Version 10.5

Physics-1

Vladimir Ivantchenko (CERN & Tomsk State University, Russia)

Geant4 Beginners Course

Outline

- Introduction
- G4VUserPhysicsList class
- G4VModularPhysicsList
- Geant4 physics lists sub-library
- Reference physics list
- Choosing appropriate physics list
- Extending of physics list
- Use of generic physics list

Introduction

- 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 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

Why do we need a Physics List?

- there are many different approximations and models to describe the same interaction:
  - very much the case both for hadronic and also 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 can select needed components in their custom-designed physics lists

Physics List interface

- G4VUserPhysicsList is the Geant4 physics list interface
  - All physics lists are derived from this base class:

```
class YourPhysicsList: public G4VUserPhysicsList {
  public:
    // CTR
    YourPhysicsList();
    // DTR
    virtual ~YourPhysicsList();

    // pure virtual => needs to be implemented
    virtual void ConstructParticle();
    // pure virtual => needs to be implemented
    virtual void ConstructProcess();

    // virtual method
    virtual void SetCuts();
    ...
    ...
};
```

- user must implement the 2 pure virtual methods: ConstructParticle() and ConstructProcess()
  - There methods are called in each thread during run initialisation
- user can implement the SetCuts() method
  - optional, not recommended to be custom for recent releases of Geant4

Construction of particles

- Particles may be constructed individually:

```
void YourPhysicsList::ConstructParticle() {
    G4Electron::Definition();
    G4Gamma::Definition();
    G4Proton::Definition();
    G4Neutron::Definition();
    // other particle definitions
    …
    …
}
```

- Particles may be constructed by using helpers:

```
void YourPhysicsList::ConstructParticle() {
    // construct baryons
    G4BaryonConstructor baryonConstructor;
    baryonConstructor.ConstructParticle();
    // construct bosons
    G4BosonConstructor bosonConstructor;
    bosonConstructor.ConstructParticle();
    // more particle definitions
    …
    …
}
```

Construction of processes

- This method is too complicate and very often is implemented via calls to logically different methods

```
void YourPhysicsList::ConstructProcess() {
    // method (provided by the G4VUserPhysicsList base class)
    // that assigns transportation process to all particles
    // defined in ConstructParticle()
    AddTransportation();
    // helper method might be defined by the user (for convenience)
    // to add electromagnetic physics processes
    ConstructEM();
    // helper method might be defined by the user
    // to add all other physics processes
    ConstructGeneral();
}
```

Electromagnetic physics

```cpp
void YourPhysicsList::ConstructEM() {
  // get the physics list helper
  // it will be used to assign processes to particles
  G4PhysicsListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
  auto particleIterator   = GetParticleIterator();
  particleIterator->reset();
  // iterate over the list of particles constructed in ConstructParticle()
  while( (*particleIterator)() ) {
    // get the current particle definition
    G4ParticleDefinition* particleDef  = particleIterator->value();
    // if the current particle is the appropriate one => add EM processes
    if ( particleDef == G4Gamma::Definition() ) {
      // add physics processes to gamma particle here
      ph->RegisterProcess(new G4GammaConversion(),  particleDef);
      ...
      ...
    } else if ( particleDef == G4Electron::Definition() ) {
      // add physics processes to electron here
      ph->RegisterProcess(new G4eBremsstrahlung(), particleDef);
      ...
      ...
    } else if (...) {
      // do the same for all other particles like e+, mu+, mu-, etc.
      ...
    }
  }
}
```

Decay physics

```
void YourPhysicsList::ConstructGeneral() {
  // get the physics list helper
  // it will be used to assign processes to particles
  G4PhysicsListHelper* ph  = G4PhysicsListHelper::GetPhysicsListHelper();
  auto particleIterator    = GetParticleIterator();
  particleIterator->reset();
  // create processes that need to be assigned to particles
  // e.g. create decay process
  G4Decay* theDecayProcess = new G4Decay();
  ...
  ...
  // iterate over the list of particles constructed in ConstructParticle()
  while( (*particleIterator)() ) {
    // get the current particle definition
    G4ParticleDefinition* particleDef  = particleIterator->value();
    // if the process can be assigned to the current particle => do it!
    if ( theDecayProcess->IsApplicable( *particleDef ) ) {
      // add the physics processes to the particle
      ph->RegisterProcess(theDecayProcess, particleDef);
    }
    // other processes might be assigned to the current particle as well
    ...
    ...
  }
}
```

Adding a process in physics list

- Get the process manager of the particle:

```
G4PhysicsListHelper* helper = G4PhysicsListHelper::GetPhysicsListHelper();
G4ParticleDefinition* electron = G4Electron::Electron();
```

- The best way to add the process:

```
helper->RegisterProcess(new G4eIonisation, electron);
```

- There is well defined order of processes
- G4PhysicsListhelper is responsible for the correct odering for processes from Geant4 sub-libraries
  - Custom user process should have ordering explicitly defined by user

About process ordering

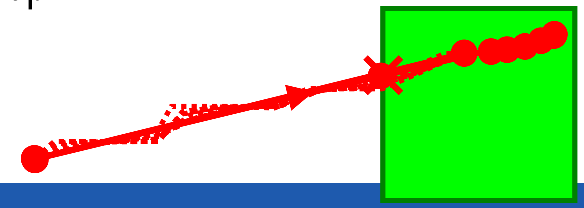
- The most strong rule for multiple-scattering, transportation, and G4Scintillation
- In your physics list, you should always have, for the ordering of the **AlongGetPhysicalInteractionLength(…)** methods:
  - Transportation last
    - For all particles
  - Multiple scattering second last
    - For charged particles only
      - assuming n processes

        [n-2] …

        [n-1] multiple scattering

        [n] transportation

- Why ?
  - Processes return a « true path length »;
  - The multiple scattering folds up this length into a ***shorter*** « geometrical » path length;
  - Based on this new length, the transportation can geometrically limits the step.



Extensions of the physics list interface

- Initial idea at time, when Geant4 was designed, was to give users full flexibility to prepare physics list
- Very soon we understood that it is unpractical
  - Too many users
  - Not easy communication between users and developers
  - Difficult to compare results obtained with custom physics lists
- A solution was proposed:
  - G4VModularPhysicsList extension
  - G4VPhysicsConstructor interface class
  - Reference physics list approach
  - physics_list sub-library added

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 components from Geant4 physics_list sub-library

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. YourPhysics) or use pre-defined physics constructors (G4EmStandardPhysics, G4DecayPhysics,…)

```
class YourProtonPhysics : public G4VPhysicsConstructor {
  public:
    // CTR
    YourProtonPhysics(const G4String& name = "proton-physics");
    // DTR
    virtual ~YourProtonPhysics();
    // particle construction:
    // only one particle i.e. proton needs to be constructed
    virtual ConstructParticle();
    // process construction:
    // create and assign all processes to proton that it can have
    virtual ConstructProcess();
};
```

Physics constructor interface

- G4VModularPhysicsList has following methods:
  - RegisterPhysics(G4VPhysicsConstructor*);
  - ReplacePhysics(G4VPhysicsConstructor*);
  - During registration and replacement of physics constructor a type of G4VPhysicsConstructor is checked to avoid double definition of physics
- Existing enumerator types of constructors:
  - bUnknown
  - bElectromagnetic
  - bEmExtra
  - bDecay
  - bHadronElastic
  - bHadronInelastic
  - bStopping
  - bIons

Electromagnetic physics constructors

- Some “standard” EM physics constructors
  - G4EmStandardPhysics - default, used by ATLAS
  - G4EmStandardPhysics_option1 - for HEP, fast but not precise for sampling calorimeters, used by CMS
  - G4EmStandardPhysics_option2 - for HEP, fast but not precise for sampling calorimeters, used by LHCb
  - G4EmStandardPhysics_option3 - for medical and space science applications
  - G4EmStandardPhysics_option4 - most accurate EM models and settings
- Many experimental, low-energy and DNA physics
  - G4EmStandardPhysicsSS – used single scattering instead of multiple
- G4EmExtraPhysics
  - gamma, electro-nuclear, G4SynchrotronRadiation, rare EM processes
- G4OpticalPhysics
  - is of Unknown type

Decay physics constructors

- G4DecayPhysics
  - main constructor of decay physics
  - defines standard list of particles
  - Defines all standard decays of “stable” particles
  - Included in all reference physics lists
- G4RadioactiveDecayPhysics
  - Defines radioactive decay of isotopes
  - Enable extra physics needed for radioactive decay
  - Physics is data driven
  - Should be registered by user
  - May slow down HEP simulation

Hadronic physics constructors

- G4HadronElasticPhysics
  - Default set for hadron elastic interactions
  - There are few alternative constructors
- Hadron inelastic physics
  - Many different constructors
    - String models for high energy
    - Cascade models for moderate energy
    - Precise data driven models for low-energy neutron transport may be added
- Ion physics
  - Light ions and G4GenericIon
  - Elastic and inelastic interactions
- Capture processes
  - Neutron capture below 15 MeV
  - μ^-, negatively charged meson and baryons capture at rest

Reference physics list approach

- Since long time Geant4 release a set of pre-packaged physics lists, which are called "Reference" physics lists
  - These physics lists are used for Geant4 validation and testing
  - These lists are recommended for to be used for R&D and validations
  - This allows having consistent set physics between various user groups
- Current Geant4 default is FTFP_BERT physics list
  - The Fritiof string model and the Bertini cascade are main components for the hadronic physics of FTFP_BERT
  - FTFP_BERT_HP is an alternative reference physics list with the addition of the high precision neutron transport
- There are ~20 "Reference" physics lists in Geant4 physics_list sub-library
  - Part of reference physics lists may be considered as "experimental" – they include not well established models

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, EMX, EMY, EMZ, LIV, PEN, WVI, GS, SS – various EM physics
- Name decoding: String_Cascade_Neutron_EM
- The complete list with description see in the web page "Guide for Physics Lists"): http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Choosing appropriate physics list

- There are groups of physics lists oriented to different application domains:
  - HEP experiments: FTFP_BERT, QGSP_FTFP_BERT, FTFP_BERT_ATL,….
  - Space applications: QBBC,….
  - Medical applications: QGSP_BIC,…
  - Radiation protection: Shielding,….
- There is no strong limitation for an application domain to use or not to use a particular physics lists for a given use case
  - We would always suggest to start from the default FTFP_BERT and to choose an alternative if there are special requirement:
    - More accurate models are needed: try FTFP_BERT_EMZ
    - Faster models are needed: try optimizing cut in range first

Extending of physics list

- On top of any physics list extra physics constructors may be registered:
  - G4RadioactiveDecayPhysics
  - G4OpticalPhysics
  - G4StepLimiterPhysics
  - G4ParallelWorldPhysics
  - .....
- UI commands allow to tune parameter definitions
  - Cuts in range
  - Tracking cuts
  - Enable/disable processes/features/options
  - Modify parameters of simulation

Use of generic physics list

- Since several years Geant4 provides helper classes allowing creation of physics lists via name
  - $G4INSTALL/examples/extended/hadronic/Hadr00
  - $G4INSTALL/examples/extended/hadronic/Hadr01
  - G4PhyListFactory helper class
    - Using command line it is possible to select desired physics list without recompilation of the application
- It is also possible to select electromagnetic physics constructor on top of a particular hadronic physics:
  - FTFP_BERT_EMV – use Opt1 EM physics
  - FTFP_BERT_EMX – use Opt2 EM physics
  - FTFP_BERT_EMY – use Opt3 EM physics
  - FTFP_BERT_EMZ – use Opt4 EM physics
  - FTFP_BERT_SS – use single scattering instead of multiple scattering

HANDS-ON TASK M2

Electromagnetic physics configuration

- Start VM and open a terminal window
- Continue with TestEm7
  - cd $G4WORKDIR/TaskM/TestEm7/build
  - ./TestEm7
  - /control/execute vis.mac
  - /run/beamOn 1
  - What processes are used for
    - gamma, e-, e+, proton
  - What are the cut in range?
- Set primary particle proton and run with different range cuts
  - /gun/particle proton
  - /run/setCut 1 km
  - /run/beamOn 10
  - /run/setCut 1 mm
  - /run/beamOn 10
  - /run/setCut 0.001 mm
  - /run/beamOn 10

Start examples/extended/hadronic/Hadr01

- Copy, compile, and build the example into working area
  - cd $G4WORKDIR
  - mkdir TaskM
  - cd TaskM
  - cp $G4INSTALL/share/Geant4-10.5.0/examples/extended/hadronic/Hadr01 ./
  - cd Hadr01
  - mkdir build
  - cd build
  - cmake -DGeant4_DIR=${G4COMP} ../
  - make
- Run in the batch mode
  - ./Hadr01 Hadr01.in FTFP_BERT >& ftfp_bert.log
  - ./Hadr01 Hadr01.in FTFP_BERT_HP >& ftfp_bert_hp.log
  - compare log files
- Study Hadr01.cc – how G4PhysListFactory is used