BEAM DELIVERY SIMULATION (BDSIM): A GEANT4 BASED TOOLKIT FOR DIAGNOSTIC AND LOSS SIMULATION

S. T. Boogert, S. M. Gibson, R. Kwee-Hinzmann, L. J. Nevay, J. Snuverink,
Royal Holloway, University of London, Egham, UK
L. C. Deacon, CERN, Geneva, Switzerland

Abstract

BDSIM is a Geant4 and C++ based particle tracking code which seamlessly tracks particles in accelerators and particle detectors, including the full range of particle interaction physics processes in Geant4. The code has been used to model the backgrounds in the International Linear Collider (ILC), Compact Linear Collider (CLIC), Accelerator Test Facility 2 (ATF2) and more recently the Large Hadron Collider (LHC). This paper outlines the current code and possible example applications and presents a roadmap for future developments.

INTRODUCTION

For computations requiring both tracking particles through long beam lines and simulating particle interactions with accelerator components, one often uses several codes. For example, a fast tracking code is used to record positions where particles hit a collimator or the beam-pipe, and these positions are then input into a detailed Monte-Carlo radiation transport code. However, this approach has several drawbacks. Firstly, the geometry description capabilities of such codes are rarely equivalent as they commonly have a very limited description of the geometry and secondly, the data must be transferred between the two codes. BDSIM [1, 2] was developed to overcome these drawbacks. It is an extension of Geant4 [3], a Monte-Carlo framework, giving access to many electromagnetic and hadronic interaction models as well as a powerful geometry description framework and visualisation tools. On top of this, fast particle tracking routines and additional physics processes are introduced, and a high level geometry description language GMAD [4] is added. Since GMAD is an extension of MAD [5], a standard for beam optics description, this allows complex accelerator descriptions to be loaded from existing repositories with just a few modifications. The architecture of BDSIM is sketched in Fig. 1. Each element has a magnetic field and a "stepper" associated with it that implements its particle transportation. The output of the simulated results can be conveniently written in either ASCII or in the ROOT analysis framework format [6], widely used in particle physics. Due to the integration with ROOT, the data analysis can be done internally 'on the fly'. This paper gives an overview of the latest code developments and some recent usage examples. More information and installation details can be found at `www.pp.rhul.ac.uk/twiki/bin/view/JAI/BdSim`.

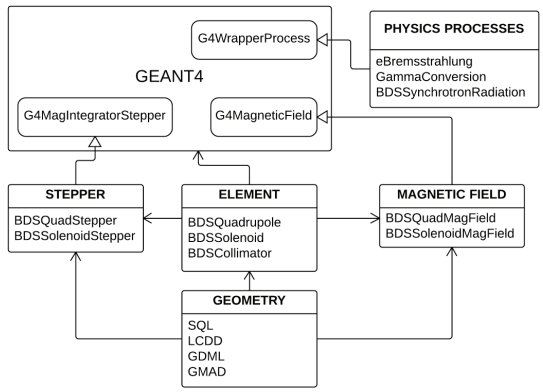

Figure 1: Schematic diagram of the BDSIM architecture.

NEW FUNCTIONALITY

Build System

BDSIM builds on top of Geant4, which has the option to include many visualisation drivers and geometry description languages, and optionally the analysis framework ROOT. The large number of package dependencies require great care in the building stage, especially cross-platform (currently Linux and Mac are supported). BDSIM previously used GNU Autoconf for building, but has switched to CMake [7] for the aforementioned reasons. CMake is a free software program for managing the build process of software using a compiler-independent method. Now that the latest versions of Geant4 have also switched to CMake, the building scripts have significantly simplified and are more robust. In addition, CMake provides a testing framework CDash, which is easily integrated in the existing CMake scripts. A server has been setup for testing the builds on various platforms automatically on a daily basis.

Test System

The major components of an accelerator system are the magnetic elements. A test suite of gmad files describing the basic components has been developed to allow the efficient unit testing of the tracking in the different classes of magnets.

PYTHON UTILITIES

Python has been selected to provide utilities to augment BDSIM, mainly because of the relative ease of developing scripts, it is open source and there are a large number

ISBN 978-3-95450-127-4

of high quality scientific calculation and plotting libraries. The utilities developed and their relationship with the main BDSIM project is shown in Fig. 2.

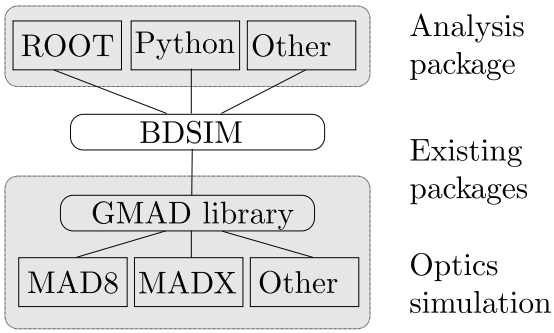

Figure 2: Python auxiliary utilities in relationship to BDSIM and the GMAD parser, the grey boxes indicate where Python wrappers have been created.

Conversion

To aid the efficient conversion of optics simulation programs to BDSIM, conversion utilities have been written from MAD8 and MADX to GMAD. The MAD8 conversion operates on the output of a MAD8 `saveline` command,

```
USE, LATTICE
SAVELINE, NAME="LAT", FILENAME="LAT.saveline"
```

whilst the MADX conversion uses the output of the `twiss` command, so

```
select,flag=twiss, clear;
twiss, file=LAT.tfs,save;
```

As GMAD is close to the MAD8 syntax, no specific conversion tools were developed, and this led to cumbersome manual checking of the conversion and could introduce errors. The new conversion requires no user input and has been tested on medium sized lattices. This conversion was tested on the ATF2 [8] MAD8 lattice and an example of the visualisation is shown in Fig. 3.

GMAD Parser

Previously the only way to debug and diagnose geometry problems with the lattice was to visualise the BDSIM model of the lattice using one of the Geant4 visualisation systems described previously. Although this is valuable, a python wrapper to the GMAD parser library has been written. This encapsulates the `C` based GMAD parser so that the lattice can be interrogated without executing the entire BDSIM executable.

Geometry

A significant drawback of using Geant4 for the description of beam lines is that the other codes typically used for

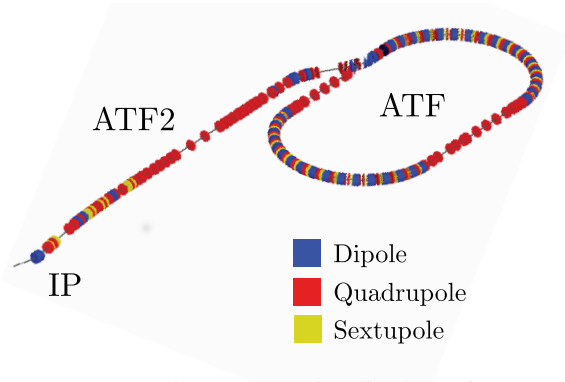

Figure 3: Example BDSIM visualisation of a MAD8-saveline conversion to GMAD of the ATF and ATF2 test accelerator.

loss simulations use constructive solid geometry (CGS) as opposed to the shape primitives used in Geant4. For example a large effort has gone into developing the FLUKA [9] representation of the accelerator around the LHC interaction regions, but conversion from this format is particularly difficult as there is no easy method to transfer between the two geometry descriptions. We have developed a simple conversion technique which generates a Standard Tessellation Language (STL), which is essentially a raw unstructured triangulated surface description generated from the constructive solid geometry representation using CUBIT [10]. There are multiple methods based of converting CGS formats to STL formats but these only convert the geometrical aspects of the information required for the simulation. Simulations such as Geant4 and FLUKA also require a material description, which must also be converted. The STL representation can be converted to a Geometry Description Markup Language (GDML) format that can be directly imported to BDSIM. It is not clear if such a conversion process will require too much memory for running a large geometry like the LHC, but this memory requirement is only transitory before the geometry is converted into voxels in Geant4.

Analysis

A Python package has been written to load and collate the BDSIM output. This collation allows addressing of hits at a single sample plane by the element name even though the hits may have occurred in different passes in a ring geometry or when the particles shower. A suite of reference plotting tools have also been implemented using the Matplotlib Python package for quick viewing of basic parameters throughout the lattice such as the number of particles, the particle trajectory in average position $\overline{x}$ and $\overline{y}$ and average angle $\overline{x'}$ and beam sigma matrix σ_{ij}.

RECENT EXAMPLES

In this section some recent representative examples of BDSIM usage are briefly summarised.

ISBN 978-3-95450-127-4

Muon Background Studies in CLIC

A study of muon backgrounds in the Compact Linear Collider (CLIC) [11], a possible future e^+/e^- linear collider, has been performed using BDSIM [12]. Halo particles in linear colliders can result in significant losses and serious background in the detectors that may reduce the overall performance. Even if most of the halo is stopped by collimators, the secondary muon background may still be significant. It is therefore important to include halo generation and tracking in collimation studies. Halo and tail particles were generated by the halo generation code HTGEN [13] based on beam gas scattering and inelastic scattering. The particles were tracked through the CLIC lattice using its interface to the tracking code PLACET [14], which was used for its collimator wakefield implementation. For particles that hit aperture limits, BDSIM was used for the detailed interaction with matter and the tracking of the secondaries towards the detector, where they could be used as input to the CLIC detector simulations. An interface of PLACET and BDSIM has been created for these kind of simulations. Particles are tracked alternately in PLACET and BDSIM [15].

CLIC Post Collision Line

The 1.5 TeV Compact Linear Collider (CLIC) beams, with a total power of 14 MW per beam, are disrupted at the interaction point due to the very strong beam-beam effects. The disrupted beam has a power of 10 MW. Some 3.5 MW reaches the main dump in the form of beamstrahlung photons, and about 0.5 MW of e^+ and e^- coherent pair particles with a very broad energy spectrum as well as the lower energy disrupted beam particles need to be disposed of along the post collision line. Background and energy deposition studies were developed using BDSIM [16].

Several shielding configurations for the special C-shaped magnets, see Fig. 4 have been tested to calculate their magnet lifetime. The results indicated that further improvements may be required if the magnets are to survive in the CLIC post-collision line radiation environment for a sufficient length of time.

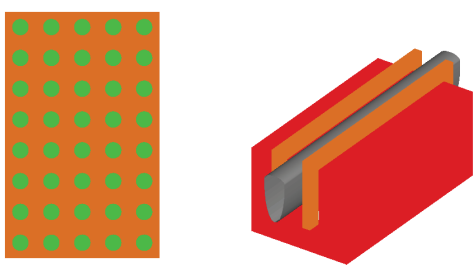

Figure 4: C-shaped CLIC post-collision line magnet simulation with detailed coil geometry. Left: a section of the coil where the copper cables are shown in green, and the insulation is in orange. Right: The magnet yoke (red), coils (orange) and beam pipe (grey).

In addition, luminosity monitoring using the post-collision line was investigated. For this study the Geant4 physics process named 'QGSP-BERT-HP' was modified to enhance the cross-sections for the resulting muons in order to be able to study muon signal in particular.

LHC

In the future the Large Hadron Collider (LHC) will be upgraded for high luminosity (HL-LHC) [17] with an order of magnitude increase in luminosity. Such an increase will require precise knowledge of the energy deposition in the accelerator itself for protection of the cryogenic systems. To this end, BDSIM has been recently modified to simulate proton transport and energy losses. The magnetic description of the lattice has been used from MADX to construct the ring and the generic component library used while conversion of the existing machine geometry in the FLUKA geometry description to that of Geant4 is undertaken. A segment of the beamline from the current simulations using generic magnetic components is shown in Fig. 5.

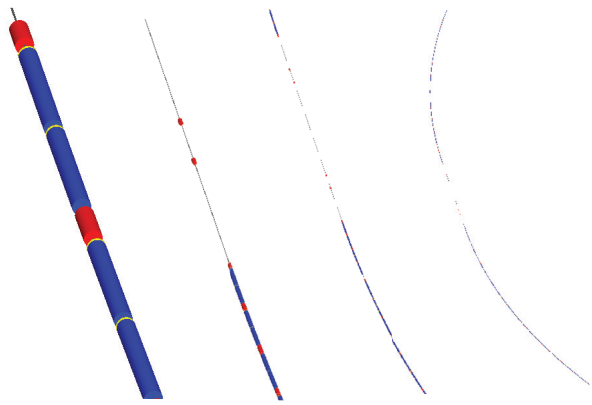

Figure 5: Progressive zoom of part of the LHC beam line showing the geometry built from the generic component library.

With 20 protons simulated for 5 turns the tracking gives the plots shown in Figs. 6 and 7.

The LHC is a particularly complex example for BDSIM, both in terms of the multi-turn tracking, but also as little of the beam geometry description is available in a Geant4 compatible format.

FUTURE PLANS

BDSIM is an accepted tool for single pass machine simulations, such as the ATF/ATF2, ILC and CLIC. Particle backgrounds at these machines mainly impact the diagnostics systems, collimation system and the low-β interaction region. The beam instrumentation that will measure Compton scattered photons, like laserwires, Shintake monitors and polarimeters will all require an accurate simulation of the background environment for successful operation.

For use at the LHC various enhancements and upgrades are required. The first is in the accelerator particle tracking,

ISBN 978-3-95450-127-4

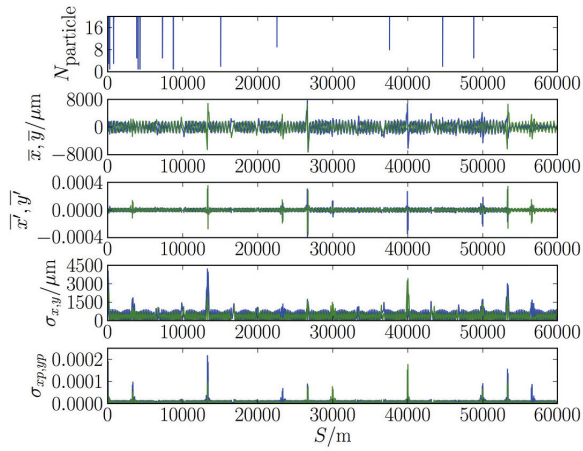

Figure 6: Proton transport parameters in the LHC lattice.

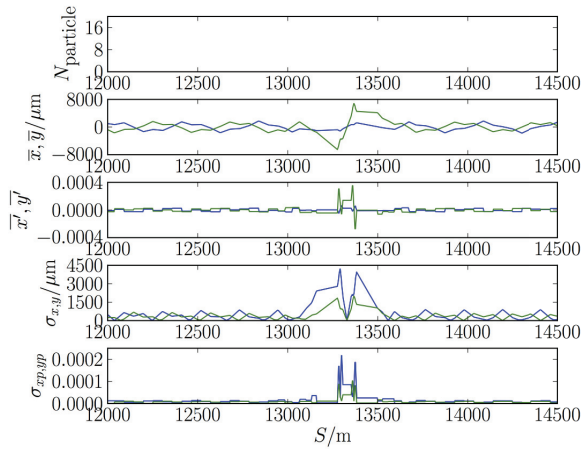

Figure 7: Proton transport parameters in the LHC lattice, zoomed around one of the interaction regions.

which should include symplectic integration schemes. For the HL-LHC the beam losses are dependent on the collimator and apertures seen by the particle beam and an accurate and flexible description of these needs to be implemented. To improve the geometry of the machine beyond the beam pipe, the conversion explained previously will be further developed. Furthermore, an interface to SixTrack [18] will be created to allow compatibility with existing LHC beam loss simulations.

CONCLUSION AND OUTLOOK

We report the development of BDSIM, there has been significant progress in refactoring the code base and build system. This is essential for adoption by users who wish to simulate beam loss and instrumentation sensitive to beam losses. There has been significant progress in automagic conversion from existing beam line optics simulation tools, such as MAD8 and MADX, to the BDSIM description using GMAD. This conversion is significantly more efficient and less prone to errors than previous methods. Finally we tested the conversion on the entire LHC 3.5 TeV ring and tracked particles around the ring for a few turns.

For ring systems, there are significant upgrades required that include symplectic tracking, recording the number of turns primary beam particles have taken, and termination conditions.

REFERENCES

[1] I. Agapov *et al.*, "The BDSIM toolkit", EUROTEV-REPORT-2006-014

[2] I. Agapov *et al.*, "BDSIM: A particle tracking code for accelerator beam-line simulations including particle- matter interactions", NIMA 606, (2009), 708

[3] S. Agostinelli *et al.*, "Geant4 - a simulation toolkit", NIMA 506 (2003) 250-303

[4] I. Agapov, "GMAD accelerator description language", EUROTeV-Memo-2006-002-1

[5] H. Grote and F. Schmidt, "MAD-X: an upgrade from MAD8", PAC 2003, `http://mad.web.cern.ch/mad/`

[6] R. Brun and F. Rademakers, "ROOT - An Object Oriented Data Analysis Framework", NIMA 389 (1997) 81-86, `http://root.cern.ch/`

[7] `http://www.cmake.org`

[8] B. I. Grishanov *et al.* (ATF2 Collaboration), "ATF2 Proposal", (2005)

[9] G. Battistoni *et al.*, "The FLUKA code: Description and benchmarking", Proc. Hadronic Shower Simulation Workshop, (2006)

[10] `https://cubit.sandia.gov`

[11] CLIC collaboration, "Compact Linear Collider Conceptual Design Report", `http://clic-study.org/accelerator/CLIC-ConceptDesignRep.php`

[12] H. Burkhardt *et al.*, "Muon backgrounds in CLIC", IPAC 2010, THPD014

[13] I. Ahmed, H. Burkhardt and M. Fitterer, "User Manual for the Halo and Tail generator HTGEN", EUROTeV-Memo-2008-003

[14] A. Latina *et al.* "Evolution of the tracking code PLACET", IPAC 2013, MOPWO053

[15] I. Agapov *et al.*, "Tracking studies of the Compact Linear Collider collimation system", PRSTAB 12:081001, (2009)

[16] L. C. Deacon, "Updates to the CLIC post-collision line", IPAC 2012, TUPPR020

[17] J. L. Rossi, "LHC Upgrade Plans: Options and Strategy", IPAC 2011, TUYA02, `http://hilumilhc.web.cern.ch/`

[18] R. de Maria *et al.*, Recent developments and future plans for SixTrack", IPAC 2013, MOPWO028

ISBN 978-3-95450-127-4