Geant 4

EM physics: The standard package

Anton Lechner, CERN

Acknowledgements:
Slides created by *M. Maire, V. Ivanchenko*

http://cern.ch/geant4

Standard Electromagnetic Packages

Overview of the packages provided by the Standard electromagnetic (EM) working group, and important physics covered by these packages

Packages Overview

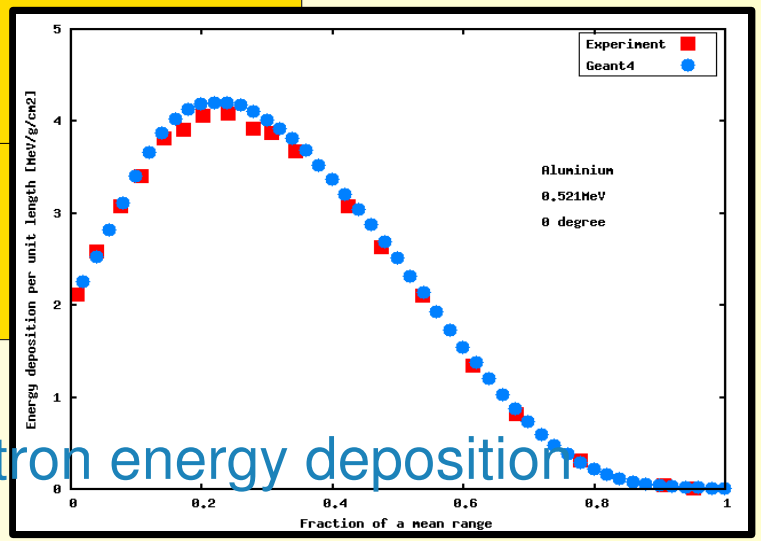
- Packages of the Standard EM working group:

Package	Description
Standard	Gamma, Electrons up to 100 TeV, Hadrons, Ions up to 100 TeV
Muons	Muons up to 1PeV, Energy loss propagator
X-rays	X-ray and optical photon production processes
Optical	Optical photon interactions
High-energy	Processes at high energy (E > 10 GeV), Physics for exotic particles
Polarization	Simulation of polarized beams

Transport of Gammas, Electrons and Positrons

- EM processes for Gammas, Electrons and Positrons:

Particle	Processes
Photons	Gamma conversion into e^+e^- pair Compton scattering Photoelectric effect
Electrons, Positrons	Ionization Coulomb scattering Bremsstrahlung
Positrons	Annihilation



Example: Electron energy deposition

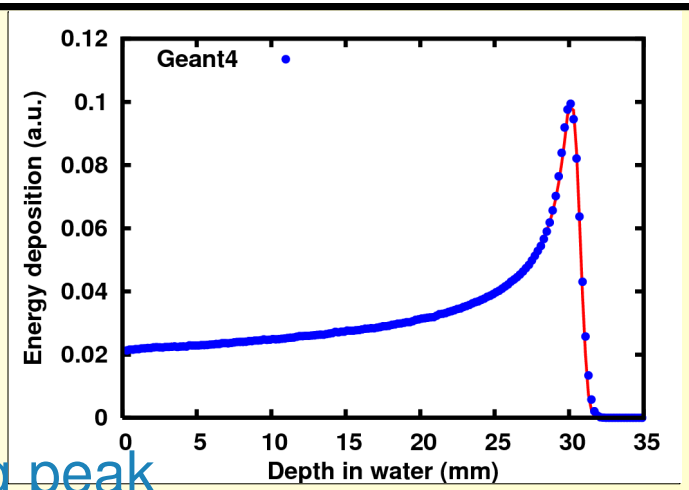
EM Processes of Hadrons and Ions

- EM processes for Hadrons and Ions:

Particle	Processes
Hadrons, Ions	Coulomb Scattering, Ionization

$$\frac{dE}{dx} = 2\pi r_e^2 mc^2 n_{el} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2mc^2\beta^2\gamma^2 T_{up}}{I^2} \right) - \beta^2 \left(1 + \frac{T_{up}}{T_{max}} \right) - \delta - \frac{2C_e}{Z} + F \right]$$

- Ionization:
  - E > 2 MeV: Bethe-Bloch formula with corrections
  - E < 2 MeV: Parametrization (ICRU 49, NIST)

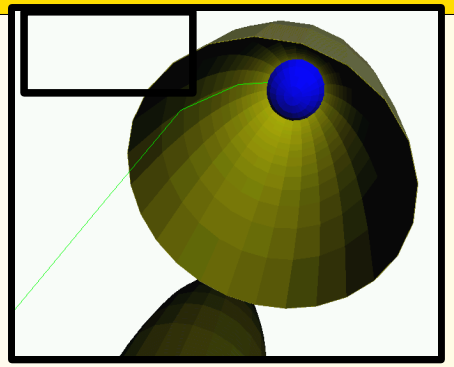


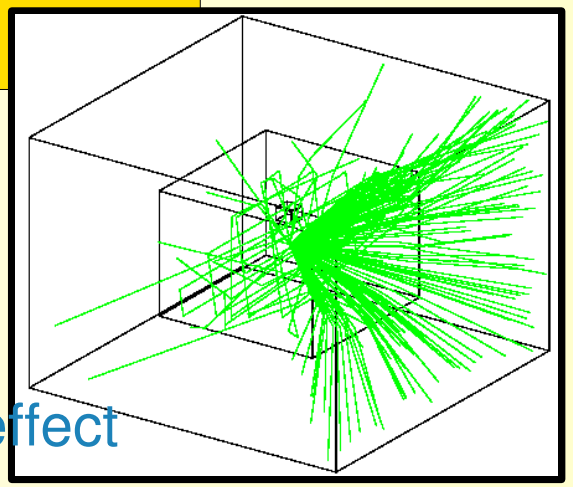
Ex: Proton Bragg peak

Simulation of x-rays and optical photons

- Optical photons are created by the Cerenkov effect, or result from transitions, or scintillation:
  - Creator processes contained in standard, xray package
- Processes of optical photons in the Optical package:

Particle	Processes
Optical photons	Refraction and reflection at medium boundaries, absorption, Rayleigh scattering





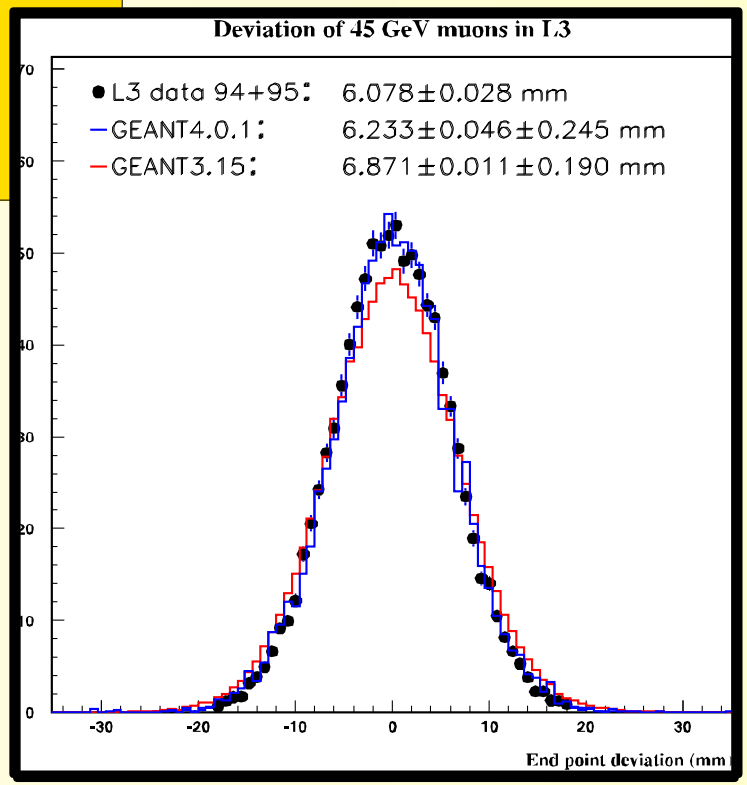
Cerenkov effect

EM Processes of muons

- EM processes for Muons:

Particle	Processes
Muons	Ionization, Bremsstrahlung, e^+e^- pair production

Figure: 45 GeV muons



Short introduction to modular physics lists

Creating a modularized physics setup

A modular physics list (1/2)

- A physics list is a mandatory part of a Geant4 application
  - Physics lists inherit from G4VUserPhysicsList
    - You can derive your own concrete class from this base class to set up the physics relevant to your simulation application
    - Implement the pure virtual methods
      - ConstructParticle() (instantiate particles)
      - ConstructProcess() (instantiate physics processes and register them to the process manager)
      - SetCuts() (particle cuts)
  - Geant4 provides also an abstract base, GVModularPhysicsList, which derives from G4VUserPhysicsList, and which allows to modularize your physics setup

A modular physics list (2/2)

- Basic strategy to set up a modular physics list:
  - G4VModularPhysicsList has a method called RegisterPhysics which you can use to register physics constructors
  - A physics constructor is a class deriving from G4VPhysicsConstructor
    - The main purpose of a physics constructor is to instantiate physics processes and particles relevant to your application
    - You may create several physics constructors, each covering a particular group of process:
      - E.g. A constructor for setting up the EM physics of electrons, another constructor, which instantiates all hadronic processes of ions, ...
      - In this way, you can modularize your physics setup
      - You can also create alternative constructors for the same group of processes (using alternative physics simulation model)

Example:

```
// main():

#include “G4RunManager.hh“
#include “MyDetectorConstruction.hh“
#include “MyPhysicsList.hh“
#include ”MPrimaryGenerator.hh“

main() {

  G4RunManager* runManager = new G4RunManager();

  // mandatory initialization classes:
  G4VUserDetectorConstruction* detector =  new MyDetectroConstruction();
  runManager -> SetUserInitialization(detector);

  G4VUserPhysicsList* physicsList = new MyPhysicsList();
  runManager -> SetUserInitialization(physicsList);

  // mandatory user action class:
  G4VUserPrimaryGeneratorAction* primaryGenerator =
                                          new MyPrimaryGenerator();
  runManager -> SetUserAction(primaryGenerator);
  ...
}
```

Example:

```
// physics list header file:

#include "G4VModularPhysicsList.hh"

class MyPhysicsList : public G4VModularPhysicsList {

 public:
   MyPhysicsList();
   virtual ~MyPhysicsList();
 ...
   void SetCuts();
};
```

```
  // physics list source file:
....
MyPhysicsList::MyPhysicsList() :
    G4VModularPhysicsList() {
  ....

  // Create and instantiate particles here

  // Registering EM physics for electrons:
  // (See next chapter for how to create this physics constructor class)
  RegisterPhysics(new PhysicsEMElectronStandard());
 ... // Register other physics constructors
}

// Implement also the SetCuts() method to set production cuts
....
```

Physics constructors with standard EM processes

Implementing EM physics constructors in your application – a few examples

Physics constructors

- Derive your concrete implementation of a physics constructor by inheriting from G4VPhysicsConstrutor
  - Instantiate physics processes in the ConstructProcess() function and register them to the process manager
- The process manager maintains a process list for each particle type
  - For a given particle, a process can be registered to three different kind of actions associated with a particle step:
    - AtRest action
    - AlongStep action
    - PostStep action

Example: Electrons

```
// header file:
#include "G4VPhysicsConstructor.hh"

class PhysicsEMElectronStandard : public G4VPhysicsConstructor {
 ...
  virtual void ConstructProcess();
};
```

```
// source file:
#include  "G4ProcessManager.hh"
#include "G4ParticleDefinition.hh"
#include "G4MultipleScatterin.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
...
void PhysicsEMElectronStandard::ConstructProcess() {

 G4ParticleDefinition* electron = G4Electron::Definition();
 G4ProcessManager* manager = electron -> GetProcessManager();


// Adding multiple Coloumb scattering
 manager -> AddProcess(new G4MultipleScattering(), -1, 1, 1);

// Adding  ionisation
 manager -> AddProcess(new G4eIonisation(), -1, 2, 2);

 // Adding bremsstrahlung
 manager -> AddProcess(new G4eBremsstrahlung(), -1, -1, 3);
}
```

-1 means that process is not registered for this action

order of post step actions

order of along step actions

order of at rest actions

process has only discrete action

Examples: Positrons, Photons

```
// source file positron: include header file for each process
...
void PhysicsEMPositronStandard::ConstructProcess() {

 G4ParticleDefinition* positron = G4Positron::Definition();
 G4ProcessManager* manager = positron -> GetProcessManager();

  manager -> AddProcess(new G4MultipleScattering(), -1, 1, 1);
  manager -> AddProcess(new G4eIonisation(), -1, 2, 2);
  manager -> AddProcess(new G4eBremsstrahlung(), -1, -1, 3);
  manager -> AddProcess(new G4eplusAnnihilation(), 0, -1, 4);
}
```

```
// source file photon: include header file for each process
...
void PhysicsEMPhotonStandard::ConstructProcess() {

 G4ParticleDefinition* gamma = G4Gamma::Definition();
 G4ProcessManager* manager = gamma -> GetProcessManager();

  manager -> AddDiscreteProcess(new G4PhotoElectricEffect());
  manager -> AddDiscreteProcess(new G4ComptonScattering());
  manager -> AddDiscreteProcess(new G4GammaConversion());

}
```

Order not important if processes have only post-step actions

is the same

```
manager -> AddProcess(new G4PhotoElectricEffect(), -1, -1, 1);
manager -> AddProcess(new G4ComptonScattering(), -1, -1, 2);
manager -> AddProcess(new G4GammaConversion(), -1, -1, 3);
```

Examples: Muons, charged Hadrons

```
// source file muons: include header file for each process
...
void PhysicsEMMuonStandard::ConstructProcess() {

 G4ParticleDefinition* muonPlus = G4MuonPlus::Definition();
 G4ProcessManager* manager = muonPlus -> GetProcessManager();

  manager -> AddProcess(new G4MultipleScattering(), -1, 1, 1);
  manager -> AddProcess(new G4MuIonisation(), -1, 2, 2);
  manager -> AddProcess(new G4MuBremsstrahlung(), -1, 3, 3);
  manager -> AddProcess(new G4MuPairProduction, -1, 4, 4);
  // similar for muonMinus
}
```

Geant4 Appl. Dev.. Manual

```
// source file charged hadrons: include header file for each process
...
void PhysicsEMHadronStandard::ConstructProcess() {

  // obtain process manager fromconsidered charged hadrons and register the
  // process for each particle type (you may loop over all particles and select
  // the desired ones by using conditional statements):

  manager -> AddProcess(new G4MultipleScattering(), -1, 1, 1);
  manager -> AddProcess(new G4hIonisation(), -1, 2, 2);
}
```

Sampling mechanism for important processes

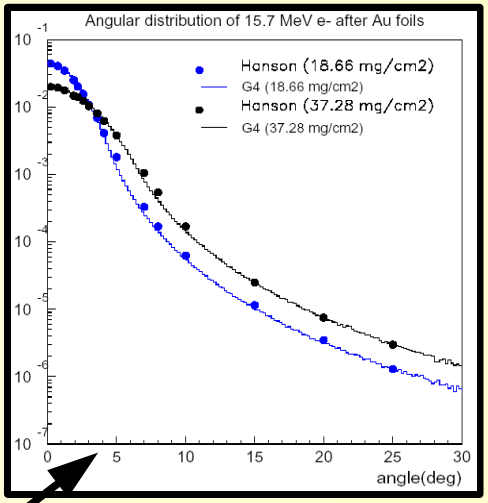
Ionization, multiple scattering

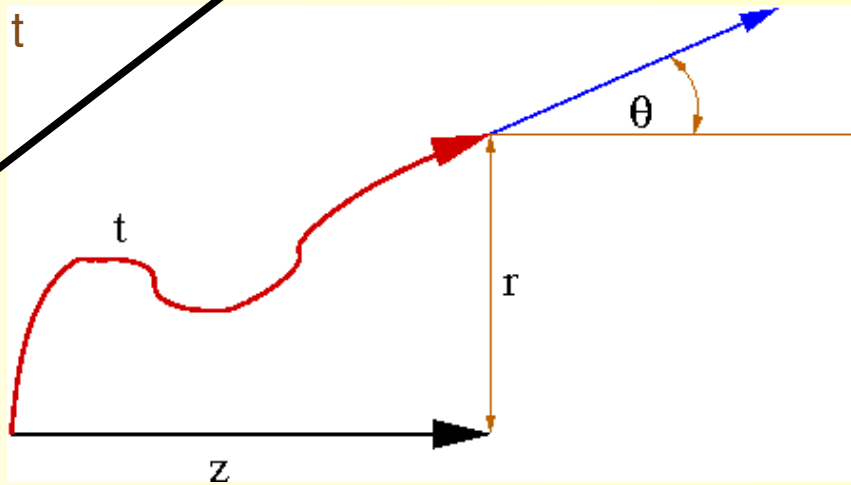
Multiple Coloumb Scattering

- Geant4 is a condensed Monte Carlo code
  - Global effects due to collisions along a macroscopic step are simulated, using approximations
  - Is in contrast to detailed simulations at a microscopic level, which are exact, but require an enormous amount of time for higher energies
- Cumulative effects
  - Charged particles, that travel through a finite material layer are subject to a large number of Coloumb interactions, where the particles are elastically scattered

Multiple Coloumb Scattering

- Cumulative effects (cont.)
  - Summing up individual effects gives a net deflection of the particles from their initial direction
  - Related quantities are:
    - longitudinal displacement z
    - lateral displacement r, omega
    - true (or corrected) path length t
    - angular deflection theta, phi





For sufficient collisions (>20) the multiple scattering angular distribution is
* a Gaussian distribution for small angles
* and is like Rutherford scattering at large angles

Geant4 Multiple Scattering Model

- The Geant4 Multiple Scattering (MSC) model by L. Urban is applicable to all charged particles
- It is based on the Lewis theory
  - Relies on transport equation of charged particles
- Uses phenomenological functions to sample angular and spatial distributions after the simulation step
  - The function parameters are chosen, in order that the moments of the distribution are the same as given by the Lewis theory
  - Set the *Physics Reference Manual* for details

Ionization

- The basics mechanism is an inelastic collision of the moving charged particle with the atomic electrons of the material, ejecting off an electron from the atom:

$$\bullet\ \mu + \text{atom} \longrightarrow \mu + \text{atom}^{+} + e^{-}$$

- In each individual collision, the energy transferred to the electron is large, and we can well define the average energy loss per (macroscopic) unit path length

Ionization

- Mean energy loss and energetic δ-rays
  - Differential cross-section per atom for the ejection of an electron with kinetic energy T by an incident charged particle of total energy E (moving in a material of density ρ):

$$\frac{d\sigma(Z, E, T)}{dT}$$

- High-energy knock-on electrons
  - Above a given threshold T_{cut} high-energy knock-on electrons are explicitly taken into account (where $T_{cut} \gg$ mean excitation energy in material), i.e. they are explicitly generated (particles excluded from continuous energy loss)
  - Below T_{cut} the soft knock-on electrons are only considered as continuous energy loss of the incident particle

Standard EM in Geant4:
$T_{cut} > 1$ keV

Ionization

- Mean rate of energy loss for soft δ-rays
  - Mean rate of the energy lost by the incident particle due to soft δ-rays is thus:

$$\frac{dE_{soft}(E, T_{cut})}{dx} = n_{at} \cdot \int_0^{T_{cut}} \frac{d\sigma(Z, E, T)}{dT} T \, dT$$

number of atoms per volume

- Ejection of high-energy electrons
  - The total cross section per atom for the ejection of an electron with an energy above T_{cut} is (T_{max} is maximum energy transferable to free e^-):

$$\sigma(Z, E, T_{cut}) = \int_{T_{cut}}^{T_{max}} \frac{d\sigma(Z, E, T)}{dT} \, dT$$

Ionization

- Fluctuations in the energy loss:
  - $\langle \Delta E \rangle = (dE/dx)\Delta x$ gives only the average energy loss by ionisation
  - Fluctuations must also be considered: Depending on the material along Δx, the distribution of ΔE can be strongly asymmetric (Landau tail)
  - Large fluctuations are due to a small number of collisions with large energy transfers.
  - The fluctuations of ΔE result in fluctuations of the actual range (straggling)

Ionization

- Geant4 models of energy loss fluctuations:
  - Urban model of fluctuations, which is based on a simple modelling approach of particle-atom interactions:
    - Atoms are assumed to have only two energy levels: E_1 and E_2
    - The particle-atom interaction can be:
      - excitation of the atom with energy loss $E = E_1 - E_2$
      - ionization with energy loss distribution $g(E) \sim 1/E^2$
  - PAI model uses photo absorption data
    - All energy transfers are sampled with production of secondary electrons and gammas
    - Very slow model, should only be applied for sensitive region

Ionization

- Sampling of δ-electrons from hadrons and ions:
  - The differential cross-section per atom for producing an electron of kinetic energy T (where $I \ll T_{cut} \leq T \leq T_{max}$) is given by :

$$\frac{d\sigma}{dT} = 2\pi r_e^2 mc^2 Z \frac{z_p^2}{\beta^2} \frac{1}{T^2} \left[1 - \beta^2 \frac{T}{T_{max}} + \frac{T^2}{2E^2} \right]$$

  - Integration gives (the last term for spin ½ only):

$$\sigma(Z, E, T_{cut}) = \frac{2\pi r_e^2 Z z_p^2}{\beta^2} mc^2 \times \left[\left(\frac{1}{T_{cut}} - \frac{1}{T_{max}} \right) - \frac{\beta^2}{T_{max}} \ln \frac{T_{max}}{T_{cut}} + \frac{T_{max} - T_{cut}}{2E^2} \right]$$

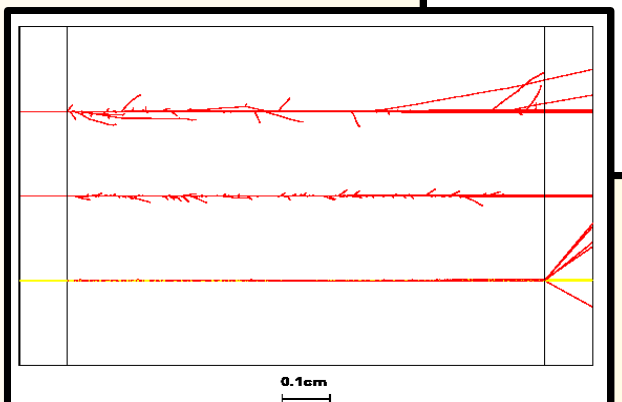


Fig.: 2000 MeV electron, proton and α in Al

Energy-range relation and Stepping

Energy-Range relation

- The mean total path length of a charge particle of kinetic Energy T is given by:

$$R(T) = \int_0^T \frac{1}{(dE/dx)}\, dE.$$

- This relation is extensively used in Geant4:
  - to control the stepping of charged particles
  - to compute the energy loss of charged particle
  - to control the production of secondaries (cut in range)

Cut in Range

- Charged particles are tracked until the end of their range
  - However, the differential cross sections for δ-electron (see before) and bremsstrahlung production increase significantly when the particle energy decreases
  - If all secondaries would be produced at low energies computational requirements would become unrealistic
  - The solution is to introduce lower production thresholds

only for sec. e-, e+, γ

- Conversion from range cut to kinetic energy
  - For reasons of coherence, the user must specify the lower production threshold as range, which is converted to an energy value

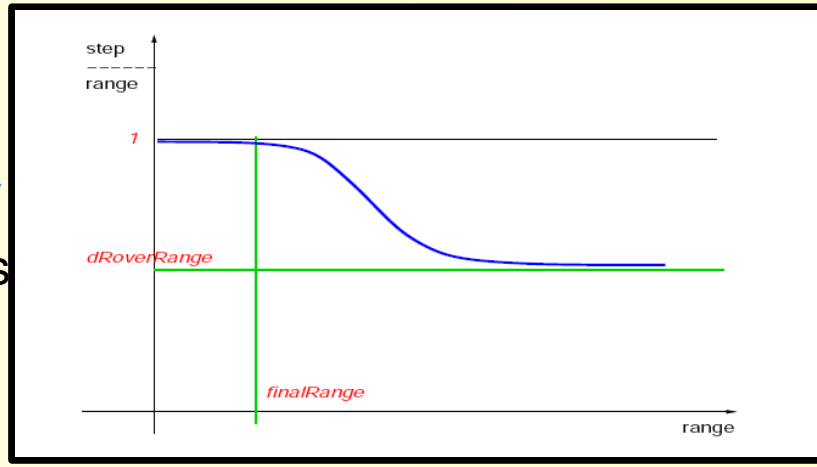
Do not confuse this range cut, with a tracking cut:
There is no tracking cut in Geant4

Stepping of Charged Particle

- Stepping control
  - The continuous energy loss imposes a limit on the step size
  - The cross sections depend on the energy E (s indicates the position of the particle, Z_i is the atomic number of the material):

$$\sigma(Z_i, E(s))$$

  - The step size must be small enough, in order that the change in the kinetic energy along the step is *small* compared to the particle energy
  - When the kinetic energy approaches zero, this constraint must be relaxed: The allowed step smoothly approaches the stopping range of the particle



Mean Energy Loss

- Computing mean energy loss of charge particles
  - The computation fo the mean energy loss on a given step is done from the range and inverse range tables
  - If the step is long, this is more accurate than $\Delta E = (dE/dx)\Delta x$ (where Δx is the step length)
  - dE/dx, range and inverse range tables are computed in the initialization phase of a Geant4 simulation using production thresholds