May 2024

Reducing Negative Weights in MC@NLO by Improved Implementation of Born Spreading

Yuxiao Che

Theoretical Particle Physics
Division of Particle and Nuclear Physics
Department of Physics
Lund University

Bachelor thesis (15 hp) supervised by Rikkert Frederix

LUND
UNIVERSITY

Abstract

In this thesis, we present the efforts towards finding an improved implementation of the newly proposed Born Spreading method for reduction of negative $\mathbb{S}$ weights in the MC@NLO matching prescription. The improved implementations are tested on various LHC processes with MADGRAPH5_AMC@NLO (MG5_aMC) and compared to default MG5_aMC, $2 \times 2 \times 1$ folding and default Born Spreading. The runtime in each step and fraction of negative $\mathbb{S}$ weights were recorded for each method to support a detailed discussion of their performance under considerations of both runtime and fraction of negative weights reduced. It was found that improved Born Spreading outperforms default Born Spreading convincingly, often achieving a greater reduction of negative $\mathbb{S}$ weights with shorter runtime. Moreover, improved Born Spreading is also comparable to $2 \times 2 \times 1$ folding in terms of reduction of negative $\mathbb{S}$ weights, but with much shorter runtime.

Popular Science Summary

In the past century, physics is amongst many disciplines that have seen tremendously rapid development. In particular, physicists have identified the fundamental constituents of matter and proposed a theory that describes the dominating interactions between them. This theory, known as the *Standard Model* is arguably physics' best candidate for a theory of everything.

However, in addition to being powerful and ambitious, the Standard Model is also extremely complex. This leads to great difficulty in linking theory with experiments. In fact, most results from accelerator experiments are completely incomprehensible, in the sense that one cannot possibly ascertain what processes produced such results by the means of simple data analysis. To bridge this gap, computer simulations were developed. These simulations, known as *event generators*, have since become an integral tool for experimental particle physics.

In recent years, improving the efficiencies of event generators has been rising quickly on physicists' priority lists. This is due to the pending upgrades to major accelerator experiments around the world, which leads to greater demand of computational resources. Unfortunately, unlike optimising an arbitrary computer program, developers of event generators will often find their hands tied by the laws of physics, since some obstacles are of physical origin. One of such obstacles is the presence of *negative weights*.

By negative weights, we refer to events that are associated with a negative contribution (weight) to the result. They are essential for obtaining the correct result, but lead to results with greater uncertainty. Since precision is an requirement as well, presence of negative weights means that more events must be generated, consuming more computational resources. Therefore, reduction of negative weights is critical for improving efficiencies of event generators.

Traditionally, the problem is tackled by *folding*, a method that trades time for accuracy. Typically, using this method represents at least two times more computing time compared to default. Since our current problem is exactly the lack of computational resources, relying on such a method might not be the best idea.

Fortunately, a good alternative was presented recently. This new method, known as *Born Spreading*, is much less demanding and gives results comparable to minimal folding in most cases. Most importantly, it's full potential is yet to be explored.

My work is exactly to contribute to this exploration with the hope of finding an improved implementation of Born Spreading. Under a general consideration (runtime and effectiveness), it is not difficult to imagine that such an improved implementation can be considered comparable or even superior to folding. Therefore, if such an implementation is found, a discussion for including Born Spreading in the default program could be considered, representing a truly exciting opportunity to make a tiny contribution to a real problem in physics.

Contents

List of Acronyms

1. LHC - Large Hadron Collider
2. HL-LHC - High-Luminosity Large Hadron Collider
3. LO - Leading Order
4. NLO - Next-to-Leading Order
5. MC - Monte Carlo
6. MG5_aMC - MADGRAPH5_AMC@NLO
7. ISR - Initial State Radiation
8. FSR - Final State Radiation

List of Figures

1 Introduction

In modern particle physics, event generators play a major role as the bridge between theory and experiment. However, due to the wide usage, they are also huge sinks for computational resources. As such, to improve efficiencies of event generators has become an important task. Moreover, the pending upgrades to the collider experiments (e.g. HL-LHC) represent increasing demand of computational resources, which, combined with the stagnation of CPU development [3], adds a sense of urgency to the problem.

One of the obstacles for improving efficiency is the presence of negative weights at NLO precision. Negative weights are events associated with a negative event weight, whose presence means that often much more events must be generated to achieve the same level of accuracy as an event sample without negative weights. For an event sample with a fraction f of negative weights, the number of extra events needed to be generated scales as $1/(1-2f)^2$ [4]. It is then obvious that having a large number of negative weights will lead to huge inefficiency. Therefore, reduction of negative weights is an essential task for increasing efficiency of event generators.

The events generated at NLO precision with MC@NLO [5] can be classified into $\mathbb{S}$ and $\mathbb{H}$ events, both containing negative weights. The negative $\mathbb{S}$ and $\mathbb{H}$ weights are of different origins and are tackled in different ways. This thesis will focus on the negative $\mathbb{S}$-weights, for which *folding* [6] has been developed as an effective solution. However, employing folding represents multiple times longer runtime and therefore cannot be considered as an efficient solution. As an alternative, *Born Spreading* [7] has been proposed recently.

Tests with a quick implementation of Born Spreading already shows a reduction of negative $\mathbb{S}$-weights comparable with $2 \times 2 \times 1$ folding but a much shorter runtime. Since the possible implementations have not been explored yet, it is certainly reasonable to expect that there exists a better performing implementation.

There are multiple directions that can be followed to find such an improved implementation. One of which is relaxing the non-negative constraint on the spreading function $F(\Phi_R)$. In isolation, this means that more negative weights will be created. However, combined with the normalisation requirement, it means that larger reduction in negative weights can be achieved elsewhere. With a careful constructed implementation, the benefit should be able to outweigh the cost, giving a net increase in negative weights reduced.

This thesis presents the efforts made towards finding such an improved implementation. In the first section, the theoretical concepts required for understanding the construction of event generators are presented, along with a discussion of folding. Then, the Theory section focuses on Born Spreading, its default implementation and the event generator it is implemented in, MADGRAPH5_AMC@NLO. After that, the Method section describes the improved implementation, with a preceding discussion about the approach took to find it. Finally, the results obtained are presented and discussed, closing with an outlook of some ideas for further improvement and an evaluation of Born Spreading as a method of negative weight reduction.

2 Background

This section starting by providing an overview of Monte Carlo event generators, from its basis in Monte Carlo integration to its usage in simulation of event structures at collider experiments such as the LHC. Then, the MC@NLO matching prescription is presented, with the goal of introducing the origin of negative weights. Lastly, folding is discussed as the existing method on negative-weight reduction, laying foundation to a comparison between it and Born Spreading based on theoretical considerations.

2.1 Monte-Carlo Integration

Monte Carlo (MC) integration is a method of numerical integration that discretises the volume being integrated over with random sampling points. To see the idea, consider an n-dimensional integral with $\boldsymbol{x} = (x_1, ..., x_n)$ representing the coordinates. MC integration relies on the realisation that such an integral can be rewritten as the following infinite sum:

$$\int_\Omega d\boldsymbol{x} f(\boldsymbol{x}) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \frac{f(\boldsymbol{x}_i)}{\rho(\boldsymbol{x}_i)} \text{ , where } \int_\Omega d\boldsymbol{x}\rho(\boldsymbol{x}) = 1 \tag{2.1}$$

One can easily imagine that at fixed N, if MC integration is employed, then the accuracy can vary depending on different sampling methods. With the goal of reducing variance σ^2 at fixed N, the two most popular methods are *importance sampling* and *stratified sampling*.

In importance sampling, $\rho(\boldsymbol{x})$ is varied to reduce σ^2. The optimal choice has been shown to be [8]:

$$\rho(\boldsymbol{x}) = \frac{|f(\boldsymbol{x})|}{\int_\Omega d\boldsymbol{x}|f(\boldsymbol{x})|} \tag{2.2}$$

That is, to sample more points in the region where the absolute value of the integrand is largest.

In stratified sampling [8], the integration volume is split into N_s sub-volumes, where N/N_s points will be sampled in each sub-volume. The sizes of each sub-volume are then varied until they contribute equally to the total variance σ^2. This has been shown to minimise σ^2. Therefore, stratified sampling samples more points in regions that contribute more to the error (i.e. large and rapidly-changing integrand).

2.2 VEGAS Algorithm

While MC integration is extremely powerful for computing high-dimensional integrals, implementing it for particle physics is not quite straight-forward. In particular, neither of these sampling methods on themselves are appropriate for use in particle physics. To implement them, one must first know about the integrand's behaviour. This is clearly unrealistic for an event generators as the integrand will (roughly) be the matrix elements, which differ for different processes. Therefore, a general-purpose MC integration algorithm must be able to gain knowledge about the integrand while running and use this knowledge to modified the sampling procedure. These algorithms are known as *adaptive* and the most relevant of them is the VEGAS algorithm.

The VEGAS program in use now can employ both importance and stratified sampling [8, 9]. However, to show the idea, only importance sampling is considered as the modification to stratified sampling is only minor. For simplicity, consider the integral I of a one-dimensional function $f(x)$ on the interval $[0, 1]$:

$$I = \int_0^1 dx f(x) \tag{2.3}$$

In the initial step, an M-point Monte Carlo integration is performed with $\rho(x) = 1$, uniform distribution. Then, the domain $[0, 1]$ is split into N intervals between $N + 1$ points ($0 = x_0 < x_1 < ... < x_{N-1} < x_N = 1$). The distribution function is then redefined such that when a random number is chosen, it has equal probability of being in any interval. That is to say:

$$\rho(x)\Delta x_i = \frac{1}{N} \implies \rho(x) = \frac{1}{N\Delta x_i} \text{ , where } \Delta x_i = x_i - x_{i-1}$$

After that, a quantity m_i will be computed for each interval $[x_{i-1}, x_i]$. m_i is defined as:

$$m_i = K\frac{\overline{f}_i\Delta x_i}{\sum_j \overline{f}_j\Delta x_j} \text{ , where } \overline{f}_i = \sum_{x\in[x_{i-1},x_i]} |f(x)| \propto \frac{1}{\Delta x_i}\int_{x_{i-1}}^{x_i} dx|f(x)| \text{ and } K \text{ a large integer.} \tag{2.4}$$

Each interval $[x_{i-1}, x_i]$ is then sub-divided into $m_i + 1$ sub-intervals[1]. Finally, the points x_i are redefined as new set of points x'_i such that new intervals $[x'_{i-1}, x'_i]$ contain an equal number of sub-intervals and the total number of intervals is thus still N. The described procedure is then iterated until $m_i = m_j$.

Notice that by the definition of $\overline{f}_i$ in Eq. 2.4, m_i is larger in regions where $|f(x)|$ is the greatest. Therefore, more points will be sampled in those regions, which is exactly what importance sampling aims to achieve.

To generalise to n dimensions, a separable probability distribution function must be used:

$$\rho(\boldsymbol{x}) = \rho_{x_1}(x_1)...\rho_{x_n}(x_n)$$

The one-dimensional procedure is then employed for each $\rho_{x_i}(x_i)$. However, to have $\overline{f}_i$ serve the same purpose of showing how much a certain interval contributes to the whole integral, it must be redefined. The algorithm will end with an optimal grid, which focuses on the regions of greatest $|f(\boldsymbol{x})|$.

Lastly, to implement stratified sampling, the definitions of $\overline{f}_i$ are simply modified to give largest m_i in regions contributing most to errors.

2.3 Event Generation

In principle, the random points sampled on the optimal grid for MC integration can be treated as events. Indeed, the coordinate $\boldsymbol{x}$ will correspond to a set of momenta, which characterises an

[1]Note that we get at most $K + 1$ sub-intervals. Therefore, to have the program efficiently showing difference between each interval $[x_{i-1}, x_i]$, we need K to be large.

event. However, these events would be *weighted*. That is to say, they do not accurately represent the distribution of events required by nature. In this case, the distribution is the absolute value of the integrand (absolute value is needed since the integrand is not necessarily positive-definite, as discussed later). Since VEGAS is not a perfect algorithm, one should not expect the optimal grid to perfectly reflect the shape of the integrand, meaning that $|f(\boldsymbol{x})|/\rho(\boldsymbol{x})$ will not be flat (or $|f(\boldsymbol{x})|$, when plotted on the optimal grid, will not be flat).

Therefore, to call these points events, they must be *unweighted*. The procedure of unweighting is simply to choose random points in an $n+1$-dimensional box, whose height is $h = \max\{|f(\boldsymbol{x})|/\rho(\boldsymbol{x})\}_{\boldsymbol{x}\in\Omega}$. Note that this height is considerably lower than $h' = \max\{|f(\boldsymbol{x})|\}_{\boldsymbol{x}\in\Omega}$. Then, all points within the plotted integrand ($x_{n+1} < |f(\boldsymbol{x})|/\rho(\boldsymbol{x})$) are accepted as events with weights ± 1, where the sign is given by $\mathrm{sgn} f(\boldsymbol{x})$. The events generated this way will be unweighted and hence accurately represent the distribution.

Then naturally, the operation of event generators will consist of three steps. In *step 0*, the VEGAS algorithm is used to set up a grid (typically the fastest step). Then in *step 1*, the value of h (upper envelope) is computed. Finally in *step 2*, the unweighted events are generated (typically the slowest step).

Now, the only remaining task is to determine the integrands that should be input. These are exactly the differential cross-sections[2] of different components of the event structure (including the parton distribution functions (PDFs) if applicable). Since event generators aim at simulating an experiment in a collider experiment, the reasonable course of action would be to consider the structure of events occurring there. A general overview of the event structure at the LHC can be found in Appendix A. It is worth noting that the dominant interactions come from QCD. For the purpose of this thesis, the only relevant ones are hard scattering and initial/final state radiation.

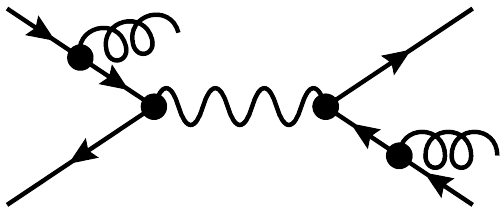


Figure 1: Example of initial and final state radiation (ISR & FSR) in a $2 \to 2$ scattering. The radiated particle is a gluon, which could either be emitted by incoming particles (ISR) or outgoing (FSR). This Feynman diagram and all subsequent ones are generated with the FEYNGAME package [1, 2].

The cross-sections of hard scatterings are rather standard. In practice, they are either built-in or generated by matrix element generators. Moreover, as a hard process, the integral will be free of divergences, hence no additional care is needed. These cross-sections will be given by *fixed-order matrix elements*.

Initial and final state radiation (examples are presented in Fig.1) are more difficult, mainly

[2]Unless specified otherwise, they are referred to simply as cross-sections from this point on.

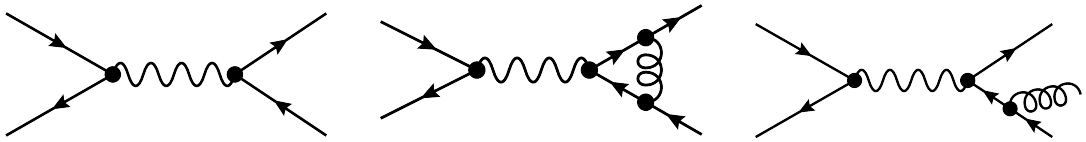
due to soft and collinear divergences. Their origins are discussed in Appendix B. Generally, the treatment involves writing the cross-sections in the soft and collinear limits and use them instead[3]. With these cross-sections, the further approach taken is known as the *parton shower evolution*, which studies the emissions of partons. A detailed description can be found in [10]. One additional complication is that initial state radiation can affect all subsequent processes, as it changes the beam energy. Therefore, instead of a natural forward evolution, employed for final state radiation, a backward evolution is performed, where the subsequent steps are fixed first, then the corresponding initial state radiation generated accordingly [10].

As illustrated above, for different components in the event structure, either fixed-order matrix elements or parton showers approach are taken to generate events. However, since the goal of building event generators is to simulate the entire event structure, we must prepare the results from fixed-order matrix elements such that it can be passed on to the subsequent parton showering program. This method of preparation is known as *matching prescriptions*.

2.4 NLO Matching

As mentioned above, the main demand for event generators are the need to study QCD processes at collider experiments. The studies of such processes are based on perturbation theory, usually up to next-to-leading order (NLO). This means that NLO matching prescriptions, which combines NLO matrix elements with partons showers are needed. This section will focus on one of such prescriptions, MC@NLO [5].

At NLO, three different terms contribute to the cross-section: 1) Born term B, corresponding to no emission (Fig.2a); 2) virtual term V, corresponding to exchange of a virtual particle (Fig.2b); 3) real emission term R, corresponding to emission of a particle (Fig.2c). Due to the extra emitted particle, R depends on an extra set of phase-space variables. That is, while $B = B(\Phi_B)$ and $V = V(\Phi_B)$, where Φ_B represent the Born phase-space variables (including the Bjorken x's), $R = R(\Phi_B, \Phi_R)$, where Φ_R represent the radiative phase-space variables.



(a) (b) (c)

Figure 2: Feynman diagrams of the Born (a), virtual (b) and real (c) contributions for NLO cross-sections

Consideration of processes at NLO leads to two issues: double counting and divergencies. By double counting, one refers to the fact that the NLO diagrams will already be included in parton showers. This is easiest to see for real emission, whose diagram is the same as for FSR.

[3]They are assumed to be accurate not only in the soft and collinear limits. Therefore, hard emissions are also generated with these cross-sections.

To avoid double counting, a shower subtraction term $K_{MC}(\Phi_B, \Phi_R)$ is defined. The integrand can then be split into two parts:

$$\underbrace{R - K_{\text{MC}}}_{\mathbb{H}} \text{ and } \underbrace{B + V + K_{\text{MC}}}_{\mathbb{S}}.$$

Events generated with the former are known as $\mathbb{H}$-events, corresponding to a Hard MC evolution with one real emission already. Events generated with latter are known as $\mathbb{S}$-events, corresponding to a Standard MC evolution with no prior emission.

However, there remains the issue of divergencies. Namely that integrals of V and R are separately divergent but their sum is finite [11, 12]. Since $\mathbb{H}$-events and $\mathbb{S}$-events are considered separately, the divergencies become a problem. For $\mathbb{H}$-events, the shower subtraction term will also cancel out the divergence in R, giving a convergent integrand. For $\mathbb{S}$-events, another term $I(\Phi_B, \Phi_R)$ is added and subtracted to cancel out the divergence. The resulting integrand is thus:

$$B + (V + I) + (K_{MC} - I)$$

Since this thesis focuses on $\mathbb{S}$-events, it is convenient to redefine the terms:

$$\widetilde{V}(\Phi_B) = V(\Phi_B) + \int I(\Phi_B, \Phi_R) d\Phi_R$$
$$\widetilde{K}_{MC}(\Phi_B, \Phi_R) = K_{MC}(\Phi_B, \Phi_R) - I(\Phi_B, \Phi_R)$$

These redefined terms will then be free of divergencies. Furthermore, the integral $I_{\mathbb{S}}$ over Born and radiative phase-space is:

$$I_{\mathbb{S}} = \int d\Phi_B \left[B(\Phi_B) + \widetilde{V}(\Phi_B) + \int d\Phi_R \widetilde{K}_{\text{MC}}(\Phi_B, \Phi_R) \right] \tag{2.5}$$

This integrand will not be positive definite as both $\widetilde{V}$ and $\widetilde{K}_{MC}$ may be negative. As a result, events with negative weights may be generated, which leads to increased statistical errors. In fact, it is possible to compute the fraction of negative-weight events as:

$$f_{\mathbb{S}} = \frac{1}{2}\left(1 - \frac{I_{\mathbb{S}}}{J_{\mathbb{S}}}\right) \tag{2.6}$$

where $J_{\mathbb{S}}$ is the same integral but of the absolute value of the integrand in Eq. 2.5. Therefore, if the integrand is positive-definite, there will be no negative-weight events, as was expected.

Lastly, the integrand with respect to Φ_B of Eq. 2.5 can be rewritten as:

$$\mathcal{F}^{(\mathbb{S})}(\Phi_B) = \int d\Phi_R \left[\frac{B(\Phi_B)}{\int d\Phi_R} + \frac{\widetilde{V}(\Phi_B)}{\int d\Phi_R} + \widetilde{K}_{MC}(\Phi_B, \Phi_R) \right], \tag{2.7}$$

known as the $\mathbb{S}$ *event generating functional*.

2.5 Folded Integration

Currently, the most common methods (e.g. [13]) used to reduce fraction of negative $\mathbb{S}$ weights make use of *folding*, implemented in the MINT algorithm and first introduced in [6]. To see the idea, consider a two-dimensional function $f(x, y)$ integrated over the unit square:

$$I = \int_0^1 \int_0^1 dxdy f(x, y) \tag{2.8}$$

Suppose that $f(x, y)$ is not positive definite but its y-integral is positive. One could rewrite I as:

$$I = \int_0^1 dx \underbrace{\left[\int_0^1 dy f(x, y)\right]}_{g(x)} \tag{2.9}$$

This gives an x-integral with a positive definite integrand $g(x)$.

To evaluate this integral numerically, $g(x)$ needs to be computed for given x, meaning that the y-integral must be taken. This is achieved by sampling N different y values per x value and averaging. Note that the evaluation of the y-integral need only to be done on a very coarse grid (small N)[4].

However, by folding over y, the information of the y-dependence of the original function $f(x, y)$ is lost. In particular, this poses a problem for event generation as the generated points will no longer be distributed according to the original integrand, which is the cross-section. Therefore, one can only fold over the variables that do not affect event generation.

Fortunately, for $\mathbb{S}$ events, the radiative phase-space variables Φ_R are such variables irrelevant in terms of event generation. This is evident through the $\mathbb{S}$ event generating functional's independence of Φ_R. Physically, as $\mathbb{S}$ events corresponds to no real emission, the generated events should only be described by Φ_B.

Therefore, on Φ_R the folded integration can be performed. The parametrisation of the radiative phase-space is done with the three FKS variables $\xi,\ y$, and ϕ [14], corresponding to energy, cosine of angle with a partner particle and azimuthal angle of the radiated particle. For these three variables, folding parameters $n_\xi,\ n_y,\ n_\phi$ (corresponding to N in the above example) are defined.

3 Theory

In this section, the theoretical underpinnings of Born Spreading, which this thesis is centred around, and its current implementation is outlined. In addition, MG5_aMC, the event generator in which Born Spreading is implemented will be discussed briefly.

[4]Suppose the x-integral is computed by sampling M points. Then $M \times N$ points will be sampled in the two-dimensional unit square. Therefore, since M should be large, N need not be.

3.1 Born Spreading

Consider first the generating functional given in Eq. 2.7:

$$\mathcal{F}^{(\mathbb{S})}(\Phi_B) = \int d\Phi_R \left[\frac{B(\Phi_B)}{\int d\Phi_R} + \frac{\widetilde{V}(\Phi_B)}{\int d\Phi_R} + \widetilde{K}_{MC}(\Phi_B, \Phi_R) \right]$$

It can be observed that the Born contribution $B(\Phi_B)$, incidentally also the only positive-definite term, is independent of Φ_R. This can be viewed as $B(\Phi_B)$ distributed uniformly in radiative phase-space. However, such distribution need not be uniform, meaning that a distribution function $F(\Phi_R)$ can be defined freely. Then as long as the distribution is normalised, an equivalent expression for the generating functional can be obtained:

$$\mathcal{F}^{(\mathbb{S})}(\Phi_B) = \int d\Phi_R \left[B(\Phi_B)F(\Phi_R) + \frac{\widetilde{V}(\Phi_B)}{\int d\Phi_R} + \widetilde{K}_{MC}(\Phi_B, \Phi_R) \right], \int F(\Phi_R)d\Phi_R = 1, \quad (3.1)$$

which is the generating functional for Born Spreading [7].

Since there is freedom in choosing $F(\Phi_R)$, it can be chosen to be larger when the no-Born contribution is more negative. This means that the integrand in Eq. 3.1 will have smaller negative regions compared to Eq. 2.7; and when it is still negative, the magnitude will be lower. In this sense, one can say that Born Spreading makes the integrand more positive-definite.

Compared to folding, the most obvious advantage of Born Spreading is speed. Folding is carried out in steps 1 & 2 and samples $N_f = n_\xi n_y n_\phi$ times more points than default. Typically, the minimal folding is considered to be $(2\times2\times1)$, which increases the number of points sampled by factor of $2 \times 2 = 4$. On the other hand, for Born Spreading, the increase in runtime is due to two sources: 1) sampling $F(\Phi_R)$; 2) integrand in Eq. 3.1 being more complicated than Eq. 2.7. Sampling of $F(\Phi_R)$ is carried out in step-0, meaning that although more (by default 10^6) points are sampled, the extra time needed will not be significant compared to total runtime. However, having a more complicated integrand influences steps 1 & 2 and leads to an increase in runtime. Nevertheless, the increase is much smaller compared to folding.

The most crucial disadvantage of Born Spreading is peak performance. After folding, the function is maximally positive-definite analytically. Therefore numerically, if sufficiently large folding parameters are chosen (i.e. integral evaluated precisely), the negative weights that can be eliminated will be eliminated completely. However, Born Spreading cannot guarantee this, even with the optimal choice of $F(\Phi_R)$ and a large number of sampling points. The reason is that the no-Born contributions may be large in magnitude and/or have strong Φ_B-dependence. Since $F(\Phi_R)$ is subject to multiple constraints (normalisation, Φ_B-independence, etc.), complete cancellation of negative no-Born contributions across the entire radiative phase-space is not always possible.

3.2 Default Implementation

The current implementation of Born Spreading is straight-forward. The no-Born contribution relative to Born contribution is defined as $W(\Phi_B, \Phi_R)$:

$$W(\Phi_B, \Phi_R) = \left(\frac{\widetilde{V}(\Phi_B)}{\int d\Phi_R} + \widetilde{K}_{MC}(\Phi_B, \Phi_R) \right) \frac{1}{B(\Phi_B)} \tag{3.2}$$

Then, to create an $F(\Phi_R)$ that is large when $W(\Phi_B, \Phi_R)$ is more negative, an option can be:

$$F(\Phi_R) = N \cdot \int |\min(0, W(\Phi_B, \Phi_R))| d\Phi_B \tag{3.3}$$

where N is a normalisation constant.

Such $F(\Phi_R)$ is 0 when the no-Born contribution is already positive, to allow it to be greater elsewhere. As for when the no-Born contribution is negative, $F(\Phi_R)$ is positive and its magnitude depends on the magnitude of the no-Born contribution relative to Born contribution. Therefore, this choice of $F(\Phi_R)$ will make the integrand in Eq. 3.1 more positive-definite.

In practice, the VEGAS grid is first set up according to Eq. 2.7. Then, the two-dimensional space of $\xi \times y$ is divided into a 40×40 uniform grid. After that, by default 10^6 points is sampled to obtain a suitable value of F in each bin. This will give a function F_{ij} defined piecewise. To arrive at the spreading function $F(\Phi_R)$, simply let $F(\Phi_R) = F_{ij}$ if the point Φ_R is in the ij-th bin. Lastly, with the generating functional defined instead as Eq. 3.1, the VEGAS grid is set up again and subsequent steps run.

3.3 MadGraph5_aMC@NLO

MG5_aMC [15] is an event generator that supports simulations at both LO and NLO (w.r.t. α_S) accuracy. Strictly speaking, MG5_aMC only generates events according to fixed-order matrix elements and matches them to parton showers. The subsequent showers are simulated by other programs, such as Pythia [16] and Herwig [17], via an interface to them.

For the purpose of this thesis, processes are always considered at NLO. In this case, the matching prescription used is MC@NLO. In addition, folding is already implemented in the program. The folding parameters can be changed in the run card, which contains all adjustable parameters for the program that will be run.

Once installed, the program is run with `./bin/mg5_aMC` executed in a terminal shell. Then, in the given prompt, event generation of an NLO process requires executing three commands consecutively:

```
MG5_aMC> generate process [QCD]
MG5_aMC> output folder_name
MG5_aMC> launch
```

For example, executing the following commands generates NLO events of $p\ p \to t\ \bar{t}$ and saves

relevant files to a folder named `pptt` in the MG5_aMC directory.

```
MG5_aMC> p p > t t~ [QCD]
MG5_aMC> output pptt
MG5_aMC> launch
```

When the run is completed, the relevant information regarding the process simulated will be summarised in the `index.html` file in the `pptt` folder.

After executing the `launch` command, the user will be prompted twice to make some modifications. The first prompt allows changes to run settings, in particular, the order, whether matching is done and how/whether shower is performed. The second prompt allows changes to the parameters and run cards (and shower card if shower is performed).

The generated events will be found in the `Events` folder within the output folder (`pptt` in example). It will be saved with extension lhe (Les Houches Events). The file format is described in [18]. See Fig.3 below for the positions of the relevant values for this thesis.

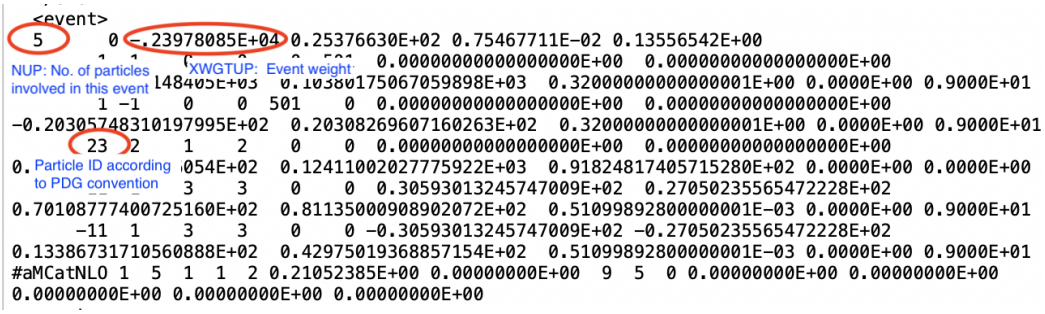


Figure 3: Example of one event from a sample events.lhe file. The mentioned Particle Data Group (PDG) convention can be found online at (https://pdg.lbl.gov/2023/mcdata/mc_particle_id_contents.html).

In the context of this thesis, only the circled values in Fig.3 will be relevant. They are:

- NUP, which is an integer value that specifies the number of particles involved in this events.
- XWGTUP, which is a double precision value that gives the weight of this event. In particular, if the events are unweighted, then this value will differ only by sign for different events.
- Particle ID, which is an integer value that corresponds to a specific particle in SM. Antiparticles will have negative values. For example, ± 6 corresponds to top/anti-top quark.

Among them, NUP values allow us to identify $\mathbb{S}$-events, XWGTUP values allow us to sort out the negative weights and particle ID's allow us to examine which particles are involved in a particular event.

4 Method

An apparent feature of the default implementation of Born Spreading outlined in Section 3.2 is that the spreading function $F(\Phi_R)$ is non-negative. However, this is not a necessary feature. In fact, as pointed out in [7], setting F negative in some regions should help further improve the performance of Born Spreading.

In this section, we will first present some methods that helped develop the improved implementations and evaluate their performance. Then, several possibilities for an improved implementation are outlined. In particular, they will be classified into naïve methods, representing some early efforts, and improved methods, which are developed in later stages and do lead to improvements.

4.1 Offline Tests

It is clear that the idea of removing the non-negative constraint on F is only a general direction. Therefore, the process of finding an improved implementation along this line will surely involve testing many ideas, among which most will not prove to be fruitful.

Since running the event generation will take around 10 minutes even for the simplest process, for efficiency, it is necessary to have a faster method for performance tests.

To this end, we make use of the 1 million sampled points for Born Spreading. We will run the event generation once and save $\{B(\Phi_B), \widetilde{V}(\Phi_B) + \widetilde{K}_{MC}(\Phi_B, \Phi_R), \Phi_R\}$ at these points to a file using the `write` command in FORTRAN. Then, a script can be written to read and process these samples offline to give F.

In particular, we can use these sampled points to compute $\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R$, which is directly related to the fraction of negative weights $f_{\mathbb{S}}$, since it is the size of the integral over the negative regions. More specifically:

$$\begin{aligned} f_{\mathbb{S}} &= \frac{1}{2}\left(1 - \frac{I_{\mathbb{S}}}{J_{\mathbb{S}}}\right) \\ &= \frac{1}{2}\left(1 - \frac{\iint |\max(0, B(F+W))| d\Phi_B d\Phi_R - \iint |\min(0, B(F+W))| d\Phi_B d\Phi_R}{\iint |\max(0, B(F+W))| d\Phi_B d\Phi_R + \iint |\min(0, B(F+W))| d\Phi_B d\Phi_R}\right) \\ &= \frac{\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R}{J_{\mathbb{S}}}, \end{aligned}$$

where $J_{\mathbb{S}}$ is approximately constant if the size of negative weights is small.

However, it should be noted that this is only an approximate measure of performance since the value of $\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R$ computed from the sample points will be an estimate and $J_{\mathbb{S}}$ is only approximately constant. Nevertheless, it should give a good enough idea as to whether the implementation being tested represents an improvement upon the default.

4.2 Computing $f_{\mathbb{S}}$

From Eq. 2.6, we see that $f_{\mathbb{S}}$ can be computed from $I_{\mathbb{S}}$ and $J_{\mathbb{S}}$, which are both outputs of MG5_aMC. However, our motivation for reducing negative weights is so that fewer events gen-

erated are associated with a negative weight. Therefore, it is much more reasonable to just look at the event file and count the number of negatively-weighted $\mathbb{S}$-events.

To this end, a PYTHON script is written to read the .lhe file. The script first counts the total number of $\mathbb{S}$-events $n_{\mathbb{S}}$, which is the number of events with the smallest NUP value (see Fig.3). Then, by looking at the first digit of the event weight XWGTUP, the number of negatively-weighted $\mathbb{S}$ events $n_{\mathbb{S}}^-$ can be counted. Finally, we print $f_{\mathbb{S}} = n_{\mathbb{S}}^- / n_{\mathbb{S}}$.

4.3 Naïve Implementation

With the restriction of non-negative F lifted, it is natural to devise a method in which a negative value is assigned to F at a certain bin based on gathered information. This is in all ways similar to the idea behind the default implementation.

Mathematically, at a bin where we wish to set F negative[5], the naïve methods set the value of F to:

$$F = -C \int W(\Phi_B, \Phi_R) d\Phi_B \tag{4.1}$$

where C is an arbitrary constant that varies from method to method.

Eq. 4.1 is indeed reasonable, as F will be more negative if the no-Born contribution is more positive. However, a closer inspection will reveal that an implementation of this cannot be straightforward.

To see this, we remark that $\int W(\Phi_B, \Phi_R) d\Phi_B \propto \text{avg}_{\Phi_B}(W(\Phi_B, \Phi_R))$ by the mean-value theorem. It is thus inevitable to have $\int W(\Phi_B, \Phi_R) d\Phi_B > W(\Phi_B, \Phi_R)$ for some Φ_B values. Moreover, if $W(\Phi_B, \Phi_R)$ has sharp peaks in Φ_B, the inequality will hold for most Φ_B. This implies that:

$$F(\Phi_R) + W(\Phi_B, \Phi_R) < 0, \tag{4.2}$$

which means that the integrand in Eq. 3.1 will be made negative in some regions by the choice of F in Eq. 4.1. In other words, creating negative weights is inevitable, which is not the case for the default implementation, where F is set to zero when the no-Born contribution is positive-definite.

Of course, setting F negative means that F can be more positive in regions where the no-Born contribution is negative. Therefore, creating negative weights is not necessarily a problem, granted that more negative weights are reduced elsewhere.

However, for any choice of C, it is difficult to justify why one expect the reduced negative weights to outweigh the created. Hence the statement that implementations of these naïve methods cannot be straightforward.

Practically, implementation of the naïve methods consists of two steps:

1. Set F according to Eq. 4.1 with certain C if the no-Born contribution is positive-definite
2. At remaining bins, use default implementation (Eq. 3.3) with the appropriate normalisation constant

Offline tests are ran with $C = \{1, 1/1600\}$[6], both performed worse than default Born Spreading.

[5]The condition for this decision also varies between methods

[6]Recall that 1600 is the total number of bins

4.4 Improved Implementation

The main issue with the naïve methods is the inability to predict its impact on performance, which the improved methods must solve. An immediate consideration is to devise an iterative method with some performance criterion in place. Alternatively, the naïve methods must be exchanged for a method of greater predictability.

4.4.1 Non-Iterative

We note first that this method requires the sampled points to be saved in an array. Then, the value of F is determined in two steps. In step 1, an un-normalised F_0 is chosen such that in every bin:

$$\int B(\Phi_B)[F_0(\Phi_R) + W(\Phi_B, \Phi_R)]d\Phi_B = a$$

where a is an arbitrary constant.

Rearranging this requirements gives the expression for this initial spreading function F_0:

$$F_0(\Phi_R) = \frac{a - \int B(\Phi_B)W(\Phi_B.\Phi_R)d\Phi_B}{\int B(\Phi_B)d\Phi_B} \tag{4.3}$$

Then, in step 2, this F will be normalised by setting

$$F(\Phi_R) = b \cdot F_0(\Phi_R) + \left(1 - b \cdot \int F_0(\Phi_R)d\Phi_R\right) \frac{\int |\min(0, B(F_0 + W))|d\Phi_B}{\iint |\min(0, B(F_0 + W))|d\Phi_B d\Phi_R}, \tag{4.4}$$

where b is another arbitrary constant.

Notice that in step 1, all Φ_R bins are made to have equal weight. Therefore, bins containing predominantly positive (negative) weights will have their weights reduced (raised) with negative (positive) F.

Then in step 2, the bin-wise values of F is modified according to the bins' contribution to total size of negative weights. This added value will always be positive. Therefore, for F at a certain bin, there can be three possible outcomes after step 2:

1. F was already positive after step 1 and remains positive;

2. F was negative after step 1, but this bin has a large contribution to negative weights, relative to its distance to the set weight a. So, F becomes positive;

3. F was negative after step 1 and this bin's contribution to negative weights is small, so it remains negative.

Among the three, outcome 3 demands particular interest. For outcome 3 to materialise, it is necessary that $\int B(1 + W)d\Phi_B > a$ and the process of setting $\int B(F + W)d\Phi_B = a$ did not make $B(F + W)$ negative in large magnitude or over a large region. But if this is the case, then the concern raised in the previous section about negative F increasing negative weights by too much is not warranted.

4.4.2 Iterative

For an iterative method, we can build upon either the naïve method or the improved method (Section 4.4.1). For the naïve method, the resulting iterative method will be of a hit-or-miss nature, since the iterations won't necessarily be progressive. On the other hand, for the improved method, it is possible to have the iterations progress towards a prescribed destination that is believed to be optimal. Therefore, the iterative method will be built upon the improved method.

From the improved method, a normalised spreading function $F(\Phi_R)$ is obtained. Considering this, an iterative procedure is designed to be:

$$F = F + c\left(\frac{\int |\min(0, B(F+W))| d\Phi_B}{\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R} - \frac{1}{\int d\Phi_R}\right), \tag{4.5}$$

where c is an arbitrary constant.

It is easy to check that upon integration over Φ_R, the expression within brackets in Eq. 4.5 vanishes. Therefore, F will continue to be normalised.

Such an iterative procedure will end naturally if all bins contribute equally to the remaining negative weights. Moreover, consider a bin that contributes more to negative weights than average, F will be increased at that bin, which reduces its contribution to negative weights. In other words, the iterative procedure does progress towards this natural end.

As for free parameters, the constant c determines the "step-size" between iterations. One can remark that extreme values should not be chosen since small c leads to negligible changes between iterations and large c introduces instability.

Lastly, the iteration procedure should not be allowed to continue indefinitely. Instead, it will be stopped when $\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R$ stops decreasing or when a preset number n_{iter} is reached. This is for runtime consideration and the fact that there is no guarantee that the minimum $f_{\mathbb{S}}$ will not be reached midway.

5 Results

Given the positive results from the offline tests, the non-iterative and iterative methods are both implemented in MG5_aMC with free parameters set to $a = 0, b = 1$ for both and $c = 1, n_{\text{iter}} = 1000$ for iterative only.

The event generations were run on a laptop with 1 million unweighted events generated for each process. All processes are generated with the four-flavour scheme, employed in MG5_aMC by default. Other relevant parameters are also unchanged from MG5_aMC default.

From Table 1, it is evident that for all four processes tested, the non-iterative and iterative methods both outperform default Born Spreading in terms of negative weight reduction, while maintaining a similar (often shorter) runtime. Moreover, they also outperform $2 \times 2 \times 1$ folding for three out of four processes, including two processes for which default Born Spreading does not, while often being considerably faster. These observations would suggest that the improved implementation of Born Spreading is indeed an efficient method for negative weight reduction.

	Step-0 (s)	Step-1 (s)	Step-2 (s)	Negative $\mathbb{S}$-weights (%)	Increase in runtime[7] (%)
$p\,p \to e^+\,e^-$					
Default MG5_aMC	3	21	252	5.9	-
$2 \times 2 \times 1$ folding	3	49	528	1.8	210
Default Born Spreading	182	49	385	1.8	223
Non-iterative	198	53	319	1.6	207
Iterative	227	100	399	1.5	263
$p\,p \to t\,\bar{t}$					
Default MG5_aMC	3	164	1674	12.1	-
$2 \times 2 \times 1$ folding	3	309	3466	3.9	205
Default Born Spreading	272	172	2033	4.4	135
Non-iterative	297	204	1634	3.5	116
Iterative	297	212	1851	3.5	128
$p\,p \to W^+\,j$					
Default MG5_aMC	13	1045	6960	27.6	-
$2 \times 2 \times 1$ folding	18	3060	21900	16.1	312
Default Born Spreading	700	1056	7800	22.3	119
Non-iterative	741	1194	7800	21.1	121
Iterative	784	1151	8700	21.2	133
$p\,p \to W^+\,t\,\bar{t}$					
Default MG5_aMC	4	841	4380	5.3	-
$2 \times 2 \times 1$ folding	7	1596	7860	3.0	198
Default Born Spreading	274	1029	8760	3.1	193
Non-iterative	289	975	5460	2.7	129
Iterative	307	1029	6180	2.8	144

Table 1: Runtime (s) at each step, fraction of negative $\mathbb{S}$-events (%), increase in total runtime w.r.t. default MG5_aMC (%) for various LHC processes with default MG5_aMC, $2 \times 2 \times 1$ folding, default Born Spreading, non-iterative method and iterative method.

In fact, when the goal is to obtain an event sample with less negative weights in a short time, improved Born Spreading can even be thought of as a superior alternative to folding.

However, runtime itself is insufficient to determine the consumption of computational resources. In particular, as mentioned in Section 4.4.1, the improved implementation require saving the sampled points. The total number of points saved could easily amount to several million depending on the number of diagrams this process can occur via and the number of integration channels for each diagram. This should translate to data of size on the order of megabytes. Therefore, in most conceivable circumstances, saving the sampled points should not be a cause of concern.

[7]Here runtime refers to the runtime sum of the three steps shown.

Another point of interest is the performance of Born Spreading in general (default and improved) for $p\ p \to W^+\ j$. As mentioned in the introduction, the increase in consumption of computational resources scales as $1/(1-2f)^2$. From the shape of this function, we see that negative weight reduction for processes with larger f should be of greater interest. In other words, if we achieve the same absolute reduction of negative weights (say 5%) for two processes with fractions of negative weights f_1 and f_2, then $f_1 > f_2$ implies that more computational resources are saved on process with f_1. With this in mind, the performance of Born Spreading on $p\ p \to W^+\ j$ becomes quite disappointing as it worsened significantly.

It is, however, possible to speculate the reason for such a worsened performance. Firstly, we should note the lack of Φ_B-dependence of the spreading function $F(\Phi_R)$, meaning that across Φ_B space, we assign the same F. Moreover, this means that the values we use to determine F will be various Φ_B-integrals to remove the Φ_B dependence. As we discussed in Section 4.3, the integral, which is proportional to the average, is at best a rough estimate of the function's behaviour. Therefore, if $W(\Phi_B, \Phi_R)$ has very different Φ_R-dependencies at different Φ_B's, $\int W d\Phi_B$ will be a bad estimate and hence cause Born Spreading to be less effective. Secondly, we have noted in Section 4.1 that $f_{\mathbb{S}}$ is directly related to $\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R$, which depends on the size of the regions where $F < W$ and their difference $|F - W|$ at those regions. Since there is a normalisation constraint on F, if we have $|W| \gg 1$ at most bins (i.e. no-Born contribution much larger than Born), then Born Spreading will surely be ineffective. Lastly, we note that $p\ p \to W^+\ j$ is a very special process, since a jet is already included at LO. In principle, this jet will be indistinguishable from the real emission at NLO. Therefore, when the event generation is ran, there is a point where the decision of which particle is which is made. Such ambiguity may also lead to unexpected influences on the Born Spreading method.

Finally, we may compare the non-iterative method to the iterative. In principle, the at most 0.1% absolute difference in negative weight fraction should be neglected as it falls into the range of statistical fluctuations. However, by construction, since a stopping condition is inserted for the iterative method, we should expect the iterative method to at least always be as good as non-iterative. This is shown, although inconclusively, to be false. If this observation were to hold after more extensive testing, then a possible reason may be over-training. Since a sample of 1 million points should only give a rough picture of the function in Φ_B and Φ_R spaces, performing minor adjustments on F based on information from the sampled points may not be entirely justified. Therefore, although the iterative method should lead to improvements by construction, its actual impact may not be as expected.

6 Extensions

Although a improved implementation is found, there is no proof that it is optimal. In fact, some ideas to pursue for further improvement can already be suggested. Due to time constraint, they cannot be tested and will instead be presented in this section.

6.1 Ratio Method

This idea returns to default Born Spreading and considers the function W_-, defined as:

$$W_-(\Phi_B, \Phi_R) = \begin{cases} 0, \ W \geq -F \\ W(\Phi_B, \Phi_R), \ W < -F \end{cases} , \tag{6.1}$$

which gives the ratio between the no-Born and Born contribution at the negative regions.

Since the spreading function is ultimately multiplied to $B(\Phi_B)$, the quantity $\int |W_-| d\Phi_B$ should play a role. To see the benefit, consider two bins A and B contributing equally to the total size of negative weights. If at the negative regions in bin A, both Born and no-Born are small, and at bin B they are both large. Then to make $\min(0, B(F+W))$ zero (i.e. eliminate negative weight), bin A needs a larger F than bin B.

However, note that the primary quantity to consider should still be the relative contribution to negative weights

$$\frac{\int |\min(0, B(F+W))| d\Phi_B}{\iint |\min(0, B(F+W))| d\Phi_B d\Phi_R}$$

and $\int |W_-| d\Phi_B$ can be used to make adjustments.

6.2 Spreading $\widetilde{V}(\Phi_B)$

As one might have noticed, by the exact same argument used to introduce $F(\Phi_R)$ for $B(\Phi_B)$, another spreading function $G(\Phi_R)$ can be introduced for $\widetilde{V}(\Phi_B)$. A brief analysis shows that F and G need not be separately normalised (see Appendix C). Instead, it is sufficient that

$$\int F(\Phi_R) d\Phi_R = 1 + \frac{\int \widetilde{V}(\Phi_B) d\Phi_B}{\int B(\Phi_B) d\Phi_B} \left(1 - \int G(\Phi_R) d\Phi_R \right)$$

which may give more freedom in assigning values for F and G at different bins.

It is, however, difficult to determine appropriate values for G. This is because $\widetilde{V}(\Phi_B)$ is not positive-definite and $G(\Phi_R)$ continues to be independent of Φ_B, where the latter means we can only look at $\int \widetilde{V} d\Phi_B$ and the former says this integral will poorly reflect the behaviour of $\widetilde{V}$.

7 Conclusion

In this thesis, we presented efforts towards finding an improved implementation of Born Spreading for reduction of negative $\mathbb{S}$ weights in MC@NLO matching prescription. The two variations (non-iterative and iterative) of the improved method are implemented in MG5_aMC for a comparison of runtime and fraction of negative weights with default MG5_aMC, $2 \times 2 \times 1$ folding and default Born Spreading.

The results show that both variations are an improvement over default Born Spreading, achieving greater negative weight reduction with often shorter runtime. Moreover, compared

to $2 \times 2 \times 1$ folding, improved Born Spreading showed great advantage in terms of runtime and achieved at least comparable reduction of negative weights. As for comparison between the two variations, the performance on negative weight reduction is almost identical but the iterative method is consistently slower. In addition, the iterative method could lead to over-training, meaning that spending extra runtime for minor adjustments may not be justified. Therefore, we may make the conclusion that the non-iterative variation of improved Born Spreading is indeed an efficient method for negative weight reduction with decent performance.

However, we must note that if the folding parameters are increased, folding will certainly achieve greater negative weight reduction than improved Born Spreading. Therefore, improved Born Spreading cannot be thought of as a substitute for folding. If the goal is to generate an event sample with almost no negative weights, folding is still the only option.

Nevertheless, as the runtime for improved Born Spreading is comparable to default MG5_aMC and its performance is comparable to $2 \times 2 \times 1$ folding, then as suggested in [7], an argument for using Born Spreading as the new default mode for MG5_aMC can be made.

Acknowledgments

I would like to express my gratitude to my supervisor Rikkert, for this wonderful opportunity and guidance throughout this journey. He encouraged me to explore the path forward myself, but is always there to lend a helping hand when I'm lost. In discussions, Rikkert is patient and open-minded. He helped me refine my ideas while pointing out things that I missed or is unaware of. I consider myself lucky to have Rikkert as my supervisor on this first research experience. Thank you!

I'm also deeply grateful to my family, who encouraged me to major in physics when I was unsure and never stopped believing in me since.

Lastly, I also want to thank my friends, for the game nights, which helped take my mind off from the project when I got stuck, and everything else.

References

[1] R.V. Harlander, S.Y. Klein and M. Lipp, *FeynGame*, *Comput. Phys. Commun.* **256** (2020) 107465 [2003.00896].

[2] R. Harlander, S.Y. Klein and M.C. Schaaf, *FeynGame-2.1 – Feynman diagrams made easy*, *PoS* **EPS-HEP2023** (2024) 657 [2401.12778].

[3] A.A. Chien and V. Karamcheti, *Moore's law: The first ending and a new beginning*, *Computer* **46** (2013) 48.

[4] HEP SOFTWARE FOUNDATION collaboration, *HL-LHC Computing Review: Common Tools and Community Software*, in *Snowmass 2021*, P. Canal et al., eds., 8, 2020, DOI [2008.13636].

[5] S. Frixione and B.R. Webber, *Matching NLO QCD computations and parton shower simulations*, *JHEP* **06** (2002) 029 [hep-ph/0204244].

[6] P. Nason, *MINT: A Computer program for adaptive Monte Carlo integration and generation of unweighted distributions*, 0709.2085.

[7] R. Frederix and P. Torrielli, *A new way of reducing negative weights in MC@NLO*, *Eur. Phys. J. C* **83** (2023) 1051 [2310.04160].

[8] G.P. Lepage, *A New Algorithm for Adaptive Multidimensional Integration*, *J. Comput. Phys.* **27** (1978) 192.

[9] G.P. Lepage, *VEGAS: AN ADAPTIVE MULTIDIMENSIONAL INTEGRATION PROGRAM*, .

[10] A. Buckley et al., *General-purpose event generators for LHC physics*, *Phys. Rept.* **504** (2011) 145 [1101.2599].

[11] T. Kinoshita, *Mass singularities of Feynman amplitudes*, *J. Math. Phys.* **3** (1962) 650.

[12] T.D. Lee and M. Nauenberg, *Degenerate Systems and Mass Singularities*, *Phys. Rev.* **133** (1964) B1549.

[13] R. Frederix, S. Frixione, S. Prestel and P. Torrielli, *On the reduction of negative weights in MC@NLO-type matching procedures*, *JHEP* **07** (2020) 238 [2002.12716].

[14] S. Frixione, Z. Kunszt and A. Signer, *Three jet cross-sections to next-to-leading order*, *Nucl. Phys. B* **467** (1996) 399 [hep-ph/9512328].

[15] 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*, *JHEP* **07** (2014) 079 [1405.0301].

[16] C. Bierlich et al., *A comprehensive guide to the physics and usage of PYTHIA 8.3*, *SciPost Phys. Codeb.* **2022** (2022) 8 [2203.11601].

[17] M. Bahr et al., *Herwig++ Physics and Manual*, *Eur. Phys. J. C* **58** (2008) 639 [0803.0883].

[18] E. Boos et al., *Generic User Process Interface for Event Generators*, in *2nd Les Houches Workshop on Physics at TeV Colliders*, 9, 2001 [hep-ph/0109068].

[19] Education, Communications and Outreach Group, CERN, "LHC the guide FAQ." https://cds.cern.ch/record/2809109/files/CERN-Brochure-2021-004-Eng.pdf, 2021.

[20] Torbjörn Sjöstrand, "Introduction to Event Generators 1." http://home.thep.lu.se/~torbjorn/talks/desy16a.pdf, 2016.

A Event structure at LHC

At the Large Hadron Collider (LHC), two counter-circulating proton beams are collided. As of the most recent run, the collisions have centre of mass energy $E_{CM} = 13$ TeV [19]. For each LHC event, there will exist various components forming the *event structure*. When two protons collide, the relevant event structure composes of [20]:

(a) *Evolution of parton distribution functions*. Since it is the partons within protons that participate in interactions, the evolution of PDFs in time (and space) will influence the processes that take place;

(b) *Hard scattering*, a high energy collision between partons. This is typically considered the primary process one wishes to study;

(c) *Resonance decays*. Once unstable resonance particles created, they will decay according to their lifetimes;

(d) *Initial state radiation*, emissions of incoming particles (i.e. partons in protons);

(e) *Final state radiation*, emissions of outgoing particles (i.e. products of the hard scattering);

(f) *Multiple parton-parton interactions*. This is a collection of all interactions between other partons present. They will scatter, either hardly or softly, and emit initial and final state radiations;

(g) *Hadronisation*. The partons present, both remnants of the initial state and products of miscellaneous processes, will form hadrons due to confinement;

(h) *Hadronic decay*. The unstable hadrons will decay into stable particles.

At the detectors, the stable particles or jets formed by them are measured. The complexity of the event structure is mainly due to the presence of QCD processes. Due to such complexity, in order to study the measurements made, event generators are required.

B Origin of Soft and Collinear Divergences

As mentioned in Section 2.3, soft and collinear divergences, which arise when considering NLO processes, demand extra care in treatment of NLO cross-sections. In this Appendix, their origin will be discussed.

Consider an NLO process with one real emission (e.g. a $2 \to 2$ scattering where one of the final state particles emit another particle). Then, looking only at this emission, it can be illustrated as:

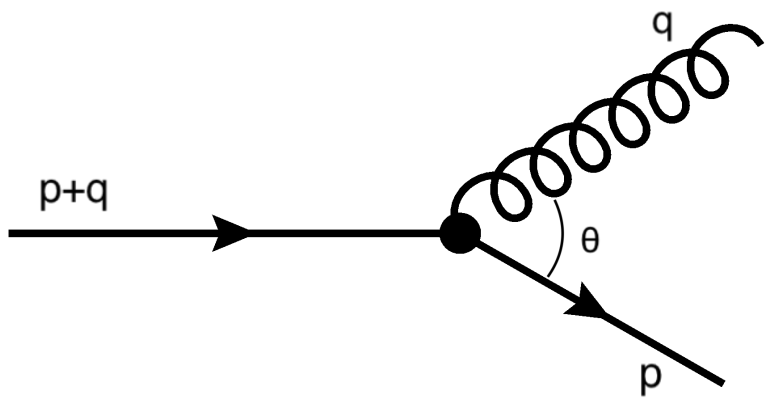


Figure 4: Diagram illustrating the real emission.

where $(p+q), p$ and q are the four-momenta of the original particle before emission, emitted particle and original particle after emission respectively. In addition, the angle between emitted particle and original particle is θ.

Considering the entire Feynman diagram, the propagator representing the original particle before emission will be an internal line. Therefore, assuming massless particles, the factor $1/(p+q)^2$ must be multiplied to the matrix element. It is then obvious that having $(p+q)^2 = 0$ will lead to divergences.

Since the particles are massless, the expression can be equivalently written as:

$$(p+q)^2 = \underbrace{p^2}_{0} + \underbrace{q^2}_{0} + 2p \cdot q = 2p \cdot q \tag{B.1}$$

Then, let

$$\begin{aligned} p &= E(1,0,0,1) \\ q &= E'(1,0,\sin\theta,\cos\theta), \end{aligned}$$

which means that $(p+q)^2$ is given by:

$$(p+q)^2 = 2p \cdot q = 2EE'(1-\cos\theta) \tag{B.2}$$

From Eq. B.2, it can be seen clearly that $(p+q)^2$ will tend to 0 when $E' \to 0$, corresponding to emission of a soft particle, and when $\theta \to 0$, corresponding to emission of a particle collinear with the original particle.

C Derivation for normalisation condition of spreading functions F and G

Recall the integral evaluated for $\mathbb{S}$-event generation (Eq. 2.5)

$$I_{\mathbb{S}} = \int d\Phi_B \left[B(\Phi_B) + \widetilde{V}(\Phi_B) + \int d\Phi_R \widetilde{K}_{\text{MC}}(\Phi_B, \Phi_R) \right]$$

If we multiply $B(\Phi_B)$ and $\widetilde{V}(\Phi_B)$ by arbitrary spreading functions $F(\Phi_R)$ and $G(\Phi_R)$, for the integral to remain unchanged, we must require that

$$\iint [B(\Phi_B)F(\Phi_R) + \widetilde{V}(\Phi_B)G(\Phi_R)] d\Phi_B d\Phi_R = \int [B(\Phi_B) + \widetilde{V}(\Phi_B)] d\Phi_B$$

Rewriting the left-hand-side gives an equation to solve for the normalisations of F and G

$$\begin{aligned} &\int B(\Phi_B)d\Phi_R \int F(\Phi_R)d\Phi_R + \int \widetilde{V}(\Phi_B)d\Phi_R \int G(\Phi_R)d\Phi_R = \int B(\Phi_B)d\Phi_B + \int \widetilde{V}(\Phi_B)d\Phi_B \\ \implies &\left(\int F(\Phi_R)d\Phi_R - 1 \right) \int B(\Phi_B)d\Phi_B = \left(1 - \int G(\Phi_R)d\Phi_R \right) \int \widetilde{V}(\Phi_B)d\Phi_B \end{aligned}$$

This equation can then be rearranged to give

$$\int F(\Phi_R)d\Phi_R = 1 + \frac{\int \widetilde{V}(\Phi_B)d\Phi_B}{\int B(\Phi_B)d\Phi_B}\left(1 - \int G(\Phi_R)d\Phi_R\right) \tag{C.1}$$

$$\int G(\Phi_R)d\Phi_R = 1 + \frac{\int B(\Phi_B)d\Phi_B}{\int \widetilde{V}(\Phi_B)d\Phi_B}\left(1 - \int F(\Phi_R)d\Phi_R\right) \tag{C.2}$$