

Efficient many-jet event generation with flow matching

E. Bothmann^{1,2}, T. Janßen^{2,3,*}, M. Knobbe⁴, B. Schmitzer^{3,5} and F. Sinz^{3,5}

¹*IT Department, CERN, 1211 Geneva 23, Switzerland*

²*Institute for Theoretical Physics, University of Göttingen, Göttingen, Germany*

³*Campus Institute Data Science, University of Göttingen, Göttingen, Germany*

⁴*Fermi National Accelerator Laboratory, Batavia, Illinois, USA*

⁵*Institute for Computer Science, University of Göttingen, Göttingen, Germany*



(Received 25 July 2025; accepted 18 March 2026; published 27 August 2026)

We apply for the first time, to the best of our knowledge, the flow matching method to the problem of phase-space sampling for event generation in high-energy collider physics. By training the model to remap the random numbers used to generate the momenta and helicities of the scattering matrix elements as implemented in the portable partonic event generator PEPPER, we find substantial efficiency improvements in the studied processes. We focus our study on the highest final-state multiplicities in Drell-Yan and top-antitop pair production used in simulated samples for the Large Hadron Collider, which computationally are the most relevant ones. We find that the unweighting efficiencies improve by factors of 184 and 25, respectively, when compared to the standard approach of using a VEGAS-based optimization. We also compare continuous normalizing flows trained with flow matching against the previously studied normalizing flows based on coupling layers and find that the former leads to better results, faster training and a better scaling behavior across the studied multiplicity range, while the latter evaluate faster. When combining the advantages of both methods using the REGFLOW approach, we find parton-level unweighted event generation walltime gains of about a factor of 10 at the highest final-state multiplicities.

DOI: [10.1103/j71d-h5w2](https://doi.org/10.1103/j71d-h5w2)

I. INTRODUCTION

The excellent experimental precision of the current and future runs at the Large Hadron Collider (LHC) and its successors must be met with a corresponding increase in the precision of theoretical simulations. This is essential to meet the objectives outlined in collider physics programs. Otherwise the uncertainty of many experimental results would be dominated by the uncertainty of the simulated data samples that enter experimental analyses in many ways, most prominently to provide background estimates [1–5]. Precision must be improved across three key areas: parametric accuracy, numerical stability, and statistical precision. In this contribution, we focus on the latter, i.e. the challenge to deliver large simulated data samples that match the statistical quality of the experimental data. To illustrate the scale of this challenge, the ATLAS Collaboration estimates [6] that 330 billion simulated events are required to match the High-Luminosity LHC data set for a

key background process vector boson production in association with additional jets, when using the general-purpose event generator SHERPA v2.2.11. This translates to 3.8 million HEPSPC06 years [6], which corresponds to about 1000 Perlmutter $2 \times$ CPU nodes [7] for an entire year [6].

At the heart of the event-generation pipeline lies the hard scattering event, which has two incoming particles for collider setups, and a fixed number of outgoing particles—up to seven in current production campaigns. Except for the experiment-specific simulation of the detector response, it is by far the most computationally demanding part of the pipeline, and therefore a prime target for performance improvements [8]. Low overall Monte Carlo simulation efficiency is driven by the highest-multiplicity processes in the simulation, for which current simulation programs only achieve very low phase-space sampling efficiency. They typically rely on multichannel techniques [9] combined with the adaptive sampling algorithm VEGAS (or similar *classical* machine learning algorithms) [10–15]. However, for processes with 7 outgoing particles, corresponding to a 19-dimensional phase space, the overall efficiency achieved with these methods can be as low as 0.01% [16], thus immensely degrading the statistical power of the generated sample.

Modern machine learning approaches to optimize sampling based on deep neural networks have been proposed to

*Contact author: timo.janssen@theorie.physik.uni-goettingen.de

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.

improve the efficiency by making use of their more flexible parametrization. There has been a particular focus on using normalizing flows as a drop-in replacement for VEGAS [17–24]. For individual partonic channels, efficiency improvements of factors between 5 and 10 over VEGAS have been demonstrated for up to five final-state particles [22]. However, a systematic study which addresses the highest and most expensive multiplicities used in standard productions at the LHC is still missing.

Here, we propose for the first time, to the best of our knowledge, to solve this issue using continuous normalizing flows [25], which implement time-continuous flows through an ordinary differential equation (ODE). To avoid expensive simulation of the ODE dynamics during training, we use the flow matching method [26–29].¹ We study the performance of our approach for the computationally, by far, most prominent simulated processes for the LHC, i.e. lepton pair production ($pp \rightarrow e^+e^- + \text{up to 5 jets}$) and top quark pair production ($pp \rightarrow t\bar{t} + \text{up to 4 jets}$), utilizing the fast event generation framework PEPPER [34] and its phase-space sampler CHILI [35]. This covers the entire multiplicity range of current production campaigns. We evaluate our results in several key measures, e.g. in the integration error of the generated sample, the effective sample size, and the unweighting efficiency, thereby comparing the performance of continuous flows with that of an unoptimized sample, with VEGAS, and with coupling-layer-based normalizing flows [36–38]. We find that our new optimization method increases the effective sample size by factors between 8.3 and 70 for the highest multiplicities studied, when compared to the conventional VEGAS-based optimization. Crucially, our method also yields large improvements in unweighting efficiency for the highest, currently, relevant multiplicity. To mitigate the increased evaluation cost of continuous flows, we also examine the REGFLOW approach [39], which combines the improved adaptability of continuous flows with the faster inference times of coupling-layer-based flows. To the best of our knowledge, we are the first to condition the phase-space sampler on the helicity states of the external particles, which allows us to optimize for individual helicity amplitudes and exploit correlations between helicities and kinematic variables. We provide a minimal interface to the PEPPER event generator allowing for straight-forward integration of similar techniques in the future.

Our article is structured as follows: In Sec. II, we introduce the main problem we are trying to solve and bring it into the larger context of phase-space sampling in high-energy physics. We then provide a summary of continuous normalizing flows and flow matching in Sec. III. In Sec. IV we describe our approach for applying

flow matching to phase-space sampling for relevant LHC processes using the PEPPER event generator. The results are then presented in Sec. V, followed by our conclusions in Sec. VI. Appendix A provides the event generation runcards used in this work, and Appendix B presents additional results. A companion Letter summarizing the key results of this work appears in Ref. [40].

II. PROBLEM STATEMENT: PHASE-SPACE SAMPLING

In this section, we give an overview of phase-space sampling in high-energy physics. In order to make it more accessible to readers that are not domain experts, we begin our description in Sec. II A in mathematical terms by formulating the problem as an integral to be estimated by importance sampling. Section II B introduces unweighted event generation as a different perspective motivated by typical High Energy Physics (HEP) applications. In both cases, integration and unweighting, a trainable remapping from a uniform distribution to a nonuniform one can improve the relevant metrics. This is described in Sec. II C, where we also derive the respective training objectives to be minimized. Finally, the connection of the mathematical treatment to physics and the discussion of domain-specific aspects follow in Sec. II D.

A. Phase-space sampling in mathematical terms

The main problem is to evaluate the integral

$$I = \int_M dx p(x) f(x) \quad (1)$$

over a manifold M . In our application, the variable x is a vector that describes a scattering event by incorporating the momentum components of the involved particles. The probability density function p describes the probability of a scattering event through the relation $p(x)dx = \sigma^{-1} d\sigma$, where σ is the scattering cross section, and the function f gives the value of the physical observable.

Our approach to solve this problem is Monte Carlo integration using importance sampling. To make the problem more suitable for Monte Carlo sampling, we pull back p to the unit hypercube $U = [0, 1]^d$ using a bijective map $\phi: U \rightarrow M$, such that

$$\begin{aligned} I &= \int_0^1 d^d x \det \left[\frac{\partial \phi}{\partial x}(x) \right] p(\phi(x)) f(\phi(x)) \\ &= \int_0^1 d^d x p_0(x) f(\phi(x)), \end{aligned} \quad (2)$$

where we introduced the density p_0 as the pull-back of p by ϕ :

$$p_0(x) = \phi^* p(x) = p(\phi(x)) \det \left[\frac{\partial \phi}{\partial x}(x) \right]. \quad (3)$$

¹Flow matching has recently been studied for neural-network based generation of LHC events and off-shell effects [30–33]. In contrast, in our study we do not model physical effects with a generative model, but use flow matching only to improve the simulation’s sampling efficiency.

In consequence, we can draw N points x_i independently from the multivariate uniform distribution u_d over U and approximate the integral I with the Monte Carlo estimator,

$$E = \frac{1}{N} \sum_{i=1}^N p_0(x_i) f(\phi(x_i)) = \frac{1}{N} \sum_{i=1}^N w_i. \quad (4)$$

We call w_i the weight of the event with index i . The expectation of E is I and its variance is given by

$$\text{Var}(E) = \mathbb{E}(E^2) - \mathbb{E}(E)^2. \quad (5)$$

In a Monte Carlo simulation, it can be estimated by the unbiased sample variance. The variance is an indicator of the sampling efficiency. When the variance is large, many points are needed to determine the integral with a given precision. The size of the variance depends on the quality of the map ϕ .

B. Unweighted event generation through rejection sampling

A related task is the production of independent and identically distributed events following the distribution with density function $p(x)$. In the context of HEP simulations, this is commonly called unweighted event generation. The need for it arises in the typical simulation toolchain used in experimental analyses. The parton-level event generation as described in Sec. II D is followed by further postprocessing steps. These can be very expensive, especially the simulation of the detector response. If we were to use the points $\phi(x_i)$ generated by importance sampling as described above, some of them would come with small weights. Running the postprocessing on these events would consume significant resources, although their contribution to the integral is small. Consequently, generating unweighted parton-level events first is often more efficient, even though they are much more expensive to produce. Another advantage is that the sample is typically much smaller after unweighting, which saves storage space.

A general method to generate unweighted events is *rejection sampling* [41]. It can be used in the situation where it is desired to sample from the distribution p on M but one can originally only sample from another distribution q . In our case, we can sample from the uniform distribution u_d on U and apply the map ϕ to find $q(x) = \phi_* u_d(x)$, where we introduced the push-forward operator

$$\phi_* u_d(x) = u_d(x) \det \left[\frac{\partial \phi^{-1}}{\partial x}(x) \right] = \det \left[\frac{\partial \phi^{-1}}{\partial x}(x) \right]. \quad (6)$$

Trial events are drawn from u_d and accepted or rejected based on an event-wise acceptance probability, given by

$$p_{\text{accept}}(x) = \frac{w(x)}{C} \quad \text{with} \quad w(x) = \frac{p_0(x)}{u_d(x)} = \frac{p(\phi(x))}{q(\phi(x))}, \quad (7)$$

where C is a scaling constant chosen such that

$$C \geq w(x) \quad \forall x \in M. \quad (8)$$

This procedure generates a sample of reduced size with uniform weights w , with a distribution which is identical to the target distribution. The resulting *unweighting efficiency* is given by the average acceptance probability,

$$\epsilon = \frac{\langle w \rangle}{C}, \quad (9)$$

which is therefore a crucial figure of merit to measure the performance of a phase-space generator. The highest efficiency can be reached by choosing $C = w_{\text{max}} := \max_x w(x)$. In practice, the maximal weight w_{max} needs to be estimated in an initial survey phase from a finite set of events. Since the unweighting efficiency is very sensitive to single large outlier weights in this survey, one often defines an effective $w_{\text{max,eff}}$ to increase the efficiency and to render its value more numerically stable. When encountering events with weights $w > w_{\text{max,eff}}$, the event weight is set to $w/w_{\text{max,eff}}$. Such weights are referred to as overweights. Choosing a suitable effective maximal weight requires balancing the introduced variance of overweights with the improved unweighting efficiency of a reduced maximal weight. A common approach is to define the desired fraction of overweight events via their contribution to the integral. For example $\epsilon_{0.001}$ denotes the unweighting efficiency for which maximally 0.1% of the integral is contributed by events with weights that are larger than $w_{\text{max,eff}}$.

Another frequently quoted measure to quantify the quality of a sample is the Kish effective sample size N_{eff} , defined as

$$N_{\text{eff}} = \frac{\langle w \rangle^2}{\langle w^2 \rangle}. \quad (10)$$

This estimates the relative size of the corresponding unit weight sample that would result in the same variance of the integral estimator [42]. Note that $N_{\text{eff}} = 1$ if all weights are equal, i.e. if the weight distribution is given by $\delta(w - \langle w \rangle)$. For a narrow weight distribution, the value of N_{eff} will still be close to one, while the value becomes smaller and approaches zero for wider distributions. In contrast to the unweighting efficiency, N_{eff} is less sensitive to large outlier weights.

C. Remapping on the unit hypercube

All measures introduced above, the variance of the integral estimate, the unweighting efficiency, and the effective sample size, depend on the quality of the map ϕ .

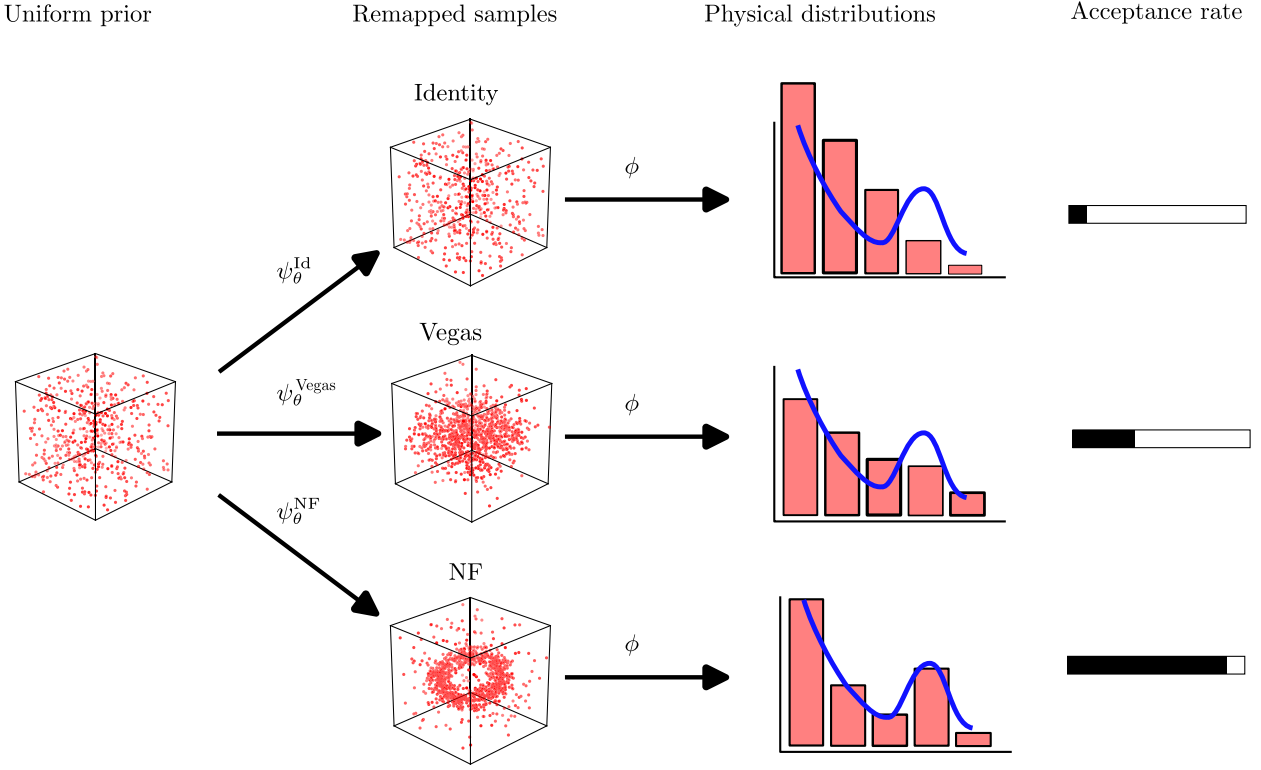


FIG. 1. Visualization of the Identity (Id), VEGAS and Normalizing Flow (NF) maps used in our integration approach. Beginning from the uniform prior, ψ_θ remaps the random numbers and ϕ maps to the physical variables. The latter are represented after a projection onto a one-dimensional physical distribution, with the true distribution depicted as a solid line. The Monte Carlo acceptance rate depends on how similar the generated and the true distribution are.

The map ϕ can be constructed for specific scattering processes based on physics: A well-chosen ϕ incorporates knowledge about the modes and general shape of p (and possibly f). Although ϕ can be designed manually for simple cases, finding an effective mapping becomes challenging for more complex scattering processes, especially those involving many particles. This leads to large variances in Eq. (5) and small unweighting efficiencies in Eq. (9). To address this problem, we use optimization techniques and combine ϕ with a second bijective map, $\psi_\theta: U \rightarrow U$, a parametric model with parameters θ that can be adapted to data. This is visualized in Fig. 1.

We first consider the case of integration, as described in Sec. II A. After introducing the map ψ_θ , the integral Eq. (2) becomes

$$I = \int_0^1 dx \left| \frac{\partial \psi_\theta}{\partial x} \right| p_0(\psi_\theta(x)) f(\phi(\psi_\theta(x))) \quad (11)$$

and its Monte Carlo estimator is now given by

$$E_\theta = \frac{1}{N} \sum_{i=1}^N \left| \frac{\partial \psi_\theta}{\partial x} \right| p_0(\psi_\theta(x_i)) f(\phi(\psi_\theta(x_i))). \quad (12)$$

Its expectation value is $\langle E_\theta \rangle = I$ by definition and its variance is

$$\text{Var}(E_\theta) = \mathbb{E}(E_\theta^2) - \mathbb{E}(E_\theta)^2. \quad (13)$$

Let

$$q_\theta = (\psi_\theta)_* u_d. \quad (14)$$

The variance, Eq. (13), becomes minimal when q_θ is proportional to $p_0 \cdot (f \circ \phi)$. Thus, if our goal is to minimize the variance, we need to optimize ψ_θ such that q_θ becomes close to $p_0 \cdot (f \circ \phi)$. If the integral is the total scattering cross section σ , the function f is constant and the task reduces to finding a q_θ close to p_0 .

We can also consider unweighted event generation, cf. Sec. II B, with ψ_θ . In this case, we sample from q_θ and accept events with probability

$$p_{\text{accept}}(x) = \frac{w_\theta(x)}{C_\theta} \quad \text{with} \quad w_\theta(x) = \frac{p_0(x)}{q_\theta(x)}, \quad (15)$$

where the ideal scaling factor is now $C_\theta = \max_x w_\theta(x)$. The unweighting efficiency, Eq. (9), becomes

$$\epsilon_\theta = \frac{\langle w_\theta \rangle}{C_\theta}. \quad (16)$$

The highest unweighting efficiency is reached when q_θ is equal to p_0 . In that case, all weights have a value of one and all events are accepted. Therefore, in order to maximize the unweighting efficiency, we need to optimize ψ_θ such that q_θ becomes close to p_0 . This is the same target as for reducing the variance of the integral estimator in the case that f is constant. The precise metric is different, however. For variance minimization, we minimize the square loss $\mathbb{E}(E_\theta^2)$ in Eq. (13), while for unweighting we minimize $\max_x w_\theta(x)$, which determines the unweighting efficiency ϵ_θ in Eq. (16). In other words, for unweighting we are first and foremost interested in a reduction of the maximum weight, while the variance cares about the average deviation from the mean. Note that in unweighting, it can be beneficial to intentionally overpopulate specific regions of phase space. This can happen, for example, when an event sample is meant to be used for several different analyses (looking at different observables f) and one of them needs a large number of events in the tail of some observable. If such an “enhancement” should be done with respect to an observable f , then it can be achieved by multiplying it to the event weight: $w_\theta^{\text{enh}}(x) = f(\phi(x)) \cdot w_\theta(x)$. In that case, the optimal q_θ is proportional to $p_0 \cdot (f \circ \phi)$, just as for the integration case.

A basic example of ψ_θ is the VEGAS algorithm, an adaptive importance sampler that factorizes the problem into one-dimensional piecewise-linear maps. The key idea of VEGAS is to construct ψ_θ as a product deformation map

$$\psi_\theta^{\text{Vegas}}(x) = \prod_{j=1}^d \psi_j(x_j), \quad (17)$$

where x_j denotes the j th component of the vector x and $\prod$ is the Cartesian product. The one-dimensional maps ψ_j are piecewise linear and can be adapted by varying the widths of the linear segments. In a Monte Carlo simulation, the adaption is typically performed in an initial optimization phase, and the bin widths are optimized such that the variance is distributed equally among the bins. The density $q_\theta^{\text{Vegas}} = (\psi_\theta^{\text{Vegas}})_* u_d$ is piecewise constant. Therefore, sampling can be performed by selecting a bin randomly and sampling uniformly between the bin edges.

While VEGAS can achieve large improvements in variance and unweighting efficiency in practice, it can fall short if p_0 not factorize. This can be alleviated by realizing $q(x) = \phi_* u_d(x)$ as a mixture density $q(x) = \sum_i \alpha_i q_i(x)$, where the components q_i exploit the known factorization of individual terms contributing to the target function $p(x)f(x)$. This *multichannel* approach, in combination with an optimization of the mixture weights α_i [9] and a VEGAS remapping of each component q_i [11], is the *de facto* state of the art in HEP. However, phase-space sampling is still a major challenge for complex scattering processes.

A more expressive alternative to VEGAS is given by normalizing flows [36,43,44], which use neural networks to parametrize ψ_θ . Through restrictions in their architecture, normalizing flows can be designed in a way so that their Jacobian determinant is tractable without having to invert the neural networks. Flows based on coupling layers [36], for example, implement the flow as a chain of discrete simpler steps. In this work, we propose to realize ψ_θ as a continuous normalizing flow [25], where the map is constructed implicitly by integrating a time-dependent vector field, which is implemented by a neural network. We argue that flow matching [26–29] is a useful training method for the continuous normalizing flow (CNF), since it does not require to numerically solve an ODE during training. We compare this method with normalizing flows based on coupling layers, and with the VEGAS algorithm, which is the default method used in the PEPPER event generator and therefore serves as our baseline. We demonstrate in Sec. V that the CNF is stable to train and outperforms the other approaches for the most challenging, i.e. highest dimensional, examples.

D. Phase-space sampling in high-energy physics

A central quantity in HEP simulations is the scattering cross section, which is a direct measure of the probability of the given process to occur. For hadronic collisions, e.g. at the LHC, it is given by

$$\begin{aligned} \sigma_{h_1 h_2 \rightarrow X} &= \sum_{i,j} \int_0^1 dx_1 \int_0^1 dx_2 f_i(x_1, \mu_F) f_j(x_2, \mu_F) \\ &\quad \times \hat{\sigma}_{ij \rightarrow X}(x_1, x_2, \mu_R, \mu_F), \end{aligned} \quad (18)$$

where the sum runs over all possible partons found within the incoming hadrons $h_{1,2}$, i.e. quarks, antiquarks, and gluons. The individual partonic contributions (called subprocesses) can be integrated separately. This means we can ignore the sum for now and concentrate on a single subprocess. The integration is performed over $x_{1,2}$, the light-cone momentum fractions of the partons i, j . Finally, we have the renormalization- and factorization scales $\mu_{R,F}$ and the functions $f_{i,j}$, which are called parton distribution functions (PDFs). They cannot be calculated from first principles and thus have to be determined from data. Typically, they are provided as precomputed grids in x and μ_F^2 and values between the knots are obtained using cubic interpolation. The PDFs are nonlinear and vary over several orders of magnitude.

The remaining ingredient of Eq. (18) is the partonic cross section $\hat{\sigma}_{ij \rightarrow X}$, which is an integral over the final-state four-momenta $p_f = (E_f, \vec{p}_f)$, with $f = 3, \dots, m$ and m being the number of incoming and outgoing particles. As a differential, it is given by

$$\begin{aligned}
d\hat{\sigma}_{ij \rightarrow X} &= \frac{1}{2E_1 E_2 |\vec{v}_1 - \vec{v}_2|} \left(\prod_f \frac{d^3 \vec{p}_f}{(2\pi)^3} \frac{1}{2E_f} \right) \\
&\times |\mathcal{M}_{ij \rightarrow X}(p_1, p_2 \rightarrow \{p_f\})|^2 (2\pi)^4 \delta^{(4)} \\
&\times \left(p_1 + p_2 - \sum_{k=3}^m p_k \right) \Theta(\{p_f\}). \quad (19)
\end{aligned}$$

We include the discontinuous cut function Θ , which depends on the final-state momenta, and assumes either the value 0 or 1. This implements the phase-space cuts, which are discussed in more details below. The squared matrix element, $|\mathcal{M}_{ij \rightarrow X}|^2$, is evaluated in perturbation theory, and is a complicated function that depends on the partonic process, $ij \rightarrow X$, and the incoming and outgoing particle momenta:

$$\begin{aligned}
&|\mathcal{M}_{ij \rightarrow X}(p_1, p_2 \rightarrow \{p_f\})|^2 \\
&\propto \sum_{\lambda_1, \dots, \lambda_d} \mathcal{A}^{\lambda_1, \dots, \lambda_m}(p_1, \dots, p_m) \mathcal{A}^{\lambda_1, \dots, \lambda_m}(p_1, \dots, p_m)^\dagger, \quad (20)
\end{aligned}$$

where most of the structure is expressed in the helicity amplitudes $\mathcal{A}$. Here, the helicity amplitudes include the external polarization vectors and spinors for the external helicities λ_i , and all required color factors. Certain helicity configurations $\lambda_1, \dots, \lambda_d$ can result in vanishing $\mathcal{A}$. These vanishing configurations can easily be identified before the sampling begins and can then be ignored therein. The number of nonvanishing helicity configuration is referred to as n_{hels} throughout this work.

The remaining component is the Lorentz-invariant phase space, which is needed to implement the symmetries of the system. These reduce the dimensionality of the partonic cross section from $4n_{\text{out}}$ (with $n_{\text{out}} = m - 2$ being the number of outgoing particles) to $3n_{\text{out}} - 4$. Together with the integrals over $x_{1,2}$, the total dimensionality is $d = 3n_{\text{out}} - 2$. For the LHC, Monte Carlo simulations typically have $n_{\text{out}} \lesssim 7$, such that $d \lesssim 19$.

The result of the integral for σ in Eq. (18) is needed for overall normalization, but can usually be estimated with sufficient accuracy without too much computational effort, at least at leading and next-to-leading order in perturbation theory. The real computational challenge lies in generating large event samples distributed according to $d\sigma$ in this high-dimensional phase space. They have to contain sufficiently many events to allow for flexible statistical comparisons of distributions for a variety of physics observables, both in the bulk and in the tails. A good phase-space sampler must therefore be efficient throughout phase space, across regions where the integrand varies by many orders of magnitudes.

We can now connect this discussion with the generic description in Sec. II A. The manifold M is given by all possible values for the momentum fractions $x_{1,2}$ and the final-state momenta $\{\vec{p}_f\}$, where the fractions assume

values between 0 and 1, and the momenta are constrained by energy-momentum conservation as implemented by the delta distribution in Eq. (19). The probability density function p is given by $p(x)dx = \sigma^{-1} d\sigma = \sigma^{-1} dx_1 dx_2 f_i f_j d\hat{\sigma}$, with the delta distribution removed. Finally, in Eq. (18), we consider the special case $f(x) = \sigma$. However, in practice, when evaluating the integral using Monte Carlo integration, one can store the generated sample of phase-space points and the corresponding values of $p(x)$ and reevaluate the integral for any $f(x)$ in a fast and straightforward way. This flexibility is one of the main advantages of using Monte Carlo sampling.

Now, to find an optimal map ψ_θ in the sense defined in Sec. II C, the task is to mimic the integrand of Eq. (18) closely, to improve the estimate of the integral or the unweighting efficiency. However, its structure makes this approximation challenging. Experimental and theoretical considerations require phase-space cuts Θ in Eq. (19), effectively setting the matrix element to zero for certain regions. These introduce discontinuities, often in regions where the integrand is large, since such cuts are commonly applied to remove singularities. Furthermore, with the matrix elements $|\mathcal{M}|^2$ being comprised of products of strongly peaking helicity amplitudes, see Eq. (20), the integrand becomes multimodal, with potentially very narrow peaks separated from each other. Finally, the integration dimensions are strongly correlated. This correlation is potentially increased by the fact that, as indicated by the delta functions in Eq. (19), the integration is only performed over a submanifold of reduced dimensionality (as defined by the Lorentz-invariant phase space). While this manifold can be mapped out efficiently, the mapping can introduce further correlations in the process.

III. BACKGROUND: CONTINUOUS NORMALIZING FLOWS

In this section, we describe how CNFs provide a viable solution to the sampling problem described in Sec. II A. We begin by introducing the concept of CNFs in Sec. III A. There we also show how to generate samples with the flow and how to evaluate its density. In Sec. III B, we show how a CNF can be trained efficiently, without solving the flow differential equation, using the flow matching method. We conclude in Sec. III C with a description of how to condition the flow on inputs with discrete values, like the helicity states of external particles in a scattering event.

A. Flows from time-dependent vector fields

As described in Sec. II, we want to generate samples from a target density² $p: \mathbb{R}^d \rightarrow \mathbb{R}$ but are only able to

²For simplicity in the notation, we here ignore the map ϕ and drop the subscript 0 accordingly. We also ignore the function f but note that it could be added to all equations with ease.

sample from a latent density q_0 . We would like to find a map $\psi: \mathbb{R}^d \rightarrow \mathbb{R}^d$ and use it to transform samples $x \sim q_0$ such that $\psi(x) \sim p$, or at least that they are close in distribution. Normalizing flows provide such a map in the form of a parametrized diffeomorphism that can be trained on data. Methods differ in how exactly the map is constructed and which loss functional is used for training. An example is a discrete-time flow based on a finite sequence of simple maps, $\psi = \psi_k \circ \dots \circ \psi_1$. These can be built, for example, with coupling layers [36] and are typically trained with a Kullback-Leibler (KL) loss. For phase space sampling in HEP, this option has been explored in Refs. [18–22,45,46]. Below, we show how ψ can also be realized as a continuous normalizing flow, which is the continuous-time analogue. As shown in Sec. V, we observe that CNFs are easier to train and scale better to higher dimensions than discrete-time flows.

Let $v_t: [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ be a smooth time-varying vector field. This vector field describes the velocity of a point $x_t \in \mathbb{R}^d$ at time $t \in [0, 1]$. The motion of the point is determined by the ODE

$$\frac{dx_t}{dt} = v_t(x_t). \quad (21)$$

We denote the solution to this ODE by $\psi_t: \mathbb{R}^d \rightarrow \mathbb{R}^d$ and the initial conditions are given by $\psi_0(x_0) = x_0$ (or $\psi_0 = \text{Id}$). The map ψ_t is the time-dependent flow of the vector field v_t . This can be used to generate samples at arbitrary times t : We sample $x_0 \sim q_0$ and apply the map ψ_t , for which we have to integrate the ODE (21):

$$x_t = \psi_t(x_0) = \int_0^t v_{t'}(\psi_{t'}(x_0)) dt'. \quad (22)$$

This allows us to determine the trajectory of any point. By the Picard-Lindelöf theorem, if v_t is Lipschitz-continuous in space and continuous in time, then the solution to the ODE (21) is unique and trajectories of different initial points never intersect, which ensures that ψ_t has an inverse.

Another quantity we are interested in is the probability density $q_t(x_t)$ for a generated point x_t at time t . To derive this, we note that due to local conservation of probability, the density has to obey the continuity equation

$$\frac{\partial}{\partial t} q_t(x) + \nabla \cdot (q_t(x) v_t(x)) = 0. \quad (23)$$

Plugging in $x = x_t = \psi_t(x_0)$, cf. Eq. (22), it follows that

$$\frac{d}{dt} q_t(\psi_t(x_0)) + q_t(\psi_t(x_0)) \nabla \cdot v_t(\psi_t(x_0)) = 0, \quad (24)$$

which simplifies to

$$\frac{d}{dt} \log q_t(\psi_t(x_0)) + \nabla \cdot v_t(\psi_t(x_0)) = 0. \quad (25)$$

So to evaluate $q_t(x_t)$, we need to solve this ODE by integrating it numerically from 0 to t . In practice, sampling and density evaluation can be done jointly by solving the system

$$\frac{d}{dt} \begin{bmatrix} \psi_t(x_0) \\ \log q_t(\psi_t(x_0)) \end{bmatrix} = \begin{bmatrix} v_t(\psi_t(x_0)) \\ -\nabla \cdot v_t(\psi_t(x_0)) \end{bmatrix} \quad (26)$$

with initial conditions

$$\begin{bmatrix} \psi_t(x_0) \\ \log q_t(x_0) \end{bmatrix}_{t=0} = \begin{bmatrix} x_0 \\ \log q_0(x_0) \end{bmatrix}. \quad (27)$$

We can also compute the density for an arbitrary point x_1 at time $t = 1$, which has not necessarily been generated with the flow. Using an auxiliary function a_t , the appropriate reverse ODE is

$$\frac{d}{dt} \begin{bmatrix} \tilde{\psi}_t(x_1) \\ a_t \end{bmatrix} = \begin{bmatrix} -v_t(\tilde{\psi}_t(x_1)) \\ \nabla \cdot v_t(\tilde{\psi}_t(x_1)) \end{bmatrix} \quad (28)$$

and has to be integrated from 1 to 0 with the initial conditions at $t = 1$,

$$\begin{bmatrix} \tilde{\psi}_t(x_1) \\ a_t \end{bmatrix}_{t=1} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}. \quad (29)$$

The auxiliary function can then be used to determine the density:

$$\log q_1(x_1) = \log q_0(x_0) - a_0. \quad (30)$$

Above, we have defined the domain of ψ_t to be $\mathbb{R}^d$ instead of the unit hypercube U used in Sec. II C. This is the natural way to define the flow, because it allows the vector field v_t to be unconstrained with regard to boundaries. In consequence, the natural (uninformed) base distribution is a standard normal. To map the output x_t of the flow to U , we use the element-wise sigmoid transform:

$$\text{sig}: \mathbb{R} \rightarrow [0, 1], x \mapsto \text{sig}(x) = \frac{1}{1 + e^{-x}}. \quad (31)$$

To account for the associated change in density, we have to multiply with the corresponding Jacobian determinant.

B. Simulation-free training with flow matching

So far we have discussed how the map ψ_t can be constructed from a velocity field v_t to generate samples x_t and how we can evaluate the density for a given point x_1 . What is still missing is a way to find a map ψ_t (or velocity field v_t) such that the density $q_1(x)$ closely approximates

the target density $p(x)$. To this end, we use a model $v_{t,\theta}$ with trainable parameters θ to parametrize the vector field v_t . A neural network is a convenient choice for $v_{t,\theta}$, since Eqs. (26) and (28) imply that for sampling or density evaluation, we only need to evaluate $v_{t,\theta}$ and not its gradient or inverse. The vector field should be continuously differentiable though.

Assume we are given training data in the form of samples $x_1 \sim p(x)$. We want to use these data to adapt $v_{t,\theta}$. Given that we can evaluate the model density $q_{t,\theta}$, using Eqs. (28) and (30), an obvious way would be maximum likelihood estimation (MLE), i.e. minimizing the KL loss between $q_{t,\theta}(x_1)$ and $p(x_1)$, as proposed in [25]. However, this approach has two major drawbacks: First, training is slow because in each step the ODE (28) has to be integrated with a numerical ODE solver, which needs many time steps to avoid large errors from the time discretization. Second, there is no direct gradient information for the vector field for intermediate times between 0 and 1, since only the endpoint of the flow is relevant for the KL loss. While there are approaches to regularize the flow, see e.g. Refs. [47,48], these still require simulation during training.

The drawbacks of maximum likelihood training motivated the development of simulation-free training methods [26–29]. The idea is to directly match the vector field $v_{t,\theta}$ to a target vector field u_t , which generates the target density p . There are various ways to construct an admissible u_t . We briefly outline the strategy of Refs. [26,49] via conditional vector fields. Let

$$u_t(x|x_0, x_1) = x_1 - x_0, \quad (32)$$

with $x_0 \sim q_0$ and $x_1 \sim p$ being samples from the base and target distribution, respectively. Clearly, a particle starting at x_0 will flow to x_1 along a straight line when following $u_t(\cdot|x_0, x_1)$, i.e.

$$x_t = tx_1 + (1-t)x_0. \quad (33)$$

Let now

$$u_t(x) = \mathbb{E}_{\substack{(x_0, x_1) \sim \pi, \\ x_t = x}} [u_t(x|x_0, x_1)]. \quad (34)$$

It can be shown [26,29,49] that the flow according to u_t transforms q_0 into p as long as the marginals of the joint law π are q_0 and p . For instance, we can set $\pi(x_0, x_1) = q_0(x_0) \otimes p(x_1)$ (independent coupling). Note that we condition the expectation to pairs (x_0, x_1) where x_t , Eq. (33), moves through the query point x . It is now natural to fit $v_{t,\theta}(x)$ to $u_t(x)$, e.g. via

$$\mathbb{E}_{\substack{(x_0, x_1) \sim \pi, \\ t \sim U_1}} \|v_{t,\theta}(x_t) - u_t(x_t)\|^2. \quad (35)$$

Note that evaluating u_t at specified points x is cumbersome due to the conditioning in Eq. (34). Fortunately, one can

show [26,49] that Eq. (35) has the same minimizers as

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\substack{(x_0, x_1) \sim \pi, \\ t \sim U_1}} \|v_{t,\theta}(x_t) - u_t(x_t|x_0, x_1)\|^2. \quad (36)$$

The flow matching objective, Eq. (36), has attractive properties. It is local in space and time, so it can be evaluated fast and without solving an ODE. Also, in contrast to a KL loss, the minimizer is unique (on the support of the data). In consequence, sample-based training becomes easy.

More generally, one can add noise to the interpolation to increase robustness. This is done by adding noise with a small standard deviation σ_{noise} to the individual straight lines x_t . This way, the sampled data cover more volume. For details, see Refs. [26,49].

We note that despite using a linear interpolation in Eq. (33), the flow trajectories x_t will, in general, not be straight lines. If the independent coupling is used, the samples x_0 and x_1 are paired randomly in Eq. (36) and the vector field $v_{t,\theta}$ will converge towards a local average of the interpolants. At time $t = 1$, it will still approximate the desired target p but when the trajectories are highly curved, sampling from the flow becomes slow because many small time steps are needed to achieve an accurate result with a numerical ODE solver. In the case of perfectly straight trajectories, a single step would suffice. Several approaches have been proposed to straighten the flow. In Ref. [29], the authors propose an iterative procedure, where the first iteration uses the independent coupling and, in each subsequent iteration, the coupling π is realized by the flow of the vector field $v_{t,\theta}$ trained in the previous iteration. This means that $x_0 \sim q_0$ and $x_1 \sim q_1 = (\psi_\theta)_* q_0$ (which implies the method is not simulation-free). The authors show that each training renders the flow trajectories more straight. However, it also accumulates estimation error, since the samples are not drawn from p anymore. Therefore, the authors recommend to not apply too many iterations. Another approach is to use a coupling between x_0 and x_1 that already encourages straight trajectories during a single training iteration. This can be done by solving the optimal transport (OT) problem and drawing x_0 and x_1 from the OT plan $\pi(x_0, x_1)$. Since this is computationally intractable for large data sets, one can instead use a minibatch approximation where the OT plan is computed only for minibatches of data. The resulting trajectories are not perfectly straight due to the bias from the approximation but it has been shown that even small batch sizes can lead to significant straightening [49,50]. In this work, we focus on the quality of the trained flows and do not investigate the use of straightening techniques. We note, however, that their application would be useful to reduce the sampling costs.

We make two modifications to the flow matching objective. First, in Sec. II A we have used the map ϕ to map our target distribution to the unit hypercube U , which means that our training data is also defined on that domain.

As noted in the previous section, the flow is defined on $\mathbb{R}^d$. Our solution for sampling was to use a sigmoid transform. Here, we need to apply its inverse, the logit transform,

$$\text{logit}: [0, 1] \rightarrow \mathbb{R}, \quad x \mapsto \text{logit}(x) = \ln \frac{x}{1-x}, \quad (37)$$

to map the training data to $\mathbb{R}^d$. To avoid numerical issues when the input to the logit is large in magnitude, we compose it with an affine transformation, $x \mapsto (1 - \epsilon)x + \epsilon/2$. We found $\epsilon = 1 \times 10^{-6}$ to work well in our case. For sampling, the inverse transformation has to be applied accordingly.

The second modification is related to the fact that, in contrast to what is assumed in Eq. (36), we do not sample $x_1 \sim p$ since this is our goal in the first place. Instead, we sample from the uniform distribution u_d . We can still use the flow matching objective by multiplying it with the importance weight w , cf. Eq. (7). This accounts for the mismatch in density. The approach is a variant of the energy conditional flow matching objective proposed in Ref. [49]. Of course, the importance sampling can lead to a high variance of the Monte Carlo estimator of Eq. (36) due to fluctuations in the weights. This is especially problematic in high dimensions. To keep this under control, it is important to use a map ϕ that reduces the variance to such an extent that model training becomes feasible. This requires a deep understanding of the physics involved. After training, the flow is itself a good sampler that can be used to generate new training data with a narrower distribution of weights. This suggests an iterative training scheme, where the flow trained in one iteration is used to generate the training data by sampling from $q_\theta = (\psi_\theta)_* u_d$ for the next iteration with weights according to (15). We note that VEGAS could also be used to generate the data for the first iteration. Similarly, a Markov-Chain Monte Carlo sampler could be used [51].

C. Helicity conditioning

Normalizing flows can be used to model conditional probability density functions. This is particularly interesting in combination with the PEPPER event generator, which allows to evaluate the sum over squared helicity amplitudes in Eq. (20) through a Monte Carlo sampling of the non-vanishing helicity configurations. In this case, the sampling of the particle momenta can be conditioned on the discrete helicity configuration corresponding to the helicity amplitude to be evaluated. Some of the helicity configurations can be highly correlated and through the use of neural networks, normalizing flows can be expected to learn these correlations well. This is a clear advantage in comparison to VEGAS, which would need to adapt to each helicity configuration individually, with poor convergence expected for configurations with low probability. By default, PEPPER uses a single VEGAS map for all helicity configurations, which averages over their respective distributions.

We note that, in principle, normalizing flows could be conditioned on any variable. In Ref. [52] this has been used for the sectors occurring in next-to-next-to-leading order QCD calculations with the sector-improved residue subtraction method. Other examples include the different partonic channels in scattering processes with hadronic initial states and jet final states or the different multiplicities contributing to inclusive cross sections, both of which we will explore in the future. Here we describe the general method.

Let k be the number of different discrete states. Each state is assigned an integer value, denoted by $h \in 1 \dots k$. To condition a continuous normalizing flow on h , we use a conditional vector field $v_{t,\theta}(\cdot|h)$. By integrating the ODE, this can be used to sample from a conditional model density $q_{1,\theta}(\cdot|h)$. Since the vector field is implemented by a neural network, we have to use a continuous representation of h to feed the values into the network. To this end, we use an embedding layer, which is trained jointly with the vector field. The representation from the embedding layer is then concatenated to the other input variables and fed into the input layer of the network. The nonvanishing helicity configurations themselves are sampled randomly, with weights determined from training data by their respective contribution to the integral.

IV. APPLICATION TO LHC EVENT SAMPLE PRODUCTION

Here we present our approach for applying our novel neural network optimization method to single partonic channels in two standard-candle processes at the LHC, lepton and top pair production in association with additional jets, at a center-of-mass energy of $\sqrt{s} = 14$ TeV. For both processes, we focus on the single partonic channel that contributes most of the overall cross section, which is $d\bar{d} \rightarrow e^+e^- + ng$ for lepton pair production and $gg \rightarrow t\bar{t} + ng$ for top pair production.

A. CHILI phase-space mapping

In this work, we make use of the CHILI phase-space mapping [35] to provide the map $\phi: U \rightarrow M$, see Eq. (2). The phase space for the final state particle f in Eq. (19) can be expressed in terms of its transverse momentum $p_{f,\perp}$, rapidity y_f and azimuthal angle φ_f via

$$\frac{d^3 \vec{p}_f}{(2\pi)^3} \frac{1}{2E_f} = \frac{1}{16\pi^2} dp_{f,\perp}^2 dy_f \frac{d\varphi_f}{2\pi}. \quad (38)$$

Using this parametrization for all final-state QCD partons allows common transverse momentum and rapidity cuts to be efficiently applied. In the phase-space generation, we generate the transverse momenta according to $dp_{f,\perp}^2 / (2p_{\perp,c} + p_{f,\perp})^2$ where $p_{\perp,c}$ denotes an adjustable parameter used to effectively map out potential phase-space

cuts. The remaining variables are distributed uniformly, the rapidity between a maximum and minimum rapidity determined by external cuts and the azimuthal angle between 0 and 2π .

To satisfy energy-momentum conservation, we use the above mapping for all but the last final state QCD parton. The phase-space integration for the last final-state momentum is then performed in combination with the integration over the light-cone momentum fractions $dx_{i,j}$ in Eq. (18). This combined integration allows to trivially satisfy energy-momentum conservation, by using the remaining final-state momentum to balance transverse momenta and the initial state momenta for the overall conservation. For vector-boson production, we add an additional s -channel propagator mapping according to the well-known two-body decay formula [35] with the virtuality of the intermediate particle distributed according to a Breit-Wigner distribution.

The key advantage of this mapping is its simplicity. The integrator consists of a single channel, while still performing similarly compared to more involved algorithms for a wide range of LHC processes [34]. The single-channel setup is an ideal starting point for the combination with remapping techniques using a trainable map ψ_θ , as introduced in Sec. II C, since no multichannel and consequently more complicated techniques such as [9,11] need to be considered.

B. PEPPER event generator

We use the Monte Carlo generator PEPPER [34,53,54] to perform the calculation of the perturbative scattering amplitudes. PEPPER is a novel portable parton-level event generator with Graphics Processing Unit (GPU) acceleration, developed specifically for computationally expensive processes at the LHC.

Momentum configurations are sampled using PEPPER's internal implementation of the CHILI phase-space generator [35]. While CHILI is comparatively simple and was thus easily ported to GPU [53], it has been shown to achieve a similar performance as more complex established generators, such as COMIX [55], which is part of the SHERPA framework [56–58].

The matrix elements as given in formula (20) are evaluated by summing over the color quantum numbers, using a minimal color decomposition [59–61]. The helicity quantum numbers are sampled instead. To achieve a good sampling efficiency, PEPPER normally samples them according to their relative contribution to the overall variance of the Monte Carlo sample with respect to the total cross section. This optimization is not normally correlated with the optimization of the phase-space sampling, and is essentially equivalent to an additional dimension in the VEGAS optimizer with the number of bins being equal to the number of nonvanishing helicities n_{hels} . However, when applying the neural network based optimization we instead

jointly optimize the phase-space and helicity sampling and thus expose the optimizer not only to the correlations between the different phase-space variables, but also between these variables and the helicity.

To write out parton-level events for training the external flow models (the VEGAS model is implemented and trained within PEPPER) we utilize PEPPER's LHEH5 event output format. The format is based on the HDF5 database library [62], accessed through the HIGHFIVE header library [63], and has been specified in Refs. [64,65]. We extend the specification for the purpose of this study to include the random numbers for each event which are required for the training, i.e. for the phase-space point and the helicity configuration.

After model training (see Sec. IV D), the generated random numbers for each event must be read in by PEPPER again to evaluate the performance of the optimization. For this, we have added a corresponding reader to PEPPER, which allows to use externally generated random numbers and phase-space weights instead of internally generated random numbers and weights.

The new input/output features have been released with version 1.2 of PEPPER. They allow for very straightforward applications of external optimization models via simple file-based interfaces, which are nevertheless flexible and scalable on High-Performance Computing due to HDF5's fast database structure and Message Passing Interface support.

C. Event generation parameters

As discussed, the two partonic processes we study are $d\bar{d} \rightarrow e^+e^- + ng$ (lepton-pair production) and $gg \rightarrow t\bar{t} + ng$ (top-pair production) at $\sqrt{s} = 14$ TeV. We set the number of additional gluons to match the currently used jet multiplicities in experimental simulation setups, with $n = 1 \dots 5$ for lepton-pair production and $n = 0 \dots 4$ for top-pair production. In both cases, we use the NNPDF3.0 PDF set [66] to evaluate the PDFs $f_{i,j}$ in Eq. (18), with the running strong coupling evaluated according to the set, using the LHAPDF6 library [67]. The renormalization and factorization scales are set to $\mu_R^2 = \mu_F^2 = H_T^2/2$ for lepton-pair production and $\mu_R^2 = \mu_F^2 = H_T^2/2$ for top-pair production [68]. The electroweak parameters are $\sin^2 \theta_w = 0.23155$ and $\alpha = 1/128.80$. We use the Z and top masses $m_Z = 91.1876$ GeV, $m_t = m_{\bar{t}} = 173.21$ GeV, respectively, and set the Z width as $\Gamma_Z = 2.4952$ GeV. All remaining quarks are considered to be massless.

We apply the following conventional phase-space cuts. For lepton-pair production, we require the invariant mass to satisfy $66 \text{ GeV} \leq m_{e^+e^-} \leq 116 \text{ GeV}$. For both processes, we require all massless final-state partons to satisfy $p_{T,j} > 30 \text{ GeV}$, $|\eta_j| < 5$ and $\Delta R_{ij} > 0.4$, for the transverse momentum p_T and the pseudorapidity η of parton j , and for the pair distance ΔR between two partons i, j , respectively.

D. Model and training parameters

For all setups and multiplicities, the VEGAS grids have 100 bins in each dimension. The bin widths are optimized using 15 optimization steps. The number of points is determined, such that the total number of points for the highest multiplicity becomes roughly 3×10^8 . This results in 2.5×10^5 training points per nonvanishing helicity configuration in the lepton pair production and 8×10^4 points in the top-quark pair production case.

The NFs are optimized in 8 iterations, similar to how VEGAS is trained. In contrast to Refs. [20,69], we do not use a buffer but completely replace the data in each iteration. This allows us to separate training and data production, so that both can be made efficient independently, and to use a minimal interface to PEPPER, which only needs to read in random numbers and corresponding Jacobians and to write out weights. For the first iteration, 10^8 weighted events are generated with PEPPER without using any remapping. In the subsequent iterations, the trained flows are used to generate 10^7 nonuniform random numbers per iteration as inputs, followed by a calculation of the corresponding momenta and weights with PEPPER, which then serve as training data. An embedding layer is used to provide a continuous representation of the nonvanishing helicity configurations, with its width being equal to the number of such configurations n_{hels} . In all cases, this representation is concatenated to the other input variables and fed into the input layer of each neural network.

For the coupling flows, the minimal number of coupling layers able to model all correlations between inputs, $2\lceil \log_2 d \rceil$ [70], is used. The implementation is based on the NFLOWS PYTHON package [71], using rational-quadratic coupling layers [38] with 16 spline knots per dimension. Between two coupling layers, the inputs are permuted with a binary permutation scheme. Each coupling transform is parametrized by a multilayer perceptron (MLP) with ReLU activation functions. The widths of the hidden layers are

growing exponentially, such that each hidden layer has twice as many nodes as the preceding layer. Hidden layers are added until the final one has more nodes than the output layer. The size of the MLPs is thus dynamic and can vary between the coupling layers. For the coupling flows, 20% of the training data are used for validation and early stopping based on the validation loss is used. This was found necessary to avoid overfitting. The ADAMW optimizer [72] is used with a batch size of 2^{16} and its learning rate is decreased on plateaus. For the lowest multiplicity processes, $e^+e^- + 1, 2$ gluons and $t\bar{t} + 1, 2$ gluons, the initial learning rate is set to 10^{-3} . However, this leads to unstable training for the higher multiplicities, requiring to decrease the learning rate. For $e^+e^- + 3, 4, 5$ gluons and $t\bar{t} + 3$ gluons, a value of 10^{-4} is thus used. For $t\bar{t} + 4$ gluons it is further decreased to 10^{-5} .

For the ODE flows, each vector field is implemented as a MLP with 4 hidden layers of width 512, using Scaled Exponential Linear Unit activation functions. As proposed in Ref. [73], the time t is sampled from the power-law distribution $p_\alpha(t) \propto t^{1/(1+\alpha)}$, with $\alpha = 1$. A 16-dimensional Fourier feature mapping [74] is used to encode the time parameter and the encoding is concatenated to the other input variables. The ODE flows are trained for 600 epochs with a batch size of 2^{19} using the ADAMW optimizer with a learning rate of 10^{-3} . Neither early stopping nor learning rate decay is used, as neither was found to improve the results. For ODE solving, necessary for sampling and density evaluation with the flow, the DORMANDPRINCE45 solver [75], as implemented in the TORCHDYN PYTHON package [76], is used. The absolute and relative tolerances of the adaptive solver are set to 10^{-4} .

For all three mappings, VEGAS, coupling flows and ODE flows, the total numbers of trainable parameters for the different processes are given in Table I. The numbers increase with the dimensionality and the number of helicity configurations. Compared to the normalizing flows,

TABLE I. Number of parameters for the different mappings. The parameters include both the number of parameters required for the phase-space remapping and the choice of the helicity configuration. We also give the number of final-state gluons n , the resulting phase-space dimensionality $d = 3n_{\text{out}} - 2$, and the number of nonvanishing helicity configurations n_{hels} .

Process	n	d	n_{hels}	Model parameters		
				VEGAS	Coupling flow	ODE flow
$d\bar{d} \rightarrow e^+e^- + ng$	1	7	8	700	616 880	1 078 878
	2	10	16	1005	1 438 515	1 086 249
	3	13	32	1318	2 450 475	1 098 300
	4	16	64	1674	3 563 647	1 120 863
	5	19	128	2008	5 215 120	1 169 058
$gg \rightarrow t\bar{t} + ng$	1	7	32	724	614 402	1 092 150
	2	10	64	1053	1 571 755	1 114 713
	3	13	128	1414	2 411 499	1 162 908
	4	16	256	1839	2 844 223	1 280 799

VEGAS uses about three orders of magnitude less parameters. For the two kinds of flows, the numbers are of the same order of magnitude.

E. Evaluation

After finishing the training of the models, we freeze their parameters and generate samples of 10 million weighted events that pass the phase-space cuts for each model and each final-state jet multiplicity for both processes. For each sample, we calculate the cut efficiency (the fraction of events that pass the phase-space cuts), the relative Monte Carlo error of the cross-section integral, the Kish effective sample size N_{eff} , Eq. (10), and the effective unweighting efficiencies $\epsilon_{0.01}$ and $\epsilon_{0.001}$. This is repeated ten times with different random seeds. Mean values and standard deviations for the different figures of merit are then estimated from these repeated samples.

V. RESULTS

The discussion of the results is split into two parts. Section VA presents measurements of several Monte Carlo quality metrics, in particular, the unweighting efficiencies, comparing the newly proposed ODE flow optimization with VEGAS and coupling-flow based optimization. Section VB then takes into account the evaluation times of the different methods and lists effective run-time gains in the full parton-level event generation, as well as break-even points considering the increased training times when compared to VEGAS.

A. Phase-space sampling efficiencies

We present results for the mean and standard deviation of several quality metrics: phase-space cut efficiency, relative integration error, effective sample size N_{eff} and unweighting efficiencies $\epsilon_{0.01}$ and $\epsilon_{0.001}$. The results are given in Table II for both $d\bar{d} \rightarrow e^+e^- + ng$ and $gg \rightarrow t\bar{t} + ng$, for the unoptimized mapping (“Identity”), a VEGAS-optimized mapping, and mappings that are implemented as coupling flows and ODE flows. As discussed in Sec. II, these metrics capture different sampling efficiency aspects. For the relative integration error, lower values are better, while in all other cases, higher values are better. We find that our newly proposed ODE flow method has the best performance in almost all cases. The only exception is the phase-space cut efficiency, where all optimization methods perform well, with only small differences between them. The results for the unweighting efficiencies differ far more significantly, in particular, for the computationally most difficult highest gluon multiplicities n in the study.

For $d\bar{d} \rightarrow e^+e^- + ng$ with $n = 5$ we find $\epsilon_{0.01} = 3.68(5)\%$ for the ODE flow, which is 13 times higher than for the coupling flow’s 0.29(9)% and 153 times higher compared to VEGAS’ 0.024(8)%. For $\epsilon_{0.001}$, we find even

larger improvements of 43 and 184 when comparing to the coupling flow and to VEGAS, respectively.

For $gg \rightarrow t\bar{t} + ng$ with $n = 4$, the ODE flow’s efficiency is $\epsilon_{0.01} = 12.53(3)\%$, which is 8 times higher than the coupling flow’s 1.6(5)%, and 17 times higher compared to VEGAS’ efficiency, 0.74(2)%. For $\epsilon_{0.001}$, the improvements over the coupling flow and VEGAS are 144 and 25, respectively, with the coupling flow’s efficiency falling below VEGAS’ efficiency in this case.

The unweighting efficiency improvements of the flow-based methods over VEGAS are consistently higher for $d\bar{d} \rightarrow e^+e^- + ng$ when compared to $gg \rightarrow t\bar{t} + ng$ at the same n . For $\epsilon_{0.01}$ and $n = 4$, we find a factor of 128 for the former and a factor of 17 for the latter. The main reason is that VEGAS performs significantly better for $gg \rightarrow t\bar{t} + ng$, for all n . This is not only true for the unweighting efficiency ϵ , but to a similar degree also for the relative Monte Carlo error, indicating that this cannot be traced back to the optimization target of VEGAS. An explanation would be that the prerequisite of VEGAS—i.e. that the distribution is factorizable into each of its dimensions—is fulfilled to a lesser degree for $d\bar{d} \rightarrow e^+e^- + ng$. Here, CHILI introduces an s -channel for the Z -boson resonance, and there is a more pronounced dependence on the helicity configuration.

In terms of the relative integration error, we find similar improvements. The normalizing flows achieve significant improvements for all multiplicities considered. For $d\bar{d} \rightarrow e^+e^- + ng$ with $n = 5$ we find a value of 0.061(4)% for the ODE flow, which is about 3 times less than the coupling flow and about 9 times less than VEGAS. Considering that the integral error scales as $1/\sqrt{N}$ we need to square these numbers to determine the improvement in efficiency, i.e. the reduction in the number of points needed to determine the integral with the same error. This results in factors of 11 and 73, respectively, in comparison to the coupling flow and VEGAS. The corresponding factors for $gg \rightarrow t\bar{t} + ng$ with $n = 4$ are 44 and 17, where in analogy to the results for the unweighting efficiency the coupling flow produces a larger error than VEGAS. Over all multiplicities, the improvements range between 14 and 224.

The effective sample size also benefits significantly from the normalizing flows. While the identity mapping achieves values of $N_{\text{eff}} > 1\%$ only for the lowest multiplicities, VEGAS achieves this in all cases except $d\bar{d} \rightarrow e^+e^- + 5g$. The normalizing flows, however, lead to much higher values and with the ODE flow the effective sample size is above 25% in all cases considered. In accordance with the results for the unweighting efficiency and the relative integration error, the highest multiplicities favor the ODE flow over the coupling flow. For the lowest multiplicity, $n = 1$, the effective sample size with the normalizing flows is close to 100%. Accordingly, the improvement with respect to VEGAS, which achieves about 25%, is close to the maximum factor of about 4. The improvement factor increases with multiplicity and is highest for $d\bar{d} \rightarrow e^+e^- + 5g$, where the

TABLE II. Comparison of the different mappings for the $d\bar{d} \rightarrow e^+e^- + ng$ and $gg \rightarrow t\bar{t} + ng$ examples using samples with 10M nonzero weight events. Columns report the phase-space cut efficiency (Cut eff.), relative integration error (Rel. err.), effective sample size N_{eff} and unweighting efficiencies $\epsilon_{0.01}$ and $\epsilon_{0.001}$. The relative error of the total cross section estimate has been determined from 10M events including zero weights. The uncertainties have been obtained as the standard deviation of 10 independent samples generated with the same mapping (without re-training). All errors and uncertainties are rounded up in the last digit given. The best value in each column is shown in bold.

Process	n	Mapping	Cut eff./% $\uparrow$	Rel. err./% $\downarrow$	$N_{\text{eff}}/\%$ $\uparrow$	$\epsilon_{0.01}/\%$ $\uparrow$	$\epsilon_{0.001}/\%$ $\uparrow$
$d\bar{d} \rightarrow e^+e^- + ng$	1	Identity	86.88(2)	0.0943(1)	11.651(8)	3.840(6)	3.00(2)
		VEGAS	98.002(6)	0.054 53(5)	25.72(2)	8.12(3)	4.24(8)
		Coupling	99.927(1)	0.0034(2)	99.1(2)	78.68(5)	49.4(3)
		ODE	99.886(2)	0.003 64(3)	98.98(3)	78.91(2)	57.6(2)
	2	Identity	75.55(2)	0.247(6)	2.2(2)	0.23(2)	0.08(3)
		VEGAS	95.287(7)	0.0949(4)	10.52(8)	1.67(2)	0.67(5)
		Coupling	99.450(2)	0.02(4)	79(28)	48.7(6)	13(5)
		ODE	98.383(6)	0.009 90(9)	92.8(2)	35.0(2)	15.5(2)
	3	Identity	65.04(2)	1.4(9)	0.15(8)	0.007(7)	0.003(2)
		VEGAS	92.75(2)	0.175(3)	3.4(1)	0.32(2)	0.10(3)
		Coupling	92.48(2)	0.030(7)	58(9)	15.5(2)	3.0(7)
		ODE	90.60(1)	0.020 85(6)	77.2(2)	22.28(6)	11.0(2)
	4	Identity	55.18(2)	2.8(9)	0.02(1)	0.0011(6)	0.0011(6)
		VEGAS	88.87(2)	0.32(2)	1.1(2)	0.07(1)	0.02(1)
		Coupling	85.24(1)	0.053(3)	31(2)	4.43(7)	1.1(2)
		ODE	83.099(9)	0.0386(2)	48.7(3)	8.96(4)	3.95(9)
	5	Identity	46.09(2)	8(7)	0.004(3)	0.0004(2)	0.0004(2)
		VEGAS	84.29(1)	0.52(4)	0.4(1)	0.024(8)	0.007(3)
		Coupling	82.90(2)	0.2(1)	4(2)	0.29(9)	0.03(3)
		ODE	77.69(2)	0.061(4)	28(2)	3.68(5)	1.29(8)
$gg \rightarrow t\bar{t} + ng$	1	Identity	77.70(2)	0.2352(6)	2.29(2)	0.434(6)	0.27(2)
		VEGAS	98.368(4)	0.053 58(6)	26.27(4)	7.10(5)	3.2(1)
		Coupling	99.9943(3)	0.004 66(3)	97.92(3)	67.17(4)	45.6(3)
		ODE	99.9843(4)	0.005 15(5)	97.47(5)	65.21(3)	45.50(9)
	2	Identity	65.40(2)	0.47(2)	0.68(3)	0.063(5)	0.021(7)
		VEGAS	95.997(6)	0.0676(2)	18.70(4)	3.55(2)	1.49(4)
		Coupling	99.205(3)	0.013(2)	85(2)	35.9(2)	10.2(9)
		ODE	98.328(5)	0.0205(3)	71.7(5)	24.39(4)	13.35(8)
	3	Identity	54.27(3)	1.1(2)	0.13(3)	0.008(3)	0.003(1)
		VEGAS	92.640(9)	0.0904(3)	11.79(8)	1.60(2)	0.59(4)
		Coupling	94.008(6)	0.0246(2)	66.4(4)	15.26(6)	6.3(3)
		ODE	93.68(2)	0.027 68(4)	60.49(6)	15.26(3)	8.7(2)
	4	Identity	44.36(2)	3(1)	0.03(2)	0.0011(8)	0.0010(7)
		VEGAS	88.304(8)	0.123(1)	7.0(1)	0.74(2)	0.23(3)
		Coupling	91.269(5)	0.2(2)	8(6)	1.6(5)	0.04(6)
		ODE	88.456(8)	0.0300(2)	59.6(5)	12.53(3)	5.76(9)

ODE flow produces a 70 times larger effective sample size in comparison to VEGAS. With respect to the coupling flow, the factor is about 9 in this case.

The improved sampling by the ODE flow method also manifests itself in a narrower weight distribution, as shown in Fig. 2 for $d\bar{d} \rightarrow e^+e^- + ng$ with $n = 5$ (left) and $gg \rightarrow t\bar{t} + ng$ with $n = 4$ (right). A narrower distribution is directly related to a larger effective sample size N_{eff} . As discussed in Sec. II C, for a good unweighting efficiency the most relevant feature of the weight distribution is the steepness of its tail towards large values w , since this will

determine the value of the denominator, $w_{\text{max,eff}}$, in Eq. (9). Indeed, the ODE flow yields a significantly steeper right tail in the weight distribution for both cases, compared to the other methods including the coupling flows.

To study the scaling behavior with the number of final-state gluons n , we compare the values of $\epsilon_{0.001}$ in Fig. 3. We find that the ratio of the efficiency for the ODE flow over the efficiency of VEGAS generally increases with n . On the other hand, the coupling flows at first show similar improvements over VEGAS, but for the largest n decrease in efficiency significantly, for both processes, indicating an

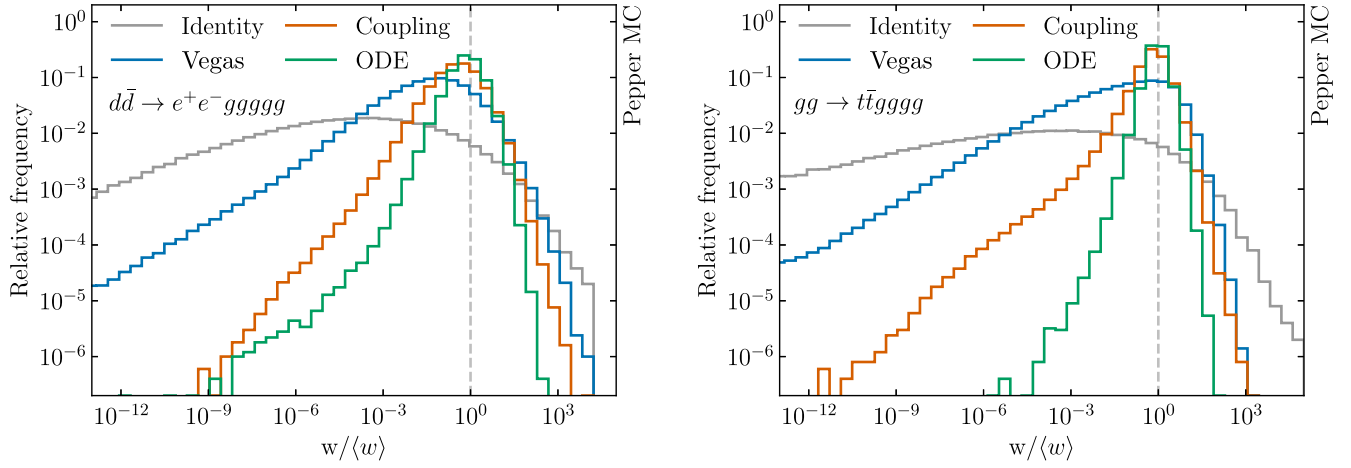


FIG. 2. Normalized weight distributions for 5×10^6 points for the $e^+e^- + 5$ gluons example (left) and the $t\bar{t} + 4$ gluons example (right). The plotted weights include matrix-element and phase-space weights, as well as PDF and, where applicable, the Jacobian of the remapping. Distributions are shown for the “Identity” mapping, and mappings optimized by VEGAS, Coupling Flows (“Coupling”), and ODE Flows (“ODE”).

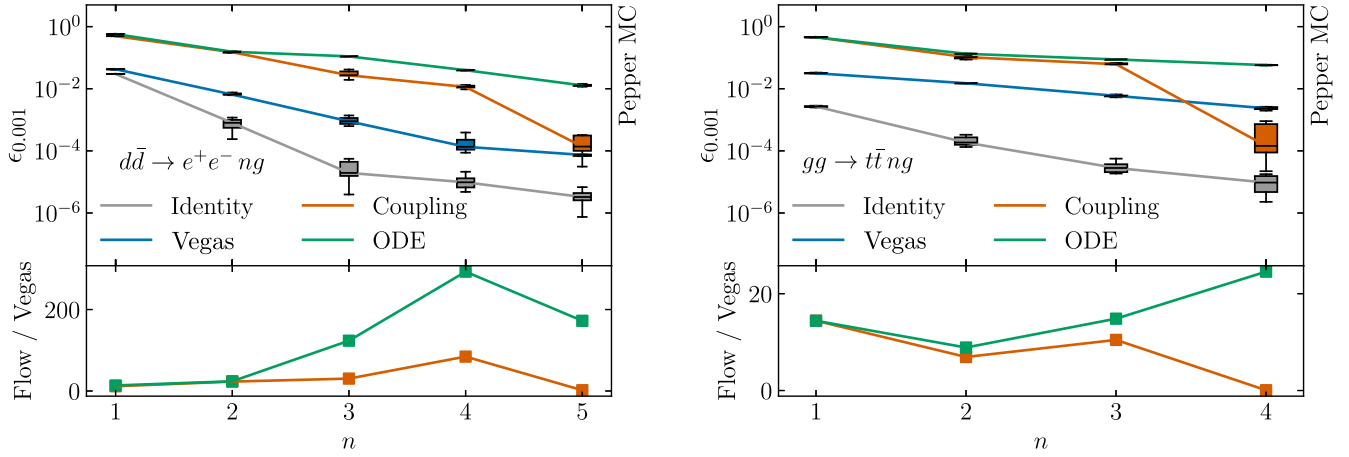


FIG. 3. Unweighting efficiencies $\epsilon_{0.001}$ for the $e^+e^- + n$ gluons example (left) and the $t\bar{t} + n$ gluons example (right), for various gluon multiplicities n . The efficiencies are shown for the “Identity” mapping, and mappings optimized by VEGAS, coupling flows (“Coupling”), and ODE flows (“ODE”). Each curve represents the mean over ten independent evaluations of the unweighting efficiency, the boxes indicate the quartiles of the distribution and the whiskers show $\frac{3}{2}$ of the interquartile range.

unfavorable scaling behavior towards large n . By evaluating the training results ten times with different random seeds, we can plot spreads for the data points. We find that they are smallest for the ODE flows, which means that this method generates the fewest very large event weight outliers.

Finally, we analyze the convergence behavior of $\epsilon_{0.001}$ as a function of training iterations in Fig. 4 for the highest multiplicity of each process. We find that all methods are reasonably converged at their last iteration. The spread of their results over ten independent samples is consistently the smallest for the ODE flows towards later iterations, with good stability from one optimization step to the next. We observe that the VEGAS results are less stable for $d\bar{d} \rightarrow e^+e^- + ng$ when compared to $gg \rightarrow t\bar{t} + ng$, consistent

with the lower overall performance we have observed earlier for the former process. Also the coupling flows give numerically less stable results when compared to the ODE flows, with a larger spread of results, indicating a larger fraction of outliers with large weights affecting the $w_{\text{max,eff}}$ determination. This might be an effect of the coupling flow’s MLE objective versus the regression-based flow matching objective.

The above observations support the conclusion that the new flow matching based method performs best, in particular, in the most relevant metrics for event generation, the unweighting efficiency. The advantage over the coupling flow based approach is most pronounced at the highest multiplicities studied, where the performance of the coupling flows drops significantly. This observation is consistent with

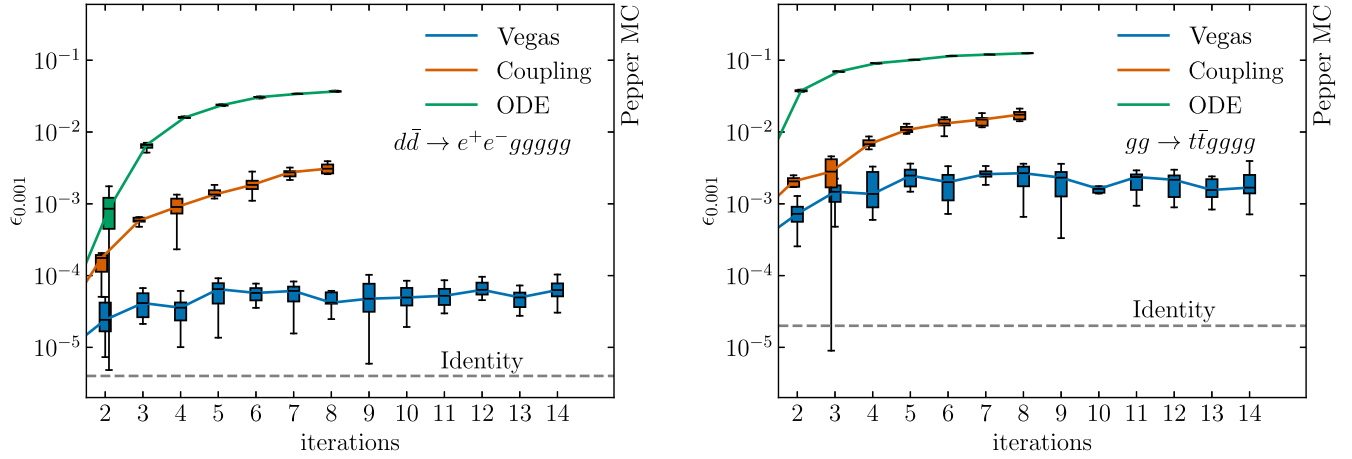


FIG. 4. Convergence of the effective unweighting efficiency $\epsilon_{0,001}$ with training iterations for the highest final-state gluon multiplicities studied: $d\bar{d} \rightarrow e^+e^- + 5g$ (left) and $gg \rightarrow t\bar{t} + 4g$ (right). Results are shown for the “Identity” mapping, VEGAS, coupling flow (“Coupling”), and ODE flow (“ODE”). Each curve represents the mean over ten independent evaluations of the unweighting efficiency, the boxes indicate the quartiles of the distribution and the whiskers show $\frac{3}{2}$ of the interquartile range.

others, which empirically show that time-discrete flows trained with MLE fall short in terms of expressivity and scalability [39]. Since the highest multiplicities drive the computational cost of event generation for current state-of-the-art production campaigns at the LHC, using a flow matching based approach would be highly beneficial.

B. Effective walltime improvements in the full parton-level event generation

In the previous section, it was demonstrated that ODE flow remapping achieves large gain factors for the unweighting efficiency. In practice, the gains do not translate directly to event generation speed-ups of the same factor: The inference time is too long due to the numerical solution of the ODE requiring many steps. We find that achieving a subpercent error on the Jacobian determinant (which is more sensitive than the phase space variables) requires very tight relative (absolute) tolerances of 10^{-7} (10^{-6}). Using a JIT-compiled code written in JAX [77], based on the DIFFRAX [78] and EQUINOX [79] libraries, generating a single point and its associated Jacobian determinant for the $e^+e^- + 5$ gluons process takes on average 4.2×10^{-4} s on an A100 GPU. For comparison, evaluating the corresponding differential cross section with PEPPER takes 9.5×10^{-6} s, roughly an order of magnitude less. Consequently, the computational overhead can reduce or even negate the potential improvements in unweighting efficiency.

We see two approaches toward improving the overall gain: Firstly, one can apply methods to straighten the flow trajectories, as discussed in Sec. III B. Viable strategies include rectified flows [29] and optimal transport conditional flow matching [49]. Straighter trajectories can be integrated with fewer steps and thus lead to faster inference times. In the optimal case, the ODE can be solved in a

single step. The improvement in inference time has to be balanced against additional training time and possibly a reduced unweighting efficiency. Secondly, the well-adapted but slow-to-evaluate ODE flow can be used as a teacher for a one-step generative model. An example of this is the REGFLOW approach [39]. In this case, some losses in performance metrics like the unweighting efficiency are unavoidable, but the gain in inference time can be by orders of magnitude. Below, we apply the REGFLOW approach.

The idea behind REGFLOW is to train a coupling flow with a regression objective similar to flow matching, instead of MLE. This is not possible if only (importance weighted) samples of the target distribution are available. However, if an already trained normalizing flow is at one’s disposal, one can simply use it to generate pairs (x_0, x_1) . These can be used to train a coupling flow by minimizing

$$\mathcal{L}_{\text{RegFlow}} = \mathbb{E}_{x_0, x_1} [\|\hat{x}_1 - x_1\|^2] + \lambda_r \mathcal{R}, \quad (39)$$

where $\hat{x}_1 = \psi_{\theta}^{\text{Coupling}}(x_0)$ denotes the image of x_0 under the coupling flow map $\psi_{\theta}^{\text{Coupling}}$. The regularization term consists of a regularization strategy $\mathcal{R}$ and a tunable coefficient $\lambda_r \in \mathbb{R}^+$. Following Ref. [39], we use the log determinant for regularization and set λ_r to 10^{-5} . Moreover, Gaussian noise with a standard deviation of 0.05 is added to the target points x_1 .

Reconsidering the highest multiplicity processes, $d\bar{d} \rightarrow e^+e^- + 5g$ and $gg \rightarrow t\bar{t} + 4g$, we train coupling flows with the same hyperparameters as before, but using the REGFLOW objective, Eq. (39). The training data are composed of pairs (x_0, x_1) generated by the trained ODE flows. To see the impact of the amount of training data, we consider two different sample sizes: 10 million and 100 million points.

TABLE III. Results for the walltime gains in full parton-level event generation for the two highest-multiplicity processes, $d\bar{d} \rightarrow e^+e^- + 5g$ and $gg \rightarrow t\bar{t} + 4g$. We compare continuous normalizing flows with flow matching (“ODE”), coupling-layer flows trained with the standard MLE objective, and coupling-layer flows trained on data generated by continuous normalizing flows using the REGFLOW method, with both 10M and 100M training points. For each case, we report the unweighting efficiency, the generation time per event t_{event} , the walltime gain relative to event generation using VEGAS, and the break-even point in event generation—i.e., the minimum number of events that must be produced with the flow-based methods to offset their increased training times relative to VEGAS (for which we conservatively assume negligible training time). Uncertainties in the gains and break-even points are propagated from the standard deviations of the unweighting efficiencies.

Process	Mapping	$\epsilon_{0.001}/\% \uparrow$	$t_{\text{event}}/\text{ms} \downarrow$	Gain over VEGAS $\uparrow$	Break-even/Mio. evts.
$d\bar{d} \rightarrow e^+e^- + 5g$	ODE	1.29(8)	0.42	3(1)	0.6(3)
	Coupling (MLE)	0.03(3)	0.003	3(3)	0.4(4)
	Coupling (REGFLOW 10M)	0.07(5)	0.003	7(5)	1.3(5)
	Coupling (REGFLOW 100M)	0.12(5)	0.003	12(6)	2.1(8)
$gg \rightarrow t\bar{t} + 4g$	ODE	5.76(9)	0.34	1.0(2)	...
	Coupling (MLE)	0.04(6)	0.002	0.2(3)	...
	Coupling (REGFLOW 10M)	1.3(2)	0.002	5.0(9)	26(4)
	Coupling (REGFLOW 100M)	2.2(2)	0.002	8(2)	42(5)

Table III compares the unweighting efficiency, the generation time per event, and the resulting overall gain for different mapping strategies across the two benchmark processes. As expected, the ODE-based mappings achieve the highest unweighting efficiencies, but are limited by comparatively long inference times. The coupling-layer-based models with MLE objective, in contrast, evaluate roughly two orders of magnitude faster, leading to substantial reductions in overall computing time, but also feature lower unweighting efficiencies. The REGFLOW variants recover a significant portion of the ODE performance while retaining the speed advantage. Their effective gains over VEGAS increase with training statistics, and the break-even points are reached at only a few million generated events for the Drell-Yan process, and several tens of millions for the top-antitop quark process.

VI. CONCLUSION

We have presented the first, to the best of our knowledge, application of the flow matching method to the problem of high-dimensional phase-space sampling in high-energy particle physics. The model is trained not only on the continuous kinematic phase-space dimensions, but also on the discrete variable used to select a helicity configuration. The training of the model and testing it in production is facilitated by a newly implemented simple file-based interface in the event generator PEPPER. The interface builds on the LHEH5 parton-level event database format, and is used to train the model and then test it in production. The interface is simple and can be used in the future to study more applications and also other novel sampling methods in a straightforward way.

In this contribution, we studied Drell-Yan and top-antitop pair production. Specifically, we evaluated various sampling efficiency metrics for the most challenging

partonic channels $d\bar{d} \rightarrow e^+e^- + ng$ with $1 \leq n \leq 5$ and $gg \rightarrow t\bar{t} + ng$ with $1 \leq n \leq 4$. The upper limits of these multiplicity ranges correspond to the highest number of jets currently simulated at the matrix-element level for experimental analyses. Their high complexity and low phase-space efficiency render them the main bottlenecks in current large-scale state-of-the-art event generation campaigns. The key performance metric for these productions is the unweighting efficiency $\epsilon_{0.001}$. Here, we find a gain factor of 184 (25) for $d\bar{d} \rightarrow e^+e^- + ng$ with $n = 5$ ($gg \rightarrow t\bar{t} + ng$ with $n = 4$). This result does not change qualitatively when considering $\epsilon_{0.01}$ instead, for which we find a gain factor of 150 (17). Studying the effect on the parton-level event generation throughput, it is found that the longer evaluation times for the ODE flow partially negate the gain factors. This is mitigated by using the REGFLOW method, combining the strengths of the coupling layers and flow matching. With that, we find effective event generation speed-up factors of 12 compared to VEGAS, and 4 compared to the standard coupling-layer based flow with MLE objective, for $d\bar{d} \rightarrow e^+e^- + ng$ with $n = 5$. For the other benchmark process, $gg \rightarrow t\bar{t} + ng$ with $n = 4$, the speed-up factors are 8 and 40, respectively. Overall, these results represent a significant advancement in the scalability and efficiency of event generation for a wide range of experimental physics analyses.

This research focused on the computationally most expensive partonic channel of each studied process. However, realistic simulations require the inclusion of all partonic channels during both training and generation. In a future work, we aim to develop a single conditional model that can simultaneously sample from all partonic channels. This combined approach enables the model to leverage correlations between channels, improving training effectiveness. We plan to make a sampling method with this

approach available as part of a future public PEPPER release.

Correlations also exist across different jet multiplicities. Consequently, a complementary strategy is to investigate the joint training of all multiplicities within a single model to further enhance training efficiency and production performance. Another improvement would be the use of rectification methods [29,49,50] that straighten the flow trajectories. These have the potential to significantly reduce the cost of the flow sampling.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the computing time granted by the Resource Allocation Board and provided on the supercomputer Emmy/Grete at NHR-Nord@Göttingen as part of the NHR infrastructure. The calculations for this research were conducted with computing resources under the project nhr_ni_starter_22045. The authors also acknowledge the use of computing resources made available by CERN to conduct some of the research reported in this work. This material is based upon work supported by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. E. B. and M. K. acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—No. 510810461. T. J. expresses his gratitude to CIDAS for supporting him with a fellowship and acknowledges financial support from the German Federal Ministry of Education and Research (BMBF; now Federal Ministry of Research, Technology and Space, BMFTR) in the ErUM-Data action plan through the KISS consortium (Verbundprojekt No. 05D2022). B. S. and F. S. were supported by the German Research Foundation (DFG) SFB 1456, Mathematics of Experiment—Project-ID No. 432680300. B. S. was supported by the Emmy Noether Program of the DFG—Project-ID No. 403056140.

DATA AVAILABILITY

The data that support the findings of this article are openly available [80].

APPENDIX A: EVENT GENERATION RUNCARDS

The PEPPER v1.3 event generator³ runcards that were used for this project are given in Tables IV and V. In addition to using the runcards, the code needs to be compiled after changing the line `#define HELICITY_BLOCK_SIZE 32` to `#define HELICITY_BLOCK_SIZE 1` in `src/event_data.h`,

³<https://gitlab.com/spice-mc/pepper/-/releases/1.3.0-kokkos>

TABLE IV. The PEPPER runcard used for generating $d\bar{d} \rightarrow e^+e^-ggggg$ events. For generating a different number of final-state gluons, only the `process` setting needs to be adjusted.

```
[main]
process = d db -> e+ e- g g g g g
mu2 = H_Tp^2/2
max_overweight_xs_contribution = 0.001
; helicity_integrator = false
[events]
random_numbers_output_enabled = true
unweighting_disabled = true
output_path = training_data_for_flow_model.hdf5
; input_path = random_numbers_from_flow_model.hdf5
[phase_space.vegas]
; disabled = true
alpha = 0.8
[phase_space.optimisation]
n_iter = 15
n_nonzero_min = 250000
n_nonzero_min_growth_factor = 1.00
```

TABLE V. The PEPPER runcard used for generating $gg \rightarrow t\bar{t}gggg$ events. For generating a different number of final-state gluons, only the `process` setting needs to be adjusted.

```
[main]
process = g g -> t tb g g g g
mu2 = H_T^2/2
max_overweight_xs_contribution = 0.001
; helicity_integrator = false
[events]
random_numbers_output_enabled = true
unweighting_disabled = true
output_path = training_data_for_flow_model.hdf5
; input_path = random_numbers_from_flow_model.hdf5
[phase_space.vegas]
; disabled = true
alpha = 0.8
[phase_space.optimisation]
n_iter = 15
n_nonzero_min = 80000
n_nonzero_min_growth_factor = 1.00
```

to avoid the generation of blocks of correlated events that share a given helicity configuration.

The runcards were used to generate the VEGAS and unoptimized “Identity” baseline results, to generate the training data for the flow models and then to measure the performance of the trained models. For the latter, the runcard line for the `input_path` setting should be used (by removing the semicolon to uncomment the line), to enable reading the random numbers generated by the model, instead of relying on PEPPER’s internal random-number generator.

APPENDIX B: ADDITIONAL FIGURES

In this appendix, we present figures analogous to Figs. 3 and 4, but for $\epsilon_{0.01}$ instead of $\epsilon_{0.001}$. Figure 5 displays the unweighting efficiency as a function of the number of additional final-state gluons, for both the $e^+e^- + n$ gluons and $t\bar{t} + n$ gluons processes, corresponding to Fig. 3. Similarly, Fig. 6 shows the convergence behavior of the unweighting efficiency for $\epsilon_{0.01}$, in correspondence with Fig. 4.

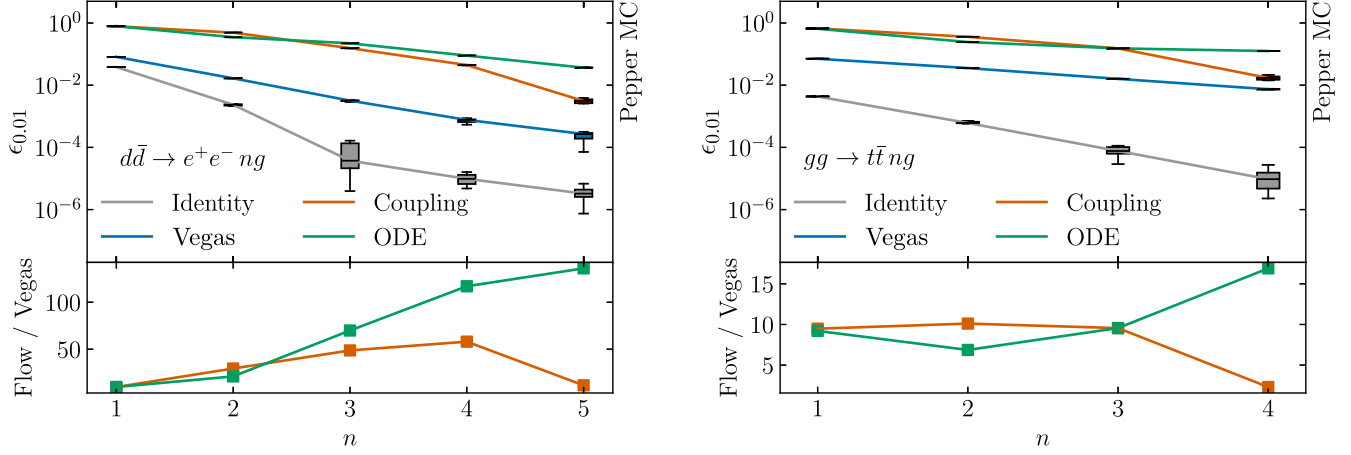


FIG. 5. Unweighting efficiencies, $\epsilon_{0.01}$ for the $e^+e^- + n$ gluons example (left) and the $t\bar{t} + n$ gluons example (right), for various gluon multiplicities n . The efficiencies are shown for the “Identity” mapping, and mappings optimized by VEGAS, Coupling Flows (“Coupling”), and ODE Flows (“ODE”).

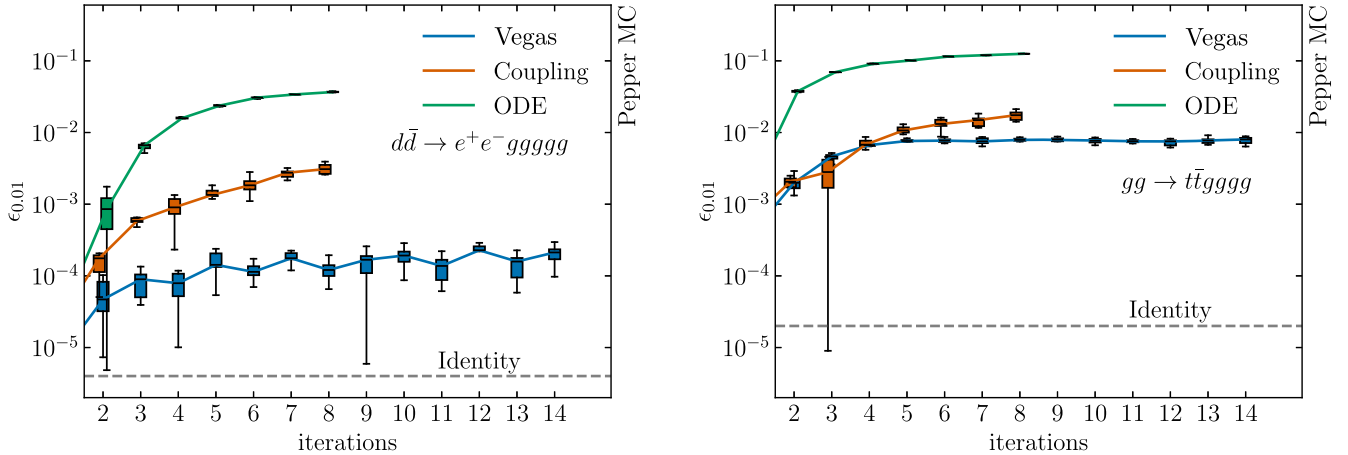


FIG. 6. Convergence of the effective unweighting efficiency $\epsilon_{0.01}$ with training iterations for the highest final-state gluon multiplicities studied: $d\bar{d} \rightarrow e^+e^- + 5g$ (left) and $gg \rightarrow t\bar{t} + 4g$ (right). Results are shown for the “Identity” mapping, VEGAS, Coupling Flow (“Coupling”), and ODE Flow (“ODE”). Each curve represents the mean over ten independent evaluations of the unweighting efficiency, the boxes indicate the quartiles of the distribution and the whiskers show $\frac{3}{2}$ of the interquartile range.

- [1] European Strategy Group, 2020 Update of the European Strategy for Particle Physics, CERN Council, Geneva (2020), [10.17181/ESU2020](https://cds.cern.ch/record/272841/files/17181-ESU2020).
- [2] M. Narain *et al.*, The future of US particle physics—The Snowmass 2021 energy frontier report, [arXiv:2211.11084](https://arxiv.org/abs/2211.11084).
- [3] HEP Software Foundation Collaboration, A roadmap for HEP software and computing R&D for the 2020s, *Comput. Software Big Sci.* **3**, 7 (2019).
- [4] HSF Physics Event Generator WG Collaboration, Challenges in Monte Carlo Event generator software for high-luminosity LHC, *Comput. Software Big Sci.* **5**, 12 (2021).
- [5] HSF Physics Event Generator WG Collaboration, HL-LHC computing review stage-2, Common software projects: Event Generators, [arXiv:2109.14938](https://arxiv.org/abs/2109.14938).
- [6] ATLAS Collaboration, Modelling and computational improvements to the simulation of single vector-boson plus jet processes for the ATLAS experiment, *J. High Energy Phys.* **08** (2022) 089.
- [7] National Energy Research Scientific Computing Center (NERSC), “Perlmutter architecture,” <https://docs.nersc.gov/systems/perlmutter/architecture> 2025.
- [8] E. Bothmann, A. Buckley, I. A. Christidi, C. Gütschow, S. Höche, M. Knobbe, T. Martin, and M. Schönherr, Accelerating LHC event generation with simplified pilot runs and fast PDFs, *Eur. Phys. J. C* **82**, 1128 (2022).
- [9] R. Kleiss and R. Pittau, Weight optimization in multichannel Monte Carlo, *Comput. Phys. Commun.* **83**, 141 (1994).
- [10] G. P. Lepage, A new algorithm for adaptive multidimensional integration, *J. Comput. Phys.* **27**, 192 (1978).
- [11] T. Ohl, VEGAS revisited: Adaptive Monte Carlo integration beyond factorization, *Comput. Phys. Commun.* **120**, 13 (1999).
- [12] G. P. Lepage, Adaptive multidimensional integration: VEGAS enhanced, *J. Comput. Phys.* **439**, 110386 (2021).
- [13] S. Jadach, Foam: A general purpose cellular Monte Carlo event generator, *Comput. Phys. Commun.* **152**, 55 (2003).
- [14] T. Hahn, CUBA: A Library for multidimensional numerical integration, *Comput. Phys. Commun.* **168**, 78 (2005).
- [15] A. van Hameren, PARNI for importance sampling and density estimation, *Acta Phys. Pol. B* **40**, 259 (2009).
- [16] S. Höche, S. Prestel, and H. Schulz, Simulation of vector boson plus many jet final states at the high luminosity LHC, *Phys. Rev. D* **100**, 014024 (2019).
- [17] M. D. Klimek and M. Perelstein, Neural network-based approach to phase space integration, *SciPost Phys.* **9**, 053 (2020).
- [18] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann, Exploring phase space with neural importance sampling, *SciPost Phys.* **8**, 069 (2020).
- [19] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, Event generation with normalizing flows, *Phys. Rev. D* **101**, 076002 (2020).
- [20] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer, and T. Plehn, MadNIS—Neural multi-channel importance sampling, *SciPost Phys.* **15**, 141 (2023).
- [21] R. Verheyen, Event generation and density estimation with surjective normalizing flows, *SciPost Phys.* **13**, 047 (2022).
- [22] T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn, and R. Winterhalder, The MadNIS reloaded, *SciPost Phys.* **17**, 023 (2024).
- [23] T. Heimel, O. Mattelaer, T. Plehn, and R. Winterhalder, Differentiable MadNIS-Lite, *SciPost Phys.* **18**, 017 (2025).
- [24] S. Badger *et al.*, Machine learning and LHC event generation, *SciPost Phys.* **14**, 079 (2023).
- [25] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, Neural ordinary differential equations, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY), pp. 6572–6583.
- [26] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, Flow matching for generative modeling, in *The Eleventh International Conference on Learning Representations* (2023), <https://openreview.net/forum?id=PqvMRDCJT9t>.
- [27] M. S. Albergio and E. Vanden-Eijnden, Building normalizing flows with stochastic interpolants, in *The Eleventh International Conference on Learning Representations* (2023), <https://openreview.net/forum?id=li7qeBbCR1t>.
- [28] M. S. Albergio, N. M. Boffi, and E. Vanden-Eijnden, Stochastic interpolants: A unifying framework for flows and diffusions, *J. Mach. Learn. Res.* **26**, 1 (2025).
- [29] X. Liu, C. Gong, and Q. Liu, Flow straight and fast: Learning to generate and transfer data with rectified flow, [arXiv:2209.03003](https://arxiv.org/abs/2209.03003).
- [30] V. Mikuni and B. Nachman, Score-based generative models for calorimeter shower simulation, *Phys. Rev. D* **106**, 092009 (2022).
- [31] E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh *et al.*, EPiC-ly fast particle cloud generation with flow-matching and diffusion, [arXiv:2310.00049](https://arxiv.org/abs/2310.00049).
- [32] A. Butter, N. Huetsch, S. Palacios Schweitzer, T. Plehn, P. Sorrenson, and J. Spinner, Jet diffusion versus JetGPT—modern networks for the LHC, *SciPost Phys. Core* **8**, 026 (2025).
- [33] L. Favaro, R. Kogler, A. Paasch, S. Palacios Schweitzer, T. Plehn, and D. Schwarz, How to unfold top decays, *SciPost Phys. Core* **8**, 053 (2025).
- [34] E. Bothmann, T. Childers, W. Giele, S. Höche, J. Isaacson, and M. Knobbe, A portable parton-level event generator for the high-luminosity LHC, *SciPost Phys.* **17**, 081 (2024).
- [35] E. Bothmann, T. Childers, W. Giele, F. Herren, S. Hoeche, J. Isaacson, M. Knobbe, and R. Wang, Efficient phase-space generation for hadron collider event simulation, *SciPost Phys.* **15**, 169 (2023).
- [36] L. Dinh, D. Krueger, and Y. Bengio, NICE: non-linear independent components estimation, in *3rd International Conference on Learning Representations, Workshop Track Proceedings* (2015), [arXiv:1410.8516](https://arxiv.org/abs/1410.8516).
- [37] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, Neural importance sampling, *ACM Trans. Graph.* **38**, 1 (2019).
- [38] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, Neural spline flows, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Vol. 32 (Curran Associates, Inc., 2019).
- [39] D. Rehman, O. Davis, J. Lu, J. Tang, M. Bronstein, Y. Bengio *et al.*, Efficient regression-based training of normalizing flows for Boltzmann generators, [arXiv:2506.01158](https://arxiv.org/abs/2506.01158).

- [40] E. Bothmann, T. Janßen, M. Knobbe, B. Schmitzer, and F. Sinz, companion Letter, Monte Carlo event generation with continuous normalizing flows, *Phys. Rev. Lett.* **137**, 091902 (2026).
- [41] J. von Neumann, *John von Neumann Collected Works, Vol. 5: Design of Computers, Theory of Automata and Numerical Analysis* (Pergamon Press, Oxford, England, 1961), pp. 768–770.
- [42] L. Kish, *Survey Sampling* (Wiley, New York, 1965).
- [43] E. Tabak and E. Vanden-Eijnden, Density estimation by dual ascent of the log-likelihood, *Commun. Math. Sci.* **8**, 217 (2010).
- [44] E. G. Tabak and C. V. Turner, A family of nonparametric density estimation algorithms, *Commun. Pure Appl. Math.* **66**, 145 (2013).
- [45] B. Stienen and R. Verheyen, Phase space sampling and inference from weighted events with autoregressive flows, *SciPost Phys.* **10**, 038 (2021).
- [46] N. Deutschmann and N. Götz, Accelerating HEP simulations with Neural Importance Sampling, *J. High Energy Phys.* **03** (2024) 083.
- [47] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman, How to train your neural ode: The world of jacobian and kinetic regularization, in *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, JMLR.org, (2020).
- [48] D. Onken, S. W. Fung, X. Li, and L. Ruthotto, OT-Flow: Fast and accurate continuous normalizing flows via optimal transport, in *AAAI Conference on Artificial Intelligence* (2021), Vol. 35, pp. 9223–9232, <https://ojs.aaai.org/index.php/AAAI/article/view/17113>.
- [49] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks *et al.*, Improving and generalizing flow-based generative models with minibatch optimal transport, [arXiv:2302.00482](https://arxiv.org/abs/2302.00482).
- [50] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Q. Chen, Multisample flow matching: Straightening flows with minibatch couplings, in *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, JMLR.org, (2023).
- [51] S. La Cagnina, C. Grunwald, T. Janßen, K. Kröninger, and S. Schumann, Phase space sampling with Markov Chain Monte Carlo methods, *EPJ Web Conf.* **337**, 01319 (2025).
- [52] T. Janßen, R. Poncelet, and S. Schumann, Sampling NNLO QCD phase space with normalizing flows, *J. High Energy Phys.* **09** (2025) 194.
- [53] E. Bothmann, W. Giele, S. Hoeche, J. Isaacson, and M. Knobbe, Many-gluon tree amplitudes on modern GPUs: A case study for novel event generators, *SciPost Phys. Codebases* **2022**, 3 (2022).
- [54] E. Bothmann, J. Isaacson, M. Knobbe, S. Höche, and W. Giele, QCD tree amplitudes on modern GPUs: A case study for novel event generators, *Proc. Sci. ICHEP2022* (2022) 222.
- [55] T. Gleisberg and S. Hoeche, COMIX, a new matrix element generator, *J. High Energy Phys.* **12** (2008) 039.
- [56] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, and J. Winter, Event generation with SHERPA 1.1, *J. High Energy Phys.* **02** (2009) 007.
- [57] SHERPA Collaboration, Event Generation with SHERPA 2.2, *SciPost Phys.* **7**, 034 (2019).
- [58] SHERPA Collaboration, Event generation with SHERPA 3, *J. High Energy Phys.* **12** (2024) 156.
- [59] T. Melia, Dyck words and multi-quark primitive amplitudes, *Phys. Rev. D* **88**, 014020 (2013).
- [60] T. Melia, Dyck words and multi-quark amplitudes, *Proc. Sci. RADCOR2013* (2013) 031.
- [61] H. Johansson and A. Ochirov, Color-kinematics duality for QCD amplitudes, *J. High Energy Phys.* **01** (2016) 170.
- [62] The HDF Group, Hierarchical data format, version 5, <https://www.hdfgroup.org/HDF5/>.
- [63] HIGHFIVE—HDF5 header-only C++ Library, <https://bluebrain.github.io/HighFive/>.
- [64] S. Höche, S. Prestel, and H. Schulz, Simulation of vector boson plus many jet final states at the High luminosity LHC, *Phys. Rev. D* **100**, 014024 (2019).
- [65] E. Bothmann, T. Childers, C. Gütschow, S. Höche, P. Hovland, J. Isaacson, M. Knobbe, and R. Latham, Efficient precision simulation of processes with many-jet final states at the LHC, *Phys. Rev. D* **109**, 014013 (2024).
- [66] NNPDF Collaboration, Parton distributions for the LHC Run II, *J. High Energy Phys.* **04** (2015) 040.
- [67] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, and G. Watt, LHAPDF6: Parton density access in the LHC precision era, *Eur. Phys. J. C* **75**, 132 (2015).
- [68] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, and K. J. Ozeren, Next-to-leading order $W + 5$ -jet production at the LHC, *Phys. Rev. D* **88**, 014025 (2013).
- [69] T. Akhound-Sadeh, J. Rector-Brooks, A. Joey Bose, S. Mittal, P. Lemos, C.-H. Liu *et al.*, Iterated denoising energy matching for sampling from Boltzmann densities, in *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, JMLR.org (2024).
- [70] C. Gao, J. Isaacson, and C. Krause, i-flow: High-dimensional integration and sampling with normalizing flows, *Mach. Learn. Sci. Tech.* **1**, 045023 (2020).
- [71] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, NFLOWS: Normalizing flows in PYTORCH, Zenodo (2020).
- [72] I. Loshchilov and F. Hutter, Decoupled weight decay regularization, in *7th International Conference on Learning Representations* (International Conference on Learning Representations (ICLR), New Orleans, Louisiana, USA, 2019).
- [73] J. Wildberger, M. Dax, S. Buchholz, S. R. Green, J. H. Macke, and B. Schölkopf, Flow matching for scalable simulation-based inference, in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, (Curran Associates Inc., Red Hook, NY, 2023).
- [74] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal *et al.*, Fourier features let networks learn high frequency functions in low dimensional domains, in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, (Curran Associates Inc., Red Hook, NY, 2020).
- [75] J. Dormand and P. Prince, A family of embedded runge-kutta formulae, *J. Comput. Appl. Math.* **6**, 19 (1980).

- [76] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, and S. Ermon, TORCHDYN: Implicit models and neural numerical methods in PYTORCH, <https://github.com/DiffEqML/torchdyn>.
- [77] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin *et al.*, JAX: Composable transformations of PYTHON+NUMPY programs, 2018.
- [78] P. Kidger, On neural differential equations, Ph.D. thesis, University of Oxford, 2021.
- [79] P. Kidger and C. Garcia, Equinox: Neural networks in JAX via callable PyTrees and filtered transformations, Differentiable Programming workshop at Neural Information Processing Systems 2021 (2021).
- [80] E. Bothmann, T. Janßen, M. Knobbe, B. Schmitzer, and F. Sinz, Efficient Many-Jet Event Generation with Flow Matching (Zenodo, 2026), [10.5281/zenodo.19949391](https://zenodo.org/record/19949391).