The manyseeds flag

If the manyseeds flag is set to 1, the BOX looks for a file named pwgseeds.dat. The program stops if this file is not found. If the file is found the program asks for an integer j for input. Let us say that this integer is 17, for sake of clarity. This integer indicates the line in pwgseed.dat where a random number seed is stored. This feature is used for running the BOX on several nodes of a cluster. In order to do this, follow the steps

1. Prepare a powheg.input file with the manyseeds flag is set to 1. Set the number of events nev to 0.

2. Prepare a file pwgseeds.dat, containing a sequence (one per line) of different random number seeds. For example: first line 1, second line 2, etc. (but any number will do).

3. Run the pwhg_main program. It will ask an integer for input. Input an integer. That integer is the line number of the random seed to be used for the current run. Assuming that the number 17 is given as input to the pwhg_main program, the run will produce files named pwgxgrid.dat and pwggrid-0017.dat. The pwhg_main program can be run with different integers as input. Each run can be sent, for example, to a different node of a cluster.

   At the end of this step, a bunch of pwggrid-[????].dat, pwgubound-[????].dat files ans pwgNLO-[????].top files will be present in the run directory. The pwgNLO-[????].top files are statistically independent topdrawer histograms. They can be combined to provide a higher statistics NLO analysis of the current analysis routines.

   If the subsequent runs are sent after the file pwgxgrid.dat was already produced, and if the flag use-old-grid is set to one, the importance sampling grid will be loaded from the pwgxgrid.dat. Otherwise it will be recreated.

4. Now set the nev token to a given number, make sure that the flag use-old-grid is set to one, and run a bunch of copies of the pwhg_main program, each with the same integers as input. The program will now load all the pwggrid-[????].dat and pwgubound-[????].dat that it can find, and combines them adequately, assuming that they are all statistically independent, and will start to generate events. The events will be in files pwgevent-[????].lhe, and they will all be independent statistically.

The sequence above is a quite simple two step procedure. It is useful, however, to better clarify the logic that the BOX follows in this procedure.

There are 3 steps in the initialization phase of POWEG. First of all, an importance sampling grid is determined. Let us call this stage ISG (Importance Sampling Grid). The second step is the calculation of the integrals, and the determination of an upper bounding envelope for the $\tilde{B}$ function, to be used for the generation of underlying Born configurations. We call this stage the UBB (Uppe Bounds for underlying Born). As a third step, the upper bound normalization for radiation is determined. We call this stage UBR (Upper Bounds for Radiation).

First of all, we remark that, if the use-old-grid flag is not set to 1, no grid file is loaded. Similarly, if the use-old-ubound flag is not set to 1, no ubound file is loaded. In other words, if these flags are set, looking for a corresponding file will always yield a negative result. The reader should keep this in mind when reading the following procedure. If the flag manyseeds is set, the pwhg_main program asks for an integer. We will call cj this integer, between 0 and 9999. We denote with [cj] the corresponding string of four digits (leading digits are set to 0; thus if ic=1 [cj]=0001. We will denote as [????] any four digit string. The logic of grid loading in the BOX is as follows:

1. If a file pwggrid.dat exist and is consistent, this file is loaded, and steps ISG and UBB are skipped (go to 6).

2. If the above fails, if the `manyseeds` flag is not set, or if it is set and a file named `pwggrid-[cj].dat` already exists and is consistent, all files of the form `pwggrid-[????].dat` are loaded and suitably combined, and the steps ISG and UBB are skipped (go to 6).

3. If the above fails, if a file `pwgxgrid.dat` exist and is consistent, this file is loaded, and steps ISG is skipped (go to 5).

4. Step ISG is performed. The resulting grid is stored in the file `pwgxgrid.dat`. This step, whether the `manyseeds` flag is set or not, is performed using the default initial seed value (i.e. not the seed found at the `cj` line of the `pwgseeds.dat` file). In this way, all copies of the program being run will use the same importance sampling greed. This is mandatory if we want to combine results.

5. Step UBB is performed. If the `manyseeds` flag is set, this step is performed using the seed found at the `cj` line of the `pwgseeds.dat` file, and the result is storedin the file `pwggrid-[cj].dat`. Otherwise, the current seed value is used, and the result is stored in a file named `pwggrid.dat`.

6. If a file named `pwgubound.dat` exists and is consistent it is loaded. The UBR step is skipped (goto 10).

7. If a file named `pwgubound.dat` exists and is consistent it is loaded. The UBR step is skipped (goto 10).

8. If the above fails, if the `manyseeds` flag is not set, or if it is set and a file named `pwgubound-[cJ].dat` exists and is consistent, all files with names of the form `pwgubound-[????].dat` are loaded and combined. The UBR step is skpped (go to 10).

9. The UBR step is performed. If the `manyseeds` flag is not set, the result is stored in a file named `pwgubound.dat`. Otherwise, it is stored in the file `pwgubound-[cj].dat`.

10. Now `nev` events are generated. If the `manyseeds` flag is not set, the result is stored in a file named `pwgevents.lhe`. Otherwise, it is stored in the file `pwgevents-[cj].dat`.

This logic has the purpose to allow several possible combinations of actions. For example, one can use grids generated in parallel runs to produce events without using the `manyseeds` flag. Or one can use grids generated without the `manyseeds` flag for generating events in parallel with the `manyseeds` flag set.