

DESY 19-028
IPPP/19/13
MCnet-19-05

HEJ 2: High Energy Resummation for Hadron Colliders

Jeppe R. Andersen^a, Tuomas Hapola^a, Marian Heil^a, Andreas Maier^{b,*},
Jennifer Smillie^c

^a*Institute for Particle Physics Phenomenology,
University of Durham, South Road, Durham DH1 3LE, UK*

^b*Deutsches Elektronen-Synchrotron, DESY, Platanenallee 6, 15738 Zeuthen, Germany*

^c*Higgs Centre for Theoretical Physics, University of Edinburgh,
Peter Guthrie Tait Road, Edinburgh EH9 3FD, UK*

Abstract

We present HEJ 2, a new implementation of the *High Energy Jets* formalism for high-energy resummation in hadron-collider processes as a flexible Monte Carlo event generator. In combination with a conventional fixed-order event generator, HEJ 2 can be used to obtain greatly improved predictions for a number of phenomenologically important processes by adding all-order logarithmic corrections in \hat{s}/p_{\perp}^2 . A prime example for such a process is the gluon-fusion production of a Higgs boson in association with widely separated jets, which constitutes the dominant background to Higgs boson production in weak-boson fusion.

Keywords: Quantum chromodynamics; Collider Physics; Resummation; Monte Carlo Event Generators

PROGRAM SUMMARY

Program Title: HEJ 2

Licensing provisions: GPLv2 or later

Programming language: C++

Nature of problem:

Conventional event generators for high-energy colliders are based on fixed-order perturbation theory, which converges poorly in regions of phase space with large invariant masses between jets. However, reliable predictions in these regions are of particular importance whenever cuts relevant to weak-boson fusion or weak-boson scattering are applied. One highly relevant example is the separation of gluon fusion and weak-boson fusion contributions to the production of a Higgs boson in association with two or more jets.

Solution method:

The perturbative description can be amended by resumming the high-energy logarithms associated with large invariant masses to all orders in perturbation theory. HEJ 2 implements this resummation following the *High Energy Jets* framework. Using HEJ 2 in conjunction with a conventional event generator allows one to obtain reliable predictions for processes involving multiple jets in high-energy regions of phase space.

Additional comments including Restrictions and Unusual features:

The current version of HEJ 2 implements multi-jet processes with at most a single Higgs boson in the final state. Matching to fixed-order perturbation theory is currently restricted to leading order.

1. Introduction

Measurements at the Large Hadron Collider (LHC) are testing our knowledge of elementary particle physics to unprecedented detail. Especially in the light of the High-Luminosity LHC upgrade it is instrumental to develop a thorough understanding of scattering processes over a wide range of phase space, including specific corners of interest.

The high-energy region of phase space is characterised by large invariant masses, or alternatively large rapidity separations, between pairs of jets. This region is of particular relevance in the production of a Higgs boson together with jets through weak-boson fusion, where the partons are typically scattered in the forward region. Similar kinematics are observed in weak-boson scattering. Experimental identification of such processes requires a precise and reliable theoretical prediction for the background, which is dominated by gluon fusion.

In the high-energy limit, the convergence of conventional perturbation theory is spoiled by large logarithms of invariant masses. While these logarithms can be predicted and resummed to all orders in the framework of BFKL[1–4], it is found that a prediction based on *only* the logarithmic behaviour is

generally not sufficient in the phase space probed by current colliders. The *High Energy Jets* framework pioneered in [5–7] remedies this shortcoming by performing a systematic power expansion of the scattering amplitude thereby retaining the logarithmic accuracy of BFKL while allowing event-by-event matching to fixed-order matrix elements. As approximations are only applied to the scattering matrix elements and not to the phase space, it is possible to obtain fully exclusive predictions. A review of *High Energy Jets* can be found in [8]. The method was implemented in the HEJ Monte Carlo event generator and validated by comparing to measurements of multi-jet production and the production of W or Z bosons in associations with jets in the high-energy region [9–16].

In order to benefit from recent progress in the formulation of *High Energy Jets*, it has been necessary to rewrite large parts of the implementation. Here, we present the new program HEJ 2. HEJ 2 is based on a recently developed matching algorithm, where events in the resummation phase space are generated from fixed-order events produced with a conventional fixed-order event generator. A detailed description of the algorithm and its implementation is given in [17]. In contrast to the previously used formulation, it allows fixed-order matching to much higher multiplicities and to loop-induced processes. As matching is no longer tied to hard-wired matrix elements, the scope of matching is much broader than with the previous version of HEJ.

HEJ 2 has a much more modular structure, which makes it easier to add and maintain code for new processes. It is straightforward to interface HEJ 2 to arbitrary fixed-order event generators producing events in the Les Houches Event File format. HEJ 2 natively supports `Rivet` [18] analyses and the standard LHEF [19, 20] and `HepMC` [21, 22] formats for event output. It is also possible to use custom analyses. Not only can HEJ 2 be used as a stand-alone program, but also as a library. This will for example facilitate future combinations with parton showers, cf. [23, 24].

In the following we explain how to install and run HEJ 2 in sections 2 and 3, respectively. Section 4 describes the HEJ fixed-order generator, which can generate leading-order high-multiplicity events in the high-energy approximation. In sections 5 and 6, we show how a user-defined analysis and custom renormalisation and factorisation scales can be defined. Finally, in section 7 we give an example how HEJ 2 can be used as a library. Continuously updated documentation for HEJ can be found at <https://hej.web.cern.ch>.

2. Installation

2.1. Download

A tar archive of the HEJ 2 source code can be downloaded and decompressed with the command

```
curl https://hej.web.cern.ch/HEJ/downloads/HEJ_2.0.tar.gz | tar -xz
```

To obtain the latest stable HEJ version, `HEJ_2.0.tar.gz` should be replaced by `HEJ.tar.gz`.

Alternatively, the HEJ source code can be obtained by installing the git version control system [25] and running

```
git clone https://phab.hepforge.org/source/hej.git
```

We also provide a Docker [26] image containing a HEJ 2 installation on <https://hub.docker.com/r/hejdock/hej>. This image can be pulled with

```
docker pull hejdock/hej
```

When using the Docker image the remaining installation steps can be skipped.

2.2. Prerequisites

Before installing HEJ 2, the following programs and libraries are needed:

- CMake version 3.1 [27, 28] or later.
- A compiler supporting the C++14 standard, for example gcc 5 [29] or later.
- FastJet [30].
- CLHEP version 2.3.1 or later [31]
- LHAPDF6 [32, 33].
- The IOStreams and uBLAS boost libraries [34].
- yaml-cpp [35].

HEJ 2 was tested with CMake version 3.12.3, gcc 5.3.0, FastJet 3.3.1, CLHEP 2.4.1.0, LHAPDF 6.1.6, boost 1.68, and yaml-cpp 0.6.2 as well as with various other combinations. For including finite top mass corrections in Higgs boson plus jets production, version 2 of the QCDLoop library [36, 37] is required in addition to the above libraries. HEJ 2 supports Rivet [18] and versions 2 and 3 of HepMC [21, 22] if any of them are installed, but does not require them.

2.3. Compilation

HEJ 2 can be compiled and installed with the commands

```
cmake HEJ2/source/directory -DCMAKE_INSTALL_PREFIX=target/directory
make install
```

where `HEJ2/source/directory` is the directory containing the file `CMakeLists.txt`. If `-DCMAKE_INSTALL_PREFIX=target/directory` is omitted HEJ 2 will be installed to some default location.

In case some of the aforementioned prerequisites are not found by `cmake` a hint can be given by adding an additional argument

```
-Dlibname_ROOT_DIR=/directory/with/library
```

where `libname` should be replaced by the name of the library in question.

2.4. Testing

The installation can be tested after downloading the NNPDF 2.3 PDF set with

```
lhpdf install NNPDF23_nlo_as_0119
```

The tests can be run with the command

```
make test
```

3. Running HEJ 2

3.1. Quick start

In order to run, HEJ 2 needs a configuration file and a file containing fixed-order events. A sample configuration is given by the `config.yml` file distributed together with HEJ 2. Events in the Les Houches Event File format can be generated with standard Monte Carlo generators like `MadGraph5_aMC@NLO` [38] or `Sherpa` [39]. It is also possible to use Les Houches Event Files compressed with `gzip` [40] as input. HEJ 2 assumes that the cross section is given by the sum of the event weights. Depending on the fixed-order generator it may be necessary to adjust the weights in the Les Houches Event File accordingly.

The processes supported by HEJ 2 so far are

- pure multijet production,
- production of a Higgs boson with jets,

where at least two jets are required in each case. Only leading-order events are supported; work is ongoing to extend the matching to next-to-leading-order.

After generating an event file `events.lhe`, adjust the parameters under the `fixed order jets` setting in `config.yml` to the settings in the fixed-order generation. Resummation can then be added by running

```
HEJ config.yml events.lhe
```

Using the default settings, this will produce an output event file `HEJ.lhe` with events including high-energy resummation.

When using the Docker image, HEJ can be run with

```
docker run -v $PWD:$PWD -w $PWD hejdock/hej HEJ config.yml events.lhe
```

3.2. Settings

HEJ 2 configuration files follow the YAML [41] format. The following configuration parameters are supported:

- **trials**: High-energy resummation is performed by generating a number of resummation phase space configurations corresponding to an input fixed-order event. This parameter specifies how many such configurations HEJ 2 should try to generate for each input event. Typical values vary between 10 and 100.
- **min extparton pt**: Specifies the minimum transverse momentum in GeV of the most forward and the most backward parton. This setting is needed to regulate an otherwise uncancelled divergence. Its value should be slightly below the minimum transverse momentum of jets specified by **resummation jets: min pt**. See also the **max ext soft pt fraction** setting.
- **max ext soft pt fraction**: Specifies the maximum fraction that soft radiation can contribute to the transverse momentum of each the most forward and the most backward jet. Values between around 0.05 and 0.1 are recommended. See also the **min extparton pt** setting.
- **fixed order jets**: This tag collects a number of settings specifying the jet definition in the event input. The settings should correspond to the ones used in the fixed-order Monte Carlo that generated the input events.

- **min pt**: Minimum transverse momentum in GeV of fixed-order jets.
 - **algorithm**: The algorithm used to define jets. Allowed settings are `kt`, `cambridge`, `antikt`, `cambridge for passive`. See the FastJet documentation for a description of these algorithms [30].
 - **R**: The R parameter used in the jet algorithm, roughly corresponding to the jet radius in the plane spanned by the rapidity and the azimuthal angle.
- **resummation jets**: This tag collects a number of settings specifying the jet definition in the observed, i.e. resummed events. These settings are optional, by default the same values as for the **fixed order jets** are assumed.
 - **min pt**: Minimum transverse momentum in GeV of resummation jets. This should be between 25% and 50% larger than the minimum transverse momentum of fixed order jets set by **fixed order jets: min pt**.
 - **algorithm**: The algorithm used to define jets. HEJ 2 can cover the resummation phase space particularly efficiently when using `antikt jets` [42], so this value is strongly recommended. For a list of possible other values, see the **fixed order jets: algorithm** setting.
 - **R**: The R parameter used in the jet algorithm.
- **FKL**: Specifies how to treat events respecting FKL rapidity ordering. These configurations are dominant in the high-energy limit (see [8]). The possible values are `reweight` to enable resummation, `keep` to keep the events as they are up to a possible change of renormalisation and factorisation scale, and `discard` to discard these events.
 - **unordered**: Specifies how to treat events with one emission that does not respect FKL ordering. In the high-energy limit, such configurations are logarithmically suppressed compared to FKL configurations. The possible values are the same as for the **FKL** setting, but `reweight` is currently only supported for Higgs boson plus jets production.

- **non-HEJ**: Specifies how to treat events where no resummation is possible. The allowed values are `keep` to keep the events as they are up to a possible change of renormalisation and factorisation scale and `discard` to discard these events.
- **scales**: Specifies the renormalisation and factorisation scales for the output events. This can either be a single entry or a list [`scale1`, \leftrightarrow `scale2`, ...]. For the case of a list the first entry defines the central scale. Possible values are fixed numbers to set the scale in GeV or the following:
 - `H_T`: The sum of the scalar transverse momenta of all final-state particles.
 - `max jet pperp`: The maximum transverse momentum of all jets.
 - `jet invariant mass`: Sum of the invariant masses of all jets.
 - `m_1j2`: Invariant mass between the two hardest jets.

Scales can be multiplied or divided by an overall factor, e.g. `H_T/2`.

It is also possible to import scales from an external library, see section 6.

- **scale factors**: A list of numeric factors by which each of the scales should be multiplied. The renormalisation scale μ_r and the factorisation scales μ_f are varied independently. For example, a list with entries [`0.5`, `2`] would give the four scale choices $(0.5\mu_r, 0.5\mu_f)$, $(0.5\mu_r, 2\mu_f)$, $(2\mu_r, 0.5\mu_f)$, $(2\mu_r, 2\mu_f)$ in this order. The ordering corresponds to the order of the final event weights.
- **max scale ratio**: Specifies the maximum factor by which renormalisation and factorisation scales may differ. For a value of 2 and the example given for the scale factors the scale choices $(0.5\mu_r, 2\mu_f)$ and $(2\mu_r, 0.5\mu_f)$ will be discarded.
- **log correction**: Whether to include corrections due to the evolution of the strong coupling constant in the virtual corrections. Allowed values are `true` and `false`.
- **event output**: Specifies the name of a single event output file or a list of such files. The file format is either specified explicitly or

derived from the suffix. For example, `events.lhe` or, equivalently `Les` \leftrightarrow \leftrightarrow `Houches: events.lhe` generates an output event file `events.lhe` in the Les Houches format. The supported formats are

- `file.lhe` or `Les Houches: file:` The Les Houches event file format.
- `file.hepmc` or `HepMC: file:` The HepMC format.

- **random generator:** Sets parameters for random number generation.
 - **name:** Which random number generator to use. Currently, `mixmax` [43, 44] and `ranlux64` [45] are supported. See the CLHEP documentation [31] for details on the generators.
 - **seed:** The seed for random generation. This should be a single number for `mixmax` and the name of a state file for `ranlux64`.
- **analysis:** Name and settings for the event analyses; either a custom analysis plug-in or `Rivet`. For the first the `plugin` sub-entry should be set to the analysis file path. All further entries are passed on to the analysis. To use `Rivet` a list of `Rivet` analyses have to be given in `Rivet` and a prefix for the yoda file has to be set through `output`. See section 5 for details.
- **Higgs coupling:** This collects a number of settings concerning the effective coupling of the Higgs boson to gluons. This is only relevant for the production process of a Higgs boson with jets and only supported if HEJ 2 was compiled with QCDLoop support.
 - **use impact factors:** Whether to use impact factors for the coupling to the most forward and most backward partons. Impact factors imply the infinite top-quark mass limit.
 - **mt:** The value of the top-quark mass in GeV. If this is not specified, the limit of an infinite mass is taken.
 - **include bottom:** Whether to include the Higgs coupling to bottom quarks.
 - **mb:** The value of the bottom-quark mass in GeV.

4. The HEJ fixed order generator

For high jet multiplicities event generation with standard fixed-order generators becomes increasingly cumbersome. For example, the leading-order production of a Higgs Boson with five or more jets is computationally prohibitively expensive.

To this end, HEJ 2 provides the HEJFOG fixed-order generator that allows to generate events with high jet multiplicities. To facilitate the computation the limit of Multi-Regge Kinematics with large invariant masses between all outgoing particles is assumed in the matrix elements. The typical use of the HEJFOG is to supplement low-multiplicity events from standard generators with high-multiplicity events before using the HEJ 2 program to add high-energy resummation.

4.1. Installation

The HEJFOG comes bundled together with HEJ 2 and the installation is very similar. After downloading HEJ 2 and installing the prerequisites as described in section 2.2 the HEJFOG can be installed with

```
cmake /path/to/FixedOrderGen -DCMAKE_INSTALL_PREFIX=target/directory ↵  
  ↵ -DCMAKE_BUILD_TYPE=Release  
make install
```

where `/path/to/FixedOrderGen` refers to the `FixedOrderGen` subdirectory in the HEJ 2 directory.

If HEJ 2 was installed to a non-standard location, it may be necessary to specify the directory containing `HEJ-config.cmake`. If the base installation directory is `/path/to/HEJ`, `HEJ-config.cmake` should be found in `/path/to/HEJ/lib/cmake/HEJ` and the commands for installing the HEJFOG would read

```
cmake /path/to/FixedOrderGen -DHEJ_DIR=/path/to/HEJ/lib/cmake/HEJ ↵  
  ↵ -DCMAKE_INSTALL_PREFIX=target/directory  
make install
```

The installation can be tested with::

```
make test
```

provided that the CT10nlo PDF set is installed.

4.2. Running the fixed-order generator

After installing the HEJFOG you can modify the provided configuration file `configF0.yml` and run the generator with:

```
HEJFOG configF0.yml
```

The resulting event file, by default named `HEJF0.he`, can then be fed into HEJ 2 like any event file generated from a standard fixed-order generator, see section 3.1.

4.3. Settings

Similar to HEJ 2, the HEJFOG uses a YAML configuration file. The settings are

- **process:** The scattering process for which events are being generated. The format is `in1 in2 => out1 out2`

The incoming particles, `in1`, `in2` can be

- quarks: `u`, `d`, `u_bar`, and so on,
- gluons: `g`,
- protons `p` or antiprotons `p_bar`.

At most one of the outgoing particles can be a boson. At the moment only the Higgs boson `h` is supported. All other outgoing particles are jets. Multiple jets can be grouped together, so `p p => h j j` is the same as `p p => h 2j`. There have to be at least two jets.

- **events:** Specifies the number of events to generate.
- **jets:** Defines the properties of the generated jets.
 - **min pt:** Minimum jet transverse momentum in GeV.
 - **peak pt:** Optional setting to specify the dominant jet transverse momentum in GeV. If the generated events are used as input for HEJ resummation, this should be set to the minimum transverse momentum of the resummation jets. The effect is that only a small fraction of jets will be generated with a transverse momentum below the value of this setting.

- **algorithm**: The algorithm used to define jets. Allowed settings are `kt`, `cambridge`, `antikt`, `cambridge for passive`. See the FastJet documentation for a description of these algorithms.
 - **R**: The R parameter used in the jet algorithm.
 - **max rapidity**: Maximum absolute value of the jet rapidity.
- **beam**: Defines various properties of the collider beam.
 - **energy**: The beam energy in GeV. For example, the 13 TeV LHC corresponds to a value of 6500.
 - **particles**: A list [`p1`, `p2`] of two beam particles. The only supported entries are protons `p` and antiprotons `p_bar`.
 - **pdf**: The LHAPDF number of the PDF set. For example, 230000 corresponds to an NNPDF 2.3 NLO PDF set.
 - **subleading fraction**: This setting is related to the fraction of events that are not FKL configurations and thus subleading in the high-energy limit. Currently only unordered emissions are implemented, and only for Higgs boson plus multijet processes. This value must be positive and not larger than 1. It should typically be chosen between 0.01 and 0.1. Note that while this parameter influences the chance of generating subleading configurations, it generally does not correspond to the actual fraction of subleading events.
 - **subleading channels**: Optional parameter to select the production of specific channels that are subleading in the high-energy limit. Only has an effect if `subleading fraction` is non-zero. Currently three values are supported:
 - `all`: All subleading channels are allowed. This is the default.
 - `none`: No subleading contribution, only FKL configurations are allowed. This is equivalent to `subleading fraction: 0`.
 - `unordered`: Unordered emission are allowed.
Unordered emission are any rapidity ordering where exactly one gluon is emitted outside the FKL rapidity ordering. More precisely, if at least one of the incoming particles is a quark or antiquark and

there are more than two jets in the final state, `subleading fraction` states the probability that the flavours of the outgoing particles are assigned in such a way that an unordered configuration arises.

- **unweight**: This setting defines the parameters for the partial unweighting of events. You can disable unweighting by removing this entry from the configuration file.
 - **sample size**: The number of weighted events used to calibrate the unweighting. A good default is to set this to the number of target events. If the number of events is large this can lead to significant memory consumption and a lower value should be chosen. Contrarily, for large multiplicities the unweighting efficiency becomes worse and the sample size should be increased.
 - **max deviation**: Controls the range of events to which unweighting is applied. A larger value means that a larger fraction of events are unweighted. Typical values are between -1 and 1.
- **Particle properties**: Specifies various properties of the different particles (Higgs, W or Z). This is only relevant if the chosen process is the production of the corresponding particles with jets. For example, for the process $p p \Rightarrow h 2j$ the `mass`, `width` and (optionally) `decays` of the `Higgs` boson are required, while all other particle properties will be ignored. In the current version, the production of W and Z bosons is not implemented and those entries will always be ignored. This will change in future versions.
 - **Higgs, W+, W- or Z**: The particle (Higgs, W^+ , W^- , Z) for which the following properties are defined.
 - * **mass**: The mass of the particle in GeV.
 - * **width**: The total decay width of the particle in GeV.
 - * **decays**: Optional setting specifying the decays of the particle. Only the decay into two particles is implemented. Each decay has the form `{into: [p1,p2], branching ratio: r}` where `p1` and `p2` are the particle names of the decay product (e.g. `photon`) and `r` is the branching ratio. Decays of a Higgs boson are treated as the production and subsequent decay of an on-shell Higgs boson, so decays into e.g. Z bosons are not supported.

- **scales**: Specifies the renormalisation and factorisation scales for the output events. For details, see the corresponding entry in section 3.2. Note that this should usually be a single value, as the weights resulting from additional scale choices will simply be ignored when adding high-energy resummation with HEJ 2.
- **event output**: Specifies the name of a single event output file or a list of such files. See the corresponding entry in section 3.2 for details.
- **random generator**: Sets parameters for random number generation. See section 3.2 for details.
- **analysis**: Specifies the name and settings for a custom analysis library. This can be useful to specify cuts at the fixed-order level. See the corresponding entry in section 3.2 for details.
- **Higgs coupling**: This collects a number of settings concerning the effective coupling of the Higgs boson to gluons. See the corresponding entry in section 3.2 for details

5. Writing custom analyses

HEJ 2 and the HEJ fixed-order generator can generate `HepMC` files, so it is always possible to run a `Rivet` analysis on these. However if HEJ 2 was compiled with `Rivet` support one can use the native `Rivet` interface. For example

```
analysis:
  rivet: [MC_XS, MC_JETS]
  output: HEJ
```

would call the generic `MC_XS` and `MC_JETS` `Rivet` analyses and write the result into `HEJ[.Scalename].yoda`. HEJ 2 will then run `Rivet` over all different scales separately and write out each into a different yoda file. Alternatively instead of using `Rivet`, one can provide a custom analysis inside a C++ library.

An analysis is a class that derives from the abstract `Analysis` base class provided by HEJ 2. It has to implement three public functions:

- The `pass_cuts` member function return true if and only if the given event (first argument) passes the analysis cuts.

- The `fill` member function adds an event to the analysis, which for example can be used to fill histograms. HEJ 2 will only pass events for which `pass_cuts` has returned true.
- The `finalise` member function is called after all events have been processed. It can be used, for example, to print out or save the analysis results.

The `pass_cuts` and `fill` functions take two arguments: the resummation event generated by HEJ 2 and the original fixed-order input event. Usually, the second argument can be ignored. It can be used, for example, for implementing cuts that depend on the ratio of the weights between the fixed-order and the resummation event.

In addition to the two member functions, there has to be a global `make_analysis` function that takes the analysis parameters in the form of a `YAML::Node` and returns a `std::unique_ptr` to the `Analysis`.

The following code creates the simplest conceivable analysis.

```
#include <memory> // for std::unique_ptr

#include "HEJ/Analysis.hh"

class MyAnalysis: public HEJ::Analysis {
public:
    MyAnalysis(YAML::Node const & /* config */) {}

    void fill(
        HEJ::Event const & /* event */,
        HEJ::Event const & /* FO_event */
    ) override { }

    bool pass_cuts(
        HEJ::Event const & /* event */,
        HEJ::Event const & /* FO_event */
    ) override {
        return true;
    }

    void finalise() override { }
};

extern "C"
std::unique_ptr<HEJ::Analysis> make_analysis(
    YAML::Node const & config
){
```

```

    return std::make_unique<MyAnalysis>(config);
}

```

After saving this code to a file, for example `myanalysis.cc`, this code can be compiled into a shared library. Using the `g++` compiler, the library can be built with

```

g++ $(HEJ-config --cxxflags) -fPIC -shared ↵
    ↵ -Wl,-soname,libmyanalysis.so -o libmyanalysis.so myanalysis.cc

```

With `g++` it is also good practice to add `__attribute__((visibility("default")))` after `extern "C"` in the above code snippet and then compile with the additional flag `-fvisibility=hidden` to prevent name clashes.

The analysis can be used in HEJ 2 or the HEJ fixed-order generator by adding

```

analysis:
  plugin: /path/to/libmyanalysis.so

```

to the `.yaml` configuration file.

As a more interesting example, here is the code for an analysis that sums up the total cross section and prints the result to both standard output and a file specified in the `.yaml` config with

```

analysis:
  plugin: analysis/build/directory/src/libmy_analysis.so
  output: outfile

```

To access the configuration at run time, HEJ 2 uses the `yaml-cpp` library. The analysis code itself is

```

#include <memory>
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>

#include "HEJ/Analysis.hh"
#include "HEJ/Event.hh"

#include "yaml-cpp/yaml.h"

class MyAnalysis: public HEJ::Analysis {
public:
  MyAnalysis(YAML::Node const & config):
    xsection_{0.}, xsection_error_{0.},
    outfile_{config["output"].as<std::string>}

```



```

{}

void fill(
    HEJ::Event const & event,
    HEJ::Event const & /* FO_event */
) override {
    const double wt = event.central().weight;
    xsection_ += wt;
    xsection_error_ += wt*wt;
}

bool pass_cuts(
    HEJ::Event const & /* event */,
    HEJ::Event const & /* FO_event */
) override {
    return true;
}

void finalise() override {
    std::cout << "cross section: " << xsection_ << " +- "
        << std::sqrt(xsection_error_) << "\n";
    std::ofstream fout{outfile_};
    fout << "cross section: " << xsection_ << " +- "
        << std::sqrt(xsection_error_) << "\n";
}

private:
double xsection_, xsection_error_;
std::string outfile_;
};

extern "C"
std::unique_ptr<HEJ::Analysis> make_analysis(
    YAML::Node const & config
){
    return std::make_unique<MyAnalysis>(config);
}

```

6. Custom scales

HEJ 2 comes with a small selection of built-in renormalisation and factorisation scales, as described in section 3.2. In addition to this, user-defined scales can be imported from custom libraries.

6.1. Writing the library

Custom scales are defined through C++ functions that take an event and compute the corresponding scale. As an example, let us consider a function returning the transverse momentum of the softest jet in an event. To make it accessible from HEJ 2, we have to prevent C++ name mangling with `extern "C"`:

```
#include "HEJ/Event.hh"

extern "C"
double softest_jet_pt(HEJ::Event const & ev){
    const auto softest_jet = sorted_by_pt(ev.jets()).back();
    return softest_jet.perp();
}
```

After saving this code to some file `myscales.cc`, we can compile it to a shared library. With the `g++` compiler this can be done with the command

```
g++ $(HEJ-config --cxxflags) -fPIC -shared ↵
    ↵ -Wl,-soname,libmyscales.so -o libmyscales.so myscales.cc
```

6.2. Importing the scale into HEJ 2

Our custom scale can now be imported into HEJ 2 by adding the following lines to the YAML configuration file

```
import scales:
  /path/to/libmyscales.so: softest_jet_pt
```

It is also possible to import several scales from one or more libraries:

```
import scales:
  /path/to/libmyscales1.so: [first_scale, second_scale]
  /path/to/libmyscales2.so: [another_scale, yet_another_scale]
```

The custom scales can then be used as usual in the `scales` setting, for example

```
scales: [H_T, softest_jet_pt, 2*softest_jet_pt]
```

7. Using HEJ 2 as a library

As mentioned before, HEJ 2 can also be used as a library, which allows lots of flexibility. The documentation of the complete functionality can be found on <https://hej.web.cern.ch/HEJ/doc/2.0/library>.

As an example, we show a toy program that computes the square of a matrix element in the HEJ approximation for a single event. First, we include the necessary header files:

```
#include "HEJ/Event.hh"
#include "HEJ/MatrixElement.hh"
```

We then specify the incoming and outgoing particles. A particle has a type and four-momentum (p_x, p_y, p_z, E) . For instance, an incoming gluon could be defined as

```
fastjet::PseudoJet momentum{0, 0, 308., 308.};
HEJ::Particle gluon_in{HEJ::ParticleID::gluon, momentum};
```

We collect all incoming and outgoing particles in a partonic event. Here is an example for a partonic $gu \rightarrow gghu$ event:

```
HEJ::UnclusteredEvent partonic_event;

// incoming particles
partonic_event.incoming[0] = {
    HEJ::ParticleID::gluon,
    { 0., 0., 308., 308.}
};
partonic_event.incoming[1] = {
    HEJ::ParticleID::up,
    { 0., 0., -164., 164.}
};
// outgoing particles
partonic_event.outgoing.push_back({
    HEJ::ParticleID::higgs,
    { 98., 82., 14., 180.}
});
partonic_event.outgoing.push_back({
    HEJ::ParticleID::up,
    { 68., -54., 36., 94.}
});
partonic_event.outgoing.push_back({
    HEJ::ParticleID::gluon,
    {-72., 9., 48., 87.}
});
partonic_event.outgoing.push_back({
    HEJ::ParticleID::gluon,
    {-94., -37., 46., 111.}
});
```

Alternatively, we could read the event from a Les Houches event file, possibly compressed with `gzip`. For this, the additional header files `HEJ/stream.hh` and `LHEF/LHEF.h` have to be included.

```
HEJ::istream in{"events.lhe.gz"};
LHEF::Reader reader{in};
reader.readEvent();
HEJ::UnclusteredEvent partonic_event{reader.hepeup};
```

In this specific example we will later choose a constant value for the strong coupling, so that the HEJ matrix element does not depend on the renormalisation scale. However, in a more general scenario, we will want to set a central scale:

```
partonic_event.central.mur = 50.;
```

It is possible to add more scales in order to perform scale variation:

```
partonic_event.variations.resize(2);
partonic_event.variations[0].mur = 25.;
partonic_event.variations[1].mur = 100.;
```

In the next step, we leverage FastJet to construct an event with clustered jets. Here, we use anti- k_t jets with $R = 0.4$ and transverse momenta of at least 30 GeV.

```
const fastjet::JetDefinition jet_def{
    fastjet::JetAlgorithm::antikt_algorithm, 0.4
};
const double min_jet_pt = 30.;
HEJ::Event event{partonic_event, jet_def, min_jet_pt};
```

In order to calculate the Matrix element, we now have to fix the physics parameters. For the sake of simplicity, we assume an effective coupling of the Higgs boson to gluons in the limit of an infinite top-quark mass and a fixed value of $\alpha_s = 0.118$ for the strong coupling.

```
const auto alpha_s = [](double /* mu_r */) { return 0.118; };
HEJ::MatrixElementConfig ME_config;
// whether to include corrections from the
// evolution of \alpha_s in virtual corrections
ME_config.log_correction = false;
HEJ::MatrixElement ME{alpha_s, ME_config};
```

If QCDDLoop is installed, we can also take into account the full loop effects with finite top and bottom quark masses:

```
HEJ::MatrixElementConfig ME_config;
ME_config.Higgs_coupling.use_impact_factors = false;
ME_config.Higgs_coupling.mt = 163;
ME_config.Higgs_coupling.include_bottom = true;
ME_config.Higgs_coupling.mb = 2.8;
```

Finally, we can compute and print the square of the matrix element with

```
std::cout << "HEJ ME: " << ME(event).central << '\n';
```

In the case of scale variation, the weight associated with the scale `event.variations[i].mur` is `ME(event).variations[i]`.

Collecting the above pieces, we have the following program:

```
#include "HEJ/Event.hh"
#include "HEJ/MatrixElement.hh"

int main(){
    HEJ::UnclusteredEvent partonic_event;
    // incoming particles
    partonic_event.incoming[0] = {
        HEJ::ParticleID::gluon,
        { 0., 0., 308., 308.}
    };
    partonic_event.incoming[1] = {
        HEJ::ParticleID::up,
        { 0., 0., -164., 164.}
    };
    // outgoing particles
    partonic_event.outgoing.push_back({
        HEJ::ParticleID::higgs,
        { 98., 82., 14., 180.}
    });
    partonic_event.outgoing.push_back({
        HEJ::ParticleID::up,
        { 68., -54., 36., 94.}
    });
    partonic_event.outgoing.push_back({
        HEJ::ParticleID::gluon,
        {-72., 9., 48., 87.}
    });
    partonic_event.outgoing.push_back({
        HEJ::ParticleID::gluon,
        {-94., -37., 46., 111.}
    });

    const fastjet::JetDefinition jet_def{
        fastjet::JetAlgorithm::antikt_algorithm, 0.4
    };
    const double min_jet_pt = 30.;
    HEJ::Event event{partonic_event, jet_def, min_jet_pt};

    const auto alpha_s = [](double /* mu_r */) { return 0.118; };
    HEJ::MatrixElementConfig ME_config;
    // whether to include corrections from the
    // evolution of \alpha_s in virtual corrections
```

```

ME_config.log_correction = false;
HEJ::MatrixElement ME{alpha_s, ME_config};

std::cout
  << "HEJ ME: " << ME(event).central
  << " = tree * virtual = " << ME.tree(event).central
  << " * " << ME.virtual_corrections(event).central
  << '\n';
}

```

After saving the above code to a file `matrix_element.cc`, it can be compiled into an executable `matrix_element` with a suitable compiler. For example, with `g++` this can be done with the command

```
g++ -o matrix_element matrix_element.cc -lhej -lfastjet
```

If HEJ or any of the required libraries (see section 2.2) was installed to a non-standard location, it may be necessary to explicitly specify the paths to the required header and library files. This can be done with the `HEJ-config` executable and similar programs for the other dependencies:

```
g++ $(fastjet-config --cxxflags) $(HEJ-config --cxxflags) -o ↵
  ↵ matrix_element matrix_element.cc $(HEJ-config --libs) ↵
  ↵ $(fastjet-config --libs)
```

8. Summary

We have presented the HEJ 2 event generator which may be used to generate Monte Carlo events for hadron colliders at leading-logarithmic (LL) accuracy in \hat{s}/p_{\perp}^2 , for multi-jet production and for Higgs boson production in association with jets. It takes as input fixed-order samples, currently at leading-order (LO), and maintains this accuracy to give combined LO+LL predictions. The addition of the LL terms has been seen to be particularly significant in regions of large invariant mass between jets (or equivalently large rapidity separation), which is particularly pertinent for gluon-fusion production of a Higgs boson in association with dijets.

The HEJ 2 code is publicly available from <https://hej.web.cern.ch>. In this contribution we have outlined all necessary details to download, install and run HEJ 2, including a full description of the possible settings which can be user-defined in an input file. We have further discussed how to create a general analysis of the events produced and how HEJ 2 can also be used as a standalone library. Therefore, this documentation will allow anyone to

generate their own predictions for arbitrary experimental setups at the LHC and future colliders.

Acknowledgements

The authors would like to thank Gavin Salam for discussions on the anti- k_t jet clustering algorithm [42].

This work has received funding from the European Union’s Horizon 2020 research and innovation programme as part of the Marie Skłodowska-Curie Innovative Training Network MCnetITN3 (grant agreement no. 722104), the Marie Skłodowska-Curie grant agreement No. 764850, SAGEX, and COST action CA16201: “Unraveling new physics at the LHC through the precision frontier”, and from the UK Science and Technology Facilities Council (STFC). JMS is supported by a Royal Society University Research Fellowship and the ERC Starting Grant 715049 “QCDforfuture”.

References

- [1] V. S. Fadin, E. A. Kuraev, L. N. Lipatov, On the Pomeranchuk singularity in asymptotically free theories, *Phys. Lett.* B60 (1975) 50–52.
- [2] E. A. Kuraev, L. N. Lipatov, V. S. Fadin, Multi - Reggeon processes in the Yang-Mills theory, *Sov. Phys. JETP* 44 (1976) 443–450.
- [3] E. A. Kuraev, L. N. Lipatov, V. S. Fadin, The Pomeranchuk singularity in nonabelian gauge theories, *Sov. Phys. JETP* 45 (1977) 199–204.
- [4] I. I. Balitsky, L. N. Lipatov, The Pomeranchuk singularity in quantum chromodynamics, *Sov. J. Nucl. Phys.* 28 (1978) 822–829.
- [5] J. R. Andersen, J. M. Smillie, Constructing All-Order Corrections to Multi-Jet Rates, *JHEP* 1001 (2010) 039. [arXiv:0908.2786](#), [doi:10.1007/JHEP01\(2010\)039](#).
- [6] J. R. Andersen, J. M. Smillie, The Factorisation of the t-channel Pole in Quark-Gluon Scattering, *Phys.Rev. D*81 (2010) 114021. [arXiv:0910.5113](#), [doi:10.1103/PhysRevD.81.114021](#).
- [7] J. R. Andersen, J. M. Smillie, Multiple Jets at the LHC with High Energy Jets, *JHEP* 1106 (2011) 010. [arXiv:1101.5394](#), [doi:10.1007/JHEP06\(2011\)010](#).

- [8] J. R. Andersen, T. Hapola, A. Maier, J. M. Smillie, Higgs Boson Plus Dijets: Higher Order Corrections, JHEP 09 (2017) 065. [arXiv:1706.01002](#), [doi:10.1007/JHEP09\(2017\)065](#).
- [9] J. R. Andersen, T. Hapola, J. M. Smillie, W Plus Multiple Jets at the LHC with High Energy Jets, JHEP 1209 (2012) 047. [arXiv:1206.6763](#), [doi:10.1007/JHEP09\(2012\)047](#).
- [10] J. R. Andersen, J. J. Medley, J. M. Smillie, Z/γ^* plus multiple hard jets in high energy collisions, JHEP 05 (2016) 136. [arXiv:1603.05460](#), [doi:10.1007/JHEP05\(2016\)136](#).
- [11] G. Aad, et al., Measurement of dijet production with a veto on additional central jet activity in pp collisions at $\sqrt{s} = 7$ TeV using the ATLAS detector, JHEP 1109 (2011) 053. [arXiv:1107.1641](#), [doi:10.1007/JHEP09\(2011\)053](#).
- [12] S. Chatrchyan, et al., Measurement of the inclusive production cross sections for forward jets and for dijet events with one forward and one central jet in pp collisions at $\sqrt{s} = 7$ TeV, JHEP 1206 (2012) 036. [arXiv:1202.0704](#), [doi:10.1007/JHEP06\(2012\)036](#).
- [13] S. Chatrchyan, et al., Ratios of dijet production cross sections as a function of the absolute difference in rapidity between jets in proton-proton collisions at $\sqrt{s} = 7$ TeV, Eur. Phys. J. C72 (2012) 2216. [arXiv:1204.0696](#), [doi:10.1140/epjc/s10052-012-2216-6](#).
- [14] V. M. Abazov, et al., Studies of W boson plus jets production in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, Phys.Rev. D88 (2013) 092001. [arXiv:1302.6508](#), [doi:10.1103/PhysRevD.88.092001](#).
- [15] G. Aad, et al., Measurements of jet vetoes and azimuthal decorrelations in dijet events produced in pp collisions at $\sqrt{s} = 7$ TeV using the ATLAS detector, Eur. Phys. J. C74 (11) (2014) 3117. [arXiv:1407.5756](#), [doi:10.1140/epjc/s10052-014-3117-7](#).
- [16] G. Aad, et al., Measurements of the W production cross sections in association with jets with the ATLAS detector, Eur. Phys. J. C75 (2) (2015) 82. [arXiv:1409.8639](#), [doi:10.1140/epjc/s10052-015-3262-7](#).

- [17] J. R. Andersen, T. Hapola, M. Heil, A. Maier, J. M. Smillie, Higgs-boson plus Dijets: Higher-Order Matching for High-Energy Predictions, JHEP 08 (2018) 090. [arXiv:1805.04446](#), [doi:10.1007/JHEP08\(2018\)090](#).
- [18] A. Buckley, J. Butterworth, L. Lönnblad, D. Grellscheid, H. Hoeth, J. Monk, H. Schulz, F. Siegert, Rivet user manual, Comput. Phys. Commun. 184 (2013) 2803–2819. [arXiv:1003.0694](#), [doi:10.1016/j.cpc.2013.05.021](#).
- [19] E. Boos, et al., Generic user process interface for event generators, in: Physics at TeV colliders. Proceedings, Euro Summer School, Les Houches, France, May 21-June 1, 2001, 2001. [arXiv:hep-ph/0109068](#).
URL <http://lss.fnal.gov/archive/preprint/fermilab-conf-01-496-t.shtml>
- [20] J. Alwall, et al., A Standard format for Les Houches event files, Comput. Phys. Commun. 176 (2007) 300–304. [arXiv:hep-ph/0609017](#), [doi:10.1016/j.cpc.2006.11.010](#).
- [21] M. Dobbs, J. B. Hansen, The HepMC C++ Monte Carlo event record for High Energy Physics, Comput. Phys. Commun. 134 (2001) 41–46. [doi:10.1016/S0010-4655\(00\)00189-2](#).
- [22] HepMC3 event record library, <https://hepmc.web.cern.ch/hepmc/>.
- [23] J. R. Andersen, L. Lönnblad, J. M. Smillie, A Parton Shower for High Energy Jets, JHEP 1107 (2011) 110. [arXiv:1104.1316](#), [doi:10.1007/JHEP07\(2011\)110](#).
- [24] J. R. Andersen, H. M. Brooks, L. Lönnblad, Merging High Energy with Soft and Collinear Logarithms using HEJ and PYTHIA, JHEP 09 (2018) 074. [arXiv:1712.00178](#), [doi:10.1007/JHEP09\(2018\)074](#).
- [25] git, <https://git-scm.com/>.
- [26] Docker, <https://www.docker.com/>.
- [27] K. Martin, B. Hoffman, An Open Source Approach to Developing Software in a Small Organization, IEEE Software 24 (1) (2007) 46–53. [doi:10.1109/MS.2007.5](#).

- [28] CMake, <https://cmake.org/>.
- [29] gcc, <https://gcc.gnu.org/>.
- [30] M. Cacciari, G. P. Salam, G. Soyez, FastJet User Manual, Eur. Phys. J. C72 (2012) 1896. arXiv:1111.6097, doi:10.1140/epjc/s10052-012-1896-2.
- [31] CLHEP, <https://proj-clhep.web.cern.ch/proj-clhep/>.
- [32] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, G. Watt, LHAPDF6: parton density access in the LHC precision era, Eur. Phys. J. C75 (2015) 132. arXiv:1412.7420, doi:10.1140/epjc/s10052-015-3318-8.
- [33] LHAPDF, <https://lhapdf.hepforge.org/>.
- [34] boost, <https://www.boost.org/>.
- [35] J. Beder, yaml-cpp, <https://github.com/jbeder/yaml-cpp/>.
- [36] S. Carrazza, R. K. Ellis, G. Zanderighi, QCDLoop: a comprehensive framework for one-loop scalar integrals, Comput. Phys. Commun. 209 (2016) 134–143. arXiv:1605.03181, doi:10.1016/j.cpc.2016.07.033.
- [37] QCDLoop, <https://github.com/scarrazza/qcdloop>.
- [38] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, 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. arXiv:1405.0301, doi:10.1007/JHEP07(2014)079.
- [39] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, J. Winter, Event generation with SHERPA 1.1, JHEP 02 (2009) 007. arXiv:0811.4622, doi:10.1088/1126-6708/2009/02/007.
- [40] gzip, <https://www.gnu.org/software/gzip/>.
- [41] YAML, <https://yaml.org>.

- [42] M. Cacciari, G. P. Salam, G. Soyez, The anti- k_t jet clustering algorithm, JHEP 04 (2008) 063. [arXiv:0802.1189](#), [doi:10.1088/1126-6708/2008/04/063](#).
- [43] G. K. Savvidy, N. G. Ter-Arutunian Savvidy, ON THE MONTE CARLO SIMULATION OF PHYSICAL SYSTEMS, J. Comput. Phys. 97 (1991) 566. [doi:10.1016/0021-9991\(91\)90015-D](#).
- [44] K. G. Savvidy, The MIXMAX random number generator, Comput. Phys. Commun. 196 (2015) 161–165. [arXiv:1403.5355](#), [doi:10.1016/j.cpc.2015.06.003](#).
- [45] M. Lüscher, A Portable high quality random number generator for lattice field theory simulations, Comput. Phys. Commun. 79 (1994) 100–110. [arXiv:hep-lat/9309020](#), [doi:10.1016/0010-4655\(94\)90232-1](#).