Preprint typeset in JHEP style - PAPER VERSION

July 20, 2010
BOX @ r95

POWHEG-BOX user manual: Single-top s- and t-channel processes

Simone Alioli

Deutsches Elektronen-Synchrotron DESY
Platanenallee 6, D-15738 Zeuthen, Germany
E-mail: simone.alioli@desy.de

Paolo Nason

INFN, Sezione di Milano-Bicocca, Piazza della Scienza 3, 20126 Milan, Italy
E-mail: Paolo.Nason@mib.infn.it

Carlo Oleari

Università di Milano-Bicocca and INFN, Sezione di Milano-Bicocca
Piazza della Scienza 3, 20126 Milan, Italy
E-mail: Carlo.Oleari@mib.infn.it

Emanuele Re

Institute for Particle Physics Phenomenology, Department of Physics
University of Durham, Durham, DH1 3LE, UK
E-mail: emanuele.re@durham.ac.uk

Abstract: This note documents the use of the package POWHEG-BOX for the single-top s- and t-channel production processes. Results can be easily interfaced to shower Monte Carlo programs, in such a way that both NLO and shower accuracy are maintained.

Keywords: POWHEG, Shower Monte Carlo, NLO.

Contents

1. Introduction

The POWHEG-BOX program is a framework for implementing NLO calculations in Shower Monte Carlo programs according to the POWHEG method. An explation of the method and a discussion of how the code is organized can be found in refs. [1, 2, 3]. The code is distributed according to the "MCNET GUIDELINES for Event Generator Authors and Users" and can be found in

http://powhegbox.mib.infn.it/~nason/POWHEG.

In the following we will focus on the implementation of single-top (s- and t-channel) production, whose source files can be found in the POWHEG-BOX/ST_sch and POWHEG-BOX/ST_tch subdirectories.

This program is an implementation of the NLO cross section calculated in [4] in the POWHEG formalism of refs. [1, 2]. A detailed description of the implementation can be found in ref. [5]. Spin correlations of the top-quark decay products are included with a method

analogous to the one described in [6], and the relevant matrix elements for the full decayed amplitudes were obtained using MadGraph [7].[1]

In this note we give all the necessary information to run the program.

2. Installation

The latest version of program can be downloaded trough SVN

```
$ svn checkout [--revision n] --username anonymous --password anonymous
               svn://powhegbox.mib.infn.it/trunk/POWHEG-BOX
```

Previous revisions are available via the `[--revision n]` option and also as a tarred-gzipped file `POWHEG-BOX-***.tar.gz`. Once downloaded, the program can be installed with the following commands[2]

```
$ cd POWHEG-BOX/ST_sch
$ make <target>
```

where the choice of the `<target>` depends upon the way one wants to interface the program with a Shower Monte Carlo implementation. In order to correctly compile and run the program, the user is asked to have the LHAPDF library installed on his/her system and to take care to insert its correct search path in the `Makefile`, or, simply, to add the path of the `lhapdf-config` executable to the `$PATH` environmental variable. We remind that in case of linking against "dynamic" shared library, the correct LHAPDF library path should also be added to the `$LD_LIBRARY_PATH` environmental variable, otherwise run time errors may occur.

The default analysis routine that comes with the package relies on jet algorithms implemented according to the `FASTJET` library [8]. It is up to the user to correctly install it and to modify the `Makefile` accordingly. For most systems, adding the `fastjet-config` executable to the `$PATH` environmental variable is enough. We provide two `c++` to `fortran` wrappers to safely call the `SISCONE` and k_{T} jet algorithms, implemented in `FASTJET`, from a `fortran` environment.

The `Makefile` is set up to use the compiler `gfortran` on Linux platforms. If one wishes to use `g77`, one should change the appropriate lines in the `Makefile` (and in the dhelas `Makefile` present in the folder `madgraph/dhelas3.2`, which is executed automatically by the main `Makefile`).

[1] The only difference of the BOX implementation with respect to the one described in [5] is the treatment of finite width effects. In the BOX program, we decided to calculate the production cross section keeping always the value of the top-quark offshellness fixed and equal to the mass value. Finite-width effects are included a posteriori, at the same stage of top decay-products generation.

[2] From here on, we will describe only the s-channel case. Same consideration hold for the t-channel case.

3. Modes of operation

The program generates hard events that can then be fed into an SMC program for subsequent showering. POWHEG-BOX saves the hard event information according to the conventions of the Les Houches Interface for User Processes (LHIUP from now on) [9]. The SMC should also comply with these conventions (as is the case for PYTHIA and HERWIG) in order to be used in conjunction with POWHEG-BOX.

The program can be run in three ways:

- POWHEG-BOX generates hard events, and stores them in a file. A SMC program reads the file and showers them.
- POWHEG-BOX is linked directly togheter with the SMC. In this case the events are generated and immediately showered, without intermediate storage.
- POWHEG-BOX is run as a standalone program, and the produced hard events are analyzed without showering. The output yields, in this case, NLO distribution with LL resummation of soft gluon effects.

3.1 Storing the user events

The easiest way to interface POWHEG-BOX to a SMC is to simply store the hard events in a file (which we call the *event file*), and in a subsequent run read the events and process them with the SMC. The format of the event file supported by POWHEG-BOX is the "Standard format for Les Houches event files", documented in ref. [10]. The program for the generation of the Les Houches Event Files (LHEF from now on) can be built with the command

```
$ make pwhg_main
```

The event file is named pwgevents.lhe (the user is given the possibility to change the file name, as documented in the next section).

An example program that reads the event file, showers it with HERWIG and analyzes it can be built as follows

```
$ make main-HERWIG-lhef
```

A similar program, named main-PYTHIA-lhef, is provided for PYTHIA, and can be built with the command

```
$ make main-PYTHIA-lhef
```

The user should take care of installing the HERWIG or PYTHIA program in the POWHEG-BOX directory. In the case of HERWIG, the appropriate include files should also be present. As can be evinced from the Makefile, the fortran files relevant for these examples are main-HERWIG-lhef.f, herwig6510.f (main-PYTHIA-lhef.f, pythia6.4.22.f for PYTHIA), pwhg_bookhist.f and pwhg_analysis.f.

The file pwhg_analysis.f contains a template analysis, that one can take as a starting point for more complex analysis. It uses pwhg_bookhist, the histogramming package of M.L. Mangano with minor modifications, and it produces topdrawer outputs in the file pwg***.top.

The routines in it are adequate for both HERWIG and PYTHIA since they rely on the standard common blocks of ref. [11]. As stated earlier, they use the jet algorithms implemented in the FASTJET package, namely the SISCONE and $k_{\rm T}$ ones. In particular, the default analysis calls the $k_{\rm T}$ jet algorithm. Implementing both the SISCONE and $k_{\rm T}$ algorithms may result in a considerable amount of running time, especially if multiple interactions and/or underlying events are switched on. If the user would like to use other analysis routines, or to use other jet-finding packages, he/she can simply modify the pwhg_analisys.f file or write his/her own.[3]

3.2 Interfacing POWHEG-BOX with a Shower Monte Carlo program

One should create a main program that initializes the SMC to make it ready to accept a user process, and provide the following routines

```
subroutine UPINIT
call pwhginit
end

subroutine UPEVNT
call pwhgevnt
end
```

that are the only link to the POWHEG-BOX program. The main program should call the appropriate subroutines to run the SMC. If the SMC is compliant with the LHIUP, it will call the routines UPINIT and UPEVNT in order to initialize and to generate the hard events. The routine pwhginit performs the initialization of POWHEG-BOX, setting up all the grids that are necessary for the efficient generation of the events, and it also initializes the process common block of the LHIUP. Each call to pwhgevnt results in the generation of one event, and its storage in the LHIUP event common block.

When using HERWIG, one must remove the dummy subroutines UPINIT and UPEVNT that are present in the HERWIG source file.

4. Input parameters

POWHEG-BOX provides an independent facility to set the input parameters for the run. All

[3]During the integration stage and/or during the generation of the event file, the possibility to perform a NLO analysis or an analysis at the level of the POWHEG output, before interfacing to the shower, is left to the user. In these cases, the analysis is executed with the string WHCPRG set to 'NLO'. In the current released version, the template analysis dedicated to single-top processes do not produce plots for an analysis performed at the NLO stage or during the event generation.

parameters are stored in a file, named `powheg.input`. Examples of these files can be found in the `testrun` subdirectory. The format of these files is as follows

1. Lines are no more than 100 characters long.

2. Empty (blank) lines are ignored

3. If a `#` or a `!` appears at any point in a line, the part of the line starting from the `#` or `!` symbol up to its end is blanked.

4. An entry has the format:
   `name  value`
   usually followed by a `!` and a comment to clarify the meaning of the variable. The `name` keyword has no more than 20 characters, and `value` is an integer or floating point number.

5. A maximum of 100 keywords are allowed.

If the file `powheg.input` is not present, the program asks the user to enter a prefix, and then looks for the file `<prefix>-powheg.input`. In this case, all the files created by `POWHEG-BOX` in the current run will carry the prefix `<prefix>-` instead of `pwg`.
The input parameters are read by the (`real * 8`) function `powheginput(string)`, in file `powheginput.f`. The statement

```
rvalue=powheginput('myparm')
```

returns the value of token `myparm` stored in `powheg.input`. If the token is not found in the input file, a message is printed, and the program is stopped. The file is read only once, on the first invocation of the function `powheginput`, and token-value pairs are stored in internal arrays, so that subsequent calls to `powheginput` are relatively fast. The statement

```
rvalue=powheginput('#myparm')
```

also returns the value of the token `myparm`. However, in case the token `myparm` is not present, the program does not stop, and returns the value -10^6. The file `powheginput.f` is a standalone code, and can be linked to any program. In this way, an SMC that is reading an event file may get parameters of the `POWHEG-BOX` run, if it needs too.

We document here a typical input file for single-top processes:

```
!  ST-schannel inputs

numevts 500000 !  number of events to be generated
ih1 1          !  hadron 1 type (1:  proton; -1:  antiproton)
```

```
ih2 1          !  hadron 2 type (1:  proton; -1:  antiproton)
lhans1 10050   !  pdf set for hadron 1 ( LHAGLUE number )
lhans2 10050   !  pdf set for hadron 2 ( LHAGLUE number )
```

The first entry is self-explanatory. The integers `ih1,ih2` and `lhans1,lhans2` characterize instead the hadron type and PDF used in `POWHEG-BOX`. The numbering scheme is that of LHAGLUE interface, leaving the possibility of re-evaluate pdf's on the fly (using number corresponding to .LHpdf file) or to interpolate from a previously calculated grid (number corresponding to .LHgrid file), as explained in ref. [12]. In the example above, 10050 corresponds to the central value of the `CTEQ6M` set in this latter case . The hadron type in `ih1` and `ih2` can be 1 for a proton or -1 for an antiproton.

```
ebeam1 7000 !  energy of beam 1 in GeV
ebeam2 7000 !  energy of beam 2 in Gev
```

We assume that beam 1 and 2 move along the third axis in the positive and negative direction respectively.

```
facscfact 1 !  factorization scale factor:  mufact=muref*facscfact
renscfact 1 !  renormalization scale factor:  muren=muref*renscfact
```

Factorization and renormalization scale factors appearing here have to do with the computation of the inclusive cross section (i.e. the $\bar{B}$ function [1, 2, 5]), and can be varied by a factor of order 1 to study scale dependence. The natural choice for this process is the mass of the top-quark. We choose to perform the NLO calculation keeping these scales fixed. The experencied user can change this setting modifying the `set_fac_ren_scales` routine.

The following parameters control the operation of the `POWHEG-BOX` program:

```
!  Parameters to allow or not the use of stored data
use-old-grid 1
use-old-ubound 1
```

The meaning of these tokens requires a little knowledge of the operation of `POWHEG-BOX`. Before the program starts generating events, the integral of the inclusive cross section is computed, and a grid is set up for the generation of Born-like configurations. Similarly, in the generation of hard radiation a grid is computed to get an upper bounding function to the radiation probability. The generation of the grids is time consuming, but the time spent in this calculation is negligible in a normal run, when hundreds of thousands of events are generated. On the other hand, sometimes it is useful (for example, when debugging an analysis program) to skip the generation stage. For this purpose, the grid for the generation of Born-like kinematics is stored in the file `pwggrid.dat`.
If `use-old-grid` is set equal to 1, and `pwggrid.dat` exists and is consistent, it is loaded,

and the old grid and old value of the cross section are used. Otherwise, a new grid is generated. Observe that the program does check the file for consistency with the current run, but the check is not exhaustive. The user should be sure that a consistent grid is used. The token `use-old-ubound` has the same role as `use-old-grid`, but it applies to the upper bounding array that is used in the generation of radiation.

The following parameters are used to control the grids generation:

```
!  Parameters that control the grid for Born variables generation
ncall1 50000    !  number of calls for initializing the integration grid
itmx1 5         !  number of iterations for initializing the integration grid
ncall2 50000    !  number of calls for computing integral
itmx2 5         !  number of iterations for computing integral
foldcsi 1       !  number of folds on x integration
foldy 1         !  number of folds on y integration
foldphi 1       !  number of folds on phi integration
nubound 20000   !  number of bbarra calls to setup upper bounds for radiation
iymax 1         !  <=100, number of intervals in y grid to compute upper bounds
icsimax 1       !  <=100, number of intervals in csi grid
xupbound 2      !  increase upper bound for radiation generation by given factor
```

The values of some of the tokens may be changed in the following cases:

- If the integration results have large errors, one may try to increase `ncall1`, `itmx1`, `ncall2`, `itmx2`.
- If the fraction of negative weights is large, one may increase `foldcsi`, `foldy`, `foldphi`. Allowed values are 1, 2, 5, 10, 25, 50. The speed of the program is inversely proportional to the product of these numbers, so that a reasonable compromise should be found. Our experiences tell us that, even at LHC energies, the fraction of negative weights in $\bar{B}$ calculation is such that the numbers provided in the examples need not to be changed. For the t- channel case, it is recommended to leave the default foldings on the csi and y variables.[4]
- If there are too many upper bound violations in the generation of radiation (see section 6), one may increase `nubound`, and/or `xupbound`.
- If the efficiency in the generation of radiation is too small, one may try to increase `iymax`, `icsimax`. We recommend to use for these parameters the values in the template input cards.

[4]In all examples, the choice of the parameters that control the grid generation is such that a reasonably small fraction of negative weights is generated, so they can be run as they are. We remind the reader that these negative weights are only due to our choice of generating $\tilde{B}$ instead of $\bar{B}$. They indeed correspond to phase space points where NLO corrections are bigger than LO contributions. Had we performed the integration over the full radiation phase space these negative weights would have disappeared completely.

- For single-top, it is recommended to activate also the `withdamp` option, to enable the Born-zero damping factor.

In order to check whether any of these conditions occurs, the user should inspect the files `pwgstat.dat` and `pwgcounters.dat` at the end of the run, as illustrated in sec. 6.

Other mandatory parameters are those specifically related to single-top processes. For the production part, the relevant parameters are:

```
!  production parameters
ttype 1       !  1 for t, -1 for tbar

topmass 175.0
wmass 80.4
sthw2 0.23113
alphaem_inv 137.0359895

CKM_Vud 0.9740
CKM_Vus 0.2225
CKM_Vub 0.000001
CKM_Vcd 0.2225
CKM_Vcs 0.9740
CKM_Vcb 0.000001
CKM_Vtd 0.000001
CKM_Vts 0.000001
CKM_Vtb 1.0
```

The value of `ttype` is used to decide if top or antitop quarks will be produced. The meaning of the other parameters is self-explanatory. We remind that it is not allowed to set any entry of the CKM matrix exactly equal to zero.

In the current released version, top-quark decay products are always generated by `POWHEG`, accordingly to a procedure very similar to the one of ref. [6]. Therefore, all the following parameters are mandatory too:

```
!  decay parameters
topwidth 1.7
wwidth 2.141

topdecaymode 10000 !  decay mode:  the 5 digits correspond to the following
                   !  top-decay channels (l,mu,tau,u,c)
                   !  0 means close, 1 open
tdec/elbranching 0.108 !  W electronic branching fraction
tdec/emass 0.000511
tdec/mumass 0.1056
```

```
tdec/taumass 1.777
```

The value of the `topdecaymode` token is formed by five digits, each representing the maximum number of the following particles at the (parton level) decay of the t ($\bar{t}$) quark: $e^{\pm}$, $\mu^{\pm}$, $\tau^{\pm}$, $\overset{(-)}{u}$, $\overset{(-)}{c}$. Thus, for example, 10000 means $t \to e^{+}\nu_e b$, 11100 means all semileptonic decays, 00011 means fully hadronic.

5. Optional parameters

In addiction to mandatory parameters presented above, `POWHEG-BOX` also accepts other parameters. The user should not worry if they are not present since, in this case, default values are used. We include them here since they can be useful for a more advanced use of the program: who is not interested in modifying them can safely skip this section. It follows a list of these parameters with corresponding default values:

```
QCDlambda5 0.25 !  for not equal pdf sets
withdamp 1 !  (default 0, do not use) use Born-zero damping factor
charmthr 1.5 !  (default 1.5 GeV) charm treshold for gluon splitting
bottomthr 5.0 !  (default 5.0 GeV) bottom treshold for gluon splitting
charmthrpdf 1.5 !  (default 1.5 GeV) pdf charm treshold
bottomthrpdf 5.0 !  (default 5.0 GeV) pdf bottom treshold
ptsqmin 0.8 !  (default 0.8 GeV) minimum pt for generation of radiation
```

For testing the correct behaviour of the program and to obtain NLO distributions, we added other parameters that may also be useful for developers. The normal user is asked not to change them, since their invocation is time consuming and/or may cause some conflicts with other settings. If instead the user is interested in changing them, a detailed explanation of their behaviour ca be found on ref. [3].

```
testsuda 0 !  (default 0, do not test) tests the Sudakov FF by numerical integration
testplots 0 !  (default 0, do not) do NLO and PWHG distributions
bornonly 0 !  (default 0) if 1 do Born only
smartsig 0 !  (default 1) remember equal amplitudes (0 do not remember)
withsubtr 0 !  (default 1) subtract real counterterms (0 do not subtract)
radregion 1 !  (default all regions) only generate radiation in the selected
singular region
iupperisr 1 !  (default 1) choice of ISR upper bounding functional form
iupperfsr 2 !  (default 2) choice of FSR upper bounding functional form
```

6. Counters and statistics

Several results relevant to the interpretation of the output of the run are written into the

files pwgstat.dat and pwgcounters.dat. The fraction of negative weights, the total cross section, the number of upper bound failures in the generation of the inclusive cross section, and the generation efficiency, together with failures and efficiency in the generation of hard radiation, are printed there. These are quite self-explanatory and we do not comment them any further. These numbers are sufficient to take action in case of problems.

7. Random number generator

POWHEG-BOX uses the RM48 random number generator, documented in the CERNLIB write-ups. This generator has default initialization. If a user wishes to start the program with different seeds, he/she should add lines similar to

```
!  Random number generator initializing parameters
iseed 6093726 !  initialize random number sequence
rand1 -1      !  initialize random number sequence
rand2 -1      !  initialize random number sequence
```

to the input card. This results in a call to the rm48in(iseed,rand1,rand2) subroutine that seeds the generator with the integer iseed, and skip the first rand1+rand2*10**8 numbers, as documented in the CERNLIB manual. This can be useful if one wants to resume a previous run. In that case, one has simply to use as initializing values those reported in the <prefix>-events.lhe file. If instead one just wants to change the seed only, he/she can comment or skip the rand1 and rand2 lines in the input card. The last option

```
manyseeds 1
```

may be 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. The event file is called pwgevents-j.lhe

We remind the reader that a change in random number generator initialization affects the POWHEG-BOX random number sequence, both in the generation of events and in NLO computation or upper bound searching, when the corresponding grids are not present. If the program is interfaced to a SMC, the user should also take care to initialize the seeds of the latter.

8. Generation of a sample with t and $\bar{t}$ events

The user can be interested in the generation of a sample where both top and antitop events appear. To this purpose, a script and a dedicated executable have been included. The script is named merge_ttb.sh and can be found in the directory testrun. It can be run in any subfolder of ST_sch. Three inputs are mandatory: the first two are the prefixes

of the input files used to generate t and $\bar{t}$ events. The third input has to be an integer and correspond to the total number of events that the final *merged* sample will contain. The script has to be run twice, using a positive integer value at the first call and its opposite afterwards. Therefore, for example, to produce a sample of 10000 events at Tevatron, starting from the input files `tev_st_s_t-powheg.input` and `tev_st_s_tb-powheg.input`, the invocation lines should be as follows:

```
$ sh merge_ttb.sh tev_st_s_t tev_st_s_tb 10000
```

and then

```
$ sh merge_ttb.sh tev_st_s_t tev_st_s_tb -10000
```

Few remarks are needed:

- it is responsability of the user to check that the 2 input files are equal. The `ttype` tokens have to be different, obviously.
- the two values of `numevts` are not really used: the program re-calculate the needed values as a function of the t and $\bar{t}$ cross sections and of the total number of events to be generated.
- the final event file is always named `t_tb_sample-events.lhe`. In the header section it also contains a copy of the two input files used to generate it, for cross-checking purposes

References

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

[2] 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 [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," [arXiv:1002.2581 [hep-ph]].

[4] B. W. Harris, E. Laenen, L. Phaf, Z. Sullivan and S. Weinzierl, "The Fully differential single top quark cross-section in next to leading order QCD," Phys. Rev. D **66**, 054024 (2002) [arXiv:hep-ph/0207055].

[5] S. Alioli, P. Nason, C. Oleari and E. Re, "NLO single-top production matched with shower in POWHEG: s- and t-channel contributions," JHEP **0909**, 111 (2009) [arXiv:0907.4076 [hep-ph]].

[6] S. Frixione, E. Laenen, P. Motylinski and B. R. Webber, "Angular correlations of lepton pairs from vector boson and top quark decays in Monte Carlo simulations," JHEP **0704**, 081 (2007) [arXiv:hep-ph/0702198].

[7] J. Alwall *et al.*, "MadGraph/MadEvent v4: The New Web Generation," JHEP **0709**, 028 (2007) [arXiv:0706.2334 [hep-ph]].

[8] M. Cacciari and G. P. Salam, "Dispelling the N^3 myth for the k_t jet-finder," Phys. Lett. B **641**, 57 (2006) [arXiv:hep-ph/0512210].

[9] E. Boos *et al.*, "Generic user process interface for event generators," [arXiv:hep-ph/0109068].

[10] J. Alwall *et al.*, "A standard format for Les Houches event files," Comput. Phys. Commun. **176** (2007) 300 [arXiv:hep-ph/0609017].

[11] T. Sjöstrand et al., in "Z physics at LEP1: Event generators and software,", eds. G. Altarelli, R. Kleiss and C. Verzegnassi, Vol 3, pg. 327.

[12] M. R. Whalley, D. Bourilkov and R. C. Group, "The Les Houches accord PDFs (LHAPDF) and LHAGLUE," [arXiv:hep-ph/0508110].