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

The VBF_Wp_Wm program is an implementation of the electroweak W^+W^-jj production cross section within the POWHEG BOX framework.

If you use this program, please quote Refs. [1–3]. This code includes a new version of some POWHEG BOX files that were provided to us by Paolo Nason and that will soon be released as part of the Version 2 of the POWHEG BOX.

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

The decay mode of the W^+ and W^- bosons can be fixed by setting the parameters vdecaymodeWp and vdecaymodeWm, respectively, in the powheg.input file. For the leptonic decay modes, these flags specify the charged leptons the bosons decay to (11: e^-; 13: μ^-; 15: τ^-; -11: e^+; -13: μ^+; -15: τ^+). The same applies for the leptonically decaying gauge boson in the semi-leptonic decay modes. In that case for the hadronically decaying W the user can choose between decay into a specific quark type (1: $W^- \to d\bar{u}$; 3: $W^- \to s\bar{c}$; -1 $W^+ \to u\bar{d}$; -3: $W^+ \to c\bar{s}$) and the sum of the light quark decay channls (7: $W^- \to d\bar{u} + s\bar{c}$; -7: $W^+ \to u\bar{d} + c\bar{s}$). Depending on the decay mode selected by the user, the program automatically chooses an appropriate sample analysis. The user can replace these sample analyses with own ones, see pwhg_analysis_all.f.

In addition, the user can set the values of the masses and widths of the Higgs and the W bosons via the parameters hmass, hwidth, wmass, wwidth.

Note that in the presence of a very sharp resonance, as is the case in a scenario with a light Higgs boson with a mass below 200 GeV, the resonant Higgs contribution has to be evaluated separately from the WW continuum, as described in Ref. [2]. In this case all the steps described below have to be performed for each of these two contributions separately in two distinct working directories, setting

```
ww_res_type 1
```

for the WW continuum and

```
ww_res_type 2
```

for the Higgs resonance contribution. After all the runs have been performed for each case, the results have to be added. This can be done, e.g., with the help of the file addplots.f in the directory auxiliary.

If the Higgs is heavy and broad, such a splitting is not necessary, and all contributions can be evaluated at the same time, setting

```
ww_res_type 0
```

For convenience, we also provide the setup for running the code in a kinematic regime with highly boosted jets. To select this mode, set

```
ww_res_type 1
fat_jet 1
```

in `powheg.input`. When the `fat_jet` flag is set to 1, generation cuts on the decay leptons are used in the phase space and an appropriate analysis routine is selected.

Running the program

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

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

and go to

```
$ cd POWHEG-BOX/VBF_Wp_Wm
```

Running is most conveniently done in a separate directory. Together with the code, we provide the directories `runs_lep`, `runs_slm_hh`, and `runs_slp_fat` that contain sample input files for setups with fully leptonic decays in the presence of a light Higgs boson, semileptonic decays of a heavy Higgs boson, and for running the code in the kinematic regime of a highly boosted fat jet.

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`.

If `LHAPDF` or `fastjet` are not installed, the code can still be run using a dummy analysis routine and built-in PDFs, see the `Makefile` in `VBF_Wp_Wm`.

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

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 50-100 runs with the number of events generated set by

```
ncall1 50000
```

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 half an hour 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 and upper bound for the generation of btilde can be performed with 50-100 runs with 15000-25000 calls each. In `powheg.input` set, for instance:

```
ncall2 25000
itmx2 1
parallelstage 2
```

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

Time is about 2.5 hours of CPU for each run with `ncall2=25000`.

Depending on the decay mode selected, the program automatically chooses an appropriate analysis routine (`pwhg_analysis_lep,pwhg_analysis_slp,pwhg_analysis_slm`), unless the option `ANALYSIS` has been set to `none` in the `Makefile` when the code was compiled. The user is free to replace these analysis routines with her own ones, depending on her 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.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. In either case, the individual results of the parallel runs can be combined with the help of the `combineplots.f` file contained in the `auxiliary` directory. To this end, just compile the file by typing, e.g.,

```
$ gfortran combineplots.f
```

and run the resulting executable. The program will ask for the the number of `pwg-*-NLO.top` files in the directory and the the number of calls per run which are set by `ncall2` in `powheg.input`. Finally, enter the integer identifier of the first file. The program will then generate the files `combinedNLO.top` and `combinedNLO-gnu.top` which contain the combined histograms in two different formats (topdraw or gnuplot friendly).

Step 3

Also this step can be run in parallel. Setting

```
nubound 5000
parallelstage 3
```

takes roughly 5 minutes per process.

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 5000
```

and run in parallel. Generating the specified number of events takes about 5-10 hours per process, 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 topdrawer output files.

Doing (from the `VBF_Wp_Wm` directory)

```
$ make lhef_analysis
```

```
$ cd testrun
```

```
../lhef_analysis
```

analyses the bare `POWHEG BOX` output, creating the topdrawer file `LHEF_analysis.top`. The target `main-PYTHIA-lhef` is instead used to perform the analysis on events fully showered using `PYTHIA`. The settings of the Monte Carlo can be modified by editing the file `setup-PYTHIA-lhef.f`.

References

[1] B. Jäger, C. Oleari, D. Zeppenfeld, *Next-to-leading order QCD corrections to W^+W^-jj production via vector-boson fusion*, JHEP **0607** (2006) 015 [hep-ph/0603177].

[2] B. Jäger, G. Zanderighi, *Electroweak W^+W^-jj prodution at NLO in QCD matched with parton shower in the POWHEG BOX*, arXiv:1301.1695.

[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]].