SIMULATING THE LHC COLLIMATION SYSTEM WITH THE ACCELERATOR PHYSICS LIBRARY MERLIN, AND LOSS MAP RESULTS

J.G. Molson*, R.B. Appleby, M. Serluca, A. Toader
The University of Manchester and the Cockcroft Institute, UK
R.J. Barlow, University of Huddersfield, UK

Abstract

We present large scale simulations of the LHC collimation system using the MERLIN code for calculations of loss maps, currently using up to 1.5×10^9 halo particles. In the dispersion suppressors following the collimation regions, protons that have undergone diffractive interactions can be lost into the cold magnets. This causes radiation damage and could possibly cause a magnet quench in the future with higher stored beam energies. In order to correctly simulate the loss rates in these regions, a high statistics physics simulation must be created that includes both accurate beam physics, and an accurate description of the scattering of a 7 TeV proton in bulk materials. The current version includes the ability to simulate new possible materials for upgraded collimators, and advances to beam-collimator interactions, including proton-nucleus interactions using the Donnachie-Landshoff Regge-Pomeron scattering model. Magnet alignment and field errors are included, in addition to collimator jaw alignment errors, and their effects on the beam losses are systematically estimated. Collimator wakefield simulations are now fully parallel via MPI, and many other speed enhancements have been made.

INTRODUCTION

The LHC is a superconducting 7 TeV proton-proton collider with a high nominal stored beam energy (360MJ) and a low quench limit on the superconducting magnets $\left(4.5\mathrm{mW/cm}^3\right)$ [1]. To protect the machine from this high stored energy, the LHC is equipped with an efficient collimation system to collimate halo particles and prevent quenching, in addition to reducing the background at the experimental regions and preventing radiation damage to sensitive electronics. There exist two collimation regions - one in interaction region 7 (IR7) which contains a series of betatron collimators for transverse collimation. The primary collimators in this regions are the aperture restriction in the machine. In IR3 there is a region of beam dispersion to perform momentum collimation.

Of critical importance are regions known as the dispersion suppressors, which match the long straight section (LSS) optics to the periodic optics of the arcs. In these regions, the dispersion rises rapidly, and any protons that have undergone any interactions in the collimators that causes them to lose momentum may be lost in a localised region, see Figure 4. Due to this, one must have an accurate simulation of the accelerator optics, the machine physical aperture, and the scattering physics of a proton inside a collimator jaw.

Merlin is a C++ accelerator physics library [2] initially developed for the ILC beam delivery system [3, 4], then later extended to model the ILC damping rings. Merlin has been extended to be used for large scale proton collimation simulations, with the aim of providing an accurate simulation of the Large Hadron Collider (LHC) collimation system, and any future upgrades. In this paper we describe the developments of the Merlin code to enable study of the LHC collimation system and present beam loss maps for 2012 running.

THE MERLIN ACCELERATOR PHYSICS LIBRARY

The Merlin library consists of a large number of classes designed to simulate a particle accelerator, and any additional systems required. The classes can be split into three main categories.

The *AcceleratorModel* and associated classes deal with the creation and storage of an accelerator lattice. The lattice is stored as a series of *AcceleratorComponent* classes, which contain information about each element. Different types of accelerator component are child classes of the main *AcceleratorComponent* class. These contain pointers to classes describing specific properties of the element: *EMField* describes any electromagnetic fields inside the element, *AcceleratorGeometry* describes any geometric transforms that the element has undergone, *Aperture* describes the experimental beam pipe, and *WakePotentials* describe any wake fields that exist for this element class. Input can take place via multiple methods: the direct creation and addition of elements, via the MAD-X [5] TFS output (*MADInterface*), or tape format (*XTFFInterface*), both of which create an *AcceleratorModel* as output.

The *ParticleTracker* and associated classes deal with the transport of particles along the accelerator optical lattice, including stepping between elements, and within individual elements, whilst applying additional physics processes at appropriate locations. These create integrator sets for tracking, and individual integrators can be overridden for selected class types, e.g. crab cavities. The *ParticleTracker* takes as its input a *ParticleBunch* class, and a *Beamline*, where the *ParticleBunch* can be one of many different types, e.g. gaussian, flat and ring amongst others. The *Beamline* is a subsection of an *AcceleratorModel*, and bunches can be passed between multiple trackers, allowing

* james.molson@hep.manchester.ac.uk

ISBN 978-3-95450-116-8

situations such as the transfer between different accelerators to be simulated.

A series of *BunchProcess* classes exist to apply additional physics within elements. These can include wakefields, collimation, synchrotron radiation, and others. Templates exist for such classes, and it is straightforward for new users to add additional physics of their choice via this method, without having to adjust other parts of the library.

This modular design allows a user to use as much or as little of the library as they wish. In addition, if one wishes to investigate additional physics relevant to their accelerator system, a new tracking code does not need to be written, but simply a new *BunchProcesses* class can be created. An example simulation run is shown in Figure 1.

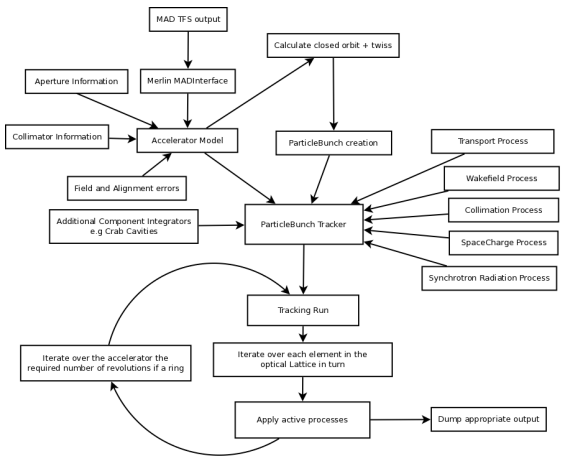

Figure 1: An example logical flow of a Merlin run.

LHC LOSS MAPS RESULTS

Merlin is used to simulate the 2012 as-built optics of the LHC in order to generate loss maps. These can be used to define the maximum possible safe beam current, and indicate areas which may need additional collimators or shielding. The first stage of this calculation is the construction of an accurate optical model of the LHC in Merlin. The machine optics are generated by MAD-X and are used as the input to Merlin. Inside Merlin, the *LatticeFunctions* class calculates beam parameters using Merlin's integrators. A comparison between Merlin and MAD-X of the β-functions and the linear dispersion is shown in Figure 2, and excellent agreement is found in all regions of the machine.

Loss maps can be generated using different optics configurations, e.g. the β-function at the interaction points (β^*), beam crossing angles, and so on. The *CollimationProcess* simulates all proton-collimator interactions and performs all aperture checking. If a proton undergoes an inelastic interaction or touches the beam pipe it is considered lost. If this takes place, the particle is removed from the bunch and the location at which this takes place is recorded. This can be done at any desired longitudinal accuracy, and by default a bin size of 10cm is used.

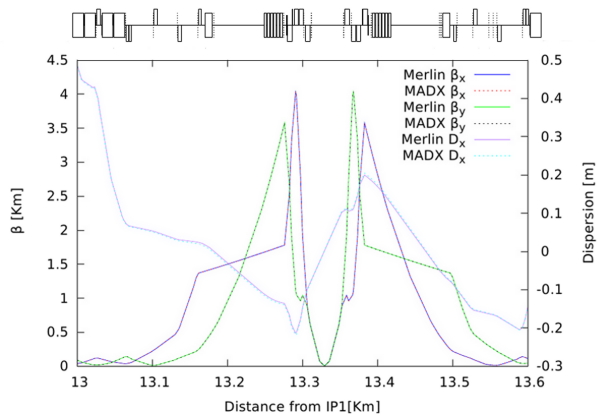

Figure 2: A comparison of optics between MAD-X and Merlin showing the β-functions and dispersion in IR5. Excellent agreement is found for both this region and the entire ring.

Figures 3 and 4 show the example loss map for 2012, 4 TeV running conditions for beam 1. A horizontal beam halo (a ring in x, x' normalized phase space, 0 in y, y') is used, which is then transformed into physical coordinate space. The initial impact parameter with the collimator can be adjusted, and in this case $1\mu m$ is used. Beam is injected at the closest (primary) collimators in IR7, and tracked for 200 turns.

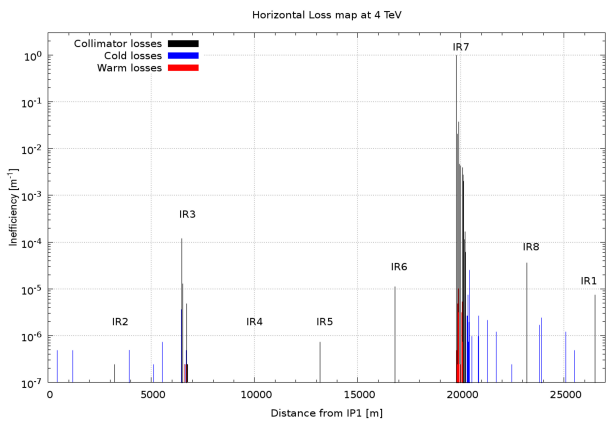

Figure 3: An example collimation loss map for 4 TeV 2012 running conditions. The initial simulated beam halo is a purely horizontal halo.

The loss map variable plotted is cleaning inefficiency around the ring, defined as:

$$\eta = \frac{n_{\text{abs}}}{\Delta s \times n_{\text{total}}}$$

where η is the inefficiency, Δs is the bin size, n_{abs} is the loss count in that bin, and n_{total} is the total number of losses.

As can be seen, the highest loss locations are in the collimation regions, specifically the IR7 betatron collimation

ISBN 978-3-95450-116-8

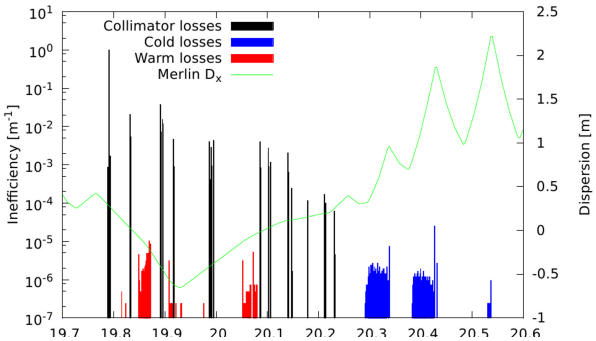

Figure 4: A zoom of the 4 TeV 2012 running conditions loss map focusing on the betatron collimation region in IR7 allowing the losses in the dispersion suppressor to be seen. The horizontal dispersion is also shown as the green line.

region. Figure 4 shows a zoom of this region. Here the effect of dispersion can be seen on the losses. Protons are lost in cold regions following the collimation region, which is minimised in the design.

Lattices have been generated in MAD-X which involve both field and alignment errors on dipole, quadrupole and sextupole magnets, to allow the effects of any lattice errors on losses to be estimated. These error configurations are then corrected using the available corrector circuits in the optical model. Collimators are aligned to the un-errored reference orbit, and then both the magnet errors are added, followed by corrections. Loss maps are generated in this configuration and it is found that as long as corrections are applied there is little quantitative difference to the loss maps generated. This includes both the locations of lost protons and the magnitude of losses.

UPDATED SCATTERING PHYSICS

In order to more accurately simulate the losses in the cold dispersion suppressor regions, more accurate simulation physics must be used over the current generation of codes. New models of proton-proton interactions have been developed, with the aim of expanding these to proton-nucleus interactions. Focus has been on two types of scattering, elastic and single diffractive scattering taking into account both theoretical considerations and experimental high energy physics data. Elastic interactions will give an angular kick to the outgoing proton, increasing the size of the beam halo, with the possibility of the proton exiting the dynamic aperture [6]. Single diffractive interactions can allow a proton to exit with an angular kick, and an energy loss [7]. This energy loss will move the outgoing proton away from the reference momentum, hence on entry to a dispersive region, they will undergo large orbit excursions, and collide with the accelerator beam pipe. Since the fits to these cross sections are mathematically highly complicated, it is faster computationally to pre-generate these distributions in an array, and interpolate them as required. The updated scattering physics will be described in a future publication.

OTHER ENHANCEMENTS

New materials, such as composites are now supported. Since Merlin is a C++ library, this is enabled via creating a new material class which inherits from the base material class, and specific access functions that are defined as virtual can be overridden. For example *GetdEdx()* will return the mean energy loss for the material mixture, whereas *GetElasticCrossSection()* will return the cross section for a randomly chosen nucleus within the material (weighted depending on the material composition). The same function calls are used for both a pure element, and material mixtures in order to model novel collimation materials.

Wakefield calculations are now fully parallel due to a parallel bunch moment calculation. Previously, particles were transferred between threads to a single node, where the wakefield calculation took place. Particles were then redistributed. This gave a speed increase over single threaded operation since standard tracking could take place in parallel. It is not the most efficient method, since a large quantity of bandwidth is required to transfer particles, and whilst this calculation takes place, CPU cores sit idle. Now the mean and standard deviation are calculated in parallel (for bunch slicing), and data is shared via a call to the *Allreduce* function.

CONCLUSION

In conclusion, the accelerator physics code Merlin has been extended in many areas to make detailed studies of the LHC collimation system and calculate halo loss maps. The loss maps have been produced for 2012 4 TeV running, and Merlin is ready to be used for studies of the LHC upgraded collimation system.

ACKNOWLEDGEMENTS

We wish to thank A. Donnachie for his assistance with the scattering physics and allowing usage of his models, and the CERN LHC collimation group for many interesting discussions.

REFERENCES

[1] O. Brüning et. al. “LHC Design Report” 2004

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

[3] D. Krücker and F. Poirier and N. Walker, “Merlin-based start-to-end simulations of luminosity stability for the ILC” PAC 2007, THPAN023

[4] F. Poirier and D. Krücker and N. Walker, “An ILC main linac simulation package based on Merlin” EPAC 2006, MOPLS065

[5] `http://mad.web.cern.ch/mad/`

[6] A. Donnachie and P.V. Landshoff, “Elastic Scattering at the LHC” arXiv:1112.2485v1 [hep-ph]

[7] A. Donnachie and P.V. Landshoff, “Soft diffraction dissociation” arXiv:hep-ph/0305246v1

ISBN 978-3-95450-116-8