Hadronic Physics II

Alberto Ribon
CERN EP/SFT

Geant4 Virtual Tutorial, 28 September – 2 October 2020

Outline

- Reminder:
  - Hadronic Models, Cross Sections, Framework, Physics Lists
- NeutronHP (ParticleHP)
- Radioactive Decay
- Biasing in Hadronic Physics
  - First part only: the “old” way

Reminder:
Hadronic Models , Cross Sections , Framework , Physics Lists

Hadronic interactions

- Hadrons (π± , K± , K°L , p , n , α , *etc.*), produced in jets and decays, travel through the detector (H , C , Ar , Si , Al , Fe , Cu , W , Pb ...)
- Therefore we need to model hadronic interactions

  hadron – nucleus -> *anything*
- In principle, QCD is the theory that describes all hadronic interactions in practice, perturbative calculations are applicable only in a tiny (but important !) phase-space region

  the hard scattering at high transverse momentum

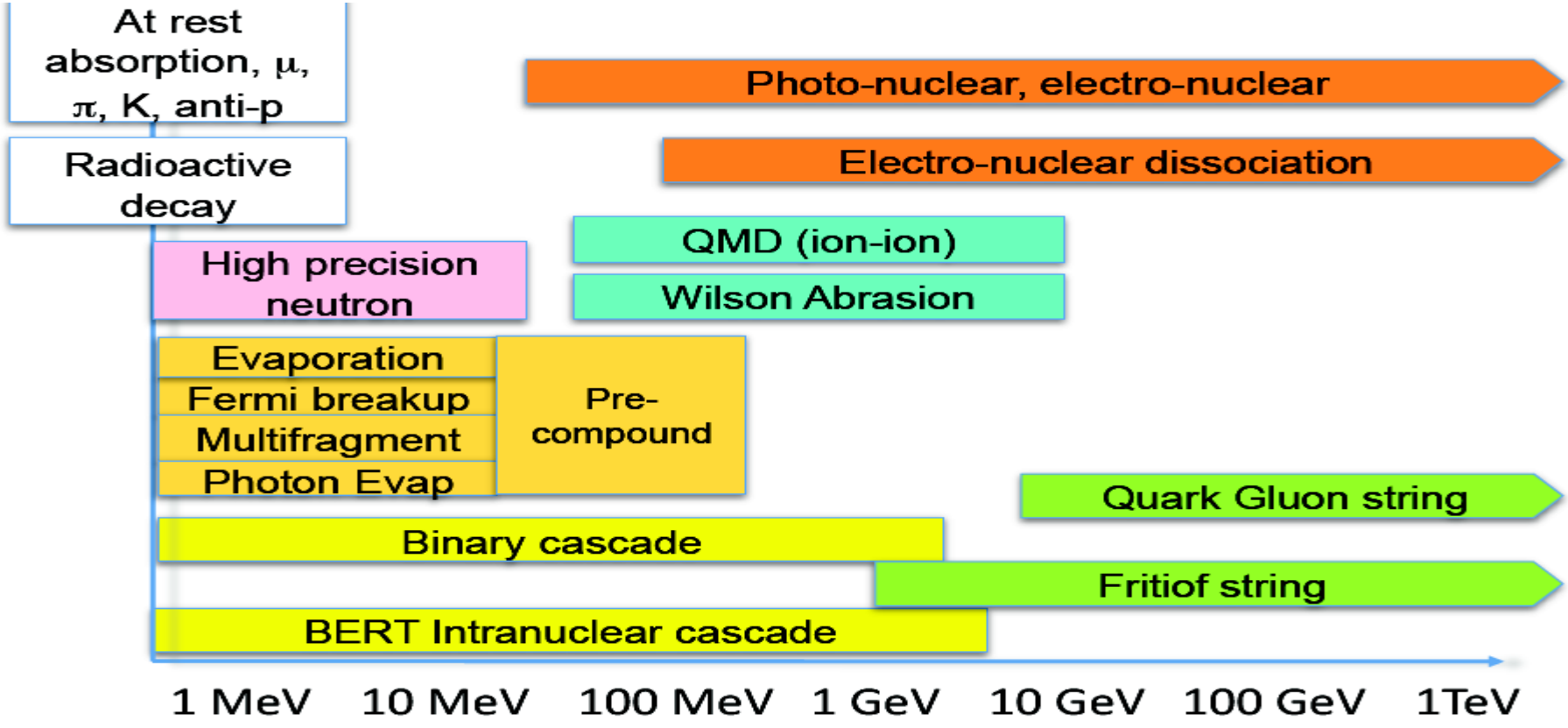
  whereas for the rest, *i.e.* most of the phase space

  soft scattering, re-scattering, hadronization, nucleus de-excitation

  only approximate models are available
- Hadronic models are valid for limited combinations of

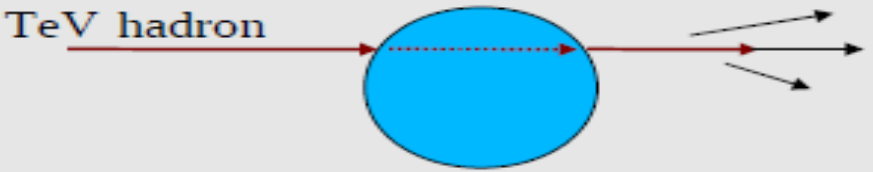
  particle type − energy − target material

Partial Hadronic Model Inventory



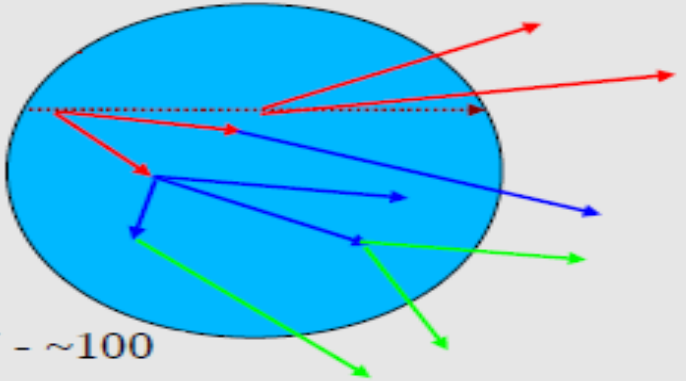
Hadronic Interactions from TeV to meV

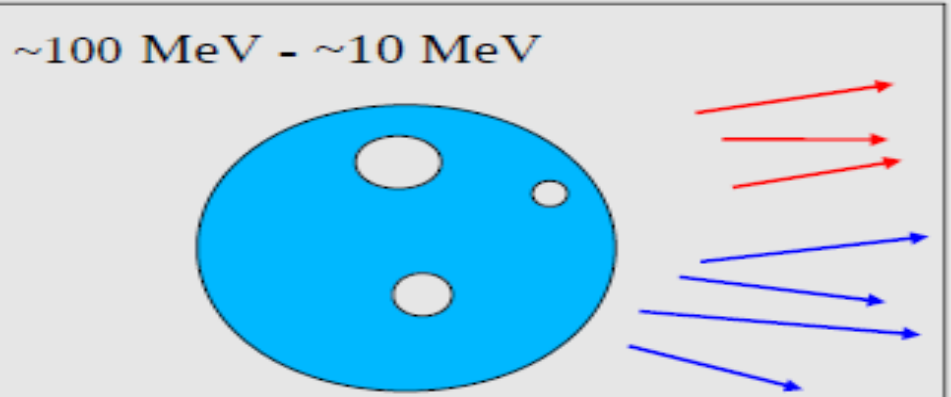
String model



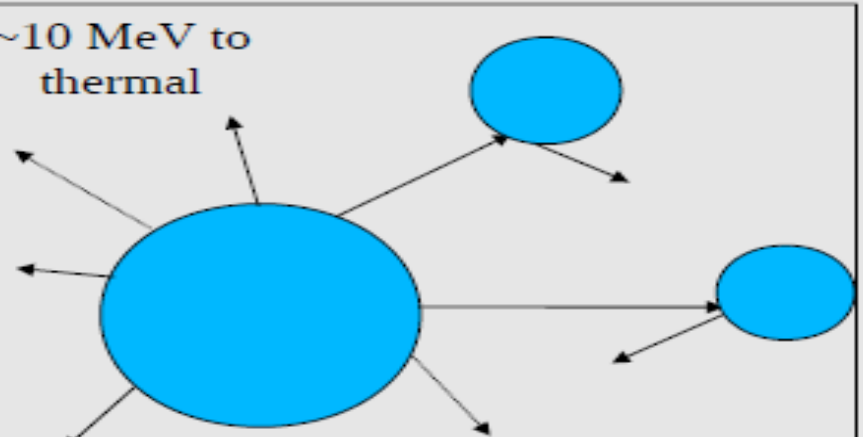
$dE/dx \sim A^{1/3}\,GeV$

Intra-nuclear cascade model





Pre-equilibrium (Precompound) model

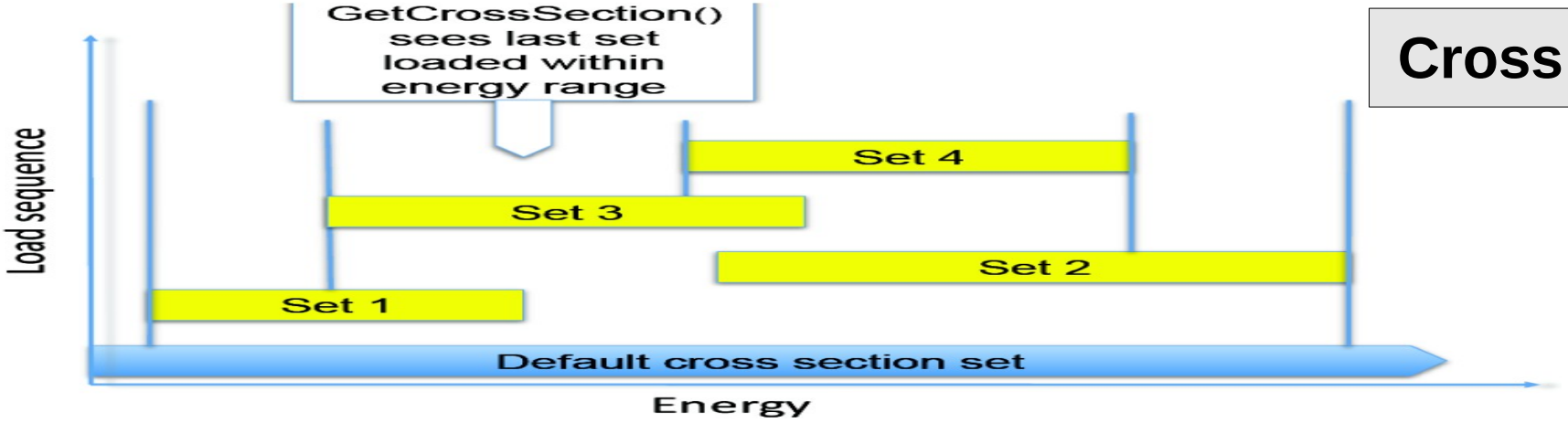


Equilibrium (Evaporation) model

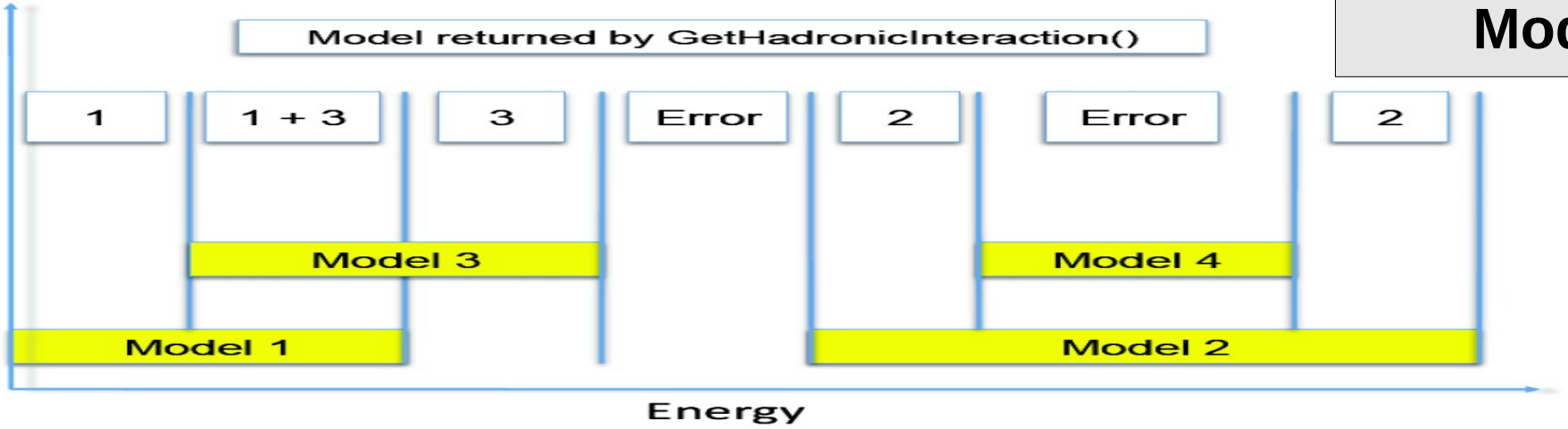
Hadronic Cross Sections and (Final-State) Models

- In Geant4, there is a clear separation between **cross sections** – related to the probability of an elastic or inelastic hadron-nucleus interaction, and therefore to the length that a hadron projectile flies in a material before interacting – and **final-state models** – related to the number, type and properties of the produced secondaries
- For each combination of projectile – energy – target
  - ≥ 1 cross sections must be specified in a physics list : the first available is used
  - 1 or 2 (final-state) models must be specified in a physics list : if two, a random number is thrown to decide which of the two models to use
    - linear probability as a function of the energy, over an interval called **transition region**, defined arbitrarily to get smooth observables

Hadronic Framework



Cross Sections



Models

FTFP_BERT

Recommended physics list for High-Energy Physics.
Its main components are :

- **FTF** (Fritiof) hadronic string model, used above 3 GeV
- **BERT** (Bertini-like) intra-nuclear cascade model, used below 6 GeV
- Nucleus de-excitation : **P**recompound + evaporation models
- Neutron capture
- Nuclear capture of negatively charged hadrons at rest
- Hadron elastic
- Gamma- , electron- , and muon-nuclear
- Standard electromagnetic physics
- NO : neutron-HP , radioactive decay , optical photons

A few other Physics Lists

- FTFP_BERT_**HP** : as FTFP_BERT, but with **NeutronHP** for neutrons E_{kin} < 20 MeV
  - **Shielding** : similar to FTFP_BERT_HP, but using for ions also **QMD** (Quantum Molecular Dynamics)
    - QMD used in the range [100 MeV, 10 GeV] : below BIC, above FTFP
- FTFP_**INCLXX** : similar to FTFP_BERT, but using **INCL**XX for some particles
    - Protons, neutrons, charged pions below 20 GeV; FTFP above 15 GeV
- **QGSP**_BERT : similar to FTFP_BERT, but using **QGS** (Quark Gluon String) model at high energies
    - [3, 6] GeV transition BERT − FTFP ; [12, 25] GeV transition FTFP − QGSP
- QGSP_**BIC** : similar to FTFP_BERT but using QGS and BIC (Binary Cascade) instead of FTF and BERT when possible
    - Protons, neutrons : BIC < 6 GeV , FTFP in [3, 25] GeV , QGSP > 12 GeV
      Pions & kaons : BERT < 6 GeV , FTFP in [3, 25] GeV , QGSP > 12 GeV

How to use a reference Physics List

Let's consider the example of FTFP_BERT :
In your main program:

```
#include "FTFP_BERT.hh"
...
int main( int argc, char** argv ) {
  ...
  G4VModularPhysicsList* physicsList = new FTFP_BERT;
  runManager->SetUserInitialization( physicsList );
  ...
}
```

NeutronHP (ParticleHP)

An interesting complication: Neutrons

- Neutrons are abundantly produced
  - Mostly “soft” neutrons, produced by the de-excitation of nuclei, after hadron-nucleus interactions
  - It is typically the 3rd most produced particle (after electrons and gammas)
- Before a neutron “disappears” via an inelastic interactions (or decays or exits the world volume), it can have many **elastic scatterings** with nuclei, and eventually can “thermalize” in the environment
- CPU time can vary by an order of magnitude depending on the physical accuracy of the **neutron transportation** simulation
  - For typical high-energy applications, a simple treatment is enough (luckily!)
  - For other applications, a more precise, **data-driven and isotope-specific** treatment is needed, especially for neutrons with kinetic energies below **~ MeV**

Neutron High Precision (**HP**)

- **H**igh **P**recision treatment of low-energy neutrons
  - **Ekin < 20 MeV** , down to thermal energies
  - Includes 4 types of interactions:
    elastic scattering, **radiative capture**, **fission**, **inelastic scattering**
  - Based on evaluated neutron scattering data libraries
    (pointed by the environmental variable **G4NEUTRONHPDATA**)
  - It is precise, but very slow !
- Not needed for most high-energy applications; useful for:
  - Cavern background, shielding, radiation damage, radio-protection
- Not used in most physics lists
  - If you need it, use one of the **_HP** physics lists:
    FTFP_BERT_**HP** , QGSP_BERT_**HP** , QGSP_BIC_(All)**HP** , Shielding(LEND)

Notes about NeutronHP (1/2)

- Because of several reasons (binned look-up data tables, inclusive - incomplete, without correlations – information, *etc.*) there will always be small energy non-conservations
  - For all types of interactions (elastic, capture, fission and inelastic)
  - To avoid that, by default Geant4 uses some “tricks” (e.g. emitting some gammas) to conserve energy-momentum. This, however, can affect the average values, so for applications (like nuclear reactors) which care about energy conservations on average, the following enviromental variable should be set to avoid any “adjustment”: **G4NEUTRONHP_DO_NOT_ADJUST_FINAL_STATE**
- Doppler broadening of the resonances, due to target thermal motion, is calculated on-the-fly (from T = 0 K values)
  - Very CPU intense: for those applications that do not need it, it can be switched off by setting the environmental variable

G4NEUTRONHP_NEGLECT_DOPPLER

Notes about NeutronHP (2/2)

- Geant4 Neutron Data Libraries:
  - The most recent one, G4NDL4.6, is based on JEFF-3.3
    - Previous ones are based on ENDF/B
    - The main motivation for using JEFF instead of ENDF/B is that it provides the closest agreemet with MCNP; moreover, it provides also trans-uranic isotopes
  - Alternative neutron data libraries for Geant4 are available from IAEA (***https://www-nds.iaea.org/geant4/***)
    - Based on ENDF, JEFF, JENDL, CENDL, BROND and ROSFOND neutron data libraries

Thermal Scattering

- A special treatment is needed for handling neutron elastic scattering at thermal energies **< 4 eV** from chemically bound atoms
  - At thermal neutron energies, atomic translational motion as well as vibration and rotation of the chemically bound atoms affect the neutron scattering cross section and the energy and angular distribution of secondary neutrons
  - Based on the **S(α , β)** model
  - Thermal neutron scattering files from ENDF/B-VII thermal data
    - There are **~ 20 materials** : al_metal, be_metal, be_beo, benzen, d_heavy_water, d_ortho_d2, d_para_d2, fe_metal, graphite, h_l_ch4, h_ortho_h2, h_para_h2, h_polyethylene, h_s_ch4, h_water, h_zrh , o_beo, o_uo2, u_uo2, zr_zrh
  - Can be activated with the elastic constructor **G4ThermalNeutrons**
    - *physicsList → RegisterPhysics(**new G4ThermalNeutrons(** 0 **)**);*
  - Example:

    examples/extended/hadronic/Hadr04

ParticleHP

- Extension of NeutronHP for : **p** , **d** , **t** , **3He** , **α**
  - For high-precision elastic and inelastic interactions below **200 MeV**
    - Of interest for medical and nuclear physics
  - Also data-driven, based on the **TENDL** database
    - Based on **TALYS** code
    - Optional database that can be downloaded from the Geant4 web site
      - **G4TENDL1.3.2**
    - Need to be pointed by the environmental variable **G4PARTICLEHPDATA**
  - The two codes, NeutronHP and ParticleHP, have been merged
  - Validation in progress, good comparisons so far with MCNP
  - Available in the **QGSP_BIC_AllHP** reference physics list

Radioactive Decay

Nuclides in Geant4 (1/2)

- Based on data files from the Evaluated Nuclear Structure Data Files (**ENSDF**), Geant4 knows about **~ 6'500 nuclides** with half-life **> 1 ns**
  - ~3'000 ground states + ~3'500 meta-stable states (isomers)
  - For Geant4 10.6 , their properties (Z , A , E , τ , *etc.*) are in **G4ENSDFSTATE2.2** (pointed by G4ENSDFSTATEDATA)

- Two ways to have **isomers** (unstable nuclides) in Geant4 :

  1. Radioactive source as a primary particle (at rest or in-flight)

  - *e.g.* Na24m :

```
/gun/particle ion
/gun/ion 11 24 0 472. keV    # Z  A  Q  E
```

Nuclides in Geant4 (2/2)

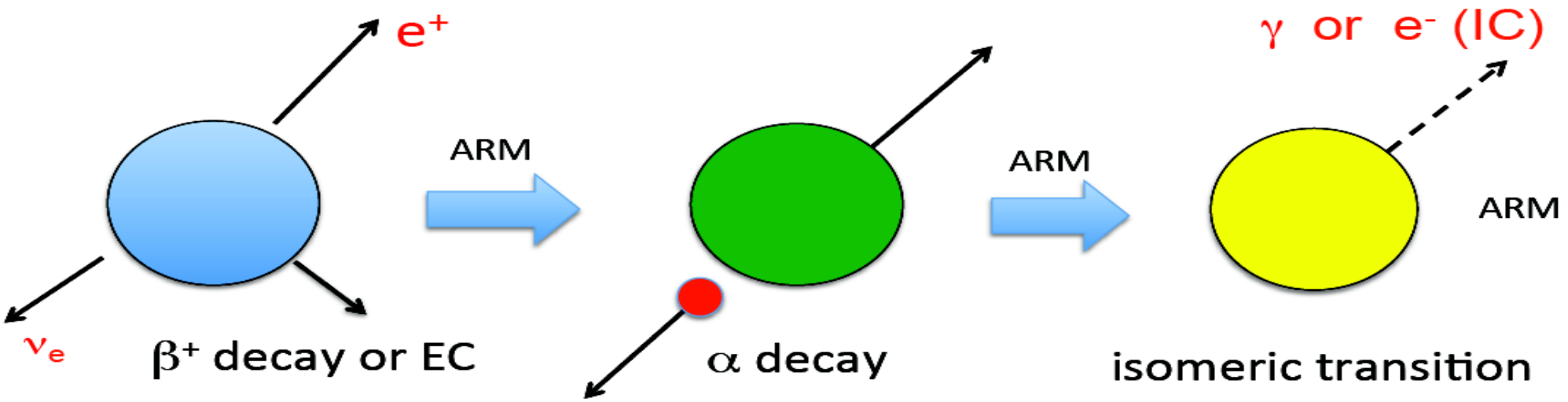
2. Induced radioactivity (activation) :
in inelastic hadronic collisions, de-excitation nuclear models can create isomers with half-life greater than a threshold:

- **1 µs** by default if RDM is not used (*i.e.* for HEP applications)
  - This threshold was 1000 seconds in Geant4 version 10.6 and 10.6.p01, but unphysical results were reported, and the threshold was changed in Geant4 version 10.6.p02 ; the same for the coming release 10.7
  - This threshold can be changed (currently) only via C++ interface
  - Obviously, without RDM, these isomers do not decay !
- **1 ns** by default when RDM is used in analogue mode
  - This threshold can be changed via UI command
- **1 µs** by default when RDM is used in biasing mode
  - This threshold can be changed via UI command

Radioactive Decay

- Process to simulate radioactive decay of (unstable) nuclei, either in flight or at rest
- So far implemented the following types of decay :
  α , **β-**, **β+** , **γ** (*i.e.* isomeric transitions, and e- Internal Conversions (IC))
  EC (Electron Capture) , **p** , **n** , **t** , **spontaneous-fission**
  - Not yet : pp , nn , β-delayed p , β-delayed n , ββ-- , ββ++ , *etc.*
- Empirical and data-driven
  - Data files from Evaluated Nuclear Structure Data Files (**ENSDF**)
    - As of Geant4 10.6 , these are in **RadioactiveDecay5.4**
      pointed by the environmental variable **G4RADIOACTIVEDATA**
    - These data files contain properties such as:
      half-lives, nuclear level structure for parent and daughter nuclides,
      type of decay, decay branching ratios, energy of decay process, *etc.*

Radioactive Decay Chain



EC: electron capture

IC: internal conversion

ARM: atomic relaxation model

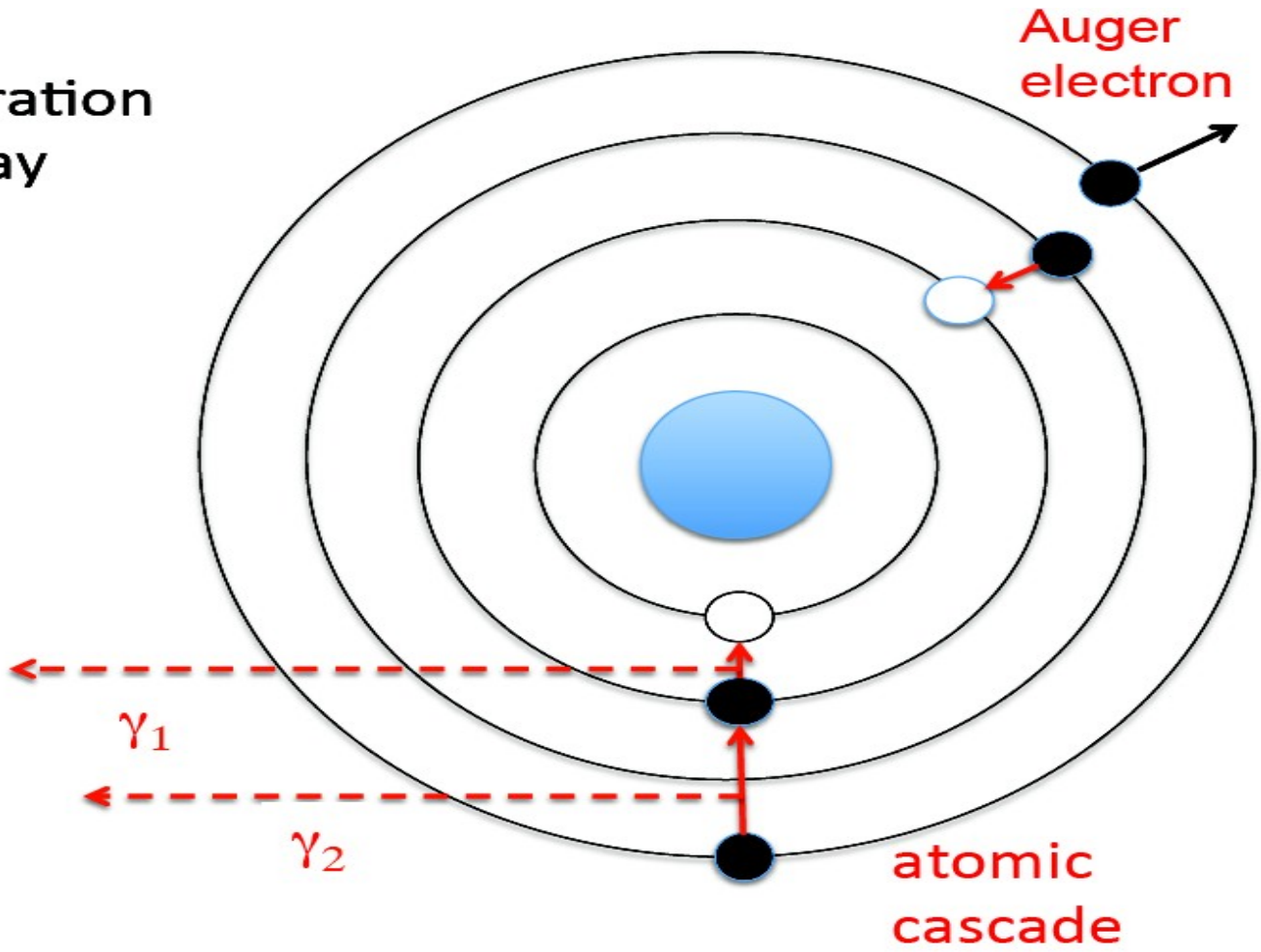
Atomic Relaxation Model

electron shell configuration may change after decay

inner holes filled by atomic cascade

either photons or Auger electrons are emitted

fluorescence option also available



Gamma (or Electron) Emission

- Nuclides with half-life **< 1 ns** are forced to decay immediately
  - This threshold can be set via UI command "**/grdm/hlThreshold**"
  - For biasing, the default value is **1 µs**
- Prompt de-excitation is done via **G4PhotonEvaporation**
  - Uses **ENSDF** files with all known gamma levels for ~ 3'000 isotopes (in total ~ 25'500 levels with half-life > 10^-23 s)
    - As of Geant4 10.6 , these are in **PhotonEvaporation5.5** pointed by the enviromental variable **G4LEVELGAMMADATA**
  - Internal conversion (*i.e.* nuclear de-excitation via emission of atomic electrons) is enabled as a competing process to gamma de-excitation
- Option to enable atomic relaxation after decay
  - When Radioactive Decay is activated, Fluorescence and Auger emissions are switched on by default (overriding any EM default settings). User can use UI command to change this

Sampling Radioactive Decay : **Analogue** mode (default)

Several options available via UI commands :

- Enable/disable radioactive decay in various geometry volumes
  "**/grdm/selectVolume**" , "**/grdm/deselectVolume**"
- Limits the nuclei in which radioactive decay can be applied (useful to limit the decay chain, i.e. to avoid decays of daughters)
  "**/grdm/nucleusLimits**"
- Supply a user-defined radioactive decay datafile for a given isotope (useful, for instance, to amplify rare decay branches)
  "**/grdm/setRadioactiveDecayFile**"
- Supply a user-defined evaporation datafile for a given isotope
  "**/grdm/setPhotoEvaporationFile**"
- Switch on/off atomic relaxation
  "**/grdm/applyARM**" (default: true)

Sampling Radioactive Decay : **Biased** mode (alternative)

Several options available via UI commands :

- Set all decay branch probabilities equal
  "**/grdm/BRbias**" default: **true**
- "Splitting" : perform nuclear decay N times for each event
  "**/grdm/splitNuclei**" default: 1
- Activation : integrate decay chain over time windows (in seconds)
  "**/grdm/sourceTimeProfile**" , "**/grdm/decayBiasProfile**"
  using Bateman equations
  "**/grdm/analogueMC false**" (default: true)
- Collimation of decay products
  "**/grdm/decayDirection**" , "**/grdm/decayHalfAngle**"

Using Radioactive Decay (1/2)

- Activate Radioactive Decay by one of the following 3 ways:

1. Choose one of the reference physics lists that have already Radioactive Decay activated, *e.g.* **_HP** physics list or **Shielding**
2. Add the physics constructor **G4RadioactiveDecayPhysics** to a reference physics list (which does not have it already), *e.g.*

```
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics( new G4RadioactiveDecayPhysics );
runManager->SetUserInitialization( physicsList );
```

3. Do something like this in your own physics list :

```
G4RadioactiveDecay* rDecay = new G4RadioactiveDecay;
G4PhysicsListHelper* plh =  G4PhysicsListHelper::GetPhysicsListHelper();
rDecay -> SetICM( true );    // Internal conversion : obsolete, always true!
rDecay -> SetARM( true );  //  Atomic relaxation
plh -> RegisterProcess( rDecay, G4GenericIon::GenericIon() );
```

Using Radioactive Decay (2/2)

- Many options can be set via UI commands

- Set environmental variables to point to the data libraries:
  - **G4ENSDFSTATEDATA -> G4ENSDFSTATE2.2**
  - **G4LEVELGAMMADATA -> PhotonEvaporation5.5**
  - **G4RADIOACTIVEDATA -> RadioactiveDecay5.4**

  in most cases, this is done automatically without the need for the user to do anything !

Examples of using RDM

examples/extended/radioactivedecay

- ***rdecay01/***
  - Shows basic features of the radioactive decay of nuclei : energy spectrum of emitted particles, time of life, activity
  - Analogue mode only, with user-defined RadioactiveDecay and PhotonEvaporation files
- ***rdecay02/***
  - Induced radioactivity by nuclear reactions
  - Shows advanced features – *e.g.* selected decay channels, time window, *etc.* – in both analogue and biasing mode
- ***Activation/***
  - Induced radioactivity by nuclear reactions
  - Shows the evolution of each meta-stable isomer as a function of time, taking into account the time of exposure of the beam; analogue mode only

Biasing in Hadronic Physics

Built-in Biasing in Hadronics

- Radioactive Decay
  - Via UI commands (see before)
- Cross Sections
  - Possibility to scale any hadronic cross section via the method
    G4HadronicProcess::BiasCrossSectionByFactor(G4double aScale)
  - No UI commands available : need to write some user code to call it, *e.g.*
    theMuonNuclearProcess->BiasCrossSectionByFactor(1000.0);
- Leading Particle Biasing
  - At each hadronic interaction, keep only the most energetic particle (and randomly one particle of each species: meson, baryon, π°, γ) via the method:
    G4HadFinalState* G4HadLeadBias::Bias(G4HadFinalState* result)
  - No UI commands available : need to modify the PostStepDoIt method of hadronic processes (to which we want to apply this biasing) to create an instance of the class G4HadLeadBias and call its Bias method

Built-in Biasing in Hadronics

- Radioactive Decay **OK**
  - Via UI commands (see before)
- Cross Sections **Need to write some user code...**
  - Possibility to scale any hadronic cross section via the method
    G4HadronicProcess::BiasCrossSectionByFactor(G4double aScale)
  - No UI commands available : need to write some user code to call it, *e.g.*
    theMuonNuclearProcess->BiasCrossSectionByFactor(1000.0);
- Leading Particle Biasing **Complicated to use in practice !**
  - At each hadronic interaction, keep only the most energetic particle (and randomly one particle of each species: meson, baryon, π°, γ) via the method: **G4HadFinalState* G4HadLeadBias::Bias(...)**
  - No UI commands available : need to modify the PostStepDoIt method of hadronic processes (to which we want to apply this biasing) to create an instance of the class G4HadLeadBias and call its Bias method

Example of Biasing a cross section (1/2)

```
int main( … ) {
   …
   G4VModularPhysicsList* physicsList = new FTFP_BERT;
   physicsList->RegisterPhysics( new MySpecialPhysicsConstructor );
   runManager->SetUserInitialization( physicsList );
   …
}

class MySpecialPhysicsConstructor : public G4VPhysicsConstructor {
   public:
      …
      virtual void ConstructParticle() {};
      virtual void ConstructProcess();
      …
};
```

Example of Biasing a cross section (2/2)

```
void MySpecialPhysicsConstructor::ConstructProcess() {
   G4HadronicProcess* theInelasticProcess = nullptr;
   G4ProcessVector* pvec =
         G4MuonMinus::MuonMinus()->GetProcessManager()->GetProcessList();
   for ( size_t i = 0; i < pvec->size(); i++ ) {
      if ( (*pvec)[i]->GetProcessName() == "muonNuclear" ) {
         theInelasticProcess = static_cast< G4HadronicProcess* >( (*pvec)[i] );
         break;
      }
   }
   if ( theInelasticProcess ) {
      theInelasticProcess->BiasCrossSectionByFactor( 1000.0 );
   }
}
```

Leading Particle Biasing and Beyond...

- Much simpler and neat to use the “generic biasing” approach
  - See tomorrow’s lectures
- Moreover, “generic biasing” allows to do much more :
  - Cross section biasing
  - Force collision biasing
  - Geometry-based biasing
    - including the possibility to use parallel geometries
  - Hadronic-model per region