GoSam BLHA Interface How-To

T. Reiter, G. Luisoni

March 27th, 2012

Abstract

This technical note describes how to interface GoSam with an external Monte Carlo program (MCP) using the Binoth-Les-Houches-Accord (BLHA) interface.

Contents

1 Make sure GoSam and the libraries are set up correctly

We assume GoSam has been downloaded and installed using the script `setup.py` which comes with the distribution. You should ensure that the file `gosam.py` is in your `$PATH` variable.

Furthermore, we recommend the use of the package `gosam-contrib.tar.gz` which contains all relevant libraries. You can have several configurations (e.g. for different compilers) in parallel. The way to go is as follows:

```
tar xzf gosam-contrib.tar.gz
cd gosam-contrib
mkdir -p build/gfortran
mkdir -p install/gfortran
cd build/gfortran
# needs to be run from this directory!!!
../../configure \
    --prefix=`pwd`/../../install/gfortran \
```

Listing 1: The order file `order.lh`

```
# This file has been written by PROGRAM/VERSION
# Process: p p > j e- ve~

# Conventions preferred by the MC program:
MatrixElementSquareType      CHaveraged
IRregularisation             CDR
OperationMode                CouplingsStrippedOff
SubdivideSubprocess          no

# Here comes the process specific part:
CorrectionType               QCD
AlphasPower                  1
AlphaPower                   2

# Subprocesses
   1   -2    ->   21   11   -12
  -2    1    ->   21   11   -12
  21    1    ->    2   11   -12
   1   21    ->    2   11   -12
  21   -2    ->   -1   11   -12
  -2   21    ->   -1   11   -12

# Process specific settings for GoSam
#@ symmetries family,generation,%+1=-,%-1=+,%11=-
#@ model.options masses:mT,widths:wZ,wW
#@ filter.nlo NOT(SCALELESS)
```

```
    FC=gfortran F77=gfortran
make FFLAGS=-ffixed-line-length-none
make install
cd ../..
# The same procedure for the next compiler
mkdir build/ifort
mkdir install/ifort
cd build/ifort
../../configure \
    --prefix=`pwd`/../../install/ifort \
    FC=ifc F77=ifc
make
make install
```

This generates for each directory `gosam-contrib/install/<compiler>/` the subdirectories `lib/`, `include/` and `share/`. In the directory `share/gosam-contrib` you find a file `gosam.conf` which will serve as our configuration file later. If Form and QGraph are not in the search path you need to provide appropriate values for `form.bin` and `qgraf.bin`.

2 Prepare the order file

This step should be done by your Monte Carlo Program (MCP). Assume your MCP has written the file `order.lh` (see Listing 1).

Remarks.

- The order file can have any name and any extension. As a convention we typically reserve the extension `.lh` for order files and `.olc` for contract files.
- The options `SubdivdeSubprocess` has the following effect on the code generation with GoSam:

  `no` : GoSam generates one label per subprocess.

  `yes` : GoSam generates one label per helicity subamplitude and therefore *many* labels per subprocess.
- GoSam specific settings can be put into commentary lines starting with the letter combination '`#@`'. This is not part of the BLHA standard. The line '`#@ symmetries`...' restricts the helicity subamplitudes being generated to the ones relevant for this particular process.

3 Run GoSam

You should now run GoSam as follows:

```
gosam.py --olp --mc=MCP_name \
   --config=<your-path-to>/gosam.conf order.lh
```

Remarks.

- The option `--olp` is mandatory whenever you process a BLHA order file (which for the sake of this tutorial is always).
- The option `--mc=<MCP-name>` is optional. By specifying the name of the Monte Carlo Program which is the intended partner program, GoSam can choose some settings simplifying the communication and linking. You can either specify `--mc=<MPC-name>` or `--mc=name/version`. Alternatively (and also optional), you can put this information in the order file:

  ```
  #@olp.mc.name <MCP-name>
  #@olp.mc.version 1.0.0
  ```

  The short option for `--mc` is `-M`. The supported MCP names so far are: `powhegbox, sherpa`.
- The option `--config` points to the configuration file. Leaving out this option is not a good idea unless you have a configuration in one of the following locations, where GoSam finds them automatically:
  - GoSam installation directory,
  - user's home directory,
  - current working directory.

  Possible names for default configuration files are `gosam.in`, `gosam.conf` and `.gosam`. Therefore you could simply copy `<your-path-to>/gosam.conf` into the current directory and leave out this option. You can have more than one `--config` options, where the latter overwrites already present information in the former appearance of this option. The short form is `-c`.

Listing 2: The contract file `order.olc`

```
#@OLP GOLEM 1.0
#@IgnoreUnknown False
#@IgnoreCase False
#@SyntaxExtensions
MatrixElementSquareType CHaveraged | OK
IRregularisation CDR | OK
OperationMode CouplingsStrippedOff | OK
SubdivideSubprocess no | OK
CorrectionType QCD | OK
AlphasPower 1 | OK
AlphaPower 2 | OK
1 -2 -> 21 11 -12 | 1 0
-2 1 -> 21 11 -12 | 1 1
21 1 -> 2 11 -12 | 1 2
1 21 -> 2 11 -12 | 1 3
21 -2 -> -1 11 -12 | 1 4
-2 21 -> -1 11 -12 | 1 5
```

- You might want to specify the option `--destination=<dir>` to place the generated files in the directory `<dir>`. The short form is `-D<dir>`.
- You may specify the name of the contract file which should be written using the option `--output-file=<contractfile>` or simply `-o<contractfile>`.
- The option `--force` will overwrite an already existing contract file without any warning. It is, however, quite useful during debugging when you regenerate your files a lot.

The contract file. If the run of GoSam was successful can be read from the contract file. If everything went smoothly it should look like Listing 2. All options are either acknowledged by the word `OK` or, in case of a failure, by the word `error` followed by an error message.

The subprocesses receive an assignment to one or more labels per subprocess. In the line

```
1 -2 -> 21 11 -12 | 1 0
```

the suffix `| 1 0` states that this subprocess has been assigned to `1` label which has the number `0`. Had we set `SubdivideSubprocess` to `yes` this line might have looked like

```
1 -2 -> 21 11 -12 | 4 0 1 2 3
```

meaning that the subamplitudes[1] have been assigned to `4` labels (which is the first number after the bar) with the numbers `0` to `3`. These labels later form the first argument of the routine `OLP_Evalsubprocess`. In order to retrieve the full amplitude the calling program should sum over the contributions from all labels. Alternatively, it is allowed to sample the different channels by Monte Carlo techniques.

[1] GoSam at the moment only splits with respect to helicity subamplitudes.

4 “Make” the files

The following sequence of commands will generate and compile the matrix element files:

```
sh ./autogen.sh --prefix=‘pwd‘
make install
```

Now you should find a subdirectory `lib/` containing the files

- `libgolem_olp.a` for static linking,
- `libgolem_olp.so` for dynamic linking,
- `libgolem_olp.la` for linking with libtool.

You can set up a variable `LIBGOLEM` by the following command:

```
export LIBGOLEM=\
"-L‘pwd‘/lib/␣-lgolem_olp␣‘sh␣./config.sh␣-olibs‘"
```

A similar construction would also work inside a make-file:

```
GOLEM_ME_DIR=<path-to-config.sh>
LIBGOLEM="-L${GOLEM_ME_DIR}/lib/␣-lgolem_olp␣\
␣␣␣‘sh␣${GOLEM_ME_DIR}/config.sh␣-olibs‘"
```

5 Call the interface routines

There are slight differences in naming (underscoring) and calling conventions (call by reference vs. call by value) depending on the extensions in use. For `--mc=powhegbox` the extension `f77` is automatically included and therefore the underscoring works such that `gfortran` used as a Fortran 77 compiler would not complain. For all other Monte Carlo programs we follow the C/C++ conventions (see `olp.h`)

5.1 Initialization

The generated GoSam library is initialized with the call

```
        call OLP_Start("path/to/contract.olc",ierr)
```

The variable `ierr` should be declared as integer. If the contract file is not found, `ierr` is set to a negative value. A non-negative value indicates success.

If the contract file contains the option `ModelFile`, which should point to a SLHA file, the matrix element code tries to load the parameters from that file.

! $\rightarrow$ Calling `OLP_Start` is mandatory even if the contract file is not present or not read.

5.2 Setting options (optional)

Parameters can be passed by calling `OLP_Option`.

```
        call OLP_Option("name=value",ierr)
```

! → The initialization of derived parameters only works correctly if the according input parameters are set with `OLP_Option` *before* `OLP_Start` is called.

Example:

```
      call OLP_Option("mZ=91.234",ierr)
      call OLP_Option("mW=80.123",ierr)
c at this point sin(theta_w) is not up to date.
      call OLP_Start("path/to/contract.olc",ierr)
c now sin(theta_w) is set consistently
```

Some options can be changed at any time: please, have a look inside `model.f90` to get a complete understanding about the available parameter names and about the impact of their settings.

5.3 Computing the matrix element

The routine which returns a value for the matrix element is `OLP_Evalsubprocess`:

```
      integer ilab
      double precision moms(5*nlegs)
      double precision mu,params(1)
      double precision res(4)
c     ...
      call OLP_Evalsubprocess(
     &        ilab,moms,mu,params,res)
```

The first argument, `ilab` is one of the labels from the contract file. The momenta are passed in the argument `moms`, which has the format

$$(/E_1, p_1^x, p_1^y, p_1^z, m_1, E_2, p_2^x, p_2^y, p_2^z, m_2, \ldots E_N, p_N^x, p_N^y, p_N^z, m_N/)$$

The momenta are expected in physical kinematics: $p_1+p_2 = p_3+\ldots+p_N$. The components are in units of GeV.

The argument `mu` is the renormalisation scale μ (not μ^2!) in GeV. The argument params is an array of which the first argument is $\alpha_s(\mu)$. Any further array entries are ignored at the moment[2].

The last argument is an array of length four which is filled by the subroutine with the result of the evaluation. The entries have as a unit some power of GeV (1GeV$^($4 $-$ N)).

$$\mathcal{M}_B^\dagger \mathcal{M}_B = \texttt{res}(4) \tag{1}$$

$$2\mathrm{Re}\left(\mathcal{M}_B^\dagger \mathcal{M}_V\right) = \frac{(4\pi)^\varepsilon}{\Gamma(1-\varepsilon)}\left(\frac{\texttt{res}(1)}{\varepsilon^2} + \frac{\texttt{res}(2)}{\varepsilon} + \texttt{res}(3)\right) \tag{2}$$

This means that the coefficients `amp(1:3)` contain an explicit factor of $\alpha_s(\mu)/(4\pi)$.

5.4 Finalize (optional)

There is also a routine `OLP_Finalize` which is only needed if the client code needs to call `OLP_Start` more than once, e.g.

[2] Passing more than one parameter is implemented by the `Parameters` option in the order file, which is currently not part of the BLHA standard.

```
       do i=1,max_i
          write(line,'(A3,F6.3)') "mZ=", mZ(i)
          call OLP_Option(line,ierr)
c Need OLP_Start to update dependent parameters
          call OLP_Start(name,ierr)
c         ...
          call OLP_Finalize()
       enddo
```