THEPEG
Toolkit for High Energy Physics Event Generation

Leif Lönnblad
Department of Theoretical Physics, Lund University, Sweden

Abstract
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 version of ARIADNE implemented using this framework.

1 Introduction

Monte Carlo Event Generators 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 event generators work satisfactorily, the next generation of experiments will substantially increase the demands both on the physics models implemented in the event generators and on the underlying software technology.

Below is a very brief description of the THEPEG [1] project for designing a general framework in `C++` for implementing event generator models, and also the ARIADNE program which uses THEPEG to implement the underlying dipole cascade model. Also HERWIG++ [2] is implemented in the THEPEG framework, but this program is described elsewhere in these proceedings.

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 [3, 4], the original project of rewriting the Lund family of event generators in `C++`. 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++ [2]. Also the coming `C++` version of ARIADNE [6] is using THEPEG. It should be noted, however, that the new `C++` version of PYTHIA, called PYTHIA8 is not built on THEPEG.

THEPEG implements a number of general utilities such as smart pointers, extended type information, persistent I/O, dynamic loading, a system for handling physical units 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. These classes communicate with each other and with the underlying framework using pre-defined virtual function definitions and a highly structured Event object.

To generate events with THEPEG one first runs a setup program where an EventGenerator object is set up to use objects implementing 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, the EventGenerator 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

THEPEG version 1.2 is available [1] and is working. As explained above, it contains the basic infrastructure for implementing and running event generation models. It also contains some simple physics models, such as some $2 \to 2$ matrix elements, a few parton density parameterizations (and an interface to LHAPDF [7]) 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 MacOS using the gcc compiler. This is mainly due to the use of dynamic linking of shared object files, which is inherently platform-dependent. However, the build procedure uses the libtool facility [8], which will hopefully allow for easy porting to other platforms in the fututre.

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 [9, 10] is included.

The documentation of THEPEG is currently quite poor. The code itself is documented using the Doxygen format [11], which provides some technical documentation. 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. However, THEPEG has a well worked through low-level interface to be able to set parameter and switches, etc. in classes introduced to the structure from the outside. This means that the running of THEPEG does not require a C++ expert, but can be handled through a simple command-line facility or through a Java-based graphical user interface.

Among the recent developments in THEPEG one can note that there is now an option to do compile-time checking of units in all mathematical expressions. Also a number of helicity classes for construction of matrix elements has been imported from HERWIG++. Furthermore, the dependence on CLHEP [12] has been dropped and the only dependence on external packages is the GNU scientific library [13], which is a standard package in all Linux distributions.

3.1 ARIADNE

The reimplementation of the ARIADNE [6] program using the framework of THEPEG has started but is not yet publically available. Although this is mainly a pure rewrite of the fortran version of ARIADNE, it will contain some improvements, such as CKKW matching [14, 15]. In addition, an improved version of the LDCMC [16] is planned.

ARIADNE is supposed to be used together with Lund string fragmentation, and for that purpose an interface of relevant parts of the PYTHIA8 program to the THEPEG framework is planned. Meanwhile there is already a simplified implementation of string fragmentation in the PYTHIA7 program [4] which was the first attempt to reimplement PYTHIA into `C++`.

4 Conclusions

THEPEG can now be considered to be a stable piece of software. Several improvements can be expected in the future, but the basic structure is fixed and has been working well for all models which have been implemented so far.

References

[1] Leif Lönnblad and others, *THEPEG program*. `http://www.thep.lu.se/ThePEG`.

[2] Bahr, M. and others (2008).

[3] Bertini, Marc and Lönnblad, Leif and Sjöstrand, Torbjörn, Comput. Phys. Commun. **134**, 365 (2001).

[4] Leif Lönnblad and others, *PYTHIA7 program*. `http://www.thep.lu.se/Pythia7`.

[5] Corcella, G. and others, JHEP **01**, 010 (2001).

[6] Lönnblad, Leif, Comput. Phys. Commun. **71**, 15 (1992).

[7] Giele, W. and others (2002).

[8] Gord Matzigkeit and others, *The `libtool` program*. `http://www.gnu.org/software/libtool`.

[9] Boos, E. and others (2001).

[10] Alwall, J. and others, Comput. Phys. Commun. **176**, 300 (2007).

[11] Dimitri van Heesch, *The doxygen documentation system*. `http://www.doxygen.org`.

[12] Lönnblad, Leif, Comput. Phys. Commun. **84**, 307 (1994).

[13] M. Galassi and others, *The gnu scientific library*. `http://www.gnu.org/software/gsl`.

[14] Catani, S. and Krauss, F. and Kuhn, R. and Webber, B. R., JHEP **11**, 063 (2001).

[15] Lönnblad, Leif, JHEP **05**, 046 (2002).

[16] Kharraziha, Hamid and Lönnblad, Leif, JHEP **03**, 006 (1998).