slharoutines — Manipulating SLHA files with PERL

Robert V. Harlander

$\boxed{\text{12 Aug 2011}}$

1 Outline of the problem

The programs ggh@nnlo and bbh@nnlo [1] work with a certain format of in- and output files (SLHA-like format). The precise definition for what we understand by this will be explained below. The typical application of an SLHA-line input file is to provide a single set of input parameters. So, for example, one value of the Higgs mass, the center-of-mass energy, etc. is provided, and the corresponding value for the cross section is evaluated by one of the programs mentioned above.

Often, however, one would like to calculate the cross section for a whole set of parameters, for example, for Higgs masses from 100 to 300 GeV in steps of 10 GeV, or for various values of the renormalization scale, etc. This requires to change the input file for every run, read the result from the output file, save it, and start over. Of course, this should be done by a script, and slharoutines provides a useful set to facilate this task.

2 SLHA-like format

What we mean by "SLHA-like format" of in- and output files is the following structure:

```
Block <NAME_1>
  <label_1.1>  <entry_1.1.1> ... <entry_1.1.n11>
  <label_1.2>  <entry_1.2.1> ... <entry_1.2.n12>
           .
           .
Block <NAME_2>
  <label_2.1>  <entry_2.1.1> ... <entry_2.1.n21>
  <label_2.2>  <entry_2.2.1> ... <entry_2.2.n22>
           .
           .
```

The file contents are divided in blocks, initialized by the word `Block` at the beginning of the line (no leading whitespace allowed), followed by the name of the block. Every block contains an arbitrary number of lines which consist of, from left to right, (i) at least one whitespace, (ii) a label of type `INTEGER`, and (iii) a list of values, separated by at least one whitespace character.

Each value corresponds to an input or output parameter, and the characterization of the block name, the label number, and the position of the value in the line is unique. An example for an input file could look like

```
Block MASS
 1  100.d0 # Higgs mass
 2  172.3d0 # top quark mass
Block VCKM
 1  0.974d0 0.2253d0 0.00347d0
 2  0.2252d0 097345d0 0.041d0
 3  0.00862d0 0.0403d0 0.999152d0
```

The character `#` introduces a comment: anything left of it in that line is ignored.

3 The main routines

For the beginning, two routines should be sufficient. The first one (`changeparam`) allows to modify parameters in the `SLHA` input file, the other one (`extractslhav2`) extracts the relevant parameters from a set of output files.

`changeparam($file,$block,$label,$value)` : In `$file`, set the parameter at position `$label` in `$block` to `$value`. Example:

```
changeparam('in.tmp','MASS',1,'110.d0')
```

Note that all of the parameters of `changeparam` should be given as strings, except for `$label` which is an `INTEGER`.

`extractslhav2($dir,$dataform,$options)` : read all input files in directory `$dir` and print its data as specified by `$dataform` which typically is defined as follows:

```
$dataform = [ [ ( $block1, $entry1 ) ],
              [ ( $block2, $entry2 ) ], ... ];
```

Note that each inner square bracket corresponds to one value. `extractslha` will successively print these entries, sorting them numerically with respect to the first column. Example:

```
$dataform = [ [ ( 'MASS', 1 ) ],
              [ ( 'SIGMA', 1 ) ] ];
extractslha("outfiles/run142",$dataform);
```

This will print two columns. `extractslha` has an optional parameter `$options` which has the form of a referenced hash. There are currently two possible options:

```
$options = {"comments" => 1,
            "header" => 1,
            "printfun" => \&printdataline}
```

where we have indicated the default values. If you set `"comments" => 1`, `extractslhav2` will add `# <file>` to each line, indicating the filename out of which the corresponding numbers were read. Also, it will include a header in the data file that contains the header of the output files (i.e., all relevant parameters). The option `"header"` writes a line before the data, indicating what the individual columns mean. The parameter `"printfun"` is a reference to a function name. You can define any function in order to modify the way `extractslhav2` prints your data. This allows you to do some calculations with the numbers etc. It is advisable to copy the source code of `printdataline` to, say, `myprintdataline` when you define a new `"printfun"`, and then say

```
$options = {"printfun" => \&myprintdataline}
```

4 A complete example

Let us assume you want to run `bbh@nnlo` for Higgs masses from 100 to 300 GeV in steps of 10 GeV. The way to do this with `slharoutines` is to run `bbh@nnlo` for each Higgs mass and save the complete output file, uniquely renamed, in a separate directory. At the end of the run, this directory will contain 21 output files. `slharoutines` then provides a routine `extractslha` to read the desired values from these output files.

In detail (all paths relative to the `bbh@nnlo` root path:

1. Provide a template input file that fixes all parameters to the desired values except the Higgs mass. Let us name it `infiles/in.run142` here.

2. Create a new output directory, e.g. `data/out.run142`

3. Write a PERL script that loops over the desired values of Higgs masses: `scripts/runbbh.pl`.

4. In each loop iteration of `scripts/runbbh.pl`:

   - call `changeparam` from `slharoutines` to modify the value of the Higgs mass in the input file:

     ```
     changeparam('in.tmp','MASS',25,"$mass");
     ```

   - run `bbh@nnlo`:

     ```
     system('./x.main in.tmp');
     ```

   - store the output file in the output directory under a unique name:

     ```
     system('cp out.bbh ../data/run142/out.'."$mass");
     ```

5. after the script has finished, call `extractslha`

An example for a `PERL` script can be found at
`http://www.robert-harlander.de/software/bbh@nnlo/scripts/runbbh.pl`.

References

[1] These programs are available from `http://www.robert-harlander.de/software`