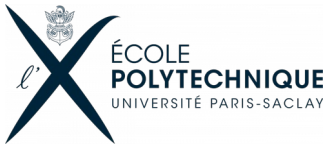
GEANT4
A SIMULATION TOOLKIT

Version 11.1

Physics Lists

Gunter Folger (CERN)

Geant4 Advanced Course



Jefferson Lab
Thomas Jefferson National Accelerator Facility

PRINCETON
UNIVERSITY

Acknowledgement

- **This lecture is by large taken from a lecture of a previous tutorials prepared by originally by Dennis Wright (SLAC), with contributions from Mihaly Novak (CERN, EP-SFT) and Vladimir Ivantchenko (CERN & Tomsk State University, Russia)**

Outline

- **Introduction**
  - What is a Physics List? Why do we need it?
- **The Geant4 Physics List interface**
  - G4VUserPhysicsListPhysics
  - ListsModular Physics List
    - A more convenient way to go...
- **Pre-packaged Physics Lists**
  - Provided by the toolkit.
  - Reference physics lists and naming conventions
  - Extend a pre-packaged physics list
- **How to choose a Physics List**
  - Validation
- **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
  - specify physics processes assigned to each individual particle
- **One out of the 3 mandatory objects the user must provide to the** *G4RunManager* **in 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 choose and specify the particles that they want to be used
  - the user can choose the physics (processes) assigned 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 will lead to poor modelling results !!

Why is a Physics List needed (1)

- **Physics is physics - shouldn't Geant4 provide, as default, a complete set of physics that everyone can use?**
- We believe **NO:**
  - Geant4 is used in different domains with different requirements on simulation
  - We offer different approximations and models to describe the same interaction:
    - very much the case for hadronic but also true for electromagnetic physics
  - No simulation application will require all the particles, all their possible interactions over all the energy range that Geant4 can provide, e.g.:
    - Most of the medical applications are not interested in multi-GeV physics
    - Few applications will require transport of thermal neutrons
  - 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
  - Make it easy to extend physics, like add new particles, processes, or models

Why is a Physics List needed (2)

- **Flexibility**
  - Choose physics best adapted to a given use case
    - Choice of process, model and cross-section per particle
- **Extendability**
  - Allow users to provide new, or more precise, or faster, or … physics
  - Allow the use of new, or more precise, cross section data sets or parameterizations
- **Consistency** with toolkit nature of Geant4

- **For these reasons, Geant4 supports both,**
  - **an atomistic approach**
    - providing many independent (for the most part) physics components, i.e. physics processes or models
  - **the equivalent of an integral approach to physics,**
    - Keeping both flexibility and extendability
    - users may select a set of components
      - In selecting a pre-defined physics list
        - With the possibility to add physics not included
      - in their custom-designed physics lists, eventually using pre-defined constructors or ‘building blocks’ from Geant4

Why is a Physics List needed (3)

- Users need help **choosing** or **constructing a physics list**
  - Which physics approximation/model is required or best for a given use case
    - Requires expert knowledge or experience from experts for a given domain
  - Offering packaged physics lists for several different domains, developed and tested by domain experts
  - Offering physics constructors to cover extra needs for packaged physics lists

How to Create a Physics List

Three options to create a physics list

Difficulty

- Create and inherit from *G4VUserPhysicsList*
  - Atomistic approach
  - Basic interface
  - Specify all particles needed
  - For each particle specify processes
    - including transportation
    - In hadronics a process is constructed from models, and cross sections should be specified – resulting large number of lines of code
    - Difficult to support users having problems
- Create and inherit from *G4VModularPhysicsList*
  - Improved and extended interface, simpler to use
  - Allows to use exiting physics constructors
  - A large set of physics constructors provided
  - User can create custom physics constructors
- Re-use prepacked physics list directly or via factory, e.g *G4PhysListFactory*
  - While allowing users to extend or modify for specific needs

Reliability, Support

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
  - Recent addition of neutrino-nuclear interactions
- **Hadronic physics:**
  - pure strong interaction physics valid from 0 to 100 TeV
  - lepto- and gamma-nuclear interactions from ~100 / 1 MeV to 100 TeV
  - high-precision package for n, p, d,..., alpha from thermal energies to ~20 MeV
- **Parameterized or “fast-simulation” physics**
- **Biasing methods**

G4VUserPhysicsList

Interface to Define Physics List (1 of 3)

- *G4VUserPhysicsList* **is the basic Geant4 physics list interface**
  - All physics lists must be derived from this base class
  - user **must** implement the 2 pure virtual methods
    - ConstructParticle()
      - Create all particles needed in simulation, including secondary particles possibly created in simulation
    - ConstructProcess()
      - Assign specific processes to each particle
  - User can implement the SetCuts() method (optional)
  - UI: */run/setCut* preferred

```
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  };
```

G4VUserPhysicsList: CreateParticles()

- **Construct particles individually one by one**
  - Many particles in G4,
  - Particle classes
    - gluons, quarks, di-quarks
    - Leptons
    - Mesons
    - Baryons
    - Ions
    - Other

```
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**
  - 'Constructors' are under particles
    - Lepton
    - Baryon
    - Meson
    - Ion
    - ShortLived
    - Excited Nucleon, Meson, Baryon, etc

```
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  }
```

G4VUserPhysicsList: ConstructProcess()

- A **process** in Geant4 describes **reaction probability** (cross section) and it **models the interaction**, i.e. creates final state of interaction
- ConstructProcess() method general split into components for EM, hadronics, etc.
- Transportation must be added

```
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  }
```

Sketch of ConstructEM()

```
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  }
```

G4VModularPhysicsList

Interface to Define Physics List (2 of 3)

- *G4VModularPhysicsList* **extends** *G4VUserPhysicsList*
  - Adding several methods:
    - RegisterPhysics(G4VPhysicsConstructor *)
    - GetPhysics(….), by index, name, or type
    - ReplacePhysics(G4VPhysicsConstructor *)
    - RemovePhysics(…), by index, name, or type
  - Provides a more convenient way to create a physics list
  - Transportation is automatically added to all constructed particles
  - *G4VPhysicsConstructor* classes are physics modules handling a well defined defined category of physics (e.g. EM physics, hadronic physics, decay, etc.)
    - An **extensive set is provided in the physics list category.**
  - User is free to add his physics constructor or to modify existing constructors.

Sketch of YourModularPhysicsList()

```
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 Lists Constructors

- **Grouped in categories**
  - Electromagnetic, hadron_(in)elastic, decay, ions, gamma_lepto_nuclear, stopping, decay, limiters
- **Some "standard" EM physics constructors** (> 30) **:**
  - 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 hadronic physics constructors**
  - G4HadronElasticPhysics – default for hadron nuclear elastic for all hadrons
  - G4HadronElasticPhysicsHP – as above, but use HP for neutrons below 20 MeV
  - G4HadronPhysicsFTFP_BERT – hadron nucleus inelastic physics for all hadrons
  - G4IonPhysics – interactions of Ions
- **The complete list of constructors can be found in your toolkit:**
  - geant4/source/physics_lists/constructors/…
- **More information at:**
  - README files in geant4/source/physics_lists/constructors/…./README
  - http://cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Types of Physics Constructors

- **Physics constructors construct a specific subset of processes**
  - e.g. all the G4EmStandardPhysics_* physics constructors construct the EM physics processes
  - Care must be taken not to add any physics process twice
- **Physics constructors have a type**
  - Type is used to check that only one physics constructor of a given type is added
  - Existing types (defined in G4BuilderType.hh)
    - bUnknown
    - bTransportation
    - bElectromagnetic
    - bEmExtra
    - bDecay
    - bHadronElastic
    - bHadronInelastic
    - bStopping
    - bIons
  - These types can be used to retrieve, replace, or delete a physics constructor from a physics list

Pre-packaged Physics Lists

Interface to Define Physics List (3 of 3)

- **Pre-packaged physics lists**
  - Geant4 toolkit provides a large number of pre-packaged physics lists
  - "**ready-to-use**", complete lists **specialized/targeted** for various use cases
  - Created and maintained by experts, often in collaboration with users
  - Provided to **help users**, but we cannot warrant that a given list is 'correct' or best for a given use case
  - **User is responsible** to validate the physics list of his choice.
  - Not all receive the same amount of attention – see later.
  - build upon physics constructors
  - Created to **help** users
    - Examples/code snippets above were using EM, but hadronics is more complicate
      - Eg. Within the hadron inelastic process several, at least two, different models must be combined. No single hadronic model in Geant4 covers the full range in energy ⇒ see lectures on hadronic physics
      - Choice of models to combine requires expertise and validation against experimental data
      - Models often have strong and less strong points ⇒ need to evaluate and choose
- **Better support:** pre-packaged physics lists help to re-produce problems

Physics Lists naming conventions

- **Name of most physics list follows name of physics constructor for hadronic inelastic,optionally followed by EM option**
- **Name of this hadronic physics constructor indicates models in use from high to low energies**
  - High energy string model: QGS or FTF, used above few (tens) of GeV
    - Extension P in QGSP/FTFP: Precompound & De-excitation model used to de-exite remnant nucleus
  - Intermediate energies: BERT, BIC, INCLXX, used up to O(10) GeV
  - Low energy neutron/particle transport: HP,
  - Various shortcuts to indicate special variants, like TRV or LEND
- **Option of electromagnetic physics:**
  - EMV –use Opt1 EM physics
  - EMX –use Opt2 EM physics
  - EMY –use Opt3 EM physics
  - EMZ –use Opt4 EM physics
  - Plus specific DNA, GS, Liv, Pen, LE, WVI, SS
- **Exceptions to naming scheme are QBBC, Shielding, LBE, and NuBeam physics lists**

Reference physics lists

- **Combinatorial explosion - maintenance difficulty:**
  - All hadronic combinations ✕ all EM options ⇒ very large number of physics lists to implement and maintain
  - Implement a subset, all with the default EM settings: **reference physics lists**

FTFP_BERT
FTFP_BERT_ATL
FTFP_BERT_HP
FTFP_BERT_TRV
FTFP_INCLXX
FTFQGSP_BERT
FTF_BIC
QBBC
QGSP_BERT
QGSP_BERT_HP
QGSP_BIC
QGSP_BIC_AllHP
QGSP_BIC_HP
QGSP_FTFP_BERT
QGSP_INCLXX
QGS_BIC
Shielding
ShieldingLEND
LBE
NuBeam

- *G4PhysListFactory* provides physics lists with all EM options (EMX, EMZ, EMY, …), including the default
  - UI command allows to add optical or radioactive decay - */physics_lists/factory/*

```
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   ...
```

Production physics lists

- These select physics lists are better documented, maintained, and validated compared to other lists
- Used by large user groups, like LHC experiments, medical users, etc.
- These lists are more reliable, changes are done conservatively, less frequent
- These currently are: (concentrating on hadronic content, ignoring EM variants)

FTFP_BERT	the current G4 default, used in HEP collider experiments
QBBC	space physics and medical applications
QGSP_BERT	An early G4 default, was used by LHC experiments
QGSP_BIC	medical/hadrontherapy, normally used with option3 or option4 electromagnetic physics
Shielding	deep shielding applications, uses HP low energy neutron transport

- **Production physics lists are documented in the Physics List Guide**
  - http://cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Extending/modifying a physics list

- **For a *G4VModularPhysicsList* object**
  - Add the physics using the physics constructor, e.g.
    - pList->RegisterPhysics(new G4RadioactiveDecayPhysics)
  - To replace/modify, delete part of the physics, use the methods corresponding methods of *G4VModularPhysicsList*
    - Select existing physics constructor by name or type

- **All prepackaged physics lists are of type *G4VModularPhysicsList***
- **When using *G4PhysListFactory***
  - Add the physics using physics constructor, or
  - Use UI command */physics_lists/factory* to add physics
    - addRadioactiveDecay
    - addOptical

Also: UI to Modify Physics Settings

- UI commands for processes
  - Disable/enable/dump a process: */particle/process/...*
- UI kernal to set cuts
  - */run/setCut 0.1 mm*
  - */run/setCutForGivenParticle e- 10 um*
  - */run/setCutForRegion COIL 1 cm*
- UI commands for electromagnetic physics
  - Lecture V.Ivantchenko, Electromagnetic physics (Monday)
  - Macro files in examples/extended/electromagnetic
  - Many commands available under */process/...*
- UI commands for some hadronics settings
  - */process/had/verbose*
  - */process/had/maxEnergy*
- UI for *G4PhysListFactory*
  - */physics_lists/factory*
- Environment variables, e.g. for particle_hp

Extend by new particle and physics

See examples in extended/exoticphysics

- **monopole**
  - Defines and implements a class derived from *G4VPhysicsConstructor*
    - Create new particle
    - Describe physics for new particle
  - Insert into prepackaged physics lists
- **dmparticle**
  - Create physics list derived from *G4VModularPhysicsList*
  - Use existing physics constructors to register physics
  - Add new particle and its process(es)

Choosing a physics list

Choosing a Physics List

- **Ideal situation: the user(s) have a good understanding of the physics relevant for a given application**
  - the user can either decide to use a pre-defined physics list or build his own
  - the chosen physics list needs to be validated for the given application
  - can be done either by the user or by someone else in case of some reference lists
  - during the validation procedure, some parts of the physics list might be changed to add physics, remove physics, change settings, etc.
- **The given application belongs to a well defined application area (e.g. medical applications)**
  - the user can choose the reference physics list recommended for the given application area as a starting point
  - the chosen physics list needs to be validated for the given application (same as above)
- **Something that may work** (depending on application area)
  - the user can take the most accurate physics settings (e.g. opt4 for EM)
    - In hadronics generally not possible
  - run some simulation with lower statistics to obtain the most accurate result
  - then step by step revise the initial physics list by using the accurate results as reference
  - then the user can take a less accurate but faster physics setting (e.g. opt0 for EM), obtain simulation results, and compare to results for accurate physics
- **Contacting experts via Geant4 forum for advice**

Validating a Physics List

- **Validating a physics list for a given use case is the responsibility of the user**
  - When using a new release, the physics performance should be re-checked.

- **Using Geant4 validation results:**
  - Geant4 provides validation, ie. comparison to data, for most of physics codes
    - Validation is an ongoing task, repeated at least for each release
    - Over time, more validation is being added
- **Geant4 validation results**
  - Geant-val, started for HEP calorimetry validation, has expanded over the last years to include many validation results from electromagnetic physics and medical applications.
  - Physics groups providing additional validation

Examples of Physics Lists

Physics Lists Examples

- **Under examples/extended/physicslists we have three examples**
  - factory: showing how to use *G4PhysListFactory*
  - genericPL: showing how to use *G4GenericPhysicsList*, an alternative factory, becoming obsolete
    - Using physics constructors to create physics list
  - extensibleFactory: (g4alt::G4PhysListFactory) a different approach to allow users to create physics lists
    - Can create physics lists by name similar to *G4PhysListFactory*
    - Allows user to add other physics constructors, including his own.
- Examples in examples/extended/{electromagnetic, hadronic}
  - Demonstrate use of physics lists
- **Examples for specific use case will give a starting point for a physics list**
  - **Extended examples** have categories like biasing, exoticphysics, medical, optical, ...
    - Caveat: examples often include physics list restricted to physics being demonstrated
  - **Advanced examples** implement complete applications for specific use cases

Summary

- **All particles and physics processes 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
- **Reference/Production physics lists are provided by the Geant4 developers**
  - *G4PhysListFactory* provides the physics lists with chosen EM option
  - these can be used as is, or as starting points
  - Addressing different applications areas
- **Choosing the appropriate physics for a given application must be done with care**
- **Validation of a physics list is the responsibility of the user/experiment**