MERLIN FOR LHC COLLIMATION*

H. Rafique†, R. J. Barlow, University of Huddersfield, Huddersfield, UK
R. B. Appleby, S. C. Tygier, Cockcroft Institute and The University of Manchester, Manchester, UK
R. Bruce, S. Redaelli, CERN, Geneva, Switzerland

Abstract

MERLIN version 5 delivers a new tool for Large Hadron Collider (LHC) collimation. Significant upgrades and optimisation have been performed to bring forth a new release of the C++ accelerator physics library. Version 5 offers modularity, including scattering routines with many physics models, which may be user supplemented trivially. The extensibility of MERLIN offers the ad hoc addition of; physics processes such as collimation, lattice elements such as the hollow electron lens (HEL), and tracking integrators. Accurate beam physics, including: synchrotron motion, symplectic tracking, and proton collimation are demonstrated. Using the LHC collimation system to produce loss maps at a beam energy of 7 TeV we validate the capabilities of MERLIN 5.

INTRODUCTION

MERLIN [1, 2] is a C++ library used for accelerator physics. Originally developed by Nick Walker *et. al.* at DESY to study the ILC [3]. For the past few years it has been developed in collaboration with the LHC collimation group for simulation of the LHC collimation system. Until recently, two code strands existed, the HEL version [4, 5], and the loss map version [6]. The two strands have now been merged into release 5, adding functionality, better structure, optimisation, and user friendliness. Here we concentrate only on MERLIN's use in LHC collimation, though as a library it has may other capabilities.

LHC COLLIMATION

The ultimate goal when simulating the LHC collimation system is the production of loss maps, which record the likely location of proton losses in the LHC to a 10 cm resolution. For loss map production, a simulation must include many elements. As well as accurate tracking, synchrotron motion, and proton scattering, a complete model of the LHC, it's apertures, and all collimator settings and materials are required. `Monospace` text denotes MERLIN classes.

Using a MADX [7] generated TFS table, MERLIN's `MADInterface` class constructs an `AcceleratorModel`, which holds a vector of `AcceleratorComponents`. Each `AcceleratorComponent` has an `Aperture`, an `AcceleratorGeometry`, and an `EMField`. Aditionally, collimator elements have a `Material`.

As the LHC is a synchrotron, many turn simulations require accurate modelling of RF cavities, this is illustrated in Fig. 1, where a bunch is populated in *dp ct* phase space and simulated in the nominal 7 TeV LHC. The colour gradient runs from yellow to red, where yellow is initial turns and red final turns. Only stable particles inside the bucket will survive long enough to be plotted in red, the others being lost.

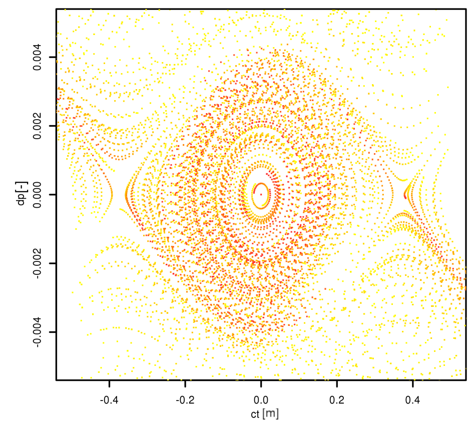

Figure 1: Poincaré section in *dp ct* phase space of the RF bucket for the nominal LHC. The colour gradient from yellow to red indicates the turn that the particle is recorded at.

A file containing the jaw openings, rotation, and material of all collimators is read by the `CollimatorDatabase` class, which sets these variables in the `AcceleratorModel` collimators. As the MADX TFS table doesn't contain an accurate aperture model an aperture file is read by the `ApertureConfiguration` class which iterates through each element in the `AcceleratorModel`, and sets the appropriate `Aperture` for it. An `InterpolatedAperture` exists to provide a better representation of the machine aperture, for example if multiple apertures are recorded within a single element. An example of the apertures created, for IR7 in the nominal LHC, is shown in Fig. 2.

As SixTrack [8] apertures are taken from the post-processing files they do not display the collimators, however these are identical as they are set using an identical collimator input file. We see that all other apertures are near identical.

Proton Bunch

The `LatticeFunction` class uses the `AcceleratorModel` to calculate the optics functions of the machine at each element, including; the β and α functions, fractional phase advance, and the closed orbit. A separate `Dispersion` and `PhaseAdvace` class exists, the latter to include the full phase advance μ, and the phase advance between any two accelerator elements. The lattice functions of the high luminosity (HL) LHC calculated in MERLIN, are shown in Fig. 3,

* Research supported by FP7 HiLumi LHC (Grant agreement 284404)
† haroon.rafique@hud.ac.uk

ISBN 978-3-95450-136-6

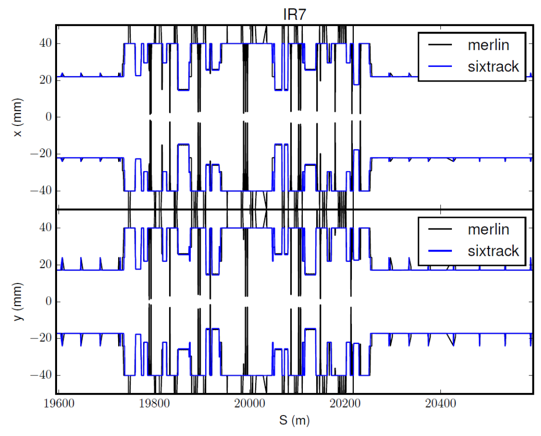

Figure 2: Apertures in IR7 in the nominal LHC, comparing MERLIN with SixTrack.

where they are compared, and seen to be identical to those calculated in MADX.

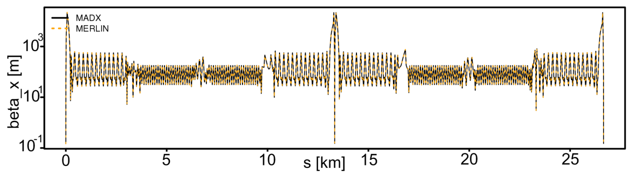

Figure 3: β_x functions for the HL LHC, comparing MERLIN (orange), and MADX (black).

A BeamData class uses the LatticeFunctions to define a ProtonBunch that accurately represents the machine bunch at any point in the accelerator.

Tracker

The ParticleTracker class takes the AcceleratorModel and the ProtonBunch as arguments, and iterates the ProtonBunch through each AcceleratorComponent in the AcceleratorModel.

The modular nature of Merlin extends to the integrators used for tracking. An integrator set, for example TRANSPORT, consists of a number of classes that each implement an integration algorithm for a set of elements, e.g. drift, bend, quadrupole etc. The tracker can be configured to use a complete integrator set or a mix of implementations, for example the user may wish to swap in a specific algorithm for some element types. MERLIN provides TRANSPORT (the default) [9], THIN_LENS and SYMPLECTIC integrator sets. A comparison of the TRANSPORT and SYMPLECTIC trackers is shown in Fig. 4.

If any PhysicsProcesses are attached to the tracker, they are called before tracking at the required elements. The structure of an LHC collimation run is shown in Fig. 5.

COLLIMATION PROCESS

The collimation process flow is shown in Fig. 6. CollimateParticleProcess identifies first whether or not the AcceleratorComponent is a Collimator and contains a

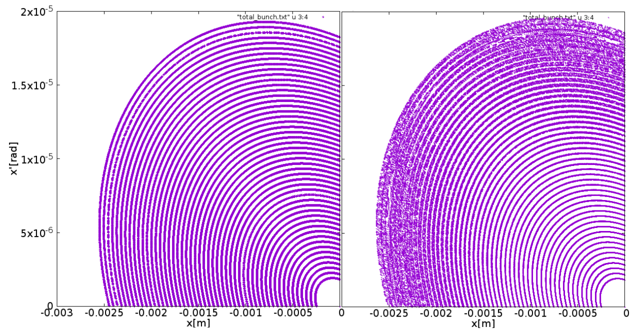

Figure 4: Poincaré section in the nominal LHC for particles with x values between 1-10σ_x, comparing the TRANSPORT (left) and SYMPLECTIC (right) integrators.

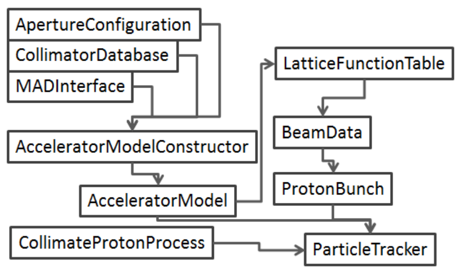

Figure 5: Flow of a standard LHC collimation simulation.

ScatterAtCollimator flag. When set to true, full collimation is performed at collimator elements, when false (the case for all non collimator elements), all particles outside the aperture are absorbed. Next we check if any particles hit the collimator jaw. When this is the case, we iterate through the rest of the particles and perform collimation.

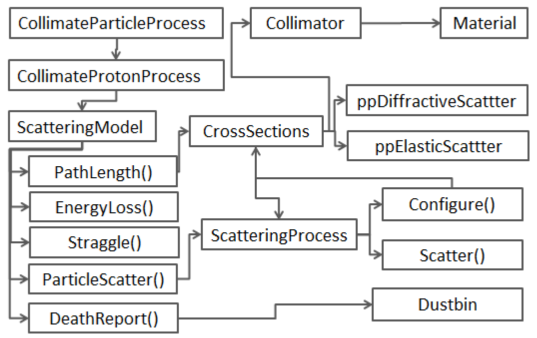

Figure 6: Flow of the collimation process.

The CollimateProtonProcess holds a pointer to the ScatteringModel class, which contains the main collimation functions. First the PathLength function is used to initialise the CrossSections class, which obtains all relevant information from the process, element, and bunch, to calculate cross sections for all ScatteringProcesses. This initialisation is done only once for each collimator jaw material, and the CrossSections object is stored for later referral, thus optimising the run. Next, energy loss and multiple coulomb scattering are performed, followed by point like processes (ScatteringProcesses).

A dictionary of standard materials is present in MERLIN, the user may add their own material by specifying certain properties, this includes mixtures which will be useful when studying novel jaw materials for HL LHC collimation.

MERLIN contains nine `ScatteringProcess`es; SixTrack+K2 [8], and advanced versions of; proton nucleon elastic, proton nuclear elastic, single diffractive, Rutherford, as well as inelastic scattering. MERLIN's own elastic and single diffractive scattering have recently undergone rigorous improvements [10]. As well as this, the ionisation process contained in the EnergyLoss function has the option of the simple SixTrack+K2 like, and advanced implementation.

As shown in Table. 1, there exist five preset combinations of `ScatteringProcess`es and ionisation functions. The combination may either be specifically chosen by the user, or a preset may be selected using the `ScatteringModel`::SetScatterType(*n*) function, where *n* is the enumerator corresponding to the column headings shown in Table 1.

Table 1: Preset Combinations of `ScatteringProcess`es and Ionisation. Note that all combinations include an inelastic process

Process	**0**	**1**	**2**	**3**	**4**
Rutherford	ST	ST	ST	ST	M
pn Elastic	ST	M	ST	ST	M
pN Elastic	ST	M	ST	ST	M
Single Diffractive	ST	ST	M	ST	M
Ionisation	ST	ST	ST	M	M

As well as a proton collimation process, MERLIN 5 contains two new processes related to LHC collimation. The first, the HEL process, has been used to study the effect of HELs as collimation enhancers in the nominal and HL LHC [4, 5]. The second, a crab cavity failure process (`CCFailureProcess`), is under development, and will allow for the simulation of voltage and phase failures of crab cavities, which are present in the HL LHC.

Performance enhancements have been made in MERLIN 5, including optimisations to memory access. This has resulted in an approximate 40% increase in throughput. For 200 turn LHC loss maps MERLIN tracks around 57 protons per second per core on a 3.3 GHz Xeon E5-2643, making a full 6.4 million particle loss map take around 30 CPU hours. Multiple processes or MPI can be used to distribute MERLIN to hundreds of cores.

LOSS MAPS

MERLIN loss maps are comparable with SixTrack generated loss maps, as shown in Fig. 7, in which both codes are run for 200 turns in the nominal LHC, SixTrack with $6.4 \cdot 10^6$ particles, MERLIN with $3 \cdot 10^7$. MERLIN's advanced scattering routines, increased speed, and multiple trackers, mean that it is perfectly adapted for LHC collimation studies. This will soon include a study of loss maps for points in the squeeze for the HL LHC.

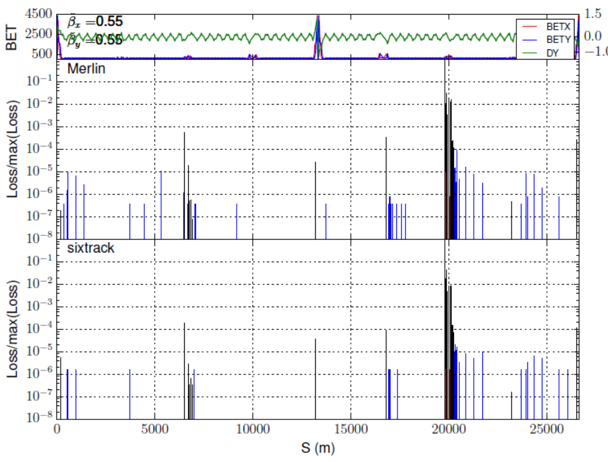

Figure 7: Nominal LHC loss map, comparing MERLIN (above), with SixTrack (below). The topmost plot shows the optics functions.

CONCLUSION

MERLIN 5 includes new; collimation structure, physics processes (HEL and CCFailure), scattering, and optimisation. Together with handling of composite materials, synchrotron motion, and symplectic tracking, MERLIN is a complete tool for LHC collimation studies, and after successful comparisons, may be used for nominal and HL physics cases. In particular, MERLIN may be exploited to study novel materials and collimation schemes for the HL LHC upgrade, including use of HELs as enhancers [5], and the impact of crab cavity failures on collimation.

ACKNOWLEDGMENT

The authors would like to acknowledge the significant contribution to MERLIN by former developers James Molson and Maurizio Serluca over the past few years.

REFERENCES

[1] `http://merlin-pt.sourceforge.net/`

[2] `https://sites.google.com/site/haroonrafiquemerlin/`

[3] D. Kruecker, F. Poirier, N. Walker, "Merlin-based start-to-end simulations of luminosity stability for the ILC", Proc. PAC2007.

[4] H. Rafique *et al.*, "Simulation of hollow electron lenses as LHC beam halo reducers using MERLIN", Proc. IPAC15, Richmond, Virginia, 2015.

[5] H. Rafique *et al.*, "High luminosity LHC hollow electron lens collimation using MERLIN", presented at ICAP'15, Shanghai, China, paper MODBC2, these proceedings.

[6] J. Molson *et al.*, "Advances with MERLIN - a beam tracking code", Proc. IPAC10, Kyoto, Japan, 2010.

[7] `http://madx.web.cern.ch/madx/`

[8] `http://sixtrack.web.cern.ch/SixTrack/`

[9] K. L. Brown *et al.*, SLAC-91, Rev. 3, 1983.

[10] J. Molson, "Proton scattering and collimation for the LHC and LHC luminosity upgrade", PhD Thesis, The University of Manchester, 2014.