CERN-LPCC-2020-002
FERMILAB-PUB-20-183-SCD-T
MCNET-20-15

Challenges in Monte Carlo event generator software for High-Luminosity LHC

The HSF Physics Event Generator WG · Andrea Valassi (editor)[1] · Efe Yazgan (editor)[2] · Josh McFayden (editor)[1,3] · Simone Amoroso[4] · Joshua Bendavid[1] · Andy Buckley[5] · Matteo Cacciari[6,7] · Taylor Childers[8] · Vitaliano Ciulli[9] · Rikkert Frederix[10] · Stefano Frixione[11] · Francesco Giuli[12] · Alexander Grohsjean[4] · Christian Gütschow[13] · Stefan Höche[14] · Walter Hopkins[8] · Dmitri Konstantinov[15] · Frank Krauss[16] · Qiang Li[17] · Leif Lönnblad[10] · Fabio Maltoni[18,19] · Michelangelo Mangano[1] · Zach Marshall[3] · Olivier Mattelaer[19] · Javier Fernandez Menendez[20] · Stephen Mrenna[14] · Servesh Muralidharan[1,8] · Tobias Neumann[13,21] · Simon Plätzer[22] · Stefan Prestel[10] · Stefan Roiser[1] · Marek Schönherr[16] · Holger Schulz[23] · Markus Schulz[1] · Elizabeth Sexton-Kennedy[14] · Frank Siegert[24] · Andrzej Siódmok[25] · Graeme A. Stewart[1]

Version 2.0 (18 May 2020)

Abstract We review the main software and computing challenges for the Monte Carlo physics event generators used by the LHC experiments, in view of the High-Luminosity LHC (HL-LHC) physics programme. This paper has been prepared by the HEP Software Foundation (HSF) Physics Event Generator Working Group as an input to the upcoming LHCC review of HL-LHC computing, which is scheduled to start in May 2020.

Keywords Monte Carlo · Physics Event Generator · LHC experiments · WLCG · High-Luminosity LHC

✉ Andrea Valassi
andrea.valassi@cern.ch
✉ Efe Yazgan
efe.yazgan@cern.ch
✉ Josh McFayden
mcfayden@cern.ch

[1]CERN, Geneva, Switzerland
[2]National Taiwan University, Taipei, Taiwan
[3]Lawrence Berkeley National Laboratory, USA
[4]DESY, Hamburg, Germany
[5]University of Glasgow, UK
[6]LPTHE, Sorbonne Université and CNRS, Paris, France
[7]Université de Paris, France
[8]Argonne National Laboratory, USA
[9]INFN and Università di Firenze, Italy
[10]Lund University, Sweden
[11]INFN and Università di Genova, Italy
[12]INFN and Università Tor Vergata, Roma, Italy
[13]University College London, UK
[14]Fermi National Accelerator Laboratory, USA
[15]NRC "Kurchatov Institute" - IHEP, Protvino, Russia
[16]University of Durham, UK
[17]Peking University, Beijing, China
[18]Università di Bologna, Italy
[19]Université Catholique de Louvain, Belgium
[20]Universidad de Oviedo, Spain
[21]Illinois Institute of Technology, Chicago, USA
[22]University of Vienna, Austria
[23]University of Cincinnati, USA
[24]Technische Universität, Dresden, Germany
[25]Institute of Nuclear Physics, Krakow, Poland

1 Introduction

Physics event generators are one of the computational pillars of any High Energy Physics (HEP) experiment, and in particular of the Large Hadron Collider (LHC) experiments. In this paper, we review the main software and computing challenges for the physics event generators used by the ALICE [1], ATLAS [2], CMS [3] and LHCb [4] experiments, in view of the high-luminosity running phase of the LHC experimental programme (HL-LHC), which should be operational from the end of 2027 [5]. This document has been prepared by the Physics Event Generators Working Group (WG) [6] of the HEP Software Foundation (HSF), as an input to the upcoming review of the HL-LHC computing strategy by the LHC Experiments Committee (LHCC) [7], which is scheduled to start in May 2020 [8–11]. As is the case for the LHCC review, this paper focuses mainly on ATLAS and CMS, but it also contains important considerations for ALICE and LHCb. A shorter summary of this paper, due to space constraints, is included in a more general document [12] prepared by the HSF for the LHCC, which covers the status and challenges in the broader area of common tools and community software.

This paper gives an overview of the many challenges in the generator area, and of the work that can be done to address them. Its outline is the following. Section 2 gives an overview of the role and challenges of physics event generators in LHC computing, summarising the steps which led to the creation of the HSF generator WG, and its current activities. Section 3 describes the collaborative challenges in the development, use and maintenance of generator software for LHC physics. Section 4 gives more details about the computational anatomy of physics event generators, and the technical challenges in their development and performance optimization. Section 5 summarizes some of the main open questions about the required physics accuracy of event generators at HL-LHC, and their impact on computational costs. Finally, in Sec. 6 we compile a list of high-priority items on which we propose that the HSF generator WG should focus, in view of the more in-depth LHCC review of HL-LHC software that is currently scheduled for Q3 2021 [11].

It should be stressed that this paper focuses on the software and computing aspects of event generators, rather than on the underlying physics. To be able to describe the overall computational goals and structure of these software applications and put them in context, many of the relevant physics concepts are in any case mentioned and briefly explained. This is done using a language that tries to be somewhat accessible also to software engineers and computing experts with no background in particle physics, even if the resulting text is not meant to be an exhaustive overview of these complex issues from a theoretical point of view.

This paper also gives an overview of some of the collaborative challenges and human resource issues related to the development and support of generator software. To some extent, these concerns have already been raised in previous community efforts, such as the HSF Community White Paper (CWP) [13] and the document [14] that was submitted as an input to the Open Symposium [15] on the Update of European Strategy for Particle Physics.

2 The HSF Physics Event Generator WG

Physics event generators are an essential component of the data processing and analysis chain of the LHC experiments, and a large consumer of resources in the Worldwide LHC Computing Grid (WLCG) [16]. All of the scientific results of the LHC experiments, such as precision measurements of physics parameters and searches for new physics, depend significantly on the comparison of experimental measurements to theoretical predictions, in most cases computed using generator software.

Using Monte Carlo (MC) techniques, generators allow both the calculation of differential and total cross sections and the generation of weighted or unweighted events for experimental studies (this is explained in more detail in Sec. 4.1, where these concepts are briefly defined). Within the experiments, generators are used primarily to produce large samples of (mostly unweighted) events: this is the first step in the production chain for simulating LHC collisions, which is followed by detector simulation and event reconstruction. In each of the two general purpose LHC experiments, ATLAS and CMS, the overall number of events that are generated by the central production teams and passed through full detector simulation and event reconstruction, across all relevant physics processes, is of the order of magnitude of $O(10^{10})$ events for every year of LHC data taking. Typically, the sizes of these samples of simulated events are approximately a factor of 3 larger than the overall number of data events collected during the corresponding time range. These large-scale event generation campaigns have a computational cost, mainly in terms of the "compute" (i.e. CPU) resources used, the majority of which are provided by the WLCG infrastructure. The limited size of the simulated samples that can be produced under resource constraints is a source of major uncertainty in many analyses (for example, in Higgs measurements of both ATLAS [17] and CMS [18]). This is an issue which is limiting the potential physics output of the LHC programme, and may get significantly worse at HL-LHC, where the projected computing needs of the experiments exceed the resources that are expected to be available [13], despite the fact that the most aggressive HL-LHC physics projections [19–22] assume no uncertainty due to the limited size of simulated samples.

When the HEP Software Foundation prepared its CWP [13] in 2017, the fraction of the ATLAS CPU resources in WLCG used for event generation was estimated [23] at around 20%. Beyond the existing projections, which assume the same level of theoretical precision as in the current event generation campaigns, concern was also raised that event generation would become computationally more expensive at the HL-LHC, where more complex calculations (e.g. beyond next-to-leading-order or with higher jet multiplicities) will be needed [24]. It was thus clear that speedups in generator software are needed to address the overall computing resource problem expected at the HL-LHC. This is of course also the case for the other big consumers of CPU (detector simulation and reconstruction), but until now these areas have had more focus, and significant speedups are already expected on the HL-LHC timescales, which has not been the case for generators. Other issues in the generator area, both technical and non-technical (e.g. funding, training and careers) also became obvious while preparing the CWP.

For these reasons, the HSF organised a three-day Workshop [25,26] at the end of 2018 to focus on the software and computing aspects of event generators. Their usage in the experiments was reviewed, revealing a large discrepancy in the CPU budgets quoted by ATLAS and CMS, 14% and 1%, respectively, for 2017 [27]. This was attributed, at least partly, to the different packages and parameter settings used by the two experiments, but it was clear that further studies were needed.

A Working Group of the HSF on Physics Event Generators [6] was therefore set up at the beginning of 2019. The main focus of the WG so far has been to get a better understanding of the current usage of generators in the experiments, and to identify and prioritise the areas where computing costs can be reduced. In particular, the ATLAS and CMS compute budgets have been analysed in detail: currently, it is estimated that the fractions of WLCG compute allocations used for generation today are around 12% for ATLAS and 5% for CMS. In absolute terms, i.e. in HEP-SPEC06 (HS06) seconds [28, 29], the ratio between ATLAS and CMS is actually larger, as the overall ATLAS budget for compute resources is larger than that of CMS. To understand what causes this difference, detailed benchmarking of the computational costs of Sherpa [30] and MadGraph5_aMC@NLO [31] (in the following abbreviated as MG5_aMC) have also started [32, 33], as these are the two generators used for some of the most expensive event generation productions in ATLAS and CMS, respectively. The WG has also been active in other areas, such as in discussing the possible sharing of common parton-level samples by ATLAS and CMS [34], and in reviewing and supporting the efforts for porting generators to modern architectures, notably GPUs. This last activity is particularly important, as it has become increasingly clear that being able to run compute-intensive WLCG software workloads on GPUs [35] would allow the exploitation of modern GPU-based supercomputers at High Performance Computing (HPC) centers, and generators look like a natural candidate for this, as discussed later on in Sec. 4.4.

Looking forward, the WG plans to continue its activities in the areas described above, but also to expand it in a few other directions. One of the goals of this paper is that of dissecting and analysing the many different challenges, both technical and non-technical, in the generator domain, to identify the specific areas where work is most urgently needed, or where the largest improvements are expected to be possible to reduce the gap between required and available computing resources at the time of HL-LHC. It should also be pointed out that the role of the WG in this context is mainly that of providing a forum for information exchange, and possibly supporting and coordinating common activities involving the collaboration of several teams or the comparison of their results, but most of the concrete work is generally expected to be done by the individual experiments or theoretical physicist teams.

3 Collaborative challenges

In this section, we give an overview of the collaboration challenges in the development, use and maintenance of generator software for LHC. By and large, these are mainly non-technical challenges that concern human resources, i.e. actual people, and their organisation, training and motivation, rather than computing resources, software modules or theoretical physics models.

3.1 A very diverse software landscape

The landscape of generator software is extremely varied, even more than in detector simulation, event reconstruction or analysis workloads. For a review, see for instance Refs. [14, 36–38]. Different generators (Sherpa, MG5_aMC, POWHEG [39], Pythia [40], Herwig [41–43], Alpgen [44], etc.) are used in the community, mainly for two reasons: firstly, one needs multiple independent calculations with potentially different approximations to cross-check one another; and secondly, the different generators vary in their features (for example, some might simulate only a subset of the physics processes of interest). A given process may be simulated with a different physics precision, e.g. leading-order (LO), next-to-leading-order (NLO), or next-to-next-to-leading-order (NNLO) in a power series expansion in the strong-force "coupling constant". Generating a sample also involves choices of hadronization and parton shower (PS) models (Pythia [40], Herwig [41–43], Ariadne [45], etc.), underlying event tunes [46–49], prescriptions for matching/merging[1] (MC@NLO [50], POWHEG [51], KrkNLO [52], CKKW [53], CKKW-L [54], MLM [55, 56], MEPS@NLO [57], MINLO [58], FxFx [59],

[1] In this paper, we use the definitions of matching and merging given in Ref. [59], which are briefly hinted at in Sec. 4.2.

UNLOPS [60], Herwig7 Matchbox [61–63], etc.), "afterburner" tools for simulating particle decays and quantum electrodynamics (QED) radiative corrections (EvtGen [64], Tauola [65], Photos [66], etc.), and other input parameters such as parton distribution functions (PDFs) [67], primarily via the LHAPDF library [68].

Various combinations of software libraries are thus possible, often written by different authors and some dating back many years, reflecting theoretical research within different teams. For a given process, the LHC experiments often use different software packages and settings from one another, and a single experiment can generate events using more than one choice. Many different packages and configurations may therefore need to be worked on to get cumulative CPU cost reductions. The large number of packages also complicates their long-term maintenance and integration in the experiments software and workflows, sometimes leading to Grid job failures and computing inefficiencies. Other packages are also absolutely critical for the whole generator community and must be maintained, even if their CPU cost is relatively low (Rivet [69], Professor [70], HepMC [71, 72], FastJet [73], etc.).

3.2 A very diverse human environment

A broad spectrum of skills and profiles are needed for the development and support of event generators: theorists (who create fundamental physics models, and design, develop and optimize most generator code); experimentalists working on research (who determine which types of event samples are required, and of which size); experimentalists working on computing (who implement, monitor and account execution of workflows on computing resources); software engineers and system performance experts (who may help to analyse and improve the efficiency of software applications and deployment models). This is a richness and opportunity, as some technical problems are best addressed by people with specific skills, but it also poses some challenges.

Training challenges. Theorists and experimentalists often lack formal training in software development and optimization. Software engineers, but also many experimentalists, are not experts in the theoretical physics models implemented in MC codes.

Communication challenges. It is difficult to find a shared terminology and set of concepts to understand one another: notions and practices that are taken for granted in one domain may be obscure for others. An example: there are many articles about the physics in generators, but software engineers would need papers describing the main software modules and overall data and control flow. Similarly, there are only very few articles where the experiments describe the software and computing workflows of their large scale MC productions (Ref. [74] is one such example for LHCb).

Career challenges. Those working in the development, optimization and execution of generator software provide essential contributions to the success of the HL-LHC physics programme and it is critical that they get the right recognition and motivation. However, theorists get recognition on published papers (which are often not even cited properly), and may not be motivated to work on software optimizations that do not have enough theoretical physics content to advance their careers. Generator support tasks in the experiments may also not be valued enough to secure jobs or funding to experimentalists pursuing a career in research.

Mismatch in usage patterns and in optimization focus. The way generators are built and used by their authors is often different from the way in which they are deployed and integrated by the experiments in their software frameworks and computing infrastructure. The goals and metrics of software optimization work may also differ. Theorists, who typically work with weighted events and fast detector parametrizations if any, are mainly interested in calculating cross sections and focus on minimising the phase space integration time for a given statistical precision. The LHC experiments typically run large scale productions for generating fully exclusive events, which are mostly unweighted as they must be processed through expensive detector simulation and event reconstruction steps: therefore, they need to maximize the throughput of events generated per unit time on a given computing system.

Programming languages. Attracting collaborators with a computer science background to work on generators, especially students, may also be complicated by the fact that critical components of some generator packages are written in Fortran, which is rarely used in industry and less popular among developers than other programming languages. Some of the generators also do not use industry standard version control systems, making it harder to contribute code.

4 Technical challenges

In this section, we give more details about the technical challenges in the software development and performance optimization of MC physics event generator codes. To this end, it is useful to first give a brief, high-level, reminder of their computational goals and internal data flows, and of the typical production workflows used by the experiments.

4.1 Computational anatomy of a MC event generator

Particle physics is based on quantum mechanics, whose description of Nature is intrinsically probabilistic. The predictions of HEP theoretical models that are numerically computed in event generators (through a combination of quantum field theory methodologies and phenomenological approximations), and which can be compared to experimental measurements, ultimately consist of probabilities and probability density functions.

In particular, the probability that a collision "event" with a given "final state", i.e. including n particles of given types, is observed in the collision of the LHC proton beams, is expressed in HEP in terms of the concept of a "cross section". In general terms, a cross section σ represents the number of events $N_{\rm exp} = \sigma\mathcal{L}$ that are expected per unit "integrated luminosity" $\mathcal{L}$ of the colliding beams (a parameter that depends on their intensities and geometries, and on the overall duration of data-taking time). More in detail, a differential cross section, $\frac{d\sigma}{dO}$, with respect to an observable O (such as a rapidity or a transverse momentum), refers to the observation of the desired final state at different points dO of the observable "phase space"; conversely, its integral $\sigma = \int_{\Omega_O} \frac{d\sigma}{dO} dO$ is referred to as the total cross section, if over the entire phase space, or as a fiducial cross section, if over a well delimited region Ω_O of the phase space (the so-called acceptance).

In this context, the computational core of a physics event generator is the code that numerically calculates, from first principles, the fully differential cross section $\frac{d\sigma}{d\Phi_n}(\mathbf{x})$ for the highest-energy interaction in the scattering process that leads to the desired n-particle final state; this is computed as a function of the complete kinematical configuration $\mathbf{x}$ of the elementary particles, or "partons", involved in this "hard interaction" for an individual collision event. In the majority of cases, the calculation of $\frac{d\sigma}{d\Phi_n}$ is implemented by identifying all Feynman diagrams contributing to this process, and calculating the "invariant amplitude" or "matrix element" (ME) for each of these diagrams (although there are also generators where matrix elements are computed using algorithms not based on Feynman diagrams [44, 75]).

For LHC processes, the kinematical configuration $\mathbf{x} = \{x_1, x_2, \Phi_n\}$ of a collision event essentially consists of a vector Φ_n, including up to four real numbers (related to their energy, mass and directions) for each of the n outgoing (final state) partons, and of two real numbers x_1 and x_2 representing the momentum fractions of the two incoming (initial state) partons. As described later on in Eq. 1, $\frac{d\sigma}{d\Phi_n}(\mathbf{x})$ is, together with two parton distribution functions $p(x_1)$ and $p(x_2)$, the central ingredient in the computation of a function $f(\mathbf{x})$, which essentially describes the probability distribution in the space of all possible kinematical configurations $\mathbf{x}$, and from whose integral in this space other relevant cross sections may be computed, $\sigma = \int f(\mathbf{x}) d\mathbf{x}$.

Integration and unweighted event generation. Given the function $f(\mathbf{x})$, physics event generators are commonly used in HEP to solve two types of computational problems, which are related to each other and generally addressed within a same execution of the software, as discussed more in detail later on. The first goal ("phase space integration") is to compute a cross section as the integral of $f(\mathbf{x})$ over the relevant phase space region. The second goal ("unweighted event generation") is to draw random samples of events whose kinematical configurations $\mathbf{x}$ are distributed according to the theoretical prediction $f(\mathbf{x})$.

Both of these goals are achieved using Monte Carlo (MC) methods, whose distinctive feature is their reliance on random number generation (see Refs. [76, 77] for early reviews of this technique in HEP). In particular, the starting point of both MC phase space integration and MC unweighted event generation is the calculation of $f(\mathbf{x})$ for a large sample of events $\mathbf{x}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, drawn at random from a known probability density function $g(\mathbf{x})$. More specifically:

1. MC phase space integration consists in drawing a random sample of events $\mathbf{x}_i$ from the sampling function $g(\mathbf{x})$, and in numerically calculating an estimator of the integral $\sigma = \int f(\mathbf{x}) d\mathbf{x}$, as the average of the "weight" $w_i = w(\mathbf{x}_i) = f(\mathbf{x}_i)/g(\mathbf{x}_i)$ for all the events $\mathbf{x}_i$ in the sample. It should be noted that this is not a deterministic approach, in the sense that the result of the calculation may change if a different random sample is used: it is easy to show, however, that the estimator is unbiased, and that its variance decreases as $1/N$ if the number of events N in the sample is increased. From a software point of view, the output of MC phase space integration is essentially only one number, the estimate of the integral $\sigma = \int f(\mathbf{x}) d\mathbf{x}$; alternatively, several numbers may also be calculated, representing the values of $\frac{d\sigma}{dO}$ computed as the MC integral of $f(\mathbf{x})$ over different regions of phase space. As discussed later on, a "phase space integration" phase is in any case also needed in the software before unweighted event generation, to iteratively optimize the choice of the

sampling function $g(\mathbf{x})$, and to compute the maximum value $w_{\max}$ of $w(\mathbf{x})$ over the relevant region of phase space.

2. MC unweighted event generation consists in drawing a random sample of events $\mathbf{x}_i$ from the sampling function $g(\mathbf{x})$, and in randomly rejecting some of them depending on the ratio of $w(\mathbf{x}_i)$ to the maximum weight $w_{\max}$ over the phase space. For each event, an accept-or-reject (or "hit-or-miss") decision is taken by drawing a random number R uniformly distributed between 0 and 1: the event is accepted if $R < w(\mathbf{x}_i)/w_{\max}$, and rejected otherwise. The resulting events, whose distribution is now described by $f(\mathbf{x})$ rather than by $g(\mathbf{x})$, are referred to as "unweighted" in the sense that they all have the same weight, which by convention is equal to 1. A special case of unweighting, producing events whose weights can be either +1 or -1, exists for calculations leading to events with negative weights: this is described later on. From a software point of view, the output of MC unweighted event generation is a sample of events, i.e. essentially a sample of vectors $\mathbf{x}_i$.

The choice of the sampling algorithm (e.g. VEGAS [78, 79]), or equivalently of the function $g(\mathbf{x})$, is very important. The closer $g(\mathbf{x})$ is to $f(\mathbf{x})$, that is to say the more constant the weight $f(\mathbf{x})/g(\mathbf{x})$ is over the entire phase space, the more precise is the integration (i.e. the lower the variance on the result) for a given sample size, and the more efficient is the unweighting procedure (i.e. the lower the fraction of events rejected).

It should be noted that the experiments also do physics analysis with samples of weighted events, which they produce for instance through "biasing" techniques, as discussed in Sec. 4.2. Wherever possible, however, unweighted events (and in particular events with a positive weight +1) are preferred, as smaller event samples are required than when using events with non-uniform weights, resulting in overall savings of compute and storage resources.

Internal software workflow. Schematically, the internal software workflow of a typical generator is the following: first, when necessary (i.e. when the process is too complex to be manually hardcoded in advance), the source code to compute the differential cross section $\frac{d\sigma}{d\Phi_n}$ of the hard process, which is needed to derive $f(\mathbf{x})$, is produced through automatic code generation, after identifying the relevant Feynman diagrams; a phase space integration step follows, where event samples are iteratively drawn to optimize the sampling function $g(\mathbf{x})$, and estimate the maximum weight $w_{\max}$; parton-level unweighted events are then generated using the final, frozen, $g(\mathbf{x})$ and $w_{\max}$; parton showers, hadronization and hadron decays to stable particles are finally applied on top of those "parton-level" events. During the unweighted event generation step, "merging" prescriptions are also applied, after parton showers and before hadronization, for so-called "merged" or "multi-leg" setups, that is to say if the required final state includes a variable number of "jets" (i.e. of quarks or gluons) n_{jets} between 0 and n; experiment-level filters and other techniques such as forced decays or forced fragmentation may also be applied, for instance to produce event samples containing specific decays of B hadrons.

The internal workflow of a generator application is actually more complex than described above, because many different hard interactions may contribute to the simulated process. To start with, for hadron colliders like the LHC, the hard interaction takes place not between two protons, but between two of the partons in their internal substructure (quarks of different flavors, and gluons): this implies that separate integrals for all possible types of initial state partons, using different sets of diagrams and of functions $f(\mathbf{x})$, must be considered. Using the factorisation theorem [80], which allows separating perturbative (i.e. ME) and non-perturbative (parton distribution function) calculations in quantum chromodynamics (QCD), the total cross section may be written [36] as

$$\sigma = \sum_{a,b} \int dx_1 p_a(x_1) \int dx_2 p_b(x_2) \int d\Phi_n \frac{d\sigma_{ab}}{d\Phi_n}(x_1, x_2, \Phi_n), \quad (1)$$

i.e. as the convolution, by the appropriate parton distribution functions $p_a(x_1)$ and $p_b(x_2)$, of the differential cross section $\frac{d\sigma_{ab}}{d\Phi_n}$ for the production of n final state particles with properties Φ_n, in the hard interaction of two partons of types a and b with momentum fractions x_1 and x_2, respectively.

In addition, NLO calculations imply the need to compute two separate classes of integrals, which involve two different classes of Feynman diagrams and of functions $f(\mathbf{x})$, because matrix elements need to be separately computed for standard "$\mathbb{S}$-events" and hard "$\mathbb{H}$-events" [50], i.e. for final states with n body kinematics Φ_n (at tree level and one loop) and n+1 body kinematics Φ_{n+1} (at tree level), respectively; "matching" prescriptions are then needed to ensure that parton showers are used appropriately in both types of events (see also for instance Refs. [81–83] for detailed presentations that include a graphical representation of these issues). The situation is similar to that of NLO calculations, and even more complex, in NNLO calculations.

Experiment production workflows. Phase space integration is a resource intensive step, but in many cases it is only executed once in a given experiment production; this is known as the creation of "gridpacks" in MG5_aMC and POWHEG, or "sherpacks" or "integration grids" in Sherpa. For instance, creating a typical MG5_aMC gridpack for V+jets at NLO may take up to several weeks on one multi-core node, or up to several days in a typical cluster usage scenario; see also Ref. [84] for further details about how gridpacks are used in CMS. The generation of unweighted event samples, conversely, is where the LHC experiments spend essentially all of their generator CPU budgets: this typically involves many Grid jobs submitted in parallel with different random number seeds and thus unrelated to one another, each storing events on its own output file. If available, integration grids are used as inputs by these jobs, but for simpler processes, or for generators lacking the option to create integration grids, every Grid job may also go through the whole event generation chain, including both phase space integration and unweighted event generation.

Computational costs. The computational cost of a MC application roughly scales with the number of points $\mathbf{x}$ where the function $f(\mathbf{x})$ is computed, i.e. with the number of events used for the integration phase and with the overall number of events drawn prior to unweighting during the event generation phase. As a consequence, the most obvious approach to reduce the overall computational cost of event generation is simply to try and decrease the number of points $\mathbf{x}$ for which $f(\mathbf{x})$ is computed. This is described in detail in Sec. 4.2, where the possible reduction of many large inefficiencies in unweighted event generation is discussed, as well as possible strategies for reusing events for more than one goal.

In addition, the intrinsic cost per event of computing $f(\mathbf{x})$ approximately scales itself with the number of Feynman diagrams contributing to that process. In particular, with respect to LO calculations for a given process, NLO and especially NNLO calculations for the same process involve much higher numbers of diagrams, some of which ("loop diagrams") are also intrinsically more complex to compute. Matrix element calculations are in fact performed as a power series expansion in terms of the strong-force coupling constant α_s (which is smaller than 1); the difference between LO, NLO and NNLO calculations is primarily that of considering the following level in this power series expansion, which leads to a roughly factorial increase in computational complexity. It should be pointed out, nevertheless, that NLO calculations for simple processes with low final state multiplicities may be computationally cheaper than LO calculations for complex processes with high final state multiplicities. In summary, it would thus seem that the intrinsic cost per event $\mathbf{x}$ of computing $f(\mathbf{x})$ is to some extent incompressible, because of the relatively fixed amount of arithmetic calculations that this involves. One of the only obvious strategies for reducing this cost consists in improving the efficiency with which these arithmetic operations are performed on modern computing systems, for instance through the use of parallel programming techniques, as discussed later in Sec. 4.4. In addition, radically new approaches are also being worked on, involving for example the approximation of matrix element calculations using Machine Learning (ML) regression methods [85, 86].

4.2 Inefficiencies in unweighted event generation

The complex workflow described above presents several challenges and opportunities for improvement. To start with, there are many sources of inefficiency in unweighted event generation, as discussed in the following.

Phase space sampling inefficiency. The algorithm used for phase space sampling is the most critical ingredient for efficient unweighted event generation. Some basic techniques, such as stratified sampling, which essentially consists in binning the phase space, and importance sampling, which is often implemented as a change of variables to parametrize the phase space, date back to more than 40 years ago [77]. Many algorithms, most notably VEGAS [78, 79] or MISER [87], are adaptive, i.e. recursive, in that their parameters are tuned iteratively as the shape of $f(\mathbf{x})$ is learnt by randomly drawing more and more phase space points. Adaptive multichannel algorithms [88, 89] are often used to address the complex peaking structures of LHC processes, by defining the sampling function $g(\mathbf{x})$ as a weighted sum of functions, each of which essentially describes a different peak. Many generic sampling algorithms exist, including very simple ones like RAMBO [90], others derived from VEGAS such as BASES/SPRING [91, 92] or MINT [93], and cellular algorithms like FOAM [94]. Other sampling algorithms have been developed specifically for a given generator: examples include MadEvent [95] and VAMP [96], which are based on modified versions of VEGAS and are used in the MG5_aMC and WHIZARD [97] generators, respectively, as well as COMIX [98], which is used in Sherpa.

In general, the larger the dimensionality of the phase space, the lower the unweighting efficiency that can be achieved: in W+jets, for instance, the Sherpa efficiency [99] is 30% for W+0jets and 0.08% for W+3jets. This is an area where research is very active, and should be actively encouraged, as significant cost reductions in WLCG compute budgets could be achieved. Improvements in this area can only start from physics-motivated approaches based on the knowledge of phase space peaks, but they can be complemented by brute-force ML algorithmic methods [99–104], therefore people with different profiles can contribute to this area. The use of one of these ML tools, Generative Adversarial Networks (GAN), is being investigated [105] not only as a way to provide a more efficient phase space sampling, but also as a possible replacement for unweighted event generation altogether, for example when complemented with maximum mean discrepancy methods [106].

In this context, it is useful to point out that maximizing the efficiency of unweighted event generation and minimizing the variance on total cross section predictions by MC integration represent two different, even if closely related, strategies for the optimization of the phase space sampling algorithm. The two strategies imply the use of different loss metrics during the learning phase of an algorithm, and result in different weight distributions. This is discussed in detail in Ref. [94], and to some extent also in Ref. [96].

Slicing and biasing. A further issue [107], somewhat related to sampling inefficiencies, is that jet production cross sections fall very sharply as the transverse momenta ($\mathrm{p_T}$) of the leading jets increase, and generating events with uniform weight generally fails to give a reasonable yield in the high-$\mathrm{p_T}$ regions of phase space. One approach to solving this problem ("slicing") is to produce several independent samples of events, using different generation cuts in each one, in order to populate all the regions of interest. An additional approach ("biasing" or "enhancement"), available for instance in POWHEG [107], MG5_aMC [108, 109], Sherpa [110], Pythia8 [111] and Herwig7.1 [43], consists in generating samples of events with non uniform weights, the shape of whose distribution can however be controlled by user-defined suppression factors. Both approaches are used in practice by the LHC experiments, as each has its pros and cons, and both reduce the resources required to populate the low-statistics tails of distributions. With additional work, these methods could help reduce the overall event generation resource requirements at HL-LHC.

Merging inefficiency. Merging prescriptions (e.g. MLM, CKKW-L at LO, and FxFx, MEPS@NLO at NLO) imply the rejection of some events to avoid double counting, between events produced with $n+1$ jets in the matrix element, and events produced with n jets in the matrix element and one jet from the parton shower [56]. The resulting inefficiencies can be relatively low depending on the process, but they are unavoidable in the algorithmic strategy used by the underlying physics modeling. Some of these issues are discussed in Ref. [112], which shows for instance that a method like shower-$\mathrm{k_T}$ MLM can reduce the merging inefficiency of MLM.

Filtering inefficiency. An additional large source of inefficiency is due to the way the experiments simulate some processes, where they generate large inclusive event samples, which are then filtered on final-state criteria to decide which events are passed on to detector simulation and reconstruction (e.g. CMS simulations of specific Λ_B decays have a 0.01% efficiency, and ATLAS B-hadron filtering in a V+jets sample has ∼10% efficiency). This inefficiency could be reduced by developing filtering tools within the generators themselves, designed for compatibility with the requirements of the experiments. A particularly wasteful example is where events are separated into orthogonal subsamples by filtering, in which case the same large inclusive sample is generated many times, once for each filtering stream: allowing a single inclusive event generation to be filtered into several orthogonal output streams would improve efficiency. Filtering is an area where the LHCb collaboration has a lot of experience and already obtained significant speedups through various techniques. In this context, one should also note that the speed of color reconnection algorithms [113, 114] is a limiting factor for simulating rare hadron decays in LHCb.

Sample sharing. In addition to removing inefficiencies, other ways could be explored to make maximal use of the CPU spent for generation by reusing samples for more than one purpose. Sharing parton-level, or even particle-level, samples between ATLAS and CMS is being discussed for some physics analyses. However, the implications of the statistical correlations that this would introduce need further investigation in the context of combinations of results across experiments.

Sample reweighting. Another way to re-use samples is through event reweighting. Recently, there have been major improvements in available tools in this area [31, 115–119], which have made it possible to obtain systematic uncertainty variations as well as reweighting to alternative model parameters. The latter may be useful for example in new physics searches, but also in the optimization of experimental measurements of model parameters [120]. This machinery is particularly important because in the past obtaining these variations would have required multiple samples to go through detector simulation and reconstruction, whereas the reweighting only requires this overhead for a single sample that can then be reused in multiple ways. This significantly reduces the CPU and storage requirements for the same end result. However, this issue can still be explored further as in some areas there are limitations to the validity of these reweighting schemes [117–119, 121, 122]. In addition, some systematic uncertainty variations, such as merging scale variations, are not yet available as weights but there is work ongoing. There are also systematic variations such as changes of the hadronisation model which are not well suited to the type of event reweighting discussed here, but for which alternative approaches using ML techniques to train an ad-hoc reweighting between samples are under investigation [123–127].

Negative weights. In NLO calculations, matching prescriptions (e.g. MC@NLO, POWHEG, etc.) are required to avoid double counting between phase space configurations that may come both from $\mathbb{H}$-events and from $\mathbb{S}$-events with parton showers. The solution of this issue becomes technically even more complex at the NNLO. A widely used NLO matching prescription, MC@NLO [50], is implemented by using a "modified subtraction method" that may lead to the appearance of events with negative weights. A MC unweighting procedure is still applied, but the resulting events are "unweighted" in the sense that their weight can only be +1 or -1. This is a source of (possibly large) inefficiency, as larger event samples must be generated and passed through the experiment simulation, reconstruction and analysis software, increasing the compute and storage requirements. For a fraction r of events with weight -1, the number of events to generate increases by a factor $1/(1\text{-}2r)^2$, because the statistical error on MC predictions is a factor $1/(1\text{-}2r)$ higher; for a more detailed explanation of these formulas, see for instance Ref. [128]. For example, negative weight fractions equal to r=25% and r=40%, which may be regarded as worst-case scenarios occurring in $\mathrm{t\bar{t}}$ and $\mathrm{Hb\bar{b}}$ production [128], respectively, imply the need to generate 4 times and 25 times as many events.

Negative weights can instead be almost completely avoided, by design, in another popular NLO matching prescription, POWHEG [51], which however is only available for a limited number of processes. POWHEG describes the relevant physics in a different way with respect to MC@NLO, so that predictions which have formally the same level of accuracy may visibly differ in the two codes, and are associated with different systematics (see Ref. [128] for an in-depth discussion). Negative weights can also be avoided in the KrkNLO [52] matching prescription, which is based on a very different approach from those used by MC@NLO and POWHEG; this method however is only available for a limited number of processes, and so far has been rarely used in practice by the LHC experiments.

Progress in this area can only be achieved by theorists, and research is active in this area. For instance, a modified MC@NLO matching procedure with reduced negative weights, known as MC@NLO-Δ, has recently been proposed [128]. Similarly, techniques to significantly reduce the negative weight fraction are also available in Sherpa [86]. Negative weights also exist for NNLO calculations, for instance in the UN2LOPS prescription [129].

One should also note that negative weights due to matching are absent in LO calculations. One possibility for avoiding negative weights, while possibly still achieving a precision beyond LO, could then be to generate LO multi-leg setups and reweight them to higher order predictions; a careful evaluation of the theoretical accuracy of this procedure would however be needed in this case. In addition, negative weights can also happen at LO because of not-definite-positive parton distribution function sets and interference terms, which is particularly relevant for effective field theory calculations.

Finally, it should be noted that developments to incorporate contributions in parton shower algorithms beyond the currently adopted approximations, see e.g. Refs. [130–132], very often necessitate weighted evolution algorithms. Overcoming the prohibitively broad weight distributions is subject to an ongoing development and might necessitate structural changes in the event generation workflow [133].

4.3 Accounting of compute budgets for generators

While progress has been made in the HSF generator WG to better understand which areas of generator software have the highest computational cost, more detailed accounting of the experiment workloads and profiling of the main generator software packages would help to further refine R&D priorities.

Accounting of CPU budgets in ATLAS/CMS. Thanks to a large effort from the generator teams in both experiments, a lot of insight into the settings used to support each experiment's physics programme was gained within the WG. It is now clear that the fraction of CPU that ATLAS spends for event generation is somewhat higher than that in CMS, although the difference is lower than previously thought: the latest preliminary estimates of these numbers are 12% and 5%, respectively. A more detailed study of the different strategies is ongoing, in particular by analysing individually the CPU costs of the main processes simulated (notably, V+jets, $t\bar{t}$, diboson and multijet).

A practical issue is that these figures had to be harvested from logs and production system databases a posteriori. Deriving precise numbers for CMS has been particularly difficult, requiring significant person hours to extract the required information, as until recently the generation (GEN) and detector simulation (SIM) steps were mixed in a single software application, and no separate accounting figures for GEN and SIM could be recovered from past job logs, therefore Grid costs had to be extrapolated from ad-hoc local tests. In addition, job monitoring information is presently kept for only 18 months in CMS, which complicates the analysis of past productions. For the future, it would be important to establish better mechanisms to collect this information, to allow for an easy comparison between different experiments. It would also help if the various types of efficiencies described above (sampling, merging and filtering) could be more easily retrieved for all simulated processes.

Profiling of generators using production setups. Another area where the WG has been active, but more work is needed, is the definition and profiling of standard generator setups, reproducing those used in production. This has been used to compare the speeds of Sherpa and MG5_aMC in the configurations used by ATLAS and CMS, respectively. For instance, Sherpa was found to be 3 to 8 times slower than MG5_aMC in the generation of NLO W+(0-2)jets, but the exact ratio depends on some of the model parameters used in Sherpa, e.g. the dynamical scale choice of Sherpa, which results in taking about 50% of the total CPU time for generation: when modifying Sherpa to use an equivalent scale to MG5_aMC, the CPU consumption for this process was reduced by over a factor of two. The choice of a scale, however, has important consequences not only on computational costs, but also on physics accuracy: an in-depth discussion of this important issue, which has been described in many research papers by different teams of theorists (see, for instance, Refs. [53, 54, 128, 134, 135]), is beyond the scope of this paper, but the WG will continue to investigate the computing and physics implications of such choices.

Detailed profiling of different generator setups has also already helped to assess the CPU cost of external PDF libraries, and to optimise their use [136]. The profiling of the memory footprint of the software would also be very useful, and may motivate in some cases a move to multithreading or multiprocessing approaches.

4.4 Modernisation of generator software

More generally, as is the case for many software packages in other areas of HEP, some R&D on generators would certainly be needed to modernise the software and make it more efficient, or even port it to more modern computing architectures (see also the discussion of these issues in the Snowmass 2013 report [137] and in the HSF CWP [13]).

Data parallelism, GPUs and vectorization. The data flow of an MC generator, where the same function $f(\mathbf{x})$, corresponding to the matrix element for the simulated HEP process, has to be computed over and over again at many phase space points $\mathbf{x}_i$, should, in principle, lend itself naturally to the data parallel approaches found in GPU compute kernels, and possibly to some extent in CPU vectorized code. In this respect, generators should be somewhat easier to reengineer efficiently for GPUs than detector simulation software (notably Geant4 [138]), where the abundance of conditional branching of a stochastic nature may lead to "thread divergence" and poor software performance (see, for examples, Refs. [139–144]).

Porting and optimizing generators on GPUs is especially important to be able to exploit modern GPU-based HPCs (such as SUMMIT [145], where 95% of the compute capacity comes from GPUs [146]). Some work in this direction was done in the past on MG5_aMC, including both a port to GPUs (HEGET [147–149]) of the library that was used in MG5_aMC, before ALOHA [150] was introduced, for the automatic generation of matrix element code (HELAS [151, 152]), and a port to GPUs of VEGAS and BASES (gVEGAS and gBASES [153, 154]). This effort, which unfortunately never reached production quality, is now being revamped by the WG, in collaboration with the MG5_aMC team, and represents one of the main R&D priorities of the WG. This work is presently focusing on Nvidia CUDA, but abstraction libraries like Alpaka [155,156] or oneAPI [157] will also be investigated.

GPUs may also be relevant to the ML-based phase space sampling algorithms discussed in Section 4.2; some recent work in this area has targeted GPUs explicitly [158, 159]. Finally, work is also ongoing [160] on the efficient exploitation of GPUs in the pseudorandom number generation libraries that are used in all MC generators (see Ref. [161] for a recent review of these components).

Task parallelism, multithreading, multiprocessing. Generators are generally executed as single-threaded, single-process software units. In most cases, this is not a problem, as the memory footprint of unweighted event generation is small and usually fits within the 2 GB per core available on WLCG nodes. However, there are cases (e.g. diboson production, or Z and Zγ+jets production with electroweak corrections, all with up to 4 additional jets) where memory requirement is higher than 2 GB and can be as much as 4 GB; this leads to inefficiencies as some processor cores remain unused, which could be avoided using multithreading approaches. The fact that some generators are not even thread safe may also be a problem, for instance to embed them in multi-threaded event processing frameworks, such as that of CMS.

Multi-processing approaches may also be useful to speed up the integration and optimization step for complex high-dimensional final states, or to reduce the overall memory footprint of a generator application. In particular, Sherpa workflows based on the Message Passing Interface (MPI) [162] which have been available for quite a long time, have been found very useful by ATLAS and CMS to speed up the preparation of integration grids on local batch clusters. A lot of work has also been done in recent years to implement and benchmark MPI-based workflows on HPC systems. For instance, the Sherpa LO-based generation of merged many-jet samples has been successfully tested [163] on the Cori [164] system at NERSC, both on traditional Intel Haswell CPUs and on many-core Intel Knights Landing (KNL) CPUs. This work has used a technique similar to that previously developed [165] for testing and benchmarking the scaling of the parallel execution of Alpgen on Mira [166] at ALCF, a supercomputer based on IBM PowerPC CPUs. New event formats, migrating LHEF [167] to HDF5, have also been instrumental in the success of the Cori tests. MPI integration has also been completed for MG5_aMC [168]. In this context, it should in any case be noted that, even if HPCs offer extremely high-speed inter-node connectivity, it is perfectly ok for WLCG workflows, including generators, to use these systems as clusters of unrelated nodes, if the computational workflow can be split up into independent tasks on those nodes.

Hybrid parallelization approaches are also possible, where multithreading or multiprocessing techniques are used internally on a single multi-core node, while the MPI protocol is used to manage the communication between distinct computing nodes. This approach is implemented for example in the WHIZARD [96] and MCFM [169] codes, both of which combine OpenMP [170] multithreading on individual multi-core nodes with MPI message passing between them.

Generic code optimizations. A speedup of generators may also be achievable by more generic optimizations, not involving concurrency. It should be studied, for instance, if data caching or different compilers and build strategies may lead to any improvements. Recent studies [136] on the way LHAPDF6 is used in Pythia have indeed resulted in significant speedups through better data caching.

5 Physics challenges (increasing precision)

In addition to software issues, important physics questions should also be addressed about more accurate theoretical predictions, above all NNLO QCD calculations, but also electroweak (EW) corrections, and their potential impact on the computational cost of event generators at HL-LHC. For a recent review of these issues, see for example Ref. [24]. Some specific NNLO calculations are already available and used today by the LHC experiments in their data analysis. For example, the measurements of fiducial $t\bar{t}$ cross sections, extrapolated to the full phase space, are compared to the predictions of TOP++ [171], accurate to NNLO: this program, however, does not use MC methods and cannot be used to generate unweighted events. Research on NNLO matching has also made significant progress, for example on the NNLOPS [172], GENEVA [173], UN2LOPS [129] and MINNLOPS [174] prescriptions. In addition, samples of unweighted events are routinely generated for Higgs boson final states using the POWHEG/MINLO NNLOPS approach [172, 175]. With a view to HL-LHC times, however, some open questions remain to be answered, as discussed below.

NNLO: status of theoretical physics research. The first question is for which processes QCD NNLO precision will be available at the time of the HL-LHC. For example, first results for triphoton results at NNLO have recently been published [176]: when would NNLO be expected for other 2$\rightarrow$3 processes or even higher multiplicity final states? Also, for final states such as $t\bar{t}$, where differential NNLO predictions exist [177, 178], but the generation of unweighted NNLO+PS events is not yet possible, when can this be expected? In particular, it would be important to clarify which are the theoretical and more practical challenges in these calculations, and the corresponding computational strategies and predicted impact on CPU time needs (e.g. more complex definition of matching procedures, higher fraction of negative weights, and more complex 2-loop MEs?).

The accuracy of shower generators is also important in this context. Current shower generators rely on first order splitting kernels, together with an appropriate scheme to handle soft emissions. Recent work aims at improving parton showers by increasing their accuracy either by developing novel shower schemes within the standard parton or dipole branching, such as DIRE [179] and Vincia [180] or by going beyond the typical probabilistic approach [181] and by incorporating higher order splitting functions [52, 130, 182, 183]. In addition, very recently, significant theoretical advance opening the way to NNL showers has been achieved [184].

To match NNLO accuracy in QCD, EW corrections must also be included. Recently, much progress has been achieved on the automation of the computation of EW corrections [185–188], to the point that fixed-order NLO QCD and EW corrections are readily available for any process of interest at the LHC. A general interface of these calculations to shower generators that correctly account for QED radiation for these computations, however, is not yet available.

An additional concern, in general but especially in higher-order phenomenology, is the control of numerical and methodological errors at the sub-percent level. This is relevant for processes where high-precision measurements and predictions are available, but also to efficiently and precisely test the input parameter dependence (PDFs, α_s, etc.). These issues, and the way in which they are addressed in the MCFM parton-level code, are discussed in detail in Ref. [169]. A key component of this code is a fully parallelized phase space integration, using both OpenMP and MPI on multi-core machines and cluster setups, where technical cutoffs can be controlled at the required level of precision.

NNLO: experimental requirements at HL-LHC. The second question is for which final states unweighted event generation with NNLO precision would actually be required ($t\bar{t}$ production is a clear candidate), and how many events would be needed. One should also ask if reweighting LO event samples to NNLO would not be an acceptable cheaper alternative to address the experimental needs, and what would be the theoretical accuracy reached by this procedure.

Size of unweighted event samples required at HL-LHC. Another question to be asked, unrelated to NNLO, is in which regions of phase space the number of unweighted events must be strictly proportional to the luminosity. For example, in the bulk (low p_T) regions of W boson production it is probably impossible to keep up with the data, due to the huge cross section. Alternative techniques could be investigated, to avoid the generation of huge samples of unweighted events.

6 Conclusions

This paper has been prepared by the HSF Physics Event Generator Working Group as an input to the upcoming LHCC review of HL-LHC computing, which is scheduled to start in May 2020. We have reviewed the main software and computing challenges for the Monte Carlo physics event generators used by the LHC experiments, in view of the HL-LHC physics programme.

Out of the many issues that we have described, we have identified the following five as the main priorities on which the WG should focus:

1. Gain a more detailed understanding of the current CPU costs by accounting and profiling.
2. Survey generator codes to understand the best way to move to GPUs and vectorized code, and prototype the port of the software to GPUs using data-parallel paradigms.
3. Support efforts to optimize phase space sampling and integration algorithms, including the use of Machine Learning techniques such as neural networks.
4. Promote research on how to reduce the cost associated with negative weight events, using new theoretical or experimental approaches.
5. Promote collaboration, training, funding and career opportunities in the generator area.

We plan to report on these issues in the more in-depth LHCC review of HL-LHC software, which is currently scheduled in Q3 2021, and reassess the WG priorities for future activities at that point in time.

Acknowledgements

This work 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). This research used resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE–AC02–07CH11359. This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231. F. Krauss acknowledges funding as Royal Society Wolfson Research fellow. M. Schönherr is funded by the Royal Society through a University Research Fellowship. E. Yazgan acknowledges funding from National Taiwan University grant NTU 109L104019.

References

1. ALICE Collaboration, *The ALICE experiment at the CERN LHC*, J. Instrum. **3** (2008) S08002. https://doi.org/10.1088/1748-0221/3/08/S08002
2. ATLAS Collaboration, *The ATLAS experiment at the CERN Large Hadron Collider*, J. Instrum. **3** (2008) S08003. https://doi.org/10.1088/1748-0221/3/08/S08003
3. CMS Collaboration, *The CMS experiment at the CERN LHC*, J. Instrum. **3** (2008) S08004. https://doi.org/10.1088/1748-0221/3/08/S08004
4. LHCb Collaboration, *The LHCb detector at the LHC*, J. Instrum. **3** (2008) S08005. https://doi.org/10.1088/1748-0221/3/08/S08005
5. High-Luminosity Large Hadron Collider (HL-LHC). https://home.cern/science/accelerators/high-luminosity-lhc
6. HSF Physics Event Generator Working Group. https://hepsoftwarefoundation.org/workinggroups/generators.html
7. LHC Experiments Committee. https://committees.web.cern.ch/lhcc
8. *Minutes of the 136th Meeting of the LHCC*, CERN-LHCC-2018-033. https://cds.cern.ch/record/2649242
9. *Minutes of the 139th Meeting of the LHCC*, CERN-LHCC-2019-010. https://cds.cern.ch/record/2689443
10. *Minutes of the 140th Meeting of the LHCC*, CERN-LHCC-2019-016. https://cds.cern.ch/record/2702745
11. *Minutes of the 141st Meeting of the LHCC*, CERN-LHCC-2020-003. https://cds.cern.ch/record/2711432
12. HEP Software Foundation, *Common Tools and Community Software*, input for the LHCC review of HL-LHC computing, HSF-DOC-2020-1 (2020). https://doi.org/10.5281/zenodo.3779249
13. HEP Software Foundation, *A Roadmap for HEP Software and Computing R&D for the 2020s*, Comput. Softw. Big Sci. **3** (2019) 7. https://doi.org/10.1007/s41781-018-0018-8
14. A. Buckley et al., *Monte Carlo event generators for high energy particle physics event simulation*, MCnet-19-02 (2019). https://arxiv.org/abs/1902.01674
15. CERN Council Open Symposium on the Update of European Strategy for Particle Physics, Granada (2019). https://cafpe.ugr.es/eppsu2019
16. Worldwide LHC Computing Grid. https://wlcg.web.cern.ch
17. ATLAS Collaboration, *Combined measurements of Higgs boson production and decay using up to 80 fb^{-1} of proton-proton collision data at $\sqrt{s}$=13 TeV collected with the ATLAS experiment*, Phys. Rev. D **101** (2020) 012002. https://doi.org/10.1103/PhysRevD.101.012002
18. CMS Collaboration, *Observation of Higgs boson decay to bottom quarks*, Phys. Rev. Lett. **121** (2018) 121801. https://doi.org/10.1103/PhysRevLett.121.121801
19. P. Azzi (ed.) et al., *Report from Working Group 1: Standard Model Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-03, HL/HE-LHC Workshop, CERN (2018). https://arxiv.org/abs/1902.04070
20. M. Cepeda (ed.) et al., *Report from Working Group 2: Higgs physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-04, HL/HE-LHC Workshop, CERN (2018). https://arxiv.org/abs/1902.00134
21. X. Cid Vidal (ed.) et al., *Report from Working Group 3: Beyond the Standard Model Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-05, HL/HE-LHC Workshop, CERN (2018). https://arxiv.org/abs/1812.07831
22. A. Cerri (ed.) et al., *Report from Working Group 4: Opportunities in Flavour Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-06, HL/HE-LHC Workshop, CERN (2018). https://arxiv.org/abs/1812.07638
23. R. Boughezal et al., *Generator and Theory Working Group Chapter for CWP*, unpublished draft (2017). https://github.com/HSF/documents/tree/master/CWP/papers/HSF-CWP-2017-11_generators
24. F. Maltoni, M. Schönherr, P. Nason, *Monte Carlo generators*, in Ref. [19].
25. HSF Physics Event Generator Computing Workshop, CERN (2018). https://indico.cern.ch/event/751693
26. A. Buckley, *Computational challenges for MC event generation*, ACAT 2019, Saas Fee, MCnet-19-18 (2019). https://arxiv.org/abs/1908.00167
27. L. Sexton-Kennedy, G. Stewart, *CWP challenges and workshop aims*, in Ref. [25]. https://indico.cern.ch/event/751693/contributions/3182927
28. M. Michelotto et al., *A comparison of HEP code with SPEC benchmarks on multi-core worker nodes*, Proc. CHEP2009, Prague, J. Phys. Conf. Ser. **219** (2010) 052009. https://doi.org/10.1088/1742-6596/219/5/052009
29. A. Valassi et al., *Using HEP experiment workflows for the benchmarking and accounting of WLCG computing resources*, submitted to Proc. CHEP2019, Adelaide. https://arxiv.org/abs/2004.01609
30. E. Bothmann et al., *Event generation with Sherpa 2.2*, SciPost Phys. **7** (2019) 034. https://doi.org/10.21468/SciPostPhys.7.3.034
31. J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP07(2014)079. https://doi.org/10.1007/JHEP07(2014)079
32. J. McFayden, *ATLAS needs and concerns*, in Ref. [25]. https://indico.cern.ch/event/751693/contributions/3182932
33. E. Yazgan, *CMS needs and concerns*, in Ref. [25]. https://indico.cern.ch/event/751693/contributions/3182936
34. ATLAS and CMS Collaborations, *Comparison of ATLAS and CMS nominal* $t\bar{t}$ *Monte Carlo simulation for Run 2*, CMS-DP-2019-011, CERN (2019). https://cds.cern.ch/record/2678959
35. A. Valassi, *Overview of the GPU efforts for WLCG production workloads*, Pre-GDB on benchmarking, CERN (2019). https://indico.cern.ch/event/739897/contributions/3559134
36. A. Buckley et al., *General-purpose event generators for LHC physics*, Physics Reports **504** (2011) 145. https://doi.org/10.1016/j.physrep.2011.03.005
37. T. Sjöstrand, *Introduction to Monte Carlo techniques in High Energy Physics*, CERN Summer Student Lectures (2012). https://indico.cern.ch/event/190076
38. T. Sjöstrand, *Status and developments of event generators*, Fourth LHC Physics Conference (LHCP2016), Lund (2016). https://arxiv.org/abs/1608.06425
39. S. Alioli et al., *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, JHEP06(2010)043. https://doi.org/10.1007/JHEP06(2010)043
40. T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, Comp. Phys. Comm. **191** (2015) 159. https://doi.org/10.1016/j.cpc.2015.01.024
41. M. Bähr et al., *Herwig++ physics and manual*, Eur. Phys. J. C **58** (2008) 639. https://doi.org/10.1140/epjc/s10052-008-0798-9

42. J. Bellm et al., *HERWIG 7.0/HERWIG++ 3.0 release note*, Eur. Phys. J. C **76** (2016) 196. https://doi.org/10.1140/epjc/s10052-016-4018-8
43. J. Bellm et al., *Herwig 7.1 Release Note*, CERN-PH-TH-2017-109 (2017). https://arxiv.org/abs/1705.06919
44. M. L. Mangano et al., *Alpgen, a generator for hard multiparton processes in hadronic collisions*, JHEP07(2003)001. https://doi.org/10.1088/1126-6708/2003/07/001
45. L. Lnnblad, *Ariadne version 4: A program for simulation of QCD cascades implementing the colour dipole model*, Comp. Phys. Comm. **71** (1992) 15. https://doi.org/10.1016/0010-4655(92)90068-A
46. ATLAS Collaboration, *Summary of ATLAS Pythia 8 tunes*, ATL-PHYS-PUB-2012-003 (2012). https://cds.cern.ch/record/1474107
47. CMS Collaboration, *Extraction and validation of a new set of CMS pythia8 tunes from underlying-event measurements*, Eur. Phys. J. C **80** (2020) 4. https://doi.org/10.1140/epjc/s10052-019-7499-4
48. X. Ju et al., *A novel workflow of generator tunings in HPC for LHC new physics searches*, in Ref. [25]. https://indico.cern.ch/event/751693/contributions/3183027
49. J. Bellm, L. Gellersen, *High dimensional parameter tuning for event generators*, Eur. Phys. J. C **80** (2020) 54. https://doi.org/10.1140/epjc/s10052-019-7579-5
50. S. Frixione, B. R. Webber, *Matching NLO QCD computations and parton shower simulations*, JHEP06(2002)029. https://doi.org/10.1088/1126-6708/2002/06/029
51. S. Frixione, P. Nason, C. Oleari, *Matching NLO QCD computations with parton shower simulations: the POWHEG method*, JHEP11(2007)070. https://doi.org/10.1088/1126-6708/2007/11/070
52. S. Jadach et al., *Matching NLO QCD with parton shower in Monte Carlo scheme the KrkNLO method*, JHEP10(2015)052. https://doi.org/10.1007/JHEP10(2015)052
53. S. Catani, F. Krauss, R. Kuhn, B. R. Webber, *QCD matrix elements + parton showers*, JHEP11(2001)063. https://doi.org/10.1088/1126-6708/2001/11/063
54. L. Lönnblad, *Correcting the colour-dipole cascade model with fixed order matrix elements*, JHEP05(2002)046. https://doi.org/10.1088/1126-6708/2002/05/046
55. M. L. Mangano, M. Moretti, R. Pittau, *Multijet matrix elements and shower evolution in hadronic collisions: $Wb\bar{b}$ + n-jets as a case study*, Nucl. Phys. B **632** (2002) 343. https://doi.org/10.1016/S0550-3213(02)00249-3
56. J. Alwall et al., *Comparative study of various algorithms for the merging of parton showers and matrix elements in hadronic collisions*, Eur. Phys. J. C **53** (2008) 473. https://doi.org/10.1140/epjc/s10052-007-0490-5
57. S. Höche, F. Krauss, M. Schönherr, *Uncertainties in MEPS@NLO calculations of h+jets*, Phys. Rev. D **90** (2014) 014012. https://doi.org/10.1103/PhysRevD.90.014012
58. K. Hamilton, P. Nason, G. Zanderighi, *MINLO: multi-scale improved NLO*. JHEP10(2012)155. https://doi.org/10.1007/JHEP10(2012)155
59. R. Frederix, S. Frixione, *Merging meets matching in MC@NLO* JHEP12(2012)061. https://doi.org/10.1007/JHEP12(2012)061
60. L. Lönnblad, S. Prestel, *Merging multi-leg NLO matrix elements with parton showers*, JHEP03(2013)166. https://doi.org/10.1007/JHEP03(2013)166
61. S. Plätzer, S. Gieseke, *Dipole showers and automated NLO matching in Herwig++*, Eur. Phys. J. C **72** (2012) 2187. https://doi.org/10.1140/epjc/s10052-012-2187-7
62. S. Plätzer, *Controlling inclusive cross sections in parton shower + matrix element merging*, JHEP08(2013)114. https://doi.org/10.1007/JHEP08(2013)114
63. J. Bellm, S. Gieseke, S. Plätzer, *Merging NLO multi-jet calculations with improved unitarization*, Eur. Phys. J. C **78** (2018) 244. https://doi.org/10.1140/epjc/s10052-018-5723-2
64. D. J. Lange, *The EvtGen particle decay simulation package*, Nucl. Instrum. Meth. A **462** (2001) 152. https://doi.org/10.1016/S0168-9002(01)00089-4
65. S. Jadach, Z. Was, R. Decker, J. H. Kühn, *The τ decay library TAUOLA, version 2.4*, Comp. Phys. Comm. **76** (1993) 361. https://doi.org/10.1016/0010-4655(93)90061-G
66. P. Golonka, Z. Was, *PHOTOS Monte Carlo: a precision tool for QED corrections in Z and W decays*, Eur. Phys. J. C **45** (2006) 97. https://doi.org/10.1140/epjc/s2005-02396-4
67. M. Botje, *The PDF4LHC Working Group Interim Recommendations* (2011). https://arxiv.org/abs/1101.0538v1
68. A. Buckley et al., *LHAPDF6: parton density access in the LHC precision era*, Eur. Phys. J. C **75** (2015) 132. https://doi.org/10.1140/epjc/s10052-015-3318-8
69. A. Buckley et al., *Rivet user manual* (2013). https://arxiv.org/abs/1003.0694v8.
70. A. Buckley et al., *Systematic event generator tuning for the LHC*, Eur. Phys. J. C **65** (2010) 331. https://doi.org/10.1140/epjc/s10052-009-1196-7
71. M. Dobbs, J. B. Hansen, *The HepMC C++ Monte Carlo event record for High Energy Physics*, Comp. Phys. Comm. **134** (2001) 41. https://doi.org/10.1016/S0010-4655(00)00189-2
72. A. Buckley et al., *The HepMC3 event record library for Monte Carlo event generators*, Comp. Phys. Comm., in press (2020). https://doi.org/10.1016/j.cpc.2020.107310
73. M. Cacciari, G. P. Salam, G. Soyez, *FastJet user manual*, Eur. Phys. J. C **72** (2012) 1896. https://doi.org/10.1140/epjc/s10052-012-1896-2
74. S. Roiser et al., *The LHCb Distributed Computing Model and Operations during LHC Runs 1, 2 and 3*, Proc. ISGC2015, Taipei. https://doi.org/10.22323/1.239.0005
75. F. Caravaglios, M. Moretti, *An algorithm to compute Born scattering amplitudes without Feynman graphs*, Phys. Lett. B **358** (1995) 332. https://doi.org/10.1016/0370-2693(95)00971-M
76. F. James, *Monte Carlo phase space*, CERN Yellow Report CERN-68-15 (1968). https://doi.org/10.5170/CERN-1968-015
77. F. James, *Monte Carlo theory and practice*, Rep. Progr. Phys. **43** (1980) 1145. https://doi.org/10.1088/0034-4885/43/9/002
78. G. P. Lepage, *A New Algorithm for Adaptive Multidimensional Integration*, J. Comp. Phys. **27** (1978) 192. https://doi.org/10.1016/0021-9991(78)90004-9
79. G. P. Lepage, *VEGAS: an adaptive multi-dimensional integration program*, Cornell report CLNS-447 (1980). https://cds.cern.ch/record/123074
80. J. Collins, D. Soper, G. Sterman, *Factorization of Hard Processes in QCD*, Adv. Ser. Direct. High Energy Phys. **5** (1989) 1. https://arxiv.org/abs/hep-ph/0409313v1
81. S. Weinzierl, *NLO Calculations*, Monte Carlo School, Hamburg (2012). https://indico.desy.de/indico/event/5064/session/8
82. M. Zaro, *MadGraph5_aMC@NLO tutorial*, QCD and event simulation for the LHC lectures, Pavia (2015). https://cp3.irmp.ucl.ac.be/projects/madgraph/wiki/Pavia2015
83. G. Luisoni, *An introduction to POWHEG*, Dartmouth-UW Experimental/Theory discussion (2017). https://indico.cern.ch/event/602457/contributions/2435408
84. E. Yazgan, *Event Generators in CMS*, CMS Heavy Flavor Tagging Workshop, Brussels (2018). https://indico.cern.ch/event/695320/contributions/2850950

85. F. Bishara, M. Montull, *(Machine) Learning amplitudes for faster event generation*, DESY 19-232 (2019). https://arxiv.org/abs/1912.11055
86. K. Danziger, *Efficiency Improvements in Monte Carlo Algorithms for High-Multiplicity Processes*, Master-Arbeit Thesis, TU Dresden, CERN-THESIS-2020-024 (2020). https://cds.cern.ch/record/2715727
87. W. H. Press, G. R. Farrar, *Recursive Stratified Sampling for Multidimensional Monte Carlo Integration*, Computers in Physics **4** (1990) 190. https://doi.org/10.1063/1.4822899
88. R. Kleiss, R. Pittau, *Weight optimization in multichannel Monte Carlo*, Comp. Phys. Comm. **83** (1994) 141. https://doi.org/10.1016/0010-4655(94)90043-4
89. T. Ohl, *Vegas Revisited: Adaptive Monte Carlo Integration Beyond Factorization*, Comp. Phys. Comm. **120** (1999) 13. https://doi.org/10.1016/S0010-4655(99)00209-X
90. R. H. Kleiss, W. J. Stirling, S. D. Ellis, *A new Monte Carlo treatment of multiparticle phase space at high energies*, Comp. Phys. Comm. **40** (1986) 359. https://doi.org/10.1016/0010-4655(86)90119-0
91. S.Kawabata, *A new Monte Carlo event generator for high energy physics*, Comp. Phys. Comm. **41** (1986) 127. https://doi.org/10.1016/0010-4655(86)90025-1
92. S.Kawabata, *A new version of the multi-dimensional integration and event generation package BASES/SPRING*, Comp. Phys. Comm. **88** (1995) 309. https://doi.org/10.1016/0010-4655(95)00028-E
93. P. Nason, *MINT: a Computer Program for Adaptive Monte Carlo Integration and Generation of Unweighted Distributions* Bicocca-FT-07-13 (2007). https://arxiv.org/abs/0709.2085
94. S. Jadach, *Foam: a general-purpose cellular Monte Carlo event generator*, Comp. Phys. Comm. **152** (2003) 55. https://doi.org/10.1016/S0010-4655(02)00755-5
95. F. Maltoni, T. Stelzer, *MadEvent: automatic event generation with MadGraph*, JHEP02(2003)027. https://doi.org/10.1088/1126-6708/2003/02/027
96. S. Brass, W. Kilian, J. Reuter, *Parallel Adaptive Monte Carlo Integration with the Event Generator WHIZARD*, Eur. Phys. J. C **79** (2019) 344. https://doi.org/10.1140/epjc/s10052-019-6840-2
97. W. Kilian, T. Ohl, J. Reuter, *WHIZARD: Simulating Multi-Particle Processes at LHC and ILC*, Eur. Phys. J. C **71** (2011) 1742. https://doi.org/10.1140/epjc/s10052-011-1742-y
98. T. Gleisberg, S. Höche, *Comix, a new matrix element generator*, JHEP12(2008)039. https://doi.org/10.1088/1126-6708/2008/12/039
99. C. Gao et al., *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, FERMILAB-PUB-20-010-T (2020). https://arxiv.org/abs/2001.05486
100. C. Gao et al., *Event generation with normalizing flows*, Phys. Rev. D **101** (2020) 076002. https://doi.org/10.1103/PhysRevD.101.076002
101. J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks* (2017). https://arxiv.org/abs/1707.00028
102. M. D. Klimek, M. Perelstein, *Neural Network-Based Approach to Phase Space Integration* (2018). https://arxiv.org/abs/1810.11509
103. S. Gleyzer, P. Seyfert, S. Schramm (eds.) et al., *Machine Learning in High Energy Physics Community White Paper* (2019). https://arxiv.org/abs/1807.02876
104. E. Bothmann et al., *Exploring phase space with Neural Importance Sampling*, MCNET-20-01 (2020). https://arxiv.org/abs/2001.05478
105. Generative Models session, ML4Jets2020 workshop, NYU. https://indico.cern.ch/event/809820/sessions/329213
106. A. Butter, T. Plehn, R. Winterhalder, *How to GAN LHC Events*, SciPost Phys. **7** (2019) 075. https://doi.org/10.21468/SciPostPhys.7.6.075
107. S. Alioli et al., *Jet pair production in POWHEG*, JHEP04(2011)081. https://doi.org/10.1007/JHEP04(2011)081
108. MadGraph, Biasing the generation of unweighted partonic events at LO, https://cp3.irmp.ucl.ac.be/projects/madgraph/wiki/LOEventGenerationBias.
109. R. Frederix et al., *Heavy-quark mass effects in Higgs plus jets production*, JHEP08(2016)006. https://doi.org/10.1007/JHEP08(2016)006
110. Sherpa Enhance_Function, Sherpa 2.0.0 Manual (2013). https://sherpa.hepforge.org/doc/SHERPA-MC-2.0.0.html#Enhance_005fFunction
111. Pythia8 Sample Main Programs, http://home.thep.lu.se/~torbjorn/pythia81html/SampleMainPrograms.html.
112. J. Alwall, S. de Visscher, F. Maltoni, *QCD radiation in the production of heavy colored particles at the LHC*, JHEP02(2009)017. https://doi.org/10.1088/1126-6708/2009/02/017
113. S. Gieseke, C. Röhr, A. Siódmok, *Colour reconnections in Herwig++*, Eur. Phys. J. C **72** (2012) 2225. https://doi.org/10.1140/epjc/s10052-012-2225-5
114. S. Gieseke et al., *Colour reconnection from soft gluon evolution*, JHEP11(2018)149. https://doi.org/10.1007/JHEP11(2018)149
115. J. S. Gainer et al., *Exploring theory space with Monte Carlo reweighting*, JHEP10(2014)78. https://doi.org/10.1007/JHEP10(2014)078
116. S. Mrenna, P. Skands, *Automated parton-shower variations in pythia 8*, Phys. Rev. D **94** (2016) 074005. https://doi.org/10.1103/PhysRevD.94.074005
117. O. Mattelaer, *On the maximal use of Monte Carlo samples: re-weighting events at NLO accuracy*, Eur. Phys. J. C **76** (2016) 674. https://doi.org/10.1140/epjc/s10052-016-4533-7
118. E. Bothmann, M. Schönherr, S. Schumann, *Reweighting QCD matrix-element and parton-shower calculations*, Eur. Phys. J. C **76** (2016) 590. https://doi.org/10.1140/epjc/s10052-016-4430-0
119. J. Bendavid et al., *Les Houches 2017: Physics at TeV Colliders Standard Model Working Group Report*, Proc. Les Houches 2017. https://arxiv.org/abs/1803.07977
120. A. Valassi, *Optimising HEP parameter fits via Monte Carlo weight derivative regression*, submitted to Proc. CHEP2019, Adelaide. https://arxiv.org/abs/2003.12853
121. J. Bellm et al., *Reweighting parton showers*, Phys. Rev. D **94** (2016) 034028. https://doi.org/10.1103/PhysRevD.94.034028
122. J. Bellm et al., *Parton-shower uncertainties with Herwig 7: benchmarks at leading order*, Eur. Phys. J. C **76** (2016) 665. https://doi.org/10.1140/epjc/s10052-016-4506-x
123. ATLAS collaboration, *Measurements of WH and ZH production in the $H \to b\bar{b}$ decay channel in pp collisions at 13 TeV with the ATLAS detector*, ATLAS-CONF-2020-006 (2020). http://cdsweb.cern.ch/record/2714885
124. K. Cranmer, J. Pavez, G. Louppe, *Approximating Likelihood Ratios with Calibrated Discriminative Classifiers* (2015). https://arxiv.org/abs/1506.02169
125. A. Andreassen, B. Nachman, *Neural Networks for Full Phase-space Reweighting and Parameter Tuning* (2019). https://arxiv.org/abs/1907.08209

126. G. Peyré, M. Cuturi, *Computational Optimal Transport* Foundations and Trends in Machine Learning **11** (2019) 355. https://arxiv.org/abs/1803.00567
127. A. Rogozhnikov, *Reweighting with Boosted Decision Trees* (2015). http://arogozhnikov.github.io/2015/10/09/gradient-boosted-reweighter.html
128. R. Frederix, S. Frixione, S. Prestel. P. Torrielli, *On the reduction of negative weights in MC@NLO-type matching procedures*, LU-TP 20-09 (2020). https://arxiv.org/abs/2002.12716
129. S. Höche, Y. Li, S. Prestel, *Drell-Yan lepton pair production at NNLO QCD with parton showers*, Phys. Rev. D **91** (2015) 074015. https://doi.org/10.1103/PhysRevD.91.074015
130. S. Höche, S. Prestel, *Triple collinear emissions in parton showers*, Phys. Rev. D **96** (2017) 074017. https://doi.org/10.1103/PhysRevD.96.074017
131. S. Plätzer, M. Sjodahl, J. Thorén, *Color matrix element corrections for parton showers*, JHEP11(2018)009. https://doi.org/10.1007/JHEP11(2018)009
132. R. Á. Martínez et al., *Soft gluon evolution and non-global logarithms*, JHEP05(2018)044. https://doi.org/10.1007/JHEP05(2018)044
133. J. Olsson, S. Plätzer, M. Sjodahl, *Resampling Algorithms for High Energy Physics Simulations*, MCnet-19-25 (2019). https://arxiv.org/abs/1912.02436
134. S. Höche et al., *A critical appraisal of NLO+PS matching methods*, JHEP09(2012)049. https://doi.org/10.1007/JHEP09(2012)049
135. S. Höche et al., *QCD matrix elements + parton showers: the NLO case*, JHEP04(2013)027. https://doi.org/10.1007/JHEP04(2013)027
136. D. Konstantinov, *Optimization of Pythia8*, EP-SFT group meeting, CERN (2020). https://indico.cern.ch/event/890670
137. C. Bauer et al., *Computing for perturbative QCD: a Snowmass White Paper* SLAC-PUB-15740 (2013). https://arxiv.org/abs/1309.3598
138. S. Agostinelli et al., *Geant4 a simulation toolkit*, Nucl. Instr. Meth. A **506** (2003) 250. https://doi.org/10.1016/S0168-9002(03)01368-8
139. O. Seiskari, J. Kommeri, T. Niemi, *GPU in Physics Computation: Case Geant4 Navigation* (2012). https://arxiv.org/abs/1209.5235
140. K. Murakami et al., *Geant4 based simulation of radiation dosimetry in CUDA*, Proc. IEEE NSS/MIC 2013, Seoul. https://doi.org/10.1109/NSSMIC.2013.6829452
141. G. Corti et al., *HEP software community meeting on GeantV R&D Panel Report* (2016). https://hepsoftwarefoundation.org/assets/GeantVPanelReport20161107.pdf
142. P. Canal, *Geant Exascale Pilot Project*, Geant4 R&D Meeting, CERN (2019). https://indico.cern.ch/event/809393/contributions/3441114
143. A. Gheata, *Design, implementation and performance results of the GeantV prototype*, Outcome of the GeantV prototype HSF meeting, CERN (2019). https://indico.cern.ch/event/818702/contributions/3559124
144. G. Amadio et al., *GeantV: results from the prototype of concurrent vector particle transport simulation in HEP* (2020). https://arxiv.org/abs/2005.00949
145. Oak Ridge Leadership Computing Facility, *Summit*. https://www.olcf.ornl.gov/summit
146. M. Feldman, *New GPU-Accelerated Supercomputers Change the Balance of Power on the TOP500*, Top500 news (2018). https://www.top500.org/news/new-gpu-accelerated-supercomputers-change-the-balance-of-power-on-the-top500
147. K. Hagiwara et al., *Fast calculation of HELAS amplitudes using graphics processing unit (GPU)*, Eur. Phys. J. C **66** (2010) 477. https://doi.org/10.1140/epjc/s10052-010-1276-8
148. K. Hagiwara et al., *Calculation of HELAS amplitudes for QCD processes using graphics processing unit (GPU)*, Eur. Phys. J. C **70** (2010) 513. https://doi.org/10.1140/epjc/s10052-010-1465-5
149. K. Hagiwara et al., *Fast computation of MadGraph amplitudes on graphics processing unit (GPU)*, Eur. Phys. J. C **73** (2013) 2608. https://doi.org/10.1140/epjc/s10052-013-2608-2
150. P. de Aquino, W. Link, F. Maltoni, O. Mattelaer, T. Stelzer, *ALOHA: Automatic libraries of helicity amplitudes for Feynman diagram computations*, Comp. Phys. Comm. **183** (2012) 2254. https://doi.org/10.1016/j.cpc.2012.05.004
151. H. Murayama, I. Watanabe, K. Hagiwara, *HELAS: HElicity Amplitude Subroutines for Feynman Diagram Evaluations*, KEK-Report 91-11 (1992). https://lib-extopc.kek.jp/preprints/PDF/1991/9124/9124011.pdf
152. I. Watanabe, H. Murayama, K. Hagiwara, *Evaluating Cross Sections at TeV Energy Scale by HELAS*, KEK preprint 92-39 (1992). https://lib-extopc.kek.jp/preprints/PDF/1992/9227/9227039.pdf
153. J. Kanzaki, *Monte Carlo integration on GPU*, Eur. Phys. J. C **71** (2011) 1559. https://doi.org/10.1140/epjc/s10052-011-1559-8
154. J. Kanzaki, *Application of graphics processing unit (GPU) to software in elementary particle/high energy physics field*, Procedia Computer Science **4** (2011) 869. https://doi.org/10.1016/j.procs.2011.04.092
155. E. Zenker et al., *Alpaka – An Abstraction Library for Parallel Kernel Acceleration*, Proc. IEEE IPDPSW 2016, Chicago. https://doi.org/10.1109/IPDPSW.2016.50
156. Alpaka – Abstraction Library for Parallel Kernel Acceleration. https://github.com/alpaka-group/alpaka
157. Intel oneAPI Toolkits (Beta). https://software.intel.com/en-us/oneapi
158. H.-Z. Wu, J.-J. Zhang, L.-G. Pang, Q. Wang, *ZMCintegral: a Package for Multi-Dimensional Monte Carlo Integration on Multi-GPUs*, Comp. Phys. Comm. **248** (2019) 106962. https://doi.org/10.1016/j.cpc.2019.106962
159. S. Carrazza, J. M. Cruz-Martinez, *VegasFlow: accelerating Monte Carlo simulation across multiple hardware platforms*, TIF-UNIMI-2020-8 (2020). https://arxiv.org/abs/2002.12921
160. S. Y. Jun et al., *Vectorization of random number generation and reproducibility of concurrent particle transport simulation*, Proc. ACAT2019, Saas Fee. https://inspirehep.net/literature/1754423
161. F. James, L. Moneta, *Review of High-Quality Random Number Generators*, Comput. Softw. Big Sci. **4** (2020) 2. https://doi.org/10.1007/s41781-019-0034-3
162. J. J. Dongarra et al., *A message passing standard for MPP and workstations*, Comm. ACM (1996). https://doi.org/10.1145/233977.234000
163. S. Höche, S. Prestel, H. Schulz, *Simulation of vector boson plus many final jets at the high luminosity LHC*, Phys. Rev. D **100** (2019) 0140124. https://doi.org/10.1103/PhysRevD.100.014024
164. NERSC, *Cori*. https://docs.nersc.gov/systems/cori
165. J. T. Childers et al., *Adapting the serial Alpgen parton-interaction generator to simulate LHC collisions on millions of parallel threads*, Comp. Phys. Comm. **210** (2017) 54. https://doi.org/10.1016/j.cpc.2016.09.013

166. Argonne Leadership Computing Facility, *Mira*. https://www.alcf.anl.gov/alcf-resources/mira
167. J. Alwall et al., *A standard format for Les Houches Event Files*, Comp. Phys. Comm. **176** (2007) 300. https://doi.org/10.1016/j.cpc.2006.11.010
168. O. Mattelaer, *MG5aMC status and plans*, in Ref. [25]. https://indico.cern.ch/event/751693/contributions/3182951
169. J. Campbell, T. Neumann, *Precision phenomenology with MCFM*, JHEP12(2019)034. https://doi.org/10.1007/JHEP12(2019)034
170. The OpenMP API specification for parallel programming, https://www.openmp.org
171. M. Czakon, A. Mitov, *Top++: A program for the calculation of the top-pair cross-section at hadron colliders*, Comp. Phys. Comm. **185** (2014) 2930. https://doi.org/10.1016/j.cpc.2014.06.021
172. K. Hamilton et al., *Merging H/W/Z + 0 and 1 jet at NLO with no merging scale: a path to parton shower + NNLO matching*, JHEP05(2013)082. https://doi.org/10.1007/JHEP05(2013)082
173. S. Alioli et al., *Matching fully differential NNLO calculations and parton showers*, JHEP06(2014)089. https://doi.org/10.1007/JHEP06(2014)089
174. P. F. Monni et al., *MiNNLOPS: A new method to match NNLO QCD to parton showers*, CERN-TH-2019-117 (2019). https://arxiv.org/abs/1908.06987
175. K. Hamilton et al., *NNLOPS simulation of Higgs boson production*, JHEP10(2013)222. https://doi.org/10.1007/JHEP10(2013)222
176. H. A. Chawdhry et al., *NNLO QCD corrections to three-photon production at the LHC*, JHEP02(2020)57. https://doi.org/10.1007/JHEP02(2020)057
177. M. Czakon, P. Fiedler, D. Heymes, A. Mitov, *NNLO QCD predictions for fully-differential top-quark pair production at the Tevatron*, JHEP05(2016)034. https://doi.org/10.1007/JHEP05(2016)034
178. S. Catani et al., *Top-quark pair production at the LHC: fully differential QCD predictions at NNLO*, JHEP07(2019)100. https://doi.org/10.1007/JHEP07(2019)100
179. S. Höche, S. Prestel, *The midpoint between dipole and parton showers*, Eur. Phys. J. C **75** (2015) 461. https://doi.org/10.1140/epjc/s10052-015-3684-2
180. N. Fischer, S. Prestel, M. Ritzmann, P. Skands, *VINCIA for hadron colliders*, Eur. Phys. J. C **76** (2016) 589. https://doi.org/10.1140/epjc/s10052-016-4429-6
181. Z. Nagy, D. E. Soper, *Jets and threshold summation in deductor*, Phys. Rev. D **98** (2018) 014035. https://doi.org/10.1103/PhysRevD.98.014035
182. S. Höche, F. Krauss, S. Prestel, *Implementing NLO DGLAP evolution in parton showers*, JHEP10(2017)093. https://doi.org/10.1007/JHEP10(2017)093
183. F. Dulat, S. Höche, S. Prestel, *Leading-color fully differential two-loop soft corrections to QCD dipole showers*, Phys. Rev. D **98** (2018) 074013. https://doi.org/10.1103/PhysRevD.98.074013
184. M. Dasgupta et al., *Parton showers beyond leading logarithmic accuracy*, CERN-TH-2020-026 (2020). https://arxiv.org/abs/2002.11114
185. S. Actis et al., *Recursive generation of one-loop amplitudes in the Standard Model*, JHEP04(2013)037. https://doi.org/10.1007/JHEP04(2013)037
186. S. Kallweit et al., *NLO electroweak automation and precise predictions for W+multijet production at the LHC*, JHEP04(2015)012. https://doi.org/10.1007/JHEP04(2015)012
187. M. Schönherr, *An automated subtraction of NLO EW infrared divergences*, Eur. Phys. J. C **78** (2018) 119. https://doi.org/10.1140/epjc/s10052-018-5600-z
188. R. Frederix et al., *The automation of next-to-leading order electroweak calculations*, JHEP07(2018)185. https://doi.org/10.1007/JHEP07(2018)185