BDSIM: Automatic Geant4 Models of Accelerators

L. J. Nevay*[1], A. Abramov[1], S. T. Boogert[1], L.C. Deacon[2], H. Garcia-Morales[1], S. M. Gibson[1], R. Kwee-Hinzmann[1], W. Shields[1], J. Snuverink[1], and S. Walker[1]

[1]John Adams Institute at Royal Holloway, University of London, Egham, UK
[2]University College London, London, UK

Abstract

BDSIM is a program that uses a suite of high energy physics software including Geant4, CLHEP & ROOT, to seamlessly track particles through a 3-dimensional accelerator model utilising the full range of particles and physics processes from Geant4. BDSIM was originally developed to simulate linear colliders such as the International Linear Collider (ILC), but has more recently been extended for application to the Large Hadron Collider (LHC), and generally to storage rings. The significant modernisation and revision of the implementation undertaken from 2013 to facilitate these simulations is presented here along with an example of the capabilities.

Keywords

Radiation simulation; tracking; energy deposition; beam loss; collimation.

1 Introduction

To ensure the expected operation of an accelerator, any beam losses must be accurately characterised for a variety of reasons. Firstly, to avoid damage to the accelerator and surrounding equipment as well as minimise radioactivation. Where superconducting magnets are used, cryogenic heat loads due to beam loss must be accurately simulated to ensure the magnets do not quench. Secondly, it's prudent to simulate the beam losses that reach detectors to understand the background in the measurements they are making. Lastly, the cleaning performance of a collimation system can determine the success of the machine operation and detector performance and so must also be simulated.

The first aspect of beam loss simulation is a tracking simulation that records the loss of a particle when it meets the definition of an aperture. Codes that are used to design accelerators such as MAD-X [1] typically include tracking codes, such as the Polymorphic Tracking Code (PTC) [2], for this purpose. Whilst such a simulation may stop at this point, a high energy particle will not and it will undergo various discrete and continuous processes that lead to energy loss and secondary particle production as the initial *primary* particle travels until it exhausts its kinetic energy and comes to rest. Depending on the energy of the primary particle and material it interacts with, the length scale of this process and final rest position may be localised to a single accelerator component or may be far away in an apparently unrelated part of the accelerator after passing through many nonlinear fields.

For example, in a proposed high energy linear collider such as the International Linear Collider (ILC) [3], loss of an electron or positron from the beam can result in the production of a high energy (up to $E \sim 230$ GeV) muon [4]. Such a muon may travel far through the accelerator, surrounding tunnel, soil and shielding, and penetrate the detector. It may also decay during its passage through these various components. In the case of a proton collider such as the LHC, the primary concern is machine protection. The LHC is designed to have a stored beam energy of 350 MJ [5], and given the superconducting magnets

*laurie.nevay@rhul.ac.uk

https://doi.org/10.23732/CYRCP-2018-002.45.

used to guide the circulating beams must remain at cryogenic temperatures, the loss of only a very small fraction of a single bunch would induce a superconducting quench and result in damage to the machine.

BDSIM [6–9] is an open-source C++ program that allows such scenarios to be simulated by building a 3D model of an accelerator. The model is constructed from its optical lattice description using a library of generic geometries for each type of magnet used in an accelerator. The model is built using Geant4 [10], a Monte-Carlo framework to simulate the passage of particles in matter. BDSIM provides the appropriate fields and numerical integrators for each type of commonly used accelerator magnet allowing particles of a chosen type and distribution to be tracked through the model in a manner similar to a conventional tracking code. Geant4 provides an extensive library of continuous and discrete processes that are used through the simulation in addition to tracking through electro-magnetic fields [10, 11].

Should a high energy particle hit the beam pipe for example, secondary particles may be produced and tracked along with the possibly surviving primary particle throughout the model. BDSIM therefore provides the ability to simulate the energy deposition and the final stopping location of particles as well as the beam loss locations. The expected flux of all particles can be simulated at given locations allowing detector background from the accelerator to be simulated. Moreover, the surviving fraction and distribution of particles from any intercepting device, such as a collimator, can be simulated, which would not normally be possible with a conventional tracking code. The ability to produce and track all secondary particles allows the accompanying energy deposition and dose to be predicted. Depending on the application and information required, suitable cuts in particle properties (kinetic energy, location, species, etc.) can be used to minimise the computational time required as the full simulation with all secondary particles is considerably longer than a tracking simulation of primary particles only.

For accurate accelerator tracking, transfer maps based on the solution to equations of motion are used to provide symplectic thick-lens tracking for linear components. For non-linear elements, numerical integrators are used to calculate the motion due to the Lorentz force. In both cases, the integrators are constructed with a strength parameter based on the nominal beam particle and rigidity that is scaled for each incoming particle. These therefore ignore the field as provided through Geant4. In all cases, Geant4 chooses a step length for the particle to be advanced by considering geometrical intersections, the occurence of discrete physics processes or to match the required degree of accuracy in tracking.

Each integrator class in BDSIM is capable of handling all particle types (stable and unstable) and rigidities and so provides extra funcitonality beyond just the tracking algorithm. In the case where the transfer maps would no longer apply, such as non-paraxial particles, the integrator falls back to using a Geant4 4^{th} order Runge-Kutta integrator that uses the field provided for that element by BDSIM. Special provision is made to efficiently deal with spiralling particles as may happen with low energy secondary particles in strong magnetic fields.

The expanded BDSIM geometry library for both aperture and magnet yoke geometry are presented alongside new tunnel geometry provision. Revision of the use of the Geant4 physics processes is discussed, followed by a description of the output format and coordinate systems used. Finally, a hypothetical beam line example is described with sample output.

2 Geometry Library

BDSIM originally only provided circular beam pipes with cylinders of iron surrounding them to represent the yoke of the magnet [7]. As the aperture is the first point where a particle leaves the vacuum, it is critical that this be an accurate part of the simulation. Furthermore, whilst a cylinder of a single material is suitable for a theoretical study of an accelerator at the design stage, more accurate geometry is required for magnets for existing accelerators. An increased variety of geometry however, leads to many potential combinations of apertures and magnet geometry styles. The code to produce such geometry should avoid duplication and should be easily extensible, so the new geometry code implemented in BDSIM is based on the *factory* design pattern [12]. Here, there is a factory class for each style of aperture that produces

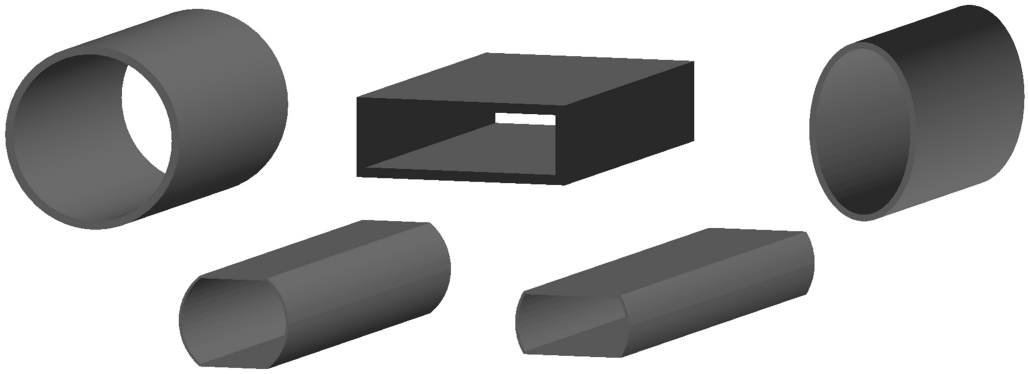

Fig. 1: Selection of aperture models in BDSIM as seen using the visualisation system. From top-left, clockwise: *circular, rectangular, elliptical, rectellipse* and *lhcscreen*.

either a straight or angled section of that aperture. Other classes, such as a magnet construction class, can use the aperture factory to build the required section of beam pipe. Like a real factory, the aperture factory does not retain ownership of its product.

Geant4 provides a variety of classes for primitive shapes that can be used in a constructive solid geometry (CSG) scheme to produce almost any desired shape [10]. The geometry factories written for BDSIM provide parameterised geometry using the Geant4 primitives to make a Geant4 model saving the user from writing a large amount of C++.

With many possible combinations of aperture and magnet styles, we must ensure that all work together without producing geometrical overlaps. Geometrical overlaps cause incorrect navigation of the geometry hierarchy in Geant4 and therefore incorrect tracking and use of physics processes due to the ambiguity of which volume a particle is in. Despite the unknown nature of the geometry at compilation, the program must avoid such scenarios. Geant4 geometry classes have no concept of extent once created, and even if the original parameters used to construct a shape are recovered through inspection, this does not indicate whether two shapes will fit beside each other or overlap. Geant4 provides overlap checking, but only by sampling particle trajectories through the geometry. To overcome this, in BDSIM, a series of extents that represent a simple cube or cylinder are recorded with each Geant4 solid in BDSIM to allow truly dynamically generated parameterised geometry from user input to be safely constructed into more complicated objects and beam lines whilst ensuring there are no geometrical overlaps.

The use of the factory pattern allows extra styles of geometry to be added as required with only minimal modification to the existing code, as well as the guarantee that this new geometry will work correctly with all other existing geometry. To provide this functionality, the previous geometry construction code was rewritten.

Additionally, the beam line construction code was rewritten to be more general and make better use of 3-dimensional transforms available through CLHEP. This allows beam lines to be built not just in a plane, but completely in 3 dimensions with tilts and rotations making BDSIM useful for transfer lines and gantry systems as well as large flat machines.

2.1 Aperture

MAD-X provides 8 different aperture models as well as a user defined boundary [1]. As these cover the most commonly used apertures in accelerators as well as MAD-X being a common source of input model, it was logical to add these to BDSIM. The 8 models (*circle, rectangle, ellipse, rectcircle, lhcscreen, rectellipse, racetrack, octagon*) were added to BDSIM as different factories. The aperture may be specified as a default for the whole model as well as on an element-by-element basis. A selection of apertures are shown in Fig. 1 as rendered by the Geant4 visualiser.

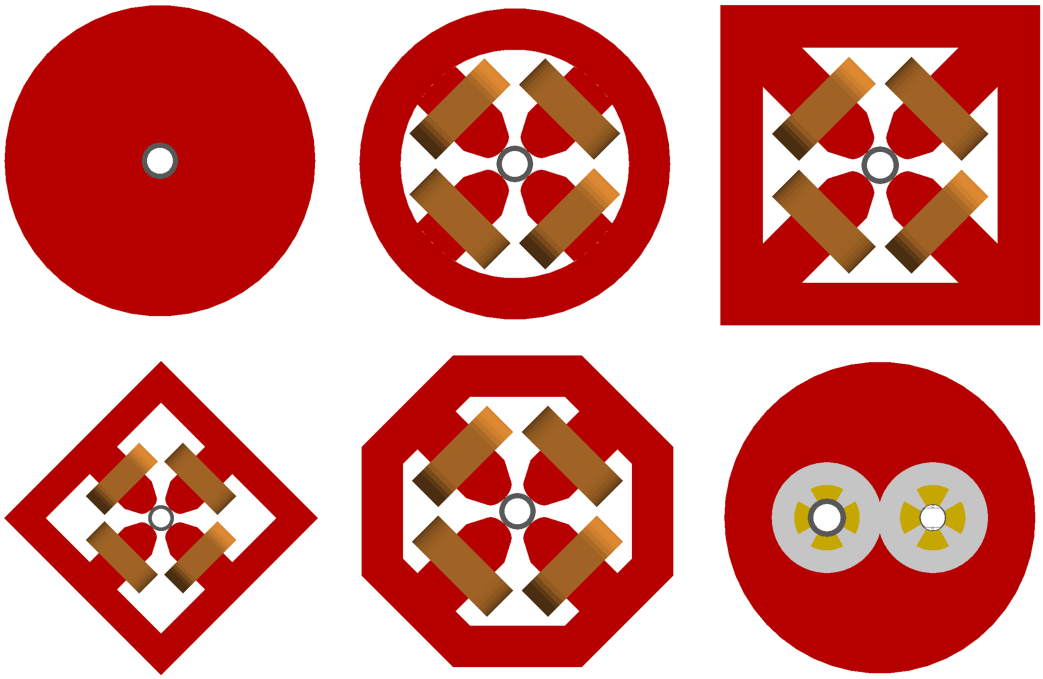

Fig. 2: Different magnet geometry styles for a quadrupole. Clockwise from the top left, these are *cylindrical, polescircular, polessquare, lhcright, polesfacetcrop* and *polesfacet*.

Aside from the aperture dimensions, the user may specify the thickness of the beam pipe that defines the aperture as well as the material (atomic composition, density and state) of the beam pipe and the vacuum volume inside the beam pipe. Additionally, a more complex model of the *lhcscreen* aperture was added including a copper screen, cooling pipes and outer layer [5, 8].

2.2 Magnet Geometry

In addition to the aperture models, more varied magnet geometry was required. The original style provided with BDSIM is an annular cylinder of solid user-assignable material surrounding the beam pipe, which is iron by default. Whilst this may be an acceptable simplification in some cases, the majority of accelerators use normal-conducting magnets with poles and coils, and so BDSIM was extended to provide different styles of magnet geometry.

To extend the magnet geometry capabilities required that the magnet construction code was rewritten in a factory pattern with one factory class providing one style of geometry for all types of magnets. Eight styles (*cylindrical, polescircular, polessquare, polesfacet, polesfacetcrop, lhcleft, lhcright and none*) are now provided with BDSIM. A selection of styles for a quadrupole magnet are shown in Fig. 2. The majority of the new styles provide magnet poles as well as coils that scale in proportion to the available space between the beam pipe and the user-defined parameter *horizontalWidth* for the full width of the magnet body. The poled geometry factories were written with common code to build the poles and a variety of yoke shapes connecting them. Both a dipole and quadrupole with the design of the LHC arc magnets are provided. Although the design of these is fixed, a small degree of tolerance in increased aperture is provided by removing parts of the coils if required.

Separate pieces of geometry for the end of the coils where they connect to the other side of the pole are provided by the poled geometry factories and these are placed only where the edge of a magnet is not adjacent to another magnet. Unused ones are cleared from memory before commencing the simulation to reduce memory usage.

A null option with no geometry was added for testing purposes, to allow a magnet where only the

beam pipe is constructed. A similar feature for the aperture definition is being added that will provide only a vacuum volume. Used together, these options will provide vacuum only tracking in a similar fashion to a more typical tracking code.

2.3 Tunnel Geometry

BDSIM originally provided no formal support for tunnel geometry. If the intention is to simulate highly penetrating radiation, it is important to include the tunnel as this will effectively filter and guide the radiation seen further along the beam line.

Similarly to the aperture and magnet geometry, a series of factories were written to provide five different styles (*square, rectangular, circular, elliptical and rectaboveground*) of tunnel geometry. *rectaboveground* represents a rectangular tunnel based on a thick flat slab of concrete, suitable for a ground level accelerator with surrounding shielding.

Aside from the factories to create segments of tunnel, a builder was written to provide a tunnel that will follow a given lattice. This surveys the already constructed beam line geometry and creates tunnel sections when the cumulative angle, offset or length reach a given tolerance. This allows an approximate tunnel to be simulated for a machine that is still under design with no definite tunnel schematic.

2.4 Curved Geometry Modelling

Nearly all accelerator components can be categorised as either straight or with a finite bending angle in one plane. The Geant4 CSG solids that can be used to provide curved geometry are a torus and a rotation solid. Using either of these for arbitrary aperture and magnet cross sections would lead to large placement offsets due to the typically large bending radii of high energy accelerator magnets. This may lead to lack of precision in the placement or length of the object as well as complicate the placement of each object in the beam line. To avoid this, the geometry in BDSIM is only made in straight sections. To achieve a bent section, the section is split into many small straight sections with angled faces. The number of sections a bend is split into is calculated from a maximum allowable tolerance between the arc and the chord of any segment. This is controllable by the user, but the default is 1 mm.

Constructing models in this way makes the relatively complex geometry construction code manageable. Straight sections with angled faces are constructed by creating a longer than desired straight section and then intersecting it with a cylinder with angled faces (the `G4CutTubs` class). A single section in a bend is constructed only once but placed many times, which significantly reduces memory usage whilst allowing the aperture to be more accurately represented.

2.5 Misalignments and Imperfections

For the most accurate simulation of an accelerator, measured magnet misalignments and field imperfections should be incorporated into a model. BDSIM is able to simulate each of these.

In the case of component misalignments, each component in BDSIM can be specified with an additional transverse offset or tilt (rotation about the curvilinear S axis). However, as a 3-dimensional model is created and surrounded by air, the components must meet exactly. In the case of dipoles (rectangular and sector bends), the transverse offset is ignored as this would change the length of the lattice. However, the tilt is taken into account and should the dipole have pole face rotations, the beam pipes before and afterwards are constructed with the correctly angled faces to allow seamless tracking. The fields are linked to the geometry so any misalignment is automatically handled in tracking.

Field imperfections are not treated directly as each beam line element has a pure field and the associated numerical integrator for tracking. However, BDSIM provides the ability to construct thin multipole elements. As each element is required to have 3 dimensions, these are constructed as a 1 μm long box. A symplectic thin-lens integrator is used for tracking. A common strategy is to place a thin

multipole at both ends of each magnet as well as in the centre of it (assuming it is split) with measured imperfections or higher-order multipole coefficients. BDSIM also provides the ability to import field maps and perform tracking using Geant4's 4^{th} order Runge-Kutta integrators, although whilst suitable on a small scale, these are non-symplectic and therefore not suitable for tracking in large circular machines.

2.6 Externally Provided Geometry

Whilst an extensive library of generic geometry is provided with BDSIM that allows quick progression from optical model to 3D geometry, more accurate geometry for a given existing accelerator may be desired. In this case, BDSIM provides the ability to load externally provided geometry in GDML format [13], and more formats may be added in future. Pieces of external geometry may be placed *as is* in the beam line, or in the case of magnets, wrapped around a beam pipe provided by BDSIM.

3 Physics Processes

Before Geant version 4.10, the developer of a program using Geant4 was required to instantiate classes for particles and physics processes as well as attach them to each other in the required way to provide a *physics list*. Geant4 has since encapsulated this construction in a set of physics lists that can be added together in a modular fashion. Custom physics processes or combinations thereof can be written to fit into this system. This provides a uniform and validated approach to a large library of physics processes as well as known outcomes. The implementation of the physics process construction in BDSIM was rewritten to use the new Geant4 modular physics lists classes. This now allows the physics lists to be used in a modular fashion that was not previously possible without modifying the source code and recompiling.

From Geant version 4.10.1 onwards, a generic interface to process cross-section biasing was introduced. This feature has now been exposed in BDSIM allowing the user to define a particle type, process and biasing factor as well as which volumes to attach it to. This allows efficient simulation of rare processes or situations such as interaction with residual gas in the accelerator vacuum.

4 Coordinate System & Data Handling

4.1 Curvilinear Coordinates

BDSIM, via Geant4, builds a 3-dimensional model in Cartesian coordinates. The model is built following a description of a lattice in curvilinear coordinates that follows the axis of the accelerator. Especially for a circular accelerator, it is more useful to describe beam losses in curvilinear coordinates than Cartesian ones.

Geant4 allows the developer to get the transform from any volume to the top of the geometry tree (the *World* volume) and vica-versa, giving *global* and *local* coordinates systems respectively. Geant4 also provides a mechanism to get the relative transform to a different level in the geometry hierarchy, however, given the potentially variable hierarchy depth of different components and styles available in BDSIM as well as the unknown depth of externally provided geometry, it is not possible to know beforehand the number of levels to navigate. Even then, curvilinear coordinates are desired and not global or local Cartesian coordinates.

To overcome this, BDSIM uses Geant4's capability to build a parallel world - a separate geometry model with a different representation of, in this case, the accelerator. A beam line of simple cylinders is built that mimics the reference trajectory of the beam line (without any orbit bumps) as shown in Fig. 3.

Any global coordinate in the original world (termed '*mass world*' by Geant4) may be read out in the parallel *curvilinear world*. The local to global transform from the simple cylinders may be used, providing the local Cartesian coordinates inside a given cylinder, which is degenerate with the transverse curvilinear coordinates of the accelerator. A registry of each cylinder is kept along with the curvilinear

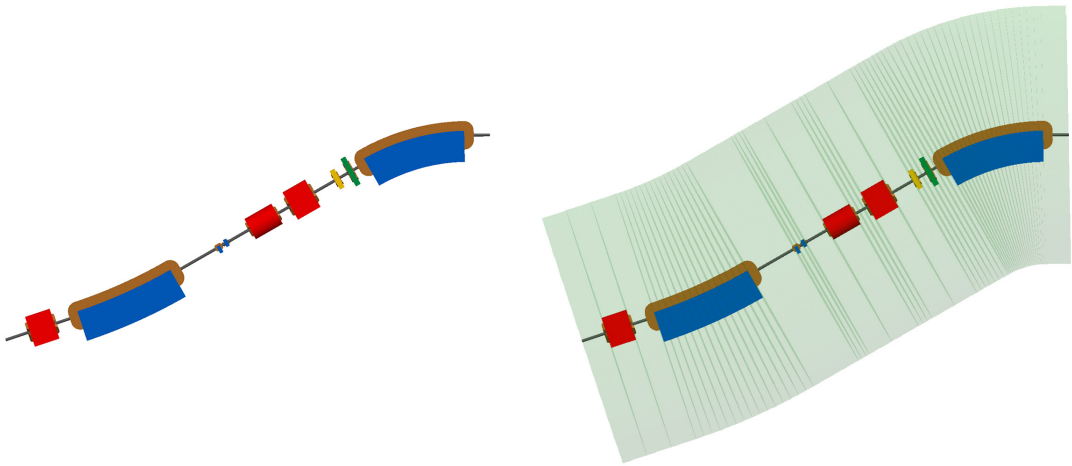

Fig. 3: Plan view of an example beam line with and without the parallel world curvilinear cylinder geometry overlaid in semi-transparent green.

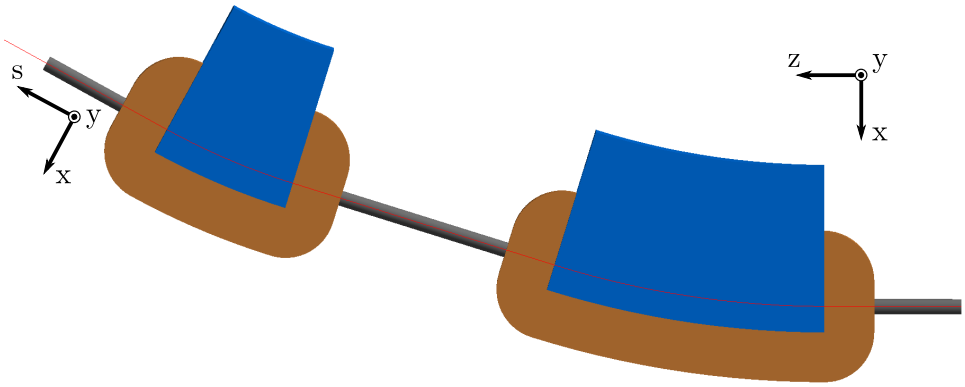

Fig. 4: Plan view of a simple beam line with two dipoles showing the difference between the curvilinear coordinate system x, y, s on the left and the global Cartesian system x, y, z on the right. The reference trajectory is shown as an overlaid red line.

s coordinate for the centre of that cylinder allowing s, x, y to be determined. The difference in these coordinate frames is shown in Fig. 4.

4.2 Output Format

Geant4 provides access to the full history of an event including all secondary particles and trajectories produced, but this is too much information to deal with in a manageable or meaningful way. Instead of this, a summary of each event as well as information requested by the user is recorded in a set of output C++ classes. At the end of each event these classes can be written to different output formats. The primary format for output is a ROOT [14] file. This allows direct serialisation of the output classes in a well-documented, freely-available compressed binary format. The ROOT framework provides a data visualiser and analysis framework that is highly suited to the analysis of this form of data and is widely used in the high energy physics community. The same classes used to write the data are used by an analysis tool, REBDSIM, provided with BDSIM to simplify loading the data. The analysis library may be used in compiled code, the ROOT interpreter and the Python interpreter with equal functionality. The BDSIM output files contain four main structures stored in ROOT *Trees*:

1. The complete set of options used for the simulation.

2. A record of the coordinates and transforms for the model.
3. A summary of all events (*Run* information)
4. Event by event summaries.

The event tree stores information on a per event level, where an event typically starts with tracking an individual particle from the primary beam distribution. For each event, the initial coordinates, energy deposition histograms, duration and primary trajectory are recorded. The user may define elements in the beam line to attach *samplers* to, which are 1 pm thin planes placed at the end of each element that record the coordinates and species of all particles passing through them. The samplers are also recorded per event in the event tree.

5 Circular Machines

BDSIM was originally developed for linear, single-pass machines. It has recently been extended to facilitate tracking in circular machines. As BDSIM builds a 3-dimensional model, no modification was required to track particles in a circular machine. However, further modification was required for control over the simulation in a multi-turn setting.

In a Geant4 application, each event finishes when all particles are tracked down to zero energy or all leave the World volume. In the case of a storage ring and in the absence of synchrotron radiation, a particle inside the dynamic aperture of a circular machine may be stored indefinitely. As each beam particle is simulated individually there are no collective or stochastic effects that would naturally cause the particle to be lost over time. Therefore, the simulation would continue indefinitely. To allow the simulation to terminate and control the maximum number of turns taken in any one event, a new element called the *terminator* was added. This counts the passage of the primary particle on each revolution and dynamically changes to an infinite absorber after a certain number of turns have passed. In the case of a conventional tracking code, the beam line sequence is a series of maps that is explicitly applied to a set of particle coordinates so the user must specify a maximum number of turns.

Due to the 3-dimensional nature of a BDSIM model, a further difference between BDSIM and conventional tracking codes was discovered. Not all optical models of circular machines close perfectly in Cartesian coordinates. In the case of the LHC, the common dipoles used to bring the beams in and out of collission are treated as sector bends in the optical model leading to a very small inaccuracy. In a Cartesian model, the $\sim$70μm transverse offset causes the beam to be lost within a few turns. This error in the optical model is extremely small compared to the alignment tolerances and scale of the machine and so is not considered problematic normally. To compensate for this effect, a new beam line element called the *teleporter* was introduced that transports the particle from the exit of the machine to the entrance such that a reference particle with no transverse displacement would return to the same position.

6 Hypothetical Example

To demonstrate the capabilities of BDSIM a hypothetical example beam line was written. The lattice consists of two dipoles, 3 quadrupoles, a sextupole and octupole as well as a pair of horizontal and vertical steering dipoles. A visualisation of the machine in BDSIM is shown in Fig. 5. This provides a selection of magnets and geometry styles, but is not designed for a specific optical purpose. This in turn leads to a desirable beam loss to demonstrate the features and analysis.

1000 particles were randomly sampled from a Gaussian beam as described in Table 1. The default options for BDSIM and the list of physics processes used was *em* and *synchrad* which map to the Geant4 physics process constructors `G4EMStandardPhysics` and the registration of the process `G4SynchrotronRadiation` to all charged particles. The resulting output file from BDSIM ($\sim$28 MB in size) was analysed using REBDSIM that provides event-by-event analysis of BDSIM output file. It produces a series of histograms as specified in a simple ASCII text file (*analysisConfig.txt*) written by

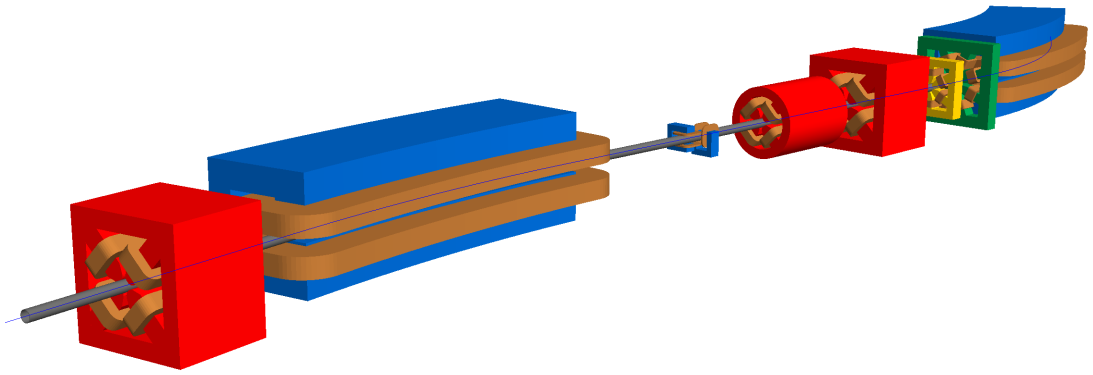

Fig. 5: Hypothetical beam line using a variety of magnets and geometry styles with the reference trajectory shown by an overlaid blue line.

Table 1: Beam parameters used to simulate hypothetical example beam line energy deposition

Parameter	Value	Unit
Particle	e^-	
E	3	GeV
σ_x	1	mm
$\sigma_{x'}$	1	μ rad
σ_y	1	mm
$\sigma_{y'}$	1	μ rad
σ_E/E	0.01	

the user. The analysis configuration file used for this analysis is shown in Fig. 6. REBDSIM produces an output file also in ROOT format with the histograms categorised in directories by whether they were derived from a run or event level analysis.

Figure 7 shows the primary *hits*, *losses* and energy deposition in the accelerator material as a function of curvilinear s position. The primary hit is the first point on the primary trajectory where a physics process determined the step length taken and was therefore not transportation or boundary crossing step. The loss point is the end of the primary trajectory and this represents the last point at which the particle is classified as a primary particle in the simulation. All geometry is sensitive to energy deposition and this is recorded in a histogram per event as a function of s.

Figure 7 has several notable features. For both vertical axes the data is the mean across all 1000 events with the vertical error bars representing the r.m.s. The vertical scales are therefore the expected rate as opposed to the total number observed in this simulation.

Also of note is that the primary hits are all within the first metre of the beam line. Here, these 'hits' are due to synchrotron radiation in the first dipole, as the synchroton radiation is the first physics process invoked and the energy of the primary particle is affected. If a different condition is desired, the user may easily write their own analysis of the primary trajectory. The near constant low-level energy deposition from $s = 1 - 3$m is predominantly due to the fan of synchrotron radiation being absorbed by the beam pipe inside the first dipole.

7 Summary & Outlook

The latest developments to BDSIM have been described that significantly revise many features throughout the code. The new geometry factories for aperture, magnets and tunnel sections described provide a much richer set of simulation abilities than previously available. The revised output format and REBD-

analysisConfig.txt

```
Debug                                   0
InputFilePath                           ./output_event.root
OutputFileName                          ./ana_1.root
CalculateOpticalFunctions               1
CalculateOpticalFunctionsFileName       ./ana_1.dat
# Object       treeName       Histogram Name              # Bins    Binning              Variable                      Selection
Histogram      Event.         ElossAgain                  {50}      {0:10.2}             Eloss.S                       Eloss.energy
Histogram      Event.         Primaryx                    {100}     {-0.1:0.1}           Primary.x                     1
Histogram      Event.         Primaryy                    {100}     {-0.1:0.1}           Primary.y                     1
Histogram      Options.       seedState                   {200}     {0:200}              Options.GMAD::OptionsBase.seed 1
Histogram      Model.         componentLength             {100}     {0.0:100}            Model.length                  1
Histogram      Run.           runDuration                 {1000}    {0,1000}             Info.duration                 1
Histogram      Event.         8XY_Distribution            {50,50}   {-2:2,-2:2}          Sampler_d2_8.x:Sampler_d2_8.y 1
Histogram      Event.         8XY_Distribution_positron   {50,50}   {-2:2,-2:2}          Sampler_d2_8.x:Sampler_d2_8.y Sampler_d2_8.partID==11
Histogram      Event.         8XY_Distribution_electron   {50,50}   {-2:2,-2:2}          Sampler_d2_8.x:Sampler_d2_8.y Sampler_d2_8.partID==-11
Histogram      Event.         8XY_Distribution_photon     {50,50}   {-2:2,-2:2}          Sampler_d2_8.x:Sampler_d2_8.y Sampler_d2_8.partID==22
Histogram      Event.         8Energy                     {100}     {0:3.01}             Sampler_d2_8.energy           1
Histogram      Event.         4XY_Distribution            {50,50}   {-0.1:0.1,-0.1:0.1}   Sampler_d2_4.y:Sampler_d2_4.x 1
Histogram      Event.         4XY_Distribution_positron   {50,50}   {-0.1:0.1,-0.1:0.1}   Sampler_d2_4.y:Sampler_d2_4.x Sampler_d2_4.partID==11
Histogram      Event.         4XY_Distribution_electron   {50,50}   {-0.1:0.1,-0.1:0.1}   Sampler_d2_4.y:Sampler_d2_4.x Sampler_d2_4.partID==-11
Histogram      Event.         4XY_Distribution_photon     {50,50}   {-0.1:0.1,-0.1:0.1}   Sampler_d2_4.y:Sampler_d2_4.x Sampler_d2_4.partID==22
Histogram      Event.         4Energy                     {100}     {0:3.01}             Sampler_d2_2.energy           1
```

Fig. 6: Example analysis configuration file for REBDSIM used to perform analysis. A series of both 1D and 2D histograms are defined along with their binning and which variable in the data to use. The selection column is used for weighting where a Boolean expression can be used for weight 1 or 0.

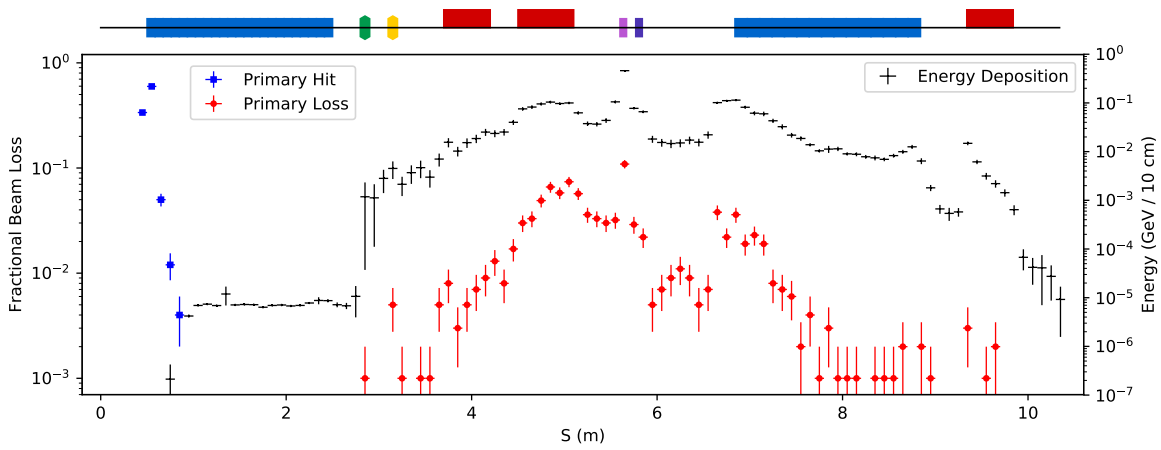

Fig. 7: Beam interaction ('*hit*') points, loss points and energy deposition throughout the lattice as a function of curvilinear s position. A schematic of the lattice is shown above the graph with the dipoles in blue, quadrupoles in red, sextupoles in yellow, octupole in green and corrector dipoles in light and dark purple.

SIM analysis tool provide the new ability to do an event-by-event analysis with the correct statistical analysis.

These developments have been implemented in a careful and generalised manner so that they are not only applicable to the developers' studies but so that BDSIM is capable of simulating beam losses and backgrounds for nearly any accelerator in the high energy physics community. Whilst it is not possible to predict every desired feature, a strongly object-orientated approach with well defined and documented patterns allows others to extend BDSIM with their own geometry and features suitable for other experiments.

Acknowledgements

The development of BDSIM has received funding from the John Adams Institute at Royal Holloway. This work was supported in part by EU FP7 EuCARD-2, Grant Agreement 312453 (WP13, ANAC2) and STFC, United Kingdom.

References

[1] The MAD-X Program, CERN, http://madx.web.cern.ch/madx

[2] F. Schmidt, E. Forest and E. McIntosh, Introduction to the Polymorphic Tracking Code: Fibre Bundles, Polymorphic Taylor Types and Exact Tracking, CERN-SL-2002-044-AP (2002).

[3] C. Adolphsen *et al.* The International Linear Collider Technical Design Report, Vol 3.II: Accelerator Baseline Design, CERN-ATS-2013-037 (2013).

[4] O. Markin, *arXiv* 1402.2515 (2014).

[5] O. Brüning *et al.*, LHC Design Report, CERN 2004-003-V-1 (2004), http://dx.doi.org/10.5170/CERN-2004-003-V-1

[6] http://www.pp.rhul.ac.uk/bdsim/

[7] I. Agapov *et al.* Nucl. Instrum. Methods A **606** (2009) 708.

[8] J. Snuverink *et al.* Beam Delivery Simulation - Recent Developments and Optimization, Proc. International Particle Accelerator Conf., Richmond, Virginia, USA, 2015.

[9] L. Nevay *et al.* Beam Delivery Simulation: BDSIM Automatic Geant4 Models of Accelerators, Proc. International Particle Accelerator Conf., Busan, Korea, 2016.

[10] S. Agnostinelli *et al.* (The Geant4 Collaboration), Nucl. Instrum. & Methods A **506** 3 (2003) 250-303.

[11] J. Allison *et al.* (The Geant4 Collaboration), Nucl. Instrum. & Methods A **835** (2016) 186-225.

[12] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns, (Addison-Wesley Professional, 1994), Chap. 3.

[13] R. Chytracek, J. McCormick and W. Pokorski, IEEE Trans. Nucl. Sci. **53** (2006) 2892.

[14] R. Brun and F. Rademakers, Nucl. Inst. & Meth. in Phys. Res. A **389** (1997) 81-86