Preprint typeset in JHEP style - PAPER VERSION

March 16, 2011
POWHEG BOX 1.0

The POWHEG BOX user manual: dijet production

Simone Alioli

Deutsches Elektronen-Synchrotron DESY
Platanenallee 6, D-15738 Zeuthen, Germany
E-mail: simone.alioli@desy.de

Keith Hamilton

INFN, Sezione di Milano-Bicocca, Piazza della Scienza 3, 20126 Milan, Italy
E-mail: Keith.Hamilton@mib.infn.it

Paolo Nason

INFN, Sezione di Milano-Bicocca, Piazza della Scienza 3, 20126 Milan, Italy
E-mail: Paolo.Nason@mib.infn.it

Carlo Oleari

Università di Milano-Bicocca and INFN, Sezione di Milano-Bicocca
Piazza della Scienza 3, 20126 Milan, Italy
E-mail: Carlo.Oleari@mib.infn.it

Emanuele Re

Institute for Particle Physics Phenomenology, Department of Physics
University of Durham, Durham, DH1 3LE, UK
E-mail: emanuele.re@durham.ac.uk

Abstract: This note documents the use of the package POWHEG BOX for jet pair production processes. Results can be easily interfaced to shower Monte Carlo programs, in such a way that both NLO and shower accuracy are maintained.

Keywords: POWHEG, Shower Monte Carlo, NLO.

Contents

1. Introduction

The POWHEG BOX program is a framework for implementing NLO calculations in Shower Monte Carlo programs according to the POWHEG method. An explanation of the method and a discussion of how the code is organized can be found in refs. [1, 2, 3]. The code is distributed according to the "MCNET GUIDELINES for Event Generator Authors and Users" and can be found at the web page

http://powhegbox.mib.infn.it.

In this manual, we describe the POWHEG NLO implementation of jet pair hadroproduction, as described in ref. [4].

2. Producing POWHEG events out-of-the-box

The POWHEG BOX dijet program may be run directly from the downloaded code as follows:

- Enter the POWHEG-BOX/Dijet directory.
- Type make here to produce an executable pwhg_main for producing POWHEG events.[1]
- Enter either directory testrun-tev or testrun-lhc.

[1]The compiler will relay numerous warnings due to our use of the -Wall compilation flag; as with most High Energy Physics software these can be safely ignored.

- Type `../pwhg_main` to run the program using the local `powheg.input` input file to produce a Les Houches format event file, `pwgevents.lhe`, containing 50000 `POWHEG` events.

The run-time for producing such an event sample should be in the region of three hours. The number of events to be generated may be freely changed and is set by the `numevts` flag in `powheg.input`. Such events are distributed according to the so-called `POWHEG` hardest emission cross section, essentially a leading-log resummed version of the next-to-leading order cross section, thus they contain three-parton final-states with very few additional, 'non-radiative', two-parton final-state events.

The user is free to analyse the `POWHEG` predictions to give NLO accurate, resummed, predictions for observables using their own analysis code, or, to fully develop them into life-like hadron level events by passing them to a shower Monte Carlo program. In any case, we have included within the `POWHEG BOX` package programs which can carry out all of these tasks with a minimum of effort, as described in the following sections.

3. NLO and `POWHEG`-level analysis with FastJet

Besides producing the `POWHEG` events the program can also analyse them on-the-fly whilst they are being generated. At least for jet production, to carry out such analysis the user should have installed a copy of the FastJet jet-clustering package[2] and the directory of the `fastjet_config` script included in his / her `PATH` environment variable. With FastJet set up one can produce some basic `POWHEG`-level topdrawer histogram output by replacing, at the beginning of the `Dijet/Makefile`,

`ANALYSIS=none` $\Rightarrow$ `ANALYSIS=default`

which simply swaps `pwhg_analysis-dummy.f` by the FastJet dependent `pwhg_analysis.f` code. Now, repeating the same steps given in Sect. 1, yields an additional topdrawer file, `pwgpwhgalone-output.top`, containing the analysis.

The `pwhg_analysis.f` file contains very basic code passing particle momenta to FastJet as well as booking and filling histograms. The pre-packaged analysis was used, in part, to produce a number of the results in our related publication [4] but can be overwritten by users as they wish.

As well as producing analysis at the level of the `POWHEG` hardest emission cross section, the software is also capable of producing the corresponding fixed order NLO analysis, since the NLO cross section must be throughly explored prior to event generation to determine an upper bound for the $\widetilde{B}(\Phi_B)$ function — the distribution of the underlying-Born kinematic configurations. This analysis proceeds through the exact same `pwhg_analysis.f` code and maybe activated by replacing

`#testplots 1` $\Rightarrow$ `testplots 1`

[2] FastJet is available at `http://www.lpthe.jussieu.fr/~salam/fastjet/`

in powheg.input. Hopefully it is obvious that enabling this feature depends on having set ANALYSIS = default in the Makefile. If one is only interested in obtaining fixed order NLO results it is not necessary to generate any POWHEG events, hence, the process is expedited by then setting numevts to zero.

Of course the Les Houches file content may be analysed 'off-line', independently of the way in which it was generated, using standalone code. Users can readily write their own programs to this end, however, for convenience we include a program to do the task using, again, the pwhg_analysis.f code. To build this program, lhef_analysis,

- Enter POWHEG-BOX/Dijet/ .
- Type make lhef_analysis .

Two inputs are required in order to run the analysis: the input file used to create the Les Houches events, foo-powheg.input, and the event file bar.lhe. If the input file has been misplaced it can be easily recovered from the head of the Les Houches event file. Assuming these two files are in the directory POWHEG-BOX/Dijet/foo-bar

- Enter POWHEG-BOX/Dijet/foo-bar/ .
- Type ../lhef_analysis .

When prompted enter the prefix of the input file 'foo' and, on further prompting, the full name of the event file. Note that the program assumes that the name of the input file is formatted as *-powheg.input. The program will run to produce a topdrawer histogram file, LHEF_analysis.top.

If run on the same event file created by pwhg_main, with no modifications to pwhg_analysis.f, one should find LHEF_analysis.top and pwgpwhgalone-output.top are basically identical, making this seem like a redundant feature. However, if one is only interested in the POWHEG predictions and not the (inferior) fixed order NLO ones, it is of course hugely faster to modify pwhg_analysis.f to your tastes and then run lhef_analysis, as opposed to running the event generation code over again from the beginning. Moreover, if one is ultimately interested in generating large samples of events it will be necessary to generate them independently on a CPU cluster whereupon they may be combined at the end and analysed *en masse* with lhef_analysis.[3]

4. Evolving POWHEG events to the hadron level

Given a Les Houches file of POWHEG events one may evolve these to the hadron level with Herwig as follows

- Enter POWHEG-BOX/Dijet/ .
- Type make main-HERWIG-lhef .

[3]For more details on parallel generation of event samples see POWHEG-BOX/Docs/Manyseeds.pdf.

As in Sect. 2, to run the program we require the input file used to create the Les Houches event file, `foo-powheg.input`, and the event file itself `bar.lhe` in the same directory. Assuming these are present in `POWHEG-BOX/Dijet/foo-bar`

- Enter `POWHEG-BOX/Dijet/foo-bar/` .
- Type `../main-HERWIG-lhef`.

When prompted enter the prefix of the input file '`foo`' and on further prompting '`bar.lhe`'. Assuming that `ANALYSIS = default` in the `Makefile` the program will run to produce a topdrawer histogram file, `foo-POWHEG+HERWIG-output.top`, based on the analysis of the hadron-level events with the (freely changeable) `pwhg_analysis.f` code. Although we have not included the facility explicitly it is of course possible to store the showered events in a binary format for direct off-line analysis and / or detector simulation with little effort.

The file `main-HERWIG.f` in the main `POWHEG-BOX` directory contains a short generic program driving the HERWIG run, while `setup-HERWIG-lhef.f`, in the `Dijet` directory, contains an equally brief user-serviceable subroutine, `setup_HERWIG_parameters`, allowing one to set options affecting the behaviour of HERWIG.

All of the discussion and instructions above here hold true for the PYTHIA event generator, up to replacing `HERWIG` → `PYTHIA` everywhere it appears in the shell commands. As in the analogous case of HERWIG, `setup_PYTHIA_parameters` isolates the setting of PYTHIA options; here one can, for example, disable the multiple interactions / underlying event model, substantially quickening studies in their preliminary stages. One minor difference with respect to the HERWIG driver code is the routine `setup_PYTHIA_tune` routine in `setup-PYTHIA-lhef.f` file. This routine simply calls `PYTUNE` with an index to a particular PYTHIA tuning, other tunings may be freely selected by simply changing the argument of `PYTUNE`.

5. Running with Lhapdf

The `POWHEG BOX` includes, for convenience, its own internal PDF evolution code and a limited number of PDF data sets (see the `POWHEG-BOX/pdfdata/`). It is, however, easy to use instead the popular LHAPDF package for this purpose. To do this edit the beginning of the `Makefile` by simply replacing the line

```
PDF=native ⇒ PDF=lhapdf
```

and in the `foo-powheg.input` file

```
ndns1 131 ⇒ !  ndns1 131

ndns2 131 ⇒ !  ndns2 131

!  lhans1 10050 ⇒ lhans1 10050

!  lhans2 10050 ⇒ lhans2 10050
```

Of course, for this to work, LHAPDF should be correctly installed and the directory containing the `lhapdf-config` script should be included in the `PATH` environment variable. Note also that the `Makefile` we include assumes the latest version of LHAPDF is installed, if this is not the case one may need to adjust the setting of the `LIBSLHAPDF` variable in it, replacing the two instances of `--libdir)` on line 58 by `--prefix)/lib`. Lastly, with performance in mind we recommend building LHAPDF with the `--enable-low-memory` configure option.[4]

6. `bornktmin`

As with all Monte Carlo simulations of processes such as this, which are already divergent at the leading order, a generation cut is required in the simulation process to avoid the singular, low p_T, region of phase space. The presence of such cuts is nothing new and will be well familiar to users who have simulated processes of the type $X+\text{jet}$ using, for example, PYTHIA and HERWIG. In the `POWHEG BOX` framework, as in PYTHIA and HERWIG, this amounts to a lower bound on the p_T of the $2 \to 2$ underlying Born configuration, from which radiation is subsequently generated. The value of this cut is set by `bornktmin` in `powheg.input`.

Since this cut has no physical origins and is merely a crude device to avoid the singular regions of the leading order configurations, it is essential that the observables which one is analysing include sufficiently high p_T / E_T cuts that they are unaffected by a decrease in it. Re-iterating, it is fundamental that the observables one is studying exhibit little or no sensitivity to the 'missing', unpopulated, low p_T region of phase space below the cut. This requirement must be more rigidly enforced in the `POWHEG BOX` case than PYTHIA or HERWIG since the latter only aim at leading order accuracy.

To give some idea of what may be considered appropriate, in our related article, `arXiv:1012.3380v1`, we were able to reliably use a generation cut, `bornktmin`, of 10 GeV to study observables for which the two leading jets were above as little as 20 GeV in transverse momentum (at the Tevatron), while a `bornktmin` cut of 20 GeV proved adequate for the study of the same observables but with the two leading jets above 50 GeV (at the LHC). In terms of what constitutes 'best practice' we recommend that serious studies involve cross checks to verify that the dependence on `bornktmin` does not modify predictions by more than 1-2%.

7. Scale choices

In the `POWHEG` algorithm, each event is built by first producing what is referred to as an underlying Born configuration, here a QCD $2 \to 2$ scattering, before proceeding to generate the hardest branching in the event. We have elected to use the p_T of the underlying-Born configuration as the renormalization and factorization scale in obtaining the fixed-order NLO predictions, effectively resumming virtual corrections to the associated t-channel gluon propagator — see the subroutine `set_fac_ren_scales` in `Dijet/Born_phsp.f`. This

[4] For more details please see `http://projects.hepforge.org/lhapdf/configure`

same scale choice is used in generating the underlying Born kinematics, of the POWHEG events, while the component of the hardest-emission cross section, responsible for the subsequent generation of the hardest branching kinematics, uses the transverse momentum of the branching, both in the evaluation of the strong coupling constant and the PDF.

References

[1] P. Nason, "A new method for combining NLO QCD with shower Monte Carlo algorithms," JHEP **0411** (2004) 040 [arXiv:hep-ph/0409146].

[2] S. Frixione, P. Nason and C. Oleari, "Matching NLO QCD computations with Parton Shower simulations: the POWHEG method," JHEP **0711** (2007) 070 [arXiv:0709.2092 [hep-ph]].

[3] S. Alioli, P. Nason, C. Oleari and E. Re, "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX," [arXiv:1002.2581 [hep-ph]].

[4] S. Alioli, K. Hamilton, P. Nason, C. Oleari and E. Re, "Jet pair production in POWHEG," arXiv:1012.3380 [hep-ph].