PROGRESS AND DEVELOPMENTS OF BEAM DELIVERY SIMULATION (BDSIM)

S. Alden[1], W. Shields [1], A. Keyken[1], M. Pereira[1], N. Nishi[1],
Royal Holloway University of London, United Kingdom
S. Boogert[2], M. Deniaud[2], D. Christie[2], J. Burvill[3], P. Mooney[3],
University of Manchester, Manchester, United Kingdom
M. Mannie-Corbisier, Université Libre de Bruxelles, Brussels
B. Lindstrom, L. Nevay, R. Kieffer,
European Organization for Nuclear Research (CERN), Geneva, Switzerland
P. Jurj, R. Kamath, Imperial College London, London, United Kingdom
C. Rogers, Science and Technology Facilities Council, Swindon, United Kingdom
[1] also at The John Adams Institute for Accelerator Science, London, United Kingdom
[2] also at Cockcroft Institute, Daresbury, United Kingdom
[3] also at European Organization for Nuclear Research (CERN), Geneva, Switzerland

Abstract

Beam Delivery Simulation (BDSIM) is a Monte Carlo program (written in C++) that creates a 3D radiation transport model designed for the simulation of accelerators and beamline modelling. BDSIM uses Geant4 for precise particle-matter interactions combined with particle-tracking through 3D geometries of accelerators and their environments. All particle species are tracked, allowing for studies for collimation, beam losses, secondary radiation generation, and dosimetry. BDSIM allows for detailed customisation, with numerous applications in the design of high-energy physics facilities, medical beamlines, particle detection experiments, and novel acceleration experiments. Recent developments to BDSIM are presented, including; coupling between BDSIM and external tracking codes, updates to the muon cooling modelling, updates to DICOM file handling, applications of BDSIM for FCC-ee, scintillation and synchrotron radiation options for secondary control, and general updates.

INTRODUCTION AND GENERAL UPDATES

BDSIM is a Monte Carlo particle-tracking code that uses the Geant4 toolkit to construct 3D models of accelerator and their environment. It is capable of modelling particle-matter interactions and is suited for studies in a wide range of accelerator applications [1, 2]. BDSIM is well supported on multiple platforms, including Conda, Docker, Apptainer, and compilation from source on all common operating systems. BDSIM developers welcome any interested users and contributors who will be supported by the BDSIM community [3].

Sparse Histogram Storage

In BDSIM, output histograms are stored on a per-event basis in a dense format, where the values of all bins are recorded even if they are zero. For situations where there is a low occupancy of the histogram at each event, this can be a source of significant inefficiency in both memory and storage as many zeroes are unnecessarily stored. An option has been developed to instead use sparse histograms which only store the positions of *filled* bins as a vector. This reduces the memory and storage requirements of, for example, scoring meshes and energy loss histograms. A comparison of the size of ROOT files storing per-event histograms is shown in Figure 1, where a significant difference is visible between the storage methods at low occupancies.

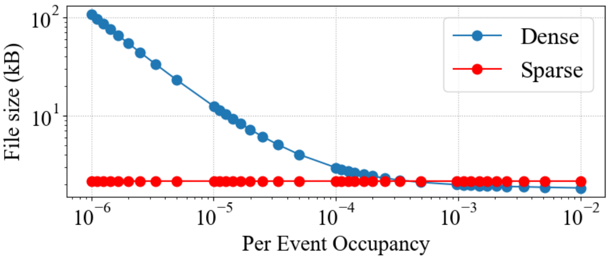

Figure 1: Per-event size of ROOT files output by BDSIM against the occupancy of the stored histograms, comparing between dense and sparse methods of storage.

Sub-Element Scoring

To score a volume in BDSIM, a scoring mesh, limited to a box or cylinder, is typically overlayed across an element within the model. Most elements, however, are composed of multiple volumes and can therefore be decomposed into sub-elements, representing the finest level of 3D geometry. A method of scoring across these specific sub-elements, using their volume as the scoring volume, is being implemented. Each physical volume is registered at construction to an integer identifier and each energy deposit is attributed to a sub element via this identifier. The total energy deposited in each sub-element can then be aggregated. An example model FCC-ee Beamstrahlung dump, shown in Figure 2,

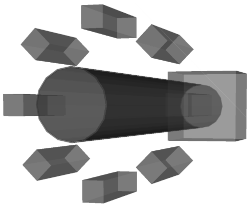

Figure 2: Pyg4ometry model of a prototype FCC Beamstrahlung Dump Monitor with eight lead glass bars surrounding the beam pipe.

is composed of bars, an entry and exit windows, a beam pipe and the dump, all of which can be scored separately. This allows users to score energy across each individual component of a model, regardless of complexity, shape and position within the model.

Python Interface

The ability to control BDSIM directly from python allows Geant4/BDSIM particle matter interactions to be used directly within beam tracking codes implemented in or having a good interface with python, like Ocelot [4], RF-Track [5] and Xsuite [6]. The latter interface extends the BDSIM coupling already present in Xcoll by adding additional functionality and flexibility [7]. In addition to allowing a wider range of simulations, the python interface allows BDSIM to be interfaced to powerful external optimisation software, e.g. XOpt. Almost all of the externally facing C++ classes used to define a simulation in BDSIM now have pybind11 based wrappers to allow users to directly call BDSIM from within these beam dynamics codes. Traditional BDSIM usage requires the user to write a text file in GMAD format which BDSIM reads. This new python interface allows users to programatically define, run and interpret the results of a simulation without the need of writing any files.

Listing 1: Python code (incomplete) to create a BDSIM element which can be used in an RF-Track simulation

```
import RF_Track as RFT
import bdsim

bds_link = bdsim.BDSLinkTrackerInterface.
    ↪ GetInstance("./trackerInterface.gmad"
    ↪ ,11,1*bdsim.clhep.GeV,0.01,1,0,True);

# create BDSIM collimator element
e = bdsim.Element();
e.name = "collimator"
e.type = bdsim.ElementType.RCOL
e['l'] = 0.05
e['xsize'] = 0.0
e['ysize'] = 0.0
e['material'] = "G4_Fe"
e = bdsim.rftrack.BDSIMElement(e);

# create RF-Track elements
d1 = RFT.Drift(0.5)
d2 = RFT.Drift(0.5)

# create RF-Track lattice
lat = RFT.Lattice()
lat.append(d1)
lat.append(e) # BDSIM element
lat.append(d2)

# track bunch b0
b1 = lat.track(b0);
```

MUON COOLING CHANNELS

Modelling muon cooling channels presents significant computational challenges due to the presence of fringe-dominated solenoids, dipoles, and RF cavities with overlapping fields [8]. Solenoids are usually modelled as blocks, each comprising a stack of n concentric sheets. The field of each sheet takes the form of a generalized complete elliptic integral and is evaluated using Bulirsch's algorithm, with distance-based early exit [9]. Dipole fringe fields are typically evaluated analytically via Enge-style exponential functions. To improve the flexibility and performance of these simulations, we introduce three upgrades [10].

First, explicit transverse offsets and Euler rotations can now be carried by each solenoid. This permits arbitrary placements and tilts, enabling investigations into the effects of physical misalignments on beam dynamics and emittance.

Second, the field of a solenoid block or dipole can optionally be precomputed on a grid and evaluated at runtime by interpolation, with user-supplied resolution and order. Grids are constructed once per unique block geometry and cached to minimise memory during runtime. The per-element computational cost is thereby collapsed to a single branch-free, vectorisable interpolation.

Third, spatial bins are mapped to overlapping field elements via a one-dimensional spatial index along the z-axis. This reduces element queries from an exhaustive $\mathcal{O}(N)$ search to a $\mathcal{O}(1)$ hash map lookup at each tracking step.

Combined, per-step field evaluations are reduced from $\mathcal{O}(nN)$ analytic calls to $\mathcal{O}(k)$ interpolations, where k represents the local field overlap, which is independent of the total lattice size. While the realized speed-up is determined by user-supplied parameters and the feasibility of amortizing field map construction over the number of simulated particles, order-of-magnitude performance gains have been observed for simulations of interest.

MEDICAL MODELLING WITH DICOM FILES

BDSIM supports loading of computed tomographic (ct) images via DICOM files in a `ct` element. Such files contains voxelised geometries which are converted from native hounsfield units into densities required for defining Geant4 materials. The most convenient method to record dose deposition in a `ct` element is with a scoring mesh. Now, a scoring mesh can be created automatically that matches the position, number of voxels, and voxel dimensions of the DICOM geometry. The scorer is defined as standard in the BDSIM model and passed to the `ct` element via the `dicomScorer` parameter. Automatic scoring mesh generation can be optionally turned off with the `createDicomScorerMesh` option. In addition, a number of fixes for the ct element have been implemented, including length checks, overlap checks for multiple DICOM slices, and the splitting of material densities is now controlled by the `changeDicomMaterialDensity` option.

SYNCHROTRON RADIATION ENERGY CUTS

Synchrotron radiation has high production in low energy ranges which can dominate and become computationally prohibitive. This is highly inefficient in models in which only higher energy photons survive transmission through model material [11]. An energy range cut has been added to BDSIM to reduce the number of secondaries tracked in simulation by deleting gammas below this energy at production. Multiple synchrotron radiation photons can be produced in a single step and those that survive are modified to carry a weight which includes the number of photons that were cut. Figure 3 shows the resulting spectrum from the generation of synchrotron radiation through a short FODO lattice. The energy cut was applied at 10 eV, 100 eV, 1000 eV and is compared to the simulation without the energy cut, where higher counts of lower energy photons are seen.

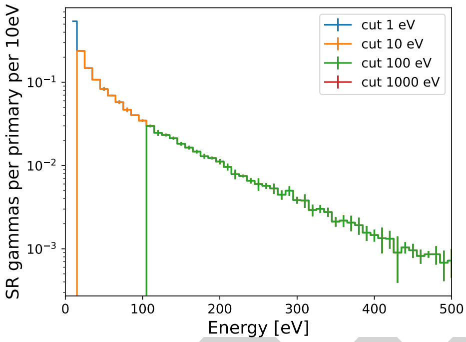

Figure 3: Energy spectra of synchrotron radiation photons with energy cuts of 10 eV to 1000 eV.

OPTICAL PHYSICS : SCINTILLATION

Geant4 optical physics provides a robust framework for generating and propagating optical photons within particle-tracking simulations. In BDSIM this enables simulation of both signal and background, from source to detector. Geant4 scintillation can be computationally expensive for high-yield materials. A biasing scheme that reduces the number of optical photons generated and tracked has been added to BDSIM. The reduction is applied per step, with suppressed photons recovered via event weighting, allowing reconstruction of the spectrum. In an example model, a spherical sampler around a Yttrium Aluminium Garnet (YAG:Ce) scintillator recorded all optical photons generated from impact by 1000 electrons. Figure 4 shows the relative residuals for the biased energy spectrum of the optical photons produced for a reduction factor of 10, and 100.

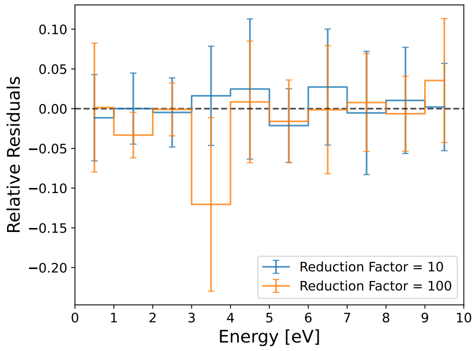

Figure 4: Residuals from optical photon energy spectrum comparing a bias factor of 10 and 100 to the unbiased model.

FCC POLARIMETER

Developments in the laser-particle model in BDSIM have allowed the simulation of inverse-Compton scattering (ICS), including the effects of particle and laser polarisation. This is being applied to simulate the polarimeter for the FCC-ee, where the average polarisation of the beam will be deduced from the angular distribution of scattered electrons and photons. The resultant distribution predicted by BDSIM has been bench-marked against a toy model based on the ICS theory described in [12]. A comparison in the predictions of the vertical profile of the scattered electrons in the polarimeter is shown in Figure 5, and a clear agreement is visible. BDSIM supports a complete model of the polarimeter, including the interaction with matter necessary to simulate the converter target for the photon sensor.

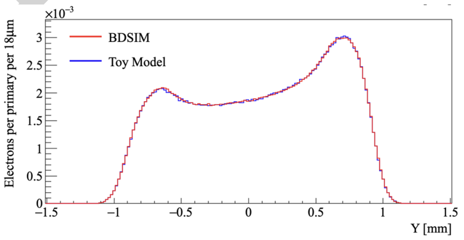

Figure 5: Distribution of scattered electrons in the FCC-ee inverse-Compton Polarimeter predicted by BDSIM, bench-marked against toy model.

CONCLUSION

The most recent updates to BDSIM have been presented. BDSIM is actively supported and developed, suited to a diverse range of applications. The BDSIM collaboration welcomes contributors and users.

REFERENCES

[1] Allison, J. *et al.*, "Recent developments in Geant4", *Nucl. Instrum. Methods Phys. Res. A*, vol. 835, pp. 0168-9002, 2016.

doi:10.1016/j.nima.2016.06.125

[2] Nevay, L.J. *et al.*, ”BDSIM: An accelerator tracking code with particle-matter interactions”, *Comput. Phys. Commun.*, vol. 252, pp. 107200, 2020. doi:10.1016/j.cpc.2020.107200

[3] BDSIM collaboration URL : *https://bdsim-collaboration.github.io/web/index.html*

[4] Tomin, S. , ”OCELOT as a Framework for Beam Dynamics Simulations of X-Ray Sources”, in *Proc. IPAC'17*, Copenhagen, Danemark, Oct. 2017, doi:10.18429/JACoW-IPAC2017-WEPAB031

[5] Latina, A. , ”The Tracking Code RF-Track and Its Application”, in *Proc. HB'23*, Geneva, Switzerland, pp. 245–248, Oct. 2024, doi:10.18429/JACoW-HB2023-WEA3C1

[6] Iadarola, G. *et al*, ”Xsuite: An Integrated Beam Physics Simulation Framework”, in *Proc. HB'23*, Geneva, Switzerland, pp. 73–80, Oct. 2023, doi:10.18429/JACoW-HB2023-TUA2I1

[7] Lindstrom, B. *et al.*, “Xcoll-BDSIM coupling for beam collimation”, in *Proc. IPAC'25*, Taipei, Taiwan, Jun. 2025, pp. 667-670. doi:10.18429/JACoW-IPAC2025-MOPS030

[8] Rogers, C. Jurj, P. B. Kamath, R. Pasternak, J., “Development of BDSIM simulation (1.0)”. Zenodo (2025). doi:10.5281/zenodo.14943349

[9] Bulirsch, R., “Numerical Calculation of Elliptic Integrals and Elliptic Functions”, *Numerische Mathematik*, 1965, 7, pp. 78-90. doi:10.1007/BF01397975

[10] Kamath, R. *et al.*, “Implementation and simulation of a rectilinear cooling channel in BDSIM”, in *Proc. IPAC'25*, Taipei, Taiwan, Jun. 2025, pp. 647–650. doi:10.18429/JACoW-IPAC2025-MOPS024

[11] Bodenstein, R. *et al.*, ”Status Update of Permanent Magnet Radiation Resiliency Studies at CEBAF”, presented at the IPAC'26, Deauville, France, May 2026, paper TUP7370, this conference.

[12] Muchnoi, N.Yu., “Electron beam polarimeter and energy spectrometer”, *J. Instrum.*, vol. 17, no. 10, p. P10014, 2022. doi:10.1088/1748-0221/17/10/P10014