proVBFH v.1.2.0 manual

August 12, 2019

This manual provides a short documentation for the proVBFH code.

Contents

1 Credits

The proVBFH program was developed by Matteo Cacciari, Frédéric Dreyer, Alexander Karlberg, Gavin Salam and Giulia Zanderighi and is based on

- M. Cacciari, F. A. Dreyer, A. Karlberg, G. P. Salam and G. Zanderighi, Phys. Rev. Lett. **115** (2015) no.8, 082002 [arXiv:1506.02660 [hep-ph]],
- F. A. Dreyer and A. Karlberg, Phys. Rev. Lett. **117** (2016) no.7, 072001 [arXiv:1606.00840 [hep-ph]].

The code itself relies heavily on the machinery of the POWHEG-BOX, the implementation of VBF Hjjj production in the POWHEG-BOX, and the VBF phase as implemented in the VBF process in the POWHEG-BOX. We therefore suggest that the following three publications be cited in addition to the above two when the program is being used to produce scientific results:

- S. Alioli, P. Nason, C. Oleari and E. Re, JHEP **1006** (2010) 043 [arXiv:1002.2581 [hep-ph]].
- P. Nason and C. Oleari, JHEP **1002** (2010) 037 [arXiv:0911.5299 [hep-ph]].
- B. Jäger, F. Schissler and D. Zeppenfeld, JHEP **1407** (2014) 125 [arXiv:1405.6950 [hep-ph]].

When using results at next-to-next-to-next-to-leading order (N^3LO), the original references on third order coefficient functions should be cited as well

- A. Vogt, S. Moch and J. A. M. Vermaseren, Nucl. Phys. B **691** (2004) 129 [hep-ph/0404111].
- S. Moch, J. A. M. Vermaseren and A. Vogt, Phys. Lett. B **606** (2005) 123 [hep-ph/041.2.0].
- J. A. M. Vermaseren, A. Vogt and S. Moch, Nucl. Phys. B **724** (2005) 3 [hep-ph/0504242].
- S. Moch, M. Rogal and A. Vogt, Nucl. Phys. B **790** (2008) 317 [arXiv:0708.3731 [hep-ph]].

Finally, when evaluating the non-factorizable NNLO corrections, the following papers should be cited

- F. A. Dreyer and A. Karlberg, [arXiv:19MM.XXXXX [hep-ph]].
- T. Liu, K. Melnikov and A. A. Penin, [arXiv:1906.10899 [hep-ph]].
- S. Buehler and C. Duhr, Comput. Phys. Commun. **185** (2014) 2703 doi:10.1016/j.cpc.2014.05.022 [arXiv:1106.5739 [hep-ph]].

1.1 Dependencies

To run `proVBFH`, you will need an installation of the following packages:

- `hoppet/struct-func-devel`: `http://hoppet.hepforge.org/`. Note that it is specifically the struct-func-devel branch of `hoppet` that is required. It can be downloaded using:

  ```
  svn checkout https://svn.hepforge.org/hoppetsvn/branches/struct-func-devel/
  ```

- `LHAPDF`: `http://lhapdf.hepforge.org/`.
- `FastJet`: `http://fastjet.fr/`.
- `gfortran` version 4.6 or later.

The corresponding references for the first three of these packages should also be cited when using `proVBFH`.

2 Installation

With the above dependencies installed, `proVBFH` can be compiled by going to the base directory and running

```
./configure [options]
make
```

Available options in the `configure` script can be accessed through the `--help` or `-h` argument, and are also described in the `INSTALL` file.

Along with the main `proVBFH` executable, a program to combine runs, `combine_runs`, will be generated in the `aux` folder.

3 Setting up a run

proVBFH is partially based on the VBF_HJJJ process of the POWHEG-BOX and uses the same machinery to set up parallel runs. Users familiar with the POWHEG-BOX will therefore find running proVBFH straightforward. Given the complexity of the code it is necessary to run it in parallel and at the end of this section we will give some recommendations on how many parallel runs should be carried out. An example configuration for running proVBFH is given in the example/ folder. The three files necessary to run the code are

- pwgseeds.dat: contains a list of integers to be read by the program
- powheg.input: contains all the technical parameters of the run. They are details below.
- vbfnlo.input: contains a number of physical parameters. They are detailed below.

From this folder, the program can be started with

```
../proVBFH
```

The program will ask for an input corresponding to a line number in the file pwgseeds.dat.

When running the program at either NLO or NNLO There are two stages (controlled by the flag parallelstage in powheg.input) that one has to go through in order to compute the cross section. During the first stage the program will generate adaptive grids to be used during stage 2, where the cross section is computed. It is possible to run stage 1 through several iterations. The program reads the flag xgriditeration. If this is set equal to '1' the program will run starting from uniform grids. If it is set to 'n' the program will look for grids generated during iteration '$n-1$' and use them as the initial grids. When stage 2 is then invoked the program looks for the most recent grids. The output of stage 2 is a file with histograms called pwg-XXXX-(N)NLO.top where XXXX is the integer which was used to invoke the program.

At LO and N^{3}LO (inclusive) the program only runs in one stage since the generation of grids is very fast for Born kinematics. In this case the two flags described above are therefore not used by the program.

3.1 The Analysis

In the event where differential distributions are requested, the analysis will be taken from the file analysis/user_analysis.f. This file can be modified to remove or add any histograms from the output files. There is an intricate relationship between the analysis used for generating histograms and for the analysis used to impose phase space cuts. Therefore, while a user can freely add or remove histograms, one should be careful with changing the cuts. In user_analysis.f there are three routines which are also used by the phase space analysis. They are setup_vbf_cuts, buildjets and vbfcuts.

- setup_vbf_cuts: This routine reads the cuts from powheg.input
- buildjets: builds an array of up to 4 jets using R from the input card and anti-k_t. If one wants to define jets in a different way this routine can be modified.
- vbfcuts: takes the jets of the above built and decides whether or not they will pass the VBF cuts defined in setup_vbf_cuts. If they are not passed the analysis exits. Otherwise it proceeds to fill the histograms.

If one wants to create an analysis with multiple sets of cuts (for instance to have a set of loose/tight cuts or a central jet veto) these cuts have to be applied *after* the three routines.

3.2 A few recommendations

proVBFH requires a significant amount of computer power to produce smooth distributions at NNLO. Under typical VBF cuts we suggest the following technical input parameters:

- `ncall1 500000`
- `ncall2 5000000`
- `itmx2 3`

Stage 1 should go through 3 iterations on 200 machines. On a typical machine in 2017 each iteration should take $2-3$ hours. Stage 2 should be carried out on 2500 machines and will take $1-2$ days depending on how complicated the analysis is and how loose the VBF cuts are. Due to the way the combination of each run is carried out it is better to have a high number of independent runs rather than few runs with a very high `ncall2`.

With these settings the fiducial cross section will be computed with a permille statistical uncertainty and distributions will have percent level uncertainties.

3.3 Scale variations

One can vary the renormalisation and factorisation scales used in the program to estimate the residual theoretical uncertainties. We recommend a 3 point scale variation $\mu_R = \mu_F$ with $\mu_{R,F} = \{1/2, 1, 2\}\mu_0$. A seven point scale variation, where the scales are varied independently, does not lead to significantly larger uncertainty estimates.

3.4 Combining runs

We provide a script to combine independent runs. The script will automatically look for outliers in each bin and remove them. It can be invoked by

```
../aux/combine_runs <list of files>
```

where the list of files can be given using wildcards, eg `*NNLO*top`. This will produce a file called `total_distrib.top` containing the combined result.

4 Input parameters

4.1 `powheg.input` card

Most parameters are changed in the `powheg.input` file. Available options:

- `qcd_order`: 1: LO, 2: NLO, 3:NNLO, 4:N3LO (only for inclusive results).
- `inclusive_only`: 0: Computes the cross section fully differentially in the jets. 1: computes only the inclusive VBF cross section differentially in the Higgs momentum. (default: 0)
- `ebeam1`: energy of beam 1 in GeV.
- `ebeam2`: energy of beam 2 in GeV.

- `lhans1`: pdf set for hadron 1 (LHA numbering).
- `lhans2`: pdf set for hadron 2 (LHA numbering).
- `renscfact`: renormalisation scale factor: muren = muref * renscfact.
- `facscfact`: factorisation scale factor: mufact = muref * facscfact.
- `nonfact`: 0: compute factorisable cross-section, 1: compute non-factorisable coefficient. (Requires qcd_order=1)
- `higgsbreitwigner`: 0: Narrow width, 1: Breit Wigner for Higgs.
- `higgsmasswindow`: How many widths we integrate around the Breit Wigner peak.
- `runningscales`: 0: m_h. 1: scale of 1506.02660. 2: Q_1 on the upper line and Q_2 on the lower line. (default: 0, i.e. no running scales)
- `ncall1`: number of calls for initializing the integration grid.
- `ncall2`: number of calls for computing the integral.
- `itmx1`: number of iterations for initializing the integration grid.
- `itmx2`: number of iterations for computing the integral .
- `xgriditeration`: identifier for grid generation.
- `parallelstage`: 1: generate grids. 2: compute integral.
- `fakevirt`: Useful for generating stage 1 grids faster as it uses the Born instead of the virt which is faster.
- `phspcuts`: (1:on ; 0: off (default)) Turns on/off analysis cuts at the phasespace generation stage. Significantly reduces run time. Turning these off will make the integration unstable.

4.1.1 Analysis cuts

- `ptalljetmin`: Minimum p_t for all jets.
- `yjetmax`: Rapidity acceptance for all jets.
- `Rsep_jjmin`: R to be used in jet algorithm.
- `ptjetmin`: Minimum p_t for the two tagging jets.
- `mjjmin`: Minimum invariant mass for the tagging jets.
- `deltay_jjmin`: Rapidity separation between tagging jets.
- `jet_opphem`: (0/1) require the tagging jets in opposite detector hemispheres.

4.1.2 Advanced settings

Several additional options are present in the code and are required by the POWHEG-BOX. In principle, the user should not modify these settings as doing so can risk producing useless results. We detail the most important ones here, additional settings can be found in the POWHEG-BOX documentation.

- `iseed`: initialize random number sequence.
- `ih1`: hadron 1 (1 for protons, -1 for antiprotons).
- `ih2`: hadron 2 (1 for protons, -1 for antiprotons).
- `maxseeds`: Maximum number of seeds that powheg can run with.
- `manyseeds`: Used to perform multiple runs with different random seeds in the same directory. If set to 1, the program asks for an integer j; The file pwgseeds.dat at line j is read, and the integer at line j is used to initialize the random sequence for the generation of the event.

4.2 `vbfnlo.input` card

- `HMASS`: Higgs mass.
- `TOPMASS`: Top mass.
- `TAU_MASS`: Tau mass.
- `BOTTOMMASS`: Bottom Pole mass.
- `CHARMMASS`: Charm Pole mass.
- `FERMI_CONST`: Fermi Constant.
- `WMASS`: W mass.
- `ZMASS`: Z mass.
- `WWIDTH`: W width.
- `ZWIDTH`: Z width.
- `NFLAVOUR`: Number of quark flavours.
- `HWIDTH`: Higgs width.