HepPDT: encapsulating the Particle Data Table

L.A. Garren[1], W. Brown[1], M. Fischler[1], M. Paterno[1]
[1](Fermi National Accelerator Laboratory, Batavia, Illinois 60510, U.S.A.)

Abstract

As a result of discussions within the HEP community, we have written a C++ package which can be used to maintain a table of particle properties, including decay mode information. The classes allow for multiple tables and accept input from a number of standard sources. In addition, they provide a mechanism by which an event generator can employ the tabulated information to actually direct the decay of particles.

Keywords: clhep, heppdt, stdhep

1 Introduction

For some time, there has been a need for a C++ class embodying the information contained in the Review of Particle Properties[1]. We have written HepPDT to fill this need. HepPDT allows access to particle name, particle ID, charge, nominal mass, total width, spin information, color information, constituent particles, and decay mode information. HepPDT is designed to be used by StdHepC++[2] or any Monte Carlo generated particle class. Generated particles will contain a pointer to the particle data information found in the HepPDT particle data table. HepPDT also has simple mechanisms to enable customized decay chains.

2 HepPDT Design

HepPDT has been designed to be used by any Monte Carlo particle generator or decay package. It contains only generic particle attributes. In principle, all information which can be found in the Review of Particle Properties[1] can be encapsulated in HepPDT. HepPDT contains particle information such as charge and nominal mass as well as decay mode information. This information is contained in a table which is accessed by a particle ID number. This ID number is defined according to the Particle Data Group's Monte Carlo numbering scheme[3].

HepPDT may be used alone or as part of the StdHepC++[2] package. StdHepC++ provides a standard generated particle class which can be used to communicate among various Monte Carlo generators and decay packages. A StdHep particle contains momentum information, generated mass, information about its generated decay, and a pointer to the appropriate HepPDT particle data.

Decay information is a crucial part of the particle data in HepPDT. Standard decay information is a list of allowed decay channels with associated branching fractions, decay model names and decay model code. There may also be extra information needed by the decay model (*e.g.*, helicity). A mechanism is provided so that the decay model code can be accessed using the decay data information instead of needing to use a series of if statements based on the decay model name. In addition, users often need the ability to "force" a particle to decay in a certain way. To do this, you must provide custom decay information. Often this information involves the entire decay chain (*e.g.*, $D^{*+} \rightarrow D^0\pi^+, D^0 \rightarrow K^-\pi^+$). The design allows the generated particle to have a pointer to a custom DecayData object. If this pointer is non-null,the pointed-to object overrides the DecayData associated with the generated particle's ParticleData. To customize the decay chain, the user may create particle aliases which use other special DecayData objects.

Methods are provided to create ParticleDataTable objects from Pythia, Herwig, Isajet, QQ, and EvtGen decay information. Methods are also provided to facilitate creation of custom

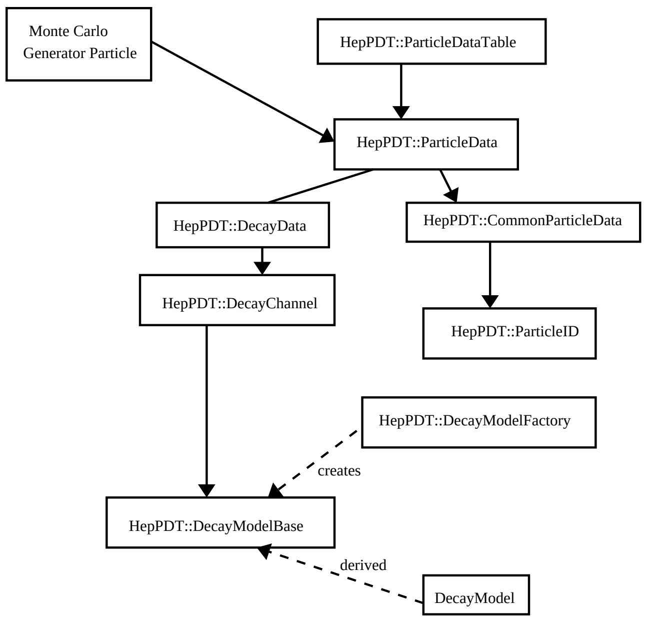

Figure 1: HepPDT Classes: Particle information is accessed by a pointer to ParticleData from any Monte Carlo generated particle. CommonParticleData contains particle information such as mass, charge, and total width. Decay information is found in DecayData. The Particle-DataTable contains a map of ParticleData objects, referenced by ParticleID, as well as lists of CommonParticleData and DecayData. ParticleData has indices to CommonParticleData and DecayData, as well as methods to access all relevant information. The DecayModelFactory is used to create DecayModelBase objects which are derived from user DecayModel classes.

particle and decay information. A ParticleDataTable object may be created from multiple information sources.

The design requires that ParticleDataTable objects must be fully created before they are used. Multiple data tables are allowed. Although potentially dangerous, we recognize that this is also a powerful option.

Figure 1 shows the interactions of the basic classes.

3 HepPDT Classes

The ParticleDataTable class contains a map of ParticleData which is keyed on the ParticleID class. Particle ID aliases can be used to add custom DecayData. ParticleDataTable also contains lists of CommonParticleData and DecayData.

The ParticleID class can be used to retrieve all the information that is implied in the particle ID (*e.g.*, charge and quark content). Boolean methods (such as isMeson, isBaryon, hasBottom, and hasTop) are provided for ease of searching for various types of particles.

The ParticleData class has iterators into the lists of CommonParticleData and Decay-

Data. CommonParticleData is extensible and includes particle name, particle ID, charge, mass, total width with cutoffs, spin information, color information, and constituent particles (*e.g.*, quark content). This class is not templated.

The DecayData class is a collection of DecayChannels. A generated particle may use the DecayData information from the ParticleDataTable entry or it may use a customized DecayData that allows, for instance, only a single DecayChannel. Users may add customized DecayData objects to the ParticleDataTable.

Each DecayChannel has a collection of decay channel products (which are pointers to ParticleData), a decay name, a branching fraction, an optional vector of extra decay model parameters, and a pointer to DecayModelBase. We recognize that other information, such as helicity, may be needed by a particular DecayChannel object. Because there are many options, this information is stored as a vector of doubles.

4 Use of Templates

Decay methods are, by definition, outside the scope of HepPDT. However, the user may wish to be able to call decay methods from DecayChannel.

DecayModelBase is the mechanism that allows the user to invoke an actual decay method from this class. Because the decay method must know what kind of generated particle will be created, DecayModelBase is a class *template*, parameterized on the type of particle to be generated. Because of the need to allow the user freedom of choice for this generated particle class, many related abstractions in this model are also class templates. In each case, a typedef is provided to allow users who do not need this degree of flexibility to ignore it. These typedefs also stand as an example for those groups who wish to standardize upon a common type for the generated particles.

5 Decay Model Factory

In order to allow users to "plug-in" new decay models (subclasses of DecayModelBase), without requiring recompilation of the entire library, we provide the class template DecayModelFactory, and a simple means by which user-written decay models will be automatically registered for creation, when requested by a DecayData object. This is done by the common technique of having each decay model register (at static initialization time) a function object that is used to create instances of that decay model.

6 Conclusions

HepPDT provides access to all useful particle data properties and is designed to be used with any generated particle. It also contains a factory to allow the user to directly access decay model code instead of needing to use a lookup table or series of if statements based on the decay model name. HepPDT will be part of the StdHepC++ package in CLHEP[4] and is available now at `http://www-pat.fnal.gov/stdhep/c++/`.

References

[1] Particle Data Group: Groom, D.E. *et al.*, *The European Physical Journal* **C3**, (2000).

[2] StdHepC++: `http://www-pat.fnal.gov/stdhep/c++/`.

[3] Particle Data Group: Groom, D.E. *et al.*, *The European Physical Journal* **C3**, (2000) 205, `http://www-pdg.lbl.gov/mc\_particle\_id\_contents.html`.

[4] CLHEP: `http://wwwinfo.cern.ch/asd/lhc++/clhep/`.