

Improving Monte Carlo simulations in high energy physics using machine learning techniques

Dissertation

for the award of the degree

‘Doctor rerum naturalium’ (Dr.rer.nat.)

at the **Georg-August-Universität Göttingen**

within the doctoral programme **Physics**

of the **Georg-August University School of Science (GAUSS)**

submitted by

Timo Janßen

from Westerstede

Göttingen, 2023

Thesis Committee

Prof. Dr. Steffen Schumann	Institut für Theoretische Physik, Georg-August-Universität Göttingen
Prof. Laura Covi, PhD	Institut für Theoretische Physik, Georg-August-Universität Göttingen
Prof. Dr. Marcus Müller	Institut für Theoretische Physik, Georg-August-Universität Göttingen

Members of the Examination Board

Reviewer:

Prof. Dr. Steffen Schumann	Institut für Theoretische Physik, Georg-August-Universität Göttingen
----------------------------	---

Second Reviewer:

Prof. Dr. Tilman Plehn	Institut für Theoretische Physik, Ruprecht-Karls-Universität Heidelberg
------------------------	--

Further members of the Examination Board

Prof. Dr. Alexander Ecker	Institut für Informatik, Georg-August-Universität Göttingen
Prof. Dr. Stan Lai	II. Physikalisches Institut, Georg-August-Universität Göttingen
Prof. Dr. Marcus Müller	Institut für Theoretische Physik, Georg-August-Universität Göttingen
Prof. Dr. Arnulf Quadt	II. Physikalisches Institut, Georg-August-Universität Göttingen

Date of the oral examination: 14 July 2023

Abstract

Monte Carlo event generators are nowadays indispensable tools for predictions based on first principles in high energy physics, and they represent one of the mainstays of particle physics research at the Large Hadron Collider. In the dawn of the high luminosity upgrade of the Large Hadron Collider, there is a push to more complex signatures and higher accuracy, rendering the generation of simulated events more expensive. At the same time, there are strict limitations on the computational budget. In this situation, the efficiency of event generators can be identified as a key issue. The recent rapid advancement of machine learning tools, first and foremost deep neural networks, and their successes in diverse applications make them a promising choice in addressing this challenge. In this thesis, I consider two central building blocks of event generation that represent bottlenecks in typical applications. The first is the sampling of phase space configurations such that their distribution closely approximates a given target. For this I present two new approaches, one based on normalizing flows and the other on nested sampling. The second is the unweighting of event samples, that is the generation of unit weight events that contribute equally to the total scattering cross-section. To accelerate the unweighting process, I present an unbiased unweighting method based on fast and accurate neural network surrogates for the event weights. Furthermore, I show how a surrogate optimized for the factorization properties of the corresponding matrix elements can significantly improve the performance for suitable processes. All methods are evaluated by means of examples, which are oriented towards realistic applications. It is also discussed how the different approaches could be combined and what opportunities there are for further developments.



Contents

1	Introduction	7
1.1	Motivation	8
1.2	Context and literature review	13
1.3	Outline of the thesis	16
2	Phase space sampling in high energy physics	17
2.1	Particle physics basics	17
2.1.1	The Standard Model	17
2.1.2	Scattering experiments	18
2.1.3	Perturbation theory and Feynman diagrams	21
2.1.4	Running coupling	21
2.1.5	Hadron collisions	22
2.1.6	Jets	23
2.2	The sampling problem in high energy physics	25
2.2.1	Sampling the differential cross-section	25
2.2.2	Monte Carlo event generators	26
2.3	Established methods and their shortcomings	28
2.3.1	Monte Carlo sampling	28
2.3.2	Adaptive importance sampling—the Vegas algorithm	30
2.3.3	Multichannel sampling	34
2.3.4	Non-uniform random variables	37
2.3.5	Rejection sampling for unweighted event generation	40
3	New sampling methods for efficiency improvements	41
3.1	Neural Importance Sampling	41
3.1.1	Artificial neural networks	41
3.1.2	Normalizing flows	43
3.1.3	<i>Publication: Exploring phase space with Neural Importance Sampling</i>	46
3.1.4	Impact	73
3.1.5	Normalization during training	73
3.1.6	Recent developments	78
3.2	Nested Sampling	83
3.2.1	Basic idea and specifics of the implementation	83
3.2.2	<i>Publication: Exploring phase space with Nested Sampling</i>	86
3.3	Comparison between the two approaches and opportunities for synthesis	108
4	Accelerating unweighted event generation	109
4.1	Partial unweighting	109
4.2	<i>Publication: Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates</i>	111
4.3	<i>To be published: Unweighted event generation for multi-jet production processes based on matrix element emulation</i>	145
4.4	Relationship to the other methods	176

5	Summary and conclusions	177
5.1	Main results	177
5.2	Prospects for the future	179

1 Introduction

Computer calculations are an essential and growing part of modern particle physics. The automated computation of event rates in high energy scattering experiments allows for complexities, in terms of accuracy and particle multiplicity, that would be impossible to do by hand. At the same time, modern collider experiments, like the Large Hadron Collider (LHC), produce enormous amounts of data. Computers are needed not only to be able to analyse the data itself, but also to produce quantitative theoretical predictions, based on first principles, that can keep up in terms of accuracy. Monte Carlo (MC) event generators, like HERWIG [1–3], PYTHIA [4, 5], and SHERPA [6, 7], are one of the main tools in this context. They represent the link between the experiment, the theory, and the computational tools, as illustrated in fig. 1.1. While MC event generators can be counted among the computational tools, they have a prominent position in that they embody our understanding of the fundamental laws of nature in a manner that allows predictions to be made in a direct way. Using an MC event generator, total scattering cross-sections, i.e. event production rates, and particle decay rates can be predicted. Another interesting, and experimentally highly useful, application is to predict differential cross-sections, i.e. production rates per phase space element, in a fully differential and exclusive way. This resembles the production of scattering events at collider experiments and allows for a detailed data analysis and comparisons between theoretical predictions and experimentally measured data.

In this thesis, the focus lies on a key element of MC event generation—the generation of the highest energy scale interaction as described by perturbative quantum field theory (QFT). This includes the efficient sampling of events with a distribution that closely approximates the predicted differential cross-section as well as producing unit-weight samples from these events. It is a challenging task that is nowadays approached in a highly automatized way, using adaptive algorithms with detailed physics knowledge built-in. Nonetheless, there is a strong interest in a further optimization of the current methods with the goal of achieving excellent efficiency in terms of computational resource demand. For that purpose, I consider the incorporation of modern machine learning (ML) techniques. These have proven to be efficient and versatile tools in a wide variety of domains. It is therefore obvious to investigate

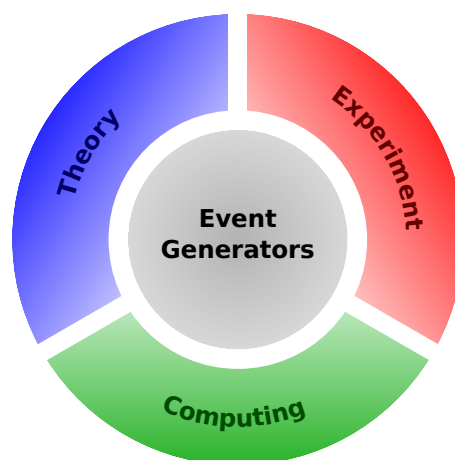



Figure 1.1: MC event generators are the link between theory, experiment, and computing. Figure taken from [8] .

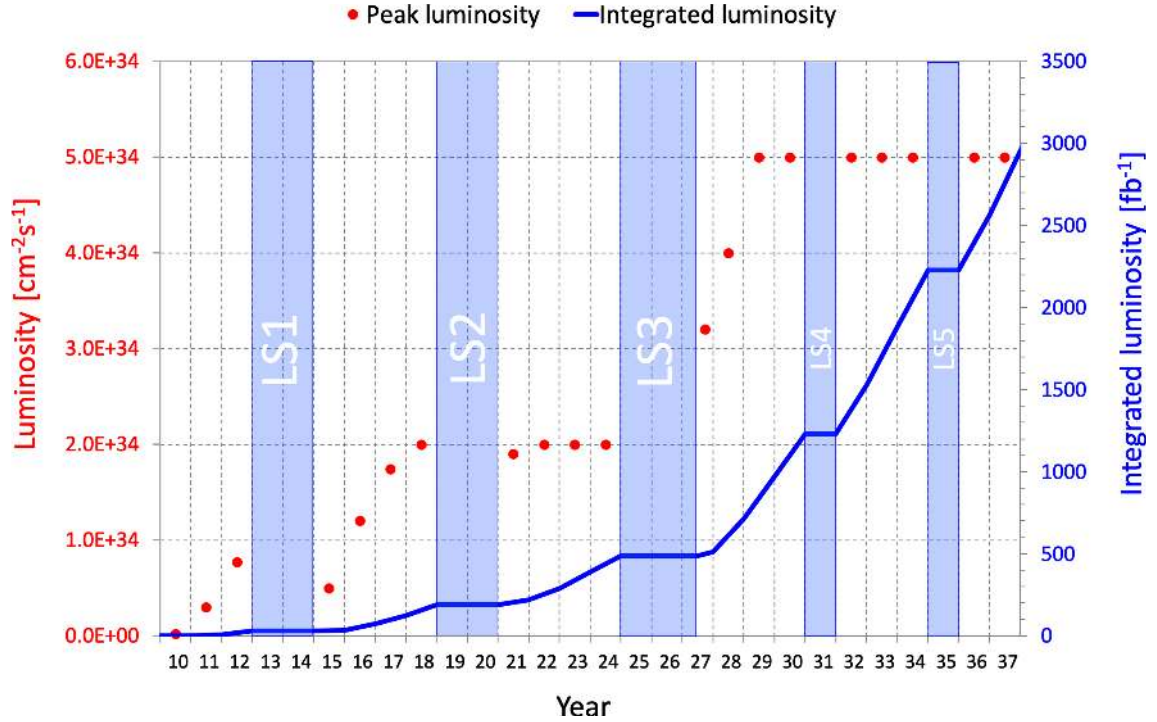


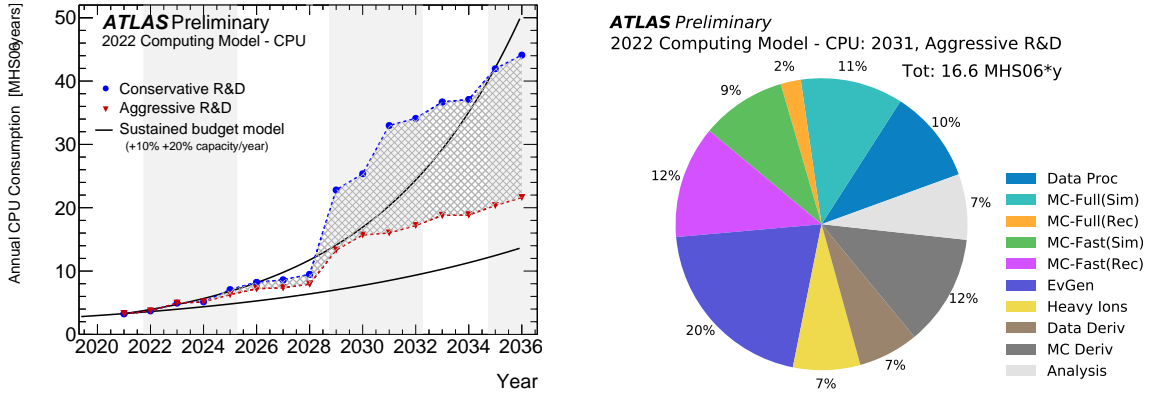
Figure 1.2: The LHC schedule and designed values for instantaneous and integrated luminosity. Figure taken from [16].

their suitability for increasing the efficiency of MC simulations for event generation. This thesis is based on four publications, refs. [9–12], which are reprinted, discussed, and put in relationship. In the following section, I comprehensively motivate the ideas that underpin this work, emphasizing the relevance of this work. Afterwards, in section 1.2, I provide a detailed review of literature in the context of this thesis. As the last section of the introduction, it is followed by an outline of the rest of this thesis.

1.1 Motivation

The high energy physics (HEP) community is facing a challenge which is due to the successes of the LHC and its experiments. Since the discovery of the Higgs boson [13, 14], the focus has shifted to precise Standard Model measurements and new discoveries. To boost the potential for these, an upgrade of the LHC is planned, the High Luminosity Large Hadron Collider (HL-LHC). It increases the luminosity of the machine, which corresponds to the number of events detected per unit of time. This will allow established analyses to improve their measurements with higher statistics and provide access to rare processes for which there is not enough data yet. In fig. 1.2, the luminosity of the LHC achieved over time and the projection for the future up to the year 2037 are shown. After the HL-LHC begins operation, planned for 2027 after long shutdown 3 (LS3), the instantaneous luminosity will reach values of $5 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$, two and a half times the value of the current run. Values as high as $7.5 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ could even be reached [15].

For the analysis of the measured data, large numbers of simulated events with small uncertainties will be needed. However, the computational budget of the detector experiments is limited and could fall short of the expected demand. Consider fig. 1.3a, which depicts the projected evolution of the annual central processing unit (CPU) consumption of the ATLAS experiment, assuming a conservative (blue) and an aggressive (red) research and development (R&D) scenario. The projection is compared to the expected budget (black), using two sustained



(a) Projected evolution of the annual CPU consumption, comparing sustained budget models (black) to conservative (blue) and aggressive (red) R&D scenarios.

(b) Projected CPU usage for LHC Run 4 under the aggressive R&D scenario, broken down to individual consumers.

Figure 1.3: Projections for ATLAS computing resources consumption. Figures taken from [17]



budget models with increases of 10 % and 20 % capacity per year. It can be seen that under the conservative scenario, the demand exceeds even the 20 % budget model for several years in succession. But also under the aggressive scenario, at least the 10 % budget model is not sufficient. One can conclude that any reduction in the need for computing time will lead to more physics analyses being possible and avoid competition for computing resources.

Traditionally, the largest fraction of the computational budget of the general-purpose detector experiments at the LHC went into the simulation of the detector response. With recent advances towards reducing the computational effort of detector simulations, however, this fraction is expected to decrease significantly. In fig. 1.3b a projection to the year 2031 of the CPU usage of different consumers for the ATLAS experiment is shown. The detector simulation (‘MC-Full(Sim)’ and ‘MC-Fast(Sim)’) has a share of 20 %, the same as the event generation (‘EvGen’). This means that the efficiency of event generation becomes more important than it has been until now. Developments towards faster event generators can lead to large savings in the overall resource need.

A multipurpose event generator has many different components, which do not contribute to the CPU consumption equally. If one wants to increase the efficiency of an event generator, it is important to know which parts of the simulation are the most demanding for the physics processes that are deemed relevant. However, one should note that these are not necessarily the ones that are easy to improve. An example for a breakdown into different components for the generation of event samples is shown in fig. 1.4. The corresponding study, ref. [18], presents computational strategies, complementary to the ones in this thesis, to reduce the computational footprint of particle-level event generation. Two processes, which were identified as standard candles for the LHC [19], are considered, namely $pp \rightarrow e^+e^- + 0, 1, 2j@NLO + 3, 4, 5j@LO$ and $pp \rightarrow t\bar{t} + 0, 1j@NLO + 2, 3, 4j@LO$ in a next-to-leading order (NLO) multijet merged calculation. The setup corresponds to a typical simulation for the ATLAS experiment using the SHERPA event generator. This includes, among others, parton distribution function (PDF) variations and electroweak corrections. In the figure it can be seen how the individual improvements cumulatively change the overall runtime and the shares of the different components from the baseline at the top to the final result at the bottom. With all the improvements applied, the two major remaining consumers are the evaluation of the tree-level matrix elements and the phase space generation. Accordingly, it is of utmost importance to work on optimizing the efficiency at these stages.

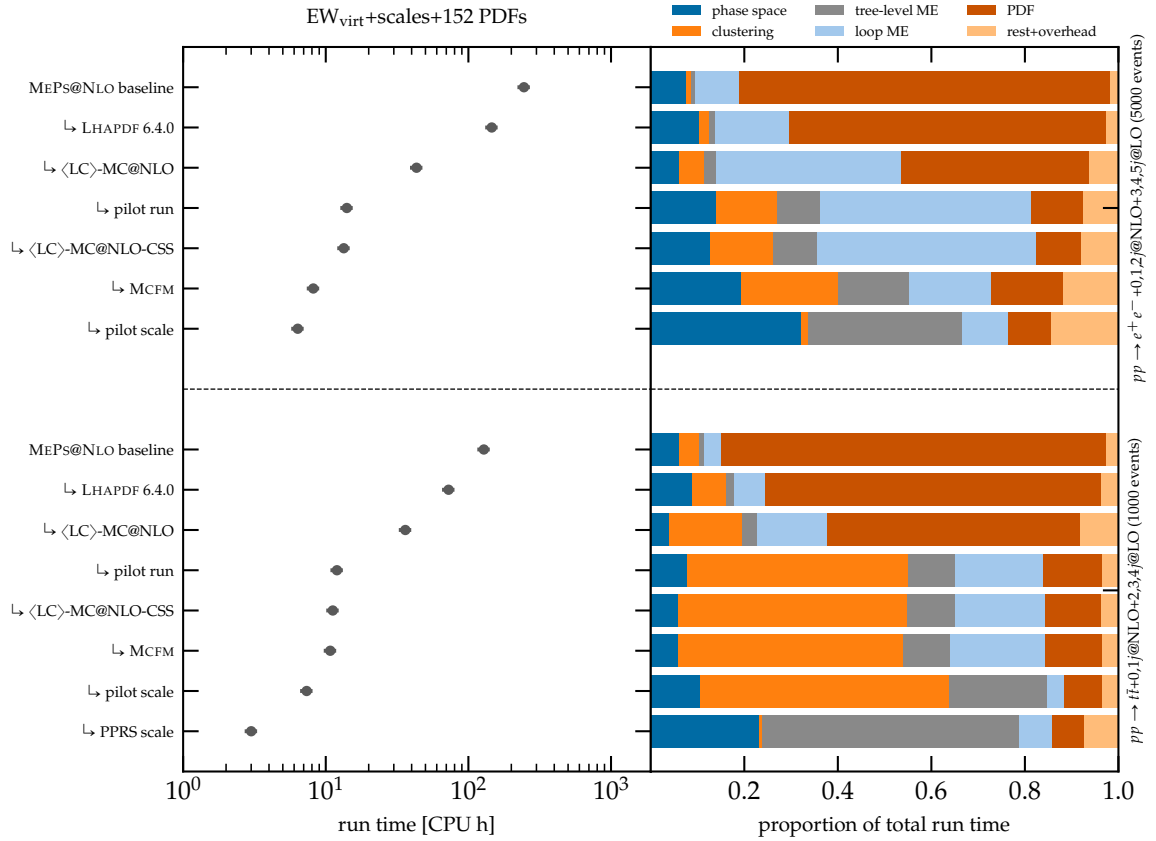


Figure 1.4: Results of different performance improvements in SHERPA as detailed in ref. [18] for the two processes $pp \rightarrow e^+e^- + 0, 1, 2j@NLO + 3, 4, 5j@LO$ and $pp \rightarrow t\bar{t} + 0, 1j@NLO + 2, 3, 4j@LO$, both using MEPS@NLO [20, 21]. Left: reduction in overall runtime. Right: breakdown into different components of event generation. Figure reproduced with friendly permission from Chris Gütschow.

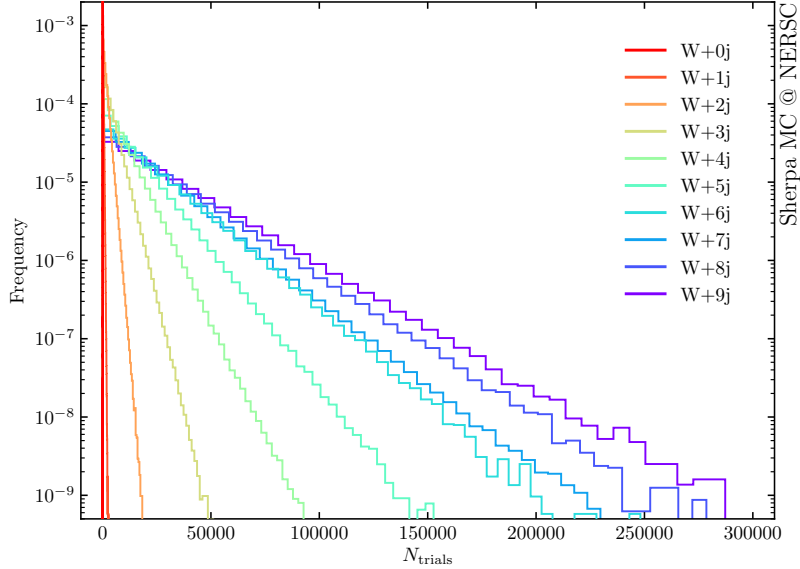



Figure 1.5: Number of trials for the unweighting of events of the process $pp \rightarrow W^+ + n$ jets at the LHC for $\sqrt{s} = 14$ TeV generated by SHERPA at leading-order (LO). Figure taken from [22] .

Both the matrix element (ME) and the phase space generation scale dramatically with the multiplicity, i.e. dimensionality, of the partonic hard interaction final state, as the number of contributing diagrams grows quickly. At the same time, there is a drop in unweighting efficiency, since due to the curse of dimensionality, supposedly small deviations in the phase space distributions lead to large fluctuations in the event weights. To demonstrate this, the distribution of the number of trials in unweighted event generation for the process $pp \rightarrow W^+ + n$ jets, with n up to nine, at the 14 TeV LHC using SHERPA is shown in fig. 1.5. It can be seen that the number of trials remains moderate for the low multiplicities, while the distributions stretch to ever larger numbers for the higher multiplicities. The average number of trials, related to the slope of the distribution, grows significantly. With the number of trials running into the hundreds of thousands and ME evaluation times reaching $\mathcal{O}(1$ s) [22], the resource demand is enormous.

A relevant question to ask is for which processes an efficiency increase is most worthwhile. The answer, of course, depends on who you ask. A pragmatic approach is to look at which processes are the most abundant at the LHC. Consider fig. 1.6, which shows a recent overview of cross-section measurements of Standard Model processes with the ATLAS detector. Among the processes with the highest cross-section, and therefore frequency, are the production of vector bosons (photons, γ , and W/Z bosons) in association with jets as well as top quark pairs ($t\bar{t}$) in association with jets. Consequently, these belong to the most important background processes for experimental analyses. As we have seen above, for these processes it is most promising to work on the high multiplicity tree-level contributions. However, in some analyses the situation can be quite different if, for example, they consider processes which are dominated by complicated higher-order contributions. Hence, it can be worthwhile to consider possible speed improvements for these as well. Overall, it would be welcome to achieve improvements that are as general as possible.

ML has been used routinely in HEP for many years, especially in experimental applications. However, in recent years the field of Deep Learning, where neural networks (NNs) with many layers are deployed, has triggered an impressive wave of developments. The size of models that can be trained successfully has grown significantly, with cutting-edge models having hundreds of billions of parameters [24]. This was also made possible by the availability of specialized hardware like graphics processing units (GPUs). Being universal function approx-

Standard Model Production Cross Section Measurements

Status: February 2022

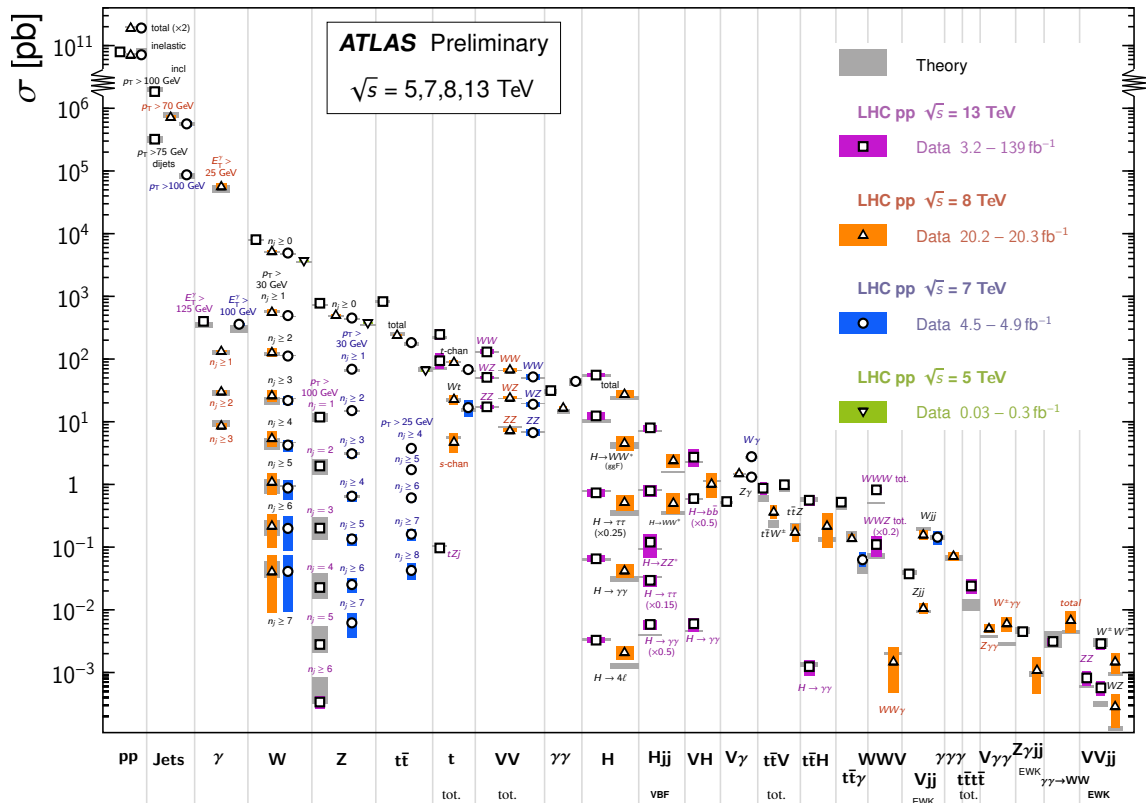


Figure 1.6: Summary of several Standard Model total and fiducial production cross-section measurements by ATLAS. Figure taken from [23] [CC BY](#).

imators [25–27], NNS can be applied to various scenarios when they are equipped with an efficient optimization strategy. Their properties make NNS a suitable and promising tool to try to achieve efficiency improvements in event generation.

There are a few obstacles that are exceptional in HEP. For example, highly accurate predictions are needed to compare with the measured data. This means that even small deviations, which would be completely negligible in image generation, for instance, must be avoided as far as possible. It is a great challenge to achieve this reliably with NNS. In addition, industrial applications of generative models often seek impressive results for individual items generated, e.g. an image based on a textual description. However, in HEP we look for an accurate description over the whole phase space, including the peaks as well as the tails of the distributions of many different observables. A significant mismatch in a small region of phase space can have adverse consequences if that region is part of an analysis. In addition, the generated data must be sensible with regard to the physics described, for example by reproducing basic laws such as Lorentz invariance and conservation of momentum. At the same time, we find ourselves in the fortunate and extraordinary position that, at least for the hard partonic interaction in focus here, the training data are exact. Our goal is to generate the same distributions as non-ML event generators but in a more efficient way. Furthermore, we can in principle generate as much training data as we want. However, their production is very costly, which means we should also look for efficient training strategies.

While NNS are certainly very flexible and easy to generalize, there are other ML tools in the sense that they automatically adapt to their inputs. In section 3.2, an approach based on nested sampling [28] for event generation is presented. Using a conceptually simple algorithm, it learns how to efficiently sample the scattering cross-section by monotonically going up along the contours. This is an unprecedented application for nested sampling. However, the algorithm is known to be very efficient for Bayesian inference in high dimensions and it is shown how this can be translated to MC event generation.

Now that the basic ideas of this thesis have been motivated, the following section aims to outline which previous work it could build on and which other approaches have already been taken. It is thereby addressed which preliminary work others have contributed. An insight into other approaches, both alternative and complementary, that have been explored is also provided. This includes other applications of ML in HEP and other methods for increasing the efficiency of event generation.

1.2 Context and literature review

The ideas in this thesis did not arise in a vacuum. Many other studies have dealt with the use of ML in HEP before and after the publication of the articles presented here. An excerpt of these is provided below in order to place this thesis in the appropriate context. Let us start with the sampling of the phase space for event generation. In a first study, boosted decision trees (BDTs) and NNS were applied to a toy example of a mixture of two Gaussians in d dimensions, with d up to nine [29]. The NN model was loosely based on the idea of generative adversarial networks (GANs) [30] in that the authors trained a generative network together with a regressive one in an iterative procedure. The regressor served as a cheap surrogate for the actual target to speed up the training. However, their model does not suffer from the typical stability problems of GANs. It was found that both BDTs and NNS had a tendency to overestimate in regions of low probability. Both models outperformed VEGAS [31, 32] and FOAM [33], with NNS being better in high dimensions. The activation functions were chosen to allow the generation of the full support of the target. In a later study, an NN was trained to generate the distributions of the two-dimensional three-body decay of a scalar, with and without intermediate resonances, as well as the leptonic scattering process $e^+e^- \rightarrow q\bar{q}g$, which has five phase-space dimensions and requires kinematic cuts for regularization [34]. Although their model guarantees to be neither

injective nor surjective, the authors argue that their training procedure favours approximately bijective maps given enough training data. They manually smoothed the kinematic cuts by a function and found good performance, with unweighting efficiencies between 30 % and 75 %. For this, they did not have to employ a multichannel sampler. With some extensions, the same approach was applied to the Higgs boson decay process $H \rightarrow \mu^+ \mu^- e^+ e^-$ in a follow-up study [35]. The process features two intermediate Z bosons and has five dimensions. An unweighting efficiency of 26 % was achieved.

Other studies based their models on normalizing flows (NFs) to design bijective maps and in turn guarantee full PS coverage. Normalizing flows were first suggested in refs. [36, 37] and were finally popularized for density estimation in ref. [38]. Besides HEP, they were also introduced to other fields, such as lattice field theory [39–43] and cosmology [44–46]. The first applications of NFs to HEP PS sampling were a study by my coauthors and me [9], as well as a similar study that was undertaken independently in parallel [47]. These approaches are introduced and discussed in section 3.1. Other authors employed autoregressive flows [48, 49] and trained their model directly on weighted event samples instead of samples from the NF, which allowed for an efficient training on a GPU [50]. They tested their model on a leading-order (LO) leptonic top quark pair production process with full decays, giving rise to a 14-dimensional phase space. Furthermore, they considered top quark pair production in proton collisions at NLO, which includes negative weights. In both cases, their unweighting efficiencies surpassed those of VEGAS. Another example of autoregressive flows, based on rational-quadratic splines [51], was applied to neutrino-nucleus cross-sections [52]. A high unweighting efficiency was obtained for a four-dimensional example. Similar to refs. [9, 47], NFs were used to remap the channels of a multichannel distribution in ref. [53]. In addition, a local version of the channel weights was introduced, and an NN was used to adapt them. To increase the training efficiency, the NFs were trained online on samples from the model and on previously generated events in parallel. As a physical example, a Drell-Yan process with an additional Z' resonance, $pp \rightarrow \gamma, Z^*, Z'^* \rightarrow e^+ e^-$, was used.

An alternative approach is to train a generative model, e.g. a GAN, on simulated data, and to use the trained model in place of the full event generator, possibly including the detector simulation as well [54–61]. This needs a sufficient amount of training data in order to reproduce the actual event distribution well. Such a method becomes interesting when it can be used to produce additional data beyond the training statistics. At a certain point, the training has paid off and one benefits from the reduced computational cost of the generative model compared to the full simulation. However, it is an ongoing discussion to what extent this approach is useful in practice, and whether a GAN can generate additional statistics beyond the training data through its interpolation and extrapolation capabilities [62, 63]. Nevertheless, useful applications are possible, such as an efficient fast simulation tool for detector simulations.

A key ingredient for the surrogate unweighting method presented in chapter 4 is a fast and accurate surrogate model that emulates the ME or the differential cross-section. In the original publication of the method, ref. [11], we trained a simple NN model to approximate the event weights. The gains could be significantly increased [12] by using the factorization-aware ME emulator of ref. [64]. It takes advantage of the factorization properties of MEs in the form of dipole formulae [65]. The dipoles capture the singular structure in the soft and collinear limits, such that the model does not need to learn it from scratch. An NN only learns coefficients that get multiplied by the dipole terms in a linear combination. The model achieves high accuracy and is able to extrapolate into regions that are more singular than the training data.

There have been several other approaches to emulating MEs with ML techniques. One of the first approaches was the application of the gradient boosting machine [66–69] XGBOOST [70] to the loop-induced process $gg \rightarrow ZZ$ [71]. The two-dimensional PS was divided into ten regions, and one regressor was trained on each of them. Errors below 0.1 % were achieved. Another study looked at the example of $e^+ e^- \rightarrow \text{jets}$ at leading and next-to-leading order [72].

The higher dimensionality makes NNs an eligible choice for the regression. However, the authors found that a single NN has difficulties finding a good fit across the whole PS due to the infrared singularities at the edges. Hence, they separated the divergent from the non-divergent part and further decomposed the latter into more sectors according to the FKS subtraction method [73, 74], thereby integrating physics knowledge into their design. They trained one NN per sector and found percent-level accuracy even for the more difficult distributions. In a comparison, the factorization-aware model mentioned above achieved significantly better fits by building knowledge about the singularities into the model [64]. In a follow-up study [75], the approach of ref. [72] was applied to the loop-induced processes $gg \rightarrow \gamma\gamma g$ and $gg \rightarrow \gamma\gamma gg$ contributing to diphoton production at hadron colliders. Although a good approximation was achieved overall, some PS regions remained challenging. Bayesian NNs were used as an alternative model in another follow-up [76]. These provide an estimate of the training-related uncertainties and can also be used to optimize the network training. Boosted training was used to improve the precision and the uncertainty estimate. It was shown that through this procedure a relatively simple and small network was able to match the performance of ref. [64] without a physics-inspired architecture. In ref. [77], the factorization-aware model was extended to the one-loop level by using it for the prediction of NLO k -factors. The authors based their factorization on antenna functions [78] and considered the process $e^+e^- \rightarrow q\bar{q} + ng$, with n up to and including 5. They obtained percent-level accuracy for the most demanding process. Finally, regression has even been attempted for two-loop helicity amplitudes [79]. An NN with skip connections [80] was used and physics domain knowledge was exploited by normalizing the amplitudes by the comparably cheap first-order terms. Furthermore, symmetry arguments were used to reduce the number of terms.

An argument in favour of ME emulators is that true MEs of high-multiplicity processes are expensive, and that a lot of computational time can be saved by relying on an emulator instead. This is certainly tempting, but makes high demands on the quality of the approximation. High accuracy, at the percent level or below, is needed over the whole fiducial phase space. Furthermore, it would be welcome to have a reliable uncertainty estimate of the approximation, which is difficult to get with NNs. This is an active area of research that has also gained attention in the HEP community [81–84]. Proposed methods for uncertainty estimation include NN ensembles [64, 85, 86] and Bayesian networks [60, 61, 76, 87, 88]. While these show that NN uncertainties can be modelled over a wide range of phase space regions, they cannot prevent overconfidence in some regions. This can lead to problems in applications if the uncertainty estimates are used blindly without knowing where they are not reliable. The surrogate unweighting method of chapter 4 offers an alternative application of ME emulators that does not rely on uncertainty estimates, since the method is unbiased by design. Nevertheless, it requires accurate predictions from the model to be efficient.

Another interesting application of ML in HEP is reweighting, which has found a multitude of possible uses. One of them is unfolding, that is the correction of measurements for detector effects to enable comparisons with theoretical predictions or measurements from other experiments, thereby going from detector level to particle level. Traditionally, this is done for individual observables based on histograms. The OMNIFOLD method [89] uses an NN classifier to iteratively reweight a sample of simulated events. This way, the full phase space information is available. The method is unbinned and not specific to any observable. Similarly, DCTR [90] can be used to reweight one particle level simulation into another. This is useful to generate new detector level samples from different particle level simulations given that one fully simulated sample (at detector level) is available. Furthermore, it can be used for determining systematic uncertainties with continuous parameter variations and for the parameter tuning of event generators. The DCTR approach of training a classifier NN is also used in the post-processing method DCTRGAN [91]. It reweights samples from a generative model, e.g. a GAN, to improve their accuracy. As an extension of this, LASER [92] pulls back

the classifier weights to the latent space, which can be done directly for invertible models like NFs or via OMNIFOLD for other models like GANs. Afterwards, another generative model is trained to generate the refined latent space such that unweighted samples can be produced. The authors propose to use a GAN to refine an NF in order to overcome its topological obstructions. However, they note that rejection sampling or Markov Chain Monte Carlo (MCMC) could also be used to sample the weighted latent space. Finally, this approach is extended by ELSA [93], which is a combination of LASER and augmented NFs [94–97]. It can be used for approximate event generation and the authors demonstrate this for $W^+ + \{2, 3\}$ jets production at the LHC.

In addition to the examples listed above, many other ML applications in HEP have been suggested. One of the most active fields, due to its large impact on computational efficiency, is detector simulation [98–116]. Among the other applications are event subtraction [117], event unweighting [118], jet super resolution [119, 120], the matrix element method [121], parton showers [122–128], simulation-based inference [129–132], and solving parametric integrals [133]. For reviews of the application of modern ML in HEP, see refs. [134–138].

A conceptually different approach is our use of nested sampling for event generation, which is presented in section 3.2. It uses a different sampling technique, not based on importance sampling. Although changes in established workflows would be necessary for its introduction, it also offers advantages. For example, nested sampling achieves remarkably good efficiency, especially in high dimensions. And while nested sampling itself is not based on modern ML techniques such as NNS, it opens up many possibilities to combine the advantages and further increase the performance of nested sampling through NNS. Originally conceived as a method to determine the evidence in Bayesian inference, nested sampling has found widespread adoption in cosmology [139–144]. However, it has also proven to be a versatile tool beyond that. Some of the main other fields of application are gravitational wave astronomy [145–147], particle physics [148–152], and materials science [153–158]. It would not be surprising if nested sampling also finds its place in HEP. This thesis describes one of the first approaches in this direction.

1.3 Outline of the thesis

In chapter 2, an overview of the theoretical foundations of HEP as well as the established approaches to the sampling of scattering cross-sections is given. Two new methods are introduced in chapter 3, the usage of NFs and nested sampling for PS sampling. In chapter 4 it is shown how NN surrogates can be used to accelerate the production of unweighted event samples. Finally, the insights are summarized in chapter 5 and an outlook on future developments is given.

2 Phase space sampling in high energy physics

To establish the necessary concepts and terminology that underlie the ideas in this thesis, this chapter is concerned with the basics of particle physics and standard sampling techniques. In section 2.1, an overview is given on how to determine scattering cross-sections in the Standard Model. The main target of this thesis, the fully differential hadronic cross-section, is defined based on the factorization theorem of quantum chromodynamics (QCD). An introduction to the problem of sampling the differential cross-section is given in section 2.2. It is followed by a brief description of MC event generators, which provide a comprehensive solution to this problem. This chapter ends after section 2.3, where the most important sampling techniques are presented. These comprise the crude MC technique, importance sampling, multichannel sampling, and methods to generate non-uniform random numbers.

2.1 Particle physics basics

The Standard Model is the accepted theory in particle physics, which is introduced in section 2.1.1. We then consider ourselves with scattering experiments at particle colliders in section 2.1.2, with the goal of defining scattering cross-sections. In section 2.1.3, it is then shown how to use perturbation theory to make quantitative predictions in HEP. The topic of running couplings due to renormalization is touched upon in section 2.1.4. We refine the definition of cross-sections for collisions of hadrons in section 2.1.5, and finish with a discussion of jets in section 2.1.6.

2.1.1 The Standard Model

The theory describing our understanding about the fundamental interactions of elementary particles is the Standard Model. It contains 17 main types of elementary particles, which are listed with some of their properties in fig. 2.1. We can identify two kinds of fermions, leptons and quarks, which come in three generations. The leptons are the electron, e , and the electron neutrino, ν_e , as well as the muon, μ , and the tau, τ , with their respective neutrinos, ν_μ and ν_τ . For the quarks, the three generations comprise the up and down quarks, u and d , the charm and strange quarks, c and s , and the top and bottom quarks, t and b . Their interactions are mediated by four vector bosons, the gauge bosons, which act as force carriers. There are three types of gauge bosons for the electroweak interaction, the massive Z and W bosons and the massless photon, γ . The W boson comes in two variations, W^+ and W^- , with electric charges $+1$ and -1 . The gauge boson of the strong interaction is the massless gluon, g . Finally, there is a scalar Higgs boson, H .

The Standard Model is a QFT collectively describing the electromagnetic, weak, and strong interactions. The electromagnetic and weak interactions can be unified in an electroweak theory in the form of a gauge theory with gauge group

$$\text{SU}(2) \times \text{U}(1), \tag{2.1}$$

where the $\text{SU}(2)$ part is related to weak isospin, and the $\text{U}(1)$ part is related to weak hypercharge. This gauge group is spontaneously broken to the electromagnetic $\text{U}(1)_{\text{EM}}$ gauge group via the Higgs mechanism. This gives rise to the masses of the three massive gauge bosons, $\{W^\pm, Z^0\}$, and a massive scalar particle, the Higgs boson, H . Yukawa coupling to the Higgs field generates

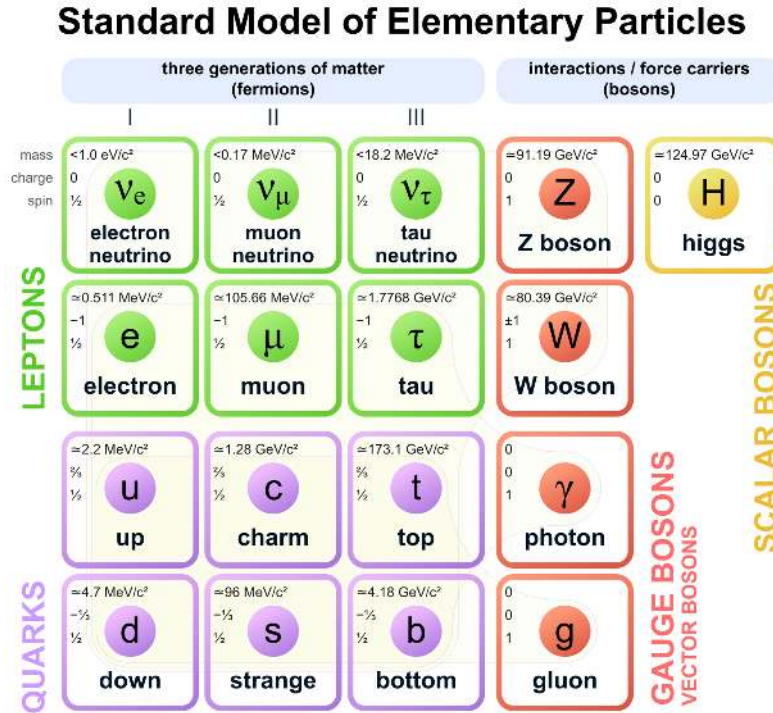


Figure 2.1: Overview of the elementary particles of the Standard Model with their masses, electric charges and spins. The masses correspond to the status of 2019, as published by the Particle Data Group [159]. Figure taken from [160] [CC](#) [iF](#).

the fermion masses. With regard to their SU(2) charges, the fermions can be grouped into left-handed doublets and right-handed singlets.

The strong sector is described by QCD, a non-abelian gauge theory with gauge group

$$\text{SU}(3). \quad (2.2)$$

QCD is a complicated theory with properties that make quantitative predictions a challenging task. The quarks, which come in colour triplets, can change their colour by emitting gluons. Furthermore, the gluons carry colour charge and can thus interact with themselves. As is explained in section 2.1.4, perturbative calculations are only possible at high energy scales, such that effects at lower scales are much more difficult to study. However, QCD is the most relevant theory for this thesis. This is because we want to make predictions mainly for the LHC. The LHC collides protons, since they can reach much higher energies than e.g. electrons in a circular collider, due to less synchrotron radiation. Protons are made of quarks and gluons and so they interact mainly via the strong interaction, as described by QCD. Furthermore, the strong interaction dominates the radiation patterns at high-energy colliders, because of its strong coupling strength.

2.1.2 Scattering experiments

Consider the case of two colliding beams made up of quantum-mechanical particles, one of type 1 and the other of type 2, as sketched in fig. 2.2. The particles themselves do not have a sharply defined diameter but if they come close there is some probability of interaction. In this sense, a particle within the beam has an effective cross-sectional area, even if we may not be able to find a physical equivalent of this area. This cross-section, σ , is proportional to the event

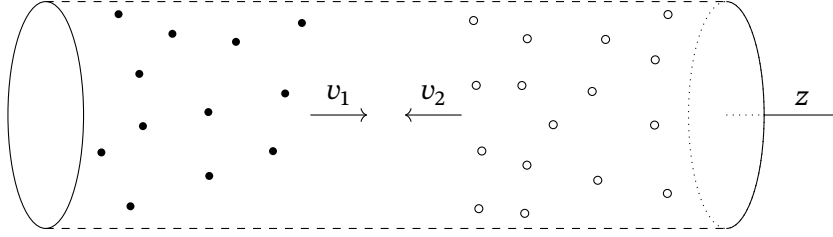


Figure 2.2: Two colliding beams.

rate

$$\frac{dN}{dt} \propto \sigma, \quad (2.3)$$

where N is the total number of scattering events. The event rate is directly related to the interaction probability. We define the proportionality constant belonging to eq. (2.3) as the instantaneous luminosity $\mathcal{L}(t)$. By integration, we get the integrated luminosity,

$$L = \int_0^T \mathcal{L} dt, \quad (2.4)$$

which is a measure of the amount of data produced by a collider. The luminosity is defined by the incoming beams, e.g. the density of particles. In contrast, the cross-section depends exclusively on the properties of the incoming particles. While not necessarily easy to do in either case, it is possible to experimentally measure as well as theoretically calculate scattering cross-sections. The conventional unit for cross-sections is the barn, defined as

$$1 \text{ b} \equiv 10^{-28} \text{ m}^2. \quad (2.5)$$

It is common to encounter fractions of that, like femtobarn, $1 \text{ fb} = 10^{-15} \text{ b}$, or picobarn, $1 \text{ pb} = 10^{-12} \text{ b}$. If there are several possible types of events, e.g. the production of different final state particles, the total cross-section is the sum over all types,

$$\sigma = \sum_i \sigma_i. \quad (2.6)$$

The cross-section provides us with a single number that contains information about the number of events of a certain type occurring in a unit of time. More inclusive information can be gained from the differential cross-section for some observable \mathcal{O} :

$$\frac{d\sigma}{d\mathcal{O}}. \quad (2.7)$$

This gives us a distribution of events, which we can visualize and compare, for example with the use of histograms. The most versatile option is to consider the cross-section differential in all phase space variables. This object, the fully differential cross-section, can be boiled down to a certain observable by integrating over the other variables.

Let us now consider how to calculate cross-sections in QFT. Skipping over a lot of details, the basic approach is to construct wavepackets for the initial-state particles in the early past and evolve them with the time evolution operator $e^{-i\mathbf{H}t}$, where \mathbf{H} is the Hamiltonian. The evolved state is then overlapped with wavepackets for the final-state particles in the distant future, which we are interested in. This provides us with the probability amplitude for the type of event defined by the desired particles. In a simplified notation, the overlap between the

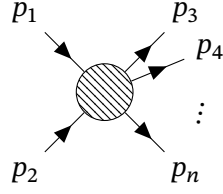


Figure 2.3: Two particles scattering and producing $n - 2$ particles.

initial and final states is described by the so-called **S**-matrix:

$$S_{fi} = \langle f|i \rangle. \quad (2.8)$$

We now confine ourselves to the case of two colliding particles, with four-momenta p_1 and p_2 , and n final-state particles, with momenta p_3, p_4, \dots, p_n . This case is illustrated in fig. 2.3. The **S**-matrix for it can be written as

$$S_{fi} = \delta_{if} + i(2\pi)^4 \delta^{(4)}\left(p_1 + p_2 - \sum_{k=3}^n p_k\right) \mathcal{M}, \quad (2.9)$$

where \mathcal{M} is called the invariant matrix element, or scattering amplitude. The δ_{if} corresponds to the case of no interaction, which is of no interest here. Four-momentum conservation is ensured by the Dirac delta distribution, and it is implied that the external particles are on their mass shell:

$$p_{1/2}^2 = m_{1/2}^2 \quad \text{and} \quad (2.10)$$

$$p_f^2 = m_f^2. \quad (2.11)$$

Let us consider the actually interesting part, the matrix element \mathcal{M} . It can, with quite general arguments, be related to the cross-section. Skipping over the calculation, the result is

$$d\sigma = \frac{1}{2E_1 E_2 |\mathbf{v}_1 - \mathbf{v}_2|} \left(\prod_f \frac{d^3 \mathbf{p}_f}{(2\pi)^3} \frac{1}{2E_f} \right) \times |\mathcal{M}(p_1, p_2 \rightarrow \{p_f\})|^2 (2\pi)^4 \delta^{(4)}\left(p_1 + p_2 - \sum_{k=3}^n p_k\right), \quad (2.12)$$

where the first factor on the right-hand side is the flux factor, with $|\mathbf{v}_1 - \mathbf{v}_2|$ being the difference of the beam velocities in the laboratory frame. To go from a probability amplitude to an observable, the matrix element is squared. Finally, there is an integral over all final-state momenta $\{p_f\}$, which is of the form

$$\int d\Phi = (2\pi)^4 \left(\prod_f \int \frac{d^3 \mathbf{p}_f}{(2\pi)^3} \frac{1}{2E_f} \right) \delta^{(4)}\left(p_1 + p_2 - \sum_{k=3}^n p_k\right). \quad (2.13)$$

This is called the Lorentz-invariant phase space and is needed to integrate the differential cross-section to a total cross-section.

With eq. (2.12) at our hands, the remaining difficulty is to find a recipe for calculating $|\mathcal{M}|^2$. It turns out that for the Standard Model this is possible using perturbation theory. A very helpful tool for the calculation are Feynman diagrams. In section 2.1.3, an overview of the approach is given.

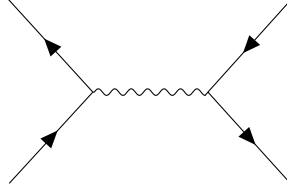


Figure 2.4: An example of a Feynman diagram: Two fermions scatter, producing an intermediate boson that decays to two fermions again.

2.1.3 Perturbation theory and Feynman diagrams

The idea of using perturbation theory for QCD can be broken down to expanding an observable f in orders of the strong coupling parameter α_s , which is defined as

$$\alpha_s = \frac{g_s^2}{4\pi}, \quad (2.14)$$

where g_s is the coupling appearing in the QCD Lagrangian. In schematic form, this expansion looks like

$$f = f_1\alpha_s + f_2\alpha_s^2 + f_3\alpha_s^3 + \dots \quad (2.15)$$

Assuming a small coupling, $\alpha_s \ll 1$, the series converges and the first few terms can deliver an acceptable approximation. However, even a divergent series can provide a reasonable approximation with appropriate truncation. We call the lowest order contributing to a given observable the leading-order (LO), the next higher one next-to-leading order (NLO) and so on. NLO calculations can be regarded the current standard as they are available largely automatized for almost arbitrary processes. Many processes are also available at next-to-next-to-leading order (NNLO) by now. Since the electromagnetic coupling is so small, the leading-order in quantum electrodynamics (QED) is often sufficient.

A widely used tool for perturbative calculations of scattering amplitudes are Feynman diagrams. They are visual representations of scattering processes as spacetime diagrams, whereby many different diagrams can contribute to a single amplitude. Feynman diagrams simplify the calculation by reducing it to a set of rules. The building blocks of a diagram are different types of external lines corresponding to real (on-shell) particles, internal lines corresponding to virtual (off-shell) particles, and vertices, where these lines connect. An example of a Feynman diagram is shown in fig. 2.4. In the calculation, the external lines are associated with spinors and polarization vectors, whereas the internal lines are associated with propagators. The vertices are related to couplings between particles. That means that the perturbative order of a diagram can be determined by counting the number of vertices. Note that vertices can differ in order and this is in fact the case for QCD, where a 4-gluon vertex exists that is of order g_s^2 while the 3-particle vertices are of order g_s .

2.1.4 Running coupling

An issue arises in the calculation of loop diagrams. There, virtual particles with high energy or momentum lead to ultraviolet divergences. Consequently, the Feynman diagrams evaluate to infinity. These divergences can be regularized by introducing a cutoff scale that implements our ignorance about high energy scales, much above the collider energy. This leads to a renormalized definition of the fields, and their masses and couplings. The renormalized masses and couplings depend on a reference scale, the renormalization scale μ_R . Their values can be measured by experiment at a specific scale. To determine their values at other scales, they can be evolved with a renormalization group equation (known as the beta function), which

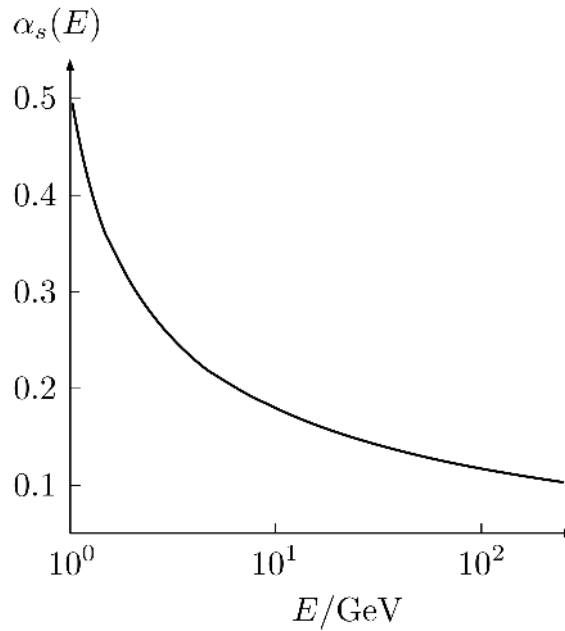


Figure 2.5: The strong coupling constant as a function of the energy. Figure taken from [161]



can be derived from perturbation theory. A consequence is that the values of the renormalized coupling constants in the Standard Model depend on the energy scale, a fact known as running coupling. For QCD, unlike QED, the coupling, α_s , decreases with energy, as can be seen in fig. 2.5. At high energies, the coupling between quarks is small, and they behave like free particles. This is called asymptotic freedom. However, at small energies we can never observe free quarks. Due to the strong coupling, they are always confined in hadrons, which are composite particles made of quarks and bound together by the strong interaction. The same running behaviour is true for the quark masses. However, in practice we often only consider running masses of the heaviest quarks, t and b , and assume the light quarks to be massless. This simplifies calculations while at high energies the approximation has negligible consequences.

Perturbation theory works well for QCD processes at high energies, where the strong coupling is small. Therefore, it allows us to describe scattering processes between hadrons at colliders like the LHC. If the hadrons are accelerated to high energies, their interaction is dominated by the scattering of individual quarks and gluons. The high energy scale suppresses softer effects that happen at a longer timescale than the hard interaction.

2.1.5 Hadron collisions

Equation (2.12) gives the differential cross-section for the scattering of two fundamental particles. However, the LHC collides protons, which are composite particles made of quarks and gluons. It is common to refer to quarks and gluons collectively as partons, and we adopt this convention here. To describe the scattering of two protons, it is not enough to calculate the cross-section of two partons, since we cannot observe which of the components interacted and what fraction of the momentum they carried. So we need to expand our toolbox. We assume that at high collision energies, the partons within the proton are almost collinear with its momentum. Furthermore, we suppose that due to their asymptotic freedom, individual partons collide mostly independently from the proton remnants. This situation is visualized in fig. 2.6, where an up-quark and a gluon leave their respective fast-moving protons to undergo a partonic interaction.

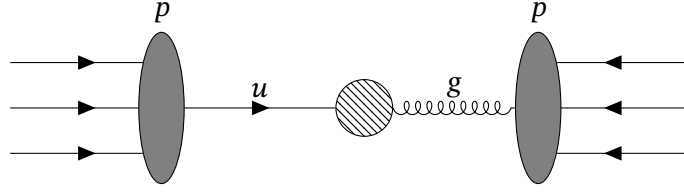


Figure 2.6: Exemplary sketch of a hadronic collision at high energy.

Let us denote the momenta of the colliding partons by

$$p_i = x_i P_i, \quad (2.16)$$

where P_i are the momenta of the two colliding protons, and x_i are the momentum fractions carried by the partons p_i . Necessarily, the fractions are restricted to $0 \leq x_i \leq 1$. With this notation, we can conceptually write down the total cross-section of a hadronic collision:

$$\sigma = \sum_{i,j} \int_0^1 dx_1 \int_0^1 dx_2 f_i(x_1) f_j(x_2) \hat{\sigma}_{p_i p_j \rightarrow \{p_f\}}. \quad (2.17)$$

The sum runs over all possible partons, i.e. quarks, antiquarks, and gluons. The partonic cross-section between partons i and j , as determined by eq. (2.12), is denoted as $\hat{\sigma}_{p_i p_j \rightarrow \{p_f\}}$. Moreover, the parton distribution functions (PDFs) $f_i(x_1)$ and $f_j(x_2)$ have been introduced. These capture the probability to find the partons of types i and j with momentum fractions x_1 and x_2 inside the protons 1 and 2, respectively. Since they comprise low-scale physics, the PDFs cannot be calculated in perturbation theory. Instead, they have to be determined from measured data.

There is a further complication due to the fact that partonic cross-sections can suffer from infrared divergences, which are triggered by soft and/or collinear emissions. To deal with this, we assume that we can factorize the cross-section into a perturbative and a non-perturbative contribution. There is good reason to believe that this is a sensible ansatz and some more specific factorization properties have been proven [162]. This factorization allows us to move the infrared divergences out of the partonic cross-section and into the definition of the PDFs. Thereby, we introduce another scale parameter, the factorization scale μ_F . This scale separates the short-distance, perturbatively calculable physics from the long-distance, non-perturbative physics. As a consequence, we have partonic cross-sections that can be calculated in some order of perturbation theory, and PDFs that can be determined from measured data at a specific scale μ_F . Similarly to the renormalized masses and couplings, the PDFs can be evolved to other energy scales using perturbation theory. The equations describing this evolution are known as Dokshitzer–Gribov–Lipatov–Altarelli–Parisi (DGLAP) equations [163–165]. Assuming factorization, the hadronic cross-section can be written as

$$\sigma = \sum_{i,j} \int_0^1 dx_1 \int_0^1 dx_2 f_i(x_1, \mu_F^2) f_j(x_2, \mu_F^2) \hat{\sigma}_{p_i p_j \rightarrow \{p_f\}}(x_1, x_2, \mu_F^2), \quad (2.18)$$

where the dependence on the factorization scale has been made explicit.

2.1.6 Jets

The result of simulating a high energy hadronic scattering event can be thousands of particles, with many of them being hadrons. Experimental data tells us that the hadrons are typically found in condensed cone-shaped arrangements, so-called jets. Consider as an example fig. 2.7, which shows an event with two jets of high transverse momentum, p_\perp , recorded by the ATLAS

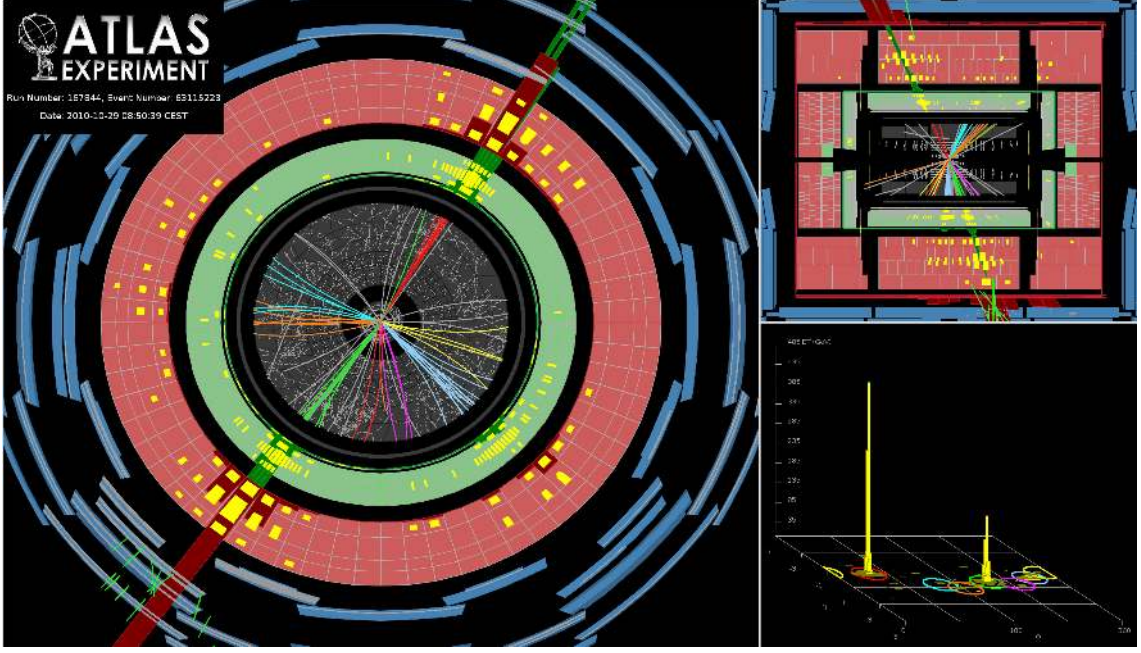


Figure 2.7: Event display of a dijet event recorded by ATLAS in 2010. Figure taken from [166]

collaboration. The left plot shows a cross-section through the detector, perpendicular to the beam axis. The tracks of the charged particles are projected onto the image layer. It can be seen that the two jets with the highest p_{\perp} , with tracks coloured in green and red, deposit a large amount of energy in the hadronic calorimeter (red annulus). In addition, the right plots show a cross-section through the detector along the beam axis and a histogram of measured transverse Energy (ET) in the pseudorapidity-azimuth (η - ϕ) plane. Clearly, the energy deposits are very localized, indicating that the particles are collimated.

From the theory perspective, jets form when highly energetic partons from the hard interaction dress themselves with soft and/or collinear emissions, so-called parton showers. At sufficiently low energy scales the partons form hadrons, whose energy deposits can be measured in a calorimeter. Jets are abundant in LHC data and almost all analyses rely on reconstructed jets.

To be able to compare experimental and simulated data, it is important to use a precise definition of jets, which can be applied to both simulated and measured events. Such a definition is available through a jet algorithm, which is a procedure for clustering a set of hadrons into jets. A useful property of a jet algorithm is infrared and collinear safety, meaning that the result of clustering is not sensitive to additional soft or collinear emissions. This is important for reliable comparisons between theory and data. The most important jet algorithm for the LHC, and the only one considered in this thesis, is the anti- k_T algorithm [167]. As a type of sequential recombination algorithm, it is infrared and collinear safe. These algorithms work by defining a distance measure d_{ij} between any two particles and clustering the particles based on their distance. For the anti- k_T algorithm, the distance measure is defined as

$$d_{ij} = \min(p_{\perp,i}^{-2}, p_{\perp,j}^{-2}) \frac{\Delta R_{ij}^2}{R^2}, \quad (2.19)$$

where $\Delta R_{ij}^2 = \Delta y^2 + \Delta \phi^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$, and $p_{\perp,i}$, y_i and ϕ_i are respectively the transverse momentum, the rapidity and the azimuth of the i -th particle. The parameter R is called the jet radius and has to be set manually. The clustering proceeds as described in algorithm 1.

Algorithm 1: The anti- k_t jet clustering algorithm.

```

1 define a list of entities containing all final state particles;
2 repeat
3   compute the distance measure  $d_{ij}$  for each pair  $(i, j)$  of entities;
4   compute the distance measure  $d_{iB}$  between each entity  $i$  and the beam  $B$ ;
5   determine the minimum  $\min(\{d_{ij}\}, \{d_{iB}\})$  of the set of all distance variables;
6   if minimum is a  $d_{ij}$  then
7     combine entities  $i$  and  $j$  by adding their four-momenta;
8   end
9   else
10    declare entity  $i$  to be a final state jet, and remove it from the list of entities;
11  end
12 until list of entities is empty;
    
```

The algorithm can also be stopped prematurely when a termination criterion has been reached. For example, one can choose to only consider jets with $p_{\perp,i} > p_{\perp,\text{cut}}$ and $d_{ij} > d_{\text{cut}}$.

2.2 The sampling problem in high energy physics

After the essential background has been introduced, the problem of sampling the phase space is explained in more detail in this section. To this end, some specific aspects are discussed in section 2.2.1. Then, in section 2.2.2, MC event generators are introduced as an important tool for tackling the sampling problem.

2.2.1 Sampling the differential cross-section

Combining eqs. (2.12) and (2.18), we can write down the fully differential cross-section of a $2 \rightarrow n$ parton level scattering process:

$$d\sigma_{p_1 p_2 \rightarrow \{p_f\}} = \frac{1}{2E_1 E_2 |\mathbf{v}_1 - \mathbf{v}_2|} f_1(x_1, \mu_F) f_2(x_2, \mu_F) |\mathcal{M}|^2 |\det J_{\Phi}| \times dx_1 dx_2 d\Phi_{p_1 p_2 \rightarrow \{p_f\}}. \quad (2.20)$$

This gives only one partonic contribution. To arrive at a hadronic cross-section, one needs to sum over all possible partonic initial and final states. The Jacobian determinant, $|\det J_{\Phi}|$, has been introduced to account for a possible change of variables that increases the sampling efficiency. For details, see section 2.3.

The main objective in this thesis is the sampling of eq. (2.20), i.e. the production of momenta $p_1, p_2, \{p_f\}$ with a distribution that follows the differential cross-section. This is in fact a challenging task. Since the parton momenta are given by four-momenta, we have to deal with $4n$ dimensions. However, four-momentum conservation and on-shell conditions reduce this number to

$$d = 3n - 4. \quad (2.21)$$

At high final state multiplicity, the number of dimensions can thus become quite large. Thereby, the costs of evaluating the matrix elements, $|\mathcal{M}|^2$, scale badly with the multiplicity. Furthermore, the matrix elements can be multimodal, and their peaks can be extremely narrow due to intermediate resonances with small decay widths and quantum-mechanical interference. Large samples, containing millions of events, are typically needed for experimental analyses.

In addition, it can be necessary to repeat the generation many times with different values of the renormalization and factorization scales, and for variations of the PDFs. This is useful for an uncertainty estimation. The variation of the scales, typically between $\mu/2$ and 2μ of the nominal value, hints at contributions from higher orders. A strong dependency on the scales is a sign that large corrections from higher orders can be expected. It is possible, though, to reduce the number of repeated runs by using on-the-fly reweighting techniques [168]. Another complication arises from the fact that the differential cross-section can be formally divergent. This necessitates regularization by using kinematic cuts, which leads to regions in PS with zero probability and vertical edges at the locations of the cuts. There can be further cuts to select specific regions of PS. At NLO, which is only touched on in passing here, we also have to deal with negative weight contributions.

As a first step, the sampling of valid PS configurations is required. In the simplest case, this is done uniformly. This task was solved long ago by the RAMBO algorithm [169]. The algorithm uniformly populates the volume of the n -body PS, eq. (2.13), which is known to be

$$V_n = \left(\frac{\pi}{2}\right)^{n-1} \frac{(Q^2)^{n-2}}{(n-1)!(n-2)!}, \quad (2.22)$$

where Q is the total momentum. Versions for massless and massive particles are available, but the massive version is not exactly uniform. Due to its simplicity, the algorithm can be easily applied to arbitrary PS sampling problems. However, the uniform sampling can be very inefficient. Therefore, realistic applications are restricted to simple processes. Furthermore, the algorithm can serve as a cross-check for more elaborate ones. A disadvantage of RAMBO is that it uses $4n$ instead of the necessary $3n - 4$ random variables. This problem is solved by the RAMBO on diet approach [170]. The idea is based on factorizing the PS by separating it into a sequence of $1 \rightarrow 2$ decays. Due to using the minimal number of random variables, the algorithm provides an invertible mapping from uniform random numbers to particle momenta.

The factorized approach of ref. [170] was also used in earlier, non-uniform approaches to PS generation [171–173]. Later, more specialized algorithms became available. These often focused on multi-parton QCD processes. An example is SARGE [174], which generates the antenna pole structure that provides the leading contribution to the structure of singularities in QCD amplitudes. It is given by the permutations of

$$\frac{1}{(p_1 \cdot p_2)(p_2 \cdot p_3) \cdots (p_{n-1} \cdot p_n)(p_n \cdot p_1)}. \quad (2.23)$$

For QCD amplitudes, SARGE is much more efficient than RAMBO. An even more efficient approach to antenna generation is given by HAAG, which uses a hierarchical strategy and a multichannel (see section 2.3.3) procedure. More recent PS generators are often integrated into ME generators and use the same strategies that are employed to evaluate the MES. These can be based on Feynman diagrams [175], like in AMEGIC [176] and MADGRAPH [177], or recursion relations [178], like in COMIX [179]. As part of full-featured MC event generators, they efficiently generate PS configurations for almost arbitrary processes.

2.2.2 Monte Carlo event generators

MC event generators are general purpose tools that can be used to calculate decay widths, total cross-sections, and exclusively simulate individual scattering events. See refs. [8, 180] for extensive reviews of modern MC event generators. They are based on first principles, models for non-perturbative phenomena, and parameters tuned with measured data. At the core of event generation lies the partonic hard scattering event, which is the part that is almost exclusively considered in this thesis. A key assumption in event generators is that the hard process is factorized from parton showers and non-perturbative phenomena. This allows to simulate

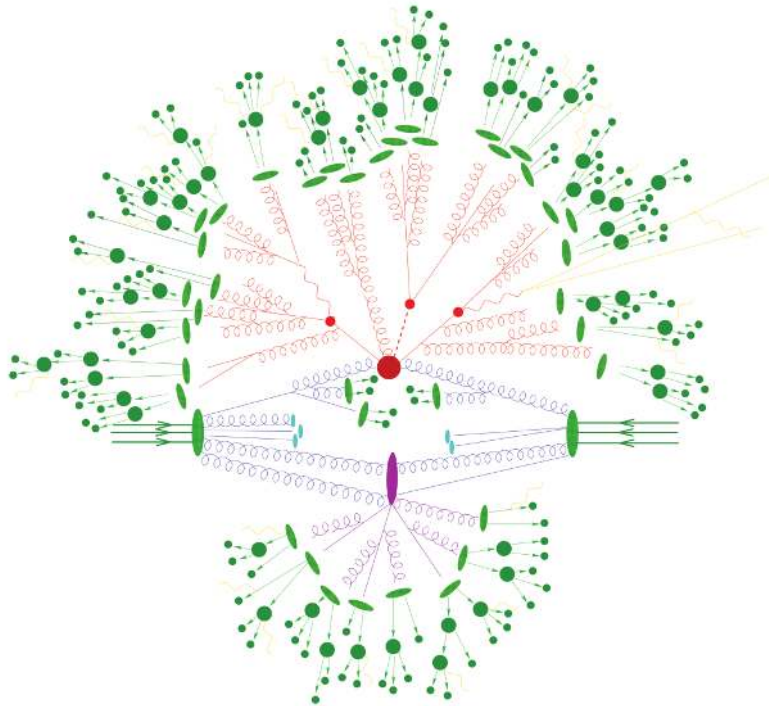



Figure 2.8: Sketch of a $t\bar{t}H$ event produced from two hadrons colliding at high energy as generated by an event generator. The two hadrons (green) enter from left and right. After initial state radiation (blue), a hard interaction between two gluons takes place (big red blob in the centre). The final state particles undergo final state radiation (red), and the top quarks and the Higgs boson decay (small red blobs). A secondary partonic interaction is shown in purple. At sufficiently low scales, the produced partons hadronize (light green blobs) and eventually the hadrons decay (dark green blobs). Electroweak radiation also occurs (yellow). Figure taken from [192] .

these parts separately. For hadronic collisions, PDFs are used to determine the contributions of the partonic channels. To calculate the scattering matrix elements of the hard process, an event generator interfaces specialized ME generators. These tools are largely automated and can be used for almost arbitrary scattering processes. Examples for tree-level ME generators are ALPGEN [181], AMEGIC [176], COMIX [179], MADGRAPH [182], and WHIZARD [183]. The set of QCD and electroweak one-loop generators includes MADLOOP [184, 185], MCFM [186, 187], NJET [188], OPENLOOPS [189, 190], POWHEG BOX [191], and RECOLA [189, 190]. In combination with an efficient PS generator, which provides the kinematic configurations as inputs, these tools can be used to generate inclusive fixed-order events at the parton level. General-purpose event generators are able to extend these into fully differential and exclusive simulations by adding a parton shower simulation, particle decays, hadronization models, and multiple interactions per collision. With all these ingredients, an event can look like the example shown in fig. 2.8. Thereby, perturbative and non-perturbative effects are integrated into a single simulation.

Although automated approaches exist by now, the combination of fixed order MEs and parton showers is a non-trivial task. They aim to describe different regions of PS, with fixed order MEs most accurate for well-separated, hard partons, while partons showers are designed for the soft and collinear parton emissions. Furthermore, with parton showers it is possible to generate many more partons than with the computationally demanding MEs. A problem for the combination of the two is that MEs deliver an inclusive description, while parton showers are exclusive in the number of final state partons. In addition, one needs to be careful to avoid double counting, i.e. that regions of PS are generated by both, MEs and showers. These

problems are addressed through matching [193–201] and merging [202–223] techniques.

2.3 Established methods and their shortcomings

Monte Carlo (MC) event generators make use of various algorithms and techniques for sampling. The common basis for them is the MC method. In this section, the established methods used in MC event generators are introduced, and their limitations are discussed. This provides the basis for the approaches to improvement that are presented in the following chapters. The crude MC method and its use for numerical integration are explained in section 2.3.1. In addition, importance sampling is introduced as a technique to reduce the variance of the integral estimate. In section 2.3.2, the adaptive importance sampling algorithm VEGAS is presented. In this context, problems with non-factorizable functions and cuts in phase space are discussed. Following that, the multichannel method is introduced in section 2.3.3 as an efficient approach to multimodal target functions. It is furthermore shown how to combine a multichannel sampler with adaptive remappings based on the VEGAS algorithm. Since all approaches based on importance sampling require the efficient generation of non-uniform random variables, this topic is covered in section 2.3.4. To this end, the inverse transform method and the rejection sampling algorithm are introduced. Finally, the relevance of rejection sampling for unweighted event generation in the context of experimental analyses is explained in section 2.3.5.

2.3.1 Monte Carlo sampling

For integration problems with more than a few dimensions the MC method is typically the tool of choice since its scaling behaviour is superior to deterministic methods such as quadrature. The idea is simple. Let $f : \Omega \subset \mathbb{R}^d \rightarrow [0, \infty)$ be a function defined over the d -dimensional unit hypercube $\Omega = [0, 1]^d$. Its integral is given by

$$I = \int_{\Omega} f(\mathbf{u}) \, d\mathbf{u}. \quad (2.24)$$

According to the MC method we sample points \mathbf{u}_i uniformly from Ω and by taking the mean over the corresponding function values we get an estimate of the integral:

$$I \approx E_N = \frac{1}{N} \sum_{i=1}^N f(\mathbf{u}_i). \quad (2.25)$$

From the law of large numbers it follows that E_N converges to I for $N \rightarrow \infty$. For finite N there is an error which can be estimated by the standard error of the mean:

$$\delta E_N = |I - E_N| \approx \widehat{\delta E_N} = \frac{\sigma_N}{\sqrt{N}}, \quad (2.26)$$

where σ_N is the sample standard deviation given by

$$\sigma_N = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (f(\mathbf{u}_i) - E_N)^2}. \quad (2.27)$$

We call $\widehat{\delta E_N}$ the MC error. Note that it is a statistical property that has meaning only for a large number of independent runs. For a single run the estimate can be completely wrong.

An example for uniform MC sampling is shown in fig. 2.9. The target is a truncated Gaussian distribution with mean $\mu = 0.5$ and variance $\sigma^2 = 0.05$ on the interval $[0, 1]$. A sample of 5000

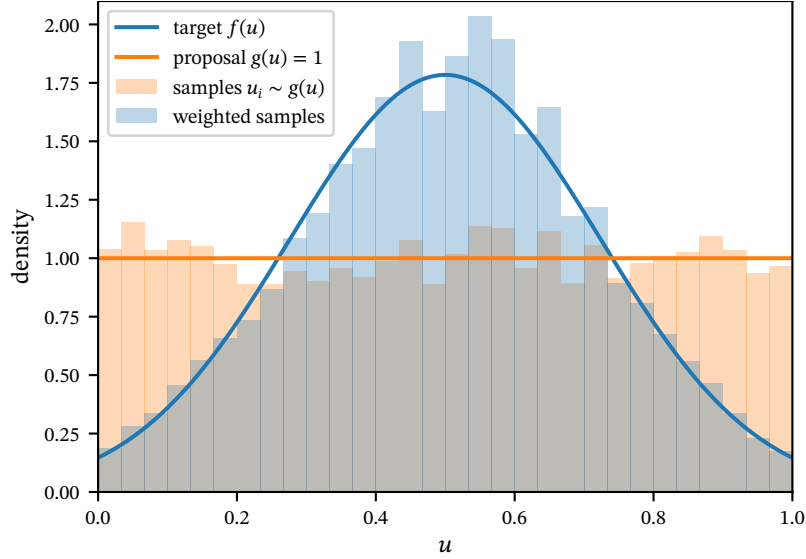


Figure 2.9: Uniform sampling of a one-dimensional target function.

points drawn from a uniform proposal is shown as a histogram. The histogram approximates the uniform distribution. However, if the points are filled into a weighted histogram, with their weight given by the value of the target function, $f(u_i)$, the histogram approximates the target function instead. In essence, fig. 2.9 is a visual illustration of the basic principle of the MC method.

Equation (2.26) represents the famous $1/\sqrt{N}$ scaling of MC integration. It is independent of the number of dimensions but at high dimensionality one might need an undesirably large number of points to get an integral estimate with an acceptable MC error. A way out is offered by variance reduction methods, which aim at reducing σ_N . These can be very effective because halving the variance has the same effect on the MC error as quadrupling the number of points. Arguably the most important method for HEP is importance sampling. The idea is to sample from a non-uniform probability density function (PDF)* $g : \Omega \rightarrow [0, \infty)$. As a consequence, the integral in eq. (2.24) becomes

$$I = \int_{\Omega} \frac{f(\mathbf{u})}{g(\mathbf{u})} g(\mathbf{u}) \, d\mathbf{u} = \int_{\Omega} \frac{f(\mathbf{u})}{g(\mathbf{u})} \, dG(\mathbf{u}), \quad (2.28)$$

where G denotes the cumulative distribution function (CDF) corresponding to g . The MC estimate of the integral is now given by

$$E'_N = \frac{1}{N} \sum_{i=1}^N \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)} = \frac{1}{N} \sum_{i=1}^N w_i, \quad (2.29)$$

where the \mathbf{x}_i are sampled from the distribution g . To make clear that they are drawn from a non-uniform distribution, we switched the notation for the random points to \mathbf{x}_i . We refer to the w_i as MC weights. The sample standard deviation becomes

$$\sigma'_N = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (w_i - E'_N)^2}. \quad (2.30)$$

It follows that the MC error can be reduced by reducing the variance of the weights w_i . This

*I use the acronym PDF for both probability density functions and parton distribution functions. It should be clear from the context, which of the two is meant.

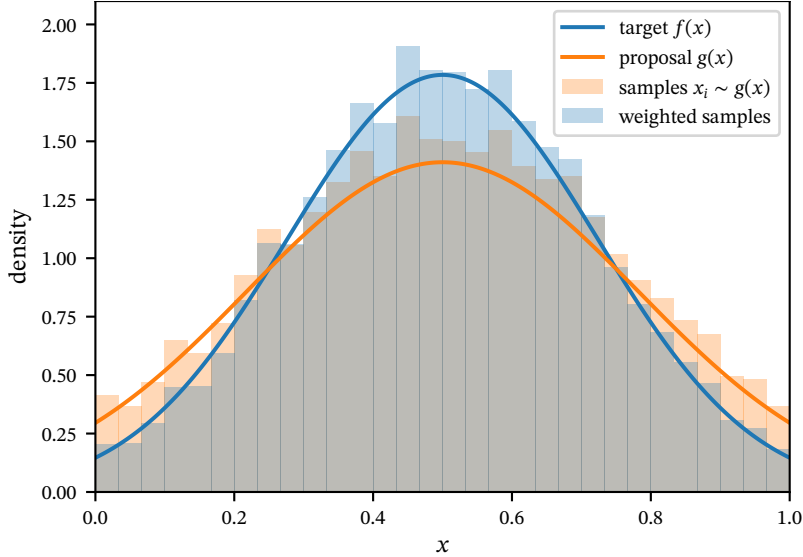


Figure 2.10: Importance sampling applied to a one-dimensional target function.

means that one should choose a $g(\mathbf{x})$ that is as similar as possible to the shape of the integrand $f(\mathbf{x})$. However, there is a restriction, namely that we need to be able to efficiently sample from g . Ideally, its quantile function, i.e. the inverse of the CDF, is known analytically such that inverse transform sampling (cf. section 2.3.4) can be applied.

In fig. 2.10 the importance sampling method is illustrated. The target is the same as in fig. 2.9. Another truncated Gaussian is used as the proposal. It has a larger variance than the target, $\sigma^2 = 0.08$ compared to $\sigma^2 = 0.05$. Again it can be seen that by weighting the points according to their weight, w_i , the sample distribution can be shifted from the proposal to the target. However, compared to uniform sampling, the distance between the proposal and the target is much shorter. Thus, the spread of the weights is smaller.

2.3.2 Adaptive importance sampling—the Vegas algorithm

VEGAS is an adaptive importance sampling algorithm that is widely used in HEP. It is based on a piecewise constant density defined over the unit hypercube $[0, 1]^d$. In d dimensions the density function is factorized into d functions:

$$q(\mathbf{x}) = \prod_{i=1}^d q_i(x_i). \quad (2.31)$$

Each function $q_i(x_i)$ is divided into N_i bins which are defined by their $N_i + 1$ boundaries $0 = x_{i,0} < x_{i,1} < \dots < x_{i,N_i} = 1$. Given the bin boundaries, the function can be evaluated at any point x_i using

$$q_i(x_i) = \frac{1}{N_i(x_{i,l} - x_{i,l-1})}, \quad \text{with } x_{i,l-1} \leq x < x_{i,l}. \quad (2.32)$$

In the VEGAS algorithm, each bin is sampled with equal probability and the bin widths are adapted to minimize the variance of the integral estimate. As a consequence, narrow peaks lead to many thin bins while flat, low probability regions are covered by few wide bins. Sampling from $q(\mathbf{x})$ is easy through the inverse transform method (cf. section 2.3.4). In each dimension, a bin is randomly chosen, and a point is sampled from it uniformly. Since the x_i are independent of each other, they can be sampled one after the other or in parallel. The actual algorithm for adjusting the bin widths is not relevant here. Details can be found in ref. [32].

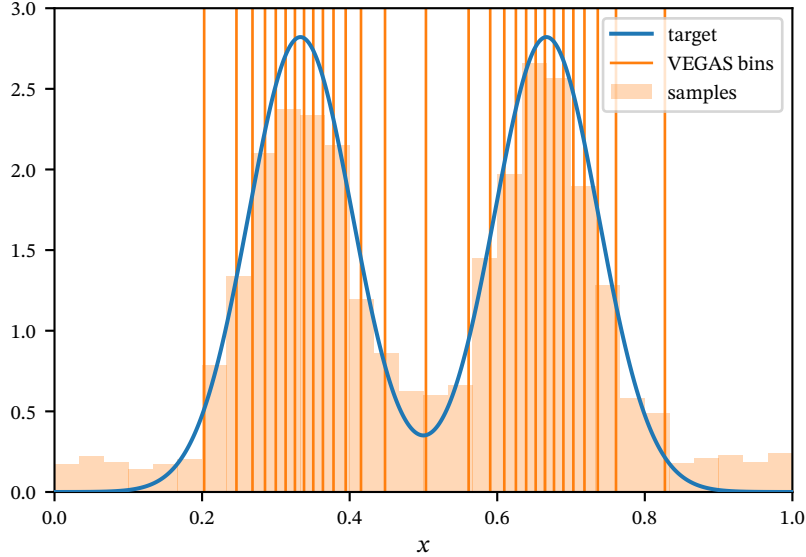


Figure 2.11: Sampling with VEGAS applied to a one-dimensional multimodal target.

In fig. 2.11 the result of applying VEGAS to a one-dimensional target function is shown. The target is a mixture of two Gaussians with peaks at $\mu_1 = 1/3$ and $\mu_2 = 2/3$. A VEGAS density with 30 bins has been trained on batches of 1000 points for 10 iterations. The adapted bin boundaries are shown in the figure. It can be seen that the bins concentrate around the peaks while there are fewer bins in regions where the target function has smaller values. The widest bins are located at the domain boundaries. A histogram of a sample from the VEGAS density is also shown. At the outermost bins, it can be seen quite clearly that VEGAS uses a uniform distribution within each of its bins. Therefore, within each bin, the target function is approximated by a constant function. Depending on the width of the bin and the gradient of the target function, this can lead to large deviations. In the example, VEGAS overestimates the target function at the domain boundaries due to the wide bins and underestimates the peaks as a consequence. Using more bins would reduce the deviations in this example.

As stated above, it is straightforward to sample a VEGAS density. Accordingly, there is a simple transformation from a uniform distribution to the VEGAS one. On the one hand, this allows to use the inverse transform method for sampling. On the other hand, this means that VEGAS can be used more generally as a mapping $\phi : \Omega \rightarrow \Omega$ to transform one distribution into another. Consider for example fig. 2.12, where the mapping $u \mapsto x$ is shown for the VEGAS density trained on the multimodal target shown in fig. 2.11. The bin boundaries are indicated in the background, and it can be seen that the mapping function is linear between the bin boundaries. At the points where the linear parts connect, the function is non-differentiable. Such an adaptive mapping can be very effective in situations where we have some but not full knowledge about the target distribution in the form of a sampling density. Consider, for example, the matrix element squared of a scattering process with an intermediate resonance, where the shape is dominated by a relativistic Breit-Wigner distribution, which can be sampled with the inverse transform method. This models the denominator of the propagator of an unstable particle. However, it neglects the numerator and other contributions. To reduce the mismatch between the target and the sampling distribution, one can train a VEGAS mapping on the function that remains after mapping out the resonance with a Breit-Wigner distribution. This approach is more efficient than using VEGAS standalone because the sharp resonance is already flattened out. It would otherwise require many bins to model. Thus, there are more bins available to adapt to other features.

The factorized definition of the VEGAS density, eq. (2.31), is a big advantage with regard to the efficiency of the adaptation. The computational costs for the optimization scale linearly

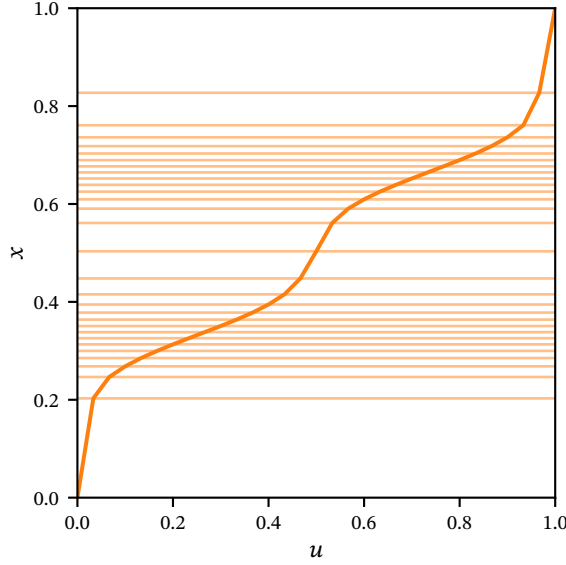


Figure 2.12: VEGAS as a mapping $u \mapsto x$ for a one-dimensional multimodal target.

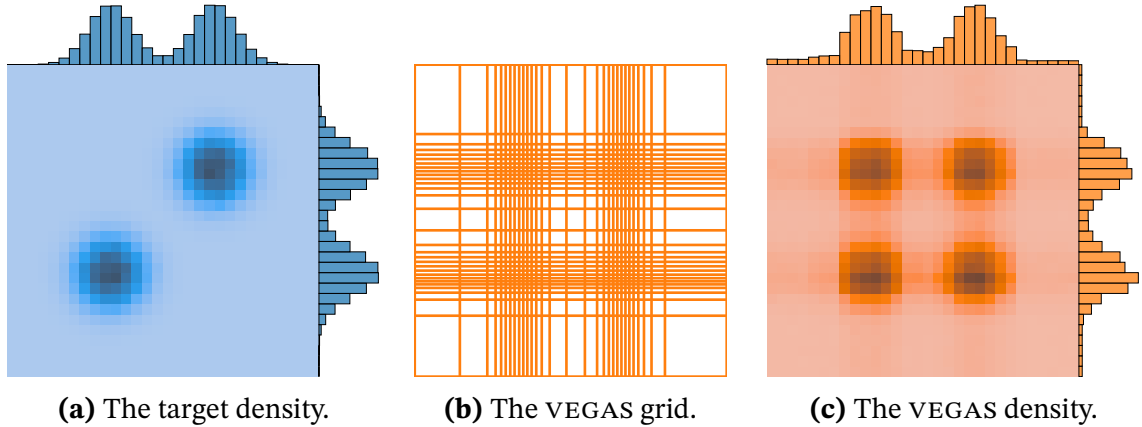


Figure 2.13: Two-dimensional example for VEGAS.

with the number of dimensions. It is, however, also a big restriction with regard to flexibility. Inevitably, VEGAS struggles to adapt to distributions that are not factorizable. An example can be seen in fig. 2.13. The target, shown in fig. 2.13a, has been chosen to be a two-dimensional mixture of two Gaussians with peaks at $\mu_1 = (1/3, 1/3)$ and $\mu_2 = (2/3, 2/3)$. An optimized VEGAS grid is shown in fig. 2.13b and the corresponding density is shown in fig. 2.13c as a two-dimensional histogram. The density features two additional peaks, at $(2/3, 1/3)$ and $(1/3, 2/3)$, which have no counterpart in the target. This is because the factorized ansatz only allows VEGAS to adapt to the marginal distributions. As a consequence, there are performance losses for unsuitable targets. In the example shown here, the problem can be solved by rotating the target function such that the peaks are aligned with one of the coordinate axes. When this is not possible, a multichannel approach, as discussed in section 2.3.3, can help. Both approaches require detailed knowledge about the target function.

Phase space cuts

Especially for unweighted event generation (cf. section 2.3.4), where large weights lead to low efficiencies, the behaviour of VEGAS in the presence of phase space cuts can be crucial. The following examples are used to analyse this behaviour. First, a quite trivial one-dimensional

function defined over $[0, 1]$ is considered, whose PDF is constant except for the range $x < 0.3$, for which it is zero. The function has been plotted in fig. 2.14a. A VEGAS grid has been adapted to the target density using 10 iterations with 1000 function evaluations each. The grid has 500 bins. In fig. 2.14b the resulting VEGAS density is shown next to the target density in a region around the cut at $x = 0.3$. It can be seen that the boundary of the leftmost bin is not aligned with the cut edge. The explanation for this is simple: VEGAS starts from a uniform grid. In the first iteration, the bin boundaries are adapted based on a uniformly distributed sample of points. Since there are no non-zero points below $x = 0.3$, the leftmost bin is stretched up to the smallest point above the cut. In the subsequent iterations the boundary can move closer to the cut if a point between the boundary and the cut is sampled. However, this rarely happens because the corresponding bin is very wide. Even with more training data or more bins a gap between the bin boundary and the cut would remain. While the VEGAS density would still be well suited for integration, the large weights for points just above the cut would pose a problem for unweighting. In this example, the weights reach values of 215 next to the cut while they are close to one in the remaining phase space.

The one-dimensional problem allows for several ways out which can also be applied to the situation where a multidimensional target features a cut that is aligned with one of the coordinate axes. If the cut is known in the sampling coordinates, the sampling space can simply be reduced to the uncut region. Otherwise, one could change the algorithm to align the boundaries of empty bins with the points closest to the cut for which $f(x) = 0$ instead of the non-zero ones. This way, the cut edge would be contained in a non-empty bin, alleviating the undersampling next to the cut. Another solution would be to use a more inclusive cut during the adaptation of the VEGAS grid and to use the desired value of the cut only afterwards. This would allow VEGAS to properly adapt to the target values next to the cut. The requirement is that the target function can be evaluated and is not singular for the more inclusive cut.

To determine the consequences of a cut that is not aligned with the coordinate axes we consider the two-dimensional example shown in fig. 2.15a. The target density is constant except for the region $x + y < 0.3$, where it is zero. A VEGAS grid is adapted to it using the same parameters as above. In fig. 2.15b the resulting density is shown. It can be seen that where $x > 0.3$ and $y > 0.3$ VEGAS reproduces the target well. However, since it only adapts to the marginal distributions, it fails to converge to the target in the remaining phase space. From $x = 0.3$ to $x = 0$ and from $y = 0.3$ to $y = 0$ the density falls off approximately linearly.

A close-up view of the region $0 < x < 0.08$, $0.23 < y < 0.31$ is shown in fig. 2.16. The individual bins of the VEGAS grid can be clearly distinguished. The location of the cut is indicated, and it can be seen how it cuts the rectangular bins at an angle. Obviously the bin boundaries are not aligned with the edge of the cut, and it can be concluded that the density close to the cut is not a good approximation of the target. More quantitative statements are facilitated by the illustration in fig. 2.17. Here we consider the two line segments $x + y = 0.3$ and $x + y = 0.9$ in the region $-0.3 < x - y < 0.3$. For the case $x + y = 0.9$, far away from the cut, the VEGAS density is a good approximation of the target density, fluctuating around it with a slight offset. However, for the case $x + y = 0.3$, at the very edge of the cut, VEGAS severely underestimates the density. The lowest density is given by $p_{\text{vegas}} = 0.84$ leading to a maximum weight of $w_{\text{max}} = 1.25$. Admittedly, this is less dramatic than in the one-dimensional example. The essential insight, though, is that the maximum weight is found next to the cut boundary. Its value is highly specific to the problem at hand since it depends on the shape of the target function, its dimensionality and the definition of the cut.

If, like in the last example, the cut is not aligned with the coordinate axes, there is no simple cure to the undersampling problem. In some cases, it may be possible to change to another coordinate system, where the cut is aligned with one of the axes. Alternatively, one can try to substitute, for the time the VEGAS grid is adapted, the target value in the cut region using some kind of inter- or extrapolation. In our examples, VEGAS would be perfectly able to learn the

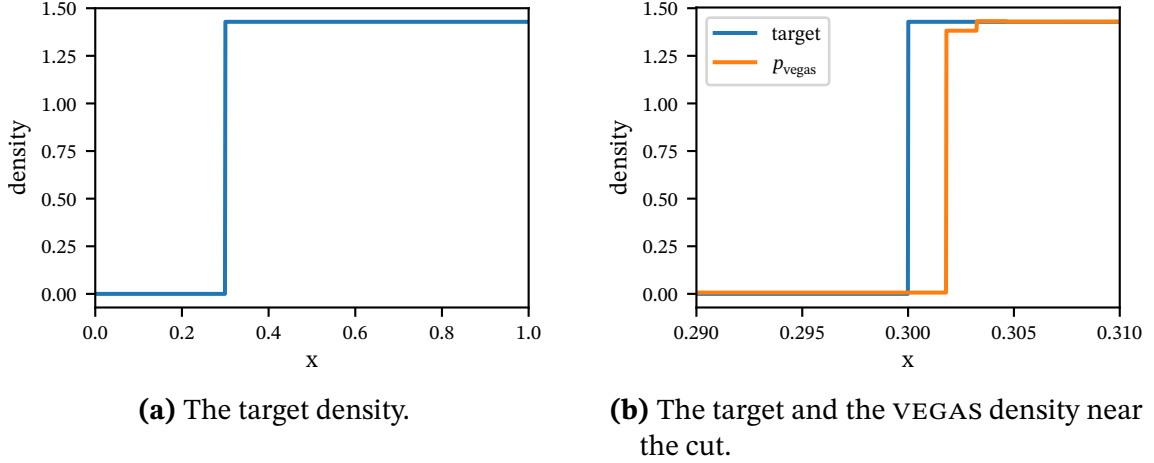


Figure 2.14: One-dimensional example for VEGAS encountering a phase space cut.

target densities if we would assume a constant value in the cut region. In many cases, however, no countermeasures will be taken for the sake of simplicity. Nevertheless, it makes sense to be aware of these characteristics.

2.3.3 Multichannel sampling

It is difficult to find proposal distributions that are well-suited for multimodal target functions, i.e. functions with multiple peaks. The difficult part is not to come up with a multimodal proposal, but to find one that we know how to produce samples from efficiently. In differential cross-sections we often find multiple peaks scattered over phase space due to e.g. intermediate resonances or quantum interference effects. While the overall distribution is complicated, we have rather good knowledge about individual terms. For example, resonances can be described by a Breit-Wigner distribution. A common trick in these situations is therefore to use a multichannel, or mixture, density with N_c channels,

$$g(\mathbf{x}) = \sum_{j=1}^{N_c} \alpha_j g_j(\mathbf{x}), \quad (2.33)$$

where the mixture weights, $\alpha_j \geq 0$, sum to one:

$$\sum_{j=1}^{N_c} \alpha_j = 1. \quad (2.34)$$

The channels, $g_j(\mathbf{x})$, are densities which we can sample easily, e.g. using inverse transform sampling (cf. section 2.3.4). Due to the normalization of the mixture weights, the multichannel distribution is a PDF. Provided the individual channels are chosen wisely, it is easy to sample the multichannel distribution: One chooses one of the channels at random, according to their mixture weights, and samples that channel. The mixture weights can be set manually or adapted automatically. A common strategy in HEP is to adapt the weights based on their influence on the variance of the integral [224]. Many other optimization algorithms, readily available in modern ML libraries, can be used as well. If a multichannel density is used for importance sampling, one should note that, according to eq. (2.33), for each point every single channel has to be evaluated. Therefore, it can be problematic if the number of channels becomes too large. It can be beneficial to monitor the contribution to the integral of each channel and deactivate the least important ones by setting their weight to zero.

A fruitful interaction results from the combination of a multichannel density with mappings

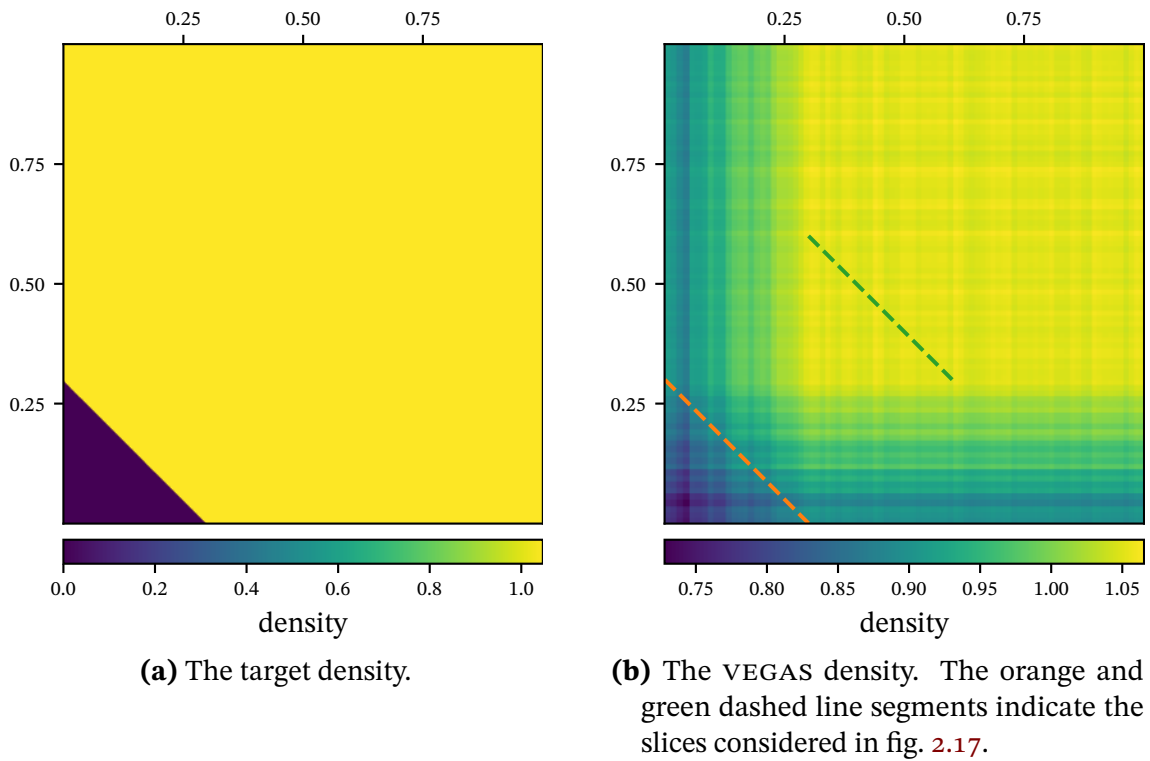


Figure 2.15: Two-dimensional example for VEGAS encountering a phase space cut that is not aligned with the coordinate axes. The x -axis is shown in the horizontal direction and the y -axis is shown in the vertical direction. The colourbars below the plots indicate the values of the densities, with blue corresponding to low values and yellow corresponding to high values.

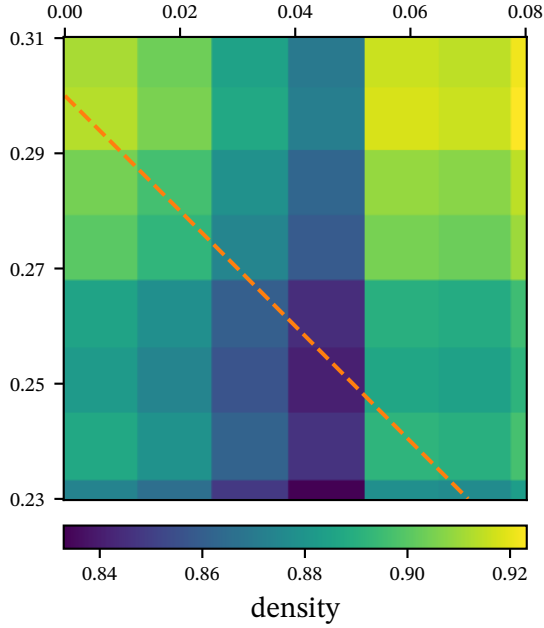


Figure 2.16: A close-up view of the VEGAS density around the edge of the cut. The orange dashed line indicates the cut at $x + y = 0.3$. Note that the rasterized appearance comes from the VEGAS grid and does not correspond to any histogram bins.

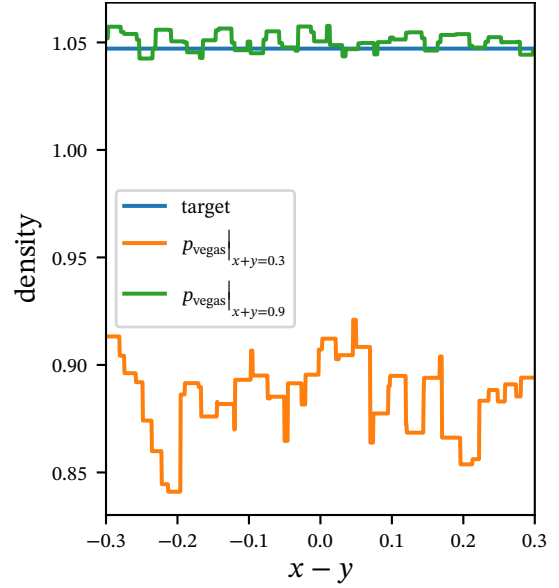


Figure 2.17: The VEGAS density along the diagonal slices $x + y = 0.3$ and $x + y = 0.9$ for $-0.3 < x - y < 0.3$. Due to VEGAS' grid, the density is piecewise constant. The target density is shown as a reference.

based on Vegas. As seen in section 2.3.2, the VEGAS algorithm can be used to automatically construct a sampling distribution for a given target. This works well as long as the target function is factorizable, i.e. there are no correlations between the dimensions. The factorizability depends on the coordinate system, hence a coordinate transform can help VEGAS to properly adapt to a target. However, as noted in ref. [225], there are functions that can not be factorized in a single coordinate system. In these cases, the multichannel technique provides an attractive solution: One identifies parts of the target function that are factorizable individually and constructs a channel in a suitable coordinate system for each of these. To further optimize the multichannel density, one can use VEGAS as a mapping for each channel, as described in section 2.3.2. By doing this, VEGAS only has to deal with factorized functions, while it is still possible to encode prior knowledge in the channel mappings. Under this approach, the MC integral estimate becomes

$$E_N = \sum_{i=1}^{N_c} \alpha_i \left\langle \frac{f \circ \phi_i}{g \circ \phi_i} \right\rangle_{g_i}, \quad (2.35)$$

where the average is evaluated for points sampled from the channel distribution g_i , and where $\phi_i : \Omega \rightarrow \Omega$ is the VEGAS map associated with the channel i . Put into words, eq. (2.35) means that for each sampled point we choose a channel at random, according to the weights α_i , and draw a uniform random number, u_j . Using the channel's VEGAS map, we map u_j to a point $v_j \in \Omega$. This point is the input for the channel g_i , which maps it to another point, x_j . Finally, we evaluate the weight $w_j = f(x_j)/g(x_j)$.

If the target function features phase space cuts, e.g. due to regularized singularities, it is possible to include knowledge about the cuts in the channels of the sampling distribution. This can increase the sampling efficiency with regard to the cuts, i.e. fewer points have to

be rejected because of cut constraints. A complication arises from the fact that the cuts are typically defined in coordinates that are motivated by physics, like transverse momentum or rapidity, and not in the ones that are used for sampling. To be considerate of the cuts in the channel definitions, they have to be translated to the respective coordinate system, which is not always possible in a factorized way. Therefore, it is often not possible to fully respect the cuts within the channel definitions. In addition, non-factorizable cuts can generally be problematic for VEGAS, cf. section 2.3.2.

For complicated scattering processes there can be numerous subprocesses contributing. This implies many channels in the multichannel density, which can be prohibitive since we need to generate a number of events from each channel in order to have training data to adapt to. To reduce the number of channels, one can use permutation invariance or group together subprocesses with similar kinematics. However, the latter can also lead to significant performance losses if not chosen carefully and is therefore a delicate balancing act.

2.3.4 Non-uniform random variables

We have seen above that our problems can require non-uniformly distributed random variables, for example if we want to apply importance sampling. However, we have not yet discussed how to generate these. Since uniform (pseudo-) random numbers are always easily available, the standard method is to use a mapping to transform these into non-uniform ones. Such a mapping can be found by inverting the CDF of the desired distribution. This can be a difficult task, but for many typical distributions the solutions are known and often implemented in standard software libraries. While the inverse transform method can be regarded as the gold standard, rejection sampling provides an alternative that can always be used, regardless of the desired distribution. However, it is inefficient because it is based on discarding points that have already been generated. Often, many trial points have to be generated for a single accepted point. Below, both methods are introduced.

Inverse transform sampling

To see how the inverse transform method works, let us consider a continuous real-valued random variable X with CDF

$$F(x) = \Pr(X \leq x) = \int_{-\infty}^x f(t) dt, \quad (2.36)$$

where $f(t)$ is the PDF. The CDF $F(x)$ is a monotone and continuous function, and has an inverse F^{-1} , the quantile function. If we now generate a uniform random variable U and map it to $X = F^{-1}(U)$, we get the cumulated probability

$$\Pr(X \leq x) = \Pr(F^{-1}(U) \leq x) = \Pr(F(F^{-1}(U)) \leq F(x)) = \Pr(U \leq F(x)) = F(x). \quad (2.37)$$

Therefore, X has the desired distribution.

As an example, consider the Cauchy distribution, whose PDF is given by

$$f(x) = \frac{1}{\pi\gamma \left[1 + \left(\frac{x-x_0}{\gamma} \right)^2 \right]}, \quad (2.38)$$

where x_0 specifies the location of the peak and γ is a scale parameter. By integrating, the CDF

$$F(x) = \frac{1}{\pi} \arctan\left(\frac{x-x_0}{\gamma}\right) + \frac{1}{2} \quad (2.39)$$

can be found. Its inverse is

$$F^{-1}(u) = x_0 + \gamma \tan\left[\pi\left(u - \frac{1}{2}\right)\right]. \quad (2.40)$$

Therefore, taking $X = x_0 + \gamma \tan[\pi(U - 0.5)]$ for a uniform U generates Cauchy random variables.

The Cauchy distribution is handy for HEP applications because it is related to the relativistic Breit-Wigner distribution, which can be used to model intermediate resonances. It is defined as

$$f(s) = \frac{k}{(s - M^2)^2 + M^2\Gamma^2}, \quad (2.41)$$

where s is the centre-of-mass energy squared, and M and Γ are the mass and the width of the resonance, respectively. The constant k ensures proper normalization. By substituting the corresponding parameters, eq. (2.40) can also be used to generate the Breit-Wigner distribution.

Sometimes we want to generate only part of a distribution, restricted to an interval $a < X \leq b$. For example, we may be interested only in the region around its peak. This can be achieved by truncating the distribution. The truncated PDF is given by

$$f(x|a < X \leq b) = \begin{cases} \frac{f(x)}{F(b) - F(a)} & \text{if } a < x \leq b, \\ 0 & \text{else,} \end{cases} \quad (2.42)$$

and is a density itself. For the CDF, the simple relation is

$$F(x|a < X \leq b) = \frac{F(x) - F(a)}{F(b) - F(a)}. \quad (2.43)$$

That means the inverse transform method can be easily adapted for truncated distributions. We can use the inverse CDF of the distribution before truncation, but transform the uniform random variable before putting it into the mapping. The transformation simply shifts and scales the variable according to

$$U \mapsto (F(b) - F(a))U + F(a). \quad (2.44)$$

Rejection sampling

The rejection sampling algorithm is simple. Consider the situation shown in fig. 2.18, where a target function $f(x)$ and a proposal density $g(x)$ are given. It is necessary to find a scaling factor c that ensures that $c \times g(x) \geq f(x)$, i.e. the scaled proposal lies above the target everywhere. If the maximum of the ratio between $f(x)$ and $g(x)$ is known, this can be achieved by taking

$$c = \max\left(\frac{f(x)}{g(x)}\right) = w_{\max}, \quad (2.45)$$

that is the maximum weight of all possible events. For a weighted trial event x_{trial} , we define the acceptance probability

$$p_{\text{accept}}(w_{\text{trial}}) = \frac{w_{\text{trial}}}{c} = \frac{1}{c} \frac{f(x_{\text{trial}})}{g(x_{\text{trial}})} \leq 1. \quad (2.46)$$

To decide whether a trial event is accepted as an unweighted event, a uniform random number, R , is drawn between 0 and 1. If $R < p_{\text{accept}}$, the event gets accepted, otherwise it gets rejected. From eq. (2.46) it is clear that events with large weights have a higher probability of being

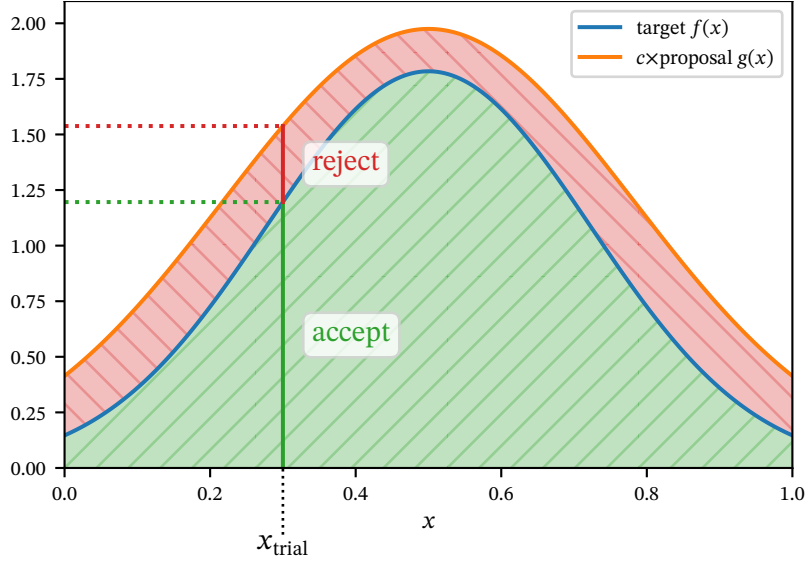


Figure 2.18: Rejection sampling applied to a one-dimensional target function.

accepted than events with small weights. In the end, all accepted events form an unweighted event sample, where all events contribute equally. The underlying distribution is equal to the target distribution.

Rejection sampling can lead to many events getting rejected and their information being lost. Nevertheless, the rejected events have to be generated and consume computational resources. This is exactly the problem highlighted by fig. 1.5 in the introduction, where for high-multiplicity processes many generated events have to be rejected. As a measure for the effectiveness of unweighted event generation, we define the unweighting efficiency as the average acceptance probability,

$$\eta = \langle p_{\text{accept}} \rangle = \frac{\langle w \rangle}{c}. \quad (2.47)$$

It tells us the average ratio between the number of accepted events and the number of trial events. Note that this definition is not robust, especially for small event samples. If phase space regions with large weights, and hence small acceptance probabilities, have not been probed, adding more events can decrease the unweighting efficiency dramatically. However, similar arguments apply to other statistical estimates, like cross-section integrals and variances. As the number of events increases, the unweighting efficiency becomes increasingly reliable. In many cases, it is certainly useful as a performance indicator and to compare the efficiency for different scattering processes or sampling algorithms. A more robust efficiency estimator can be defined for partial unweighting, discussed in section 4.1.

Typically, variance reduction techniques, e.g. importance sampling, imply an increase of the unweighting efficiency. This is clear, since reducing the variance of the event weights should often bring the mean closer to the maximum. However, the two goals, variance reduction and increasing the unweighting efficiency, do not align. In general, a reduction of the variance is easier to achieve than an increase of the unweighting efficiency, since the latter is sensitive to individual large-weight events. An example that aims at directly increasing the unweighting efficiency is FOAM [33], an adaptive algorithm that provides an alternative to VEGAS. However, FOAM has a less favourable scaling behaviour than VEGAS, which hinders its application to high-dimensional problems.

2.3.5 Rejection sampling for unweighted event generation

If an event sample is to be processed by a more time-consuming calculation (on a per-event basis), e.g. a detector simulation, it can be beneficial to unweight the events. That means to apply rejection sampling in order to generate a unit-weight sample. In this process, part of the events (and often the vast majority) is discarded. As a consequence, there is a loss of information which can be quantified in a reduction of the effective sample size given by

$$N_{\text{eff}} = \frac{\left(\sum_{i=1}^N w_i\right)^2}{\sum_{i=1}^N w_i^2}. \quad (2.48)$$

To see why it can still pay off to generate unweighted events, a back-of-the-envelope calculation is helpful. Let us assume that generating one weighted event takes $t_{\text{weighted}} = 10^{-3}$ s and the unweighting efficiency is $\eta = 10^{-3}$, such that generating one unweighted event takes $t_{\text{unweighted}} = 1$ s on average. Suppose that $N = 1\text{M}$ weighted events have been generated and the effective sample size of the weighted sample is $N_{\text{eff}} = 500\text{k}$. After unweighting, the sample size and the effective sample size would be $N_{\text{unweighted}} = 1\text{k}$. Let the detector simulation take $t_{\text{detector}} = 100$ s per event. The full simulation of one effective event using weighted events would then take

$$\begin{aligned} t_{\text{full, weighted}} &= (t_{\text{weighted}} + t_{\text{detector}}) \cdot \frac{N}{N_{\text{eff}}} \\ &= (10^{-3} \text{ s} + 100 \text{ s}) \cdot \frac{1 \times 10^6}{5 \times 10^3} \\ &= 200.002 \text{ s}. \end{aligned} \quad (2.49)$$

Using unweighted events it would take

$$\begin{aligned} t_{\text{full, unweighted}} &= t_{\text{unweighted}} + t_{\text{detector}} \\ &= 1 \text{ s} + 100 \text{ s} \\ &= 101 \text{ s}, \end{aligned} \quad (2.50)$$

which is almost twice as fast. Regardless of how realistic these figures are, it is clear that the use of unweighted events in appropriate situations can save a lot of resources. In addition, there are other advantages, such as lower memory requirements and easier downstream processing when weights do not have to be taken into account. The latter can even be a major argument in experimental analyses, since unweighted simulated events can be treated in the same way as events reconstructed from measured data.

3 New sampling methods for efficiency improvements

In this chapter, two new methods for efficient PS sampling are presented. The first is the use of normalizing flows (NFs) as expressive, trainable mappings. Basically, the idea is to use them as a replacement for VEGAS in existing multichannel samplers, whereby some of the limitations of VEGAS can be avoided. As a preparation, section 3.1 begins with an introduction to neural networks (NNS) in general and NFs in particular. Afterwards, ref. [9] is reprinted in its original form. In this publication, the idea of NF-based sampling is presented, and its performance is studied on representative examples. It is followed by a discussion and the presentation of some more recent developments.

The second new approach is the use of nested sampling. This idea is introduced in section 3.2. After introducing the nested sampling algorithm and discussing some of its properties, the original article, ref. [10], is presented. In the article, it is shown how to apply the nested sampling algorithm to the sampling and integration of partonic scattering cross-sections. This idea is then applied to gluon scattering processes as physics-motivated examples. A comparison between NFs and nested sampling follows. It is also shown which possibilities there are for combining the two techniques.

3.1 Neural Importance Sampling

We begin this section with a brief introduction to artificial NNS in section 3.1.1. Subsequently, in section 3.1.2, it is shown how to construct NFs and use them as a generative model. In addition, their properties are discussed and some variants are presented. This provides the basis for section 3.1.3, where an article is presented that applies NFs to PS sampling in HEP. The article is followed by a discussion in section 3.1.4. In section 3.1.5, an interesting observation regarding the combination of NFs and phase space cuts is illustrated. Finally, some more recent developments are discussed in section 3.1.6. These include the idea of local multichannel weights that can be learned by a NN, and mixtures of NFs.

3.1.1 Artificial neural networks

Before explaining the structure of an NN, let us begin with its building block, the artificial neuron. An artificial neuron has k inputs and one output. The neuron takes a linear combination of the inputs, with weights w_i , and adds a bias b . Finally, a non-linear activation function φ is applied to the sum. The output of the neuron is therefore given by

$$\varphi\left(\sum_{i=1}^k w_i x_i + b\right). \quad (3.1)$$

A visualization of an artificial neuron is shown in fig. 3.1. The power of NNS comes from combining many of these neurons in an interconnected network, which can be done in many ways. The most popular structure is the fully-connected feedforward NN, or multilayer perceptron. This structure is shown in fig. 3.2. The neurons are arranged in layers, and each neuron in a layer gets inputs from all the neurons in the previous layer. There are one input layer, one output layer, and a number of hidden layers. The number of neurons for the input

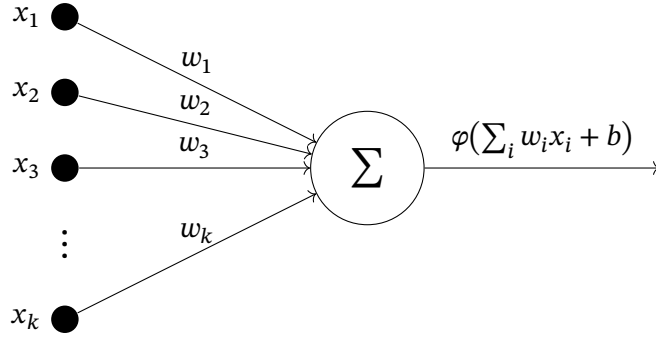


Figure 3.1: An artificial neuron.

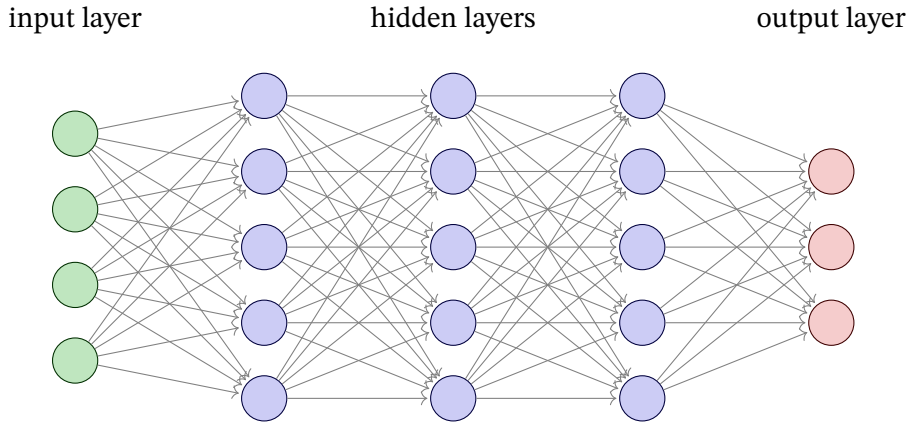


Figure 3.2: A feedforward neural network.

and output layers is defined by the application, while the hidden layers can be of arbitrary size. The information flows from the input layer through the hidden layers to the output layer.

There are many different choices for activation functions and the different layers in an NN need not use the same activation functions. Popular choices are the sigmoid function,

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad (3.2)$$

the hyperbolic tangent

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad (3.3)$$

and the rectified linear unit (ReLU),

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (3.4)$$

Although it is not differentiable at $x = 0$, the ReLU activation function has proven to be particularly useful in many applications, allowing fast and effective training [226].

To apply an NN to a problem, one chooses a loss function that quantifies the quality of the output of the network with respect to some given input. After initializing the weights and biases, the NN can be trained by adjusting their values in a way that optimizes the loss function for some training data. This can be done via stochastic gradient descent, using backpropagation for an efficient determination of the gradients. A popular optimization algorithm for this task is ADAM [227]. Such a training procedure allows NNS to efficiently approximate continuous

functions in a flexible way.

3.1.2 Normalizing flows

NNS are undoubtedly expressive and flexible tools, and can be used to approximate almost arbitrary functions. It is in principle also possible to directly use them as generative models. We can train an NN to learn a transformation from a simple base distribution, e.g. a Gaussian, to a more complex distribution. The model can then be sampled by generating points from the base distribution and transforming them with the NN. For the points generated in this way, we can even calculate the probability by backpropagating through the network. However, there are two major problems with this approach. First, a generic NN is not invertible and therefore the density function is not tractable for arbitrary points from the feature space. This prohibits, among other things, the use in mixture models (i.e. multichannel densities). Second, it is not guaranteed that the model covers the full support of the distribution we want to model, since the NN is not necessarily surjective. This is hard to excuse in phase space sampling, where the part not covered may contain the interesting feature.

An alternative approach that seems more suited to our needs is provided by normalizing flows. Put simply, NFs are generative models based on NNS, where some expressivity is sacrificed in order to obtain some useful guarantees for modelling probability densities. We can think of NFs as a sequence of invertible and differentiable mappings that realize a transformation from a simple distribution to a more complex one. In this regard, they share many similarities with VEGAS and the two can be used interchangeably or in combination, taking into account their respective advantages and disadvantages.

Before moving on to more specific properties and applications, the basics of normalizing flows shall be laid out here. The presentation is partly based on ref. [228]. Let $\mathbf{u} \in \mathbb{R}^d$ be a random vector sampled from a base distribution $p_{\mathbf{u}}(\mathbf{u})$:

$$\mathbf{u} \sim p_{\mathbf{u}}(\mathbf{u}). \quad (3.5)$$

Note that, despite its name, \mathbf{u} is not necessarily a uniform variable. In principle, $p_{\mathbf{u}}(\mathbf{u})$ can be any PDF, but due to their simplicity, the most common choices are uniform and Gaussian distributions. We now construct a transformation T that maps \mathbf{u} to another real random vector \mathbf{x} :

$$\mathbf{x} = T(\mathbf{u}). \quad (3.6)$$

Let us assume that T is a diffeomorphism, i.e. it is bijective and both T and T^{-1} are differentiable. Then we can define the density of \mathbf{x} by the change of variables formula:

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{u}}(\mathbf{u}) |\det J_T(\mathbf{u})|^{-1}, \quad \text{where } \mathbf{u} = T^{-1}(\mathbf{x}). \quad (3.7)$$

The Jacobian, $J_T(\mathbf{u})$, quantifies the change of volume due to the transformation T . Since T is invertible, it is equivalent to consider the inverse direction and write

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{u}}(T^{-1}(\mathbf{x})) |\det J_{T^{-1}}(\mathbf{x})|^{-1}. \quad (3.8)$$

This allows us to change between the two distributions at will. In the forward, or generative, direction we can move samples from a simple base distribution to a more complex one. In the inverse, or normalizing, direction we can ‘normalize’ samples from a complicated distribution by mapping them to the base distribution. The transformation T can be thought of as warping the input space, as is exemplified in fig. 3.3 for a rectangular grid on a square.

A useful property of diffeomorphisms is that they can be composed. If T_1 and T_2 are diffeomorphisms, their composition $T_2 \circ T_1$ is also a diffeomorphism. Its inverse is given by

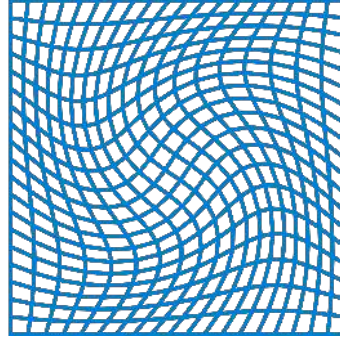




Figure 3.3: Illustration of a diffeomorphism from a square to a square. It is shown how a rectangular grid gets distorted under the diffeomorphism. Figure taken from [229]  .

$$(T_2 \circ T_1)^{-1} = T_1^{-1} \circ T_2^{-1}, \quad (3.9)$$

and the chain rule implies that the Jacobian determinant is given by

$$\det J_{T_2 \circ T_1}(\mathbf{u}) = \det J_{T_2}(T_1(\mathbf{u})) \det J_{T_1}(\mathbf{u}). \quad (3.10)$$

This is used in the construction of NFNs, where multiple simpler transformations are chained together to define one complex transformation $T = T_L \circ \dots \circ T_1$. In this way, highly expressive transformations can be composed by stacking much simpler building blocks. All computations can be done on the individual blocks T_i , and the transformation properties of the composition can be found by eqs. (3.9) and (3.10). In the simplest case, all T_i have the same structure.

The efficiency of NFNs depends on the definition of the transformations T_i . Increasing the number of blocks the flow is composed of only comes with a cost of $\mathcal{O}(L)$. The most demanding part of the calculation comes from the Jacobian determinant. For an arbitrary matrix, using lower-upper decomposition, the computational costs for calculating the determinant grow as the cube of the number of dimensions. For practical applications, it is therefore desirable to find transformations for which the Jacobian has a simpler structure, such that its determinant can be efficiently evaluated. Here we consider coupling layers [38] as one possible solution. Another popular choice is given by masked autoregressive flows [48, 49]. For coupling layers, evaluation and inversion are equally fast, while for masked autoregressive flows one of them is always slower by a factor depending on the number of dimensions.

To consider individual coupling layers T_i , let us introduce another variable \mathbf{z}_i , defined as

$$\mathbf{z}_i = T_i(\mathbf{z}_{i-1}), \quad (3.11)$$

with $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{z}_L = \mathbf{x}$. A coupling layer splits its input \mathbf{z}_{i-1} into two parts and transforms only the second part elementwise using a function that depends on the first part, while the other part is not transformed at all. This can be denoted as follows: The input is split into two partitions A and B that do not need to be in sequence or even of the same size:

$$\mathbf{z}_{i-1} = (\mathbf{z}_{i-1}^A, \mathbf{z}_{i-1}^B). \quad (3.12)$$

The transformation of the coupling layer is then given by

$$\mathbf{z}_i^A = \mathbf{z}_{i-1}^A, \quad (3.13)$$

$$\mathbf{z}_i^B = C_i(\mathbf{z}_{i-1}^B; m_i(\mathbf{z}_{i-1}^A)), \quad (3.14)$$

where C_l is called the coupling transform and m_l is an arbitrary function. The coupling transform is applied elementwise and needs to be invertible. It can be followed from eqs. (3.13) and (3.14) that the Jacobian of the transformation is lower triangular, with an identity block from the derivative of eq. (3.13). The determinant therefore depends only on the derivative of eq. (3.14):

$$\det\left(\frac{\partial \mathbf{z}_l}{\partial \mathbf{z}_{l-1}}\right) = \prod_{i=1}^{n_B} \frac{\partial C_l(\mathbf{z}_{l-1}^B; m_l(\mathbf{z}_{l-1}^A))}{\partial (\mathbf{z}_{l-1}^B)^T}, \quad (3.15)$$

where n_B is the number of elements in B . This implies that the computational costs scale linearly. Furthermore, we see that the Jacobian determinant does not involve derivatives of the function m_l . It is therefore obvious to represent it by an NN. Except for the training, we do not have to calculate its gradient. Its bijectivity renders the training of an NF very convenient, e.g. via maximum likelihood. Depending on what is considered suitable for the application, we can use samples from the target distribution as training data and evaluate the density of the NF, or we use samples from the base distribution and transform them with the NF, which then requires us to evaluate the target function.

Obviously, a single coupling layer is not very expressive, as it transforms only some components of its input. To obtain a useful transformation, several coupling layers have to be chained in a way, such that every component has the chance to influence the transformation of every other component in order to capture all correlations. This can be achieved by permuting the components between the coupling layers. In consequence, the attribution of the components to the parts A and B changes. It can be shown that the number of coupling layers required is

$$\begin{cases} 2\lceil \log_2 d \rceil & \text{for } d > 5, \text{ and} \\ d & \text{for } d \leq 5, \end{cases} \quad (3.16)$$

where d denotes the number of dimensions [230].

There are various implementations of coupling layers that differ in complexity and expressivity. The simplest choices are affine functions [38, 231]. More flexible coupling layers can be constructed with monotonic splines in the form of quadratic splines [232], cubic splines [233], linear-rational splines [234], and rational-quadratic splines [51].

As an example, we come back to the two-dimensional density of section 2.3.2, where it is demonstrated that VEGAS has difficulties adapting to non-factorizable target functions. The example density is shown again in fig. 3.4a. An NF has been trained to find a transformation from a uniform distribution to the target density of the example. The flow features two coupling layers based on rational-quadratic splines, with 16 spline knots in each layer. The splines are parameterized by NNS, but the network architecture is not important here. The ADAM optimizer has been used to train the flow for 500 epochs using batches of size 50000. In fig. 3.4b, the resulting density is shown. Clearly, the NF has been able to learn this simple distribution very well, even though it has two separated peaks that are not present in the base distribution. In contrast to VEGAS (cf. fig. 2.13c), the NF got the correlation right and produced no phantom peaks.

For an interesting deeper insight, the splines of the coupling layers are shown in fig. 3.5. On the left side of the figure, the transformation $u_0 \mapsto x_0$ is shown, which happens in the first coupling layer. For comparison, the marginal distribution of the target is shown as a shade in the background. It can be seen how the splines transform the input variable depending on the value of the untransformed variable, $z_l^A = u_1$. This dependence is what makes NFs so flexible. For VEGAS, on the other hand, we would see only a single curve here, like for the one-dimensional example shown in fig. 2.12. The shape of the spline in the coupling layer, by contrast, can vary greatly depending on the untransformed variable. We see that the NN

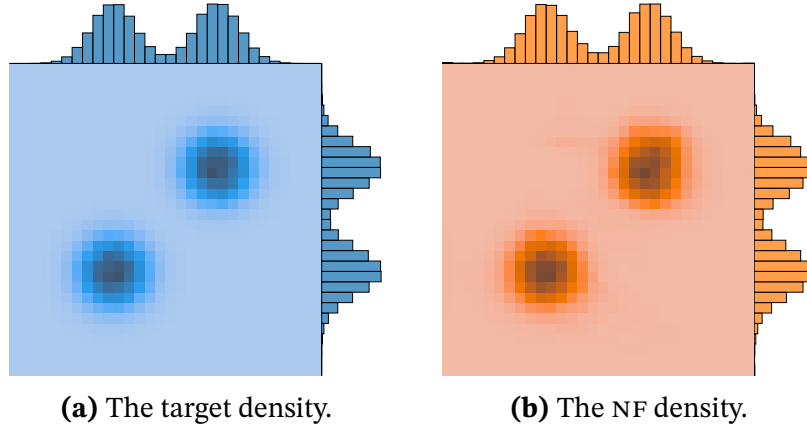


Figure 3.4: Two-dimensional example for density estimation with an NF.

learned to separate the peaks in this way. For $u_1 < 0.5$, probability mass is moved towards the peak at $x_0 = 1/3$, while for $u_1 > 0.5$ it is moved towards the other peak at $x_0 = 2/3$. In between, for $u_1 = 0.5$, the marginal distribution is more or less reproduced. On the right side of the figure, the transformation $u_1 \mapsto x_1$ is shown, which happens in the second coupling layer. Interestingly, the splines look quite different. They are much closer together and for all values of the untransformed variable there are approximately equal contributions to both peaks. The reason for this striking difference is that in the second coupling layer the untransformed variable is $z_1^A = x_0$, which is not uniform. By separating the peaks in the first coupling transform, much of the correlation in the target distribution has been accomplished already. Therefore, the dependence on the untransformed variable is less pronounced in the second coupling transform. The symmetry of the target grants the model a certain freedom. In this case, the splines cross at a point in the centre of the peak at $x_1 = 2/3$, while they run alongside each other at the other peak around $x_1 = 1/3$. This could look different if the training was repeated with new data or a different initialization of the NNS.

3.1.3 *Publication:* Exploring phase space with Neural Importance Sampling

In this section, the article ‘Exploring phase space with Neural Importance Sampling’ is presented. It is the first application of NFs to phase space sampling in HEP. A similar study, ref. [47], discussed in section 3.1.4, appeared shortly after. The idea for this article was motivated by ref. [232], in which the authors use NFs for sampling in light-transport simulation and introduce the piecewise quadratic coupling layers. A lot of similarities between their application and the ones considered in this thesis can be identified, e.g. the usage of random inputs from a hypercube and multichannel mappings. Therefore, the proposed method seems well suited to be transferred to the sampling of cross-sections.

Within the article, the work is embedded in the context of the literature current at the time of publication. Limitations of existing approaches are pointed out and the advantages of the proposed method are explained. The performance is studied for top quark decays, leptonic top quark pair production and gluon scattering to multi-gluon final states. It is shown that significant gains can be achieved. Finally, problems with scalability are discussed and possible further developments are suggested.

The article was first published as a preprint on ARXIV in January 2020. Subsequently it was submitted to the journal *SciPost Physics*. After the referees’ comments were addressed, it was finally published in April 2020. The version published in the journal is reprinted below. Copyright and license notices as well as a link to the material are provided on the first page of the article.

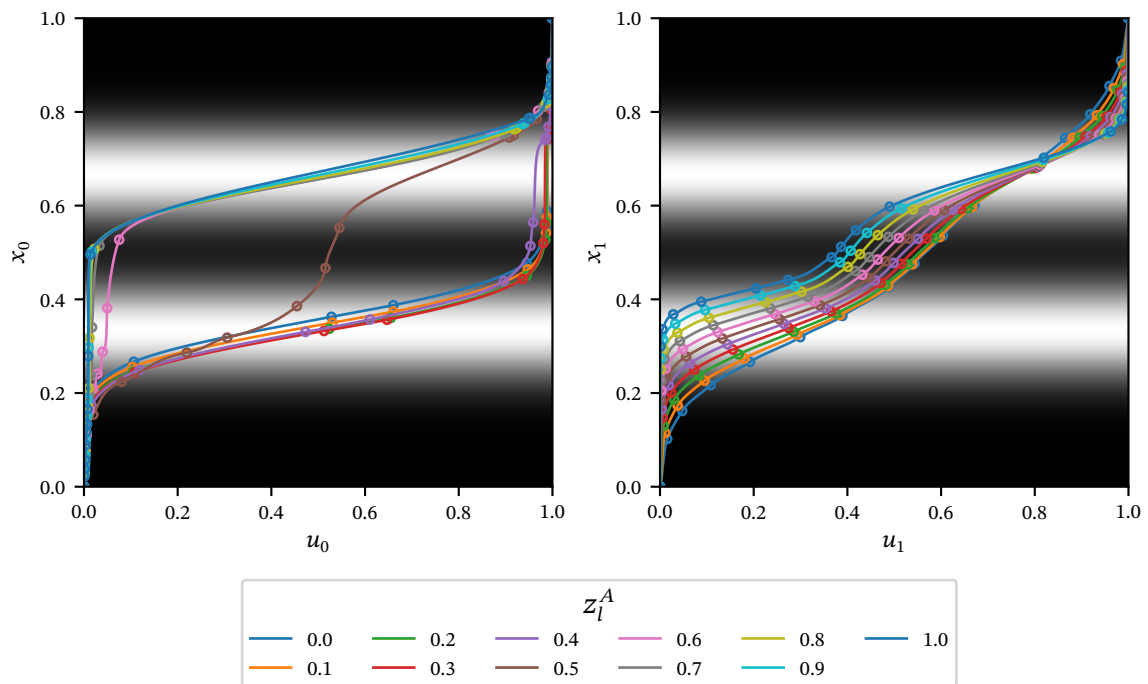


Figure 3.5: The one-dimensional coupling transforms of the NF for the two-dimensional example. Left: first coupling transform, $u_0 \mapsto x_0$. The untransformed variable is u_1 . Right: second coupling transform, $u_1 \mapsto x_1$. The untransformed variable is x_0 , which is the output of the first coupling transform. The legend at the bottom shows the value of the untransformed variable, z_l^A . In the backgrounds of the plots, the marginal distributions of the target are shown as a greyscale gradient, with black corresponding to low values and white corresponding to high values.

Author contributions

While dealing with new sampling methods for applications in HEP as part of my master's thesis [235], the idea for this article arose in the collaboration with Steffen Schumann. Subsequently, I did first studies on sampling with normalizing flows. For my master's thesis, I studied the use of an importance sampling method based on REAL NVP [231] on toy examples and validated the method for the physics process $e^+e^- \rightarrow q\bar{q}g$. Thereupon, we extended the method with the ideas of ref. [232]. Max Knobbe implemented, trained, and evaluated the normalizing flow models based on piecewise quadratic coupling layers, whereas all results not involving NNS were generated by me. While the top quark examples were standalone and implemented by Max Knobbe, I implemented the interface to use matrix elements from SHERPA in PYTHON for the gluon scattering examples. Furthermore, I implemented the HAAG phase space mapping used therein, provided the RIVET analyses, and contributed to the evaluation of the results. Tobias Schmale worked on the PS coverage studies presented in sec. 2.3. All authors contributed significantly to the writing of the article.*

*It is common practice in our community that the names of authors are sorted strictly alphabetically. Accordingly, the order says nothing about the respective contribution to the work.

Exploring phase space with Neural Importance Sampling

Enrico Bothmann, Timo Janßen, Max Knobbe, Tobias Schmale and Steffen Schumann

Institut für Theoretische Physik, Georg-August-Universität Göttingen,
D-37077 Göttingen, Germany

Abstract

We present a novel approach for the integration of scattering cross sections and the generation of partonic event samples in high-energy physics. We propose an importance sampling technique capable of overcoming typical deficiencies of existing approaches by incorporating neural networks. The method guarantees full phase space coverage and the exact reproduction of the desired target distribution, in our case given by the squared transition matrix element. We study the performance of the algorithm for a few representative examples, including top-quark pair production and gluon scattering into three- and four-gluon final states.



Copyright E. Bothmann *et al.*

This work is licensed under the Creative Commons

[Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Published by the SciPost Foundation.

Received 30-01-2020

Accepted 22-04-2020

Published 29-04-2020

doi:[10.21468/SciPostPhys.8.4.069](https://doi.org/10.21468/SciPostPhys.8.4.069)



Check for
updates

Contents

1	Introduction	1
2	Phase space sampling: existing approaches	3
2.1	Importance Sampling	4
2.2	VEGAS algorithm	5
2.3	Existing proposals for neural-network based sampling and possible pitfalls	6
3	Neural-Network assisted Importance Sampling	9
3.1	Coupling Layers	9
3.2	Piecewise Quadratic Coupling Layers	10
3.3	Importance Sampling with Coupling Layers	10
4	Results	11
4.1	Top quarks	12
4.2	Gluon-induced multi-jet production	14
5	Conclusions	18
A	Auxiliary jet p_{\perp} distributions in multi-jet production	19
	References	20

1 Introduction

An important deliverable in high-energy particle physics are quantitative predictions for the outcome of collider experiments. This includes total and differential production rates in the framework of the Standard Model or hypothetical New Physics scenarios. To allow for a direct comparison with experimental data, multi-purpose event generators such as PYTHIA [1], HERWIG [2] or SHERPA [3, 4] proved to be vital tools. Starting from the evaluation of partonic hard-scattering cross sections they accomplish a fully differential and exclusive simulation of individual scattering events by invoking parton shower simulations, particle decays, models for the parton-to-hadron transition and, in case of composite colliding entities such as protons, multiple interactions per collision. See [5] for a recent review of Monte Carlo event generators.

In contemporary Standard Model analyses as well as searches for New Physics, hard-scattering processes featuring a rather high multiplicity of final-state particles are of enormous phenomenological relevance. This includes in particular signatures with multiple hard jets or a number of intermediate resonances that decay further on. Illustrative examples are the production of V + jets final states or top-quark pair production in association with a boson $V = \gamma, H, Z^0, W^\pm$ in proton-proton collisions at the LHC. Such cutting-edge channels require the efficient evaluation of the corresponding partonic scattering matrix elements, featuring up to 8 final-state particles, with easily thousands of Feynman diagrams contributing. This clearly goes well beyond the traditional realm of multi-purpose generators, such that specialised tools for this computationally very intense task have emerged over time, known as matrix element generators or parton-level event generators. These tools largely automate the generation and evaluation of almost arbitrary scattering matrix elements. At tree-level this includes tools such as AMEGIC [6], COMIX [7], MADGRAPH [8, 9] or WHIZARD [10]. For one-loop matrix elements widely-used examples are MADGRAPH5_AMC@NLO [9], OPENLOOPS [11], POWHEGBOX [12] or RECOLA [13, 14]. Equipped with a phase space generator these tools can be used to compile partonic cross section evaluations, to calculate decay widths and to probabilistically generate partonic events. When incorporated into or interfaced to a multi-purpose event generator they provide the momentum-space partonic scattering events that seed the evolution to fully exclusive particle-level final states.

State-of-the-art matrix element generators use adaptive Monte Carlo techniques for generating phase space points with a distribution that reasonably approximates the target distribution, such that event weight fluctuations are reduced. Samples of unit-weight events can then be generated with a distribution given by the actual target function by applying a simple hit-or-miss algorithm. However, nowadays matrix element generators are often limited by the performance of their phase space sampler. An insufficient mapping of the target distribution results in significant fluctuations of the event weights and correspondingly a large number of target-function evaluations are needed when generating unit-weight events.

Typically the sampling performance deteriorates significantly with the phase space dimensionality, i.e. particle multiplicity [15], and the complexity of the integrand. In particular the appearance of intermediate resonances, regularised singularities or quantum-interference effects complicate the situation. Further limitations arise from non-trivial kinematical cuts that the integrator can not address, i.e. adapt to.

Compared to the efforts that went into the development of improved scattering-amplitude construction algorithms, the field of phase space sampling has seen rather little conceptual developments. For some recent works see [16–21]. Besides the matrix element generator implementations, there are public libraries like CUBA [22] (implementing the VEGAS, DIVONNE, SUAVE, CUHRE algorithms) or FOAM [23, 24] that are widely used. There have been some efforts to employ Markov Chain techniques for phase space sampling, cf. [25, 26]. However, very recently there has been significant interest to employ modern machine-learning techniques

to the problem of phase space sampling in particle physics, cf. [27–31]. The tremendous advances in the field of machine learning, driven from very different applications such as image generation or light-transport simulation, also fuel the work we present here.

The paper is organised as follows. In Sec. 2 we review the basics of Monte Carlo integration and phase space sampling techniques as used in high-energy event generators, and discuss potential pitfalls when extending or replacing these methods using neural networks. In Sec. 3 we present our novel sampler that inherits all the properties of an importance sampler, but with the phase space mapping optimised through bijective maps, so-called coupling layers [32]. These are adjusted by training neural networks, which has originally been proposed in [32]. Our work is in principle an application of ‘Neural Importance Sampling’ [33] as we employ the ‘polynomial coupling layers’ introduced therein, although we want to point out that the usage of coupling layers for importance sampling has also been studied in [34]. In Sec. 4 we discuss benchmark applications of our method from high-energy physics, including top-quark pair production and gluon scattering into three- and four-gluon final states. Conclusions and a brief outlook are presented in Sec. 5.

An independent study of applying Neural Importance Sampling to high-dimensional integration problems is simultaneously presented in [35], and a follow-up application of this approach to HEP processes appeared in [36].

2 Phase space sampling: existing approaches

To set the scene we start out with a brief review of the basics of Monte Carlo integration and event sampling. For ease of having a clean nomenclature we consider a simple positive-definite target distribution $f : \Omega \subset \mathbb{R}^d \rightarrow [0, \infty)$ defined over the unit hypercube, i.e. $\Omega = [0, 1]^d$. In our use case hypercube points $u_i \in \Omega$ are mapped onto a set of final-state four-momenta $\{p_i\}$, the corresponding Jacobian is considered part of the integrand $f(u_i)$. The phase space dimensionality d is set by the number of final-state particles n , i.e. $d = 3n - 4$. We thereby implement on-shell constraints for all external particles and total four-momentum conservation. There are two standard tasks that we wish to address in what follows, the probabilistic generation of phase space points according to the target distribution f and the evaluation of integrals over f .

The Monte Carlo estimate of the integral over the unit hypercube

$$I = \int_{\Omega} f(u') du' \quad (1)$$

is given by

$$I \approx E_N = \frac{1}{N} \sum_{i=1}^N f(u_i) = \langle f \rangle, \quad (2)$$

where we assumed uniformly distributed random variables $u_i \in \Omega$. The corresponding standard deviation, when assuming large N , is given by

$$\sigma_N(f) = \sqrt{\frac{V_N(f)}{N}} = \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}, \quad (3)$$

with V_N the corresponding variance.

Interpreting the random points u_i as individual *events*, we call $f(u_i)$ the corresponding *event weight* w_i , such that the integral is estimated by the average event weight $\langle w \rangle_N$. When asked to generate N unit-weight events according to the distribution $f(u)$, a simple hit-or-miss

algorithm can be employed to convert a sample of weighted events into a set of unweighted events. The corresponding unweighting efficiency is given by

$$\epsilon_{\text{uw}} := \frac{\langle w \rangle_N}{w_{\text{max}}}, \quad (4)$$

with w_{max} the (numerically pre-determined) maximal event weight in the integration region. An efficient integrator, i.e. sampler, foremost aims for a reduction of the variance V_N that will typically result in an increased unweighting efficiency ϵ_{uw} . Even though these two figures of merit are interrelated, they provide complementary means for the optimisation of a sampler, i.e. reducing the variance does not necessarily yield an improved unweighting efficiency. In the next section we will discuss established methods that achieve a variance reduction.

We close this introductory section by specifying requirements we impose on our improved cross section integration and parton-level event generation algorithm:

- (i) The samples produced by the algorithm should converge to the true target distribution everywhere in phase space.¹
- (ii) We demand that the full physical phase space is to be covered for the limit $N \rightarrow \infty$. This should be guaranteed, even if potential training samples only feature finite statistics and thus provide no full coverage of the available phase space volume.
- (iii) The method should be general, lending itself to automation. By that we wish the algorithm to be self-adaptive to new integrands, without the need of manual intervention.
- (iv) The method should be capable of producing samples of uncorrelated events.²

As discussed in the following, these conditions are naturally fulfilled by traditional sampling algorithms used in high-energy physics, such as importance and stratified sampling. However, this is not necessarily true for some of the recently proposed samplers based on neural networks as discussed in Sec. 2.3. In Sec. 3 we will present our novel algorithm employing neural-network techniques, that indeed fulfils all the above criteria.

2.1 Importance Sampling

As can be seen from Eq. (3) the standard deviation of a Monte Carlo integral estimate scales as $1/\sqrt{N}$, independent of the dimensionality of the problem. However, besides the sample size, the variance of the integrand over the integration region determines the quality of the integral estimate and in turn the unweighting efficiency ϵ_{uw} . In particular for strongly structured, possibly multi-modal target distributions it is therefore vital to introduce specific variance-reduction techniques to obtain more accurate integral estimates for a given sample size.

To this end a suitable variable transformation can be utilised, i.e. producing phase space points with a positive definite non-uniform distribution function $G(u) : \Omega \mapsto \Omega$, such that

$$I = \int_{\Omega} \frac{f(u')}{g(u')} g(u') du' = \int_{\Omega} \frac{f(u')}{g(u')} dG(u') = \left\langle \frac{f}{g} \right\rangle, \quad (5)$$

with $g(u) : \Omega \mapsto \mathbb{R}$. The relevant variance is thus $V(f/g)$. Hence it can be significantly reduced by picking $g(u)$ similar in shape to $f(u)$. Obviously, the optimal choice would be

$$G(u) = \int_{\Omega} f(u') du', \quad \text{i.e.} \quad g(u) = f(u). \quad (6)$$

¹On the same basis, in [37] it has been cautioned against the usage of Generative Adversarial Networks to extrapolate from finite-statistics training data to large-scale event samples for physics analyses.

²This limits the use of algorithms based on Markov Chain samplers, cf. [26].

However, this presupposes the solution of the actual integration problem.

We often have to deal with multimodal targets. In that case it can be very hard to find a density that allows for efficient importance sampling. To simplify the task, we can use a mixture distribution

$$g(x) = \sum_{j=1}^{N_c} \alpha_j g_j(x), \quad (7)$$

where the g_j are distributions and $\sum_{j=1}^{N_c} \alpha_j = 1$. Using a mixture distribution for importance sampling is known as *multi-channel importance sampling*. The corresponding integral estimate is given by

$$I \approx E_N = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{g(x_i)} = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{\sum_{j=1}^{N_c} \alpha_j g_j(x_i)}, \quad (8)$$

where the x_i are non-uniform random numbers drawn from $g(x)$.

It is easy to sample from the multi-channel distribution: for each point one channel is chosen at random according to the α_j and then sampled from using the inverse-transform method. It is possible to approximate a multimodal target function by using one channel per peak. The channel weights α_j can be optimised automatically [38].

The performance of the multi-channel method is intimately connected with the choice of channels. In practice, information about the physics problem at hand is used to choose a suitable distribution g . When integrating squared transition matrix elements in high-energy physics, the propagator and spin structures of a given process are known and this knowledge can be used to construct appropriate channels [39], a procedure that is fully automated in matrix-element generators.

2.2 VEGAS algorithm

It can be very time consuming to find a sampling distribution that results in an efficient sampler for a given target. Because of this, adaptive importance sampling algorithms have been developed. These are able to adapt automatically to a target distribution. In the following we describe the VEGAS algorithm [40]. It uses a product density

$$q(x) = \prod_{j=1}^d q_j(x_j), \quad (9)$$

where each q_j is a piecewise-constant function. The idea is to split the range $[0, 1)$ into N_j bins $I_{j,l} = [x_{j,l-1}, x_{j,l})$, where we have defined the break points between the constant pieces as $0 = x_{j,0} < x_{j,1} < \dots < x_{j,N_j} = 1$. The corresponding bin widths are $\Delta_{j,l} = x_{j,l} - x_{j,l-1}$ for $1 \leq l \leq N_j$. The functions q_j are then defined by

$$q_j(x) = \frac{1}{N_j \Delta_{j,l}} \quad \text{for} \quad x_{j,l-1} \leq x \leq x_{j,l}. \quad (10)$$

The width of the bins can vary but per component j they all have the same probability content $1/N_j$. This means that if we approximate a function with VEGAS we use many thin bins for narrow peaks and few wide bins for flat regions.

Sampling and evaluating the Jacobian for the density q is straightforward. The important part is the update of the bin widths. This happens through an iterative procedure, where in each iteration we sample a number of points with the current q , calculate the importance weights with respect to the target f and determine the new bin widths by minimising the variance for this sample. More details can be found in [40].

VEGAS is very effective for unimodal targets but has difficulties with multimodal functions if the peaks are not aligned with the coordinate axes. The density then features ‘ghost peaks’ which are not present in the target distribution and which can decrease the efficiency significantly.

In the simplest case, we use VEGAS to approximate a target directly in order to use the resulting density in an importance sampling scheme. However, it can be even more effective if we use it to remap the input variables (i.e. uniform random numbers) of another density, e.g. a single channel of a multi-channel distribution [41]. This amounts to adding variable transforms $\phi_j : \Omega \rightarrow \Omega$ to Eq. (7) (corresponding to the VEGAS densities q_j) as follows:

$$g = \sum_{j=1}^{N_c} \alpha_j (g_j \circ \phi_j^{-1}) \left| \frac{\partial \phi_j^{-1}}{\partial u} \right|, \quad (11)$$

with densities $g_j : \Omega \rightarrow [0, \infty)$ and hence $g : \Omega \rightarrow [0, \infty)$. As above we assume that the α_j sum to 1. To sample a point from this distribution we

1. randomly choose a channel according to the channel weights α_j ,
2. generate a uniform random number $u \in \Omega$,
3. use the channel-specific map ϕ_j to map u to a non-uniform number v and
4. use the inverse transform method to transform v to a point x according to the distribution g_j .

The Monte Carlo estimate of the integral is then still given by Eq. (8) but the Jacobians $\left| \frac{\partial \phi_j^{-1}}{\partial u} \right|$ of the different channels have to be taken into account.

2.3 Existing proposals for neural-network based sampling and possible pitfalls

A multi-layer feedforward fully connected artificial neural network (NN in the following) consists of artificial neurons arranged in several layers which are stacked on top of each other. Every neuron in a layer is connected to every neuron in the preceding layer. We distinguish the input layer, the output layer and the hidden layers in between. A single artificial neuron produces the weighted sum of its inputs and optionally adds a scalar bias. The output of the neuron is then transformed by a non-linear activation function. This means that the output $z_i^{[l]}$ of the i -th neuron in the l -th layer is given by

$$z_i^{[l]} = \vec{w}_i^{[l]T} \cdot \vec{a}^{[l-1]} + b_i^{[l]}, \quad (12)$$

where $\vec{w}_i^{[l]}$ denotes the vector of input weights of the neuron, $b_i^{[l]}$ the respective bias and $\vec{a}^{[l-1]}$ the output of the activation function of the preceding layer. After applying the activation function $\sigma^{[l]}$ the output is given by

$$a_i^{[l]} = \sigma^{[l]}(z_i^{[l]}). \quad (13)$$

We assume that all hidden layers use the same activation function. The choice of output activation function is limited by the particular application. In our case it has to be a function that maps to the unit hypercube. The input layer does not use an activation function as it only passes the input variables to the neurons of the first hidden layer.

Two previous studies use this kind of NN to improve phase space sampling [27, 28]. The number of input and output neurons is there chosen equal to the number of phase space

dimensions d . Hence the neural network gives effectively an importance sampling mapping g in the language of Sec. 2.1. The output value of the i th output layer neuron then gives the i th component of $g(u)$. Both studies described in [27, 28] use output functions that map \mathbb{R} into $(0, 1)$, such that $g(u) \in (0, 1)^d$. As the hidden-layer activation function, either the hyperbolic sine, the exponential linear unit (ELU) [28] or a hyperbolic tangent [27] is used. Note that ELUs map \mathbb{R} into $(-1, \infty)$ and \tanh maps \mathbb{R} into $(-1, 1)$, whereas \sinh maps \mathbb{R} into itself. The input space is \mathbb{R}^d (sampled from a Gaussian distribution) in [27], and $(0, 1)^d$ in [28]. Both studies then set up different training procedures for the NN based on minimising the Kullback–Leibler (KL) divergence [42] between the NN output and the real target distribution. The details of these procedures are not important here.

What we want to point out in regards of the requirements set up in Sec. 2 is that restricting the input space to a subspace of \mathbb{R} with an upper and/or lower bound will in general have the consequence that the NN map g is not guaranteed to be surjective any more. The same is true if an activation function of the hidden layer maps onto a subspace of \mathbb{R} with an upper and/or lower bound, such as the ELU or the \tanh function. In both cases, such finite boundaries will be transformed several times, but in the end this will yield finite boundaries for the target-space coordinates, such that the support of the target distribution will be a proper subspace of the desired target space $(0, 1)^d$. Although a sufficiently long training will guarantee that the bulk of the target distribution will be within this subspace (the NN will adapt its weights to extend this subspace as required), the phase space coverage might never reach 100%. A sample generated with such a NN will hence suffer from artificial phase space boundaries far away from the peaks of the distribution and will thus not be distributed according to the desired target distribution. Instead, it will be suppressed in the tails and enhanced in the peaks. Moreover, the artificial phase space boundaries will also yield wrong integration results. The NN structure in [28] is affected by this problem, whereas the structure in [27] is not, since it uses surjective functions throughout and the input points are given by a Gaussian distribution without a cut-off, such that the input space is given by \mathbb{R}^d .

To illustrate this issue, we study a simple distribution given by a 2d Gaussian centred in $(0, 1)^2$, i.e. at $(x, y) = (0.5, 0.5)$. The width of the Gaussian is set to 1/10 of the length of the phase space edges, hence, close to the phase space boundaries the target-function values are much smaller than around the peak. We test different combinations of activation and input functions for a fully-connected NN architecture with 5 hidden layers and 64 nodes per hidden layer, always with a bounded input space of $(0, 1)^2$, as in [28]. We train the networks using the ADAM optimiser [43] with the learning rate set to 10^{-2} . With a training data set of $N_{\text{train}} = 500\text{k}$ events, this setup yields a very poor phase space coverage for the NN regardless of the activation/output functions, namely around 25 % only:

Input space	Activation function	Output function	Coverage (asymptote)
$(0, 1)^2$	Sinh	Sigmoid	0.235 ± 0.027
$(0, 1)^2$	ELU	Soft Clipping	0.269 ± 0.037
$(0, 1)^2$	Sigmoid	Sigmoid	0.234 ± 0.050

The coverage is estimated by the convex hull of the respective event samples, as introduced in [44]. Note that the first two rows follow the two choices discussed in [28]. The error of the asymptotic coverage is given by an average over 10 independently trained NN with different random initial weights. In Fig. 1a, we show the obtained phase space coverage as a function of the sample size N for such a NN with sigmoid activation and output functions. This is compared to unweighted event samples generated from a uniform distribution and through VEGAS. In addition, the phase space coverage is also shown for a NN with surjective functions only, and with input points given by an unbounded Gaussian distribution, as in [27]. This surjective NN is guaranteed to sample the entire phase space and indeed its coverage increases

with the sample size N in the same way as the uniform and VEGAS samples do, whereas the non-surjective NN shows an asymptotic behaviour in terms of the coverage. We illustrate this with three non-surjective NN setups, which are trained using $N_{\text{train}} = 250\text{k}$, 1M and 5M events, respectively. We choose these N_{train} to have similar successive ratios between them, to illustrate that increasing the size of the training data successively leads to a diminishing return of investment in terms of the achieved asymptotic phase space coverage.

We study the consequence of an incomplete phase space coverage in Fig. 1b. The target distribution is averaged over bins with $x + y = \text{const.}$ (resulting in one-dimensional Gaussians) and compared with the distributions averaged in the same way given by the NN (here with $N_{\text{train}} = 500\text{k}$), by an unweighted uniform sample, by an unweighted VEGAS sample and by the averaged distribution given by the strictly surjective NN. The uniform and VEGAS samples and the one from the strictly surjective NN agree very well with the target, whereas the non-surjective NN undershoots the tails and puts too many events in the peak. We have also studied the distribution of phase space points in the two-dimensional plane, where we find that the NN is mapping the input space $(0, 1)^2$ to a slightly deformed rectangular region around the peak, which is strictly smaller than the target space.

3 Neural-Network assisted Importance Sampling

With the requirements stated in Sec. 2 in mind, we present our NN based approach to importance sampling. In order to be usable for multi-channel sampling, our adaptive model needs to be invertible. For this reason, we adopt the ‘‘Neural Importance Sampling’’ algorithm of [33]. The method of using a trainable mapping to redistribute the random numbers going into the generation of a sample is similar to how VEGAS is often used in practice. We begin this section by discussing this remapping of a distribution.

Consider a mapping $h : X \rightarrow Y, x \mapsto y$, where x is distributed as $p_X(x)$. If we know $p_X(x)$ and the Jacobian determinant of $h(x)$, we can compute the PDF of y using the change of variable formula:

$$p_Y(y) = p_X(x) \left| \det \left(\frac{\partial h(x)}{\partial x^T} \right) \right|^{-1}. \quad (14)$$

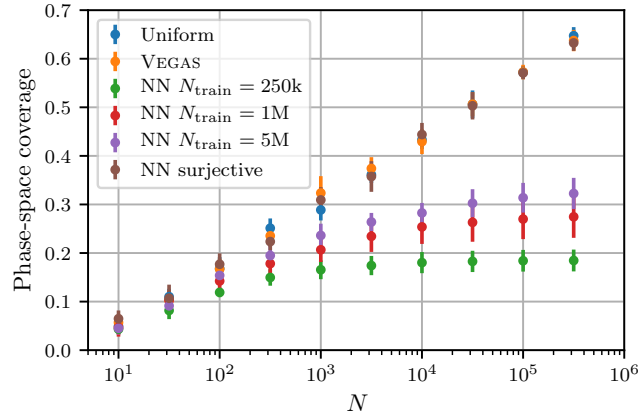
Using lower-upper decomposition, the cost of computing the determinant for arbitrary matrices grows as the cube of the number of dimensions and can therefore be obstructive. However, it is possible to design mappings for which the computation of the Jacobian determinant is cheap.

In [32], Dinh et al. introduce *coupling layers* which have a triangular Jacobian. As the determinant of a triangular matrix is given by the product of its diagonal terms, the computation scales linearly with the number of dimensions only. In the following, we describe the basic idea of coupling layers.

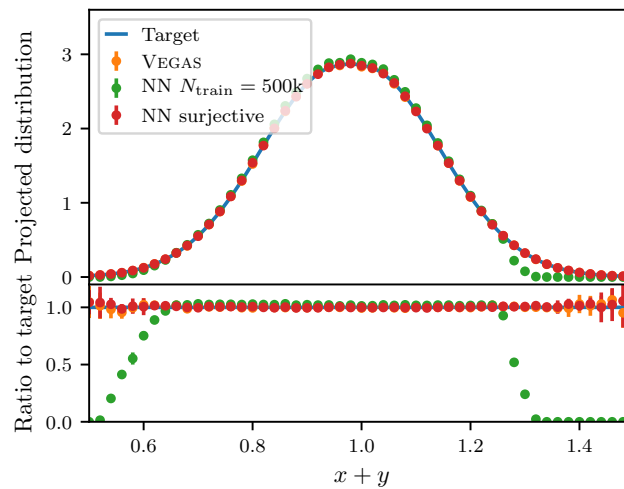
3.1 Coupling Layers

A coupling layer takes a d -dimensional input $x \in \mathbb{R}^d$. It uses a partition $\{A, B\}$ of the input dimensions x_i such that $x = (x^A, x^B)$. The output $y = (y^A, y^B)$ of the coupling layer is defined as

$$\begin{aligned} y^A &= x^A, \\ y^B &= C(x^B; m(x^A)), \end{aligned} \quad (15)$$



(a) The phase space coverage for different sampling methods as a function of the number of unweighted events N . The error bars indicate the spread over 10 statistically independent samples. For the NN, also the training has been independently repeated, each time with a new set of randomly initialised weights. For the non-surjective NN setup, the effect of using different number of training points N_{train} is illustrated.



(b) The two-dimensional distribution of sampling points averaged over bins with $x + y = \text{const.}$, compared to the corresponding average of the target distribution.

Figure 1: The phase space coverage and the distribution averaged over diagonals of the two-dimensional plane for different sampling techniques, with the target distribution being a two-dimensional Gaussian centred in $(0, 1)^2$. The Gaussian width is $\sigma = 0.1$. Besides the Uniform and the VEGAS samples we also show NN-generated samples. The NN architecture is described in the main text, it uses sigmoids as activation and output functions, and $N_{\text{train}} = 500\text{k}$. The input space is given by $(0, 1)^2$. The “surjective” NN on the other hand only uses surjective functions and the input space is unbounded.

where the *coupling transform* C is a map that is invertible and separable, where the latter means that

$$C(x^B; m(x^A)) = \left(C_1(x_1^B; m(x^A)), \dots, C_{|B|}(x_{|B|}^B; m(x^A)) \right)^T. \quad (16)$$

By $|B|$ we denote the cardinality of the set B .

According to Eq. (15) only the subset B is transformed by the coupling layer, while the subset A is left unchanged. Because of this, $\partial y^A / \partial (x^B)^T = 0$ and the Jacobian determinant simplifies to

$$\det \left(\frac{\partial y(x)}{\partial x^T} \right) = \prod_{i=1}^{|B|} \frac{\partial C_i(x^B; m(x^A))}{\partial (x^B)^T}. \quad (17)$$

To see this, we assume that without loss of generality we split the input dimensions in two consecutive blocks $A = [1, n]$ and $B = [n + 1, d]$. In this case, the Jacobian matrix is of block form

$$\frac{\partial y(x)}{\partial x^T} = \begin{pmatrix} I_n & 0 \\ \frac{\partial C(x^B; m(x^A))}{\partial (x^A)^T} & \frac{\partial C(x^B; m(x^A))}{\partial (x^B)^T} \end{pmatrix}, \quad (18)$$

with the determinant given by Eq. (17). As the determinant does not involve the derivative $\frac{\partial m(x^A)}{\partial x^A}$, the function m can be arbitrarily complex. Following [33], we represent m through a NN.

A single coupling layer transforms only part of the input. To ensure that all components can be transformed, we use a layered mapping $h = h_L \circ \dots \circ h_2 \circ h_1$, where each h_i is a coupling layer. Between two layers, we exchange the roles of A and B . For the functions m_i , we use one NN per layer. If $d > 3$, we need at least 3 coupling layers if we want each input component to be able to influence every output component.

Different choices for the coupling transform C are possible. Additive coupling layers result in a *NICE* model [32], while affine coupling layers result in a *Real NVP* model [45]. In the following, we restrict ourselves to piecewise quadratic coupling layers, which have been proposed in [33].

3.2 Piecewise Quadratic Coupling Layers

We assume that our variables live in a unit hypercube: $x, y \in \Omega = [0, 1]^d$. This allows us to interpret each component C_i of the coupling transform as a cumulative distribution function (CDF) in a straightforward manner. The idea is to use the output of a NN to construct unnormalised distributions \hat{q}_i and get C_i by integration. We normalise the distributions to get the PDF q_i and model them with piecewise linear functions which have K bins and $K + 1$ vertices (bin edges) each. The parameters of these functions can be stored in two matrices: The $|B| \times (K + 1)$ matrix V contains the height (vertical coordinate) of the functions at each vertex and the $|B| \times K$ matrix W contains the bin widths (which are adaptive).

A NN outputs the unnormalised matrices \hat{V} and \hat{W} . The bin widths should sum to 1, so we normalise the rows of the matrix \hat{W} using the softmax function σ and define

$$W_i = \sigma(\hat{W}_i). \quad (19)$$

We want the piecewise linear function q_i to be a PDF, and therefore normalise the rows of \hat{V} according to:

$$V_{i,j} = \frac{\exp(\hat{V}_{i,j})}{\sum_{k=1}^K \frac{1}{2} (\exp(\hat{V}_{i,k}) + \exp(\hat{V}_{i,k+1})) W_{i,k}}. \quad (20)$$

Finally, we use linear interpolation to define our PDFs as

$$q_i(x_i^B) = V_{i,b} + \alpha(V_{i,b+1} - V_{i,b}), \quad (21)$$

where b is the bin that contains the point x_i^B and $\alpha = (x_i^B - \sum_{k=1}^{b-1} W_{i,k})/W_{i,b}$ is the relative position within that bin. By integration we get the piecewise quadratic coupling transform:

$$C_i(x_i^B) = \int_0^{x_i^B} q_i(t)dt = \frac{\alpha^2}{2}(V_{i,b+1} - V_{i,b})W_{i,b} + \alpha V_{i,b}W_{i,b} + \sum_{k=1}^{b-1} \frac{V_{i,k} + V_{i,k+1}}{2} W_{i,k}. \quad (22)$$

The corresponding Jacobian determinant is given by

$$\det\left(\frac{\partial C(x^B; m(x^A))}{\partial (x^B)^T}\right) = \prod_{i=1}^{|B|} q_i(x_i^B). \quad (23)$$

3.3 Importance Sampling with Coupling Layers

Having defined the coupling layers, their application for importance sampling is straightforward as they can be used in the same way VEGAS is already applied in existing event generators. The algorithm for a single phase space map, i.e. channel, proceeds as follows.

For each event, we generate a suitable number of uniformly distributed random numbers $x \in \Omega$. These get mapped to non-uniform numbers $y \in \Omega$ using a layered mapping consisting of several coupling layers, as described above. These numbers then serve as input variables for a channel mapping that generates a point z in the target domain. The weight w associated with an event depends on the value of the target function and the Jacobians involved, namely the ones from the coupling layers and the channel mapping itself:

$$w = \left| \det\left(\frac{\partial y(x)}{\partial x^T}\right) \right| \left| \det\left(\frac{\partial z(y)}{\partial y^T}\right) \right| f(z). \quad (24)$$

Note that we do not use the NN model to generate points in the target domain directly as this could be highly inefficient. For example, if we wanted to generate four-momenta the NN would have to learn four-momentum conservation and on-shell conditions exactly. Using a channel mapping we can implement four-momentum conservation and mass shell conditions directly, lowering the dimensionality of the problem significantly, and also map out known peak structures that might be difficult to infer otherwise.

As for VEGAS we need a mechanism to train our model in order to actually improve the efficiency of the sampler. For this purpose, we define a loss function which gets minimised iteratively using gradient descent. As a loss function we use the Pearson χ^2 -divergence between the target function f and the sample distribution g in a minibatch that consists of n sampling points:

$$D_{\chi^2} = \frac{1}{n} \sum_{i=1}^n \frac{(f(z_i) - g(z_i))^2}{g(z_i)}, \quad (25)$$

with points z_i in the target domain, generated from a uniform distribution and transformed by a channel mapping.

Minimising D_{χ^2} will minimise the variance of a Monte Carlo estimator, as recognised in [33]. Empirically we find that for our applications the mean squared error distance performs better in terms of variance reduction and unweighting-efficiency increase than the Kullback–Leibler divergence.

Our method can be used in a multi-channel approach in the same way as described for VEGAS in Sec. 2.2. It has the additional advantage that we are able to train all mappings for the different channels simultaneously. The channel mappings are aware of each other and do not try to adapt to the same features.

4 Results

We have implemented the NN architecture described in the previous section using TENSORFLOW [46]. The NN training is guided using the ADAM optimiser [43]. The default learning rate we use is 10^{-4} , and gradients calculated for the training are clipped at a value of 100 to avoid instabilities in the training.

We apply our NN-assisted sampling to three standard applications in high-energy physics: the three-body decay of a top quark which features a single importance sampling channel modelling the Breit–Wigner distribution of the intermediate W boson; top-quark pair production in e^+e^- annihilation with the subsequent decay of both top quarks (which also can be modelled by a single importance sampling channel by using the same mapping for both decays); and finally QCD multi-gluon production, with two gluons in the initial state colliding at a fixed centre-of-mass energy and 3 or 4 final-state gluons. For the latter, a multi-channel algorithm as described in the previous section is used, with one NN per independent channel.³ The required multi-gluon tree-level matrix elements are obtained from SHERPA through its dedicated PYTHON interface [47].

In all cases we compare the performance of the novel NN-assisted importance sampling algorithm with the VEGAS-assisted one which serves as benchmark. We checked that the performance of the VEGAS grids used were not limited by the number of bins or by the number of optimisation steps used.

4.1 Top quarks

Top quarks decay predominantly to a W boson and a bottom quark. In turn, the W decays either leptonically or hadronically. This induces an s -channel resonance for the W propagator, which is usually described in a phase space sampler by a strongly-peaked Breit–Wigner channel, i.e.

$$g(u) = \frac{1}{(s(u) - M_W^2)^2 + M_W^2 \Gamma_W^2}, \quad \text{with } s(u) = M_W \Gamma_W \tan(u) + M_W^2. \quad (26)$$

This channel captures the behaviour of the denominator of the corresponding squared matrix element, but assumes a constant numerator, which renders the channel imperfect for the actual integrand. In the following we study for single top-quark decays and top-quark pair production with subsequent decays how our NN optimisation compares with VEGAS optimisation to remedy such imperfections.

The NN architecture for both top-quark examples consists of 6 piecewise-quadratic coupling layers and 150 bins. The trainings conclude after 6000 optimisation steps, where each step uses a minibatch of 200 phase space points to guide the optimisation.

Top-quark decays: We simulate the decay sequence of a top quark, i.e. $t \rightarrow W^+b \rightarrow e^+\nu_e b$. With three on-shell final-state particles we have 5 dimensions for the kinematics (the top quark is considered at rest and on-shell). However, we integrate out all dimensions except for the invariant mass of the W-boson decay products and the angle between them. The number of phase space dimensions is therefore $d = 2$. The s -channel propagator of the W boson is modelled by a Breit–Wigner distribution in the importance sampling, reducing the variance caused by sampling the strongly-peaked invariant-mass distribution of the lepton-neutrino pair.

The results of a run with $N = 10^6$ events are compared in Tab. 1 with an unoptimised (“Uniform”) sampling and a VEGAS-optimised sampling. The Monte Carlo integration result,

³Here and in the following, “one NN” refers to a connected set of coupling layers, not to the “sub-NN” used within each single coupling layer.

Table 1: Results for sampling the partial top-quark decay width and the total cross section of top-pair production, i.e. the Monte Carlo integral E_N is an estimator for $\Gamma_{t \rightarrow b e^+ \nu_e}$ and σ for $e^+ e^- \rightarrow \gamma \rightarrow t[\bar{b} e^+ \nu_e] \bar{t}[b e^- \bar{\nu}_e]$ at $\sqrt{s} = 500$ GeV, respectively. Besides E_N and its MC error, we also show the unweighting efficiency ϵ_{uw} of the sample, comparing VEGAS optimisation, NN-based optimisation and an unoptimised (“Uniform”) distribution. All samples consist of $N = 10^6$ (weighted) points.

Sample	top decays		top-pair production	
	ϵ_{uw}	E_N [GeV]	ϵ_{uw}	E_N [fb]
Uniform	59 %	0.1679(2)	35 %	1.5254(8)
VEGAS	50 %	0.16782(4)	40 %	1.5251(1)
NN	84 %	0.167865(5)	78 %	1.52531(2)

i.e. the partial decay width E_N given by the estimator for $\Gamma_{t \rightarrow b e^+ \nu_e}$, is given as a consistency check and to compare its statistical deviation when generating the same number of points N with the alternative sampling methods. The standard deviation obtained with VEGAS is 5 times smaller than for the Uniform sample. Improving on that, the NN sampling has a standard deviation which is 8 times smaller than the VEGAS one. As another figure of merit the unweighting efficiencies ϵ_{uw} are compared. Again, the NN has the best (i.e. largest) efficiency, with a value of 0.84 compared to 0.50 for the VEGAS and 0.59 for the Uniform sampling. So while for VEGAS the integral variance is indeed reduced, the unweighting efficiency is somewhat reduced in comparison to the Uniform sampling. This originates from rare outliers in the event weight distribution.

This is illustrated in Fig. 2a, where the distributions of event weights, cf. Eq. (24), for the three samples are shown. The optimal sampler would result in events with identical weights, what leads to a vanishing variance of the integral estimate and an unweighting efficiency of one, cf. Eq. (4). The NN sample features the sharpest peak here and a steeply falling tail towards larger weights, which corresponds to the significantly improved unweighting efficiency. Although the VEGAS sample is also more peaked than the Uniform one, it features large-weight outliers causing the reduced unweighting efficiency.

Leptonic top-quark pair production: As a second application, we study the leptonic production of a top–anti-top pair via a virtual photon, and their subsequent leptonic decay, i.e. $e^+ e^- \rightarrow \gamma \rightarrow t[\bar{b} e^+ \nu_e] \bar{t}[b e^- \bar{\nu}_e]$, at $\sqrt{s} = 500$ GeV. This gives us effectively two copies of the top-quark decay chain considered in the previous example, plus the scattering angle between the incoming lepton and the outgoing top quark. This yields a phase space dimensionality of $d = 5$.

Both s -channel propagators of the W bosons are modelled by Breit–Wigner distributions, using a single importance sampling channel. Again, a NN sample with $N = 10^6$ points is generated. It is compared in Tab. 1 with an unoptimised and a VEGAS sample of same sizes. The standard deviation of the VEGAS sample is 8 times smaller than the one of the unoptimised sample. The NN sample has the smallest standard deviation, being yet 5 times smaller than the one of the VEGAS sample. The unoptimised and the VEGAS sample have a similar unweighting efficiency of 35 % and 40 %, respectively. The NN one’s is about two times better, at 78 %.

Figure 2b depicts the event weight distributions of the three samples. As for the top-decay samples, the NN-optimised sample for top–anti-top production is most strongly peaked, which is in accordance with the small standard deviation and the good unweighting efficiency. The other two samples are significantly broader and have long tails towards large weights.

Overall, the results for top decays and top–anti-top production are similar, which is ex-

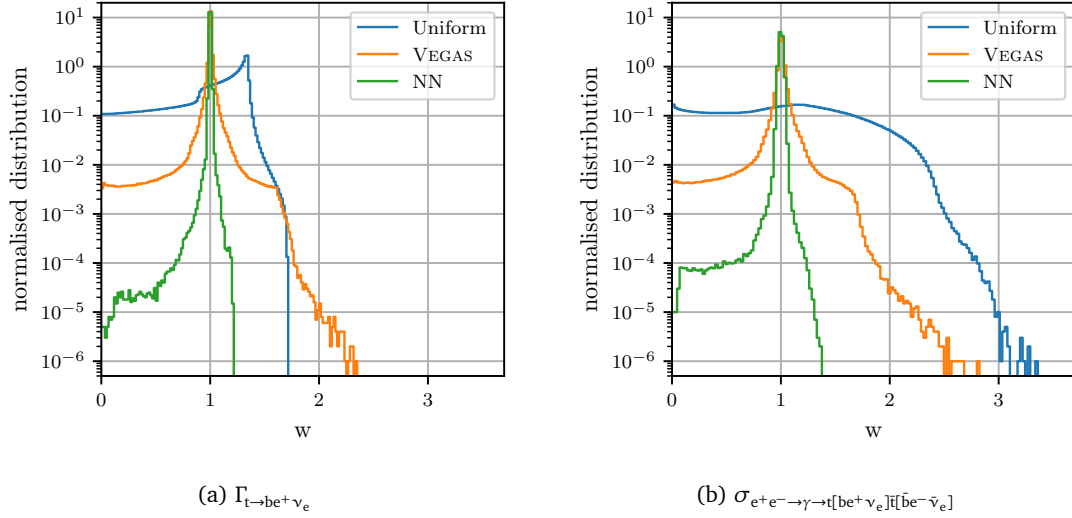


Figure 2: Event weight distributions for sampling the partial decay width $\Gamma_{t \rightarrow be^+ \nu_e}$ and the total cross section σ for $e^+e^- \rightarrow \gamma \rightarrow t[be^+ \nu_e] \bar{t}[\bar{b}e^- \bar{\nu}_e]$ at $\sqrt{s} = 500$ GeV, each with $N = 10^6$ points, comparing VEGAS optimisation, NN-based optimisation and an unoptimised (“Uniform”) distribution.

pected because the main difference is that the Breit–Wigner peak appears in one additional dimension for the top–anti-top production, with all other dimensions in phase space not featuring any (strongly) peaked structures. Hence we see a similar shape in the weight distributions, only the unoptimised sample is significantly broader now due to yet another peak it can not adapt to. Compared to the single top decay setup, there is a moderate degradation of the Monte Carlo integration/sampling. The unweighting efficiency is reduced by 7 (20) % for the NN (VEGAS) samples. The unoptimised sample’s efficiency is reduced by 40 %.

Finally, we want to study for the case of top–anti-top production how the overall reduction in the width of the weight distributions shown in Fig. 2b translates to more differential observables. We show in Fig. 3 the differential cross section for two observables, the invariant mass of the electron-positron pair m_{ee} and the angle between the electron and the anti-bottom quark $\theta_{e-\bar{b}}$. Note that the invariant mass m_{ee} depends on the lepton momenta of both top-quark decay sequences, whereas the angle $\theta_{e-\bar{b}}$ is an observable that depends on the momenta of only the anti-top quark decay sequence. Comparing the results for VEGAS and NN optimisation (again using the samples with equal sizes, $N = 10^6$), we find that both distributions agree and feature nearly equal MC errors across the whole range of the observable. However, the two samples behave differently when we consider the mean weights per bin in the lower panels. With the weights given by the ratio between the integrand and the sampling distribution, cf. Eq. (8), the plots illustrate how close the sampling distribution approximates the actual target. In the perfect case a constant line at 1 would be seen. Any distortion away from 1 directly translates into a broader global weight distribution. For m_{ee} , we find that VEGAS samples both tails too often to the expense of the intermediate region between 100 and 250 GeV, whereas the NN sample is nearly constant in comparison. Both samples feature distortions for low $\theta_{e-\bar{b}}$, although in different directions. As for both VEGAS and the NN most of the weights are very close to 1, which is also reflected in the weight distribution shown in Fig. 2b, the distortions only have a minor impact on the relative MC errors shown in the middle panels.

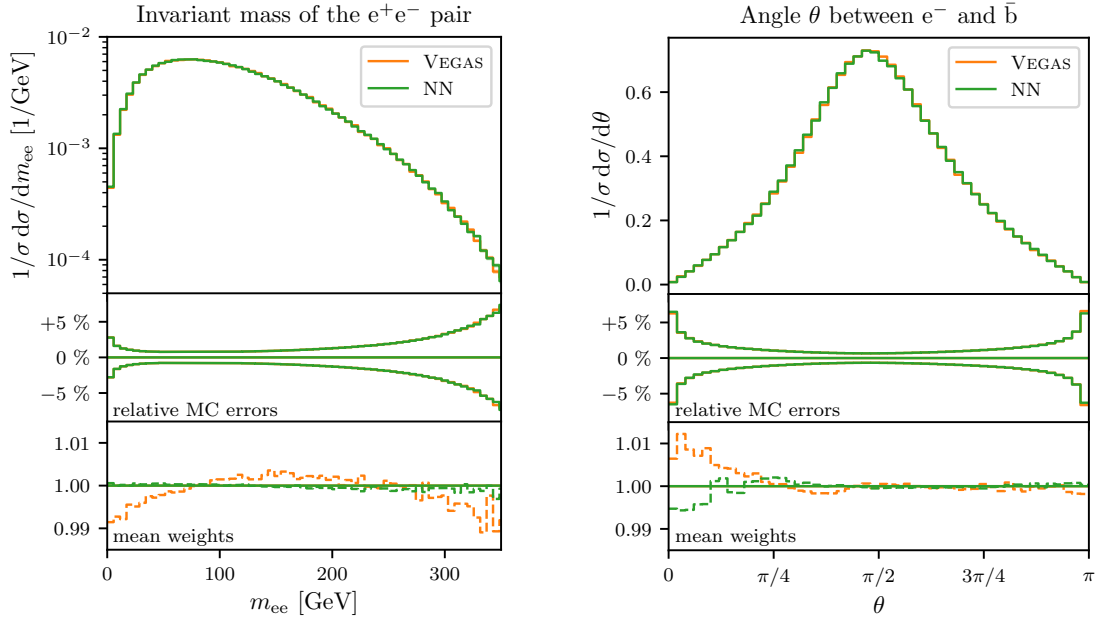


Figure 3: The invariant mass of the electron-positron pair (left) and the angle between the electron and the anti-bottom quark in top-pair production (right). For each observable, we compare the nominal distributions for a VEGAS-optimised and a NN-optimised phase space sampling (upper panes). In the middle panes, Monte Carlo errors for both samples are compared. The lower panes show the mean event weights per bin, highlighting regions of the observables where the sampling distributions over- or undershoot the target.

4.2 Gluon-induced multi-jet production

Finally, we test our approach for $gg \rightarrow n$ gluons with $n = 3$ and $n = 4$, at a fixed centre-of-mass energy $\sqrt{s} = 1$ TeV. For this application, the basic importance sampling density follows the QCD antenna radiation pattern realised by the HAAG algorithm [48, 49], with a number of channels that depends on the number of final-state particles. For $n = 3$, HAAG constructs 24 channels, but after mapping channels that differ in the permutation of the momenta only, this boils down to 2 independent channels. For $n = 4$, there are 120 HAAG channels that can be mapped onto 3 independent channels. Therefore, in contrast to the top-quark applications, a multi-channel algorithm is employed, with one independent NN (or VEGAS) per channel. During the training, the NN are all optimised simultaneously, cf. Sec. 3.3.

Another difference with respect to the top-quark examples is the presence of phase space cuts, used to regularise the n -gluon cross sections. Hence, the optimisation has to deal with “dead” regions in phase space and therefore with non-continuous integrands.

For regularisation, HAAG uses a cut-off parameter which we set to $s_0 = 900 \text{ GeV}^2$. On the final state we employ a cut on the invariant masses of all parton pairs, i.e. $m_{ij} > 30 \text{ GeV}$, and on the transverse momenta of all particles, $p_{\perp,i} > 30 \text{ GeV}$. To select the jets, we use the anti- k_t algorithm [50] with $R = 0.4$. The renormalisation scale is given by $\mu_R = \sqrt{s}$. Each NN consists of 5 coupling layers and 32 bins. The trainings conclude after a maximum of 10^4 optimisation steps, where at each step we train the NN on a minibatch of at least 2048 non-zero phase space points.

In Tab. 2, we show the results of sampling the cross section without optimisation (“Uniform”), with VEGAS optimisation and with our NN optimisation. The unweighting efficiencies

Table 2: Results for sampling the total cross section for gluonic jet production at $\sqrt{s} = 1$ TeV, i.e. the Monte Carlo integral E_N is an estimator for $\sigma_{gg \rightarrow n \text{ jets}}$. Besides E_N and its MC error, we also show the unweighting efficiency ϵ_{uw} of the sample, comparing VEGAS optimisation, NN-based optimisation and an unoptimised (“Uniform”) distribution. All samples consist of $N = 10^6$ points with non-zero weights. The table also lists the acceptance rate P_{acc} .

Sample	3 jets			4 jets		
	ϵ_{uw}	E_N [pb]	P_{acc}	ϵ_{uw}	E_N [pb]	P_{acc}
Uniform	3.0 %	24806(55)	89 %	2.7 %	9869(20)	57 %
VEGAS	27.7 %	24813(23)	32 %	31.8 %	9868(10)	17 %
NN	64.3 %	24847(21)	34 %	33.6 %	9859(10)	16 %

ϵ_{uw} for $n = 3, 4$ are about 3 % for the unoptimised sampling, and increase to about 30 % by VEGAS optimisation. The NN optimisation achieves to surpass VEGAS for $n = 3$ by a factor of two, whereas for $n = 4$ we find no significant improvement over VEGAS. Both VEGAS and NN optimisation gives similar improvements for the estimate of the standard deviation for $n = 3$ and $n = 4$. We also quote in Tab. 2 the acceptance rate $P_{\text{acc}} = N/N_{\text{trials}}$, i.e. the probability that a proposed point passes the phase space cuts and hence provides a finite contribution to the integral result. In our gluon production setup, the cuts regularise the matrix elements, and therefore the matrix element value is expected to be larger close to these cuts than elsewhere. It is therefore unsurprising that both VEGAS and NN optimisation lead to a decrease in P_{acc} , as they enhance the sampling rate close to the cuts, with the side effect of proposing points also outside of the cuts (since the bin edges of both methods will not perfectly coincide with the cuts).

The event weight distributions for the samples are compared with each other in Fig. 4. For 3-jet production, we find that the NN optimisation gives the most strongly peaked weight distribution. The situation is more ambiguous for 4-jet production. Both the VEGAS and NN optimisation significantly sharpen the weight distribution, in fact providing quite similar outcomes. However, while the NN optimisation results in a slightly more pronounced peak compared to VEGAS and a slightly faster fall-off towards large weights, it depletes less quickly towards small weights. In particular for the 3-jet case it might be surprising that we find a comparable estimate for the standard deviation for NN and VEGAS optimisation, although the weight distribution is narrower in the NN case. This apparent discrepancy originates from the higher fraction of zero-weight events for the optimised samples, i.e. events that fall outside the physical phase space volume and are thus not accepted. The standard deviation of the integral estimate is in such a case largely determined by the corresponding acceptance rate, since the weight distribution will then actually contain two peaks: the one at a finite value and one at $w = 0$. A further improvement in the sampling accuracy would therefore require a modification of the optimisation to reduce the number of discarded phase space points. The unweighting efficiency is not affected by $P_{\text{acc}} < 1$, since it takes into account non-zero weights only.

In Fig. 5 we depict the transverse momentum distributions for the jet with the smallest transverse momentum p_{\perp} in three- and four-jet production, i.e. the third and the fourth jet, respectively, again comparing the NN-optimised sample with a VEGAS-optimised one. In the comparisons of the mean weight per bin distributions (lower panels) we find a different behaviour for the two optimisation methods. For three-jet production, Fig. 5a, the NN weights stay very close to one for $p_T \lesssim 240$ GeV, whereas VEGAS samples the lower-most two p_{\perp} bins with weights smaller than unity, which is compensated by weights larger than unity already

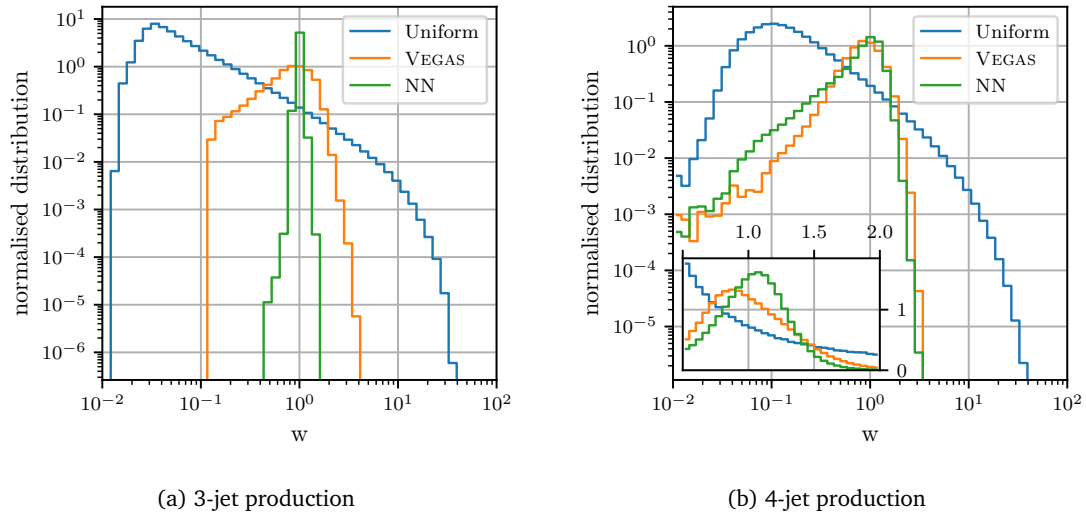


Figure 4: Event weight distributions for sampling the total cross section for $gg \rightarrow n$ jets for $\sqrt{s} = 1$ TeV with $N = 10^6$ points, comparing VEGAS optimisation, NN-based optimisation and an unoptimised (“Uniform”) distribution. Note that we now use a logarithmic scale for the x axis. The inset plot in (b) shows the peak region in more detail and using a linear scale.

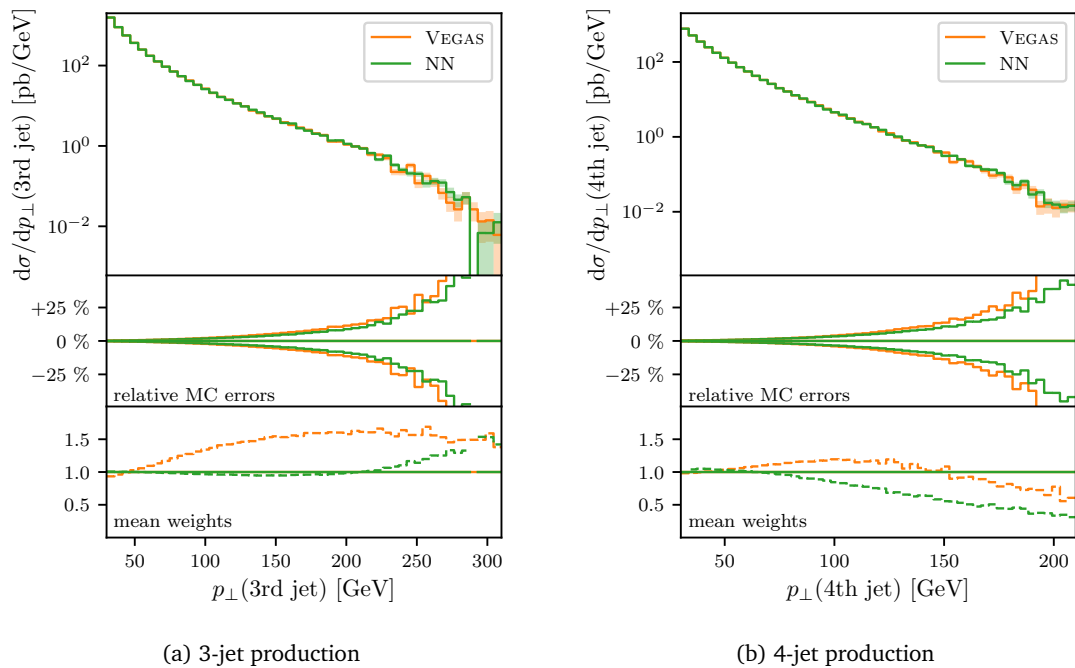


Figure 5: The transverse momentum of the smallest- p_{\perp} jet in three-gluon production (left) and in four-gluon production (right). For both observables, we compare the nominal distributions for a VEGAS-optimised and a NN-optimised phase space sampling (upper panes). In the middle panes, Monte Carlo errors for both samples are compared. The lower panes show the mean event weights per bin.

above 50 GeV. For four-jet production, Fig. 5b, the NN sample differs from unity for p_{\perp} values larger than 80 GeV. Then the weights become increasingly smaller, which corresponds to the long tail of the weight distribution towards smaller weights in Fig. 4b. For VEGAS, weights again begin to differ from unity above 50 GeV. However, there is a turning point and therefore the weights remain closer to unity compared to the ones of the NN sample above 100 GeV. Hence, judging the sample quality is less straightforward in the four-jet case, whereas the NN sample is clearly better in the three-jet case. This is in agreement with the very similar global sample performance given in Tab. 2.

Considering the relative MC errors in the middle panel of Fig. 5b we observe that although the NN distribution differs from the integrand more than the VEGAS distribution in the high- p_{\perp} bins, it still leads to smaller relative errors. This is, however, just a consequence of the statistics: the NN sample features smaller weights in this region as it oversamples the target. Therefore, it generates more events per bin than VEGAS which results in a smaller variance.

5 Conclusions

We have conducted a proof-of-principle study for applying Neural Importance Sampling with piecewise-quadratic coupling layers to optimise phase space sampling in Monte Carlo integration problems in high-energy physics. The approach fulfils the requirements needed to guarantee a faithful sampling of the target distribution. In particular, full phase space coverage is guaranteed. We have investigated the performance of the approach by employing it as a drop-in replacement of the widely used VEGAS optimiser, which we use for comparison benchmarks. Specifically, we have studied the efficiency of the approach both for the integration result and for the generation of weighted and unweighted event samples for the decay width of a top quark, for the cross section of leptonic production of a top-quark pair with subsequent decays; and for the cross sections of gluonic 3-jet and 4-jet production.

We find a significantly improved sampling performance for the simpler examples with a phase space dimensionality up to $d = 5$, namely top decays, top pair production and 3-jet production. For the more complex example of 4-jet production with $d = 8$ and an increased number of importance sampling channels, we have not been able to outperform VEGAS, e.g. the gain factor in the unweighting efficiency dropped from 2.3 for 3-jet production to 1.1 for 4-jet production. Since the complexity of the NN architecture and the number of events per training batch was limited by our computing resources, we expect that the result for the 4-jet case can be improved by using more powerful hardware and/or optimising the implementation. Though, even then the computational challenge would emerge again for 5-jet production, and it is left to further studies to improve the scaling behaviour of the ansatz. Our findings are consistent with those in another study [36], where increasing the final-state multiplicity (and hence the number of channels) in $V + \text{jets}$ production also leads to a rapid reduction in the gain factor.

However, the results for the top quarks and the 3-jet production are promising and indicate that conventional optimisers such as VEGAS can potentially be outperformed by NN-based approaches also for more complex problems in the future. To this end the computational challenges outlined above need to be addressed. In future research we will therefore aim to extend the range in final-state multiplicity while keeping the training costs at an acceptable level, and—if successful—to implement the new sampling techniques within the SHERPA general-purpose event generator framework. A starting point should be the further study and comparison of alternative ways to integrate our NN approach within multi-channel sampling, beginning with our ansatz and the one proposed in [36], to find out if the scaling behaviour can be optimised. On the purely NN side, the exploration of possible extensions or alternatives to piecewise-quadratic coupling layers is promising, such as [51]. Also adversarial training has

the potential to reduce training times significantly. The limitation of the statistical accuracy by a large number of zero-weight events found in the jet-production examples furthermore suggests that it is worthwhile to investigate the construction of optimised importance sampling maps that better respect common phase space cuts, or alternatively to modify the optimisation procedure to further reduce the generation of points outside the fiducial phase space volume.

Acknowledgements

We are grateful to Stefan Höche for fruitful discussion. We would also like to thank Tilman Plehn, Anja Butter and Ramon Winterhalder for useful discussions.

This work has received funding from the European Union's Horizon 2020 research and innovation programme as part of the Marie Skłodowska-Curie Innovative Training Network MCnetITN3 (grant agreement no. 722104). SS acknowledges support through the Fulbright-Cottrell Award and from BMBF (contract 05H18MGCA1).

A Auxiliary jet p_{\perp} distributions in multi-jet production

In this appendix we compile additional plots of the jet p_{\perp} distributions in 3- and 4-gluon production from gluon annihilation at $\sqrt{s} = 1$ TeV. Details on the calculational setup are given in Sec. 4.2.

The leading and second-leading jet p_{\perp} distribution in 3-gluon production are depicted in Fig. 6a. The leading, second- and third-leading jet p_{\perp} distributions in 4-gluon production are shown in Fig. 6b.

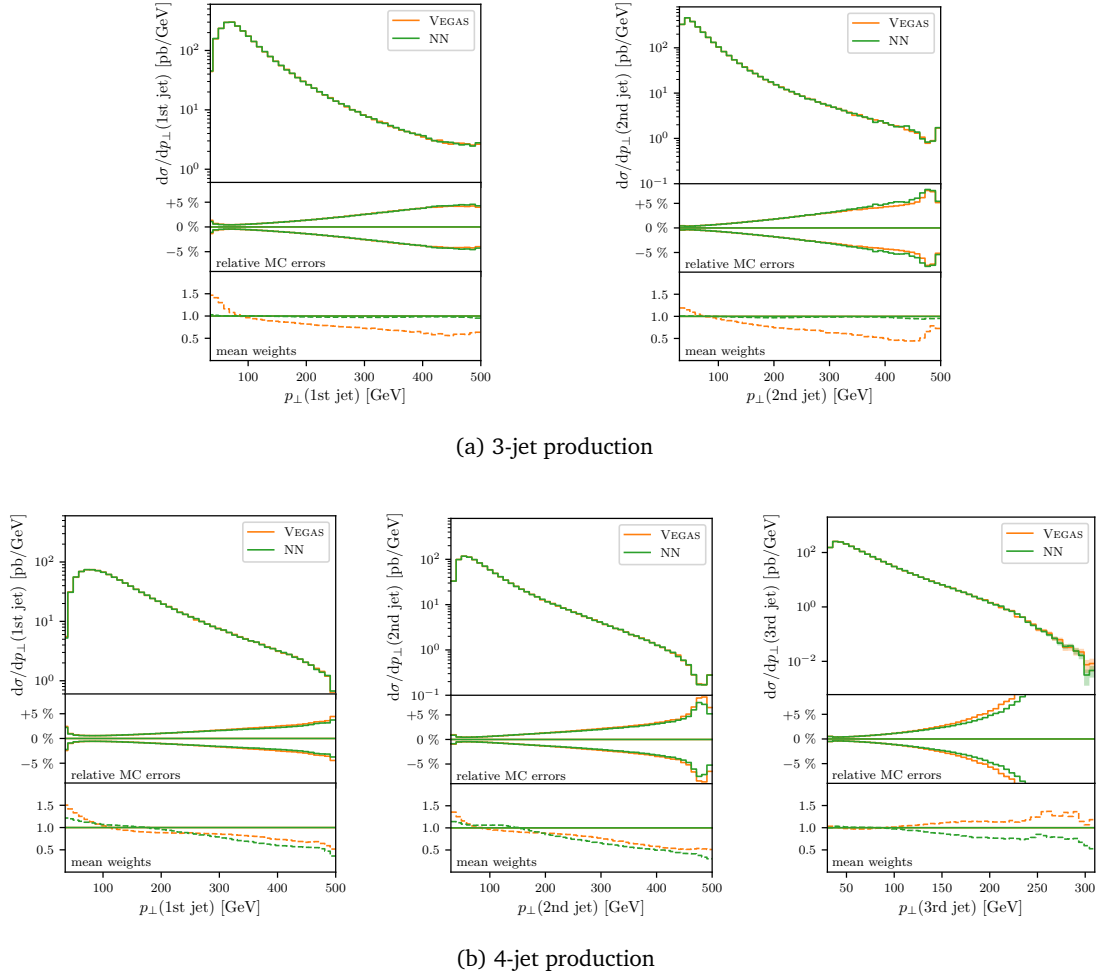


Figure 6: Distributions of the transverse momentum p_{\perp} of the leading and second-leading jets in three-gluon production (a) and for the leading, second- and third-leading jets in four-gluon production (b). For each observable, we compare the nominal distributions for a VEGAS-optimised and a NN-optimised phase space sampling (upper panes). In the middle panes of each plot, Monte Carlo errors for both samples are compared. The lower panes show the mean event weights per bin.

References

- [1] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, Comput. Phys. Commun. **191**, 159 (2015), doi:[10.1016/j.cpc.2015.01.024](https://doi.org/10.1016/j.cpc.2015.01.024).
- [2] M. Bähr et al., *Herwig++ physics and manual*, Eur. Phys. J. C **58**, 639 (2008), doi:[10.1140/epjc/s10052-008-0798-9](https://doi.org/10.1140/epjc/s10052-008-0798-9).
- [3] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert and J. Winter, *Event generation with SHERPA 1.1*, J. High Energ. Phys. **02**, 007 (2009), doi:[10.1088/1126-6708/2009/02/007](https://doi.org/10.1088/1126-6708/2009/02/007).
- [4] E. Bothmann et al., *Event generation with Sherpa 2.2*, SciPost Phys. **7**, 034 (2019), doi:[10.21468/SciPostPhys.7.3.034](https://doi.org/10.21468/SciPostPhys.7.3.034).
- [5] A. Buckley et al., *General-purpose event generators for LHC physics*, Phys. Rep. **504**, 145 (2011), doi:[10.1016/j.physrep.2011.03.005](https://doi.org/10.1016/j.physrep.2011.03.005).
- [6] F. Krauss, R. Kuhn and G. Soff, *AMEGIC++ 1.0, A Matrix Element Generator In C++*, J. High Energ. Phys. **02**, 044 (2002), doi:[10.1088/1126-6708/2002/02/044](https://doi.org/10.1088/1126-6708/2002/02/044).
- [7] T. Gleisberg and S. Höche, *Comix, a new matrix element generator*, J. High Energ. Phys. **12**, 039 (2008), doi:[10.1088/1126-6708/2008/12/039](https://doi.org/10.1088/1126-6708/2008/12/039).
- [8] J. Alwall, P. Demin, S. de Visscher, R. Frederix, M. Herquet, F. Maltoni, T. Plehn, D. L. Rainwater and T. Stelzer, *MadGraph/MadEvent v4: the new web generation*, J. High Energ. Phys. **09**, 028 (2007), doi:[10.1088/1126-6708/2007/09/028](https://doi.org/10.1088/1126-6708/2007/09/028).
- [9] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, J. High Energ. Phys. **07**, 079 (2014), doi:[10.1007/JHEP07\(2014\)079](https://doi.org/10.1007/JHEP07(2014)079).
- [10] W. Kilian, T. Ohl and J. Reuter, *WHIZARD—simulating multi-particle processes at LHC and ILC*, Eur. Phys. J. C **71**, 1742 (2011), doi:[10.1140/epjc/s10052-011-1742-y](https://doi.org/10.1140/epjc/s10052-011-1742-y).
- [11] F. Cascioli, P. Maierhöfer and S. Pozzorini, *Scattering amplitudes with open loops*, Phys. Rev. Lett. **108**, 111601 (2012), doi:[10.1103/PhysRevLett.108.111601](https://doi.org/10.1103/PhysRevLett.108.111601).
- [12] S. Alioli, P. Nason, C. Oleari and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, J. High Energ. Phys. **06**, 043 (2010), doi:[10.1007/JHEP06\(2010\)043](https://doi.org/10.1007/JHEP06(2010)043).
- [13] S. Actis, A. Denner, L. Hofer, A. Scharf and S. Uccirati, *Recursive generation of one-loop amplitudes in the Standard Model*, J. High Energ. Phys. **04**, 037 (2013), doi:[10.1007/JHEP04\(2013\)037](https://doi.org/10.1007/JHEP04(2013)037).
- [14] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf and S. Uccirati, *RECOLA—REcursive Computation of One-Loop Amplitudes*, Comput. Phys. Commun. **214**, 140 (2017), doi:[10.1016/j.cpc.2017.01.004](https://doi.org/10.1016/j.cpc.2017.01.004).
- [15] 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), doi:[10.1103/PhysRevD.100.014024](https://doi.org/10.1103/PhysRevD.100.014024).
- [16] W. T. Giele, G. C. Stavenga and J.-C. Winter, *A forward branching phase-space generator* (2011), [arXiv:1106.5045](https://arxiv.org/abs/1106.5045).

- [17] A. van Hameren, *PARNI for importance sampling and density estimation*, Acta Phys. Polon. B **40**, 259 (2009), [arXiv:0710.2448](#).
- [18] P. Nason, *MINT: a computer program for adaptive Monte Carlo integration and generation of unweighted distributions* (2007), [arXiv:0709.2085](#).
- [19] A. van Hameren, *Kaleu: a general-purpose parton-level phase space generator* (2010), [arXiv:1003.4953](#).
- [20] T. M. Figy and W. T. Giele, *A forward branching phase space generator for hadron colliders*, J. High Energ. Phys. **10**, 203 (2018), doi:[10.1007/JHEP10\(2018\)203](#).
- [21] T. Chen, T. M. Figy and W. T. Giele, *A projective phase space generator for hadronic vector boson plus one jet production* (2019), [arXiv:1907.03893](#).
- [22] T. Hahn, *Cuba—a library for multidimensional numerical integration*, Comput. Phys. Commun. **168**, 78 (2005), doi:[10.1016/j.cpc.2005.01.010](#).
- [23] S. Jadach, *Foam: Multi-dimensional general purpose Monte Carlo generator with self-adapting simplicial grid*, Comput. Phys. Commun. **130**, 244 (2000), doi:[10.1016/S0010-4655\(00\)00047-3](#).
- [24] S. Jadach, *Foam: A general-purpose cellular Monte Carlo event generator*, Comput. Phys. Commun. **152**, 55 (2003), doi:[10.1016/S0010-4655\(02\)00755-5](#).
- [25] H. Kharraziha and S. Moretti, *The Metropolis algorithm for on-shell four-momentum phase space*, Comput. Phys. Commun. **127**, 242 (2000), doi:[10.1016/S0010-4655\(99\)00504-4](#).
- [26] K. Kröniger, S. Schumann and B. Willenberg, *(MC)³ – a multi-channel Markov chain Monte Carlo algorithm for phase-space sampling*, Comput. Phys. Commun. **186**, 1 (2015), doi:[10.1016/j.cpc.2014.08.024](#).
- [27] J. Bendavid, *Efficient Monte Carlo integration using boosted decision trees and generative deep neural networks* (2017), [arXiv:1707.00028](#).
- [28] M. D. Klimek and M. Perelstein, *Neural network-based approach to phase space integration* (2018), [arXiv:1810.11509](#).
- [29] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri and R. Verheyen, *Event generation and statistical sampling for physics with deep generative models and a density information buffer* (2019), [arXiv:1901.00875](#).
- [30] R. Di Sipio, M. Fauci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC*, J. High Energ. Phys. **08**, 110 (2019), doi:[10.1007/JHEP08\(2019\)110](#).
- [31] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC events*, SciPost Phys. **7**, 075 (2019), doi:[10.21468/SciPostPhys.7.6.075](#).
- [32] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9 (2015), [arXiv:1410.8516](#).
- [33] T. Müller, B. McWilliams, F. Rousselle, M. Gross and J. Novák, *Neural importance sampling* (2018), [arXiv:1808.03856](#).

- [34] Q. Zheng and M. Zwicker, *Learning to importance sample in primary sample space*, Comput. Graph. Forum **38**, 169 (2019), doi:[10.1111/cgf.13628](https://doi.org/10.1111/cgf.13628).
- [35] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional integration and sampling with normalizing flows* (2020), [arXiv:2001.05486](https://arxiv.org/abs/2001.05486).
- [36] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event generation with normalizing flows*, Phys. Rev. D **101**, 076002 (2020), doi:[10.1103/PhysRevD.101.076002](https://doi.org/10.1103/PhysRevD.101.076002).
- [37] K. T. Matchev and P. Shyamsundar, *Uncertainties associated with GAN-generated datasets in high energy physics* (2020), [arXiv:2002.06307](https://arxiv.org/abs/2002.06307).
- [38] R. Kleiss and R. Pittau, *Weight optimization in multichannel Monte Carlo*, Comput. Phys. Commun. **83**, 141 (1994), doi:[10.1016/0010-4655\(94\)90043-4](https://doi.org/10.1016/0010-4655(94)90043-4).
- [39] E. Byckling and K. Kajantie, *N-particle phase space in terms of invariant momentum transfers*, Nucl. Phys. B **9** (1969), doi:[10.1016/0550-3213\(69\)90271-5](https://doi.org/10.1016/0550-3213(69)90271-5).
- [40] G. P. Lepage, *A new algorithm for adaptive multidimensional integration*, J. Comput. Phys. **27**, 192 (1978), doi:[0.1016/0021-9991\(78\)90004-9](https://doi.org/10.1016/0021-9991(78)90004-9).
- [41] T. Ohl, *Vegas revisited: Adaptive Monte Carlo integration beyond factorization*, Comput. Phys. Commun. **120**, 13 (1999), doi:[10.1016/S0010-4655\(99\)00209-X](https://doi.org/10.1016/S0010-4655(99)00209-X).
- [42] S. Kullback and R. A. Leibler, *On information and sufficiency*, Ann. Math. Statist. **22**, 79 (1951), doi:[10.1214/aoms/1177729694](https://doi.org/10.1214/aoms/1177729694).
- [43] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2014), [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [44] C. B. Barber, D. P. Dobkin and H. Huhdanpaa, *The Quickhull algorithm for convex hulls*, ACM Trans. Math. Software **22**, 469 (1996).
- [45] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP* (2016), [arXiv:1605.08803](https://arxiv.org/abs/1605.08803).
- [46] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous distributed systems* (2015), [arXiv:1603.04467](https://arxiv.org/abs/1603.04467), Software available from [tensorflow.org](https://www.tensorflow.org).
- [47] S. Höche, S. Kuttimalai, S. Schumann and F. Siegert, *Beyond standard model calculations with Sherpa*, Eur. Phys. J. C **75**, 135 (2015), doi:[10.1140/epjc/s10052-015-3338-4](https://doi.org/10.1140/epjc/s10052-015-3338-4).
- [48] A. van Hameren and R. Kleiss, *Generating QCD antennas*, Eur. Phys. J. C **17**, 611 (2000), doi:[10.1007/s100520000508](https://doi.org/10.1007/s100520000508).
- [49] A. van Hameren and C. G. Papadopoulos, *A hierarchical phase space generator for QCD antenna structures*, Eur. Phys. J. C **25**, 563 (2002), doi:[10.1007/s10052-002-1000-4](https://doi.org/10.1007/s10052-002-1000-4).
- [50] M. Cacciari, G. P. Salam and G. Soyez, *The anti- k_t jet clustering algorithm*, J. High Energ. Phys. **04**, 063 (2008), doi:[10.1088/1126-6708/2008/04/063](https://doi.org/10.1088/1126-6708/2008/04/063).
- [51] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural spline flows* (2019), [arXiv:1906.04032](https://arxiv.org/abs/1906.04032).
- [52] R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, *Neural ordinary differential equations* (2018), [arXiv:1806.07366](https://arxiv.org/abs/1806.07366).

- [53] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever and D. Duvenaud, *FFJORD: Free-form continuous dynamics for scalable reversible generative models* (2018), [arXiv:1810.01367](#).
- [54] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1x1 convolutions* (2018), [arXiv:1807.03039](#).
- [55] E. Hoogeboom, R. van den Berg and M. Welling, *Emerging convolutions for generative normalizing flows* (2019), [arXiv:1901.11137](#).
- [56] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud and J.-H. Jacobsen, *Invertible residual networks*, Proceedings of the International Conference on Machine Learning (ICML) (2019), [arXiv:1811.00995](#).
- [57] J.-H. Jacobsen, A. Smeulders and E. Oyallon, *i-RevNet: Deep invertible networks*, ICLR 2018 - International Conference on Learning Representations, Apr 2018, Vancouver, Canada (2018), [arXiv:1802.07088](#).
- [58] R. Cornish, A. L. Caterini, G. Deligiannidis and A. Doucet, *Relaxing bijectivity constraints with continuously indexed normalising flows* (2019), [arXiv:1909.13833](#).
- [59] A. Wehenkel and G. Louppe, *Unconstrained monotonic neural networks*, Adv. Neural Inf. Process. Syst. (2019), [arXiv:1908.05164](#).

3.1.4 Impact

Together with an almost simultaneously published study [47, 230], our article presents the first application of NFS to HEP PS sampling. It shows that NFS are a promising tool that is able to improve the efficiency of current tools. The paper was well received by the community. As of 2 June 2023, according to *SciPost* there are 46 citations in peer-reviewed journals [236], and there are several later works with direct relation to our ideas. These include discussions about bijectivity [35, 237], sampling with autoregressive flows instead of coupling layers [50], neural importance sampling applied to neutrino-nucleus cross-section models [238], a discussion about the topological issues of NFS [92], and the usage of NFS in refs. [53, 60, 239].

The work was presented by Enrico Bothmann at ‘The Eighth Annual Conference on Large Hadron Collider Physics’ (LHCP 2020) in May 2020. This contribution was published as part of the proceedings in ref. [240]. Another presentation of the work was given by me as part of a talk at the ‘20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research’ (ACAT 2021). A conference paper was subsequently published in ref. [241]. Therein are also new results which were obtained after the publication of the original article. The new results are based on a reimplementaion in PYTORCH done by me. It uses rational-quadratic splines [51] and residual NNS [242] in the coupling layers, but otherwise resembles the original model. Using this setup, an unweighting efficiency of 49 % could be achieved for the 4 jets gluon scattering example. This is significantly better than the value of 34 % printed in tab. 2 of the original article. As a consequence, the model outperforms the VEGAS mapping. Two figures from the conference paper, showing the improvements for the 4 jets example, are reproduced in fig. 3.6. The first one, fig. 3.6a, shows the weight distributions using the unoptimized (‘uniform’), VEGAS-optimized, and NF-optimized samplers. Compared to fig. 4b of the original article, it can be seen that the weight distribution from the NF-based sampler is significantly improved. The width of the distribution is clearly reduced and it features a smaller maximum. In addition, the distribution of the transverse momentum of the smallest p_{\perp} -jet is shown in fig. 3.6b. A clear improvement is visible in this figure, too. Compared to fig. 5b of the original article, the mean weights per bin are now much closer to one for the NF-optimized sampler. Obviously, the sampling distribution approximates the target distribution much better over a wide range of p_{\perp} . In the original article, there was considerable oversampling of the high- p_{\perp} region, as is visible in the lower pane showing the mean weights. With increasing p_{\perp} , the mean weights drift away from one towards lower values. At the highest p_{\perp} , the values are below 0.5, meaning the sampling distribution oversamples the target by more than a factor of two. As a result of this, the relative MC errors, shown in the middle pane, are smaller than with VEGAS, for which the oversampling is less pronounced. With the updated results in fig. 3.6b, this effect is not visible anymore. For both VEGAS and NF, the relative MC errors are similar, since the distance between the two sampling distributions has decreased.

3.1.5 Normalization during training

There is an interesting detail that was realized only after the publication of ref. [9]. Specific loss functions like the mean squared error (MSE) allow to choose a normalization of the concerned distributions. In the presence of phase space cuts, this normalization can have a big influence on the result of training. In this section, the observation is illustrated by a toy example. We consider a function $f : [0, 1]^2 \rightarrow \mathbb{R}_{\geq 0}$, $(x, y) \mapsto f(x, y)$. Let us first define coordinates that

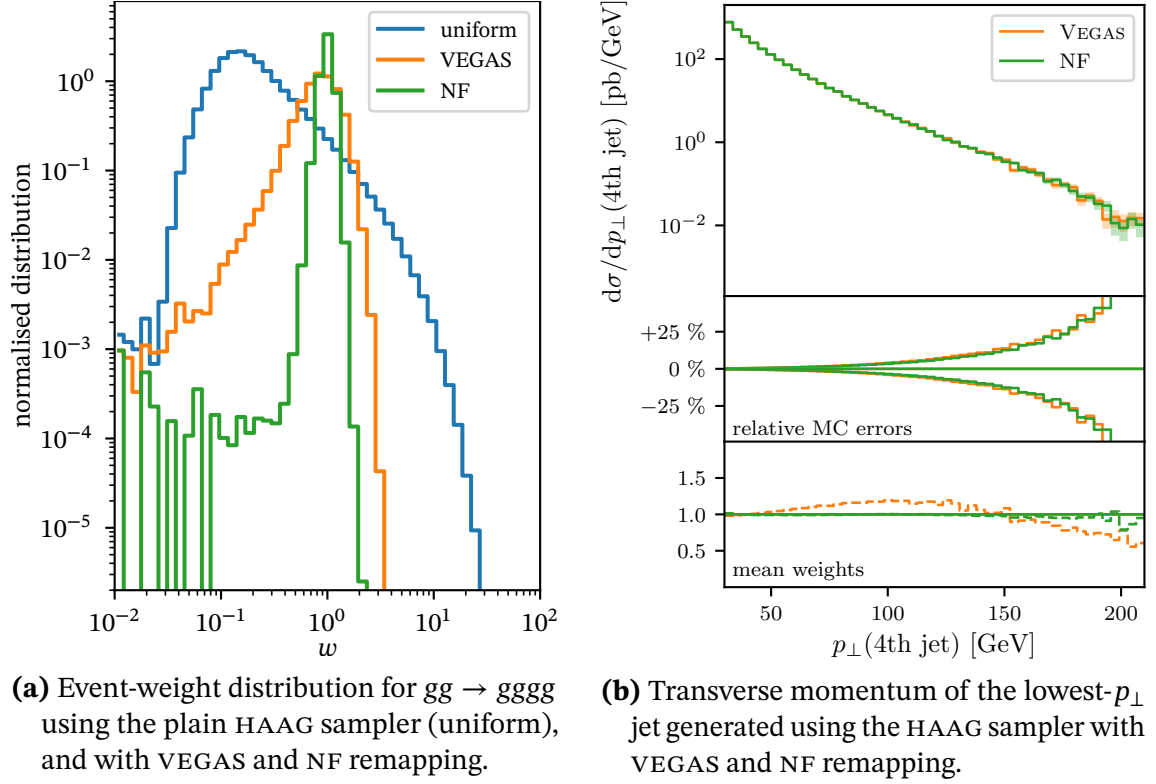




Figure 3.6: Sample results for the process $gg \rightarrow gggg$. Figures taken from [241].  

allow us to write the target function in a compact form:

$$a(x, y) = \left[\left(x - \frac{1}{2} \right) \cdot \cos\left(\frac{\pi}{4}\right) - \left(y - \frac{1}{2} \right) \cdot \sin\left(\frac{\pi}{4}\right) + \frac{1}{2} \right], \quad (3.17)$$

$$b(x, y) = \left[\left(x - \frac{1}{2} \right) \cdot \sin\left(\frac{\pi}{4}\right) + \left(y - \frac{1}{2} \right) \cdot \cos\left(\frac{\pi}{4}\right) + \frac{1}{2} \right], \quad (3.18)$$

$$r(x, y) = \sqrt{\left(x - 1 \right)^2 + \left(y - \frac{1}{2} \right)^2}. \quad (3.19)$$

The coordinates a and b are shifted and rotated such that they correspond to the diagonals of the square $[0, 1]^2$, while r is a radial coordinate centred at $(1, 0.5)$. Using these coordinates, the target function is defined as

$$f(a, b, r) = \begin{cases} e^{-\alpha \cdot a} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(b-\mu)^2} + \frac{1}{\beta} r^{-3} & \text{if } a > 0.1 \text{ and } r > 0.3, \\ 0 & \text{else.} \end{cases} \quad (3.20)$$

It features a Gaussian distribution in the b direction with mean μ and standard deviation σ and an exponential distribution in the a direction. Added to this is an r^{-3} term centred at $r = 0$. The parameters are set to $\alpha = 7$, $\beta = 30$, $\sigma = 0.1$ and $\mu = 0.5$. While the definition of this toy function looks complicated, it is in the end a combination of simple functions in different coordinate systems. The motivation for this is that similar complications can be expected to appear in squared amplitudes. There are multiple peaks of different shape and large regions of low probability. In our example, the regions with $a < 0.1$ or $r < 0.3$ are deliberately cut off to imitate PS cuts used in HEP. Due to these cuts there are two modes with vertical flanks. This is also a typical situation in the sampling of differential cross-sections, where regularized divergences lead to functions that peak towards the edge of PS. Being defined in terms of

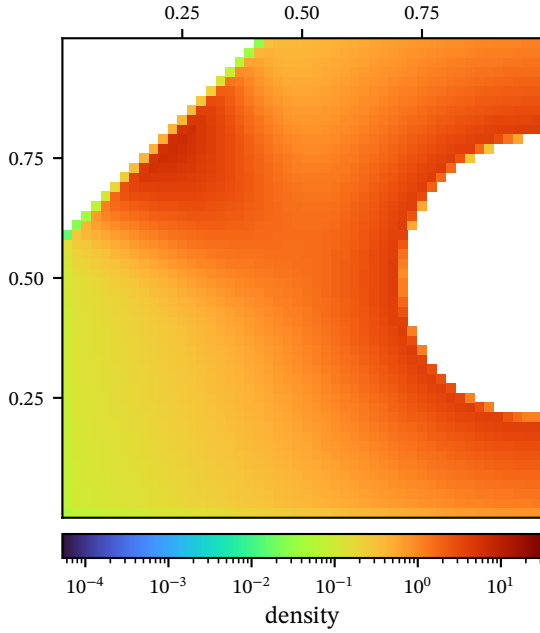


Figure 3.7: The distribution of the function f . The horizontal axis shows the x coordinate and the vertical axis shows the y coordinate. For comparability, the normalization of the colourmap was chosen to be the same as in fig. 3.9. As a result, the plotted values cover only a small range of the colourmap. Note that the ‘turbo’ colourmap is not perceptually uniform.

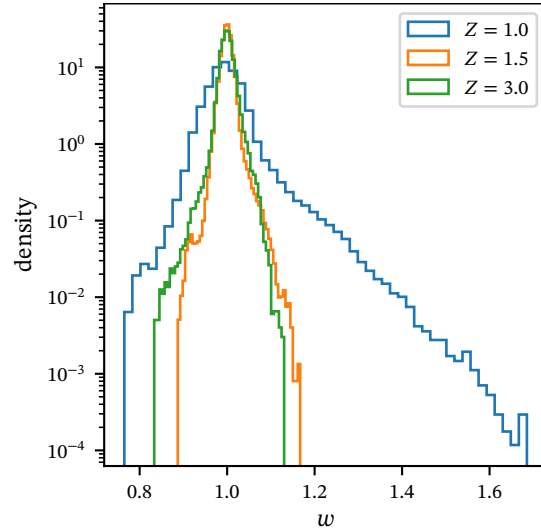


Figure 3.8: Distributions of the weights from samples of size 1M produced by the NFs trained with different normalization factors. Zero weights are excluded and the weights are normalized to one for comparability.

physics-motivated variables, the cuts are not necessarily aligned with the sampling coordinates. In fig. 3.7, the distribution of the toy example in the x - y plane is shown. The function can be expected to be a difficult target for VEGAS, since neither the modes nor the cuts are aligned with the coordinate axes. For this example, we deliberately refrain from using a multi-channel sampler in suitable coordinates.

An NF has been trained on the target function using a batch size of 5000. It uses two coupling blocks with rational-quadratic splines that have 24 knots each. The parameters of the coupling transforms have been determined by residual NNs with three hidden layers and 32 nodes. For optimization, the ADAM optimizer with a learning rate of 0.001 has been used. The learning rate has been reduced on plateaus and the training has been stopped when the validation loss, based on a sample of 1M points, did not increase for 150 consecutive epochs. In each epoch, the training data has been produced anew by sampling from a uniform distribution and transforming the points with the NF. An MSE loss function has been used. The target function is normalized such that it integrates to one. The NF is normalized to one by construction but here we introduce a normalization factor Z that increases the normalization by $Z \in \{1.0, 1.5, 3.0\}$. The training has been repeated for each value of Z . In fig. 3.8 the resulting distributions of the MC weights are shown for the models with the best validation loss. It can be seen that the normalization has a major impact. The larger normalization factors result in much narrower distributions with a steeper decline towards large weights and lower maximum values.

To understand how the narrowing of the weight distribution is achieved, we can look at the distributions defined by the three NFs as shown in fig. 3.9. For $Z = 1.0$, the distribution looks very similar to the target function and the density is low where $f(x, y) = 0$. Due to its surjectivity, the NF’s density cannot be zero anywhere and therefore it ‘spills over’ into the cut

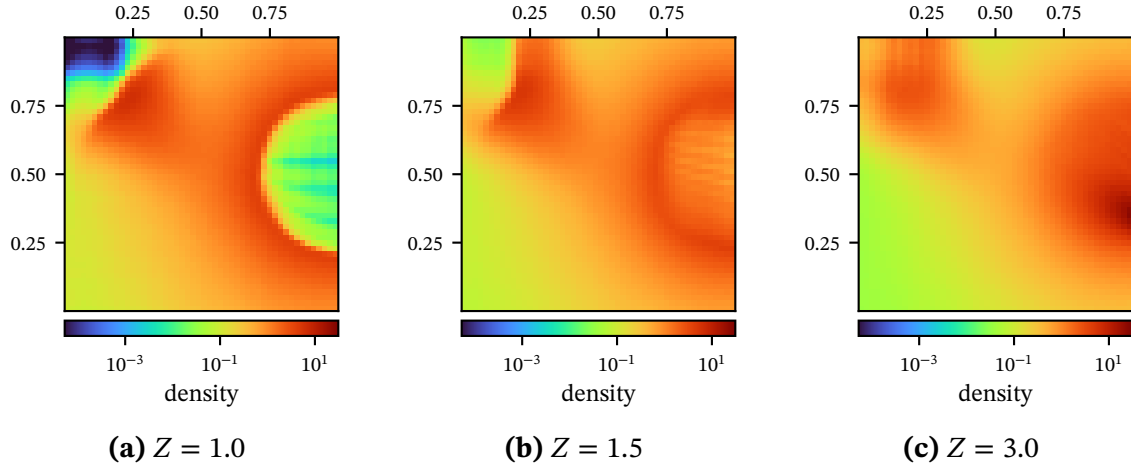


Figure 3.9: Densities defined by the NFs using three different values of the normalization factor Z . The colourmaps are normalized to the same values, such that the plots can be easily compared. Note that the ‘turbo’ colourmap is not perceptually uniform.

region. However, it tries to reproduce the sharp edge at the cut. For $Z = 1.5$, there is much more ‘spillover’ since the NF is allowed to put $\frac{1}{3}$ of the points into the cut regions. Especially around $r = 0.3$ it can be seen that the vertical edge of the circular cut is smoothed out. Finally, for $Z = 3.0$, most of the points land in the target’s cut regions such that the NF produces peaks there. It is no longer apparent from the plot where the cuts actually are.

We can investigate the behaviour around the cuts in more detail by looking at the coordinates in which the cuts are defined. In figs. 3.10a and 3.10b, the distributions of a and r are shown, respectively. For $Z = 1.0$, the distribution is close to the target. Some points are generated below the cuts at $a < 0.1$ and $r < 0.3$, and accordingly other regions are slightly undersampled. However, for the larger normalization factors the distributions significantly differ from the target. Especially for $Z = 3.0$, it can be seen that the peaks are displaced from their target positions and there are no more indications of the vertical edges.

In figs. 3.10c and 3.10d, a close-up view around the cuts is shown. For these plots, the points outside of the cuts have been ignored, such that only points with $a > 0.1$ and $r > 0.3$ enter the histograms. In an event generation setting, this would correspond to the situation where we have applied all phase space cuts and look at the events entering the unweighting procedure. Clearly, in the cases with $Z > 1.0$ the distributions are much closer to the target, and we see the best approximation for the largest value, $Z = 3.0$. With unit normalization, the NF underestimates the peak, since it has trouble to smoothly fit the sharp edge.

Finally, the performance of the NFs is quantified in table 3.1. As expected, the acceptance probability P_{acc} is close to one for $Z = 1.0$, $\frac{2}{3}$ for $Z = 1.5$ and $\frac{1}{3}$ for $Z = 3.0$. The relative MC uncertainty is smallest for unit normalization and grows for increasing normalization factor, being almost five times as large for $Z = 3.0$. This can be explained by the fact that the variance of the MC estimator depends strongly on P_{acc} . If P_{acc} is low, the weight distribution has a high peak at zero, and thus a high variance. For the unweighting efficiency, however, the behaviour is vice versa. An increase of efficiency can be observed from 71 % to over 90 % by using $Z > 1.0$. While the unweighting efficiency is independent of the number of zero weights, it benefits from the improved approximation near the cuts as shown in fig. 3.10. It leads to smaller weight maxima as confirmed by fig. 3.8.

In summary, in the presence of cuts, the unweighting efficiency can be positively influenced by normalizing the NF during training. The ‘spillover’ into the cut-off regions allows the NF to better model the behaviour at the cut edges. This advantage comes at the price of a lower acceptance probability with regard to the cuts. In the relevant applications, however, the

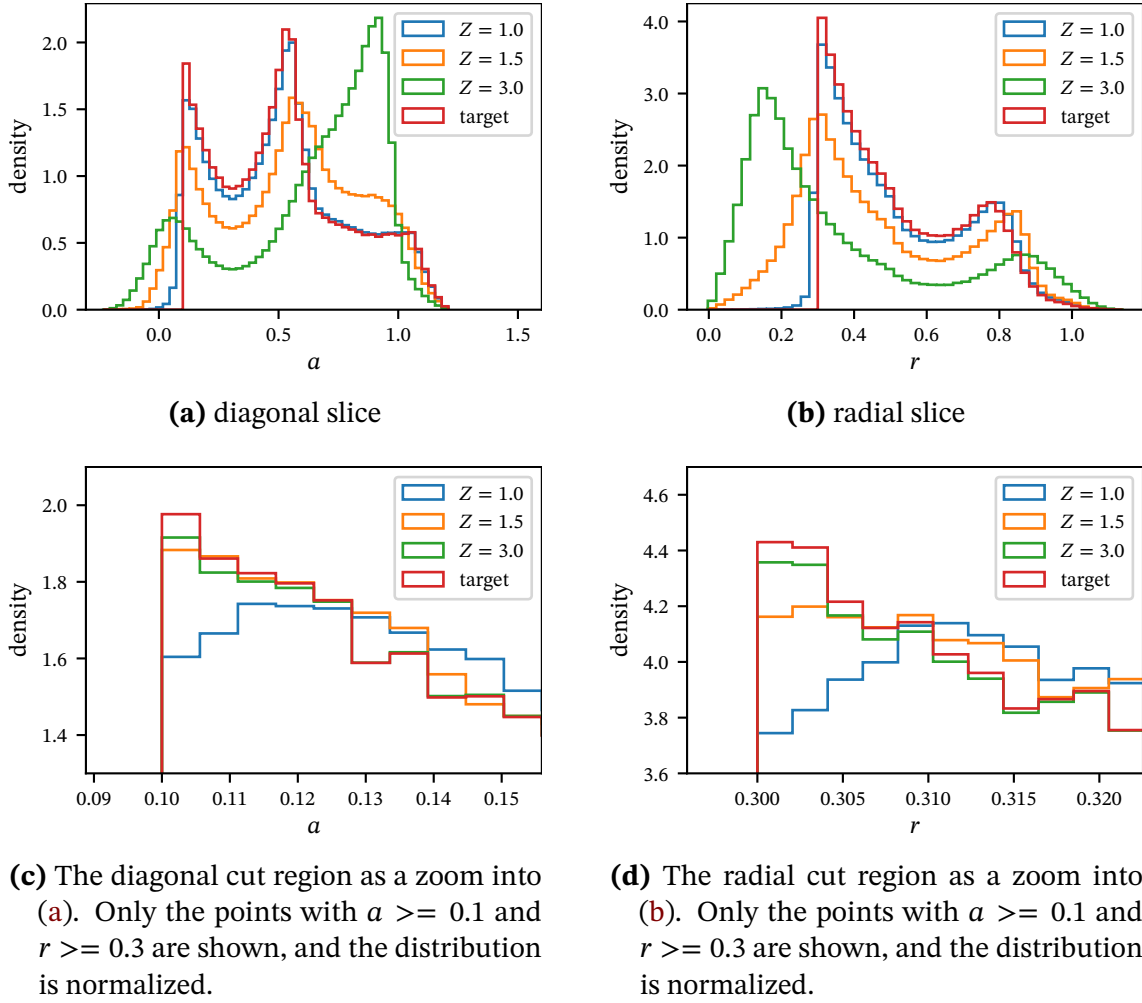


Figure 3.10: Distributions of 1M points sampled from the NFS using three different values of the normalization factor Z . The diagonal coordinate a (a, c) and the radial coordinate r (b, d) are shown. For comparison, the target distribution is shown. In the bottom row, (c, d) the regions around the cuts are shown without including the points lying outside of the cuts. All histograms have been normalized to unit area after applying the cuts.

Table 3.1: The acceptance probability P_{acc} , the relative MC uncertainty $\Delta I/I$ and the unweighting efficiency η for the three normalization factors Z , using samples of 1M points. For the determination of η , a reduced weight maximum as described in section 4.1 has been used.

Z	P_{acc}	$\Delta I/I$	η
1.0	.92	0.000 29	.71
1.5	.67	0.000 71	.91
3.0	.33	0.001 41	.92

evaluation of the target function is likely to be many times more expensive than the generation of a phase space point. The optimal value of Z must be found specifically for the application.

Interestingly, the behaviour of NFs in the presence of phase space cuts is strikingly different from VEGAS', which is examined in section 2.3.2. For $Z = 1.0$, the NF behaves similar to VEGAS in that it struggles to properly sample the target function close to the cut. Larger values of Z , however, allow it to inter- and extrapolate in the cut region. This is something that VEGAS is not able to achieve. Furthermore, for the NF it does not matter whether the cuts or any other features are aligned with the coordinate axes.

3.1.6 Recent developments

In this section, two more recent developments are presented. The first is the idea to make the weights of a multichannel density phase-space dependent by defining them in a local way. These local multichannel weights lend themselves to automatic optimization with NNS. The second is a discussion about mixtures of NFs. An interesting idea is the combination of the two approaches.

Phase space dependent multichannel weights

In section 2.3.3, it is explained how the channel weights of a multichannel density can be automatically adapted. An obvious idea is to make the channel weights phase space dependent. This amounts to changing eq. (2.33) into

$$g(\mathbf{x}) = \sum_{j=1}^{N_c} \alpha_j(\mathbf{u}) g_j(\mathbf{x}). \quad (3.21)$$

Of course the $\alpha_j(\mathbf{u})$ still need to sum to one, so we require

$$\sum_{j=1}^{N_c} \alpha_j(\mathbf{u}) = 1. \quad (3.22)$$

It seems sensible to let an NN learn the $\alpha_j(\mathbf{u})$. A standard feedforward NN should certainly be able to do so provided that the output layer uses a softmax activation function to ensure that eq. (3.22) is satisfied. However, applying this naively can lead to non-optimal results, as is shown below. The problem is related to the fact that we have taken the channel weights to be dependent on the input variable $\mathbf{u} \sim p_{\mathbf{u}}(\mathbf{u})$, since when we want to sample $g(\mathbf{x})$ we need to choose a channel based on \mathbf{u} . Each channel then maps a given input point, \mathbf{u}_i , to a different output point, \mathbf{x}_i , leading to some counter-intuitive behaviour. This shall be illustrated by an example to become more clear.

Consider the one-dimensional target function

$$f(x) = \begin{cases} 4x & \text{if } x < 0.5, \\ 4(1-x) & \text{else,} \end{cases} \quad (3.23)$$

defined over the unit interval $x \in [0, 1)$. The function has two linear parts, one with positive slope for $x < 0.5$ and one with negative slope for $x \geq 0.5$. The graph of the function is shown in fig. 3.11. To sample the target function, we define two channels that can be combined in a multichannel density:

$$g_1(x) = 2x, \quad (3.24)$$

$$g_2(x) = 2(1-x). \quad (3.25)$$

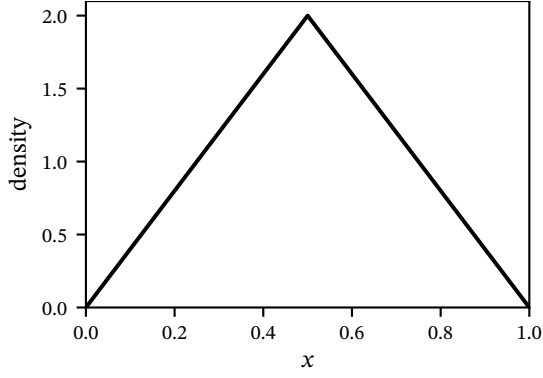


Figure 3.11: The target function of the toy example.

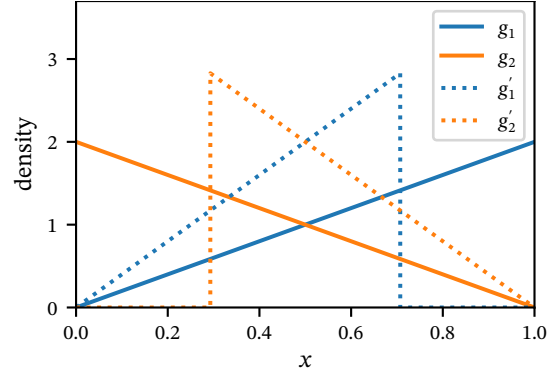


Figure 3.12: The channels used to sample the target distribution. For the dotted versions, the channels have been scaled and shifted.

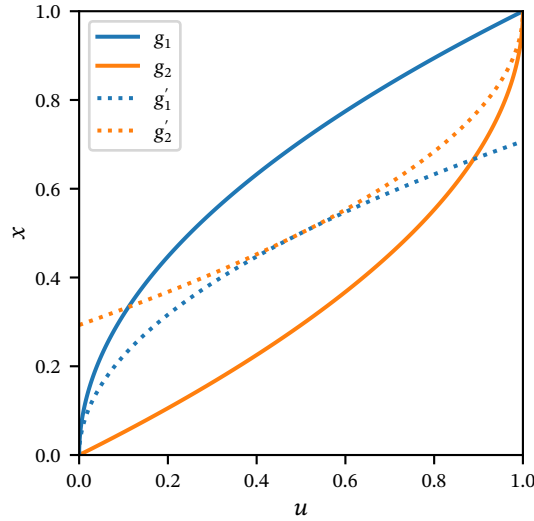


Figure 3.13: The transformations used to sample the channels by mapping the input variable u to the output variable x . For the dotted versions, the channels have been scaled and shifted.

These are simply two linear functions, which are visualized in fig. 3.12 (solid lines). To sample the channels, we use the inverse transform method, which provides us with mappings to transform a uniform input variable u into a linearly distributed output variable x as defined by $g_1(x)$ and $g_2(x)$. The transformations are given by square root functions:

$$T_1 : u \mapsto x = \sqrt{u}, \quad (3.26)$$

$$T_2 : u \mapsto x = 1 - \sqrt{1 - u}. \quad (3.27)$$

These are shown in fig. 3.13 (solid lines). In the conventional multichannel approach, the channels can be mixed with constant weights. An equal mixture, $\alpha_1 = \alpha_2 = 0.5$, leads to uniform sampling and an unweighting efficiency of $\eta = 0.5$.

In order to try to improve upon this, we make the multichannel weights phase space dependent. Our naive first attempt is to exclusively sample g_1 in the left half, $u < 0.5$, and g_2 in the

right half. This choice corresponds to the weight functions

$$\alpha_1(u) = \begin{cases} 1 & \text{if } u < 0.5, \\ 0 & \text{else,} \end{cases} \quad (3.28)$$

$$\alpha_2(u) = 1 - \alpha_1(u). \quad (3.29)$$

However, by looking at fig. 3.13 it is easy to see that this choice does not lead to perfect sampling. We have split up the u -space between the channels, but there is significant overlap in x -space. The transformation T_1 maps $[0, 0.5)$ to $[0, 1/\sqrt{2})$, and T_2 maps $[0.5, 1)$ to $[1 - 1/\sqrt{2}, 1)$. It follows that within $[1 - 1/\sqrt{2}, 1/\sqrt{2})$ the sampling is effectively uniform, while for $x < 1 - 1/\sqrt{2}$ and for $x > 1/\sqrt{2}$ the target is undersampled by a factor of two. Consequently, the unweighting efficiency is again $\eta = 0.5$, which means there is no improvement compared to before.

A small change opens up a way out for us. We slightly modify the definition of our channels to allow shift and scale transformations:

$$g'_1(x) = 2\sigma_1 x, \quad T'_1 : u \mapsto x = \sqrt{\frac{u - \rho_1}{\sigma_1}}, \quad (3.30)$$

$$g'_2(x) = 2\sigma_2(1 - x), \quad T'_2 : u \mapsto x = 1 - \sqrt{1 - \frac{u - \rho_2}{\sigma_2}}. \quad (3.31)$$

If we choose to keep the mixing weight functions, eqs. (3.28) and (3.29), the optimal choices for the shift and scale parameters are $\sigma_1 = \sigma_2 = 2$, $\rho_1 = 0$, and $\rho_2 = -1$. In figs. 3.12 and 3.13 (dotted lines), the resulting PDFs, $g'_1(x)$ and $g'_2(x)$, and the transformations, T'_1 and T'_2 , are shown, respectively. Note that T'_1 and T'_2 are non-surjective, which means that an individual channel is not able to generate all values of x . This is, however, not an issue here, since our choice of mixing weights ensures that in the combination as a multichannel any x can be reached. With the given parameters, an optimal sampler for the target function has been found, such that the unweighting efficiency reaches $\eta = 1$.

The conclusion of the above example is that phase space dependent multichannel weights are only effective if they are combined with parameterized channels, which can be optimized in conjunction with the channel weights. It should be sufficient to let the parameters correspond to simple shift and scale transformations, which allow moving probability mass to where it is needed. In the ML literature, the combination of a parameterized mixture model with a NN that learns the mixture weights and the parameters is known as mixture density networks [243]. They are typically used with Gaussian mixtures for regression and uncertainty estimation. Deep Gaussian mixture models [244, 245] provide an alternative, where Gaussian mixtures are stacked in layers.

At this point it is appropriate to point out that the event generator MADGRAPH also allows a multichannel sampler with phase space dependent weights [53, 246]. However, it uses a different definition of multichannel, which is based on single diagram enhancement. It assumes that the integrand can be written in terms of a basis of n functions,

$$f(\mathbf{u}) = \sum_{i=1}^n f_i(\mathbf{u}), \quad \text{with } f_i(\mathbf{u}) \geq 0, \quad (3.32)$$

such that each f_i can be efficiently mapped by a single mapping g_i . Accordingly, we introduce a multichannel density

$$g(\mathbf{u}) = \sum_{i=1}^n g_i(\mathbf{u}). \quad (3.33)$$

In the same way as in eq. (2.28), the integral over f can then be written as

$$\int_{\Omega} f(\mathbf{u}) d\mathbf{u} = \sum_{i=1}^n \int_{\Omega} g_i(\mathbf{u}) \frac{f_i(\mathbf{u})}{g_i(\mathbf{u})} d\mathbf{u} = \sum_{i=1}^n \int_{\Omega} \frac{f_i(\mathbf{u})}{g_i(\mathbf{u})} dG_i(\mathbf{u}). \quad (3.34)$$

The integral has been rewritten as a sum of potentially simpler integrals. Note that in contrast to the conventional multichannel approach, as introduced in section 2.3.3, we do not have to evaluate the sum over all channels for every sampled point. Instead, we only have to determine the cheaper ratio $f_i(\mathbf{u})/g_i(\mathbf{u})$. In MADGRAPH, the proposed basis for eq. (3.32) is

$$f_i = \frac{|A_i|^2}{\sum_{j=1}^n |A_j|^2} |A_{\text{tot}}|^2, \quad (3.35)$$

where A_i is the amplitude corresponding to a single Feynman diagram and $A_{\text{tot}} = \sum_{j=1}^n A_j$ is the total amplitude. Naturally, the g_i are then chosen to be approximations of the squared amplitudes.

We can identify a quantity that is somewhat similar to our multichannel weights:

$$\alpha_i(\mathbf{u}) = \frac{g_i(\mathbf{u})}{\sum_{j=1}^n g_j(\mathbf{u})}. \quad (3.36)$$

These sum to unity by definition, which can be used to write

$$f(\mathbf{u}) = \sum_{i=1}^n \alpha_i(\mathbf{u}) f(\mathbf{u}). \quad (3.37)$$

For the integral it follows that

$$\int_{\Omega} f(\mathbf{u}) d\mathbf{u} = \sum_{i=1}^n \int_{\Omega} \alpha_i(\mathbf{u}) f(\mathbf{u}) d\mathbf{u}. \quad (3.38)$$

Note that the α_i depend on the same variable as the function f unlike above, for the conventional multichannel, where the channel weights are defined in the input space, \mathbf{u} , while all other functions are evaluated in the mapped space, \mathbf{x} . Therefore, they are now defined in the same space as the target function and the channel densities, such that the difficulties described above do not apply here.

From eq. (3.34) it is not obvious how to generate samples from the multichannel distribution, specifically how to choose the probabilities for selecting the channels. Ideally, the channels would be selected by their respective contributions to the integral. This would correspond to the selection weights

$$\beta_i = \frac{\int \alpha_i(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}}{\int f(\mathbf{x}) d\mathbf{x}}. \quad (3.39)$$

In practice, these can be estimated from a finite sample of points.

The single diagram enhanced multichannel has the advantage that the weight in one channel can be evaluated independently of the others. Remember that the conventional multichannel always requires evaluating the sum over all channels. It follows that a large number of channels can be prohibitive in the conventional case and less so in the single diagram enhanced case. The main disadvantage of the method is that its performance depends entirely on the validity of the assumption eq. (3.32). Usually, the channel mappings model single Feynman diagrams squared, whose peak structures can be derived from the involved propagators. If there are large interference terms, however, this is not a good approximation of the full amplitude. As noted in ref. [247], this can for example happen for vector boson fusion-type processes, where

a conventional multichannel can be much faster. The authors propose an alternative definition of the channel weights to alleviate this problem for that type of process.

Mixtures of normalizing flows

There are known limitations of NFs with regard to the shapes of the targeted functions. These are related to functions with discrete structure, for example disconnected modes, holes where the function is zero, or discrete symmetries. When the base distribution, which is usually normal or uniform, does not have a compatible topology, it is impossible to perfectly learn the target with an NF, and good approximations require very deep NFs with prohibitively high training costs [248]. To address this, in ref. [249] the authors propose to use a mixture of several NFs, where the individual components are restricted to disjoint subsets of the target space. This results in a piecewise invertible model. As noted in [250], though, the discontinuity at the transitions between the subsets leads to training difficulties. Instead of mixing several NFs, it is also possible to use a base distribution that has the desired topology built-in. If we lack the knowledge to manually construct such a distribution, it can be found automatically by resampling a simpler distribution, e.g. a Gaussian. In ref. [251], the authors introduce a method based on learned accept/reject sampling [252], where the base distribution is resampled using rejection sampling with an acceptance probability learned by an NN. The drawback is that the resampled base distribution is intractable, since the normalization constant can only be estimated. Moreover, sampling becomes less efficient due to the rejected proposals. Another approach, which we will not pursue further in this thesis, is to sacrifice the invertibility of the flows [248], which, however, makes it impossible to compute the probability density exactly. A proposal better suited to our needs is a mixture of NFs with learnable mixture weights [250, 253]. This corresponds to a multichannel sampler, where the channels are given by NFs, and the channel weights are learned by an NN with a softmax output layer. When the channel weights and the channel mappings are optimized at the same time, such a model fulfils the requirement introduced above, i.e. the phase space dependent channel weights are combined with parameterized channels.

While in ref. [9] we only considered static channel weights, it can be expected that phase space dependent weights are able to further boost the performance. However, although mixtures of NFs are very powerful models, accurately learning a complex target distribution from the ground up can require extensive training on large datasets. As for other sampling techniques it should be beneficial to include prior knowledge.

For the sampling of differential cross-sections, several applications can be imagined. As an extension of ref. [9], one could use NFs to remap and fine-tune physics-inspired channels and extend this approach with phase space dependent channel weights. It can be expected that the variable weights allow the model to yield better approximations in the presence of non-factorizable structures and phase space cuts. Instead of remapping given channels, one could also try to use NFs pretrained on normalized versions of typical channels. These could then be further optimized during the training on the target. In both cases, after an initial warm-up phase, one can reduce the number of channels by using only those contributing significantly to the total integral. This can reduce the training costs considerably, while the model should still be flexible enough, as long as an appropriate number of channels is kept. Finally, one can also use NFs as additional channels next to physics-inspired ones to learn the residual structures. In this case, one should optimize the parameters of the physics-inspired channels together with the channel weights in order to benefit from the latter in the most effective way. It remains for future work to find out which of the proposed approaches turns out to be the most promising.

3.2 Nested Sampling

Markov Chain Monte Carlo (MCMC) methods like the Metropolis-Hastings algorithm [254, 255], and variants thereof, are popular tools for sampling with numerous applications. For example, they are a standard choice in lattice field theory [256]. While there have been attempts towards using MCMC for phase space sampling in HEP [257, 258], it never came to large-scale use. The main reason for this, besides conservatism, can be assumed to be the autocorrelation intrinsic to MCMC samples. Especially when the MCMC chain does not converge well, there can be high autocorrelation and even many identical samples. Typical HEP analyses cannot cope with this since they are designed for conventional MC methods where the samples are statistically independent. The autocorrelation can be reduced by thinning the chain, but this in turn decreases the efficiency.

Nested sampling [28, 259] is not an MCMC algorithm per se, but in implementations MCMC can be used for a crucial step of the algorithm, as is shown below. However, unlike pure MCMC algorithms, nested sampling then produces many short Markov chains instead of a single long one. Therefore, one expects only low autocorrelation and the algorithm does not get stuck at points or small regions of phase space. This makes nested sampling an interesting alternative to MCMC in the HEP context, and it seems useful to look at what properties the algorithm has and how it performs in representative problems.

This section is meant to provide some context around the article ‘Exploring phase space with nested sampling’, which presents the first application of nested sampling to the sampling of the high-dimensional phase space of particle collision events. Below, in section 3.2.1, the algorithm is introduced in more detail than the restricted format of the article allows. The publication itself is reprinted in section 3.2.2.

3.2.1 Basic idea and specifics of the implementation

The nested sampling algorithm was originally introduced in the context of Bayesian inference. A detailed introduction to the topic of Bayesian inference, as well as a comparison of Bayesian and frequentist statistics, would go beyond the scope of this thesis. Instead, a pragmatic approach is appropriate: for our purposes, we consider nested sampling as a generic integration algorithm that also generates samples from the integrand. Where necessary, terms from the respective jargon can be translated into each other. These are in particular the following:

phase space mapping	→	prior,
target function / integrand / differential cross-section	→	likelihood,
target integral / total cross-section	→	evidence.

This simple translation is already a big step towards applying nested sampling to HEP phase space sampling.

Algorithm 2 gives an overview of the nested sampling meta algorithm as applied to the calculation of scattering cross-sections. The algorithm maintains an ensemble of n_{live} live points, which get updated until a preselected termination criterion, T_C , is reached. It proceeds iteratively, and in every step the live point with the smallest value of the target function is chosen for a replacement. In our case, the target function corresponds to the differential cross-section, while in Bayesian inference it is the likelihood function. The replacement is the crucial step of the algorithm. It is required that the new point is drawn uniformly from the phase space volume under the condition that it has a higher value of $d\sigma$. The meta-algorithm does not specify how this step is to be implemented. Below, several options are discussed. The new live point is used to update the estimate of the total cross-section, σ . It is based on the value of the differential cross-section, $d\sigma_{\text{min}}$, and an estimate of the change in phase space volume, $\Delta\Theta$. The estimate of the change in volume is made on a statistical basis. It can be shown

Algorithm 2: The nested sampling meta algorithm applied to the calculation of cross-sections in particle physics phase space.

```

1 Choose a termination criterion,  $T_C \in [0, 1]$ ;
2 initialize the starting phase space volume,  $\Theta = 1$ , and the cross section integral,  $\sigma = 0$ ;
3 sample  $n_{\text{live}}$  points uniformly from the volume — the live point ensemble;
4 define an estimate of the volume compression per iteration,  $t = e^{-1/n_{\text{live}}}$ ;
5 while  $\Theta > T_C$  do
6     find the point in  $n_{\text{live}}$  with the smallest value of  $d\sigma$ ,  $d\sigma_{\text{min}}$ ;
7     replace this point with a new sample in the volume defined by the boundary
        $d\sigma > d\sigma_{\text{min}}$ ;
8     increment the estimate of the cross-section,  $\sigma = \sigma + d\sigma_{\text{min}} \Delta\Theta$ , with  $\Delta\Theta = (1 - t)\Theta$ ;
9     contract the remaining volume,  $\Theta = \Theta - \Delta\Theta$ ;
10 end
11 add an estimate of the contribution to the integral from the remaining live points,
     $\sigma = \sigma + \langle d\sigma \rangle \Delta\Theta$ , where  $\langle d\sigma \rangle$  is the average differential cross-section of the remaining
    live points;
12 return  $\sigma$ ;
```

that the compression factor t associated with discarding a live point follows a Beta($n_{\text{live}}, 1$) distribution [260]. Thus, the expectation value of $\log t$ is

$$\langle \log t \rangle = -\frac{1}{n_{\text{live}}}. \quad (3.40)$$

Based on this, the volume compression can be estimated to be

$$t = e^{-1/n_{\text{live}}}. \quad (3.41)$$

The absolute change in volume is $\Delta\Theta = (1 - t)\Theta$, where Θ is the volume before discarding the live point. Note that this is an estimate based on probability and does not equal the true change in volume. This fact should be taken into account where uncertainty estimates are concerned.

The output of the nested sampling algorithm consists of the final ensemble of live points and an estimate of the total cross-section. Furthermore, we can keep track of the discarded points, which form the ensemble of dead points. These can be used as samples from the distribution of the integrand, or posterior distribution in Bayesian terms, by assigning weights to them. The weights are given by

$$P_i = \frac{w_i d\sigma_{\text{min},i}}{\sigma}, \quad (3.42)$$

where the w_i are defined as

$$w_i = \frac{1}{2}(\Theta_{i-1} + \Theta_{i+1}). \quad (3.43)$$

If desired, the weighted dead points can be unweighted using rejection sampling.

As stated above, there are various options how to do the compression step, i.e. how to sample new live points under the likelihood constraint. Among them are variants of MCMC and rejection sampling. In the article reprinted in section 3.2.2, the POLYCHORD [261] implementation of nested sampling is used. It realizes the compression step by slice sampling [262]. Therefore, only this method is considered here. Slice sampling is a kind of Markov chain sampling. Similar to the Metropolis-Hastings algorithm, it commonly uses a step size parameter. For Metropolis-Hastings, this is a crucial parameter that needs to be tuned to find a balance between

random walk behaviour and high rejection rates of proposals. In slice sampling, however, the step size is adapted automatically in dependence of local properties of the target function. Slice sampling is based on the observation that sampling from a distribution is equivalent to uniformly sampling points from underneath its graph. Starting from a position x_0 , this can be done in the univariate case by randomly selecting a point y between zero and $f(x_0)$, where $f(x)$ is the target function. We then need to sample a new position x_1 from the slice defined by $f(x) \geq y$, which can consist of several line segments. This is the starting position for the next iteration. The x_i form a sample that, in the long run, converges to the target distribution. In the multivariate case, we can use the same procedure with multidimensional slices.

The critical step is the sampling from the slice, since the boundaries are not easily available. This is especially true in the multimodal case, where several disconnected segments can appear. A possible method is to find boundaries that enclose all segments and to use rejection sampling until a point within one of the segments has been found. POLYCHORD uses a different procedure, which is informed by the set of live points and can be more efficient. At each iteration, it chooses one of the live points as a starting point. It takes a step of initial width w in a random direction \hat{n} chosen from a probability distribution $P(\hat{n})$, generating the next starting point. This step is repeated n_{repeats} times, where n_{repeats} should be chosen such that the final point is sufficiently decorrelated from the starting point. The width can be expanded during the procedure. Furthermore, POLYCHORD uses a cluster recognition algorithm to identify modes of the target distribution, and tries to evolve points within the clusters.

The similarities between slice sampling and sampling from the iso-likelihood contours in nested sampling are obvious. Therefore, the two seem to be a natural fit. In nested sampling, additional knowledge about the target distribution is available in the form of the live point ensemble. These points are used to define the distribution $P(\hat{n})$. The points from the first $n_{\text{repeats}} - 1$ steps are called phantom points and are saved alongside the dead points. Although the phantom points are significantly correlated, they contain valuable information. For example, they can be used to further inform $P(\hat{n})$. If we do not mind the correlation, we can even combine the phantom points with the dead points to get a larger sample from the posterior distribution. More details about slice sampling can be found in the original publication, ref. [262], and in the description of the POLYCHORD algorithm, ref. [261].

So far we have assumed that the live points are drawn from a uniform distribution. However, if prior knowledge about the target is available, it can be highly beneficial to use a non-uniform prior distribution which encodes some of this knowledge. In that case, new live points get drawn from the prior distribution, which is subject to evolving likelihood constraints. A prior distribution can prevent that modes are missed, and can increase the sampling efficiency by mimicking the shape of the target. Similar to the PS mappings used for importance sampling and rejection sampling in HEP, prior distributions are implemented in the form of mappings from the unit hypercube to the parameter space of the target.

Nested sampling has some properties that distinguish it from other sampling methods. In contrast to importance sampling and rejection sampling, it is not possible to generate a fixed number of events. Based on a few tunable parameters, essentially n_{live} and n_{repeats} , the algorithm runs until the termination criterion has been reached. A choice for this criterion could be, for example, that the remaining live points are expected to contribute only some small fraction to the total cross-section. Now if less events are needed than the algorithm generates, one still has to wait for the run to finish before the chain of events can be thinned. This is because of the hierarchy of the events, where events with small values of $d\sigma$ are generated earlier than events with larger values. If, on the other hand, more events are needed, one has to increase the number of live points or combine several runs, which is equivalent. In comparison to MCMC methods, it can be noted that they are primarily sampling algorithms and do, in general, not provide good estimates of the integral. By contrast, for nested sampling the integral is the prime target, and it aims at giving reliable integral estimates even in high dimensions.

3.2.2 *Publication: Exploring phase space with Nested Sampling*

In this section, the article ‘Exploring phase space with Nested Sampling’ is presented. Like the earlier publication by my co-authors and me, ref. [9], it deals in principle with improving phase space sampling by trying to bring the distribution of drawn events closer to the target. However, instead of enhancing the phase space mappings used for importance sampling, it is proposed to replace the existing sampling approach, i.e. adaptive multichannel importance sampling, by nested sampling. The main motivation of the article is the realization that there are great similarities between sampling evidence integrals in Bayesian inference and sampling cross-sections in HEP. Since nested sampling is widely and successfully used for Bayesian inference, especially in cosmology [260], the question naturally arises to what extent this can be transferred to other applications. In the article reprinted below, this is examined for the integration of partonic cross-sections and the generation of corresponding distributions of events, which to my knowledge is the first such application of nested sampling. The scattering of high-energy gluons with 3- 4- and 5-gluon final states serves as an example. Although this example is much less complex than, for example, hadronic collisions with more diverse final states, it already contains many typical difficulties that arise in the structure of QCD matrix elements. It is therefore a good testbed, while not oversimplifying, so that the results should be qualitatively transferable to more realistic applications.

In the article, it is shown how nested sampling can be used for event generation. The implications and possibly necessary changes in established workflows are discussed. Furthermore, it is shown how nested sampling can be compared quantitatively with current sampling methods, also with regard to uncertainties. When applied to the physical example, it is shown that competitive results can be achieved even without prior knowledge, and that the method scales advantageously to higher dimensions. Based on this, various ideas for future developments are presented.

The article was first published as a preprint on ARXIV in May 2022. Subsequently it was submitted to the journal *The European Physical Journal C*. After the referees’ comments were addressed, it was finally published in August 2022. The version published in the journal is reprinted below. A copyright notice and a link to the material are provided on the first page of the article. The article is licensed under a [Creative Commons Attribution 4.0 International License](#).

Author contributions


The idea of using nested sampling for HEP PS sampling is attributable to David Yallup. Will Handley contributed his expertise on the nested sampling algorithm and the POLYCHORD implementation of it. The examples considered in the article are based on the experience of Steffen Schumann and me, especially through our earlier article, ref. [9]. We were able to reuse my implementations of the RAMBO and HAAG phase space mappings as well as my PYTHON interface to SHERPAS matrix elements for the gluon scattering processes. For use with POLYCHORD, I provided an implementation of the RAMBO mapping and an interface to the gluon scattering matrix elements in C++. All nested sampling calculations for the physics example presented in sec. 3 were done by David Yallup. The equivalent importance sampling and rejection sampling calculations were done by me. The evaluation and the presentation of the results in figures and tables were a joint effort of both of us. I contributed the implementation and evaluation of the toy example and its extension to non-uniform priors as presented in secs. 2.1 and 4.2. David Yallup added the examination of the posterior mass profile, which is shown in Fig. 3. The discussion of uncertainties for the nested sampling results in the appendix is due to David Yallup and Will Handley. All authors contributed significantly

to the writing of the article.*

*In this work, we collaborated with researchers from the astrophysics community. Therefore, in contrast to the other publications presented in this thesis, the order of the authors is not alphabetical. The respective contributions are as indicated above.



Exploring phase space with nested sampling

David Yallup^{1,a} , Timo Janßen², Steffen Schumann², Will Handley¹¹ Cavendish Laboratory and Kavli Institute for Cosmology, University of Cambridge, JJ Thomson Avenue, Cambridge CB3 0HE, UK² Institut für Theoretische Physik, Georg-August-Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, GermanyReceived: 17 May 2022 / Accepted: 23 July 2022 / Published online: 5 August 2022
© The Author(s) 2022

Abstract We present the first application of a Nested Sampling algorithm to explore the high-dimensional phase space of particle collision events. We describe the adaptation of the algorithm, designed to perform Bayesian inference computations, to the integration of partonic scattering cross sections and the generation of individual events distributed according to the corresponding squared matrix element. As a first concrete example we consider gluon scattering processes into 3-, 4- and 5-gluon final states and compare the performance with established sampling techniques. Starting from a flat prior distribution Nested Sampling outperforms the VEGAS algorithm and achieves results comparable to a dedicated multi-channel importance sampler. We outline possible approaches to combine Nested Sampling with non-flat prior distributions to further reduce the variance of integral estimates and to increase unweighting efficiencies.

1 Introduction

Realistic simulations of scattering events at particle collider experiments play an indispensable role in the analysis and interpretation of actual measurement data for example at the Large Hadron Collider (LHC) [1, 2]. A central component of such event simulations is the generation of hard scattering configurations according to a density given by the squared transition matrix element of the concrete process under consideration. This is needed both for the evaluation of corresponding cross sections, as well as the explicit generation of individual events that potentially get further processed, e.g. by attaching parton showers, invoking phenomenological models to account for the parton-to-hadron transition, and eventually, a detector simulation. To adequately address the physics needs of the LHC experiments requires the evaluation of a wide range of high-multiplicity hard processes that feature a highly non-trivial multimodal target density

that is rather costly to evaluate. The structure of the target is thereby affected by the appearance of intermediate resonances, quantum interferences, the emission of soft and/or collinear massless gauge bosons, or non-trivial phase space constraints, due to kinematic cuts on the final state particles. Dimensionality and complexity of the phase space sampling problem make the usage of numerical methods, and in particular Monte Carlo techniques, for its solution indispensable.

The most widely used approach relies on adaptive multi-channel importance sampling, see for example [3–7]. However, to achieve good performance detailed knowledge of the target distribution, i.e. the squared matrix element, is needed. To this end information about the topology of scattering amplitudes contributing to the considered process is employed in the construction of individual channels. Alternatively, and also used in combination with importance sampling phase space maps, variants of the self-adaptive VEGAS algorithm [8] are routinely applied [9–12].

An alternative approach for sampling according to a desired probability density is offered by Markov Chain Monte Carlo (MCMC) algorithms. However, in the context of phase space sampling in high energy physics these techniques attracted rather limited attention, see in particular [13, 14]. More recently a mixed kernel method combining multi-channel sampling and MCMC, dubbed (MC)³, has been presented [15]. A typical feature of such MCMC based algorithms is the potential autocorrelation of events that can affect their direct applicability in typical use case scenarios of event generators.

To meet the computing challenges posed by the upcoming and future LHC collider runs and the corresponding event simulation campaigns, improvements of the existing phase space sampling and event unweighting techniques will be crucial [16, 17]. This has sparked renewed interest in the subject, largely driven by applications of machine learning techniques, see for instance [18–36].

In this article we explore an alternative direction. We here study the application of Nested Sampling [37] as imple-

^a e-mail: dy297@cam.ac.uk (corresponding author)

mented in POLYCHORD [38] to phase space integration and event generation for high energy particle collisions. We here assume no prior knowledge about the target and investigate the ability of the algorithm to adapt to the problem. Nested Sampling has originally been proposed to perform Bayesian inference computations for high dimensional parameter spaces, providing also the evidence integral, i.e. the integral of the likelihood over the prior density. This makes it ideally suited for our purpose. In Sect. 2 we will introduce Nested Sampling as a method to perform cross section integrals and event generation, including a reliable uncertainty estimation. In Sect. 3 we will apply the method to gluon scattering to 3-, 4-, and 5-gluon final states as a benchmark for jet production at hadron colliders, thereby comparing results for total cross sections and differential distributions with established standard techniques. Evaluation of the important features of the algorithm when applied in the particle physics context is also discussed in this section. In Sect. 4 we illustrate several avenues for future research, extending the work presented here. Finally, we present our conclusions in Sect. 5.

2 Nested sampling for event generation

The central task when exploring the phase space of scattering processes in particle physics is to compute the cross section integral, σ . This requires the evaluation of the transition squared matrix element, $|\mathcal{M}|^2$, integrated over the phase space volume, Ω , where Ω is composed of all possible kinematic configurations, Φ , of the external particles. Up to some constant phase space factors this amounts to performing the integral,

$$\sigma = \int_{\Omega} d\Phi |\mathcal{M}|^2(\Phi). \quad (1)$$

In practice rather than sampling the physical phase space variables, i.e. the particles' four-momenta, it is typical to integrate over configurations, $\theta \in [0, 1]^D$, from the D -dimensional unit hypercube. Some mapping, $\Pi : [0, 1]^D \rightarrow \Omega$, is then employed to translate the sampled variables to the physical momenta. The mapping is defined as, $\Phi = \Pi(\theta)$, and the integral in Eq. (1) is written,

$$\sigma = \int_{[0,1]^D} d\theta |\mathcal{M}|^2(\Pi(\theta)) \mathcal{J}(\theta) = \int_{[0,1]^D} d\theta \mathcal{L}(\theta). \quad (2)$$

A Jacobian associated with the change of coordinates between θ and Φ has been introduced, \mathcal{J} , and then absorbed into the definition of $\mathcal{L}(\theta) = |\mathcal{M}|^2(\Pi(\theta)) \mathcal{J}(\theta)$. With no general analytic solution to the sorts of scatterings considered

at the high energy frontier, this integral must be estimated with numerical techniques. Numerical integration involves sampling from the $|\mathcal{M}|^2$ distribution in a manner that gives a convergent estimate of the true integral when the samples are summed. As a byproduct this set of samples can be used to estimate integrals of arbitrary sub-selections of the integrated phase space volume, decomposing the total cross section into differential cross section elements, $d\sigma$. Additionally these samples can be unweighted and used as pseudo-data to emulate the experimental observations of the collisions. The current state of the art techniques for performing these tasks were briefly reviewed in Sect. 1.

Importance Sampling (IS) is a Monte Carlo technique used extensively in particle physics when one needs to draw samples from a distribution with an unknown *target* probability density function, $P(\Phi)$. Importance Sampling approaches this problem by instead drawing from a known *sampling* distribution, $Q(\Phi)$ (A number of standard texts for inference give more thorough exposition of the general sampling theory used in this paper, see e.g. [39]). Samples drawn from Q are assigned a weight, $w = P(\Phi)/Q(\Phi)$, adjusting the importance of each sampled point. The performance of IS rests heavily on how well the sampling distribution can be chosen to match the target, and adaptive schemes like VEGAS are employed to refine initial proposals. It is well established that as the dimensionality and complexity of the target increase, the task of constructing a viable sampling distribution becomes increasingly challenging.

Markov Chain based approaches fundamentally differ in that they employ a local sampling distribution and define an acceptance probability with which to accept new samples. Markov Chain Monte Carlo (MCMC) algorithms are widely used in Bayesian inference. Numerical Bayesian methods have to be able to iteratively refine the prior distribution to the posterior, even in cases where the two distributions are largely disparate, making stochastic MCMC refinement an indispensable tool in many cases. This is an important conceptual point; in the particle physics problems presented in this work we are sampling from exact theoretically derived distributions. The lack of noise and a priori well known structure make methods with deterministic proposal distributions such as IS more initially appealing, however at some point increasing the complexity and dimensionality of the problem forces one to use stochastic methods. Lattice QCD calculations are a prominent example set of adjacent problems sampling from theoretical distributions that make extensive use of MCMC approaches [40]. MCMC algorithms introduce an orthogonal set of challenges to IS; a local proposal is inherently simpler to construct, however issues with exploration of multimodal target distributions and autocorrelation of samples become new challenges to address.

Nested Sampling (NS) is a well established algorithm for numerical evaluation of high dimensional integrals [37]. NS

differs from typical MCMC samplers as it is primarily an integration algorithm, hence by definition has to overcome a lot of the difficulties MCMC samplers face in multimodal problems. A recent community review of its various applications in the physical sciences, and various implementations of the algorithm has been presented in [41].

At its core NS operates by maintaining a number, n_{live} , of *live point* samples. This ensemble of live points is initially uniformly sampled from $\theta \in [0, 1]^D$ – distributed in the physical volume Ω according to the shape of the mapping Π . These live points are sorted in order of $\mathcal{L}(\theta)$ evaluated at the phase space point, and the point with the lowest \mathcal{L} , \mathcal{L}_{min} , in the population is identified. A replacement for this point is found by sampling uniformly under a hard constraint requiring, $\mathcal{L} > \mathcal{L}_{\text{min}}$. The volume enclosed by this next iteration of live points has contracted and the procedure of identifying the lowest \mathcal{L} point and replacing it is repeated. An illustration of three different stages of this iterative compression on an example two-dimensional function are shown in Fig. 1. The example function used in this case has four identical local maxima to find, practical exploration and discovery of the modes is achieved by having a sufficient ($\mathcal{O}(10)$) initial samples in the basis of attraction of each mode. This can either be achieved by brute force sampling a large number of initial samples, or by picking an initial mapping distribution that better reflects the multi-modal structure. By continually uniformly sampling from a steadily compressing volume, NS can estimate the density of points which is necessary for computing an integral as given by Eq. (1). Once the iterative procedure reaches a point where the live point ensemble occupies a predefined small fraction of the initial volume, T_C , the algorithm terminates. The fraction T_C can be characterised as the *termination criterion*. The discarded points throughout the evolution are termed *dead points* which can be joined with the remaining live points to form a representative sample of the function, that can be used to estimate the integral or to provide a random sample of events.

To estimate the integral and generate (weighted) random samples, Nested Sampling achieves this by probabilistically estimating the volume of the shell between the two outermost points as approximately $\frac{1}{n_{\text{live}}}$ of the current live volume. The volume X_j within the contour \mathcal{L}_j – defined by the point with \mathcal{L}_{min} – at iteration j may therefore be estimated as,

$$\begin{aligned}
 X_j &= \int_{\mathcal{L}(\theta) > \mathcal{L}_j} d\theta \\
 &\Rightarrow X_0 = 1, \\
 P(X_j | X_{j-1}) &= \frac{X_j^{n_{\text{live}}-1}}{n_{\text{live}} X_{j-1}^{n_{\text{live}}}} \\
 &\Rightarrow \log X_j \approx \frac{-j \pm \sqrt{j}}{n_{\text{live}}}.
 \end{aligned}$$

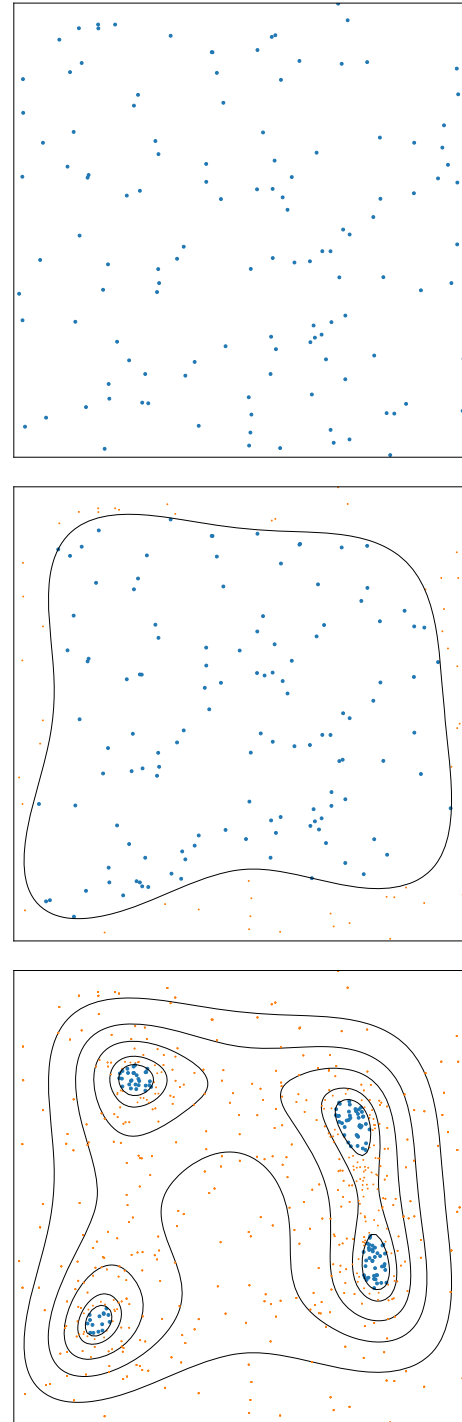


Fig. 1 Schematic of live point evolution (blue dots) in Nested Sampling, over a two-dimensional function whose logarithm is the negative Himmelblau function (contours). Points are initially drawn from the unit hypercube (top panel). The points on the lowest contours are successively deleted, causing the live points to contract around the peak(s) of the function. After sufficient compression is achieved, the dead points (orange) may be weighted to compute the volume under the surface and samples from probability distributions derived from the function

The cross section and probability weights can therefore be estimated as,

$$\begin{aligned}\sigma &= \int d\theta \mathcal{L}(\theta) = \int dX \mathcal{L}(X) \\ &\approx \sum_j \mathcal{L}_j \Delta X_j, \quad w_j \approx \frac{\Delta X_j \mathcal{L}_j}{\sigma}.\end{aligned}\quad (3)$$

Importantly, for all of the above the approximation signs indicate errors in the procedure of probabilistic volume estimation, which are fully quantifiable.

The method to sample new live points under a hard constraint can be realised in multiple ways, and this is one of the key differences in the various implementations of NS. In this work we employ the POLYCHORD implementation of Nested Sampling [38], which uses slice sampling [42] MCMC steps to evolve the live points. NS can be viewed as being an ensemble of many short Markov Chains.

Much of the development and usage of NS has focused on the problem of calculation of marginal likelihoods (or evidences) in Bayesian inference, particularly within the field of Cosmology [43–48]. We can define the Bayesian evidence, \mathcal{Z} , analogously to the particle physics cross section, σ . NS in this context evaluates the integral,

$$\mathcal{Z} = \int d\theta \mathcal{L}(\theta) \pi(\theta), \quad (4)$$

where the likelihood function, \mathcal{L} , plays a similar role to $|\mathcal{M}|^2$. In the Bayesian inference context, the phase space over which we are integrating, θ , has a measure defined by the prior distribution, $\pi(\theta)$, which without loss of generality under a suitable coordinate transformation can be taken to be uniform over the unit hypercube. Making the analogy between the evidence and the cross section explicit will allow us to apply some of the information theoretic metrics commonly used in Bayesian inference to the particle physics context [49], and provide terminology used throughout this work. Among a wide array of sampling methods for Bayesian inference, NS possesses some unique properties that enable it to successfully compute the high dimensional integral associated with Eq. (4). These properties also bear a striking similarity to the requirements one would like to have to explore particle physics phase spaces. These are briefly qualitatively described as follows:

- NS is primarily a *numerical integration method that produces posterior samples as a by product*. In this respect it is comfortably similar to Importance Sampling as the established tool in particle physics event generation. It might initially be tempting to approach the particle physics event generation task purely as a posterior sampling problem. Standard Markov Chain based sampling

tools cannot generically give good estimates of the integral, so are not suited to compute the cross section. Additionally issues with coverage of the full phase space from the resulting event samples are accounted for by default by obtaining a convergent estimate of the integral over all of the phase space.

- NS naturally *handles multimodal problems* [45,46]. The iterative compression can be augmented by inserting steps that cluster the live points periodically throughout the run. Defining subsets of live points and evolving them separately allows NS to naturally tune itself to the modality of unseen problems.
- NS requires a construction that can handle *sampling under a hard likelihood constraint* in order to perform the compression of the volume throughout the run. Hard boundaries in the physics problem, such as un-physical or deliberately cut phase space regions, manifest themselves in the sampling space as a natural extension of these constraints.
- NS is *largely self-tuning*. Usage in Bayesian inference has found that NS can be applied to a broad range of problems with little optimisation of hyper-parameters necessary [50–52]. NS can adapt to different processes in particle physics *without any prior knowledge of the underlying process needed*.

The challenge to present NS in this new context is to find an even comparison of sampling performance between NS and IS. It is typical in phase space sampling to compare the difference between the target and the sampling distribution as reducing the variation between these two distributions gives a clear metric of performance for IS. For NS there is no such global sampling distribution; the closest analogue being the prior which is then iteratively refined with local proposals to an estimate of the target. In Sect. 2.1 we attempt to compare the sampling distribution between NS and IS using a toy problem, however in the full physical gluon scattering example presented in Sect. 3 we instead focus directly on the properties of the estimated target distribution as this is the most direct equitable point of comparison.

2.1 Illustrative example

To demonstrate the capabilities of NS we apply the algorithm to an illustrative sampling problem in two dimensions. Further examples validating POLYCHORD on a number of challenging sampling toy problems are included in the original paper [38], here we present a modified version of the *Gaussian Shells* scenario. An important distinction of the phase space use case not present in typical examples is the emphasis on calculating finely binned *differential* histograms of the total integral. As a comparison to NS, we sample the same problem with a method that is well-known in high energy

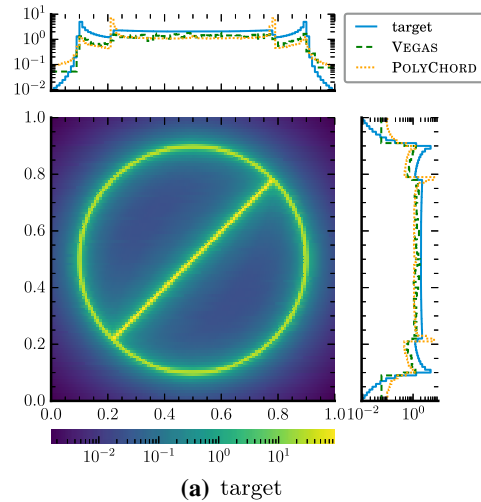
physics – adaptive Importance Sampling (IS), realised using the VEGAS algorithm.

For our toy example we introduce a “stop sign” target density, whose unnormalised distribution is defined by

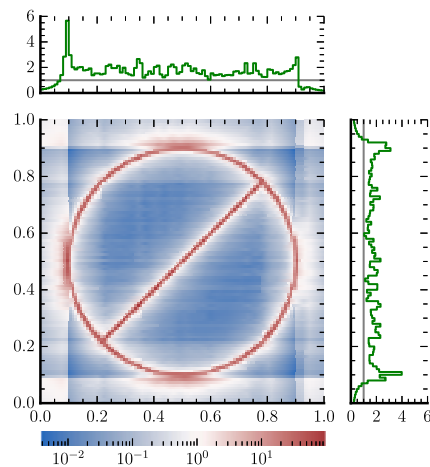
$$f(x, y) = \frac{1}{2\pi^2} \frac{\Delta r}{\left(\sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0\right)^2 + (\Delta r)^2} \cdot \frac{1}{\sqrt{(x-x_0)^2 + (y-y_0)^2}} + \frac{1}{2\pi r_0} \frac{\Delta r}{((y-y_0) - (x-x_0))^2 + (\Delta r)^2} \cdot \Theta\left(r_0 - \sqrt{(x-x_0)^2 + (y-y_0)^2}\right), \quad (5)$$

where $\Theta(x)$ is the Heaviside function. It is the sum of a ring and a line segment, both with a (truncated) Cauchy profile. The ring is centred at $(x_0, y_0) = (0.5, 0.5)$ and has a radius of $r_0 = 0.4$. The line segment is located in the inner part of the ring and runs through the entire diameter. We set the width of the Cauchy profile to $\Delta r = 0.002$. This distribution can be seen as an example of a target where it makes sense to tackle the sampling problem with a multi-channel distribution. One channel could be chosen to sample the ring in polar coordinates and one to sample the line segment in Cartesian coordinates. However, here we deliberately use VEGAS as a single channel in order to highlight the limitations of the algorithm. From the perspective of a single channel, there is no coordinate system to factorise the target distribution. That poses a serious problem for VEGAS, as it uses a factorised sampling distribution where the variables are sampled individually. Both algorithms are given zero prior knowledge of the target, thus starting with a uniform prior distribution.

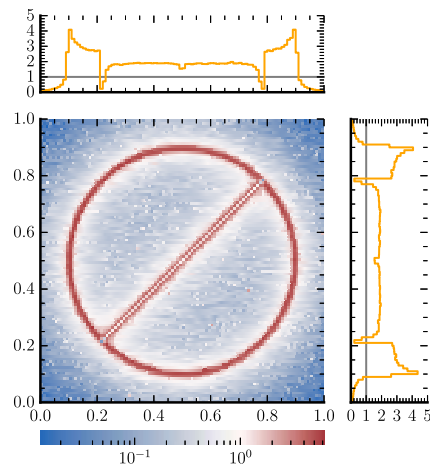
Our VEGAS grid has 200 bins per dimension. We train it over 10 iterations where we draw 30k points from the current VEGAS mapping and adapt the grid to the data. The distribution defined by the resulting grid is then used for IS without further adaptation. This corresponds to the typical use in an event generator, where there is first an integration phase in which, among other things, VEGAS is adapted, followed by a non-adaptive event generation phase. We note that VEGAS gets an advantage in this example comparison as we do not include the target evaluations from the training into the counting. However, it should be borne in mind that in a realistic application with a large number of events to be generated, the costs for training are comparatively low. For NS we use POLYCHORD with a number of live points $n_{\text{live}} = 1000$ and a chain length $n_{\text{repeats}} = 4$, more complete detail of POLYCHORD settings and their implication are given in Sect. 3.1. Figure 2a shows the bivariate target distribution along with the marginal x and y distributions of the target, VEGAS and POLYCHORD. For this plot (as well as for Fig. 2c) we merged 70 independent runs of POLYCHORD to get a better visual



(a) target



(b) ratio target/VEGAS sampling density



(c) ratio target/POLYCHORD sampling density

Fig. 2 A two-dimensional toy example: **a** Histogram of the target function along with the marginal sampling distributions of VEGAS and POLYCHORD. **b** Ratio of the target function and the probability density function of VEGAS. **c** Ratio of the target density to the sampling density of POLYCHORD

representation due to the larger sample size. It can be seen that both algorithms reproduce the marginal distributions reasonably well. There is some mismatch at the boundaries for VEGAS. This can be explained by the fact that VEGAS, as a variance-reduction method, focuses on the high-probability regions, where it puts many bins, and uses only few bins for the comparably flat low-probability regions. As a result, the bins next to the boundaries are very wide and overestimate the tails. POLYCHORD also oversamples the tails, reflecting the fact that in this example the prior is drastically different from the posterior, meaning the initial phase of prior sampling in POLYCHORD is very inefficient. In addition it puts too many points where the ring and the line segment join, which is where we find the highest values of the target function. This is not a generic feature of NS at the termination of the algorithm, rather it reflects the nature of having two intersecting sweeping degenerate modes in the problem, a rather unlikely scenario in any physical integral.

Figure 2b shows the ratio between the target distribution and the sampling distribution of VEGAS, representing the IS weights. It can be seen that the marginals of the ratio are relatively flat, with values between 0.1 and 5.7. However, in two dimensions the ratio reaches values up to 1×10^{-2} . By comparing Fig. 2a and b, paying particular attention to the very similar ranges of function values, it can be deduced that VEGAS almost completely misses to learn the structure of the target. It tries to represent the peak structure from the ring and the line segment by an enclosing square with nearly uniform probability distribution.

The same kind of plot is shown in Fig. 2c for the POLYCHORD data. NS does not strictly define a sampling distribution, however a proxy for this can be visualised by plotting the density of posterior samples. Here the values of the ratio are much smaller, between 1×10^{-2} and 7. POLYCHORD produces a flatter ratio function than VEGAS while not introducing additional artifacts that are not present in the original function. The smallest/largest values of the ratio are found in the same regions as the smallest/largest values of the target function, implying that POLYCHORD tends to overestimate the tails and to underestimate the peaks. This can be most clearly explained by examining the profile of where posterior mass is distributed throughout a run, an important diagnostic tool for NS runs [53]. It is shown in Fig. 3, where the algorithm runs from left to right; starting with the entire prior volume remaining enclosed by the live points, $\log X = 0$, and running to termination, when the live points contain a vanishingly small remaining prior volume. The posterior mass profile, shown in blue, is the analogue to the sampling density in VEGAS. To contextualise this against the target function, a profile of the log-likelihood of the lowest live point in the live point ensemble is similarly shown as a function of the

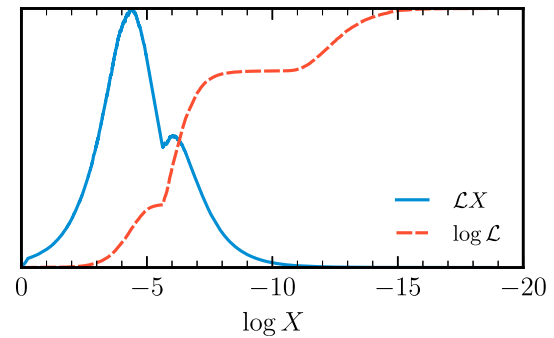


Fig. 3 Likelihood ($\log \mathcal{L}$) and posterior mass ($\mathcal{L}X$) profiles for a run of POLYCHORD on the example target density. The x -axis tracks the prior volume remaining as the run progresses, with $\log X = 0$ corresponding to the start of the run, with the algorithm compressing the volume from left to right, where the run terminates

remaining prior volume, X . Nested Sampling can be motivated as a *likelihood scanner*, sampling from monotonically increasing likelihood shells. These two profiles indicate some features of this problem, firstly a *phase transition* is visible in the posterior mass profile. This occurs when the degenerate peak of the ring structure is reached, the likelihood profile reaches a plateau where the iterations kill off the degenerate points at the peak of the ring, before proceeding to scan up the remaining line segment feature. An effective second plateau is found when the peak of the line segment is reached, with a final small detail being the superposition of the ring likelihood on the line segment. Once the live points are all occupying the extrema of the line segment, there is a sufficiently small prior volume remaining that the algorithm terminates. The majority of the posterior mass, and hence sampling density is distributed around the points where the two peaks are ascended. This reflects the stark contrast between the prior initial sampling density and the target, the samples are naturally distributed where the most information is needed to effectively compress the prior to the posterior.

We compare the efficiencies of the two algorithms for the generation of equal-weight events in Table 1. It shows that POLYCHORD achieves an overall efficiency of $\epsilon = 0.0113(90)$ which is almost three times as high as the efficiency of VEGAS. While for VEGAS the overall efficiency ϵ is identical to the unweighting efficiency ϵ_{uw} , determined by the ratio of the average event weight over the maximal weight in the sample, for POLYCHORD we also have to take the slice sampling efficiency ϵ_{ss} into account, which results from the thinning of the Markov Chain in the slice sampling step. Here, the total efficiency $\epsilon = \epsilon_{ss}\epsilon_{uw}$ is dominated by the slice sampling efficiency. We point out that it is in the nature of the NS algorithm that the sample size is not deterministic. However, the variance is not very large and it is easily possible to merge several NS runs to obtain a larger sample.

Table 1 Comparison of VEGAS and NS for the toy example in terms of size of event samples produced. $N_{\mathcal{L}}$ gives the number of target evaluations, N_W the number of weighted events and N_{equal} the derived number of equal weight events. A MC slice sampling efficiency, ϵ_{ss} , is listed for

Algorithm	$N_{\mathcal{L}}$	ϵ_{ss}	N_W	ϵ_{uw}	N_{equal}	ϵ
VEGAS	300,000		300,000	0.004(2)	1267 (460)	0.004 (2)
NS	308,755 (17505)	0.041 (3)	12,669 (147)	0.273 (7)	3462 (96)	0.0113 (9)

Table 2 Comparison of integrals calculated in the toy example with VEGAS and NS, along with the respective uncertainties

Algorithm	I	$\Delta\sigma_{\text{tot}}$	Δ_w	Δ_{MC}
VEGAS	1.71	0.02		0.02
NS	1.65	0.05	0.04	0.02

Table 2 shows the integral estimates along with the corresponding uncertainty measures. While the pure Monte Carlo errors are of the same size for both algorithms, there is an additional uncertainty for NS. It carries an uncertainty on the weights of the sampled points, listed as Δ_w . This arises due to the nature of NS using the volume enclosed by the live points at each iteration to estimate the volume of the likelihood shell. The variance in this volume estimate can be sampled, which is reflected as a sample of alternative weights for each dead point in the sample. Summing up these alternative weight samples gives a spread of predictions for the total integral estimate, and the standard deviation of these is quoted as Δ_w . This additional uncertainty compounds the familiar statistical uncertainty, listed as Δ_{MC} for all calculations. In Appendix A, we present the procedure needed to combine the two NS uncertainties to quote a total uncertainty, $\Delta\sigma_{\text{tot}}$, as naively adding in quadrature will overestimate the true error.

3 Application to gluon scattering

As a first application and benchmark for the Nested Sampling algorithm, we consider partonic gluon scattering processes into three-, four- and five-gluon final states at fixed centre-of-mass energies of $\sqrt{s} = 1$ TeV. These channels have a complicated phase space structure that is similar to processes with quarks or jets, while the corresponding amplitude expressions are rather straightforward to generate. The fixed initial and final states allow us to focus on the underlying sampling problem. For regularisation we apply cuts to the invariant masses of all pairs of final state gluons such that $m_{ij} > 30$ GeV and on the transverse momenta of all final state gluons such that $p_{T,i} > 30$ GeV. The renormalisation scale is fixed to $\mu_R = \sqrt{s}$. The matrix elements are calculated

NS. A total, ϵ , and unweighting, ϵ_{uw} , efficiency are listed for both algorithms. We report the mean and standard deviation of ten independent runs of the respective algorithm

using a custom interface between POLYCHORD and the matrix element generator AMEGIC [5] within the SHERPA event generator framework [54]. Three established methods are used to provide benchmarks to compare NS to. Principle comparison is drawn to the HAAG sampler, optimised for QCD antenna structures [55], illustrating the exploration of phase space with the best a priori knowledge of the underlying physics included. It uses a cut-off parameter of $s_0 = 900$ GeV². Alongside this, two algorithms that will input no prior knowledge of the phase space, i.e. the integrand, are used; adaptive importance sampling as realised in the VEGAS algorithm [8] and a flat uniform sampler realised using the RAMBO algorithm [56,57]. VEGAS remaps the variables of the RAMBO parametrisation using 50, 70, 200 bins per dimension for the three-, four-, and five-gluon case, respectively. The grid is trained in 10 iterations using 100k training points each. Note, the dimensionality of the phase space for n -gluon production is $D = 3n - 4$, where total four-momentum conservation and on-shell conditions for the external particles are implicit.

As a first attempt to establish NS in this context, we treat the task of estimating the total and differential cross sections of the three processes *starting with no prior knowledge* of the underlying phase space distribution. For the purposes of running POLYCHORD we provide the flat RAMBO sampler as the prior, and the likelihood function provided is the squared matrix element. In contrast to HAAG, POLYCHORD performs the integration without any decomposition into channels, removing the need for any multichannel mapping. NS is a flexible procedure, and the objective of the algorithm can be modified to perform a variety of tasks, a recent example has presented NS for computation of small p -values in the particle physics context [58]. To establish NS for the task of phase space integration in this study, a standard usage of POLYCHORD is employed, mostly following default values used commonly in Bayesian inference problems.

The discussion of the application of NS to gluon-scattering processes is split into four parts. Firstly, the hyperparameters and general setup of POLYCHORD are explained in Sect. 3.1. In Sect. 3.2 a first validation of NS performing the core tasks of (differential) cross-section estimation from weighted events – against the HAAG algorithm – is presented. In Sect. 3.3 further information is given to contextualise the computational efficiency of NS against the alter-

Table 3 POLYCHORD hyperparameters used for this analysis, parameters not listed follow the POLYCHORD defaults

Parameter	POLYCHORD name	Value	Description
Number of dimensions	n_{dim}	[5, 8, 11]	Dimension of sampling space
Number of live points	n_{live}	10,000	Resolution of the algorithm
Number of repeats	n_{rep}	$n_{\text{dim}} \times 2$	Length of Markov chains
Number of prior samples	n_{prior}	n_{live}	Number of initial samples from prior
Boost posterior		n_{rep}	Write out maximum number of posterior samples

native established tools for these tasks. Finally a consideration of unweighted event generation with NS is presented in Sect. 3.4.

3.1 POLYCHORD hyperparameters

The hyperparameters chosen to steer POLYCHORD are listed in Table 3. These represent a typical set of choices for a high resolution run with the objective of producing a large number of posterior samples. The number of live points is one of the parameters that is most free to tune, being effectively the resolution of the algorithm. Typically n_{live} larger than $\mathcal{O}(1000)$ gives diminishing returns on accuracy, Bayesian inference usage in particle physics has previously employed $n_{\text{live}} = 4000$ [59] to provide some context for the choice made in this work. The particular event generation use case, partitioning the integral into arbitrarily small divisions (differential cross sections), logically favours a large n_{live} (resolution). The number of repeats is a parameter that controls the length of the slice sampling chains, the value chosen is the recommended default for reliable posterior sampling, whereas $n_{\text{rep}} = n_{\text{dim}} \times 5$ is recommended for evidence (total integral) estimation. As this study aims to cover both differential and total cross sections, the smaller value is favoured as there is a strong limit on the overall efficiency imposed by how many samples are needed to decorrelate the Markov Chains.

An important point to note is in how POLYCHORD treats unphysical values of the phase space variables, e.g. if they fall outside the fiducial phase space defined by cuts on the particle momenta. This is not an explicit hyperparameter of POLYCHORD, rather how the algorithm treats points with zero likelihood. In both the established approaches and in POLYCHORD the sampling is performed in the unit hypercube, which is then translated to the physical variables which can be evaluated for consistency and rejected if they are not phys-

ically valid. One of the strengths of NS is that the default behavior is to consider points which return zero likelihood¹ as being excluded at the prior level. During the initial prior sampling phase, unphysical points are set to log-zero and the sampling proceeds until n_{prior} initial physical samples have been obtained. Provided each connected physical region contains some live points after this initial phase, the iterative phase of MCMC sampling will explore up to the unphysical boundary. This effect necessitates a correction factor to be applied to the integral, derived as the ratio of total initial prior samples to the physically valid prior samples. In practice the correction factor is found in the `prior_info` file written out by POLYCHORD. An uncertainty on this correction can be derived from order statistics [60], however it was found to be negligibly small for the purposes of this study so is not included.

Another standout choice of hyperparameter is the chosen value of n_{prior} . The number of prior samples is an important hyperparameter that would typically be set to some larger multiple of n_{live} in a Bayesian inference context, $n_{\text{prior}} = 10 \times n_{\text{live}}$ would be considered sensible for a broad range of tasks. For the purpose of generating weighted events, using a larger value would generally be advantageous, however increasing n_{prior} will strongly decrease the efficiency in generating *unweighted* events. As the goal is to construct a generator taking an uninformed prior all the way through to unweighted events, the default value listed is used. However it is notable that this is a particular feature of starting from an uninformed prior, if more knowledge were to be included in the prior then a longer phase of prior sampling becomes advantageous. The final parameter noted, the factor by which to boost posterior samples, has no effect on POLYCHORD at runtime. Setting this to be equal to the number of repeats simply writes out the maximum number of dead points, hence is needed in this scenario. All plots and tables in the remainder of this section are composed of one single run of POLYCHORD with these settings, with the additional entries in Table 4 demonstrating a join of ten such runs.

3.2 Exploration and integrals

Before examining the performance of NS in detail, it is first important to validate that the technique is capable of fully exploring particle physics phase spaces in these chosen examples. The key test to validate this is to compare if various differential cross sections calculated with NS are statistically consistent with the established techniques. To do this, a single NS and HAAG sample of weighted events is produced, using approximately similar levels of computational

¹ Since POLYCHORD operates in log space, to avoid the infinity associated with $\log(0)$, log-zero is defined as a settable parameter. By default this is chosen to -1×10^{-25} .

Table 4 Comparison of integrals calculated for the three-, four- and five-gluon processes using RAMBO, VEGAS, NS and HAAG, along with the respective uncertainties

Process	Algorithm	σ	$\Delta\sigma_{\text{tot}}$	Δ_w	Δ_{MC}
3-jet	RAMBO	24.580	0.191		0.191
	VEGAS	24.807	0.017		0.017
	NS	24.669	0.467	0.484	0.100
	NS ($\times 10$)	24.888	0.145	0.150	0.030
	HAAG	24.840	0.017		0.017
4-jet	RAMBO	9.876	0.107		0.107
	VEGAS	9.849	0.009		0.009
	NS	9.837	0.194	0.196	0.036
	NS ($\times 10$)	9.778	0.064	0.066	0.011
	HAAG	9.853	0.006		0.006
5-jet	RAMBO	2.644	0.024		0.024
	VEGAS	2.680	0.003		0.003
	NS	2.612	0.051	0.048	0.009
	NS ($\times 10$)	2.667	0.017	0.017	0.003
	HAAG	2.685	0.001		0.001

overhead (more detail on this is given in Sect. 3.3). Both sets of weighted events are analysed using the default MC_JETS RIVET routine [61]. RIVET produces binned differential cross sections as functions of various physical observables of the outgoing gluons. For each process, the total cross section for the NS sample is normalised to the HAAG sample, and a range of fine grained differential cross sections is calculated using both algorithms covering the following observables; $\eta_i, y_i, p_{T,i}, \Delta\phi_{ij}, m_{ij}, \Delta R_{ij}, \Delta\eta_{ij}$, where $i \neq j$ label the final state jets, reconstructed using the anti- k_T algorithm [62] with a radius parameter of $R = 0.4$ and $p_T > 30$ GeV. The normalised difference between the NS and HAAG differential cross section in each bin can be computed as,

$$\chi = \frac{d\sigma_{\text{HAAG}} - d\sigma_{\text{NS}}}{\sqrt{\Delta_{\text{HAAG}}^2 + \Delta_{\text{NS}}^2}}, \tag{6}$$

in effect this is the differences between the two algorithms normalised by the combined standard deviation. By summing up this χ deviation across all the available bins in each process, a test to see if the two algorithms are convergent within their quoted uncertainties can be performed. Since over 500 bins are populated and considered in each process, it is expected that the rate of these χ deviations should be approximately normally distributed. This indeed appears to hold, and these summed density estimates across all observables are shown in Fig. 4, alongside an overlaid normal distribution with mean zero and variance one, $\mathcal{N}(0, 1)$, to illustrate the expected outcome. Two example variables that were used to build this global deviation are also shown; the leading jet

p_T in Fig. 5a and ΔR_{12} , the distance of the two leading jets in the (η, ϕ) plane, in Fig. 5b.

The composition of the quoted uncertainty for the two algorithms differs, demonstrating an important feature of an NS calculation. For HAAG, and IS in general, it is conventional to quote the uncertainty as the standard error from the effective number of fills in a bin. Nested Sampling on the other hand introduces an uncertainty on the weights used to fill the histograms themselves, effectively giving rise to multiple weight histories that must be sampled to derive the correct uncertainty on the NS calculation. Details on this calculation are supplied in Appendix A. In summary the alternative weight histories give an overlapping measure of the statistical uncertainty, so this effect must be accounted for in situ alongside taking the standard deviation of the weight histories. To contextualise this, the middle panels in Fig. 5 show the correct combined uncertainty (using the recipe from Appendix A) as a grey band, against the bands derived from the standard error of each individual algorithm (henceforth Δ_{MC}) as dashed lines, and the complete NS error treatment as a dotted line. The standard error (dashed) NS band in these panels is a naive estimation of the full NS uncertainty (dotted), however this illustrates an important point; at the level of fine grained differential observables the NS uncertainty is dominated by statistics and is hence reducible as one would expect by repeated runs. Based on the example observables we can initially conclude that whilst both algorithms appear compatible, when using weighted events NS generally has a larger uncertainty than HAAG across most of the range (given a roughly equivalent computational overhead). However, further inspection of the resulting unweighted event samples

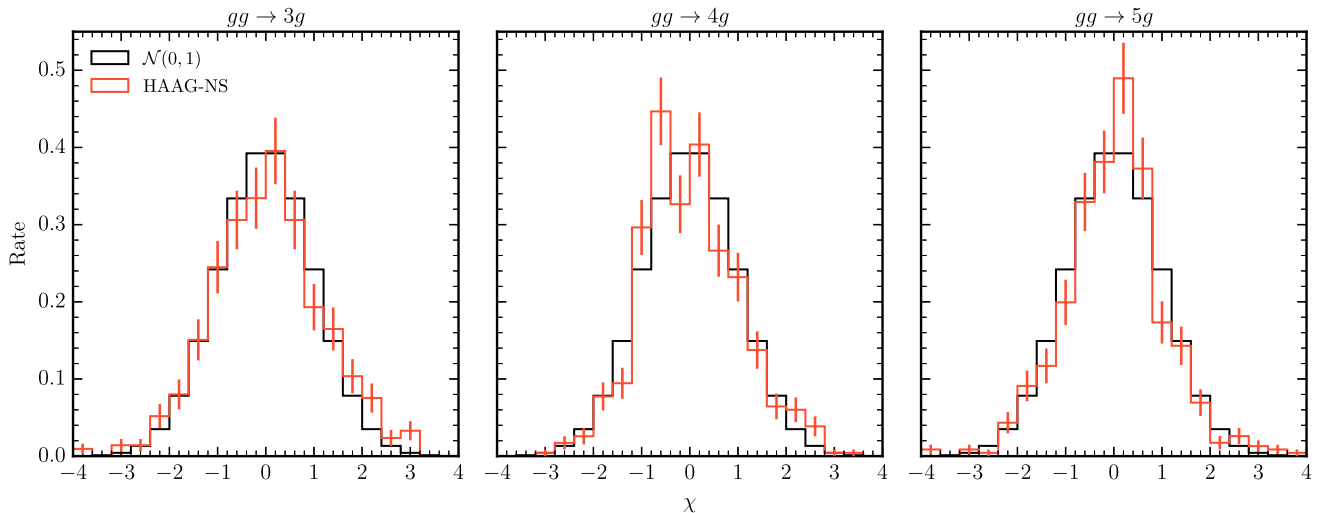


Fig. 4 Global rate of occurrence of per bin deviation, χ , between HAAG and NS, for each considered scattering process. A normally distributed equivalent deviation rate is shown for comparison

derived from these weighted samples in the remaining sections reveals a more competitive picture between the two algorithms.

The estimates of the total cross sections, derived from the sum of weighted samples, provided in Table 4, give an alternative validation that NS is sufficiently exploring the phase space by ensuring that compatible estimates of the cross sections are produced between all the methods reviewed in this study. The central estimates of the total cross sections are generally consistent within the established error sources for all calculations considered. In this table the components of the error calculation for NS are listed separately; Δ_w being the standard deviation resulting from the alternative weight histories and Δ_{MC} being the standard error naively taken from the mean of the alternative NS weights. In contrast to the differential observables, the naive counting uncertainty is small so has negligible effect at the level of total cross sections. In summary, for a total cross section the spread of alternative weight histories gives a rough estimate of the total error, whereas for a fine grained differential cross section the standard error dominates. The way to correctly account for the effect of counting statistics within the weight histories is given in Appendix A.

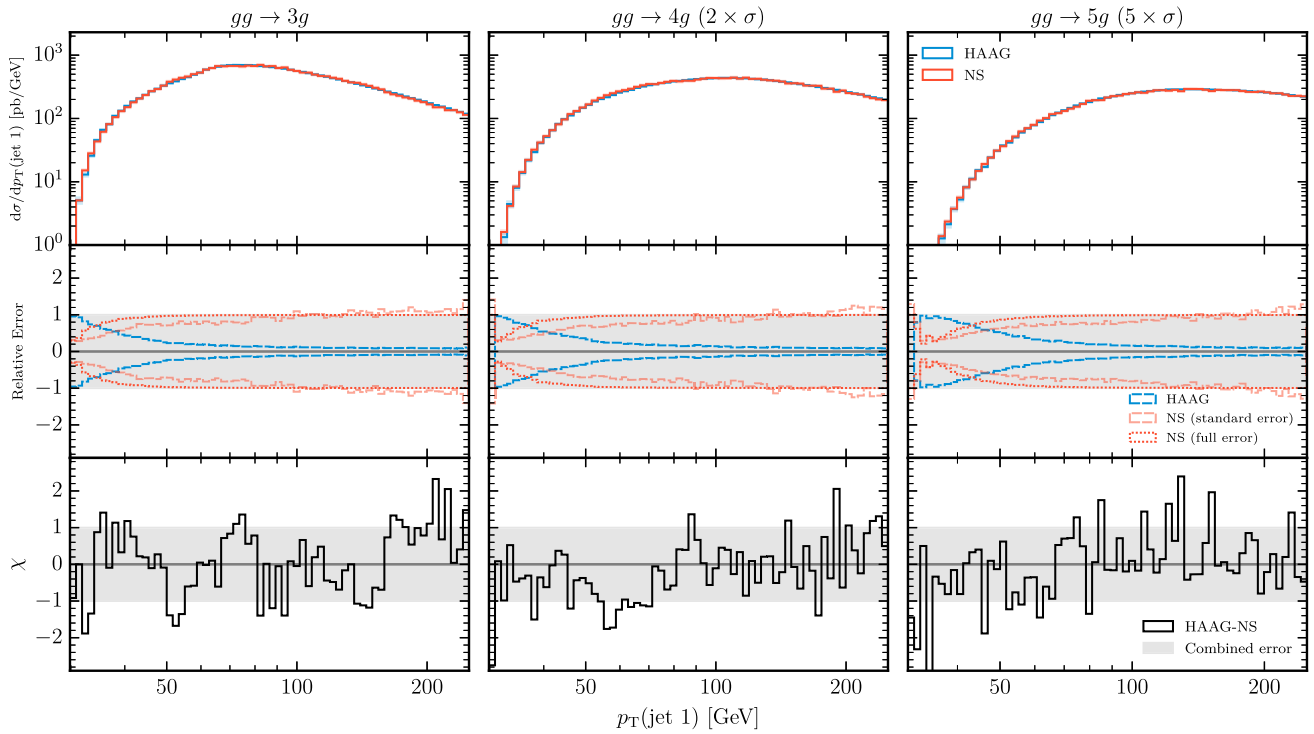
Repeated runs of NS will reduce these uncertainties. The `anesthetic` package [63] is used to analyse the NS runs throughout this paper, and contains a utility to join samples. Once samples are joined consistently into a larger sample, the uncertainties can be derived as already detailed. The result of joining 10 equivalent NS runs with the previously motivated hyperparameters is also listed in Table 4. Joining 10 runs affects the $\Delta\sigma_{tot}$ for NS in two ways; reducing the spread of weighted sums composing Δ_w (i.e. reducing Δ_{MC}), and reducing the variance of distribution for each weight itself

(i.e. the part of Δ_w that does not overlap with Δ_{MC}). The former is reduced by simply having an increased size of samples produced, increasing the number of effective fills by a factor of ~ 10 in this case, with the latter reduced due to the increased effective number of live points used for the volume estimation.

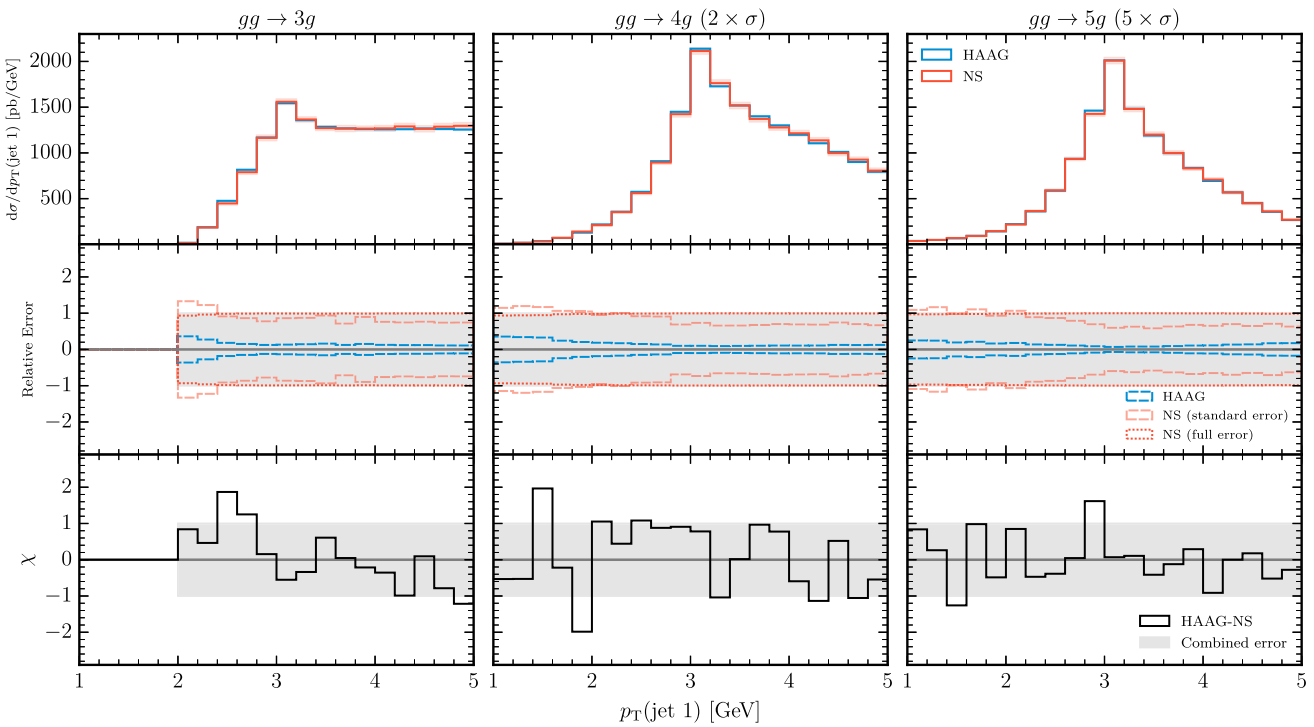
3.3 Efficiency of event generation

An example particle physics workflow on this gluon scattering problem would be to take HAAG as an initial mapping of the phase space (effectively representing the best prior knowledge of the problem), and using VEGAS to refine the proposal distribution to optimally efficiently generate weighted events. Of the three existing tools presented in this study for comparison (HAAG, RAMBO, and VEGAS), NS bears most similarity to VEGAS, in that both algorithms learn the structure of the target integrand. To this end an atypical usage of VEGAS is employed, testing how well VEGAS could learn a proposal distribution from an uninformed starting point (RAMBO). This is equivalent to how NS was employed, starting from an uninformed prior (RAMBO) and generating posterior samples via Nested Sampling. It was motivated so far that roughly similar computational cost was used for the previous convergence checks, and that the hyperparameters of POLYCHORD were chosen to emphasise efficient generation of unweighted events. In what follows, we analyse more precisely this key issue of computational efficiency.

The statistics from a single run of the four algorithms for the three selected processes is listed in Table 5. NS is non deterministic in terms of number of matrix element evaluations ($N_{\mathcal{L}}$), instead terminating from a pre determined convergence criterion of the integral. HAAG, VEGAS, RAMBO



(a) Differential cross section binned as a function of the leading jet transverse momentum, for the three considered processes.



(b) Differential cross section binned as a function of the separation of the leading two jets in the (η, ϕ) plane, *i.e.* $\Delta R_{12} = \sqrt{\Delta\eta_{12}^2 + \Delta\phi_{12}^2}$, for the three considered processes.

Fig. 5 Two example physical differential observables computed with weighted events using the HAAG and NS algorithms. The top panels show the physical distributions, the middle panels display the relative component error sources, and the bottom panel displays the normalised

deviation. The deviation plot has been normalised such that $\chi = 1$ corresponds to an expected 1σ deviation of a Gaussian distribution. Note that for illustrative purposes the cross sections for the four- and five-gluon processes have been scaled by global factors

Table 5 Comparison of the four algorithms for the three processes in terms of size of event samples produces. $N_{\mathcal{L}}$ gives the number of matrix element evaluations, N_W the number of weighted events, $N_{W,\text{eff}}$ the effective number of weighted events and N_{equal} the derived number

of equal-weight events. A MC slice sampling efficiency, ϵ_{ss} , is listed for NS. A total, ϵ , and an unweighting, ϵ_{uw} , efficiency are listed for all algorithms

Process	Algorithm	$N_{\mathcal{L}} (\times 10^6)$	ϵ_{ss}	$N_W (\times 10^6)$	ϵ_{uw}	$N_{\text{equal}} (\times 10^6)$	ϵ
3-jet	RAMBO	10.00		10.00	0.0001	0.001	0.0001
	VEGAS	10.00		10.00	0.02	0.20	0.02
	NS	6.43	0.03	0.17	0.37	0.06	0.01
	HAAG	10.00		10.00	0.03	0.29	0.03
4-jet	RAMBO	10.00		10.00	0.00003	0.0003	0.00003
	VEGAS	10.00		10.00	0.005	0.049	0.005
	NS	7.94	0.02	0.19	0.43	0.08	0.01
	HAAG	10.00		10.00	0.02	0.23	0.02
5-jet	RAMBO	10.00		10.00	0.00004	0.0004	0.00004
	VEGAS	10.00		10.00	0.001	0.013	0.001
	NS	9.17	0.02	0.19	0.44	0.08	0.01
	HAAG	10.00		10.00	0.03	0.25	0.03

are all used to generate exactly 10M weighted events. The chosen POLYCHORD hyperparameters roughly align the NS method with the other three in terms of computational cost. One striking difference comes from the Markov Chain nature of NS. Default usage only retains a fraction of the total \mathcal{L} evaluations, inversely proportional to n_{rep} . This results in a smaller number of retained weighted events, N_W , than the number of \mathcal{L} evaluations, $N_{\mathcal{L}}$, for NS. However the retained weighted events by construction match the underlying distribution much closer than the other methods, resulting in a higher unweighting efficiency, ϵ_{uw} , for the NS sample. Exact equal-weight unweighting can be achieved by accepting events with a probability proportional to the share of the sample weight they carry, this operation is performed for all samples of weighted events and the number of retained events is quoted as N_{equal} . NS as an unweighted event generator has some additional complexity due to the uncertainty in the weights themselves, this is given more attention in Sect. 3.4.

Due to differences in $N_{\mathcal{L}}$ between NS and the other methods, it is most effective to compare the total efficiency in producing unweighted events, $\epsilon = N_{\text{equal}}/N_{\mathcal{L}}$. RAMBO as the baseline illustrates the performance one would expect, inputting no prior knowledge and not adapting to any acquired knowledge. As such RAMBO yields a tiny ϵ . HAAG represents the performance using the best state of prior knowledge but without any adaptation, in these tests this represents the best attainable ϵ . VEGAS and NS start from a similar point, both using RAMBO as an uninformed state of prior knowledge, but adapting to better approximate the phase space distribution as information is acquired. VEGAS starts with a higher efficiency than NS for the 3-gluon process, but the VEGAS efficiency drops by approximately an

order of magnitude as the dimensionality of phase space is increased to the 5-gluon process. NS maintains a consistent efficiency of approximately a percent, competitive with the consistent approximately three percent efficiency obtained by HAAG.

As the key point of comparison for this issue is the efficiency, ϵ , this is highlighted with an additional visualisation in Fig. 6. The scaling behavior of the efficiency of each algorithm as a function of the number of outgoing gluons (corresponding to an increase in phase space dimensionality) is plotted for NS, HAAG and VEGAS. From the same starting point, NS and VEGAS can both learn a representation of the phase space, and do so in a way that yields a comparable efficiency to the static best available prior knowledge in HAAG. As the dimensionality of the space increases it appears that VEGAS starts to suffer in how accurately it can learn the mapping, however NS is still able to learn the mapping in a consistently efficient manner.

3.4 Unweighted event generation

The fact that NS leads to a set of alternative weight histories poses a technical challenge in operating as a generator of unweighted events in the expected manner. Exact unweighting, compressing the weighted sample to strictly equally weighted events leads to a different set of events being accepted for each weight history. Representative yields of unweighted events can be calculated as shown in Table 5 using the mean weight for each event, but the resulting differential distributions will underestimate the uncertainty if this is quoted simply as the standard error in the bin, as described in Appendix A. The correct uncertainty recipe can be propagated through naively, by separately unweight-

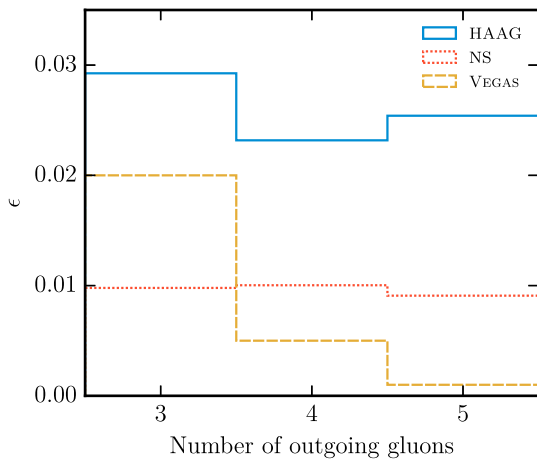


Fig. 6 Visualisation of the efficiencies listed in Table 5

ing each weight history, however this requires saving as many event samples as required weight variations. Partial unweighting is commonly used in HEP event generation to allow a slight deviation from strict unit weights, to increase efficiency in practical settings. A modification to the partial unweighting procedure could be used to propagate the spread of weights to variations around accepted, approximate unit weight, events.

To conclude the exploration of the properties of NS as a generator for particle physics, a representative physical distribution calculated from a sample of exact unit-weight events is shown in Fig. 7. This sample is derived from the same weighted sample described in Table 5 and previously presented as a weighted event sample in Fig. 5a. The full set of NS variation weights is used to calculate the mean weight for each event, which is used to unweight the sample, for the chosen observable this is a very reasonable approximation as the fine binning means the standard error is the dominant uncertainty. The range of the leading jet transverse momenta has been extended into the tail of this distribution by modifying the default RIVET routine. This distribution largely reflects the information about the total efficiency previously illustrated in Fig. 6, projected onto a familiar differential observable. The total efficiency, ϵ , was noted as being approximately one percent from NS, compared to approximately three percent from HAAG across all processes. If the total number of matrix element evaluations, $N_{\mathcal{L}}$, were to be made equal across all algorithms and processes, the performance would be further consistent.

4 Future research directions

Throughout Sect. 3, the performance of Nested Sampling in the context of particle physics phase space sampling and event generation was presented. A single choice of hyperpa-

rameters was made, effectively performing a single NS run as an entire end-to-end event generator; starting from zero knowledge of the phase space all the way through to generating unweighted events. Simplifying the potential options of NS to a single version of the algorithm was a deliberate choice to more clearly illustrate the performance of NS in this new context, using the same settings for multiple tasks gives multiple orthogonal views on how the algorithm performs. However this was a limiting choice, NS has a number of variants and applications that could more effectively be tuned to a subset of the tasks presented. Some of the possible simple alterations – such as increasing n_{prior} to improve weighted event generation at the expense of unweighting efficiency – were motivated already in this paper. In this section we outline four broad topics that extend the workflow presented here, bringing together further ideas from the worlds of Nested Sampling and phase space exploration.

4.1 Physics challenges in event generation

The physical processes studied in this work, up to 5-gluon scattering problems, are representative of the complexity of phase space calculation needed for the current precision demands of the LHC experiment collaborations [64]. However part of the motivation for this work, and indeed the broader increased interest in phase space integration methods, is due to the impending breaking point current pipelines face under the increased precision challenges of the HL-LHC programme. Firstly we observe that the phase space dimensionality of the highest multiplicity process studied here is 11. In broader Bayesian inference terms this is rather small, with NS being typically used for problems $\mathcal{O}(10)$ to $\mathcal{O}(100)$ dimensions, where it is uniquely able to perform numerical integration without approximation or strictly matching prior knowledge. The POLYCHORD implementation is styled as *next-generation* Nested Sampling, designed to have polynomial scaling with dimensionality aiming for robust performance as inference is extended to $\mathcal{O}(100)$ dimensions. Earlier implementations of NS, such as MULTINEST [46], whilst having worse dimensional scaling properties, may be a useful avenue of investigation for the lower dimensional problems considered in this paper.

This work validated NS in a context where current tools still can perform the required tasks, albeit at times at immense computational costs. Requirements from the HL-LHC strain the existing LHC event generation pipeline in many ways and pushing the sampling problem to higher dimensions is no exception [2]. Importance Sampling becomes exponentially more sensitive to how close the proposal distribution matches the target in higher dimensions, a clear challenge for particle physics in two directions; multileg processes rapidly increasing the sampling dimension [65] and corresponding radiative corrections (real and virtual) make it increasingly

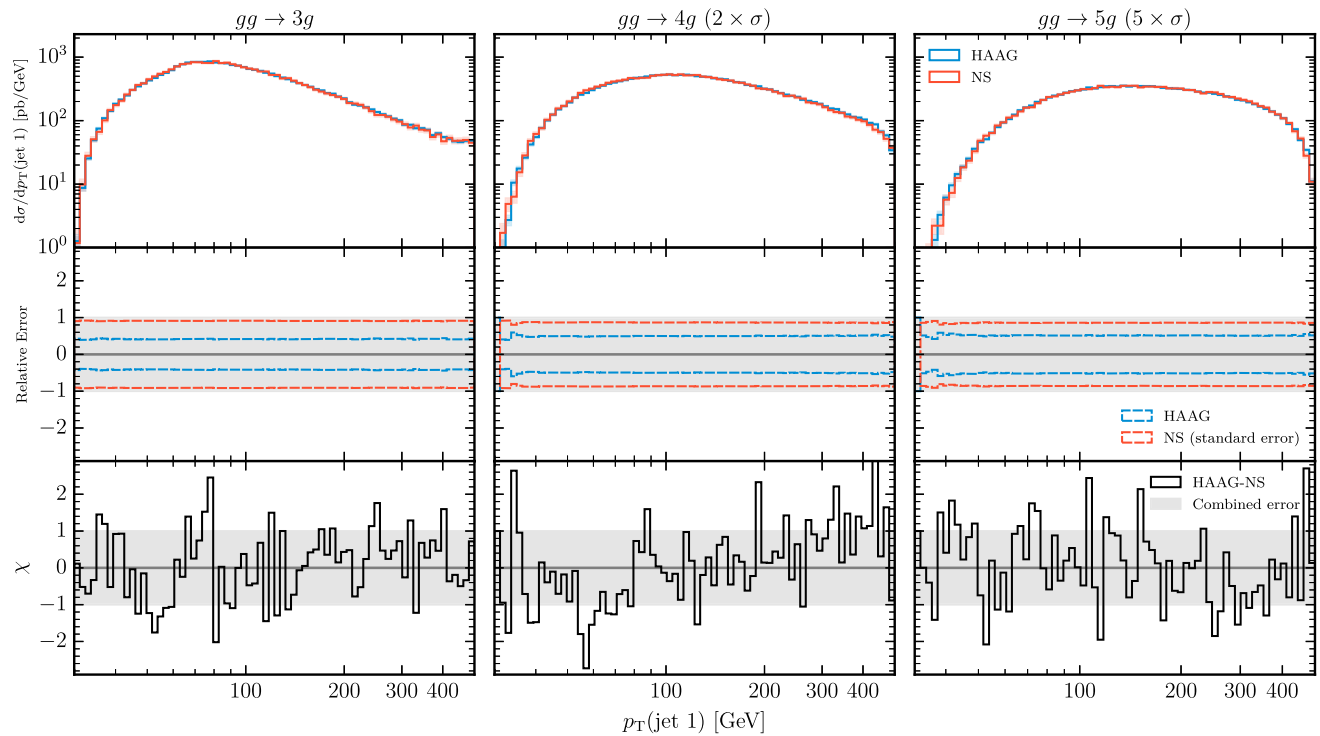


Fig. 7 The equivalent leading jet transverse momentum observable as calculated in Fig. 5a, using an exact unit weight compression of the same samples. A modified version of the default MC_JETS routine has been used to extend the p_T range shown

hard to provide an accurate proposal, e.g. through the sheer number of phase space channels needed and by having to probe deep into singular phase space regions [66]. We propose that NS is an excellent complement to further investigation on both these fronts. The robust dimensional scaling of NS illustrated against VEGAS in Fig. 6 encapsulates both solid performance with increasing dimension, and the adherence to an uninformed prior whilst still attaining this scaling is promising for scenarios where accurate proposals are harder to construct.

4.2 Using prior knowledge

Perhaps the most obvious choice that makes the application here stylised is in always starting from an uninformed prior state of knowledge. Using Equations (2) and (4), the cross section integral with a phase space mapping was motivated as being exactly the Bayesian evidence integral with a choice of prior. To that end there is no real distinction between taking the non-uniform HAAG distribution as the prior instead of the flat RAMBO density that was used in this study. In this respect NS could be styled as learning an additional compression to the posterior distribution, refining the static proposal distributions typically employed to initiate the generation of a phase space mapping (noting that this is precisely what VEGAS aims to do in this context).

Naively applying a non-flat mapping exposes the conflicting aims at play in this set of problems however; efficiently generating events from a strongly peaked distribution, and generating high statistics estimates of the tails of the same distribution. Taking a flat RAMBO prior is well suited to the latter problem, whereas taking a HAAG prior is better suited to the former. One particular hyperparameter of POLYCHORD that was introduced can be tuned to this purpose; the number of prior samples, n_{prior} . If future work is to use a non flat, partially informed starting point, increasing n_{prior} well above the minimum (equal to the number of live points required) used in this study would be needed. A more complete direction for further work would be to investigate the possibility of mixing multiple proposal distributions [67,68].

As a demonstration, we again apply NS to the toy example of Sect. 2.1 but this time using a non-uniform prior distribution. While a good prior would be an approximation of the target distribution, we choose to purposely miss an important feature of the target, the straight line segment, that the sampler still has to explore. Considering that in HEP applications the prior knowledge may be encoded in the mixture distributions of a multi-channel importance sampler, this is an extreme version of a realistic situation. As typically the number of channels grows dramatically with increasing final-state particle multiplicity, e.g. factorially when channels correspond to the topologies of contributing Feynman diagrams, one might choose to disable some sub-dominant channels in

order to avoid a prohibitively large set of channels. However, this would lead to a mis-modelling of the target in certain phase-space regions.

Here we use only the ring part of the target, truncated on a circle that covers the unit hypercube, as our prior. Without an additional coordinate transformation this prior would not be of much use for VEGAS as the line part remains on the diagonal. To sample from the prior, we first transform to polar coordinates. Then we sample the angle uniformly and the radial coordinate using a Cauchy distribution truncated to the interval $(0, 1/\sqrt{2}]$. In order to have good coverage of the tails, despite the strongly peaked prior, we increase n_{prior} to $50 \times n_{\text{live}}$. This results in a total efficiency of $\epsilon = 0.037(4)$, more than three times the value obtained with a uniform prior, cf. Table 1. While the unweighting efficiency reduces to $\epsilon_{\text{uw}} = 0.17(2)$, the slice sampling efficiency increases to $\epsilon_{\text{ss}} = 0.216(7)$. In Fig. 8 we show the ratio between the target function and the POLYCHORD sampling distribution. Compared to Fig. 2c, the ratio has a smaller range of values. Along the peak of the ring part of the target function, the ratio is approximately one. The largest values can be found around the line segment with POLYCHORD generating up to ten times less samples than required by the target distribution. It can be concluded that even with an intentionally poor prior distribution, POLYCHORD benefits from the prior knowledge in terms of efficiency and still correctly samples the target distribution including the features absent from the prior.

4.3 Dynamic nested sampling

In addition to using a more informed prior to initiate the Nested Sampling process, a previous NS run can be used to further tune the algorithm itself to a particular problem. This is an existing idea in the literature known as dynamic Nested Sampling [69]. Dynamic NS uses information acquired about the likelihood shells in a previous NS run to varying the number of live points *dynamically* throughout the run. This results in a more efficient allocation of the computation towards the core aim of compressing the prior to the posterior. We expect that this would only increase the efficiency of the unweighting process, as the density of weighted events would be trimmed to even more closely match the underlying phase space density. Dynamic Nested Sampling naturally combines with the proposal of *using prior knowledge* to make a more familiar generator chain, however one that is driven primarily by NS. This mirrors the current established usage of VEGAS in this context; using VEGAS to refine the initial mapping by a redistribution of the input variables, to more efficiently generate from the acquired mapping.

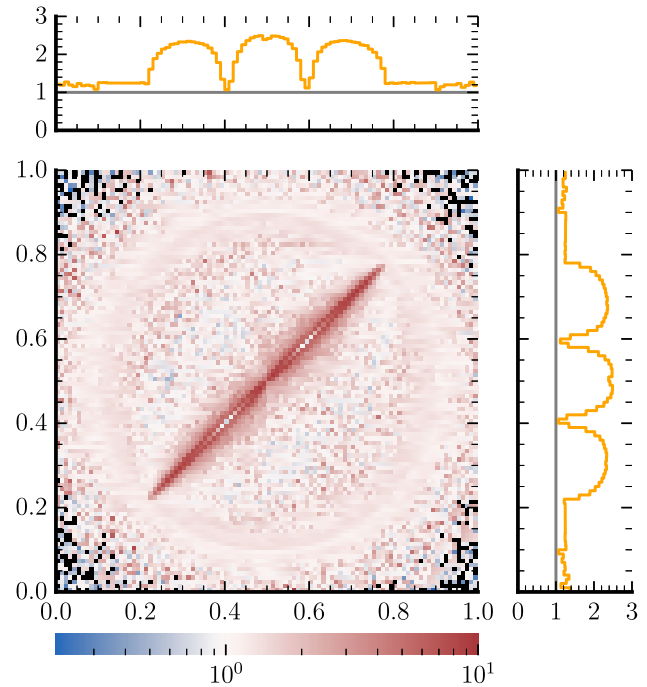


Fig. 8 The ratio of the target function of the two-dimensional toy example and the probability density function of POLYCHORD using a non-uniform prior distribution. Black histogram bins have not been filled by any data due to limited sample size

4.4 Connection to modern machine learning techniques

There has been a great deal of recent activity coincident to this work, approaching similar sets of problems in particle physics event generation using modern Machine Learning (ML) techniques [70]. Much of this work is still exploratory in nature, and covers such a broad range of activity that comprehensively reviewing the potential for combining ML and NS is beyond the scope of this work. It is however clear that there is strong potential to include NS into a pipeline that modern ML is already aiming to optimise. To that aim, we identify a particular technique that has been studied previously in the particle physics context; using Normalising Flows to train phase space mappings [29–31]. In spirit a flow based approach, training an invertible probabilistic mapping between prior and posterior, bears a great deal of similarity to the core compression idea behind Nested Sampling. The potential in dovetailing Nested Sampling with a flow based approach has been noted in the NS literature [71], further motivating the potential for synergy here.

The ability of NS to construct mappings of high dimensional phase spaces without needing any strong prior knowledge, can be motivated as being an ideal forward model with

which to train a Normalising Flow. In effect this replaces the generator part of the process with an importance sampler, whilst still using NS to generate the mappings. This is particularly ideal in this context, as the computational overhead required to decorrelate the Markov Chains imposes a harsh limit on the efficiency of a pure NS based approach. Combining these techniques in this way could retain the desirable features of both and serve to mitigate the ever increasing computational demands of energy frontier particle physics.

We close by noting that also in the area of lattice field theory Normalising Flows have recently attracted attention, see e.g. [72, 73], to address the sampling of multimodal target function. We envisage that also in these applications Nested Sampling could be applied.

5 Conclusions

The establishing study presented here had two main aims. Firstly to introduce the technique of Nested Sampling, applied to a realistic problem, to researchers in the particle physics community. Secondly to provide a translation back to researchers working on Bayesian inference techniques, presenting an important and active set of problems in particle physics that Nested Sampling could provide a valuable contribution to. The physical example presented used POLYCHORD to perform an end-to-end generation of events without any input prior knowledge. This is a stylised version of the event generator problem, intended to validate Nested Sampling in this new context and demonstrate some key features. For the considered multi-gluon production processes Nested Sampling was able to learn a mapping in an efficient manner that exhibits promising scaling properties with phase space dimension. We have outlined some potential future research directions; highlighting where the strengths of this approach could be most effective, and how to embed Nested Sampling in a more complete event generator workflow. Along these lines, we envisage an implementation of the Nested Sampling technique for the SHERPA event generator framework [54], possibly also supporting operation on GPUs [74]. This will provide additional means to address the computing challenges for event generation posed by the upcoming LHC runs.

Acknowledgements This work has received funding from the European Union's Horizon 2020 research and innovation programme as part of the Marie Skłodowska-Curie Innovative Training Network MCnetITN3 (Grant agreement no. 722104). SS and TJ acknowledge support from BMBF (contract 05H21MGCAB). SS acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 456104544. WH and DY are funded by the Royal Society. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2

funding from the Engineering and Physical Sciences Research Council (capital Grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: The results involve purely theoretical comparisons, publicly available software packages to recreate the calculations are listed and referenced where appropriate.]

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Funded by SCOAP³. SCOAP³ supports the goals of the International Year of Basic Sciences for Sustainable Development.

Appendix: Uncertainties in nested sampling

Typical Nested Sampling literature focuses on two main sources of uncertainty; an uncertainty on the weights of the dead points due to the uncertainty in the volume contraction at each iteration, and an uncertainty on the overall volume arising from the path the Markov Chain takes through the space to perform each iteration. The former source is what we consider in this work, and can be calculated as a sample of weights for each dead point using `anesthetic`. The latter source can be estimated using the `nestcheck` package [75], the method presented here uses combinations of multiple runs to form integral estimates meaning the best strategy to minimise this effect is already baked in. Further use cases would benefit from more thorough cross checks using `nestcheck`.

The usual source of uncertainty in a binned histogram in particle physics comes from the standard error. Importance Sampling draws sample events with associated weights w_i , with the sum of these sample weights giving the estimated cross section in a bin. The effective number of fills in a bin using weighted samples is,

$$N = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}. \quad (7)$$

The inverse square root of N then constitutes the standard error on the cross section in the bin. In practice this means that the standard deviation of an integral estimated with Importance Sampling can be quoted as $\Delta_{MC} = \sqrt{\sum_i (w_i^2)}$. In typical NS applications this is significantly smaller than the pre-

viously mentioned sources, and thus often not considered. However, when using NS as a phase space event generator for finely binned differential observables, the statistical uncertainty can become a significant effect so must be taken into account. Adding the standard error to the weight uncertainty in quadrature is a suitable upper bound for the NS uncertainty but is found to overestimate the uncertainty in some bins. While the standard error gives a measure of the spread of weights around the mean weight in a bin, alternative weights from the sampling history in NS also give an overlapping measure of this.

To correctly account for the statistical error in this context a revised recipe is needed. The following proposed procedure reweights the alternative weight samples to account for the spread of the resulting effective fills in each bin. The effective number of entries in a bin arising from a NS run can be written as,

$$N_j = \frac{(\sum_i w_{j,i})^2}{\sum_i w_{j,i}^2}, \quad (8)$$

where i indexes the number of weighted samples in each bin, and j indexes the alternative weights. The result of the j sampled weight variations is a set of j different effective counts in each bin. These counts can be modelled as j trials of a multinomial distribution with j categories, written as,

$$P(N | \alpha) = \frac{j!}{\prod_j N_j!} \prod_j \alpha_j^{N_j}, \quad (9)$$

where a probability of sampling each category, α_j , has been introduced. The desired unknown distribution of α_j can be found using Bayes theorem to invert the arguments. If an uninformative conjugate prior to the multinomial distribution is used, the Dirichlet distribution, the desired inverted probability can also be written in the form of a Dirichlet distribution,

$$P(\alpha | N) = \Gamma \left(\sum_j N_j \right) \prod_j \frac{\alpha_j^{N_j-1}}{\Gamma(N_j)}. \quad (10)$$

A sample vector of α_j from this Dirichlet distribution, will give a probability of observing each category N_j . This probability can be used to weight the categories giving a weighted set of effective number of fills, $\{\alpha_j N_j\}$. This considers each alternative weight sample as a discrete sample from an underlying continuous distribution N_j is sampled from. The set of weighted effective fills can be used to quote a weighted set of samples of the bin cross section by multiplying by the square of the sum of the weights, $\{\sigma_j\} = \{\alpha_j N_j \sum_i (w_{j,i}^2)\}$. The estimated cross section in the bin is then the expected value of this

set, $\sigma = E[\sigma_j]$, and the total standard deviation on this cross section is derived from the variance, $\Delta\sigma_{\text{tot}} = (\text{Var}[\sigma_j])^2$.

References

1. A. Buckley et al., General-purpose event generators for LHC physics. *Phys. Rep.* **504**, 145 (2011). <https://doi.org/10.1016/j.physrep.2011.03.005>. arXiv:1101.2599
2. J.M. Campbell et al., Event Generators for High-Energy Physics Experiments, in 2022 Snowmass Summer Study, 3, (2022). arxiv:2203.11110
3. R. Kleiss, R. Pittau, Weight optimization in multichannel Monte Carlo. *Comput. Phys. Commun.* **83**, 141 (1994). [https://doi.org/10.1016/0010-4655\(94\)90043-4](https://doi.org/10.1016/0010-4655(94)90043-4). arXiv:hep-ph/9405257
4. C.G. Papadopoulos, PHEGAS: a phase space generator for automatic cross-section computation. *Comput. Phys. Commun.* **137**, 247 (2001). [https://doi.org/10.1016/S0010-4655\(01\)00163-1](https://doi.org/10.1016/S0010-4655(01)00163-1). arXiv:hep-ph/0007335
5. F. Krauss, R. Kuhn, G. Soff, AMEGIC++ 1.0: a matrix element generator in C++. *JHEP* **02**, 044 (2002). <https://doi.org/10.1088/1126-6708/2002/02/044>. arXiv:hep-ph/0109036
6. F. Maltoni, T. Stelzer, MadEvent: automatic event generation with MadGraph. *JHEP* **02**, 027 (2003). <https://doi.org/10.1088/1126-6708/2003/02/027>. arXiv:hep-ph/0208156
7. T. Gleisberg, S. Hoeche, Comix, a new matrix element generator. *JHEP* **12**, 039 (2008). <https://doi.org/10.1088/1126-6708/2008/12/039>. arXiv:0808.3674
8. G.P. Lepage, A new algorithm for adaptive multidimensional integration. *J. Comput. Phys.* **27**, 192 (1978). [https://doi.org/10.1016/0021-9991\(78\)90004-9](https://doi.org/10.1016/0021-9991(78)90004-9)
9. T. Ohl, Vegas revisited: adaptive Monte Carlo integration beyond factorization. *Comput. Phys. Commun.* **120**, 13 (1999). [https://doi.org/10.1016/S0010-4655\(99\)00209-X](https://doi.org/10.1016/S0010-4655(99)00209-X). arXiv:hep-ph/9806432
10. S. Jadach, Foam: multidimensional general purpose Monte Carlo generator with selfadapting symplectic grid. *Comput. Phys. Commun.* **130**, 244 (2000). [https://doi.org/10.1016/S0010-4655\(00\)00047-3](https://doi.org/10.1016/S0010-4655(00)00047-3). arXiv:physics/9910004
11. T. Hahn, CUBA: a Library for multidimensional numerical integration. *Comput. Phys. Commun.* **168**, 78 (2005). <https://doi.org/10.1016/j.cpc.2005.01.010>. arXiv:hep-ph/0404043
12. A. van Hameren, PARNI for importance sampling and density estimation. *Acta Phys. Pol. B Ser* **40**, 259 (2009). arXiv:0710.2448
13. H. Kharraziha, S. Moretti, The Metropolis algorithm for on-shell four momentum phase space. *Comput. Phys. Commun.* **127**, 242 (2000). [https://doi.org/10.1016/S0010-4655\(99\)00504-4](https://doi.org/10.1016/S0010-4655(99)00504-4). arXiv:hep-ph/9909313
14. S. Weinzierl, A general algorithm to generate unweighted events for next-to-leading order calculations in electron positron annihilation. *JHEP* **08**, 028 (2001). <https://doi.org/10.1088/1126-6708/2001/08/028>. arXiv:hep-ph/0106146
15. K. Kröninger, S. Schumann, B. Willenberg, (MC)**3: a multi-channel Markov chain Monte Carlo algorithm for phase-space sampling. *Comput. Phys. Commun.* **186**, 1 (2015). <https://doi.org/10.1016/j.cpc.2014.08.024>. arXiv:1404.4328
16. H.S.F. Physics Event Generator, WG collaboration, Challenges in Monte Carlo event generator software for high-luminosity LHC. *Comput. Softw. Big Sci.* **5**, 12 (2021). <https://doi.org/10.1007/s41781-021-00055-1>. arxiv:2004.13687
17. HSF Physics Event Generator WG collaboration, E. Yazgan et al., HL-LHC Computing Review Stage-2, Common Software Projects: Event Generators, 9 (2021)
18. J. Bendavid, Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks, 6 (2017)

19. M.D. Klimek, M. Perelstein, Neural network-based approach to phase space integration. *SciPost Phys.* **9**, 053 (2020). <https://doi.org/10.21468/SciPostPhys.9.4.053>. arXiv:1810.11509
20. S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen et al., Event generation and statistical sampling for physics with deep generative models and a density information buffer. *Nat. Commun.* **12**, 2985 (2021). <https://doi.org/10.1038/s41467-021-22616-z>. arXiv:1901.00875
21. R. Di Sipio, M.F. Giannelli, S.K. Haghighat, S. Palazzo, DijetGAN: a generative-adversarial network approach for the simulation of QCD Dijet Events at the LHC. *JHEP* **08**, 110 (2019). [https://doi.org/10.1007/JHEP08\(2019\)110](https://doi.org/10.1007/JHEP08(2019)110). arxiv:1903.02433
22. A. Butter, T. Plehn, R. Winterhalder, How to GAN LHC Events. *SciPost Phys.* **7**, 075 (2019). <https://doi.org/10.21468/SciPostPhys.7.6.075>. arXiv:1907.03764
23. Y. Alanazi et al., Simulation of electron–proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN), 1 (2020)
24. Y. Alanazi et al., AI-based Monte Carlo event generator for electron-proton scattering, 8. (2020)
25. S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman, D. Shih, DCTRGAN: Improving the Precision of Generative Models with Reweighting. *JINST* **15** (2020). <https://doi.org/10.1088/1748-0221/15/11/P11004>. arXiv:2009.03796
26. A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, GANplifying event samples. *SciPost Phys.* **10**, 139 (2021). <https://doi.org/10.21468/SciPostPhys.10.6.139>. arXiv:2008.06545
27. A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, GANplifying Event Samples. *SciPost Phys.* **10**, 139 (2021). <https://doi.org/10.21468/SciPostPhys.10.6.139>. arXiv:2008.06545
28. K.T. Matchev, A. Roman, P. Shyamsundar, Uncertainties associated with GAN-generated datasets in high energy physics. *SciPost Phys.* **12**, 104 (2022). <https://doi.org/10.21468/SciPostPhys.12.3.104>. arXiv:2002.06307
29. E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, S. Schumann, Exploring phase space with neural importance sampling. *SciPost Phys.* **8**, 069 (2020). <https://doi.org/10.21468/SciPostPhys.8.4.069>. arXiv:2001.05478
30. C. Gao, J. Isaacson, C. Krause, i-flow: high-dimensional integration and sampling with normalizing flows. *Mach. Learn. Sci. Tech.* **1**, 045023 (2020). <https://doi.org/10.1088/2632-2153/abab62>. arXiv:2001.05486
31. C. Gao, S. Höche, J. Isaacson, C. Krause, H. Schulz, Event generation with normalizing flows. *Phys. Rev. D* **101**, 076002 (2020). <https://doi.org/10.1103/PhysRevD.101.076002>. arXiv:2001.10028
32. B. Stienen, R. Verheyen, Phase space sampling and inference from weighted events with autoregressive flows. *SciPost Phys.* **10**, 038 (2021). <https://doi.org/10.21468/SciPostPhys.10.2.038>. arXiv:2011.13445
33. K. Danziger, T. Janßen, S. Schumann, F. Siegert, Accelerating Monte Carlo event generation—rejection sampling using neural network event-weight estimates, 9 (2021)
34. M. Backes, A. Butter, T. Plehn, R. Winterhalder, How to GAN event unweighting. *SciPost Phys.* **10**, 089 (2021). <https://doi.org/10.21468/SciPostPhys.10.4.089>. arXiv:2012.07873
35. M. Bellagente, M. Haußmann, M. Luchmann, T. Plehn, Understanding event-generation networks via uncertainties, 4 (2021)
36. A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousset et al., Generative networks for precision enthusiasts, 10 (2021)
37. J. Skilling, Nested sampling for general Bayesian computation. *Bayesian Anal.* **1**, 833 (2006). <https://doi.org/10.1214/06-BA127>
38. W.J. Handley, M.P. Hobson, A.N. Lasenby, polychord: next-generation nested sampling. *Mon. Not. R. Astron. Soc.* **453**, 4385 (2015). <https://doi.org/10.1093/mnras/stv1911>. arXiv:1506.00171
39. D.J.C. MacKay, *Information Theory. Inference & Learning Algorithms* (Cambridge University Press, Cambridge, 2002)
40. Particle Data Group collaboration, Review of Particle Physics. *PTEP* **2020**, 083C01 (2020). <https://doi.org/10.1093/ptep/ptaa104>
41. G. Ashton et al., Nested sampling for physical scientists. *Nature* **2** (2022). <https://doi.org/10.1038/s43586-022-00121-x>. arXiv:2205.15570
42. R.M. Neal, Slice sampling. *Ann. Stat.* **31**, 705 (2003). <https://doi.org/10.1214/aos/1056562461>
43. P. Mukherjee, D. Parkinson, A.R. Liddle, A nested sampling algorithm for cosmological model selection. *Astrophys. J. Lett.* **638**, L51 (2006). <https://doi.org/10.1086/501068>. arXiv:astro-ph/0508461
44. R. Shaw, M. Bridges, M.P. Hobson, Clustered nested sampling: efficient Bayesian inference for cosmology. *Mon. Not. R. Astron. Soc.* **378**, 1365 (2007). <https://doi.org/10.1111/j.1365-2966.2007.11871.x>. arXiv:astro-ph/0701867
45. F. Feroz, M.P. Hobson, Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis. *Mon. Not. R. Astron. Soc.* **384**, 449 (2008). <https://doi.org/10.1111/j.1365-2966.2007.12353.x>. arXiv:0704.3704
46. F. Feroz, M.P. Hobson, M. Bridges, MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon. Not. R. Astron. Soc.* **398**, 1601 (2009). <https://doi.org/10.1111/j.1365-2966.2009.14548.x>. arXiv:0809.3437
47. F. Feroz, M.P. Hobson, E. Cameron, A.N. Pettitt, Importance nested sampling and the MultiNest algorithm. *Open J. Astrophys.* **2**, 10 (2019). <https://doi.org/10.21105/astro.1306.2144>. arXiv:1306.2144
48. W.J. Handley, M.P. Hobson, A.N. Lasenby, PolyChord: nested sampling for cosmology. *Mon. Not. R. Astron. Soc.* **450**, L61 (2015). <https://doi.org/10.1093/mnras/rlv047>. arXiv:1502.01856
49. W. Handley, P. Lemos, Quantifying dimensionality: Bayesian cosmological model complexities. *Phys. Rev. D* **100**, 023512 (2019). <https://doi.org/10.1103/PhysRevD.100.023512>. arXiv:1903.06682
50. S.S. AbdusSalam et al., Simple and statistically sound recommendations for analysing physical theories, 12 (2020)
51. GAMBIT collaboration, Comparison of statistical sampling methods with ScannerBit, the GAMBIT scanning module. *Eur. Phys. J. C* **77**, 761 (2017). <https://doi.org/10.1140/epjc/s10052-017-5274-y>. arXiv:1705.07959
52. A. Fowlie, W. Handley, L. Su, Nested sampling with plateaus. *Mon. Not. R. Astron. Soc.* **503**, 1199 (2021). <https://doi.org/10.1093/mnras/stab590>. arXiv:2010.13884
53. E. Higson, W. Handley, M. Hobson, A. Lasenby, Sampling errors in nested sampling parameter estimation. *Bayesian Analysis series* **13** (2018). <https://doi.org/10.1214/17-ba1075>
54. Sherpa collaboration, Event Generation with Sherpa 2.2. *SciPost Phys.* **7**, 034 (2019). <https://doi.org/10.21468/SciPostPhys.7.3.034>. arXiv:1905.09127
55. A. van Hameren, C.G. Papadopoulos, A hierarchical phase space generator for QCD antenna structures. *Eur. Phys. J. C* **25**, 563 (2002). <https://doi.org/10.1007/s10052-002-1000-4>. arXiv:hep-ph/0204055
56. R. Kleiss, W.J. Stirling, S.D. Ellis, A new Monte Carlo treatment of multiparticle phase space at high-energies. *Comput. Phys. Commun.* **40**, 359 (1986). [https://doi.org/10.1016/0010-4655\(86\)90119-0](https://doi.org/10.1016/0010-4655(86)90119-0)
57. S. Plätzer, RAMBO on diet, 8 (2013). arXiv:1308.2922
58. A. Fowlie, S. Hoof, W. Handley, Nested sampling for frequentist computation: fast estimation of small p values. *Phys. Rev. Lett.* **128**, 021801 (2022). <https://doi.org/10.1103/PhysRevLett.128.021801>. arXiv:2105.13923
59. E. Carragher, W. Handley, D. Murnane, P. Stangl, W. Su, M. White et al., Convergent Bayesian global fits of 4D compos-

- ite Higgs models. *JHEP* **05**, 237 (2021). [https://doi.org/10.1007/JHEP05\(2021\)237](https://doi.org/10.1007/JHEP05(2021)237). arXiv: 2101.00428
60. A. Fowlie, W. Handley, L. Su, Nested sampling cross-checks using order statistics. *Mon. Not. R. Astron. Soc.* **497**, 5256 (2020). <https://doi.org/10.1093/mnras/staa2345>. arXiv:2006.03371
 61. C. Bierlich et al., Robust independent validation of experiment and theory: rivet version 3. *SciPost Phys.* **8**, 026 (2020). <https://doi.org/10.21468/SciPostPhys.8.2.026>. arXiv:1912.05451
 62. M. Cacciari, G.P. Salam, G. Soyez, The anti- k_t jet clustering algorithm. *JHEP* **04**, 063 (2008). <https://doi.org/10.1088/1126-6708/2008/04/063>. arXiv:0802.1189
 63. W. Handley, anesthetic: nested sampling visualisation. *J. Open Sour. Softw.* **4**, 1414 (2019). <https://doi.org/10.21105/joss.01414>. arXiv:1905.04768
 64. ATLAS collaboration, Modelling and computational improvements to the simulation of single vector-boson plus jet processes for the ATLAS experiment, 12 (2021). arXiv:2112.09588
 65. S. Höche, S. Prestel, H. Schulz, Simulation of vector boson plus many jet final states at the high luminosity LHC. *Phys. Rev. D* **100**, 014024 (2019). <https://doi.org/10.1103/PhysRevD.100.014024>. arXiv:1905.05120
 66. T. Gleisberg, F. Krauss, Automating dipole subtraction for QCD NLO calculations. *Eur. Phys. J. C* **53**, 501 (2008). <https://doi.org/10.1140/epjc/s10052-007-0495-0>. arXiv: 0709.2881
 67. A. Petrosyan, W. Handley, SuperNest: accelerated nested sampling applied to astrophysics and cosmology, Maximum Entropy (accepted for Oral presentation) (2022)
 68. “SuperNest.” <https://gitlab.com/a-p-petrosyan/sspr>. <https://pypi.org/project/supernest/>
 69. E. Higson, W. Handley, M. Hobson, A. Lasenby, Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Stat. Comput.* **29**, 891–913 (2018). <https://doi.org/10.1007/s11222-018-9844-0>
 70. S. Badger et al., Machine Learning and LHC Event Generation, in 2022 Snowmass Summer Study, A. Butter, T. Plehn and S. Schumann, eds., 3 (2022). arXiv:2203.07460
 71. J. Alsing, W. Handley, Nested sampling with any prior you like. *Mon. Not. R. Astron. Soc.* **505**, L95 (2021). <https://doi.org/10.1093/mnras/slab057>. arXiv:2102.12478
 72. L. Del Debbio, J.M. Rossney, M. Wilson, Efficient modeling of trivializing maps for lattice ϕ^4 theory using normalizing flows: a first look at scalability. *Phys. Rev. D* **104**, 094507 (2021). <https://doi.org/10.1103/PhysRevD.104.094507>. arXiv:2105.12481
 73. D.C. Hackett, C.-C. Hsieh, M.S. Alberg, D. Boyda, J.-W. Chen, K.-F. Chen et al., Flow-based sampling for multimodal distributions in lattice field theory, 7 (2021)
 74. E. Bothmann, W. Giele, S. Höche, J. Isaacson, M. Knobbe, Many-gluon tree amplitudes on modern GPUs: a case study for novel event generators, 6 (2021)
 75. E. Higson, W. Handley, M. Hobson, A. Lasenby, Nestcheck: diagnostic tests for nested sampling calculations. *Mon. Not. R. Astron. Soc.* **483**, 2044 (2019). <https://doi.org/10.1093/mnras/sty3090>. arXiv:1804.06406

3.3 Comparison between the two approaches and opportunities for synthesis

In sections 3.1.3 and 3.2.2, two very different approaches with a common goal are presented: NFS as a replacement for the classic VEGAS algorithm in adaptive multichannel sampling, and nested sampling as an alternative method for PS sampling. Both try to find an efficient way to generate PS points whose distribution closely approximates the desired differential cross-section distribution. In some sense, the NF approach is a minimal change to the current standard. It does not change any of its guarantees and does not require changes to existing workflows, if training and evaluation of the NFS is automated. Note, however, that the NFS benefit greatly from running on GPUs in a parallelized way, which are not yet available in standard workflows. Nested sampling, on the other hand, is a fundamentally different sampling algorithm. Notable differences include the non-deterministic number of events and the estimation of uncertainties.

In the examples considered in the publication, we have applied nested sampling without any prior knowledge, and achieved impressive efficiencies and scaling. It can be expected that the performance can be further improved by using an informed prior that implements some of our physics knowledge, e.g. the location and shape of peaks in phase space. Such a prior distribution can be provided in the form of a transformation from the unit hypercube to the parameterization of the target function. This is reminiscent of the use of PS mappings for importance sampling and it should be possible to construct prior distributions for nested sampling based on these mappings. Consequently, it should also be possible to use adaptive bijections like VEGAS and NFS to further tune these prior distributions to the target. Such an approach was suggested and demonstrated for a cosmological application in ref. [263]. To deal with the topological issues of NFS, e.g. in the case of multimodal targets, ref. [264] introduced piecewise normalizing flows, where the target distribution is divided into clusters and an NF is trained on each cluster. In this way, one can hope to combine the benefits of NFS and nested sampling. Implementation and evaluation for HEP examples is left to future research.

Another approach that uses NFS for nested sampling was presented in ref. [265]. The authors trained an NF on a set of live points to map their distribution to a Gaussian latent space. This can then be used as a proposal to sample new live points within an iso-likelihood contour. If the mapping is perfect, the complicated iso-likelihood contours in the target space are mapped to spherical contours in the latent space, which are much easier to sample from. The idea then is to take the current worst point, map it to latent space, and sample a new point in latent space with a larger likelihood. The new point can be mapped back to target space, and serve as a proposal for a new live point. Since the mapping is not perfect in practice, the proposals have to be resampled to ensure their distribution matches the prior. To this end, the authors proposed to use rejection sampling. Assuming a well-trained NF, the acceptance rate is high and the method requires few likelihood evaluations to sample new live points. The method is suited for applications where the likelihood is expensive to evaluate, such that the costs of training the NF are offset by the reduced number of likelihood evaluations. The authors used gravitational-wave inference as an example, finding that they were able to half the number of likelihood evaluations required. In ref. [266], the method was further developed, including by the use of a mixture of NFS.

4 Accelerating unweighted event generation

In chapter 3, two methods for the efficient generation of weighted scattering events, with the possibility of unweighting them, are presented. Below we consider an alternative, complementary approach. It targets solely unweighted event generation, and offers no benefits for the generation of weighted events or for calculating the total cross-section. The idea is to approximate a computationally expensive matrix element by a fast and accurate surrogate in the form of a neural network. This surrogate is then used for unweighting by taking it as the target in the rejection sampling step. Unavoidably, we introduce an error by doing this: we accept or reject events based on the wrong probability, and therefore the distribution of the unweighted events is biased. The key insight here is that this bias can be fully corrected by introducing a second unweighting step, which uses the true value of the ME. However, we only have to call the expensive true ME for those events that have been accepted in the first unweighting step. Thus, if the first unweighting efficiency is sufficiently low, we can significantly reduce the number of calls to the true ME and gain overall efficiency.

An important ingredient for the method is partial unweighting. This method is not introduced in the previous chapters since it is not relevant to the discussion there. It is, however, a standard method in MC event generation and routinely used. In the context of this chapter, it takes on a more important role. Partial unweighting is therefore introduced in section 4.1.

The surrogate unweighting method was first published in the article titled ‘Accelerating Monte Carlo event generation—rejection sampling using neural network event-weight estimates’ [11]. After a short introduction, the article is reprinted in section 4.2. In a second article, ref. [12], the method was combined with the more sophisticated ME emulation model of ref. [64]. This article is the subject of section 4.3. Section 4.4 concludes this chapter by explaining the relationship of the two-step surrogate unweighting method to the efficiency improvement methods presented in chapter 3.

4.1 Partial unweighting

Partial unweighting is based on the rejection sampling algorithm, which is introduced in section 2.3.4. In section 2.3.5, it is furthermore explained why it is often desired to use rejection sampling to generate unweighted event samples. To apply rejection sampling, a scaling factor is needed that ensures that the proposal always lies above the target. Therefore, it has to be at least as large as the maximum weight appearing in the weighted event sample. In practical applications, though the maximum weight is typically unknown a priori, and has to be determined somehow. One could search for the maximum using a blackbox optimization algorithm, e.g. Nelder-Mead [267] or Basin-hopping [268]. However, this can take many function evaluations and be very costly for target functions that are expensive to evaluate. Therefore, another approach is typically used. It is necessary anyway to have an initial run of the event generator where weighted events are generated and used to determine the total cross-section, as well as to optimize the PS sampler. Thus, it is a natural idea to use this run also to determine the maximum event weight. Note, however, that this can only be done after the PS optimization has finished, since the maximum weight depends on the shape of the sampling density.

It is clear that it is unlikely to find the global maximum from a finite event sample. Therefore, it can happen that we use a scaling factor $c < w_{\max}$ and later, during unweighting, encounter

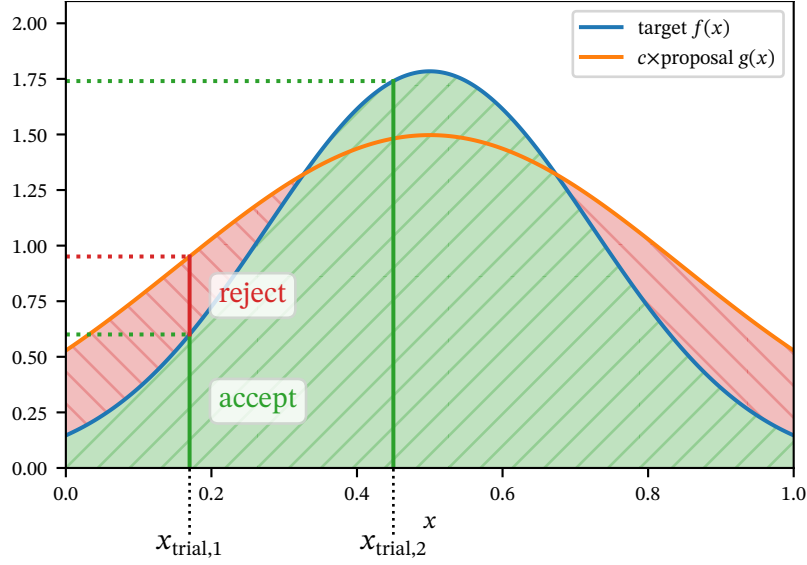


Figure 4.1: Partial unweighting applied to a one-dimensional target function.

events with weights larger than c . Technically, this means that they have an acceptance probability larger than one, and will always be accepted. However, if we treat these events the same as the other accepted events, this will distort their distribution and introduce a deviation from the target distribution. This situation is demonstrated in fig. 4.1, where the scaled proposal underestimates the target in the peak region. As a consequence, the second trial point, $x_{\text{trial},2}$, has an acceptance probability above one. The correct way to deal with such an event is to assign a weight to it, given by

$$\tilde{w}_i = \frac{w_i}{c} = \frac{f(\mathbf{x}_i)}{c g(\mathbf{x}_i)}. \quad (4.1)$$

Since \tilde{w}_i is always larger than one, we call it an overweight. The resulting event sample, consisting of unit-weighted and overweighted events, is not properly unweighted any longer. We call it a partially unweighted sample.

While at first glance it may not seem desirable to have overweighted events, partial unweighting is often used intentionally. In SHERPA, as well as other generators, it is the default setting. The reasoning behind this is based on the fact that the appearance of overweights can not be avoided except by deliberately overestimating the maximum weight, which in turn leads to low unweighting efficiencies and wasted computational efforts. In this way, the efficiency of event generation can be dominated by a few rare outliers. At the same time, the actual impact of overweights is typically small, since they appear at low frequencies and are mostly close to one. In order to end up with reasonable unweighting efficiencies, it can thus be sensible to work with a reduced scaling factor, $c < w_{\text{max}}$, in a way that keeps the impact of overweighted events under control. This can be done in several ways, leading to different choices of scaling factors. The method that is used by SHERPA monitors the weights appearing in the initial run and chooses a value for the scaling factor for which it can be expected that the overweights do not contribute more than a certain percentage to the total cross-section. Note that this requires a large enough sample of weighted events to be somewhat reliable. However, in practice this is anyway needed for a good estimate of the total cross-section. More details on the method are given in section 4.2. An alternative method is also presented there.

It is important to note that a correct statistical treatment requires to always take the overweights into account. Ignoring the overweights, and thereby treating a partially unweighted sample the same as a fully unweighted one, inadvertently introduces bias. However, when

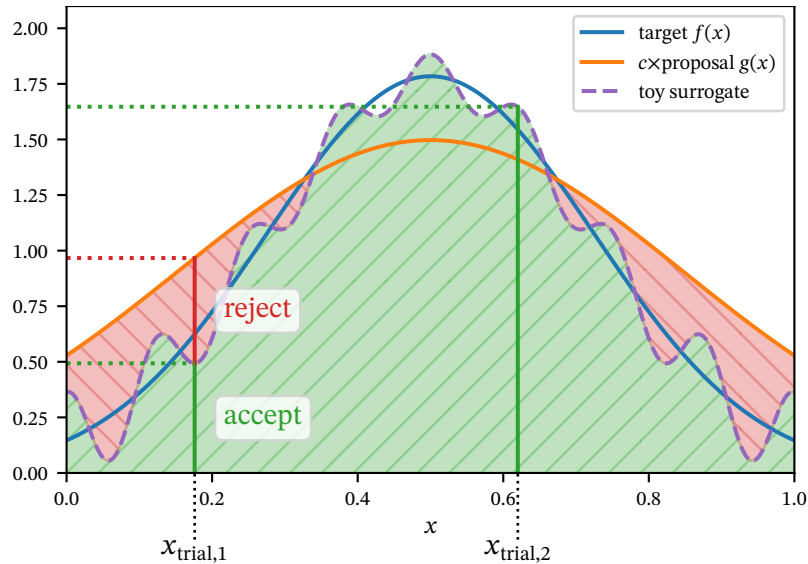


Figure 4.2: Surrogate unweighting applied to a one-dimensional target function.

the overweights are not too large and not too frequent, the bias can be small in comparison to other uncertainties.

4.2 Publication: Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates

In this section, the article ‘Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates’ is presented. In contrast to the publications presented in chapter 3, this article is not concerned with increasing the unweighting efficiency of event generation. Instead, it introduces the idea of speeding up unweighted event generation by using fast and accurate surrogates for the event weights. This is illustrated in fig. 4.2, where in comparison to fig. 4.1 the target function has been replaced by a toy surrogate that changes the acceptance probability. However, the surrogate unweighting method is unbiased, because it adopts a second unweighting step that corrects for the approximation errors of the surrogate. The method is a promising approach in situations where the event weight is expensive to evaluate, due to complicated matrix elements, and where the unweighting efficiency of an event generator is low, due to limitations of the phase space mappings. This is typically the case with high-multiplicity tree-level processes, among others.

In the article, the new unweighting algorithm is introduced, and it is discussed how it differs from the established approach and in which situations one can expect benefits. To test the method, a simple NN based surrogate model is presented. Using this model, the performance is evaluated for different examples contributing to $Z, W + 4$ jets and $t\bar{t} + 3$ jets production at the LHC. It is shown that the time for the generation of unweighted events can be reduced by factors up to ten for individual partonic channels. Based on differential distributions of physical observables, the validity of the method is demonstrated. Furthermore, it is shown how the method can be generalized to non-positive event weights, which appear in NLO (and higher order) calculations based on subtraction methods. This is demonstrated for a toy example.

The article was first published as a preprint on ARXIV in September 2021. Subsequently, it was submitted to the journal *SciPost Physics*. After the referees’ comments were addressed, it was finally published in May 2022. The version published in the journal is reprinted below. Copyright and license notices as well as a link to the material are provided on the first page of

the article.

Author contributions

Frank Siegert came up with the initial idea and the first studies were done by Johannes Krause as part of his M.Sc. thesis [269]. He considered different choices of surrogates and found NNS to be the most performant. His performance claims for realistic LHC applications were obtained by statistical considerations without an actual implementation. Such an implementation within the SHERPA event generator was first realized by Katharina Danziger for her M.Sc. thesis [270]. She optimized the NN architecture and benchmarked the performance for partonic channels contributing to the tree-level LHC production processes $Z/W + 4$ jets and $t\bar{t} + 3$ jets. Working towards a publication, I built upon her implementation. Our changes required me to regenerate all the event samples for the results presented in the paper. Steffen Schumann contributed the generalization to non-positive event weights presented in sec. 2.2 for a toy example. Apart from that, all computational work was done by me. This includes implementation, generation of events, and evaluation of the results. Steffen Schumann, Frank Siegert and I contributed significantly to the writing of the article. This includes the presentation of the results in figures and tables.

Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates

Katharina Danziger¹, Timo Janßen², Steffen Schumann² and Frank Siegert¹

¹ Institut für Kern- und Teilchenphysik, TU Dresden, Dresden, Germany

² Institut für Theoretische Physik, Georg-August-Universität Göttingen, Göttingen, Germany

Abstract

The generation of unit-weight events for complex scattering processes presents a severe challenge to modern Monte Carlo event generators. Even when using sophisticated phase-space sampling techniques adapted to the underlying transition matrix elements, the efficiency for generating unit-weight events from weighted samples can become a limiting factor in practical applications. Here we present a novel two-staged unweighting procedure that makes use of a neural-network surrogate for the full event weight. The algorithm can significantly accelerate the unweighting process, while it still guarantees unbiased sampling from the correct target distribution. We apply, validate and benchmark the new approach in high-multiplicity LHC production processes, including $Z/W+4$ jets and $t\bar{t}+3$ jets, where we find speed-up factors up to ten.



Copyright K. Danziger *et al.*

This work is licensed under the Creative Commons

[Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Published by the SciPost Foundation.

Received 28-09-2021

Accepted 28-04-2022

Published 18-05-2022

doi:[10.21468/SciPostPhys.12.5.164](https://doi.org/10.21468/SciPostPhys.12.5.164)



Check for updates

Contents

1	Introduction	1
2	Phase-space sampling and event unweighting	3
2.1	A novel unweighting procedure	5
2.2	Generalisation to non-positive event weights	6
3	Machine learning event weights	9
3.1	Neural-network based matrix-element emulation	9
3.2	An example: $gg \rightarrow e^-e^+ggd\bar{d}$	11
4	Surrogate-based unweighting: Implementation, Validation and Results	14
4.1	Implementation in the SHERPA framework	14
4.2	An example: $gg \rightarrow e^-e^+ggd\bar{d}$	14
4.3	Results for LHC production processes	16
4.3.1	$W+4$ jets	18
4.3.2	$t\bar{t}+3$ jets	20
4.3.3	Summary of physics validation for LHC processes	23
5	Conclusions	25
	References	26

1 Introduction

Multi-purpose Monte Carlo event generators such as HERWIG [1,2], PYTHIA [3,4] or SHERPA [5,6], are indispensable tools for the analysis and interpretation of high-energy particle-collision experiments, *e.g.* at the Large Hadron Collider (LHC). They encapsulate our present-day understanding of the fundamental laws of nature, and provide means to simulate individual scattering events in a fully exclusive manner. With such virtual collisions we can quantify expected event yields and predict detailed final-state properties for in principle arbitrary scattering processes.

The central and often the computationally most expensive element of event simulations is a hard-scattering process – addressing the highest momentum-transfer interactions – that gets described by transition matrix elements evaluated in fixed-order perturbation theory. Given the enormous collision energies and impressive luminosities achieved at the LHC, paired with the excellent performance of the experiments, the need to provide evaluations of higher multiplicity hard-scattering processes is steadily growing. In view of the upcoming HL-LHC this becomes an even more pressing problem, requiring *much faster* event generation in order to match the expected event yields with the projected computing resources [7,8]. The underlying matrix-elements are calculated by dedicated matrix-element generators. Widely used tree-level tools such as ALPGEN [9], AMEGIC [10], COMIX [11], MADGRAPH [12] and WHIZARD [13] automatically construct tree-level amplitudes, but also provide efficient means to generate momentum configurations for the initial- and final-state particles taking part in the hard scattering. Furthermore, there exist dedicated tools for the construction and evaluation of one-loop amplitudes in QCD and the electroweak coupling, *e.g.* MADLOOP [14,15], MCFM [16,17], NJET [18], OPENLOOPS [19,20], POWHEGBOX [21], and RECOLA [22,23]. These tools can be used to compile fixed-order partonic cross-section computations and to probabilistically generate parton-level events. When incorporated into or interfaced to a multi-purpose event generator they provide the momentum-space partonic scattering events that get dressed by QCD parton showers, if applicable supplemented by an underlying event simulation, and finally transitioned to fully exclusive hadron-level final states by invoking a hadronisation model [24].

An efficient sampling of the final-state phase space is particularly crucial for complex scattering processes, where a single evaluation of the matrix element can take $\mathcal{O}(1s)$ [25]. Especially for experimental applications, *i.e.* the actual generation of pseudo data, including a simulation of the detector response, see *e.g.* refs. [26–28], unit-weight event samples are required, that are conventionally obtained from weighted events via *rejection sampling*. The resulting unit-weight events are unbiased random samples of fully uncorrelated probes of the target distribution given by the squared transition matrix element. They appear with frequencies that we would expect in a corresponding experiment. Although information about the target is lost in the unweighting step, the expensive detector simulation or other post processing of many events with minuscule weight gets avoided.

In modern matrix-element generators importance-sampling techniques are used, that account and possibly adapt [29] to the modal structures of the target, thereby employing knowledge about the propagator and spin structures of a given process [30]. These methods aim to reduce the inherent variance of the weight distribution of weighted event samples, and in turn also improve the unweighting efficiencies.

There have recently been a number of different strands of research to make optimal use of event-weight information, and, largely driven by algorithmic opportunities provided by novel machine-learning (ML) techniques, to optimise phase-space sampling and also event unweighting. On-the-fly reweighting methods are meanwhile routinely used to account for systematic uncertainties [31–33], or alternative physics models [34,35]. The use of MCMC techniques for exploring high-dimensional phase spaces has been studied in [36]. In [37]

the application of analysis-specific optimal sampling distributions was proposed, similar to methods of biasing event generation, *e.g.* to oversample tails of physical distributions [38]. A number of approaches to accelerate event generation based on (generative adversarial) neural networks have been presented [39–47]¹. An alternative and particularly attractive class of algorithms is based on *normalizing flows* [49–51], *i.e.* trainable bijectors parametrised by neural networks, see for instance [52–55], that can represent highly expressive importance-sampling maps [56,57]. Corresponding implementations and first applications of normalizing flows to Monte Carlo event generation in high-energy physics have been presented in [58–61]. Ref. [62] discussed the usage of GANs, trained on weighted Monte Carlo samples to produce unit-weight events. However, in order to guarantee the reproduction of the true target distribution, an additional post-processing step is needed. Possible solutions to this problem based on reweighting have been presented in [63,64]. The application of Bayesian networks for event generation including the quantification of uncertainties has been presented in [65,66].

We here propose an alternative approach to accelerate the unweighting procedure using ML methods. During the initial integration phase of a standard importance sampler we train a deep neural network to predict the event weight for given phase-space points. For complex processes, this surrogate is much cheaper to evaluate than the actual event weight. We therefore employ it in an initial rejection sampling. Only when the surrogate event weight gets accepted, we invoke a second unweighting step, where we account for the difference between the surrogate and the actual event weight. While a two-step unweighting procedure has been applied before [9], our combination with a neural-network surrogate gives it a new purpose. Given the neural network approximates the weight distribution reasonably well, we can significantly reduce the number of evaluations of the computationally expensive target function. Our approach easily generalises to non-positive targets and is thus suitable also for unweighted-event generation beyond the leading order in perturbation theory. We have implemented, validated and benchmarked the method in the SHERPA event-generator framework and here present results for tree-level $Z/W+4$ jets and $t\bar{t}+3$ jets production at the LHC.

The paper is organised as follows, in Sec. 2 we briefly review the basics of Monte Carlo event generation and event unweighting in the canonical approach. We then introduce our novel unweighting procedure, exemplified for a simple toy example. In Sec. 3 we discuss the neural-network setup and the used training procedure to obtain a predictor for the weight of scattering events. In Sec. 4 we describe our implementation of the new method in the SHERPA framework and present exemplary results for high-multiplicity LHC production processes. We conclude and give an outlook in Sec. 5.

2 Phase-space sampling and event unweighting

For sake of simplicity, we begin by considering the generic integral

$$I = \int_{\Omega} f(u') du', \quad (1)$$

with f a positive-definite target distribution $f : \Omega \subset \mathbb{R}^d \rightarrow [0, \infty)$ defined over the unit hypercube $\Omega = [0, 1]^d$. The Monte Carlo estimate of the integral is given by

$$I \approx E_N = \frac{1}{N} \sum_{i=1}^N f(u_i) = \langle f \rangle, \quad (2)$$

¹A critical review on the application of Generative Adversarial Networks (GANs) in the context of event generation has been presented in [48].

where we assumed N uniformly distributed random variables $u_i \in \Omega$. The random points u_i are interpreted as individual *events* and $w_i \equiv f(u_i)$ is called the corresponding *event weight*²; the integral is thus estimated by the average of the event weights $\langle w \rangle_N$. The standard deviation of the integral estimate is given by

$$\sigma_N(f) = \sqrt{\frac{V_N(f)}{N}} = \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}, \quad (3)$$

with V_N the corresponding variance. Variance-reduction techniques aim for a minimisation of V_N , e.g. by a remapping of the input random variables u to a non-uniform distribution $v: \Omega \rightarrow \bar{\Omega}$, called *importance sampling* [67]. For the desired integral this results in

$$I = \int_{\Omega} \frac{f(u')}{g(u')} g(u') du' = \int_{\bar{\Omega}} \frac{f(u')}{g(u')} \Big|_{u'=u'(v')} dv' \quad \text{with} \quad g(u) = \left| \frac{\partial v(u)}{\partial u} \right|. \quad (4)$$

With suitably chosen probability density $g(u)$, the variance of the integrand can be significantly reduced. A prominent example widely used in particle physics is VEGAS [68]. Given the multimodal nature of high-energy scattering matrix elements, state-of-the-art generators employ adaptive multi-channel importance samplers [10, 11, 69, 70]. Thereby the probability density $g(u)$ is decomposed into a sum of N_c *channels*, i.e.

$$g(u) = \sum_{j=1}^{N_c} \alpha_j g_j(u), \quad \text{with} \quad \sum_{j=1}^{N_c} \alpha_j = 1 \quad \text{and} \quad 0 \leq \alpha_j \leq 1, \quad (5)$$

yielding

$$I = \int_{\Omega} \frac{f(u')}{g(u')} \sum_{j=1}^{N_c} \alpha_j g_j(u') du' = \sum_{j=1}^{N_c} \alpha_j \int_{\bar{\Omega}} \frac{f(u')}{g(u')} \Big|_{u'=u'(v'_j)} dv'_j. \quad (6)$$

The channel weights α_j can thereby be adjusted dynamically such that the variance of the integral gets minimised [29].

To sample *unit-weight events* from the target function $f(u)$, typically a rejection sampling algorithm [71] is employed that utilises the maximal event weight in the integration volume, w_{\max} . A sample of N^{trials} weighted events is thus converted into a set of $N \leq N^{\text{trials}}$ unweighted events, where N corresponds to the number of accepted events. The related unweighting efficiency for large N is given by

$$\epsilon := \frac{N}{N^{\text{trials}}} = \frac{\langle w \rangle_{N^{\text{trials}}}}{w_{\max}}. \quad (7)$$

Its inverse determines the average number of target-function evaluations needed before an event is accepted with unit-weight.

An exact determination of w_{\max} is often neither possible – given finite statistics – nor desirable in a numerical calculation that might exhibit a few points with spuriously large weights, as this would yield a prohibitively small unweighting efficiency. Instead, to avoid being dominated by such rare outliers, there are various possibilities to define a reduced maximum such that some “*overweight*” events with $w > w_{\max}$ are allowed and will be assigned a correction weight $\tilde{w} = w/w_{\max}$, effectively leading to *partially unweighted events*³. Ref. [59] proposed a bootstrap method where the maximum is given by the median of n determinations from

²In the following we drop the index i as we are always referring to the generation of a single event.

³While the event weight w is typically a dimensionful quantity, in unweighted events the weights \tilde{w} are considered dimensionless. To obtain the correct normalisation of a differential cross section, e.g. in a histogram, they need to be normalised to the *generated inclusive cross section* as reported by the event generator, $\tilde{w}_i \rightarrow \tilde{w}_i \cdot \frac{\sigma_{\text{gen}}}{\sum_j \tilde{w}_j}$.

independent event batches. A more conventional approach would be the exclusion (from the maximum definition) of large-weight events with a certain quantile of the cross section⁴. In what follows we will make use of both techniques. The classical unweighting algorithm with overweight treatment for generating a single event is sketched in Alg. 1.

Algorithm 1: The classic rejection-sampling unweighting algorithm.

```

while true do
  generate phase-space point  $u$ ;
  calculate exact event weight  $w$ ;
  generate uniform random number  $R \in [0, 1)$ ;
  if  $w > R \cdot w_{\max}$  then
    | return  $u$  and  $\tilde{w} = \max(1, w/w_{\max})$ 
  end
end

```

The application of variance-reduction methods will typically also lead to an improved unweighting efficiency ϵ . In fact, an optimal sampler would directly produce event weights $w = \text{const}$, resulting in an unweighting efficiency of 100%. However, in realistic use cases this is never achieved. For high-multiplicity scattering processes unweighting efficiencies are instead often well below 1% [25, 59]. To systematically improve ϵ one needs to reduce w_{\max} . The FOAM [72, 73] algorithm attempts to achieve this and aims for an optimised unweighting efficiency by gaining control over the maximal event weight.

2.1 A novel unweighting procedure

We here propose an alternative method aiming for a reduction of the computational resources needed to produce unweighted events that follow the desired target distribution. This can be achieved through a light-weight surrogate for the full event-weight calculation that enters a two-staged rejection-sampling algorithm. Given such a local surrogate s for the true event weight w , that can for example be obtained from a well-trained neural-network predictor, cf. Sec. 3, we can use this surrogate in an initial rejection sampling against the maximal event weight w_{\max} . However, to ultimately sample from the correct distribution, we need to account for the mismatch between the estimated and the actual event weight. This is accomplished with a correction factor $x = w/s$. This factor could be applied as an additional weight to accepted events, or a second rejection sampling step can be added to unweight this against the (predetermined) maximum, x_{\max} , see below. The resulting unweighting algorithm for generating a single unit-weight event is sketched in Alg. 2 and explained in more detail in the following.

For a fast surrogate perfectly reproducing the exact weights, *i.e.* $x = 1$, the potential for saving resources is maximal, even though the unweighting efficiency obtained with the standard approach is not altered. This is the case, because for all trial configurations failing the first step only the surrogate gets evaluated, while the full weight is computed for accepted events only. However, in practice this is not realistic and the x will vary around unity. Note, we do not require the approximation s to overestimate w , and thus will also face values $x > 1$.

The appearance of non-unit relative weights x makes a second unweighting step convenient. To this end, we need to predetermine the maximum x_{\max} , against which to perform the additional rejection-sampling. Again, to avoid being dominated by rare outliers, we reduce x_{\max} in a controlled way by either excluding a certain quantile of the largest weights or using

⁴This is also the default in SHERPA for the standard rejection-sampling method.

Algorithm 2: Two-stage rejection-sampling unweighting algorithm using an event-wise weight estimate.

```

while true do
  generate phase-space point  $u$ ;
  calculate approximate event weight  $s$ ;
  generate uniform random number  $R_1 \in [0, 1)$ ;
  # first unweighting step
  if  $s > R_1 \cdot w_{max}$  then
    calculate exact event weight  $w$ ;
    determine ratio  $x = w/s$ ;
    generate uniform random number  $R_2 \in [0, 1)$ ;
    # second unweighting step
    if  $x > R_2 \cdot x_{max}$  then
      return  $u$  and  $\tilde{w} = \max(1, s/w_{max}) \cdot \max(1, x/x_{max})$ 
    end
  end
end

```

the median from several independent x_{max} determinations. We correct for the mismatch with the overweight x/x_{max} when $x > x_{max}$. The final weight for an accepted event u is then given by

$$\tilde{w} = \max\left(1, \frac{s}{w_{max}}\right) \cdot \max\left(1, \frac{x}{x_{max}}\right). \quad (8)$$

As consequence of this residual weight, one might need to generate more events using the surrogate approach to achieve the same statistical accuracy as in standard unweighting. To account for this, we use the Kish effective sample size N_{eff} [74] in the following,

$$N_{eff} := \frac{(\sum_i \tilde{w})^2}{\sum_i \tilde{w}^2} = \alpha N, \quad (9)$$

where the sums run over all N events passing the second unweighting and we introduced the proportionality factor $\alpha \leq 1$. The statistical accuracy of the sample is given by $1/\sqrt{N_{eff}}$. Only when using the true maximal weight x_{max} , the effective sample size equals N , corresponding to $\alpha = 1$.

We can now introduce the effective gain factor f_{eff} of the described two-staged unweighting procedure:

$$\begin{aligned}
 f_{eff} &:= \frac{T_{standard}}{T_{surrogate}} \\
 &= \frac{N_{eff} \cdot \frac{\langle t_{full} \rangle}{\epsilon_{full}}}{N \cdot \left(\frac{\langle t_{surr} \rangle}{\epsilon_{1st,surr} \epsilon_{2nd,surr}} + \frac{\langle t_{full} \rangle}{\epsilon_{2nd,surr}} \right)} \\
 &= \alpha \cdot \frac{1}{\frac{\langle t_{surr} \rangle}{\langle t_{full} \rangle} \cdot \frac{\epsilon_{full}}{\epsilon_{1st,surr} \epsilon_{2nd,surr}} + \frac{\epsilon_{full}}{\epsilon_{2nd,surr}}}. \quad (10)
 \end{aligned}$$

It accounts for all timing, efficiency, and statistical differences in the proposed event generation with Alg. 2 compared to standard (partially) unweighted event generation with Alg. 1. Here

$\langle t_{\text{full}} \rangle$ and $\langle t_{\text{surr}} \rangle$ denote the average evaluation times of the full weight and the surrogate, respectively. The quoted unweighting efficiencies are given by

$$\epsilon_{\text{full}} := \frac{N}{N_{\text{full}}^{\text{trials}}}, \quad \epsilon_{1\text{st,surr}} := \frac{N_{2\text{nd,surr}}^{\text{trials}}}{N_{1\text{st,surr}}^{\text{trials}}} \quad \text{and} \quad \epsilon_{2\text{nd,surr}} := \frac{N}{N_{2\text{nd,surr}}^{\text{trials}}}, \quad (11)$$

where the $N_{\text{step}}^{\text{trials}}$ denote the number of trials used in the respective unweighting step. We point out that phase-space cuts are applied before unweighting and therefore events rejected due to cuts do not count towards the number of trials here.

Significant speed gains can be expected if the standard unweighting efficiency ϵ_{full} is rather low and the surrogate approximates the true weights well, *i.e.* $\epsilon_{1\text{st,surr}} \approx \epsilon_{\text{full}}$ and $\epsilon_{2\text{nd,surr}} \approx 1$, while still being significantly faster, *i.e.* $\langle t_{\text{surr}} \rangle \ll \langle t_{\text{full}} \rangle$.

Note, the gain factor f_{eff} has to be understood as an *upper bound* of a potential CPU time saving in an overall budget, as it does not apply to other stages of the event generation like parton showering and, more importantly, also not to post-processing steps like a detector simulation.

2.2 Generalisation to non-positive event weights

The above described unweighting method can easily be extended to the case of non-positive event weights. These naturally appear in higher-order perturbative calculations based on local subtraction methods such as Catani–Seymour [75] or Frixione–Kunszt–Signer [76] subtraction for next-to-leading-order (NLO) QCD calculations. In approaches matching and merging NLO matrix elements with QCD parton showers negative-weight events can resolve potential double counting of hard real-emission contributions and shower emissions off Born-like configurations, see for instance [77, 78]. However, the appearance of such negative weights reduces the statistical significance of a fixed-size event sample as possibly large cancellations take place. In corresponding unweighted samples events contribute with weights ± 1 . The generalisation of the standard unweighting algorithm allowing for negative weights is given in Alg. 3. We thereby make use of a single maximal weight $w_{\text{max}} = |w|_{\text{max}} > 0$ in the rejection sampling, that is determined by the largest weight modulus observed in an initial exploration run⁵.

Algorithm 3: Standard rejection-sampling unweighting algorithm allowing for negative-weight events.

```

while true do
  generate phase-space point  $u$ ;
  calculate exact event weight  $w$ ;
  generate uniform random number  $R \in [0, 1)$ ;
  if  $|w| > R \cdot w_{\text{max}}$  then
    return  $u$  and  $\tilde{w} = \text{sgn}(w) \cdot \max(1, |w|/w_{\text{max}})$ 
  end
end

```

This can be extended to our two-staged unweighting approach, using a surrogate for the full event weight that can now also become negative, *cf.* Alg. 4. We still employ a single maximal weight modulus in the first rejection step, where correspondingly we have to use the modulus of the surrogate, *i.e.* $|s|$. Similarly, for the second rejection sampling we use the modulus of the estimate for the maximal ratio between the full and the surrogate weights. Note

⁵As before, we consider the reduction of the maximum, compensated for by partial over-weighting.

that the sign of the ratio w/s is not unique, as the surrogate s might sometimes get the sign of the true weight wrong. Accordingly, we have to use $x = |w/s|$ also in the (partial) overweighting. The absolute weight value of an accepted event is still given by Eq. (8), however, its sign is determined by $\text{sgn}(\tilde{w}) = \text{sgn}(w)$.

Algorithm 4: Two-stage rejection-sampling algorithm, allowing for negative valued (surrogate) weights. We hereby assume $w_{\max} > 0$ and $x_{\max} > 0$ given by the respective maximal modulus determined in a pre-run.

```

while true do
  generate phase-space point  $u$ ;
  calculate approximate event weight  $s$ ;
  generate uniform random number  $R_1 \in [0, 1]$ ;
  # first unweighting step with  $w_{\max} > 0$ 
  if  $|s| > R_1 \cdot w_{\max}$  then
    calculate exact event weight  $w$ ;
    determine ratio  $x = |w/s|$ ;
    generate uniform random number  $R_2 \in [0, 1]$ ;
    # second unweighting step with  $x_{\max} > 0$ 
    if  $x > R_2 \cdot x_{\max}$  then
      return  $u$  and  $\tilde{w} = \text{sgn}(w) \cdot \max(1, |s|/w_{\max}) \cdot \max(1, x/x_{\max})$ 
    end
  end
end
end

```

To illustrate and validate the proposed algorithm we consider a simple 1d example by sampling from the target distribution

$$f(u) = u^2 - 0.25, \quad \text{for } u \in [0, 1]. \quad (12)$$

As surrogate we here just use a piecewise constant function over $u \in [0, 1]$ given by

$$s(u) = -0.25\chi_{[0,0.2)}(u) - 0.15\chi_{[0.2,0.4)}(u) + 0.05\chi_{[0.4,0.6)}(u) + 0.25\chi_{[0.6,0.8)}(u) + 0.75\chi_{[0.8,1]}(u), \quad (13)$$

where

$$\chi_M(u) = \begin{cases} 1 & : u \in M \\ 0 & : u \notin M \end{cases}. \quad (14)$$

This encloses the cases that the surrogate over- or underestimates the target, as well as predicting its sign wrongly. In the left panel of Fig. 1 we compile the target distribution, the surrogate and their ratio. Furthermore, we mark the maximum used in the second unweighting step, *i.e.* $x_{\max} = 1.5$. This is chosen such that there are regions where $|f(u)/s(u)| > x_{\max}$, triggering the appearance of events with weight $|w| > 1$. In the right panel of Fig. 1 we present the distributions obtained from 500k events generated with the standard unweighting algorithm and the two-staged approach. Comparing to the true target distribution we see that both methods produce the desired density. To further confirm the proper treatment for those events where $x > x_{\max}$, we provide a close-up view of the region around $u = 0.6$. In the standard approach the unweighting efficiency is $\epsilon_{\text{full}} = 0.33$, requiring $N_{\text{full}}^{\text{trials}} \approx 1.5\text{M}$ calls of the target function to generate 500k unit-weight events. In contrast, with the given surrogate and the choice of x_{\max} we obtain $\epsilon_{1\text{st,surr}} = 0.39$ and $\epsilon_{2\text{nd,surr}} = 0.58$, corresponding to $N_{\text{surr}}^{\text{trials}} \approx 2.2\text{M}$. However, for the given event sample we only had to evaluate the target $N_{\text{full}}^{\text{trials}} \approx 875\text{k}$ times.

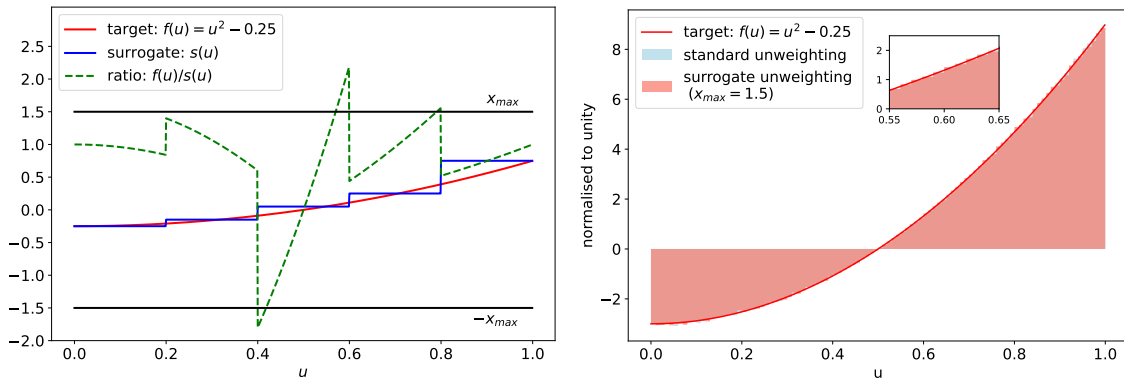


Figure 1: One-dimensional toy example for applying the standard unweighting algorithm and the two-staged surrogate method to a non-positive target function. The left panel shows the target (red) and the employed surrogate (blue), given by Eq. (13), as well as their ratio (green dashed). Indicated is the (capped) maximal ratio x_{\max} used in the second unweighting step. The right panel contains the comparison of the distributions of generated events with the true target.

3 Machine learning event weights

The calculation of transition matrix elements for complicated scattering processes, in particular when considering higher-order corrections, becomes computationally very expensive. In applications that require a large number of repeated evaluations this poses a severe bottleneck. The generation of unweighted events considered here is only one such example, others include the fitting of parton density functions (PDFs), or scans over large parameter spaces in searches for New Physics, *i.e.* corresponding limit-setting procedures.

For the fast evaluation of fixed-order differential cross sections needed in the determination of PDFs interpolation grids such as APPLGRID [79], FASTNLO [80], and PINEAPPL [81] are widely used, and there exist tools for their largely automated construction [82, 83]. To facilitate and accelerate analyses searching for New Physics, there have recently been efforts to use deep-learning techniques for the regression of cross-section integrals [84–86]. Very recently also the approximation of scattering matrix elements rather than integrated cross sections through neural networks has been addressed by several groups [87–90]. These approaches suggest that high-quality surrogates for full scattering matrix elements are feasible, offering potential for significant speed-ups in the event-generation process when applied within the unweighting framework described above.

3.1 Neural-network based matrix-element emulation

For a first application of the surrogate-based unweighting, we introduce a custom ML model, which learns and predicts the complete weight of partonic scattering events. This combines the squared matrix element and the phase-space weight, the latter including Jacobian factors J_{Φ_n} from variable mappings of the Lorentz invariant phase-space element Φ_n used by the underlying integrator. For a given $2 \rightarrow n$ parton-level process our surrogate $s(p_a, p_b, p_1, \dots, p_n)$ thus approximates the following part of the fully differential cross section:

$$d\sigma_{ab \rightarrow n} \Big|_{p_a, p_b, \{p_i\}} = \underbrace{f_a(x_a, \mu_F) f_b(x_b, \mu_F) |\mathcal{M}_{ab \rightarrow n}|^2}_{\approx s} \Big|_{J_{\Phi_n}} dx_a dx_b d\Phi_n \Big|_{p_a, p_b, \{p_i\}} \cdot \quad (15)$$

Here $f_{a/b}$ denotes the PDF for the incoming parton a/b with momentum fraction $x_{a/b}$,

evaluated at factorisation scale μ_F . Note, the PDF contribution could also be factored out of the surrogate and evaluated exactly on an event-wise basis but we here decided to include it. The external particle momenta satisfy four-momentum conservation and on-shell conditions:

$$p_a + p_b = \sum_{i=1}^n p_i, \quad p_{a/b}^2 = 0, \quad \text{and} \quad p_i^2 = m_i^2 \quad (\forall i = 1, \dots, n). \quad (16)$$

Accordingly, the dimensionality of the physical phase space is $d = 3n - 4 + 2$.

When comparing Eq. (15) to the first identity in the multi-channel integral given by Eq. (6), we identify the phase-space element $dx_a dx_b d\Phi_n$ in momentum space with the differential du' multiplied by the multi-channel density $\sum_j \alpha_j g_j(u')$. The Jacobian factor $|J_{\Phi_n}|$ corresponds to $1/g(u')$. Our NN thus has to approximate the ratio $f(u')/g(u')$, that is obviously dependent on the total importance sampling density g , but not on the very channel used to produce the phase-space point, see also Ref. [69].

Alternatively to Eq. (15) one could approximate the squared matrix element only, *i.e.* $s' \approx |\mathcal{M}_{ab \rightarrow n}|^2$, and fully calculate the Jacobian factors for each phase-space point. Due to its factorised nature, this approach would in fact be easier to implement. However, it suffers from the significant costs of evaluating the phase-space weight for multi-leg processes, which can sometimes even rival the evaluation cost of the matrix element. Furthermore, the combination of Jacobian factors and matrix elements often yields a smoother function over phase space. We thus only consider the approach of replacing the combined matrix-element and phase-space weight with a fast surrogate here.

Various test cases for surrogate models were considered in the course of this work, including (boosted) decision trees, random forests and neural networks. While being faster⁶, random forests and (boosted) decision trees yield a poorer prediction accuracy, rendering them inadequate for an application in the surrogate-based unweighting [91]. Thus, only neural networks are discussed further in the following.

Given the specific role of the surrogate in the proposed unweighting procedure, we seek for light-weight network architectures, flexible enough to approximate the weight of high-multiplicity scattering events well, and fast to evaluate. To this end we employ rather simple multi-layer feedforward fully connected neural networks (NN).

As input-layer variables we use the three-momentum components of the initial- and final-state particles⁷, *i.e.* $3n + 2$ inputs. In general, any set of variables that has an injective mapping to the phase-space point could be used, even with different dimensionality if adding or removing variables.

One might alternatively consider a particular set of input variables, namely the random numbers v_i from the phase-space sampling, which have been mapped into momenta as described by Eqs. (4) and (6). While this is straightforward for simple sampling methods, it becomes more tricky for multi-channel samplers. Here, the mapping between random numbers and phase-space point is not unique, but depends on the randomly chosen channel $j = 1 \dots N_c$. To remedy this situation, one could either train a separate NN for each phase-space channel j , or one could add the channel number j (or the random number determining it) as another input variable. We postpone a study of these possibilities to future works.

The single output variable of our NN corresponds to the real-valued event weight. The network is further defined by the number of hidden layers and the set of nodes per layer as detailed in Table 1. As output activation function for the network nodes we use the Rectified

⁶The prediction speed of the machine-learning models depends on their architecture. One can construct simple neural networks which are able to predict faster than a very deep decision tree. However, the accuracy and ability to generalise may decrease with simpler topologies.

⁷Note, we here assume the initial-state momenta of partonic scattering events to be collinear with the incoming beams, *i.e.* along the $\pm z$ -axis, such that their x and y components vanish.

Linear Unit (ReLU) [92]. We use HE weight initialisation [93] and train the NN with the ADAM optimiser [94].

The practical implementation of NN training in the SHERPA framework and the interface for (general) surrogate models for application in event unweighting will be detailed in Sec. 4. In the remainder of this section, however, details on the hyperparameters of our NN and the training procedure are given. The NN performance is first studied for the example process $gg \rightarrow e^-e^+ggd\bar{d}$. We used this channel as a test bed for investigations on the NN performance in terms of timing and the quality of the event-weight predictions as a function of the hyperparameters. Being primarily interested in a conceptual proof-of-concept and an initial estimation of the method’s potential to save resources in event unweighting, we do not attempt to systematically optimise the NN setup. Furthermore, while in principle different scattering processes might get better approximated by a different NN architecture, we will employ the hyperparameter set found in the following example also in our other applications presented in Sec. 4.

3.2 An example: $gg \rightarrow e^-e^+ggd\bar{d}$

We consider the partonic channel $gg \rightarrow e^-e^+ggd\bar{d}$ at the leading order, *i.e.* $\mathcal{O}(\alpha^2\alpha_s^4)$, that represents a tree-level contribution to $Z+4$ jets production at the LHC. Correspondingly, the input-parameter space for the NN here is 20-dimensional. The fiducial phase space used in the training and for the predictions is constrained by requiring a dilepton invariant mass $m_{e^-e^+} > 66$ GeV and four anti- k_t jets [95] with radius parameter $R = 0.4$ and $p_{T,j} > 20$ GeV. We consider a proton–proton centre-of-mass energy of $\sqrt{s} = 13$ TeV, and use the NNPDF-3.0 NNLO PDF set [96]. As matrix-element and phase-space generator we employ AMEGIC [10] in the framework of SHERPA-2.2.

Our NN has four hidden layers with 128 nodes each. The training dataset consists of 1M events generated with SHERPA after the optimisation phase of the AMEGIC integrator. We split the dataset such that 80% of the events are used for training and 20% for validation. In order to normalise the input features, we scale the momenta to the range $[-1, 1]$ using min-max normalisation with the min-max values given by $\pm\sqrt{s}/2$. As the values of the weights can span several orders of magnitude, we take the logarithm of the weights in order to avoid numerical problems. The NN model is fitted to the data by minimising the mean squared error (MSE) loss using the ADAM optimiser with a learning rate of 10^{-3} . We use a batch size of 1000 and train in epochs containing all training points in random order. Early stopping is used to end the training when the validation loss does not decrease for 30 epochs and save the model with the lowest validation loss. Like for the training we also use the MSE loss for validation. Fig. 2 shows the convergence behaviour of our model. One can see that the loss decreases fairly smoothly and that the variations between different initialisations of the model are small.

To test the quality of our trained NN surrogate s for the true event weights w we present in Fig. 3a the resulting distribution of $x = w/s$ for 1M phase-space points generated with SHERPA. The x -distribution is centred around $x = 1$, rather symmetric, and falls off quite steeply. This confirms that the chosen NN is indeed suitable for a prediction of the event weight. Still we observe that the tails of the distribution stretch beyond $|\log_{10}(x)| > 4$, meaning the NN sometimes severely over- or underestimates the true weight. In particular the largest x -values will affect the performance of the unweighting algorithm proposed in Sec. 2.1, as they determine the maximum x_{\max} against which to perform the second rejection sampling. Fig. 3b shows that the largest and smallest values of x correspond to small values of w . As opposed to this, the NN approximation is much better for higher values of w as can be recognized by the smaller spread of the x -values. This behaviour can be expected given the MSE loss function used for the training of the NN. While Fig. 3b shows that the largest relative deviations can be found for small values of w , the absolute deviations in that region are actually small. The MSE

Table 1: Summary of hyperparameters specifying the employed feedforward NN architecture and the means of training.

NN Hyperparameter	Value
Hidden Layers	4
Nodes per Layer	128
Activation Function	ReLU
Loss Function	MSE
Optimiser	ADAM
Learning Rate	10^{-3}
Batch Size	1000

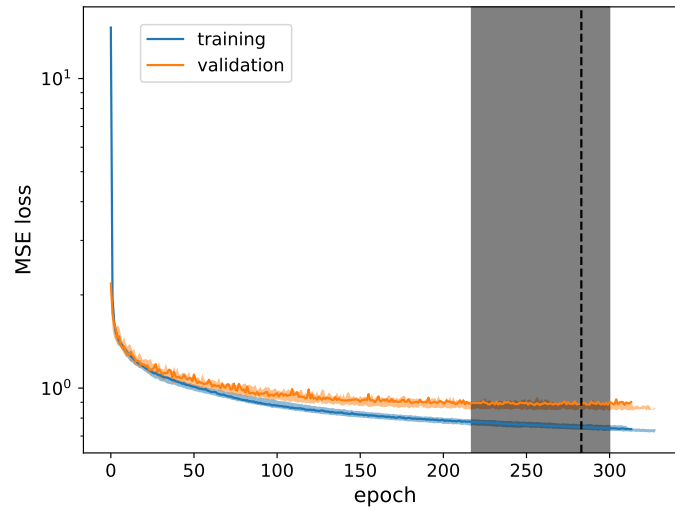


Figure 2: Training (blue) and validation (orange) MSE loss of the best performing NN during training. The dashed line illustrates the stopping point due to early stopping. The coloured bands show the variations from ten independently trained initialisations of the same model.

loss function penalizes absolute deviations at larger w -values more than at smaller w -values which leads to larger relative deviations for small values of w .

As described already in the context of the first unweighting step in Alg. 1, we can use maximum-reduction techniques also for x_{\max} in the second rejection sampling. These will reduce the sensitivity to the tail of the weight distribution and in particular rare outliers by using a reduced maximum, again at the price of a partial overweighting of events. In our performance study in Sec. 4 we will employ two reduction techniques. The first being the *quantile reduction method*, where we define $x_{\max}^{\text{p.m.}}$ such that the remaining overweights contribute at most 1‰ to the total cross section σ . We consider an event sample of $N = 1\text{M}$ events with weights $\{w_i\}$. For reference, in the standard unweighting method we can determine $w_{\max}^{\text{p.m.}}$ by sorting the sequence of weights $\{w_i\}$ such that $w_i \leq w_{i+1}$ and requiring that

$$w_{\max}^{\text{p.m.}} := \min \left(w_j \left| \sum_{i=j+1}^N w_i < 0.001 \cdot \sum_{i=1}^N w_i \right. \right). \quad (17)$$

The equivalent procedure for our two-stage unweighting method is to calculate the values of

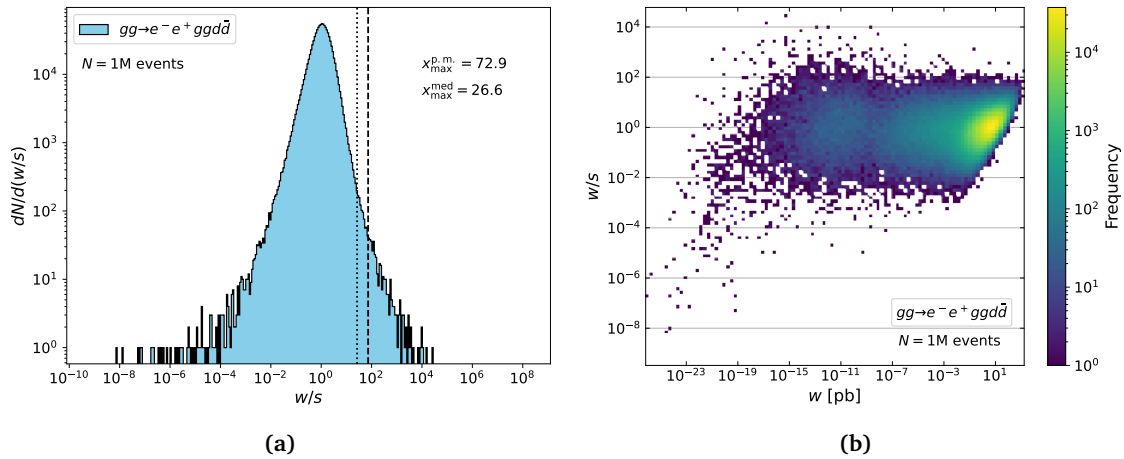


Figure 3: Distribution of weights using 1M test points generated with SHERPA for the process $gg \rightarrow e^-e^+ggd\bar{d}$ in proton–proton collisions at $\sqrt{s} = 13$ TeV. **(a)** One-dimensional histogram of the ratio $x = \frac{w}{s}$. The two vertical lines indicate the values of the reduced weight maxima $x_{\max}^{\text{p.m.}}$ (dashed) and x_{\max}^{med} (dotted). **(b)** Two-dimensional histogram showing the relationship between the ratio $x = \frac{w}{s}$ and the true event weight w .

s and x for all events and to sort the sequence $\{x_i\}$ such that $x_i \leq x_{i+1}$ and to use the same order for the $\{s_i\}$. The reduced maximum is then defined as

$$x_{\max}^{\text{p.m.}} := \min \left(x_j \left| \sum_{i=j+1}^N x_i s_i < 0.001 \cdot \sum_{i=1}^N x_i s_i \right. \right). \quad (18)$$

As a somewhat more aggressive alternative we introduce the *median reduction method*. Here we consider $N_{\text{1st,surr}}^{\text{trials}} = 1\text{M}$ trial points for which we perform the first unweighting $n = 50$ times with different random seeds. For the accepted events in each iteration we determine x_{\max} . From the final set of maxima we then determine the median x_{\max}^{med} , *i.e.*

$$x_{\max}^{\text{med}} := \text{med} \left(\{x_{\max}^i\} \right). \quad (19)$$

For our example process the resulting values of $x_{\max}^{\text{p.m.}}$ and x_{\max}^{med} are illustrated by the vertical dashed and dotted line in Fig. 3a, respectively. In this specific example we obtain $x_{\max}^{\text{p.m.}} \approx 73$ and $x_{\max}^{\text{med}} \approx 27$, which corresponds to a reduction of $x_{\max}^{\text{p.m.}}$ by about two orders of magnitude with respect to the naive maximum, and an additional factor of three when using the median approach.

We close this section with a comment on the timings for the evaluation of the matrix element and the NN surrogate for a single phase-space point. On average the evaluation of the full event weight for the $gg \rightarrow e^-e^+ggd\bar{d}$ process from AMEGIC takes about 85 ms^8 . In contrast, for the NN model this just takes 0.13 ms , which translates into a speed-up of

$$\frac{\langle t_{\text{full}} \rangle}{\langle t_{\text{surr}} \rangle} \approx 650. \quad (20)$$

⁸The quoted times correspond to the evaluation on a single core of an Intel[®] Xeon[®] Processor E5-2680 v3 @ 2.50GHz.

4 Surrogate-based unweighting: Implementation, Validation and Results

The event-weight estimator from Sec. 3.1 is optimally suited to be used as light-weight surrogate in the two-stage unweighting method presented in Sec. 2.1. In the following, we will briefly describe the implementation of the algorithm in the SHERPA generator framework. As a first application we will again consider the $gg \rightarrow e^-e^+ggd\bar{d}$ process. We will then benchmark the method in a variety of partonic channels contributing to $W+4$ jets and $t\bar{t}+3$ jets production at the LHC and validate the obtained results.

4.1 Implementation in the SHERPA framework

The SHERPA framework embeds modules to automatically construct the transition matrix elements and suitable multi-channel integrators for in-principle arbitrary tree-level processes. To this end it has two matrix-element generators built-in, AMEGIC and COMIX. Our current implementation of the novel unweighting algorithm employs the AMEGIC generator.

In an initial optimisation phase the integrator is adapted to the specific process and fiducial phase space using the channel-weight optimisation described in [29]. During the integration phase the value of w_{\max} is determined based on the quantile approach. We use the SHERPA default of letting overweighted events have a relative contribution of 1‰ to the inclusive cross section. The optimised generator is then used to produce a sample of 2M weighted events. We use the first 1M events as training (80%) and validation (20%) data for our NN model.⁹ For the NN implementation and training we use KERAS [97] with the TENSORFLOW [98] backend. The model parameters leading to the lowest validation loss are written out as an HDF5 [99] file. While KERAS is based on Python, SHERPA is written in C++. To use the KERAS model in SHERPA without having to rely on an interface we use the header-only library frugally-deep [100] which runs the model in prediction mode on a single CPU core.

The second 1M events are used to determine the x_{\max} for the second unweighting using the per mille quantile or median approach. For the latter we consider $n = 50$ independent iterations over the data set. This procedure is repeated for ten independently trained NN models and we finally choose the one achieving the lowest x_{\max} on the test dataset to be used in the following. The NN and the value of x_{\max} then serve as inputs to SHERPA for subsequent event-generation runs. We use different events for the determination of x_{\max} than for the training of the NN. If one were to use the same data set, x_{\max} would likely be underestimated. With data not seen by the model during training, however, we get a much more reliable estimate.

For the performance analysis we log several quantities during the event generation. To determine the efficiencies, we count the numbers of trials for the first and second unweighting steps. Also, we measure the time it takes on average to evaluate the surrogate by taking the sum of user and system time spent in the respective parts of the code.

4.2 An example: $gg \rightarrow e^-e^+ggd\bar{d}$

Before proceeding with the application of our novel unweighting approach to $W+4$ jets and $t\bar{t}+3$ jets production at the LHC, we examine its technical and physics performance in more detail for the example process of $gg \rightarrow e^-e^+ggd\bar{d}$. This is the channel initially used to optimise the NN performance in terms of timing and accuracy, cf. Sec. 3.2.

⁹In a production implementation in the future, one could also perform the training on the same events that are generated during the integration phase after the integrator optimisation.

Performance analysis

The evaluation of the NN surrogate for a single phase-space point was found to be about 650-times faster than the full weight calculation with AMEGIC. In Fig. 3a we have presented the obtained distribution of $x = w/s$, where we also indicated the reduced maxima for the per mille quantile and the median approach, *i.e.* $x_{\max}^{\text{p.m.}} \approx 73$ and $x_{\max}^{\text{med}} \approx 27$, respectively. Using the trained NN and each of these maxima, we generate from scratch 100k events with our surrogate unweighting algorithm. In Tab. 2 we summarise the obtained efficiency of the default single-stage unweighting, ϵ_{full} , the efficiencies of the first and second rejection-sampling step in the surrogate unweighting, as well as the α -parameter that determines the effective sample size, *cf.* Eq. (9), for the two maximum-reduction methods¹⁰. Lastly, we give the resulting gain factors f_{eff} , *cf.* Eq. (10).

Table 2: Sampling measures for the $gg \rightarrow e^-e^+ggd\bar{d}$ partonic channel in pp collisions at $\sqrt{s} = 13$ TeV. All efficiencies, the sample-size parameters and effective gain factors are determined in the generation of 100k unweighted events.

process: $gg \rightarrow e^-e^+ggd\bar{d}$									
ϵ_{full}	$\epsilon_{1\text{st,surr}}$	$x_{\max}^{\text{p.m.}}$	$\epsilon_{2\text{nd,surr}}^{\text{p.m.}}$	$\alpha^{\text{p.m.}}$	$f_{\text{eff}}^{\text{p.m.}}$	x_{\max}^{med}	$\epsilon_{2\text{nd,surr}}^{\text{med}}$	α^{med}	$f_{\text{eff}}^{\text{med}}$
8.8e−3	6.4e−3	72.9	1.9e−2	0.9982	1.73	26.6	5.1e−2	0.9962	4.69

Using the default unweighting algorithm, AMEGIC achieves an unweighting efficiency of about 0.9%. This in fact is quite remarkable, given that we consider a six-particle final state. When using the NN surrogate we obtain a similar performance, $\epsilon_{1\text{st,surr}} \approx 0.64\%$, and given the fast evaluation time for the surrogate this slightly lower efficiency barely affects the overall performance. More relevant is the second unweighting, for which we find efficiencies of $\epsilon_{2\text{nd,surr}}^{\text{p.m.}} = 1.9\%$ and $\epsilon_{2\text{nd,surr}}^{\text{med}} = 5.1\%$. Accordingly, when using the median-reduction technique, we need to evaluate the full weight roughly a factor 2.7 less often than for the quantile approach. For the considered process this almost directly transfers to the effective gain factors that yield $f_{\text{eff}}^{\text{p.m.}} = 1.73$ and $f_{\text{eff}}^{\text{med}} = 4.69$. These gains are a consequence of the speed of the surrogate evaluation, and its excellent approximation of the true weights, *i.e.* the very steep fall-off of the $x = w/s$ distribution. In fact, the effective sample size reduces only to 99.8% and 99.6% of a unit-weight sample, which will be negligible in practical applications.

The obtained α values close to unity reflect the fact that only few events retain non-unit weights \tilde{w} in the end, *cf.* Eq. (8). This is confirmed by Fig. 4 where we present the final event-weight distribution for the sample of 100k events generated using the more aggressively reduced maximum x_{\max}^{med} in the second unweighting step. Indeed, only a small fraction of events exhibits weights $\tilde{w} > 1$. Furthermore, the overweights rarely exceed $\tilde{w} = 3$ and the maximum we observe within this sample is $\tilde{w} \approx 9$.

Physics validation

To prove that our algorithm indeed produces the correct target distribution we now move to the validation of differential cross sections. Figure 5 collects various physical observables comparing the predictions of SHERPA with and without the novel unweighting approach. For both methods we produced samples of 1M events at the parton level. Parton shower and hadronisation effects are disabled in these and the following simulations to increase the resolution and sensitivity to potential differences between the two approaches. These were analysed with

¹⁰Note, the w_{\max} used in the first unweighting is always reduced using the per mille quantile approach to keep the full and the surrogate approach comparable.

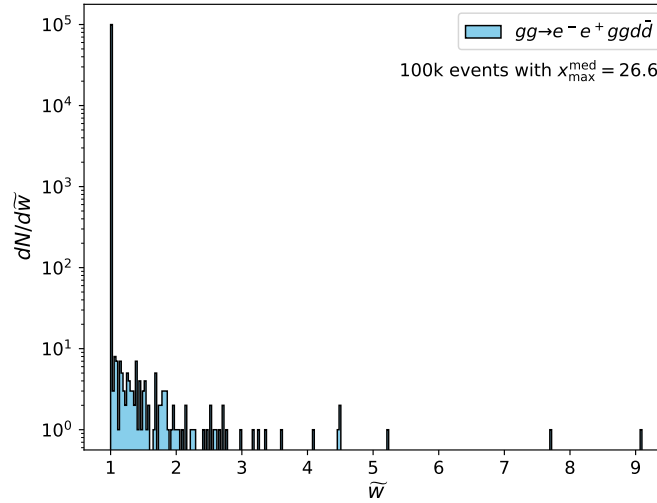


Figure 4: Final event weights \tilde{w} of 100k $gg \rightarrow e^- e^+ gg d \bar{d}$ events in proton–proton collisions at $\sqrt{s} = 13$ TeV generated using surrogate unweighting with χ_{\max}^{med} .

the RIVET3 toolkit [101] using the MC_ZINC and MC_ZJETS analyses. In panel (a) we show the dilepton invariant mass, (b) the dilepton rapidity distribution, (c) the p_T of the jet with highest transverse momentum, and (d) the azimuthal distance between the two leading jets. For each plot we provide two sub-panels. In the first we depict the ratio of the predictions obtained from the surrogate approach and nominal SHERPA, where the errorbars indicate the bin-wise statistical uncertainty. The second panel displays directly the statistical compatibility of the two predictions measured in terms of standard deviations.

For all four observables we find full statistical agreement, which proves that the surrogate approach produces the correct target function. This also applies to the tail of the distributions. No significant increase in the statistical errors is observed for the surrogate-based prediction, which verifies the negligible reduction of $\alpha^{\text{med}} = 0.9962$. Furthermore, there is no visible imprint of statistical fluctuations from the events that exceed the maximum in the second unweighting.

For the considered example process we can conclude that when using the surrogate unweighting approach we can generate samples of almost identical statistical accuracy that reproduce the exact physical distribution. Depending on the method used to reduce the maximum in the second unweighting step we find effective gain factors up to 4.7.

4.3 Results for LHC production processes

In this section we present results for processes contributing to $W+4$ jets and $t\bar{t}+3$ jets production at the LHC, providing further insight into the potential and limitations of the surrogate-unweighting method. Both final states receive contributions from a large number of partonic channels, from which we pick representatives here. Given the high final-state multiplicity, the large number of contributing Feynman diagrams, and the complexity in QCD colour space, these matrix elements are highly non-trivial functions over phase space and rather expensive to evaluate, such that we can expect gains from employing the surrogate method.

In the following we employ the same network architecture and training measures as described in Sec. 3.1 and used in the $Z+4$ jets example and apply them in each partonic channel separately. We do not attempt to specifically adjust and optimise the hyperparameters, though this could potentially further improve performance. As before, all setups are studied with SHERPA-2.2 for pp collisions at $\sqrt{s} = 13$ TeV, using the NNPDF-3.0 NNLO PDF set and AMEGIC

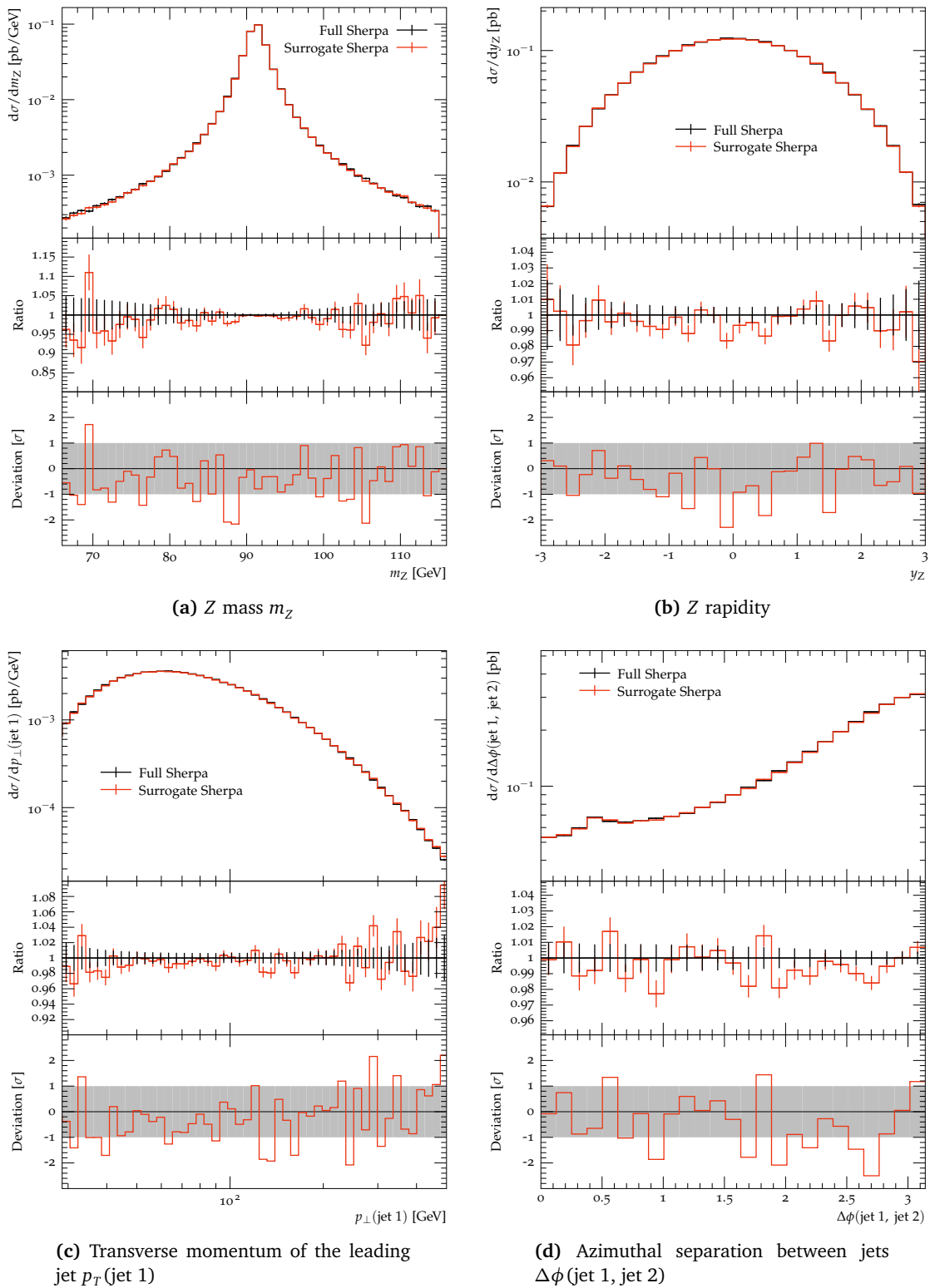


Figure 5: Comparison of different differential distributions generated using SHERPA with (red) and without (black) an NN weight surrogate for the process $gg \rightarrow e^- e^+ gg d \bar{d}$ in proton–proton collisions at $\sqrt{s} = 13 \text{ TeV}$.

as matrix-element and phase-space generator.

Besides quantifying timing improvements, we scrutinise the physics description by validating observable distributions against the standard unweighting approach. It is worth mentioning that all presented timing improvements can likewise be translated into energy savings as no notable additional computing resources are needed for the new approach.

4.3.1 $W+4$ jets

We consider three partonic channels with varying numbers of external gluons that contribute to $W+4$ jets in proton–proton collisions. These are listed along with their respective tree-level production cross section in Tab. 3. While the dimensionality of the input-parameter space for the NN surrogate is identical to the $Z+4$ jets example, we now consider the charged-current weak interaction and different combinations of initial- and final-state partons.

Table 3: Selection of partonic channels contributing to $W+4$ jets production at the LHC and their corresponding leading-order production cross sections.

process	cross section [pb]
$dg \rightarrow e^- \bar{\nu}_e g g g u$	24.5(2)
$dd \rightarrow e^- \bar{\nu}_e g g d u$	4.62(3)
$ud \rightarrow e^- \bar{\nu}_e d u u \bar{d}$	0.0572(3)

The quoted cross sections correspond to a fiducial phase space requiring four anti- k_t jets with $R = 0.4$ and $p_{T,j} > 20$ GeV, and $m_{e^- \bar{\nu}_e} > 1$ GeV. Due to the high total production rate, $W+4$ jets final states constitute an important background to top-quark pair-production and many searches for new physics phenomena. From Tab. 3 we can infer that the cross sections of different partonic channels vary significantly. In particular processes with more external gluons dominate over quarks. An additional driver are the initial-state flavour PDFs. The larger the contribution of a partonic channel to the total $W+4$ jets cross section the more events it will contribute to an inclusive sample. Accordingly, it is desirable to speed up event generation in particular for the dominant production channels.

Performance analysis

In Tab. 4 we compile the performance measures for unweighted event generation separately for the three considered $W+4$ jets partonic channels. They are determined from samples of 100k events generated with the standard and the NN surrogate approach.

Notably, for all three processes the standard unweighting efficiency is lower than for the $Z+4$ jets channel. For the process with four external gluons the evaluation of the surrogate model is again more than 600 times faster than the full weight calculation. However, for the other two cases we achieve speed-up factors of 162 and 25 only. These lower gains originate from shorter evaluation times for the full weights of 20 ms for $dd \rightarrow e^- \bar{\nu}_e g g d u$ and 3 ms for $ud \rightarrow e^- \bar{\nu}_e d u u \bar{d}$, while the NN surrogate takes about 0.12 ms for *each* channel. While the maxima x_{\max}^{med} are all of a similar size as in the $Z+4$ jets case, the values for $x_{\max}^{\text{p.m.}}$ are significantly higher, ranging up to 1650 for $ud \rightarrow e^- \bar{\nu}_e d u u \bar{d}$. This suggests that the NN provides an inferior approximation of the weights for the processes and fiducial phase space considered here. To illustrate this we show in Fig. 6a the distribution of $x = w/s$ for 1M events for the process $dd \rightarrow e^- \bar{\nu}_e g g d u$. When comparing to Fig. 3a we indeed observe a broader distribution that exhibits more pronounced tails. The two vertical lines indicate the values of $x_{\max}^{\text{p.m.}}$ (dashed) and x_{\max}^{med} (dotted). By comparing the relationship between x and w shown in Fig. 6b to the one shown in Fig. 3b, we see that the spread of x -values is much broader overall. However,

Table 4: Performance measures for partonic channels contributing to $W+4$ jets production at the LHC.

	$dg \rightarrow e^- \bar{\nu}_e gggu$	$dd \rightarrow e^- \bar{\nu}_e ggdu$	$ud \rightarrow e^- \bar{\nu}_e duu\bar{d}$
ϵ_{full}	1.4e-3	3.1e-4	3.6e-4
$\epsilon_{1\text{st,surr}}$	7.1e-4	1.1e-4	1.3e-4
$\langle t_{\text{full}} \rangle / \langle t_{\text{surr}} \rangle$	667	162	25
$x_{\text{max}}^{\text{p.m.}}$	234.03	544.96	1642.77
$\epsilon_{2\text{nd,surr}}^{\text{p.m.}}$	8.5e-3	5.2e-3	1.8e-3
$\alpha^{\text{p.m.}}$	0.9953	0.9958	0.9953
$f_{\text{eff}}^{\text{p.m.}}$	1.93	0.29	0.02
$x_{\text{max}}^{\text{med}}$	40.28	30.53	38.53
$\epsilon_{2\text{nd,surr}}^{\text{med}}$	5.3e-2	8.5e-2	7.3e-2
α^{med}	0.9285	0.8204	0.4323
$f_{\text{eff}}^{\text{med}}$	10.36	3.91	0.25

otherwise it shows a similar behaviour with the more extreme values of x corresponding to small values of w .

The efficiencies of the initial unweighting step are also consistently lower than for the neutral gauge-boson channel. In particular for the process without external gluons, where $\langle t_{\text{full}} \rangle / \langle t_{\text{surr}} \rangle$ is 'only' 25, the factor of three between ϵ_{full} and $\epsilon_{1\text{st,surr}}$ might not be negligible. As expected given the larger values of $x_{\text{max}}^{\text{p.m.}}$ the corresponding efficiencies for the second unweighting step are all below 1%, *i.e.* as low as 2‰ for $ud \rightarrow e^- \bar{\nu}_e duu\bar{d}$. However, for the median-reduced maximum the situation improves significantly, with $\epsilon_{2\text{nd,surr}}^{\text{med}}$ in the range of 5 – 8%. This efficiency improvement comes at the expense of the statistical power of the sample. While in the quantile approach the resulting $\alpha^{\text{p.m.}}$ factors are very close to unity, *i.e.* the effective sample size is larger than 99.5% of a true unit-weight sample, we observe more significant fractions of overweights with the median approach. This is true in particular for $dd \rightarrow e^- \bar{\nu}_e ggdu$ ($N_{\text{eff}} \approx 82\%N$) and $ud \rightarrow e^- \bar{\nu}_e duu\bar{d}$ ($N_{\text{eff}} \approx 43\%N$).

These performance measures are condensed into the resulting effective gain factor f_{eff} according to Eq. (10). For the dominant $dg \rightarrow e^- \bar{\nu}_e gggu$ channel we find quite significant gains, even exceeding a factor of ten for the median approach. For the other channels the situation is different. For the all-fermion process surrogate unweighting needs *more* resources than the standard approach. This can be traced back to the relatively fast evaluation of the full weight, due to the simpler form of the matrix element, and the inferior performance of the NN in approximating the true event weights. However, in the global $W+4$ jets context, this channel contributes little to the total production rate and thus relatively few events need to be generated for such a channel. For the intermediate process, $dd \rightarrow e^- \bar{\nu}_e ggdu$, we find $f_{\text{eff}}^{\text{med}} \approx 4$. This speed-gain, however, also goes along with a more sizeable fraction of overweights, yielding $\alpha^{\text{med}} \approx 0.82$. We will therefore compare differential distributions for physical observable for this channel in the median approach next.

Physics validation

In Fig. 7 we present a comparison of physical distributions for the channel $dd \rightarrow e^- \bar{\nu}_e ggdu$ generated with and without the NN surrogate, employing $x_{\text{max}}^{\text{med}}$ in the second unweighting. We show results for (a) the transverse momentum of the charged boson, (b) the k_t 4-jet resolution

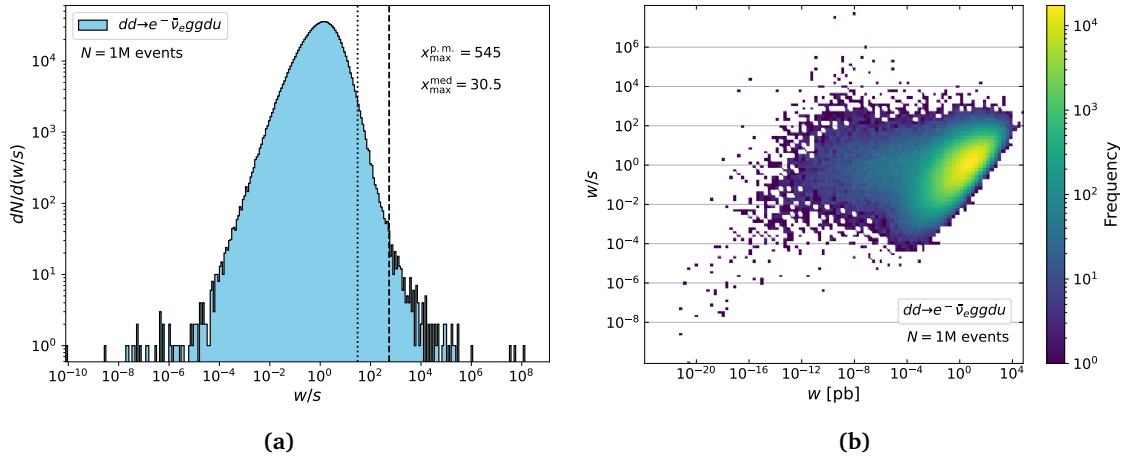


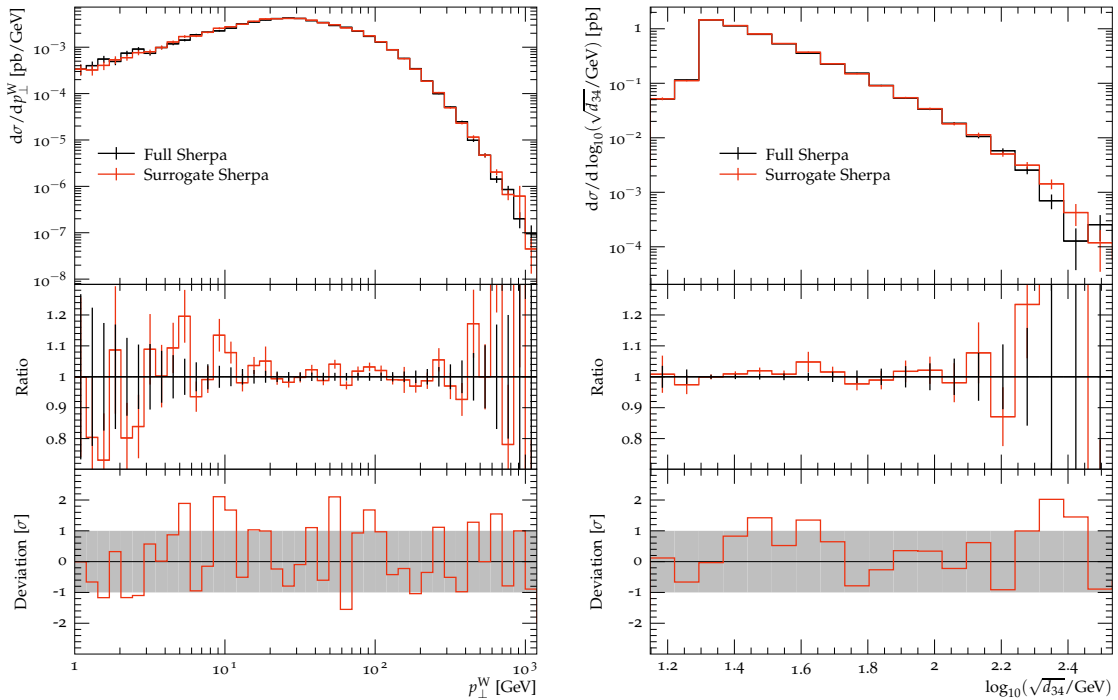
Figure 6: Distribution of weights using 1M test points generated with SHERPA for the process $dd \rightarrow e^- \bar{\nu}_e ggdu$ in proton–proton collisions at $\sqrt{s} = 13$ TeV. **(a)** One-dimensional histogram of the ratio $x = \frac{w}{s}$. The two vertical lines indicate the values of the reduced weight maxima $x_{\max}^{\text{p.m.}}$ (dashed) and x_{\max}^{med} (dotted). **(b)** Two-dimensional histogram showing the relationship between the ratio $x = \frac{w}{s}$ and the true event weight w .

d_{34} , (c) the scalar sum of the four jet transverse momenta, H_T , and (d) the invariant mass of the two leading p_T jets within the RIVET analyses MC_WINC, MC_WJETS, and MC_WKTSPLITTINGS.

For all four differential distributions we observe full statistical compatibility between the two samples of 1M events each. This further underlines that our surrogate-unweighting approach produces the exact target distribution. The considered observables all deeply probe the high- p_T tails of phase space. In fact, the p_T^W and d_{34} distributions extend over five orders of magnitude in cross section. While for the given sample size of $N = 1\text{M}$ we observe significant statistical fluctuations in the tails, these are fully consistent between standard and NN-surrogate generated samples. Even given $\alpha^{\text{med}} \approx 0.82$, corresponding to an effective sample size of $N_{\text{eff}} = 820\text{k}$, neither spikes or bumps are manifest in the nominal distributions, nor a significant increase in the statistical uncertainties for particular observable bins. And with a resulting gain factor of $f_{\text{eff}} \approx 4$, the surrogate method outperforms standard unweighting drastically. However, to some extent and as noted earlier, this statement depends on the post-processing procedures for the events. If the overall generation time of parton-level predictions is small compared to e.g. a full detector simulation, the standard unweighting might be preferable, at least for sub-channels with medium or low f_{eff} .

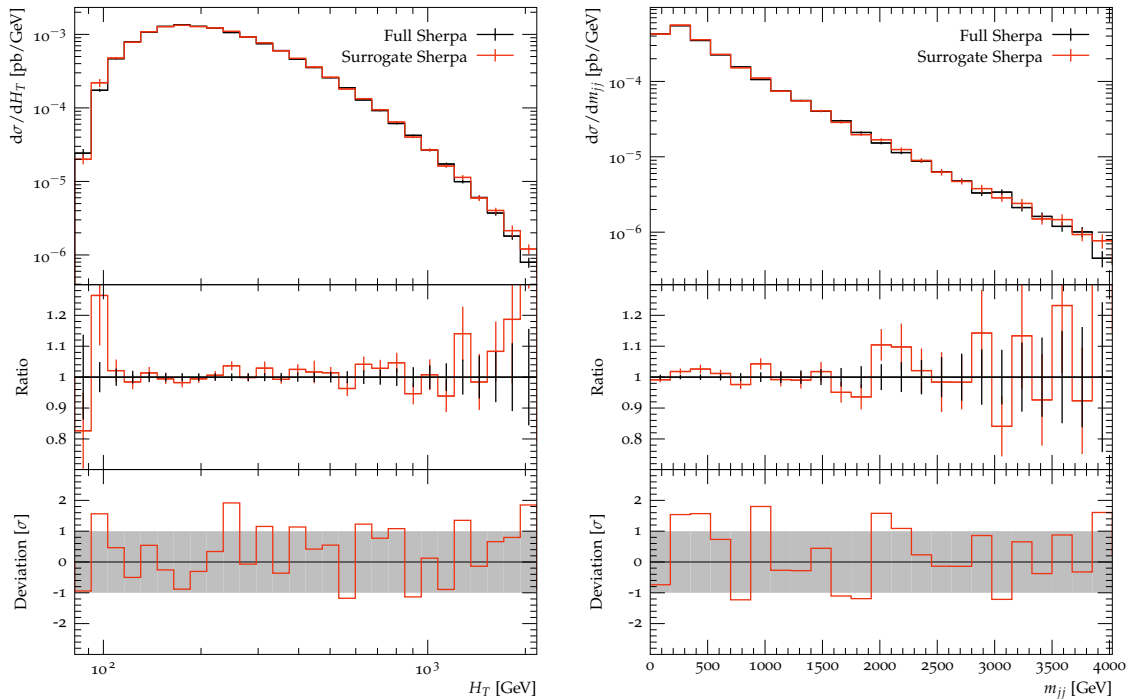
4.3.2 $t\bar{t}+3$ jets

Finally, we present results for processes belonging to the $t\bar{t}+3$ jets group. This probes the generalisation beyond the production of a single electroweak gauge boson in association with jets to a pure QCD process with massive particles. Even though the final state contains one particle less this process still poses a severe challenge. As top quarks carry colour charge there is a significant proliferation of Feynman diagrams when considering their jet-associated production. Despite these differences we employ the same neural-network architecture as before, adjusting the input-space dimensionality for the NN to 17, again utilising the three-momenta as input variables. We require three anti- k_t jets with $R = 0.4$ and $p_{T,j} > 20$ GeV and do not impose phase-space cuts for the external top quarks. The latter are treated as on-shell in the matrix-element calculation, $p_t^2 = p_{\bar{t}}^2 = m_t^2$ with $m_t = 173.4$ GeV, and only decayed a posteriori to allow a more realistic definition of observables in the following physics validation.



(a) W -boson transverse momentum p_T^W

(b) $4 \rightarrow 3$ k_t clustering scale d_{34}



(c) Scalar sum of jet transverse momenta H_T

(d) Dijet invariant mass spectrum m_{jj}

Figure 7: Comparison of different differential distributions generated using SHERPA with (red) and without (black) an NN weight surrogate for the process $dd \rightarrow e^- \bar{\nu}_e ggdu$ in proton–proton collisions at $\sqrt{s} = 13$ TeV.

In Tab. 5 we list the four considered partonic channels and their respective leading-order cross section in proton–proton collisions at $\sqrt{s} = 13$ TeV.

Table 5: Selection of partonic channels contributing to $t\bar{t}+3$ jets production at the LHC and their corresponding leading-order production cross sections.

process	cross section [pb]
$gg \rightarrow t\bar{t}ggg$	108.4(2)
$ug \rightarrow t\bar{t}ggu$	26.00(4)
$uu \rightarrow t\bar{t}guu$	3.733(8)
$u\bar{u} \rightarrow t\bar{t}gd\bar{d}$	0.01840(6)

Clearly, under LHC conditions the all-gluon process has the largest production rate. In the second channel, *i.e.* $ug \rightarrow t\bar{t}ggu$ we instead consider an initial-state up-quark. Given that the QCD interaction does not change flavour, this parton species also appears in the final state. The third channel contains two up-quarks in the initial- and final state, corresponding to t -channel dominance in the top-quark production. The last considered process is $u\bar{u} \rightarrow t\bar{t}gd\bar{d}$, here top-quarks can be produced through s -channel gluons. Note, its production rate and correspondingly its contribution to an inclusive sample of unweighted events is significantly suppressed.

Performance analysis

In Table 6 we collect the performance measures for the surrogate-unweighting approach applied to the four top-quark productions channels. The reference unweighting efficiencies ϵ_{full} for standard unweighting with AMEGIC are typically higher than the ones found for $W+4$ jets before.

Table 6: Performance measures for partonic channels contributing to $t\bar{t}+3$ jets production at the LHC.

	$gg \rightarrow t\bar{t}ggg$	$ug \rightarrow t\bar{t}ggu$	$uu \rightarrow t\bar{t}guu$	$u\bar{u} \rightarrow t\bar{t}gd\bar{d}$
ϵ_{full}	1.1e−2	7.3e−3	6.8e−3	6.6e−4
$\epsilon_{1\text{st,surr}}$	8.7e−3	5.8e−3	4.7e−3	3.6e−4
$\langle t_{\text{full}} \rangle / \langle t_{\text{surr}} \rangle$	39312	2417	199	64
$\chi_{\text{max}}^{\text{p.m.}}$	52.03	32.52	69.76	326.19
$\epsilon_{2\text{nd,surr}}^{\text{p.m.}}$	2.4e−2	3.8e−2	2.1e−2	5.6e−3
$\alpha^{\text{p.m.}}$	0.9989	0.9984	0.9994	0.9981
$f_{\text{eff}}^{\text{p.m.}}$	2.21	4.89	1.47	0.19
$\chi_{\text{max}}^{\text{med}}$	30.40	19.14	27.78	25.34
$\epsilon_{2\text{nd,surr}}^{\text{med}}$	4.3e−2	6.4e−2	5.1e−2	7.1e−2
α^{med}	0.9983	0.9966	0.9943	0.9321
$f_{\text{eff}}^{\text{med}}$	3.90	8.26	3.91	2.22

When comparing the evaluation times for the full event weights and the NN surrogate, quite significant speed-ups are found for $gg \rightarrow t\bar{t}ggg$ and $ug \rightarrow t\bar{t}ggu$. As before, a single evaluation of the surrogate weight takes about 0.12 ms. However, the weight calculation for

the all-gluon channel takes around 5 s, for $ug \rightarrow t\bar{t}gg$ it is still around 0.3 s. Although we observe this high ratio of weight evaluation times, the effective gain factor f_{eff} is much smaller in the end and of the same order of magnitude than for the other processes. This can be mostly attributed to the relatively high unweighting efficiencies we start from. With a value of $\epsilon_{\text{full}} = 1.1 \times 10^{-2}$ the process $gg \rightarrow t\bar{t}gg$ has the highest unweighting efficiency of the examples considered here. According to Eq. 10, this clearly limits the possible gains. The reason for the high values of ϵ_{full} is that this kind of multi-gluon channel is well-optimised in the integrator used by SHERPA.

The results obtained for $x^{\text{p.m.}}$ and x^{med} are less spread out than for the $W+4$ jets processes. For $ug \rightarrow t\bar{t}gg$ the NN performs best, with $x_{\text{max}}^{\text{p.m.}} \approx 33$ and $x_{\text{max}}^{\text{med}} \approx 19$. Only for $u\bar{u} \rightarrow t\bar{t}g\bar{d}\bar{d}$ do we find an inferior performance with $x_{\text{max}}^{\text{p.m.}} > 300$. The values for the efficiency of the first unweighting step are comparable to what we found for the $Z+4$ jets channel, only for $u\bar{u} \rightarrow t\bar{t}g\bar{d}\bar{d}$ it is significantly lower. Similar findings hold for $\epsilon_{\text{2nd,surr}}^{\text{p.m.}}$, which is lowest for $u\bar{u} \rightarrow t\bar{t}g\bar{d}\bar{d}$. All effective sample size parameters are found to be larger than 0.99, with the exception of the $u\bar{u}$ process when using the median reduction method, where $\alpha^{\text{med}} \approx 0.93$.

However, when using $x_{\text{max}}^{\text{med}}$ in the rejection sampling the effective gain factors are all higher than two, being largest for $ug \rightarrow t\bar{t}gg$ with $f_{\text{eff}}^{\text{med}} \approx 8$. For the two computationally most expensive channels, that also feature the largest production rates, we obtain gains larger than two even with the per mille maximum reduction.

Physics validation

We close again by comparing predictions for physical observables, obtained with and without using the weight surrogate for the partonic channel $uu \rightarrow t\bar{t}guu$. Note, the on-shell top-quarks produced in the hard scattering get decayed with SHERPA's decay handler [6] prior to the final-state analysis. We here consider the semi-leptonic decay channel, *i.e.* $t\bar{t} \rightarrow l\nu_l q\bar{q}' b\bar{b}$ and employ the RIVET analysis MC_TTBAR.

In Fig. 8 we present exemplary results for (a) the invariant mass of hadronic W -boson candidates, (b) the H_T distribution of all final-state jets, (c) the invariant mass of the hadronic top-quark candidates, and (d) the transverse momentum of the harder of the two final-state b -quark jets.

As before, we find full statistical agreement between the two samples for all considered observables. The rather fine binning of the invariant-mass distributions leads to larger statistical fluctuations for the given sample size of $N = 1\text{M}$. However, as we will illustrate in Sec. 4.3.3 the deviations are in agreement with perfect statistical compatibility, *i.e.* both samples follow the same target distribution. Given $\alpha^{\text{med}} = 0.9966$ we do not expect and in fact do not observe any visible effects from a reduced statistical accuracy of the sample produced with the surrogate approach.

4.3.3 Summary of physics validation for LHC processes

In addition to the selected observables for the three processes shown in the previous sections, we have performed a statistical compatibility analysis between the full and the surrogate setups based on 190 observables with almost 16,000 bins in total. The predictions are normalised in each observable for this analysis, to avoid a sensitivity to differences in the integrated cross section of each run, which would otherwise have to be accounted for as a correlation between different bins. As can be seen in Fig. 9, the deviations follow a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 0$ and $\sigma = 1$, thereby validating our approach as faithful and unbiased.

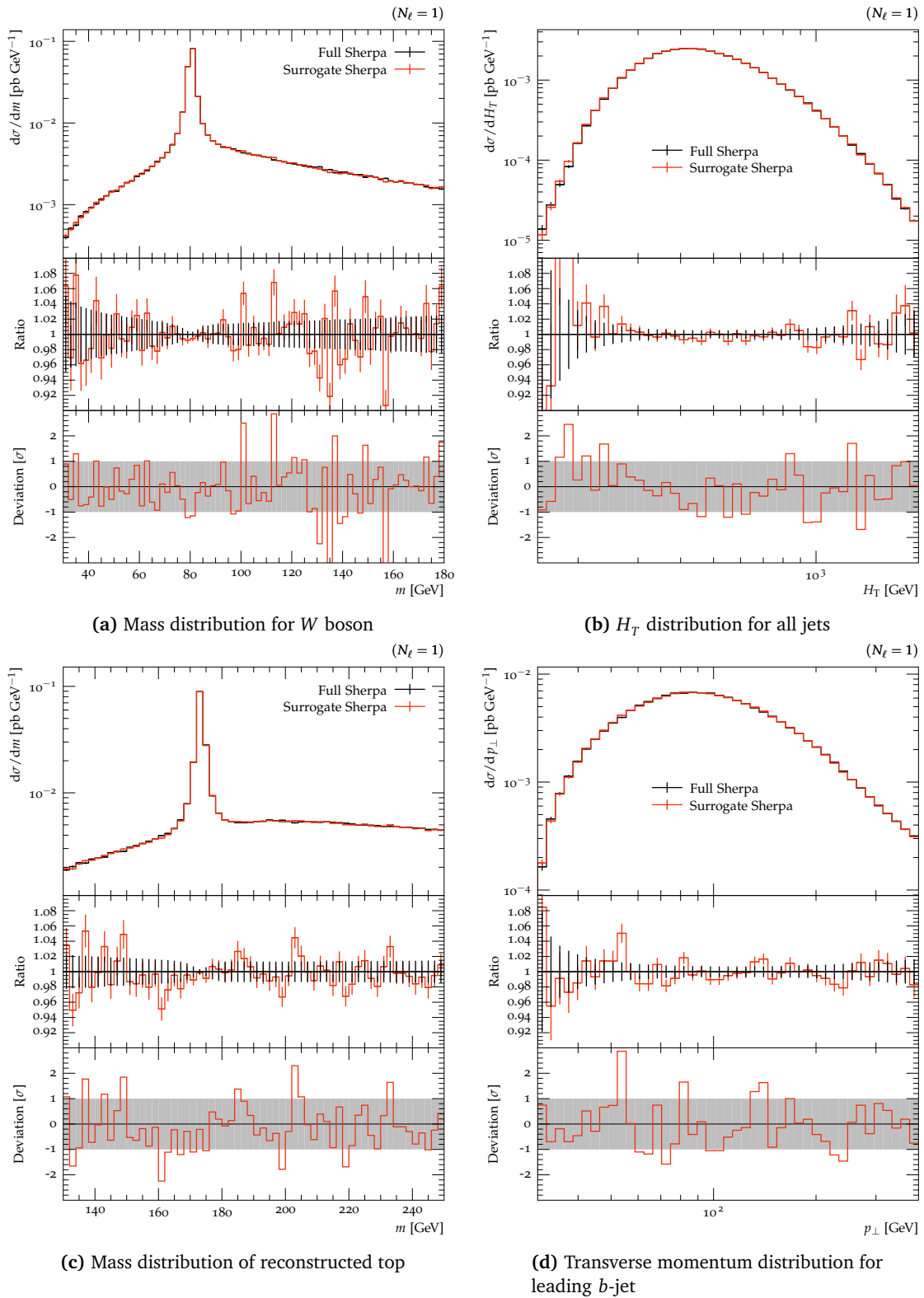


Figure 8: Comparison of different differential distributions generated using SHERPA with (red) and without (black) an NN weight surrogate for the process $uu \rightarrow t\bar{t}guu$ with subsequent leptonic top-quark decays in proton-proton collisions at $\sqrt{s} = 13$ TeV.

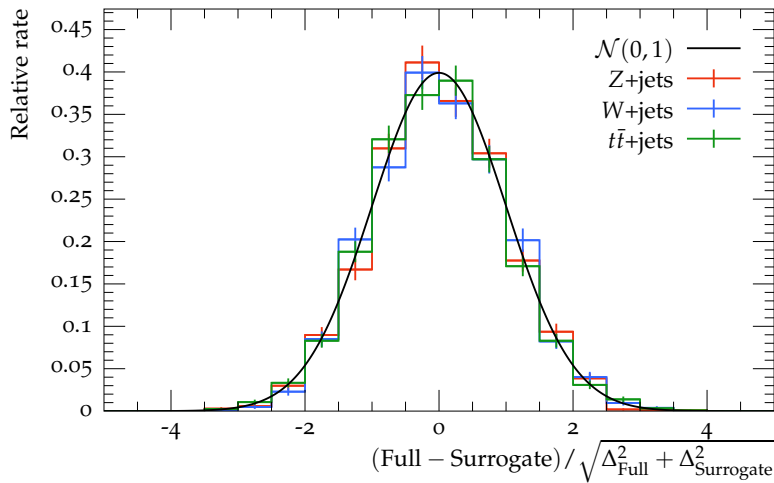


Figure 9: Distribution of deviations between the full and the surrogate approach in SHERPA from a comparison in almost 16,000 observable bins for all three processes.

5 Conclusions

Virtual particle collisions as simulated by Monte Carlo event generators play a central role in high-energy physics. Representing our best-knowledge theoretical expectations, they are used in the design and development of particle detectors for collider experiments, the planning and preparation of measurements, and, foremost, in the actual analysis and interpretation of real experimental data. To match actual measurements, particle-level virtual events need to be supplemented by a detailed simulation of the detector response. Given the calculational complexity and resource consumption of the detector emulation, ideally particle-level events with unit weight should be provided. However, the growing need in high-statistics simulations for a wide range of complex, high-multiplicity partonic scattering processes, including higher-order perturbative corrections, makes event unweighting a severe and very relevant computational challenge.

We have presented a novel two-staged unweighting algorithm that has the potential to significantly accelerate event unweighting. In an initial rejection-sampling step we employ a light-weight neural-network surrogate for the computationally expensive exact integrand, *i.e.* the matrix-element and phase-space weight. The mismatch of the surrogate and the true event weight is then corrected for in a second unweighting step. To protect against rare outliers in the true weight distribution as well as in the point-wise ratio of the true and the surrogate weight, we systematically reduce the respective numerically found maxima using a quantile or median approach, resulting in a partial overweighting of events. The relevant performance measures for the algorithm are the quality of the approximation, as well as the evaluation time per phase-space point, which can be combined into an effective per-event gain factor f_{eff} with respect to conventional rejection sampling. This measure accounts for the reduced statistical power of the sample due to overweighting. It is used throughout this work to give a rigorous assessment of the effective improvement to be expected in various example processes. While the proposed unweighting algorithm has been developed in the context of collision-event simulations, it is in fact more general and can be used in other applications as well.

In Sec. 3 we have discussed the setup and training procedure used to approximate event weights with deep feedforward neural networks. As an initial test bed we have used a representative partonic channel contributing to tree-level $Z+4$ jets production at the LHC. We found

that our neural network is well capable of estimating the true event weights, thereby being more than 600 times faster.

In Sec. 4 we presented the practical implementation of the novel two-staged unweighting algorithm in the SHERPA event-generator framework. To further validate, benchmark and gauge the potential of the method, we applied it to high-multiplicity partonic channels contributing to $W+4$ jets and $t\bar{t}+3$ jets at the LHC. For the dominant partonic channels with sizeable cross sections and expensive matrix elements we found gain factors from using surrogate unweighting ranging from two up to ten. By comparing differential distributions of physical observables we were able to show that the proposed method indeed reproduces the correct target distribution. We were furthermore able to show that the partial overweighting of events, due to employing reduced maxima in the rejection sampling, barely affects the statistical accuracy and leaves no visible effect in physical distributions.

The unweighting algorithm presented here can also be applied in event generation beyond the leading order, where in parts of the phase space the event weights can become negative. While the proposed algorithm can take negative-valued weights into account, our SHERPA implementation is currently limited to tree-level matrix elements, where only positive weights appear. We leave the generalisation to NLO event generation and corresponding performance studies for future work. It will furthermore be interesting to apply our algorithm with alternative and potentially more powerful surrogate methods on the market, and evaluate their performance using the measures introduced in this work.

Acknowledgements

We are grateful for fruitful discussions with Johannes Krause, Stefan Höche and Marek Schönherr, and Stephen Jiggins for reading the manuscript. We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocations of computing time.

Funding information This work has received funding from the European Union's Horizon 2020 research and innovation programme as part of the Marie Skłodowska-Curie Innovative Training Network MCnetITN3 (grant agreement no. 722104). SS and TJ acknowledge support from BMBF (contracts 05H18MGCA1 and 05H21MGCAB). SS acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 456104544. FS's research was supported by the German Research Foundation (DFG) under grant No. SI 2009/1-1.

References

- [1] G. Corcella, I. G. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson, M. H. Seymour and B. R. Webber, *HERWIG 6: An event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)*, J. High Energy Phys. **01**, 010 (2001), doi:[10.1088/1126-6708/2001/01/010](https://doi.org/10.1088/1126-6708/2001/01/010).
- [2] J. Bellm et al., *Herwig 7.0/Herwig++ 3.0 release note*, Eur. Phys. J. C **76**, 196 (2016), doi:[10.1140/epjc/s10052-016-4018-8](https://doi.org/10.1140/epjc/s10052-016-4018-8).
- [3] T. Sjöstrand, S. Mrenna and P. Skands, *PYTHIA 6.4 physics and manual*, J. High Energy Phys. **05**, 026 (2006), doi:[10.1088/1126-6708/2006/05/026](https://doi.org/10.1088/1126-6708/2006/05/026).

- [4] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, Comput. Phys. Commun. **191**, 159 (2015), doi:[10.1016/j.cpc.2015.01.024](https://doi.org/10.1016/j.cpc.2015.01.024).
- [5] 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**, 007 (2009), doi:[10.1088/1126-6708/2009/02/007](https://doi.org/10.1088/1126-6708/2009/02/007).
- [6] E. Bothmann et al., *Event generation with Sherpa 2.2*, SciPost Phys. **7**, 034 (2019), doi:[10.21468/SciPostPhys.7.3.034](https://doi.org/10.21468/SciPostPhys.7.3.034).
- [7] P. Azzi et al., *Report from working group 1: Standard Model physics at the HL-LHC and HE-LHC*, CERN Yellow Rep. Monogr. **7**, 1 (2019), doi:[10.23731/CYRM-2019-007.1](https://doi.org/10.23731/CYRM-2019-007.1).
- [8] P. Calafiura, J. Catmore, D. Costanzo and A. Di Girolamo, *ATLAS HL-LHC computing conceptual design report*, Tech. Rep., CERN, Geneva, <http://cds.cern.ch/record/2729668>, CERN-LHCC-2020-015, LHCC-G-178 (2020).
- [9] M. L. Mangano, F. Piccinini, A. D. Polosa, M. Moretti and R. Pittau, *ALPGEN, a generator for hard multiparton processes in hadronic collisions*, J. High Energy Phys. **07**, 001 (2003), doi:[10.1088/1126-6708/2003/07/001](https://doi.org/10.1088/1126-6708/2003/07/001).
- [10] F. Krauss, R. Kuhn and G. Soff, *AMEGIC++ 1.0, A Matrix Element Generator In C++*, J. High Energy Phys. **02**, 044 (2002), doi:[10.1088/1126-6708/2002/02/044](https://doi.org/10.1088/1126-6708/2002/02/044).
- [11] T. Gleisberg and S. Höche, *Comix, a new matrix element generator*, J. High Energy Phys. **12**, 039 (2008), doi:[10.1088/1126-6708/2008/12/039](https://doi.org/10.1088/1126-6708/2008/12/039).
- [12] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, J. High Energy Phys. **07**, 079 (2014), doi:[10.1007/JHEP07\(2014\)079](https://doi.org/10.1007/JHEP07(2014)079).
- [13] W. Kilian, T. Ohl and J. Reuter, *WHIZARD—simulating multi-particle processes at LHC and ILC*, Eur. Phys. J. C **71**, 1742 (2011), doi:[10.1140/epjc/s10052-011-1742-y](https://doi.org/10.1140/epjc/s10052-011-1742-y).
- [14] V. Hirschi, R. Frederix, S. Frixione, M. Vittoria Garzelli, F. Maltoni and R. Pittau, *Automation of one-loop QCD computations*, J. High Energy Phys. **05**, 044 (2011), doi:[10.1007/JHEP05\(2011\)044](https://doi.org/10.1007/JHEP05(2011)044).
- [15] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H.-S. Shao and M. Zaro, *The automation of next-to-leading order electroweak calculations*, J. High Energy Phys. **07**, 185 (2018), doi:[10.1007/JHEP07\(2018\)185](https://doi.org/10.1007/JHEP07(2018)185).
- [16] J. M. Campbell and R. K. Ellis, *Update on vector boson pair production at hadron colliders*, Phys. Rev. D **60**, 113006 (1999), doi:[10.1103/PhysRevD.60.113006](https://doi.org/10.1103/PhysRevD.60.113006).
- [17] J. M. Campbell, S. Höche and C. T. Preuss, *Accelerating LHC phenomenology with analytic one-loop amplitudes*, Eur. Phys. J. C **81**, 1117 (2021), doi:[10.1140/epjc/s10052-021-09885-0](https://doi.org/10.1140/epjc/s10052-021-09885-0).
- [18] S. Badger, B. Biedermann, P. Uwer and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, Comput. Phys. Commun. **184**, 1981 (2013), doi:[10.1016/j.cpc.2013.03.018](https://doi.org/10.1016/j.cpc.2013.03.018).
- [19] F. Cascioli, P. Maierhöfer and S. Pozzorini, *Scattering amplitudes with open loops*, Phys. Rev. Lett. **108**, 111601 (2012), doi:[10.1103/PhysRevLett.108.111601](https://doi.org/10.1103/PhysRevLett.108.111601).

- [20] F. Buccioni, J.-N. Lang, J. M. Lindert, P. Maierhöfer, S. Pozzorini, H. Zhang and M. F. Zoller, *OpenLoops 2*, Eur. Phys. J. C **79**, 866 (2019), doi:[10.1140/epjc/s10052-019-7306-2](https://doi.org/10.1140/epjc/s10052-019-7306-2).
- [21] S. Alioli, P. Nason, C. Oleari and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: The POWHEG BOX*, J. High Energy Phys. **06**, 043 (2010), doi:[10.1007/JHEP06\(2010\)043](https://doi.org/10.1007/JHEP06(2010)043).
- [22] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf and S. Uccirati, *RECOLA: Recursive Computation of One-Loop Amplitudes*, Comput. Phys. Commun. **214**, 140 (2017), doi:[10.1016/j.cpc.2017.01.004](https://doi.org/10.1016/j.cpc.2017.01.004).
- [23] A. Denner, J.-N. Lang and S. Uccirati, *RECOLA2: Recursive Computation of One-Loop Amplitudes 2*, Comput. Phys. Commun. **224**, 346 (2018), doi:[10.1016/j.cpc.2017.11.013](https://doi.org/10.1016/j.cpc.2017.11.013).
- [24] A. Buckley et al., *General-purpose event generators for LHC physics*, Phys. Rep. **504**, 145 (2011), doi:[10.1016/j.physrep.2011.03.005](https://doi.org/10.1016/j.physrep.2011.03.005).
- [25] 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), doi:[10.1103/PhysRevD.100.014024](https://doi.org/10.1103/PhysRevD.100.014024).
- [26] G. Aad et al., *The ATLAS simulation infrastructure*, Eur. Phys. J. C **70**, 823 (2010), doi:[10.1140/epjc/s10052-010-1429-9](https://doi.org/10.1140/epjc/s10052-010-1429-9).
- [27] S. Abdullin, P. Azzi, F. Beaudette, P. Janot and A. Perrotta, *The fast simulation of the CMS detector at LHC*, J. Phys.: Conf. Ser. **331**, 032049 (2011), doi:[10.1088/1742-6596/331/3/032049](https://doi.org/10.1088/1742-6596/331/3/032049).
- [28] M. Clemencic, G. Corti, S. Easo, C. R. Jones, S. Miglioranza, M. Pappagallo and P. Robbe, *The LHCb simulation application, Gauss: Design, evolution and experience*, J. Phys.: Conf. Ser. **331**, 032023 (2011), doi:[10.1088/1742-6596/331/3/032023](https://doi.org/10.1088/1742-6596/331/3/032023).
- [29] R. Kleiss and R. Pittau, *Weight optimization in multichannel Monte Carlo*, Comput. Phys. Commun. **83**, 141 (1994), doi:[10.1016/0010-4655\(94\)90043-4](https://doi.org/10.1016/0010-4655(94)90043-4).
- [30] E. Byckling and K. Kajantie, *n-particle phase space in terms of invariant momentum transfers*, Nucl. Phys. B **9**, 568 (1969), doi:[10.1016/0550-3213\(69\)90271-5](https://doi.org/10.1016/0550-3213(69)90271-5).
- [31] S. Mrenna and P. Skands, *Automated parton-shower variations in pythia 8*, Phys. Rev. D **94**, 074005 (2016), doi:[10.1103/PhysRevD.94.074005](https://doi.org/10.1103/PhysRevD.94.074005).
- [32] J. Bellm, S. Plätzer, P. Richardson, A. Siódmok and S. Webster, *Reweighting parton showers*, Phys. Rev. D **94**, 034028 (2016), doi:[10.1103/PhysRevD.94.034028](https://doi.org/10.1103/PhysRevD.94.034028).
- [33] E. Bothmann, M. Schönherr and S. Schumann, *Reweighting QCD matrix-element and parton-shower calculations*, Eur. Phys. J. C **76**, 590 (2016), doi:[10.1140/epjc/s10052-016-4430-0](https://doi.org/10.1140/epjc/s10052-016-4430-0).
- [34] O. Mattelaer, *On the maximal use of Monte Carlo samples: Re-weighting events at NLO accuracy*, Eur. Phys. J. C **76**, 674 (2016), doi:[10.1140/epjc/s10052-016-4533-7](https://doi.org/10.1140/epjc/s10052-016-4533-7).
- [35] D. Buarque Franzosi, F. Fabbri and S. Schumann, *Constraining scalar resonances with top-quark pair production at the LHC*, J. High Energy Phys. **03**, 022 (2018), doi:[10.1007/JHEP03\(2018\)022](https://doi.org/10.1007/JHEP03(2018)022).

- [36] K. Kröniger, S. Schumann and B. Willenberg, *(MC)**3 – a multi-channel Markov chain Monte Carlo algorithm for phase-space sampling*, Comput. Phys. Commun. **186**, 1 (2015), doi:[10.1016/j.cpc.2014.08.024](https://doi.org/10.1016/j.cpc.2014.08.024).
- [37] K. Matchev and P. Shyamsundar, *OASIS: Optimal Analysis-Specific Importance Sampling for event generation*, SciPost Phys. **10**, 034 (2021), doi:[10.21468/SciPostPhys.10.2.034](https://doi.org/10.21468/SciPostPhys.10.2.034).
- [38] S. Amoroso et al., *Challenges in Monte Carlo event generator software for high-luminosity LHC*, Comput. Softw. Big Sci. **5**, 12 (2021), doi:[10.1007/s41781-021-00055-1](https://doi.org/10.1007/s41781-021-00055-1).
- [39] J. Bendavid, *Efficient Monte Carlo integration using boosted decision trees and generative deep neural networks*, [arXiv:1707.00028](https://arxiv.org/abs/1707.00028).
- [40] M. Klimek and M. Perelstein, *Neural network-based approach to phase space integration*, SciPost Phys. **9**, 053 (2020), doi:[10.21468/SciPostPhys.9.4.053](https://doi.org/10.21468/SciPostPhys.9.4.053).
- [41] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri and R. Verheyen, *Event generation and statistical sampling for physics with deep generative models and a density information buffer*, Nat. Commun. **12**, 2985 (2021), doi:[10.1038/s41467-021-22616-z](https://doi.org/10.1038/s41467-021-22616-z).
- [42] R. Di Sipio, M. Fucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: A Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC*, J. High Energy Phys. **08**, 110 (2019), doi:[10.1007/JHEP08\(2019\)110](https://doi.org/10.1007/JHEP08(2019)110).
- [43] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC events*, SciPost Phys. **7**, 075 (2019), doi:[10.21468/SciPostPhys.7.6.075](https://doi.org/10.21468/SciPostPhys.7.6.075).
- [44] Y. Alanazi et al., *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, in *Proceedings of the thirtieth international joint conference on artificial intelligence*, 2126 (2021), doi:[10.24963/ijcai.2021/293](https://doi.org/10.24963/ijcai.2021/293).
- [45] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying event samples*, SciPost Phys. **10**, 139 (2021), doi:[10.21468/SciPostPhys.10.6.139](https://doi.org/10.21468/SciPostPhys.10.6.139).
- [46] Y. Alanazi et al., *Machine learning-based event generator for electron-proton scattering*, [arXiv:2008.03151](https://arxiv.org/abs/2008.03151).
- [47] I.-K. Chen, M. Klimek and M. Perelstein, *Improved neural network Monte Carlo simulation*, SciPost Phys. **10**, 023 (2021), doi:[10.21468/SciPostPhys.10.1.023](https://doi.org/10.21468/SciPostPhys.10.1.023).
- [48] K. Matchev, A. Roman and P. Shyamsundar, *Uncertainties associated with GAN-generated datasets in high energy physics*, SciPost Phys. **12**, 104 (2022), doi:[10.21468/SciPostPhys.12.3.104](https://doi.org/10.21468/SciPostPhys.12.3.104).
- [49] D. Jimenez Rezende and S. Mohamed, *Variational inference with normalizing flows*, [arXiv:1505.05770](https://arxiv.org/abs/1505.05770).
- [50] I. Kobyzev, S. J.D. Prince and M. A. Brubaker, *Normalizing flows: An introduction and review of current methods*, IEEE Trans. Pattern Anal. Mach. Intell. **43**, 3964 (2021), doi:[10.1109/TPAMI.2020.2992934](https://doi.org/10.1109/TPAMI.2020.2992934).
- [51] G. Papamakarios, E. Nalisnick, D. Jimenez Rezende, S. Mohamed and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, [arXiv:1912.02762](https://arxiv.org/abs/1912.02762).

- [52] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, [arXiv:1410.8516](https://arxiv.org/abs/1410.8516).
- [53] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP*, [arXiv:1605.08803](https://arxiv.org/abs/1605.08803).
- [54] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling, *Improving variational inference with inverse autoregressive flow*, [arXiv:1606.04934](https://arxiv.org/abs/1606.04934).
- [55] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1×1 convolutions*, [arXiv:1807.03039](https://arxiv.org/abs/1807.03039).
- [56] T. Müller, B. McWilliams, F. Rousselle, M. Gross and J. Novák, *Neural importance sampling*, *ACM Trans. Graph.* **38**, 1 (2019), doi:[10.1145/3341156](https://doi.org/10.1145/3341156).
- [57] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional integration and sampling with normalizing flows*, *Mach. Learn.: Sci. Technol.* **1**, 045023 (2020), doi:[10.1088/2632-2153/abab62](https://doi.org/10.1088/2632-2153/abab62).
- [58] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with neural importance sampling*, *SciPost Phys.* **8**, 069 (2020), doi:[10.21468/SciPostPhys.8.4.069](https://doi.org/10.21468/SciPostPhys.8.4.069).
- [59] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event generation with normalizing flows*, *Phys. Rev. D* **101**, 076002 (2020), doi:[10.1103/PhysRevD.101.076002](https://doi.org/10.1103/PhysRevD.101.076002).
- [60] B. Stienen and R. Verheyen, *Phase space sampling and inference from weighted events with autoregressive flows*, *SciPost Phys.* **10**, 038 (2021), doi:[10.21468/SciPostPhys.10.2.038](https://doi.org/10.21468/SciPostPhys.10.2.038).
- [61] S. Pina-Otey, F. Sánchez, T. Lux and V. Gaitan, *Exhaustive neural importance sampling applied to Monte Carlo event generation*, *Phys. Rev. D* **102**, 013003 (2020), doi:[10.1103/PhysRevD.102.013003](https://doi.org/10.1103/PhysRevD.102.013003).
- [62] M. Backes, A. Butter, T. Plehn and R. Winterhalder, *How to GAN event unweighting*, *SciPost Phys.* **10**, 089 (2021), doi:[10.21468/SciPostPhys.10.4.089](https://doi.org/10.21468/SciPostPhys.10.4.089).
- [63] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih, *DCTRGAN: Improving the precision of generative models with reweighting*, *J. Inst.* **15**, P11004 (2020), doi:[10.1088/1748-0221/15/11/P11004](https://doi.org/10.1088/1748-0221/15/11/P11004).
- [64] R. Winterhalder, M. Bellagente and B. Nachman, *Latent space refinement for deep generative models*, [arXiv:2106.00792](https://arxiv.org/abs/2106.00792).
- [65] M. Bellagente, M. Haußmann, M. Luchmann and T. Plehn, *Understanding event-generation networks via uncertainties*, [arXiv:2104.04543](https://arxiv.org/abs/2104.04543).
- [66] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, *Generative networks for precision enthusiasts*, [arXiv:2110.13632](https://arxiv.org/abs/2110.13632).
- [67] F. James, *Monte Carlo theory and practice*, *Rep. Prog. Phys.* **43**, 1145 (1980), doi:[10.1088/0034-4885/43/9/002](https://doi.org/10.1088/0034-4885/43/9/002).
- [68] G. Peter Lepage, *A new algorithm for adaptive multidimensional integration*, *J. Comput. Phys.* **27**, 192 (1978), doi:[10.1016/0021-9991\(78\)90004-9](https://doi.org/10.1016/0021-9991(78)90004-9).
- [69] T. Ohl, *Vegas revisited: Adaptive Monte Carlo integration beyond factorization*, *Comput. Phys. Commun.* **120**, 13 (1999), doi:[10.1016/S0010-4655\(99\)00209-X](https://doi.org/10.1016/S0010-4655(99)00209-X).

- [70] F. Maltoni and T. Stelzer, *MadEvent: Automatic event generation with MadGraph*, J. High Energy Phys. **02**, 027 (2003), doi:[10.1088/1126-6708/2003/02/027](https://doi.org/10.1088/1126-6708/2003/02/027).
- [71] J. von Neumann, *Various techniques used in connection with random digits*, in A. S. Householder, G. E. Forsythe and H. H. Germond, eds., *Monte Carlo method*, National Bureau of Standards Applied Mathematics Series **12** 36, US Government Printing Office, Washington, US, (1951).
- [72] S. Jadach, *Foam: Multi-dimensional general purpose Monte Carlo generator with self-adapting simplicial grid*, Comput. Phys. Commun. **130**, 244 (2000), doi:[10.1016/S0010-4655\(00\)00047-3](https://doi.org/10.1016/S0010-4655(00)00047-3).
- [73] S. Jadach, *Foam: A general-purpose cellular Monte Carlo event generator*, Comput. Phys. Commun. **152**, 55 (2003), doi:[10.1016/S0010-4655\(02\)00755-5](https://doi.org/10.1016/S0010-4655(02)00755-5).
- [74] L. Kish, *Survey sampling*, John Wiley & Sons, New York, US, ISBN 9780471109495 (1965).
- [75] S. Catani and M. H. Seymour, *A general algorithm for calculating jet cross sections in NLO QCD*, Nucl. Phys. B **485**, 291 (1997), doi:[10.1016/S0550-3213\(96\)00589-5](https://doi.org/10.1016/S0550-3213(96)00589-5).
- [76] S. Frixione, Z. Kunszt and A. Signer, *Three-jet cross sections to next-to-leading order*, Nucl. Phys. B **467**, 399 (1996), doi:[10.1016/0550-3213\(96\)00110-1](https://doi.org/10.1016/0550-3213(96)00110-1).
- [77] S. Frixione and B. R. Webber, *Matching NLO QCD computations and parton shower simulations*, J. High Energy Phys. **06**, 029 (2002), doi:[10.1088/1126-6708/2002/06/029](https://doi.org/10.1088/1126-6708/2002/06/029).
- [78] S. Höche, F. Krauss, M. Schönherr and F. Siegert, *QCD matrix elements + parton showers. The NLO case*, J. High Energy Phys. **04**, 027 (2013), doi:[10.1007/JHEP04\(2013\)027](https://doi.org/10.1007/JHEP04(2013)027).
- [79] T. Carli, D. Clements, A. Cooper-Sarkar, C. Gwenlan, G. P. Salam, F. Siegert, P. Starovoitov and M. Sutton, *A posteriori inclusion of parton density functions in NLO QCD final-state calculations at hadron colliders: The APPLGRID project*, Eur. Phys. J. C **66**, 503 (2010), doi:[10.1140/epjc/s10052-010-1255-0](https://doi.org/10.1140/epjc/s10052-010-1255-0).
- [80] T. Kluge, K. Rabbertz and M. Wobisch, *FastNLO: Fast pQCD calculations for PDF fits*, in *14th international workshop on deep inelastic scattering*, (2007), doi:[10.1142/9789812706706_0110](https://doi.org/10.1142/9789812706706_0110).
- [81] S. Carrazza, E. R. Nocera, C. Schwan and M. Zaro, *PineAPPL: Combining EW and QCD corrections for fast evaluation of LHC processes*, J. High Energy Phys. **12**, 108 (2020), doi:[10.1007/JHEP12\(2020\)108](https://doi.org/10.1007/JHEP12(2020)108).
- [82] L. Del Debbio, N. Hartland and S. Schumann, *MCgrid: Projecting cross section calculations on grids*, Comput. Phys. Commun. **185**, 2115 (2014), doi:[10.1016/j.cpc.2014.03.023](https://doi.org/10.1016/j.cpc.2014.03.023).
- [83] V. Bertone, R. Frederix, S. Frixione, J. Rojo and M. Sutton, *aMCfast: Automation of fast NLO computations for PDF fits*, J. High Energy Phys. **08**, 166 (2014), doi:[10.1007/JHEP08\(2014\)166](https://doi.org/10.1007/JHEP08(2014)166).
- [84] S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz de Austri and J. Tattersall, *DeepXS: Fast approximation of MSSM electroweak cross sections at NLO*, Eur. Phys. J. C **80**, 12 (2020), doi:[10.1140/epjc/s10052-019-7562-1](https://doi.org/10.1140/epjc/s10052-019-7562-1).

- [85] A. Buckley, A. Kvellestad, A. Raklev, P. Scott, J. Vegard Sparre, J. Van den Abeele and I. A. Vazquez-Holm, *Xsec: The cross-section evaluation code*, Eur. Phys. J. C **80**, 1106 (2020), doi:[10.1140/epjc/s10052-020-08635-y](https://doi.org/10.1140/epjc/s10052-020-08635-y).
- [86] F. Bury and C. Delaere, *Matrix element regression with deep neural networks — Breaking the CPU barrier*, J. High Energy Phys. **04**, 020 (2021), doi:[10.1007/JHEP04\(2021\)020](https://doi.org/10.1007/JHEP04(2021)020).
- [87] F. Bishara and M. Montull, *(Machine) Learning amplitudes for faster event generation*, [arXiv:1912.11055](https://arxiv.org/abs/1912.11055).
- [88] S. Badger and J. Bullock, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes*, J. High Energy Phys. **06**, 114 (2020), doi:[10.1007/JHEP06\(2020\)114](https://doi.org/10.1007/JHEP06(2020)114).
- [89] J. Aylett-Bullock, S. Badger and R. Moodie, *Optimising simulations for diphoton production at hadron colliders using amplitude neural networks*, J. High Energy Phys. **08**, 066 (2021), doi:[10.1007/JHEP08\(2021\)066](https://doi.org/10.1007/JHEP08(2021)066).
- [90] D. Maître and H. Truong, *A factorisation-aware Matrix element emulator*, J. High Energy Phys. **11**, 066 (2021), doi:[10.1007/JHEP11\(2021\)066](https://doi.org/10.1007/JHEP11(2021)066).
- [91] J. Krause, *Efficiency improvements using machine learning in event generators for the LHC*, M.Sc. thesis, CERN-THESIS-2015-486, (2015).
- [92] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas and H. Sebastian Seung, *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit*, Nature **405**, 947 (2000), doi:[10.1038/35016072](https://doi.org/10.1038/35016072).
- [93] K. He, X. Zhang, S. Ren and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, in *Proceedings of the IEEE international conference on computer vision*, (2015).
- [94] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [95] M. Cacciari, G. P. Salam and G. Soyez, *The anti- k_t jet clustering algorithm*, J. High Energy Phys. **04**, 063 (2008), doi:[10.1088/1126-6708/2008/04/063](https://doi.org/10.1088/1126-6708/2008/04/063).
- [96] R. D. Ball et al., *Parton distributions for the LHC run II*, J. High Energy Phys. **04**, 040 (2015), doi:[10.1007/JHEP04\(2015\)040](https://doi.org/10.1007/JHEP04(2015)040).
- [97] F. Chollet et al., *Keras*, (2015), <https://keras.io>.
- [98] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, (2015), <https://tensorflow.org>.
- [99] The HDF Group, *Hierarchical Data Format, version 5*, (2021), <https://www.hdfgroup.org/HDF5/>.
- [100] T. Hermann, *frugally-deep*, Github, (2016), <https://github.com/Dobiasd/frugally-deep>.
- [101] C. Bierlich et al., *Robust independent validation of experiment and theory: Rivet version 3*, SciPost Phys. **8**, 026 (2020), doi:[10.21468/SciPostPhys.8.2.026](https://doi.org/10.21468/SciPostPhys.8.2.026).

4.3 To be published: Unweighted event generation for multi-jet production processes based on matrix element emulation

In this section the article ‘Unweighted event generation for multi-jet production processes based on matrix element emulation’ is presented. It is a follow-up to ref. [11], presented above, where the simple surrogate model used therein is replaced by the more sophisticated, factorization-aware model of ref. [64]. While that model was originally developed for jet production in e^+e^- collisions, it was extended to allow hadronic initial states and massive quarks in the final state for this study. By incorporating knowledge about the factorization properties of QCD matrix elements in the model, it provides much better approximations to these matrix elements compared to a naive model.

In the article, the results of applying the unweighting method in combination with the factorization-aware surrogate to partonic channels contributing to $Z + 4, 5$ jets and $t\bar{t} + 3, 4$ jets production at the LHC are presented. Thereby, we went one multiplicity higher than in ref. [11] and dropped the production of W bosons to reduce the number of examples. A comparison to the naive model shows that the gain factors increase significantly, reaching values between 16 and 354. It is shown how the results are influenced by the number of events in the training data. Furthermore, a discussion about summing versus sampling over the $SU(3)$ colour configurations is given. While the main results of the article relate to colour summed matrix elements, the scaling behaviour of colour sampling makes it a popular choice for high-multiplicity processes. In the article, it is shown how to extend the model to make it applicable in a colour sampling scenario. The achieved gains are not satisfying, though, and ideas for improving the performance are presented.

The article was first published as a preprint on ARXIV in January 2023. Subsequently, it was submitted to the journal *SciPost Physics*, where it is currently undergoing peer review. It received positive comments from the two referees, with only minor changes demanded. The editor asked for a minor revision to address these points, which we provided in May 2023. The revised preprint version is reprinted below. The copyright belongs to the authors and the article is licensed under a [Creative Commons Attribution 4.0 International License](#).

Author contributions

Since the article is based on the combination of two previous publications by the authors, refs. [11, 64], the implementations used therein provided the foundation for the work on this project. Accordingly, the surrogate model was implemented by Henry Truong, while I contributed the implementation of the unweighting algorithm and the naive, non-factorization-aware model, based on the earlier work of Katharina Danziger and myself. Henry Truong extended his model to include initial state partons and massive quarks in the final state. He trained his model on training data that I provided. Subsequently, I used the trained versions within SHERPA to evaluate the performance by generating unweighted events with our two stage unweighting method. To make the model available in SHERPA, I ported the PYTHON implementation to C++ code. I also contributed the interface to the ONNX RUNTIME. The presentation of the surrogate model in sec. 2 was almost solely prepared by Henry Truong, while the description of the unweighting method in sec. 3 was mostly done by me. For the rest of the article, all authors contributed significantly to the writing.

MCNET-23-01, IPPP/23/04

Unweighting multijet event generation using factorisation-aware neural networks

T. Janßen¹, D. Maître², S. Schumann¹, F. Siegert³, H. Truong²

1 Institut für Theoretische Physik, Georg-August-Universität Göttingen, Göttingen, Germany

2 Institute for Particle Physics Phenomenology, Department of Physics, Durham University, United Kingdom

3 Institut für Kern- und Teilchenphysik, TU Dresden, Dresden, Germany

May 17, 2023

Abstract

In this article we combine a recently proposed method for factorisation-aware matrix element surrogates with an unbiased unweighting algorithm. We show that employing a sophisticated neural network emulation of QCD multijet matrix elements based on dipole factorisation can lead to a drastic acceleration of unweighted event generation. We train neural networks for a selection of partonic channels contributing at the tree-level to $Z+4, 5$ jets and $t\bar{t}+3, 4$ jets production at the LHC which necessitates a generalisation of the dipole emulation model to include initial state partons as well as massive final state quarks. We also present first steps towards the emulation of colour-sampled amplitudes. We incorporate these emulations as fast and accurate surrogates in a two-stage rejection sampling algorithm within the SHERPA Monte Carlo that yields unbiased unweighted events suitable for phenomenological analyses and post-processing in experimental workflows, e.g. as input to a time-consuming detector simulation. For the computational cost of unweighted events we achieve a reduction by factors between 16 and 350 for the considered channels.

arXiv:2301.13562v2 [hep-ph] 16 May 2023

Contents

1	Introduction	2
2	Improved matrix element emulation using neural networks	4
2.1	Neural networks based on dipoles	4
2.2	Extension to initial-state and massive partons	7
2.3	Colour-sampled matrix elements	9
3	Event unweighting utilising matrix element surrogates	12
3.1	Two-stage unweighting method	12
3.2	Performance analysis	15
4	Implementation and application to LHC processes	16
4.1	Implementation in the SHERPA framework	16
4.2	Results for LHC multijet production processes	17
5	Conclusions	23
A	Auxiliary weight distributions	24
	References	26

1 Introduction

Physics simulations for current and future high-energy accelerator experiments pose a severe computational challenge not only due to the complexity of the studied signatures and the demand for higher theoretical accuracy, but also owing to the sheer number of simulated events needed to match the enormous collider luminosities. This has sparked a wide range of algorithmic developments to accelerate key elements of the simulation tool chain and to improve their computational efficiency and thus reduce their resource requirements. Machine learning based methods play a prominent role in these developments [3].

Traditionally, the largest fraction of resources has been spent on the complex simulation of the detector response to collision final states, while the generation of the collision events constituted only $\mathcal{O}(10\% - 20\%)$ of the budget. With many recent activities reducing the computational footprint of detector simulations, the event generation speed has become a more and more important area to facilitate the full exploitation of future collider data.

This situation is amplified with the upcoming increased luminosity at the LHC and its focus turning to more complex processes. With the advent of matching and merging techniques theoretical predictions of increased precision have become accessible for these high multiplicity final states. But as the computational cost increases strongly with the multiplicity the resources needed for the theoretical description of processes of interest has surged, see for example [4]. Besides conventional approaches for improving the performance of Monte Carlo methods in event generators, more recently also machine learning methods are explored [5].

A particular challenge is related to the generation of the hard scattering component that forms the core of the event evolution, thereby representing the parton-level truth signal and/or background hypotheses in physics analyses [6, 7]. In Monte Carlo event generators the hard process is assumed to be factorised from parton showers in the initial and final state as well as non-perturbative phenomena such as hadronisation and the underlying event. Algorithmically the generation of hard scattering events corresponds to the evaluation, *i.e.* the stochastic sampling, of the phase space integral over the squared transition matrix element of the considered process at a given order in perturbation theory.

There has been renewed interest in improving phase space integration and sampling techniques, mostly based on neural networks [8–14], using a variety of methods, but also Nested Sampling [15], and mixed-kernel Markov chain algorithms [16] have been investigated. Broadly speaking these approaches share the goal of adjusting a sampling distribution as closely as possible to the true target distribution, *i.e.* the actual transition matrix element. In the case of a traditional Monte Carlo integration this will typically result in events with weights of ideally small spread. However, for a resource efficient generation process for physics analyses, in particular due to very time-consuming components such as a detector simulation, events with (largely) unit weight are desirable. This is typically solved via von Neumann rejection sampling, *i.e.* an accept–reject procedure for weighted event samples. However, even with advanced adaptive sampling techniques in particular for multi-particle final states the efficiency of the unweighting procedure can be quite small, resulting in the repeated trial evaluation of the scattering matrix element that ultimately get rejected. For LHC key processes such as the multijet-associated production of gauge bosons this might result in $\mathcal{O}(10^5)$ evaluations of the computationally expensive matrix element for a single unit-weight event [17].

This suggests a complementary opportunity for saving resources, namely the usage of a fast *and* accurate surrogate for the trial weights. A corresponding two-stage unweighting procedure that fully corrects for the potential mismatch between the surrogate weight and the actual value of the full matrix element has recently been presented in Ref. [2]. To demonstrate the algorithm, a rather simple neural network designed to replicate the weight of partonic events, represented by their external momenta was used. For tree-level contributions to $Z/W + 4$ jets and $t\bar{t} + 3$ jets production at the LHC significant gains have been observed, however, for less complex partonic channels ordinary unweighting could not be outperformed. Over the last few years more sophisticated matrix element surrogates based on neural networks have been developed [1, 18, 19], addressing tree-level and one-loop amplitudes. In particular, Ref. [1] presented a method to emulate scattering matrix elements employing the factorisation properties of QCD amplitudes in the soft- and collinear limits.

In this work we explore the potential of a combination of the approaches in Refs. [2] and [1] in unweighted event generation for multijet production processes at the LHC. To this end we generalise the method presented in Ref. [1] to the case of colour-charged initial states and massive final-state partons. We also explore, for the first time, the emulation of colour-sampled QCD amplitudes in the colour-flow decomposition. With an implementation in the SHERPA event generator framework [20, 21] we benchmark tree-level contributions to $Z + 4, 5$ jets and $t\bar{t} + 3, 4$ jets production. The paper is structured as follows: In Sec. 2 we review the dipole emulation model of Ref. [1] and present our new developments to address hadronic collisions and massive final-state partons. In Sec. 3 we review the unweighting procedure worked out in Ref. [2]. In Sec. 4 we discuss the implementation of both algorithms in the SHERPA framework, and present our results obtained for selected partonic channels contributing to $pp \rightarrow Z + 4, 5$ jets and $pp \rightarrow t\bar{t} + 3, 4$ jets.

2 Improved matrix element emulation using neural networks

To demonstrate the accelerated surrogate unweighting algorithm in Ref. [2] the authors used only a simple neural network as model for the event weights. It was clear from the study that a bottleneck in this procedure was the accuracy of the event weight approximations coming from the surrogate model, leading to low efficiencies in the second unweighting step. In this work we consider replacing that simple model with the factorisation-aware neural network model introduced in Ref. [1], which has been shown to exhibit more accurate predictions on a per-point basis compared to existing methods. Below we review the construction of this model and detail the necessary extensions to facilitate multijet production processes at the LHC.

2.1 Neural networks based on dipoles

In this section we briefly review the framework from Ref. [1], where an ansatz for matrix elements based on the factorisation properties of QCD matrix elements in their soft and collinear limits was used. This factorisation can be depicted as

$$|\mathcal{M}_{n+1}|^2 \rightarrow |\mathcal{M}_n|^2 \otimes \mathbf{V}_{ijk}, \quad (1)$$

where the matrix element in the $(n + 1)$ -body phase space reduces to a matrix element in the momentum mapped n -body phase space multiplied by a singular factor, \mathbf{V}_{ijk} . For single infrared limits, \mathbf{V}_{ijk} contains all the singularity structure of the matrix element. In the dipole formalism (originally presented by Catani and Seymour for massless partons in [22] and later generalised by Catani, Dittmaier, Seymour and Trócsányi to account for finite masses in [23]) these singular factors are encapsulated in the dipole functions D_{ijk} . This factorisation property of matrix elements leads to the form of our ansatz, which is inspired by the dipole factorisation formula. It is given by

$$|\mathcal{M}_{n+1}|^2 \simeq \sum_{\{ijk\}} C_{ijk} D_{ijk}, \quad (2)$$

where i , j , and k denote the three partons involved in the dipole function. Instead of fitting the matrix element directly where there are divergences in the infrared regions of phase space, we let the neural network fit the coefficients C_{ijk} as a function of phase space. These coefficients are more well-behaved than the full matrix element in the soft and collinear limits as the singular behaviour is described by the dipole functions. By combining these well-behaved coefficients with the analytically known dipoles, we produce an approximation for the matrix element. This enables the fitting of matrix elements across the entire sampled phase space with a single neural network. Whilst the model predicts the C_{ijk} coefficients, it should be noted that these are not entirely meaningful by themselves. Only once they are combined with the corresponding dipoles do we get the approximation of the matrix element. It is this approximation of the matrix element which should be seen as the model prediction and which appears in the loss function.

In singly unresolved limits, only relevant dipoles are large and so constrain the corresponding C_{ijk} in the fit. Outside of these limits all dipoles are of similar order of magnitude and the ansatz in Eq. (2) is more under-constrained. In these regions of phase space, the excellent fitting capabilities of neural networks are leveraged to interpolate the non-singular matrix elements. The accuracy achieved in this approach is due to the fact that the coefficients being fit by the network, for single soft or collinear kinematics, are free of divergences. This facet of the emulation model makes it particularly apt for the case of multijet production processes where matrix elements are plagued with many

Table 1: Hyperparameters of the neural network and their values.

Parameter	Value
Hidden layers	4
Nodes in hidden layers	128
Activation function	swish [26,27]
Weight initialiser	Glorot uniform [28]
Loss function	MSE
Batch size	512
Optimiser	ADAM [29]
Initial learning rate	10^{-3}
Callbacks	EarlyStopping, ReduceLROnPlateau

well-understood divergent structures. In Sec. 4 we will apply the method to emulate jet-production processes at the tree level, where the fiducial phase space is constrained by jet cuts, *i.e.* a minimal separation of all QCD parton pairs and a moderate transverse momentum threshold.

We note that Eq. (1) is not to be interpreted as a recursion relation, instead it depicts the isolation of a single infrared limit into the dipole D_{ijk} leaving the coefficients C_{ijk} to capture the behaviour of the non-divergent n -body matrix element. In principle, this model could be extended to cases of multiple unresolved partons where the divergences are captured by multiple D_{ijk} functions, again leaving C_{ijk} well-behaved. This was examined briefly in Section 3.3 of Ref. [1] where the model performance was tested on more complex infrared configurations.

In this work we employ networks of similar size and complexity to those in Ref. [2], namely, we use KERAS [24] and TENSORFLOW [25] to build a neural network (NN) model with four hidden layers, each consisting of 128 nodes. These hidden layers use the swish activation function [26,27] and their weights are initialised according to the Glorot Uniform distribution [28], which aims to keep the variance of activations similar across all hidden layers in order to prevent exploding or vanishing gradients during network training.

The swish activation function, $\frac{x}{1+\exp(-x)}$, has a similar shape to the more well-known ReLU activation function, but it is a smooth continuous function allowing small negative values, instead of thresholding them to 0 like ReLU does. We find that in practice swish outperforms ReLU in all our trained models. We also find that instead of using a linear activation function in the output layer, using a swish activation function is more performant.

The NN is fitted to the data generated from SHERPA (see Section 4.2 for more information on the generation of data) by minimising the loss function encoding the discrepancy between the prediction made with the neural network to the true matrix element provided by SHERPA. We use the mean squared error (MSE) as the loss function, with the training optimised using ADAM [29] with an initial learning rate of 10^{-3} . The learning rate is reduced when the validation loss shows no improvement for 30 epochs of training by using the `ReduceLROnPlateau` callback, and `EarlyStopping` is used to terminate training when there is no improvement in the validation loss after 60 epochs. A summary of the neural network hyperparameters is given in Tab. 1 for reference.

As inputs to our generalised network model we feed: the 4-momenta of all initial- and

final-state particles, the phase-space mapping variables corresponding to the dipoles in the ansatz, denoted as y_{ijk} ¹, and the kinematic invariants s_{ij} for all pairs of particles in the process considered. Dipoles can be classified depending on whether the emitter and spectator are in the initial- or final-state, and whether they are massless or massive. The four classes of dipoles are FF (final-state emitter, final-state spectator), FI (final-state emitter, initial-state spectator), IF (initial-state emitter, final-state spectator), and II (initial-state emitter, initial-state spectator), where each class can be massless or massive. Each of these dipole configurations have a corresponding phase-space mapping. To illustrate the general form of these mappings, we quote the massless FF case here,

$$y_{ijk} = \frac{p_i p_j}{p_i p_j + p_j p_k + p_i p_k}, \quad (3)$$

where it is understood that the other mappings are also functions of the external momenta (and masses if present). Note that the y_{ijk} and the s_{ij} input variables are *not* independent from the external momenta. To aid the neural network training process, we pre-process the y_{ijk} , such that the shape and widths of their distributions are similar for the different dipole configurations. This amounts to

$$y_{ijk} \rightarrow \begin{cases} \log(1 - y_{ijk}) & \text{if massless FI, IF, or II dipole,} \\ \log(y_{ijk}) & \text{otherwise.} \end{cases} \quad (4)$$

The kinematic invariants are also transformed with the logarithm $s_{ij} \rightarrow \log(s_{ij})$ as they can span many orders of magnitude. It should be noted that the particles involved in the dipole functions and mapping variables denoted by the subscript ijk are the colour-charged particles in the initial- and final-state. Non colour-charged particles, for example, electrons and positrons, do not appear in the dipoles, but nevertheless their momenta and kinematic invariants are fed into the network as inputs such that it learns of their dependence. All of these inputs are standardised to zero mean and unit variance, with the 4-momenta being standardised along each component.

To use them more effectively in the loss function, we pre-process the matrix elements as

$$|\mathcal{M}_{n+1}|^2 \rightarrow \operatorname{arsinh} \left(\frac{|\mathcal{M}_{n+1}|^2}{S_{\text{pred}}} \right), \quad (5)$$

and standardise to zero mean and unit variance. S_{pred} is the prediction scale taken to be the minimum matrix element value found in our training set. This transformation aids the neural network in training by reducing the span of the target distribution.

The output nodes of our neural network correspond not to the matrix elements directly, but instead to the dipole coefficients. The raw outputs, denoted by c_{ijk} , are transformed to the coefficients appearing in Eq. (2), C_{ijk} , via the transformation

$$C_{ijk} = S_{\text{coef}} \times \sinh(c_{ijk}) \quad (6)$$

where S_{coef} is the coefficient scale, taken to be $S_{\text{pred}}/S_{\text{dipole}}$. S_{dipole} is the representative value of a dipole, which we take to be the median of all dipoles in our training set. The neural network prediction is made by using Eq. (2) to combine the predicted C_{ijk} coefficients with the corresponding dipoles D_{ijk} . In order to compare with the scaled target matrix elements, we have to transform the neural network predicted matrix element with Eq. (5) with the same S_{pred} . We can then compare the matrix element as predicted by the neural network, with the truth value, as given by SHERPA, in the MSE loss function. A diagram illustrating the NN emulator architecture is given in Fig. 1.

¹The initial-state phase-space mapping variables are referred to as x_{ijk} in [22, 23] but we will refer to all phase-space mappings as y_{ijk} for brevity.

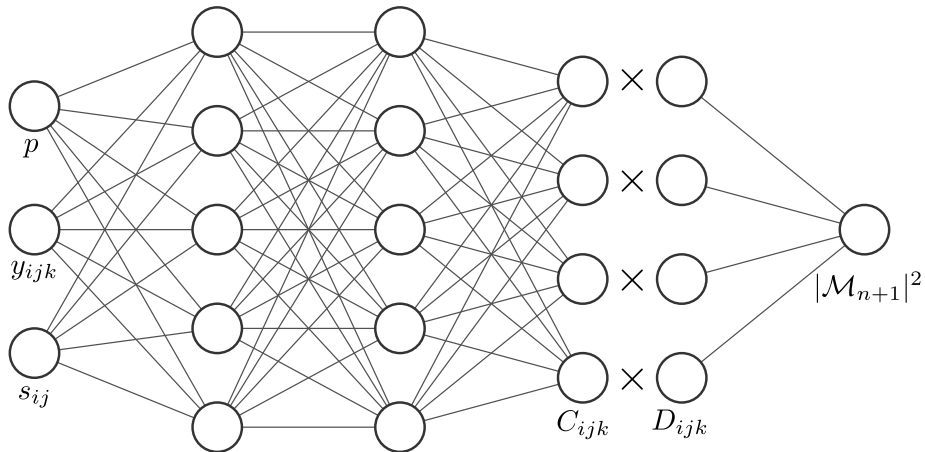


Figure 1: A simplified sketch of our neural network emulator showing inputs, hidden layers, and outputs C_{ijk} .

In Ref. [1], the neural network predictions were given by the average over an ensemble of 20 independent replicas trained on different shuffled subsets of the training set and with different initial random seeds for model weight initialisation. Here we take a similar approach by training a set of 10 replica models, however, for predictions we select the model with the lowest validation loss. We stress that this is not a special choice as all individual replica models converge to a similar point. As an illustrative example, we plot in Fig. 2 the loss curves for the partonic channel $gg \rightarrow e^-e^+ggd\bar{d}$, which is a leading-order contribution to $Z + 4$ jets production at the LHC. We observe convergence across all replicas with training terminating at similar values of the MSE.

The reasoning behind ensembling a prediction is to reduce the effects of stochasticity of the training process, to reduce random model weight initialisation, and to reduce variance in the prediction. In this work we strive for a balance of accuracy and speed, meaning it is advantageous to use a single model to make predictions. The reasoning is as follows. In an ensemble of models where replicas are trained on different subsets of the same training data, there is overlapping information learnt by the individual models. This leads to diminishing returns in predictive accuracy, meaning that whilst evaluation time grows linearly with the number of replicas, accuracy does not. We have therefore observed a single model to be the most performant configuration. It is important to stress that this does not mean that one model cannot be sufficiently accurate, as we will demonstrate in Sec. 4.2.

This decision to use only a single NN for predictions also guided our choice of number of nodes in the hidden layers. With 128 nodes we reach a balance of having enough parameters to model the matrix elements whilst reducing the effects of overfitting. Decreasing the number of nodes in the hidden layers to create a more compact NN has little effect on the evaluation time for a single network when we use the ONNX Runtime [30] for evaluation, there would only be loss in accuracy which represents a decrease in overall unweighting efficiency.

2.2 Extension to initial-state and massive partons

In Ref. [1], the authors considered jet production processes initiated via electron–positron annihilation where only final-state QCD radiation occurs, meaning the set of dipoles built into the emulation model were of the FF kind. Furthermore, the model was restricted to

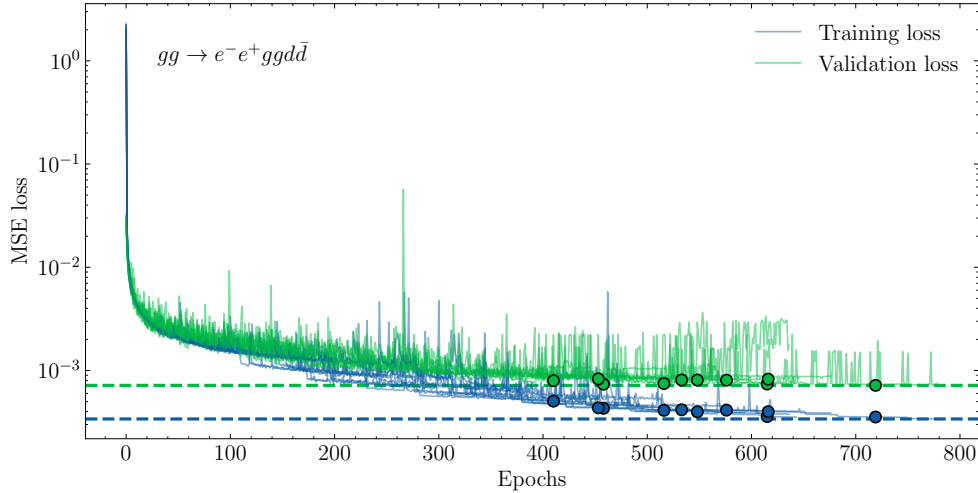


Figure 2: Training and validation loss recorded during training for 10 replica models, for the $gg \rightarrow e^-e^+ggd\bar{d}$ channel, shown as solid lines. The MSE loss is the mean squared difference between the transformed predictions and transformed truth values. The epochs at which training is terminated are illustrated as the solid circles. We depict the training and validation loss of the selected model in dashed horizontal lines.

the production of massless QCD partons.

In this work we consider the extension to hadronic initial states, which is relatively straightforward: we need to account for the additional radiation that comes from the colour-charged initial-state particles. To this end, we add the initial-state dipoles to the ansatz, namely, we add the IF, FI and II splitting configurations. This means that the emitter i , and spectator k , in the ansatz can now be in the initial-state. Illustrations for the complete set of dipoles now included in the model are shown in Fig. 3.

To showcase the extension to massless initial state dipoles we consider the emulation of tree-level matrix elements for the partonic channels $gg \rightarrow e^-e^+ggd\bar{d}$ and $gg \rightarrow e^-e^+ggg\bar{d}\bar{d}$, which are leading order contributions to $Z + 4$ jets and $Z + 5$ jets production at the LHC, respectively. As validation of the emulation accuracy of the NN model for this extension to initial states, we examine the ability of the model to predict matrix elements across the sampled phase space, but in particular for the case of soft and collinear kinematics, where QCD matrix elements are strongly enhanced. We plot in Fig. 4 a 2d histogram of the truth-to-prediction ratio, $|\mathcal{M}|_{\text{true}}^2/|\mathcal{M}|_{\text{pred}}^2$, against the true value, $|\mathcal{M}|_{\text{true}}^2$, for 1M $gg \rightarrow e^-e^+ggd\bar{d}$ test events with standard cuts as described in Sec. 4.2. Along the sides, we plot the marginal distributions of the matrix element (top) and the ratio (right). The results illustrate that the ratio depicting model accuracy is centred around the ideal value of 1, with a steep drop off. This applies to the bulk of the events, as depicted by yellow coloured bins, tightly constrained to a narrow band. The purple coloured bins represent low population bins, or single points, which shows that the tails of the ratio distribution are primarily seen for smaller matrix element weights. Furthermore, the model accuracy remains high for the largest values of the matrix element, signalling that the infrared behaviour is well controlled. This is a key property of the factorisation-aware model. The emulation performance for the $Z + 5$ jets process is presented in Sec. 4.2.

An additional extension we study in this article is the inclusion of massive dipoles to our ansatz. This allows us to examine QCD processes with massive partons which is of particular importance for top-quark pair production in association with jets. We include the massive FF, FI, and IF dipoles from Ref. [23] into the emulation model. The massive dipoles are generalisations of the massless dipoles, meaning in principle it would be possible

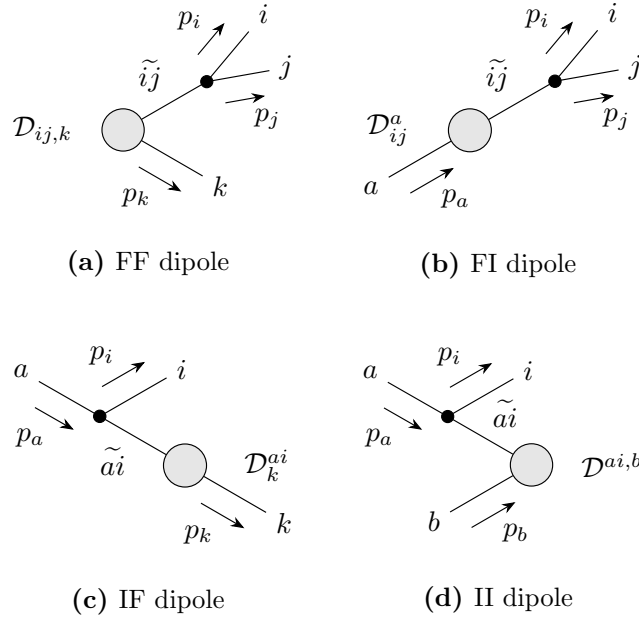


Figure 3: Schematic diagrams of the four classes of dipoles. The dipoles are named according to whether the emitter and spectator are in the initial (upper indices) or final state (lower indices). Each dipole consists of a composite particle (denoted by tilde) that decays into two partons, and a spectator that recoils to conserve momentum. The grey blob represents the hard scattering process, with incoming and outgoing lines representing initial- and final-state partons, respectively. The black circle represents the splitting function within the dipole function which contains the divergent behaviour.

to remove the massless dipoles from the ansatz. However, in practice, we only include the minimal set of necessary dipoles for a given partonic channel and so the inclusion of the massless dipoles reduces overall computational cost due to their relatively simpler expressions. With the massive dipoles implemented, our model contains the complete set of dipoles and is in principle able to take advantage of the factorisation-aware model for arbitrary processes involving QCD-enhanced behaviour at tree-level.

In order to showcase the extension to massive dipoles, we consider emulating tree-level matrix elements of three partonic channels: $gg \rightarrow t\bar{t}ggg$, and $u\bar{u} \rightarrow t\bar{t}g d\bar{d}$, contributing to leading order $t\bar{t} + 3$ jets production, and $ug \rightarrow t\bar{t}gggu$ which is a leading order contribution to $t\bar{t} + 4$ jets production. To validate the inclusion of these massive dipoles into the model, we show in Fig. 5 the deviation similar to Fig. 4 but for 1M tree-level events of $gg \rightarrow t\bar{t}ggg$ in proton–proton collisions at $\sqrt{s} = 13$ TeV, with cuts described in Sec. 4.2. We again observe the narrow yellow band, indicating that the bulk of the test events are accurately predicted, with the outliers corresponding to smaller matrix element values. The infrared behaviour is well captured by the model as can be seen by the narrow head for the largest matrix element values. For emulation performance of channels not described here, we refer the reader to Sec. 4.2 and App. A.

2.3 Colour-sampled matrix elements

The discussion so far has been focused on the emulation of colour-summed matrix elements as it was the case for Ref. [1]. In this work we take the first steps towards emulating colour-sampled matrix elements, such as those obtained from the COMIX generator [31, 32].

Based on the colour-flow decomposition of QCD amplitudes [33, 34], for each event, the generator samples a momentum configuration *and* a valid colour assignment, *i.e.* colour

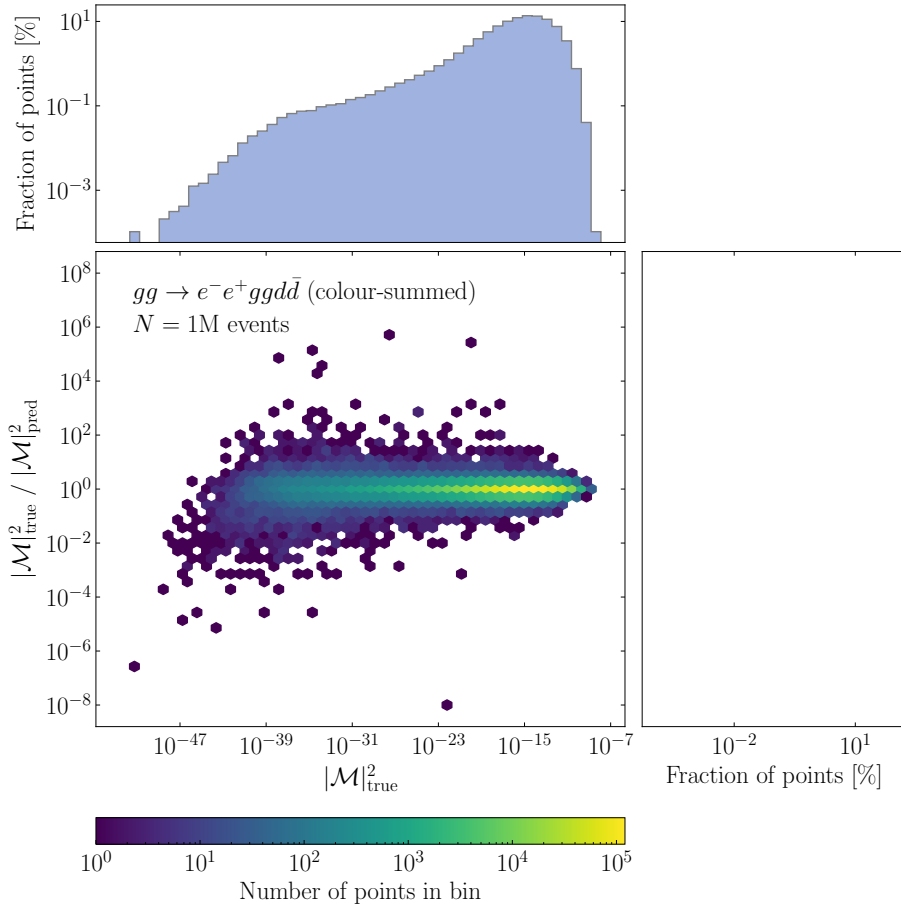


Figure 4: 2d histogram showing the distribution of truth-to-prediction ratios of the matrix element against the value of the true matrix element for the $Z + 4j$ process $gg \rightarrow e^- e^+ gg d \bar{d}$. Along the axes, we plot the marginal distributions of the matrix element (top), and the truth-to-prediction ratio (right). High population bins are illustrated as yellow, with low population bins, down to single points, are depicted in purple.

indices. The colour assignment thereby is represented by a vector of integers, C , where entries in the vector, $c_i \in \{1, 2, 3\}$, denote the colour assigned to a colour-charged parton in the process. Gluons have two colour indices corresponding to colour and anti-colour, whereas quarks/anti-quarks carry only one index.

We add this vector of colour assignments as an additional input to the NN to include the colour-sampled information from the generator. We one-hot encode the colour assignments such that colours are represented by 3-element vectors, *e.g.* $R = [0, 0, 1]$, $G = [0, 1, 0]$, and $B = [1, 0, 0]$, as the integer representation of colour assignments is not useful to the NN.

Given the actual colour of a parton is ambiguous, the matrix element should be invariant to any cyclic permutation of the specific colour assigned to a given quark or gluon. To give an example, the three permutations $C_1 = [R, B, B, G, R]$, $C_2 = [G, R, R, B, G]$, and $C_3 = [B, G, G, R, B]$ of a five colour assignment would lead to the same matrix element weight. To aid the NN in learning this behaviour, we take the three permutations and duplicate the other model inputs such that the training data is enlarged by a factor of three. This did not cause us to run into any computational bottlenecks in terms of memory or time taken to train the models. Note that this duplication of data is not required when making predictions.

The rest of the inputs to the NN model remain identical. We study to what extent

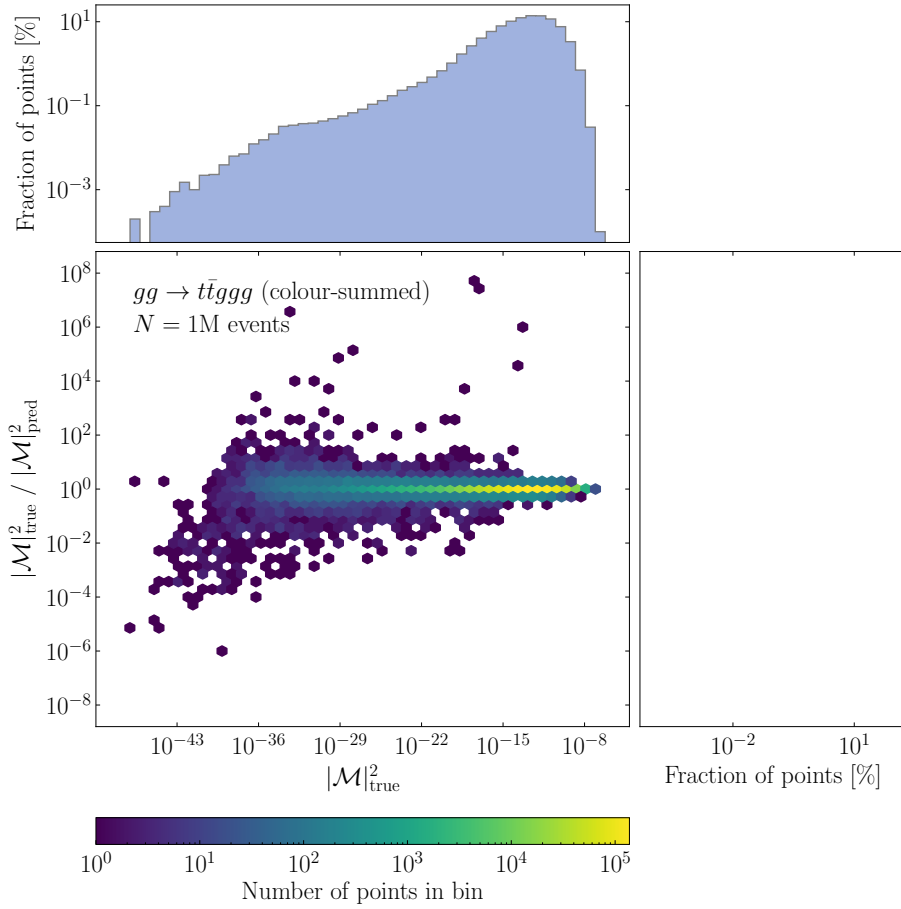


Figure 5: 2d histogram showing the distribution of truth-to-prediction ratios of the matrix element against the value of the true matrix element for the $t\bar{t} + 3j$ process $gg \rightarrow t\bar{t}ggg$. Along the axes, we plot the marginal distributions of the matrix element (top), and the truth-to-prediction ratio (right).

a naive approach of using the same dipole functions, which are most suitable for colour-summed matrix elements, works for the case of colour-sampled matrix elements. In the future, a more promising approach might be the application of coloured dipole terms directly. Their form has already been derived [35] and implemented for the dipole subtraction in the COMIX event generator [31] but is not implemented in our NN-based model yet.

To illustrate the emulation accuracy of colour-sampled matrix elements, here denoted $|\mathcal{M}|^2$, we plot the truth-to-prediction ratio in Fig. 6 for the $gg \rightarrow t\bar{t}ggg$ channel. While we again observe the property of well-behaved predictions for the larger matrix elements, evidently, the ratio distribution is much wider than in the colour-summed case. This decrease in accuracy directly translates to a lower expected gain factor when using this emulator as a surrogate model for event unweighting. This is discussed further in Sec. 4.2 where we elaborate on specific reasons for this decrease in accuracy and present possible future endeavours.

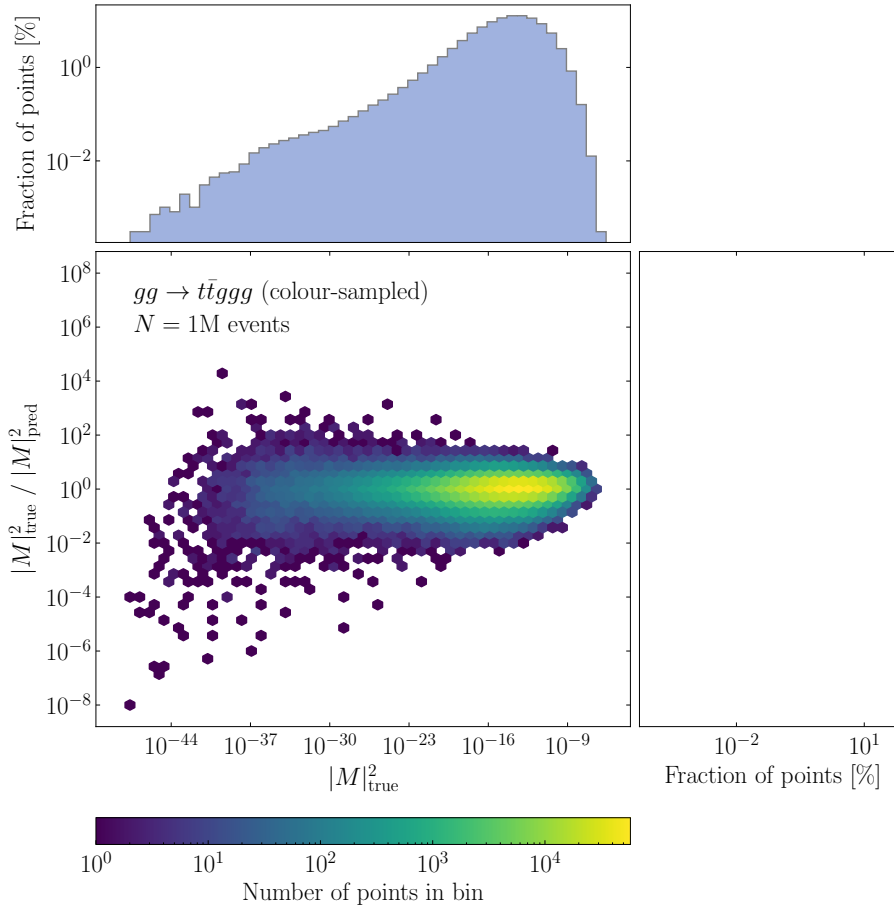


Figure 6: Truth-to-prediction ratio for colour-sampled $gg \rightarrow t\bar{t}ggg$ matrix elements against the colour-ordered partial amplitudes, $|M|^2$. Marginal distributions are plotted for the matrix elements (top) and ratios (right).

3 Event unweighting utilising matrix element surrogates

The unweighting of hard-scattering parton-level event samples constitutes an important step in the simulation of scattering events. The obtained unit-weight events then get passed on to subsequent evolution stages, including QCD parton showers, hadronisation, and possibly a detector simulation. However, the unweighting, based on rejection sampling, can pose a severe computational challenge, in particular when the evaluation time of the matrix element is long and the efficiency of the unweighting is rather low. To address this challenge Ref. [2] proposed a novel two-stage rejection sampling algorithm based on fast surrogates that we briefly review in this section. We furthermore generalise the performance measures to the case where the surrogate replaces the matrix element only, rather than its combination with the phase space weight as was the case in Ref. [2].

3.1 Two-stage unweighting method

The Monte Carlo method provides a numerical procedure to estimate integrals, *e.g.* partonic cross sections in high energy physics. When the integrand is non-trivial we use importance sampling to reduce the variance of the integral estimate. For a positive-definite target function $f : \Omega \subset \mathbb{R}^d \rightarrow [0, \infty)$ defined over the unit hypercube $\Omega = [0, 1]^d$ and a

probability density function g the Monte Carlo estimate of the integral

$$I = \int_{\Omega} f(u') du' = \int_{\Omega} \frac{f(v')}{g(v')} dv' \quad \text{with} \quad \int_{\Omega} g(u') du' = 1 \quad (7)$$

is given by

$$I \approx \frac{1}{N} \sum_{i=1}^N \frac{f(v_i)}{g(v_i)} = \langle w \rangle_g \quad (8)$$

with the pointwise event weight $w_i = f(v_i)/g(v_i)$. The points v_i are drawn from the distribution g . A suitable g can reduce the variance of the integral estimate and thereby increase the efficiency of the numerical integration. Finding such a function g is a difficult task, though, as one needs a way to efficiently draw samples from it. For multimodal target functions it is attractive to use a multi-channel approach, where g is defined by a mixture distribution. The weights of the channels can then be adapted automatically [36]. VEGAS [37] is an algorithm to automatically construct a sampling distribution g by optimising the bin widths of a piecewise-constant function. It can also be used to remap a given g or even the individual channels of a multi-channel distribution [38].

Besides the total integral we are typically interested in differential distributions of the points u_i , *i.e.* histograms of physical observables. Monte Carlo sampling produces weighted events so every entry in a histogram comes with a weight. Variance reduction methods like importance sampling also reduce the spread of weights but only a perfect sampler results in strictly uniform weights. A large weight spread is problematic when the samples are to be post-processed by detector simulations, as these are very expensive in terms of computation time per event. It is inefficient to apply them to events that yield a minuscule contribution to the total cross section. The alternative is to first impose a rejection sampling step to extract unit-weight samples. This converts a sample of N^{trials} weighted events into a set of $N \leq N^{\text{trials}}$ unweighted events by randomly accepting or rejecting every weighted event with the acceptance probability w/w_{max} where w_{max} is the maximal event weight. Even though the information of the rejected events is lost the overall efficiency can be significantly increased when detector simulation is more expensive than event generation.

A convenient measure for the performance of a Monte Carlo event generator is the unweighting efficiency ϵ of the rejection sampling step, defined as

$$\epsilon := \frac{N}{N^{\text{trials}}}. \quad (9)$$

For a large number of trial events it can be estimated by

$$\epsilon \approx \frac{\langle w \rangle}{w_{\text{max}}}, \quad (10)$$

where $\langle w \rangle$ is the mean of the N_{trials} weights in the event sample. The average number of target function evaluations needed to get one accepted event is then given by $1/\epsilon$. Similar to how the uncertainty on the integral estimate can be diminished by variance reduction methods, the unweighting efficiency can be increased by optimising the sampling density g for smaller w_{max} .

There is another way of reducing the computational footprint especially if the target function takes a long time to evaluate and has a rather low unweighting efficiency. This is typically the case for high multiplicity scattering processes. The enormous growth in the number of contributing Feynman diagrams makes high multiplicity matrix elements increasingly expensive. At the same time, the high dimensionality of phase space renders

it difficult to find a sampling density g that is well adapted to the target everywhere in the integration volume. Consequently, the unweighting efficiency typically decreases with increasing multiplicity, see for example [17]. In this situation one can reduce the overall event generation time through replacing the expensive matrix element by a fast and accurate surrogate. The inaccuracy inevitably introduced in this procedure can be fully corrected for in a second unweighting step, resulting in an unbiased method [2]. An outline of the algorithm is given in Alg. 1. In addition, a more extensive explanation follows below.

Algorithm 1: Two-stage rejection-sampling unweighting algorithm using an event-wise weight estimate.

```

while true do
  generate phase-space point  $u$ ;
  calculate approximate event weight  $s$ ;
  generate uniform random number  $R_1 \in [0, 1)$ ;
  # first unweighting step
  if  $s > R_1 \cdot w_{\max}$  then
    calculate exact event weight  $w$ ;
    determine ratio  $x = w/s$ ;
    generate uniform random number  $R_2 \in [0, 1)$ ;
    # second unweighting step
    if  $x > R_2 \cdot x_{\max}$  then
      return  $u$  and  $\tilde{w} = \max(1, s/w_{\max}) \cdot \max(1, x/x_{\max})$ 
    end
  end
end
end

```

We begin by generating a weighted trial event in the conventional way. In a first unweighting step we then compare the surrogate weight s to the weight maximum w_{\max} and accept the event with probability s/w_{\max} . For an event that gets rejected at this point we only had to evaluate the cheap surrogate. If the event gets accepted, however, we need to evaluate the true weight w and attach a correction weight $x = w/s$ to the event. In a second unweighting step, the event has an acceptance probability of x/x_{\max} . Like w_{\max} , x_{\max} has to be predetermined. When the surrogate yields an accurate approximation of the true weight, a large proportion of events gets accepted in the second unweighting step. We note that the algorithm can easily be extended to the case of not strictly positive event weights as shown in [2].

Alg. 1 contains a crucial detail regarding the weight maxima, namely that even after unweighting events can end up with weights $\tilde{w} > 1$ if s is larger than w_{\max} or if x is larger than x_{\max} . If the true maxima were used, this could never happen. However, given finite-sized samples an exact determination of w_{\max} is realistically not possible. It is often not even desirable since a small number of points with large weights can induce a prohibitively small unweighting efficiency without contributing significantly to the total integral. It can therefore be useful to work with a deliberately reduced maximum, provided the rare mismatches are corrected for by event weights. The resulting events will be partially unweighted since there can be some events that overshoot the maximum. These will receive an overweight $\tilde{w} = w/w_{\max} > 1$. Hereinafter, we adopt the approach used in SHERPA for finding the reduced maximum. The aim is that the remaining overweights do not contribute more than a fixed proportion to the integral. We set this share to 0.1%. This can be achieved by taking the sorted weights of a sample of weighted points and

finding the weight that cuts off the desired quantile. In SHERPA this is done automatically during the integration phase. We point out that using a reduced maximum is a fully unbiased technique commonly used in event generators. It is especially helpful when weight surrogates are used since the limited approximation quality of the surrogate can lead to particularly large outliers.

3.2 Performance analysis

To fairly evaluate the performance gain of the two-stage unweighting algorithm shown in Alg. 1 we take the average time it takes to generate a single (partially) unweighted event and compare it to the time it would take to generate the statistical equivalent using the standard unweighting procedure. We call the ratio between the two the effective gain factor f_{eff} :

$$f_{\text{eff}} := \frac{T_{\text{standard}}}{T_{\text{surrogate}}}. \quad (11)$$

In order to separate the actual unweighting from program initialisation and other aspects of event generation, we break the calculation down to the relevant ingredients:

$$f_{\text{eff}} = \frac{N_{\text{full}}^{\text{trials}} \cdot (\langle t_{\text{ME}} \rangle + \langle t_{\text{PS}} \rangle)}{N_{\text{1st,surr}}^{\text{trials}} \cdot (\langle t_{\text{surr}} \rangle + \langle t_{\text{PS}} \rangle) + N_{\text{2nd,surr}}^{\text{trials}} \cdot \langle t_{\text{ME}} \rangle} \quad (12)$$

$$= \frac{1}{\frac{\langle t_{\text{surr}} \rangle + \langle t_{\text{PS}} \rangle}{\langle t_{\text{ME}} \rangle + \langle t_{\text{PS}} \rangle} \cdot \frac{\epsilon_{\text{full}}}{\epsilon_{\text{1st,surr}} \epsilon_{\text{2nd,surr}}} + \frac{\langle t_{\text{ME}} \rangle}{\langle t_{\text{ME}} \rangle + \langle t_{\text{PS}} \rangle} \cdot \frac{\epsilon_{\text{full}}}{\epsilon_{\text{2nd,surr}}}}. \quad (13)$$

The average evaluation times of the full matrix element weight, the phase space weight and the matrix element surrogate, respectively, are denoted as $\langle t_{\text{ME}} \rangle$, $\langle t_{\text{PS}} \rangle$ and $\langle t_{\text{surr}} \rangle$. By $N_{\text{step}}^{\text{trials}}$ we denote the number of trials in the respective unweighting step. The unweighting efficiencies are defined as

$$\epsilon_{\text{full}} := \frac{N}{N_{\text{full}}^{\text{trials}}}, \quad \epsilon_{\text{1st,surr}} := \frac{N_{\text{2nd,surr}}^{\text{trials}}}{N_{\text{1st,surr}}^{\text{trials}}} \quad \text{and} \quad \epsilon_{\text{2nd,surr}} := \frac{N}{N_{\text{2nd,surr}}^{\text{trials}}}. \quad (14)$$

It should be noted that events rejected due to phase space constraints do not affect the unweighting efficiencies since the selection cuts can be applied solely based on the kinematics without having to evaluate the matrix element.

From Eq. (13) it is clear that an important requirement for significant gains are short evaluation times for the surrogate in comparison to the full matrix element, *i.e.* $\langle t_{\text{surr}} \rangle \ll \langle t_{\text{ME}} \rangle$. Furthermore, even with a fast and accurate surrogate gains are only possible when the original unweighting efficiency ϵ_{full} is small enough. Therefore, the surrogate unweighting method is of limited use when the sampling density is very well adapted to the target. For suitable processes it will thus be important to find a good balance between fast evaluation and high accuracy of the surrogate.

The efficiency ϵ_{full} can be estimated by

$$\epsilon_{\text{full}} \approx \frac{\langle w \rangle}{w_{\text{max}}} \quad (15)$$

from the weights w generated during an initial integration run, *i.e.* after adapting the phase space generator. For w_{max} we use the reduced value as described in Sec. 3.1. Analogously, we estimate $\epsilon_{\text{1st,surr}}$ by

$$\epsilon_{\text{1st,surr}} \approx \frac{\langle s \rangle}{w_{\text{max}}} \quad (16)$$

using the same weight maximum and the surrogate weights s determined for the events in the test dataset. Since the deviations of the surrogate should average out, one can expect the values of ϵ_{full} and $\epsilon_{1\text{st,surr}}$ to be close. The second unweighting efficiency $\epsilon_{2\text{nd,surr}}$ can be estimated by

$$\epsilon_{2\text{nd,surr}} \approx \frac{\langle x \rangle}{x_{\text{max}}} \quad (17)$$

using the values $x = w/s$ determined for the events in the test dataset. The reduced maximum x_{max} can be calculated analogously to w_{max} with the restriction that we have to weight the values of x by their corresponding values of s to take into account the acceptance probability in the first unweighting step.

To determine the times $\langle t_{\text{ME}} \rangle$, $\langle t_{\text{PS}} \rangle$ and $\langle t_{\text{surr}} \rangle$ we repeat the calculation of the full/surrogate matrix element and phase space weights for a number of events from the test dataset. Depending on the complexity of the process we need between 10 and 10 000 events for a reliable time estimate. Note, the value of $\langle t_{\text{surr}} \rangle$ includes the time for preprocessing the inputs and post-processing the outputs of the surrogate model.

4 Implementation and application to LHC processes

In this section we present the application of the dipole model emulation of QCD matrix elements in the unweighting of event samples for high-multiplicity scattering processes at the LHC, *i.e.* $Z + 4, 5$ jets and $t\bar{t} + 3, 4$ jets production at the LHC. Results presented in Ref. [2] were based on a simplified neural network surrogate, however, also included an approximation for the phase space weight. We will here contrast the results obtained before to the sophisticated dipole model surrogate and also comment on the challenges when using colour-sampled QCD amplitudes. We furthermore briefly describe an implementation in the workflow of the SHERPA framework [20,21]. Note that we here only need to consider the generation of the hard process partons, as this is factorised from the generation of initial- and final-state parton showers as well as non-perturbative phases such as hadronisation and the underlying event [6]. Furthermore we note that systematic variations of the hard event related to alternative PDF sets, or modifications in the scale choices can be evaluated on-the-fly for unweighted events, represented by variational weights, see for example [39,40].

4.1 Implementation in the SHERPA framework

The two-stage unweighting algorithm described in Sec. 3.1 has been implemented in SHERPA [2]. The framework provides two built-in tree-level matrix element generators: AMEGIC [41] and COMIX [31]. We use AMEGIC to evaluate colour-summed matrix elements and COMIX for colour-sampled ones. To adapt the integrator to the integrand SHERPA runs an initial optimisation phase. This is followed by an integration phase in which the optimised integrator is used to calculate the total cross section of the process. From the event weights produced in this phase the value of w_{max} is determined, based on the 0.1% maximum reduction method introduced in Sec. 3.1. We take 2M events from the integration phase as a training dataset by saving the momenta, matrix element and phase space weights, and, when using colour sampling, colour assignments. From the training dataset we use 800k events for training the model, 200k for validation during the training and 1M for testing the performance afterwards. We train the dipole model described in Sec. 2 using KERAS [24] with the TENSORFLOW [25] backend and save it in the ONNX format [42]. The 1M events from the test dataset are used to determine the value of x_{max} .

After the training of the surrogate model has been completed successfully, the determination of the surrogate matrix element value during event generation with SHERPA proceeds as follows. At the point where normally the matrix element would be calculated with AMEGIC or COMIX, we use the momenta of the current trial event to determine the additional inputs y_{ijk} and s_{ij} . Along with the momenta these are then fed into the model which we evaluate on a single CPU core using the C++ API of the ONNX Runtime package [30]. We find that ONNX Runtime evaluates the model several times faster than the header-only library frugally-deep [43], which was used in [2]. It is important to note that this introduces an additional dependency on a software library. We would, however like to emphasise that our method does not depend on the code with which the surrogate is evaluated. This affects only the evaluation time. It would even be possible to create an interface through which any suitable tool could be used for this purpose. The model evaluation yields the dipole coefficients C_{ijk} which are then combined with the dipole functions D_{ijk} according to Eq. (2). To determine the relevant dipoles we use a custom implementation, although there already exists an implementation of the dipole functions in SHERPA (used in the automated construction of infrared subtraction terms for NLO QCD and EW calculations [44,45]) which could in principle also be employed for the case considered here.

4.2 Results for LHC multijet production processes

To study the performance of the method we consider various partonic multijet processes at tree-level accuracy. We thereby follow the validation and benchmark strategies outlined in Ref. [2], considering Z +jets and $t\bar{t}$ +jets production in proton–proton collisions at $\sqrt{s} = 13$ TeV. In particular we present results for $Z + \{4, 5\}$ jets and $t\bar{t} + \{3, 4\}$ jets final states, thereby extending our previous study by one multiplicity. Jets get reconstructed with the anti- k_t algorithm [46] with $R = 0.4$. As parton density functions we use the NNPDF-3.0 NNLO set [47].

Z +jets

We examine the partonic channels $gg \rightarrow e^-e^+ggd\bar{d}$ and $gg \rightarrow e^-e^+ggg\bar{d}$ at the tree-level that represent leading-order contributions to $Z + 4$ jets and $Z + 5$ jets production at the LHC. Correspondingly, using the four-momenta as inputs for the surrogate model we have parameter spaces with 32 and 36 dimensions. These get supplemented by the corresponding dipole mapping variables and kinematic invariants, see Sec. 2.2. Cuts are implemented to constrain the fiducial phase space and, in turn, to regulate QCD infrared divergences. A dilepton invariant mass $m_{e^-e^+} > 66$ GeV and four, respectively, five jets with $p_{T,j} > 20$ GeV are enforced. Identical cuts are used for the training and the prediction.

As a first assessment of the quality of the surrogate we show in Fig. 7a the distribution of the ratio between the true event weight w and the surrogate event weight s for 1M test events for the exemplar channel $gg \rightarrow e^-e^+ggg\bar{d}$. The corresponding plot for the process with the lower multiplicity is shown in App. A. We compare the results of the dipole model with the naive model from Ref. [2]. We point out that the naive model learns the entire event weight, while the dipole model learns only the matrix element weight. For the representation in Fig. 7a, the approximated matrix element weight of the dipole model was therefore multiplied by the true phase space weight. While a perfect model would reproduce the true weight exactly, such that the ratio would be one for all events, our surrogates show deviations. In both cases the distribution is peaked at one and falls off rather symmetrically towards higher and lower values. For the dipole model the peak is more pronounced and has a steep slope towards the tails of the distribution. This indicates

Table 2: Performance measures for partonic channels contributing to $Z + \{4, 5\}$ jets production at the LHC.

Process	SHERPA default			with dipole-model surrogate				
	$t_{\text{ME}}[\text{ms}]$	$t_{\text{PS}}[\text{ms}]$	ϵ_{full}	$t_{\text{surr}}[\text{ms}]$	x_{max}	$\epsilon_{1\text{st,surr}}$	$\epsilon_{2\text{nd,surr}}$	f_{eff}
$gg \rightarrow e^- e^+ gg d \bar{d}$	54	0.40	1.411 %	0.14	2.6	1.418 %	39 %	16
$gg \rightarrow e^- e^+ ggg d \bar{d}$	16 216	5.70	0.076 %	0.20	3.6	0.085 %	29 %	269

that for the bulk of the events the dipole model produces results that are much closer to the true values than the ones from the naive model. While the naive model seems to tend to generate an excessive number of large weights, *i.e.* $s > w$, both models generate a small number of outliers with $s \ll w$, reaching values for $x = w/s$ of up to 10^7 . We also indicate the points where the values of x_{max} lie to show which parts of the distributions are cut off in the partial unweighting. The dipole model achieves a much smaller x_{max} than the naive model, 3.6 compared to 84.8. There are orders of magnitude between the large- x outliers and the value of x_{max} used for the unweighting. To underline that this does not contradict each other, we refer again to Figs. 4–6. There we can see that the large ratios x are suppressed with respect to the matrix element weight. These events therefore have a low probability of being accepted in the unweighting. Accordingly, the cut by the reduced x_{max} is justifiable because the outliers on average contribute little to the total cross-section due to their low frequency.

In Tab. 2 we summarise the evaluation times of the full and dipole-model surrogate weights, the efficiencies of the single- and two-stage unweighting, the maximum x_{max} for the second unweighting step and, finally, the effective gain factor f_{eff} . The evaluation of the surrogate is found to be orders of magnitude faster than the full matrix element calculation with AMEGIC. In the 4-jet case it is more than 300, and in the 5-jet case more than 80.000 times as fast. The evaluation of the phase space weights is fast in comparison to the full matrix element. However, it is of order, or even larger than $\langle t_{\text{surr}} \rangle$. We find that the additional complexity when increasing the multiplicity from four to five jets increases the matrix element evaluation time by a factor of 300 and reduces the unweighting efficiency by a factor of 20. Nevertheless, $\langle t_{\text{surr}} \rangle$ grows only by a factor less than two, while the approximation accuracy, reflected by x_{max} and $\epsilon_{2\text{nd,surr}}$, remains very similar. We obtain the values $x_{\text{max}} = 2.6$ and $\epsilon_{2\text{nd,surr}} = 0.39$ in the 4-jet case compared to $x_{\text{max}} = 3.6$ and $\epsilon_{2\text{nd,surr}} = 0.29$ for five jets. The effective gain factors yield 16 and 269, respectively.

$t\bar{t}$ +jets

As contributions to the processes $t\bar{t} + 3$ jets and $t\bar{t} + 4$ jets in hadronic collisions we here consider three partonic channels with varying number of external gluons, namely $u\bar{u} \rightarrow t\bar{t}g d \bar{d}$, $gg \rightarrow t\bar{t}g g g$ and $ug \rightarrow t\bar{t}g g g u$. In contrast to the previous examples these are pure QCD processes featuring massive coloured particles. Even though the final states contain one particle fewer than the Z +jets channels, these processes still pose a severe computational challenge. The direct coupling of gluons to the top quarks leads to a significant proliferation of Feynman diagrams in their jet-associated production. The input space dimensionalities are now 28 and 32, respectively. For the processes contributing to $t\bar{t} + 3$ jets we require three anti- k_t jets with $p_{T,j} > 20$ GeV. The fiducial phase space of the $t\bar{t} + 4$ jets channel is constrained by requiring four jets with staggered transverse-momentum cuts, namely $p_{T,1} > 100$ GeV, $p_{T,2} > 50$ GeV, $p_{T,3} > 40$ GeV and $p_{T,4} > 20$ GeV. We do not impose phase space restrictions on the external top quarks, that we treat as on-shell

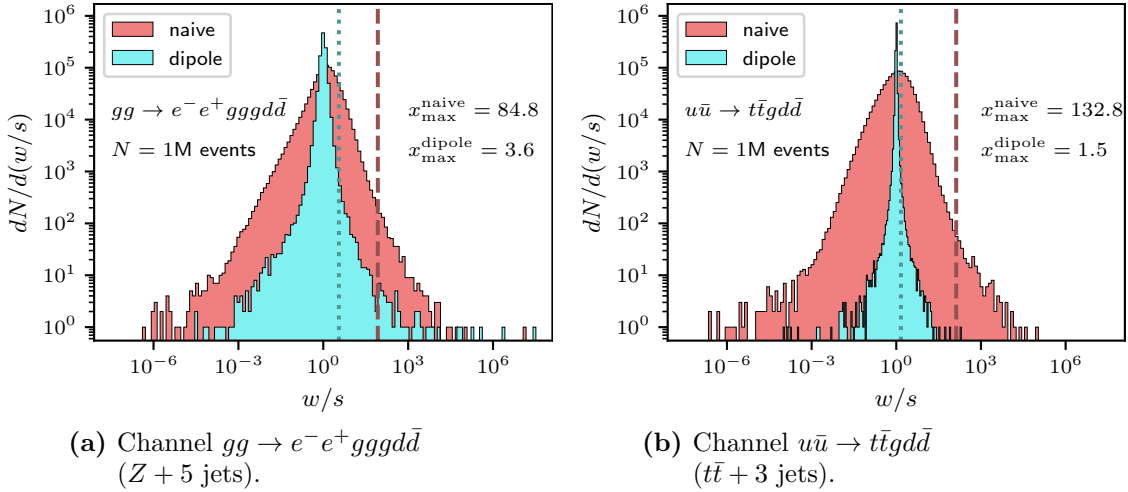


Figure 7: Ratio distributions of exact weights and their surrogate for the factorisation-aware emulation of the matrix-element weight (dipole) and the combined matrix-element and phase-space weight from Ref. [2] (naive).

Table 3: Performance measures for partonic channels contributing to $t\bar{t} + \{3, 4\}$ jets production at the LHC.

Process	SHERPA default			with dipole-model surrogate				f_{eff}
	$t_{\text{ME}}[\text{ms}]$	$t_{\text{PS}}[\text{ms}]$	ϵ_{full}	$t_{\text{surr}}[\text{ms}]$	x_{max}	$\epsilon_{1\text{st,surr}}$	$\epsilon_{2\text{nd,surr}}$	
$u\bar{u} \rightarrow t\bar{t}gdd\bar{d}$	5	0.04	0.092 %	0.14	1.5	0.092 %	69 %	20
$gg \rightarrow t\bar{t}ggg$	3262	0.90	1.093 %	0.18	1.4	1.128 %	69 %	61
$ug \rightarrow t\bar{t}gggu$	51 200	4.00	0.153 %	0.24	1.8	0.160 %	57 %	354

in the matrix element calculation, *i.e.* $p_t^2 = p_{\bar{t}}^2 = m_t^2$ with $m_t = 173.4$ GeV.

In Fig. 7b we show the ratio distributions of the true event weights and their surrogates for the dipole model and the naive model using the example of the partonic channel $u\bar{u} \rightarrow t\bar{t}gdd\bar{d}$. Note that the corresponding distributions for the other channels are shown in App. A. In comparison to Fig. 7a it can be seen that the distribution of the naive model is wider while the one of the dipole model is even narrower in this example. Moreover, it has visibly fewer outliers. This is also reflected in the values of x_{\max} , where the excellent result of 1.5 for the dipole model is two orders of magnitude smaller than the one for the naive model.

In Tab. 3 we compile the results obtained for the three partonic channels comparing the ordinary unweighting procedure with the two-stage surrogate technique. Again, we find significant speedups when using the dipole-model surrogate. For the process $ug \rightarrow t\bar{t}gggu$ the surrogate is in fact more than 200.000 times faster than the full matrix element weight evaluation. For all three examples the surrogate gives accurate approximations leading to values of x_{\max} between 1.4 and 1.8. The gain factors f_{eff} lie between 20 for the process $u\bar{u} \rightarrow t\bar{t}gdd\bar{d}$ and 354 for $ug \rightarrow t\bar{t}gggu$.

We compare the results for the effective gain factors for all five example processes in Fig. 8. For comparison we also include the results obtained using the simpler NN surrogate from Ref. [2] that were not contained in the tables. The values differ from the original publication because the definition of the renormalisation and factorisation scales has changed from a momenta-dependent one as used in Ref. [2] to a fixed value

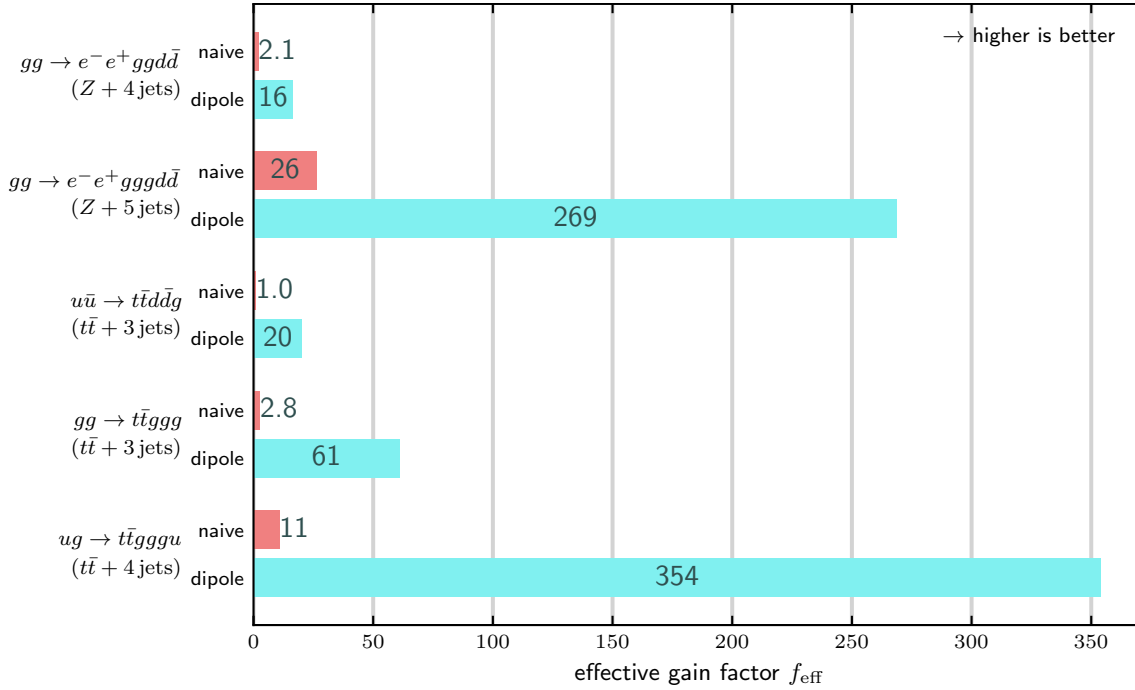


Figure 8: Effective gain factors for different processes. For comparison the results obtained using the *naive* neural network surrogate model from Ref. [2] are shown. Note that the naive model includes the phase space weight while the dipole model learns the matrix element weight only.

as used in Ref. [1]. This change leads to a slightly simpler learning problem and thus to slightly better performance. It can be seen that the dipole model achieves much larger gain factors. This can be attributed to the fact that the dipole model approximates the matrix elements much better because it already knows the relevant dipole structures for QCD emissions that dominate the multijet processes considered here. Furthermore, it is found that the respective highest multiplicity channels of the two process groups yield the largest gain factors. Adding an additional external particle causes the complexity of the calculation of the matrix element to grow significantly. This leads to a considerably increased evaluation time t_{ME} for the full weight, while the time t_{surr} for the surrogate changes only insignificantly. The impressive performance of the dipole surrogate model facilitates high gains even for those channels where the naive model from Ref. [2] led to minor gains only.

The influence of the training dataset size

In Fig. 9 we show how the value of x_{max} depends on the event sample size used to train the surrogate model for the different example processes. The number of training events is varied between 10^5 and 10^6 . A hierarchy can be identified: the models with the highest, *i.e.* worst, values of x_{max} gain the most from additional training data. For the process $gg \rightarrow e^-e^+gggd\bar{d}$ for example the resulting x_{max} is more than halved by going from 10^5 to 10^6 events. The processes with smaller x_{max} in comparison benefit less. For the process $gg \rightarrow t\bar{t}ggg$ the gain is only 23%. These observations carry over to Fig. 10 where the dependence of f_{eff} on the training dataset size is shown. According to Eq. (13) we have $f_{\text{eff}} \propto \epsilon_{2\text{nd},\text{surr}}$ and according to Eq. (15) we have $\epsilon_{2\text{nd},\text{surr}} \propto 1/x_{\text{max}}$. Therefore f_{eff} is inversely proportional to x_{max} . The largest improvement can again be seen for the process $gg \rightarrow e^-e^+gggd\bar{d}$ where the value of f_{eff} increases by 125% when going from 10^5 to 10^6

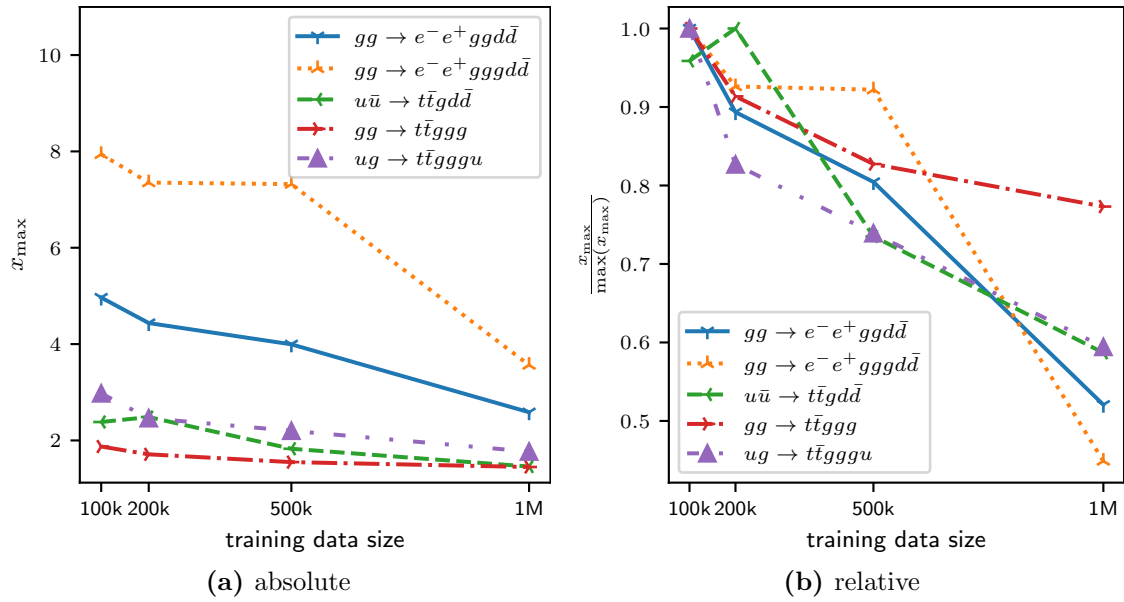


Figure 9: Influence of the training data size on the value of x_{\max} .

events. Likewise, the smallest improvement relates to the process $gg \rightarrow t \bar{t} g g g$ where the increase is only 22 %.

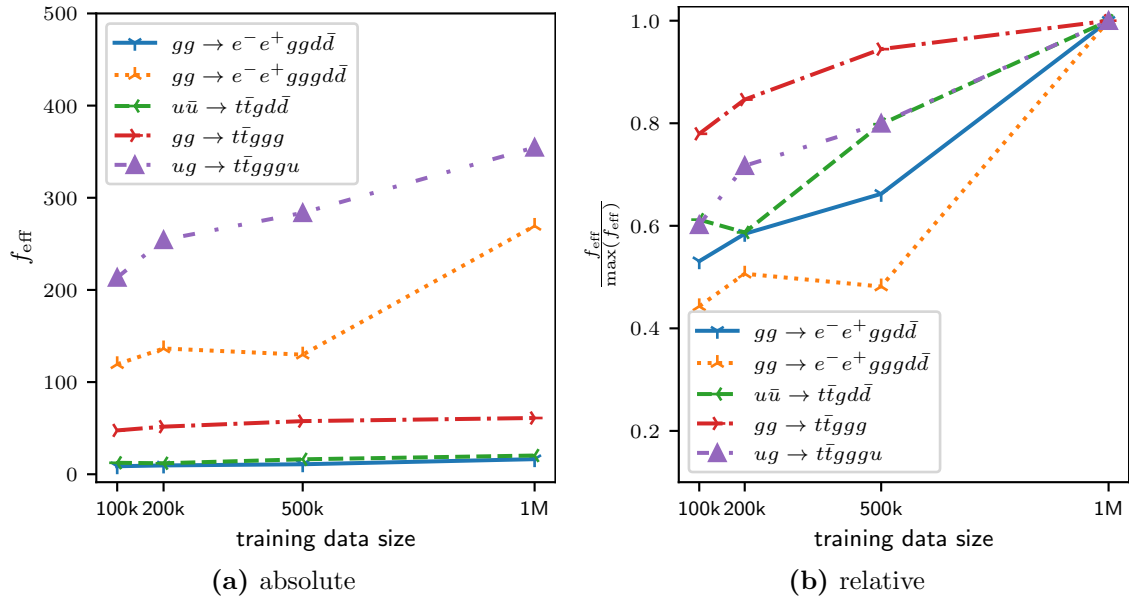


Figure 10: Influence of the training data size on the value of f_{eff} .

Results for colour-sampled amplitudes

The above examples are based on matrix elements with an explicit sum over the $SU(3)$ colour configurations of the involved partons. Using Monte Carlo integration techniques for phase space sampling, and possibly partonic flavours, a further option arises: just like the kinematic variables, we can also sample the colour assignments for the external partons. It can be shown [48] that colour sampling has a superior scaling behaviour compared to colour summation and therefore becomes much faster for large parton multiplicities. This holds even though colour sampling needs more points to reach a certain target precision. With

6–8 colour charged legs our examples already feature quite high dimensional colour spaces. It is thus worthwhile to test the performance of our method for colour sampled matrix elements. As a benchmark we use SHERPA with its built-in matrix element generator COMIX that implements colour sampling based on the colour-flow decomposition of QCD amplitudes [31, 32]. To keep things simple, we use a naive approach and employ basically the same surrogate model as before with the colour configuration as an additional input, see Sec. 2.3. While our model ansatz Eq. (2) is averaged over the colours, the neural network can try to learn the colour structure and encode it in the coefficients. As discussed in Sec. 2.3, an improved approach could use a new set of dipoles with explicit colour assignment in the future.

We trained the dipole model on the processes $gg \rightarrow e^-e^+ggd\bar{d}$ and $gg \rightarrow t\bar{t}ggg$ and found gain factors of 0.23 and 0.26, respectively. The performance is thus worse than using the standard unweighting when sampling colours. We checked that increasing the size of the training dataset does not lead to much higher gains. Three effects come into play here: first, the approximation quality of the model is worse because the complexity of the emulation problem increases significantly due to the additional colour degrees of freedom. Secondly, the evaluation time t_{ME} for the matrix element is now much shorter because instead of the whole sum only a single colour point needs to be evaluated. Thirdly, the evaluation time t_{PS} for the phase space weight is now no longer negligible. With COMIX it is of the same order of magnitude as t_{ME} . This makes it much more difficult to achieve large gains.

A way to deal with the last two points would be to let the surrogate also approximate the phase space weight such that

$$s' \approx w_{\text{ME}} \cdot w_{\text{PS}}. \quad (18)$$

Let us demonstrate this for the effective gain factor. In the limit of a highly accurate surrogate with $\epsilon_{1\text{st},\text{surr}} \approx \epsilon_{\text{full}}$ and $\epsilon_{2\text{nd},\text{surr}} \approx 1$ Eq. (13) becomes:

$$f_{\text{eff}} \approx \frac{1}{\frac{\langle t_{\text{surr}} \rangle + \langle t_{\text{PS}} \rangle}{\langle t_{\text{full}} \rangle} + \frac{\langle t_{\text{ME}} \rangle}{\langle t_{\text{full}} \rangle}} \cdot \epsilon_{\text{full}}. \quad (19)$$

Even in the ideal case where $\langle t_{\text{surr}} \rangle \rightarrow 0$ and $\epsilon_{\text{full}} \rightarrow 0$ there is an upper limit given by

$$f_{\text{eff}} \leq \frac{\langle t_{\text{full}} \rangle}{\langle t_{\text{PS}} \rangle}. \quad (20)$$

This is unproblematic as long as the evaluation of w_{PS} is cheap compared to w_{ME} . If this is not the case a surrogate that emulates the full weight is beneficial and results in an effective gain factor of:

$$f'_{\text{eff}} = \frac{1}{\frac{\langle t'_{\text{surr}} \rangle}{\langle t_{\text{full}} \rangle} \cdot \frac{\epsilon_{\text{full}}}{\epsilon'_{1\text{st},\text{surr}} \epsilon'_{2\text{nd},\text{surr}}} + \frac{\epsilon_{\text{full}}}{\epsilon'_{2\text{nd},\text{surr}}}}. \quad (21)$$

Considering again the limit of a highly accurate surrogate leads to

$$f'_{\text{eff}} \approx \frac{1}{\frac{\langle t'_{\text{surr}} \rangle}{\langle t_{\text{full}} \rangle} + \epsilon_{\text{full}}}. \quad (22)$$

The largest possible gain factor is thus $f'_{\text{eff}}{}^{\text{max}} = \epsilon_{\text{full}}^{-1}$. This corresponds to the same acceptance rate as without surrogate but with zero evaluation time. As was done in Ref. [2] we adapted the dipole surrogate model to include the phase space weight and evaluated the performance for the same two processes as above. We find gain factors of

0.02 and 0.22, respectively. Again, we do not achieve any gains compared to the standard unweighting. In this case the problem is that the neural network gives an even worse approximation since we include the phase space mapping which already tries to flatten the structures in the soft and collinear regions. So the model has to deal with a situation it was originally not designed for. The resulting losses eat up the gain from not having to calculate w_{PS} for every trial event.

The observations described above open up various options for improvement. One possibility, as mentioned before, would be to develop a surrogate model with colour-dependent dipoles, adequately representing amplitudes in a specific colour-flow assignment. In addition, one could attempt to explicitly incorporate knowledge about the employed phase space mappings.

5 Conclusions

We presented a case study of using a fast and accurate neural network emulation model for scattering matrix elements in the context of unweighted event generation for multijet processes. To this end we have generalised the model originally presented in Ref. [1], based on dipole factorisation, to account also for initial-state emissions and massive final-state partons. When considering QCD multijet processes this factorisation-aware model – using the parton four-momenta, dipole variables and kinematic invariants as inputs – provides very precise estimates for the squared transition amplitudes. This has been showcased for a selection of partonic channels contributing at the tree-level to hadronic $Z + 4, 5$ jets and $t\bar{t} + 3, 4$ jets production.

We then considered the trained networks in the ONNX format as fast surrogates for the full squared matrix elements in a two-stage rejection algorithm, originally presented in Ref. [2], in the SHERPA framework. This enables the production of unbiased samples of unweighted events that reproduce the exact target distribution, *i.e.* the true squared matrix element of the considered scattering process. Given a fast *and* accurate surrogate model, the effective gains are largest when two conditions are met: (i) the unweighting efficiency of the phase space integrator is rather low, and, (ii) the matrix element is time consuming to evaluate. For example, for the channels $gg \rightarrow e^-e^+ggg\bar{d}\bar{d}$ and $ug \rightarrow t\bar{t}gggu$ in proton–proton collisions at $\sqrt{s} = 13$ TeV, featuring default unweighting efficiencies for the AMEGIC integrator of 0.08% and 0.153%, we found gain factors of 269 and 354, respectively, when using our dipole-model surrogate. Accordingly, the computational resources needed to generate a given number of unweighted events get reduced by more than two orders of magnitude. At the same time, the overheads for training the surrogate network model are very modest, given that events from the compulsory integration phase prior to the generation process can be used for that purpose.

The underlying workflow for colour-summed squared matrix elements should be easily adaptable also for other matrix element providers and usage in experimental computing frameworks, given that in contrast to the original treatment from Ref. [2] we only employ the emulation of the matrix element expression and no longer include the generator specific phase space weight in the first-stage approximation. Furthermore, the ONNX standard allows one to easily store, transfer and exchange the trained neural networks, offering much flexibility in the method used to train the model.

Our results are valid for a sequential event generation workflow where events are generated one after the other on a single CPU core. We expect that the performance can be further increased by moving to a parallel workflow that generates multiple events at the same time using parallel hardware. The evaluation of the neural networks, which form

the basis for the surrogate models, can be easily vectorised and benefits in particular from accelerators such as GPUs.

Our study targeted high-multiplicity tree-level contributions that constitute a severe computational challenge in state-of-the-art matrix element plus parton shower simulations of multijet production processes [49–54], given that one can typically achieve NLO QCD accuracy only for somewhat lower multiplicities, see for instance [55]. However, in particular for the highest multiplicities sampling the colour assignments of the external partons outperforms their explicit summation. This poses new challenges to emulation models, given the high-dimensionality of the colour space. We explored naive extensions towards a suitably adjusted network model, though we were not able to achieve significant gains using a surrogate based on colour-summed dipoles. This is partly also due to the reduced evaluation times for partial amplitudes in the colour-flow decomposition. We are confident that under the same strategy but using colour-stripped dipoles in the surrogate ansatz and incorporating the phase space weight into the emulation useful gain factors could be achieved.

While the current paper addressed the emulation of high-multiplicity tree-level matrix elements, an extension to NLO calculations is rather straightforward, though there are additional challenges that have to be addressed. In Ref. [2] it was already shown that the two-staged unweighting algorithm is applicable also to non-positive definite target functions, *i.e.* negative event weights as they appear in subtraction-based higher-order calculations. In Ref. [56] an extended version of the factorisation-aware emulation model used here to QCD one-loop matrix elements has been presented. For the considered multijet production channels in electron–positron annihilation percent-level accuracy has been achieved. Upon generalisation to initial-state partons this could be used to lift unweighted event generation for LHC applications based on fast neural-network surrogates to NLO accuracy.

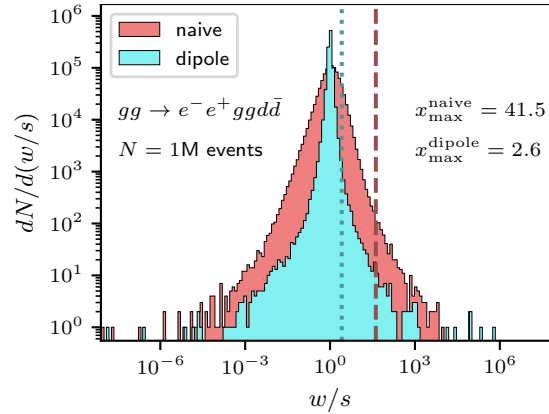
Acknowledgements

We are grateful for fruitful discussions with Enrico Bothmann, Stefan Höche, and Max Knobbe.

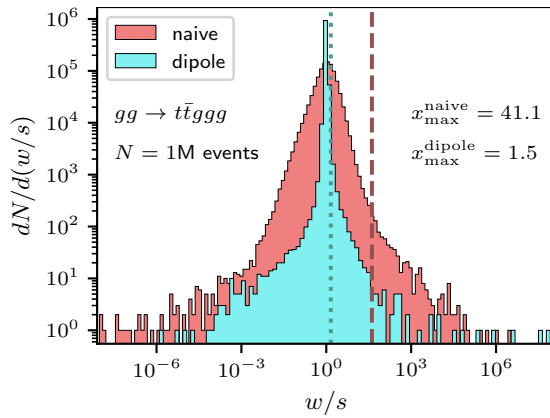
Funding information The work of SS and TJ was supported by BMBF (contract 05H21MGCAB) and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 456104544. FS’s research was supported by the German Research Foundation (DFG) under grant No. SI 2009/1-1. DM’s research was supported by STFC under grant ST/X003167/1.

A Auxiliary weight distributions

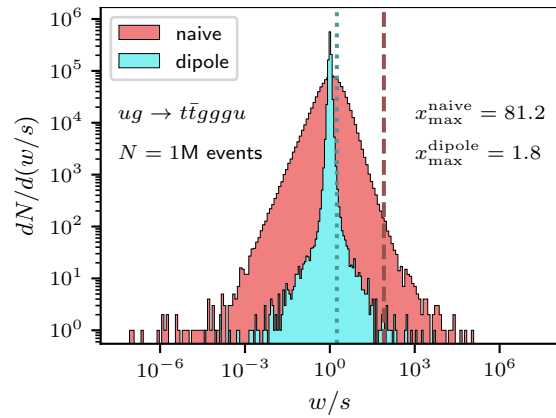
In this appendix we collect in Figs. 11a–11c auxiliary plots for the emulation accuracy of our dipole surrogate model (dipole) and the combined neural network surrogate for the matrix-element and phase-space weight from Ref. [2] (naive) for the remaining partonic channels. Shown are the ratios of the true weights and the respective surrogates. The two vertical lines indicate the corresponding maxima based on the 0.1% maximum reduction method, see Sec. 3.1.



(a) Channel $gg \rightarrow e^+ e^- gg d \bar{d}$
($Z + 4$ jets).



(b) Channel $gg \rightarrow t \bar{t} g g g$
($t \bar{t} + 3$ jets).



(c) Channel $ug \rightarrow t \bar{t} g g g u$
($t \bar{t} + 4$ jets).

Figure 11: Ratio distributions of exact weights and their surrogate for the factorisation-aware emulation of the matrix-element weight (dipole) and the combined matrix-element and phase-space weight from Ref. [2] (naive) for different partonic channels.

References

- [1] D. Maître and H. Truong, “A factorisation-aware Matrix element emulator,” *JHEP* **11** (2021) 066, [arXiv:2107.06625 \[hep-ph\]](#).
- [2] K. Danziger, T. Janßen, S. Schumann, and F. Siegert, “Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates,” *SciPost Phys.* **12** no. 5, (2022) 164, [arXiv:2109.11964 \[hep-ph\]](#).
- [3] P. Shanahan *et al.*, “Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning,” [arXiv:2209.07559 \[physics.comp-ph\]](#).
- [4] **HSF Physics Event Generator WG** Collaboration, S. Amoroso *et al.*, “Challenges in Monte Carlo Event Generator Software for High-Luminosity LHC,” *Comput. Softw. Big Sci.* **5** no. 1, (2021) 12, [arXiv:2004.13687 \[hep-ph\]](#).
- [5] S. Badger *et al.*, “Machine Learning and LHC Event Generation,” [arXiv:2203.07460 \[hep-ph\]](#).
- [6] A. Buckley *et al.*, “General-purpose event generators for LHC physics,” *Phys. Rept.* **504** (2011) 145–233, [arXiv:1101.2599 \[hep-ph\]](#).
- [7] J. M. Campbell *et al.*, “Event Generators for High-Energy Physics Experiments,” in *2022 Snowmass Summer Study*. **3**, 2022. [arXiv:2203.11110 \[hep-ph\]](#).
- [8] J. Bendavid, “Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks,” [arXiv:1707.00028 \[hep-ph\]](#).
- [9] M. D. Klimek and M. Perelstein, “Neural Network-Based Approach to Phase Space Integration,” *SciPost Phys.* **9** (2020) 053, [arXiv:1810.11509 \[hep-ph\]](#).
- [10] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann, “Exploring phase space with Neural Importance Sampling,” *SciPost Phys.* **8** no. 4, (2020) 069, [arXiv:2001.05478 \[hep-ph\]](#).
- [11] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, “Event Generation with Normalizing Flows,” *Phys. Rev. D* **101** no. 7, (2020) 076002, [arXiv:2001.10028 \[hep-ph\]](#).
- [12] B. Stienen and R. Verheyen, “Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows,” *SciPost Phys.* **10** (2021) 038, [arXiv:2011.13445 \[hep-ph\]](#).
- [13] I.-K. Chen, M. D. Klimek, and M. Perelstein, “Improved Neural Network Monte Carlo Simulation,” *SciPost Phys.* **10** (2021) 023, [arXiv:2009.07819 \[hep-ph\]](#).
- [14] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer, and T. Plehn, “MadNIS – Neural Multi-Channel Importance Sampling,” [arXiv:2212.06172 \[hep-ph\]](#).
- [15] D. Yallup, T. Janßen, S. Schumann, and W. Handley, “Exploring phase space with Nested Sampling,” *Eur. Phys. J. C* **82** (2022) 8, [arXiv:2205.02030 \[hep-ph\]](#).
- [16] K. Kröninger, S. Schumann, and B. Willenberg, “(MC)**3 – a Multi-Channel Markov Chain Monte Carlo algorithm for phase-space sampling,” *Comput. Phys. Commun.* **186** (2015) 1–10, [arXiv:1404.4328 \[hep-ph\]](#).

- [17] S. Höche, S. Prestel, and H. Schulz, “Simulation of Vector Boson Plus Many Jet Final States at the High Luminosity LHC,” *Phys. Rev.* **D100** no. 1, (2019) 014024, arXiv:1905.05120 [hep-ph].
- [18] J. Aylett-Bullock, S. Badger, and R. Moodie, “Optimising simulations for diphoton production at hadron colliders using amplitude neural networks,” *JHEP* **08** (2021) 066, arXiv:2106.09474 [hep-ph].
- [19] S. Badger, A. Butter, M. Luchmann, S. Pitz, and T. Plehn, “Loop Amplitudes from Precision Networks,” arXiv:2206.14831 [hep-ph].
- [20] **Sherpa** Collaboration, E. Bothmann *et al.*, “Event Generation with Sherpa 2.2” *SciPost Phys.* **7** no. 3, (2019) 034, arXiv:1905.09127 [hep-ph].
- [21] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, and J. Winter, “Event generation with SHERPA 1.1” *JHEP* **02** (2009) 007, arXiv:0811.4622 [hep-ph].
- [22] S. Catani and M. H. Seymour, “A General algorithm for calculating jet cross-sections in NLO QCD,” *Nucl. Phys. B* **485** (1997) 291–419, arXiv:hep-ph/9605323. [Erratum: *Nucl.Phys.B* 510, 503–504 (1998)].
- [23] S. Catani, S. Dittmaier, M. H. Seymour, and Z. Trocsanyi, “The Dipole formalism for next-to-leading order QCD calculations with massive partons,” *Nucl. Phys. B* **627** (2002) 189–265, arXiv:hep-ph/0201036.
- [24] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [25] M. Abadi, A. Agarwal, *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems.” 2015. <https://www.tensorflow.org/>. Software available from [tensorflow.org](https://www.tensorflow.org/).
- [26] S. Elfving, E. Uchibe, and K. Doya, “Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning,” *arXiv e-prints* (Feb., 2017) arXiv:1702.03118, arXiv:1702.03118 [cs.LG].
- [27] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” *arXiv e-prints* (Oct., 2017) arXiv:1710.05941, arXiv:1710.05941 [cs.NE].
- [28] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterton, eds., vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy, 13–15 may, 2010. <https://proceedings.mlr.press/v9/glorot10a.html>.
- [29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv:1412.6980 [cs.LG].
- [30] O. R. developers, “Onnx runtime,” <https://github.com/microsoft/onnxruntime>, 11, 2018.
- [31] T. Gleisberg and S. Höche, “Comix, a new matrix element generator,” *JHEP* **12** (2008) 039, arXiv:0808.3674 [hep-ph].

- [32] S. Höche, S. Kuttimalai, S. Schumann, and F. Siegert, “Beyond Standard Model calculations with Sherpa,” *Eur. Phys. J. C* **75** no. 3, (2015) 135, [arXiv:1412.6478 \[hep-ph\]](#).
- [33] F. Maltoni, K. Paul, T. Stelzer, and S. Willenbrock, “Color Flow Decomposition of QCD Amplitudes,” *Phys. Rev. D* **67** (2003) 014026, [arXiv:hep-ph/0209271](#).
- [34] C. Duhr, S. Höche, and F. Maltoni, “Color-dressed recursive relations for multi-parton amplitudes,” *JHEP* **08** (2006) 062, [arXiv:hep-ph/0607057](#).
- [35] S. Höche, private communication, 2023.
- [36] R. Kleiss and R. Pittau, “Weight optimization in multichannel Monte Carlo,” *Comput. Phys. Commun.* **83** (1994) 141–146, [arXiv:hep-ph/9405257 \[hep-ph\]](#).
- [37] G. P. Lepage, “A new algorithm for adaptive multidimensional integration,” *Journal of Computational Physics* **27** no. 2, (1978) 192 – 203. <http://www.sciencedirect.com/science/article/pii/0021999178900049>.
- [38] T. Ohl, “Vegas revisited: Adaptive Monte Carlo integration beyond factorization,” *Comput. Phys. Commun.* **120** (1999) 13–19, [arXiv:hep-ph/9806432](#).
- [39] E. Bothmann, M. Schönherr, and S. Schumann, “Reweighting QCD matrix-element and parton-shower calculations,” *Eur. Phys. J. C* **76** no. 11, (2016) 590, [arXiv:1606.08753 \[hep-ph\]](#).
- [40] 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** no. 12, (2022) 1128, [arXiv:2209.00843 \[hep-ph\]](#).
- [41] F. Krauss, R. Kuhn, and G. Soff, “AMEGIC++ 1.0: A Matrix element generator in C++,” *JHEP* **02** (2002) 044, [arXiv:hep-ph/0109036](#).
- [42] J. Bai, F. Lu, K. Zhang, *et al.*, “Onnx: Open neural network exchange,” <https://github.com/onnx/onnx>, 2019.
- [43] T. Hermann, “frugally-deep,” <https://github.com/Dobiasd/frugally-deep>, 2016-2021.
- [44] T. Gleisberg and F. Krauss, “Automating dipole subtraction for QCD NLO calculations,” *Eur. Phys. J. C* **53** (2008) 501–523, [arXiv:0709.2881 \[hep-ph\]](#).
- [45] M. Schönherr, “An automated subtraction of NLO EW infrared divergences,” *Eur. Phys. J. C* **78** no. 2, (2018) 119, [arXiv:1712.07975 \[hep-ph\]](#).
- [46] M. Cacciari, G. P. Salam, and G. Soyez, “The anti- k_t jet clustering algorithm,” *JHEP* **04** (2008) 063, [arXiv:0802.1189 \[hep-ph\]](#).
- [47] NNPDF Collaboration, R. D. Ball *et al.*, “Parton distributions for the LHC Run II,” *JHEP* **04** (2015) 040, [arXiv:1410.8849 \[hep-ph\]](#).
- [48] E. Bothmann, W. Giele, S. Höche, J. Isaacson, and M. Knobbe, “Many-gluon tree amplitudes on modern GPUs: A case study for novel event generators,” *SciPost Phys. Codebases* (2022) 3. <https://scipost.org/10.21468/SciPostPhysCodeb.3>.

- [49] S. Catani, F. Krauss, R. Kuhn, and B. R. Webber, “QCD matrix elements + parton showers,” *JHEP* **11** (2001) 063, [arXiv:hep-ph/0109231](#).
- [50] L. Lönnblad, “Correcting the color dipole cascade model with fixed order matrix elements,” *JHEP* **05** (2002) 046, [arXiv:hep-ph/0112284](#).
- [51] J. Alwall *et al.*, “Comparative study of various algorithms for the merging of parton showers and matrix elements in hadronic collisions,” *Eur. Phys. J. C* **53** (2008) 473–500, [arXiv:0706.2569 \[hep-ph\]](#).
- [52] S. Höche, F. Krauss, S. Schumann, and F. Siegert, “QCD matrix elements and truncated showers,” *JHEP* **05** (2009) 053, [arXiv:0903.1219 \[hep-ph\]](#).
- [53] S. Höche, F. Krauss, M. Schönherr, and F. Siegert, “QCD matrix elements + parton showers: The NLO case,” *JHEP* **04** (2013) 027, [arXiv:1207.5030 \[hep-ph\]](#).
- [54] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations,” *JHEP* **07** (2014) 079, [arXiv:1405.0301 \[hep-ph\]](#).
- [55] **ATLAS** Collaboration, G. Aad *et al.*, “Modelling and computational improvements to the simulation of single vector-boson plus jet processes for the ATLAS experiment,” *JHEP* **08** (2022) 089, [arXiv:2112.09588 \[hep-ex\]](#).
- [56] D. Maître and H. Truong, “One-loop matrix element emulation with factorisation awareness,” [arXiv:2302.04005 \[hep-ph\]](#).

4.4 Relationship to the other methods

The surrogate-based unweighting method presented in sections 4.2 and 4.3 can improve the efficiency of unweighted event generation by reducing the number of calls to an expensive integrand, and shows the greatest potential when the initial unweighting efficiency is low. In contrast, the methods presented in sections 3.1.3 and 3.2.2 aim at directly increasing the unweighting efficiency. The question naturally arises to what extent the approaches can be combined. Since all the approaches presented here leave the physics untouched and do not introduce biases, there is no danger of introducing errors through combining them. However, it is possible that a single method achieves a higher efficiency than a combination with another.

It is to be expected that at low to moderate dimensionalities, i.e. partonic final state multiplicities, the NF-based importance sampling can reach high efficiencies. At the same time, the MES in these cases are at the most moderately expensive. Therefore, in these cases, no or only small efficiency gains can be expected from the surrogate-based unweighting. That method is more suited for the highest multiplicities, where, due to the curse of dimensionality, expensive integrands and low unweighting efficiencies typically occur. As shown above, the time for event generation can be reduced by orders of magnitude for specific partonic subprocesses. If the unweighting method would be combined with an improved PS sampler, e.g. based on NFs, these gains would of course be reduced. However, this should be combined with the gain from improving the PS sampler in the first place. Note that at high dimensionalities it becomes increasingly hard to improve the unweighting efficiency due to the large PS volume. Thus, it can still be worthwhile to combine the approaches.

The situation is somewhat different for nested sampling. There, the overall efficiency is dominated by the efficiency of finding new points under the hard likelihood constraints. The actual unweighting efficiency is comparatively large, as can be seen in section 3.2.2. This means that using a surrogate for the unweighting does not appear useful here. However, a combination of nested sampling with an NF-based prior would be interesting to consider and this option is discussed in section 3.3. It is worth mentioning that the nested sampling implementation we used, POLYCHORD, seems to be especially well suited for very high dimensionalities due to its excellent scaling behaviour. There are, however, other implementations that can be more efficient for a moderate number of dimensions. One example is MULTINEST [271], which relies on rejection sampling for finding new live points. It approximates the iso-likelihood contours by ellipsoids determined from clusters of the current live points. The ellipsoidal bounds are used to sample new points until one is found within the iso-likelihood contour. For each trial point, the target likelihood has to be evaluated, and this can be quite expensive if the efficiency is low. Using a cheaper NN surrogate may actually provide some performance improvements. Our surrogate unweighting method is in fact generic and can in principle be applied to any problem involving rejection sampling. It would be an interesting future project to evaluate the combination of MULTINEST with likelihood surrogates.

5 Summary and conclusions

In this thesis, I have presented different approaches to improving the efficiency of MC event generators, which I have studied during my time as a doctoral student. As motivated in the introduction, these aim at contributing to the ongoing efforts towards improving the efficiency of the computational toolchain in the light of the upcoming HL-LHC. The articles that came out of my research have been introduced, reprinted and discussed in the previous chapters. In this chapter I formulate an overall summary, state my conclusions and present my ideas for future research that could build upon the work presented so far.

A lesson that I learned through this work is that a lot of effort has been spent on achieving the state of the art of MC event generation. This includes not only refinements in physics but also various efficiency improvements. Without these, some features of the simulations would not even be usable. Accordingly, it is not an easy task to further improve the efficiency of the existing tools. In this thesis, I set the focus on exploring the suitability of modern ML techniques for this challenge. I applied three different methods to realistic examples and obtained promising results that could lead to more efficient event generation for relevant applications by extending or complementing our existing frameworks.

The main results of this thesis are summarized in section 5.1. I conclude in section 5.2, where I also highlight some opportunities to combine the ideas in this thesis in order to maximize the overall efficiency improvement.

5.1 Main results

In section 3.1 the neural importance sampling method is considered, which uses NFs as adaptive mappings for efficient importance sampling and rejection sampling. Due to the bijectivity of the NFs, the method guarantees unbiased sampling. Spline-based transformers, using e.g. piecewise-quadratic splines, are defined on the unit hypercube and allow NFs to be used as a drop-in replacement of the popular adaptive importance sampling tool VEGAS. Furthermore, with an NN parameterization and multiple transformation layers, the NF mapping becomes highly expressive, and it can overcome the limitations of VEGAS with respect to non-factorizable target functions.

A proof of principle study for applying neural importance sampling to efficient PS sampling in high energy scattering event generation was presented in our publication, ref. [9], and reprinted in section 3.1.3. We implemented NFs with piecewise-quadratic coupling layers and used them to optimize process-specific, but non-optimal, PS mappings. The decay width of the top quark, as well as the cross-sections of leptonic top quark pair production and gluonic scattering with 3-jet and 4-jet production served as examples. Thereby, the PS dimensionality ranged from 2 to 8 dimensions. For the top quark examples, we used a single importance sampling channel, where the s -channel propagators of the W boson were modelled by Breit-Wigner distributions. A single NF was used to remap that channel. For the more complex gluon scattering examples, we employed the HAAG multichannel PS sampler, which is based on QCD antenna functions. It features 2 and 3 channels, respectively, which we optimized with one NF each.

For all examples, except for the one with the highest dimensionality, 4-jet production, our approach outperformed VEGAS. We were able to significantly improve the unweighting efficiency as well as the variance of the cross-section estimate. As expected, no relevant deviations

were found in differential distributions of observables. For the example of 4-jet production, the results were on par with VEGAS, but not strictly better. We attributed this to the complexity of the example, and to our limited computing resources that did not allow to increase the size of the NF model and its training time arbitrarily. However, as stated in section 3.1.4, I was able to improve the performance significantly at a later stage, so that the unweighting efficiency is better than VEGAS by a factor of about 1.5. Still, the gain is below the one of the 3-jet production example, where the factor is 2.3. This is consistent with the results in another study [47], which used NFs to optimize a recursive multichannel sampler for the production of vector bosons in association with jets at the LHC. There, the gains also decreased when the final state multiplicity was increased.

In section 3.2, I presented a study that introduced nested sampling as a technique for PS sampling. On the one hand, this study was targeted at researchers in the HEP community to make them aware of this technique and its possible applications. On the other hand, it showed researchers in the field of Bayesian inference the similarities between their typical problems and the sampling and integration of high-energy cross-sections. Using the example of gluon scattering processes with the production of 3-, 4-, and 5-gluon final states, it was shown that the nested sampling implementation POLYCHORD can efficiently generate unweighted events without any prior knowledge. The only ingredient we used was a RAMBO mapping that maps the gluon four-momenta to the unit hypercube, and in doing so implements four-momentum conservation and on-shell conditions. However, the RAMBO mapping is uniform and therefore does not provide any information about the shape of the differential cross-section. We were able to show that nested sampling outperformed VEGAS for the higher multiplicities and was comparable, in terms of unweighting efficiency, to the dedicated multichannel sampler HAAG. Thereby, nested sampling featured an excellent scaling behaviour that was basically constant over the considered multiplicities. One could therefore hope that this scaling continues to even higher multiplicities, such that nested sampling provides an efficient approach in these cases, which needs very little tuning.

A technique that is complementary to the previous two approaches is the surrogate unweighting method presented in chapter 4. It is fully general and can in principle be applied to any application of rejection sampling. The requirements for the method to be useful are that the efficiency is low to begin with, and that the target function is expensive in terms of computational resources needed to evaluate it. It also requires a fast and accurate surrogate, which is used in place of the actual target function during the unweighting process. The outstanding quality of the method is that it is fully unbiased. In the worst case, it can reduce the total efficiency of event generation, in which case it should not be used. However, it will not alter the distributions of events. This is achieved by correcting the approximation errors of the surrogate in a second unweighting step, where the value of the true target function is used. The second unweighting step is comparably expensive, but it only has to be executed for the events accepted in the first unweighting step. Since the efficiency in the first step is typically low, and since an accurate surrogate can achieve high efficiency in the second step, the true target function needs to be evaluated less often in total. This results in an overall reduction of the time needed for unweighted event generation.

In ref. [11], reprinted in section 4.2, we implemented the method in the SHERPA event generator and applied it to high-multiplicity partonic channels contributing to the production of $Z+4$ jets, $W+4$ jets, and $t\bar{t}+3$ jets at the LHC. We used a simple NN surrogate for the full event weight, i.e. the product of the ME weight and the PS weight. The surrogate was significantly faster to evaluate than the true weight, between 25 and 40000 times, while it was well capable of estimating it. For the dominant partonic processes, which contribute the most to the total cross-sections and feature expensive MES, we found accelerated unweighted event generation by factors between two and ten. We found no significant deviations from the expected differential distributions of physical observables. Since we applied partial unweighting to protect against

rare outliers, we also looked for imprints of overweighted events in the distributions, which we did not find. Furthermore, it was shown on a toy example how the surrogate unweighting method can be applied to NLO event generation, where negative event weights have to be dealt with.

The surrogate model used in ref. [11] was quite simple and not particularly optimized for the examples. However, the method itself is completely indifferent to what kind of surrogate is used. The main constraint is that one has to find a good balance between fast evaluation and high accuracy. As an extension of our previous work, we combined it with the ME emulation model of ref. [64]. The results of our follow-up study were published in a preprint [12], which is reprinted in section 4.3. The model is based on dipole factorization and has built-in knowledge about the soft and collinear singularity structure of tree-level QCD MES, allowing it to interpolate smoothly between the divergent regions. For our study, the model was generalized to account also for initial-state emissions, which are necessary for hadronic collisions, and massive quarks in the final state. We applied our surrogate unweighting method in combination with the factorization-aware model to partonic channels contributing to $Z + 4, 5$ jets and $t\bar{t} + 3, 4$ jets production at the LHC, thereby going one multiplicity higher than in our previous study. We found that, in comparison to the naive model used previously, the factorization-aware model increased the efficiency gains by an order of magnitude due to its better approximation. For the partonic processes $gg \rightarrow e^-e^+ggg\bar{d}$ and $ug \rightarrow t\bar{t}gggu$, contributing to the highest multiplicities of our examples, we found a reduction in the time needed for unweighted event generation by factors of 269 and 354, respectively. Although the new model is more complex in that it requires a more sophisticated pre- and post-processing, it still evaluates much faster than the true ME. Furthermore, the time needed for training the NN is negligible compared to the time spent later on in unweighted event generation.

While our naive model represented a surrogate for the full event weight, the factorization-aware model is an approximation to the ME weight only. This frees the surrogate unweighting method from dependence on the integrator-specific PS generator. Therefore, the trained models can be easily transferred to other frameworks and used there for the same processes. This is especially simplified by the usage of the ONNX format, which allows for various ML libraries to be used for the training of the model and an easy exchange and interfacing of trained models.

In our studies of the surrogate unweighting method, we focused on the use of colour-summed MES. However, it is known that an explicit sampling of the colour assignments of the external partons exhibits a superior scaling behaviour and outperforms their summation at high multiplicities. Given the higher dimensionality of the input space, it is more difficult to find a good surrogate model in this situation. This must be seen in the light of the fact that partial amplitudes in the colour flow decomposition can be evaluated much faster than colour-summed matrix elements. This gives the evaluation time of the surrogate a new significance. Since also the evaluation time of the PS weight can be a significant factor, it should be considered again to include the PS weight in the surrogate, as we did with the naive model in our first study. Unfortunately, a naive extension of the factorization-aware model with colour assignments as additional inputs did not achieve satisfactory gains. However, we are confident that also with colour sampling the efficiency of unweighted event generation can be increased by finding suitable surrogate models, e.g. based on a colour-stripped dipole ansatz.

5.2 Prospects for the future

Several paths can be identified in which the ideas in this thesis can be extended and used for further studies in the future. An obvious extension is to pursue the perturbative series and move to NLO calculations. The work contributing to this thesis concentrated on tree-level calculations for mainly two reasons. Firstly because tree-level simulations are much easier than loop calculations and therefore the obvious first choice for proof-of-principle studies.

Secondly, the high-multiplicity tree-level part is often the most expensive one in key processes. This is because the NLO contributions in multijet merged calculations are typically at a lower multiplicity. However, the situation can be different for specific processes, rendering efficiency improvements for NLO calculations an interesting research path.

For the NF-based sampling there would be no major changes necessary, since it is basically a replacement for VEGAS mappings. These are already used for NLO calculations in state-of-the-art event generators. For nested sampling the situation is more difficult. The algorithm requires a non-negative integrand, which means the negative weights at NLO would pose a problem. However, nested sampling in the form of the POLYCHORD implementation is possibly best suited for the highest multiplicity tree-level contributions. The reason lies in its polynomial scaling with dimensionality, which makes it an excellent tool for sampling and integration in very high dimensions. For the surrogate unweighting method, we have already shown in ref. [11] how it can be applied to non-positive integrands. The extension is minimal and requires a surrogate that can become negative. Given a suitable surrogate model for loop MEs, the method can be readily applied to NLO calculations. In fact, several emulation models for loop amplitudes have already been developed [72, 76, 77, 79].

An open question is whether NFs can provide an efficient solution for high-multiplicity processes. In our study, we achieved efficiency gains for all considered examples, up to 4-gluon final states with 8 PS dimensions. However, the gains slowed down with increasing dimensionality. At the same time, the costs for training the model rise, especially with the number of channels that are remapped by individual NFs. One approach to this challenge is the use of more expressive models, e.g. based on autoregressive NFs possibly trained on weighted events [50]. Autoregressive transformations are more expressive than coupling layers [272], but either sampling or evaluating the probability density is more expensive. However, one can argue that at the rather moderate dimensionalities encountered in PS sampling, compared to e.g. image generation, this is of secondary importance. Since the evaluation of the ME should be even more expensive, one can afford to use a model that is somewhat more costly to evaluate. Other approaches to improving NF-based samplers are the use of local multichannel weights, learned by a NN, and mixtures of NFs. Furthermore, NFs could be a promising companion for newly developed GPU-based PS generators. These allow efficient training and sampling in a fully parallelized way. Furthermore, it was shown that good unweighting efficiencies can be achieved with only few channels [273]. This reduces the number of NF mappings necessary. In combination with ME generators on GPUS [274–276], a full event generation pipeline on GPUS would be possible.

As stated above, nested sampling shines at high-dimensional problems. In our study, we considered scattering problems with up to 5 gluons in the final state. This is representative of the PS complexity of current simulations for the LHC experiments. The precision demands increase, though, with the upcoming HL-LHC upgrade, such that even higher multiplicities can become necessary. Nested sampling may be an interesting choice for these, due to its scaling properties. An obvious extension of our approach would be the use of prior knowledge. This can be in the form of physics-motivated mappings or even machine-learned adaptive mappings, e.g. based on NFs [263]. A well-chosen prior distribution can significantly increase the efficiency of nested sampling. Similarly, a dynamical refinement in the form of dynamic nested sampling [277] can also improve the performance.

Concerning the surrogate unweighting procedure, one of the main improvements besides the extension to NLO would be a fast and accurate surrogate model for the partial amplitudes used in colour-sampling PS generators. The extension of the dipole-based factorization-aware model to coloured inputs is a promising idea that we plan to further explore. Lastly, also the surrogate unweighting method would benefit significantly from a parallel event generation pipeline on GPUS. The vectorized evaluation of the NNs could lead to a much faster unweighting, which could also be done on a previously generated set of weighted events.

Danksagung

Zuallererst möchte ich mich bei meinem Doktorvater Steffen Schumann bedanken. Vielen Dank für die enge und erstklassige Betreuung in wissenschaftlicher wie auch persönlicher Hinsicht, die Zugewandtheit, Offenheit und Unterstützung, das Vertrauen und die Hilfe bei Problemen jeglicher Art! Ich freue mich immer wieder aufs Neue darüber, in dieser Gruppe arbeiten zu dürfen. Des Weiteren bedanke ich mich bei Laura Covi und Marcus Müller für ihre Mitwirkung im Rahmen meines Betreuungskomitees. Zudem danke ich meiner Familie und meinen Freunden für ihre Unterstützung, sowie allen, die durch ihre Lehre zu meiner Ausbildung beigetragen haben und mir dadurch die Möglichkeit eröffnet haben, in der Wissenschaft tätig zu sein. Besonderer Dank gilt auch meiner Arbeitsgruppe, sowie allen anderen im Institut für Theoretische Physik (insbesondere auch allen, die über die Wissenschaft hinaus beschäftigt sind) für die freundliche und angenehme Atmosphäre, durch welche ich stets gerne den Weg zur Physik-Fakultät auf mich genommen habe. Ein ausdrückliches Dankeschön dabei auch an Max Knobbe für das Korrekturlesen dieser Arbeit. Außerordentlich wichtig ist es mir auch, mich bei meiner Freundin Linda und unserer Tochter Ronja für ihre Geduld zu bedanken. Ich bin mehr als dankbar dafür, dass ihr auch in anstrengenden Zeiten immer an meiner Seite standet und mein Leben mehr als alles andere bereichert.

Bibliography

- [1] G. Corcella, I. G. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson, M. H. Seymour and B. R. Webber, ‘HERWIG 6.5: An Event Generator for Hadron Emission Reactions With Interfering Gluons (Including Supersymmetric Processes)’, *Journal of High Energy Physics* **2001**, 010 (2001) [10.1088/1126-6708/2001/01/010](https://doi.org/10.1088/1126-6708/2001/01/010), [arXiv:hep-ph/0011363](https://arxiv.org/abs/hep-ph/0011363).
- [2] M. Bahr, S. Gieseke, M. A. Gigg, D. Grellscheid, K. Hamilton, O. Latunde-Dada, S. Platzer, P. Richardson, M. H. Seymour, A. Sherstnev, J. Tully and B. R. Webber, ‘Herwig++ Physics and Manual’, *The European Physical Journal C* **58**, 639 (2008) [10.1140/epjc/s10052-008-0798-9](https://doi.org/10.1140/epjc/s10052-008-0798-9), [arXiv:0803.0883](https://arxiv.org/abs/0803.0883).
- [3] J. Bellm et al., ‘Herwig 7.0 / Herwig++ 3.0 Release Note’, *The European Physical Journal C* **76**, 196 (2016) [10.1140/epjc/s10052-016-4018-8](https://doi.org/10.1140/epjc/s10052-016-4018-8), [arXiv:1512.01178](https://arxiv.org/abs/1512.01178).
- [4] T. Sjostrand, S. Mrenna and P. Skands, ‘PYTHIA 6.4 Physics and Manual’, *Journal of High Energy Physics* **2006**, 026 (2006) [10.1088/1126-6708/2006/05/026](https://doi.org/10.1088/1126-6708/2006/05/026), [arXiv:hep-ph/0603175](https://arxiv.org/abs/hep-ph/0603175).
- [5] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen and P. Z. Skands, ‘An Introduction to PYTHIA 8.2’, *Computer Physics Communications* **191**, 159 (2015) [10.1016/j.cpc.2015.01.024](https://doi.org/10.1016/j.cpc.2015.01.024), [arXiv:1410.3012](https://arxiv.org/abs/1410.3012).
- [6] T. Gleisberg, S. Hoeche, F. Krauss, M. Schoenherr, S. Schumann, F. Siegert and J. Winter, ‘Event Generation with SHERPA 1.1’, *Journal of High Energy Physics* **2009**, 007 (2009) [10.1088/1126-6708/2009/02/007](https://doi.org/10.1088/1126-6708/2009/02/007), [arXiv:0811.4622](https://arxiv.org/abs/0811.4622).
- [7] E. Bothmann, G. S. Chahal, S. Höche, J. Krause, F. Krauss, S. Kuttimalai, S. Liebschner, D. Napoletano, M. Schönherr, H. Schulz, S. Schumann and F. Siegert, ‘Event Generation with Sherpa 2.2’, *SciPost Physics* **7**, 034 (2019) [10.21468/SciPostPhys.7.3.034](https://doi.org/10.21468/SciPostPhys.7.3.034), [arXiv:1905.09127](https://arxiv.org/abs/1905.09127).
- [8] J. M. Campbell et al., *Event Generators for High-Energy Physics Experiments*, (2022) [arXiv:2203.11110](https://arxiv.org/abs/2203.11110), preprint.
- [9] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, ‘Exploring Phase Space with Neural Importance Sampling’, *SciPost Physics* **8**, 069 (2020) [10.21468/SciPostPhys.8.4.069](https://doi.org/10.21468/SciPostPhys.8.4.069), [arXiv:2001.05478](https://arxiv.org/abs/2001.05478).
- [10] D. Yallup, T. Janßen, S. Schumann and W. Handley, ‘Exploring Phase Space with Nested Sampling’, *The European Physical Journal C* **82**, 678 (2022) [10.1140/epjc/s10052-022-10632-2](https://doi.org/10.1140/epjc/s10052-022-10632-2), [arXiv:2205.02030](https://arxiv.org/abs/2205.02030).
- [11] K. Danziger, T. Janßen, S. Schumann and F. Siegert, ‘Accelerating Monte Carlo Event Generation – Rejection Sampling Using Neural Network Event-Weight Estimates’, *SciPost Physics* **12**, 164 (2022) [10.21468/SciPostPhys.12.5.164](https://doi.org/10.21468/SciPostPhys.12.5.164), [arXiv:2109.11964](https://arxiv.org/abs/2109.11964).
- [12] T. Janßen, D. Maître, S. Schumann, F. Siegert and H. Truong, *Unweighting Multijet Event Generation Using Factorisation-Aware Neural Networks*, (2023) [arXiv:2301.13562](https://arxiv.org/abs/2301.13562), preprint.

- [13] The ATLAS Collaboration, ‘Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC’, *Physics Letters B* **716**, 1 (2012) [10.1016/j.physletb.2012.08.020](https://doi.org/10.1016/j.physletb.2012.08.020), arXiv:1207.7214.
- [14] The CMS Collaboration, ‘Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC’, *Physics Letters B* **716**, 30 (2012) [10.1016/j.physletb.2012.08.021](https://doi.org/10.1016/j.physletb.2012.08.021), arXiv:1207.7235.
- [15] O. Brüning and L. Rossi, ‘Chapter 1: High-Luminosity Large Hadron Collider’, in *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report*, Vol. 10, edited by I. Béjar Alonso, O. Brüning, P. Fessia, L. Rossi, L. Tavian and M. Zerlauth, CERN Yellow Rep.Monogr. (CERN, Geneva, 2020), pages 1–16.
- [16] LHC commissioning, *Nominal HL-LHC Luminosity - Ions Stopped after LS4*, <https://web.archive.org/web/20220706170326/https://lhc-commissioning.web.cern.ch/schedule/images/LHC-nominal-lumi-projection.png> (visited on 28/04/2023).
- [17] The ATLAS Collaboration, *ATLAS Software and Computing HL-LHC Roadmap*, CERN-LHCC-2022-005 ; LHCC-G-182 (CERN, Geneva, 2022), <https://cds.cern.ch/record/2802918>.
- [18] 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’, *The European Physical Journal C* **82**, 1128 (2022) [10.1140/epjc/s10052-022-11087-1](https://doi.org/10.1140/epjc/s10052-022-11087-1), arXiv:2209.00843.
- [19] The ATLAS Collaboration, ‘Modelling and Computational Improvements to the Simulation of Single Vector-Boson plus Jet Processes for the ATLAS Experiment’, *Journal of High Energy Physics* **2022**, 89 (2022) [10.1007/JHEP08\(2022\)089](https://doi.org/10.1007/JHEP08(2022)089), arXiv:2112.09588.
- [20] S. Hoeche, F. Krauss, M. Schonherr and F. Siegert, ‘QCD Matrix Elements + Parton Showers: The NLO Case’, *Journal of High Energy Physics* **2013**, 27 (2013) [10.1007/JHEP04\(2013\)027](https://doi.org/10.1007/JHEP04(2013)027), arXiv:1207.5030.
- [21] T. Gehrmann, S. Hoeche, F. Krauss, M. Schonherr and F. Siegert, ‘NLO QCD Matrix Elements + Parton Showers in $E+e^- \rightarrow$ Hadrons’, *Journal of High Energy Physics* **2013**, 144 (2013) [10.1007/JHEP01\(2013\)144](https://doi.org/10.1007/JHEP01(2013)144), arXiv:1207.5031.
- [22] S. Höche, S. Prestel and H. Schulz, ‘Simulation of Vector Boson plus Many Jet Final States at the High Luminosity LHC’, *Physical Review D* **100**, 014024 (2019) [10.1103/PhysRevD.100.014024](https://doi.org/10.1103/PhysRevD.100.014024), arXiv:1905.05120.
- [23] The ATLAS Collaboration, *Standard Model Summary Plots February 2022*, ATL-PHYS-PUB-2022-009 (CERN, Geneva, 2022), <https://cds.cern.ch/record/2804061>.
- [24] A. Chowdhery et al., *PaLM: Scaling Language Modeling with Pathways*, (2022) [10.48550/arXiv.2204.02311](https://arxiv.org/abs/2204.02311), arXiv:2204.02311, preprint.
- [25] K. Hornik, M. Stinchcombe and H. White, ‘Multilayer Feedforward Networks Are Universal Approximators’, *Neural Networks* **2**, 359 (1989) [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- [26] A. Pinkus, ‘Approximation Theory of the MLP Model in Neural Networks’, *Acta Numerica* **8**, 143 (1999) [10.1017/S0962492900002919](https://doi.org/10.1017/S0962492900002919).
- [27] Z. Lu, H. Pu, F. Wang, Z. Hu and L. Wang, ‘The Expressive Power of Neural Networks: A View from the Width’, in *Advances in Neural Information Processing Systems*, Vol. 30 (2017).

- [28] J. Skilling, ‘Nested Sampling’, *AIP Conference Proceedings* **735**, 395 (2004) [10.1063/1.1835238](#).
- [29] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, (2017) [10.48550/arXiv.1707.00028](#), arXiv:1707.00028, preprint.
- [30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, ‘Generative Adversarial Nets’, in *Advances in Neural Information Processing Systems*, Vol. 27 (2014).
- [31] G. Peter Lepage, ‘A New Algorithm for Adaptive Multidimensional Integration’, *Journal of Computational Physics* **27**, 192 (1978) [10.1016/0021-9991\(78\)90004-9](#).
- [32] G. P. Lepage, *VEGAS - an Adaptive Multi-Dimensional Integration Program*, CLNS-447 (Cornell Univ. Lab. Nucl. Stud., Ithaca, NY, 1980), <https://cds.cern.ch/record/123074>.
- [33] S. Jadach, ‘Foam: A General-Purpose Cellular Monte Carlo Event Generator’, *Computer Physics Communications* **152**, 55 (2003) [10.1016/S0010-4655\(02\)00755-5](#), arXiv:physics/0203033.
- [34] M. D. Klimek and M. Perelstein, ‘Neural Network-Based Approach to Phase Space Integration’, *SciPost Physics* **9**, 053 (2020) [10.21468/SciPostPhys.9.4.053](#), arXiv:1810.11509.
- [35] I.-K. Chen, M. D. Klimek and M. Perelstein, ‘Improved Neural Network Monte Carlo Simulation’, *SciPost Physics* **10**, 023 (2021) [10.21468/SciPostPhys.10.1.023](#), arXiv:2009.07819.
- [36] E. G. Tabak and E. Vanden-Eijnden, ‘Density Estimation by Dual Ascent of the Log-Likelihood’, *Communications in Mathematical Sciences* **8**, 217 (2010) [10.4310/CMS.2010.v8.n1.a11](#).
- [37] E. G. Tabak and C. V. Turner, ‘A Family of Nonparametric Density Estimation Algorithms’, *Communications on Pure and Applied Mathematics* **66**, 145 (2013) [10.1002/cpa.21423](#).
- [38] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, (2015) [10.48550/arXiv.1410.8516](#), arXiv:1410.8516, preprint.
- [39] M. S. Albergó, G. Kanwar and P. E. Shanahan, ‘Flow-Based Generative Models for Markov Chain Monte Carlo in Lattice Field Theory’, *Physical Review D* **100**, 034515 (2019) [10.1103/PhysRevD.100.034515](#), arXiv:1904.12072.
- [40] G. Kanwar, M. S. Albergó, D. Boyda, K. Cranmer, D. C. Hackett, S. Racanière, D. J. Rezende and P. E. Shanahan, ‘Equivariant Flow-Based Sampling for Lattice Gauge Theory’, *Physical Review Letters* **125**, 121601 (2020) [10.1103/PhysRevLett.125.121601](#), arXiv:2003.06413.
- [41] K. A. Nicoli, C. J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel, S. Nakajima and P. Stornati, ‘Estimation of Thermodynamic Observables in Lattice Field Theories with Deep Generative Models’, *Physical Review Letters* **126**, 032001 (2021) [10.1103/PhysRevLett.126.032001](#).
- [42] D. C. Hackett, C.-C. Hsieh, M. S. Albergó, D. Boyda, J.-W. Chen, K.-F. Chen, K. Cranmer, G. Kanwar and P. E. Shanahan, *Flow-Based Sampling for Multimodal Distributions in Lattice Field Theory*, (2021) [10.48550/arXiv.2107.00734](#), arXiv:2107.00734, preprint.
- [43] M. S. Albergó, G. Kanwar, S. Racanière, D. J. Rezende, J. M. Urban, D. Boyda, K. Cranmer, D. C. Hackett and P. E. Shanahan, ‘Flow-Based Sampling for Fermionic Lattice Field Theories’, *Physical Review D* **104**, 114507 (2021) [10.1103/PhysRevD.104.114507](#).

- [44] A. Rouhiainen, U. Giri and M. Münchmeyer, *Normalizing Flows for Random Fields in Cosmology*, (2021) [10.48550/arXiv.2105.12024](https://arxiv.org/abs/2105.12024), [arXiv:2105.12024](https://arxiv.org/abs/2105.12024), preprint.
- [45] B. Dai and U. Seljak, ‘Translation and Rotation Equivariant Normalizing Flow (TRENFlow) for Optimal Cosmological Analysis’, *Monthly Notices of the Royal Astronomical Society* **516**, 2363 (2022) [10.1093/mnras/stac2010](https://arxiv.org/abs/2202.05282), [arXiv:2202.05282](https://arxiv.org/abs/2202.05282).
- [46] A. Rouhiainen and M. Münchmeyer, *De-Noising Non-Gaussian Fields in Cosmology with Normalizing Flows*, (2022) [10.48550/arXiv.2211.15161](https://arxiv.org/abs/2211.15161), [arXiv:2211.15161](https://arxiv.org/abs/2211.15161), preprint.
- [47] C. Gao, S. Hoeche, J. Isaacson, C. Krause and H. Schulz, ‘Event Generation with Normalizing Flows’, *Physical Review D* **101**, 076002 (2020) [10.1103/PhysRevD.101.076002](https://arxiv.org/abs/2001.10028), [arXiv:2001.10028](https://arxiv.org/abs/2001.10028).
- [48] G. Papamakarios, T. Pavlakou and I. Murray, ‘Masked Autoregressive Flow for Density Estimation’, in *Advances in Neural Information Processing Systems*, Vol. 30 (2017).
- [49] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling, ‘Improved Variational Inference with Inverse Autoregressive Flow’, in *Advances in Neural Information Processing Systems*, Vol. 29 (2016).
- [50] B. Stienen and R. Verheyen, ‘Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows’, *SciPost Physics* **10**, 038 (2021) [10.21468/SciPostPhys.10.2.038](https://arxiv.org/abs/2011.13445), [arXiv:2011.13445](https://arxiv.org/abs/2011.13445).
- [51] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, ‘Neural Spline Flows’, in *Advances in Neural Information Processing Systems*, Vol. 32 (2019).
- [52] S. Pina-Otey, F. Sánchez, T. Lux and V. Gaitan, ‘Exhaustive Neural Importance Sampling Applied to Monte Carlo Event Generation’, *Physical Review D* **102**, 013003 (2020) [10.1103/PhysRevD.102.013003](https://arxiv.org/abs/2005.12719), [arXiv:2005.12719](https://arxiv.org/abs/2005.12719).
- [53] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer and T. Plehn, ‘MadNIS – Neural Multi-Channel Importance Sampling’, 2022, [arXiv:2212.06172](https://arxiv.org/abs/2212.06172).
- [54] A. Butter, T. Plehn and R. Winterhalder, ‘How to GAN LHC Events’, *SciPost Physics* **7**, 075 (2019) [10.21468/SciPostPhys.7.6.075](https://arxiv.org/abs/1907.03764), [arXiv:1907.03764](https://arxiv.org/abs/1907.03764).
- [55] R. Di Sipio, M. F. Giannelli, S. K. Haghighat and S. Palazzo, ‘DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC’, *Journal of High Energy Physics* **2019**, 110 (2019) [10.1007/JHEP08\(2019\)110](https://arxiv.org/abs/1903.02433), [arXiv:1903.02433](https://arxiv.org/abs/1903.02433).
- [56] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, *LHC Analysis-Specific Datasets with Generative Adversarial Networks*, (2019) [10.48550/arXiv.1901.05282](https://arxiv.org/abs/1901.05282), [arXiv:1901.05282](https://arxiv.org/abs/1901.05282), preprint.
- [57] L. Velasco, E. McClellan, N. Sato, P. Ambrozewicz, T. Liu, W. Melnitchouk, M. P. Kuchera, Y. Alanazi and Y. Li, ‘cFAT-GAN: Conditional Simulation of Electron-Proton Scattering Events with Varying Beam Energies by a Feature Augmented and Transformed Generative Adversarial Network’, in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2020), pages 372–375, [10.1109/ICMLA51294.2020.00066](https://arxiv.org/abs/2020.00066).
- [58] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, P. Ambrozewicz, F. Hauenstein, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li, ‘Simulation of Electron-Proton Scattering Events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)’, in *Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 3 (2021), pages 2126–2132, [10.24963/ijcai.2021/293](https://arxiv.org/abs/2021/293).

- [59] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri and R. Verheyen, ‘Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer’, *Nature Communications* **12**, 2985 (2021) [10.1038/s41467-021-22616-z](https://doi.org/10.1038/s41467-021-22616-z).
- [60] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, ‘Generative Networks for Precision Enthusiasts’, *SciPost Physics* **14**, 078 (2023) [10.21468/SciPostPhys.14.4.078](https://doi.org/10.21468/SciPostPhys.14.4.078), arXiv:2110.13632.
- [61] A. Butter, N. Huetsch, S. P. Schweitzer, T. Plehn, P. Sorrenson and J. Spinner, *Jet Diffusion versus JetGPT – Modern Networks for the LHC*, (2023) [10.48550/arXiv.2305.10475](https://doi.org/10.48550/arXiv.2305.10475), arXiv:2305.10475, preprint.
- [62] K. T. Matchev, A. Roman and P. Shyamsundar, ‘Uncertainties Associated with GAN-generated Datasets in High Energy Physics’, *SciPost Physics* **12**, 104 (2022) [10.21468/SciPostPhys.12.3.104](https://doi.org/10.21468/SciPostPhys.12.3.104), arXiv:2002.06307.
- [63] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, ‘GANplifying Event Samples’, *SciPost Physics* **10**, 139 (2021) [10.21468/SciPostPhys.10.6.139](https://doi.org/10.21468/SciPostPhys.10.6.139), arXiv:2008.06545.
- [64] D. Maître and H. Truong, ‘A Factorisation-Aware Matrix Element Emulator’, *Journal of High Energy Physics* **2021**, 66 (2021) [10.1007/JHEP11\(2021\)066](https://doi.org/10.1007/JHEP11(2021)066), arXiv:2107.06625.
- [65] S. Catani and M. H. Seymour, ‘A General Algorithm for Calculating Jet Cross Sections in NLO QCD’, *Nuclear Physics B* **485**, 291 (1997) [10.1016/S0550-3213\(96\)00589-5](https://doi.org/10.1016/S0550-3213(96)00589-5), arXiv:hep-ph/9605323.
- [66] L. Mason, J. Baxter, P. Bartlett and M. Frean, ‘Boosting Algorithms as Gradient Descent’, in *Advances in Neural Information Processing Systems*, Vol. 12 (1999).
- [67] L. Mason, J. Baxter, P. Bartlett and M. Frean, ‘Boosting Algorithms as Gradient Descent in Function Space’, (1999).
- [68] J. H. Friedman, ‘Greedy Function Approximation: A Gradient Boosting Machine.’, *The Annals of Statistics* **29**, 1189 (2001) [10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451).
- [69] J. Friedman, ‘Stochastic Gradient Boosting’, *Computational Statistics & Data Analysis* **38**, 367 (2002) [10.1016/S0167-9473\(01\)00065-2](https://doi.org/10.1016/S0167-9473(01)00065-2).
- [70] T. Chen and C. Guestrin, ‘XGBoost: A Scalable Tree Boosting System’, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pages 785–794, [10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785), arXiv:1603.02754.
- [71] F. Bishara and M. Montull, ‘Machine Learning Amplitudes for Faster Event Generation’, *Physical Review D* **107**, L071901 (2023) [10.1103/PhysRevD.107.L071901](https://doi.org/10.1103/PhysRevD.107.L071901).
- [72] S. Badger and J. Bullock, ‘Using Neural Networks for Efficient Evaluation of High Multiplicity Scattering Amplitudes’, *Journal of High Energy Physics* **2020**, 114 (2020) [10.1007/JHEP06\(2020\)114](https://doi.org/10.1007/JHEP06(2020)114).
- [73] S. Frixione, Z. Kunszt and A. Signer, ‘Three-Jet Cross Sections to next-to-Leading Order’, *Nuclear Physics B* **467**, 399 (1996) [10.1016/0550-3213\(96\)00110-1](https://doi.org/10.1016/0550-3213(96)00110-1), arXiv:hep-ph/9512328.
- [74] R. Frederix, S. Frixione, F. Maltoni and T. Stelzer, ‘Automation of Next-to-Leading Order Computations in QCD: The FKS Subtraction’, *Journal of High Energy Physics* **2009**, 003 (2009) [10.1088/1126-6708/2009/10/003](https://doi.org/10.1088/1126-6708/2009/10/003), arXiv:0908.4272.
- [75] J. Aylett-Bullock, S. Badger and R. Moodie, ‘Optimising Simulations for Diphoton Production at Hadron Colliders Using Amplitude Neural Networks’, *Journal of High Energy Physics* **2021**, 66 (2021) [10.1007/JHEP08\(2021\)066](https://doi.org/10.1007/JHEP08(2021)066).

- [76] S. Badger, A. Butter, M. Luchmann, S. Pitz and T. Plehn, ‘Loop Amplitudes from Precision Networks’, *SciPost Physics Core* **6**, 034 (2023) [10.21468/SciPostPhysCore.6.2.034](https://doi.org/10.21468/SciPostPhysCore.6.2.034).
- [77] D. Maître and H. Truong, ‘One-Loop Matrix Element Emulation with Factorisation Awareness’, *Journal of High Energy Physics* **2023**, 159 (2023) [10.1007/JHEP05\(2023\)159](https://doi.org/10.1007/JHEP05(2023)159), [arXiv:2302.04005](https://arxiv.org/abs/2302.04005).
- [78] A. G.-D. Ridder, T. Gehrmann and E. W. N. Glover, ‘Antenna Subtraction at NNLO’, *Journal of High Energy Physics* **2005**, 056 (2005) [10.1088/1126-6708/2005/09/056](https://doi.org/10.1088/1126-6708/2005/09/056), [arXiv:hep-ph/0505111](https://arxiv.org/abs/hep-ph/0505111).
- [79] F. Bishara, A. Paul and J. Dy, *High-Precision Regressors for Particle Physics*, (2023) [10.48550/arXiv.2302.00753](https://doi.org/10.48550/arXiv.2302.00753), [arXiv:2302.00753](https://arxiv.org/abs/2302.00753), preprint.
- [80] R. K. Srivastava, K. Greff and J. Schmidhuber, *Highway Networks*, (2015) [10.48550/arXiv.1505.00387](https://doi.org/10.48550/arXiv.1505.00387), [arXiv:1505.00387](https://arxiv.org/abs/1505.00387), preprint.
- [81] B. Nachman and C. Shimmin, *AI Safety for High Energy Physics*, (2019) [10.48550/arXiv.1910.08606](https://doi.org/10.48550/arXiv.1910.08606), [arXiv:1910.08606](https://arxiv.org/abs/1910.08606), preprint.
- [82] B. Nachman, ‘A Guide for Deploying Deep Learning in LHC Searches: How to Achieve Optimality and Account for Uncertainty’, *SciPost Physics* **8**, 090 (2020) [10.21468/SciPostPhys.8.6.090](https://doi.org/10.21468/SciPostPhys.8.6.090).
- [83] T. Y. Chen, B. Dey, A. Ghosh, M. Kagan, B. Nord and N. Ramachandra, *Interpretable Uncertainty Quantification in AI for HEP*, FERMILAB-FN-1179-SCD (2022), [10.2172/1886020](https://doi.org/10.2172/1886020), [arXiv:2208.03284](https://arxiv.org/abs/2208.03284).
- [84] L. Anderlini, C. Chimpoesh, N. Kazeev and A. Shishigina, ‘Generative Models Uncertainty Estimation’, *Journal of Physics: Conference Series* **2438**, 012088 (2023) [10.1088/1742-6596/2438/1/012088](https://doi.org/10.1088/1742-6596/2438/1/012088), [arXiv:2210.09767](https://arxiv.org/abs/2210.09767).
- [85] L. Hansen and P. Salamon, ‘Neural Network Ensembles’, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, 993 (1990) [10.1109/34.58871](https://doi.org/10.1109/34.58871).
- [86] O. Sagi and L. Rokach, ‘Ensemble Learning: A Survey’, *WIREs Data Mining and Knowledge Discovery* **8**, e1249 (2018) [10.1002/widm.1249](https://doi.org/10.1002/widm.1249).
- [87] B. Kronheim, M. Kuchera, H. Prosper and A. Karbo, ‘Bayesian Neural Networks for Fast SUSY Predictions’, *Physics Letters B* **813**, 136041 (2021) [10.1016/j.physletb.2020.136041](https://doi.org/10.1016/j.physletb.2020.136041), [arXiv:2007.04506](https://arxiv.org/abs/2007.04506).
- [88] M. Bellagente, M. Haußmann, M. Luchmann and T. Plehn, ‘Understanding Event-Generation Networks via Uncertainties’, *SciPost Physics* **13**, 003 (2022) [10.21468/SciPostPhys.13.1.003](https://doi.org/10.21468/SciPostPhys.13.1.003), [arXiv:2104.04543](https://arxiv.org/abs/2104.04543).
- [89] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman and J. Thaler, ‘OmniFold: A Method to Simultaneously Unfold All Observables’, *Physical Review Letters* **124**, 182001 (2020) [10.1103/PhysRevLett.124.182001](https://doi.org/10.1103/PhysRevLett.124.182001), [arXiv:1911.09107](https://arxiv.org/abs/1911.09107).
- [90] A. Andreassen and B. Nachman, ‘Neural Networks for Full Phase-space Reweighting and Parameter Tuning’, *Physical Review D* **101**, 091901 (2020) [10.1103/PhysRevD.101.091901](https://doi.org/10.1103/PhysRevD.101.091901), [arXiv:1907.08209](https://arxiv.org/abs/1907.08209).
- [91] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih, ‘DCTRGAN: Improving the Precision of Generative Models with Reweighting’, *Journal of Instrumentation* **15**, P11004 (2020) [10.1088/1748-0221/15/11/P11004](https://doi.org/10.1088/1748-0221/15/11/P11004), [arXiv:2009.03796](https://arxiv.org/abs/2009.03796).
- [92] R. Winterhalder, M. Bellagente and B. Nachman, *Latent Space Refinement for Deep Generative Models*, (2021) [10.48550/arXiv.2106.00792](https://doi.org/10.48550/arXiv.2106.00792), [arXiv:2106.00792](https://arxiv.org/abs/2106.00792), preprint.

- [93] B. Nachman and R. Winterhalder, *ELSA – Enhanced Latent Spaces for Improved Collider Simulations*, (2023) [10.48550/arXiv.2305.07696](https://arxiv.org/abs/2305.07696), [arXiv:2305.07696](https://arxiv.org/abs/2305.07696), preprint.
- [94] E. Dupont, A. Doucet and Y. W. Teh, ‘Augmented Neural ODEs’, in *Advances in Neural Information Processing Systems*, Vol. 32 (2019).
- [95] C.-W. Huang, L. Dinh and A. Courville, *Augmented Normalizing Flows: Bridging the Gap Between Generative Flows and Latent Variable Models*, (2020) [10.48550/arXiv.2002.07101](https://arxiv.org/abs/2002.07101), [arXiv:2002.07101](https://arxiv.org/abs/2002.07101), preprint.
- [96] J. Chen, C. Lu, B. Chenli, J. Zhu and T. Tian, ‘VFlow: More Expressive Generative Flows with Variational Data Augmentation’, in *Proceedings of the 37th International Conference on Machine Learning* (2020), pages 1660–1669.
- [97] D. Nielsen, P. Jainsi, E. Hooeboom, O. Winther and M. Welling, ‘SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows’, in *Advances in Neural Information Processing Systems*, Vol. 33 (2020), pages 12685–12696.
- [98] L. de Oliveira, M. Paganini and B. Nachman, ‘Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters’, *Journal of Physics: Conference Series* **1085**, 042017 (2018) [10.1088/1742-6596/1085/4/042017](https://arxiv.org/abs/1711.08813), [arXiv:1711.08813](https://arxiv.org/abs/1711.08813).
- [99] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, ‘Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed’, *Computing and Software for Big Science* **5**, 13 (2021) [10.1007/s41781-021-00056-0](https://arxiv.org/abs/2005.05334), [arXiv:2005.05334](https://arxiv.org/abs/2005.05334).
- [100] C. Krause, I. Pang and D. Shih, *CaloFlow for CaloChallenge Dataset 1*, (2022) [arXiv:2210.14245](https://arxiv.org/abs/2210.14245), preprint.
- [101] V. Mikuni and B. Nachman, ‘Score-Based Generative Models for Calorimeter Shower Simulation’, *Physical Review D* **106**, 092009 (2022) [10.1103/PhysRevD.106.092009](https://arxiv.org/abs/2206.11898), [arXiv:2206.11898](https://arxiv.org/abs/2206.11898).
- [102] The ATLAS Collaboration, ‘AtlFast3: The next Generation of Fast Simulation in ATLAS’, *Computing and Software for Big Science* **6**, 7 (2022) [10.1007/s41781-021-00079-7](https://arxiv.org/abs/2109.02551), [arXiv:2109.02551](https://arxiv.org/abs/2109.02551).
- [103] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *CaloClouds: Fast Geometry-Independent Highly-Granular Calorimeter Simulation*, (2023) [arXiv:2305.04847](https://arxiv.org/abs/2305.04847), preprint.
- [104] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh and D. Shih, *L2LFlows: Generating High-Fidelity 3D Calorimeter Images*, (2023) [arXiv:2302.11594](https://arxiv.org/abs/2302.11594), preprint.
- [105] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown and L. Rustige, *New Angles on Fast Calorimeter Shower Simulation*, (2023) [arXiv:2303.18150](https://arxiv.org/abs/2303.18150), preprint.
- [106] H. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn and T. Kuhr, *Ultra-High-Resolution Detector Simulation with Intra-Event Aware GAN and Self-Supervised Relational Reasoning*, (2023) [arXiv:2303.08046](https://arxiv.org/abs/2303.08046), preprint.
- [107] C. Krause and D. Shih, *CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows*, (2023) [arXiv:2110.11377](https://arxiv.org/abs/2110.11377), preprint.
- [108] C. Krause and D. Shih, *CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows*, (2023) [arXiv:2106.05285](https://arxiv.org/abs/2106.05285), preprint.

- [109] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, ‘Generating and Refining Particle Detector Simulations Using the Wasserstein Distance in Adversarial Networks’, *Computing and Software for Big Science* **2**, 4 (2018) 10.1007/s41781-018-0008-x.
- [110] M. Paganini, L. de Oliveira and B. Nachman, ‘Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multi-Layer Calorimeters’, *Physical Review Letters* **120**, 042003 (2018) 10.1103/PhysRevLett.120.042003, arXiv:1705.02355.
- [111] M. Paganini, L. de Oliveira and B. Nachman, ‘CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks’, *Physical Review D* **97**, 014021 (2018) 10.1103/PhysRevD.97.014021, arXiv:1712.10321.
- [112] The ATLAS Collaboration, *Deep Generative Models for Fast Shower Simulation in ATLAS*, Geneva, 2018, <https://cds.cern.ch/record/2630433>.
- [113] M. Erdmann, J. Glombitza and T. Quast, ‘Precise Simulation of Electromagnetic Calorimeter Showers Using a Wasserstein Generative Adversarial Network’, *Computing and Software for Big Science* **3**, 4 (2019) 10.1007/s41781-018-0019-7, arXiv:1807.01954.
- [114] D. Belayneh et al., ‘Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics’, *The European Physical Journal C* **80**, 688 (2020) 10.1140/epjc/s10052-020-8251-9, arXiv:1912.06794.
- [115] The ATLAS Collaboration, *Fast Simulation of the ATLAS Calorimeter System with Generative Adversarial Networks*, Geneva, 2020, <https://cds.cern.ch/record/2746032>.
- [116] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, ‘Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network’, *EPJ Web of Conferences* **251**, 03003 (2021) 10.1051/epjconf/202125103003, arXiv:2102.12491.
- [117] A. Butter, T. Plehn and R. Winterhalder, ‘How to GAN Event Subtraction’, *SciPost Physics Core* **3**, 009 (2020) 10.21468/SciPostPhysCore.3.2.009, arXiv:1912.08824.
- [118] M. Backes, A. Butter, T. Plehn and R. Winterhalder, ‘How to GAN Event Unweighting’, *SciPost Physics* **10**, 089 (2021) 10.21468/SciPostPhys.10.4.089, arXiv:2012.07873.
- [119] P. Baldi, L. Blecher, A. Butter, J. Collado, J. N. Howard, F. Keilbach, T. Plehn, G. Kasieczka and D. Whiteson, ‘How to GAN Higher Jet Resolution’, *SciPost Physics* **13**, 064 (2022) 10.21468/SciPostPhys.13.3.064.
- [120] F. A. Di Bello, S. Ganguly, E. Gross, M. Kado, M. Pitt, L. Santi and J. Shlomi, ‘Towards a Computer Vision Particle Flow’, *The European Physical Journal C* **81**, 107 (2021) 10.1140/epjc/s10052-021-08897-0, arXiv:2003.08863.
- [121] A. Butter, T. Heimel, T. Martini, S. Peitzsch and T. Plehn, *Two Invertible Networks for the Matrix Element Method*, (2023) 10.48550/arXiv.2210.00019, arXiv:2210.00019, preprint.
- [122] L. de Oliveira, M. Paganini and B. Nachman, ‘Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis’, *Computing and Software for Big Science* **1**, 4 (2017) 10.1007/s41781-017-0004-6, arXiv:1701.05927.
- [123] A. Andreassen, I. Feige, C. Frye and M. D. Schwartz, ‘JUNIPR: A Framework for Unsupervised Machine Learning in Particle Physics’, *The European Physical Journal C* **79**, 102 (2019) 10.1140/epjc/s10052-019-6607-9, arXiv:1804.09720.

- [124] E. Bothmann and L. Del Debbio, ‘Reweighting a Parton Shower Using a Neural Network: The Final-State Case’, *Journal of High Energy Physics* **2019**, 33 (2019) 10 . 1007 / JHEP01(2019)033, arXiv:1808 . 07802.
- [125] K. Dohi, *Variational Autoencoders for Jet Simulation*, (2020) arXiv:2009 . 04842, preprint.
- [126] E. Buhmann, G. Kasieczka and J. Thaler, *EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets*, (2023) 10 . 48550 / arXiv . 2301 . 08128, arXiv:2301 . 08128, preprint.
- [127] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch and T. Golling, *PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics*, (2023) arXiv:2303 . 05376, preprint.
- [128] V. Mikuni, B. Nachman and M. Pettee, *Fast Point Cloud Generation with Diffusion Models in High Energy Physics*, (2023) arXiv:2304 . 01266, preprint.
- [129] J. Brehmer, G. Louppe, J. Pavez and K. Cranmer, ‘Mining Gold from Implicit Models to Improve Likelihood-Free Inference’, *Proceedings of the National Academy of Sciences* **117**, 5242 (2020) 10 . 1073 / pnas . 1915980117, arXiv:1805 . 12244.
- [130] J. Brehmer, F. Kling, I. Espejo and K. Cranmer, ‘MadMiner: Machine Learning-Based Inference for Particle Physics’, *Computing and Software for Big Science* **4**, 3 (2020) 10 . 1007 / s41781-020-0035-2.
- [131] S. Bieringer, A. Butter, T. Heimel, S. Höche, U. Köthe, T. Plehn and S. T. Radev, ‘Measuring QCD Splittings with Invertible Networks’, *SciPost Physics* **10**, 126 (2021) 10 . 21468 / SciPostPhys . 10 . 6 . 126, arXiv:2012 . 09873.
- [132] T. Bister, M. Erdmann, U. Köthe and J. Schulte, ‘Inference of Cosmic-Ray Source Properties by Conditional Invertible Neural Networks’, *The European Physical Journal C* **82**, 171 (2022) 10 . 1140 / epjc / s10052-022-10138-x, arXiv:2110 . 09493.
- [133] D. Maître and R. Santos-Mateos, ‘Multi-Variable Integration with a Neural Network’, *Journal of High Energy Physics* **2023**, 221 (2023) 10 . 1007 / JHEP03(2023)221, arXiv:2211 . 02834.
- [134] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao and T. Wongjirad, ‘Machine Learning at the Energy and Intensity Frontiers of Particle Physics’, *Nature* **560**, 41 (2018) 10 . 1038 / s41586-018-0361-2.
- [135] K. Albertsson et al., ‘Machine Learning in High Energy Physics Community White Paper’, *Journal of Physics: Conference Series* **1085**, 022008 (2018) 10 . 1088 / 1742-6596 / 1085 / 2 / 022008.
- [136] D. Bourilkov, ‘Machine and Deep Learning Applications in Particle Physics’, *International Journal of Modern Physics A* **34**, 1930019 (2019) 10 . 1142 / S0217751X19300199, arXiv:1912 . 08245.
- [137] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman and D. Shih, ‘Machine Learning in the Search for New Fundamental Physics’, *Nature Reviews Physics* **4**, 399 (2022) 10 . 1038 / s42254-022-00455-1.
- [138] A. Butter et al., ‘Machine Learning and LHC Event Generation’, *SciPost Physics* **14**, 079 (2023) 10 . 21468 / SciPostPhys . 14 . 4 . 079, arXiv:2203 . 07460.
- [139] F. Feroz and M. P. Hobson, ‘Multimodal Nested Sampling: An Efficient and Robust Alternative to MCMC Methods for Astronomical Data Analysis’, *Monthly Notices of the Royal Astronomical Society* **384**, 449 (2008) 10 . 1111 / j . 1365-2966 . 2007 . 12353 . x, arXiv:0704 . 3704.

- [140] M. C. March, R. Trotta, P. Berkes, G. D. Starkman and P. M. Vaudrevange, ‘Improved Constraints on Cosmological Parameters from Type Ia Supernova Data’, *Monthly Notices of the Royal Astronomical Society* **418**, 2308 (2011) [10.1111/j.1365-2966.2011.19584.x](#).
- [141] W. J. Handley, M. P. Hobson and A. N. Lasenby, ‘PolyChord: Nested Sampling for Cosmology’, *Monthly Notices of the Royal Astronomical Society: Letters* **450**, L61 (2015) [10.1093/mnrasl/slv047](#), arXiv:1502.01856.
- [142] Planck Collaboration, ‘Planck 2015 Results. XX. Constraints on Inflation’, *Astronomy & Astrophysics* **594**, A20 (2016) [10.1051/0004-6361/201525898](#), arXiv:1502.02114.
- [143] W. Handley and P. Lemos, ‘Quantifying Tensions in Cosmological Parameters: Interpreting the DES Evidence Ratio’, *Physical Review D* **100**, 043504 (2019) [10.1103/PhysRevD.100.043504](#), arXiv:1902.04029.
- [144] The GAMBIT Cosmology Workgroup, ‘CosmoBit: A GAMBIT Module for Computing Cosmological Observables and Likelihoods’, *Journal of Cosmology and Astroparticle Physics* **2021**, 022 (2021) [10.1088/1475-7516/2021/02/022](#), arXiv:2009.03286.
- [145] J. Veitch and A. Vecchio, ‘A Bayesian Approach to the Follow-up of Candidate Gravitational Wave Signals’, *Physical Review D* **78**, 022001 (2008) [10.1103/PhysRevD.78.022001](#), arXiv:0801.4313.
- [146] R. Smith, G. Ashton, A. Vajpeyi and C. Talbot, ‘Massively Parallel Bayesian Inference for Transient Gravitational-Wave Astronomy’, *Monthly Notices of the Royal Astronomical Society* **498**, 4492 (2020) [10.1093/mnras/staa2483](#), arXiv:1909.11873.
- [147] The LIGO Scientific Collaboration and the Virgo Collaboration, ‘Model Comparison from LIGO-Virgo Data on GW170817’s Binary Components and Consequences for the Merger Remnant’, *Classical and Quantum Gravity* **37**, 045006 (2020) [10.1088/1361-6382/ab5f7c](#), arXiv:1908.01012.
- [148] R. Trotta, F. Feroz, M. P. Hobson, L. Roszkowski and R. R. de Austri, ‘The Impact of Priors and Observables on Parameter Inferences in the Constrained MSSM’, *Journal of High Energy Physics* **2008**, 024 (2008) [10.1088/1126-6708/2008/12/024](#), arXiv:0809.3792.
- [149] F. Feroz, K. Cranmer, M. Hobson, R. R. de Austri and R. Trotta, ‘Challenges of Profile Likelihood Evaluation in Multi-Dimensional SUSY Scans’, *Journal of High Energy Physics* **2011**, 42 (2011) [10.1007/JHEP06\(2011\)042](#), arXiv:1101.3296.
- [150] The GAMBIT Scanner Workgroup, ‘Comparison of Statistical Sampling Methods with ScannerBit, the GAMBIT Scanning Module’, *The European Physical Journal C* **77**, 761 (2017) [10.1140/epjc/s10052-017-5274-y](#), arXiv:1705.07959.
- [151] The DarkMachines High Dimensional Sampling Group, ‘A Comparison of Optimisation Algorithms for High-Dimensional Particle and Astrophysics Applications’, *Journal of High Energy Physics* **2021**, 108 (2021) [10.1007/JHEP05\(2021\)108](#), arXiv:2101.04525.
- [152] A. Fowlie, S. Hoof and W. Handley, ‘Nested Sampling for Frequentist Computation: Fast Estimation of Small p -Values’, *Physical Review Letters* **128**, 021801 (2022) [10.1103/PhysRevLett.128.021801](#), arXiv:2105.13923.
- [153] L. B. Pártay, A. P. Bartók and G. Csányi, ‘Efficient Sampling of Atomic Configurational Spaces’, *The journal of physical chemistry B* **114**, 10502 (2010) [10.1021/jp1012973](#), pmid: 20701382.

- [154] S. Martiniani, J. D. Stevenson, D. J. Wales and D. Frenkel, ‘Superposition Enhanced Nested Sampling’, *Physical Review X* **4**, 031034 (2014) [10.1103/PhysRevX.4.031034](https://doi.org/10.1103/PhysRevX.4.031034), [arXiv:1402.6306](https://arxiv.org/abs/1402.6306).
- [155] R. J. N. Baldock, L. B. Pártay, A. P. Bartók, M. C. Payne and G. Csányi, ‘Determining Pressure-Temperature Phase Diagrams of Materials’, *Physical Review B* **93**, 174108 (2016) [10.1103/PhysRevB.93.174108](https://doi.org/10.1103/PhysRevB.93.174108), [arXiv:1503.03404](https://arxiv.org/abs/1503.03404).
- [156] P. G. Bolhuis and G. Csányi, ‘Nested Transition Path Sampling’, *Physical Review Letters* **120**, 250601 (2018) [10.1103/PhysRevLett.120.250601](https://doi.org/10.1103/PhysRevLett.120.250601).
- [157] B. Szekeres, L. B. Pártay and E. Mátyus, ‘Direct Computation of the Quantum Partition Function by Path-Integral Nested Sampling’, *Journal of Chemical Theory and Computation* **14**, 4353 (2018) [10.1021/acs.jctc.8b00368](https://doi.org/10.1021/acs.jctc.8b00368), pmid: 29944376.
- [158] L. B. Pártay, G. Csányi and N. Bernstein, ‘Nested Sampling for Materials’, *The European Physical Journal B* **94**, 159 (2021) [10.1140/epjb/s10051-021-00172-1](https://doi.org/10.1140/epjb/s10051-021-00172-1).
- [159] Particle Data Group, ‘Review of Particle Physics’, *Physical Review D* **98**, 030001 (2018) [10.1103/PhysRevD.98.030001](https://doi.org/10.1103/PhysRevD.98.030001).
- [160] MissMJ, Cush and Acrux13, *Standard Model of Elementary Particles*, Wikimedia Commons, (2008) https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles_edit.svg (visited on 17/05/2023).
- [161] Franzl aus tirol, *Strong Coupling as Function of Energy*, Wikimedia Commons, (2012) https://commons.wikimedia.org/wiki/File:Strong_coupling_as_function_of_energy.svg (visited on 05/05/2023).
- [162] J. C. Collins, D. E. Soper and G. Sterman, ‘Factorization of Hard Processes in Qcd’, in *Perturbative QCD*, Vol. Volume 5, Advanced Series on Directions in High Energy Physics Volume 5 (WORLD SCIENTIFIC, 1989), pages 1–91, [10.1142/9789814503266_0001](https://doi.org/10.1142/9789814503266_0001).
- [163] Y. L. Dokshitzer, ‘Calculation of the Structure Functions for Deep Inelastic Scattering and e^+e^- Annihilation by Perturbation Theory in Quantum Chromodynamics.’, *Sov. Phys. JETP* **46**, 641 (1977).
- [164] V. N. Gribov and L. N. Lipatov, ‘Deep Inelastic $e p$ Scattering in Perturbation Theory’, *Sov. J. Nucl. Phys.* **15**, 438 (1972).
- [165] G. Altarelli and G. Parisi, ‘Asymptotic Freedom in Parton Language’, *Nuclear Physics B* **126**, 298 (1977) [10.1016/0550-3213\(77\)90384-4](https://doi.org/10.1016/0550-3213(77)90384-4).
- [166] The ATLAS Collaboration, *Measurement of Inclusive Jet and Dijet Cross Sections in Proton-Proton Collision Data at 7 TeV Centre-of-Mass Energy Using the ATLAS Detector*, ATLAS-CONF-2011-047 (CERN, Geneva, 2011), <https://cds.cern.ch/record/1338578>.
- [167] M. Cacciari, G. P. Salam and G. Soyez, ‘The Anti- K_t Jet Clustering Algorithm’, *Journal of High Energy Physics* **2008**, 063 (2008) [10.1088/1126-6708/2008/04/063](https://doi.org/10.1088/1126-6708/2008/04/063), [arXiv:0802.1189](https://arxiv.org/abs/0802.1189).
- [168] E. Bothmann, M. Schönherr and S. Schumann, ‘Reweighting QCD Matrix-Element and Parton-Shower Calculations’, *The European Physical Journal C* **76**, 590 (2016) [10.1140/epjc/s10052-016-4430-0](https://doi.org/10.1140/epjc/s10052-016-4430-0), [arXiv:1606.08753](https://arxiv.org/abs/1606.08753).
- [169] R. Kleiss, W. J. Stirling and S. D. Ellis, ‘A New Monte Carlo Treatment of Multiparticle Phase Space at High-energies’, *Comput. Phys. Commun.* **40**, 359 (1986) [10.1016/0010-4655\(86\)90119-0](https://doi.org/10.1016/0010-4655(86)90119-0).
- [170] S. Plätzer, *RAMBO on Diet*, (2013) [10.48550/arXiv.1308.2922](https://doi.org/10.48550/arXiv.1308.2922), [arXiv:1308.2922](https://arxiv.org/abs/1308.2922), preprint.

- [171] F. E. James, *Monte Carlo Phase Space*, Geneva, 1968, [10.5170/CERN-1968-015](#).
- [172] E. Byckling and K. Kajantie, 'N-Particle Phase Space in Terms of Invariant Momentum Transfers', *Nuclear Physics B* **9**, 568 (1969) [10.1016/0550-3213\(69\)90271-5](#).
- [173] E. E. R. O. BYCKLING and K. KAJANTIE, 'Reductions of the Phase-Space Integral in Terms of Simpler Processes', *Physical Review* **187**, 2008 (1969) [10.1103/PhysRev.187.2008](#).
- [174] P. Draggiotis, A. van Hameren and R. Kleiss, 'SARGE: An Algorithm for Generating QCD-antennas', *Physics Letters B* **483**, 124 (2000) [10.1016/S0370-2693\(00\)00532-3](#), [arXiv:hep-ph/0004047](#).
- [175] C. G. Papadopoulos, 'PHEGAS: A Phase-Space Generator for Automatic Cross-Section Computation', *Computer Physics Communications* **137**, 247 (2001) [10.1016/S0010-4655\(01\)00163-1](#), [arXiv:hep-ph/0007335](#).
- [176] F. Krauss, R. Kuhn and G. Soff, 'AMEGIC++ 1.0, A Matrix Element Generator In C++', *Journal of High Energy Physics* **2002**, 044 (2002) [10.1088/1126-6708/2002/02/044](#), [arXiv:hep-ph/0109036](#).
- [177] T. Stelzer and W. F. Long, 'Automatic Generation of Tree Level Helicity Amplitudes', *Computer Physics Communications* **81**, 357 (1994) [10.1016/0010-4655\(94\)90084-1](#), [arXiv:hep-ph/9401258](#).
- [178] F. A. Berends and W. T. Giele, 'Recursive Calculations for Processes with n Gluons', *Nuclear Physics B* **306**, 759 (1988) [10.1016/0550-3213\(88\)90442-7](#).
- [179] T. Gleisberg and S. Hoeche, 'Comix, a New Matrix Element Generator', *Journal of High Energy Physics* **2008**, 039 (2008) [10.1088/1126-6708/2008/12/039](#), [arXiv:0808.3674](#).
- [180] A. Buckley et al., 'General-Purpose Event Generators for LHC Physics', *Physics Reports* **504**, 145 (2011) [10.1016/j.physrep.2011.03.005](#), [arXiv:1101.2599](#).
- [181] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau and A. D. Polosa, 'ALPGEN, a Generator for Hard Multiparton Processes in Hadronic Collisions', *Journal of High Energy Physics* **2003**, 001 (2003) [10.1088/1126-6708/2003/07/001](#), [arXiv:hep-ph/0206293](#).
- [182] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli and M. Zaro, 'The Automated Computation of Tree-Level and next-to-Leading Order Differential Cross Sections, and Their Matching to Parton Shower Simulations', *Journal of High Energy Physics* **2014**, 79 (2014) [10.1007/JHEP07\(2014\)079](#), [arXiv:1405.0301](#).
- [183] W. Kilian, T. Ohl and J. Reuter, 'WHIZARD: Simulating Multi-Particle Processes at LHC and ILC', *The European Physical Journal C* **71**, 1742 (2011) [10.1140/epjc/s10052-011-1742-y](#), [arXiv:0708.4233](#).
- [184] V. Hirschi, R. Frederix, S. Frixione, M. Vittoria Garzelli, F. Maltoni and R. Pittau, 'Automation of One-Loop QCD Computations', *Journal of High Energy Physics* **2011**, 44 (2011) [10.1007/JHEP05\(2011\)044](#).
- [185] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H.-S. Shao and M. Zaro, 'The Automation of Next-to-Leading Order Electroweak Calculations', *Journal of High Energy Physics* **2018**, 185 (2018) [10.1007/JHEP07\(2018\)185](#), [arXiv:1804.10017](#).
- [186] J. M. Campbell and R. K. Ellis, 'Update on Vector Boson Pair Production at Hadron Colliders', *Physical Review D* **60**, 113006 (1999) [10.1103/PhysRevD.60.113006](#).

- [187] J. M. Campbell, S. Höche and C. T. Preuss, ‘Accelerating LHC Phenomenology with Analytic One-Loop Amplitudes’, *The European Physical Journal C* **81**, 1117 (2021) [10.1140/epjc/s10052-021-09885-0](https://doi.org/10.1140/epjc/s10052-021-09885-0).
- [188] S. Badger, B. Biedermann, P. Uwer and V. Yundin, ‘Numerical Evaluation of Virtual Corrections to Multi-Jet Production in Massless QCD’, *Computer Physics Communications* **184**, 1981 (2013) [10.1016/j.cpc.2013.03.018](https://doi.org/10.1016/j.cpc.2013.03.018), arXiv:1209.0100.
- [189] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf and S. Uccirati, ‘RECOLA: REcursive Computation of One-Loop Amplitudes’, *Computer Physics Communications* **214**, 140 (2017) [10.1016/j.cpc.2017.01.004](https://doi.org/10.1016/j.cpc.2017.01.004), arXiv:1605.01090.
- [190] A. Denner, J.-N. Lang and S. Uccirati, ‘Recola2: REcursive Computation of One-Loop Amplitudes 2’, *Computer Physics Communications* **224**, 346 (2018) [10.1016/j.cpc.2017.11.013](https://doi.org/10.1016/j.cpc.2017.11.013), arXiv:1711.07388.
- [191] S. Alioli, P. Nason, C. Oleari and E. Re, ‘A General Framework for Implementing NLO Calculations in Shower Monte Carlo Programs: The POWHEG BOX’, *Journal of High Energy Physics* **2010**, 43 (2010) [10.1007/JHEP06\(2010\)043](https://doi.org/10.1007/JHEP06(2010)043), arXiv:1002.2581.
- [192] J. Campbell, J. Huston and F. Krauss, *The Black Book of Quantum Chromodynamics: A Primer for the LHC Era* (Oxford University Press, 2018), [10.1093/oso/9780199652747.001.0001](https://doi.org/10.1093/oso/9780199652747.001.0001).
- [193] M. Bengtsson and T. Sjöstrand, ‘Coherent Parton Showers versus Matrix Elements - Implications of PETRA/PEP Data’, *Physics Letters B* **185**, 435 (1987) [10.1016/0370-2693\(87\)91031-8](https://doi.org/10.1016/0370-2693(87)91031-8).
- [194] G. Gustafson and U. Pettersson, ‘Dipole Formulation of QCD Cascades’, *Nuclear Physics B* **306**, 746 (1988) [10.1016/0550-3213\(88\)90441-5](https://doi.org/10.1016/0550-3213(88)90441-5).
- [195] M. H. Seymour, ‘A Simple Prescription for First Order Corrections to Quark Scattering and Annihilation Processes’, *Nuclear Physics B* **436**, 443 (1995) [10.1016/0550-3213\(94\)00554-R](https://doi.org/10.1016/0550-3213(94)00554-R), arXiv:hep-ph/9410244.
- [196] M. H. Seymour, ‘Matrix-Element Corrections to Parton Shower Algorithms’, *Computer Physics Communications* **90**, 95 (1995) [10.1016/0010-4655\(95\)00064-M](https://doi.org/10.1016/0010-4655(95)00064-M), arXiv:hep-ph/9410414.
- [197] L. Lonnblad, ‘Small-x Effects in W + Jets Production at the Tevatron’, *Nuclear Physics B* **458**, 215 (1996) [10.1016/0550-3213\(95\)00576-5](https://doi.org/10.1016/0550-3213(95)00576-5), arXiv:hep-ph/9508261.
- [198] G. Miu and T. Sjostrand, ‘W Production in an Improved Parton-Shower Approach’, *Physics Letters B* **449**, 313 (1999) [10.1016/S0370-2693\(99\)00068-4](https://doi.org/10.1016/S0370-2693(99)00068-4), arXiv:hep-ph/9812455.
- [199] S. Frixione and B. R. Webber, ‘Matching NLO QCD Computations and Parton Shower Simulations’, *Journal of High Energy Physics* **2002**, 029 (2002) [10.1088/1126-6708/2002/06/029](https://doi.org/10.1088/1126-6708/2002/06/029), arXiv:hep-ph/0204244.
- [200] P. Nason, ‘A New Method for Combining NLO QCD with Shower Monte Carlo Algorithms’, *Journal of High Energy Physics* **2004**, 040 (2004) [10.1088/1126-6708/2004/11/040](https://doi.org/10.1088/1126-6708/2004/11/040), arXiv:hep-ph/0409146.
- [201] S. Frixione and B. R. Webber, *The MC@NLO 3.3 Event Generator*, (2006) [10.48550/arXiv.hep-ph/0612272](https://doi.org/10.48550/arXiv.hep-ph/0612272), arXiv:hep-ph/0612272, preprint.
- [202] S. Catani, F. Krauss, R. Kuhn and B. R. Webber, ‘QCD Matrix Elements + Parton Showers’, *Journal of High Energy Physics* **2001**, 063 (2001) [10.1088/1126-6708/2001/11/063](https://doi.org/10.1088/1126-6708/2001/11/063), arXiv:hep-ph/0109231.

- [203] K. Hamilton, P. Richardson and J. Tully, ‘A Modified CKKW Matrix Element Merging Approach to Angular-Ordered Parton Showers’, *Journal of High Energy Physics* **2009**, 038 (2009) 10 . 1088 / 1126-6708 / 2009 / 11 / 038, arXiv:0905 . 3072.
- [204] S. Hoeche, S. Schumann and F. Siegert, ‘Hard Photon Production and Matrix-Element Parton-Shower Merging’, *Physical Review D* **81**, 034026 (2010) 10 . 1103 / PhysRevD . 81 . 034026, arXiv:0912 . 3501.
- [205] T. Carli, T. Gehrmann and S. Höche, ‘Hadronic Final States in Deep-Inelastic Scattering with Sherpa’, *The European Physical Journal C* **67**, 73 (2010) 10 . 1140 / epj c / s 10052-010-1261-2, arXiv:0912 . 3715.
- [206] L. Lonnblad and S. Prestel, ‘Matching Tree-Level Matrix Elements with Interleaved Showers’, *Journal of High Energy Physics* **2012**, 19 (2012) 10 . 1007 / JHEP03 (2012) 019, arXiv:1109 . 4829.
- [207] L. Lonnblad and S. Prestel, ‘Unitarising Matrix Element + Parton Shower Merging’, *Journal of High Energy Physics* **2013**, 94 (2013) 10 . 1007 / JHEP02 (2013) 094, arXiv:1211 . 4827.
- [208] W. T. Giele, D. A. Kosower and P. Z. Skands, ‘A Simple Shower and Matching Algorithm’, *Physical Review D* **78**, 014026 (2008) 10 . 1103 / PhysRevD . 78 . 014026, arXiv:0707 . 3652.
- [209] N. Lavesson and L. Lonnblad, ‘Extending CKKW-merging to One-Loop Matrix Elements’, *Journal of High Energy Physics* **2008**, 070 (2008) 10 . 1088 / 1126-6708 / 2008 / 12 / 070, arXiv:0811 . 2912.
- [210] K. Hamilton and P. Nason, ‘Improving NLO-parton Shower Matched Simulations with Higher Order Matrix Elements’, *Journal of High Energy Physics* **2010**, 39 (2010) 10 . 1007 / JHEP06 (2010) 039, arXiv:1004 . 1764.
- [211] S. Hoeche, F. Krauss, M. Schonherr and F. Siegert, ‘NLO Matrix Elements and Truncated Showers’, *Journal of High Energy Physics* **2011**, 123 (2011) 10 . 1007 / JHEP08 (2011) 123, arXiv:1009 . 1127.
- [212] T. Gehrmann, S. Hoeche, F. Krauss, M. Schonherr and F. Siegert, ‘NLO QCD Matrix Elements + Parton Showers in $E+e^- \rightarrow$ Hadrons’, *Journal of High Energy Physics* **2013**, 144 (2013) 10 . 1007 / JHEP01 (2013) 144, arXiv:1207 . 5031.
- [213] L. Lonnblad, ‘Correcting the Colour-Dipole Cascade Model with Fixed Order Matrix Elements’, *Journal of High Energy Physics* **2002**, 046 (2002) 10 . 1088 / 1126-6708 / 2002 / 05 / 046, arXiv:hep-ph / 0112284.
- [214] L. Lonnblad and S. Prestel, ‘Merging Multi-leg NLO Matrix Elements with Parton Showers’, *Journal of High Energy Physics* **2013**, 166 (2013) 10 . 1007 / JHEP03 (2013) 166, arXiv:1211 . 7278.
- [215] S. Hoeche, F. Krauss, M. Schonherr and F. Siegert, ‘QCD Matrix Elements + Parton Showers: The NLO Case’, *Journal of High Energy Physics* **2013**, 27 (2013) 10 . 1007 / JHEP04 (2013) 027, arXiv:1207 . 5030.
- [216] W. T. Giele, L. Hartgring, D. A. Kosower, E. Laenen, A. J. Larkoski, J. J. Lopez-Villarejo, M. Ritzmann and P. Skands, *The Vincia Parton Shower*, (2013) 10 . 48550 / arXiv . 1307 . 1060, arXiv:1307 . 1060, preprint.
- [217] M. L. Mangano, M. Moretti and R. Pittau, ‘Multijet Matrix Elements and Shower Evolution in Hadronic Collisions: $W b \bar{b} + n$ Jets as a Case Study’, *Nuclear Physics B* **632**, 343 (2002) 10 . 1016 / S0550-3213 (02) 00249-3, arXiv:hep-ph / 0108069.
- [218] F. Krauss, ‘Matrix Elements and Parton Showers in Hadronic Interactions’, *Journal of High Energy Physics* **2002**, 015 (2002) 10 . 1088 / 1126-6708 / 2002 / 08 / 015, arXiv:hep-ph / 0205283.

- [219] S. Mrenna and P. Richardson, ‘Matching Matrix Elements and Parton Showers with HERWIG and PYTHIA’, *Journal of High Energy Physics* **2004**, 040 (2004) [10.1088/1126-6708/2004/05/040](#), [arXiv:hep-ph/0312274](#).
- [220] S. Hoeche, F. Krauss, N. Lavesson, L. Lonnblad, M. Mangano, A. Schaelicke and S. Schumann, *Matching Parton Showers and Matrix Elements*, (2006) [10.48550/arXiv.hep-ph/0602031](#), [arXiv:hep-ph/0602031](#), preprint.
- [221] M. L. Mangano, M. Moretti, F. Piccinini and M. Treccani, ‘Matching Matrix Elements and Shower Evolution for Top-Quark Production in Hadronic Collisions’, *Journal of High Energy Physics* **2007**, 013 (2007) [10.1088/1126-6708/2007/01/013](#), [arXiv:hep-ph/0611129](#).
- [222] J. Alwall et al., ‘Comparative Study of Various Algorithms for the Merging of Parton Showers and Matrix Elements in Hadronic Collisions’, *The European Physical Journal C* **53**, 473 (2008) [10.1140/epjc/s10052-007-0490-5](#), [arXiv:0706.2569](#).
- [223] S. Hoeche, F. Krauss, S. Schumann and F. Siegert, ‘QCD Matrix Elements and Truncated Showers’, *Journal of High Energy Physics* **2009**, 053 (2009) [10.1088/1126-6708/2009/05/053](#), [arXiv:0903.1219](#).
- [224] R. Kleiss and R. Pittau, ‘Weight Optimization in Multichannel Monte Carlo’, *Computer Physics Communications* **83**, 141 (1994) [10.1016/0010-4655\(94\)90043-4](#), [arXiv:hep-ph/9405257](#).
- [225] T. Ohl, ‘Vegas Revisited: Adaptive Monte Carlo Integration Beyond Factorization’, *Computer Physics Communications* **120**, 13 (1999) [10.1016/S0010-4655\(99\)00209-X](#), [arXiv:hep-ph/9806432](#).
- [226] X. Glorot, A. Bordes and Y. Bengio, ‘Deep Sparse Rectifier Neural Networks’, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (2011), pages 315–323.
- [227] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, (2017) [10.48550/arXiv.1412.6980](#), [arXiv:1412.6980](#), preprint.
- [228] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, ‘Normalizing Flows for Probabilistic Modeling and Inference’, *Journal of Machine Learning Research* **22**, 1 (2021).
- [229] O. Alexandrov, *Diffeomorphism of a Square*, Wikimedia Commons, (2008) https://commons.wikimedia.org/wiki/File:Diffeomorphism_of_a_square.svg (visited on 30/05/2023).
- [230] C. Gao, J. Isaacson and C. Krause, ‘I-Flow: High-dimensional Integration and Sampling with Normalizing Flows’, *Machine Learning: Science and Technology* **1**, 045023 (2020) [10.1088/2632-2153/abab62](#), [arXiv:2001.05486](#).
- [231] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density Estimation Using Real NVP*, (2017) [10.48550/arXiv.1605.08803](#), [arXiv:1605.08803](#), preprint.
- [232] T. Müller, B. McWilliams, F. Rousselle, M. Gross and J. Novák, ‘Neural Importance Sampling’, *ACM Transactions on Graphics* **38**, 145:1 (2019) [10.1145/3341156](#).
- [233] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Cubic-Spline Flows*, (2019) [10.48550/arXiv.1906.02145](#), [arXiv:1906.02145](#), preprint.
- [234] H. M. Dolatabadi, S. Erfani and C. Leckie, *Invertible Generative Modeling Using Linear Rational Splines*, (2020) [10.48550/arXiv.2001.05168](#), [arXiv:2001.05168](#), preprint.
- [235] T. Janßen, ‘New Sampling Algorithms for High Energy Physics Simulations’, Master’s thesis (Georg-August-Universität Göttingen, 2019), unpublished.

- [236] *SciPost: SciPost Phys.* **8**, 069 (2020) - *Exploring Phase Space with Neural Importance Sampling*, (2023) <https://web.archive.org/web/20230602164212/http://scipost.org/SciPostPhys.8.4.069> (visited on 02/06/2023).
- [237] R. Verheyen, ‘Event Generation and Density Estimation with Surjective Normalizing Flows’, *SciPost Physics* **13**, 047 (2022) [10.21468/SciPostPhys.13.3.047](https://doi.org/10.21468/SciPostPhys.13.3.047), arXiv:2205.01697.
- [238] S. Pina-Otey, F. Sánchez, T. Lux and V. Gaitan, ‘Exhaustive Neural Importance Sampling Applied to Monte Carlo Event Generation’, *Physical Review D* **102**, 013003 (2020) [10.1103/PhysRevD.102.013003](https://doi.org/10.1103/PhysRevD.102.013003), arXiv:2005.12719.
- [239] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone and U. Köthe, ‘Invertible Networks or Partons to Detector and Back Again’, *SciPost Physics* **9**, 074 (2020) [10.21468/SciPostPhys.9.5.074](https://doi.org/10.21468/SciPostPhys.9.5.074), arXiv:2006.06685.
- [240] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, ‘Applying Neural Importance Sampling to Gluon Scattering’, in *Proceedings of The Eighth Annual Conference on Large Hadron Collider Physics — PoS(LHCP2020)*, Vol. 382, edited by B. Mansoulie, G. Marchiori, R. Salern and T. Bos (2020), page 056, [10.22323/1.382.0056](https://doi.org/10.22323/1.382.0056).
- [241] T. Janßen and S. Schumann, ‘Machine Learning Efforts in Sherpa’, *Journal of Physics: Conference Series* **2438**, 012144 (2023) [10.1088/1742-6596/2438/1/012144](https://doi.org/10.1088/1742-6596/2438/1/012144).
- [242] K. He, X. Zhang, S. Ren and J. Sun, ‘Deep Residual Learning for Image Recognition’, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pages 770–778, [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [243] C. M. Bishop, ‘Mixture Density Networks’, *Mixture density networks* (1994).
- [244] A. van den Oord and B. Schrauwen, ‘Factoring Variations in Natural Images with Deep Gaussian Mixture Models’, in *Advances in Neural Information Processing Systems*, Vol. 27 (2014).
- [245] C. Viroli and G. J. McLachlan, ‘Deep Gaussian Mixture Models’, *Statistics and Computing* **29**, 43 (2019) [10.1007/s11222-017-9793-z](https://doi.org/10.1007/s11222-017-9793-z).
- [246] F. Maltoni and T. Stelzer, ‘MadEvent: Automatic Event Generation with MadGraph’, *Journal of High Energy Physics* **2003**, 027 (2003) [10.1088/1126-6708/2003/02/027](https://doi.org/10.1088/1126-6708/2003/02/027), arXiv:hep-ph/0208156.
- [247] O. Mattelaer and K. Ostrolenk, ‘Speeding up MadGraph5_aMC@NLO’, *The European Physical Journal C* **81**, 435 (2021) [10.1140/epjc/s10052-021-09204-7](https://doi.org/10.1140/epjc/s10052-021-09204-7).
- [248] R. Cornish, A. Caterini, G. Deligiannidis and A. Doucet, ‘Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows’, in *Proceedings of the 37th International Conference on Machine Learning* (2020), pages 2133–2143.
- [249] L. Dinh, J. Sohl-Dickstein, H. Larochelle and R. Pascanu, *A RAD Approach to Deep Mixture Models*, (2020) [10.48550/arXiv.1903.07714](https://doi.org/10.48550/arXiv.1903.07714), arXiv:1903.07714, preprint.
- [250] J. Postels, M. Liu, R. Spezialetti, L. Van Gool and F. Tombari, *Go with the Flows: Mixtures of Normalizing Flows for Point Cloud Generation and Reconstruction*, (2021) [10.48550/arXiv.2106.03135](https://doi.org/10.48550/arXiv.2106.03135), arXiv:2106.03135, preprint.
- [251] V. Stimper, B. Schölkopf and J. M. Hernandez-Lobato, ‘Resampling Base Distributions of Normalizing Flows’, in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics* (2022), pages 4915–4936.

- [252] M. Bauer and A. Mnih, ‘Resampled Priors for Variational Autoencoders’, in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* (2019), pages 66–75.
- [253] G. G. P. F. Pires and M. A. T. Figueiredo, *Variational Mixture of Normalizing Flows*, (2020) [10.48550/arXiv.2009.00585](https://arxiv.org/abs/2009.00585), arXiv:2009.00585, preprint.
- [254] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, ‘Equation of State Calculations by Fast Computing Machines’, *The Journal of Chemical Physics* **21**, 1087 (1953) [10.1063/1.1699114](https://doi.org/10.1063/1.1699114).
- [255] W. K. Hastings, ‘Monte Carlo Sampling Methods Using Markov Chains and Their Applications’, *Biometrika* **57**, 97 (1970) [10.1093/biomet/57.1.97](https://doi.org/10.1093/biomet/57.1.97).
- [256] A. Joseph, *Markov Chain Monte Carlo Methods in Quantum Field Theories: A Modern Primer* (2020), [10.1007/978-3-030-46044-0](https://arxiv.org/abs/1912.10997), arXiv:1912.10997.
- [257] H. Kharraziha and S. Moretti, ‘The Metropolis Algorithm for On-Shell Four-Momentum Phase Space’, *Computer Physics Communications* **127**, 242 (2000) [10.1016/S0010-4655\(99\)00504-4](https://doi.org/10.1016/S0010-4655(99)00504-4), arXiv:hep-ph/9909313.
- [258] K. Kroeninger, S. Schumann and B. Willenberg, ‘(MC)**3 – a Multi-Channel Markov Chain Monte Carlo Algorithm for Phase-Space Sampling’, *Computer Physics Communications* **186**, 1 (2015) [10.1016/j.cpc.2014.08.024](https://doi.org/10.1016/j.cpc.2014.08.024), arXiv:1404.4328.
- [259] J. Skilling, ‘Nested Sampling for General Bayesian Computation’, *Bayesian Analysis* **1**, 833 (2006) [10.1214/06-BA127](https://doi.org/10.1214/06-BA127).
- [260] G. Ashton et al., ‘Nested Sampling for Physical Scientists’, *Nature Reviews Methods Primers* **2**, 39 (2022) [10.1038/s43586-022-00121-x](https://doi.org/10.1038/s43586-022-00121-x), arXiv:2205.15570.
- [261] W. J. Handley, M. P. Hobson and A. N. Lasenby, ‘PolyChord: Next-Generation Nested Sampling’, *Monthly Notices of the Royal Astronomical Society* **453**, 4385 (2015) [10.1093/mnras/stv1911](https://doi.org/10.1093/mnras/stv1911), arXiv:1506.00171.
- [262] R. M. Neal, ‘Slice Sampling’, *The Annals of Statistics* **31**, 705 (2003) [10.1214/aos/1056562461](https://doi.org/10.1214/aos/1056562461).
- [263] J. Alsing and W. Handley, ‘Nested Sampling with Any Prior You Like’, *Monthly Notices of the Royal Astronomical Society: Letters* **505**, L95 (2021) [10.1093/mnrasl/slab057](https://doi.org/10.1093/mnrasl/slab057), arXiv:2102.12478.
- [264] H. Bevins and W. Handley, *Piecewise Normalizing Flows*, (2023) [10.48550/arXiv.2305.02930](https://arxiv.org/abs/2305.02930), arXiv:2305.02930, preprint.
- [265] M. J. Williams, J. Veitch and C. Messenger, ‘Nested Sampling with Normalising Flows for Gravitational-Wave Inference’, *Physical Review D* **103**, 103006 (2021) [10.1103/PhysRevD.103.103006](https://doi.org/10.1103/PhysRevD.103.103006), arXiv:2102.11056.
- [266] M. J. Williams, J. Veitch and C. Messenger, *Importance Nested Sampling with Normalising Flows*, (2023) [10.48550/arXiv.2302.08526](https://arxiv.org/abs/2302.08526), arXiv:2302.08526, preprint.
- [267] J. A. Nelder and R. Mead, ‘A Simplex Method for Function Minimization’, *The Computer Journal* **7**, 308 (1965) [10.1093/comjnl/7.4.308](https://doi.org/10.1093/comjnl/7.4.308).
- [268] D. Wales and J. Doye, ‘Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms’, *The Journal of Physical Chemistry A* **101**, 5111 (1997) [10.1021/jp970984n](https://doi.org/10.1021/jp970984n), arXiv:cond-mat/9803344.
- [269] J. Krause, ‘Efficiency Improvements Using Machine Learning in Event Generators for the LHC’, Master’s thesis (Dresden, Tech. U., 2015), <https://cds.cern.ch/record/2804526>.

- [270] K. Danziger, ‘Efficiency Improvements in Monte Carlo Algorithms for High-Multiplicity Processes’, Master’s thesis (Dresden, Tech. U., 2020), <https://cds.cern.ch/record/2715727>.
- [271] F. Feroz, M. P. Hobson and M. Bridges, ‘MultiNest: An Efficient and Robust Bayesian Inference Tool for Cosmology and Particle Physics’, *Monthly Notices of the Royal Astronomical Society* **398**, 1601 (2009) [10.1111/j.1365-2966.2009.14548.x](https://doi.org/10.1111/j.1365-2966.2009.14548.x), [arXiv:0809.3437](https://arxiv.org/abs/0809.3437).
- [272] A. Coccaro, M. Letizia, H. Reyes-Gonzalez and R. Torre, *On the Curse of Dimensionality for Normalizing Flows*, (2023) [10.48550/arXiv.2302.12024](https://doi.org/10.48550/arXiv.2302.12024), [arXiv:2302.12024](https://arxiv.org/abs/2302.12024), preprint.
- [273] 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*, version 1, (2023) [10.48550/arXiv.2302.10449](https://doi.org/10.48550/arXiv.2302.10449), [arXiv:2302.10449](https://arxiv.org/abs/2302.10449), preprint.
- [274] 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’, 2022, [arXiv:2106.06507](https://arxiv.org/abs/2106.06507).
- [275] A. Valassi, S. Roiser, O. Mattelaer and S. Hageboeck, ‘Design and Engineering of a Simplified Workflow Execution for the MG5aMC Event Generator on GPUs and Vector CPUs’, *EPJ Web of Conferences* **251**, 03045 (2021) [10.1051/epjconf/202125103045](https://doi.org/10.1051/epjconf/202125103045), [arXiv:2106.12631](https://arxiv.org/abs/2106.12631).
- [276] A. Valassi, T. Childers, L. Field, S. Hageböck, W. Hopkins, O. Mattelaer, N. Nichols, S. Roiser and D. Smith, ‘Developments in Performance and Portability for MadGraph5_aMC@NLO’, in *Proceedings of 41st International Conference on High Energy Physics — PoS(ICHEP2022)* (2022), page 212, [10.22323/1.414.0212](https://doi.org/10.22323/1.414.0212), [arXiv:2210.11122](https://arxiv.org/abs/2210.11122).
- [277] E. Higson, W. Handley, M. Hobson and A. Lasenby, ‘Dynamic Nested Sampling: An Improved Algorithm for Parameter Estimation and Evidence Calculation’, *Statistics and Computing* **29**, 891 (2019) [10.1007/s11222-018-9844-0](https://doi.org/10.1007/s11222-018-9844-0), [arXiv:1704.03459](https://arxiv.org/abs/1704.03459).