# Table of Contents

# Tips for improving CPU Performance of programs using Geant4

Some suggestions for obtaining good/better CPU performance in Geant4 simulations.

- Critical parts of user code in a Geant4 simulation
- First tips for a performant Geometry Description
- Describing the electromagnetic field as Geant4 expects
- Selection of Physics Lists
- Selection of options for Geant4 electromagnetic (EM) physics
- Tips for Event Biasing usage

## Critical parts of user code in a Geant4 simulation

A first list of user-written code which is time critical - and things to avoid in each one:

- *User Stepping Action* . It is the **most important** user code: it is called every single step of the simulation, so if you really need it must be very small and fast. Avoid any heavy operation if possible, especially combining two or more: string comparisons, nested loops, .. ;
- *Sensitive Detector:: Compute Hits* - it is called only for each step of any track in the sensitive volume, so its impact is proportional to the fraction of steps inside the sensitive volume. If most steps are inside a particular sensitive volume (eg crystal) ensure that this one has none or very few heavy operations (see last point);
- *User Tracking Action* methods: Called for every track - many of which can have just one or two steps. Each method should have as few heavy operations as possible.
- *User Event Action* methods: Called for every event - keep it light if events are relatively fast t <~ O(1 sec) . In particular avoid/reduce writing to a file after every event in this case - else it can slow the simulation. ( Saving the RNG state after every event makes sense if a typical event takes more than a minute to simulate! )

## First tips for a performant Geometry Description

A few things to use, if possible, in describing a geometry:

- Use CSG solids, if appropriate, are faster than others;
- Avoid, if possible, using a large number of sections in a polycone/polyhedra. Break up the volume into a set of sections - only then can each section be optimised using the optimisation ('smart') voxels.
- Avoid complicated boolean solids, composed of a large number of boolean operations (time is proportional to the number of constituents. )
- Create a hierarchy of volumes, if possible, when dealing with thousands of volumes, rather than placing all volumes in one flat space. Especially if there are areas of a setup or detector which have very different typical volume size (eg millimeters near an interaction point, meters far away) there will be a benefit in navigating if the parts of the setup are separated into different volumes (Not an absolute rule.)
- Counterpoint: Generally do not create very complicated envelope volume(s) just to create a hierarchy (e.g. a complicated boolean, polycone, or polyhedra).

# Describing the electromagnetic field as Geant4 expects

When a field object exists for a volume, all charged particles inside will take more CPU time to move - even if the field value is zero. This is because the fact that the field value is zero at one point is not a guarantee that it will remain zero for the full length of a step - Geant4 must evaluate the field value during a step, and will utilize exactly the same code as if the field was finite. To avoid the extra work, you need to describe your field following a few guidelines:

- If a field exists only in a small part of a setup, try to avoid using a global field (use a local field);
- The only way to make sure that a sub-volume has no field, if its parent has a field, is to give it a G4FieldManager whose field pointer is null. In this case there is no field in this volume, and charged particle move in straight lines.
- If a geometrical region is know to have zero (or negligible) field, and its parent not, assign to it an empty G4FieldManager - one with a null field pointer for it.
- If a small region has non-zero field and is inside a large volume which has negligible or zero field, try to create a new 'envelope-like' new volume for the finite field in the geometry. If you can, give the larger volume an empty G4FieldManager

# Selection of Physics Lists

Geant4 physics can be constructed by user from components, however, this requires some expertise. More simple solution is to use one of Reference Physics Lists provided in Geant4 Physics List library. Different approaches and variants of Physics Lists are shown in Geant4 novice, extended and advanced examples. The choice depends on concrete task and reqired accuracy:

- If only electromagnetic physics is needed for an application the best choice would be to use Physics Lists from extended electromagnetic examples ( TestEm3 );
- For hadron therapy and similar applications also hadronic processes should be taken into account. The way how to add hadronic physics is shown in several extended examples ( TestEm7 );
- If main physics processes in an application are hadronic then it is possible to use reference Physics Lists via *G4PhysicsListFactory* class, as it is shown in extended hadronic example ( Hadr00 );

# Selection of options for Geant4 electromagnetic (EM) physics

The main parameter of Geant4 EM simulation is the production cut. A user defines cut in range. The default value in reference Physics Lists is 0.7 mm. At initialisation stage of Geant4 values of production thresholds in energy for electrons and gamma are computed per material used in the simulation. For lower cuts number of simulated secondary particles increases and correspondingly increases CPU time. The choice of the cut value depends on the concrete task and geometry:

- Qualitatively the value of the cut corresponds to accuracy of simulation of energy deposition in space, so cut in range should be about the smallest thickness of absober layers;
- If in the setup there are very different geometry parts which require very different accuracy (for example, micro-detector inside a complecate apparatus), then the setup can be subdivided by different geometry regions and range cuts can be defined per region.

There are other options for EM physics. It is possible to choose different level of accuracy by selecting different constructors of the EM part of Physics Lists. Few guidelines to create an efficient EM physics:

- Try to use if possible constructors provided with Geant4 source code or examples;
- Start validation of a new application using the standard EM physics;
- EM standard models usually provides similar or better accuracy and much better CPU performance for particles with energy above 1 MeV;
- For electrons and gamma below 1 MeV some models of low-energy EM physics are more precise (for example, angular distribution of bremsstrahlung, Doppler broading of Compton scattering);
- For muons, hadrons and ions standard EM physics should be used;
- If X-ray emission or Auger electron production are required then use low-energy Livermore EM physics;
- For simulation of PIXE low-energy Livermore EM physics should be used;
- For more details about accuracy and performance of EM physics, please, visit EM validation page.

-- JohnApostolakis - First version: 04 Dec 2008, Last Edits February 2009

# Tips for Event Biasing usage

Available biasing options in Geant4 are documented in section 3.7 of the User s Guide for Application Developers.

When the focus of an application is on rare events, the **event biasing** technique(s) can provide very significant performance gains, and can even allow estimation of mean values that would be unreachable in practice by standard simulation. Compared to other acceleration techniques like shower parameterization or deterministic field computation, the event biasing technique still involves detailed tracking keeping valid information like track history, propagation time, etc.

The two main techniques used are importance sampling and splitting [1]:

- The **importance sampling** technique enhances the production of some rare event by changing its natural probability of occurrence *p_natural* by a biased probability *p_biasing*. The event here can be for example a given interaction, with *p_natural* being its physical cross-section, or can be a forced interaction in a given volume with *p_natural* being the total interaction probability in this volume. This technique thus changes the underneath physical laws.
- The **splitting** technique defines importance regions (not to be confused with the above importance sampling ) with importance increasing as long as we approach the region of the rare event to target. When a particle goes from one region, with importance *I_small*, to the next one, with higher importance *I_high*, it is split , or actually cloned, into *n_split* identical copies, with *n_split = I_high / I_small* being the ratio of the two importance values. In the other case, where the particle goes to a lower importance region, it is killed with the probability *p_kill = I_small / I_high*. This killing procedure is called the Russian Roulette . This splitting technique keeps unchanged the underneath physical laws.

In both techniques, the particle is given a weight ***w***, also called likelihood ratio in many books, which evolves multiplicatively:

- at each interaction by the ratio *p_natural/p_biasing* in the importance sampling case, meaning in passing that we must care that *p_biasing* > 0 wherever *p_natural* > 0;
- at each change of region by *1/n_split* in case of splitting or by *1/p_kill* in case the particle survives the killing in the splitting technique case.

If secondary particles are created by an interaction of a mother particle of weight ***w***, they are all given the weight ***w***. If this interaction is in addition biased, the secondary particles are given the weight ***w*** multiplied by the likelihood ratio of this interaction.

*This weight **w** is used to correct the contribution of the particle to a given tally (dose, flux, etc.) for the biasing procedure.*

**Tips:**

- Geant4 tracking takes care of the weight evolution in the various biasing options available, and no user s action is needed at this point.
- If you will be using the `G4MultiFunctionalDetector` facility (4.4.5 of the User s Guide for Application Developers) to score the quantities you are interested in through the various available primitive scorers (dose, flux, etc.) the reweighting will automatically be applied.
- If you make your own scoring, either by using your concrete implementation of a `G4UserSteppingAction` based class (section 5.1.4), or by using a sensitive volume object with your concrete implementation of a `G4VSensitiveVolume` (section 4.4.2), you will have to take care of the reweighting. If track is the `G4Track` pointer in your code, you will get the current track weight with `G4double weight = track->GetWeight();`
- When using the splitting options(see section 3.7.1.1 for geometrical biasing) you have to choose the levels of splitting and importance you need. Assuming the case of a shielding (see examples in examples/extended/biasing), with importance regions as simple slices, you have to choose the slices thin enough so that the flux at the exit of the slice is not dramatically reduced compared to what it was at the entrance of the slice. If not doing so, this will result in an inefficient simulation, with many particles simulated and few of them contributing to the signal. In this example, the importance values should be chosen so that the (unweighted) flux, ie total number of particles crossing a surface, stays roughly constant : if the increase of the importance values is too small, the flux will reduce while progressing into the shielding, resulting again in a inefficient simulation : this is an undersplit case; on the other hand, if the increase of the importance values if too fast, this will result in an exponential explosion of the number of particles, and with no benefit on a statistical basis : this is the oversplit case. While the above may look scary, and receives quite theoretical attention (see for example [1] and references there in), an empirical approach leads often to satisfactory performances, and has no deleterious effects on the statistical validity of the results.
- Biasing techniques can be mixed: i.e. it is possible to use in a same application geometrical biasing as above, and, for example, hadronic cross-section biasing (section 3.7.2.1.4).
- Weights contributing to a signal, or a tally, should be kept of the same order of magnitudes [10]. While this can be easily the case for the above simple shielding example, this might be more complicated with more complex setups, or when mixing biasing techniques. If few particles with very high weights compared to other particles score, this will result in instabilities of the scored quantities, with jumps of these scored quantities. Such jumps should be monitored. There are no automatic procedures in Geant4 at this point today. At the opposite, very low weight particles will not contribute significantly to the scored quantity and result just in a waste of time. To control this, the weight window technique can be used (see section 3.7.1.5). This technique uses the same splitting and killing as exposed above, but applies everywhere : particles with high weight will be split so that each copy has a weight into the your pre-defined weight window; particles with low weight will be killed, with the Russian Roulette algorithm as above, so that the survivors are back into the weight window. The same undesplit and oversplit issues, as explained above, exist with this technique.

**References:**

There are many books and articles on the rare event simulation topic. The below one is recent, and gives many references to other documents.


- [1] *Rare Event Simulation using Monte Carlo Methods*, edited by Gerardo Rubino and Bruno Tuffin, John Wiley & Sons, Ltd; and references there in.
- [2] Chapter 10 of reference [1], Particle Transport Application, Thomas Booth; and references there in this chapter.

This topic: Geant4 > Geant4PerformanceTips
Topic revision: r7 - 2010-05-31 - MarcVerderi