Weighted events in the BOX

Two flags control the nature of the output in the BOX: withnegweights and ptsupp. If neither of these flags is set, events are output with weight 1, i.e. the XWGTUP variable is set to 1. The IDTUP variable in the Les Houches interface is set to 3 in this case. The total cross section is stored in the xsecup variable. Negative weighted events are neglected with this choice.

If withnegweights is set to 1 (true in our convention), negative weighted events are not discarded. The IDTUP variable in the Les Houches interface is set to -4, and the weight of the event is set to its sign times the total cross section for positive weighted events plus the (absolute value) of the cross section for negative weighted events. In this way, the average value of XWGTUP equals the real cross section, as required by the Les Houches convention when the IDTUP variable is set to -4. The variable xsecup always stores the real cross section.

If the ptsupp token is set, a suppression factor that depends upon the underlying Born configuration of each event is supplied with it. The cross section computed by the pwhg_main program is in this case not valid. It is the integral of the cross section times the suppression factor. Events are generated using this "fake" cross section, and thus are weighted with the inverse of the suppression factor. The IDTUP variable in the Les Houches interface is set to -4. The weight of the event is in this case the sign, times the total cross section for positive weighted events plus the (absolute value) of the cross section for negative weighted events, times the inverse of the weighting factor. The weight factor is returned by the user routine born_suppression, that can use the value of the ptsupp token as a parameter to compute the suppression factor. Also in this case, the average value of the weight of the event is equal to the real cross section. This option can be active in conjunction with the withnegweights flag.

These flags have many uses. On one side, one might like to know where negative weighted events end up. The fact that they constitute a small fraction leaves us to worry that they may end up in some tiny tail of some important distribution. One may also prefer to work with negative weight in cases when getting rid of them requires high folding numbers (the foldcsi, foldy and foldphi tokens), and thus has a high cost in computer time. The ptsupp feature can be use to enhance a region of phase space (like a high k_T tail) where it would be otherwise difficult to get high statistics. These features, however, become really useful for processes where the Born contribution itself is singular. The simplest examples are the $Z+$jet and the dijet production processes. Here we discuss $Z+$jet. The dijet case is fully analogous.

1 Generation cut and Born suppression factor

The $Z+1j$ process differs substantially from all processes previously implemented in POWHEG, in the fact that the Born diagram itself is collinear and infrared divergent. In all previous implementations, the Born diagram was finite, and it was thus possible to generate an unweighted set of underlying Born configurations covering the whole phase space. In the present case, this is not possible, since they would all populate the very low transverse momentum region. Of course, this problem is also present in standard Shower Monte Carlo programs, where it is dealt with by generating the Born configuration with a cut $k_{\rm gen}$ $k_{\rm gen}$ on the transverse momentum of the Z boson. After the shower, one must discard all events that fail some transverse momentum analysis cut $k_{\rm an}$ $k_{\rm an}$ in order to get a realistic sample. The analysis cut $k_{\rm an}$ may be applied to the transverse momentum of the Z, or to the hardest jet. We assume here, for sake of discussion that the analysis cut is applied to the Z transverse momentum.

Taking $k_{\rm an} \gtrsim k_{\rm gen}$ is not enough to get a realistic sample. In fact, in an event generated at the Born level with a given $k_T < k_{\rm gen}$, the shower may increase the transverse momentum of the jet so that $k_T^Z > k_{\rm an}$. Thus, the generation cut, even if it is below the analysis cut, may reduce

the number of events that pass the analysis cut. Of course, as we lower $k_{\rm gen}$ keeping $k_{\rm an}$ fixed, we will reach a point when very few events below $k_{\rm gen}$ will pass the analysis cut $k_{\rm an}$. In fact, generation of radiation with transverse momentum larger than $k_{\rm gen}$ is strongly suppressed in `POWHEG`, and, in turn, radiation from subsequent shower is required to be not harder than the hardest radiation of `POWHEG`. Thus, given the fact that we want to generate a sample with a given $k_{\rm an}$ cut, we should choose $k_{\rm gen}$ small enough, so that the final sample remains substantially the same if $k_{\rm gen}$ is lowered even further.

A second option for the implementation of processes with a divergent Born contribution is also available. It requires that we generate weighed events, rather than unweighted ones. This is done by using a suppressed cross section for the generation of the underlying Born configurations:

$$\bar{B}_{\rm supp} = \bar{B} \times F(k_T), \tag{1}$$

wher $\bar{B}$ is the inclusive NLO cross section at fixed underlying Born variables, and k_T is the transverse momentum of the vector boson in the underlying Born configuration. In this way $\bar{B}_{\rm supp}$ is integrable, and one can use it to generate underlying Born configurations according to its value. The generated event, however, should be given a weight $1/F(k_T)$ rather than a constant one, in order to compensate for the initial $F(k_T)$ suppression factor. With this method, events do not concentrate in the low k_T region. However, their weight in the low k_T region becomes divergent. After shower, if one imposes the analysis cut, one gets a finite cross section, since it is unlikely that events with small transverse momentum at the Born level may pass the analysis cut after shower. In fact, shower transverse momenta larger than the one present in the initial Born process must be suppressed in the Monte Carlo generator.

In recent `POWHEG BOX` revisions, both methods can be implemented at the same time. We wanted i fact to be able to implement the following three options:

- Generate events using a transverse momentum generation cut.
- Generate events using a Born suppression factor, and a small transverse momentum cut, just enough to avoid unphysical values of the strong coupling constant and of the factorization scale that appears in the parton density functions.
- Apply a Born suppression factor, and set the transverse momentum cut to zero. In this case the program cannot be used to generate events. It can be used, however, to produce NLO fixed order distributions, provided the renormalization and factorization scales are set in such a way that they remain large enough even at small k_T^Z. This feature is only used for the generation of fixed order distributions.

The generation cut is activated by setting the token `bornktmin` to the desired value in the `powheg.input` file. The Born suppression is activated by setting the token `ptsupp` to a positive real value. The process-specific subroutine `born_suppression` sets the suppression factor to $k_T^2/(k_T^2+\texttt{ptsupp}^2)$. If `psupp` is negative, the suppression factor is set to 1.

The need of a transverse momentum cut is not only a technical issue. The NLO calculation of $Z+1j$ production holds only if the transverse momentum of the Z is not too small. In fact, as the k_T decreases, large Sudakov logarithms arise in the NLO correction, and the value of the running coupling increases, up to the point when the cross section at fixed order becomes totally unreliable. These large logarithms should all be resummed in order to get a sensible answer in this region. In the `POWHEG` implementation of Z production, in fact, these logarithms are all resummed. It is clear then that some sort of merging between the $Z+1j$ and the Z production processes should be performed at relatively small transverse momentum, in order to properly deal with these large logarithms. In the present work we will not attempt to perform such merging, that we leave for future publications. We will simply remember, when looking at our results, that we expect to get unphysical distributions when the Z transverse momentum is too small, and we will discuss this fact in a more quantitative way.

In the POWHEG approach, negative weighted events can only arise if one is approaching a region where the NLO computation is no longer feasible. In our studies for the $Z + 1j$ process we approach this region at small transverse momentum. In order to better see what happens there, rather than neglecting negative weights (that is the default behaviour of the POWHEG BOX), we have introduced a new feature in the program, that allows one to track also the negative weighted events. This feature is activated by setting the token withnegweights to 1 (true). If withnegweights is set to 1, events with negative weight can thus appear in the Les Houches event file. While we normally set the IDWTUP flag in the Les Houches interface to 3, in this case we set it to -4. With this flag, the SMC is supposed to simply process the event, without taking any other action. Furthermore, the XWGTUP (Les Houches) common block variable is set by the POWHEG-BOX to the sign of the event times the integral of the absolute value of the cross section, in such a way that its average equals the true total cross section.

Notice that, if withnegweights is set and a Born suppression factor is also present, the events will have variable XWGTUP of either signs. In this case XWGTUP is set to the sign of the event, times the absolute value of the cross section, divided by the suppression factor ptsupp. Also in this case the average value of XWGTUP coincides withe the true total cross section. We preferred not to use the option -3 in case of signed events with constant absolute value. This option is advocated by the Les Houches interface precisely in such cases. However, the Les Houches interface does not provide a standard way to store the integral of the absolute value of the cross section, that would be needed to compute correctly the weight of the event. In fact, the XSECUP variable is reserved for the true total cross section. More specifically, if we have N events of either sign, they should be weighted with the sum of the positive plus the absolute value of the negative part of the cross section, in such a way that

$$\sum_{i=1}^{N} W_i\big(\sigma_{(+)} + |\sigma_{(-)}|\big) = N\big(\sigma_{(+)} - |\sigma_{(-)}|\big) = N\sigma_{\rm NLO}, \tag{2}$$

(where W_i are the sign of the event ± 1), because

$$\frac{\sum_i W_i}{N} = \frac{\big(\sigma_{(+)} - |\sigma_{(-)}|\big)}{\big(\sigma_{(+)} + |\sigma_{(-)}|\big)}\,. \tag{3}$$

Weighted events are also useful if one wants to generate a homogeneous sample from relatively low up to very high transverse momenta. It is convenient in this case to pick a very large ptsupp value, of the order of the maximum transverse momentum one is interested in. The large momentum region will be more populated in this way.