

# Acceleration beyond lowest order event generation

## An outlook on further parallelism within MADGRAPH5\_AMC@NLO

Zenny Wettersten<sup>1,2\*</sup>, Olivier Mattelaer<sup>3</sup>, Stefan Roiser<sup>1</sup>, Robert Schöfbeck<sup>2</sup>, and Andrea Valassi<sup>1</sup>

<sup>1</sup>CERN

<sup>2</sup>HEPHY

<sup>3</sup>UCLouvain

**Abstract.** An important area of high energy physics studies at the Large Hadron Collider (LHC) currently concerns the need for more extensive and precise comparison data. Important tools in this realm are event reweighting and evaluation of more precise next-to-leading order (NLO) processes via Monte Carlo event generators, especially in the context of the upcoming High Luminosity LHC. Current event generators need to improve throughputs for these studies. MADGRAPH5\_AMC@NLO (MG5AMC) is an event generator being used by LHC experiments which has been accelerated considerably with a port to GPU and vector CPU architectures, but as of yet only for leading order processes. In this contribution a prototype for event reweighting using the accelerated MG5AMC software, as well as plans for an NLO implementation, are presented.

## 1 Introduction

As Moore's law's death throes echo throughout the world of high performance computing, a need for optimisation in hardware and software alike grows. Within high energy physics (HEP) this becomes especially apparent as we approach the era of the High Luminosity Large Hadron Collider (HL-LHC), where experimental precision and consequently both experimental and simulated measurements are expected to increase by an order of magnitude [1–4]. Many different treatments for these difficulties are being studied, largely organised under the umbrella of the HEP Software Foundation [5, 6], but here we consider exclusively event generation, and particularly *event reweighting*.

Event generation is the first step in HEP simulation, where process cross sections are evaluated and relevant instances of that process are stochastically generated to be used for later stages of simulation. In total, event generation currently makes up  $\sim 5 - 15\%$  of CPU hours at experiments [7, 8]. Due to the embarrassingly parallel and non-divergent nature of event generation, it is a prime candidate for exploring parallel architectures such as vectorised CPUs and GPUs, and over the last few years we have been working on porting leading order (LO) event generation within the MADGRAPH5\_AMC@NLO (MG5AMC) [9] framework to such systems. Here, we consider what developments to pursue in the future as the accelerated MG5AMC port (AMGAMC) [10–12] approaches an alpha release.

---

\*e-mail: zenny.wettersten@cern.ch

## 2 Leading order event reweighting

Particle collision simulations factorise [13] into stages, and it is often unnecessary to resimulate samples for distinct physics models. Instead, it suffices to regenerate events under the new physics model, and simulations can be recycled to accommodate these new physics. Such reuse allows for more extensive work in fitting physics model parameters to experimental observations. Furthermore, scattering amplitudes factorise from MC event weights. Rather than regenerating events, amplitudes can be reevaluated under the assumption of the new model, and event weights *reweighted* by this factor.

### 2.1 Event generation and weights

Given the inherently stochastic nature of particle physics, Monte Carlo (MC) methods are a natural fit for event generation. Normalising weights such that the total cross section is the sum of all event weights<sup>1</sup>, the weights  $W$  of LO events are given by [13]

$$W = |M|^2 \left( \prod_i f_i(x_i, k^2) \right) \Omega_{PS}, \quad (1)$$

with  $|M|^2$  the scattering amplitude<sup>2</sup> of the event,  $f_i(x_i, k^2)$  the parton distribution function of parton  $i$  with momentum fraction  $x_i$  at renormalisation scale  $k^2$ , and  $\Omega_{PS}$  the phase space measure. Defined as such, the total cross section  $\sigma$  is

$$\sigma = \int |M|^2 \left( \prod_i f_i(x_i, k^2) \right) d\Omega \cong \sum_k W_k, \quad (2)$$

with  $d\Omega$  the full angle differential, or equivalently the phase space measure. However, as mentioned, HEP events are stochastic; it is impossible to directly observe a total cross section, making the use of MC methods obvious: As only singular events can be observed in experiments, measurements naturally come about in a manner equivalent to a MC phase space distribution. For simulation purposes, MC methods are equivalent to real-world observations.

### 2.2 Event reweighting

Looking at (1), the scattering amplitude is the sole factor dependant on internal physics of the interaction<sup>3</sup>. Consequently, if the physics model is modified,

$$|M|^2 \rightarrow |M'|^2 \implies W \rightarrow W' = \frac{|M'|^2}{|M|^2} W, \quad (3)$$

i.e. so long as the relevant phase space for the new physics is a subset of the original one (ensuring  $|M|^2 \neq 0$ ), the new event weight can be determined by refactoring the original weight with the new amplitude. This process is known as event reweighting, and although we treat reweighting an LO process to a modified LO process there are other use cases, e.g. reweighting LO events to NLO to increase precision without full NLO event generation [8].

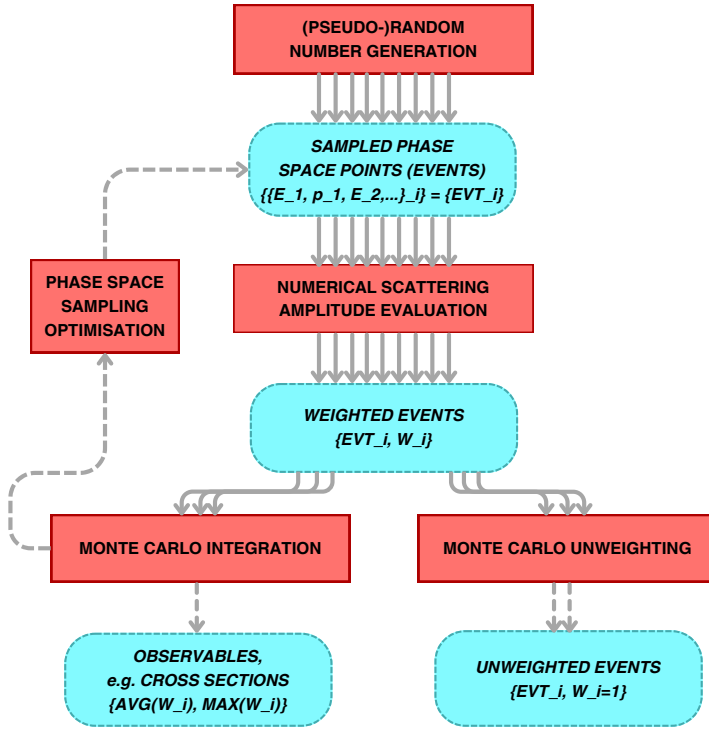
## 3 Parallel event generation with MG5aMC

Due to the embarrassingly parallel and non-divergent nature of event generation, it makes a good candidate for heterogeneous computing in HEP. Over the last few years there has been

<sup>1</sup>As long as the normalisation is kept in mind, this choice is arbitrary.

<sup>2</sup>Commonly referred to as *matrix element*, *matrix element squared*, *amplitude*, or *differential cross section*.

<sup>3</sup>The parton distribution functions depend only on the external partons, and the phase space measure is a volume.



**Figure 1.** Flowchart detailing the general structure of a scattering element generator. Red rectangles with right-angled corners correspond to numerical operations, while turquoise rounded nodes indicate their returned structures. Arrow abundance between nodes roughly hints at the amount of data in arguments and return values, save for all weighted events returned from the scattering amplitude evaluation being used for integration or unweighting. Returned observables and unweighted events can then be used by further simulation software.

work in porting MG5<sub>AMC</sub> event generation to the CUDA and SYCL standards [10–12], currently nearing release for LO event generation. Although we here primarily consider further development, a quick overview of the current framework is given below to contextualise work on parallel event reweighting and parallel NLO event generation.

### 3.1 LO parallelisation

The MG5<sub>AMC</sub> framework evaluates scattering amplitudes in terms of *helicity amplitudes*, using ALOHA-generated HELAS-based code [14, 15]. Event-level parallelism is an “appropriate approach” for acceleration [8], and exactly what has been implemented: The code structure is inherited from the MAD<sub>EVENT</sub> event generator, but amplitudes<sup>4</sup> are evaluated with vectorised C++ code on CPUs, and using the CUDA API on GPUs.

An overview of the MC event generator structure is provided in Fig. 1, where the green bubble specifies the part event-level parallelism applies to. Note that MG5<sub>AMC</sub> is a meta-program, a *code generator*, rather than a singular implementation. MG5<sub>AMC</sub> generates, exports, and runs a program for considered physics processes. Particularly, scattering amplitude evaluations are called as external subroutines distinct from phase space integration. Consequently, it is simple (although not easy) to replace these with e.g. vectorised routines. Scattering amplitudes being the primary bottleneck in event generation, this is the consideration for acceleration in  $\Delta$ MG5<sub>AMC</sub>.

<sup>4</sup>Note that *only* scattering amplitude evaluations are parallelised here. Other event-specific parts, such as random number generation, are kept on the host. Further points of acceleration are being investigated, but are not a priority.

<b>Process</b>	$e^+e^- \rightarrow 5\gamma$ (SM)
<b>CPU</b>	Intel Core i5-1145G7
<b>GPU</b>	NVIDIA A100 PCIe 40GB

Table 1: Details for the time measurements displayed in Figures 2 and 3, showing the process considered and hardware used for the two implementations. Note that all reweighting is done on within the standard model, setting the electroweak coupling constant to be a randomly determined non-zero value for each reweight iteration.

### 3.2 Parallel LO scattering amplitudes and reweighting

As detailed in Sec. 3.1, not only are scattering amplitude routines distinct from other event generation steps, but MG5AMC and AMGAMC generate them independently. In light of Sec. 2, where event reweighting was shown to only use amplitude evaluation, these routines can be repurposed for reweighting. This is considered here: The usage of HELAS-like CUDA code, in line with the native MG5AMC reweighting module [16].

Although AMGAMC does not yet support out-of-the-box event generation — generated code takes a posteriori modifications — standalone processes (i.e. amplitude generation, the green bubble in Fig. 1) *do* work. Using this standalone output, a minimal program calling an in-house generic reweighting library<sup>5</sup> can perform event reweighting on a provided event set.

This comes with a caveat: There exists no interface between these parts yet. With AMGAMC, the necessary amplitude subroutines can largely<sup>6</sup> be created; our reweighting library reads and runs events through generated subroutines; and a separate program calls these libraries to perform the reweighting.

Once set up and compiled, reweighting goes roughly as follows: **1)** Input reweight parameters; **2)** Parse events; **3)** Evaluate original scattering amplitudes; **4)** Overwrite physics parameters with new values; **5)** Evaluate new amplitudes; **6)** Evaluate new weight with eq. (3); **7)** Repeat steps 4 – 6 for all parameter sets, and; **8)** Return new weights. We note that this procedure only holds for parameters that can be modified at runtime, making reweighting between structurally different amplitude routines tedious (albeit possible).

### 3.3 Results

To evaluate the reweight acceleration of AMGAMC, two independent degrees of complexity need be considered: For statistics, it is necessary to increase the amount of events; for physics studies, it becomes important to consider more parameter sets. Practically, these are close to identical, as reweighting over multiple parameter sets means performing more amplitude evaluations for the same event. Runtimes are thus expected to rise linearly with both, at least beyond some minimum number of events (per parameter set) such that amplitudes dominate runtime. In Table 1, the measurement details are shown<sup>7</sup>, and time measurements are plotted against number of events and of parameter sets in Figs. 2 and 3, respectively.

As can be seen in Figs. 2 and 3, runtimes increase linearly<sup>8</sup> with reweight process complexity for both MG5AMC and AMGAMC. The left-hand graphs show total CPU times, while for the right-hand graphs the runtimes have been independently normalised to the time to reweight a single event for one parameter set. For small event sets, both implementation runtimes are dominated by (initialisation, I/O, etc.), making this choice unfit for AMGAMC due

<sup>5</sup>This library is expected to see an official release before the end of the year. We forego further details here.

<sup>6</sup>AMGAMC does not yet support fully generic beyond-the-Standard Model process generation.

<sup>7</sup>Note that the CPU used is consumer grade, making one-to-one runtime comparisons less relevant.

<sup>8</sup>Note that Fig. 2 has log-log graphs, whereas Fig. 3 has log-linear graphs.

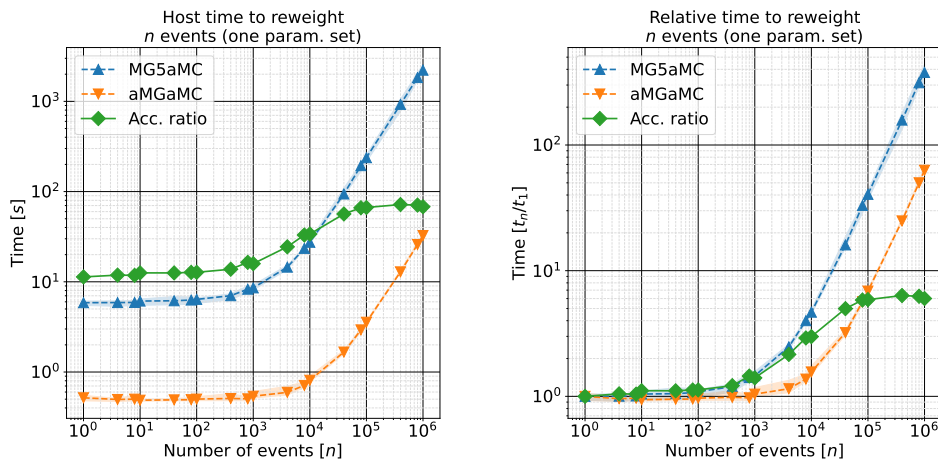


Figure 2: Comparison of real (left) and relative (right) host (CPU) time to reweight  $n$  events for one set of parameters for MG5aMC and aMGaMC. Average time over five measurements is shown, with min- and maximum value for each  $n$  denoted by coloured regions. In the right plot, times have been independently normalised to the time  $t_1$  it takes to reweight one event once. The green solids show the ratio between the two dashed lines. See Table 1 for details on the reweight procedure.

to the overhead occurring on the host rather than on the device, resulting in a bias against the GPU.

In Fig. 2, similar runtime complexities are observed between the two implementations; initial domination by statistical effects, but a trend towards linear growth in the large- $n$  limit. The main difference is an initial constant runtime for aMGaMC, explained by two factors: Predominant overhead in the small- $n$  limit is transfer rate between host and device, which for small  $n$  is roughly constant; and amplitude evaluation runtime will not increase until the device is saturated, which will happen at  $\sim 10^4$  events. Acceleration from the GPU is depicted here by green lines, which show the ratio between the two implementation runtimes. Fig. 3, which details the runtimes as functions of the number of parameter sets, shows a clear linear growth for both implementations. As mentioned, reweighting over several parameter sets is equivalent to reweighting more events when amplitude evaluation is the dominant runtime contribution, and for Fig. 3 sufficiently many events per iteration have been taken for both implementations to land in the linear large- $n$  limit shown in Fig. 2. Note that the green lines depicting the runtime ratio in Fig. 3 have been multiplied by the runtime quotient  $\sim 8.66$  between  $10^5$  and  $10^4$  events for MG5aMC, to account for these measurements being made for  $10^4$  events rather than the  $10^5$  events of aMGaMC. This is a logistical limitation in the runtime for such large event sets, but the factor used is biased towards MG5aMC<sup>9</sup>, and this factor is nevertheless not much smaller than the naive maximum factor 10.

<sup>9</sup>Most overhead (particularly I/O) depends only on the number of events, not on the number of parameter sets: Each event needs to be parsed only once, overcounting this overhead for each parameter set past the first one.

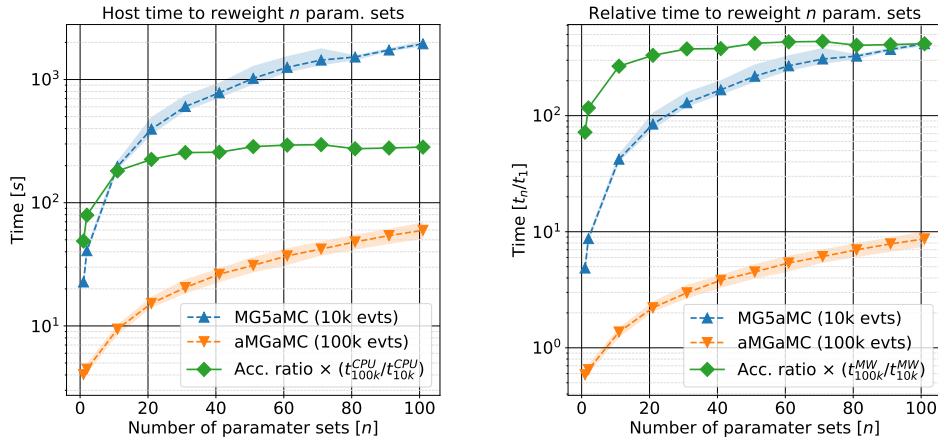


Figure 3: Comparison of real (left) and relative (right) host (CPU) time to reweight  $10^4$  (MG5aMC) or  $10^5$  (aMGaMC) events for  $n$  parameter sets. Average time over five measurements is shown, with min- and maximum value for each  $n$  denoted by coloured regions. In the right plot, the normalisation shown in Fig. 2 (right plot) is used. The green solid lines show the ratio between the dashed lines, multiplied by a factor  $\sim 8.66$  longer it takes MG5aMC to reweight  $10^5$  events than  $10^4$ . See Table 1 for details on the reweight procedure.

## 4 Expectations for NLO parallelism

While we approach an official LO aMGaMC release, we turn to the next major goal: NLO event generation. Work is just commencing to this end, but considerations of aMGaMC alongside the structure of NLO event generation in native MG5aMC can be presented.

In MG5aMC, NLO event generation uses the FKS subtraction scheme [17–19]. We forego theoretical details here, but note that this splits NLO amplitudes into three distinct subsets:

**LO amplitudes.** In perturbation theory, higher order contributions are *corrections* to LO amplitudes. Thus, LO amplitudes need to be evaluated also for NLO calculations.

**Real emission amplitudes.** Here, additional unobserved particles are added to the interaction. These are *tree-level*, and can be evaluated using the same machinery as LO amplitudes.

**One-loop amplitudes.** These loop amplitudes in NLO corrections require integration over undetermined momenta for singular amplitudes. While many clever techniques for such evaluations exist, it requires different machinery to that at tree-level.

In porting LO event generation to parallel architectures the structures used for tree-level amplitudes have already been developed; we already have the tools to evaluate real emissions. The first two points for NLO event generation can be treated as LO multiprocesses, and although their NLO use has not been implemented, we expect few problems in doing so.

Loops, though, present an issue. Native MG5aMC calls external libraries to evaluate loop integrals [9]; development of such GPU-compatible libraries is largely unaddressed as of yet<sup>10</sup>. Additionally, depending on the numerical stability, it is at times necessary to evaluate

<sup>10</sup>GPU-focused loop developments [20–23] largely focus on speeding up the loop integrals themselves, whereas we consider event-level parallelism.

loop integrals at quadruple precision, which no GPU or TPU manufacturer currently supports. Nevertheless, it appears that loops could be evaluated on GPUs for at least a large fraction of events.

## 5 Outlook

Over the course of this paper, we have detailed expected difficulties in development of `AMGAMC` past LO event generation, as well as presented a simple comparison in how already supported development can be extended for purposes other than pure scattering amplitude generation, here using `AMGAMC` to parallelise scattering amplitudes for LO event reweighting. Although resulting runtimes are difficult to compare one-to-one, parallelism evidently makes the encroaching need for larger event sets in HEP less problematic.

Nevertheless, LO event generation parallelism will be insufficient to tackle computational needs in the coming decade. The difficulties in porting also NLO scattering amplitude generation are being investigated, and while the computation of one-loop amplitudes is predicted to pose a problem, we are optimistic. The architecture for parallel evaluation of tree-level diagram amplitudes already exists, and a first partial port of NLO event generation to `AMGAMC` could be done by performing these tree-level evaluations on vectorised architecture, keeping loop evaluations on the host.

Somewhat orthogonal to this development, we are working on running also LO event reweighting on heterogeneous architectures. A C++ library for parsing the LHE file format and interfacing with generic structures for performing event reweighting has been developed, and we hope to have it released publicly by the end of the year. Interfacing this library with the current pre-alpha release of `AMGAMC`, a test bed for LO event reweighting on GPUs has been established, demonstrating the sizeable acceleration attained with vectorisation and GPU parallelism. In Figs. 2 and 3, we display that in the high complexity limit, the `AMGAMC` reweighting implementation will be a sizeable factor faster than `MG5AMC`. The specific factor is, of course, hardware-dependent.

Although some complications must be dealt with prior to an official release of the `AMGAMC` plugin, deliberation on future development plans is becoming increasingly important. A favoured candidate is immediately moving onto NLO event generation, and here we have discussed some of the foreseen obstacles. However, other paths to success have revealed themselves as well: Here we have treated event reweighting, which allows for recycling generated event sets to probe physics parameters without the need to regenerate or resimulate events. Continued effort in exploring attainable benefits in heterogeneous architectures and in how to apply these efforts in multiple synergistic directions appears to be not only effective, but necessary, for the future of particle physics.

## References

- [1] O. Brüning, L. Rossi, eds., *The High Luminosity Large Hadron Collider*, Vol. 24 (2015), ISBN 978-981-4675-46-8, 978-981-4678-14-8
- [2] L. Rossi, O. Brüning, *Progress with the High Luminosity LHC project at CERN*, in *10th International Particle Accelerator Conference* (2019)
- [3] CMS Offline Software and Computing, Tech. rep., CERN, Geneva (2022), <https://cds.cern.ch/record/2815292>
- [4] ATLAS Collaboration, Tech. rep., CERN, Geneva (2022), <https://cds.cern.ch/record/2802918>

- [5] J. Albrecht, A.A. Alves, G. Amadio, G. Andronico, N. Anh-Ky, L. Aphécetche, J. Apostolakis, M. Asai, L. Atzori, M. Babik et al., *Computing and Software for Big Science* **3** (2019)
- [6] G.A. Stewart, P. Elmer, E. Sexton-Kennedy, *The HEP Software Foundation Community* (2022), 2205.08193
- [7] T. Aarrestad et al. (HEP Software Foundation), *HL-LHC Computing Review: Common Tools and Community Software*, in *Snowmass 2021*, edited by P. Canal et al. (2020), 2008.13636
- [8] S. Amoroso et al. (HSF Physics Event Generator WG), *Comput. Softw. Big Sci.* **5**, 12 (2021), 2004.13687
- [9] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.S. Shao, T. Stelzer, P. Torrielli, M. Zaro, *JHEP* **07**, 079 (2014), 1405.0301
- [10] A. Valassi, S. Roiser, O. Mattelaer, S. Hageboeck, *EPJ Web Conf.* **251**, 03045 (2021), 2106.12631
- [11] A. Valassi, T. Childers, L. Field, S. Hageboeck, W. Hopkins, O. Mattelaer, N. Nichols, S. Roiser, D. Smith, *PoS ICHEP2022*, 212 (2022), 2210.11122
- [12] A. Valassi, T. Childers, L. Field, S. Hageböck, W. Hopkins, O. Mattelaer, N. Nichols, S. Roiser, D. Smith, J. Teig et al., *Speeding up Madgraph5 aMC@NLO through CPU vectorization and GPU offloading*, in *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research* (2023), 2303.18244
- [13] O. Mattelaer, *Eur. Phys. J. C* **76**, 674 (2016), 1607.00763
- [14] H. Murayama, I. Watanabe, K. Hagiwara, *Tech. Rep. KEK-91-11*, National Lab. for High Energy Physics, Tsukuba, Ibaraki (1998)
- [15] P. de Aquino, W. Link, F. Maltoni, O. Mattelaer, T. Stelzer, *Computer Physics Communications* **183**, 2254–2263 (2012)
- [16] P. Artoisenet, V. Lemaître, F. Maltoni, O. Mattelaer, *JHEP* **12**, 068 (2010), 1007.3300
- [17] S. Frixione, Z. Kunszt, A. Signer, *Nucl. Phys. B* **467**, 399 (1996), hep-ph/9512328
- [18] S. Frixione, *Nucl. Phys. B* **507**, 295 (1997), hep-ph/9706545
- [19] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H.S. Shao, M. Zaro, *JHEP* **07**, 185 (2018), [Erratum: *JHEP* 11, 085 (2021)], 1804.10017
- [20] F. Yuasa, T. Ishikawa, N. Hamaguchi, T. Koike, N. Nakasato, *Journal of Physics: Conference Series* **454**, 012081 (2013)
- [21] Z. Li, J. Wang, Q.S. Yan, X. Zhao, *Chin. Phys. C* **40**, 033103 (2016), 1508.02512
- [22] A. Smirnov, *Computer Physics Communications* **204**, 189 (2016)
- [23] R. Winterhalder, V. Magerya, E. Villa, S.P. Jones, M. Kerner, A. Butter, G. Heinrich, T. Plehn, *SciPost Phys.* **12**, 129 (2022), 2112.09145