KaTie: for parton-level event generation with k_T-dependent initial states

A. van Hameren

Institute of Nuclear Physics Polish Academy of Sciences
PL-31-342 Kraków, Poland

Abstract

KaTie is a parton-level event generator for hadron scattering processes that can deal with partonic initial-state momenta with an explicit transverse momentum dependence causing them to be space-like. Provided with the necessary transverse momentum dependent parton density functions, it calculates the tree-level off-shell matrix elements and performs the phase space importance sampling to produce weighted events, for example in the Les Houches Event File format. It can deal with arbitrary processes within the Standard Model, for up to at least four final-state particles. Furthermore, it can produce events for single-parton scattering as well as for multi-parton scattering.

Contents

1 Introduction

Event generators are indispensable for the scientific activity in high-energy physics. Scattering experiments performed at for example the Large Hadron Collider involve physical processes evolving at a large range of energy scales. Typical energy scales reach several TeV in the so-called hard scattering process, in which heavy elementary particles like the Higgs boson are created, and evolve down to below the level of a GeV when the eventual scattering remnants reach the detector. Multi-purpose event generators [1–3] are designed to simulate scattering process over the whole range of scales. Their development greatly benefited from dedicated studies of the parton-level hard scattering process, which is computationally accessible with perturbative quantum chromodynamics (QCD). The goal of computing cross sections for arbitrary tree-level processes with many final-state particles (four or more) led to creation of several programs around the year 2000 [4–9]. The descriptive power of both the multi-purpose event generators and the parton-level event generators was greatly enhanced by combining their efforts using so-called merging procedures [10, 11]. The challenge of the parton-level generators was twofold: phase space points must be generated efficiently, and matrix elements must be evaluated quickly. The program presented in this paper is of this type, but with an extended range of applicability, as will be explained below. It must be mentioned that developments have gone beyond tree-level. Several parton-level event generators operate at next-to-leading (NLO) precision [12–18]. Obtaining the necessary one-loop amplitudes was a bottleneck, and several programs dedicated to solve this were developed [19–23]. The multi-purpose generators have been highly advanced, steering full calculations using internal libraries for hard scattering matrix elements or using external programs in various combinations, leading to precise predictions, *e.g.* [24–29].

All these programs function within a factorization prescription that separates the low-scale physics of the colliding protons from the high-scale physics of the hard process. More specifically, the cross section is a convolution of the parton distribution functions (PDFs) describing the protons, and the matrix element, describing the hard process. The PDFs only depend on the factorization scale and the fraction of the light-like proton momenta that enters the hard process, as prescribed by *collinear factorization* [30–32].

The initial-state partons in the hard process being parallel to the incoming hadrons is an approximation. For example, it implies that the final-state momenta can be separated into two groups that are back-to-back, which in a realistic collision is never the case. This approximation is traditionally cured by including higher fixed-order corrections, and/or by augmentation with a parton shower. It was argued in [33] that it would be desirable to remove this approximation already at lowest order in perturbation theory, and allow for non-vanishing transverse momentum components of the initial-state momenta in the hard process. The higher-order corrections may become much smaller, and it would open the possibility to include resummation corrections via transverse momentum dependent (TMD), or *un-integrated*, PDFs. An example of such an approach is high-energy factorization (k_T-factorization) [34–36].

On the side of the TMDs necessary for such an approach, there has been substantial activity [37]. Several evolution equations for TMDs have been developed, and the CCFM evolution [38–41] in particular is employed in the event generator CASCADE [42]. Several TMDs are provided by the library TMDLIB [43]. Besides the TMDs, such calculation requires matrix elements with off-shell initial-state partons. These cannot be obtained from collinear matrix elements by just changing the kinematics, because that would in general break gauge invariance. The problem of defining and calculating them, at least at tree-level, has been solved [44–50], and an automated implementation for arbitrary processes within the Standard Model with off-shell initial-state partons and with several final-state particles can be found in AVHLIB [51]. Furthermore, the program OGIME [52] can generate analytic expressions for multi-gluon amplitudes with several of them off-shell.

Besides implementations of functions to evaluate the TMDs and the matrix elements, the calculation of a cross section or other observables requires Monte Carlo tools to perform the necessary phase space integrals. The program LXJET [53] can be used for a selection of processes. For arbitrary processes, the tools are publicly available in AVHLIB [54, 55], which has already been used for calculations of various processes at the LHC, like for example $pp \to 4j$, $pp \to Z + j$ and $pp \to c\bar{c}\,c\bar{c}$ productions [56–59].

While AVHLIB provides all the numerical tools to generate phase space points, interpolate PDF grids, evaluate matrix elements etc., it does not provide a practical environment to perform phenomenological studies. KATIE fills this hiatus. In particular, it produces event files in the Les Houches Event File (LHEF) format [60], or a custom format for which it provides the tools to produce distributions of arbitrary observables. The definition of the process one would like to study happens via a single input file, containing all the information about which subprocesses are included, which PDFs are used, which cuts are applied, which values of model parameters are used, etc.. KATIE does not provide any PDFs, but uses LHAPDF [61] for collinear PDFs, and automatically interpolates any TMD PDF provided in the form of a (hyper-)rectangular grid (further described in Section 3.1.2). Alternatively, TMDLIB can be used to provide TMD PDFs. Finally, KATIE offers a convenient environment to perform calculations for multi-parton scattering (MPI), in which more than one hard process occurs in each event simultaneously.

The outline of the paper is as follows. Section 2 introduces the formalism along with some notation, and the usage of the program is explained in Section 3. Section 4 contains the summary, and the appendices containing some details close the paper.

2 Formalism

We consider collision processes of specific hadrons resulting in a specific final state and refer to those with the symbol Y. The process $Y = p\,p \to \mu^+\mu^-\,j\,j$ (proton-proton to $\mu^+\mu^-$ plus two jets) is a typical example. Several partonic processes contribute in this example, *e.g.* $Y \ni y = \bar{u}\,g \to \mu^+\mu^-\,\bar{u}\,g$. The separate particles in the partonic process are refered to by y_i. In the example given before, we have $y_2 = g$ and $y_4 = \mu^-$. The number of final-state particles in Y is $n = 4$ in the given example. In the following, we assume that all processes in Y share the same kinematical situation, and that the mass of final-state particle i is the same for every $y \in Y$.

A generally factorized formula for the differential cross section of a hadron collision process with a single parton-level scattering is given by

$$d\sigma_Y(p_1,p_2;k_3,\dots,k_{2+n}) = \sum_{y\in Y}\int d^4k_1\,\mathcal{P}_{y_1}(k_1)\int d^4k_2\,\mathcal{P}_{y_2}(k_2)\,d\hat{\sigma}_y(k_1,k_2;k_3,\dots,k_{2+n})\ . \quad (1)$$

It is differential in the final-state momenta $k_3,\dots,k_{2+n}$. The symbols p_1, p_2 refer to the momenta of the hadrons, while k_1, k_2 are the momenta of the initial-state partons. In collinear factorization, the distributions denoted with $\mathcal{P}$ are given by

$$\mathcal{P}_{y_i}(k_i) = \int\frac{dx_i}{x_i}\,f_{y_i}(x_i,\mu)\,\delta^4(k_i - x_ip_i)\ , \quad (2)$$

where f_{y_i} is the collinear PDF for a parton of type y_i coming from hadron i, and μ is the, possibly dynamical, factorization scale. For k_T-factorization the distributions typically look like

$$\mathcal{P}_{y_i}(k_i) = \int\frac{d^2\mathbf{k}_{iT}}{\pi}\int\frac{dx_i}{x_i}\,\mathcal{F}_{y_i}(x_i,|\mathbf{k}_{iT}|,\mu)\,\delta^4(k_i - x_ip_i - k_{iT})\ , \quad (3)$$

where now $\mathcal{F}_{y_i}$ is the TMD or unintegrated PDF, and k_{iT} is the embedding of $\mathbf{k}_{iT}$ in four-dimensional Minkowski space. The partonic differential cross section can be dissected futher as follows:

$$\begin{aligned} d\hat{\sigma}_y(k_1,k_2;k_3,\dots,k_{2+n}) = {} & d\Phi_Y(k_1,k_2;k_3,\dots,k_{2+n})\,\Theta_Y(k_3,\dots,k_{2+n}) \\ & \times \text{flux}(k_1,k_2)\times\mathcal{S}_y\,|\mathcal{M}_y(k_1,\dots,k_{2+n})|^2\ , \end{aligned} \quad (4)$$

where $d\Phi_Y$ is the $(3n-4)$-dimensional differential phase space element for a final state with n particles and masses dictated by Y:

$$d\Phi_Y(k_1,k_2;k_3,\dots,k_{2+n}) = (2\pi)^{4-3n}\left[\prod_{i=3}^{2+n} d^4k_i\,\delta_+(k_i^2-m_i^2)\right]\delta^4\big(k_1+k_2-k_3-\cdots-k_{2+n}\big)\ . \quad (5)$$

The (squared) matrix element $|\mathcal{M}_y|^2$ includes the summation over spins and colors of all external particles. It is turned into the average over color and spin degrees of freedom for the initial-state particles through the factor $\mathcal{S}_y$, which also includes the symmetry factor for the final state. The matrix elements used in KaTie are tree-level matrix elements, where k_1 and k_2 can have non-vanishing transverse components, and contain singularities which need to be avoided. This is established by phase space cuts Θ_Y, typically consisting of minimum transverse momenta, maximum absolute rapidities, minimum distance between momenta in the two-dimenional space of rapidity and azimutal angle (ΔR), etc. (examples of implemented cuts and scales can be found in Section 3.1.4). The flux factor, finally, includes the demand that the energy of the partonic process is positive:

$$\text{flux}(k_1,k_2) = \frac{\theta\big((k_1+k_2)^2\big)}{4\sqrt{(k_1\cdot k_2)^2 - k_1^2k_2^2}}\ . \quad (6)$$

The particular denominator is not prescribed by a specific factorization theorem, and is just an analytic continuation of the textbook formula for massive initial-state particles. It is discussed further in Section 3.1.6.

2.1 Event generation

KATIE creates event files consisting of a list of weighted phase space points such that

$$\frac{1}{N}\sum_{I=1}^{N} W_I\,\mathcal{B}\Big(k_3^{(I)},\ldots,k_{2+n}^{(I)}\Big) \approx \int d\sigma_Y(p_1,p_2;k_3,\ldots,k_{2+n})\,\mathcal{B}(k_3,\ldots,k_{2+n})\,. \tag{7}$$

The sum is over the events, and the approximation formally becomes an equality when the number of events N goes to infinity. W_I is the weight of phase space point I, and $k_j^{(I)}$ is the j-th momentum in event I. $\mathcal{B}$ is a test function, for example the bin $[z, z+\delta z]$ for a distribution of an observable $\mathcal{O}$:

$$\mathcal{B}(k_3,\ldots,k_{2+n}) = \theta\big(\mathcal{O}(k_3,\ldots,k_{2+n}) - z\big)\,\theta\big(z+\delta z - \mathcal{O}(k_3,\ldots,k_{2+n})\big)\,. \tag{8}$$

Of course, KATIE also provides k_1, k_2 in the event file, and "observables" depending on these momenta can be studied too.

2.1.1 Importance sampling

KATIE uses importance sampling to reduce the fluctuation of the weights. Because of the multitude of possible PDFs and matrix elements causing the fluctuations, the importance sampling is partly achieved in an adaptive manner, and KATIE employs an optimization phase before it starts generating the actual events. For each partonic subprocess, a probability density P_y is adaptively created with the aim that

$$\mathcal{W}_y(k_1,\ldots,k_{2+n}) = \frac{\mathcal{P}_{y_1}(k_1)\,\mathcal{P}_{y_2}(k_2)\,\mathrm{flux}(k_1,k_2)\,\mathcal{S}_y\,|\mathcal{M}_y(k_1,\ldots,k_{2+n})|^2}{P_y(k_1,\ldots,k_{2+n})} \tag{9}$$

is as constant as possible. The differential cross section can be written as

$$d\sigma_Y = \sum_{y\in Y} dF_y(k_1,\ldots,k_{2+n})\,\Theta_Y(k_3\ldots,k_{2+n})\,\mathcal{W}_y(k_1,\ldots,k_{2+n})\,, \tag{10}$$

where we say that phase space points are generated following the distributions F_y given by

$$dF_y(k_1,\ldots,k_{2+n}) = d^4k_1\,d^4k_2\,d\Phi_Y(k_1,k_2;k_3,\ldots,k_{2+n})\,P_y(k_1,\ldots,k_{2+n})\,. \tag{11}$$

Integration over k_1, k_2 is understood in Eq. (10). The creation of the distributions F_y leads to crude estimates of the partonic cross sections

$$\sigma_y = \int dF_y(k_1,\ldots,k_{2+n})\,\Theta_Y(k_3\ldots,k_{2+n})\,\mathcal{W}_y(k_1,\ldots,k_{2+n})\,. \tag{12}$$

During the event generation, first a subprocess is chosen with relative probability σ_y, and then a phase space point is generated following the distribution F_y. If the event does not satisfy the phase space cuts Θ_Y, it is rejected, and else it is accepted with weight

$$W = \mathcal{W}_y(k_1,\ldots,k_{2+n})\,\frac{\sum_{y'\in Y}\sigma_{y'}}{\sigma_y}\,. \tag{13}$$

So each event automatically has a subprocess assigned to it.

2.1.2 Un-weighting

Despite the importance sampling, the event weights may still fluctuate wildly, and supplementary techniques need to be applied to reduce this behavior. In the crude un-weighting method, the maximum weight W_{max} is determined, and event I is accepted with probability $W_{\mathrm{I}}/W_{\mathrm{max}}$. The main advantage is that all events get the same weight W_{max}. The disadvantage is that in practice W_{max} can only be determined within the total sample of events, and generating extra events may lead to a new increased W_{max}, effectively reducing the total number of events.

KaTie uses partial un-weighting to reduce the weight fluctuation while avoiding the problem mentioned above. Using the first N_0 events, a weight W_{thrs} is determined, and from then on, events with weight $W_{\mathrm{I}} \geq W_{\mathrm{thrs}}$ are accepted and keep their weight, while events with weight $W_{\mathrm{I}} < W_{\mathrm{thrs}}$ are accepted with probability W/W_{thrs}. If accepted, the latter get weight W_{thrs}. The threshold W_{thrs} is estimated with the aim to reach (the order of) a desired number of accepted events after reaching a desired statistical precision for the estimate of the total cross section. This way, a compromise can be chosen between accepting all events with wildly fluctuating weights, and accepting (possibly only very) few events with constant weight. More details can be found in the next section. It is important to mention that event files created with different values of W_{thrs} can be safely mixed without creating a bias, as long as they were created with *exactly the same set of distributions* F_y.

2.2 TMD interpolation

KaTie does not include any k_T-dependent PDFs. These can be provided by TMDlib [43]. Alternatively, they can be provided in the form of data files consisting of (hyper) rectangular grids representing the PDFs. The columns in the files must contain

$$\ln(x) \quad \ln\left(|\mathbf{k}_T|^2\right) \quad x\mathcal{F}(x, |\mathbf{k}_T|) \tag{14}$$

or

$$\ln(x) \quad \ln\left(|\mathbf{k}_T|^2\right) \quad \ln\left(\mu^2\right) \quad x\mathcal{F}(x, |\mathbf{k}_T|, \mu) \tag{15}$$

if the PDF also depends on the factorization scale. KaTie employs routines from avhlib, which use straightforward multi-linear interpolation, to interpolate these grids itself. As mentioned, the grids must be rectangular, but they do not need to be regular.

2.3 Matrix elements

The necessary matrix elements are provided by avhlib. They are evaluated via numerical Dyson-Schwinger recursion, which is implemented in a way very similar to Helac [4, 62]. Before the start of a Monte Carlo run, a list is prepared for each process, encoding the vertex operations and propagator operations that have to be executed numerically for each phase space point in order to arrive at the value of the amplitude. So the vertices and propagators etc. exist as compiled routines in the executable, whereas the amplitudes exist as such lists. Color is treated in terms of color-ordered decomposition, more specifically in the color-connection decomposition as in HELAC. This has the advantage that it naturally generalizes to arbitrary numbers of quark-antiquark pairs, but has the disadvantage that it does not automatically lead to the most efficient calculation of multi-gluon amplitudes. Because of this, the maximum number of color pairs in a process for KATIE to deal with comfortably is 6 for now.

2.4 MPI

It is possible to generate event files with KATIE in which each event is the result of two or more independent scatterings. These can be useful if one wants to study multiple parton interactions (MPI) [63]. In this case we consider the process Y to be separated into N_{MPI} processes $Y^{(h)}$. For example $Y = p\,p \to \mu^+\mu^-\,j\,j$ could be separated into $Y^{(1)} = p\,p \to \mu^+\mu^-$ and $Y^{(2)} = p\,p \to j\,j$. The hard processes are imagined to originate from one and the same pair of hadrons of course, but for our notation the above separation is most convenient. It has to be stated that KATIE does not provide a list of possible separations, and that a list of desired separations has to be provided by the user.

KATIE treats MPI by simply factorizing all distributions, the matrix element, and the phase space. The only function that is not necessarily factorized is Θ_Y representing the phase space cuts. Furthermore, there are extra restrictions requiring the sum of hadron momentum fractions coming from the same hadron to be smaller than one. The differential cross section becomes

$$d\sigma_Y = \frac{\mathcal{S}_{\mathrm{MPI}}}{\sigma_{\mathrm{eff}}} \prod_{h=1}^{N_{\mathrm{MPI}}} \left[\sum_{y \in Y^{(h)}} dF_y\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big)\, \mathcal{W}_y\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big) \right] \tag{16}$$
$$\times\, \Theta_Y(k_3 \dots, k_{2+n})\, \theta\left(1 - \sum_{h=1}^{N_{\mathrm{MPI}}} x_1^{(h)}\right) \theta\left(1 - \sum_{h=1}^{N_{\mathrm{MPI}}} x_2^{(h)}\right),$$

where σ_{eff} is some phenomenologically determined normalization with units of a surface to the power $N_{\mathrm{MPI}} - 1$, and $\mathcal{S}_{\mathrm{MPI}}$ is the symmetry factor associated with the MPI. For each set of, say l, identical $Y^{(h)}$, it contains a factor $1/l!$.

In the event generation, a subprocess $y^{(h)}$ is chosen for each $Y^{(h)}$ with relative probability

$$\sigma_{y^{(h)}} = \int dF_{y^{(h)}}\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big)\, \Theta_{Y^{(h)}}\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big)\, \mathcal{W}_{y^{(h)}}\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big)\,. \tag{17}$$

These are estimated during the optimization of the distributions $F_{y^{(h)}}$, which happens as if they concerned single-parton scattering processes, with cuts $\Theta_{Y^{(h)}}$ that are chosen such that they cover enough phase space. Each event in the event file gets a weight

$$W = \frac{\mathcal{S}_{\mathrm{MPI}}}{\sigma_{\mathrm{eff}}} \prod_{h=1}^{N_{\mathrm{MPI}}} \left[\mathcal{W}_{y^{(h)}}\big(k_1^{(h)}, \dots, k_{2+n_h}^{(h)}\big) \frac{\sum_{y' \in Y^{(h)}} \sigma_{y'}}{\sigma_{y^{(h)}}} \right]. \tag{18}$$

The PDFs used to create the event file are of the single-parton type. At the moment, it is not possible to use PDFs which depend in non-factorizable manner on the variables of more than one parton. Such dependencies can be included after the event file has been created, by re-weighting the events. The event file provides the value of $\mathcal{P}_{y_1(h)}\mathcal{P}_{y_2(h)}$ for each $Y^{(h)}$, and non-factorizable PDFs can be included by multiplying the event weight with

$$\frac{\mathcal{D}\left(k_1^{(1)},k_2^{(1)};\ldots;k_1^{(N_{\text{MPI}})},k_2^{(N_{\text{MPI}})}\right)}{\prod_{h=1}^{N_{\text{MPI}}}\mathcal{P}_{y_1(h)}\left(k_1^{(h)}\right)\mathcal{P}_{y_2(h)}\left(k_2^{(h)}\right)}\,, \tag{19}$$

where $\mathcal{D}$ is the desired (arbitrary) multi-parton PDF.

In order to studie for example both the contributions from single-parton scattering (SPS) and double-parton scattering (DPS) to a process, the user must generate separate event files for each. The (differential) cross sections from each can simply be added together. KATIE does not provide the means to generate “mixed” event files.

3 Usage

KATIE uses many Fortran 2003 features, but not all of them (for example not parametrized derived types). It requires a compiler which provides at least the Fortran 2003 features that gfortran-4.6 provides. Furthermore, it requires Python-2.x with x≥6 or Python-3.x, and Bash. The program can be obtained from

```
http://bitbucket.org/hameren/katie/downloads
```

It comes as a `.zip` file, that can be extracted with `unzip`. For the following, we refer to the directory where KATIE is installed, that is the directory created by the unzipping procedure, with `$KaTie`. Before KATIE can be used, the user must download AVHLIB, which can be obtained from

```
http://bitbucket.org/hameren/avhlib/downloads
```

and must also be unzipped. We refer to the resulting directory with `$AVHLIB`. Next, the user must edit the file `$KaTie/settings.py`, and set the path to the AVHLIB-directory (so the value of `$AVHLIB`), the path to the LHAPDF directory (the directory of the library file), and the Fortran compiler. LHAPDF is also required if no collinear PDFs are used, in order to provide the running coupling constant. It is advised to use at least version 6.x. If the user has TMDLIB available and wishes to use it, then they also need to set the path to the directory where its library file is, and the path to the directory where the library file of the GNU Scientific Library (GSL) is. Since TMDLIB version 2.0.2 it is possible to use more than one TMD set at a time, and the user must set the appropriate keyword to have this feature available.

After finishing the settings file, the user must execute

```
$ $KaTie/config.py lib
```

in order to create a library. This also configures AVHLIB. The easiest way to proceed is to choose an example in the directory `$KaTie/examples` that is closest to the user’s wishes, and use this as a starting point. Let us say it is `pp_to_4j`. The user must choose a non-existent directory, from now on referred to as `$project`, and execute

```
$ $KaTie/work.sh pp_to_4j $project
```

This will create `$project` with some files and a symbolic link to the run-script inside. In this example, there are two input files: one for single-parton scattering (SPS) and one for double-parton scattering (DPS).

If, for some reason, the library needs to be re-compiled at some point, this can be achieved with

```
$ $KaTie/run.sh clean
$ $KaTie/run.sh lib
```

This is the run-script that is linked in `$project`, so the re-compilation can also be invoked from there.

3.1 Input file

The input file consist of lines with the structure `keyword = value`. The '`=`' must be separated from `keyword` and `value` by at least one space. Words in a line in the input file must always be separated by at least one space, also when "words" consist of a single character, like '`=`'. As a first example of a keyword, we mention the statement

```
include file = someInputFile
```

with which the lines of another input file can be included.

3.1.1 Processes

By default, KaTie expects to be dealing with SPS, and the partonic processes the user wants to contribute can be simply listed as for example[1]

```
process = g g -> g g
process = g g -> u u~
process = g g -> d d~
process = g u -> g u
process = g d -> g d
```

etcetera for the production of two jets. The names of all possible particles are

```
ve   ve~   e-   e+   u u~ d d~
vmu  vmu~  mu-  mu+  c c~ s s~
vtau vtau~ tau- tau+ t t~ b b~
g H A Z W+ W-
```

[1]The old format with extra space-separated keywords in a specific order is still valid. Here, a more convenient new format with optional comma-separated keywords is presented.

They should all be obvious, except maybe the neutrinos which start with `v`, and the photon `A`. In case the user wants a partonic process like the second one above to refer to all light-quark contributions, KATIE can be told to multiply the partonic cross section the appropriate factor:

```
process = g g -> u u~ , factor = Nf
```

where `Nf` is the number of light flavors. Then the third process above can be omitted. This can also be done for more complicated situations, like for four jets:

```
process = g g -> u u~ d d~ , factor = (Nf*(Nf-1))/2
```

The number of flavors is set by

```
Nflavors = 5
```

If the user wants to stick to a parton-level calculation, and does not plan to process the eventfile further with a parton shower, they can sum over processes with identical matrix elements regarding initial-state quarks by setting the keyword

```
partlumi = combined
```

and using the symbols `q` and `r` to refer to quarks, like for example

```
process = g q  -> g q
process = q q~ -> r r~
```

This would be incorrect if electro-weak interactions are involved, and then the quarks should be denoted `u` for up-type quarks and `d` for down-type quarks. Then there is still a summation over combinations that have equal matrix elements. All combinations are explicitly given in Appendix A.

In case the process involves other interactions than pure QCD, the powers of the couplings can be set by, for example

```
pNonQCD = 2 0 0
```

The numbers refer to the power of the electro-weak coupling, the Higgs-gluon coupling, and the Higgs-photon coupling respectively. This is relevant again for example for $\mathrm{pp} \to \mu^+\mu^-\mathrm{jj}$, to which both $\mathcal{O}\left(\alpha_\mathrm{S}^2\alpha_\mathrm{EW}^2\right)$ and $\mathcal{O}\left(\alpha_\mathrm{EW}^4\right)$ Feynman graphs contribute, while one would like to exclude the latter. This keyword can also be set in the process line

```
process = g g -> mu+ mu- u u~ , pNonQCD = 2 0 0 , factor = Nf
```

The order of the keywords does not matter, they just must be separated by commas. Also, the commas and equality signs must be separated by spaces from other symbols.

For the application of MPI, KATIE needs to know the number of interactions, which is expressed as the number of groups the listed processes are divided into. This number is set by, for example

```
Ngroups = 2
```

for DPS. In this case the number of final-state particles must be set for each group, for example for four jets in DPS:

```
Nfinst = 2 2
```

A keyword `groups` must now be added in each process line, to indicate to which groups the process contributes, *e.g.*:

```
process = g g -> u u~ , factor = Nf , groups = 1 2
```

In the case of DPS for 4 jets, all $2 \to 2$ processes contribute to both groups, but for the production of $\mu^+\mu^- jj$ for example this would not be the case. Then, the $2 \to 2$ processes with the muons in the final state would only contribute to one group, and the processes with partons in the final state to the other group.

3.1.2 PDFs

The next parameter sets the pdf set from LHAPDF, for example

```
lhaSet = MSTW2008nlo68cl
```

It needs to be set also for off-shell initial-state partons, because KaTie takes α_S from there. The line

```
offshell = 0 0
```

determines that no partons are off-shell. The other possibilities are `0 1`, `1 0` to set one of the initial states off-shell, and `1 1` to set both of them off-shell. If only one argument is given, like `offshell = 1`, then only the positive-rapidity initial state is off-shell.

In order to use a TMD PDF set from TMDlib, the associated keyword from Table 1 in [43], for example `KS-2013-linear`, can be set with

```
TMDlibSet = KS-2013-linear
```

The initial state(s) marked as off-shell will then be treated with the PDFs from that set. From TMDlib version 2.0.0 to at least 2.0.2, it is not possible to use the keywords, and the user must put the integer key instead (`400001` for the example above).

Alternatively, the user can provide TMD PDFs as grids directly. They must be given in the form of files consisting of three or four columns, containing

$$\ln(x) \quad \ln\left(|\mathbf{k}_T|^2\right) \quad xf(x, |\mathbf{k}_T|)$$

or

$$\ln(x) \quad \ln\left(|\mathbf{k}_T|^2\right) \quad \ln\left(\mu^2\right) \quad xf(x, |\mathbf{k}_T|, \mu)$$

if the pdf also depends on the factorization scale. The scale dependence, *i.e.* the fact that the file has four columns, does not have to be indicated anywhere, and is recognized automatically. KaTie interpolates the grids itself. The TMD PDFs above are indicated with the symbol xf in order to highlight that KaTie alsway divides the result of the interpolation by the value of x. The directory where KaTie can find the grids is indicated in the input file by

```
tmdTableDir = /home/user/projects/tmdgrids/
```

where here of course an example path is given. The actual grid file must be indicated for each parton separately with

```
tmdpdf = g gluon.dat
tmdpdf = u uQuark.dat
tmdpdf = u~ uBar.dat
```

etc., where the file names are examples again. The path can be changed (or rather updated) between lines indicating the files, in case the files are distributed over several directories. In the case of k_T-factorization, a TMD for the gluon *must always* be provided, also if the user happens to only want to study quark-initiated processes. Setting the keyword `TMDlibSet` overrules the grids.

3.1.3 More than one TMD set

For the application of proton-ion collisions, one may want to have two different TMD sets. This can be achieved by specifying the grids as, for example

```
tmdTableDir = /home/user/projects/protongrids/
tmdpdf A = g gluon.dat
tmdpdf A = u uQuark.dat
tmdpdf A = u~ uBar.dat
tmdTableDir = /home/user/projects/iongrids/
tmdpdf B = g gluon.dat
tmdpdf B = u uQuark.dat
tmdpdf B = u~ uBar.dat
```

Here it was assumed that the grids are in different directories, but have the same name, which however is not necessary. The letter `A` refers to the positive-rapidity initial state, and the letter `B` to the negative-rapidity initial state. In the process list, they refer to partons as

```
B A -> 1 2 3 etc.
```

So the first one is `B` and the second one `A`.

From TMDLIB version 2.0.2 it is also possible to have two different TMD sets, *e.g.*:

```
TMDlibSet A = 400002
TMDlibSet B = 102200
```

The user must provide the integer keys of the TMD sets and cannot use the keywords.

3.1.4 Kinematics and cuts

The center-of-mass energy of the scattering is set in GeV with

```
Ecm = 7000
```

Instead of this, the user can also set beam energies `EbeamPosRap` and `EbeamNegRap`, or equivalently `EbeamA` and `EbeamB`, for the positive-rapidity and negative-rapidity beam respectively. In the list of processes, the first initial state has negative rapidity, while the second one has positive rapidity.

The phase space pre-sampler needs information about the typical value of the softest scale, like for example a minimum p_T, which is set to 20GeV by default, and can be changed to another value in GeV with for example

```
Esoft = 30
```

This number *is not a cut-off* and only influences the efficiency of the optimization, as explained in Appendix B. It must be larger than zero. The actual phase space cuts are set explicitly in the input file.

```
cut = {deltaR|2,4|} > 0.5
```

sets the minimum value of ΔR between final-state particle 2 and 4. At tree-level, to which KATIE currently is restricted, this corresponds to the value of R in a jet algorithm. The particle numbers above refer to the order as given in the process list.

```
cut = {pT|i|j,k,...} > 50
```

sets the minimum p_T in GeV for the i-th final-state in the p_T-ordered list of final-state number $j, k, \ldots$. For example, if the process is, for some unpractical reason, given as $pp \to j\,\mu^+\,j\,\mu^-\,j\,j$, then the jets are coming from final-state items `{1,3,5,6}`. One of them will have the second-highest jet p_T, and demanding that it is at least 50GeV is done by

```
cut = {pT|2|1,3,5,6} > 50
```

A minimum p_T for final-state 3 directly can be set with `{pT|3|}`, and a maximum p_T can be set with <. The

```
cut = {mass|1+3+4|} > 100
```

sets a minimum for the invariant mass of the sum of final-state momenta 1, 3, and 4. Similarly, other variables also can take arguments that consist of sums, *e.g.* the p_T of the sum of final-state momenta 2 and 3 is `{pT|2+3|}`. Other possible variables are `rapidity`, `pseudoRap`, `deltaPhi`, and `ET` (also called the transverse mass):

$$\sqrt{p_T^2 + p^2} = \sqrt{(E - p_z)(E + p_z)}\,. \tag{20}$$

Furthermore there are the polar angle `theta` and the angle between two finial-state momenta `angle`. The values of `rapidity` and `pseudoRap` can be negative and do not refer to the absolute value. On the other hand, `deltaPhi` is positive between 0 and π, while `theta` and `angle` are between 0 and 180. Also the momentum components `plus` and `minus` are available, referring to the inner product of the momentum with the light-like four-momenta $(1,0,0,1)$ and $(1,0,0,-1)$ respectively.

Orderings can also be relative to different variables, for example

```
cut = {pT|2|rapidity|1,3,5,6} > 50
```

demands that the p_T of the momentum from the list $\{1,3,5,6\}$ with the second largest rapidity is larger than 50GeV. For even more complicated cuts, the user can provide blocks of FORTRAN pseudo source code like

```
cut source = if ({rapidity|1|}.gt.{rapidity|2|}) then
cut source = if ({pT|2|1,2,3|}.lt.30d0) REJECT
cut source = endif
```

The statement `REJECT` has the consequence that the weight is set to zero, and thus should be invoked if the phase space point is outside the desired region.

3.1.5 Scales

By default, it is assumed that the factorization scale and the renormalization scale are identical in KaTie. It goes both into the PDFs and α_S, and is set with a line of FORTRAN source like for example

```
scale = ({pT|1|}+{pT|2|}+{pT|3|}+{pT|4|})/2
```

All variables mentioned in the previous section are allowed. In case the user wants a different renormalization scale with which α_S is to be evaluated, it can then be set with for example

```
renormalization scale = ({pT|1|}+{pT|2|}+{ET|3|}+{ET|4|})/2
```

If the scales going to the PDFs are to be different as well, then they can be set with for example

```
scaleA = {pT|1|}+{pT|2|}+91.2d0
scaleB = ({pT|1|}+{pT|2|}+{pT|3|}+{pT|4|})/4
```

As mentioned before, `A` refers to the positive-rapidity initial state, and `B` to the negative-rapidity initial state.

In the case of DPS, one would like to have separate scales for the different groups. This can be achieved with for example

```
scale = entry 1 ({pT|1|}+{pT|2|})/2
scale = entry 2 ({pT|3|}+{pT|4|})/2
```

In the case of DPS, cuts and scale also have to be given for each group separately for the phase space optimization. The phase spaces are optimized independently, as if they were SPS processes. To set these, the lines in the input file need, to look like

```
cut = group 1 {pT|2|1,2} > 50
scale = group 1 ({pT|1|}+{pT|2|})/2
cut = group 2 {pT|2|1,2} > 50
scale = group 2 ({pT|1|}+{pT|2|})/2
```

etc.. Notice that the particle numbering for `group 2` is also $1,2$ now, as if it was an SPS process.

3.1.6 Flux factor

As mentioned earlier, the denominator of the flux factor in Eq. (6) is not prescribed by a specific factorization theorem. It is argued in [64] that within the framework of Kimber, Martin, and Ryskin [65] the appropriate denominator is the generalization of the one in collinear factorization, so

$$\mathrm{flux}(k_1,k_2) = \frac{\theta\big((k_1+k_2)^2\big)}{8k_1^0k_2^0}\,. \tag{21}$$

This is the default in KATIE. The one of Eq. (6) can be set with the option

```
flux factor = textbook
```

3.1.7 Model parameters

Particle masses and widths are set in GeV with

```
mass = Z 91.1882 2.4952
mass = W 80.419 2.21
mass = H 125.0 0.00429
mass = t 173.5
```

The absence of a value for the width, like for the top quark above, implies it is set to zero. Other particles are massless by default, but can be given a mass and width too. The user can indicate which interactions are active with the lines

```
switch = withQCD Yes
switch = withQED No
switch = withWeak No
switch = withHiggs No
switch = withHG No
switch = withHA No
```

where the last two refer to the effective Higgs-gluon and Higgs-photon interactions (explicit expressions for the vertices can for example be found in [66]). These and `withHiggs` are switched off by default, while the others are switched on. The electro-weak coupling is fixed, and can be set with for example

```
coupling = alphaEW 0.00794
```

Alternatively, the value of Fermi's constant G_F can be set with for example

```
coupling = Gfermi 1.16639d-5
```

The electo-weak coupling is then set to $\alpha_{EW} = G_F\frac{\sqrt{2}}{\pi}m_W^2\big(1-\frac{m_W^2}{m_Z^2}\big)$. The value of the effective Higgs-gluon coupling is set with

```
coupling = Higgs-gluon -0.431d-3
```

This is the amplitude-level effective coupling and should contain everything except the overall factor of α_S, so the value of g_{rest} in $g_h = \alpha_S\, g_{\text{rest}}$. Possible higher powers of α_S can only be included as fixed numbers in g_{rest} and cannot run. For `Higgs-photon` the user provides the whole, fixed, value of the amplitude-level effective coupling.

3.1.8 DIS

Deep inelastic scattering (DIS) can be studied with KaTie by setting processes as follows

```
process = DIS g  -> g g
process = DIS u  -> g u
process = DIS u~ -> g u~
```

for the production of two jets, for example. The, for example, first process line implies the process `e- g -> g g e-`, so the initial-state electron has negative rapidity. The user should not forget to set the proper number of electro-weak interactions, so at least

```
pNonQCD = 2 0 0
```

As mentionend earlier, the user can set beam energies `EbeamPosRap` and `EbeamNegRap`, but also the equivalents `EbeamHadron` and `EbeamElectron`. In order to set cuts involing the final-state electron, it is indicated by the word `electron`, for example

```
cut = {energy|electron|} > 11
cut = {theta|electron|} > 30
cut = {plus|1+2+electron|} > 35
```

In this example, the kinematical variable `plus` is introduced, refering to the value of $E - p_z$ of the momentum represented by the argument, that is the Lorentz-invariant product of this momentum with $n_+^\mu = (1, 0, 0, 1)$. Of course, also `minus` is available. Other available variables useful for DIS are `{Qsquare}`$=Q^2$, `{inelast}`$=y$, and `{xBjorken}`$=x_B$, satisfying $x_B y = Q^2/E_{\text{cm}}^2$. These three variables do not take arguments. Furthermore, there are `deltaRbreit` and `pTbreit`, referring to ΔR and p_T in the Breit-frame of the difference between the momenta of the initial-state electron and the final-state electron.

For e^+p scattering, the keyword is `DIS+`, while `DIS-` is equivalent to `DIS`. In the former case, the keyword `positron` can be used to refer to the final-state positron.

In case the requested process does not involve any other electro-weak interactions, it may be convenient to set the values of Q^2 and x_B. This is possible via, for example,

```
xBjorken bin = 0.001 0.01
Qsquare bin = 10 350
```

The bins can be chosen arbitrarily small, and setting the upper limit equal (or smaller) than the lower limit results in the calculation of the differential cross section. By setting

```
DISF2 = yes
```

the factor

$$\frac{1}{2\pi\alpha_{\mathrm{EW}}^2}\,\frac{x_B\,Q^4}{1+(1-y)^2}\,\frac{[\mathrm{GeV}^{-2}]}{[\mathrm{nb}]} \tag{22}$$

is included to produce F_2 instead of the cross section.

3.1.9 Extra weight factors

If the user wishes multiply the event weight with extra factors, then this can be accommodated by providing strings of FORTRAN source via for example

```
weight factor = {pT|1|}**4/(1d0+{pT|1|}**2)**2
```

The user may want to access variables associated to initial-state momenta for this purpose, which can be done using the argument `A` for the positive-rapidity initial state, and `B` for the negative-rapidity initial state, for example in `{pT|A|}`.

3.1.10 Optimization parameters

If there are many final-state particles, the sum over helicities becomes unnecessarily time consuming, and the user should choose to sample over helicities instead:

```
helicity = sampling
```

Other possible values are `sum` and `polarized`. The latter chooses the continuous sampling method of [67]. The number of events to be spent on the optimization of the pre-sampler also needs to be set in the input file.

```
Noptim = 100,000
```

sets the number of non-vanishing-weight events to a hundred thousand. This will be discussed in more detail in the following.

3.2 Code generation

The input file contains the information to create dedicated source files that are immediately, and quickly, compiled with the command

```
$ $project/run.sh prepare $project/inputFileName $project/trial01
```

Here, `inputFileName` is the name of the input file, and `trial01` is the name of a new directory that is being created. It contains a directory for each process listed in the input files, each of which contains a FORTRAN source file and an executable. The directory `trial01` itself also contains a FORTRAN source file and executable. These are the eventual Monte Carlo programs. The source files can be edited and then re-compiled by executing the script `recompile.sh`. For example, even more complicated phase space cuts can be written to the file `extra_cuts.h90`. This file is included in all source files, also to those in the process directories, after the line starting with

`!]cuts`. The user must familiarize themselves with the syntax of the source files, and may need to contact the author, but in principle it is possible without complications.

3.3 Optimization stage

Event generation happens in two stages. During the first stage, the pre-sampler is optimized for each process given in the input file separately. After code generation, the optimization of all processes is started by executing

```
$ $project/trial01/optimize.sh
```

If there are very many processes, one might want the optimization to happen in batches, and by executing for example

```
$ $project/trial01/optimize.sh Nparallel=4
```

only 4 optimizations will run simultaneously until all have been performed. One can also select processes to be optimized, *e.g.*

```
$ $project/trial01/optimize.sh Nparallel=4 proc=1,7,13,24,25
```

The progress can be monitored by viewing the output files in each process directory, for example with

```
$ tail -f $project/trial01/proc*/output
```

The final precision should not be more than a few percent. For example

```
MESSAGE from Kaleu stats: Ntot = 38,887
MESSAGE from Kaleu stats: +  25,600 (.13072883+/-.00132728)E+06 1.015%
```

means that 38887 events were generated, of which 25600 passed the cuts, leading to an estimated cross section of 0.1307×10^6nb with an estimated statistical error of 1.015%. The $+$ in front of `25,600` means that it concerns positive-weight events. In case the optimization of a process does not seem to converge, the user can try to increase the number of events for that process and/or change the random number seed. Say the user wants to rerun process number 3 with seed=273846 and with Noptim=200000. Then the user needs to execute

```
$ prefix=$project/trial01/proc03
$ $prefix/main.out seed=273846 Noptim=200000 > $prefix/output &
```

Alternatively, the user can edit the appropriate lines in `$project/trial/optimize.sh`. For any seed, the result of the optimization will be an unbiased, but not necessarily efficient, phase space pre-sampler. The seed can be different for every process. The only rule is that once you start to generate event files, the pre-sampler must be fixed, and you MUST NOT rerun the optimization.

3.4 Event generation

The second stage is the actual event generation, and happens after the optimization is finished. The user may just execute

```
$ cd $project/trial01
$ nohup ./main.out seed=732415 > output1 &
$ nohup ./main.out seed=232984 > output2 &
```

etc. for several different random number seeds. So-called "raw" event files will be produced in `$project/trial01` with the names `raw732415.dat`, `raw232984.dat` etc. They contain weighted events that form a partially un-weighted collection of events from all that are generated by the pre-sampler. The number of events and the rate of fluctuation of their weights can be steered to some degree, as explained in Section 2. By default, of the order of 10^5 events are accepted when the cross section is estimated to a statistical precision of $0.001 = 0.1\%$. These numbers can be changed by providing optional arguments. For example

```
$ nohup ./main.out Nevents=1e6 precision=0.01 seed=732415 > output1 &
```

will try to accept of the order of 10^6 events until a statistical precision of $0.01 = 1\%$ is reached. In this case, of course, the events will be of "lower quality" in the sense that their weights will fluctuate more.

The standard output of a Monte Carlo run, sent to output files in the examples above, will, after some initialization, consist of lines looking as follows:

```
1,579,644       19,600  (.28868124+/-.01110617)E+03   3.847%
```

These data are the number of generated events so far, the number of events that passed the cuts, the estimate of the cross section in nb with error estimate, and the estimated relative error.

Given a number of raw files, an event file can be created in the LHEF format by executing

```
$ ./create_eventfile.out lhef raw*
```

This will use all available raw files. The user may also select some by listing them separately in the above command. This command may be executed before the event generations have finished. The executable `create_eventfile.out` is the result of the compilation of the source file `create_eventfile.f90` located in the same diretory. The user may edit this source file and then recompile it by executing

```
$ bash create_eventfile.sh
```

Having access to the source file gives the user the opportunity to manipulate the events, for example by re-weighting them. The available variables and functions are listed at the beginning of the source file. Also, the user has the opportunity to create histograms using the routines listed in the next section. If the user wishes to use functions from other external libraries, then the file `create_eventfile.sh` must be edited accordingly. If this script is provided with the same arguments as discussed here for `create_eventfile.out`, it will actually execute the latter after compilation.

Alternatively to the LHEF format, an event file in a custom format can be created by executing

```
$ ./create_eventfile.out custom raw*
```

The format is explained in Appendix C. The cross section will be in nano-barn. This can be changed to pico-barn using the comma-separated keywords `custom,pb` (no spaces). The LHEF can be obtained in nano-barn with the keywords `lhef,nb`. Other available keywords are `dir=<dirname>` setting the directory to which the event file is written (the default is `./`), and `name=<filename>` setting the file name (the default is `eventfile.dat`). No spaces are allowed. So a complete example could be

```
$ ./create_eventfile.out lhef,dir=/tmp,name=events1.dat raw*
```

In the next section, it is explained how to create histogram files together with the event file. If the user is only interested in these, and does not need the event file, then they can use the keyword `noEventFile` in the option string, like

```
$ ./create_eventfile.out custom,noEventFile raw*
```

Finally, event files (still) can also be created using the run script:

```
$ $project/run.sh lhef $project/trial01/raw*
```

or

```
$ $project/run.sh merge $project/trial01/raw*
```

but then there is no opportunity to manipulate the events.

3.5 Creation of histograms

The program `create_eventfile` has a declaration block, an initialization block to set the bins for the histograms etc., a block that runs through the event file to collect the data, and a block to write the histograms to files with user-chosen file names. An array of histograms can, for example, be declared as

```
type(histo_1d_type) ::  h_pT(1:4)
```

and the entries can be initialized with

```
do ii=1,4
call h_pT(ii)%init( left=0d0 ,right=200d0 ,Nbins=100 )
enddo
```

It is also possible to set bins explicitly, *e.g.*

```
call h_pT(1)%init([0d0,50d0 ,50d0,100d0 ,100d0,150d0 ,150d0,200d0])
```

will set four bins of size 50GeV (the entries in the array may also be single precision). An example of the available variables is the array `pFinst(0:3,:)`, containing the final-state four-momenta of the event (pFinst(0,i) is the energy of final-state momentum i), and an example of the available functions is pTrans, returning the size of the transverse momentum of a four-momentum. Then,

```
do ii=1,4
pT(ii) = pTrans(pFinst(0:3,ii))
enddo
```

will fill the array pT with the value of the size of the transverse momenta of the first four final-state momenta These can be ordered with

```
call sort_big2small( pTordered ,pT ,Njet )
```

which alters pT, but also returns the integer array pTordered containing the associated permutation: pT(i)=pTrans(pFinst(0:3),pTordered(i)). So pT(1) will be the largest, pT(2) the next to hardest etc. The histograms are filled with

```
do ii=1,4
call h_pT(ii)%collect( pT(ii) ,eventWeight )
enddo
```

where eventWeight is the event weight. Finally, the histograms are written to files with

```
do ii=1,4
call h_pT(ii)%write('pT'//numeral(ii)//'.hst')
enddo
```

The character array numeral(0:9) contains the numbers 0 to 9 as single characters, so the filenames will be pT1.hst, pT2.hst, etc. The files are written to the directory where the user executed create_eventfile.out. They consist of four columns, containing the left bin-border, right bin-border, the value, and the statistical error estimate. Histograms can be plotted with, for example, gnuplot via

```
plot 'pT1.hst' using ($1+$2)/2:3 with boxes
```

3.6 ITMD

From the technical point of view, Improved Transverse Momentum Dependent factorization [68] is special because it requires the matrix elements to be separated into color structures, each of which is associated with a different TMD. This can conveniently be organized by using TMD-valued color matrices [69]. Amplitudes in KATIE are follow the *color connection* decomposition [70, 71], which differs from the color flow decomposition [72] in that the $1/N_c$ terms are in the gluon-quark vertex instead of the gluon propagator. This has the consequence that the TMD-valued color matrices become particularly simple, and each element of the matrices consist of a single so-called $\mathcal{F}$-TMD and a single power of N_c. There are 10 possible $\mathcal{F}$-TMDs and they are defined in equation (25) to (34) in [69].

Calculations can be performed within ITMD factorization with KATIE by including the following lines in the input file. ITMD needs one off-shell initial-state gluon, and in KATIE this must be in the positive-rapidity direction

```
offshell = 0 1
```

ITMD factorization is activated with

```
itmdf = yes
```

Then, the necessary grids for the TMDs are loaded as

```
tmdpdf = qg1 gridFileForFqg1.dat
tmdpdf = qg2 gridFileForFqg2.dat
tmdpdf = gg1 gridFileForFgg1.dat
tmdpdf = gg2 gridFileForFgg2.dat
```

etcetera. The file names are just examples. In general, not all 10 TMDs are required. The beginning of the output files produced during the optimization phase contain the TMD-valued color matrix. The leading color approximation can be invoked with

```
leadingColor = yes
```

It puts the elements of the color matrix with non-leading powers of N_c to zero.

3.7 Amplitudes

The library makes the

```
module katie_amplitudes
```

available, which provides routines to evaluate squared amplitudes, summed over color, as function of momenta and helicities. The command

```
$ $KaTie/run.sh help compile
```

prints the lines the user must include when compiling a program that uses this, or any other, module from the library. Alternatively, the command

```
$ $KaTie/run.sh katamp
```

creates a single source file `katamp.f90` containing the module and all dependencies, which the user can compile themselves. I just needs the location of a data file `katie_ITMDmatrix.tbl` containing tables, but the path should automatically have been set correctly.

Routines are imagined to be mainly used within Monte Carlo applications, so initialization is separated from evaluation. A list of processes can be specified at the initialization stage, the amplitudes of which will from then on be available. First, the user must specify a “class” of processes via their kinematics with

```
call katamp_add_kinematics( kinID ,Nxtrn ,inStates )
```

The integer input `Nxtrn` is the number of external particles, and the input `inStates` is an integer array of size `Nxtrn` stating for each particle whether it is an initial state, by having a non-zero entry. If such a non-zero entry is equal to `2`, then the initial state will be off-shell. The output `kinID` is an integer identifier for the kinematics defined by that particular call.

In case initial states are chosen to be off-shell, the first non-zero entry corresponds to the positive-rapidity direction, and the second non-zero entry to the negative-rapidity direction. The default directions can be changed by first calling

```
call katamp_set_directions( pA ,pB )
```

Where the real input `pA(0:3)` of `kind(1d0)` will be the direction of the first initial state and `pB(0:3)` the direction of the second initial state. The default corresponds to `pA=[-1d0,0d0, 0d0,-1d0] pB=[-1d0,0d0,0d0,1d0]`.

For a given kinematics, processes can be defined with

```
call katamp_add_process( kinID ,prcID ,process, pNonQCD )
```

Now `kinID` is input and `prcID` is an output integer identifier for the process defined by that particular call. The integer input `process` is an array containing the particle flavors. These are defined by the following obviously-named integer parameters provided by the module:

```
eleNeu eleon  uQuark dQuark
muNeu  muon   cQuark sQuark
tauNeu tauon  tQuark bQuark
Wboson photon Zboson gluon  Higgs
```

Anti-particles and the W^- boson are obtained by putting a minus sign. So, an example process array is `[gluon,uQuark,-uQuark,gluon]`. The “charges” must add up to zero. The integer input array `pNonQCD` has size 3 and is equivalent to the eponymous keyword for KATIE input files (Section 3.1.1). Interactions can be switched on and off *for all processes* by calling the following routines with arguments equal to `0` or `1`:

```
set_withQCD set_withQED set_withWeak
set_withHiggs set_withHG set_withHA
```

where the last two refer to the effective Higgs-gluon and Higgs-photon interactions. Masses and widths can be set as, for example

```
call set_mass_and_width( Zboson ,91.2d0 ,2.5d0 )
```

The QCD and QED couplings α_S and α_{EW} are equal to one. This means that the coupling strenghts g_{EW} associated with the photon-lepton interaction vertices is equal to $\sqrt{4\pi}$. The weak couplings are proportional to this number, and further defined via $\cos\theta_{\text{Weinberg}}$, which by default is set to M_W/M_Z. It can be set to $\sqrt{M_W^2 - iM_W\Gamma_W}/\sqrt{M_Z^2 - iM_Z\Gamma_Z}$ with

```
call set_complex_mass_scheme
```

If the user wants to perform calculations within ITMD factorization, then they must call

```
call katamp_prepare_itmd( kinID ,prcID ,option )
```

after defining the process. All arguments are input. The only value for the integer `option` that is not ignored is `2`, which invokes the leading color approximation.

The previous routines should be called before the "Monte Carlo loop". Within the loop, the user can provide momenta for a kinematics class with

```
call katamp_put_momenta( kinID ,momenta ,momSqr )
```

where `kinID` is input, and `momenta` is a real array of `kind(1d0)` with shape `(0:3,*)` containing the momentum values, such that `momenta(0,2)` is the energy for particle number 2, `momenta(1:3,2)` are the $x-y-z$-components of its momentum, etc. The real array `momSqr` must contain the values of the squared momenta (the desired values, eg. squared masses, not the ones computed from `momenta`). The momenta must sum up to zero. They will be available for all processes defined using the particular value of `kinID`. Then, amplitudes are evaluated for given helicity configurations with

```
call katamp_evaluate( kinID ,prcID ,iEval ,helicities )
```

`kinID` and `prcID` are input, while `iEval` is an output integer identifier for the evaluation of amplitudes in that particular call. The integer array `helicities` contains the helicities of the particles, which have one of the values `-1,0,1`. The idea is to call this routine several times with different values of `helicities`, and storing the corresponding values of `iEval`. The value of the squared amplitude summed over color is then obtained with

```
call katamp_sqr( kinID ,prcID ,iEval ,rslt )
```

Now, `kinID` and `prcID` and `iEval` are input, and `rslt` is real output of `kind(1d0)`, corresponding to the helicity configurations associated with the value of `iEval`. A minimal but complete example program is

```
program testIt
use katie_amplitudes
implicit none
integer ::  kinID,prcID,iEval
real(kind(1d0)) ::  momenta(0:3,4),momSqr(4),rslt
call katamp_add_kinematics( kinID ,4 ,[1,0,0,1] )
call katamp_add_process( kinID ,prcID ,[gluon,uQuark,-uQuark,gluon] &
                                      ,[0,0,0] )
momenta = reshape( [-3d0, 0d0, 0d0,-3d0&
                   , 3d0, 0d0, 3d0, 0d0&
                   , 3d0, 0d0,-3d0, 0d0&
                   ,-3d0, 0d0, 0d0, 3d0] , [4,4] )
momSqr(1:4) = 0
call katamp_put_momenta( kinID ,momenta ,momSqr )
call katamp_evaluate( kinID,prcID ,iEval ,[1,-1,1,-1] )
call katamp_sqr( kinID,prcID ,iEval ,rslt )
write(*,*) rslt
end program
```

For ITMD factorization, the user must instead of `katamp_sqr`

```
call katamp_itmd_sqr( kinID ,prcID ,iEval ,tmds ,rslt )
```

where the input `tmds` is a real array of `kind(1d0)` containing the values of the evaluated TMDs. The order of the entries should be as follows: $\mathtt{tmds}(1:3) = \mathcal{F}_{qg}^{(1:3)}$ and $\mathtt{tmds}(4:10) = \mathcal{F}_{gg}^{(1:7)}$, where the $\mathcal{F}$-TMDs are defined in equation (25) to (34) in [69]. In this case, `rslt` includes the matrixelements and the TMDs inside `tmds`. It does not include any other PDFs. The TMD-valued color matrix can be printed with

```
call katamp_print_itmd( kinID ,prcID ,writeUnit )
```

where the integer input `writeUnit` sets the unit to which the matrix is written.

4 Summary

KATIE, a program for parton-level generation of events with initial-states that can have non-vanishing transverse momentum components, was presented. It provides all necessary ingredients, including off-shell matrix elements and an efficient importance sampler, except the transverse momentum dependent parton density functions. The latter can be provided in the form of hyper-rectangular grids which KATIE will automatically interpolate, or by TMDLIB. Events are produced in the Les Houches Event File format, or in a custom format with which distributions of arbitrary observables can be produced with tools also provided by KATIE. Finally, a convenient environment is available to perform calculations involving multi-parton interactions.

Acknowledgments

The author would like to thank Marcin Bury, Hannes Jung, Mirko Serino, and Krzysztof Bożek for their help in beta testing the program. Also, the author would like to thank Maxim Nefedov and Vladimir Saleev for valuable feedback. This work was supported by grant of National Science Center, Poland, No. 2015/17/B/ST2/01838.

5 References

[1] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, *An Introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159–177, [1410.3012]. http://home.thep.lu.se/\~{}torbjorn/Pythia.html.

[2] J. Bellm *et al.*, *Herwig 7.0/Herwig++ 3.0 release note*, *Eur. Phys. J.* **C76** (2016), no. 4 196, [1512.01178]. https://herwig.hepforge.org.

[3] T. Gleisberg, S. Hoeche, F. Krauss, M. Schönherr, S. Schumann, *et al.*, *Event generation with SHERPA 1.1*, *JHEP* **0902** (2009) 007, [0811.4622]. https://sherpa.hepforge.org/trac/wiki.

[4] A. Kanaki and C. G. Papadopoulos, *HELAC: A Package to compute electroweak helicity amplitudes*, *Comput.Phys.Commun.* **132** (2000) 306–315, [hep-ph/0002082].

[5] F. Krauss, R. Kuhn, and G. Soff, *AMEGIC++ 1.0: A Matrix element generator in C++*, *JHEP* **0202** (2002) 044, [hep-ph/0109036].

[6] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau, and A. D. Polosa, *ALPGEN, a generator for hard multiparton processes in hadronic collisions*, *JHEP* **0307** (2003) 001, [hep-ph/0206293].

[7] M. Moretti, T. Ohl, and J. Reuter, *O'Mega: An Optimizing matrix element generator*, hep-ph/0102195.

[8] W. Kilian, T. Ohl, and J. Reuter, *WHIZARD: Simulating Multi-Particle Processes at LHC and ILC*, 0708.4233.

[9] F. Maltoni and T. Stelzer, *MadEvent: Automatic event generation with MadGraph*, *JHEP* **0302** (2003) 027, [hep-ph/0208156].

[10] A. Buckley *et al.*, *General-purpose event generators for LHC physics*, *Phys. Rept.* **504** (2011) 145–233, [1101.2599].

[11] T. Sjöstrand, *Status and developments of event generators*, *PoS* **LHCP2016** (2016) 007, [1608.06425].

[12] S. Frixione and B. R. Webber, *The MC and NLO 3.4 Event Generator*, 0812.0770.

[13] K. Arnold *et al.*, *VBFNLO: A Parton level Monte Carlo for processes with electroweak bosons*, *Comput. Phys. Commun.* **180** (2009) 1661–1670, [0811.4559].

[14] J. M. Campbell and R. Ellis, *MCFM for the Tevatron and the LHC*, *Nucl.Phys.Proc.Suppl.* **205-206** (2010) 10–15, [1007.3492].

[15] S. Alioli, P. Nason, C. Oleari, and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, *JHEP* **06** (2010) 043, [1002.2581].

[16] G. Bevilacqua, M. Czakon, M. Garzelli, A. van Hameren, A. Kardos, *et al.*, *HELAC-NLO*, *Comput.Phys.Commun.* **184** (2013) 986–997, [1110.1499].

[17] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079, [1405.0301].

[18] C. Weiss, B. Chokoufe Nejad, W. Kilian, and J. Reuter, *Automated NLO QCD Corrections with WHIZARD*, *PoS* **EPS-HEP2015** (2015) 466, [1510.02666].

[19] C. Berger, Z. Bern, L. Dixon, F. Febres Cordero, D. Forde, *et al.*, *An Automated Implementation of On-Shell Methods for One-Loop Amplitudes*, *Phys.Rev.* **D78** (2008) 036003, [0803.4180].

[20] F. Cascioli, P. Maierhofer, and S. Pozzorini, *Scattering Amplitudes with Open Loops*, *Phys.Rev.Lett.* **108** (2012) 111601, [1111.5206].

[21] 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].

[22] 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].

[23] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf, and S. Uccirati, *RECOLA: REcursive Computation of One-Loop Amplitudes*, *Comput. Phys. Commun.* **214** (2017) 140–173, [1605.01090].

[24] S. Badger, A. Guffanti, and V. Yundin, *Next-to-leading order QCD corrections to di-photon production in association with up to three jets at the Large Hadron Collider*, *JHEP* **03** (2014) 122, [1312.5927].

[25] H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, and T. Peraro, *Next-to-Leading-Order QCD Corrections to Higgs Boson Production in Association with a Top Quark Pair and a Jet*, *Phys. Rev. Lett.* **111** (2013), no. 17 171801, [1307.8437].

[26] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Hoeche, H. Ita, D. A. Kosower, N. A. Lo Presti, and D. Maitre, *Next-to-leading order $\gamma\gamma + 2$-jet production at the LHC*, *Phys. Rev.* **D90** (2014), no. 5 054004, [1402.4127].

[27] S. Kallweit, J. M. Lindert, P. Maierhofer, S. Pozzorini, and M. Schönherr, *NLO QCD+EW predictions for V + jets including off-shell vector-boson decays and multijet merging*, *JHEP* **04** (2016) 021, [1511.08692].

[28] M. Rauch and S. Plätzer, *Parton Shower Matching Systematics in Vector-Boson-Fusion WW Production*, *Eur. Phys. J.* **C77** (2017), no. 5 293, [1605.07851].

[29] J. Bellm, S. Gieseke, N. Greiner, G. Heinrich, S. Plätzer, C. Reuschle, and J. F. von Soden-Fraunhofen, *Anomalous coupling, top-mass and parton-shower effects in* W^+W^- *production*, *JHEP* **05** (2016) 106, [1602.05141].

[30] S. B. Libby and G. F. Sterman, *Jet and Lepton Pair Production in High-Energy Lepton-Hadron and Hadron-Hadron Scattering*, *Phys. Rev.* **D18** (1978) 3252.

[31] R. K. Ellis, H. Georgi, M. Machacek, H. D. Politzer, and G. G. Ross, *Perturbation Theory and the Parton Model in QCD*, *Nucl.Phys.* **B152** (1979) 285.

[32] J. C. Collins, D. E. Soper, and G. F. Sterman, *Factorization of Hard Processes in QCD*, *Adv. Ser. Direct. High Energy Phys.* **5** (1989) 1–91, [hep-ph/0409313].

[33] J. Collins and H. Jung, *Need for fully unintegrated parton densities*, in *HERA and the LHC: A Workshop on the implications of HERA for LHC physics. Proceedings, Part B*, 2005. hep-ph/0508280.

[34] S. Catani, M. Ciafaloni, and F. Hautmann, *High-energy factorization and small x heavy flavor production*, *Nucl.Phys.* **B366** (1991) 135–188.

[35] J. C. Collins and R. K. Ellis, *Heavy quark production in very high-energy hadron collisions*, *Nucl.Phys.* **B360** (1991) 3–30.

[36] E. M. Levin, M. G. Ryskin, Yu. M. Shabelski, and A. G. Shuvaev, *Heavy quark production in semihard nucleon interactions*, *Sov. J. Nucl. Phys.* **53** (1991) 657. [Yad. Fiz.53,1059(1991)].

[37] R. Angeles-Martinez *et al.*, *Transverse Momentum Dependent (TMD) parton distribution functions: status and prospects*, *Acta Phys. Polon.* **B46** (2015), no. 12 2501–2534, [1507.05267].

[38] M. Ciafaloni, *Coherence Effects in Initial Jets at Small q**2 / s*, *Nucl.Phys.* **B296** (1988) 49.

[39] S. Catani, F. Fiorani, and G. Marchesini, *QCD Coherence in Initial State Radiation*, *Phys.Lett.* **B234** (1990) 339.

[40] S. Catani, F. Fiorani, and G. Marchesini, *Small x Behavior of Initial State Radiation in Perturbative QCD*, *Nucl.Phys.* **B336** (1990) 18.

[41] G. Marchesini, *QCD coherence in the structure function and associated distributions at small x*, *Nucl. Phys.* **B445** (1995) 49–80, [hep-ph/9412327].

[42] H. Jung, S. Baranov, M. Deak, A. Grebenyuk, F. Hautmann, *et al.*, *The CCFM Monte Carlo generator CASCADE version 2.2.03*, *Eur.Phys.J.* **C70** (2010) 1237–1249, [1008.0152].

[43] F. Hautmann, H. Jung, M. Krämer, P. J. Mulders, E. R. Nocera, T. C. Rogers, and A. Signori, *TMDlib and TMDplotter: library and plotting tools for transverse-momentum-dependent parton distributions*, *Eur. Phys. J.* **C74** (2014) 3220, [1408.3015]. http://tmdlib.hepforge.org.

[44] L. Lipatov, *Gauge invariant effective action for high-energy processes in QCD*, *Nucl.Phys.* **B452** (1995) 369–400, [hep-ph/9502308].

[45] L. Lipatov and M. Vyazovsky, *QuasimultiRegge processes with a quark exchange in the t channel*, *Nucl.Phys.* **B597** (2001) 399–409, [hep-ph/0009340].

[46] E. Antonov, L. Lipatov, E. Kuraev, and I. Cherednikov, *Feynman rules for effective Regge action*, *Nucl.Phys.* **B721** (2005) 111–135, [hep-ph/0411185].

[47] A. van Hameren, P. Kotko, and K. Kutak, *Helicity amplitudes for high-energy scattering*, *JHEP* **1301** (2013) 078, [1211.0961].

[48] A. van Hameren, K. Kutak, and T. Salwa, *Scattering amplitudes with off-shell quarks*, *Phys.Lett.* **B727** (2013) 226–233, [1308.2861].

[49] A. van Hameren, *Scattering amplitudes for high-energy factorization*, 1307.1979.

[50] P. Kotko, *Wilson lines and gauge invariant off-shell amplitudes*, *JHEP* **1407** (2014) 128, [1403.4824].

[51] M. Bury and A. van Hameren, *Numerical evaluation of multi-gluon amplitudes for High Energy Factorization*, *Comput. Phys. Commun.* **196** (2015) 592–598, [1503.08612].

[52] P. Kotko, *Ogime*. http://nz42.ifj.edu.pl/~pkotko/OGIME.html.

[53] P. Kotko, *LxJet*. http://nz42.ifj.edu.pl/~pkotko/LxJet.html.

[54] A. van Hameren, *PARNI for importance sampling and density estimation*, *Acta Phys.Polon.* **B40** (2009) 259–272, [0710.2448].

[55] A. van Hameren, *Kaleu: a general-purpose parton-level phase space generator*, 1003.4953.

[56] K. Kutak, R. Maciuła, M. Serino, A. Szczurek, and A. van Hameren, *Search for optimal conditions for exploring double-parton scattering in four-jet production: $\mathrm{k_t}$-factorization approach*, *Phys. Rev.* **D94** (2016), no. 1 014019, [1605.08240].

[57] K. Kutak, R. Maciuła, M. Serino, A. Szczurek, and A. van Hameren, *Four-jet production in single- and double-parton scattering within high-energy factorization*, *JHEP* **04** (2016) 175, [1602.06814].

[58] A. van Hameren, P. Kotko, and K. Kutak, *Resummation effects in the forward production of Z_0+jet at the LHC*, *Phys. Rev.* **D92** (2015), no. 5 054007, [1505.02763].

[59] A. van Hameren, R. Maciuła, and A. Szczurek, *Production of two charm quark-antiquark pairs in single-parton scattering within the k_t-factorization approach*, *Phys. Lett.* **B748** (2015) 167–172, [1504.06490].

[60] J. Alwall *et al.*, *A Standard format for Les Houches event files*, *Comput. Phys. Commun.* **176** (2007) 300–304, [hep-ph/0609017].

[61] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, and G. Watt, *LHAPDF6: parton density access in the LHC precision era*, *Eur. Phys. J.* **C75** (2015) 132, [1412.7420]. http://lhapdf.hepforge.org/.

[62] A. Cafarella, C. G. Papadopoulos, and M. Worek, *Helac-Phegas: A Generator for all parton level processes*, *Comput.Phys.Commun.* **180** (2009) 1941–1955, [0710.2427].

[63] *Proceedings of the Sixth International Workshop on Multiple Partonic Interactions at the Large Hadron Collider*, 2015.

[64] M. Nefedov and V. Saleev, *Diphoton production at the Tevatron and the LHC in the NLO approximation of the parton Reggeization approach*, *Phys. Rev.* **D92** (2015), no. 9 094033, [1505.01718].

[65] M. A. Kimber, A. D. Martin, and M. G. Ryskin, *Unintegrated parton distributions and prompt photon hadroproduction*, *Eur. Phys. J.* **C12** (2000) 655–661, [hep-ph/9911379].

[66] H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro, J. F. von Soden-Fraunhofen, and F. Tramontano, *NLO QCD corrections to the production of Higgs plus two jets at the LHC*, *Phys. Lett.* **B721** (2013) 74–81, [1301.0493].

[67] P. Draggiotis, R. H. Kleiss, and C. G. Papadopoulos, *On the computation of multigluon amplitudes*, *Phys.Lett.* **B439** (1998) 157–164, [hep-ph/9807207].

[68] P. Kotko, K. Kutak, C. Marquet, E. Petreska, S. Sapeta, and A. van Hameren, *Improved TMD factorization for forward dijet production in dilute-dense hadronic collisions*, *JHEP* **09** (2015) 106, [1503.03421].

[69] M. Bury, P. Kotko, and K. Kutak, *TMD gluon distributions for multiparton processes*, *Eur. Phys. J.* **C79** (2019), no. 2 152, [1809.08968].

[70] A. Kanaki and C. G. Papadopoulos, *HELAC-PHEGAS: Automatic computation of helicity amplitudes and cross-sections*, hep-ph/0012004.

[71] C. G. Papadopoulos and M. Worek, *Multi-parton cross sections at hadron colliders*, *Eur.Phys.J.* **C50** (2007) 843–856, [hep-ph/0512150].

[72] F. Maltoni, K. Paul, T. Stelzer, and S. Willenbrock, *Color flow decomposition of QCD amplitudes*, *Phys.Rev.* **D67** (2003) 014026, [hep-ph/0209271].

[73] F. James, *RANLUX: A FORTRAN implementation of the high quality pseudorandom number generator of Luscher*, *Comput. Phys. Commun.* **79** (1994) 111–114. [Erratum: Comput. Phys. Commun.97,357(1996)].

A PDF sums

In the following, the indices $1, 2$ refer to initial-state 1 and 2. Each left-hand-side below implies the sum on the right-hand side, because for each term on the right-hand side the numerical value of the matrix element is identical. We consider the 5-flavor case. For the 4-flavor case b and $\bar{\mathrm{b}}$ quarks are removed, for the 3-flavor case also c and $\bar{\mathrm{c}}$ quarks are removed, and for the 2-flavor case also s and $\bar{\mathrm{s}}$ quarks are removed. In case of no electro-weak interactions, there are the following initial-state cases, which all need to be included for example for the processes $\mathrm{pp} \to$ jets:

$$\mathtt{g\ g}: g_1 g_2 \tag{23}$$

$$\mathtt{g\ q}: g_1(u_2+\bar{u}_2+c_2+\bar{c}_2+d_2+\bar{d}_2+s_2+\bar{s}_2+b_2+\bar{b}_2) \tag{24}$$

$$\mathtt{q\ g}: (u_1+\bar{u}_1+c_1+\bar{c}_1+d_1+\bar{d}_1+s_1+\bar{s}_1+b_1+\bar{b}_1)g_2 \tag{25}$$

$$\mathtt{q\ q}: u_1u_2+\bar{u}_1\bar{u}_2+c_1c_2+\bar{c}_1\bar{c}_2+d_1d_2+\bar{d}_1\bar{d}_2+s_1s_2+\bar{s}_1\bar{s}_2+b_1b_2+\bar{b}_1\bar{b}_2 \tag{26}$$

$$\mathtt{q\ q\sim}: u_1\bar{u}_2+\bar{u}_1u_2+c_1\bar{c}_2+\bar{c}_1c_2+d_1\bar{d}_2+\bar{d}_1d_2+s_1\bar{s}_2+\bar{s}_1s_2+b_1\bar{b}_2+\bar{b}_1b_2 \tag{27}$$

$$\begin{aligned}\mathtt{q\ r}: &u_1(\quad c_2+d_2+s_2+b_2+) + \bar{u}_1(\quad \bar{c}_2+\bar{d}_2+\bar{s}_2+\bar{b}_2)\\ &+c_1(u_2+\quad d_2+s_2+b_2+) + \bar{c}_1(\bar{u}_2+\quad \bar{d}_2+\bar{s}_2+\bar{b}_2)\\ &+d_1(u_2+c_2++\quad s_2+b_2) + \bar{d}_1(\bar{u}_2+\bar{c}_2+\quad \bar{s}_2+\bar{b}_2)\\ &+s_1(u_2+c_2++d_2+\quad b_2) + \bar{s}_1(\bar{u}_2+\bar{c}_2+\bar{d}_2+\quad \bar{b}_2)\\ &+b_1(u_2+c_2++d_2+s_2\quad) + \bar{b}_1(\bar{u}_2+\bar{c}_2+\bar{d}_2+\bar{s}_2\quad)\end{aligned} \tag{28}$$

$$\begin{aligned}\mathtt{q\ r\sim}: &u_1(\quad \bar{c}_2+\bar{d}_2+\bar{s}_2+\bar{b}_2) + \bar{u}_1(\quad c_2+d_2+s_2+b_2+)\\ &+c_1(\bar{u}_2+\quad \bar{d}_2+\bar{s}_2+\bar{b}_2) + \bar{c}_1(u_2+\quad d_2+s_2+b_2+)\\ &+d_1(\bar{u}_2+\bar{c}_2+\quad \bar{s}_2+\bar{b}_2) + \bar{d}_1(u_2+c_2++\quad s_2+b_2)\\ &+s_1(\bar{u}_2+\bar{c}_2+\bar{d}_2+\quad \bar{b}_2) + \bar{s}_1(u_2+c_2++d_2+\quad b_2)\\ &+b_1(\bar{u}_2+\bar{c}_2+\bar{d}_2+\bar{s}_2\quad) + \bar{b}_1(u_2+c_2++d_2+s_2\quad)\end{aligned} \tag{29}$$

In the case electro-weak interactions are involved, the foregoing is incorrect because the matrix elements are not equal for all those combination. Then, the correct decomposition is as follows:

$$\begin{aligned}
\texttt{g u}&: g_1(u_2+c_2) & \texttt{g d}&: g_1(d_2+s_2+b_2) & (30)\\
\texttt{g u\~{}}&: g_1(\bar{u}_2+\bar{c}_2) & \texttt{g d\~{}}&: g_1(\bar{d}_2+\bar{s}_2+\bar{b}_2) & (31)\\
\texttt{u g}&: (u_1+c_1)g_2 & \texttt{d g}&: (d_1+s_1+b_1)g_2 & (32)\\
\texttt{u\~{} g}&: (\bar{u}_1+\bar{c}_1)g_2 & \texttt{d\~{} g}&: (\bar{d}_1+\bar{s}_1+\bar{b}_1)g_2 & (33)
\end{aligned}$$

$$\begin{aligned}
\texttt{u u}&: u_1u_2+c_1c_2 & \texttt{d d}&: d_1d_2+s_1s_2+b_1b_2 & (34)\\
\texttt{u\~{} u\~{}}&: \bar{u}_1\bar{u}_2+\bar{c}_1\bar{c}_2 & \texttt{d\~{} d\~{}}&: \bar{d}_1\bar{d}_2+\bar{s}_1\bar{s}_2+\bar{b}_1\bar{b}_2 & (35)\\
\texttt{u u\~{}}&: u_1\bar{u}_2+c_1\bar{c}_2 & \texttt{d d\~{}}&: d_1\bar{d}_2+s_1\bar{s}_2+b_1\bar{b}_2 & (36)\\
\texttt{u\~{} u}&: \bar{u}_1u_2+\bar{c}_1c_2 & \texttt{d\~{} d}&: \bar{d}_1d_2+\bar{s}_1s_2+\bar{b}_1b_2 & (37)
\end{aligned}$$

$$\begin{aligned}
\texttt{u d}&: (u_1+c_1)(d_2+s_2+b_2) & \texttt{u d\~{}}&: (u_1+c_1)(\bar{d}_2+\bar{s}_2+\bar{b}_2) & (38)\\
\texttt{u\~{} d}&: (\bar{u}_1+\bar{c}_1)(d_2+s_2+b_2) & \texttt{u\~{} d\~{}}&: (\bar{u}_1+\bar{c}_1)(\bar{d}_2+\bar{s}_2+\bar{b}_2) & (39)\\
\texttt{d u}&: (d_1+s_1+b_1)(u_2+c_2) & \texttt{d\~{} u}&: (\bar{d}_1+\bar{s}_1+\bar{b}_1)(u_2+c_2) & (40)\\
\texttt{d u\~{}}&: (d_1+s_1+b_1)(\bar{u}_2+\bar{c}_2) & \texttt{d\~{} u\~{}}&: (\bar{d}_1+\bar{s}_1+\bar{b}_1)(\bar{u}_2+\bar{c}_2) & (41)
\end{aligned}$$

B The meaning of `Esoft`

Pre-sampling is performed with KALEU [55], which constructs phase space points from invariants and angles that are generated following pre-defined probability distributions augmented with adaptive grids, following the method of [54][2]. The generation of the invariants is such that it mimics the behavior of the squared matrix element as function of the invariants. For example, if the final-state momenta p_i, p_j belong to on-shell gluons, then the squared matrix element behaves as $1/s_{ij}$ where $s_{ij} = (p_i + p_j)^2$. The singularity at $s_{ij} = 0$ is protected by the phase space cuts, but the squared matrix element will still show a steep behavior as function of s_{ij}. Therefore, the predefined density for the invariant is

$$P_{ij}(s_{ij}) = \frac{\theta\big(E_{\text{cm}}^2 - s_{ij}\big)}{1+\log\left(E_{\text{cm}}^2/E_{\text{soft}}^2\right)}\left[\frac{\theta\big(E_{\text{soft}}^2 - s_{ij}\big)}{E_{\text{soft}}^2} + \frac{\theta\big(s_{ij} - E_{\text{soft}}^2\big)}{s_{ij}}\right]. \tag{42}$$

It increases for decreasing s_{ij} until $s_{ij} = E_{\text{soft}}^2$, from where it stays constant, ensuring the coverage of the whole phase space. The exact value of E_{soft} does not matter too much, since also a bad choice will be corrected by the adaptive grid. A good choice will, however, help in the efficiency.

[2]AVHLIB employs RANLUX [73] for the generation of pseudo random numbers.

C The custom event file format

After obvious information in the header, the events are listed. Each event block starts with, for example

```
EVENT WEIGHT: 0.5778970106138136E+09
```

where the floating point number is the weight value W assigned to the event. The weights are normalized such that

$$\frac{\sum_{\text{events}} W}{\sum_{\text{events}} 1} = \text{total cross-section} \;.$$

The next line contains a single integer, referring to the process corresponding to the event. The process numbering is given in the header of the event file. The next lines consist of 5 floating point numbers and 3 integers, containing the momenta, the color flow, and the helicities of the event:

$$E \qquad p_x \qquad p_y \qquad p_z \qquad E^2 - p_x^2 - p_y^2 - p_z^2 \qquad \text{color} \qquad \text{anti-color} \qquad \text{helicity}$$

Initial-state momenta have a negative value of the energy E. For them, the helicity is not defined. Then follows a line consisting of 4 floating point numbers, containing the value of

$$\text{matrix element} \qquad \text{parton luminosity} \qquad \alpha_S \qquad \text{renormalization scale}$$

The matrix element is not averaged regarding the initial-state partons. The parton luminosity is just the product of the PDFs, as $x_1 f_1\, x_2 f_2$, not $f_1 f_2$. This number is also included in the LHEF, in lines starting with `#pdf1pdf2` . If the user did not set `partlumi = combined`, then the next two lines contain the values of the individual pdfs as

$$\begin{array}{cccc} \text{pdf}_B & x_B & k_{TB} & \mu_B \\ \text{pdf}_A & x_A & k_{TA} & \mu_A \end{array}$$

where A refers to the initial state with the postive rapidity, and B to the initial state with the negative helicity. These lines are included into the LHEF starting with the strings `#pdf_posRap` and `#pdf_negRap`.

For multi-parton scattering, event blocks are repeated for a single event weight. For MPI, event files can only be created in the custom format for now, and not in the LHEF format.