ThePEG, herwig++ AND Ariadne

LEIF LÖNNBLAD
Department of Theoretical Physics, Lund University, Sweden

I present the status of the ThePEG project for creating a common platform for implementing C++ event generators. I also describe briefly the status of the new versions of herwig and Ariadne which are implemented using this framework.

1. Introduction

Monte Carlo Event Generators (EGs) have developed into essential tools in High Energy Physics. Without them it is questionable if it at all would be possible to embark on large scale experiments such as the LHC. Although the current EGs work satisfactorily, the next generation of experiments will substantially increase the demands both on the physics models implemented in the EGs and on the underlying software technology.

The current EGs are typically written in Fortran and their basic structure was designed almost two decades ago. Meanwhile there has been a change in programming paradigm, towards object oriented methodology in general and C++ in particular. This applies to almost all areas of high-energy physics, but in particular for the LHC program, where all detector simulation and analysis is based on C++. When designing the next generation of EGs it is therefore natural to use C++. Below is a brief description of the ThePEG [1] project for designing a general framework in C++ for implementing EG models, and also the herwig++ and Ariadne programs which uses ThePEG to implement their respective physics models.

2. Basic structure

ThePEG is a general platform written in C++ for implementing models for event generation. It is made up from the basic model-independent parts of Pythia7 [2, 3], the original project of rewriting the Lund family of EGs in

`C++` [a]. When the corresponding rewrite of the HERWIG program [5] started it was decided to use the same basic infrastructure as PYTHIA7 and therefore the THEPEG was factorized out of PYTHIA7 and is now the base of both PYTHIA7 and HERWIG++ [6]. Also the coming `C++` version of ARIADNE [7] is using THEPEG.

THEPEG uses CLHEP [8] and adds on a number of general utilities such as smart pointers, extended type information, persistent I/O, dynamic loading and some extra utilities for kinematics, phase space generation etc.

The actual event generation is then performed by calling different *handler* classes for hard partonic sub-processes, parton densities, QCD cascades, hadronization etc. To implement a new model to be used by THEPEG, the procedure is then to write a new `C++` class inheriting from a corresponding handler class and implement a number of pre-defined virtual functions. Eg. a class for implementing a new hadronization model would inherit from the abstract `HandronizationHandler` class, and a new parton density parameterization would inherit from the `PDFBase` class.

To generate events with THEPEG one first runs a setup program where an `EventGenerator` object is set up to use different models for different steps of the generation procedure. All objects to be chosen from are stored in a *repository*, within which it is also possible to modify switches and parameters of the implemented models in a standardized fashion, using so called *interface* objects. Typically the user would choose from a number of pre-defined `EventGenerator` objects and only make minor changes for the specific simulation to be made. When an `EventGenerator` is properly set up it is saved persistently to a file which can then be read into a special run program to perform the generation, in which case special `AnalysisHandler` objects may be specified to analyze the resulting events. Alternatively it can be read into eg. a detector simulation program or a user supplied analysis program, where it can be used to generate events.

3. Status of THEPEG

THEPEG version 1.0α is available [1] and is working. As explained above, it contains the basic infrastructure for implementing and running even gen-

[a]In an unfortunate turn of events, the principal PYTHIA author, Torbjörn Sjöstrand, has decided to leave the THEPEG collaboration and is currently developing a new `C++` version of PYTHIA (called PYTHIA8 [4]) on his own. This means that the current version of PYTHIA7, including only basic string fragmentation and parton showers, is frozen. Hopefully it will be possible to interface the different modules in PYTHIA8 so that they can be used within the general framework of THEPEG.

eration models. It also contains some simple physics models, such as some $2 \to 2$ matrix elements, a few parton density parameterizations and a near-complete set of particle decays. However, these are mainly in place for testing purposes, and to generate realistic events, the PYTHIA7 and/or HERWIG++ programs are needed.

Currently the program only works under Linux and Mac OSX, using the `gcc` compiler. This is mainly due to the extensive use of dynamic linking of shared object files, which is inherently platform-dependent. Recently, the build procedure has been redesigned using the `libtool` facility [9], which should allow for easy porting to other platforms in the future.

Although THEPEG includes a general structure for implementing basic fixed-order matrix element generation to produce the initial hard subprocesses in the event generation, a general procedure for reading such parton level events from external programs using the Les Houches accord [10] has been developed and will be included in the next release[b].

The documentation of THEPEG is currently quite poor. Recently the actual code documentation was converted to Doxygen format [11], which will hopefully facilitate the documentation process. The lack of documentation means that there is currently a fairly high threshold for a beginner to start using and/or developing physics modules for THEPEG. The situation is somewhat alleviated by the recent addition of a Java-based graphical user interface for the setup.

3.1. *ARIADNE*

The reimplementation of the ARIADNE [7] program using the framework of THEPEG has just started and is, hence, not publically available yet. Although this is mainly a pure rewrite of the Fortran version of ARIADNE, it will contain some improvements, such as the CKKW-matching [12–14]. In addition, an improved version of the LDCMC [15] is planned.

3.2. *HERWIG++*

The first version of HERWIG++ was released in 2003 [6] and contained a re-implementation of the HERWIG cluster hadronization and an improved version [16] of the angular ordered HERWIG parton shower. This version was only able to handle e^+e^--annihilation, but a new version is expected this year which will also be able to handle hadronic collisions. This release will

[b]A snapshot of the current development version is available from [1]

include eg. an extended list of hard matrix elements, a rudimentary model of underlying events, initial state showers, and an improved modeling of secondary hadronic decays. Work has also started to include in a future version eg. an improved modeling of gluon radiation in the decay of heavy coloured particles, beyond-the-standard-model processes, CKKW-matching and a more sophisticated underlying-event model [17,18]. More information about these improvements and a more extensive description of the program is available on the web [19].

4. Conclusions

ThePEG was intended to be *the* standard platform for event generation for the LHC era of collider physics. Unfortunately, this does not seem to become a reality. Besides the recent split between Pythia and ThePEG, there will also be other separate programs such as Sherpa [20]. This is, of course, not an optimal situation, especially not for the LHC experiments, which presumably would have preferred a uniform interface to different event generator models.

References

1. L. Lönnblad *et al.*, “ThePEG program.” `http://www.thep.lu.se/ThePEG`.
2. M. Bertini *et al.* *Comp. Phys. Comm.* **134** (2001) 365, `hep-ph/0006152`.
3. L. Lönnblad *et al.*, “Pythia7 program.” `http://www.thep.lu.se/Pythia7`.
4. T. Sjöstrand, “Pythia8 program.” `http://www.thep.lu.se/~torbjorn/future/`.
5. G. Corcella *et al.* *JHEP* **01** (2001) 010, `hep-ph/0011363`.
6. S. Gieseke *et al.* *JHEP* **02** (2004) 005, `hep-ph/0311208`.
7. L. Lönnblad *Comput. Phys. Commun.* **71** (1992) 15–31.
8. L. Lönnblad *Comput. Phys. Commun.* **84** (1994) 307–316.
9. G. Matzigkeit *et al.*, “`libtool`.” `http://www.gnu.org/software/libtool`.
10. E. Boos *et al.* `hep-ph/0109068`.
11. D. van Heesch, “The Doxygen system.” `http://www.doxygen.org`.
12. S. Catani *et al.* *JHEP* **11** (2001) 063, `hep-ph/0109231`.
13. L. Lönnblad *JHEP* **05** (2002) 046, `hep-ph/0112284`.
14. N. Lavesson and L. Lönnblad *JHEP* **07** (2005) 054, `hep-ph/0503293`.
15. H. Kharraziha and L. Lönnblad *JHEP* **03** (1998) 006, `hep-ph/9709424`.
16. S. Gieseke *et al.* *JHEP* **12** (2003) 045, `hep-ph/0310083`.
17. J. M. Butterworth *et al.* *Z. Phys.* **C72** (1996) 637–646, `hep-ph/9601371`.
18. I. Borozan and M. H. Seymour *JHEP* **09** (2002) 015, `hep-ph/0207283`.
19. S. Gieseke *et al.*, “HERWIG++ program.” `http://hepforge.cedar.ac.uk/herwig/`.
20. T. Gleisberg *et al.* *JHEP* **02** (2004) 056, `hep-ph/0311263`.