

Ntuples for NLO events at hadron colliders

Z. Bern, L. J. Dixon, F. Febres Cordero, S. Hoeche, H. Ita, David A Kosower, D. Maitre

▶ **To cite this version:**

Z. Bern, L. J. Dixon, F. Febres Cordero, S. Hoeche, H. Ita, et al.. Ntuples for NLO events at hadron colliders. Computer Physics Communications, 2014, 185 (5), pp.1443-1460. 10.1016/j.cpc.2014.01.011 . cea-01037882

HAL Id: cea-01037882

https://cea.hal.science/cea-01037882

Submitted on 21 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:1310.7439v1 [hep-ph] 28 Oct 2013

UCLA/13/TEP/108 SLAC–PUB–15739 SB/F/422-13
IPhT–T13/228 IPPP-13-86 CERN–PH–TH/2013-243

Ntuples for NLO Events at Hadron Colliders

Z. Bern[a], L. J. Dixon[b], F. Febres Cordero[c], S. Höche[b], H. Ita[d], D. A. Kosower[e], D. Maître[f,g,*]

[a] *Department of Physics and Astronomy, UCLA, Los Angeles, CA 90095-1547, USA*
[b] *SLAC National Accelerator Laboratory, Stanford University, Stanford, CA 94309, USA*
[c] *Departamento de Física, Universidad Simón Bolívar, Caracas 1080A, Venezuela*
[d] *Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, D-79104 Freiburg, Germany*
[e] *Institut de Physique Théorique, CEA–Saclay, F–91191 Gif-sur-Yvette cedex, France*
[f] *Theory Division, Physics Department, CERN, CH–1211 Geneva 23, Switzerland*
[g] *Department of Physics, University of Durham, Durham DH1 3LE, UK*

Abstract

We present an event-file format for the dissemination of next-to-leading-order (NLO) predictions for QCD processes at hadron colliders. The files contain all information required to compute generic jet-based infrared-safe observables at fixed order (without showering or hadronization), and to recompute observables with different factorization and renormalization scales. The files also make it possible to evaluate cross sections and distributions with different parton distribution functions. This in turn makes it possible to estimate uncertainties in NLO predictions of a wide variety of observables without recomputing the short-distance matrix elements. The event files allow a user to choose among a wide range of commonly-used jet algorithms and jet-size parameters.

We provide event files for a W or Z boson accompanied by up to four jets, and for pure-jet events with up to four jets. The files are for the Large Hadron Collider with a center of mass energy of 7 or 8 TeV.

*Corresponding author. *E-mail address:* daniel.maitre@durham.ac.uk

A C++ library along with a Python interface for handling these files are also provided and described in this article. The library allows a user to read the event files and recompute observables transparently for different pdf sets and factorization and renormalization scales.

Keywords: QCD, vector boson, jets, LHC, Tevatron.

PROGRAM SUMMARY

Manuscript Title:
Authors:
Program Title:
Journal Reference:
Catalogue identifier:
Licensing provisions:
Programming language: C++ with Python interface
Computer:
Operating system: Linux, MacOS
RAM: varying bytes
Keywords: QCD, NLO, vector boson, jets.
Classification:
External routines/libraries: ROOT, LHAPDF
Nature of problem: NLO QCD predictions for vector boson + jets and jet processes for generic observables.

Solution method: Event files

1. Introduction

With last year's discovery of a Higgs-like boson [1, 2] at the Large Hadron Collider (LHC), the ATLAS and CMS experiments at CERN have rounded out our knowledge of the particle content of the Standard Model. Direct

searches for new physics beyond it, and indirect searches via precision measurements of the properties of the Higgs-like boson and of the top quark, remain as challenges for ongoing research. Both of these avenues beyond the Standard Model require extensive calculations in QCD to next-to-leading order (NLO), to high jet multiplicity. NLO is the first order in perturbation theory to provide a quantitatively reliable estimate of backgrounds due to Standard-Model processes [3, 4].

NLO calculations require the computation of both virtual and real-emission corrections to processes with high jet multiplicity. The former require the evaluation of one-loop corrections to the basic tree-level process, and the latter require the integration of matrix elements with an additional emitted parton over the phase space for the additional emission. Both parts of the correction are computationally intensive. In order to obtain sufficiently small statistical uncertainties, one must typically evaluate the matrix elements at millions of different kinematic points when performing phase-space integrals by Monte Carlo sampling techniques.

The end results of the calculations are total cross sections or differential distributions for various observables. In developing an analysis[1], one may wish to evaluate these quantities with different choices of experimental cuts. Different cuts may also be required for comparison with different experimental analyses. The estimation of uncertainties due to variation of the unphysical renormalization and factorization scales, and due to our imperfect knowledge of the proton's parton distribution functions (pdfs), also require that we we recompute the same differential distributions many times. The latter alone can require dozens or grosses of evaluations. A naive rerunning of the entire calculation would force the computationally expensive short-distance matrix elements to be recomputed a comparable number of times.

The actual evaluation of multiple differential distributions, given a list of kinematic points and matrix-element weights, is however computationally relatively cheap. It is therefore very desirable to amortize the expensive task of matrix-element computation over many evaluations of differential distributions. We can do this by storing each of the phase-space points, along with the matrix elements and other information, in data files. In order to compress these sizeable volumes of data, we store them as ROOT n-tuple files, or simply 'n-tuple files' for short. We produce n-tuple files using the BLACKHAT

[1]We use this term in the sense used by experimenters when analyzing data.

one-loop library [5] in conjuction with SHERPA [6, 7, 8, 9, 10]. In this paper we describe in more detail an implementation of this strategy for a class of interesting multi-jet final states at the Large Hadron Collider (LHC). We document n-tuple files for processes producing a single electroweak vector boson ($W^{\pm}$ or Z) in association with 1, 2, 3, or 4 jets, and files for pure-QCD processes producing 2, 3, or 4 jets. In addition, we supply a C++ library that provides an interface to the files. The n-tuple files also allow other researchers, and especially experimenters, to perform their own NLO analyses, using refined cuts, without the computational expense and management complexity of running high-multiplicity codes. The computations recorded in the n-tuple files do not include showering or hadronization, and the events are *not* a suitable starting point for matched showers with existing implementations. In contrast to files containing showered events, the n-tuple files documented here have a relatively small number of parton momenta recorded per event, which keeps file sizes manageable. A reweighting technique for estimating scale and pdf uncertainties has also been applied to particle-level event samples that have been generated by merging NLO calculations with a parton shower [11].

The paper is organized as follows. In section 2, we discuss the organization of an NLO calculation. In section 3, we present the content of event files. In section 4 we explain how to compute differential cross sections using the event files. In section 5, we discuss additional details needed to change scales or parton distribution functions from those in the original calculation. In section 6, we present a software library for reading event files. We summarize in section 7.

2. NLO Calculations

We wish to store and reuse the matrix-element information computed during an NLO calculation. In order to explain what information we need to record, we first review the components of an NLO calculation. We further refine the information needed in section 5.

Let us consider a process that produces n final-state objects (jets or electroweak bosons). At leading order (LO) in QCD, the prediction for associated observables would be given in terms of a tree-level matrix element for a process scattering two colored partons into n final-state partons or electroweak bosons. Schematically, at this order we can write the differential cross-section

in an observable $\mathcal{O}$ as follows,

$$\begin{aligned}\frac{d\sigma^{\rm LO}}{dv} &= \int d\hat{\sigma}_n\, \delta_v \\ &= \int dx_1 dx_2 \int d\mathrm{LIPS}_n\; f_1(x_1) f_2(x_2)\, \hat{\sigma}_n\, \delta\big(v - \mathcal{O}(\{k\}_n)\big)\,,\end{aligned} \tag{1}$$

where $f_1(x_1)$ and $f_2(x_2)$ are the pdfs for the partons inside the two incoming protons, $d\mathrm{LIPS}_n$ is the n-particle Lorentz-invariant phase-space measure for momenta $\{k\}_n$, and $\hat{\sigma}_n$ is the short-distance $2 \to n$ squared matrix element. We sum over all allowed parton species implicitly. The pdfs $f_{1,2}$ also depend on a factorization scale $\mu_{\rm F}$, and the parton-level squared matrix element $\hat{\sigma}_n$ depends on a renormalization scale $\mu_{\rm R}$. These two scales will in general depend on the phase-space point $\{k\}_n$. However, at LO $\hat{\sigma}_n$ depends on the renormalization scale only through its (homogeneous) dependence on the strong coupling $\alpha_s(\mu_{\rm R})$.

In order for eq. (1) to produce a meaningful answer, the phase space must be cut off in regions where the squared matrix elements are singular. We are primarily interested in jet cross sections, and so we shall use a jet algorithm to impose such cuts. We leave them implicit in eq. (1).

The integral will typically be performed by Monte-Carlo integration, using a random sample of phase-space points. More sophisticated approaches will adapt the sample to the squared matrix element. The evaluation will generate a list of phase-space points and momentum fractions, along with the associated integration weight. At each phase-space point, we evaluate only a single subprocess, fixing the incoming and outgoing parton types and flavors. In order to recompute any observable in an LO calculation, or compute a new one, it then suffices to save the generated phase-space configurations, along with each configuration's integration weight. Each configuration can be saved by saving the four-momenta of each final-state parton it contains. In order to estimate the scale sensitivity of observables, we must recompute them with different choices of renormalization scale. For this purpose, it suffices to save the choice of $\mu_{\rm R}$ for each configuration, as the weight is homogeneous in α_s with a fixed power. In order to estimate the uncertainty in an observable due to imprecise knowledge of the pdfs, we must recompute them with different pdfs drawn from an error ensemble. For this purpose, we must also save the initial-state momentum fractions and choice of $\mu_{\rm F}$ for each configuration. It is also convenient to save separately the matrix-element weight multiplied

by the phase-space volume element; this is the quantity that would yield the integration weight upon multiplication by the pdfs.

The NLO prediction for the same process is more intricate. We can decompose the computation of an observable into a Born-level contribution, along with virtual and real-emission corrections,

$$\frac{d\sigma^{\mathrm{NLO}}}{dv} = \int \left(d\hat{\sigma}_n^{\mathrm{Born}} + d\hat{\sigma}_n^{\mathrm{virt}}\right) \delta_v + \int d\hat{\sigma}_{n+1}^{\mathrm{real}}\, \delta_v \,, \tag{2}$$

where the subscripts indicate the number of particles in the final-state phase space. The virtual and the real-emission corrections are separately infrared divergent. The virtual corrections, given by the interference of one-loop and tree matrix elements, have explicit divergences arising from the integration over the loop momenta. These are usually regulated using dimensional regularization ($D = 4 - 2\epsilon$), and appear as double and single poles in ϵ. The infrared divergences in the real-emission contributions arise from regions of $(n+1)$-body phase space in which two massless partons become collinear or a gluon becomes soft. We can think of the $(n+1)$-body phase space factorizing into a phase space for n final-state objects and an unresolved phase space over soft or collinear partons. Were we to integrate these contributions over a dimensionally-regulated unresolved phase space, the resulting poles in ϵ would cancel against those in the virtual contributions for infrared-safe observables, allowing us to obtain a finite result in the $\epsilon \to 0$ limit.

However, the integration over the dimensionally-regulated real-emission phase space is intractable analytically in the presence of kinematical cuts. Accordingly, we must integrate numerically; but it is hard to extract singularities, and more importantly, the underlying finite term, by numerical integration in D dimensions. Instead, we seek to separate the real-emission contributions into two parts: a simple part, containing all singularities, to be integrated analytically in D dimensions; and a remainder, whose numerical integral is finite over the complete phase space. The universality of the singular limits of the matrix elements makes this possible. The most common approach, which we use as well, is to add and subtract an approximation to the real-emission squared matrix element that captures all of its divergent collinear and soft limits, and yet is simple enough to be integrated analytically over the unresolved phase-space. This approach started with applications to specific processes [12, 13, 14, 15] and was later generalized by Catani and Seymour [16, 17] to a process-independent method. We use an implementation of the Catani–Seymour dipole method within the AMEGIC++ [8, 9] and

Comix [10] tree-level matrix-element package (both parts of SHERPA [6, 7], with Comix used for the highest-multiplicity processes). Other subtraction methods have also been applied, especially in NNLO calculations [15, 18], and related methods are under development [19]. Some of these have also been automated [20]. With a subtraction term, we obtain the following formula for a differential cross section computed to NLO,

$$\frac{d\sigma^{\text{NLO}}}{dv} = \int d\hat{\sigma}_n^{\text{Born}}\,\delta_v + \int \left(d\hat{\sigma}_n^{\text{virt}} + d\hat{\sigma}_n^{\text{int}}\right)\delta_v + \int \left(d\hat{\sigma}_{n+1}^{\text{real}} - d\hat{\sigma}_{n+1}^{\text{sub}}\right)\delta_v\,, \quad (3)$$

where $\hat{\sigma}_n^{\text{int}}$ is the integral of $\hat{\sigma}_{n+1}^{\text{sub}}$ over the unresolved phase space. Each integral in eq. (3) is now finite as $\epsilon \to 0$. (The observable must be infrared and collinear safe, that is $\mathcal{O}_{n+1}$ must approach $\mathcal{O}_n$ in every singular limit, for this to be true.) As the divergent terms in $\hat{\sigma}_n^{\text{virt}}$ cancel those in $\hat{\sigma}_n^{\text{int}}$, we can drop them all, retaining only the finite contributions,

$$\frac{d\sigma^{\text{NLO}}}{dv} = \int d\hat{\sigma}_n^{\text{Born}}\,\delta_v + \int d\hat{\sigma}_n^{\text{fin. virt}}\,\delta_v + \int d\hat{\sigma}_n^{\text{fin. int}}\,\delta_v + \int \left(d\hat{\sigma}_{n+1}^{\text{real}} - d\hat{\sigma}_{n+1}^{\text{sub}}\right)\delta_v\,. \quad (4)$$

As in an LO computation, the integrals in eq. (4) are most easily computed by Monte-Carlo sampling. (In our setup, using SHERPA [6, 7], the code first adapts an integration grid to the integrand, and many independent integrations are done using these grids.) We generate events separately for each of the four types of contributions, over n-particle phase space for the Born (B), virtual (V), and integrated-subtraction (I) contributions, and over $(n+1)$-particle phase space for the subtracted real-emission (R) contributions. At lower multiplicities, this split-up is sufficient. At higher multiplicities, however, a further subdivision of contributions is desirable. Different parts of each contribution have very different computational complexities and magnitudes; for example, the subleading-color contributions are more costly by at least an order of magnitude with three or more colored partons, yet give only a small contribution to cross sections. A subdivision according to initial-state parton types is also helpful in this regard. For each part, we adapt the number of phase-space points according to its relative contribution to cross sections. The subdivision also allows us to add more statistics to selected parts without having to perform additional computations for parts whose statistics are adequate. In order to keep the subdivision flexible, and make code using them more robust to future evolution, we simply label the different parts sequentially within each of the four types, for example (R001, R002, R003).

As in an LO calculation, we must save all particle four-momenta in each phase-space configuration in order to be able to recompute (or compute afresh) observables. We must also save the renormalization scale $\mu_{\rm R}$ in order to be able to vary it, and the initial-state parton momentum fractions $x_{1,2}$ and factorization scale $\mu_{\rm F}$ in order to be able to vary the pdfs and thereby estimate pdf uncertainties. We must also save each configuration's integration weight; alongside it, we again save the matrix-element weight as well.

Unlike for an LO calculation, however, these elements do not suffice in order to be able to vary scales or pdfs. They do suffice for the Born and subtracted real-emission contributions, as both the strong coupling and the pdfs appear as simple overall factors in the integrand. However, the virtual contribution contains additional dependence on $\mu_{\rm R}$ arising from the one-loop amplitude; and the integrated-subtraction contribution contains both additional dependence on $\mu_{\rm R}$ and $\mu_{\rm F}$, and different dependence on pdfs for different parton species. We must save more detailed information for these contributions, as we discuss in greater depth in section 5.

The n-tuple collections we describe are focused on the computation of jet cross sections. The framework we describe is applicable to other infrared-safe observables, but the specific files we have generated and are documenting herein rely on a jet algorithm, and can only be used to compute observables which also are defined using one of a selected set of jet algorithms. Modern jet algorithms are characterized by a clustering (or seedless cone) algorithm; a minimum jet $p_{\rm T}$; and a jet size R. The event samples described here allow for any one of the anti-$k_{\rm T}$ [21], $k_{\rm T}$[22], and SISCone [23] algorithms, in each case with jet sizes R chosen from the set $\{0.4, 0.5, 0.6, 0.7\}$. The SISCone merging fraction parameter is taken to be $f = 0.75$. We use the FastJet library [24] to implement these jet algorithms. The minimum $p_{\rm T}$ is specified in each file, but is typically 20 or 25 GeV. Only observables imposing this cut, or a tighter one, are allowed for use with n-tuple files. (If the minimum $p_{\rm T}$ cut is too tight, the number of events in the sample passing the cut may be too small for adequate statistical accuracy.) These restrictions on jet algorithm and minimum jet $p_{\rm T}$ could of course be relaxed by generating new event samples within our framework.

The main practical tradeoff in this approach is the sizable storage requirement for the n-tuple files. For example, the provided 7 TeV n-tuple files for W^++3 jets all together require 50 GB of storage, while the corresponding files for $W^+ + 4$ jets would require 375 GB. Of course, the storage requirements depend greatly on the process and on the desired statistics. The files are

quite voluminous, especially at higher multiplicities; the compression offered by ROOT yields a significant reduction compared to a naive binary format both in disk-space usage and in transmission times.

3. The *n*-Tuple Files

As described in the previous section, we save collections of phase-space configurations along with additional information. Each phase-space configuration represents one event at LO. The same is true for three of the four types of contributions at NLO: the Born (B), the virtual (V), and the integrated-subtraction (I) contributions. In the fourth type of contribution, the subtracted real-emission (R) one, each event in general will contain multiple configurations: one corresponding to an emission, with the rest of the configurations corresponding to subtractions; we will call each of the configurations 'entries'. Each ROOT n-tuple file contains a collection of events. Each (numbered) part within each of the contribution types (B,V,I,R at NLO) may in general be split up into a number of different files for convenience. The event files we describe in this section were produced using BLACKHAT [5] and SHERPA [6, 7, 8, 9, 10], according to the setups described in refs. [25, 26, 27, 28, 29, 30, 31]. Each file is a ROOT file, containing a set of events, along with information about the file content, and sample histograms which can be used to cross-check analyses using the file. (For a review of ROOT and its file formats, the reader may consult ref. [32].)

Information for the entries in each file is stored in a ROOT tree called `BHSntuples`. The branches of this tree are listed in Table 1. We have chosen to restrict the numerical precision for the momenta to single precision in order to limit disk space usage. This means that the weights cannot be recomputed exactly from the momenta for each entry; but there is of course no need to do so. For this reason, because each individual weight is in any case not necessarily accurate even to single precision (see refs. [5, 26, 27, 33, 34] for examples of numerical uncertainty distributions of matrix elements), and because the weights include phase-space and Jacobian factors arising from SHERPA's integration grids, they are not suitable as reference points for verifying or comparing matrix elements. For such purposes, reference points and matrix-element values quoted in refs. [26, 27] should be used.

Branch name	Type	Notes
id	I	ID of the event. Real-emission entries and their associated counterterms share the same ID.
nparticle	I	number of particles in the final state
px, py, pz	F[nparticle]	array of p_x, p_y, p_z respectively, for final-state particles
E	F[nparticle]	array of energies E for final-state particles
kf	I[nparticle]	PDG codes of the final-state particles
weight	D	total weight of the entry
weight2	D	secondary or correlated weight used to compute the subtracted real-emission's statistical errors. Identical to weight for the B, V, and I contributions; the normalization differs for the R contribution
me_wgt	D	coefficient of the product of parton-distribution functions in weight. For the B, V, and R contributions, this is the squared matrix element multiplied by the phase-space measure and the Jacobian from SHERPA's phase-space mapping
me_wgt2	D	coefficient of the product of parton-distribution functions in weight2
x1, x2	D	fraction of hadron momentum carried by the first and second incoming partons, respectively
x1p,x2p	D	secondary momentum fractions $x'_{1,2}$ used in integrated subtraction entries [9]
id1, id2	I	PDG codes of the first and second incoming partons respectively
fac_scale	D	factorization scale used ($\mu_{\mathrm{F},0}$)
ren_scale	D	renormalization scale used ($\mu_{\mathrm{R},0}$)
nuwgt	I	number of additional weights
usr_wgts	D[nuwgt]	additional weights needed to recompute the entry's weight for a different scale or pdf choices
part	C	type of contribution: B, V, I, or R
alphas_power	S	power of the coupling
alphas	D	α_s value used for this entry

Table 1: Branches in a BlackHat+SHERPA Root file. The type of the data entry follows Root's notation. In the second column, "D" stands for "double-precision floating point number", "F" for "single-precision floating point number", "I" for "integer", "S" for "short integer", and "C" for "character array". Square brackets denote an array.

Process	n-tuple file sets	References
$W^{\pm}(\to e^{\pm}\overset{(-)}{\nu}) + 0, 1, 2$ jets	B001, I001, R001, V001	[25, 26]
$W^{\pm}(\to e^{\pm}\overset{(-)}{\nu}) + 3$ jets	B001, I001, R001, V001–V002	[25, 26]
$W^{-}(\to e^{-}\bar{\nu}) + 4$ jets	B001, I001, R001, V001	[28, 35]
$W^{+}(\to e^{+}\nu) + 4$ jets	B001, I001, R001–R005, V001	[28, 35]
$Z(\to e^{+}e^{-}) + 0, 1, 2$ jets	B001, I001, R001, V001	[27]
$Z(\to e^{+}e^{-}) + 3$ jets	B001, I001, R001, V001–V002	[27]
$Z(\to e^{+}e^{-}) + 4$ jets	B001, I001–I003, R001–R006, V001–V006	[29]
n jets $(n = 1, 2, 3, 4)$	B001, I001, R001, V001	[31]

Table 2: Available processes at NLO, and their decomposition into n-tuple file sets.

3.1. Coordinates and units

We take the coordinate axes such that the beams are directed along the z axis, with the initial-state parton with momentum fraction `x1` and Particle Data Group (PDG) code `id1` moving in the positive z direction. The transverse directions are labeled by x and y, with (x, y, z) forming a right-handed coordinate system. The energies of particles are denoted by E and the spatial momenta by (p_x, p_y, p_z). We use natural units ($\hbar = c = 1$) and GeV for energies, masses and momenta. Weights are normalized to yield cross sections in picobarns.

3.2. n-Tuple Collections

In this article, we document n-tuple files for the LHC processes listed in Table 2, which we have made available as described in section 3.3. The corresponding references for each class of processes are given in the last column.

As explained in section 2, the events for each process are split up into different types according to the different terms in eq. (4). They are further subdivided into different parts, organized in subdirectories, with each part in turn split into multiple files. We must add together contributions from all parts within all types of contributions to obtain the complete NLO cross sections or distributions, as we describe in greater detail in sections 4 and 5. The different parts are independent, analogous to different subprocesses, so their statistical integration errors can be added in quadrature. We show the list of the different parts for each process in Table 2.

The n-tuples were generated using the MSTW2008 [36] NLO parton distribution functions, and a five-flavor running $\alpha_s(\mu)$ where the value of $\alpha_s(M_Z)$

is specified by the parton distribution set. The renormalization and factorization scales are,

$$\mu_{\mathrm{F},0} = \mu_{\mathrm{R},0} = \tfrac{1}{2}\hat{H}'_{\mathrm{T}} \equiv \frac{1}{2}\Big(\sum_j p_{\mathrm{T}}^j + E_{\mathrm{T}}^V\Big)\,. \tag{5}$$

In this equation, originally given in ref. [28], the sum runs over all partons in the final state, with p_{T}^j the transverse momentum of the jth parton, and E_{T}^V is the transverse energy of the vector boson,

$$E_{\mathrm{T}}^V = \sqrt{m_V^2 + (p_{\mathrm{T}}^V)^2}\,. \tag{6}$$

(For pure-jet processes, the E_{T}^V term is of course omitted.) As we explain in more detail in section 5, the n-tuples can be used to generate predictions for other parton distributions, and other choices of renormalization and factorization scales[2].

The `getInfo` program (see section 6.4) provides useful information about the process, input parameters, cuts and jet algorithms for a specific n-tuple file.

For processes containing a single electroweak boson, we fold in its decay into a massless lepton pair. We label the pair by $e^+\nu$ for a W^+ boson, $e^-\bar{\nu}$ for a W^- boson, and e^+e^- for a Z boson. Results for the corresponding muon channels are identical in the massless-lepton approximation. Off-shell effects are taken into account by distributing the lepton-pair invariant mass in a relativistic Breit-Wigner profile with width Γ_V around the mass m_V of the electroweak vector boson V. In the case of the Z boson (that is, e^+e^- final states), we include virtual photon γ^* exchange as well. For all processes we neglect the contribution of a massive top quark in the quark loops.

3.2.1. W + n-jet processes

For W + n-jet processes, we take the CKM matrix to be diagonal. We provide separate sets of files for W^- + n-jet and W^+ + n-jet production. The files we provide yield results for the W + 4-jet process in the leading-color

[2]Cross sections and distributions computed using the n-tuples documented here may not match the results quoted in the references in the last column of Table 2 exactly, because the earlier results used six-flavor running (above the top-quark mass) and a different definition of the leading-color approximation.

approximation for the virtual contributions. The n-tuple collections impose no cuts on the generated momenta of the final-state electron or neutrino, no isolation cuts between the leptons and the jets, and no cuts on the jet rapidities. We impose cuts only on the transverse momenta of the jets,

$$\begin{aligned} p_{\mathrm{T}}^{\mathrm{jet}} &> 25 \text{ GeV}\,, \qquad 7 \text{ TeV } n\text{-tuples for } W + 1,2,3,4 \text{ jets}\,, \\ p_{\mathrm{T}}^{\mathrm{jet}} &> 20 \text{ GeV}\,, \qquad 8 \text{ TeV } n\text{-tuples for } W + 1,2,3 \text{ jets}\,. \end{aligned} \tag{7}$$

As noted in section 2, the allowed jet algorithms are the anti-k_{T}, k_{T}, and SISCone ones, in each case with jet sizes R chosen from the set $\{0.4, 0.5, 0.6, 0.7\}$. In the SISCone case the merging parameter is taken to be $f = 0.75$.

3.2.2. $Z, \gamma^* + n$-jet processes

In contrast to the $W + n$-jet processes, for Z processes we also impose a cut on the invariant mass of the e^+e^- pair in a window around the Z mass, $60 < M_{e^+e^-} < 120$ GeV, in order to suppress the contribution of the virtual photon. Otherwise, we again impose only a cut on the jet transverse momenta,

$$\begin{aligned} p_{\mathrm{T}}^{\mathrm{jet}} &> 25 \text{ GeV}\,, \qquad 7 \text{ TeV } n\text{-tuples for } Z, \gamma^* + 1,2 \text{ jets}\,, \\ p_{\mathrm{T}}^{\mathrm{jet}} &> 20 \text{ GeV}\,, \qquad 7 \text{ TeV } n\text{-tuples for } Z, \gamma^* + 3,4 \text{ jets}\,, \\ p_{\mathrm{T}}^{\mathrm{jet}} &> 20 \text{ GeV}\,, \qquad 8 \text{ TeV } n\text{-tuples for } Z, \gamma^* + 1,2,3 \text{ jets}\,. \end{aligned} \tag{8}$$

The allowed jet algorithms are the same as given above for $W + n$-jet processes.

The virtual contributions for the Z + 4-jet process are again computed in the leading-color approximation described in ref. [29]. Based on studies of lower multiplicities [26, 27] and also in the case of $W + 4$-jets [35], we expect these neglected pieces to be on the order of 3% of the total cross section. As in ref. [29], we also drop the axial- and vector-coupling loop contributions, along with the effects of top quarks in the loop. We expect these neglected pieces to contribute under 1%. If we neglect the small effect from the muon mass, these n-tuples are just as valid for the Z boson decaying into a pair of muons.

3.2.3. Pure-jet processes

For pure-jet processes, we impose the cut,

$$p_{\mathrm{T}}^{\mathrm{jet}} > 40 \text{ GeV}\,. \tag{9}$$

`xsection`	total cross section	1 bin of width 1
`h_pt_jN`	transverse momentum of jet N	50 bins: $[0, 1000]$
`h_eta_jN`	pseudo-rapidity of jet N	22 bins: $[-4.4, 4.4]$
`h_pt_NP`	trans. momentum of the non-parton NP	200 bins $[0, 2000]$

Table 3: List of the histograms provided to test validate the analysis of an n-tuple file. A non-parton can be an electron, positron, photon, neutrino or anti-neutrino

We treat the five light-flavor quarks as massless, and drop top-quark loops, following ref. [31] (this has a sub-percent effect). We include the full color dependence of all contributions.

3.3. Location

An up-to-date list of available processes is maintained at
`http://blackhat.hepforge.org/trac/wiki/Availability`

The locations from which the n-tuple files may be obtained are given in `http://blackhat.hepforge.org/trac/wiki/Location`. They are currently available at CERN on CASTOR and on the LHC Grid. At each location, the files are in

⟨base⟩`/BHSNtuples/PROCESS/ENERGY/PART`

where `ENERGY` is either `7TeV` or `8TeV`, and `PART` is one of `B`, `V`, `I`, or `R`.

3.4. Checks

All n-tuple files contain several histograms which can be used to check their consistency and the implementation of the program reading them. The histograms are listed in Table 3.

Each of the histograms listed in Table 3 comes in two copies, one in which the distributions are computed using the original weights, without change of renormalization or factorization scale, and a second in which both $\mu_{\rm R}$ and $\mu_{\rm F}$ have been changed to $H_{\rm T}^{\rm all}$, defined to be the scalar sum of the transverse momentum of all particles (including the neutrino) in the event[3]. The histogram names have `_Orig` and `_HTallp` respectively appended to them.

The program `getInfo` (see section 6.4) can be used to obtain the cuts and jet algorithm used for the histograms.

[3]This scale is not appropriate for observables sensitive to the polarization of the vector boson, as explained in ref. [37].

4. Using the n-tuple files

One can analyze the entries in the n-tuple files to generate histograms with a wide variety of experimental cuts, so long as all of the following conditions are met:

- One of the jet algorithms used in generating the n-tuple file is applied. (The list of compatible jet algorithms and parameters are stored in each file but are the same for all files for the same process and center-of-mass energy. See section 6.4 for a description of this information and how to extract it from an n-tuple file.)

- The number of jets passing the algorithm and associated cuts is at least n for an n-jet process.

- All cuts and observables are defined in terms of jets passing the cuts and not in terms of partons. (It is perfectly acceptable, and indeed often desirable, to define the renormalization and/or factorization scales using parton momenta so long as the definition is infrared- and collinear-safe.)

As explained above, each type of contribution ('B', 'R', ...) to a process is split up into a number of parts (R001, R002, ...). The number of parts will depend on the process and type of contribution. An analysis should in general sum over all available parts and all types of contributions. (Once one has verified that a given part yields a negligible contribution to the observables of interest compared to the desired accuracy, one may choose to omit it.) The events for each part may be split up into a number of files; each file will contain an independent sample of events. An analysis should sum over as many files as required to obtain the desired statistical accuracy for the given part and type of contribution, but need not sum over all available files. Summing over more files for a given part will increase the statistical accuracy of estimates for that part.

To compute the contribution of a given type t and part p to an observable, one must sum the weight times the observable's value over all entries in all chosen files, and normalize by the total number of entries in these files,

$$\langle \mathcal{O} \rangle^{(t,p)} = \frac{1}{N_{t,p}} \sum_{e=1}^{N_{t,p}} w_{t,p,e} \mathcal{O}_{t,p,e} \,, \tag{10}$$

where $N_{t,p}$ is the total number of entries (across all files) for the given part of the given type, $w_{t,p,e}$ the weight of the entry (given by the `weight` branch of the ROOT file), and $\mathcal{O}_{t,p,e}$ the value of the observable evaluated on the entry. The value of an observable is then giving by summing over all parts of all types of contributions,

$$\langle \mathcal{O} \rangle = \sum_{t \in T, p \in P_t} \langle \mathcal{O} \rangle^{(t,p)} \,. \tag{11}$$

where T is the set of types, and P_t the set of parts in type t. As a simple example, if we choose $\mathcal{O}_{t,p,e} = \Theta_{\text{cuts}}$ (where $\Theta_{\text{cuts}} = 1$ if the entry is allowed by our chosen jet algorithm and all other applied analysis cuts, and 0 otherwise), we obtain the total cross section. In eq. (10), all types of contributions are computed in a similar manner. The weight itself can be recomputed to alter the factorization scale, the renormalization scale, or the pdf, as we shall explain in section 5. The overwhelming fraction of the computation effort in an analysis is in the execution of eq. (10). One can combine different batches or sets of files part-by-part by weighting with the total number of entries in each,

$$\langle \mathcal{O} \rangle_{1+2}^{(t,p)} = \frac{1}{N_{t,p,1+2}} \big(N_{t,p,1} \langle \mathcal{O} \rangle_{1}^{(t,p)} + N_{t,p,2} \langle \mathcal{O} \rangle_{2}^{(t,p)} \big) \tag{12}$$

(where $N_{t,p,1+2} = N_{t,p,1} + N_{t,p,2}$), and then re-evaluating eq. (11). One would of course typically compute several observables in parallel during a single pass over the selected files.

In order to compute the error estimate for the observable, we must however treat the subtracted real-emission contributions (R) specially. For each part in each of the other types of contributions (B, V, and I), the error estimate is the standard one for Monte-Carlo integration,

$$\varepsilon_{\mathcal{O}}^{(t,p)} = \frac{1}{\sqrt{N_{t,p}(N_{t,p}-1)}} \left[\sum_{e=1}^{N_{t,p}} \big(w_{t,p,e} \mathcal{O}_{t,p,e} \big)^2 - \frac{1}{N_{t,p}} \left(\sum_{e=1}^{N_{t,p}} w_{t,p,e} \mathcal{O}_{t,p,e} \right)^2 \right]^{1/2} . \tag{13}$$

(We denote the error estimate by ε in order to avoid confusion with the cross section σ.) The R contributions are special, because each event in general contains more than one entry. (In the ROOT file, all entries within a given event share the same `id`.) The entries (or phase-space configurations) are of two different kinds, a real-emission configuration and counter-configurations. The latter correspond to the subtraction term that regulates the squared matrix element in the soft and collinear limits, where the unregulated matrix

element diverges. The real-emission configurations and subtraction counter-configurations for a given event are strongly anticorrelated. This means that simply adding the weights independently as in eq. (13) will grossly overestimate the statistical error. (The estimated error would typically be of order the central value. Using these weights will, however, yield the correct central value in eq. (10).) The anticorrelation must be taken into account properly, which we can do using the following formula,

$$\varepsilon_{\mathcal{O}}^{(\mathrm{R},p)} = \frac{1}{\sqrt{\widetilde{N}_{\mathrm{R},p}(\widetilde{N}_{\mathrm{R},p}-1)}} \Bigg[\sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \Bigg(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \mathcal{O}_{\mathrm{R},p,e,j} \Bigg)^2 - \frac{1}{\widetilde{N}_{\mathrm{R},p}} \Bigg(\sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \mathcal{O}_{\mathrm{R},p,e,j} \Bigg)^2 \Bigg]^{1/2}, \tag{14}$$

where $\widetilde{N}_{\mathrm{R},p}$ is the total number of *events* (or equivalently, different (filename,ID) pairs with the ID given by the `id` branch) in the files analyzed for the given part p, in contrast to the total number of *entries* $N_{\mathrm{R},p}$; where $\widehat{N}_{p,e}$ is the number of different entries in the given event, and where $\widetilde{w}_{\mathrm{R},p,e,j}$ is the secondary or correlated weight of the entry (given by the `weight2` branch of the ROOT file). For simplicity of use, events are ordered in the n-tuple files, with real-emission entries and counter-entries with the same `id` appearing contiguously.

If we choose to combine different batches of files using eq. (12), we also need the corresponding formula for combining uncertainty estimates,

$$\varepsilon_{\mathcal{O},1+2}^{(t,p)} = \frac{1}{\sqrt{N_{t,p,1+2}(N_{t,p,1+2}-1)}} \times \Big[N_{t,p,1}(N_{t,p,1}-1)\big(\varepsilon_{\mathcal{O},1}^{(t,p)}\big)^2 + N_{t,p,2}(N_{t,p,2}-1)\big(\varepsilon_{\mathcal{O},2}^{(t,p)}\big)^2 \Big]^{1/2}, \tag{15}$$

for the B, V, and I contributions, and a similar equation with $N_{t,p} \to \widetilde{N}_{\mathrm{R},p}$ for the R contribution.

The different types of contributions are statistically independent, as they are generated using independent sets of phase-space configurations, and so the overall error estimate simply adds the error estimates for the different parts in quadrature,

$$\varepsilon_{\mathcal{O}} = \Big[\sum_{t\in T, p\in P_t} \big(\varepsilon_{\mathcal{O}}^{(t,p)}\big)^2 \Big]^{1/2}. \tag{16}$$

In addition to overall values of observables, we will typically want to obtain values of distributions as well. To obtain the prediction for the distribution in a variable v, we must divide its range B into bins (possibly including overflow and underflow bins). For each bin, the expected value is given by,

$$\left\langle \frac{d\sigma}{dv} \right\rangle_b = \sum_{t\in T, p\in P_t} \left\langle \frac{d\sigma}{dv} \right\rangle_b^{(t,p)} , \tag{17}$$

where

$$\left\langle \frac{d\sigma}{dv} \right\rangle_b^{(t,p)} = \frac{1}{\Delta_b} \frac{1}{N_{t,p}} \sum_{e=1}^{N_{t,p}} w_{t,p,e} \hat{\delta}(v_{t,p,e}, b) , \tag{18}$$

where Δ_b is the bin's width (in the units of the observable), and where $\hat{\delta}(v, b)$ is 1 if the value lies in the given bin b, and 0 otherwise. The values of the distribution for different bins can of course be obtained in parallel, by assigning each file entry to the appropriate bin. This formula is what is typically implemented by histogramming codes, which can be used straightforwardly to process the n-tuple files for all the types of contributions.

For the error estimates, we must again treat the subtracted real-emission contribution specially. For each part in each of the other types (B, V, and I), we have a standard form for the error estimate,

$$\begin{aligned} \varepsilon_{d\sigma/dv,b}^{(t,p)} = \frac{1}{\Delta_b} \frac{1}{\sqrt{N_{t,p}(N_{t,p}-1)}} \Bigg[& \sum_{e=1}^{N_{t,p}} w_{t,p,e}^2 \hat{\delta}(v_{t,p,e}, b) \\ & - \frac{1}{N_{t,p}} \left(\sum_{e=1}^{N_{t,p}} w_{t,p,e} \hat{\delta}(v_{t,p,e}, b) \right)^2 \Bigg]^{1/2} . \end{aligned} \tag{19}$$

This again is the formula that will typically be the one implemented by histogramming routines.

For the R contribution, we must take the anticorrelation into account, so that the error estimate is given by,

$$\begin{aligned} \varepsilon_{\mathcal{O}}^{(\mathrm{R},p)} = \frac{1}{\Delta_b} \frac{1}{\sqrt{\widetilde{N}_{\mathrm{R},p}(\widetilde{N}_{\mathrm{R},p}-1)}} \Bigg[& \sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \left(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b) \right)^2 \\ & - \frac{1}{\widetilde{N}_{\mathrm{R},p}} \left(\sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b) \right)^2 \Bigg]^{1/2} . \end{aligned} \tag{20}$$

A given event is counted as being in a bin if any of its entries are in a bin; a given event can therefore appear in more than one bin. The extent to which this happens, and hence the error estimates, will depend on the value of the integrated subtraction's $\alpha_{\rm dipole}$ parameter [38], which is 0.03 for all the n-tuple files documented here. This formula is *not* what is implemented by standard histogramming routines; in particular, current versions of RIVET [39] do not take anticorrelations into account, and so will not compute statistical error estimates correctly for this contribution. The user will typically need to supply his or her own histogramming routines to obtain correct estimates.

Histograms with statistical uncertainty estimates obtained using eq. (20) cannot be rebinned in the usual way (that is keeping track of the sum of the weights and the sum of the weights squared for each bin). To understand why, let us consider two neighboring bins b_1 and b_2 and see what happens if we want to compute the error associated with the combined bins which we will refer to as $b_1 + b_2$. The second term in the calculation of the error will be the same, whether one considers the bins separately or not. Let us consider the first term,

$$\begin{aligned}
&\sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \left(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b_1 + b_2) \right)^2 \\
&\quad = \sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \left(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \left(\hat{\delta}(v_{t,p,e,j}, b_1) + \hat{\delta}(v_{t,p,e,j}, b_2) \right) \right)^2 \\
&\quad = \sum_{e=1}^{\widetilde{N}_{\mathrm{R},p}} \left[\left(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b_1) \right)^2 + \left(\sum_{j=1}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b_2) \right)^2 \right. \\
&\qquad \left. + 2 \left(\sum_{j,j'=1; j \neq j'}^{\widehat{N}_{p,e}} \widetilde{w}_{\mathrm{R},p,e,j} \hat{\delta}(v_{t,p,e,j}, b_1) \widetilde{w}_{\mathrm{R},p,e,j'} \hat{\delta}(v_{t,p,e,j'}, b_2) \right) \right] \qquad (21)
\end{aligned}$$

The terms on the penultimate line are typically recorded (as the statistical uncertainty estimate) for each bin alongside the sum of weights. The term on the last line is not. In the absence of correlations between entries in an event — if $\widehat{N}_{p,e} = 1$ — the double sum over j and j' would disappear, and we would recover the usual formula for combining uncertainty estimates of two bins. Because the weights of real-emission and counter-configuration entries in an event are typically anticorrelated, however, the last term typically will

be negative and will reduce the uncertainty estimate significantly. Accordingly, naively rebinning histograms with uncertainties computed according to eq. (20) but then combined in quadrature will overestimate the statistical uncertainty. The same argument applies to extracting cumulative distributions from a histogram. If the user plans to rebin or extract cumulative distributions, the last term in eq. (21) should be recorded for all bin pairs along with the sum of the squared weights.

We again obtain the overall error estimate for each bin by adding the separate contributions in quadrature,

$$\varepsilon_{d\sigma/dv,b} = \Big[\sum_{t\in T, p\in P_t} \big(\varepsilon^{(t,p)}_{d\sigma/dv,b}\big)^2 \Big]^{1/2} . \tag{22}$$

5. Changing Scales and Parton Distributions

The BlackHat+SHERPA n-tuple files contain the information needed to recompute the weight of the event for a different renormalization or factorization scale, or for a different pdf set. The new weights can be used to compute new central values, or to compute scale-variation bands and pdf-uncertainty estimates. In the following subsections, we explain the additional information stored in the n-tuple files, and how to make use of it. We denote by f_1 and f_2 the pdf of the first and second hadron respectively. For the LHC, in which both beams are protons, they both correspond to proton pdfs. `Variables` shown in a distinct font correspond to branches in the Root file. Indices in arrays such as `usr_wgts` are zero-based.

5.1. Born and subtracted-real contributions

The new weight is given by

$$n = \texttt{alphas_power}\,, \tag{23}$$

$$w = \texttt{me_wgt2}\, f_1(\texttt{id1}, \texttt{x1}, \mu_{\rm F})\, f_2(\texttt{id2}, \texttt{x2}, \mu_{\rm F})\, \frac{\alpha_s(\mu_{\rm R})^n}{(\texttt{alphas})^n}\,, \tag{24}$$

where $\mu_{\rm F}$ is the new factorization scale, $\mu_{\rm R}$ the new renormalization scale, $f_{1,2}$ the new pdf, α_s the corresponding running coupling, and n the power of the strong coupling α_s. (In the case of W or Z + n_j-jet processes, this power is n_j in the Born (B) contribution and $n_j + 1$ in the subtracted-real (R) contribution.) The new scales must be infrared- and collinear-safe functions of the final-state momenta. If the factorization scale and pdf set are

left unchanged, one can simplify the computation of w, eliminating the pdf function call:

$$w \;=\; \texttt{weight2}\,\frac{\alpha_s(\mu_\mathrm{R})^n}{(\texttt{alphas})^n}\,. \tag{25}$$

Alternatively, if we leave the renormalization scale unaltered, we could simplify the computation of w to:

$$w \;=\; \texttt{me_wgt2}\,f_1(\texttt{id1},\texttt{x1},\mu_\mathrm{F})\,f_2(\texttt{id2},\texttt{x2},\mu_\mathrm{F})\,. \tag{26}$$

For the Born contribution (but not the subtracted-real one), `weight` and `weight2` are the same.

5.2. Virtual contribution

The virtual contribution (V) is treated in a similar way as the real and Born contributions, except that the matrix element has an explicit dependence on the renormalization scale. In dimensional regularization, this dependence arises from the introduction of a scale to give the coupling g the required dimension, $g \to g\mu^\epsilon$, along with the $\overline{\mathrm{MS}}$ ultraviolet subtraction that replaces the bare coupling g_0 with the physical coupling $g(\mu_\mathrm{R})$,

$$g^n\mu_\mathrm{R}^\epsilon c_\Gamma\hat{\mathcal{A}}_n^{(1)} \longrightarrow g^n\mu_\mathrm{R}^\epsilon c_\Gamma\hat{\mathcal{A}}_n^{(1)} - b_0(n-2)g^n c_\Gamma\hat{\mathcal{A}}_n^{(0)}\,, \tag{27}$$

where $b_0 = 11/2 - n_f/3$, $c_\Gamma = \Gamma(1+\epsilon)\Gamma^2(1-\epsilon)/\big((4\pi)^{2-\epsilon}\Gamma(1-2\epsilon)\big)$, and $\hat{\mathcal{A}}^{(0,1)}$ are respectively the tree-level and one-loop amplitude with factors of g and c_Γ removed. This gives rise to a term of the form $b_0\alpha_s\mathcal{A}^{(0)}\ln(\mu_\mathrm{R}/s)$. Because μ_R also enters terms with infrared divergences, there are also double-logarithmic terms in the virtual contribution. In the sum over all contributions, they will cancel statistically against similar terms in the integrated-subtraction term, leaving only double logs of ratios of invariants as well as the linear term in $\ln\mu_\mathrm{R}$. A dependence on $\ln\mu_\mathrm{R}$ beyond linear will remain from the running of the strong coupling α_s. In calculations using n-tuples, the dependence on $\mu_{\mathrm{R,F}}$ is taken into account using the additional weights `usr_wgts`. These weights are computed from the coefficients $\mathcal{L}^{(1,2)}$ of the $1/\epsilon$ and $1/\epsilon^2$ poles respectively in the virtual contribution [with the coupling removed],

$$\begin{aligned}\texttt{usr_wgts}[0] &= \frac{1}{2\pi}\mathcal{L}^{(1)}\Phi\,,\\ \texttt{usr_wgts}[1] &= \frac{1}{2\pi}\mathcal{L}^{(2)}\Phi\,,\end{aligned} \tag{28}$$

and saved in the n-tuple file. The coefficient $\mathcal{L}^{(1)}$ depends on b_0. In this equation, Φ is the phase-space measure at the configuration for the event, including Jacobian factors from any remappings used by the phase-space generator. The b_0 terms mentioned above reside within $\mathcal{L}^{(1)}$. An analysis code can then recompute the event weight using the following formulas,

$$n = \texttt{alphas_power}\,, \tag{29}$$

$$l = \ln\left(\frac{\mu_{\rm R}^2}{\texttt{ren_scale}^2}\right)\,, \tag{30}$$

$$m = \texttt{me_wgt} + l\,\texttt{usr_wgts[0]} + \frac{l^2}{2}\,\texttt{usr_wgts[1]}\,, \tag{31}$$

$$w = m\,f_1(\texttt{id1},\texttt{x1},\mu_{\rm F}) f_2(\texttt{id2},\texttt{x2},\mu_{\rm F})\,\frac{\alpha_s(\mu_{\rm R})^n}{(\texttt{alphas})^n}\,. \tag{32}$$

In these equations, `ren_scale` is the reference renormalization scale $\mu_{\rm R,0}$. If a new pdf set is chosen, the α_s should correspond to that set. If we do not change the renormalization scale, the scale-changing logarithm in eq. (30) will vanish, and along with it the additional terms in eq. (31). (In the case of W or Z + n_j-jet processes, n is $n_j + 1$ for this contribution.)

5.3. Integrated subtraction contribution

The computation of the new weight for the integrated subtraction contribution (I) is the most complicated. The n-tuple file has 16 additional weights {`usr_wgts[2]`,...,`usr_wgts[17]`} which make a re-computation possible. These weights are computed from the virtual pole coefficients defined in sect. 5.2, along with other quantities extracted from the Catani–Seymour subtraction formalism [17]. These additional quantities include the coefficients $\mathcal{V}_i^{(1,2)}$ of the $1/\epsilon$ and $1/\epsilon^2$ poles respectively in the integrated subtraction functions for parton i, extracted from eqs. (5.32–5.34) of ref. [17],

$$\mathcal{V}_i^{(2)} = \begin{cases} C_F\,, & i = q\,, \\ C_A\,, & i = g\,; \end{cases} \tag{33}$$

$$\mathcal{V}_i^{(1)} = \begin{cases} \dfrac{3}{2}C_F\,, & i = q\,, \\ \dfrac{11}{6}C_A - \dfrac{n_f}{3}\,, & i = g\,, \end{cases} \tag{34}$$

(C_R is the Casimir in representation R of SU(3)); the color- (and helicity-) summed squared tree-level matrix element with factors of the strong coupling removed, $|\mathcal{M}|^2$; a logarithm $\ell_{ik} = \ln \mu^2_{\mathrm{F},0}/|2p_i \cdot p_k|$; and the color-correlated squared tree-level matrix element as given [up to a different normalization] in eq. (3.13) of ref. [17],

$$\mathcal{M}_{ik} = \langle 1, \ldots, n| \, \frac{\mathbf{T}_i \cdot \mathbf{T}_k}{\mathbf{T}_i^2} \, |1, \ldots, n\rangle \ , \tag{35}$$

where $|1, \ldots, n\rangle$ is the tree amplitude in the color–helicity basis of ref. [17] (again, with factors of the strong coupling removed).

We also need the four-dimensional regularized Altarelli–Parisi splitting functions $P^{ab}(x)$, given in eqs. (5.85–5.88) of ref. [17]; an additional function, related to the splitting functions,

$$\begin{aligned}
\breve{P}^{q,q}(x) &= 2C_F \Big(\ln \frac{1}{1-x} - x - \frac{x^2}{2} \Big) \,, \\
\breve{P}^{g,g}(x) &= 2C_A \ln \frac{1}{1-x} \,, \\
\breve{P}^{g,q}(x) &= \breve{P}^{q,g}(x) = 0 \,;
\end{aligned} \tag{36}$$

the auxiliary functions $\overline{K}^{\,ab}(x)$, defined in eqs. (8.32–8.25) of ref. [17]; and additional functions,

$$\begin{aligned}
\breve{K}^{q,q}(x) &= 2C_F \Big(\mathrm{Li}_2(x) - \frac{1}{2} \ln^2(1-x) + \ln(x) \ln(1-x) \Big) \\
&\quad - C_F \Big(5 - \pi^2 \Big) \,, \\
\breve{K}^{g,g}(x) &= 2C_A \Big(\mathrm{Li}_2(x) - \frac{1}{2} \ln^2(1-x) + \ln(x) \ln(1-x) \Big) \\
&\quad - \Big[\Big(\frac{50}{9} - \pi^2 \Big) C_A - \frac{8}{9} n_f \Big] \,, \\
\breve{K}^{q,g}(x) &= \breve{K}^{g,q}(x) = 0 \,,
\end{aligned} \tag{37}$$

and

$$\widetilde{P}_i(x) = \gamma_i \big(1 - \ln(1-x) \big) \,, \tag{38}$$

where $\gamma_i = \mathcal{V}_i^{(1)}$ is the collinear anomalous dimension given in eq. (5.90) of ref. [17], with $\mathcal{V}_i^{(1)}$ defined in eq. (34).

In terms of these quantities, we have the following expressions for the set of `usr_wgts`,

$$
\begin{aligned}
\texttt{usr_wgts[0]} &= |\mathcal{M}|^2\Phi\Big[\mathcal{L}^{(1)} + n_B\, b_0 \,-\sum_{i\neq k}\frac{\mathcal{M}_{ik}}{|\mathcal{M}|^2}\Big(\mathcal{V}_i^{(1)} + \mathcal{V}_i^{(2)}\ln\frac{\mu^2_{\mathrm{R},0}}{|2p_i\cdot p_k|}\Big)\Big]\,,\\
\texttt{usr_wgts[1]} &= |\mathcal{M}|^2\Phi\Big[\mathcal{L}^{(2)} - \sum_{i\neq k}\frac{\mathcal{M}_{ik}}{|\mathcal{M}|^2}\mathcal{V}_i^{(2)}\Big]\,,\\
\texttt{usr_wgts[2]} &= |\mathcal{M}|^2\Phi\Big[\breve{K}^{q,a}(x) + \Big(\delta^{qa}\sum_{i\neq a}\frac{\mathcal{M}_{ia}}{|\mathcal{M}|^2}\widetilde{P}_i(x) - \breve{P}^{q,a}(x)\ell_{ia}\Big)\Big]\,,\\
\texttt{usr_wgts[3]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}\Big[\overline{K}^{\,q,a}(x') + \sum_{i\neq a}\Big(\frac{\mathcal{M}_{ia}}{|\mathcal{M}|^2}\delta^{qa}\frac{\gamma_i}{1-x'} + \frac{\mathcal{M}_{ai}}{|\mathcal{M}|^2}P^{q,a}(x')\ell_{ia}\Big)\Big]\,,\\
\texttt{usr_wgts[4]} &= |\mathcal{M}|^2\Phi\Big[\breve{K}^{g,a}(x) + \Big(\delta^{ga}\sum_{i\neq a}\frac{\mathcal{M}_{ia}}{|\mathcal{M}|^2}\widetilde{P}_i(x) - \breve{P}^{g,a}(x)\ell_{ia}\Big)\Big]\,,\\
\texttt{usr_wgts[5]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}\Big[\overline{K}^{\,g,a}(x') + \sum_{i\neq a}\Big(\frac{\mathcal{M}_{ia}}{|\mathcal{M}|^2}\delta^{ga}\frac{\gamma_i}{1-x'} + \frac{\mathcal{M}_{ai}}{|\mathcal{M}|^2}P^{g,a}(x')\ell_{ia}\Big)\Big]\,,\\
\texttt{usr_wgts[6]} &= |\mathcal{M}|^2\Phi\Big[\breve{K}^{q,b}(x) + \Big(\delta^{qb}\sum_{i\neq b}\frac{\mathcal{M}_{ib}}{|\mathcal{M}|^2}\widetilde{P}_i(x) - \breve{P}^{q,b}(x)\ell_{ib}\Big)\Big]\,,\\
\texttt{usr_wgts[7]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}\Big[\overline{K}^{\,q,b}(x') + \sum_{i\neq b}\Big(\frac{\mathcal{M}_{ib}}{|\mathcal{M}|^2}\delta^{qb}\frac{\gamma_i}{1-x'} + \frac{\mathcal{M}_{bi}}{|\mathcal{M}|^2}P^{q,b}(x')\ell_{ib}\Big)\Big]\,,\\
\texttt{usr_wgts[8]} &= |\mathcal{M}|^2\Phi\Big[\breve{K}^{g,b}(x) + \Big(\delta^{gb}\sum_{i\neq b}\frac{\mathcal{M}_{ib}}{|\mathcal{M}|^2}\widetilde{P}_i(x) - \breve{P}^{g,b}(x)\ell_{ib}\Big)\Big]\,,\\
\texttt{usr_wgts[9]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}\Big[\overline{K}^{\,g,b}(x') + \sum_{i\neq b}\Big(\frac{\mathcal{M}_{ib}}{|\mathcal{M}|^2}\delta^{gb}\frac{\gamma_i}{1-x'} + \frac{\mathcal{M}_{bi}}{|\mathcal{M}|^2}P^{g,b}(x')\ell_{ib}\Big)\Big]\,,\\
\texttt{usr_wgts[10]} &= |\mathcal{M}|^2\Phi\breve{P}^{q,a}(x)\,, \qquad & \texttt{usr_wgts[14]} &= |\mathcal{M}|^2\Phi\breve{P}^{q,b}(x)\,,\\
\texttt{usr_wgts[11]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}P^{q,a}(x')\,, \qquad & \texttt{usr_wgts[15]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}P^{q,b}(x')\,,\\
\texttt{usr_wgts[12]} &= |\mathcal{M}|^2\Phi\breve{P}^{g,a}(x)\,, \qquad & \texttt{usr_wgts[16]} &= |\mathcal{M}|^2\Phi\breve{P}^{g,b}(x)\,,\\
\texttt{usr_wgts[13]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}P^{g,a}(x')\,, \qquad & \texttt{usr_wgts[17]} &= |\mathcal{M}|^2\Phi\frac{1}{x'}P^{g,b}(x')\,.
\end{aligned}
\tag{39}
$$

In these expressions, Φ is again the phase-space measure at the configuration for the event, including Jacobian factors from any remappings used by the

phase-space generator; n_B is the order in α_s of the Born process. In these equations, a is the initial-state parton type and flavor in hadron 1, and b is the initial-state type and flavor of the parton from hadron 2. In the weights where a appears (2–5 and 10–13), x is `x1` and x' is `x1p`, whereas in the weights where b appears (6–9 and 14–17), x is `x2` and x' is `x2p`.

With the additional `usr_wgts` coefficients, an analysis code can use the following formulas to recompute the event weight,

$$n = \texttt{alphas_power}\,, \tag{40}$$

$$l = \ln\left(\frac{\mu_{\rm R}^2}{\texttt{ren_scale}^2}\right)\,, \tag{41}$$

$$\omega_0 = \texttt{me_wgt} + l\,\texttt{usr_wgts}[0] + \frac{l^2}{2}\,\texttt{usr_wgts}[1]\,, \tag{42}$$

$$\omega_i = \texttt{usr_wgts}[i{+}1] + \texttt{usr_wgts}[i{+}9]\ln\left(\frac{\mu_{\rm F}^2}{\texttt{fac_scale}^2}\right)\,, \tag{43}$$

$$\begin{aligned} m = {} & \omega_0\, f_1(\texttt{id1},\texttt{x1},\mu_{\rm F})\, f_2(\texttt{id2},\texttt{x2},\mu_{\rm F}) \\ & + \left(\sum_{j=1}^{4} f_1^{(j)}(\texttt{id1},\texttt{x1},\texttt{x1p},\mu_{\rm F})\,\omega_j\right) f_2(\texttt{id2},\texttt{x2},\mu_{\rm F}) \\ & + f_1(\texttt{id1},\texttt{x1},\mu_{\rm F})\left(\sum_{j=1}^{4} f_2^{(j)}(\texttt{id2},\texttt{x2},\texttt{x2p},\mu_{\rm F})\,\omega_{j+4}\right)\,, \end{aligned} \tag{44}$$

$$w = m\,\frac{\alpha_s(\mu_{\rm R})^n}{(\texttt{alphas})^n}\,, \tag{45}$$

where ($r = 1$ or 2)

$$f_r^{(1)}(i,x,x',\mu_{\rm F}) = \begin{cases} i = \text{quark}: & f_r(i,x,\mu_{\rm F})\,, \\ i = \text{gluon}: & \sum_{\text{quarks } q} f_r(q,x,\mu_{\rm F})\,, \end{cases} \tag{46}$$

$$f_r^{(2)}(i,x,x',\mu_{\rm F}) = \begin{cases} i = \text{quark}: & f_r(i,x/x',\mu_{\rm F})/x'\,, \\ i = \text{gluon}: & \sum_{\text{quarks } q} f_r(q,x/x',\mu_{\rm F})/x'\,, \end{cases} \tag{47}$$

$$f_r^{(3)}(i,x,x',\mu_{\rm F}) = f_r(g,x,\mu_{\rm F})\,, \tag{48}$$

$$f_r^{(4)}(i,x,x',\mu_{\rm F}) = f_r(g,x/x',\mu_{\rm F})/x'\,. \tag{49}$$

The sums over quarks are taken over the quark flavors active at the scale $\mu_{\rm F}$, typically five. In these equations, `ren_scale` is the reference renormalization scale $\mu_{\rm R,0}$, and `fac_scale` is the reference factorization scale $\mu_{\rm F,0}$. (In the case of W or Z + n_j-jet processes, n is $n_j + 1$ for this contribution.)

6. Library for Reading BHS n-Tuple Files

This section describes a C++ library, `nTupleReader`, which provides an easy-to-use interface to the n-tuple files described in the above sections.

6.1. Dependencies

The library depends on LHAPDF [40] (`http://lhapdf.hepforge.org/`) and ROOT [32] (`http://root.cern.ch/drupal/`).

6.2. Installation

The library is an autotools package, and is installed using the usual paradigm of `configure; make; make install`. Along with the usual autotools options, the library's `configure` script offers the following options:

`--with-lhapdf-path=`⟨path to LHAPDF installation⟩
: *Sets the location of the* LHAPDF *installation; this option is needed only if the helper program* `lhapdf-config` *is not in the executable search path.*

`--with-root-path=`⟨path to ROOT installation⟩
: *Sets the location of the root installation; this option is needed only if the helper program* `root-config` *is not in the executable search path.*

`--enable-pythoninterface`
: *Compiles the python interface (see section 6.7).*

To install the library, unpack the `.tar.gz` file,

`tar -xzf ntuplereader-`⟨version⟩`.tar.gz`

where ⟨version⟩ is a string like '1.0'. Configure, with options as needed,

```
cd ntuplereader-⟨version⟩
./configure ⟨options⟩
```

and then compile and install,

```
make
make install
```

To install in a different location than the standard one, use the `--prefix` option to the `configure` script.

6.3. Usage

Your source code must include the header file `nTupleReader.h` to use the library. Your code may access information in the n-tuple files via an object of the `nTupleReader` class. We give an example of a C++ program using the library in section 6.8.

The `nTupleReader-config` script returns the compiler and linker flags needed to compile a user program and link it to the library. The script is located in the `bin` subdirectory of the installation directory. Assuming this script is in your executable search path, you would compile and link a program as follows,

```
CFLAGS=‘nTupleReader-config --include‘
g++ -c $CFLAGS -o NTRexample.o NTRexample.cpp
LDFLAGS=‘nTupleReader-config --libs‘
g++ $LDFLAGS -o NTRexample NTRexample.o
```

6.4. The `getInfo` *program*

The `make` step creates a program called `getInfo` along with the library. This program takes one argument, the name of a n-tuple file. The program prints useful information about events in the file:

- their center-of-mass energy;
- their initial-state hadrons;
- the process that generated them;
- the part of the NLO calculation to which they contribute;
- the jet algorithms and parameters allowed in an analysis;
- the minimum jet transverse-momentum cut allowed;
- additional generation-level cuts, such as the lepton invariant-mass cut for processes with a Z boson.;
- the electroweak parameters used;
- the pdf set used in the n-tuple generation.

6.5. Member functions of an `nTupleReader` *object*

This section describes the member functions of the `nTupleReader` class, which allow access to the data in n-tuple files. We give an example of their use in section 6.8.

The member functions are,

`void addFile(const std::string &fileName)`
Adds a file to the reader.
`fileName` *is the name of the file.*

`void addFiles(std::vector<std::string> fileNames)`
Adds a list of files to the reader
`fileNames` *is a vector of* `std::string` *containing the names of the files to be added, which will be read in the order given.*

```
double computeWeight(double newFactorizationScale,
                     double newRenormalizationScale)
```
Returns the weight (`weight`) of the current entry recomputed for the new scales, using the current pdf member number in the current pdf set.
`newFactorizationScale` *is the new factorization scale (in GeV)*
`newRenormalizationScale` *is the new renormalization scale (in GeV)*

```
double computeWeight2(double newFactorizationScale,
                      double newRenormalizationScale)
```
Returns the secondary weight (`weight2`) of the current entry recomputed for the new scales, using the current pdf member number in the current pdf set. One should use this weight for the real part in order to take into account the correlation between the entry and counter entries.
`newFactorizationScale` *is the new factorization scale (in GeV)*
`newRenormalizationScale` *is the new renormalization scale (in GeV)*

`short getAlphasPower()`
Returns the power of the strong coupling constant in the current entry.

`long getEndEntryIndex()`
Returns the (1-based) index of the entry at which reading will stop.

`double getEnergy(int i)`
Returns the energy of the i^{th} particle in the current entry.
`i` *is a 0-based index; an argument equal to or larger than the number of final state particles will throw an* `nTR_OutOfBounds` *exception.*

`void getEntry(long index)`
Reads the entry corresponding to the index specified; `nextEntry`() will start from that position.
`index` *is the index of the entry to be read.*

`double getFactorizationScale()`
Returns the factorization scale used to compute the weights for the current entry.

`int getID()`
Returns the ID of the current event.

`double getId1()`
Returns the PDG code for the first (forward) incoming parton in the current entry.

`double getId2()`
Returns the PDG code for the second (backward) incoming parton in the current entry.

`long getIndexOfNextEntry()`
Returns the index of the next entry.

`double getMEWeight()`
Returns the weight for the current entry omitting pdf factors.

`double getMEWeight2()`
Returns the secondary weight for the current entry omitting the pdf factors, to be used as described in section 4 to obtain the correct estimate of the statistical uncertainty.

`long getMaxEvent()`
Returns the ID of the event (phase-space configuration–counter-configuration group of entries, or simply entries for contribution types with one entry per event) after which reading will stop.

`long getNumberOfEntries()`
Returns the total number of entries.

`int getPDGcode(int i)`
Returns the PDG code of the i^{th} particle in the current entry.
`i` *is a 0-based index; an argument equal to or larger than the number of final state particles will throw an* `nTR_OutOfBound` *exception.*

`int getParticleNumber()`
Returns the number of final state particles in the current entry.

`double getRenormalizationScale()`
Returns the renormalization scale used to compute the weights for the current entry.

`long getStartEntryIndex()`
Returns the (1-based) index of the entry at which reading will start.

`char getType()`
Returns the type of the current entry, 'B' standing for born, 'I' for integrated subtraction, 'V' for the virtual, and 'R' for the subtracted real emission.

`double getWeight()`
Returns the weight (`weight`) for the current entry.

`double getWeight2()`
Returns the secondary weight (`weight2`) for the current entry, to be used as described in section 4 to obtain the correct estimate of the statistical uncertainty.

`double getX(int i)`
Returns the x component of the i^{th} particle's momentum in the current entry.
`i` *is a 0-based index; an argument equal to or larger than the number of final state particles will throw an* `nTR_OutOfBound` *exception.*

`double getX1()`
Returns the momentum fraction x_1 in the current entry.

`double getX2()`
Returns the momentum fraction x_2 in the current entry.

`double getY(int i)`
Returns the y component of the i^{th} particle's momentum in the current entry.

`i` *is a 0-based index; an argument equal to or larger than the number of final state particles will throw an* `nTR_OutOfBound` *exception.*

`double getZ(int i)`
Returns the z component of the i^{th} particle's momentum in the current entry.

`i` *is a 0-based index; an argument equal to or larger than the number of final state particles will throw an* `nTR_OutOfBound` *exception.*

`bool nextEntry()`
Reads the next entry and returns true upon success, false otherwise (including when the end of the file is reached).

`void setEndEntryIndex(long index)`
Sets the index of the entry at which reading will stop.

`index` *1-based index at which the reading will stop (the* `index`*-th entry will not be read)*

`void setMaxEvent(long count)`
Sets the reader to stop reading entries when the given number of events (phase-space configuration–counter-configuration groups of entries, or simply entries for contribution types with one entry per event) have been read.

`count` *is a 1-based sequence number specifying the first event that will not be read.*

`void setPDF(const std::string &name)`
Sets the pdf set to be used.

`name` *is the name of the file to be loaded by* LHAPDF*, for example* `CT10.LHgrid`.

`void setPDFmember(int member)`
Sets the pdf member number to be used.

`member` *is an integer labeling the member; 0 is typically used to denote the central value.*

`void setPP()`
Sets the initial state to proton–proton. This is the default if no calls to `setPP()` or `setPPbar()` are issued. This routine should only be invoked before using files generated for proton–proton colliders.

`void setPPbar()`
Sets the initial state to proton–antiproton. This routine should only be invoked before using files generated for proton–antiproton colliders.

`void setStartEntryIndex(long index)`
Sets the index of the entry at which reading will start.
`index` *1-based index at which the reading will start (the* `index`*-th entry will be read at the next call of* `nextEntry()`*)*

`void setStartEvent(long count)`
Sets the number for the event sequence counter at which the library will start reading in events (phase-space configuration–counter-configuration groups of entries, or simply entries for contribution types with one entry per event).
`count` *is a 1-based sequence number specifying the first event that will be read.*

6.6. Histogram Implementation Example

This section gives an example of the implementation of a histogramming procedure, as described in section 4.

```
nTupleReader r;
⋮
bool notFinished=r.nextEntry();
int lastID=r.getID();

while(notFinished){
  double wgt;
  // compute the value of the weight wgt here...
  int ID=r.getID()
```

```
  int bin=findBin(x);
  finalHistogram[bin]+=wgt;
  tempHistogram[bin]+=wgt;

  if (ID != lastID){
    for (int bin=0;bin<NbrBins;bin++){
      weightsSquare[bin]+=tempHistogram[bin]*tempHistogram[bin];
      if (   tempHistogram[bin] != 0.0 ){
        NbrEntries[bin]++;
      }
      tempHistogram[bin]=0.;
    }
    lastID=id;
  }
  notFinished=r.nextEvent()
}
```

where `NbrBins` is the numbers of bins in the histograms.

6.7. Example of the usage of the python interface

To use the python interface, make sure that the installation path for the libraries installed with the C++ library is included in the locations searched for python modules. The following is a sample python program,

```
import nTupleReader as NR

r=NR.nTupleReader()
r.addFile('@prefix@/share/ntuplereader/sample.root')

r.nextEntry()

for i in range(r.getParticleNumber()):
    print "p(%d)=(%f,%f,%f,%f)" % (
              i,
              r.getEnergy(i),
              r.getX(i),
              r.getY(i),
```

```
            r.getZ(i)
        )
```

6.8. Example of the usage of the library in a C++ program

The following listing shows an example of a C++ program using the `nTupleReader` library. This example is included after installation in the directory `share/ntuplereader/`.

```
#include "nTupleReader.h"

using namespace std;

int main(){

        std::vector<std::string> fs;
        fs.push_back("@prefix@/share/ntuplereader/sample.root");

        nTupleReader r;


        r.setPDF("cteq6ll.LHpdf");
        r.addFiles(fs);

        while(r.nextEntry()){
                int id=r.getID();
                cout <<
                   "Checking momentum conservation for event ID: "
                                << id << std::endl;
                double sumX=0;
                double sumY=0;
                double sumZ=0;
                int nbrP=r.getParticleNumber();
                std::cout << "Number of particles: "
                          << nbrP << std::endl;
                for (int i=0;i<nbrP;i++){
                        sumX+=r.getX(i);
```

```
                sumY+=r.getY(i);
                sumZ+=r.getZ(i);
            }
            std::cout << "Sum X: " << sumX << std::endl;
            std::cout << "Sum Y: " << sumY << std::endl;
            std::cout << "Sum Z: " << sumZ
                      << " to be compared with: "
                      <<  3500 * (r.getX1()-r.getX2())
                      <<  std::endl;
    }

    return 0;

}
```

7. Conclusions

In this Article we described software tools for obtaining predictions to NLO in QCD using an n-tuple event-file format generated by SHERPA using the BlackHat software library. This framework offers a convenient means for evaluating cross-sections and distributions with different parton distribution functions, bypassing repeated and computationally costly evaluations of matrix elements. The set-up described here makes it straightforward to compute scale-variation bands and to obtain uncertainty estimates due to imprecise knowledge of the parton distribution functions. The n-tuple files also make it convenient to study the effects of varying jet algorithms and cuts, within a wide range of commonly-used ones. These tools have already proved useful in a number of theoretical [31, 41, 34] and experimental studies [42, 43]. The reader may find an up-to-date list of available n-tuple sets at `http://blackhat.hepforge.org/trac/wiki/Availability`, stored at locations given in `http://blackhat.hepforge.org/trac/wiki/Location`. We look forward to further theoretical and experimental studies using the tools described here.

Acknowledgments

We thank Tanju Gleisberg for his collaboration in the initial stages of the development described above; he contributed in an important way to the tools described here. We are also grateful to Kemal Ozeren for his collaboration and input, and we thank Joey Huston for his continual encouragement. We also thank David Saltzberg, and Eric Takasugi for discussions and feedback. This research was supported by the US Department of Energy under contracts DE–AC02–76SF00515 and DE–FG02–13ER42022. DAKs research is supported by the European Research Council under Advanced Investigator Grant ERC–AdG–228301. DM's work was supported by the Research Executive Agency (REA) of the European Union under the Grant Agreement number PITN–GA–2010–264564 (LHCPhenoNet). SH's work was partly supported by a grant from the US LHC Theory Initiative through NSF contract PHY–0705682. This research used resources of Academic Technology Services at UCLA, and of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE–AC02–05CH11231.

Appendix A. Tables of Cross Section Values

In this appendix we provide tables of cross sections obtained using the $\sqrt{s} = 8$ TeV n-tuples currently available on CASTOR and on the LHC Grid. These can serve as a reference for comparison purposes, and as a cross check that the sum over different contributions has been carried out correctly in user code. We list cross sections for a specific set of cuts identical to the ones used at 7 TeV in refs. [28, 29, 31]. (Sample distributions listed in Table 3 are included with each n-tuple file.)

Appendix A.1. Inclusive W + n-jet Production

Process	W^- LO	W^- NLO	W^+ LO	W^+ NLO
$W + 1$	341.4(0.2)	422.3(0.6)	487.4(0.4)	597(2)
$W + 2$	105.1(0.1)	104.1(0.3)	158.1(0.2)	154.0(0.5)
$W + 3$	27.6(0.04)	23.9(0.1)	43.85(0.08)	37.2(0.4)

Table A.4: Total cross sections in pb for $W + n$ jet production at the LHC at $\sqrt{s} = 8$ TeV, using the anti-k_T jet algorithm with $R = 0.5$.

We display the $\sqrt{s} = 8$ TeV cross sections for $W + n$ jet production in Table A.4. We define jets using the anti-k_T algorithm [21] with parameter $R = 0.5$. We apply the following cuts,

$$\begin{aligned} &E_{\rm T}^e > 20 \text{ GeV}\,, && |\eta^e| < 2.5\,, && \not{E}_{\rm T} > 20 \text{ GeV}\,, \\ &p_{\rm T}^{\rm jet} > 25 \text{ GeV}\,, && |\eta^{\rm jet}| < 3\,, && M_{\rm T}^W > 20 \text{ GeV}\,. \end{aligned} \tag{A.1}$$

The transverse mass of the W-boson is computed from the kinematics of its decay products, $W \to e\nu_e$: $M_{\rm T}^W = \sqrt{2E_{\rm T}^e E_{\rm T}^\nu (1 - \cos(\Delta\phi_{e\nu}))}$. In this case, the factorization and renormalization scales are set to,

$$\mu_{\rm R,0} = \mu_{\rm F,0} = \tfrac{1}{2}\hat{H}'_{\rm T}\,, \tag{A.2}$$

where $\hat{H}'_{\rm T}$ is defined in eq. (5). The LO cross sections are computed using MSTW2008 [36] LO pdfs, and the NLO cross sections using the MSTW2008 NLO pdfs. In each case, we use the $\alpha_s(\mu)$ corresponding to the parton distribution set. The corresponding 7 TeV cross sections are given in table I of ref. [34].

Appendix A.2. Inclusive $Z, \gamma^* + n$-jet Production

Process	LO	NLO
$Z + 1$	84.05(0.04)	102.6(0.2)
$Z + 2$	26.86(0.02)	26.50(0.06)
$Z + 3$	7.452(0.009)	6.58(0.09)

Table A.5: Total cross sections in pb for $Z, \gamma^* + n$ jet production at the LHC at $\sqrt{s} = 8$ TeV, using the anti-k_T jet algorithm with $R = 0.5$ and the cuts given in eq. (A.3).

For the 8 TeV cross sections listed in Table A.5, we choose the anti-k_T algorithm with $R = 0.5$ and impose the following set of cuts,

$$\begin{aligned} &E_{\rm T}^e > 20 \text{ GeV}\,, && |\eta^e| < 2.5\,, && 66 \text{ GeV} < M_{e^+e^-} < 116 \text{ GeV}\,, \\ &p_{\rm T}^{\rm jet} > 25 \text{ GeV}\,, && |\eta^{\rm jet}| < 3\,, \end{aligned} \tag{A.3}$$

where η is the pseudorapidity and $M_{e^+e^-}$ is the invariant mass of the e^+e^- decay pair. The invariant-mass cut is tighter than that used in generating the n-tuples, and other cuts are added in this calculation, illustrating precisely

this flexibility of the n-tuple setup. The factorization and renormalization scales are set to

$$\mu_{\mathrm{R},0} = \mu_{\mathrm{F},0} = \frac{1}{2}\hat{H}'_{\mathrm{T}}\,, \tag{A.4}$$

where $\hat{H}'_{\mathrm{T}}$ is defined in eq. (5). The MSTW2008 parton distribution functions are used. Cross sections at 7 TeV are given in ref. [29].

Appendix A.3. Inclusive n-jet Production

Jets	LO	NLO
2	1232.5(0.2)	1526.(2)
3	126.74(.03)	71.9(0.3)
4	14.36(0.01)	8.12(0.17)

Table A.6: Total cross sections in nb for pure jet production at the LHC at $\sqrt{s} = 8$ TeV, using the anti-k_T jet algorithm with $R = 0.4$ and the cuts given in eq. (A.5).

The cross sections for $2, 3, 4$-jet production at $\sqrt{s} = 8$ TeV are given in Table A.6. In this case we use the anti-k_T jet algorithm with $R = 0.4$ with the same cuts as in ref. [31]:

$$p_{\mathrm{T}}^{\mathrm{jet}} > 60\ \mathrm{GeV}\,, \qquad p_{\mathrm{T}}^{\text{leading jet}} > 80\ \mathrm{GeV}\,, \qquad |y|^{\mathrm{jet}} < 2.8\,, \tag{A.5}$$

where $p_{\mathrm{T}}^{\text{leading jet}}$ is the transverse momentum of the leading jet ordered in transverse momentum and y^{jet} is the rapidity of a jet. In this case, the factorization and renormalization scale are chosen to be,

$$\mu_{\mathrm{R},0} = \mu_{\mathrm{F},0} \equiv \mu = \frac{1}{2}\sum_j p_T^j\,, \tag{A.6}$$

where the sum runs over all final-state partons j. Again we use the MSTW2008 parton distribution functions. The 7 TeV cross sections are given in table I of ref. [31].

References

[1] G. Aad, et al., *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Phys. Lett. B716 (2012) 1–29. arXiv:1207.7214, doi:10.1016/j.physletb.2012.08.020.

[2] S. Chatrchyan, et al., *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Phys. Lett. B716 (2012) 30–61. arXiv:1207.7235, doi:10.1016/j.physletb.2012.08.021.

[3] Z. Bern, et al., *The NLO multileg working group: Summary report*, (2008) 1–120arXiv:0803.0494.

[4] J. Andersen, et al., *The SM and NLO Multileg Working Group: Summary report*, (2010) 21–189arXiv:1003.1241.

[5] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, H. Ita, D. A. Kosower, D. Maître, *An automated implementation of on-shell methods for one-loop amplitudes*, Phys. Rev. D78 (2008) 036003. arXiv:0803.4180, doi:10.1103/PhysRevD.78.036003.

[6] T. Gleisberg, S. Höche, F. Krauss, A. Schälicke, S. Schumann, J. Winter, *SHERPA 1.α: A proof of concept version*, JHEP 0402 (2004) 056. arXiv:hep-ph/0311263, doi:10.1088/1126-6708/2004/02/056.

[7] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, J. Winter, *Event generation with SHERPA 1.1*, JHEP 0902 (2009) 007. arXiv:0811.4622, doi:10.1088/1126-6708/2009/02/007.

[8] F. Krauss, R. Kuhn, G. Soff, *AMEGIC++ 1.0: A matrix element generator in C++*, JHEP 0202 (2002) 044. arXiv:hep-ph/0109036, doi:doi:10.1088/1126-6708/2002/02/044.

[9] T. Gleisberg, F. Krauss, *Automating dipole subtraction for QCD NLO calculations*, Eur. Phys. J. C53 (2008) 501–523. arXiv:0709.2881, doi:10.1140/epjc/s10052-007-0495-0.

[10] T. Gleisberg, S. Höche, *Comix, a new matrix element generator*, JHEP 0812 (2008) 039. arXiv:0808.3674, doi:10.1088/1126-6708/2008/12/039.

[11] R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, R. Pittau, et al., *Four-lepton production at hadron colliders: aMC@NLO predictions with theoretical uncertainties*, JHEP 1202 (2012) 099. arXiv:1110.4738, doi:10.1007/JHEP02(2012)099.

[12] R. K. Ellis, D. A. Ross, A. E. Terrano, *The Perturbative Calculation of Jet Structure in e^+e^- Annihilation*, Nucl. Phys. B178 (1981) 421. doi:10.1016/0550-3213(81)90165-6.

[13] M. L. Mangano, P. Nason, G. Ridolfi, *Heavy quark correlations in hadron collisions at next-to-leading order*, Nucl. Phys. B373 (1992) 295–345. doi:10.1016/0550-3213(92)90435-E.

[14] Z. Kunszt, D. E. Soper, *Calculation of jet cross-sections in hadron collisions at order α_3^3*, Phys. Rev. D46 (1992) 192–221. doi:10.1103/PhysRevD.46.192.

[15] S. Frixione, Z. Kunszt, A. Signer, *Three jet cross-sections to next-to-leading order*, Nucl. Phys. B467 (1996) 399–442. arXiv:hep-ph/9512328, doi:10.1016/0550-3213(96)00110-1.

[16] S. Catani, M. Seymour, *The Dipole formalism for the calculation of QCD jet cross-sections at next-to-leading order*, Phys. Lett. B378 (1996) 287–301. arXiv:hep-ph/9602277, doi:10.1016/0370-2693(96)00425-X.

[17] S. Catani, M. Seymour, *A General algorithm for calculating jet cross-sections in NLO QCD*, Nucl. Phys. B485 (1997) 291–419. arXiv:hep-ph/9605323, doi:10.1016/S0550-3213(96)00589-5.

[18] A. Gehrmann-De Ridder, T. Gehrmann, E. N. Glover, *Antenna subtraction at NNLO*, JHEP 0509 (2005) 056. arXiv:hep-ph/0505111, doi:10.1088/1126-6708/2005/09/056.

[19] Z. Nagy, D. E. Soper, *General subtraction method for numerical calculation of one loop QCD matrix elements*, JHEP 0309 (2003) 055. arXiv:hep-ph/0308127.

[20] R. Frederix, S. Frixione, F. Maltoni, T. Stelzer, *Automation of next-to-leading order computations in QCD: The FKS subtraction*, JHEP 0910 (2009) 003. arXiv:0908.4272, doi:10.1088/1126-6708/2009/10/003.

[21] M. Cacciari, G. P. Salam, G. Soyez, *The Anti-$k(t)$ jet clustering algorithm*, JHEP 0804 (2008) 063. arXiv:0802.1189, doi:10.1088/1126-6708/2008/04/063.

[22] S. Catani, Y. L. Dokshitzer, M. H. Seymour, B. Webber, *Longitudinally invariant K_t clustering algorithms for hadron hadron collisions*, Nucl. Phys. B406 (1993) 187–224. doi:10.1016/0550-3213(93)90166-M.

[23] G. P. Salam, G. Soyez, *A Practical Seedless Infrared-Safe Cone jet algorithm*, JHEP 0705 (2007) 086. arXiv:0704.0292, doi:10.1088/1126-6708/2007/05/086.

[24] 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.

[25] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, D. Maître, *Precise predictions for W + 3-jet production at hadron colliders*, Phys. Rev. Lett. 102 (2009) 222001. arXiv:0902.2760, doi:10.1103/PhysRevLett.102.222001.

[26] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, D. Maître, *Next-to-leading order QCD predictions for W + 3-jet distributions at hadron colliders*, Phys. Rev. D80 (2009) 074036. arXiv:0907.1984, doi:10.1103/PhysRevD.80.074036.

[27] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, D. Maître, *Next-to-leading order QCD predictions for Z, γ^* + 3-jet distributions at the Tevatron*, Phys. Rev. D82 (2010) 074002. arXiv:1004.1659, doi:10.1103/PhysRevD.82.074002.

[28] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, D. Maître, *Precise predictions for W + 4-jet production at the Large Hadron Collider*, Phys. Rev. Lett. 106 (2011) 092001. arXiv:1009.2338, doi:10.1103/PhysRevLett.106.092001.

[29] H. Ita, Z. Bern, L. J. Dixon, F. Febres Cordero, D. A. Kosower, D. Maître, *Precise predictions for Z + 4 jets at hadron colliders*, Phys. Rev. D85 (2012) 031501. arXiv:1108.2229, doi:10.1103/PhysRevD.85.031501.

[30] Z. Bern, G. Diana, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, K. Ozeren, *Driving missing data at next-to-leading order*, Phys. Rev. D84 (2011) 114002. arXiv:1106.1423, doi:10.1103/PhysRevD.84.114002.

[31] Z. Bern, G. Diana, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, K. Ozeren, *Four-jet production at the Large Hadron Collider at next-to-leading order in QCD*, Phys. Rev. Lett. 109 (2011) 042001. arXiv:1112.3940, doi:10.1103/PhysRevLett.109.042001.

[32] R. Brun, F. Rademakers, *ROOT: An object-oriented data analysis framework*, Nucl. Instrum. Meth. A389 (1997) 81–86. doi:10.1016/S0168-9002(97)00048-X.

[33] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, D. Maître, *Vector Boson + Jets with BlackHat and Sherpa*, Nucl. Phys. Proc. Suppl. 205-206 (2010) 92–97. arXiv:1005.3728, doi:10.1016/j.nuclphysbps.2010.08.025.

[34] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, K. Ozeren, *Next-to-Leading Order $W + 5$-Jet Production at the LHC*, Phys. Rev. D88 (2013) 014025. arXiv:1304.1253, doi:10.1103/PhysRevD.88.014025.

[35] H. Ita, K. Ozeren, *Colour Decompositions of Multi-quark One-loop QCD Amplitudes*, JHEP 1202 (2012) 118. arXiv:1111.4193, doi:10.1007/JHEP02(2012)118.

[36] A. Martin, W. Stirling, R. Thorne, G. Watt, *Parton distributions for the LHC*, Eur. Phys. J. C63 (2009) 189–285. arXiv:0901.0002, doi:10.1140/epjc/s10052-009-1072-5.

[37] Z. Bern, G. Diana, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, S. Höche, H. Ita, D. A. Kosower, D. Maître, *Left-Handed W Bosons at the LHC*, Phys. Rev. D84 (2011) 034008. arXiv:1103.5445, doi:10.1103/PhysRevD.84.034008.

[38] Z. Nagy, Next-to-leading order calculation of three jet observables in hadron hadron collision, Phys.Rev. D68 (2003) 094002. arXiv:hep-ph/0307268, doi:10.1103/PhysRevD.68.094002.

[39] A. Buckley, J. Butterworth, L. Lönnblad, H. Hoeth, J. Monk, H. Schulz, J. E. von Seggern, F. Siegert, L. Sonnenschein, *Rivet user manual*, arXiv:1003.0694.

[40] M. R. Whalley, D. Bourilkov, R. C. Group, *The Les Houches Accord PDFs (LHAPDF) and Lhaglue*, arXiv:hep-ph/0508110.

[41] Z. Bern, G. Diana, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, K. Ozeren, *Missing Energy and Jets for Supersymmetry Searches*, Phys. Rev. D87 (2013) 034026. arXiv:1206.6064, doi:10.1103/PhysRevD.87.034026.

[42] G. Aad, et al., *Measurement of the production cross section for Z/γ^* in association with jets in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector*, Phys. Rev. D85 (2012) 032009. arXiv:1111.2690, doi:10.1103/PhysRevD.85.032009.

[43] G. Aad, et al., *Study of jets produced in association with a W boson in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector*, Phys. Rev. D85 (2012) 092002. arXiv:1201.1276, doi:10.1103/PhysRevD.85.092002.