Simulating the LHC Collimation System with the Accelerator Physics Library MERLIN, and Loss Map Results

J. Molson, R. Appleby, R. Barlow, M. Serluca, A. Toader

University of Manchester, Cockcroft Institute, University of Huddersfield

20th August, 2012
The Large Hadron Collider (LHC)

- 7 TeV proton-proton synchrotron, 26.65km length
- Beams collide at 4 experimental regions - (ATLAS, ALICE, CMS, LHCb)
- 2 collimation regions
- Additional regions for RF, and the beam dump
- Injection at 450 GeV, ramp to up 7000 GeV (Currently running at 4000 GeV)
- Superconducting magnet system, 1.9K, 8.33T dipoles
- High stored beam energy!
Why do we need to collimate

- 360MJ stored beam energy.
- $4.5 \text{mW/cm}^3$ will quench a magnet at top energy!
Collimation

Impact parameter \( \leq 1 \mu m \)

Core

Primary halo (p)

Beam propagation

Unavoidable losses

Secondary halo

\( \pi \)

\( p \)

Shower

\( e \)

\( p \)

Tertiary halo

Absorber

Absorber

Absorber

CFC

CFC

W/Cu

W/Cu

Superconducting magnets

SC magnets and particle physics exp.
## Collimator families

<table>
<thead>
<tr>
<th>Collimator</th>
<th>Type</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>Primary</td>
<td>C</td>
</tr>
<tr>
<td>TCSG</td>
<td>Secondary</td>
<td>C</td>
</tr>
<tr>
<td>TCLA</td>
<td>Absorber</td>
<td>W</td>
</tr>
<tr>
<td>TCT</td>
<td>Triplet protection</td>
<td>W</td>
</tr>
<tr>
<td>TCLP</td>
<td>Physics debris Absorber</td>
<td>Cu</td>
</tr>
<tr>
<td>TCDQ</td>
<td>Dump protection</td>
<td>C</td>
</tr>
<tr>
<td>TDI</td>
<td>Injection protection</td>
<td>C</td>
</tr>
</tbody>
</table>
What is Merlin?

- C++ Accelerator physics library
- Provides a set of useful functions for accelerator modelling
- Initially used to simulate ground motion in the ILC BDS and linac
- Later the ILC damping rings
- Written by Nick Walker et al (DESY)
- Now adapted for large scale proton collimation simulations by Manchester and Huddersfield
- Three main sections of the library:
  - Accelerator lattice loading/creation and storage
  - Tracker
  - Physics processes
- Modular design - easy to modify and extend
Accelerator Lattice

- Can load directly from MAD (TFS table output)
- Can also use XTFF format
- Direct element addition
- The created AcceleratorModel element can be further manipulated in the future, e.g. adjust aperture, alignment errors, etc
- AcceleratorComponent: The base class for each element in the lattice that all elements inherit from.
- EMField: The field associated with the element
- AcceleratorGeometry: Any Geometry transforms for the element, e.g. tilt
- Aperture: The aperture for the element, e.g. the beam pipe or collimator jaws
- WakePotentials: Any wakes for the element - resistive wall, geometric and cavity wakes
Different types of tracker, particle tracking and bunch moment tracking

- Takes the input of a bunch and beamline, and tracks the bunch along the beamline
- Can use specific integrator sets, e.g. transport, thin lens, symplectic
- Can override specific integrators, e.g. crab cavities
- Step both along the accelerator lattice and within individual elements
Physics processes

- Additional physics on top of tracking to be applied at selected elements and positions
- Can be enabled or disabled as required - processes are attached to trackers
- Examples: Synchrotron radiation, collimation, wakefields, etc
- Easy to create, template examples exist
- Trackers manage stepping within processes - inputs are the AcceleratorComponent and bunch
Accelerator errors

- Can offset element positions, x,y,z
- Can adjust angular tilts
- Can add in field errors including additional multipoles
- Can generate errors inside Merlin
- We generate errors in MAD, correct for errors, then transfer this information to merlin
- Tested loss maps with an errored and corrected lattice, with collimators aligned to the perfect orbit
- Little difference from the perfect configuration in loss maps
Parallel running

- Wish to run large simulations - very cpu heavy - use MPI
- Must use multiple physical machines with interconnects
- Run multiple copies of the same binary that can communicate with each other
- Tracking, collimation, etc, are all independent on a per-particle basis, do not need any knowledge about other particles
- Collective effects such as space charge and wakefields do require this information
- Functions exist such as parallel bunch moment calculations (mean, standard deviation) in addition to the ability to move particles between computers
- Parallel running is implemented at a per process algorithm level
Parallel running

MPI::Init() → Bunch Creation → Linear Tracking
Node 1 → Node 2 → Master → Node n-1 → Node n
Collective effects

Linear Tracking → Master → Node n-1 → Node n

MPI::Finalize()
Example run

1. MAD TFS output
2. Calculate closed orbit + twiss
3. ParticleBunch creation
4. ParticleBunch Tracker
5. Tracking Run
6. Application of active processes
7. Output dumping

Currently working on parallelising processes that contain collective effects.

*Example:* Wakefield + Space charge effects

This is Algorithm dependent.
General simulation strategy

1: Comparison and benchmarking with existing codes - Sixtrack, MAD-X
2: Enhance features - e.g. New scattering, new materials
3: Predict future operations - e.g. New collimator/optics layouts, new collimator materials and methods
Currently working on 1 and 2 in parallel as will be shown
Collimation simulation configuration

- Want to have a comparison with Sixtrack
- Thick-lens version V6.5.2012.02.seq
- Using Beam 1
- $\beta^*$ for IP1 and IP5: 0.6m
- $\beta^*$ for IP2 and IP8: 3m
- 6.4 M particles simulated
- No field or alignment errors
- Energy = 4 TeV, $\epsilon_n = 3.5$ mm-mrad, $dp/p = 0$, $\sigma_z = 0$
- Crossing angle [$\mu$rad]: X1=-145, X2=-90, X5=145, X8=-220
- Parallel separation on at all IP: sep = $\pm 0.65$mm
- Horizontal halo distribution
- Impact parameter = $1\mu m$ and 10 cm longitudinal loss resolution
Collimation simulation configuration

- Beam injected at TCP.C6L7.B1 (primary horizontal collimator)
- Collimators aligned to orbit and beam envelope

<table>
<thead>
<tr>
<th>Collimator</th>
<th>Aperture (σ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP IR7</td>
<td>4.3</td>
</tr>
<tr>
<td>TCP IR3</td>
<td>12</td>
</tr>
<tr>
<td>TCSG IR7</td>
<td>6.3</td>
</tr>
<tr>
<td>TCSG IR3</td>
<td>15.6</td>
</tr>
<tr>
<td>TCLA IR7</td>
<td>8.3</td>
</tr>
<tr>
<td>TCLA IR3</td>
<td>17.6</td>
</tr>
<tr>
<td>TCLP</td>
<td>10</td>
</tr>
<tr>
<td>TCT IR1/IR5</td>
<td>9</td>
</tr>
<tr>
<td>TCT IR2/8</td>
<td>12</td>
</tr>
<tr>
<td>TCDQ IR6</td>
<td>7.6</td>
</tr>
<tr>
<td>TCLI</td>
<td>open</td>
</tr>
<tr>
<td>TDI</td>
<td>open</td>
</tr>
</tbody>
</table>
Beta functions

MAD-X vs. Merlin Optics

Merlin $\beta_x$
MADX $\beta_x$
Beta functions - IR7

The graph depicts the beta functions across different scales, comparing Merlin and MADX results. The x-axis represents the scale [m], and the y-axis represents the beta function [m]. The graph shows periodic peaks corresponding to different sections of the IR7, with Merlin β\(_x\) and β\(_y\) and MADX β\(_x\) and β\(_y\) curves indicated distinctly.
Reference orbit
Loss map results comparison (Sixtrack plots from LHC collimation group (R. Bruce))
Loss map Results IR7

MERLIN

SIXTRACK

J. Molson, R. Appleby, R. Barlow, M. Serluca, A. Toader

ICAP 2012 22/28
Enhanced scattering physics

- Have been working on enhanced scattering physics inside a collimator jaw
- Nuclear interactions - pA scattering
- Nucleon scattering: elastic and single diffractive
- Higher level electron interactions - multiple coulomb scattering, atomic ionization
- Do not care about any secondary particles, similar to sixtrack
- Aiming to be precise and fast
- Our range of interest is of beam energy between 450 and 7000 GeV/c
- This is \( \sqrt{s} = 30 \rightarrow 115 \text{GeV} \) for fixed target interactions
Elastic scattering

- Interested in the differential cross section $\frac{d\sigma}{dt}$
- Fit all appropriate existing $pp$ and $p\bar{p}$ data
- Data exists on either side of the region of interest so interpolation is possible
- Add low $t$ coulomb peak to the fit
Elastic scattering experimental data

Preliminary Results

- 30.54 GeV pp
- pp Fit at 30.54 GeV

- 44.64 GeV pp
- pp Fit at 44.64 GeV

- 52.8 GeV pp
- pp Fit at 52.8 GeV

- 62.5 GeV pp
- pp Fit at 62.5 GeV
Single diffractive Scattering

- Incident proton interacts with a target nucleon
- Exits with reduced energy ($M_X$), and an angular kick
- Two regions of $M_X$: baryon resonances at low mass, triple regge at higher mass

\[ t \]

\[ a \quad p \quad \rightarrow \quad \xi p \quad \rightarrow \quad a \]

\[ b \quad p' \]

Diffractive dissociation at $\sqrt{s} = 81$ and 115 GeV

- Diffractive dissociation: $pp \rightarrow pX$
- Data of variable quality
  - Normalisation difference of 15% between the two principal experiments
  - Only one experiment at small $t$ ($t = -0.05$) above required energies so need to extrapolate

- Two very different regimes of $M_X$
  - $M_X \lesssim 2.5$ GeV: dominated by baryon resonances
  - $M_X \gtrsim 2.5$ GeV: triple-regge model
Tripple Regge exchange region

- Fitting over 6000 data points, over 8 experiments!
- More than 70% of the cross section exists at large $M_x$
Conclusions

- We are developing the code Merlin to operate with proton machines for high energy collimation simulations.
- When running in sixtrack like scattering mode, similar loss maps to sixtrack are generated for 4 TeV 2012 running.
- Enhanced scattering physics models are almost complete and will soon be implemented into Merlin.
- New material classes allow simulation of novel collimator materials, e.g. SiC, CuD, etc.
- Parallel running allows large scale simulations.
- We welcome new code users (and brave developers).
Aperture configuration

- Load aperture information file
  - Save data into a vector
  - Extract all AcceleratorModel elements
    - Loop over elements
      - If element is a collimator or has zero length - skip
      - Create a new aperture vector for this element
      - Check if the position of the aperture vector is greater than the element position
      - If we are at the start of the aperture list, extract the aperture from the last entry in the vector and add this to the element specific vector
      - Iterate over the aperture vector whilst we are still inside the component
      - Add one aperture entry past the current AcceleratorComponent
      - Create interpolated RectEllipse aperture and attach to element
        - Check if each sub-entry is constant along the element length.
        - Create interpolated circular aperture and attach to element
        - Move to next element until at the end of the Accelerator
      - Add aperture entry to element specific vector
        - If at the last entry, match with the entry from the start.
      - Now loop over element specific aperture vector
        - Check if all entries within each entry are the same.
        - Create RectEllipse aperture and attach to element
        - Create circular aperture and attach to element.
Collimation Process

For non-collimators, create a copy of the bunch at the entry of each element

Log the position in the bunch of any lost particles

Copy the lost particles from the initial conditions bunch copy into the new bunch

Create a new ParticleBunch

Create a new particle tracker and assign it the new bunch

While there are still particles remaining and we are inside the element, check aperture

Add lost particles to the lost particle list as for a collimator

Track a step ds

If at end of element, add any remaining particles to the last bin

Call DoOutput() function and clean up
Reference orbit - zoom x
Reference orbit - zoom y
Cross sections

\[ \sigma \text{ (mb)} \]

\[ \sigma_{pp \text{ total}} \]

\[ \sigma_{p\bar{p} \text{ total}} \]

\[ \sigma_{pp \text{ elastic}} \]

\[ \sigma_{p\bar{p} \text{ elastic}} \]