Recent developments in the Thick-GEM charging-up calculations and Garfield++ simulations acceleration.

Pedro M. M. Correia
University of Aveiro
pmcorreia@ua.pt
Outline

• Charging-up calculations with Thick-GEMs
  • Recent developments
  • Discussion of a possible Garfield++ new class

• OpenMP integration with Garfield++
  • Code example
  • SpeedUp

• Conclusions
Charging-up calculation

• Iterative-based calculation

• Calculation of field maps require several FEM calls

• Dynamic method is available (http://dx.doi.org/10.1088/1748-0221/9/07/P07025)

• Only static method will be discussed for simplicity
Principle of superposition

• The need of several executions of FEM is tedious

• Solution: Calculate Potential field map + Charged field maps at the beginning of the simulation

• Garfield++ reads potentials as a vector of nodes vs E. Potential

• For each node, the new E. Potential for each iterations is calculated inside Garfield++!
Principle of superposition

• Ex: Thick-GEM, insulator with 20 slices

• For each voltage between electrodes, field map is calculated as usually (without charging-up)

• For each slice, the field map correspondent to 1 electron accumulated on the correspondent slice surface is calculated

• 22 field maps due to charges + 1 (at least) potential field map are needed for full simulation
Garfield++ Method

- A new Garfield++ class has been developed (~500 lines of code, up to now).
- Responsible to find the field maps (only ANSYS at the moment)
- Writes a temporary field map depending on the number of accumulated charges for each iteration
- It allows restart simulation at a specific iteration if previous field maps are stored
- (code for demonstration only)
Gain Results

- Thick-GEM
  - Insulator with 20 slices
  - Pitch 0.4 mm
  - Thickness 0.2 mm
  - Hole Diameter 0.2 mm
  - Rim 40 μm
  - $V_{THGEM} = 1000$ V
  - Standard temp. and pressure.

Assuming 10 kHz/mm$^2$
Gain Results

Assuming 10 kHz/mm²

Despite gain differences, time-behaviour is the same.
Still does not explain the long-term gain increase observed after some hours.

Experimental data. Same geometry. Unknown VTHGEM and Rate “The gain in Thick GEM multipliers and its time-evolution”, 2015 http://dx.doi.org/10.1088/1748-0221/10/03/P03026
Integration of OpenMP in Garfield++

- OpenMP is a parallelization tool first released in 1997, can be used with Fortran, C and C++
- Used for multithread parallelization of parts of the code
- Set of simple compiler directives and code lines are needed to parallelize simple applications:
  - Ex. Compilation flag for gnu:
    - `-fopenmp`

- Uses the Fork-join\(^1\) model of parallel execution

\[^1\]M. E. Conway. A multiprocessor system design. In Proceedings, November 12-14 1963

```cpp
// Typical C++
for (int i = 0; i < 8; ++i) {
    do_some_task(i);
}
```

```cpp
#include <omp.h>
omp_set_num_threads(8);
#pragma omp parallel for
for (int i = 0; i < 8; ++i) {
    do_some_task(i);
}
```
Integration of OpenMP in Garfield++

- Careful must be taken when objects are created outside the parallel region.
- Some Garfield objects need to be defined for each thread (as arrays of pointers), otherwise race conditions will happen between different threads.
- Parallel region is defined with the line `#pragma omp parallel for`.
- `Omp_set_lock(&writelock)` is called to lock everything that is not thread-safe or needs to be synchronized (ex: `std::cout`).
SpeedUp
OpenMP in Garfield++

• Lxplus processor:
  • Intel Xeon E312xx (Sandy Bridge)
  • 8 threads
  • 2.6 GHz
  • 4 Mb Cache Size

• Optimization of the code would allow even faster simulations.

• Speedup is not linear with the increase of available threads.
Conclusions

• Charging-up gain calculations are now performed completely inside Garfield++
  • A new class has been written, only needs the initial field maps corresponding of each insulator slice
  • No need to call FEM software after each iteration -> **time and memory saving**
  • Results coherent with previous simulations; Short-time gain evolution as described by experimental work.

• Garfield++ simulations acceleration with multithread tools - **OpenMP**.
  • Race-conditions must be take into consideration, otherwise software will crash.
  • Speed-up factor > 2 can be achieved with Lxplus machines with 8 threads.