Simulation features request: Trajectories and replayability

Rich Holmes
27 Mar 2015

There are a couple of features I would argue are important for simulations, which as far as I know standard GEMC 1.x does not provide. I've kludged them into what I do but I would argue they should be built into the simulation package.

One is trajectory access. Normally GEANT does not preserve trajectory information, that is, the information on the history of all particles it generates and propagates. Doing so uses a lot of time and memory and in many cases isn't useful. But it can be turned on. GEMC provides a SAVE_ALL_MOTHERS which does turn on trajectory storage. But all it does with the trajectory information is it stores the PID and vertex of the immediate mother of the hit particle in the hit output (and it can draw the ancestral tracks if you're doing graphics). For understanding backgrounds it's invaluable to have more information (track ID, momentum vector) about more ancestors (from the mother all the way back to the primary). That information is there, but GEMC doesn't provide a way to output it.

The other, and it's kind of related, is replayability. By that I mean, for example, suppose you want to study backgrounds by running 11 GeV e- into the target and seeing what hits the detectors. The hit rate may be very low, a hundred per million e- or something like that. And suppose, for instance, you want to know where these backgrounds are coming from, so you want full trajectory information on the hits. But generating and storing trajectories for millions of events consumes lots of time and storage. What makes more sense is to identify the events that give rise to detector hits, and then replay those events with trajectory storage turned on. Likewise if you want to step through your hundred hit events to look at them and see if you find patterns to them — you can't sit around for half an hour waiting for the next hit. You want to generate the hits overnight and then replay them interactively. To do this the software needs to store one or two things: the state of the random number generator (it has to be stored internally at the start of each event, and then written out at the end of the event if it produced a hit, or better yet if it produced a hit that passes some user defined cut), and the primary track input if you're using an external generator. Then you need to be able tell the software to run using the saved RNG information and the saved primary inputs. Again, I've kludged this and I think I even checked it into solgemc, but it really should be in GEMC or whatever framework we decide to use (in less kludgey form).