Manual for electroweak $WZjj$ production in the POWHEG BOX V2

The VBF $WZjj$ program is an implementation of the electroweak (EW) W^+Z production cross section within the POWHEG BOX framework, with the Z boson decaying to charged leptons, and the W^+ boson decaying to a charged lepton and the associated neutrino.

If you use this program, please quote Refs. [1–3]. The code is taking full advantage of the version 2 of the POWHEG BOX in order to handle the computational complexity of the process. It was designed for and tested with svn 3596 of the POWHEG BOX V2. We successfully compiled and run the code on Ubuntu Linux 14.04 with GNU Fortran 5.5.0.

This document describes the input parameters that are specific to the implementation of the EW W^+Z channel. The parameters that are common to all POWHEG BOX implementations are given in the manual-BOX.pdf document, in the POWHEG-BOX-V2/Docs directory.

The flags vdecaymodeZ and vdecaymodeW, respectively, in the powheg.input file specify the leptonic decay products of the Z boson (11: e^+e^-; 13: $\mu^+\mu^-$) and the charged lepton that the W^+ boson decays into (-11: $e^+\nu_e$; -13: $\mu^+\nu_\mu$). Semileptonic and fully hadronic decay modes are not supported. In addition, the user can set the values of the masses and widths of the Higgs and the W and Z bosons via the parameters hmass, hwidth, wmass, wwidth, zmass, zwidth.

It is possible to restrict the process to the production of on-shell bosons by setting the parameter zerowidth to 1.

The flag channel_type allows the user to only evaluate single quark flavour channels (see the file init_processes.f for a description of the channels). This option is designated for debugging and we recommend setting it to the default value of 0 for normal usage.

The program allows the use of a Born suppression factor in order to increase the efficiency of phase-space integration which can be activated by setting the option bornsuppfact to 1. Moreover, it is possible to employ a generation cut on the invariant mass of the same-type lepton pair to remove contributions with a quasi on-shell photon, leading to divergences. The value of this cut (expressed in GeV) is set with the parameter mll_gencut.

The renormalization and factorization scales can be fixed to $\mu_0 = (m_W + m_Z)/2$ by setting the parameter runningscales to zero. For using running factorization and renormalization scales instead, the parameter runningscales can be set to either of the following values:

- 1 for $\mu_0 = \sum_{i=1}^{n_{\text{final}}}(p_{T,i} + \sqrt{m_W^2 + m_Z^2 + p_T^2(W) + p_T^2(Z)})/2$,
- 2 for $\mu_0 = \sqrt{Q_1 + Q_2}$, where Q_1 and Q_2 denote the momentum transfer on the upper and lower quark line of the underlying Born configuration, respectively, and

- 3 for $\mu_0 = \sqrt{p_{T,q_1} + p_{T,q_2}}$, with the q_j denoting the final-state partons of the underlying Born configuration.

Running the program

Download the `POWHEG BOX V2`, following the instructions at the web site

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

and go to the relevant directory by typing

```
$ cd POWHEG-BOX-V2/VBF_WZ
```

Running is most conveniently done in a separate directory. Together with the code, we provide the directory `test` that contains a sample input file.

For your runs, generate your own directory, for instance by doing

```
$ 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`, `ifort` or `g77` is in the path, and the appropriate libraries are in the environment variable `LD_LIBRARY_PATH`.

After compiling, enter the testrun directory:

```
$ cd testrun
```

and perform all your runs there.

We recommend running the program in a parallel mode in several consecutive steps, with the following common settings in `powheg.input`:

```
foldcsi 1
foldy 1
foldphi 1
withdamp 1
manyseeds 1
```

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.

Step 1

In this first step, the importance sampling grids are generated. Make sure that the relevant technical parameters in `powheg.input` are set to the following values:

```
xgriditeration 1
parallelstage 1
```

We recommend to generate the grids 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 grids.

For a default setup one needs 100-150 runs with the number of calls set by

```
ncall1 100000
```

for each. 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.

The completion of the first iteration of the grid production takes roughly four hours of CPU per job. Once all jobs of the first iteration are completed, the grids are refined in further iterations. We recommend to perform at least a second iteration, setting

```
xgriditeration 2
parallelstage 1
```

and rerunning the program as in the first iteration. If more iterations are needed, the value of `xgriditeration` has to be adapted accordingly. Each iteration takes roughly the same amount of CPU time as the first one.

Step 2

In order to proceed, perform subsequent runs in the directory where the previously generated grids are located. Comment out the `fakevirt` token from `powheg.input`.

The integration for the generation of btilde can be performed with 100-150 runs with 200000 calls each. In `powheg.input` set, for instance:

```
ncall2 200000
itmx2 1
parallelstage 2
```

Run jobs in parallel, in the same way as explained for `step 1` above.

The program automatically chooses the default analysis routine (`pwhg_analysis_lep`), which provides basic differential histograms. Users are free to replace this analysis routine with their own ones, depending on their needs.

Upon the completion of `step 2`, for each parallel run a file `pwg-*-NLO.top` is generated (where the * denotes the integer identifier of the run). These files contain the histograms defined in `pwhg_analysis_lep.f` at NLO-QCD accuracy, if the variable `bornonly` is set to zero in `powheg.input`. Setting `bornonly` to 1 yields the respective LO results.

Step 3

Also this step can be run in parallel. We recommend 100-150 jobs with the following settings:

```
nubound 100000
parallelstage 3
```

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 job, for example

```
numevts 1000
```

and run in parallel. Generating the specified number of events can take several days, depending on the setup.

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
```

At the end of the generated file `pwgevents.lhe`, add a line containing the expression

```
</LesHouchesEvents>
```

Analyzing the events

It is straightforward to feed the combined event file `pwgevents.lhe` (or each individual file `pwgevents-*.lhe`) 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 gnuplot output files.

Doing (from the `VBF_WZ` directory)

```
$ make lhef_analysis
```

```
$ cd testrun

../lhef_analysis
```

analyses the bare POWHEG BOX output, creating the gnuplot file pwgLHEF_analysis.top. The targets main-PYTHIA-lhef or main-PYTHIA8-lhef are instead used to perform the analysis on events fully showered using PYTHIA 6 or PYTHIA 8. The settings of the Monte Carlo can be modified by editing the file setup-PYTHIA-lhef.f. For using HERWIG7.1, we refer the interested reader to the information provided within the folder of the same name in the POWHEG BOX V2 directory.

We remind the user that it is possible to change factorization and renormalization scales and the parton distribution functions a posteriori through a reweighting procedure, provided the events have been generated using the flag storeinfo_rwgt 1. The reweighting can be done by running pwhg_main using the flag compute_rwgt 1.

References

[1] G. Bozzi, B. Jäger, C. Oleari and D. Zeppenfeld, *Next-to-leading order QCD corrections to W+ Z and W- Z production via vector-boson fusion*, Phys. Rev. D **75** (2007) 073004 [hep-ph/0701105].

[2] B. Jäger, A. Karlberg, J. Scheller, *Parton-shower effects in electroweak $WZjj$ production at the next-to-leading order of QCD*, arXiv:1812.05118 [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*, JHEP **1006** (2010) 043 [arXiv:1002.2581 [hep-ph]].