Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and One-Loop Programs

S. Alioli [a], S. Badger [b], J. Bellm [c], B. Biedermann [d], F. Boudjema [e], G. Cullen [f], A. Denner [g], H. van Deurzen [h], S. Dittmaier [i], R. Frederix [j], S. Frixione [j], M.V.Garzelli [k], S. Gieseke [c], E.W.N. Glover [p], N. Greiner [h], G. Heinrich [h], V. Hirschi [l], S. Höche [m], J. Huston [n], H. Ita [i], N. Kauer [o], F. Krauss [p], G. Luisoni [h], D. Maître [p], F. Maltoni [q], P. Nason [t], C. Oleari [s], R. Pittau [r], S. Plätzer [u], S. Pozzorini [v], L. Reina [w], C. Reuschle [c], T. Robens [x], J. Schlenk [h], M. Schönherr [p], F. Siegert [i], J.F.von Soden-Fraunhofen [h], F. Tackmann [u], F. Tramontano [y], P. Uwer [d], G. Salam [j], P. Skands [j], S. Weinzierl [z], J. Winter [h], V. Yundin [b], G. Zanderighi [aa], M. Zaro [q]

[a] *Lawrence Berkeley National Laboratory and University of California, Berkeley, CA 94720, USA*

[b] *The Niels Bohr Institute,University of Copenhagen, DK-2100 Copenhagen, Denmark*

[c] *KIT Karlsruhe, Germany*

[d] *Humboldt-Universität zu Berlin, Institut für Physik, D-12489 Berlin, Germany*

[e] *LAPTH, Université de Savoie and CNRS, F-74941 Annecy-le-Vieux, France*

[f] *Deutsches Elektronen-Synchrotron DESY, Platanenallee 6, 15738 Zeuthen, Germany*

[g] *Universität Würzburg, Institut für Theoretische Physik und Astrophysik,D-97074 Würzburg, Germany*

[h] *Max Planck Institute for Physics, 80805 Munich, Germany*

[i] *Albert-Ludwigs-Universität Freiburg, Physikalisches Institut, D-79104 Freiburg, Germany*

[j] *CERN/PH, CH–1211 Geneva 23, Switzerland*

[k] *University of Nova Gorica, SI 5000 Nova Gorica, Slovenia*

[l] *EPFL Lausanne, Switzerland*

[m] *SLAC, Stanford University, Stanford, CA 94309, USA*

Preprint submitted to Elsevier *13 August 2013*

[n] *Michigan State University, East Lansing, MI 48840, USA*

[o] *Department of Physics, Royal Holloway, University of London, Egham TW20 0EX, UK*

[p] *Institute for Particle Physics Phenomenology, University of Durham, Durham, DH1 3LE, UK*

[q] *CP3, Université Catholique de Louvain,1348 Louvain-la-Neuve, Belgium*

[r] *Departamento de Fisica Teorica y del Cosmos CAFPE, Universidad de Granada, E-18071 Granada, Spain*

[s] *Università di Milano-Bicocca and INFN, Sezione di Milano-Bicocca, 20126 Milano, Italy*

[t] *INFN, Sezione di Milano-Bicocca, 20126 Milano, Italy*

[u] *DESY Hamburg, Germany*

[v] *University of Zurich, Institute for Theoretical Physics, CH-8057 Zurich, Switzerland*

[w] *Florida State University, Tallahassee, FL 32306-4350, USA*

[x] *Technical University Dresden, Germany*

[y] *Dipartimento di Scienze Fisiche, Università degli studi di Napoli "Federico II" and INFN, Sezione di Napoli, I-80125 Napoli, Italy*

[z] *Prisma cluster of Excellence, Institute for Physics, Johannes-Gutenberg-Universität Mainz, D-55099 Mainz, Germany*

[aa] *Rudolf Peierls Centre for Theoretical Physics, Oxford OX13PN, UK*

draft version 13 August 2013

Abstract

We present an update of the Binoth Les Houches Accord (BLHA) to standardise the interface between Monte Carlo programs and codes providing one-loop matrix elements.

Key words: Monte Carlo tools, one-loop computations, Les Houches Accord, automation

[1] Corresponding author: gudrun@mpp.mpg.de

1 Introduction

The past years have seen an enormous progress in the development of programs providing next-to-leading order corrections for multi-particle final states. This is due to new developments concerning the calculation of one-loop amplitudes as well as important progress on the Monte Carlo side to account for real radiation at NLO. The modular structure of NLO calculations allows to share the tasks between a "One-Loop Provider (OLP)", providing the virtual corrections, and a Monte Carlo program (MC) taking care of all the parts which do not involve loops. To facilitate the cross-talk between those two engines, a standard interface has been worked out during the workshop on Physics at TeV Colliders at Les Houches in June 2009, called the "Binoth Les Houches Accord (BLHA)" [1].

Meanwhile, the use of this interface [2–16] and further developments in OLP and MC codes have brought up the necessity to extend it with further options. The aim of this article is to provide a public document where an update of the BLHA is proposed and conventions are defined to pass parameters, calculational schemes etc., and to return less inclusive information, such as matrix elements which are not summed over all colours and helicities.

2 Existing features of the interface

We do not aim at an exhaustive description of the complete framework of the original interface here, referring to [1] for more details. However, we sketch the main features any extension will build upon.

The interaction between an OLP and a MC proceeds in two phases: the *pre-runtime phase* where order and contract files are established, and the actual *runtime* phase. In the *pre-runtime phase*, the MC creates a file called `order file` containing information about the setup and the subprocesses it will need from the OLP to perform the computation. A subprocess can be either a partonic subprocess or a component thereof (e.g. a specific helicity amplitude or a colour partial amplitude). The particles are identified by specifying their particle data group (PDG) code.

A flowchart of the setup between Monte Carlo program and One Loop Provider (where the new functions defined in BLHA version 2 are included already) is shown in Fig. 1. As an example, an order file written by SHERPA for $pp \to (Z \to e^+e^-) + 1\,\text{jet}$, using the original standards, is given in Fig. 2.

The OLP reads the order file and checks availability for each item. Then it

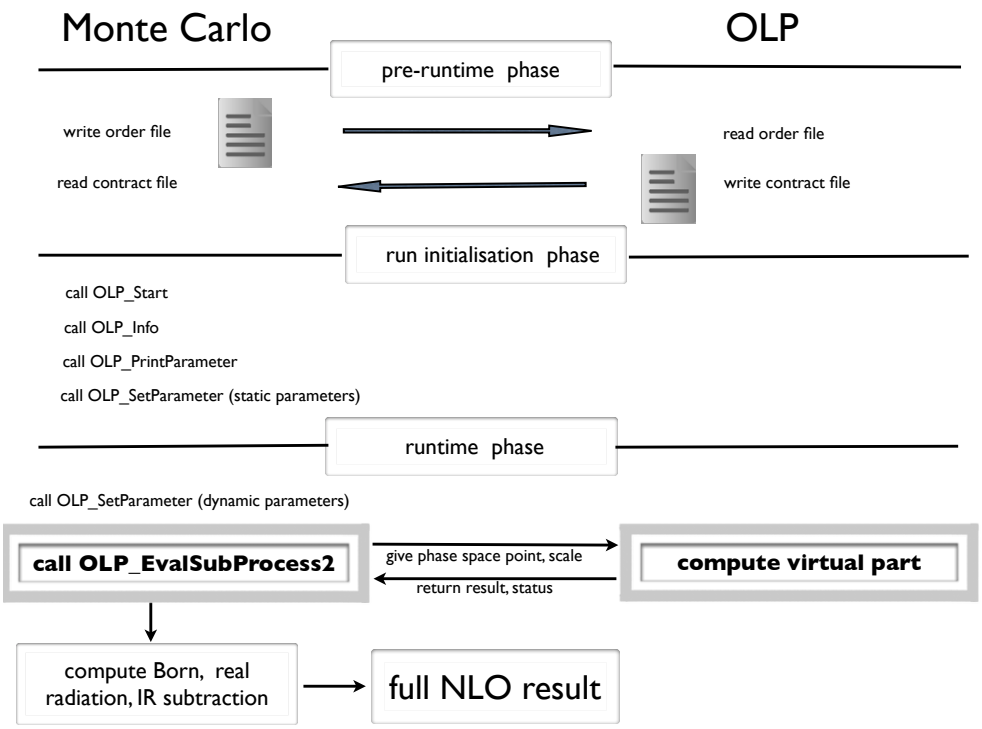

Fig. 1. Illustration of an interplay between Monte Carlo program and One Loop Provider (OLP). In the pre-runtime phase of the interface, the OLP receives an `order file` placed by the MC and checks availability of the contents. Then it returns a `contract file` to the MC where the contents are confirmed if available. At runtime, the Monte Carlo program provides Born, real radiation part minus infrared subtraction terms and integrated subtraction terms. The OLP provides the virtual amplitude for each phase space point. The phase space integration is done by the Monte Carlo program.

returns a `contract file` telling the MC what it can provide, and labelling the individual subprocesses. An example contract file generated by GoSam as a response to Sherpa's order file, looks like the one in Fig. 3. From this file, one can already see where an upgrade of the interface is clearly needed: as the original interface version did not contain a standard way to pass parameters, the definition of masses and widths is marked as "`Ignored by OLP`" in the contract file. Certainly the parameters are passed in the actual calculation, but in a non-standardised way, as an individual agreement between the particular MC and OLP. How to define a standard for the passing of parameters is described in Section 3.1.

The first integer label after each subprocess specifies that this subprocess contains only one component (if it was composed e.g. of several helicity configurations to be evaluated separately, this first label would be an integer larger than one). The second integer acts as a label for each subprocess, used at runtime to call the individual subprocesses.

```
# OLE_order.lh
# Created by Sherpa-1.4.1

MatrixElementSquareType CHsummed
CorrectionType          QCD
IRregularisation        DRED
AlphasPower             1
AlphaPower              2
OperationMode           CouplingsStrippedOff

Z_mass                  91.188
Z_width                 2.49
W_mass                  80.419
W_width                 2.0476
sin_th_2                0.221051079833

# process list
1 -1 -> 11 -11 21
21 1 -> 11 -11 1
21 -1 -> 11 -11 -1
2 -2 -> 11 -11 21
21 2 -> 11 -11 2
21 -2 -> 11 -11 -2
```

Fig. 2. Example of a `BLHA1` order file for the process Z+jet, created by SHERPA.

After the contract has been "signed", the communication between MC and OLP proceeds via function calls. Signing a contract means that the OLP basically copies the order file and appends an `OK` after each setting, separated by a "|" character. For the requested subprocesses, the OLP answer should be the integer labels described above, denoting the components and the labels of the individual subprocesses. If a setting is not supported or a subprocess is not available, the OLP should indicate this with `Error` instead of `OK` after the setting. Preferably, the message "`Error`" should be supplemented by a specification of the error, like `unsupported flag`, `process not available`, etc. More examples can be found in [1]. In any case, the program should not proceed if `Error` appears in the contract file.

If the contract file does not contain any `Error` statements, the communication via function calls can be started. In the original standard, there were only two functions which allowed the transfer of information between the two programs. One was the function `OLP_Start(char* fname, int* ierr)` which should be called by the MC when initializing the runtime phase. The character string in the first argument contains the name of the contract file. The integer in the second argument is set to 1 by the function call if the contract file is accepted. In case of failure, the second argument is different from one, and an error message of the type `Error: can not handle contract file` should

```
# vim: syntax=olp
#@OLP GOSAM 1.0
#@IgnoreUnknown True
#@IgnoreCase False
IRregularisation DRED | OK
AlphaPower 2 | OK
sin_th_2 0.221051079833 | OK # Ignored by OLP
Z_width 2.49 | OK # Ignored by OLP
Z_mass 91.188 | OK # Ignored by OLP
W_mass 80.419 | OK # Ignored by OLP
CorrectionType QCD | OK
AplhasPower  1 | OK
W_width 2.0476 | OK # Ignored by OLP
OperationMode CouplingsStrippedOff | OK
MatrixElementSquareType CHsummed | OK
1 -1 -> 11 -11 21 | 1 3
21 1 -> 11 -11 1 | 1 4
21 -1 -> 11 -11 -1 | 1 5
2 -2 -> 11 -11 21 | 1 0
21 2 -> 11 -11 2 | 1 1
21 -2 -> 11 -11 -2 | 1 2
```

Fig. 3. Example of a `BLHA1` contract file for the process Z+jet, created by GoSam. As the original interface did not define a standard way how to pass parameters, the definition of masses and widths is marked as "`Ignored by OLP`" in the contract file, while the parameters are passed in a non-standardised way.

be issued.

The second function which was already in place, but will have a different argument list with the new standards, is the function `OLP_EvalSubProcess`. The parameters to be passed to the `OLP_EvalSubProcess` function according to the original version of the interface were (in this order):

- The integer label of the subprocess (as given in the contract file).
- An array containing the components of the momenta. The momenta are placed in a one dimensional array, where physical scattering kinematics is used, i.e. $k_1+k_2 = k_3+\cdots+k_m$. For each particle, the kinematics is specified by a 5-tuple: $(E_j, k_j^x, k_j^y, k_j^z, M_j)$. Thus a full m-particle event is specified by an array of $5 \times m$ double precision numbers filled with the m 5-tuples, ordered by the particle labels.
- The renormalisation scale, μ_{R}, as a double precision number, or an array of scales, if different scales need to be passed.
- The strong coupling $\alpha_{\mathrm{S}}(\mu_{\mathrm{R}})$, where $\alpha_{\mathrm{S}}(\mu_{\mathrm{R}}) = 1$ can be used to indicate that the `MC` multiplies the returned values with the adequate coupling constants.
- The array where the computed results are returned.

The returned array is expected to contain at least four real-valued double precision numbers

```
PoleCoeff2, PoleCoeff1, PoleCoeff0, BornSquare
```

which correspond to the colour- and helicity-summed (resp. averaged for the initial state) terms A_2, A_1, A_0, $|\mathrm{Born}|^2$.

The conventions for the overall prefactor are as in the original proposal [1] and are briefly repeated here. The general structure of the virtual correction is given by

$$\mathcal{I}\Big(\{k_j\}, \mathrm{R.S.}, \mu_{\mathrm{R}}^2, \alpha_{\mathrm{S}}(\mu_{\mathrm{R}}^2), \alpha, \ldots\Big) = C(\epsilon)\left(\frac{A_2}{\epsilon^2} + \frac{A_1}{\epsilon} + A_0\right), \tag{1}$$

where R.S. defines the infrared regularisation scheme. The Laurent coefficients A_j are real-valued. The overall constant is given by

$$C(\epsilon) = \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)}\left(\frac{\mu^2}{\mu_{\mathrm{R}}^2}\right)^\epsilon = (4\pi)^\epsilon \frac{\Gamma(1+\epsilon)\Gamma(1-\epsilon)^2}{\Gamma(1-2\epsilon)}\left(\frac{\mu^2}{\mu_{\mathrm{R}}^2}\right)^\epsilon. \tag{2}$$

3 New features of the interface

The new features should serve to pass more detailed information between the MC and the OLP. However, it should always be kept in mind that the general setup is such that the MC is steering the calculation. The MC orders the virtual amplitude for the given settings, and gets back the result and a relative accuracy informing about the quality of the result. The interface does not foresee a setup where the OLP can cause changes in the MC settings.

For the *pre-runtime phase*, we define a number of new keywords to allow for more options in the order/contract files. The valid keywords are listed in Appendix A.

As the new standards are not backwards compatible, we propose to place the keyword `InterfaceVersion`, which can take the values `BLHA1` or `BLHA2`, on top of the order file. This way, if the OLP does not support one or the other, it can issue an error message and stop without proceeding further.

Once order and contract files are established, one can distinguish two phases: The *run initialization phase* and the actual *runtime phase*. The functions which are used for the communication between MC and OLP before exchang-

ing phase space points and results are the following (described in more detail in the subsections below and listed in Appendix B):

- `OLP_Start(char* fname, int* ierr)`: same as in `BLHA1`.
- `OLP_Info(char olp_name[15],char olp_version[15],char message[255])`: the function serves to keep track of the type and version of the OLP which has been used, and to encourage proper citation. The arguments are the name of the OLP, the version, and a string which contains information about the relevant publications, for example the bibtex identifier.
- `OLP_SetParameter(char* para, double* re, double* im, int* ierr)`: This function is used to define static parameters at the beginning of a run, and to exchange dynamic parameters at runtime.
- `OLP_PrintParameter(char* filename)`: prints out a list of the actual parameter settings to the file *filename*.

At *runtime*, the OLP returns the values for the virtual amplitude to the MC via the function `OLP_EvalSubProcess2`. As compared to the original function `OLP_EvalSubProcess` described in Section 2, the new function `OLP_EvalSubProcess2` does not contain the passing of coupling constants anymore, as their values are now passed separately, using `OLP_SetParameter`. It also has a new argument appended which is useful to assess the accuracy of the result returned by the OLP. With the new argument list, the function is not backwards compatible with the original standard. Therefore, to avoid confusion with different versions, the new function is called `OLP_EvalSubProcess2`.

```
OLP_EvalSubProcess2(int* i,double* pp,double* mu,double* rval,double* acc)
```

The arguments are:

- i: pointer to a (one element) array with the label of the subprocess as given in the contract file
- pp: pointer to an array of momenta, conventions $(E_j, k_j^x, k_j^y, k_j^z, M_j)$
- mu: pointer to the renormalisation scale
- rval: pointer to an array of return values
- acc: pointer to a one element array with the outcome of the OLP internal accuracy check (see Section 3.3).

Note: originally, the argument list of `OLP_EvalSubProcess` contained both μ_r and $\alpha_s(\mu_r)$: `OLP_EvalSubProcess(int i,double* pp,double mu,double alphas,double* rval)`. However, $\alpha_s(\mu_r)$ can now be set using the new function `OLP_SetParameter` to pass also dynamical parameters. This setup is also clearer for mixed (e.g. QCD-EW) corrections or corrections where α_s at different scales should be used within the same calculation.

The length of the array `rval` is at least four, containing the Laurent coefficients A_2, A_1, A_0 and $|\text{Born}|^2$. However, for the case where colour/spin correlated matrix elements are returned, the array `rval` must be longer. Details about the labelling conventions for such cases are given in Section 3.8.

The last argument `acc` should return information about the OLP internal accuracy check(s), denoting the relative accuracy of the virtual amplitude at this phase space point as estimated by the OLP. OLPs which only provide a binary stability test will return 0. for *passed*, or a large number, say 10^5, for *failed*. More details about unstable phase space points are given in Section 3.3.

The list of defined functions is given in Appendix B, where the `C/C++` version (with pointers in the argument list) is given. The `C++` version with function calls by reference as well as a `Fortran (2003)` module for binding with `C/C++` can be found at `http://phystev.in2p3.fr/wiki/2013:groups:sm:blha:api`.

3.1 Passing parameters

In the first version of the interface, the standard only allowed to pass a fixed amount of information at the level of the order/contract files. However, to be able to pass also dynamical parameters like running masses, and to have more flexibility in the definition of individual parameters, we suggest the following extension.

Parameters can be passed by the function

```
OLP_SetParameter(char* para, double* re, double* im, int* ierr)
```

where the first argument is a (pointer to a) string serving as a keyword for the parameter to be set, followed by two double precision numbers so that complex parameters can also be passed (in case of real parameters, the second double is zero). The integer in the fourth argument is set by the OLP to tell the MC whether the setting of the parameter was successful.

`ierr=1` means the parameter has been set successfully,
`ierr=0` means failure: issue an error message,
`ierr=2` means that the parameter is unknown or the setting is ignored (for example because it is irrelevant for the considered case), but the MC program should proceed.

The function `OLP_SetParameter` can be called at runtime, for every phase space point, if used to define a dynamic parameter. Obviously it can also be called once (for each particular parameter that needs to be passed) if this is a

static parameter needed only at the run initialization phase.

Further, we propose a routine `OLP_PrintParameter(char* filename)` giving out a list of the actual parameter settings used in the calculation, where `filename` is the name of the output file. The intention of this function is just to inform the user, consistency between the parameters will not be checked. The output format of `OLP_PrintParameter` should be
`ParameterName Value State`,
where the separator is a space, and `Value` can be complex, denoted by `(real part, imaginary part)` in case it is complex. `State` can serve to distinguish the parameters set by `OLP_SetParameter` or defined by `Model` from fixed internal ones and is optional (as this info may not be readily available in all programs).

3.2 Defining the model

We distinguish two alternative ways of model definition, which we will denote by "keyword model" respectively "`UFO` model" in the following.

Model definitions offer the possibility to define some global settings in the order file, which are intrinsic to the model (e.g. SM, MSSM), which is used. This is done using the required keyword `Model`. For example, `Model: SMdiag` should set the CKM matrix to unity globally.

In the "keyword model" setup, the parameters that need to be set within a certain model are passed via PDG codes [17] and keywords with naming conventions as specified in Fig. 4 for the Standard Model. The numbers in parenthesis after `mass` and `width` denote the particle's PDG code.

In the "`UFO` model" setup, the parameters are defined in `UFO` (Universal Feynrules Output) [18] format, which is particularly useful for calculations beyond the Standard Model. The import of the `UFO` model file should be specified in the `order file` by
`Model ufo:/path_to_ufo_model-directory/`.

The `UFO` format also provides human readable name attributes for the model parameters, as well as the SLHA identifiers [19–21]. OLPs which support the import of `UFO` model files typically also support the name attributes. The `UFO` model setup entails the use of a SLHA parameter card to initialize the runtime phase. This requires an additional keyword `ParameterCard`, followed by the path to the SLHA parameter card, to be placed into the order file when using the `UFO` model setup. The parameters which are set by reading in the SLHA parameter card do not need to be set again by `OLP_SetParameter`. However, `OLP_SetParameter` needs to be used at runtime for the dynamic

keyword	parameter
`mass(5)`	b quark mass
`mass(6)`	top quark mass
`width(6)`	top quark width
`sw2`	$\sin^2\theta_w$
`vev`	SM vacuum expectation value
`Gf`	G_{Fermi}
`VV12`	V_{ud}
⋮	

Fig. 4. List of keywords to define parameters to be passed by the function `OLP_SetParameter`.

parameters. In this case the SLHA block name should appear as a prefix prepended to the parameter name, in the form `<BlockName>&&<ParamName>`. To avoid confusion, this requires that the characters '`&&`' should never appear in any block or parameter name.

Note that we do not require the capability to import `UFO` model files to belong to the "minimal standard". However, at least one of the two ways described above to define the model parameters should be implemented to comply with the standard.

3.3 Treatment of unstable phase-space points

In a complex multi-leg one-loop calculation, some of the phase space points may lead to large numerical cancellations within the virtual amplitude, leading to a result which is not trustworthy at this particular point. It is the task of the OLP to make precision tests, such that it can trigger some rescue procedure in case the phase space point is found to be problematic. In order to steer precision requirements from the order file, we introduce the following:

- The OLP should pass information about the quality of the virtual result at a given phase space point back to the MC, by means of the function `OLP_EvalSubProcess2`. The last argument of `OLP_EvalSubProcess2` should be a (double precision) number "`acc`", giving the relative accuracy the OLP attributes to the finite part of the loop amplitude at this phase space point. Note that this value will depend on the type of internal stability test(s) made by the OLP. The assessment of the internal "relative accuracy" can therefore vary between different OLPs and even within the same OLP if it

can switch between different types of stability tests. However, it allows the MC to give a more reliable overall accuracy estimate for the cross section than if it had just a "pass" or "fail" information from the OLP side.

- The user should be able to specify the overall accuracy he would like to reach (this is typically done in the MC runcard). For this purpose there should be an (optional) keyword in the order file which specifies the target accuracy, called `AccuracyTarget`, denoting a relative accuracy. The OLP can use it to decide whether an internal rescue system should be triggered if its stability test outcome gives a value larger than `AccuracyTarget`. If `AccuracyTarget` is not specified, the OLP will use its internal settings to decide if a rescue system should be triggered.

  It could be that a particular OLP is not able to provide a number for the relative accuracy, but only can deliver a "binary" test outcome *passed* or *failed*. In this case, `acc=0.0` stands for *passed*, and in the case of *failed*, `acc` can be set to a large value, for example 10^5. In order to allow the MC to distinguish the "binary test" return value for *passed* from a "quantitative test" return value for *passed*, we can require that quantitative stability tests will return a perfect accuracy as working precision (typically 1d-17, 1d-34), while binary stability tests return exactly 0.0.
- There should be the possibility that the OLP prints points classified as "unstable" to a file for monitoring purposes. Therefore we propose an optional keyword `DebugUnstable` in the order file which can be used to keep information about points classified as "unstable". In the case of `DebugUnstable True`, the phase space point should be printed to a file to allow further diagnostics. The format of such a debug file can be defined internally within the OLP. The threshold which defines whether a point is classified as "unstable" (after eventual rescue attempts) does not need to be equal to the value of `AccuracyTarget`, but this should be the default. In any case, `DebugUnstable` is mostly for OLP developers for monitoring purposes.

We emphasize again that after all, it is left to the MC to decide what to do with the phase space point, based on `acc` returned by `OLP_EvalSubProcess2`.

3.4 Different powers of coupling constants, merging different jet multiplicities

So far, the interface was tailored to NLO calculations for a fixed jet multiplicity, and focused on QCD corrections rather than electroweak corrections. However, mixed QCD-EW corrections, or expansions in parameters other than α_s or α, require a more flexible scheme to define the desired orders in coupling constants.

In addition, recent developments [15, 22–27] propose a merging method for matched NLO predictions with varying jet multiplicity. In order to calculate

merged samples, the Monte Carlo program needs to ask the OLP for one-loop matrix elements with different jet multiplicities and therefore different powers of the coupling constant.

```
CouplingPower QCD  2
# process list 2j
1 1 -> 1 1
1 -1 -> 1 -1
1 -1 -> 2 -2
1 -1 -> 21 21
21 21 -> 21 21

CouplingPower QCD  3
# process list 3j
1 1 -> 21 1 1
1 -1 -> 21 1 -1
1 -1 -> 21 2 -2
1 -1 -> 21 21 21
21 21 -> 21 21 21

CouplingPower QCD  4
AmplitudeType   Tree
# process list 4j
21 21 -> 21 21 21 21
1 -1 -> 21 21 21 21
1 -1 -> -2 2 -2 2
1 -1 -> -1 1 -1 1
21 21 -> 21 21 21 21
```

Fig. 5. Example of the part of an order file containing different settings for different sets of subprocesses.

These situations can be accounted for by allowing different settings for different subprocesses, see Fig. 5. A setting is valid until it is explicitly overwritten. This setup can be used for merged samples as well as mixed QCD/EW corrections. It can also be used to pass additional information referring only to particular subprocesses, as indicated e.g. by `AmplitudeType Tree`. In the latter case, the first three elements of the returned array, i.e. A_2, A_1, A_0, will be zero.

3.5 Extras

The keyword `Extra` can be used to write special requirements relevant to the OLP into the order file. If there is a way to generate the order file for the OLP from the MC run card, the MC will write them to the order file, but otherwise ignore them. The difference between *optional keywords* and keywords preceded

by the "`Extra`" flag is that optional keywords have defined names (as listed in Appendix A.2), while keywords flagged by "`Extra`" can be very OLP specific and do not have a standard name.

For example, requirements concerning a colour expansion, or passing only particular helicity configurations, could be put under the `Extra` flag. The `Extra` flag can also be used to define that MC and OLP should use the MiNLO [28] procedure, if the latter is available on both sides.

With regards to overall averaging and symmetry factors, the default is that amplitudes are summed over final state colours and polarisations and averaged over initial state colours and polarisations, and that symmetry factors for identical final state particles are included. A possibility to change this could look as follows:

```
Extra HelAvgInitial False
Extra ColAvgInitial False
Extra MCSymmetrizeFinal False
```

3.6 Electroweak corrections

In the case of electroweak (EW) corrections, it is of particular importance to check the consistency of the parameters, for example the relation between M_Z, M_W and $\sin^2\theta_w$. The scheme is set in the order file by the keyword `EWScheme`, which can take the values `alphaGF` (also known as G_μ-scheme), `alpha0`, `alphaMZ`, `alphaRUN`, `alphaMSbar`, `OLPDefined`. Then the parameters are set using `OLP_SetParameter`. The OLP imports these parameters. The integer in the last argument only indicates if the import was successful or not. It is left to the user to ensure consistency of the parameters within the given scheme.

3.7 An example order file

An example for an order file is shown in Fig. 6. The lines following the keyword `Extra` stand for any extra information which may be relevant to the OLP (which, in this example, is parsed from the Sherpa runcard to the order file).

```
# OLE_order.lh
# Created by Sherpa-2.0.0

InterfaceVersion         BLHA2
Model                    SMdiag
AmplitudeType            CHsummed
CorrectionType           QCD
IRregularisation         DRED
WidthScheme              ComplexMass
EWScheme                 alphaGF
AccuracyTarget           0.0001
DebugUnstable            True
Extra                    Line1
Extra                    Line2

# process list
CouplingPower QCD        2
CouplingPower QED        0
1 -1 -> 6 -6
-1 1 -> 6 -6
21 21 -> 6 -6
CouplingPower QCD        3
CouplingPower QED        0
1 -1 -> 6 -6 21
1 21 -> 6 -6 1
-1 1 -> 6 -6 21
-1 21 -> 6 -6 -1
21 1 -> 6 -6 1
21 -1 -> 6 -6 -1
21 21 -> 6 -6 21
```

Fig. 6. Example of an order file for $t\bar{t} + 0, 1$ jets produced by `Sherpa-2.0.0` . The expressions `Line1,Line2` following the keyword `Extra` denote any extra information which may be relevant for the OLP.

3.8 Colour- and spin correlated tree amplitudes

Going beyond the generic case of colour and helicity summed matrix elements, it becomes difficult to satisfy special needs of different programs with one global standard. Results for polarized amplitudes in general depend on phase conventions, and approximations in a colour expansion, like e.g. "leading colour", will very likely be defined differently in different codes. Nonetheless, the interface in principle allows to pass very specific information using the `Extra` flag. However, the details probably remain to be implemented and tested individually between specific programs before trying to establish any standards.

Below we only focus on a particular example going beyond `CHsummed`, which is passing colour- and spin correlated tree amplitudes. The definitions are oriented at the case where they are used for the construction of NLO subtraction terms in the Catani-Seymour formalism [29].

The MC can ask for colour- or spin correlated tree matrix elements by writing one (or several) of the following keywords into the order file:

```
AmplitudeType ccTree   # colour correlated tree amplitude
AmplitudeType scTree   # spin correlated tree amplitude
```

The defaults for `AmplitudeType` are `Loop` and `CHSummed`. If only `Tree` or `Loop` are specified, this automatically means `CHSummed`.

The new setup to define individual settings for blocks of subprocesses can be used to call certain subprocesses twice: one for the colour/spin correlated tree case and one for the loop case. An example is given by

```
AmplitudeType ccTree
1 -1 -> 21 1 -1
21 21 -> 21 21 21

AmplitudeType Loop
1 -1 -> 21 1 -1
21 21 -> 21 21 21
```

If the amplitude returned by the OLP is not colour/polarisation summed, the values to be returned by `OLP_EvalSubProcess2` form a matrix in colour respectively Lorentz space. To pass the values in an unambiguous way, it is necessary to define the order in which the matrix elements are written into the array `rval` returned by the OLP.

Colour correlations

The colour correlated matrix elements

$$C_{ij} = \langle \mathcal{M} | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M} \rangle \tag{3}$$

can be defined to be real valued quantities and independent of the particular colour basis chosen. Note that $\mathbf{T}_i \cdot \mathbf{T}_j$ is symmetric under exchange of i and j. Further, $\mathbf{T}_i^2 = C_i \mathbf{1}$ with $C_i = C_F$ if the leg i belongs to the $\mathbf{3}$ or $\bar{\mathbf{3}}$ representation of SU(3), and $C_i = C_A$ if the leg belongs to the $\mathbf{8}$ representation, and $\langle \mathcal{M} | \mathbf{T}_i \cdot \mathbf{T}_i | \mathcal{M} \rangle = C_i \langle \mathcal{M} | \mathcal{M} \rangle = C_i |\mathcal{M}|^2$ is just proportional to the tree level amplitude squared.

The minimal information that needs to be passed for colour correlated matrix elements are the upper non-diagonal $n(n-1)/2$ elements of an $n \times n$ matrix, where n is the number of legs attached to the process of interest. Entries belonging to non-coloured legs should be ignored.

For a process which is flagged as `AmplitudeType ccTree` the OLP should (through `rval` returned by `OLP_EvalSubProcess2`) return an array of length $n(n-1)/2$ such that the element at position $i+j(j-1)/2$ (counting external legs and elements in the array starting from zero) contains the result for C_{ij} with $i < j$.

Here and in the following we assume that i labels the emitter and j the spectator.

Spin correlations

When gluons are present at the Born level, the spin correlated Born amplitude is obtained by Lorentz-contracting the corresponding Born terms with the gluon polarization vectors. It can be shown that these amplitudes can be written in terms of

$$S_{ij} = \langle \mathcal{M}_{i,-} | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M}_{i,+} \rangle \,, \tag{4}$$

with

$$\begin{aligned} &\langle \mathcal{M}_{i,-} | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M}_{i,+} \rangle = \\ &\sum_{\lambda_1,\ldots,\lambda_{i-1},\lambda_{i+1},\ldots,\lambda_n} {}_C\langle \mathcal{M}_{\lambda_1,\ldots,\lambda_{i-1},-,\lambda_{i+1},\ldots,\lambda_n} | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M}_{\lambda_1,\ldots,\lambda_{i-1},+,\lambda_{i+1},\ldots,\lambda_n} \rangle_C \,. \end{aligned} \tag{5}$$

Here, $|\rangle_C$ indicates an object which is a vector in colour space only, and we have made explicit the spin dependence by the subscripts λ and the fixed $\pm$ polarization of the gluon i of interest. Depending on the organisation of the subtraction terms, non-trivial colour correlations $i \neq j$ may be needed. Since i is defined to be the gluon index, we have $S_{ij} \neq S_{ji}$ and $S_{ij} \in \mathbb{C}$, so in general the communication of a complex $n \times n$ matrix is required. As for the plain colour correlated matrix elements, entries corresponding to non-coloured legs or coloured legs for which the infrared singular limit does not involve spin correlations (*i.e.* everything but gluons for all practical purposes) should be ignored.

For a process flagged as `AmplitudeType scTree` the OLP should (through the standard `OLP_EvalSubProcess2` method) return an array of length $2n^2$, such that the elements at positions $2i + 2nj$ and $2i + 2nj + 1$ (counting external legs and elements in the array starting from zero) contain $\mathrm{Re}(S_{ij})$ and $\mathrm{Im}(S_{ij})$, respectively.

A consistent setup of spin correlations in this way further requires the MC code to obtain the gluon polarization vector $\epsilon^\mu_+(p,q)$ as used by the OLP, given the gluon momentum p and a reference vector q. The gluon ploarisation vectors can be passed via a dedicated function

```
OLP_Polvec(double* p, double* q, double* eps)
```

where `p` and `q` are arrays of size 4, denoting the momentum p of gluon i and the corresponding reference vector, and `eps` is an array of size 8 containing the four complex components of $\epsilon^\mu_\pm(p,q) = \pm\frac{1}{\sqrt{2}}\frac{\langle q^\mp|\gamma^\mu|p^\mp\rangle}{\langle q^\mp|p^\pm\rangle}$, in a form where real and imaginary parts are alternating.

Returning all possible colour and spin correlations could potentially become a significant overhead; we leave it to the OLP code to support the calculation of a single correlator. Information about the correlator to be returned could be passed via the `OLP_SetParameter` method right before the `OLP_EvalSubProcess2` call in question.

Obviously, if one of the contractors cannot provide the detailed colour or helicity information requested, the calculation should exit at the stage of `OLP_Start`.

3.9 Restrictions such as diagram filters, exploitation of special symmetries, etc.

The keyword `ExcludedParticles` in the order file can be used to remove unwanted particles from the code generation. Restrictions like confining the set of diagrams to resonant diagrams only can be set in the MC input card. OLP specific restrictions can be imposed using the keyword `Extra`.

4 Conclusions

This writeup summarises the update of the standard interface between Monte Carlo programs and one-loop matrix element providers which has been initiated at the Les Houches 2009 workshop on Physics at TeV Colliders, called the "Binoth Les Houches Accord (BLHA)". The setup meanwhile has been implemented by several groups and facilitates the automation of NLO calculations.

The experience gained meanwhile with the original setup fed into the discussion about an extension of the standards, such that the interface can be used

in a wider and more flexible context. The outcome of the discussion between a large number of Monte Carlo and One-Loop Providers (OLPs) is summarized in the present document, which is intended to serve as a reference for the new standards. This should increase the flexibility of both Monte Carlo programs to import virtual corrections where available and of OLPs to team up with different Monte Carlo programs. This is an important step forward, as different MCs and OLPs have different focus and strengths concerning for example the multiplicity of final states, particle masses, electroweak corrections, BSM capability, etc. We therefore hope that BLHA version 2 will contribute to the goal of extending the comparison of LHC data with theoretical results consistently beyond the leading order.

Acknowledgements

We would like to thank the CERN TH/LPCC Institute on SM at the LHC for hospitality and for providing a stimulating environment to discuss this interface. We would also like to thank the Les Houches 2013 organizers for providing a platform (e.g. Wikipages) and again a stimulating environment to work out this Accord. GH would like to thank everybody who contributed to the creation and implementation of the Accord, both BLHA1 and BLHA2.

A List of valid keywords for order/contract file

A.1 Required keywords

`InterfaceVersion:` can take the values `BLHA1` or `BLHA2`. This clarifies already in the pre-runtime phase if the new standards are supported by both MC and OLP.

`Model:` `SMdiag`, `SMnondiag`, `MSSM`, `ufo:/abs-path-to-ufo-file/`. For BSM standards, the `UFO` [18] format is most convenient.

`CouplingPower QCD:` integer which specifies the α_s power of the Born cross section. Can be used for sub-processes as well, where it also refers to cross section rather than amplitude level.

`CorrectionType:` the type of higher order correction. Standard values are `QCD`, `QED`, corresponding to expansions in α_s, α.

`IRregularisation:` the infrared regularisation scheme used. Possible choices for QCD are `CDR`, `DRED`.

`AccuracyTarget`: The target accuracy the user would like to achieve. This is a relative accuracy the OLP can use to decide if a rescue system should be triggered if the result of internal stability checks returns a value which is larger than `AccuracyTarget`.

`DebugUnstable`: can take the values `True` or `False`. Indicates that the unstable phase space points should be printed to a file for further diagnostics. In order to decide what is classified as "unstable phase space point", the threshold given by `AccuracyTarget` or an internal OLP threshold can be used.

`CouplingPower QED`: integer which specifies the α power of the Born cross section. The default is zero. Can also be used for other couplings in the form `CouplingPower gX` if the coupling `gX` is defined (for example through an `UFO` module). Note that `CouplingPower QCD` needs to be defined explicitly, while `CouplingPower QED` will be set to the default value (zero) if not specified.

`AmplitudeType`: the default is `Loop`. It can also take the values `ccTree` (colour correlated tree), `scTree` (spin correlated tree), `Tree`, `LoopInduced`. `Tree` without "`cc`" or "`sc`" implies `CHsummed`.

`Extra`: can be used to write special requirements relevant to the OLP into the order file. If the MC generates the order file from a run card where such requirements are specified, it will copy them into the order file but otherwise ignore them.

`ParameterCard`: gives the path to the SLHA parameter card if the `UFO` model setup is used.

`MassiveParticles`: defines a list of massive particles at the level of the order file, for example `MassiveParticles 5 6`. The separator is a space. This also implies that the light-quark masses are set to zero.

`LightMassiveParticles`: useful if mass regularisation instead of dimensional regularisation is used (e.g. for electroweak corrections). It defines the set of particles where only $\log(m)$ terms are kept, but not power suppressed ones.

`ExcludedParticles`: can be used to exclude particles which are contained by default in `Model`. The particles should be listed after the keyword, denoted by their PDG numbers, and separated by a space.

`MassiveParticleScheme`: a standard choice is `OnShell`.

`SubdivideSubprocess`: this flag defines whether a given process is represented in a split form to allow for multi-channel Monte Carlo sampling. May be useful e.g. in the case of colour expansions. Can take the values `True` or `False` (default).

`EWScheme`: used schemes (discussed in the text) can be flagged by the keywords `alpha0`, `alphaMZ`, `alphaGF`, `alphaRUN`, `alphaMSbar`, `OLPDefined` (default).

`WidthScheme`: defines the treatment of unstable particles. Standard values are `ComplexMass`, `FixedWidth`, `RunningWidth`, `PoleApprox`.

A.3 List of keywords contained in the original proposal which will be dismissed

`MatrixElementSquareType:` replaced by `AmplitudeType`, declared `CHsummed` as default. Originally this flag was intended to specify colour (C) and helicity (H) treatment. Possible values were defined as `CHsummed`, `Csummed`, `Hsummed`, `NOTsummed`.
The new default `CHsummed` also implies an average over initial state colours and polarizations. The flag `Extra` can be used to accommodate for the keywords `HelAvgInitial`, `ColAvgInitial`, `MCSymmetrizeFinal`, which can be set to `False` if the factors included by default should be switched off.

`ModelFile:` the model file from which parameters have to be read. Keyword replaced by `Model`.

`OperationMode:` the operating mode of the OLP. This optional flag was intended to specify OLP-defined conventions or approximations to the one-loop contribution. Typical operating modes are `CouplingsStrippedOff`, `LeadingColour`, `HighEnergyLimit`. The keyword `CouplingsStrippedOff` turned out to be ambiguous in the presence of EW couplings and therefore will not be used any longer.

`ResonanceTreatment:` has been replaced by `WidthScheme`, as, for example, the complex-mass scheme also affects non-resonant propagators.

B List of defined functions

We list here the definitions of the functions needed for a working interface in the `C/C++` version (with pointers in the argument list). A `Fortran (2003)` module for binding with `C/C++` can be found at
`http://phystev.in2p3.fr/wiki/2013:groups:sm:blha:api`.

```
#ifndef __OLP_H__
#define __OLP_H__

#ifdef __cplusplus
extern "C" {
#endif

void OLP_Start(char* fname, int* ierr);
void OLP_Info(char[15] olp_name,char[15] olp_version,char[255] message);
void OLP_SetParameter(char* para, double* re, double* im, int* ierr);
void OLP_PrintParameter(char* filename);
void OLP_Polvec(double* p, double* n, double* eps); // (optional)
void OLP_EvalSubProcess2(int* i,double* pp,double* mu,double* rval,double* acc);

#ifdef __cplusplus
}
#endif // __cplusplus
#endif // __OLP_H__
```

References

[1] T. Binoth, F. Boudjema, G. Dissertori, A. Lazopoulos, A. Denner, *et al.*, *A Proposal for a standard interface between Monte Carlo tools and one-loop programs*, *Comput.Phys.Commun.* **181** (2010) 1612–1622, [1001.1307].

[2] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, *et al.*, *Event generation with SHERPA 1.1*, *JHEP* **0902** (2009) 007, [0811.4622].

[3] T. Gleisberg and F. Krauss, *Automating dipole subtraction for QCD NLO calculations*, *Eur.Phys.J.* **C53** (2008) 501–523, [0709.2881].

[4] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, *et al.*, *Precise Predictions for W + 3 Jet Production at Hadron Colliders*, *Phys.Rev.Lett.* **102** (2009) 222001, [0902.2760].

[5] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, *et al.*, *Precise Predictions for W + 4 Jet Production at the Large Hadron Collider*, *Phys.Rev.Lett.* **106** (2011) 092001, [1009.2338].

[6] H. Ita, Z. Bern, L. Dixon, F. Febres Cordero, D. Kosower, *et al.*, *Precise Predictions for Z + 4 Jets at Hadron Colliders*, *Phys.Rev.* **D85** (2012) 031501, [1108.2229].

[7] Z. Bern, G. Diana, L. Dixon, F. Febres Cordero, S. Hoeche, *et al.*, *Four-Jet Production at the Large Hadron Collider at Next-to-Leading Order in QCD*, *Phys.Rev.Lett.* **109** (2012) 042001, [1112.3940].

[8] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, *et al.*, *Automated One-Loop Calculations with GoSam*, *Eur.Phys.J.* **C72** (2012) 1889, [1111.2034].

[9] S. Badger, B. Biedermann, P. Uwer, and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, *Comput.Phys.Commun.* **184** (2013) 1981–1998, [1209.0100].

[10] E. Re, *NLO corrections merged with parton showers for Z+2 jets production using the POWHEG method*, *JHEP* **1210** (2012) 031, [1204.5433].

[11] Z. Bern, L. Dixon, F. Febres Cordero, S. Hoeche, H. Ita, *et al.*, *Next-to-Leading Order W + 5-Jet Production at the LHC*, 1304.1253.

[12] R. Frederix, S. Frixione, F. Maltoni, and T. Stelzer, *Automation of next-to-leading order computations in QCD: The FKS subtraction*, *JHEP* **0910** (2009) 003, [0908.4272].

[13] V. Hirschi, R. Frederix, S. Frixione, M. V. Garzelli, F. Maltoni, *et al.*, *Automation of one-loop QCD corrections*, *JHEP* **1105** (2011) 044, [1103.0621].

[14] R. Frederix, S. Frixione, K. Melnikov, and G. Zanderighi, *NLO QCD corrections to five-jet production at LEP and the extraction of $\alpha_s(M_Z)$*, *JHEP* **1011** (2010) 050, [1008.5313].

[15] G. Luisoni, P. Nason, C. Oleari, and F. Tramontano, *Merging HW/HZ + 0 and 1 jet at NLO with no merging scale using the POWHEG BOX interfaced to GoSam*, 1306.2542.

[16] S. Hoeche, J. Huang, G. Luisoni, M. Schoenherr, and J. Winter, *Zero and one jet combined NLO analysis of the top quark forward-backward asymmetry*, 1306.2703.

[17] **Particle Data Group** Collaboration, J. Beringer *et al.*, *Review of Particle Physics (RPP)*, *Phys.Rev.* **D86** (2012) 010001.

[18] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, *et al.*, *UFO - The Universal FeynRules Output*, *Comput.Phys.Commun.* **183** (2012) 1201–1214, [1108.2040].

[19] P. Z. Skands, B. Allanach, H. Baer, C. Balazs, G. Belanger, *et al.*, *SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators*, *JHEP* **0407** (2004) 036, [hep-ph/0311123].

[20] T. Hahn, *SUSY Les Houches Accord 2 I/O made easy*, *Comput.Phys.Commun.* **180** (2009) 1681–1693, [hep-ph/0605049].

[21] B. Allanach, C. Balazs, G. Belanger, M. Bernhardt, F. Boudjema, *et al.*, *SUSY Les Houches Accord 2*, *Comput.Phys.Commun.* **180** (2009) 8–25, [0801.0045].

[22] S. Hoeche, F. Krauss, M. Schonherr, and F. Siegert, *QCD matrix elements + parton showers: The NLO case*, *JHEP* **1304** (2013) 027, [1207.5030].

[23] T. Gehrmann, S. Hoche, F. Krauss, M. Schonherr, and F. Siegert, *NLO QCD matrix elements + parton showers in e^+e^- to hadrons*, *JHEP* **1301** (2013) 144, [1207.5031].

[24] R. Frederix and S. Frixione, *Merging meets matching in MC@NLO*, *JHEP* **1212** (2012) 061, [1209.6215].

[25] K. Hamilton, P. Nason, C. Oleari, and G. Zanderighi, *Merging H/W/Z + 0 and 1 jet at NLO with no merging scale: a path to parton shower + NNLO matching*, *JHEP* **1305** (2013) 082, [1212.4504].

[26] S. Plätzer, *Controlling inclusive cross sections in parton shower + matrix element merging*, 1211.5467.

[27] L. Lonnblad and S. Prestel, *Merging Multi-leg NLO Matrix Elements with Parton Showers*, *JHEP* **1303** (2013) 166, [1211.7278].

[28] K. Hamilton, P. Nason, and G. Zanderighi, *MINLO: Multi-Scale Improved NLO*, *JHEP* **1210** (2012) 155, [1206.3572].

[29] S. Catani and M. Seymour, *A General algorithm for calculating jet cross-sections in NLO QCD*, *Nucl.Phys.* **B485** (1997) 291–419, [hep-ph/9605323].