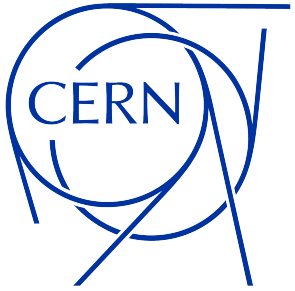
Hadronic Physics II

Geant4–11.1 reference - Based on previous Geant4 courses

Lorenzo Pezzotti
CERN EP-SFT

10th International Geant4 Course in Korea 2023
@ Jeju National University



Covered topics

- Reminder of *Hadronic Physics I*
- **High-precision neutron treatment (NEUTRON_HP)**
- **Nuclides and radioactive decay**
- **Brief: nucleus-nucleus (ion-ion), lepto-nuclear and gamma-nuclear interactions**
- **Hadronic physics biasing**
- **Hadronic physics validation**

Hadronic cross sections (recap)

Geant4 separates hadronic cross section datasets from hadronic final states models

- Hadronic cross section datasets refer to the *total cross section* for a hadronic-nucleus interaction (`G4HadronicProcess::GetElementCrossSection()`)
  - For each combination of particle, energy and target-material ≥ 1 cross sections must be specified in a physics list (in case more than one is available the last one set is used)
- In Geant4 there are only 2 types of hadronic cross sections (neutrons are an exception):
  - The elastic cross section describing the process for which the projectile and the target nucleus survive and no additional particles are created
  - The inelastic cross section describing the process for which any other final state is created

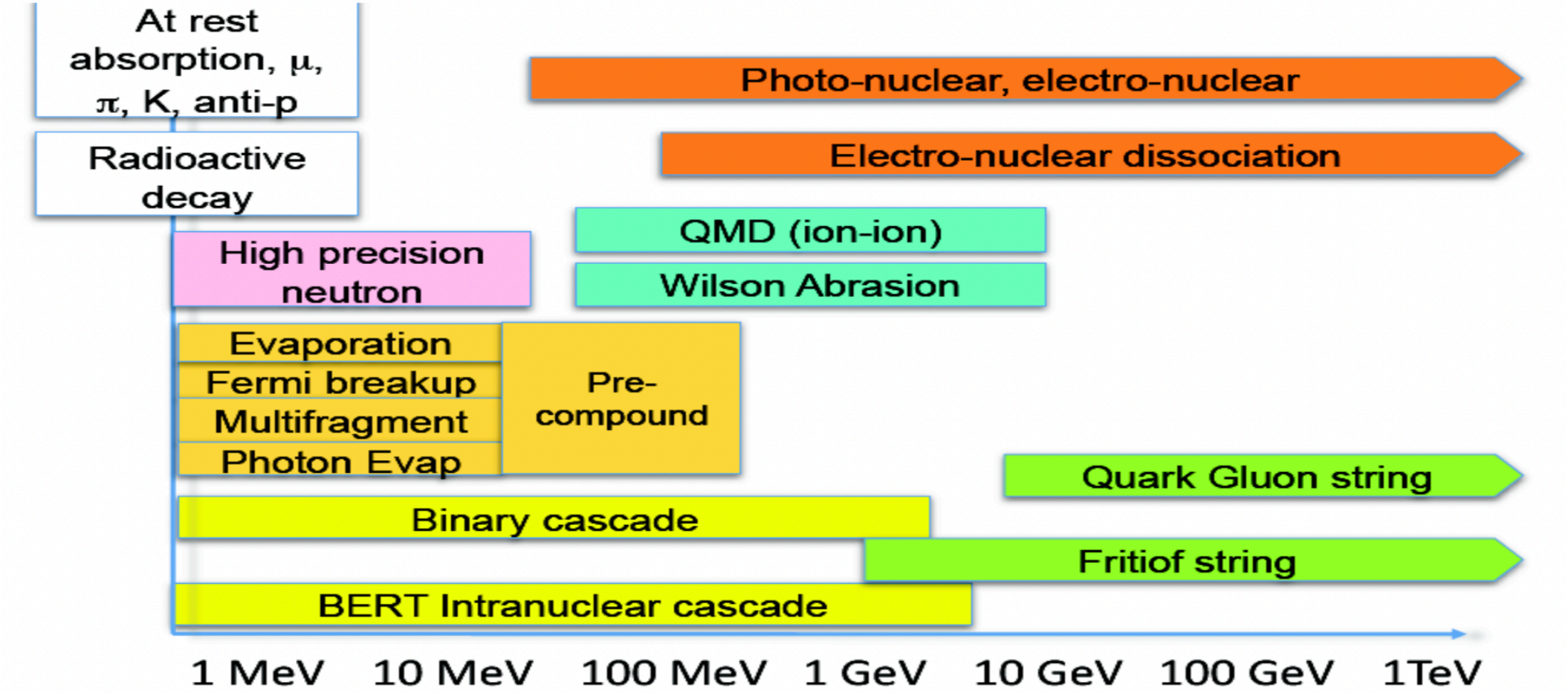
NOTE: there are no *differential cross sections* for hadronic interactions. To get the differential cross section for secondaries in a given phase space one has to multiply the total cross section by the fraction of events corresponding to that final state

Hadronic final states models (recap)

`Geant4` separates hadronic cross section datasets from hadronic final states models

- Hadronic final states models describe the properties of the *secondaries* from the hadronic interaction (`G4HadronicProcess::GetHadronicInteraction()`)
  - For each combination of particle, energy and target-material, 1 or 2 final states model(s) must be specified in a physics list
    (in case two are present in the same energy range, one is selected according to a linear probability interpolation)
- The hadron elastic process competes with the hadron inelastic one
  (as with any other process, e.g. ionization, bremsstrahlung, transportation, …)
- When an inelastic hadronic process occurs, i.e. the process is selected against the others,
  a final state (distribution of secondaries) is produced by one of the hadronic models working the the current energy range

Partial hadronic final states models inventory (recap)



NOTE: Physics list names are derived from the hadronic models used (see lesson *how to choose a physics list*)

A special case: neutrons (recap) (1/2)

- Neutrons are crucial particles for every simulation involving hadronic physics (*e.g. hadronic calorimeters, nuclear reactors, shielding, …*)
- Typically several *soft* neutrons are released from nuclei de-excitation after the hadron inelastic process (take care *lepto-nuclear* and *gamma-nuclear* processes release neutrons too)

Bash - Example: inspect neutron processes (FTFP_BERT)

```
/particle/select neutron # UI command
/particle/process/dump   # UI command

G4ProcessManager: particle[neutron]
[0]=== process[Transportation :Transportation] Active
[1]=== process[Decay :Decay] Active
[2]=== process[hadElastic :Hadronic] Active
[3]=== process[neutronInelastic :Hadronic] Active
[4]=== process[nCapture :Hadronic] Active
[5]=== process[nKiller :General] Active
```

- The `nCapture` (radiative capture) process can happen *in-flight* (differently from the radiative capture at rest for negatively charged hadrons), the neutron is absorbed by a nucleus which emits γ

NOTE: In hadronic showers neutrons are typically the third most abundant particles produced, after γ and e^-

A special case: neutrons (recap) (2/2)

- As neutrons do not loose energy by ionization, they usually undergo many elastic collisions (eventually down to thermalization) with nuclei before being killed
- In `Geant4` neutrons can be killed by the hadron inelastic process, the decay process, the radiative capture process, the `nKiller` process (default threshold $10\mu s$) or by the Geant4 kernel (*out-of-world killed*)
- Neutron cross sections are wild: they depend, sometimes dramatically and unpredictably, on the neutron energy and the target element/isotope
- The CPU time varies up to one order of magnitude depending on the accuracy of neutron simulation
  - For most high-energy physics and nuclear applications, a simplified and fast description is usually sufficient (`FTFP_BERT, QGSP_BERT, …`)
  - Where more accuracy is needed, `Geant4` offers a more precise, data-driven and isotope-specific treatment for neutrons, for kinetic energies < 20 MeV (`physics_list_HP, …`)

Neutron high precision treatment

The `Geant4` high-precision treatment of neutrons is applicable to neutron with kinetic energy < 20 MeV

- With respect to the default neutron description it includes:
  - More precise models for the *elastic*, *inelastic* and *radiative capture* processes below 20 MeV, and,
  - the *fission* process for which the target nucleus breaks up (see next slide)
- These models are based on neutron data libraries pointed by the `G4NEUTRONHPDATA env`
- It is precise but much slower (up to one order of magnitude) than the default neutron treatment
- It is typically used for cavern background, neutron shielding and radio protection studies
- The user picks up the high-precision neutron treatment by registering any of the _HP physics lists (`FTFP_BERT_HP, QGSP_BERT_HP, …`) or the `SHIELDING` physics list

Bash - execution using **FTFP_BERT**

```
Hadronic Processes for neutron


Process: hadElastic
Model:             hElasticCHIPS: 0 eV  ---> 100 TeV
Cr_sctns:     G4NeutronElasticXS: 0 eV  ---> 100 TeV

Process: neutronInelastic
Model:                      FTFP: 3 GeV ---> 100 TeV
Model:            BertiniCascade: 0 eV  ---> 6 GeV
Cr_sctns:   G4NeutronInelasticXS: 0 eV  ---> 100 TeV

Process: nCapture
Model:               nRadCapture: 0 eV  ---> 100 TeV
Cr_sctns:     G4NeutronCaptureXS: 0 eV  ---> 100 TeV
```

NOTE: to avoid printout of processes list add UI command
`/process/had/verbose 0`
`/process/em/verbose 0`
before `/run/initialize`

Bash - execution using **FTFP_BERT_HP**

```
Hadronic Processes for neutron


Process: hadElastic
Model:             hElasticCHIPS: 19.5 MeV ---> 100 TeV
Model:          NeutronHPElastic: 0 eV  ---> 20 MeV
Cr_sctns:     NeutronHPElasticXS: 0 eV  ---> 20 MeV
Cr_sctns:     G4NeutronElasticXS: 0 eV  ---> 100 TeV

Process: neutronInelastic
Model:                      FTFP: 3 GeV ---> 100 TeV
Model:            BertiniCascade: 19.9 MeV ---> 6 GeV
Model:          NeutronHPInelastic: 0 eV  ---> 20 MeV
Cr_sctns:   NeutronHPInelasticXS: 0 eV  ---> 20 MeV
Cr_sctns:   G4NeutronInelasticXS: 0 eV  ---> 100 TeV

Process: nCapture
Model:          NeutronHPCapture: 0 eV  ---> 20 MeV
Model:               nRadCapture: 19.9 MeV ---> 100 TeV
Cr_sctns:     NeutronHPCaptureXS: 0 eV  ---> 20 MeV
Cr_sctns:     G4NeutronCaptureXS: 0 eV  ---> 100 TeV

Process: nFission
Model:          NeutronHPFission: 0 eV  ---> 20 MeV
Model:                 G4LFission: 19.9 MeV ---> 100 TeV
Cr_sctns:     NeutronHPFissionXS: 0 eV  ---> 20 MeV
Cr_sctns:                 ZeroXS: 0 eV  ---> 100 TeV
```

More on neutron HP (1/3)

- The high-precision models rely on data libraries.
  Most often the look-up data tables provide *inclusive* information (without full kinematic of interaction), they are discontinuous due to binning, and sometimes do not preserve correlation between secondaries

  → there might be *event-by-event* violation of energy, momentum or baryon number conservation in inelastic scattering (but also in elastic, capture and fission)

- To avoid that, by default `Geant4` uses internally some tricks (*e.g.* emitting soft gammas) to conserve energy on an event-by-event basis (but the experimental measurement of the average deposited energy is not reproduced)

  - If the average deposited energy is of primary importance (dosimetry, nuclear reactors, …)
    the user can turn off every adjustment via

```
/process/had/particle_hp/do_not_adjust_final_state true
=== G4ParticleHPMessenger CHANGED PARAMETER DoNotAdjustFinalState TO 1 ===
-> Disabled the adjustment of the final state for getting better conservation !
```

More on neutron HP (2/3)

- At low energies (< 20 MeV), the temperature of the material plays a role on the neutron elastic scattering due to target thermal motion
- Data libraries for `Neutron_HP` refer to $T = 0$ K. On-the-fly Geant4 adjusts the values due to target thermal motion (*Doppler broadening*) according to the material temperature
  - CPU intensive and important for epithermal neutrons (kinetic energy $< k \times 10^2$ KeV).
    It is possible to use only data at $T = 0$ K while speeding-up the simulation, via

```
/process/had/particle_hp/neglect_Doppler_broadening true
=== G4ParticleHPMessenger CHANGED PARAMETER NeglectDoppler TO 1 ===
-> Switched off the Doppler broadening due to the thermal motion of the target nucleus:
on-the-fly Doppler broadening will be neglected in the cross section calculations of
capture, elastic, fission and inelastic reactions/scatterings of neutrons below 20 MeV.
This option provides a significant CPU performance advantage !
```

More on neutron HP (3/3)

- The most recent Neutron Data Library is `G4NDL4.7`
  - It is based on JEFF-3.3 (european), previously ENDF/B (american) was used
  - JEFF currently provides a better agreement w.r.t. the MCNP Monte Carlo code that is considered a standard reference in neutronics
  - Additionally, JEFF provides also transuranic isotopes
- Alternative neutron data libraries for `Geant4` are available from IAEA website based on ENDF, JEFF, JENDL, CENDL, BROND and ROSFOND

Neutron thermal scattering

- To describe elastic scattering of thermal neutrons (kinetic energy < 4 eV) it is mandatory to include the effect of the molecular structure of the medium (not only the nucleus)
  - Geant4 description is based on the *Thermal Scattering Law* (TSL) and relies both on experimental measurements and molecular dynamics calculations (see `examples/extended/hadronic/Hadr04`)
  - Thermal neutron scattering file come from JEFF-3.3 and ENDF/B-VIII-0 thermal data
    - There are only ~30 materials for which we can have this special thermal scattering modelling: *benzene, para-hydrogen, ortho-hydrogen, para-deuterium, ortho-deuterium, uranium-dioxide, ice, heavy-water, beryllium-oxide, uranium-nitride, liquid-methane, zirconium-hydride, yttrium-hydride, water, polyethylene, polymethy-methacrylate, graphite, silicium-carbide, graphite- porosity-10%, graphite-porosity-30%, beryllium-metal, iron-metal, solid-metane, aluminium-metal, sapphir, Si28, Si29, Si30, magnesium-metal, …*
- HP physics lists by default do not include the thermal scattering description for neutron < 4 eV, it can be included by the user, as

```
G4VModularPhysicsList *physicsList = new FTFP_BERT_HP();
physicsList->SetVerboseLevel(1);
physicsList->RegisterPhysics(new G4ThermalNeutrons());
runManager->SetUserInitialization(physicsList);
```

```
                    Hadronic Processes for neutron
Process: hadElastic
  Model:              hElasticCHIPS: 19.5 MeV ---> 100 TeV
  Model:           NeutronHPElastic: 4 eV  ---> 20 MeV
  Model: NeutronHPThermalScattering: 0 eV  ---> 4 eV
  Cr_sctns: NeutronHPThermalScatteringData: 0 eV  ---> 4 eV
  Cr_sctns:         NeutronHPElasticXS: 0 eV  ---> 20 MeV
  Cr_sctns:         G4NeutronElasticXS: 0 eV  ---> 100 TeV
```

Nuclides (1/2)

Nuclides are nuclei characterized by their number of protons, neutrons and nuclear energy state

In `Geant4` there are ~6500 nuclides with half-life > 1 ns, as described in the Evaluated Nuclear Structure Data Files (ENSDF)

- ~6500 nuclides divided as: ~3000 *ground state* and ~3500 *meta-stable state* (also referred to as *isomers*, they are not stable but their half-life is > 1 ns)
- Nuclides properties are stored in the G4ENSDF2.3 data library (pointed by the `env G4ENSDFSTATEDATA`)

Bash - geant4-install/share/Geant4/data

```
$ cat G4ENSDFSTATE2.3/README

Data file for G4NuclideTable.
Properties of each state are introduced from
Evaluated Nuclear Structure Data File (ENSDF)



An example of state
6   14          6728.2  0      0.09521787    6    4.12143936e-27
and format of each enetry is
1:Z
2:A
3:Excitation energy [keV]
4: String for floating levels -,+X,+Y,+Z,+U,+V,+W,..
   - means not floating level
5:Mean life time [ns]
6:Spin [h_bar/2]
7:Dipole magnetic moment [joule/tesla]
```

Nuclides (2/2)

There are two cases for which nuclides/isomers are simulated in Geant4

- The user sets a radioactive source as primary particle, as for instance *Na24m*,

```
/gun/particle ion
/gun/ion 11 24 0 472. # <Z> <A> [<Q>] [<E>] [<flb>]
/run/beamOn 1
```

- Or, the hadron inelastic process create nuclides with half-life above a certain threshold:
  - 1 ns if Radioactive Decay process is not used. This threshold can be changed via C++ interface (not recommended)
  - 1 ns if Radioactive Decay process is used. This threshold can be changed via UI command (shown later)

NOTE: if the radioactive decay process is not registered, these meta-stable nuclides do not decay (see next slide)

Radioactive decay

Radioactive decay is a `Geant4` *process* used to simulate the decay of unstable nuclides at any energy (both *at-rest* or *in-flight*)

- Currently the following decay modes are implemented:
  α (emission of α particle), β (emission of e^- or e^+), γ (isomeric transition with emission of photons at $\simeq$ MeV), internal conversion (orbital e^- expelled due to isometric transition), electron capture (capture of orbital e^-), as well as emission of p, n, t and spontaneous fission

- Radioactive decay models are described in the `RadioActiveDecay5.6` data library pointed by the env `G4RADIOACTIVEDATA` **(**It is a database for nuclides half-lives, nuclear level of parent and daughter nuclides, decays and branching ratios)

Bash - inspect data for ^{60}Co radioactive decays

```
$ cat RadioactiveDecay5.6/z27.a60

# 60CO ( 1925.28 D )
#  Excitation  flag   Halflife  Mode    Daughter Ex flag   Intensity          Q
P            0  - 1.663442e+08
                            BetaMinus             0              1
                            BetaMinus     1332.514  -         0.12     1490.299
                            BetaMinus     2158.632  -        1e-27      664.181
                            BetaMinus     2505.753  -        99.88      317.06
```

Registering the radioactive decay process

- To include the radioactive decay process, the appropriate constructor must be registered, as

main() - register radioactive decay physics

```
#include "G4RadioactiveDecayPhysics.hh" //additional .hh files not specified

int main(){
  // code before
  G4VModularPhysicsList *physicsList = new FTFP_BERT();
  physicsList->SetVerboseLevel(1);
  physicsList->RegisterPhysics(new G4RadioactiveDecayPhysics());
  runManager->SetUserInitialization(physicsList);
  // code after
}
```

- `G4RadioactiveDecayPhysics` is included by default in _HP and `SHIELDING` reference physics lists

PrimaryGeneratorAction.cc - Example: observe the radioactive decay(s) of ^{60}Co

Available on GitHub

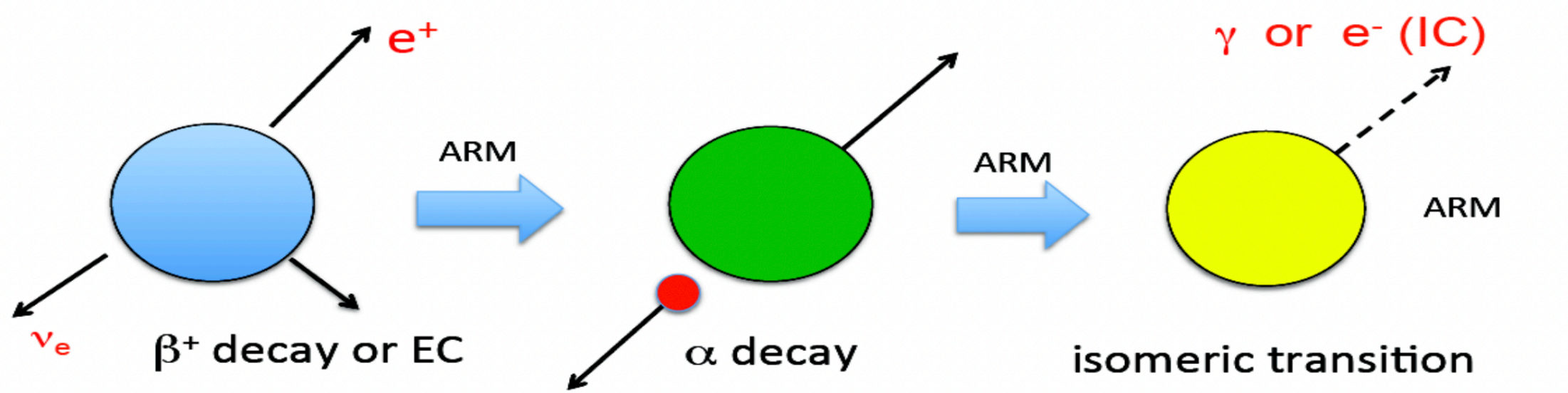
```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event *anEvent) {

  G4int Z = 27;
  G4int A = 60;
  G4double charge = 0. * eplus;
  G4double energy = 0. * keV;
  G4ParticleDefinition *ion = G4IonTable::GetIonTable()->GetIon(Z, A, energy);

  fParticleGun->SetParticleDefinition(ion);
  fParticleGun->SetParticleCharge(charge);
  fParticleGun->SetParticleEnergy(0. * MeV);
  fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

Bash - execution with /tracking/verbose 2 - ^{60}Co $\rightarrow$ ^{60}Ni + e^- + $\overline{\nu}_e$

```
*********************************************************************************
* G4Track Information:   Particle = Co60,   Track ID = 1,   Parent ID = 0
*********************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0        0        0        0         0        0        0         0       World initStep
    1        0        0        0         0        0        0         0       World RadioactiveDecay
    :----- List of 2ndaries - #SpawnInStep=  3(Rest= 3,Along= 0,Post= 0), #SpawnTotal=  3 --------------
    :         0         0         0  1.31e-07     Ni60[2505.753]
    :         0         0         0     0.264          anti_nu_e
    :         0         0         0    0.0534                e-
    :-------------------------------------------------------- EndOf2ndaries Info --------------
```

Radioactive decay chains

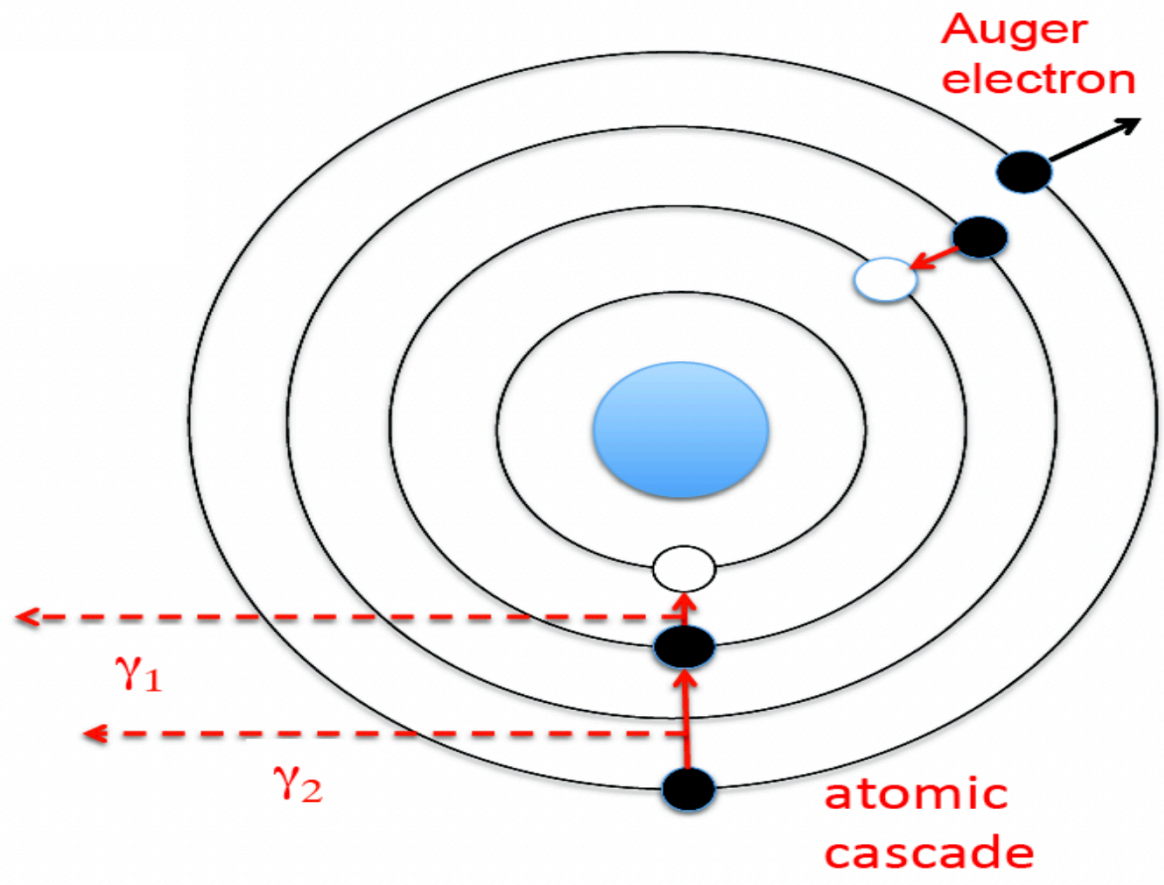
Typically the radioactive decay process triggers a *chain* of decays, as, for instance



EC = electron capture - IC = internal conversion - ARM = atomic relaxation model

Atomic relaxation model

- After a radioactive decay the electron configuration can be perturbed and result in an unstable state
- As most radioactive decays involve internal electrons (k-shell),
  the inner holes left are filled by external electrons with the emission of
  - x-rays (~keV γ's) or
  - Auger electrons
    (when the emitted gamma interacts with an external electron and that is emitted by the atom)
- Atomic relaxation is switched on by default (overriding default EM settings)
  when `G4RadioactiveDecayPhysics` is registered



More on γ/e^- emission

- By default nuclides with half-time < 1 ns are forced to decay immediately
  - The threshold can be changed with `/process/had/rdm/hlThreshold`
- Prompt de-excitation is done via `G4PhotonEvaporation` (not via radioactive decay)
  - It uses ENSDF files with gamma-levels for ~3000 isotopes (~25500 levels with half lives $> 10^{-23}$ s)
  - They are part of the PhotonEvaportation5.7 data library pointed by the `env G4LEVELGAMMADATA`
  - Internal conversion (i.e. atomic relaxation by emission of atomic e^-) is a competing process with the gamma de-excitation

Sampling radioactive decay: analogue mode

Several UI commands are available to adapt *analogue* radioactive decay simulation to the user needs

- Enable/disable radioactive decay in specific geometry volumes

```
/process/had/rdm/selectVolume <AVolume>
/process/had/rdm/deselectVolume <AVolume>
```

- Limit nuclei with radioactive decay (i.e. limit the decay chain by stopping daughters decays)

```
/process/had/rdm/nucleusLimits [<AMin>] [<AMax>] [<ZMin>] [<ZMax>]
```

- Pass a user-defined radioactive decay datafile for a given isotope

```
/process/had/rdm/setRadioactiveDecayFile [<Z_isotope>] [<A_isotope>] [<filename>]
```

- Similarly for a user-defined evaporation datafile for a given isotope

```
/process/had/rdm/setEvaporationFile [<Z_isotope>] [<A_isotope>] [<filename>]
```

- Turn-on/off atomic relaxation model

```
/process/had/rdm/applyARM [<applyARM>] # default true
```

Sampling radioactive decay: biased mode

Several UI commands are available to *bias* radioactive decay simulation

- Set all decay branching mode equally probable

```
/process/had/rdm/BRbias [<BRbias>] # default true
```

- Perform nuclear decay N times at each event

```
/process/had/rdm/splitNuclei [<NSplit>] # default 1
```

- Collimate decay products

```
/process/had/rdm/decayDirection <X> <Y> <Z>
/process/had/rdm/decayHalfAngle <HalfAngle> <Unit>
```

NOTE: to use biased radioactive decay the `G4Radioactivation` process must be registered instead of `G4RadioaciveDecay` (see examples in next slide)

Examples on radioactive decay

Three examples are available on nuclides and radioactive decays simulation in `/examples/extended/radioactivedecay/`

- **rdecay01**: illustrates basic features of nuclei radioactive decay in analogue mode with user files for radioactive decay and photo evaporation
- **rdecay02**: illustrates induced radioactivity by nuclear reactions, as well as advanced options in analogue and biased modes (e.g. selected decay channels, …)
- **Activation**: illustrates the radioactivity induced by nuclear reactions, and the evolution of each metastable isomer as a function of time and exposure time (analogue mode only)

Nucleus-nucleus interactions (brief)

All reference physics lists implement nucleus-nucleus (ion-ion) interactions via dedicated constructors

- Nearly all reference physics lists use the physics constructor `G4IonPhysics`
  - It adopts the FTF model > 3 GeV/nucleon and the BIC (binary cascade) model < 6 GeV/nucleon
- Alternative constructors are `G4IonINCLXXPhysics`, `G4IonQMDPhysics` (used by the `SHIELDING` physics list) and `G4IonPhysicsPHP` (used by the `QGS_BIC_AllHP` physics list)
- All reference physics lists adopt the Glauber-Gribov model for both nucleus-nucleus elastic and inelastic cross sections

Gamma-nuclear interactions (brief)

Currently in Geant4 there is only one approach for the photonNuclear process, common to any reference physics list

- Cross section models:
  - Cross section provided by a parameterization of experimental data from M. Kossov
  - Below 140 MeV an extended collection of photonuclear data by IAEA is used
  - Optionally, LEND gamma cross section data below 20 MeV can be used
- Final states models:
  - Above 3 GeV the QGSP model is used
  - Between 199 MeV and 6 GeV the BERT model is used
  - Below 200 MeV the Precoumpund/de-excitation model is used
  - Optionally, the LEND final-state model can be used below 20 MeV

Example: dump γ processes (FTFP_BERT)

```
/particle/select gamma
/particle/process/dump

G4ProcessManager: particle[gamma
[0]=== process[Transportation :Transportation] Active
[1]=== process[phot :Electromagnetic] Active
[2]=== process[compt :Electromagnetic] Active
[3]=== process[conv :Electromagnetic] Active
[4]=== process[Rayl :Electromagnetic] Active
[5]=== process[photonNuclear :Hadronic] Active
```

Lepto-nuclear interactions (brief)

Currently in Geant4 there is only one approach for the leptoNuclear process, common to any reference physics list (τ-nuclear process currently missing)

- Cross section models:
  - $e^{\pm}$ cross-section provided at all energies by a parameterization of experimental data from M. Kossov
  - $\mu^{\pm}$ cross-section provided at all energies by theoretical computation from Kokoulin
- Final states models:
  - First a virtual gamma is produced, then converted to a real gamma and the gamma interacts with the nucleus (*equivalent photon approximation*)
  - Below 10 GeV the BERT model is used
  - Above 10 GeV the FTFP model is used

Example: dump e^- processes (FTFP_BERT)

```
/particle/select e-
/particle/process/dump

G4ProcessManager: particle[e-]

[0]=== process[Transportation :Transportation] Active
[1]=== process[msc :Electromagnetic] Active
[2]=== process[eIoni :Electromagnetic] Active
[3]=== process[eBrem :Electromagnetic] Active
[4]=== process[CoulombScat :Electromagnetic] Active
[5]=== process[electronNuclear :Hadronic] Active
```

Hadronic physics biasing

Some *biasing* techniques are specific-to and built-in hadronic physics

- Biasing of the radioactive decay process
  - via UI commands (see previous slides)
- Biasing of the `G4HadronicProcess` cross sections via the method `BiasCrossSectionByFactor(G4double factor)`
  - No UI command available, the user must develop some code (see example in the next slide)

NOTE: We recommend to use generic biasing for any other biasing related to hadronic physics (see lesson on biasing at this course)

Biasing μ^- nuclear inelastic cross section (1/2)

main() - registering biasing constructor

```
#include "BiasPhysics.hh" # additional .hh omitted

int main(){
  // some code
  G4VModularPhysicsList *physicsList = new FTFP_BERT();
  physicsList->RegisterPhysics(new BiasPhysics());
  runManager->SetUserInitialization(physicsList);
  //some code
}
```

BiasPhysics.hh - create the biasing constructor

```
class BiasPhysics : public G4VPhysicsConstructor {
public:
  BiasPhysics();
  ~BiasPhysics() override;

public:
  void ConstructParticle() override;
  void ConstructProcess() override;
};
```

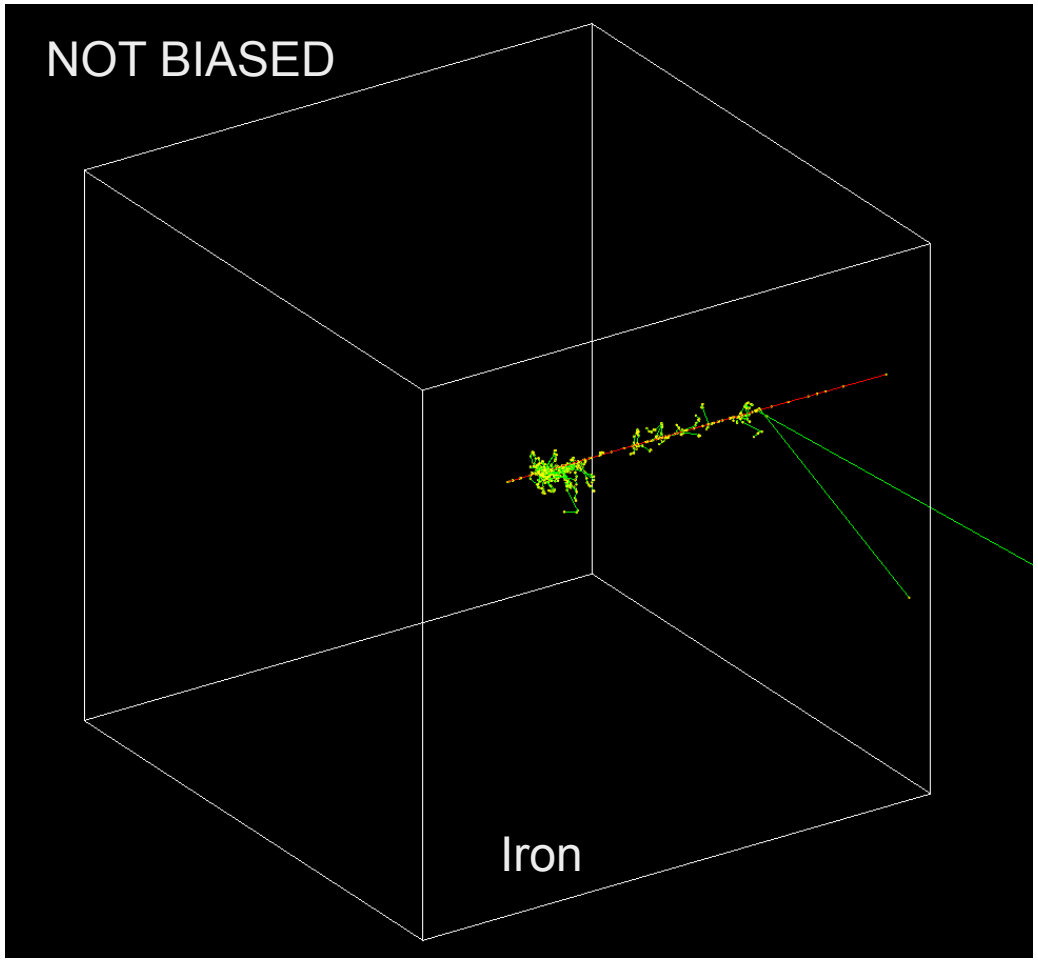
BiasPhysics.cc - implement the biasing constructor

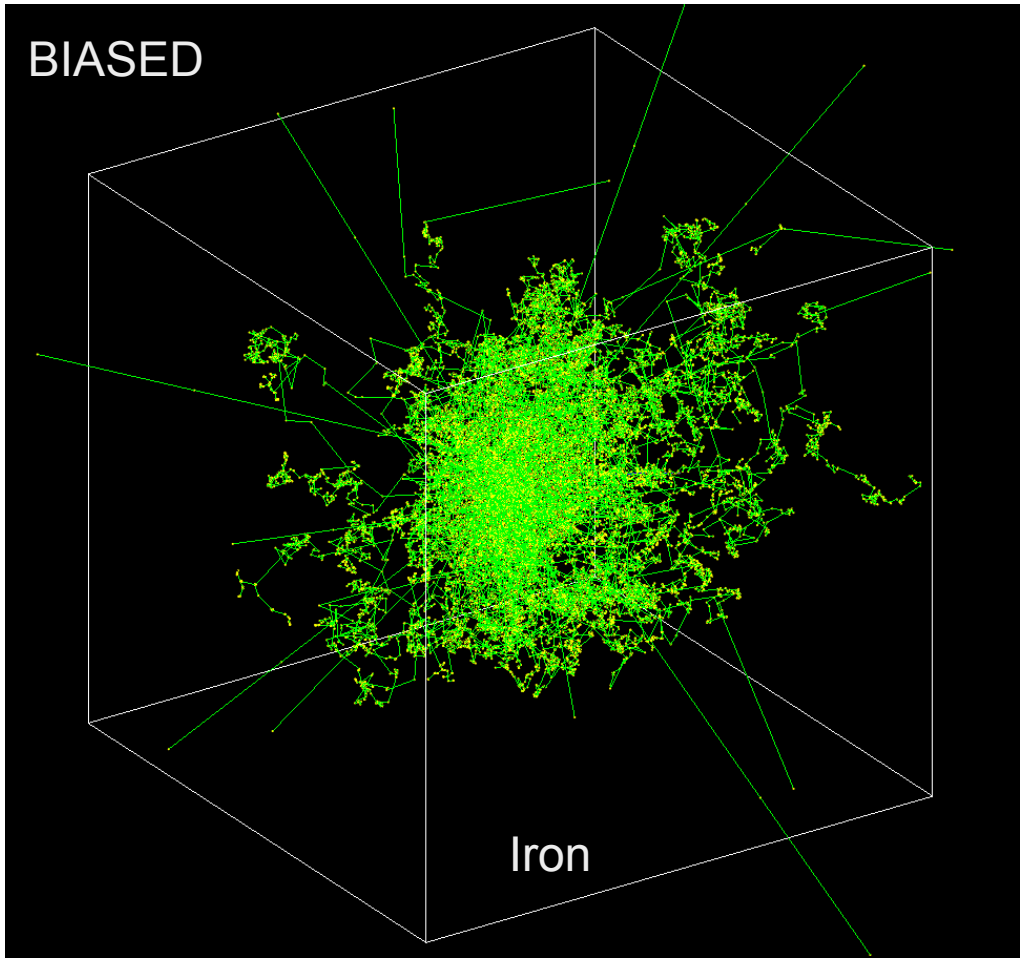
```
#include "BiasPhysics.hh" # additional .hh omitted

void BiasPhysics::ConstructProcess() {

  G4HadronicProcess *muInelasticProcess = nullptr;
  G4ProcessVector *pvec =
      G4MuonMinus::MuonMinus()->GetProcessManager()
                              ->GetProcessList();
  for (std::size_t i = 0; i < pvec->size(); i++) {
    if ((*pvec)[i]->GetProcessName() == "muonNuclear") {
      muInelasticProcess =
      static_cast<G4HadronicProcess *>((*pvec)[i]);
      break;
    }
  }
  if (muInelasticProcess) {
    muInelasticProcess
      ->BiasCrossSectionByFactor(1e7);
  }
}
```

Available on GitHub

Biasing μ^- nuclear inelastic cross section (2/2)

```
/gun/particle mu-
/gun/energy 10 GeV
/run/beamOn 1
```

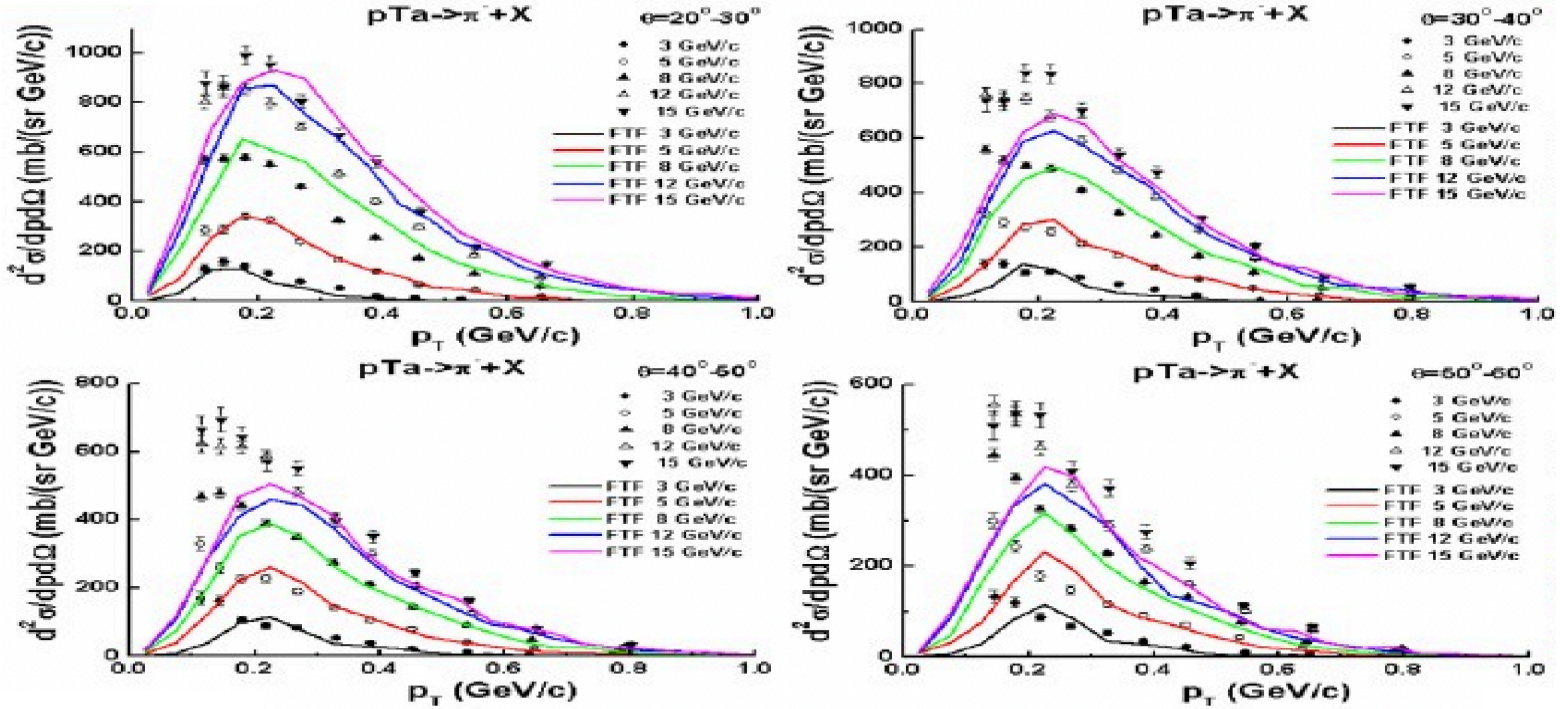




Hadronic physics validation (1/3)

Validation of hadronic models in `Geant4` stands on two main activities

- Tuning and testing of microscopic features of hadronic interactions via thin-target experiments
  - Possible to isolate a single hadronic process for a given projectile (p, π^-, …) on a thin target (Beryllium, Tallium, …)



Hadronic physics validation (2/3)

Validation of hadronic models in `Geant4` stands on two main activities

- Tuning and testing of macroscopic features of hadronic interactions via thick-target experiments
  - Possible to measure macroscopic features of hadronic showers in detectors (calorimeters)

- Example from the CALICE SiW calorimeter features:
  - 30 longitudinal layers (silicon + tungsten) with a total thickness of $24X_0$ ($\simeq 1\lambda$)
  - each silicon layer readout by 36×9 Si-cells
  - with an active area of 18×18 cm²

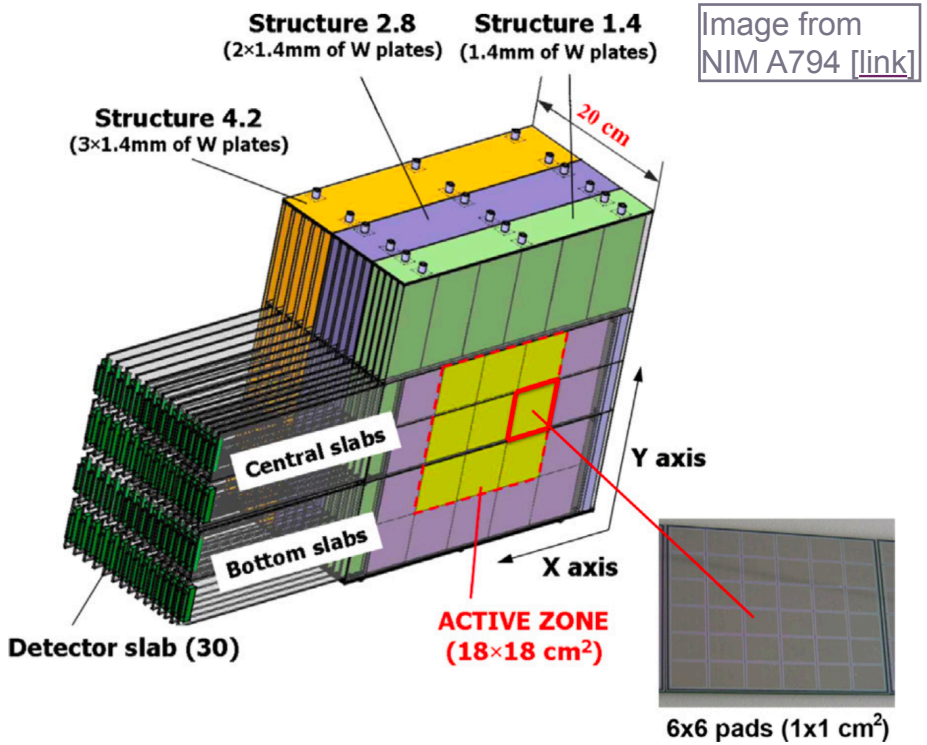
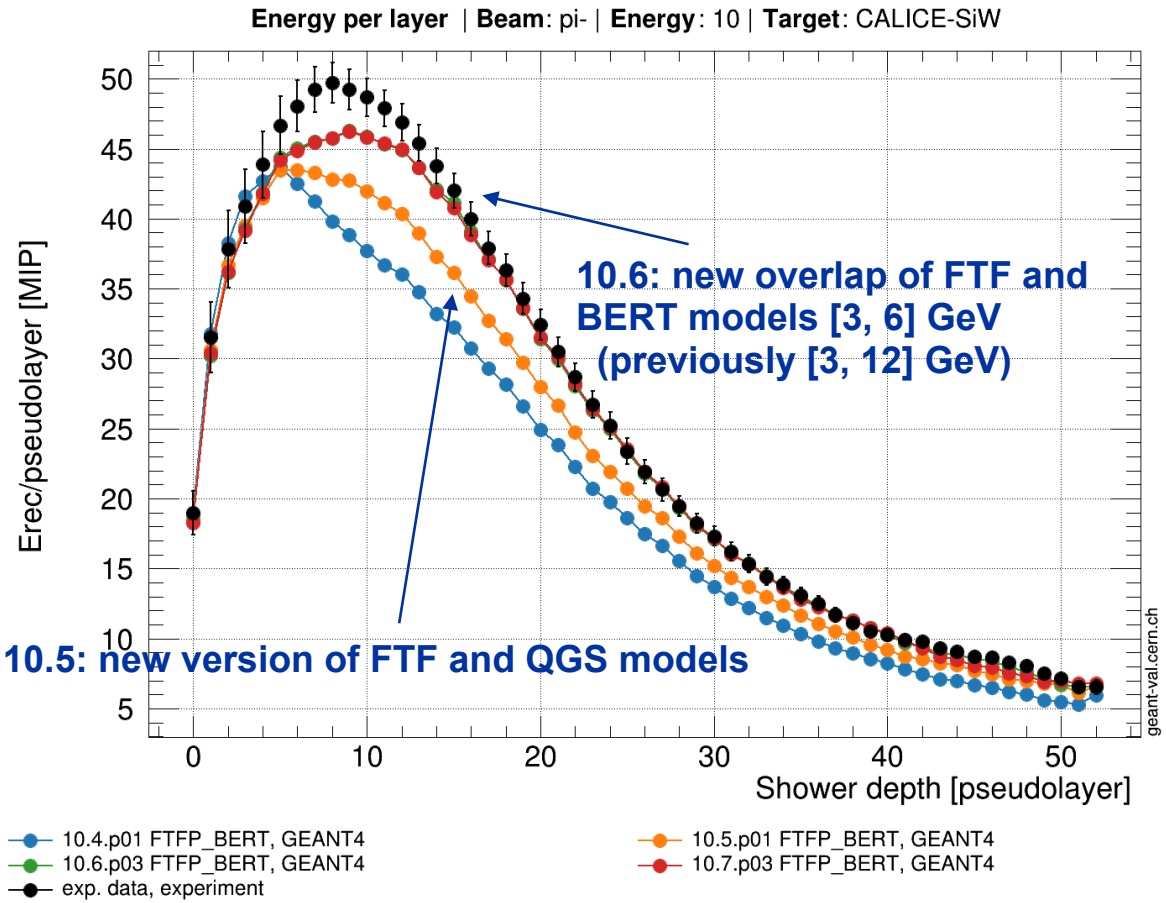


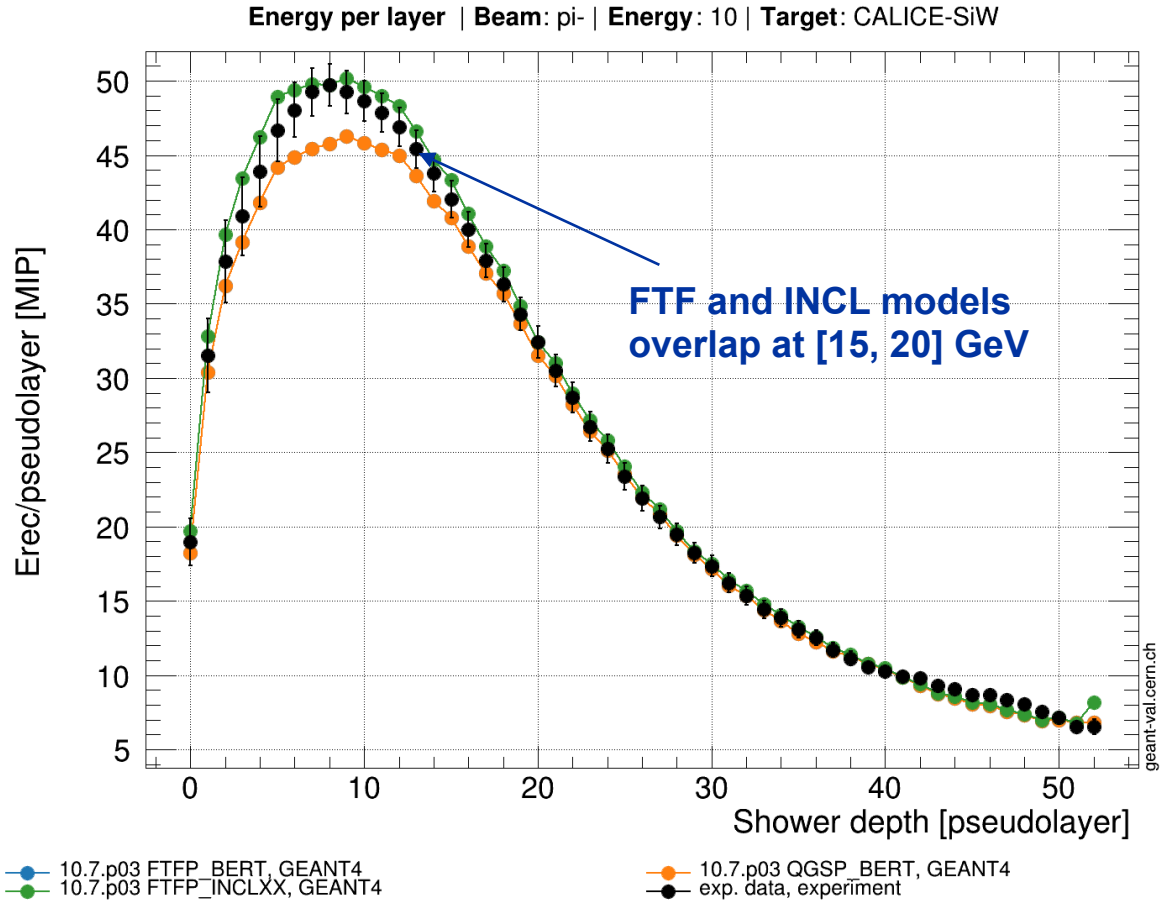
Image from NIM A794 [link]

Hadronic physics validation (3/3)

10 GeV π^-, exp. data from NIM A794



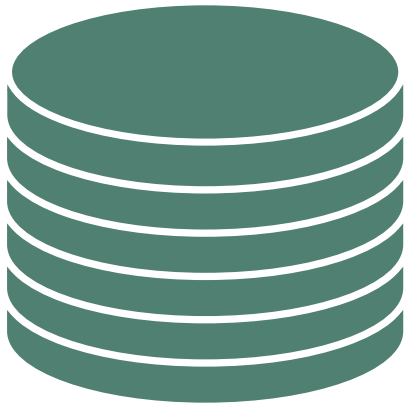
FTFP_BERT Physics List regression testing 2017-2020



Physics Lists comparison - Geant4.10.7.p03

Recap of *Hadronic Physics II*

- The `Geant4` high-precision treatment of neutrons is applicable to neutron with kinetic energy < 20 MeV, it includes:
  - more precise models for the *elastic*, *inelastic* and *radiative capture* processes below 20 MeV, and the *fission* process
  - these models are based on neutron data libraries pointed by the `G4NEUTRONHPDATA`
  - to describe elastic scattering of thermal neutrons (kinetic energy < 4 eV) it is mandatory to include the effect of the molecular structure of the medium via the constructor `G4ThermalNeutrons`
- In `Geant4` there are ~6500 nuclides with half-life > 1 ns, as described in the data files pointed by `G4ENSDF2.3`
  - To simulate their radioactive decays, the `G4RadioactiveDecayPhysics` constructor must be registered
- Some biasing techniques are built-in for hadronic physics:
  - UI commands biasing the radioactive decay process and `C++` methods biasing cross sections



Backup material

ParticleHP

ParticleHP is an extension of NeutronHP. Recently introduced by `Geant4`, it provides a high-precision treatment for *proton*, *deuterium*, *tritium*, ^3He and α

- Includes high-precision model for inelastic interaction < 200 MeV
- Similarly to NeutronHP, it is data driven and based on the TENDL data library
- To use particleHP the user must
  - Download the `G4TENDL1.4` dataset from the Geant4 website and
  - Set the `env` `G4PARTICLEHPDATA` pointing to it
- NeutronHP and ParticleHP have been merged
- Validation in progress and good comparison found so far with MCNP (mostly used for medical and nuclear physics)
- Available only in one reference physics list, QGSP_BIC_AllHP

Neutrino interactions

- Currently, most neutrino experiments use an external software package GENIE, interfaced to Geant4, to simulate neutrino interactions
- By default, in any reference physics list, there are no processes (except transportation) registered to neutrinos

Example: dump ν_e processes (FTFP_BERT)

```
/particle/select nu_e
/particle/process/dump

G4ProcessManager: particle[nu_e]
[0]=== process[Transportation :Transportation] Active
```

- However, it is possible to add charged and neutral interactions for ν_e, ν_μ, ν_τ (anti anti-neutrino):
  - On top of any reference physics list
  - Implemented very recently, validation is ongoing
  - Documentation for such processes should be included in the next release (`Geant4-11.2`)
  - Neutrino oscillations should be included in the next release (`Geant4-11.2`)