

Manual for electroweak W^+W^+jj production in the POWHEG BOX

The `VBF_Wp_Wp` program is an implementation of the electroweak W^+W^+jj production cross section within the POWHEG BOX framework. It complements the `Wp_Wp-j-j` program, which provides the QCD-induced contributions to W^+W^+jj production in hadronic collisions.

This document describes the input parameters that are specific to the implementation of the EW channel. Note that the recommended settings are very similar to those for the QCD production mode. The parameters that are common to all POWHEG BOX implementations are given in the `manual-BOX.pdf` document, in the `POWHEG-BOX/Docs` directory.

If you use this program, please quote Refs. [1–3].

Running the program

Download the POWHEG BOX, following the instructions at the web site

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

and do to

```
$ cd POWHEG-BOX/VBF_Wp_Wp
```

Running is most conveniently done in a separate directory, for instance do

```
$ mkdir testrun
```

The directory must contain the `powheg.input` file and, for parallel running, a `pwgseeds.dat` file (see `manual-BOX.pdf` and `Manyseeds.pdf`).

Before compiling make sure that:

- `fastjet` is installed and `fastjet-config` is in the path,
- `lhapdf` is installed and `lhapdf-config` is in the path,
- `gfortran` is in the path,
- `gfortran` libraries are in the environment variable `LD_LIBRARY_PATH`.

If `LHAPDF` or `fastjet` are not installed, it is still possible to run the code using a dummy analysis routine and built-in PDFs, see the `Makefile` in `VBF_Wp_Wp`. If you use `LHAPDF`, make sure that you have a version compatible with `gfortran`, and insert the appropriate path for `LHAPDFCONFIG` in the `Makefile`.

After compiling, enter the `testrun` directory:

```
$ cd testrun
```

A `powheg.input` and a `pwgseeds.dat` file are present there. When executing

```
$../pwhg_main
```

the program will ask you to

```
enter which seed
```

The program requires you to enter an index that specifies the line number in the `pwgseeds.dat` file where the seed of the random number generator to be used for the run is stored. All results generated by the run will be stored in files named `*-[index].*`. When running on parallel CPUs, make sure that each parallel run has a different index.

The program is most conveniently run in several steps. Each new step requires the completion of the previous step.

The timings given in the following refer to the program compiled with `gfortran` and run on a cluster with 2.7 GHz Opteron processors.

Step 1

Consists of a single run to generate the grid. At this point the user has to decide whether the weak bosons are to be generated on-shell or off-shell, distributed according to a Breit-Wigner distribution. By default the parameter `zerowidth` in `powheg.input` is set to zero and off-shell weak bosons are generated. If instead the user sets `zerowidth 1`, the W^+ bosons are produced on-shell. The user can also select the decay modes of the weak bosons by assigning appropriate PDG code to the parameters `vdecaymodew1` and `vdecaymodew2` in `powheg.input`. Note that only leptonic decays are supported. The template analysis file `pwhg_analysis.f` needed in subsequent steps of the analysis is designed for the $e^+\nu_e e^+\nu_e$ mode.

We recommend to generate the grid with the option `fakevirt 1` in `powheg.input`. When using this option, the virtual contribution is replaced by a fake one proportional to the Born term. This speeds up the generation of the grid tremendously.

One needs at least 1000000 events and 10 iterations. Set the following tokens in the `powheg.input` file:

```
ncall1 1000000
```

```
itmx1 10
```

```
ncall2 0
```

```
fakevirt 1
```

Run the program via

```
$../pwhg_main
```

When prompted

```
enter which seed
```

enter 1 or any other valid seed number.

This step takes roughly 48 hours of CPU. By setting

```
ncall2 0
```

in the `powheg.input` file the program stops after the compilation of this step.

Step 2

Runs in parallel can be performed now. Comment out the `fakevirt` token from `powheg.input`.

The runs must be performed where the previously generated grid is, unless the grid is moved to a different directory, this is the same directory where **Step 1** was performed.

The integration and upper bound for the generation of `btilde` can be performed with 50-100 runs with 2500-5000 calls each. Set for instance

```
ncall2 5000
```

```
itmx2 1
```

in `powheg.input`.

Folding numbers that are appropriate for runs at LHC energy are:

```
foldcsi 5 ! number of folds on csi integration
```

```
foldy 5 ! number of folds on y integration
```

```
foldphi 10 ! number of folds on phi integration
```

Time is about 3 hours of CPU for each run with `ncall2=5000`.

Setting

```
nubound 0
```

in `powheg.input` causes the program to stop after the completion of this step.

In order to run, for example, 100 processes in parallel do:

```
$../pwhg_main
```

When prompted

```
enter which seed
```

enter an index for each run (from 1 to 100). The `pwgseeds.dat` must contain at least 100 lines, each with a different seed.

Step 3

Also this step can be run in parallel. The number of processes must not be larger than the one used in the previous step. The runs are best performed in the same directory where all processes

of `step 2` have been completed. Setting

```
numevts 0
```

```
nubound 100000
```

takes roughly 2.5 hours per process.

The setting `numevts 0` causes the program to stop after completion of this step. The parallel execution of the program is performed as in the previous step.

Step 4

Set `numevts` to the number of events you want to generate per process, for example

```
numevts 1000
```

and run in parallel. The number of processes must not be larger than the one used in the previous step. Generating the specified number of events takes about 7 hours per process.

At this point, files of the form `pwgevents-[index].lhe` are present in the run directory.

Count the events:

```
$ grep '/event' pwgevents-*.lhe | wc
```

The events can be merged into a single event file by

```
cat pwgevents-*.lhe | grep -v '/LesHouchesEvents' > pwgevents.lhe
```

Analyzing the events

It should be easy to feed the `*.lhe` events into a generic shower Monte Carlo program, within the analysis framework of each experiment. We also provide a sample analysis that computes several histograms and stores them in topdrawer output files.

Doing (from the `VBF_Wp_Wp` directory)

```
$ make lhef_analysis
```

```
$ cd testrun
```

```
../lhef_analysis
```

analyses the bare POWHEG BOX output, creating the topdrawer file `LHEF_analysis.top`. The targets `main-HERWIG-lhef` and `main-PYTHIA-lhef` are instead used to perform the analysis on events fully showered using HERWIG or PYTHIA.

References

- [1] B. Jäger, C. Oleari, D. Zeppenfeld, *Next-to-leading order QCD corrections to W^+W^+jj and W^-W^-jj production via weak-boson fusion*, Phys. Rev. **D80** (2009) 034022. [arXiv:0907.0580 [hep-ph]].

- [2] B. Jäger, G. Zanderighi, *NLO corrections to electroweak and QCD production of W^+W^+ plus two jets in the POWHEG BOX*, to appear.
- [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*, JHEP **1006** (2010) 043. [arXiv:1002.2581 [hep-ph]].