XCOLL-BDSIM COUPLING FOR BEAM COLLIMATION[*]

B. Lindström[1,2], A. Abramov[2], S. T. Boogert[3], G. Broggi[2], R. Bruce[2], S. Gibson[1],
S. Redaelli[2], S. Solstrand[2], F. Van der Veken[2]
[1]John Adams Institute at Royal Holloway, University of London, Egham, UK
[2]CERN, Geneva, Switzerland
[3]Cockcroft Institute, Daresbury Laboratory, Daresbury, UK

Abstract

Xsuite is a comprehensive simulation toolkit for accelerator physics, with Xcoll serving as the dedicated module for collimation studies. These studies involve tracking particles through an accelerator and simulating their interactions with matter, taking into account non-linear elements and large betatron and off-momentum deviations. Particle-matter interactions can be modeled using an internal scattering model, Everest, or coupled to external codes. This paper presents the integration of Xcoll with the Geant4 libraries by utilizing the Beam Delivery Simulation (BDSIM) code. This coupling enables efficient tracking of diverse particle species through materials, along with realistic simulations of energy deposition and secondary particle production. Apertures are included to identify the precise location of particle losses throughout the machine.The implementation is designed to be flexible, with recent additions including support for heavy ions and re-entry safety, and with planned extensions such as crystal collimation and detailed collimator geometries.

INTRODUCTION

Xsuite [1] is a Python-based simulation framework developed at CERN to unify tools for various accelerator physics applications within a single modular environment. Its design emphasizes flexibility, with dedicated modules targeting different types of simulations. Xcoll [2–4] is the module dedicated to collimation studies, building on functionalities developed in previous frameworks. It provides tracking of particles and simulation of their interactions with matter, and includes an internal scattering model, Everest, which is a modern implementation of the legacy K2 code [5] formerly used in SixTrack [6,7]. In addition, Xcoll retains compatibility with external scattering models such as FLUKA [8–10] and Geant4 [11–13], in line with established collimation workflows.

The coupling to Geant4 is implemented via the Beam Delivery Simulation (BDSIM) code [14], enabling realistic simulation of particle–matter interactions, including energy deposition and secondary particle production. An initial integration of BDSIM with Xcoll was developed in 2023 in a separate branch, with a limited interface, enabling studies for FCC-ee [15–20] as well as benchmarks with measured data at SuperKEKB [21]. However, the initial implementation lacked re-entry safety due to incomplete memory cleanup on the Geant4 side [22], which posed a challenge for the Xsuite philosophy of supporting rapid prototyping and flexible simulation workflows.

To address this, a new architecture was introduced based on Remote Python Call (RPyC) [23], where BDSIM runs in a subprocess on a locally hosted server spawned automatically from within Xcoll. Upon completion of a simulation, the subprocess is terminated, ensuring proper memory cleanup. An additional update is the ability to return unstable particles from BDSIM to Xcoll for continued tracking. This is particularly important for heavy-ion studies, where a large fraction of secondaries would otherwise be lost. Decay processes for unstable particles are planned for a future release.

This paper announces the public release of the Xcoll-BDSIM coupling into the Xcoll main branch, and summarizes its current status, including recent developments such as support for heavy-ion tracking.

SIMULATION FLOW

Figure 1 provides an overview of the simulation workflow. Particles are generated by Xcoll and tracked through the accelerator line using Xtrack [1]. When a particle encounters a collimator, it is passed to the scattering model – in this case BDSIM – for interaction with the material using Geant4. Any surviving particles, including secondaries produced in the interaction, are returned to Xtrack for continued tracking. Energy deposited in collimators is recorded.

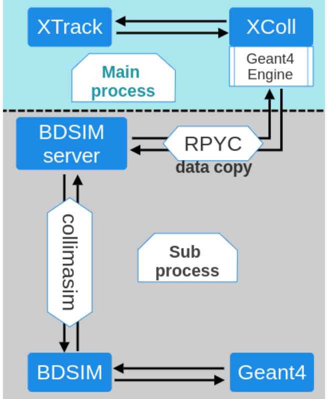

Figure 1: Flow chart illustrating the main components of the Xcoll-BDSIM coupling. See the text for details.

RPyC

Before running a simulation, the Geant4 collimators must be inserted in the Xsuite sequence and the Geant4 server initialized. This launches BDSIM in a subprocess on a lo-

---

[*] Work supported by the High Luminosity LHC project

Table 1: Performance Benchmark of the New RPyC Implementation Compared to the Previous Implementation (The four different test cases are detailed in the text. The left column shows the number of active particles.)

	pre-RPyC runtime [s]				RPyC incl. unstable part. runtime [s]			
test	1	2	3	4	1	2	3	4
10	1.4e-3	4.5e-2	7.8e-3	6.5	8.2e-3	4.9e-2	1.4e-2	71.2
100	4.1e-3	0.16	7.7e-2	20.1	1.1e-2	0.16	6.3e-2	95.9
500	1.7e-2	0.50	0.27	59.2	2.5e-2	0.50	0.30	300.3
1000	3.3e-2	1.0	0.52	97.7	4.1e-2	0.93	0.58	931.3
5000	0.16	5.0	2.6	467.1	0.25	4.7	2.8	1141.4
10000	0.33	10.0	5.2	896.3	0.38	9.5	5.6	2137.1
50000	1.6	50.0	26.5	4488.4	1.8	48.7	30.2	8798.4
100000	3.3	100.7	51.0	8939.2	3.6	102.9	64.8	17895.5

cally hosted server. Python functions are implemented on the server side to interface with the C++ layer of BDSIM via PyBind11 [24], and are made accessible to the client through RPyC. These functions provide an interface for collimator installation, particle injection, tracking, and data retrieval. The PyBind11 interface is implemented through Collimasim [25].

In the earlier implementation, BDSIM could directly access and modify the memory buffers of the particle ensemble. With the RPyC-based architecture, particles are instead serialized and transferred between processes. This introduces a performance overhead due to the copying of particle data to and from the C++ process. To reduce this overhead, the NumPy [26] arrays containing particle data are serialized into a single compressed binary block that is streamed between the client and the server.

CPU-Performance Benchmarking

To evaluate the performance impact of the new architecture, four benchmark cases were considered: (1) all particles miss a single collimator, (2) all particles interact with the collimator, (3) 90% of particles are dead and the remaining are split evenly between hitting and missing, and (4) a full LHC loss map simulation over 200 turns. Each test was run using a different number of particles and compared across two configurations: the pre-RPyC implementation and the RPyC implementation with unstable particles. Results are summarized in Table 1. All benchmarks were performed single-threaded on a system running Manjaro Linux 6.6.75-2 with an AMD Ryzen 7 7735U CPU.

In cases (1) and (2), the overhead introduced by RPyC depends on whether particles interact with the collimator. When all particles hit the collimator, the overhead is negligible. When all particles miss, the overhead is significant for small particle numbers – up to a factor of six for ten particles – but diminishes with larger samples, reaching approximately 10% at 100,000 particles (case (1), pre-PRYC faster).

In case (3), a large discrepancy appears due to how dead particles are handled. Although dead particles are not tracked, they are still transferred across the client-server interface. In the pre-RPyC implementation, performance scales with the number of active particles, whereas the RPyC implementation transfers all particles regardless of state. In this test, the RPyC implementation was roughly 27% slower.

In practical loss map simulations, case (4), each collimator will encounter a mixture of active and inactive particles. For a typical simulation involving 50,000 particles, the new implementation is about twice as slow. Most of this performance penalty stems not from RPyC itself, but from the need to copy particle data between processes. Future optimizations, such as pre-selecting only active particles for transfer, could significantly reduce the overhead. Additionally, including unstable particles in the simulation contributes to the total runtime. For example, a 10,000-particle simulation with unstable particles disabled completed in 1831.6 s.

LOSS MAP SIMULATION EXAMPLES

While BDSIM supports the tracking of any particle defined in the Geant4 libraries, the Xcoll-BDSIM coupling, via the Collimasim interface, initially supported only (anti-)protons, positrons and electrons. With the release of Xcoll-BDSIM, support for heavy ions is added, including stable and unstable isotopes (currently excluding nuclear isomers). A comparison between the BDSIM and FLUKA couplings for protons was presented in [3], while a corresponding study for heavy ions is planned for future work.

Figure 2 shows a typical LHC loss map with protons, focused on the betatron collimation insertion (IR7). A comparison is made with the pre-RPyC implementation, with 10^7 particles simulated in each setup. The results are statistically consistent.

To demonstrate heavy-ion capabilities, a beam of ^{192}Os was tracked through the LHC. Although heavy-ion collimation typically employs crystal collimators [27], the standard proton collimation setup was used in this test. The results are shown in Fig. 3, with the top plot showing full-ring losses and the bottom a zoom into the betatron cleaning insertion, IR7. The loss pattern is characteristic of heavy-ion collimation [28, 29]: the highest losses occur on secondary collimators downstream of the primaries, along with significant leakage into the dispersion suppressor. This is a known

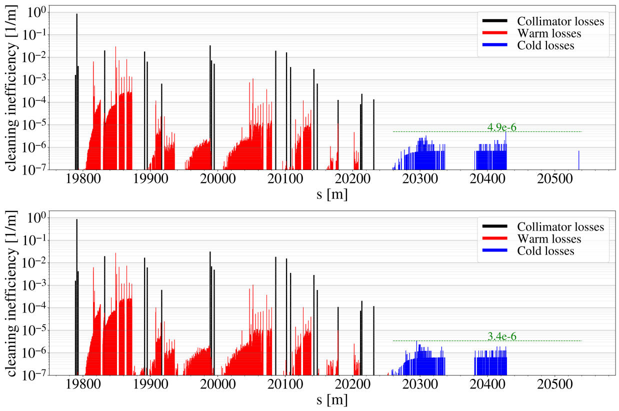

Figure 2: Beam 1 horizontal betatron loss map with protons. Top: pre-RPyC Bottom: RPyC

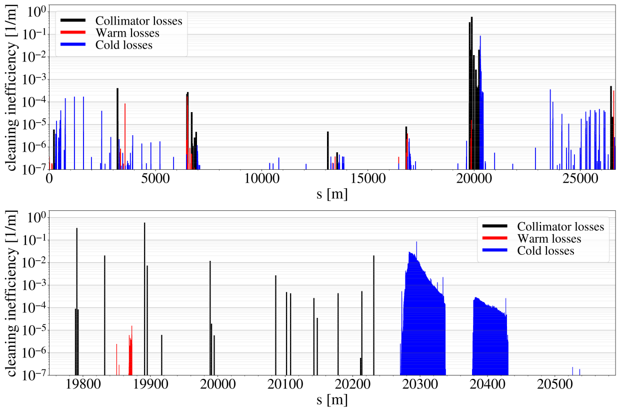

Figure 3: Beam 1 horizontal betatron loss map with ^{192}Os. Top: full LHC ring. Bottom: zoom into IR7.

limitation of the present system for heavy-ion collimation, which called for the deployment of crystal collimation.

Figure 4 shows a histogram of isotopes produced during the simulation. The prominent peak corresponds to the initial isotope (^{192}Os), followed by a smaller peak for ^{191}Os, attributed to electromagnetic dissociation [29]. Beyond this, the distribution is relatively flat, with minor peaks toward both ends. The lower bound (Z = 11, A = 27) arises from the relative energy cut: in BDSIM, the cut is applied to the kinetic energy of the entire particle. Lighter isotopes are less likely to exceed this threshold and are therefore suppressed. Lowering the energy cut would allow inclusion of lighter fragments, such as hydrogen isotopes, at the expense of increased simulation time.

UPCOMING DEVELOPMENT

Several updates are planned for future releases. The main feature under development is support for crystal collimation. Additional improvements include the ability to simulate more complex collimator geometries and the integration of generic magnetic field maps through the coupling.

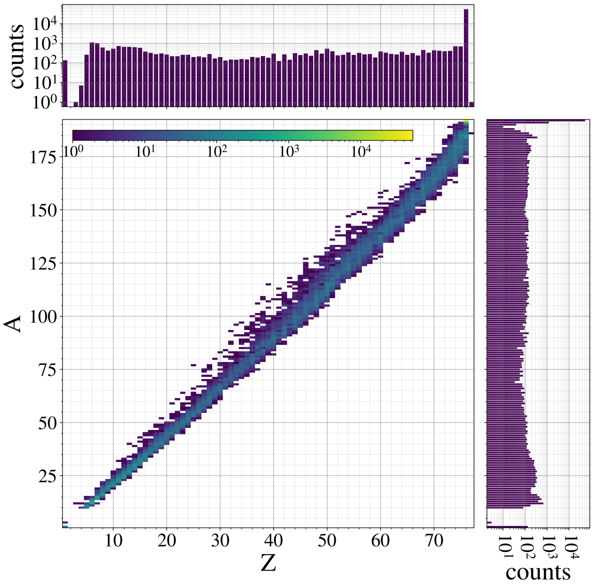

Figure 4: Histogram of isotopes produced in the heavy-ion simulation with ^{192}Os.

Handling of unstable particle decay, currently unsupported in Xcoll, is also planned. On the performance side, two key developments are anticipated: reducing the volume of data exchanged between Xcoll and BDSIM, and replacing the Collimasim interface with native C bindings integrated directly into Xcoll.

A general-purpose Python interface to BDSIM is also under development [30], and may serve as a future foundation for the Xcoll-BDSIM coupling.

CONCLUSIONS

The coupling of BDSIM to Xcoll has been under development for some time, and although initially available only in a separate branch with a limited interface, it has already enabled valuable studies, including for FCC-ee. With this release, the interface has been streamlined, made more user-friendly, and integrated into the main branch of Xcoll to ensure long-term support and maintainability. The implementation was also made re-entry safe by running BDSIM in a subprocess – an essential workaround to avoid memory issues on the Geant4 side when running multiple simulations in sequence.

In addition, support for tracking unstable particles and heavy ions was introduced, extending the scope of simulations that can be performed. While the new architecture adds some computational overhead due to data transfer, the performance is acceptable for typical use cases, and further optimizations are planned. Upcoming developments include support for crystal collimation, particle decay, and tighter integration of the BDSIM interface, aiming to further expand the range of applications and improve usability.

REFERENCES

[1] G. Iadarola *et al.*, “Xsuite: An Integrated Beam Physics Simulation Framework”, in *Proc. HB’23*, Geneva, Switzerland, 2024, pp. 73–80. `doi:10.18429/JACoW-HB2023-TUA2I1`

[2] D. Demetriadou, A. Abramov, G. Iadarola, and F. F. Van der Veken, “Tools for integrated simulation of collimation processes in Xsuite”, in *Proc. IPAC’23*, Venice, Italy, May 2023, pp. 2801–2804. `doi:10.18429/JACoW-IPAC2023-WEPA066`

[3] F. Van der Veken *et al.*, “Recent Developments with the New Tools for Collimation Simulations in Xsuite”, in *Proc. HB’23*, 2024, pp. 474–478. `doi:10.18429/JACoW-HB2023-THBP13`

[4] F. F. Van der Veken, “Introducing Xcoll: A Streamlined Approach to Collimation and Beam Loss Simulations Using Xsuite”, presented at ICAP’24, Seeheim, Germany, Oct. 2024, unpublished. `https://indico.gsi.de/event/19249/contributions/82656/attachments/48844/70973/ICAP_Xcoll.pdf`

[5] S. Redaelli, *ICFA Mini-Workshop on Tracking for Collimation in Particle Accelerators*. Geneva: CERN, 2018. `doi:10.23732/CYRCP-2018-002`

[6] R. D. Maria *et al.*, “SixTrack Version 5: Status and New Developments”, in *Proc. IPAC’19*, Melbourne, Australia, 2019, pp. 3200–3203. `doi:10.18429/JACoW-IPAC2019-WEPTS043`

[7] R. De Maria *et al.*, SixTrack – 6D Tracking Code. `http://sixtrack.web.cern.ch/SixTrack/`

[8] C. Ahdida *et al.*, “New capabilities of the FLUKA multipurpose code”, *Front. Phys.*, vol. 9, 2022. `doi:10.3389/fphy.2021.788253`

[9] G. Battistoni *et al.*, “Overview of the FLUKA code”, *Ann. Nucl. Energy*, vol. 82, pp. 10–18, 2015. `doi:10.1016/j.anucene.2014.11.007`

[10] FLUKA website, 2020. `https://fluka.cern`

[11] J. Allison *et al.*, “Recent developments in geant4”, *Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip.*, vol. 835, pp. 186–225, 2016. `doi:10.1016/j.nima.2016.06.125`

[12] J. Allison *et al.*, “Geant4 developments and applications”, *IEEE Trans. Nucl. Sci.*, vol. 53, no. 1, pp. 270–278, 2006. `doi:10.1109/TNS.2006.869826`

[13] S. Agostinelli *et al.*, “Geant4—a simulation toolkit”, *Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip.*, vol. 506, no. 3, pp. 250–303, 2003. `doi:10.1016/S0168-9002(03)01368-8`

[14] LJ. Nevay *et al.*, “Bdsim: an accelerator tracking code with particle–matter interactions”, *Comput. Phys. Commun.*, vol. 252, p. 107200, 2020. `doi:10.1016/j.cpc.2020.107200`

[15] A. Abramov *et al.*, “Collimation simulations for the FCC-ee”, *JINST*, vol. 19, T02004, 2024. `doi:10.1088/1748-0221/19/02/T02004`

[16] A. Abramov *et al.*, “Studies of layout and cleaning performance for the FCC-ee collimation system”, in *Proc. IPAC’23*, Venice, Italy, 2023, pp. 356–359. `doi:10.18429/JACoW-IPAC2023-MOPA128`

[17] A. A. G. Broggi and R. Bruce, “Beam dynamics studies for the FCC-ee collimation system design”, in *Proc. IPAC’23*, Venice, Italy, 2023, pp. 360–363. `doi:10.18429/JACoW-IPAC2023-MOPA129`

[18] G. Broggi, “Tracking studies for the FCC-ee collimation system design”, *Nuovo Cimento C*, vol. 47, 2024. `doi:10.1393/ncc/i2024-24273-x`

[19] G. Broggi *et al.*, “Optimizations and updates of the FCC-ee collimation system design”, in *Proc. IPAC’24*, Nashville, Tennessee, USA, 2024, pp. 1192–1195. `doi:10.18429/JACoW-IPAC2024-TUPC76`

[20] G. Broggi *et al.*, “Beam losses due to beam-residual gas interactions in the FCC-ee”, presented at IPAC’25, Taipei, Taiwan, 2025, paper MOPM036, this conference.

[21] G. Broggi *et al.*, “Comparison of Xsuite simulations with measured backgrounds at SuperKEKB”, presented at IPAC’25, Taipei, Taiwan, 2025, paper MOPM035, this conference.

[22] S. Boogert, Rerunning simulation – Geant4 Forum Discussion, 2024. `https://geant4-forum.web.cern.ch/t/rerunning-simulation/13507`

[23] T. Filiba and contributors, RPyC – Remote Python Call, 2024. `https://github.com/tomerfiliba-org/rpyc`

[24] W. Jakob, J. Rhinelander, and D. Moldovan, pybind11 – seamless operability between C++11 and Python, 2017. `https://github.com/pybind/pybind11`

[25] A. Abramov and G. Broggi, collimasim – C++/Python interface for BDSIM coupling, 2025. `https://gitlab.cern.ch/anabramo/collimasim`

[26] C. R. Harris *et al.*, “Array programming with NumPy”, *Nature*, vol. 585, pp. 357–362, 2020. `doi:10.1038/s41586-020-2649-2`

[27] S. Redaelli *et al.*, “Crystal collimation of heavy-ion beams at the large hadron collider”, *Phys. Rev. Accel. Beams*, vol. 28, no. 5, p. 051001, May 2025. `doi:10.1103/PhysRevAccelBeams.28.051001`

[28] M. D’Andrea, “Applications of Crystal Collimation to the CERN Large Hadron Collider (LHC) and its High Luminosity Upgrade Project (HL-LHC)”, Ph.D. thesis, Padua U., 2021. `https://cds.cern.ch/record/2758839`

[29] R. Cai, “Studies of crystal collimation for heavy ion operation at the LHC”, Ph.D. thesis, Ecole Polytechnique, Lausanne, 2024. `http://cds.cern.ch/record/2896673`

[30] W. Shields *et al.*, “Updates and developments of BDSIM”, presented at IPAC’25, Taipei, Taiwan, Jun. 2025, paper WEPS039, this conference.