Physics I.: Physics Lists

Mihaly Novak (CERN, EP-SFT)

Geant4 Tutorial at Lund University, Lund (Sweden), 3-7 September 2018

Geant4.10.4

Slides based on Dennis Wright (SLAC) & Vladimir Ivantchenko (CERN) lectures

OUTLINE

- **Introduction**
  - What is a Physics List? Why do we need it?
- **The Geant4 Physics List interface**
  - `G4VUserPhysicsList`
- **Modular Physics List**
  - A more convenient way to go…
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
- **Examples**

INTRODUCTION

- **Introduction**
  - What is a Physics List? Why do we need it?
- **The Geant4 Physics List interface**
  - `G4VUserPhysicsList`
- **Modular Physics List**
  - A more convenient way to go…
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
- **Examples**

What is a Physics List?

- **Physics List is an object that is responsible to:**
  - specify all the particles that will be used in the simulation application
  - together with the list of physics processes assigned to each individual particles
- **One out of the 3 mandatory objects that the user needs to provide to the `G4RunManager` in case of all Geant4 applications:**
  - it provides the information to the run-manager when, how and what set of physics needs to be invoked
- **Provides a very flexible way to set up the physics environment:**
  - the user can chose and specify the particles that they want to be used
  - the user can chose the physics (processes) to assign to each particle
- **BUT, the user must have a good understanding of the physics required to describe properly the given problem:**
  - omission of relevant particles and/or physics interactions could lead to poor modelling results !!!

Why do we need a Physics List?

- **Physics is physics - shouldn't Geant4 provide, as default, a complete set of physics that everyone can use?**
- **NO:**
  - there are many different approximations and models to describe the same interaction:
    - very much the case for hadronic but also true for electromagnetic physics
  - computation time is an issue:
    - some users may want a less accurate but significantly faster model for a given interaction while others need the most accurate description
  - there is no any simulation application that would require all the particles, all their possible interactions that Geant4 can provide:
    - e.g. most of the medical applications are not interested in multi-GeV physics
- **For this reason, Geant4 provides an *atomistic*, rather than an integral approach to physics:**
  - provides many independent (for the most part) physics components i.e. physics ***processes***
  - users select these components in their custom-designed physics lists
  - exceptions: few electromagnetic processes must be used together; `G4Transportation` process must be assigned to all stable particles

Physics processes provided by Geant4?

- **EM physics:**
  - the “standard” i.e. default processes are valid between ~keV to PeV
  - the “low energy” processes can be used from ~100 eV to PeV
  - Geant4-DNA: valid down to ~eV (only for liquid water)
  - optical photons
- **Weak interaction physics:**
  - decay of subatomic particles
  - radioactive decay of nuclei
- **Hadronic physics:**
  - pure strong interaction physics valid from 0 to ~TeV
  - electro- and gamma-nuclear interactions valid from 10 MeV to ~TeV
  - high-precision neutron package valid from thermal energies to ~20 MeV
- **Parameterized or “fast-simulation” physics**

THE GEANT4 PHYSICS LIST INTERFACE

- Introduction
  - What is a Physics List? Why do we need it?
- **The Geant4 Physics List interface**
  - `G4VUserPhysicsList`
- Modular Physics List
  - A more convenient way to go…
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
- **Examples**

Physics List interface:

- `G4VUserPhysicsList` **is the Geant4 physics list interface**
- **All physics lists must be derived from this base class:**

```
4   class YourPhysicsList: public G4VUserPhysicsList {
5     public:
6       // CTR
7       YourPhysicsList();
8       // DTR
9       virtual ~YourPhysicsList();
10
11      // pure virtual => needs to be implemented
12      virtual void ConstructParticle();
13      // pure virtual => needs to be implemented
14      virtual void ConstructProcess();
15
16      // virtual method
17      virtual void SetCuts();
18      ...
19      ...
20  };
```

  - user must implement the 2 pure virtual methods: `ConstructParticle()` and `ConstructProcess()`
  - user can implement the `SetCuts()` method (optional)

Physics List interface: `ConstructParticle()`

- **Interface method:** to define the list of particles to be used in the simulation
- **Construct particles individually:**

```
23  void YourPhysicsList::ConstructParticle() {
24      G4Electron::Definition();
25      G4Gamma::Definition();
26      G4Proton::Definition();
27      G4Neutron::Definition();
28      // other particle definitions
29      …
30      …
31  }
```

- **Construct particles by using helpers:**

```
35  void YourPhysicsList::ConstructParticle() {
36      // construct baryons
37      G4BaryonConstructor baryonConstructor;
38      baryonConstructor.ConstructParticle();
39      // construct bosons
40      G4BosonConstructor bosonConstructor;
41      bosonConstructor.ConstructParticle();
42      // more particle definitions
43      …
44      …
45  }
```

Physics List interface: `ConstructProcess()`

- **What Process is?**
  - an object that defines the way in which a given particle interacts with matter through a given type of interaction (e.g. *electron ionisation process*)
  - more on this later
- **Interface method:** to define the list of physics processes to be used in the simulation for a given particle (constructed above)

```
48  void YourPhysicsList::ConstructProcess() {
49      // method (provided by the G4VUserPhysicsList base class)
50      // that assigns transportation process to all particles
51      // defined in ConstructParticle()
52      AddTransportation();
53      // helper method might be defined by the user (for convenience)
54      // to add electromagnetic physics processes
55      ConstructEM();
56      // helper method might be defined by the user
57      // to add all other physics processes
58      ConstructGeneral();
59  }
```

Physics List interface: ConstructProcess()

```
62  void YourPhysicsList::ConstructEM() {
63    // get the physics list helper
64    // it will be used to assign processes to particles
65    G4PhysicsListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
66    auto particleIterator   = GetParticleIterator();
67    particleIterator->reset();
68    // iterate over the list of particles constructed in ConstructParticle()
69    while( (*particleIterator)() ) {
70      // get the current particle definition
71      G4ParticleDefinition* particleDef  = particleIterator->value();
72      // if the current particle is the appropriate one => add EM processes
73      if ( particleDef == G4Gamma::Definition() ) {
74        // add physics processes to gamma particle here
75        ph->RegisterProcess(new G4GammaConversion(),  particleDef);
76        ...
77        ...
78      } else if ( particleDef == G4Electron::Definition() ) {
79        // add physics processes to electron here
80        ph->RegisterProcess(new G4eBremsstrahlung(), particleDef);
81        ...
82        ...
83      } else if (...) {
84        // do the same for all other particles like e+, mu+, mu-, etc.
85        ...
86      }
87    }
88  }
```

Physics List interface: ConstructProcess()

```
 93  void YourPhysicsList::ConstructGeneral() {
 94    // get the physics list helper
 95    // it will be used to assign processes to particles
 96    G4PhysicsListHelper* ph  = G4PhysicsListHelper::GetPhysicsListHelper();
 97    auto particleIterator    = GetParticleIterator();
 98    particleIterator->reset();
 99    // create processes that need to be assigned to particles
100    // e.g. create decay process
101    G4Decay* theDecayProcess = new G4Decay();
102    ...
103    ...
104    // iterate over the list of particles constructed in ConstructParticle()
105    while( (*particleIterator)() ) {
106      // get the current particle definition
107      G4ParticleDefinition* particleDef  = particleIterator->value();
108      // if the process can be assigned to the current particle => do it!
109      if ( theDecayProcess->IsApplicable( *particleDef ) ) {
110        // add the physics processes to the particle
111        ph->RegisterProcess(theDecayProcess, particleDef);
112      }
113      // other processes might be assigned to the current particle as well
114      ...
115      ...
116    }
117  }
```

Physics List interface: SetCuts()

- **Interface method (optional):** to set secondary production threshold values

```
119  // optional: default cut value = 1.0 mm
120  void YourPhysicsList::SetCuts() {
121    // set the base (G4VUserPhysicsList) class member value
122    // to the required one
123    defaultCutValue = 0.7*CLHEP::mm;
124    // then set each production threshold individually
125    // NOTE: order is important! First "gamma" then the others.
126    SetCutValue(defaultCutValue, "gamma");
127    SetCutValue(defaultCutValue, "e-");
128    SetCutValue(defaultCutValue, "e+");
129    SetCutValue(defaultCutValue, "proton");
130    //
131    // These are all the production cuts:
132    // - not required for any other particle
133  }
```

- **Or in a simpler and perfectly equivalent way:**

```
135  // optional: default cut value = 1.0 mm
136  void YourPhysicsList::SetCuts() {
137    G4double yourCutValue = 0.7*CLHEP::mm;
138    // use the base (G4VUserPhysicsList) class method
139    SetDefaultCutValue( yourCutValue );
140  }
```

MODULAR PHYSICS LIST

- **Introduction**
  - What is a Physics List? Why do we need it?
- **The Geant4 Physics List interface**
  - `G4VUserPhysicsList`
- **Modular Physics List**
  - A more convenient way to go…
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
- **Examples**

Modular physics list:

- **Why?**
  - our previous physics list example was very simple and very incomplete
  - a realistic physics list will have much more particles and processes
  - such a list can be quite long, complicated and hard to maintain
- **Modular physics list provides a solution:**
  - the interface is defined in `G4VModularPhysicsList`
  - this interface is derived from the `G4VUserPhysicsList` interface (as `YourPhysicsList` in the previous example)
  - transportation is automatically added to all constructed particles
  - allows to use “physics modules”
  - a given physics module handles a well defined category of physics (e.g. EM physics, hadronic physics, decay, etc.)

Modular physics list:

```
145  class YourModularPhysicsList : public G4VModularPhysicsList {
146    public:
147      // CTR
148      YourModularPhysicsList();
149      ...
150  };
151
152  // CTR implementation
153  YourModularPhysicsList::YourModularPhysicsList()
154  : G4VModularPhysicsList() {
155    // set default cut value (optional)
156    defaultCutValue = 0.7*CLHEP::mm;
157    // use pre-defined physics constructors
158    // e.g. register standard EM physics using the pre-defined constructor
159    // (includes constructions of all EM processes as well as the
160    // corresponding particles)
161    RegisterPhysics( new G4EmStandardPhysics() );
162    // user might create their own constructor and register it
163    // e.g. all physics processes having to do with protons (see below)
164    RegisterPhysics( new YourProtonPhysics()   );
165    // add more constructors to complete the physics
166    ...
167  }
```

Modular physics list: physics constructors

- **Physics constructor:**
  - allows to group particle and their processes construction according to physics domain
  - implements the `G4VPhysicsConstructor` interface
  - kind of sub-set of a complete physics list
  - user might create their own (e.g. `YourProtonPhysics`) or use pre-defined physics constructors (e.g. `G4EmStandardPhysics`, `G4DecayPhysics`, etc.)

```
169   class YourProtonPhysics : public G4VPhysicsConstructor {
170     public:
171       // CTR
172       YourProtonPhysics(const G4String& name = "proton-physics");
173       // DTR
174       virtual ~YourProtonPhysics();
175       // particle construction:
176       // only one particle i.e. proton needs to be constructed
177       virtual ConstructParticle();
178       // process construction:
179       // create and assign all processes to proton that it can have
180       virtual ConstructProcess();
181   };
```

Modular physics list: constructors

- **Some "standard" EM physics constructors:**
  - `G4EmStandardPhysics` - default
  - `G4EmStandardPhysics_option1` - for HEP, fast but not precise settings
  - `G4EmStandardPhysics_option2` - for HEP, experimental
  - `G4EmStandardPhysics_option3` - for medical and space science applications
  - `G4EmStandardPhysics_option4` - most accurate EM models and settings
- **Some "low energy" EM physics constructors:**
  - `G4EmLivermorePhysics`
  - `G4EmLivermorePolarizedPhysics`
  - `G4EmPenelopePhysics`
  - `G4EmDNAPhysics`
- **The complete list can be found in your toolkit:**

  `geant4/source/physics_lists/constructors/` ==> all built in CTR

  `geant4/source/physics_lists/constructors/electromagnetic`

  `geant4/source/physics_lists/constructors/hadron_elastic`

  `geant4/source/physics_lists/constructors/hadron_inelastic`
- **More information at:**

  `geant4/source/physics_lists/constructors/xxx/README`

  http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

PRE-PACKAGED PHYSICS LISTS

- Introduction
  - What is a Physics List? Why do we need it?
- The Geant4 Physics List interface
  - `G4VUserPhysicsList`
- Modular Physics List
  - A more convenient way to go…
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
- Examples

Packaged Physics Lists:

- **Our examples dealt mainly with electromagnetic physics**
- **A realistic physics list can be found in basic example B3:**
  - a modular physics list that includes “standard” EM physics and decay physics by using built in physics constructors
  - serves as a good starting point to construct your own physics list
  - add any other physics according to your needs
- **Adding hadronic physics is more involved:**
  - for any hadronic process, the user might chose from several “models”
  - choosing the most appropriate model for a given application requires significant experience
- **Pre-packaged physics lists:**
  - in order to help the users, the toolkit provides pre-packaged physics lists according to some reference use cases (see later)
  - these are “ready-to-use”, complete physics lists provided by the toolkit and constructed by the expert developers
  - each pre-packaged physics list includes different combinations of EM and hadronic physics
  - the list of these pre-packaged physics lists can be found in the toolkit at: `geant4/source/physics_lists/lists/include`

Packaged Physics Lists:

- **Caveats:**
  - these lists are provided as a “best guess” of the physics needed in some given use cases
  - when a user decide to use them, the user is responsible for “validating” the physics for that given application and adding (or removing) the appropriate physics
  - they are intended to give a starting point or template
- **“Production physics lists”:**
  - these physics lists are used by large user groups like ATLAS, CMS, etc.
  - because of their importance, they are well-maintained and tested physics lists
  - they are changed, updated less frequently: very stable physics lists
  - they are extensively validated by the developers and the user communities
  - FTFP_BERT, QGSP_BERT, QGSP_FTFP_BERT_EMV, FTFP_BERT_HP, QGSP_BIC_EMY, QGSP_BIC_HP, QBBC, Shielding

Packaged Physics Lists: naming convention

- **Some Hadronic options:**
  - "QGS" Quark Gluon String model (> ~15 GeV)
  - "FTF" FRITIOF String model (> ~5 GeV)
  - "BIC" Binary Cascade model (< ~10 GeV)
  - "BERT" Bertini Cascade model (< ~10 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "HP" High Precision neutron model (< ~20 MeV)
- **Some EM options:**
  - No suffix: standard EM i.e. the default `G4EmStandardPhysics` constructor
  - "EMV" `G4EmStandardPhysics_option1` CTR: HEP, fast but less precise
  - "EMY" `G4EmStandardPhysics_option3` CTR: medical, space sci., precise
  - "EMZ" `G4EmStandardPhysics_option4` CTR: most precise EM physics
- **Name decoding:** String(s)_Cascade_Neutron_EM
- **The complete list of pre-packaged physics list with detailed description can be found in the documentation ("*Guide for Physics Lists*"):**
  - **http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html**

Packaged Physics Lists: naming convention (example)

- **FTFP_BERT:**
  - Recommended by Geant4 developers for HEP applications
  - Includes the standard EM physics i.e. `G4EmStandardPhysics` CTR
  - "FTF" FRITIOF string model (> 4 GeV)
  - "BERT" Bertini Cascade model (< 5 GeV)
  - "P" `G4Precompound` model used for de-excitation

- **QGSP_BIC_HP(_EMZ):**
  - Recommended for medical applications (experimental QGSP_BIC_AllHP)
  - "QGS" Quark Gluon String model (> 12 GeV)
  - "FTF" FRITIOF String model (9.5 - 25 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "BIC" Binary Cascade model (200 MeV - 9.9 GeV)
  - "HP" High Precision neutron model (< ~20 MeV)
  - "EMZ" `G4EmStandardPhysics_option4` CTR (or EMY that's a bit less precise)

EXAMPLES

- Introduction
  - What is a Physics List? Why do we need it?
- The Geant4 Physics List interface
  - `G4VUserPhysicsList`
- Modular Physics List
  - A more convenient way to go…
- Pre-packaged Physics Lists
  - Provided by the toolkit.
- **Examples**

Examples: using physics constructors

- **QGSP_BIC_HP_EMZ:**
  - the QGSP_BIC_HP_EMZ list (used in the previous example) doesn't exist!
  - however, constructors for both the hadronic and the EM parts are available:
    - `G4HadronPhysicsQGSP_BIC_HP` : for the hadron inelastic part i.e. QGSP_BIC_HP
    - `G4EmStandardPhysics_option4`: for the EM part i.e. for EMZ

```
187  class YourQGSP_BIC_HP_EMZ : public G4VModularPhysicsList {
188    public:
189      // CTR
190      YourQGSP_BIC_HP_EMZ();
191      ...
192  };
193
194
195  // CTR implementation
196  YourQGSP_BIC_HP_EMZ::YourQGSP_BIC_HP_EMZ()
197  : G4VModularPhysicsList() {
198    // set default cut value (optional)
199    defaultCutValue = 0.7*CLHEP::mm;
200    // use pre-defined physics constructor for EM: EM-opt4
201    RegisterPhysics( new G4EmStandardPhysics_option4() );
202    // use pre-defined physics constructor for hadron inelastic: QGSP_BIC_HP
203    RegisterPhysics( new G4HadronPhysicsQGSP_BIC_HP()  );
204    // ADD MORE CONSTRUCTORS TO COMPLETE THE PHYSICS WITH:
205    // Hadron Elastic, Decay, Stopping, Ion, etc. Physics !!!!
206    ...
207  }
```

Examples: using reference physics lists (easier!!!)

- **QGSP_BIC_HP_EMZ:**
  - a QGSP_BIC_HP reference physics list, including all the above mentioned CTRs is available (but with the standard EM physics)
  - the `G4PhysListFactory` knows everything about the available reference lists
  - moreover, it makes possible to replace their EM option with a new one

```
212   // IM YOUR MAIN APPLICATION
213   //
214     // create your run manager
215   #ifdef G4MULTITHREADED
216     G4MTRunManager* runManager = new G4MTRunManager;
217     // number of threads can be defined via macro command
218     runManager->SetNumberOfThreads(4);
219   #else
220     G4RunManager* runManager = new G4RunManager;
221   #endif
222     //
223     // create a physics list factory object that knows
224     // everything about the available reference physics lists
225     // and can replace their default EM option
226     G4PhysListFactory physListFactory;
227     // obtain the QGSP_BIC_HP_EMZ reference physics lists
228     // which is the QGSP_BIC_HP refrence list with opt4 EM
229     const G4String plName = "QGSP_BIC_HP_EMZ";
230     G4VModularPhysicsList* pList = physListFactory.GetReferencePhysList(plName);
231     // (check that pList is not nullptr, that I skipp now)
232     // register your physics list in the run manager
233     runManager->SetUserInitialization(pList);
234     // register further mandatory objects i.e. Detector and Primary-generator
235     ...
```

SUMMARY

- **All particles, physics processes and production cuts, needed for the simulation application, must be defined and given in a physics list**
- **Two kinds of physics list interfaces are available for the users:**
  - `G4VUserPhysicsList` - for relatively simple physics environment
  - `G4VModularPhysicsList` - for more complex physics environment
- **Some reference physics lists are provided by the Geant4 developers that can be used as starting points:**
  - pure EM physics constructors
  - complete hadronic, EM and other extra physics
- **Choosing the appropriate physics for a given application must be done by special care**