REVIEW OF PRODUCTION THRESHOLDS (DRAFT OF PLENARY)

Parallel 4B

Marc Verderi
LLR/Ecole polytechnique
Jefferson Lab Collaboration Meeting
September 2019

Layout

- Introduction
- Cuts & Kernel classes
- Stating on Present Scheme
- Considering an Other Scheme

Introduction

Production Thresholds: initial scheme

- Production thresholds (aka “cuts”) were initially considered in RD44/Geant4 as an issue fundamental enough to be addressed at the kernel classes level:
  1. With a mandatory definition in `G4VUserPhysicsList` :

     ```
     virtual void SetCuts() = 0;
     ```

     - that, up to 2011
     - and then a default implementation came for `SetCuts()`.
  2. An explicit declaration of particles subject to cuts in

     `G4ParticleDefinition`

     - With the predefined and fixed set $\{e^-, e^+, \gamma$ and $p\}$
  3. A control at tracking time by the `G4SteppingManager` of the conformance of the produced secondaries wrt to their thresholds
     - Done after each process `DoIt` invocation
     - But allowing exceptions, though, with the “`GoodForTracking`” flag

Particles Under Production Thresholds

Particle produced	Production process	Motivation
e^-	Ionization	Heavy production (limited by energy binding to atoms). These are actually "recoil electrons". Threshold needed to limit the production.
e^+	Conversion	No divergence nor heavy production. Use case : production cut in mountain rock for, e.g., dark matter experiments.
γ	Bremsstrahlung	Cross-section divergence (actually limited by dielectric effects at very low energies). Threshold needed to limit the production.
p	Hadron elastic	Threshold for recoil protons, e.g. n scattering on proton, ejecting it. Threshold defines the "visibility" cut.
Ion	Hadron elastic	Threshold on recoil, as for protons, defined internally in `G4Hadron-ElasticProcess` as (100*keV)*proton_cut_in_mm

- We use cuts for very different reasons:
  - For $e^-$and γ they are essentially unavoidable, vital
  - For proton and (silently) for ions they are physically very important but not vital
  - For e^+they are for convenience for a quite special use-case
    - And if we accept the use-case for e^+ we have to accept it for all particles !
- Promoting these cuts, on the same foot, at the kernel level, is certainly puzzling.

Questions motivating this review

- Where are we compared to the initial scheme ?
- Isn't this scheme "overkilling" ?
  - Because only a few processes need thresholds
  - And because of the control at tracking after every process `DoIt()`
- Is this scheme effective ?
  - Because of the `GoodForTracking` flag which "offers" to bypass the control anyway
- Why having a "production cut" for e^+ and not for all other particles ?
- Why attaching production cuts to particles while these are essentially a matter of processes ?

- Could we consider a simpler scheme ?
  - Giving full responsibility to the few processes concerned to handle "their" cuts
    - Which does not prevent to have centralized tools to configure the cuts
  - Offloading kernel classes, in particular the `G4SteppingManager`, from this responsibility
  - Leaving open to all processes the opportunity to define cuts (as for e^+) if they wish ?

Cuts & Kernel classes

Cuts in particles category

- `G4ParticleDefinition` allows particles to remember if they are subjects to cuts:
  - Public methods:

```
void   SetApplyCutsFlag(G4bool);
G4bool GetApplyCutsFlag() const;
```

  - Implementation:

```
void G4ParticleDefinition::SetApplyCutsFlag(G4bool flg)
{
  if(theParticleName=="gamma"
  || theParticleName=="e-"
  || theParticleName=="e+"
  || theParticleName=="proton")
  { fApplyCutsFlag = flg; }
  else
  {
    G4cout
     << "G4ParticleDefinition::SetApplyCutsFlag() for " << theParticleName
     << G4endl;
    G4cout
     << "becomes obsolete. Production threshold is applied only for "
     << "gamma, e- ,e+ and proton." << G4endl;
  }
}
```

  - Note also the typedef G4ParticleWithCuts:
    - `typedef G4ParticleDefinition G4ParticleWithCuts;`
    - Used in some places.
- But it looks that `SetApplyCutsFlag(G4bool flg)` is never called !
  - at least for FTFP_BERT, FTFP_BERT_LIV/PEN & QBBC
  - Making `G4bool GetApplyCutsFlag()` always false

Cuts in run category

- Cut flags for `G4ParticleDefinition` objects are set in the physics list base class:

```
void G4VUserPhysicsList::SetApplyCuts(G4bool value, const G4String& name)
{
(…)
  if(name=="all") {
    theParticleTable->FindParticle("gamma")->SetApplyCutsFlag(value);
    theParticleTable->FindParticle("e-")->SetApplyCutsFlag(value);
    theParticleTable->FindParticle("e+")->SetApplyCutsFlag(value);
    theParticleTable->FindParticle("proton")->SetApplyCutsFlag(value);
  } else {
    theParticleTable->FindParticle(name)->SetApplyCutsFlag(value);
  }
}
```

- This is this method which does not look to be called
  - hence leaving all G4ParticleDefinition objects with `GetApplyCutsFlag()` being false.
  - Making void the control of conformance of secondary particles to cuts at tracking time…
  - (Am I missing something ?!?)

- **But be reassured, processes do apply production cuts !**
  - Through the G4ProductionCuts and G4MaterialCutCouple objects

Cuts in tracking category

- The stepping manager DoIt methods:
  - `void G4SteppingManager::InvokeAtRestDoItProcs()`
  - `void G4SteppingManager::InvokeAlongStepDoItProcs()`
  - `void G4SteppingManager::InvokePostStepDoItProcs()`
    - `void G4SteppingManager::InvokePSDIP(size_t np)`
- call for each secondary created by the current process
  - the method `ApplyProductionCut( secondary )`
    - If cuts apply to this secondary particle type
  - Stepping manager snippet code involved:

```
if(tempSecondaryTrack->GetDefinition()->GetApplyCutsFlag())
{ ApplyProductionCut(tempSecondaryTrack); }
```

Each of these methods are 90 – 130 lines long, among which ~50 are for cuts

- `ApplyProductionCut` method:
  - Checks if the secondary conforms to production cuts
  - Two cases:
    - If the track is set “GoodForTracking” by the process, it is accepted anyway
      - Use case: production near boundary
      - Mainly (and likely only) for EM processes
    - Otherwise if its energy is below the cut, it is set to zero kinetic energy, transferring the energy to local deposit
      - And will be later killed if not AtRest processes are attached to it.

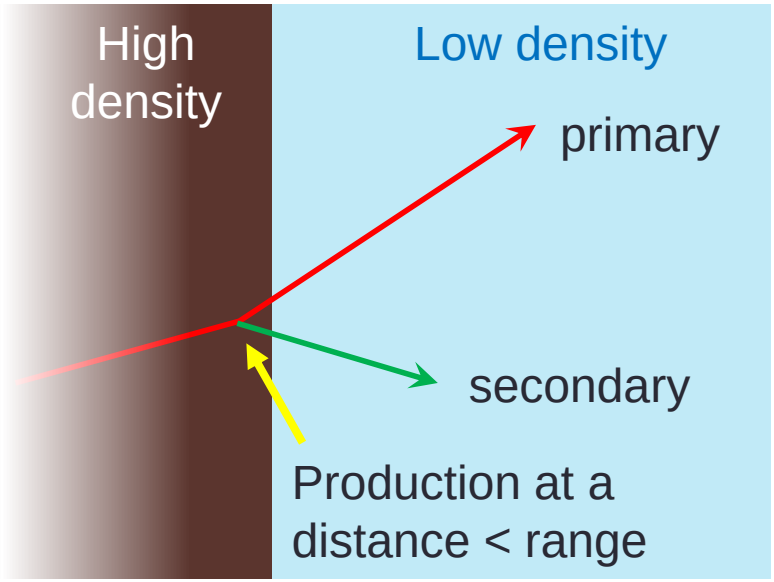
- But, because of the previous issue, `ApplyProductionCut(…)` is not called
  - And the control at tracking time is not effective

Stating on Present Scheme

Stating on present scheme : machinery

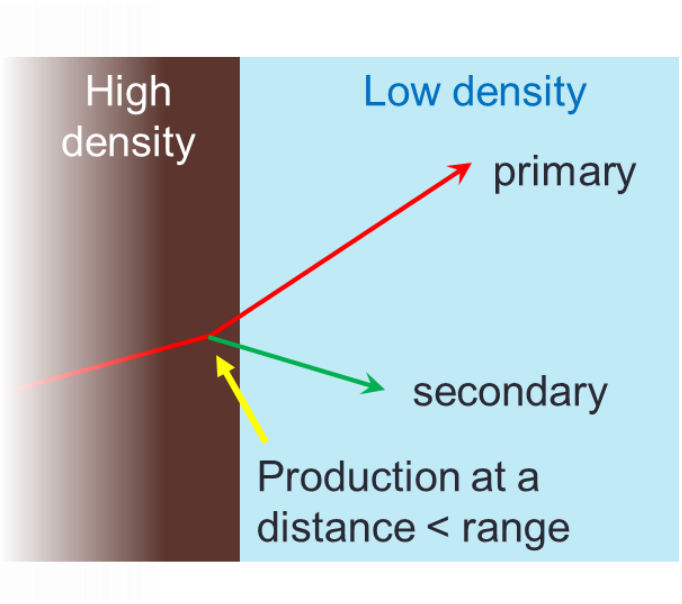
1. Where are we are compared to the initial scheme ?
   - Apparently far…
   - `G4ParticleDefinition` not used anymore to define particle with cuts.
     - Making ineffective the stepping manager conformance check implemented in the `ApplyProductionCut( secondary )` method !
   - In addition, the `GoodForTracking` flag is not used:
     - Was meant to allow production of secondary tracks below threshold (in a “high” density material) that could escape into a low density volume next to it
     - Was not giving much improvement, as reported by Vladimir.

☞ So, in practice, the stepping manager is void wrt threshold conformance checks.



Stating on present scheme

- It looks (unless I'm mistaken…) that:
  - We are far from the initial scheme
  - And that the related code is "dormant" :
    - Making no control of conformance anymore…
    - And no mechanism of exception to it -the `GoodForTracking` flag-.
    - ☞ **All related code might/could be dropped !** (?)
- What about the motivation for the `GoodForTracking` flag ?
  - Was meant to authorize production of secondary tracks below threshold
    - Produced in a high density material…
    - … that could escape in a low density volume
  - But the issue of simulating properly the interface == the issue of simulating properly the lower energy demand
    - The tracking can't judge by itself !
    - And so just follows the process advice
  - So what is the point of the secondary production control by the `G4SteppingManager` ?
- Even more: if a process is unduly producing lots of secondaries, should this be shadowed by a control or made obvious ?



Considering an Other Scheme

Proposal

- Kernel classes are offloaded from cuts control
  - Including control at tracking time
  - Classes involved: `G4ParticleDefinition, G4SteppingManager`
- Processes are given the full responsibility to manage their production thresholds
  - Whatever if this is due to divergences or not
  - Common tools are used to expose the cuts configuration to the user and allow her/him to set it up
- The machinery for material-cut couple becomes extendable:
  - It has the set $\{e^-, \gamma$ and $p\}$ by default
  - But is extendable to any other type of particle
- Dedicated tests are added to check for conformance of secondary production
    - A test using a simple user stepping action could do it
- Backward compatibility should be considered as well
  - At least for some time