Physics II: Physics Overview, Processes and Production Thresholds

20-23 June, 2016
Geant4 tutorial @ MIT
Whitaker Building – Room 56-154

http://www.geant4.org
Outline

• Overview of Geant4 Physics

• Processes

• Production Thresholds (aka cuts)

• Cuts per Region
Geant4 Physics

• Geant4 provides a wide variety of physics components for use in simulation

• Physics components are coded as processes
  – a process is a class which tells a particle how to interact
  – Geant4 provides many of these
  – users may write their own, but must be derived from a Geant4 process

• Processes are classified as
  – electromagnetic, hadronic, decay, parameterized or transportation
Geant4 Physics: Electromagnetic

• standard – complete set of processes covering charged particles and gammas
  — energy range 1 keV to ~PeV

• low energy – specialized routines for e-, γ, charged hadrons
  — more atomic shell structure details
  — some processes valid down to 250 eV or below
  — others not valid above a few GeV

• Optical photon – only for long wavelength photons (x-rays, UV, visible)
  — processes for reflection/refraction, absorption, wavelength shifting, Rayleigh scattering
Geant4 Physics: Hadronic

• pure hadronic (0 - ~TeV)
  – elastic
  – inelastic
  – capture
  – fission

• radioactive decay
  – at rest and in-flight

• photo-nuclear (~10 MeV – ~TeV)
• lepto-nuclear (~10 MeV - ~TeV)
  – e+, e- induced nuclear reactions
  – muon induced nuclear reactions
• decay processes include
  – weak decay (leptonic, semi-leptonic decays, radioactive decay of nuclei)
  – electromagnetic decay ($\pi^0$, $\Sigma^0$, etc.)
  – strong decays not included here (they are part of hadronic models)

• parameterized process
  – electromagnetic showers propagated according to parameters averaged
    over many events
  – faster than detailed shower simulation

• transportation
  – only one process which is responsible for moving the particle through the
    geometry
Physics Processes

• All the work of particle decays and interactions is done by processes

• A process does two things:
  – decides when and where an interaction will occur
    – method: GetPhysicalInteractionLength()
    – this requires a cross section or decay lifetime
    – for the transportation process, the distance to the nearest object along the track is required

  – generates the final state of the interaction (changes momentum, generates secondaries, etc.)
    – method: DoIt()
    – this requires a model of the physics
Physics Processes

• There are three flavors of processes:
  • well-located in space -> PostStep
  • distributed in space -> AlongStep
  • well-located in time -> AtRest

• A process may be a combination of all three of the above
  – in that case six methods must be implemented, one
    GetPhysicalInteractionLength() and one DoIt() for each type

“Shortcut” processes are defined which invoke only one
  – Discrete process (has only PostStep physics)
  – Continuous process (has only AlongStep physics)
  – AtRest process (has only AtRest physics)
Example Processes

• Discrete process: **Compton scattering**
  • length of step determined by cross section, interaction is at end of step
    - PostStepGetPhysicalInteractionLength()
    - PostStepDoIt()

• Continuous process: **Cerenkov effect**
  — photons created along step, # proportional to step length
    - AlongStepGetPhysicalInteractionLength()
    - AlongStepDoIt()

At rest process: **positron annihilation at rest**
  — no spatial displacement, time is the relevant variable
    - AtRestGetPhysicalInteractionLength()
    - AtRestDoIt()
Example Processes

• Continuous + discrete: ionization
  • energy loss is continuous
  • Moller/Bhabha scattering and knock-on electrons are discrete

• Continuous + discrete: process: bremsstrahlung
  — energy loss due to soft photons is continuous
  — hard photon emission is discrete

In both cases, the production threshold separates the continuous and discrete parts of the process
  — more on this later

Multiple scattering is also continuous + discrete
Handling Multiple Processes

• Many processes (and therefore many interactions) may be assigned to the same particle
Handling Multiple Processes

• How does Geant4 decide which interaction happens at any one time?

  • interaction length (or decay length) is sampled from each process

  • sampling is done from the distribution $1 - e^{-\alpha p L}$

  • done for each process assigned to the particle

• now we have several lengths, including distance to next volume boundary

• the interaction (or boundary crossing) that happens is the one with the shortest length

• processes which did not occur are not re-sampled, but have the previous step length subtracted from their originally sampled lengths
Handling Multiple Processes

- **Step 1:**
  - all lengths sampled
  - Compton occurs

- **Step 2:**
  - Compton re-sampled
  - boundary is crossed

- **Step 3:**
  - Compton occurs again
  - new boundary found

- **Step 4:**
  - Compton re-sampled
  - pair production occurs
• Every simulation developer must answer the question: how low can you go?
  — at what energy do I stop tracking particles?

• This is a balancing act:
  — need to go low enough to get the physics you’re interested in
  — can’t go too low because some processes have infrared divergence causing CPU to skyrocket

• The traditional Monte Carlo solution is to impose an absolute cutoff in energy
  — particles are stopped when this energy is reached
  — remaining energy is dumped at that point
Threshold for Secondary Production

• But, such a cut may cause imprecise stopping location and deposition of energy

• There is also a particle dependence
  — range of a 10 keV $\gamma$ in Si is a few cm
  — range of a 10 keV e- in Si is a few microns

• And a material dependence
  — suppose you have a detector made of alternating sheets of Pb and plastic scintillator
  — if the cutoff is OK for Pb it will likely be wrong for the scintillator which does the actual energy measurement
Threshold for Secondary Production

- **Geant4 solution**: impose a production threshold
  - this threshold is a distance, not an energy
  - default = 0.7 mm
  - the primary particle loses energy by producing secondary electrons or gammas
- if primary no longer has enough energy to produce secondaries which travel at least 0.7 mm, two things happen:
  - discrete energy loss ceases (no more secondaries produced)
  - the primary is tracked down to zero energy using continuous energy loss

- Stopping location is therefore correct
- Only one value of production threshold distance is needed for all materials because it corresponds to different energies depending on material
Production Threshold vs. Energy Cut
Example: 500 MeV p in LAr-Pb Sampling Calorimeter

Geant3 (and others)

Cut = 2 MeV

Cut = 450 keV

Geant4

0.5 mm
Threshold for Secondary Production

• Geant4 recommends the default value of 0.7 mm
  – user needs to decide the best value
  – this will depend on the size and sensitive elements within the simulated detector, and on available CPU

• This value is set in the SetCuts() method of your physics list

• Instead of “secondary production threshold distance” it is more convenient to simply say “cuts”
  – but please remember that this does not mean that any particle is actually stopped before it runs out of energy
Cuts per Region

• In a complex detector there may be many different types of sub-detectors involving
  – finely segmented volumes
  – very sensitive materials
  – large, undivided volumes
  – inert materials

• The same value of the secondary production threshold may not be appropriate for all of these
  – user must define regions of similar sensitivity and granularity and assign a different set of production thresholds (cuts) for each

• Warning: this feature is for users who are
  • simulating the most complex detectors
  • experienced at simulating EM showers in matter
Cuts per Region

• A default region is created automatically for the world volume
  — it has the cuts which you set in SetCuts() in your physics list
  — these will be used everywhere except for user-defined regions

In the geometry an instance of G4Region must be created which corresponds to the volume where the cuts are to be changed

• To define different cuts for this special region, user must
  — create a G4ProductionCuts object
  — initialize it with the new cuts
  — assign it to a region which has already been created
void MyPhysicsList::SetCuts() {
    SetCutValue(defaultCutValue, "gamma"); // same for e-, e+, p

    // Get the region
    G4Region* aRegion = 
        G4RegionStore::GetInstance()->GetRegion("RegionA");

    // Define cuts object for the new region and set values
    G4ProductionCuts* cuts = new G4ProductionCuts();
    cuts->SetProductionCut(0.01*mm); // here, same for all
    // Assign cuts to region
    aRegion->SetProductionCuts(cuts);
}
Summary

• Processes handle all the physics of particle interactions
• Geant4 provides processes to cover nearly all particles over energies ranging from 0 to ~TeV
  • users may define their own processes
• Many processes may be assigned to a particle

• The precision of particle stopping and the production of secondary particles are determined by a secondary production threshold
• For complex detectors with different types of sensitive volumes, different production thresholds may be defined for different regions within the detector
This is the main function used in G4 to sample the cross section. Consider a process with a cross section \( \sigma \), the distance \( x \) a particle travels without interacting is distributed as:

\[
PDF(x) = \sigma e^{-\sigma x}
\]

The mean free path being \( 1/\sigma \)

So, for a given particle and a process, if we want to know when this process will limit the step length (e.g. because is a discrete step length) we have to generate a random number according to this PDF.

As it is well known a way to obtain random numbers distributed according to a logarithmic PDF is to generate a random variable \( x' \) distributed uniformly in \([0,1]\) and calculate \( x = -\sigma \log(x') \).

This is very useful also because it factors out the dependency on \( \sigma \) (that usually depends on the energy of the particle and so on).

In addition remember that the PDF we consider has the important properties of not having "memory": if a particle survived up to \( x_1 \), the probability to interact in (i.e. survive up to) \( x_2 \) is distributed according to the same PDF with \( x = x_2 - x_1 \).

So you have two choices for a tracking algorithm in a MC: perform one step, throw a random number according to the PDF to calculate when the particle will do the next interaction OR when starting a new track, throw a single random number (the number of interactions length left), at each step you reduce this number by \( step_{length}/\sigma \) and when this reaches zero, if the particle survived, you throw another random number.

Since throwing random numbers is expensive (maybe today we can do differently on GPUs), the second approach was chosen.

Let's demonstrate that indeed the two approaches are equivalent, i.e. give raise to the same PDF.