COMPUTING METHODS IN HIGH ENERGY PHYSICS

S. Lehti

Helsinki Institute of Physics

Spring 2024

Event generators

Introduction

Event generators generate simulated high-energy physics events. Despite the simple structure of the tree-level perturbative quantum field theory description of the collision and decay processes in an event, the observed high-energy process usually contains significant amount of modifications, like photon and gluon bremsstrahlung or loop diagram corrections, that usually are too complex to be easily evaluated in real calculations directly on the diagrammatic level. Any realistic test of the underlying physical process in a particle accelerator experiment, therefore, requires an adequate inclusion of these complex behaviors surrounding the actual process. Based on the fact that in most processes, a factorization of the full process into individual problems is possible (which means a negligible effect from interference), these individual processes are calculated separately, and the probabilistic branching between them are performed using Monte Carlo methods.

The final-state particles generated by event generators can be fed into a detector simulation, allowing a precise prediction and verification for the entire system of experimental setup. However, as the detector simulation is usually a complex and

Event generators

computationally expensive task, simple event analysis techniques are also performed directly on event generator results.

A typical hadronic event generator simulates the following subprocesses:

- Initial-state composition and substructure
- Initial-state radiation
- The hard process
- Resonance decay
- Final-state radiation
- Accompanying semi-hard processes
- Hadronization and further decay

A typical *heavy ion* event generator usually can be less strict in simulating the rare and rather negligible processes found in a hadronic generator, but it would need to simulate the following subprocesses, in addition to those in a hadronic generator:

- Nuclear initial-state
- High multiplicity, soft processes
- In-medium energy loss
- Collective behavior of the medium (not handled properly by any generators so far)

The major event generators that are used by current experiments are e.g.:

- Pythia6 and Pythia8

Event generators

- Herwig6, Herwig++, Herwig7
- Powheg
- Alpgen
- MG5_aMC@NLO
- CalcHEP

As the name indicates, the output of an event generator should be in the form of events, with the same average behavior and the same fluctuations as in the real data.

Event generators can be used in many different ways. The five main applications are probably the following:

- To give a feeling for the kind of events one may expect/hope to find, and at what rates
- As a help in the planning of a new detector, so that detector performance is optimized, within other constraints, for the study of interesting physics scenarios.
- As a tool for devising the analysis strategies with real data, to optimize signal-to-background conditions
- As a method for estimating detector acceptance corrections that have to be applied to raw data
- As a convenient framework within which to interpret the observed phenomena in terms of a more fundamental underlying theory

Event generators

PYTHIA

The PYTHIA program is frequently used for event generation in high-energy physics. The emphasis is on multiparticle production in collisions between elementary particles. This in particular means hard interactions in $\mathrm{e}^+\mathrm{e}^-$, pp and ep colliders, although also other applications are envisaged. The program is intended to generate complete events, in as much detail as experimentally observables ones, within the bounds of our current understanding of the underlying physics.

The documentation is extensive, and extremely useful, and it can be found in
https://pythia.org

The PYTHIA package is completely self-contained. In addition interfaces to externally defined subprocesses, PDF libraries, τ decay libraries and a time routine are provided, plus a few optional interfaces. A worksheet, found in the distribution, contains a step-by-step procedure, and example main programs are provided in the examples/ directory.

The program philosophy of PYTHIA is that the Monte Carlo program is built as a slave system, and the user has to supply the main program. The details how the program should perform specific tasks is done by

Event generators

configuring it by a Pythia::readString function. The program is told to generate events according to rules established by the user. To extract information on the generated events, one can call various functions to analyse the events.

In the core process generation machinery, like selection of hard process kinematics, a sizable amount of initialization is performed in the Pythia::init() call, and thereafter the events generated by Pythia::next() all obey rules established at that point. The variables specifying methods and constraints to be used have to be set before calling Pythia::init().

PYTHIA is extremely versatile, but the price to be paid for this is having a large number of adjustable parameters and switches for alternative modes of operation. Since all these parameters and switches are assigned sensible default values, there is no reason to worry about them until the need arises. Unless explicitly stated, all switches and parameters can be changed independently of each other.

The normal usage of PYTHIA can be subdivided into three steps:

- initialization
- event generation loop
- printing and histogramming

Event generators

In the initialization step all basic characteristics of the coming generation are specified. This includes the selection of required processes, the selection of kinematics, definition of the underlying physics scenario, e.g. the Higgs mass, selection of PDF's, switching off of generation parts not needed, and initialization of the event generation procedure.

In the event loop the events are generated and studied. The generation of next event is done with call

Pythia::next()

The events are then saved on disk, or interfaced to a detector simulation.

In the final step, in addition to producing output, the events can be normalized to expected rates as the number of events simulated is usually not the number of events expected. For example a process with a very large cross section may be expected to give millions of events, while simulating that many events may not always be feasible. In that case the events must be given weights.

Example 1: How to get a particle listing

In PYTHIA every particle is given a unique code, a KF code. This code is needed e.g. when forcing decays, or when you want to select a particle in your analysis for some reason. To

Event generators

get a full list of the KF codes, one can call Pythia::ParticleData::list()

Main program:

```
#include 'Pythia8/Pythia.h'
using namespace Pythia8;
int main() {
  Pythia pythia;
  pythia.particleData.list();

}
```

This list is so useful that you might want to save the output on your disk or even print it.

The decay modes for each particle are also included in the list. These can be used for switching decay modes on and off.

Example 2: Simulating Z boson production

Let's assume we want to study the Z boson in the $\mu^+\mu^-$ final state. For that we need to initialize PYTHIA to produce Z bosons, and then force them to decay into muons, as there is no need to include the other decay modes producing very different signatures in the detector. The initialization step looks like following:

```
pythia.readString(
'Beams:eCM = 13000.');
pythia.readString(
'WeakSingleBoson:ffbar2gmZ = on');
pythia.readString('23:onMode = off');
```

Event generators

```
pythia.readString(
'23:onIfMatch = -13 13');
pythia.readString(
'PhaseSpace:mHatMin = 80.');
pythia.readString(
'Next:numberShowEvent=0');
pythia.init();
```

The first line selects the CM energy. Next the process is selected. The list of processes available can be found in the Pythia manual, e.g. the Z (+gamma*) process is found under 'Electroweak processes, Single boson'

flag WeakSingleBoson:ffbar2gmZ (default = off)
The third line switches all decay modes of particle 23 (Z boson) off, and the fourth line sets decays to particles 13 (muon) on. The phase space is limited by mHatMin to simulate only events with the mu+mu invariant mass 80 GeV or larger. By default the Pythia::next() function lists the event content of the first event, but here that feature is disabled.

If one looks at the listing of the hard processes, it quickly becomes obvious, that the list contains only Z bosons, despite the process being described as Z/gamma*. Pythia manual explains: *"Note: irrespective of the option used, the particle produced will always be assigned code 23 for Z0, and open decay channels is purely dictated by what is set for the Z0."*

Event generators

The event loop consists of a loop over as many events as we want. Choosing too low a value for the number of simulated events results in poor statistics, but choosing too large a value will make the simulation computationally expensive and very time consuming. Within the event loop the events are generated by calling Pythia::next()

The final step consists of event normalization. If we assume an integrated luminosity in units of fb^{-1}, the normalization factor (event weight) will be

xsec = Pythia.info.sigmaGen()
//[mb]
CNOR = xsec*1e12 *LUMI/NEV

where xsec is the total integrated cross section, NEV is the number of events and LUMI the chosen luminosity. The factor 1e12 comes from unit conversion, in PYTHIA the cross sections are in mb's, while the luminosity here is in fb^{-1}.

Example 3: How to set PDF's

In the previous example we used PYTHIA's default parton distribution function (PDF), which is NNPDF2.3 QCD+QED LO (Neural Network PDF). If some other PDF is preferred, it can be selected in the initialization.

The complete set of PDF's available in PYTHIA are listed in the PYTHIA manual section PDF Selection.

Event generators

More can be found in external libraries of parton distributions, PDFLIB and LHAPDF. PDFLIB is the older one, and no new parton distributions has been added in it for a couple of years. The newer LHAPDF (the Les Houches Accord PDF interface) contains all new sets of the last five years or so. LHAPDF has an interface LHAGLUE, which is identical to the PDFLIB interface.

To change PDF to one already available in PYTHIA, use

pythia.readString('PDF:pSet = value');

to select the set. Use PYTHIA manual to check which value of PDF:pSet selects the chosen PDF.

To change the PDF used in example 2 to a PDF from an external library, the external library must be available, and Pythia must be linked with it. In the Pythia manual it is stated that 'Pythia comes with a reasonably complete list of recent LO fits built-in, so there is no real need to link any external PDF sets.' We will, however, do that anyway, for the exercise. As PDFLIB is becoming obsolete, LHAPDF is used as an example. The procedure is the following:

Download and install LHAPDF, and download the PDF set you are planning to use. We use LHAPDF6

Event generators

and cteq6l1 as an example.

Install LHAPDF6, and copy the relevant data (cteq6l1) in $LHAPDF_DATA_PATH.

Configure Pythia8 installation with

./configure –with-lhapdf6=<path to LHAPDF>

and recompile.
In the main program add line

```
pythia.readString('PDF:pSet=
    LHAPDF6:cteq6l1');
```

If you need to change the value of alpha_S, add
pythia.readString("SigmaProcess: alphaSvalue = 0.1298");

HERWIG

HERWIG7 is another general purpose Monte Carlo event generator. It includes the simulation of hard lepton-lepton, lepton-hadron and hadron-hadron scattering and soft hadron-hadron collisions. It uses parton-shower approach for initial and final state QCD radiation, including colour coherence effects and azimuthal corrections both within and between jets.

A generic hard process, i.e. process with high momentum transfer, is divided in four components in HERWIG. These are

- Elementary hard subprocess
- Initial and final state parton showers

Event generators

- Heavy object decays
- Hadronization process

The official HERWIG homepage for documentation and source code is http://herwig.hepforge.org

Example 4: Simulating Z boson production

Let's repeat the example 2, now using HERWIG instead of PYTHIA. Download and install Herwig7. Now we dont write a main program, but use config files to inform what Herwig is supposed to do. A good starting point is in the Herwig manual section Getting started.

cp Herwig/share/Herwig/LHC-matchbox.in .

In the LHC-matchbox.in edit the final state particles and ChargedLeptonPairMassCut to match the value we used in Pythia, 80 GeV. Type Herwig read

LHC-Matchbox.in
This creates a .run file, which is run with:

Herwig/bin/Herwig run LHC-matchbox.run -N 1000

Here the -N specifies the number of events to be created. In the output one can see the different tried processes, their cross sections, total cross section etc.

Event generators

cross section etc.

Next, let's create the output in HepMC format. For that we need to download and install HepMC, and configure Herwig to use it.

herwig-bootstrap –with-hepmc=<path to HepMC>

In the Herwig input file we need

insert EventGenerator:AnalysisHandlers 0 /Herwig/Analysis/HepMCFile

set /Herwig/Analysis/HepMCFile:PrintEvent 100

set /Herwig/Analysis/HepMCFile:Format GenEvent

set /Herwig/Analysis/HepMCFile:Units GeV_mm

set /Herwig/Analysis/HepMCFile:Filename events.fifo

The HepMC output can then be processed separately. Herwig supports also an internal mechanism for performing the analysis of the events generated by Herwig but it is recommended that it is only to make use of it to either output the events to a file in the HepMC format, or call the Rivet package to analyse the events. This mechanism works by calling a number of AnalysisHandler classes from the EventGenerator.

Event generators

To be able to use LHAPDF instead of the built-in PDF sets, one needs to specify the installation path of LHAPDF

herwig-bootstrap –with-lhapdf=<path to LHAPDF>

To use a PDF set from LHAPDF one needs to create a new PDF object. This is done in the input file by adding lines

cd /Herwig/Partons

create ThePEG::LHAPDF myPDFset ThePEGLHAPDF.so

set myPDFset:PDFName NNPDF30_nlo_as_0118

set myPDFset:RemnantHandler HadronRemnants

More about Herwig and Parton Distribution Functions in the Herwig manual.

OTHER EVENT GENERATORS

The number of event generators is increasing. Often the general purpose event generators can be used, or they can serve as a good starting point, but in many cases there are special event generators available giving more precise predictions. The event generators can be specialized in e.g. top physics, or vector boson fusion, or associated production with n jets etc. It is often very healthy to compare the results from different event generators, and to try to

Event generators

understand the differences. Which event generator to use is up to the user, but when presenting results the choice of the event generator may be criticized.

Many of the specialized event generators include code only for the core calculations, and use general purpose event generators like PYTHIA and HERWIG as a user interface. This simplifies the HEP simulation, as the event generators can be easily switched from one to another.