Manual for electroweakino pair production in the POWHEG BOX V2

This manual summarizes the settings and input parameters that are specific to the implementation of electroweakino pair production at the LHC within the POWHEG BOX V2 framework. The parameters that are common to all POWHEG BOX V2 implementations are given in the documents manual-BOX.pdf and Manyseeds.pdf in the POWHEG-BOX-V2/Docs directory.

Electroweakino pair-production processes can be split into the sub-processes neutralino-pair, chargino-pair and neutralino-chargino production, which we put into the sub-folders neuIneuJ, chaIchaJ and neuIchaJ, respectively. This makes the main directory tidier and the organization of the code simpler. In the following we refer to the directory POWHEG-BOX-V2/weakinos/ as the main directory and to the three folders neuIneuJ, chaIchaJ and neuIchaJ within the main directory as the sub-folders.

If you use our program, please cite Refs. [1–4].

Compiling the program

To get started, compile the static libraries libdhelas3.a [5], liblooptools.a [6], and libSLHA.a [7] for the operating system you are using. To this end, call the configuration scripts in the main directory by simply typing

```
$ ./configure.sh
```

Afterwards you can compile the libraries by executing

```
$ make libdhelas3.a
$ make liblooptools.a
$ make libSLHA.a
```

or, in short,

```
$ make libs
```

If you want to use your own libraries, copy them into the directory ./Tools/ or provide paths to the libraries in the makefile.

The neutralino-pair, chargino-pair and neutralino-chargino production processes are implemented as three independent programs. The respective process-dependent routines can be found in the sub-folders neuIneuJ, chaIchaJ and neuIchaJ, which each contain a sample makefile. In order to generate the executables change into the respective sub-folder and run

```
$ make -j4 all.
```

Per default the programs require fastjet and lhapdf, so make sure that fastjet-config and lhapdf-config are set correctly in your path, otherwise you have to adapt the makefile.

To test the program with default parameters type

```
$ make clean-results && make -j4 do
```

in the desired process sub-folder which will clean preceding results, compile and run the main executable in the provided `testrun` directory. Use the flag `-j`n, where n is an integer, to compile on multiple cores. This will speed up the compilation significantly.

The following commands can be used in the sub-folders `neuIneuJ`, `chaIchaJ`, or `neuIchaJ` to clean the compiled executables and object files:

```
$ make clean
```

removes all object files in `./build`, which has no effect on the compiled program. The command

```
$ make clean-results
```

removes old results in `./testrun`, and

```
$ make clean-all
```

removes old results, the object files and the compiled programs. To clean the code thoroughly from within the main directory use

```
$ make clean
```

which will remove all object files, the libraries, and executables.

Important note for Mac OSX and some Linux users: In order to link the object files properly with newer compiler versions it might be advisable to recompile all your own libraries using the `-lstdc++` flag.

Precompiler Flags

In the current code version several C preprocessor (cpp) flags are implemented. The preprocessor runs in traditional mode for gfortran. Any restrictions of the file format, especially the limits on line length, apply for preprocessed output as well, so it might be advisable to use the `-ffree-line-length-none` or `-ffixed-line-length-none` options (activated by default). If you want to change a preprocessor flag, it is imperative to run

```
$ make clean
```

before recompiling the source code.

The flags `FORM_BORN`, `MAD_BORN`, `DR_I`, `DR_II`, `DSUB_I`, `DSUB_II` are mandatory, you should not change them unless you know what you do. Please refer to the Makefile for more details.

Model parameters

All parameters are read from a single slha-file in the run directories which is formated according to the SUSY Les Houches Accord (`SLHA`) [8, 9] and can be taken from standard SUSY spectrum generators. Runtime variables, such as integration points, number of events to generate, etc. have to be specified in `powheg.input`. If you want to change the Z-mass, Z-width or $\alpha_{\rm em}$, etc. you can do this in `powheg.input`, too.

Running the executables and helpful scripts

In order to run the generated executables you have to provide a `powheg.input` file, which contains the settings for different technical parameters, and a SLHA file containing the masses of the sparticles. Some sample files can be found in the sub-folders `testrun` and `testrun-clean` of the process directories `neuIneuJ`, `chaIchaJ` and `neuIchaJ`. Copy these files to a directory of your choice and adapt the parameters as desired. Then either execute `pwhg_main_ninj`, `pwhg_main_xixj`, or `pwhg_main_nixj` from within the folder which contains the input files.

We have added several helpful shell scripts to the whole package, which can be used to generate results or clean old runs. The most important one is `./Scripts/runparallel.sh`, which is used to run the `POWHEG BOX V2` executables fully automated in parallel mode. Type `./runparallel.sh -h` to get an overview of the full functionality of the script. This script uses standard shell commands, so it should work on every ∗NIX-based operating system.

A few examples how to use the `./runparallel.sh` script are given in the following.

- Run the executable `pwhg_main_nixj` in the directory `testrun` on 4 cores (default) and clean preceding results:

```
$ ./runparallel.sh -c -d testrun -e pwhg_main_nixj
```

- Run the executable `pwhg_main_nixj` in `testrun` on 4 cores and overwrite some parameters for integration in `powheg.input`:

```
$ ./runparallel.sh -c -d testrun -e pwhg_main_nixj --itmx1 4 \
 --itmx2 4 --itmx1osres 6 --itmx2osres 8 --ncall1 2000 \
 --ncall2 2000 --ncall1osres 20000 --ncall2osres 20000
```

- Copy the folder `testrun_clean`, rename it to `run_wevents` and generate events (`nubound` and `nevents` in `powheg.input` must be greater than zero):

```
$ ./runparallel.sh -g -c -e pwhg_main_nixj -d run_wevents \
 --genevents > log_wevents
```

- Generate a total of 20Mio. events with high precision on a cluster with the MOAB submission system in the directory `run_mSUGRA_n2x1+` distributed to 50 jobs. The input file `input_mSUGRA_1410.4999.slha` and the final states $\tilde{\chi}_2^0$ and $\tilde{\chi}_1^+$ are specified:

```
$ nohup ./runparallel.sh -g -c -e pwhg_main_nixj \
 -d run_mSUGRA_n2x1+ -p 50 --fin1 1000023 --fin2 1000024 \
 --slha input_mSUGRA_1410.4999.slha --ncall1 200000 \
 --ncall2 300000 --ncall1osres 20000 --ncall2osres 30000 \
 --nevents 400000 --nubound 400000 --genevents \
 --usemsub --offset 0 > log_run_mSUGRA_n2x1+ &
```

- Same for LO event generation:

```
$ nohup ./runparallel.sh -g -c -e pwhg_main_nixj \
 -d runLO_mSUGRA_n2x1+ -p 4 --fin1 1000023 --fin2 1000024 \
 --slha input_mSUGRA_1410.4999.slha --ncall1 200000 \
 --ncall2 300000 --ncall1osres 20000 --ncall2osres 30000 \
 --lopdf 90400 --nevents 5000000 --nubound 5000000 --genevents \
 --merge > log_runLO_mSUGRA_n2x1+ &
```

To merge the event files, the script `./Scripts/merge.sh` could be used. If you want to calculate results for a different factorization or renormalization scale, but you don't want to generate new events, you can use your old events and reweight them with the script `./Scripts/reweight.sh`.

Analyzing the events

It is straightforward to feed the combined event 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

```
$ ../lhef_analysis
```

analyzes the bare `POWHEG BOX` output and writes histograms to the file `pwgLHEF_analysis.top`.

Parton showers, decays, and further effects can be simulated when the events are handed over to `PYTHIA` [10] by

```
$ ../main-PYTHIA-lhef
```

The corresponding histograms can be found in `pwgPOWHEG+PYTHIA-output.top`.

Various settings of `PYTHIA` can be modified by editing the file `setup-PYTHIA-lhef.F` prior to compilation. `PYTHIA` generates the decays according to the masses provided in the previous mentioned slha-file.

Additional notes

In order to implement the on-shell subtraction scheme to regularize divergences in the real matrix elements (refer to [1] for more details), we had to make several modifications to a few files provided by the `POWHEG BOX V2`. Since the `POWHEG BOX V2` is constantly being updated and changes to important routines could be made in the future, we are not able to guarantee the full functionality of our code with newer releases of the `POWHEG BOX V2`. We have built our code on the `POWHEG BOX V2` svn-revision 3154. Within this revision our code was tested extensively, and compiling and running the code should work without problems. In case non-circumventable problems during compilation with a newer revision occur, you can always revert the `POWHEG BOX V2` to an older revision with the command

```
$ svn update -r 3154
```

References

[1] J. Baglio, B. Jäger, M. Kesenheimer, *Electroweakino pair production at the LHC: NLO SUSY-QCD corrections and parton-shower effects*, arXiv:1605.06509 [hep-ph].

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

[3] 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.

[4] 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.

[5] H. Murayama, I. Watanabe and K. Hagiwara, *HELAS: HELicity amplitude subroutines for Feynman diagram evaluations*, KEK-91-11 (1992).

[6] T. Hahn and M. Perez-Victoria, *Automatized one loop calculations in four-dimensions and D-dimensions*, Comput. Phys. Commun. **118**, 153 (1999), [hep-ph/9807565].

[7] T. Hahn, *SUSY Les Houches Accord 2 I/O made easy*, Comput. Phys. Commun. **180**, 1681 (2009), [hep-ph/0605049].

[8] P. Z. Skands *et al.*, *SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators*, JHEP **0407** (2004) 036 [hep-ph/0311123].

[9] B. C. Allanach *et al.*, *SUSY Les Houches Accord 2*, Comput. Phys. Commun. **180** (2009) 8 [arXiv:0801.0045 [hep-ph]].

[10] T. Sjostrand, S. Mrenna, P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, JHEP **0605** (2006) 026 [hep-ph/0603175].