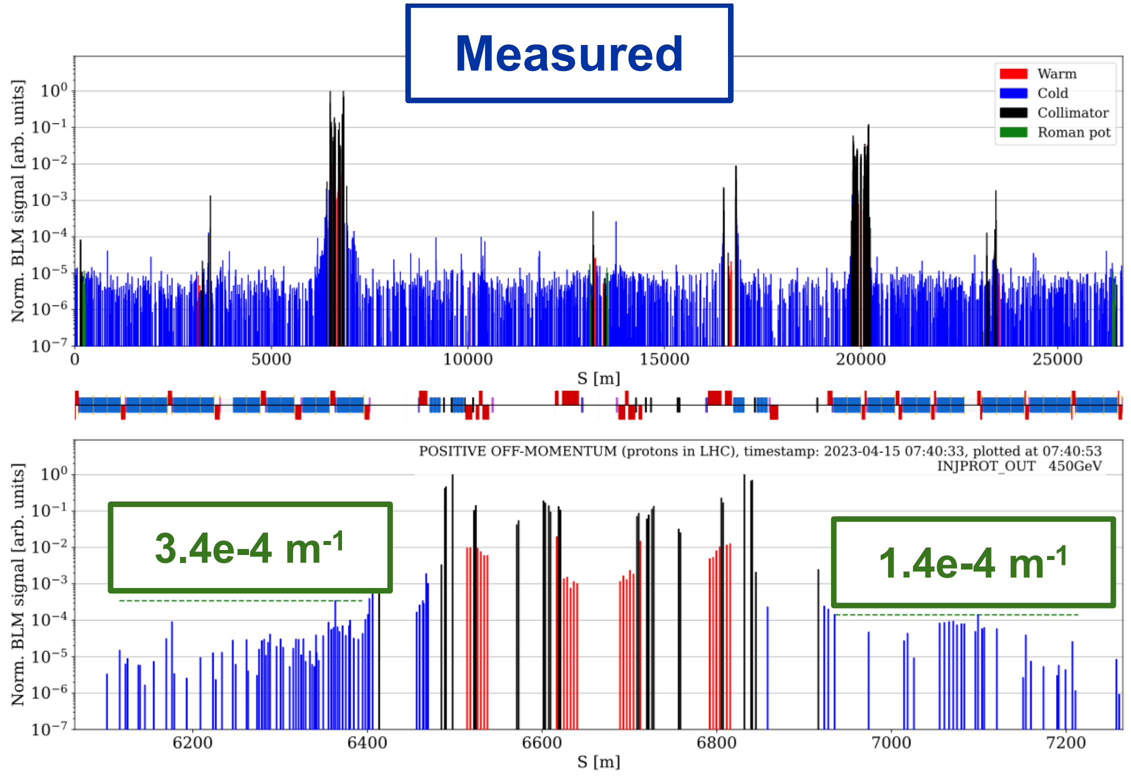
- Sweeping frequency of RF cavities moves separatrix up/down along delta
- Tracking assumes particles synchronous to longitudinal reference trajectory
  => need to **shorten/lengthen trajectory** to simulate sweep

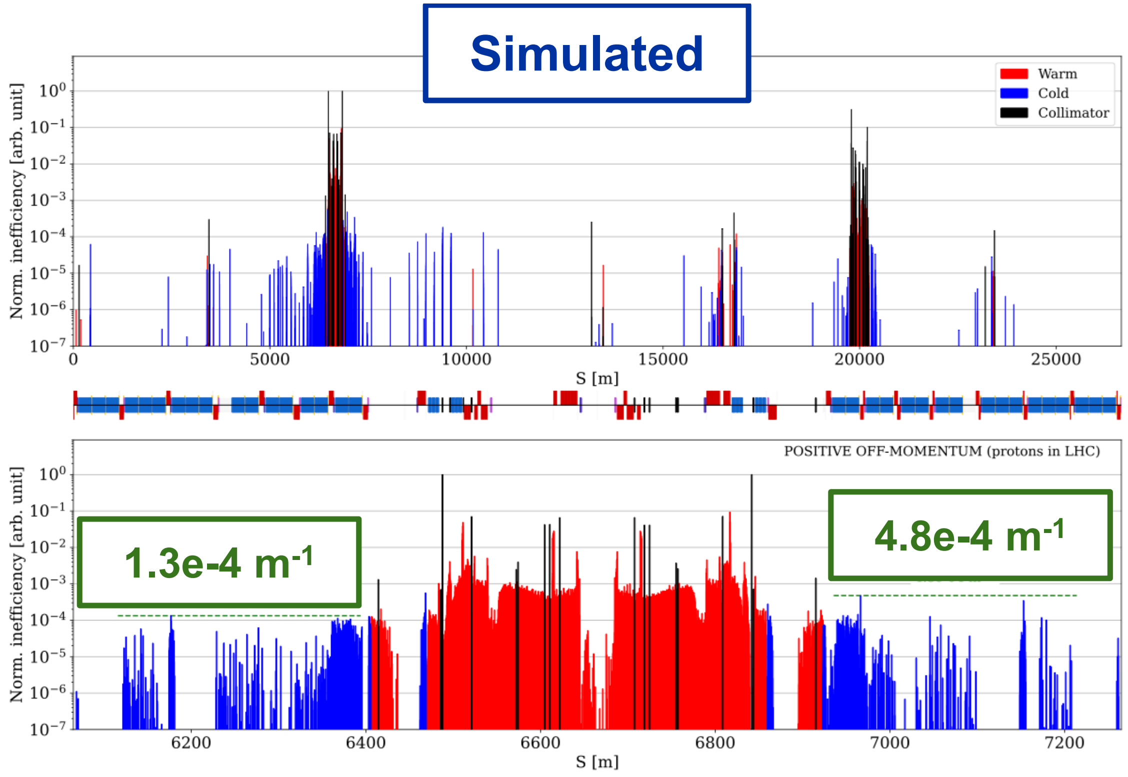




LHC Example: Off-Momentum Cleaning

- **Very good qualitative agreement** between measurement and simulation
  - Similar losses patterns in both collimator insertion regions





N. Triantafyllou, HB'23

Outline

Conclusions and Outlook

- Very active development, several new features

- Quickly becoming **new standard** for collimation team at CERN:
  - **Everest:** default for proton - matter interactions *(K2 discontinued)*
  - **Geant4:** only used with Xsuite *(SixTrack+Geant4 discontinued)*
  - **FLUKA:** first prototype *(SixTrack+FLUKA still in active use)*

- Future outlook:
  - **Everest:** material investigation, crystals exact channeling, and GPU-compatibility
  - **Geant4 & FLUKA:** complete integration into xcoll (natively in C)

Thank You

Cool stuff, right?

Thanks for your attention!

Feel free to use at will, comment/correct/contribute!

https://github.com/xsuite/xcoll

home.cern