Physics Particles and Physics lists

G. Petringa (LNS-INFN)

XV Seminar on Software for Nuclear, Sub-Nuclear and Applied Physics

Alghero, Italy

27.05.2018 - 1.06.2018

Contents

- Particles
- Processes
- Tracking
- Cuts
- Physics lists

User classes

At initialization

G4VUserDetectorConstruction

G4VUserPhysicsList

G4VUserActionInitialization

Global: only one instance exists in memory, shared by all threads.

At execution

G4VUserPrimaryGeneratorAction

G4UserRunAction

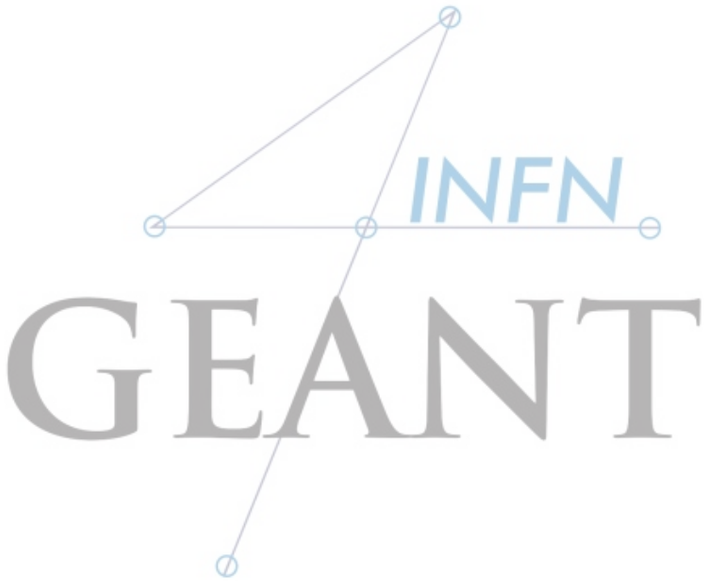
G4UserEventAction

G4UserStackingAction

G4UserTrackingAction

G4UserSteppingAction

Thread-local: an instance of each action class exists for each thread.



Particles and Processes

Particles: basic concepts

There are three levels of class to describe particles in Geant4:

G4ParticleDefinition

Particle static properties: name, mass, spin, PDG number, etc.

G4DynamicParticle

Particle dynamic state: energy, momentum, polarization, etc.

G4Track

Information for tracking in a detector simulation: position, step, current volume, track ID, parent ID, etc.

Particles: common hadrons & ions table

Particle name	Class name	Name (in GPS...)	PDG
(anti)proton	G4Proton G4AnitProton	proton anti_proton	2212 -2212
(anti)neutron	G4Neutron G4AntiNeutron	neutron anti_neutron	2112 -2112
(anti)lambda	G4Lambda G4AntiLambda	lambda anti_lambda	3122 -3122
pion	G4PionMinus G4PionPlus G4PionZero	pi- pi+ pi0	-211 211 111
kaon	G4KaonMinus G4KaonPlus G4KaonZero G4KaonZeroLong G4KaonZeroShort	kaon- kaon+ kaon0 kaon0L kaon0S	-321 321 311 130 310
(anti)alpha	G4Alpha G4AntiAlpha	alpha anti_alpha	1000020040 -1000020040
(anti)deuteron	G4Deteuron G4AntiDeuteron	deuteron anti_deuteron	1000010020 -1000010020
Heavier ions	G4Ions	ion	100ZZZAAAI*

*ZZZ=proton number, AAA=nucleon number, I=excitation level

Processes: basic concepts

How do particles interact with materials?

Responsibilities:

1. *decide **when** and **where** an interaction occurs*
   - **GetPhysicalInteractionLength...()**
     → *limit the step*
   - this requires a cross section
   - for the **transportation** process, the distance to the nearest object

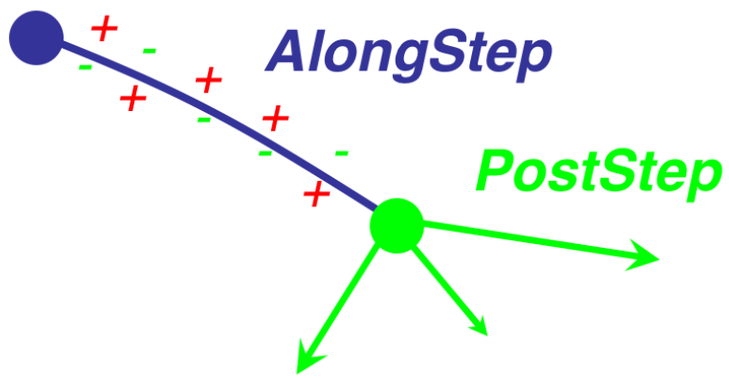
2. *generate the **final state** of the interaction*
   - changes momentum, generates secondaries, etc.)
   - method: **DoIt...()**
   - this requires a model of the physics

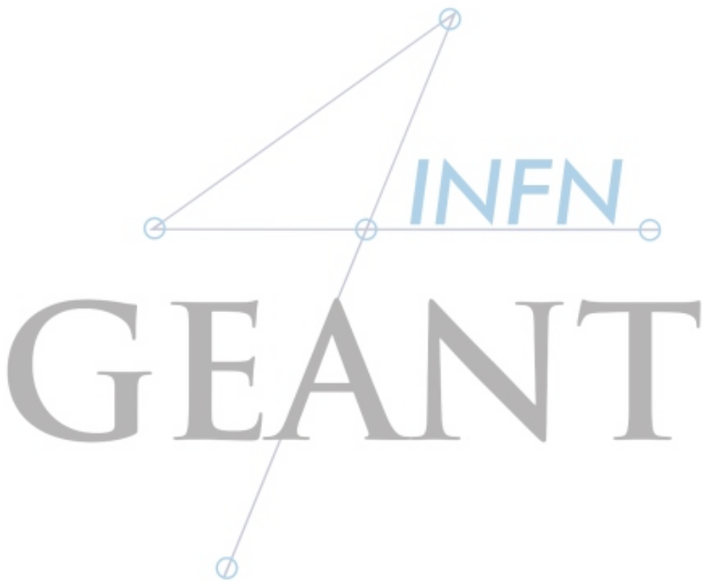
G4VProcess class

- Physics processes describe HOW particles interact with material
- Are derived from G4VProcess base class
- Abstract class defining the common interface of all processes in Geant4, used by all physics processes

Three kinds of "actions":

- AtRest actions
  - Decays, e^+ annihilation
- AlongStep actions
  - To describe continuous (inter)actions, occurring along the path of the particle, i.e. **"soft" interactions**
- PostStep actions
  - To describe the point-like (inter)actions, like decay in flight, hadronic interactions, i.e. **"hard" interactions**





Tracking and Cuts

Tracking: basic concepts

- G4Track keeps current information of the particle and has static information

- G4Track keeps information at the beginning of the step. After finishing all AlongStepDoIts, G4Track is updated. It is updated after each invocation of a PostStepDoIt.

- All Geant4 processes, including the transportation of particles, are treated generically. In spite of the name "*tracking*", particles are not *transported* in the tracking category.

Tracking Verbosity

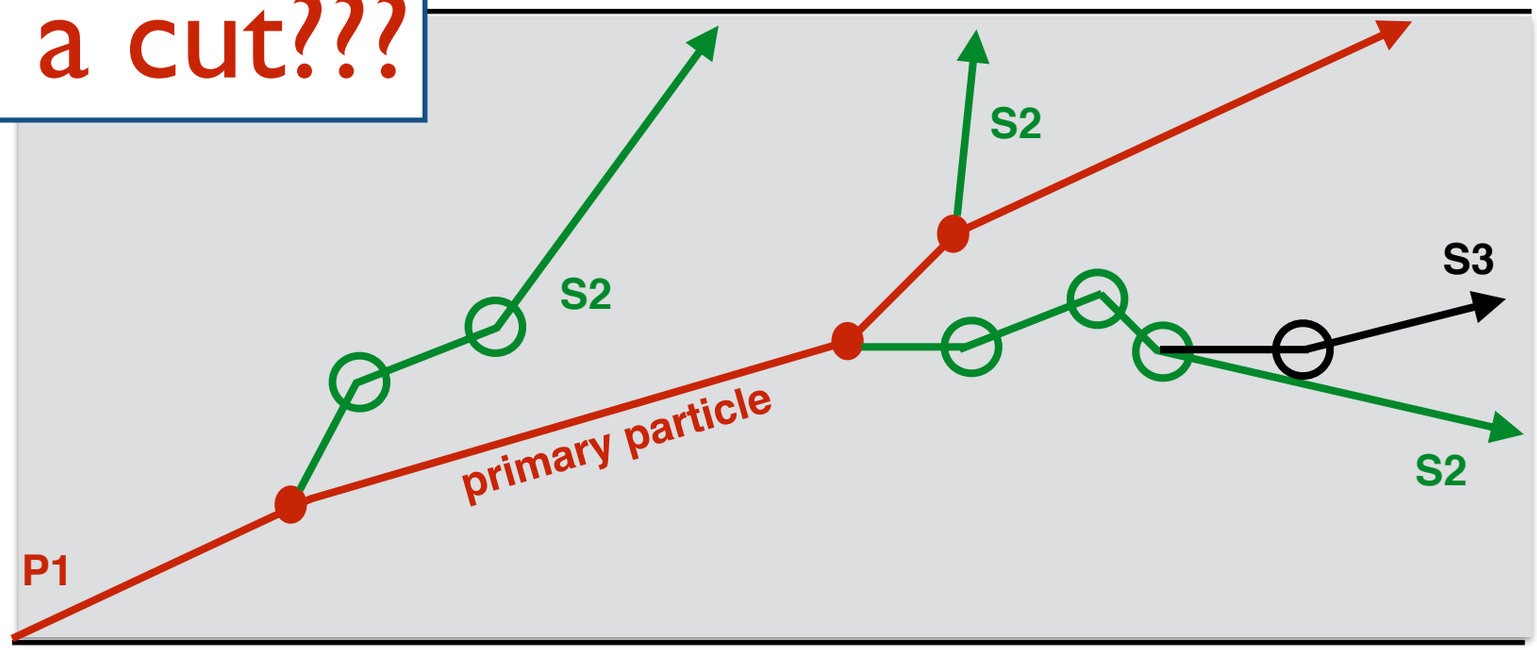
UI command: **`/tracking/verbose 1`**

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

Step#     X(mm)     Y(mm)     Z(mm) KinE(MeV)   dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0      47.4       -53      -150          6         0        0         0    Envelope initStep
    1      47.4       -53       -58      0.844         0       92        92    Envelope compt
    2       -46      15.9      5.55       0.47         0      132       224    Envelope compt
    3      -100      6.37     -3.62       0.47         0     55.6       280       World
Transportation
    4      -120      2.84     -7.02       0.47         0     20.6       301  OutOfWorld
Transportation
**************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 3,   Parent ID = 1
**************************************************************************
Step#     X(mm)     Y(mm)     Z(mm) KinE(MeV)   dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0       -46      15.9      5.55      0.375         0        0         0    Envelope initStep
    1     -46.1      16.4      5.98     0.0482     0.327     1.16      1.16    Envelope eIoni
    2     -46.1      16.3      5.98          0    0.0482   0.0408       1.2    Envelope eIoni
```

Production cuts: basic concepts

What is a cut???



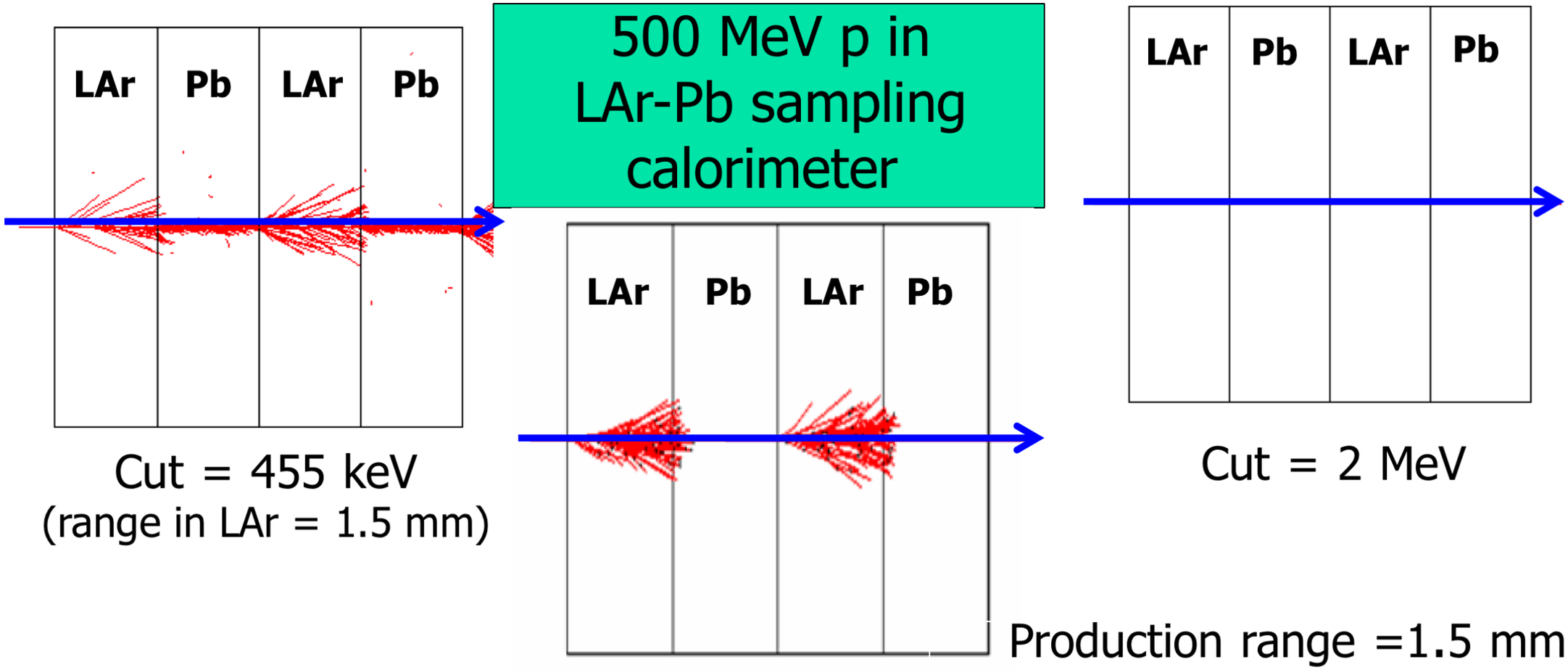
You can set a **"range" production threshold**

- this threshold is a distance, not an energy
- default = 1 mm
- Particles unable to travel at least the range cut value are not produced

Production threshold is internally converted to the energy threshold W_0, depending on particle type and material

Effective energy threshold is different in each material

Cut in range: an example



Threshold in range: 1.5 mm

455 keV electron energy in liquid Ar

2 MeV electron energy in Pb

Cuts: UI Commands

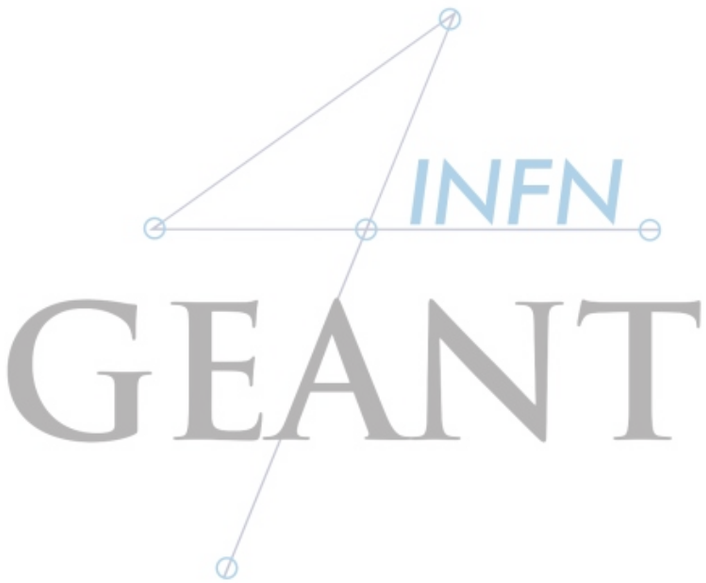

```
# Universal cut (whole world, all particles)
/run/setCut 10 mm

# Override low-energy limit
/cuts/setLowEdge 100 eV
```

```
# Set cut for a specific particle (whole world)
/run/setCutForAGivenParticle gamma 0.1 mm

# Set cut for a region (all particles)
/run/setCutForARegion myRegion 0.01 mm

# Print a summary of particles/regions/cuts
/run/dumpCouples
```



Physics in Geant4

Physics - the challenge

- Huge amount of different processes for various purposes (only a handful relevant)
- Competing descriptions of the same physics phenomena (necessary to choose)
  - fundamentally different **approaches**
  - balance between **speed** and **precision**
  - different **parameterizations**
- Hypothetical processes & exotic physics

Solution: Atomistic approach with modular **physics lists**

Physics models: basic concepts

There are currently 28 "packaged" physics lists available

- **but you will likely interested in only a few, namely the "reference physics lists"**

- **many physics lists are either developmental or cutomized in some way, and so not very useful to new users**

Reference physics lists

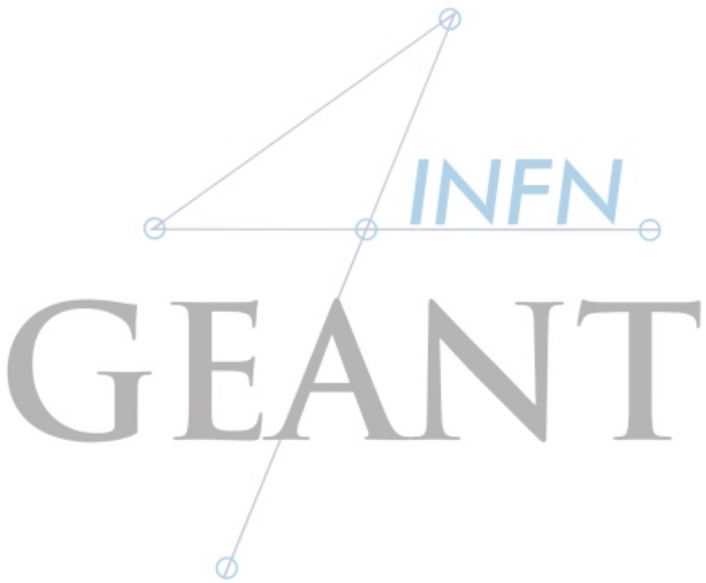
- QGSP_BERT, QGSP_BERT_EMV, QGSP_BERT_HP, QGSP_BIC, FTFP_BERT, LBE, LHEP
- plus a few more

Conventional Physics List

- Two families of builders for the high-energy part
- **QGS**, or list based on a model that use the Quark Gluon String model for high energy hadronic interactions of protons, neutrons, pions and kaons
- **FTF**, based on the FTF (FRITIOF like string model) for protons, neutrons, pions and kaons
- Three families for the cascade energy range
- **BIC**, binary cascade
- **BERT**, Bertini cascade
- **INCLXX**, Liege Intranuclear cascade model

- New Model: ParticleHP

Data based on TENDL-2014 (charged particles) and ENDFVII.r1 (neutrons).



Physics lists

Physics List

- One instance per application
  - registered to run manager in main()
  - inheriting from G4VUserPhysicsList
- Responsibilities
  - all particle types (electron, proton, gamma, ...)
  - all processes (photoeffect, bremsstrahlung, ...)
  - all process parameters (...)
  - production cuts (e.g. 1 mm for electrons, ...)

Physics List

3 ways to get a physics list

1) Manual: Specify all particles & processes that may occur in the simulation. (difficult)

2) Physics constructors: Combine your physics from pre-defined sets of particles and processes. Still you define your own class – modular physics list (easier)

3) Reference physics lists: Take one of the pre-defined physics lists. You don't create any class (easy)

Physics List

Way 1

G4VUserPhysicsList class

Implement 3 methods:

```
class MyPhysicsList : public G4VUserPhysicsList {
public:
    // ...
    void ConstructParticle();  // pure virtual
    void ConstructProcess();   // pure virtual
    void SetCuts();
    // ...
}
```

Advantage: most flexible

Disadvantages:

- most verbose
- most difficult to get right

Physics List

G4VUserPhysicsList: implementation

ConstructParticle():

– choose the particles you need in your simulation, define all of them here

ConstructProcess() :

– for each particle, assign all the physics processes relevant to your simulation

SetCuts() :

– set the range cuts for secondary production for processes with infrared divergence

Physics List

Way 2

G4VModularPhysicsList

- Similar structure as `G4VUserPhysicsList` (same methods to override – though not necessary):

```
class MyPhysicsList : public G4VModularPhysicsList {
public:
    MyPhysicsList();            // define physics constructors
    void ConstructParticle();   // optional
    void ConstructProcess();    // optional
    void SetCuts();             // optional
}
```

Differences to “manual” way:

- Particles and processes typically handled by **physics constructors** (still customizable)
- **Transportation** automatically included

Physics List

Physics constructor

= “**module**” of the **modular physics list**

- Inherits from **`G4VPhysicsConstructor`**
- Defines **ConstructParticle**() and **ConstructProcess**()
  - to be fully imported in modular list (behaving in the same way)
- **GetPhysicsType**()
  - enables switching physics of the same type, if possible (see next slide)

Physics List

Physics constructors

- Huge set of pre-defined ones
  - **EM**: Standard, Livermore, Penelope
  - **Hadronic inelastic:** QGSP_BIC, FTFP_Bert, ...
  - **Hadronic elastic:** G4HadronElasticPhysics, ...
  - ... (decay, optical physics, EM extras, ...)
- You can implement your own (of course) by inheriting from the G4VPhysicsConstructor class

Physics List

Replace physics constructors

You can **add** or **remove** the physics constructors after the list instance is created:

- e.g. in response to **UI command**
- only **before initialization**
- physics of the same type can be **replaced**

```
void MyModularList::SelectAlternativePhysics() {
    AddPhysics(new G4OpticalPhysics);
    RemovePhysics(fDecayPhysics);
    ReplacePhysics(new G4EmLivermorePhysics);
}
```

Physics List

SetCuts()

- Define all **production** cuts for gamma, electrons and positrons
  - Recently also for protons
- Notice: this is a **production cut**, not a tracking cut

Physics List

Way 3

Reference physics lists

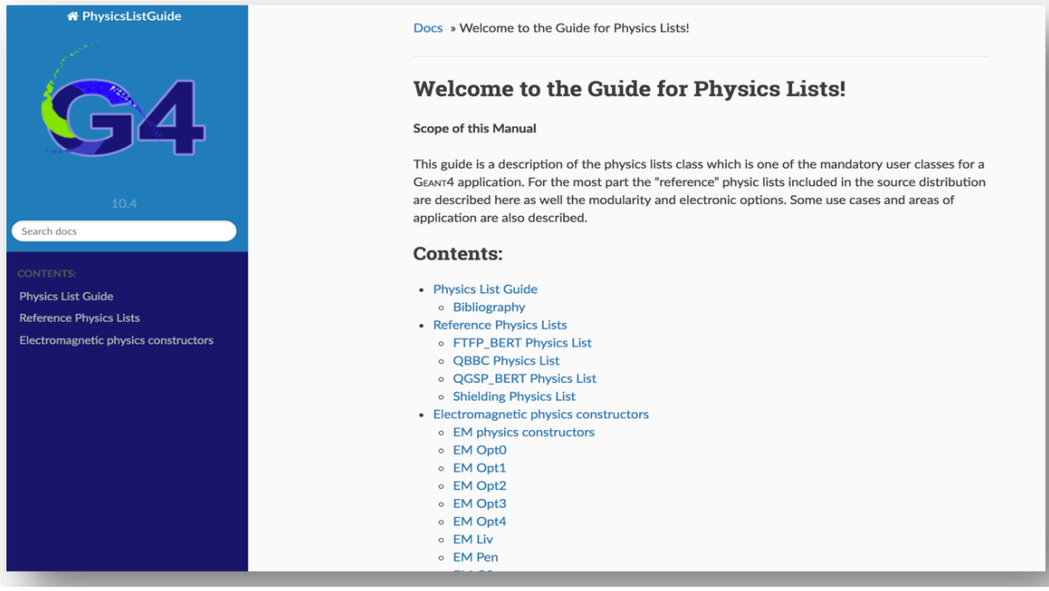
- Pre-defined physics lists
  - already containing a complete set of particles & processes (that work together)
  - targeted at specific area of interest (HEP, medical physics, ...)
  - constructed as **modular physics lists**, built on top of **physics constructors**
  - customizable (by calling appropriate methods before initialization)

Physics List

Lists of reference physics lists

Source code: `$G4INSTALL/source/physics_lists/lists`

```
FTF_BIC.hh                    G4PhysListRegistry.hh         QGSP_BIC_AllHP.hh
FTFP_BERT.hh                  G4PhysListStamper.hh          QGSP_BIC.hh
FTFP_BERT_HP.hh               INCLXXPhysicsListHelper.hh    QGSP_BIC_HP.hh
FTFP_BERT_TRV.hh              LBE.hh                        QGSP_FTFP_BERT.hh
FTFP_INCLXX.hh                NuBeam.hh                     QGSP_INCLXX.hh
FTFP_INCLXX_HP.hh             QBBC.hh                       QGSP_INCLXX_HP.hh
G4GenericPhysicsList.hh       QGS_BIC.hh                    Shielding.hh
G4PhysListFactoryAlt.hh       QGSP_BERT.hh
G4PhysListFactory.hh          QGSP_BERT_HP.hh
```



====> Task 3

link to the task: http://geant4.lngs.infn.it/alghero2018/task3/index.html

Task 3 - Physics and Physics Lists

- ❑ Processes and particles
- ❑ Physics constructors
- ❑ Physics lists
- ❑ Production and cuts