BDSIM Development
2014 - 2015

L. Nevay, J. Snuverink, S. Boogert,
H. Garcia-Morales, S. Gibson, L. Deacon
R. Kwee-Hinzmann, S. Walker, A. Abramov

(LHC – not coaxial cable)
Beam Delivery Simulation - BDSIM

- Tracking code that uses Geant4
- Open source C++
- Automatically builds Geant4 model
- Uses MadX-like syntax for test input
- Mixes normal accelerator tracking & Monte Carlo particle physics
- Full showers of secondaries created by Geant4 processes
- Ability to simulate synchrotron radiation
- Simulate energy deposition and detector backgrounds
- Ability to import external geometry and field maps
Previously…

- Initial conversion and loss maps using BDISM
- First Geant4 based loss map of LHC
- Simplistic geometry
  - symmetric geometry
  - only circular & elliptical aperture
BDSIM Development

- BDSIM heavily developed since 2012 / 2013
- Recent development followed 3 main themes:

  - **Geometry**
  - **Tracking**
  - **Physics processes**

- Previous questions & issues fell into these three categories

- LHC Specific developments
- Documentation & general development
- Analysis tools & workflow
Geometry

• Previous geometry relatively simple
  — Adequate for conceptual studies
  — Great detail required for real machines
• Main geometry library rewritten
• Extensive use of factory pattern
  — Each factory represents a style and can make every type of say magnet
• 6 different **aperture** types (including detailed LHC)
• 6 different **magnet styles** (again with LHC style)
• 4 different **tunnel styles**
  — can follow the beam line
  — will be able to have external geometry and customise for certain ranges
• Most importantly all geometry works together
• Any beam pipe will work with any magnet!
• Very simple to extend with new geometry
  — guaranteed to work with all magnets
• New collimators by **H. Garcia-Morales**
Geometry

• 6 aperture models
  — circular, rectangular, elliptical, lhc, lhcdetailed, rectellipse
• Modelled on MadX aperture parameterisation
• Works with any other geometry
• 6 different magnet styles

Circular

RectEllipse

LHC screen

SRF Cavities

(L. Walker)

LHC Style

Poles circular yoke

Rectangular / square

Elliptical

LHC detailed

Poles square yoke
Tunnel Geometry

- Was only partially implemented previously
- Rewritten using factories
- Currently 4 different styles
- Can automatically follow beam line
- Can describe different styles for different sections*
- Can use external geometry for sections*

* nearly complete
External Geometry?

- For when the generic components just won’t suffice
- Can import external geometry
  - SQL, Mokka, GDML, STL
- Can also overlay field maps and interpolate
  - 2D, 3D, etc.
- You can also **export to GDML** from BDSIM!
Tracking

• Quantitative comparison with PTC & SixTrack underway

• Very good agreement with PTC
  — Tracking & optical function calculation

• Factorising tracking into library
  — will reduce tracking time by order of magnitude
  — Will allow choice of integrators
  — Will be able to use other tracking libraries shortly
  — Expected complete early 2016

A. Abramov

Double Bend Achromat agreement with PTC
Symplecticity

- Clearly not symplectic
- Factorising into a tracking library opens door to this
- Track first and test aperture continuously
- When approaches aperture -> shift to Geant4 model
- Non-symplectic tracking sufficient from then on
  - impact inherently non-symplectic!
- Faster
  - get the particle to the impact point
- Ideally use an external library!

Current LHC model very stable to 1000s of turns

“I thought I was on to something but I can’t figure out how to move it.”
Physics & Processes

• Benefit from regular Geant4 updates to many models
• Moved entirely to range cuts
  — several bugs fixed in this
• Cut particles not on energy but on range to produce a secondary particle
• Much more accurate stopping location
  — and therefore energy deposition
• Improved physics accuracy for lower CPU usage
• Modular physics list implemented in Geant4
  — can mix and add to physics processes very easily
• Remember, if it can be wrapped in C++, you can add the physical processes
Processes

- Quantitative comparison of scattering cross-sections on-going
- Benefiting from updated Geant4 all the time
- Example spectra from scattering at end of TCP
  — $1 \times 10^8$ primaries simulated
Process Biasing

- Introduced interface to Geant4 process biasing
- **Any** process for **any** particle can be biased for **any** volume or set of volumes

Define bias ‘object’

- which particles
- which processes
- cross-section scaling

Attach sets of biases to objects

- Extremely flexible interface
- Attach to vacuum or general accelerator material
- Previously required specially written wrapper class for each

```cpp
biasDef0: xsecBias, particle="e-", proc="msc eIoni eBrem CoulombScat", xsecfact={10,10,10,10}, flag={1,1,1,1};
biasDef1: xsecBias, particle="e-", proc="msc eIoni eBrem CoulombScat", xsecfact={10,10,10,10}, flag={1,1,1,1};
biasDef2: xsecBias, particle="e+", proc="msc eIoni eBrem annihil CoulombScat", xsecfact={10,10,10,10,10}, flag={1,1,1,1,1};
biasDef3: xsecBias, particle="e-", proc="msc eIoni eBrem CoulombScat", xsecfact={10,10,10,10}, flag={1,1,1,1};

d1 : drift, l=1*m, bias="biasDef0 biasDef1"; ! uses the process biasDef0 and biasDef1
q1 : quadrupole, l=1*m, material="Iron", biasMaterial="biasDef2", biasVacuum="biasDef3";
d2 : drift, l=1*m, bias="biasDef3";

beamLine: line = (d1,q1,d2); !defines the beam line
use, period=beamLine; !chooses the beam line to use
```
LHC Specific Developments

- Beam pipe geometry & magnet style
- Most development *can* and *has been* very general
- Rewriting of geometry was necessary
  - gives the required aperture flexibility
  - key to correct results!
- Python utilities extend for large automatic conversion
  - definitely required for the LHC
  - now integral part of work flow for all studies
- Primary trouble in piecing together information
- Aperture information significantly different from magnetic
- Tools nearly finished to split and match elements
Loss Map & 6T Hits -> Loss Map

- Unfortunately, this will have to follow
- Significant time to get aperture correct
- Large combination of files in many formats to be converted
- Model must be carefully checked
- Will present at CoLUSM in November
Direct Injection

- Had the ability to read out in curvilinear coordinates – now in too
- Introduced ability to inject particles anywhere in lattice
- Any beam distribution as function of S
  - Interpolation of trajectory within arcs
  - Efficient look up of transforms
- Sixtrack loader written by R. Kwee
- Can therefore convert SixTrack hits to energy deposition
Code & Software Development

- 60,000 lines of C++
- Revised *class hierarchy* & *factory patterns* – improved geometry
- Increased factorisation – much easier to extend
- Consolidation of development branches
- C++ 11 adoption & latest versions of Geant4, ROOT, CLHEP,
- Parser significantly revised by J. Snuverink
  - memory leaks, and problems fixed
  - written in object-orientated C++
- CTest test suite, CMake build system
  - much easier to use as compared to old configuration scripts
- CDash **nightly** and **on demand** automated building & testing
- Issue tracking & reporting
- Built in configuration for AFS
- Automated manual updates
- Regular release cycle
Output Analysis

- ASCII & ROOT output formats
  - ASCII useful for initial and single particle trials
    - ASCII isn’t suited to large data
    - ASCII inefficient storage and not as strongly structured
  - ROOT output used for studies
    - ability to introduce other formats if / when required (HDF5)
- RoBDSIM analysis tool written in C++
  - compiled C++
- Can use in 3 ways with exactly the same functionality
  - C++ linked to, use interactively in ROOT, use in python
- RoBDSIM typically used in Python, ROOT and on farm
- Can add and produce histograms
- Suitable for farm analysis of job files
- [https://bitbucket.org/jairhul/robdsim](https://bitbucket.org/jairhul/robdsim)
Statistical Comparison of Tracking

- Can calculate optical functions from particle distribution
- Another measure as compared to single particle tests
- Used to compare BDSIM with design of lattice
  — useful for spotting conversion discrepancies
- Can compare with PTC
- Significant work on uncertainty calculation of optical functions from particle distribution

\[
\epsilon_x = \sqrt{\langle x^2 \rangle \langle x'^2 \rangle + \langle xx' \rangle^2},
\]

\[
\beta_x = \frac{\langle x^2 \rangle}{\epsilon_x},
\]

\[
\Sigma = \begin{pmatrix}
\langle x^2 \rangle & \langle xx' \rangle \\
\langle x' x \rangle & \langle x'^2 \rangle
\end{pmatrix}
\]

\[
\sigma_f^2 = \left( \frac{\partial f}{\partial a_1} \right)^2 \text{cov}[a_1, a_1] + \left( \frac{\partial f}{\partial a_2} \right)^2 \text{cov}[a_2, a_2] + \left( \frac{\partial f}{\partial a_3} \right)^2 \text{cov}[a_3, a_3] +
\]

\[
+ 2 \left( \frac{\partial f}{\partial a_1} \frac{\partial f}{\partial a_2} \right) \text{cov}[a_1, a_2] + 2 \left( \frac{\partial f}{\partial a_1} \frac{\partial f}{\partial a_3} \right) \text{cov}[a_1, a_3] + 2 \left( \frac{\partial f}{\partial a_2} \frac{\partial f}{\partial a_3} \right) \text{cov}[a_2, a_3].
\]

\[
\text{cov}[a_3, a_3] = -\frac{(-2 + n)\mu_{1,1}^2}{(-1 + n)n} + \frac{\mu_{0,2}\mu_{2,0}}{(-1 + n)n} + \frac{\mu_{2,2}}{n},
\]

A. Abramov, S. Boogert
Regression Testing

• Rapid development of BDSIM
• Occasionally, simple / basic things break
• Code too large to test all features yourself
• Automated build & testing system implemented
• Each example is also a test
• Reference histograms and results compared
• RoBDSIM executable used for comparison and testing
• **145 tests** so far
• **Run nightly**

  • Hadronic & EM Shower development
  • Tracking in each component
  • Parser
  • Geometry construction
  • Geometry overlaps
  • Many more…. 
Python Utilities Galore

• pymadx, pybdsim, pymad8, robdsim, pytransport, pylhc
  • pymadx
    — loading and manipulation of TFS files
    — range iterating, filtering, matching
    — PTC segments supported
    — use to plot a lattice above a graph – interactive too!
  • pybdsim
    — conversion from Madx, Mad8, Transport etc
    — ASCII output analysis
    — programmatic model construction
  • pylhc
    — utilities for parsing lhcb model specific information
    — collimation files, aperture information (filtering, matching etc)
• Again, all open source and distributed with BDSIM
Documentation

- New manual (html & pdf) automatically updated weekly
  - lots of syntax examples
  - www.pp.rhul.ac.uk/bdsm/manual
- Detailed Doxygen code documentation similarly
  - www.pp.rhul.ac.uk/bdsm/doxygen
Public Git Access

- www.bitbucket.org/jairhul/bdsim
- Full open source development
- Issue tracking - (100 this year, 20 open)
- ~ 10 regular developers
- ~ 5 branches

Many developers working at once without issue on many versions

300 – 500 commits per version
3 releases per year typically

A successful git branching model
Timeline

• Autumn 2015
  — v0.9 released next week
  — Finish magnet support & BLM geometry
  — Implement magnet coils
  — Implement magnet pole face rotation flexibility
  — Implement Racetrack aperture
  — Complete regression testing suite
  — Support of crystal collimation processes & geometry

• V1.0 by end of 2015
  — paper in preparation

• Spring 2016
  — Factorise tracking
  — Include external tracking library
  — Consider incorporating collimation specific processes
Summary

- BDSIM has great potential for a variety of applications
- Complimentary to existing codes
- Under rapid development – 7 regular developers
- Flexible and easy to explore new scenarios
- Weekly meetings & bi-annual code weeks
- Clear road map for development
- Paper in preparation
Thank you

Acknowledgements: Regina, Stefano, Roderik, Andrea, Hector

www.pp.rhul.ac.uk/bdsim
Reference Slide for URLs

- *We’ve moved!*
  - [www.bitbucket.org/stewartboogert/bdsim](http://www.bitbucket.org/stewartboogert/bdsim)
  - [www.bitbucket.org/lnevay/pybdsim](http://www.bitbucket.org/lnevay/pybdsim)
  - [www.pp.rhul.ac.uk/bdsim](http://www.pp.rhul.ac.uk/bdsim)
  - [www.pp.rhul.ac.uk/bdsim/doxygen](http://www.pp.rhul.ac.uk/bdsim/doxygen)
  - [www.bitbucket.org/jairhul/bdsim](http://www.bitbucket.org/jairhul/bdsim)
  - [www.bitbucket.org/jairhul/pymadx](http://www.bitbucket.org/jairhul/pymadx)
  - [www.bitbucket.org/jairhul/pybdsim](http://www.bitbucket.org/jairhul/pybdsim)