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

The VBF_Z_Z program is an implementation of the electroweak (EW) $ZZjj$ production cross section within the POWHEG BOX framework, with the Z bosons decaying to charged leptons, neutrinos, or quarks.

If you use this program, please quote Refs. [1–3]. The code is taking full advantage of the new Version 2 of the POWHEG BOX in order handle the computational complexity of the process.

This document describes the input parameters that are specific to the implementation of the EW $ZZjj$ 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 decay mode of the two Z bosons can be fixed by setting the parameters vdecaymodeZ1 and vdecaymodeZ2, respectively, in the powheg.input file. For the leptonic decay modes, these flags specify the lepton pair which each boson decays into (11: e^+e^-; 12; $\bar{\nu}_e\nu_e$; 13: $\mu^+\mu^-$; 14: $\bar{\nu}_\mu\nu_\mu$). The same applies for the leptonically decaying gauge boson in the semi-leptonic decay modes. In that case for the hadronically decaying Z the user can choose between decay into a specific quark type (1: $Z \to d\bar{d}$; 3: $Z \to s\bar{s}$; 2: $Z \to u\bar{u}$; 4: $Z \to c\bar{c}$) and the sum of the up - or down-type quark channel (7: $Z \to d\bar{d} + s\bar{s}$; 8: $Z \to u\bar{u} + c\bar{c}$). 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 and Z bosons via the parameters hmass, hwidth, wmass, wwidth, zmass, zwidth.

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 ZZ 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

```
zz_res_type 1
```

for the ZZ continuum and

```
zz_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 mergedata.f in the directory plot-aux.

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

```
zz_res_type 0
```

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_Z_Z
```

Running is most conveniently done in a separate directory. Together with the code, we provide the directory runs that contains a sample input file for a setup with fully leptonic decays in the presence of a light Higgs boson.

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.

The timings given in the following refer to the program compiled with ifort and run on a cluster with 2.4 GHz Xeon 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 2
foldy 2
foldphi 2
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 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 50-100 runs with 50000-100000 calls each. In `powheg.input` set, for instance:

```
ncall2 100000
```

```
itmx2 1
parallelstage 2
```

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

Time is about 48 hours of CPU for each run with `ncall2=100000`.

Depending on the decay mode selected, the program automatically chooses an appropriate analysis routine (`pwhg_analysis_lep,pwhg_analysis_lnu,pwhg_analysis_slp`), 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 `mergedata.f` file contained in the `plot-aux` directory. To this end, just compile the file by typing, e.g.,

```
$ gfortran mergedata.f -o mergedata
```

and run the resulting executable in your run directory `mergedata 1 *NLO.top`. The program expects a number between 1 and 5 and a list of files to merge. If no number or no list is specified, the user will be prompted by the program to enter them. Running mergedata without any arguments will also explain what the 5 different options are. `mergedata` will combine the histograms into a file called `fort.12`. We provide a file `plot-aux/genplots.sh` and `plot-aux/gnuplotsplit.gp` which can be used to plot the resulting histograms. First run `genplots.sh`

```
./genplots.sh file nameoutput
```

or if a comparison between two different runs is wanted

```
./genplots.sh file1 file2 nameoutput
```

This will result in a file called `genplots.gp`. In the later case the file `plot-aux/pastegnudata.f` has to be compiled and either put in the working directory or in the users path. After `genplots.sh` has been run, the user can produce a set of .eps files with gnuplot running

```
./gnuplotsplit.gp genplots.gp
```

Step 3

Also this step can be run in parallel. Setting

```
nubound 5000
parallelstage 3
```

takes roughly 25 minutes per job.

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 5000
```

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_Z_Z` directory)

```
$ make lhef_analysis
```

```
$ cd testrun
```

```
../lhef_analysis
```

analyses the bare `POWHEG BOX` output, creating the topdrawer file `pwgLHEF_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`.

We remind 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`.

Enabling effective operator contributions

As described in some detail in Ref. [2], the `VBF_Z_Z` implementation in the `POWHEG BOX` accounts for physics beyond the Standard Model by means of an effective field theory approach with operators of dimension six that affect triple and quartic gauge boson vertices, thus giving rise to anomalous couplings. All files needed to run the code in this effective operator approach are found in the directory `anomalous`. As many more diagrams have to be evaluated when running

in this mode (even if the coefficients of the anomalous contributions are all set to zero), the code is about a factor of two slower than in the Standard Model mode.

We provide a script `anomalous.sh` which, upon execution, first creates a backup of the original Standard Model files in the `VBF_Z_Z` directory and puts them in a separate directory called `no_anomalous`. Then all files of the directory `anomalous` are copied to the main directory and compiled. The code can now be run using the executable `pwhg_main_anom`, performing the same fours steps as described above for the Standard Model mode. The script `no_anomalous.sh` that is created automatically during the preparation of the code in the effective operator approach can be used to restore the original Standard Model code.

The coefficients of the dimension-six operators are set in the file `powheg.input` through the five parameters `CWWWL2`, `CWL2`, `CBL2`, `CPWWWL2` and `CPWL2`, corresponding to the c_i/Λ^2 of Ref. [2], where $\Lambda = 1$ TeV. Hence the coefficients have to be specified in units of TeV^{-2}. For instance, if the user wants to set the coupling $c_{WWW}/\Lambda^2 = -3\ \text{TeV}^{-2}$ in the input file, she would set

```
CWWWl2      -3d0
```

The couplings are assumed real. Using complex couplings would require slight modifications of the file `anomalous/convert_coup.f`.

References

[1] B. Jäger, C. Oleari, D. Zeppenfeld, *Next-to-leading order QCD corrections to Z boson pair production via vector-boson fusion*, Phys. Rev. **D73** (2006) 113006. [arXiv:hep-ph/0604200].

[2] B. Jäger, A. Karlberg, G. Zanderighi, *Electroweak $ZZjj$ production in the Standard Model and beyond in the POWHEG-BOX V2*, arXiv:1312.3252 [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]].