Whizard threshold scan on lxplus

Introduction

This document contains instructions to run a threshold scan of benchmark point two for the CLIC CDR. It is assumed that the user has a valid lxplus account.

Setup

Please prepare your environment, so that it finds all relevant executables and libraries.

```
# we assume you are in bash
export PATH=/afs/cern.ch/sw/lcg/contrib/gcc/4.5.2/x86_64-slc5/bin:${PATH}
export LD_LIBRARY_PATH=/afs/cern.ch/sw/lcg/contrib/gcc/4.5.2/x86_64-slc5/lib64:${LD_LIBRARY_PATH}

export PATH=/afs/cern.ch/user/j/jfstrube/public/PYTHON/bin:${PATH}

export PATH=/afs/cern.ch/user/j/jfstrube/public/Ocaml-3.12.1/bin:/afs/cern.ch/user/j/jfstrube/public/Whizard-
2.0.6/bin:${PATH}
export LD_LIBRARY_PATH=/afs/cern.ch/user/j/jfstrube/public/Ocaml-3.12.1/lib:${LD_LIBRARY_PATH}
```

After executing these steps, whizard is available as `/afs/cern.ch/user/j/jfstrube/public/Whizard-2.0.6/bin/whizard`. The following provides a framework for the threshold scan.

Running the code

The input data

Please copy the directory with the input files from afs

```
cp -a /afs/cern.ch/user/j/jfstrube/public/CLIC_Amplitudes/SUSY/ .
```

You may not be able to copy everything. In this case, the only important files to keep are the **`*.txt`** files and the LHA inputs **`bench2_slha.dat`**. Remove everything else, it will be re-generated. If you are not interested the current set of channels (subdirectories), feel free to remove (or add) some.

The code

The script that does all the work is called `compile_thresholdscan.py`. It carries out the following steps

1. obtain a list of the subdirectories
2. in each of the subdirectories, look for a **`.txt`** file with the same name as the directory
3. for each of the given energies, create the Whizard **`.sin`** file from the **`.txt`** file
4. call Whizard on this file
5. parse the resulting log file and remember the result
6. rename the logfile, so that it ends with the energy at which the cross-section was computed
7. move to the next directory
8. at the end print the output on the screen and write to file

Call the script with `/afs/cern.ch/user/j/jfstrube/public/PYTHON/bin/python compile_thresholdscan.py`

Analyzing the output

Conversion to comma-separated values

The first step is to turn the output to a csv file. This is not strictly necessary, but it provides a threshold to involuntarily overwriting the output from the previous run.
`/afs/cern.ch/user/j/jfstrube/public/PYTHON/bin/python convert2csv.py` looks for the file `results.txt` and creates the file `results.csv`

Plotting

The csv file can be turned into a plot using `/afs/cern.ch/user/j/jfstrube/public/PYTHON/bin/python plot_results.py`. This simple script uses the matplotlib library to create pdf files from `results.csv`