Speeding up MadGraph5 _aMC@NLO

O. Mattelaer, K. Ostrolenk

1 Center for Cosmology, Particle Physics and Phenomenology (CP3), Université Catholique de Louvain (UCLouvain), B1348 Louvain-la-Neuve, Belgium
2 Centre de Calcul Intensif et de Stockage de Masse, Université Catholique de Louvain (UCLouvain), B1348 Louvain-la-Neuve, Belgium
3 University of Manchester, School of Physics and Astronomy, Schuster Building, Oxford Road, Manchester M13 9PL, United Kingdom

Received: date / Accepted: date

Abstract In this paper we will describe two new optimisations implemented in MadGraph5 _aMC@NLO, both of which are designed to speed up the computation of leading-order processes (for any model). First we implement a new method to evaluate the squared matrix element, dubbed helicity recycling, which results in factor of two speed-up. Second, we have modified the multi-channel handling of the phase-space integrator providing tremendous speed-up for VBF-like processes (up to thousands times faster).

1 Introduction

While the LHC is still running, preparation is starting for the High-Luminosity LHC. As part of this preparation, the CPU efficiency of our Monte-Carlo generators is crucial due to the sheer number of events that need to be generated. Given the current constraints on the LHC IT budget this will not be possible without significant software improvement [1,2]. While the full CPU time of the LHC experiment is not dominated by event generation, it is still estimated to represent between ten and twenty percent of it. Consequently, we received a specific request to speed up our code by at least 20% and ideally by a factor of two [1].

In addition to the High-Luminosity LHC, work is also starting for future high-energy accelerators [3]. Going to the high-energy regime will amplify multi-scale issues which in turn can reduce the efficiency of event generation. This issue is particularly striking for VBF-like processes where the current phase-space integration algorithm either fails to generate the requested number of events or takes an unreasonable time to do so.

The different avenues for speeding up Monte-Carlo integration is a well covered topic. Such efforts can be classified into four different categories. First one can optimise the evaluation of the function being integrated, which in our case is the matrix element [4–7]. Second one can optimise the integration method to minimise the number of times such functions need to be evaluated [8–12]. Third, one can try to use more efficiently the various types of hardware (e.g. GPU, MPI, vectorization) [13–16] and finally one can play with the weights of the sample to optimise/re-use information [17–22].

In the context of this work, we will focus on optimising MadGraph5 _aMC@NLO (MG5aMC) [23–24], one of the main Monte-Carlo generators [25–27] and we will combine two different methods to achieve our goal, one optimising the time to evaluate the matrix element and one optimising the phase-space integrator. The strategy we employed was to keep the main design choices in place (in particular the helicity amplitude method [28] and the single diagram enhancement method [29]) and to deeply study them to see how they could be further improved.

To reduce the time needed to evaluate a given squared matrix element, we use a quite standard Memory/CPU trade-off guided by physics considerations. We have identified parts of the computation that can be stored in memory in order to avoid their re-computation later. We dubbed this method helicity recycling since the additional terms stored in memory correspond to identical sub-expressions shared between different helicity configurations. This improvement will be presented in section (3.1). We start in section (2.1) by explaining the main idea behind helicity recycling. The details of the associated speed-up will be then presented in section (2.3).

The second improvement that we have implemented improves the phase-space integrator. Contrary to the first method, it is more challenging to introduce an improvement that acts positively on all processes. On the other hand, the expected (and obtained) gain can be much more impressive with this method. For this paper we will mainly focus on the speed-up of VBF-like processes since they are the ones where MG5aMC has some specific issues.

This will be covered in section (3) where we start by reviewing the current multi-channel strategy (the single diagram enhancement method) in section (3.1). We then explain in section (3.2) the methods used to have a better handling on the integration of t-channel propagator. We then turn in section (3.3) to the deeper changes we made to the multi-channel strategy as well as the impact on some physical observables – which are beyond LO accurate. Our speed-ups are then compared to the older version of the code in section (3.4).
2 Helicity recycling within the Helicity Amplitude Method

2.1 Helicity Amplitudes and previous optimisations in MG5aMC

When evaluating a matrix element one can identify two key structures: the Lorentz structure of the matrix element and its colour structure. Within MG5aMC evaluation of these two structures factorises at the amplitude level [5]. Hence it is possible to discuss one without the other. The helicity-recycling optimisation only impacts the evaluation of the Lorentz structure and in this section we will explain how this evaluation is performed.

2.1.1 Helicity amplitude formalism

Matrix elements typically contain factors of spinors (from the external fermions) and Lorentz vectors (from internal propagators). When analytically evaluating the square of this matrix element it is common to remove the dependence on spinors via their on-shell condition:

$$\sum_{s=\pm 1} u_s(p)\bar{u}_s(p) = \delta + m, \quad (1)$$

where the sum is over helicity. The squared matrix element can then be reduced to a series of scalar products between the external fermions and Lorentz vectors (from internal propagators). When analytically evaluating the square of this matrix element it is written as:

$$iM = (-ie)^2\bar{u}(p_2)\gamma_\mu u(p_1)\frac{-ig^{\mu\nu}}{s}\bar{u}(p_3)\gamma_\nu \gamma_5 v(p_4). \quad (5)$$

Remember that $u(p)$ and $v(p)$ are Dirac spinors and can either be left- or right-handed. In the massless limit this corresponds to them having $+1$ or $-1$ helicity eigenvalues respectively. In what follows we will also use $\psi$ to denote a general Dirac spinor. The subscript $L$ and $R$ will be used to denote left- and right-handedness respectively. It is common to use the following notation when expressing helicity amplitudes, for a spinor of momentum $p_n$:

$$\psi_L = \langle n | \psi_L | n \rangle, \quad \psi_R = \langle n | \psi_R | n \rangle. \quad (6)$$

One can show that $\langle n^+m^- \rangle = \langle n^+m^- \rangle = 0$. Hence, for the matrix element to be non-zero, the incoming fermions must have opposite helicity and similarly for the outgoing. Hence, for all the possible helicity combinations we could construct, the only ones that give a non-zero matrix element are:

$$\begin{align*}
1^-2^-3^+4^+, & \quad (8) \\
1^+2^-3^+4^-, & \quad (9) \\
1^+2^-3^-4^+, & \quad (10) \\
1^-2^+3^-4^-. & \quad (11)
\end{align*}$$

Here, the notation $n^m$ means the fermion with momentum $p_n$ has helicity $m$. The first helicity combination will give:

$$iM(1^-2^+3^-4^+) = (-ie)^2(2\gamma^\mu)\frac{-ig^{\mu\nu}}{s}(3\gamma_\nu 4), \quad (12)$$

$$= 2\frac{ie^2}{s}[41|23]. \quad (13)$$

The Feynman diagrams in this paper were generated using the Tikz-Feynman package [38].
Using the relation $|mn\rangle\langle mn| = (mn)[nm] = p_m \cdot p_n$, one finds the square to be:

$$\left|\mathcal{M} (1-2^+3^+4^-)\right|^2 = 4e^4 \frac{u^2}{s^2},$$

(14)

By parity, the helicity combination $1^+2^-3^+4^-$ will give the same result. The only remaining contributing helicity combinations are the same two but with $1 \leftrightarrow 2$. By symmetry arguments, one can see they will give the same result as in equation (14) but with $u \rightarrow t$. Hence the final result for this matrix element becomes:

$$\frac{1}{4} \sum_{\text{helicity}} \left|\mathcal{M}\right|^2 = 2e^4 \frac{t^2 + u^2}{s^2}.$$  

(15)

### 2.1.2 MadGraph implementation

The main points from the simple example looked at in section 2.1.1 are:

1. Diagrams are evaluated at the amplitude level (before squaring the matrix element).
2. To get the final result these diagrams must be summed over helicity combinations.
3. Only some of these helicity combinations will actually contribute.

MG5aMC follows the HELAS strategy [28] to compute amplitudes. It starts by computing the spinorial representation of all external particles. These spinors are then iteratively combined using some model specific functions (generated thanks to the Aloha [30] package). In this way the full diagram can be built up. This process is represented in Fig. (2). In the helicity amplitude formalism one distinguishes three types of functions:

- **external wave function**: function calculating the spinorial representation (e.g. $u, \bar{v}, e^\mu$) of the external particles evaluated for a given helicity.
- **internal wave function**: function calculating the spinorial representation (see Eq. (16)) of an internal particle (i.e. of a propagator) evaluated for a given helicity combination.
- **amplitude**: function fully contracting all spinorial representations and therefore returning the value for a given helicity combination of the amplitude of the associated Feynman diagram.

As depicted in Fig. (2), the evaluation of the matrix element starts by evaluating all the external wave functions, before combining them to get the internal wave functions associated to the propagators. Finally it reaches the last vertex and at that point returns an amplitude.

For example, consider again the $e^-e^+ \rightarrow \mu^-\mu^+$ process. After the computation of the spinors associated to the electrons and muons (which depends only their momenta and helicity), the algorithm will call the routine for calculating the internal photon wave function. The analytic expression of this photon wave function $\phi_\gamma$ will be:

$$\phi^\mu_h h_i = \frac{-ie^\mu}{s} \psi^\mu_h \bar{\psi}^h_i,$$

(16)

where $\psi_\mu$ has been used to represent the muon spinors. As already mentioned, these will be dependent on the helicity of the muons, labelled $h_3$ and $h_4$. Note that the wave function associated to this propagator is not unique since it depends on which vertex is used to compute it.

In this example, we are already at the last vertex and all that remains is to contract the various wave functions, taking into account the Lorentz/spinor structure of the associated vertex. Analytically, this is written as:

$$\mathcal{M}^{h_1 h_2 h_3 h_4} = \left(-ie^\mu \frac{\psi_{h_3}^\mu}{s} \bar{\psi}_{h_4}^\mu \phi^\mu h_4 h_3\right)\phi^{h_2 h_1},$$

(17)

$$= (-ie)^2 \psi_{h_3}^\mu \gamma_\mu \bar{\psi}_{h_4}^\mu \left(-\frac{ig_{\mu\nu}}{s} \psi_{h_3}^{\gamma \mu} \gamma_\mu \bar{\psi}_{h_4}^\gamma \right),$$

(18)

where, just as with the muons, $\psi_\mu$ has been used to represent the spinors of the electrons. This is the same as the expression in (12) but without the specific choice of helicity combination.

MG5aMC generates a FORTRAN subroutine that carries out the above process for a given matrix element, a given helicity combination and then returns the squared amplitude.

Further, the same propagator can appear in multiple Feynman diagrams, so being moved outside of the loop over diagrams. Furthermore, the same propagator can appear in multiple Feynman diagrams and in such a case it is also highly beneficial to move

5Evaluation of the matrix element for a single phase-space point is also possible in C++, CUDA and a python wrapper is also available.

6This representation/meta-code is technically accurate only for colourless processes but the general idea is still valid for QCD processes since MG5aMC is using the colour-flow formalism [5].
it out of the Feynman diagram’s loop. Therefore a more accurate representation of MG5aMC, prior to this paper, can be written like:

```plaintext
result = 0
loop over CONTRIBUTING helicity
  loop over external particles
    ext_i = get_spinsor(p_i , hel_i)
  loop over all possible propagator
    propa_i = Vertex(ext_x, propa_y)
  # amplitude/ final vertex
  sumamp = 0
  loop over all Feynman Diagram
    | sumamp += Vertex(ext_x, propa_y)
  # squaring the sum of amplitudes
  result += sumamp ** sumamp
```

Secondly, recall the point made at the start of this section that only a subset of helicity combinations contributes. Hence, another optimisation already included in the above meta-code exploits this fact and makes sure the loop over helicity combinations only includes “contributing” ones. We shall refer to this optimisation as ‘helicity filtering’.

The matrix routine discussed in this section is called by the madevent binary. This binary is also responsible for generating phase space points, evaluating the PDFs and writing the events (amongst other things). We can evaluate the computational cost of the matrix routine by comparing the number of instructions it executes to the total number executed by madevent. This is presented in table (1) for top-quark pair production plus gluons. The diagrams for $t\bar{t}$ are simpler and fewer in number than those of $t\bar{g}g$. Hence the total number of madevent instructions executed by madevent increases across these processes. Furthermore, this means the matrix routine is also responsible for an increasing percentage of the instructions: $23\% \rightarrow 96\% \rightarrow \sim 100\%$.

We also see that for $t\bar{g}g$ and $t\bar{g}gg$ (the more complex processes) the amplitude routines account for 44% and 51% of the computation, making it the dominant contribution to the matrix element calculation. This is again due to the higher number of diagrams and since the number of unique propagators does not scale as fast as the diagram multiplicity. Hence it is important that any future optimisation targets not just wave-function routines but also the amplitude routines.

Table 1: Here we present the number of instructions evaluated by the matrix routine (matrix1) and the total number of instructions evaluated by madevent. In brackets we also present these numbers as a percentage of the madevent total. These results are presented for three processes: $gg \rightarrow t\bar{t}$, $gg \rightarrow t\bar{g}g$ and $gg \rightarrow t\bar{g}gg$. We have also broken down the matrix routine into the number of instructions (again alongside their percentage of the total) evaluated by calls to functions that evaluate external spinors (ext), internal wavefunctions (int) and amplitudes (amp). The data in this table was obtained using a combination of Valgrind [39] and KCacheGrind [40].

Finally, one can also see from the table that the wavefunction and amplitude routines do not add up to the number of matrix routine instructions. This is because the routine has other things to evaluate, most noticeably the colour factors for the relevant amplitudes. Such computation is even the second hot-spot for the three gluon multiplicity case and therefore will limit the potential impact of our optimisation.

2.2 Helicity recycling

In general, when summing over helicity combinations, the same spinor with the same helicity can appear multiple times. For example, in the combinations $[5] \rightarrow [1]$, each helicity spinor (such as $1^+$) appears twice. Hence when MG5aMC loops over the combinations it will calculate the same wave function for these spinors multiple times (see the above meta-code). This is a waste of computation time—even if ultimately irrelevant for external wavefunctions. It would be cheaper to only calculate the wave functions once, save their values and reuse it when appropriate.
The same principle also applies to the wave function of internal particles. Such routines take other wave functions as input, therefore for a subset of helicity combinations the same input will be given and the code will waste time by re-computing the same quantity. For example, when looking at the internal (photon) wave function given in Eq. (16) the helicity combination \([8, 10]\) will give exactly the same result. Hence, the computational efficiency can again be improved by only calculating such internal wave functions once and then re-using the value when necessary.

This technique of calling the wave-function routines only once is what we refer to as “helicity recycling” and can be written in terms of pseudo-code as:

```plaintext
1. loop over external particles
2. | loop over helicity
3. | | propa_kl = Vertex (ext_ij, propa_xy)
4. | | ext_ij = get_spinor (p_i, hel_j)
5. | loop over all possible propagator
6. | loop over all Feynman Diagram
7. | sumamp = Vertex(ext_xy, ..)
8. | # squaring the sum of amplitudes
9. | result += sumamp * sumamp
10. loop over all helicity combination
11. result = 0
```

Here you can note that all stored variables have an additional index indicating which helicity combination was used to compute them.

While such an optimisation sounds natural, it has two potential issues. First, the amount of RAM needed for the computation will increase. However the RAM is currently dominated by meta-data related to the phase-space integrator and not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is dominated by meta-data related to the phase-space integrator. This corresponds to (see Eq. (17)):\[ \mathcal{M}_{h_1h_2h_3} = \bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \phi_{h_3}. \] (19)

Where we have explicitly added indices representing the associated helicity (or helicity combination for internal wave functions) of each component. Now, when computing \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_4} \), the factor \(\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \) will be the identical and can therefore be re-used multiple times. Therefore we have been able to optimise the code further by implementing a recycling of this factor. However, a similar factor can also be recycled between \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_5} \). Therefore it is advantageous to compare (for each Feynman diagram) which one of the following expressions:

- \[\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{1h_1} \gamma_\mu \phi_{h_5}.\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \phi_{h_5}.\]

The latter reductions are much larger because evaluating the内部 wavefuction who is vanishing only for some helicity combination. The former reductions are more advantageous to compare (for each Feynman diagram) which helicity combinations and amplitudes do not contribute when creating the internal (photon) wave function given in Eq. (16) the same input will be given and the code will waste time by re-computing the same quantity. For example, when looking at the internal (photon) wave function given in Eq. (16) the helicity combination \([8, 10]\) will give exactly the same result. Hence, the computational efficiency can again be improved by only calculating such internal wave functions once and then re-using the value when necessary.

This technique of calling the wave-function routines only once is what we refer to as “helicity recycling” and can be written in terms of pseudo-code as:

```plaintext
1. loop over external particles
2. | loop over helicity
3. | | propa_kl = Vertex (ext_ij, propa_xy)
4. | | ext_ij = get_spinor (p_i, hel_j)
5. | loop over all possible propagator
6. | loop over all Feynman Diagram
7. | sumamp = Vertex(ext_xy, ..)
8. | # squaring the sum of amplitudes
9. | result += sumamp * sumamp
10. loop over all helicity combination
11. result = 0
```

Here you can note that all stored variables have an additional index indicating which helicity combination was used to compute them.

While such an optimisation sounds natural, it has two potential issues. First, the amount of RAM needed for the computation will increase. However the RAM is currently dominated by meta-data related to the phase-space integrator and not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is dominated by meta-data related to the phase-space integrator. This corresponds to (see Eq. (17)):\[ \mathcal{M}_{h_1h_2h_3} = \bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \phi_{h_3}. \] (19)

Where we have explicitly added indices representing the associated helicity (or helicity combination for internal wave functions) of each component. Now, when computing \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_4} \), the factor \(\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \) will be the identical and can therefore be re-used multiple times. Therefore we have been able to optimise the code further by implementing a recycling of this factor. However, a similar factor can also be recycled between \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_5} \). Therefore it is advantageous to compare (for each Feynman diagram) which one of the following expressions:

- \[\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{1h_1} \gamma_\mu \phi_{h_5}.\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \phi_{h_5}.\]

The latter reductions are much larger because evaluating the internal wavefunction who is vanishing only for some helicity combination. The former reductions are more advantageous to compare (for each Feynman diagram) which helicity combinations and amplitudes do not contribute when creating the internal (photon) wave function given in Eq. (16) the same input will be given and the code will waste time by re-computing the same quantity. For example, when looking at the internal (photon) wave function given in Eq. (16) the helicity combination \([8, 10]\) will give exactly the same result. Hence, the computational efficiency can again be improved by only calculating such internal wave functions once and then re-using the value when necessary.

This technique of calling the wave-function routines only once is what we refer to as “helicity recycling” and can be written in terms of pseudo-code as:

```plaintext
1. loop over external particles
2. | loop over helicity
3. | | propa_kl = Vertex (ext_ij, propa_xy)
4. | | ext_ij = get_spinor (p_i, hel_j)
5. | loop over all possible propagator
6. | loop over all Feynman Diagram
7. | sumamp = Vertex(ext_xy, ..)
8. | # squaring the sum of amplitudes
9. | result += sumamp * sumamp
10. loop over all helicity combination
11. result = 0
```

Here you can note that all stored variables have an additional index indicating which helicity combination was used to compute them.

While such an optimisation sounds natural, it has two potential issues. First, the amount of RAM needed for the computation will increase. However the RAM is currently dominated by meta-data related to the phase-space integrator and not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is not to the amount of RAM used in the matrix element. The increase of memory needed by the helicity recycling is dominated by meta-data related to the phase-space integrator. This corresponds to (see Eq. (17)):\[ \mathcal{M}_{h_1h_2h_3} = \bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \phi_{h_3}. \] (19)

Where we have explicitly added indices representing the associated helicity (or helicity combination for internal wave functions) of each component. Now, when computing \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_4} \), the factor \(\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2} \) will be the identical and can therefore be re-used multiple times. Therefore we have been able to optimise the code further by implementing a recycling of this factor. However, a similar factor can also be recycled between \(\mathcal{M}_{h_1h_2h_3} \) and \(\mathcal{M}_{h_1h_2h_5} \). Therefore it is advantageous to compare (for each Feynman diagram) which one of the following expressions:

- \[\bar{\psi}_{1h_1} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{1h_1} \gamma_\mu \phi_{h_5}.\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \psi_{2h_2},\]
- \[\bar{\psi}_{2h_2} \gamma_\mu \phi_{h_5}.\]

The latter reductions are much larger because evaluating the matrix element represents a larger percentage of the overall computation for those processes. This is because the diagrams are more complex and numerous for \(tggg \) and \(tggg \).

Looking at table (2), we observe that both external and internal wavefuction routines represent, after helicity recycling, a relatively insignificant computational cost. First they were not that significant before the optimization and second they have been highly reduced by the helicity recycling (by at least 10x factor). The final speed-up is actually more dependent on the reduction in calls to amplitude routines. Focusing on the \(tggg \) process, one can see the amplitude routines have

6These vanishing terms are determined during the initial sampling of the matrix element.
Table 2: The same as Table [1] but this time with helicity recycling enabled. We also present the percentage reduction between the two tables.

|                  | $gg \to tl$ | $gg \to t\bar{g}g$ | $gg \to t\bar{g}gg$ | Instructions | Reduction | Instructions | Reduction | Instructions | Reduction |
|------------------|-------------|---------------------|---------------------|---------------|-----------|---------------|-----------|---------------|-----------|
| madevent         | 11G         | 15%                 | 180G                | 62%           | 5T        | 55%           |
| matrix1          | 1G (9.3%)   | 68%                 | 160G (90%)          | 64%           | 4.9T (98%)| 55%           |
| $L_{\text{ext}}$| 76M (<1%)   | 83%                 | 100M (<1%)          | 97%           | 110M (<1%)| 98%           |
| $L_{\text{int}}$| 540M (4.8%) | 72%                 | 16G (8.9%)          | 90%           | 180G (3.6%)| 91%           |
| $L_{\text{amp}}$| 280M (2.6%) | 47%                 | 77G (42%)           | 63%           | 1.7T (33%)| 69%           |

seen a large reduction in the number of associated instructions (by a factor of two) but still represent 46% of the overall computation. Although not shown in this table, roughly half of this computation (19% of the total) is spent evaluating simply scalar products (the contraction of Eq. (20-22) with the remaining wave function) which strongly limit the hope of further optimisation.

For the three gluon final state the situation is similar, even if the reduction of amplitude routine instructions is closer to a factor of 4. However, for this process the limiting factor is now the computation of the colour factor (taking around 60% of the computation). We have also investigated how that step could be optimised and introduced two simple improvements of the code. First we use a common sub-expression reduction algorithm [11] on that segment of the code. Second we merge the numerator and denominator of the colour-matrix into a single matrix, reducing the number of operations and allowing for a better memory pattern. Combined together those modifications lead to a speed-up of around 20%.

### 2.3.2 Overall speed-up

Having looked at how the computational cost of madevent breaks down into different functions, we now present its overall factor speed increase. This is shown for a range of processes in table (3). If $t_{\text{with}}$ and $t_{\text{without}}$ are the times it takes to run madevent with and without helicity recycling respectively then the speed-up is represented as

$$ \text{speed-up} = \frac{t_{\text{with}}}{t_{\text{without}}} $$

As has already been alluded to in tables (1) and (2), the speed-up we gain from helicity recycling is highly process dependent. Helicity recycling reduces the number of times we must calculate wave functions and amplitudes and so processes with more complicated diagrams and with a higher number of total diagrams see the biggest boost. For example, the $gg \to tl$ results shown in table (3a). As more gluons are added to the final state the gain increases, with the $t\bar{g}g$ final state seeing a dramatic $2.27 \times$ speed increase.

In contrast to this large increase the similar process $qg \to t\bar{qq}$ (where $q \in \{u, d, c, s\}$) sees a noticeably lower speed increase of $1.27 \times$. This is because processes involving fermions have a higher number of non-contributing helicity combinations and so helicity filtering will have a bigger effect. Hence, there will be fewer wave functions/ampitudes to evaluate and so helicity recycling will have a smaller impact.

One can see that the other processes presented in table (3) also follow these general principles regarding diagram complexity and fermion multiplicity. $W$ bosons allow for minimal helicity filtering and so table (3b) displays a large speed increase, whereas electrons - being fermions - suppress the speed-up in table (3c).

In table (3) we present these results for a wider range of processes and with a more detailed breakdown. In the ‘hel’ column we present how many helicity combinations are evaluated per phase-space point. In this column we use a ‘$\rightarrow$’ to indicate that the maximum value across the various matrix element is being shown. The columns “survey” and “refine” present the timing of the two main phases of the phase-space integration/event generation. The “survey” is designed to get a first (un-precise) estimate of the cross-section and to know the relative importance of each contribution. The amount of time spent in the “survey” is therefore independent of the number of requested events. The “refine” is targeting to generate the number of requested events and therefore scale linearly with the number of events. In both cases, the timing is the same for the ‘early’ with the number of events. In the column we present the timing of the two main phases of the phase-space integration/event generation. The “survey” is designed to get a first (un-precise) estimate of the cross-section and to know the relative importance of each contribution. The amount of time spent in the “survey” is therefore independent of the number of requested events. The “refine” is targeting to generate the number of requested events and therefore scale linearly with the number of events. In both cases, the timing is the same for the ‘early’ with the number of events.

Again one can see the same principles at play: the biggest speed increase is seen for complex processes dominated by QCD and vector boson interactions as they have the most non-vanishing helicity combinations and diagrams. The disparity of the gain between “survey” and “refine” can be explained due to different relative importance of the various matrix element in each steps.

Notice that helicity recycling’s tendency to more greatly speed up more intensive processes is very convenient. For example, in a $pp \to 4j$ simulation, the processes with $j = q$ will heavily dominate the calculation and so speeding those up.
is the most effective way to speed up the overall simulation. This is why the simulation sees an impressive 2.1× speed-up in table (1).

3 Phase-Space integrator

3.1 Monte-Carlo Integration and Single Diagram enhancement

In addition to the speed of the matrix-element evaluation, the efficiency of the phase-space integrator is another crucial factor determining the speed of the computation since it controls how many times we need to evaluate the matrix element. Due to both the number of dimensions of integration and also the requirement to generate uncorrelated events, the only suitable method is Monte-Carlo integration. However, the convergence of the method is quite slow (1/√N, where N is the number of times the matrix element is evaluated).

In Monte-Carlo methods (see [12] for a nice review), one evaluates the integrand for random values of the variable of integration. The estimator of the integral (IN) is simply given by the average of the function estimated across these random points x_i:

\[ \int f(x) \approx I_N = \frac{1}{N} \sum_{i=1}^{N} f(x_i). \]  

(23)

The statistical error (\Delta I_N) of this estimator I_n is controlled by the variance of the integrand and can be estimated as

\[ \Delta I_N = \frac{\text{Var}_f}{\sqrt{N}} \approx \frac{1}{\sqrt{N}} \sum_{i=1}^{N} f^2(x_i) - I_N^2. \]  

(24)

Since the convergence rate is fixed to 1/√N, the various avenues of optimisation consist in modifying the function being integrated to effectively reduce its variance. In MG5aMC, we use the single diagram enhancement method [20], which combines various importance sampling methods – both analytical [18] and numerical [43][44] – on top of a multi-channel strategy. The integral is decomposed as a sum of various contributions:

\[ \int |M|^2 = \sum_i \int |M_i|^2 \frac{|M|^2}{\sum_j |M_j|^2}, \]  

(25)

\[ = \sum_i \int \alpha_i |M|^2, \]  

(26)

where the indices i and j range over individual (or potentially a subset of) Feynman diagram. The values \( \alpha_i \equiv \frac{|M_i|^2}{\sum_j |M_j|^2} \) are called the channel weights, they do not modify the value of the integral (as long as \( \sum_i \alpha_i = 1 \) for every phase-space point) but do impact the variance of the integrand and therefore the speed of the computation.

While in general the value \( \alpha_i \) could be any arbitrary function, the single diagram enhancement method makes a specific choice a-priori. This choice is particularly motivated by the classical limit where interference terms between the Feynman diagrams are small. In that case

\[ \frac{1}{\sum_j |M_j|^2} \approx 1, \]  

(27)

and therefore we have

\[ \int |M|^2 = \sum_i \int |M_i|^2 \frac{|M|^2}{\sum_j |M_j|^2} \approx \sum_i \int |M_i|^2. \]  

(28)

In other words, with this choice the assumption is that each of the terms of the sum –called channels of integration– are mainly behaving as a single diagram squared from which the poles are easily identifiable and importance sampling is relatively easy to implement (see section (3.2)). The role of machine learning algorithm (MG5aMC use a modified VEGAS algorithm [43]) is then mainly to catch the impact of \( \frac{|M_j|^2}{\sum_j |M_j|^2} \) term as well as other source of deviation from the ideal case (like for example the impact of generation cuts or the impact of the running of the strong coupling).

While this method works extremely well in general, it is not the most suited approach for VBF-like processes, especially at high energy where large interference occurs due to gauge symmetry. As a matter of fact, MG5aMC is much slower than many other programs (in particular VBFNLO [45]) for such types of generation.

3.2 t-channel strategy

When running MG5aMC, one can easily identify that the slowest channels of integration are the ones with multiple t-channel propagators. In MG5aMC, we handle t-channel propagators with the following change of variable/phase-space

\[
\begin{array}{c|c|c|c|c}
\text{Process} & \text{Speed up} & \text{Process} & \text{Speed up} & \text{Process} & \text{Speed up} \\
\hline
gg \rightarrow t\bar{t} & 1.36 \times & qq \rightarrow W^+W^- qq & 1.67 \times & qq \rightarrow e^+e^- & 1.02 \times \\
gg \rightarrow t\bar{t}g & 1.43 \times & qq \rightarrow W^+W^- gg & 1.89 \times & qq \rightarrow e^+e^- g & 1.03 \times \\
gg \rightarrow t\bar{t}gg & 2.27 \times & gg \rightarrow W^+W^- qq & 1.89 \times & gg \rightarrow e^+e^- qq & 1.09 \times \\
qq \rightarrow t\bar{t}qq & 1.27 \times & \end{array}
\]

(a) A selection of \( t\bar{t} \) processes.

(b) \( pp \rightarrow W^+W^- jj \) processes.

(c) A selection of \( e^+e^- \) processes.

Table 3: The factor speed up of madevent as a result of using helicity recycling for a selection of SM processes. Here \( q \in \{ u, d, c, s \} \). One can see two general principles at play. Firstly, the more diagrams a process has and the more complicated those diagrams are, the bigger the speed increase. For example, see the effect of adding more gluons in (a). Secondly, the more helicity combinations we filter away, the lower the speed increase. This is why processes with more fermions see a lower speed up.
| process | hel   | 2.9.0 nohel | 2.9.0 speed-up |
|---------|-------|-------------|----------------|
| pp → W^+W^- jj [gS = 0] | 9     | 21s         | 1.5x           |
| pp → W^+W^- jj, W → lv [gS = 0, 0.13 TeV] | 6     | 10m0s       | 1.4x           |
| pp → W^+W^- jj, W → lv [gS = 0, 0.10 TeV] | 6     | 7m0s        | 1.4x           |
| ud → W^+W^- L ud [gS = 0] | 4     | 17s         | 1.6x           |
| ud → W^+W^- L ud, W → dū, W^- → τ^+ν_τ [gS = 0] | 8     | 1m0s        | 1.7x           |
| ud → W^+W^- L ud, W^- → dū, W^- → τ^+ν_τ [gS = 0] | 8     | 40s         | 1.1x           |
| μ^+ μ^- → hhś uν_μ [1.4 TeV] | 1     | 1s          | 1.1x           |
| μ^+ μ^- → W^- μ^- W^-[13 TeV] | 64    | 23s         | 4.4x           |
| μ^+ μ^- → W^- μ^- [4 TeV] | 122   | 1m0s        | 3.8x           |

Table 4: Integration timing (time to solution) on a mac-book pro (quad-core 2016) to generate 10k events. Comparing the impact with and without helicity recycling.

\[
p \quad k_1 \quad t_1 \quad k_2 \quad t_2 \quad k_3 \quad \vdots \quad k_{n-1} \quad t_{n-1} \quad k_n \quad q
\]

Fig. 3: Feynman diagram used to fix the convention in the variable.

The t-channel propagator given by \( t_1 = (p - k_1)^2 \) and where \( \lambda(a, b, c) = a^2 + b^2 + c^2 - 2ab - 2ac - 2bc \).

The integration over \( t_1 \) is bounded by the following values:

\[
t_1^+ = m_p^2 + m_T^2 - \frac{1}{2σ} \left( (S + m_p^2 - m_T^2) (S + m_T^2 - m_b^2) \right)
\]

\[
\lambda^2 (S, m_p^2, m_b^2, k_0) \lambda^2 (S, m_b^2, m_T^2) \right),
\]

In the presence of multiple t-channel propagators, MG5aMC writes them as

\[
t_1 = (p - k_1)^2
t_2 = (p - k_1 - k_2)^2
\]

\[
\ldots
\]

\[
t_{n-1} = (p - k_1 - k_2 - \ldots - k_{n-1})^2
\]

and performs the integration in the following (ordered) way

\[
\int dt_{n-1} \int dt_{n-2} \cdots \int dt_1,
\]

meaning that we first integrate over \( t_1 \), then \( t_2 \) and continues up to \( t_{n-1} \).

The combination of such ordering with the boundary condition of Eq. (31) creates correlations between the variables of integration. Such type of correlation are problematic for any type of VEGAS-like algorithm. The importance of the ordering can be seen in figure (4) where we compare various orderings for a given channel of integration (the associated diagram is displayed in the figure). We present both the un-weighting efficiency and the estimated statistical uncertainty after three and five iterations which gives a hint on the convergence of the integration grid.

In this example, it is clear that the original ordering strategy picked by MG5aMC is sub-optimal (between 3 and 6 times slower than the optimal strategy depending of the number of iterations). Not only is the best strategy more efficient
at generating events but also the associated grid seems to need fewer iterations to converge.

While we hand-picked an example of channel of integration where the ordering was sub-optimal, it would not have been difficult to also present cases where it was optimal (e.g. just flip the initial state for the example). Indeed the optimal ordering is deeply dependent on the channel of integration under consideration.

On top of the old ordering, we have added support for three additional orderings (see details in Appendix A). Each channel of integration is then associated to one of those orderings such that the virtuality of the most singular $t$-channel propagator is integrated first. In principle one could use the values of $t^i$ (Eq. 51) to choose such an ordering, but this is technologically not possible and we used a simple heuristic approach for such determination.

Our resulting timing is presented in table (5) which contains the same type of information as table (4). The comparison is between the version of the code containing all the optimisations of this paper (2.9.0) with the same version of the code where the ordering strategy was forced to the old one (Eq. 55). While in general the “survey” presents only a small speed-up (see table (5)) a more sizeable gain is achieved during the “refine”. The actual speed-up will therefore slightly increase when requesting more events. Additionally, one can note that the biggest gain are achieved for the slowest processes.

One processes (VBF with transverse polarised W boson) shows a significant slow-down (1.6 times slower). Our investigation shows that the ordering picked in that case was correct but the convergence of the grid was actually slower than what it was for the previous strategy leading to the slow-down. Such an issue can only happen for relatively easy channels of integration since most complex processes would need more iteration and then this effect would disappear (as observed on the VBF process at 100 TeV).

| order | unweighted events | relative precision | comment |
|-------|-------------------|--------------------|---------|
| 3 iterations/20k evaluations | | | |
| $\int dt_3 \int dt_2 \int dt_1$ | 161 (1.8%) | 2.9% | old default |
| $\int dt_1 \int dt_2 \int dt_3$ | 1034 (5.0%) | 1.4% | new default |
| $\int dt_2 \int dt_3 \int dt_1$ | 564 (2.8%) | 2.2% | |
| $\int dt_2 \int dt_1 \int dt_3$ | 194 (1.0%) | 2.8% | |
| 5 iterations/ 80k evaluations | | | |
| $\int dt_3 \int dt_2 \int dt_1$ | 1703 (2.1%) | 1.0% | old default |
| $\int dt_1 \int dt_2 \int dt_3$ | 5421 (6.8%) | 0.6% | new default |
| $\int dt_2 \int dt_3 \int dt_1$ | 3636 (4.5%) | 0.8% | |
| $\int dt_2 \int dt_1 \int dt_3$ | 4224 (5.3%) | 0.8% | |

Fig. 4: Comparison of various ordering of the three variable of integration corresponding to the invariant of time-like particles for the channel associated to the Feynman diagram represented on the left. We present both the relative error, the number of events generated at a given iteration and the associated un-weighting efficiency.

3.3 New Diagram enhancement strategy

A recent paper [49] pointed to the importance of the gauge choice when generating events within MG5aMC. Even if the amplitude is fully gauge invariant, the definition of the channel weights ($\alpha_i$) is not. The presence of large gauge cancellations are related to large interference term and therefore the assumptions used by the phase-space integrator (Eq. (27)) will not hold anymore. Consequently, the $\alpha_i$ associated with Feynman diagrams with $t$-channel propagators will be artificially large at high-energy. This will increase the variance of the function and reduce the efficiency of the method.

We propose here, a second strategy for the single diagram enhancement method. Instead of using Eq. (25), we replace the $|M|^2$ by the product of the denominator (both $s$ and $t$ channel) associated to the single diagram under consideration (and normalise them as needed).

$$\alpha_i \equiv \prod_{k \in \text{propag}} \frac{1}{|p_k^2 - M_k^2 - iM_k \Gamma_k|^2},$$

$$\alpha_i = \frac{\alpha_i}{\sum_j \alpha_j}. \tag{35}$$

Such a change in the definition of the multi-weights is in principle not impacting the cross-section (see Eq. (26)). However in practise the choice of the dynamical scale is done on a CCKW inspired clustering [29] which depends on the Feynman diagram selected by the single diagram enhancement method. This modification of the channel weight will therefore impact the scale associated to each event and therefore the cross-section and shape, both within scale uncertainty. In general this can be avoided -if needed- by changing the scale computation to $H_{T}/2$ or other pure kinematical scale choice [50]. However, one should note that it is not possible to avoid such effects when running matched/merged generation within MLM or shower-kt MLM [51,52], since the CKKW clustering is mandatory in those type of generation.

A more subtle effect of the modification of the channel weight is related to the parton-shower. When writing an event inside the output file MG5aMC provides, in addition to the kinematical variables, the colour dipole representation in the leading colour approximation. The determination of the eligible dipole depends on the Feynman diagram (e.g. in presence of mixed expansion) and therefore the modification of the multi-channel strategy can impact such a cesselection. One can therefore expect some change, within theoretical uncertainties, after parton-shower for some QCD related observable.

In table (6), we compare the time needed to generate ten thousands events with the previous strategy and the one introduced in this paper (all other optimisations of this paper are included in both cases). As one can expect for such deep modification of the phase-space integration strategy, the observable spectrum of speed-up/down is extremely broad going
| process | 2.9.0 old ordering | 2.9.0 new | speed-up |
|---------|-------------------|----------|----------|
| $pp \to W^+W^+j$ [$g_s = 0$] | 17s | 7m1s | 14s | 7m28s | 1.1x |
| $pp \to W^+W^-jj$, $W \to ll[|g_s| = 0.13 \text{ TeV}]$ | 8m0s | 1m30s | 7m0s | 1m29s | 1.1x |
| $pp \to W^+W^-jj$, $W \to ll[|g_s| = 0.100 \text{ TeV}]$ | 6m0s | 1h0m/3821 | 5m0s | 1m26s | 3.1x |
| $ud \to W^+_L W^-_L$ [$g_s = 0$] | 12s | 4m23s | 11s | 1m36s | 2.6x |
| $ud \to W^+_L W^-_L$ $ud, W^- \to d\bar{u}, W^- \to \tau^+\nu_\tau[g_s = 0]$ | 1m0s | 24m19s | 1m0s | 4m27s | 4.6x |
| $ud \to W^+_L W^-_L$ $ud, W^- \to d\bar{u}, W^- \to \tau^+\nu_\tau[g_s = 0]$ | 33s | 1m9s | 33s | 2m6s | 1.6x |
| $\mu^+\mu^- \to hh\bar{h}_1\nu_\tau[14 \text{ TeV}]$ | 1s | 29s | 1s | 10s | 2.7x |
| $\mu^+\mu^- \to tt\mu^+\mu^-[13 \text{ TeV}]$ | 7s | 24s | 5s | 22s | 1.1x |
| $\mu^+\mu^- \to W^+W^-\mu^+\mu^-[4 \text{ TeV}]$ | 13s | 15s | 14s | 14s | 1.0x |

**Table 5:** Integration timing (time to solution) on a mac-book pro (quad-core 2016) to generate 10k events depending on the t-channel ordering strategy. If the generation fails to generate enough events the timing is rescaled accordingly. In such case the timing is set in bold and the number of events actually generated is then indicated. Timing in red means an actual slowdown of the new strategy compare to the previous one.

| process | old strategy | new strategy | speed-up | default |
|---------|--------------|--------------|----------|---------|
| $pp \to W^+W^+j$ [$g_s = 0$] | 13s | 2h12m/1290 | 16s | 8m1s | 16x | new |
| $pp \to W^+W^-jj$, $W \to ll[|g_s| = 0.13 \text{ TeV}]$ | 19m0s | 9m6s | 10m0s | 1m43s | 2.4x | new |
| $pp \to W^+W^-jj$, $W \to ll[|g_s| = 0.100 \text{ TeV}]$ | 10m0s | 24m8s | 7m0s | 18m10s | 1.4x | new |
| $ud \to W^+_L W^-_L$ $ud, W^- \to d\bar{u}, W^- \to \tau^+\nu_\tau[g_s = 0]$ | 23s | 27h56m/203 | 14s | 1m53s | 792x | new |
| $ud \to W^+_L W^-_L$ $ud, W^- \to d\bar{u}, W^- \to \tau^+\nu_\tau[g_s = 0]$ | 2m0s | 15h52m/793 | 1m0s | 5m42s | 142x | new |
| $ud \to W^+_L W^-_L$ $ud, W^- \to d\bar{u}, W^- \to \tau^+\nu_\tau[g_s = 0]$ | 36s | 2m54s | 37s | 2m28s | 1.1x | new |
| $\mu^+\mu^- \to hh\bar{h}_1\nu_\tau[14 \text{ TeV}]$ | 3s | 8h50m/641 | 1s | 11s | 2653x | new |
| $\mu^+\mu^- \to tt\mu^+\mu^-[13 \text{ TeV}]$ | 20s | 3h6m/948 | 6s | 25s | 362x | new |
| $\mu^+\mu^- \to W^+W^-\mu^+\mu^-[4 \text{ TeV}]$ | 1m0s | 33m26s | 16s | 15s | 66x | new |

**Table 6:** Integration timing (time to solution) on a mac-book pro (quad-core 2016) to generate 10k events depending of the multi-channel strategy. If the generation fails to generate enough event the timing is re-scaled accordingly. In such case the timing is set in bold and the number of events actually generated is then indicated. Timing in red means an actual slowdown of the new strategy compare to the previous one.
from three orders of magnitude speed-up to five times slower. It is clear that such an optimisation is a must have in the context of VBF processes but must be avoided for most QCD multi-jet processes. While the user can easily switch from one strategy to the other (see Appendix A), we have configured the code such that the default value is process dependent. All processes with only one colour-flow will use the new method while others processes will use the old method. We made and exception for pure multi-jet processes which are using the new method as well. The default for each processes is indicated as the last column in table (6). Since for most QCD processes we keep the previous integration strategy, the caveat on the change in the scale/leading colour choice, mentioned above, are naturally mitigated.

3.4 Comparison with older version of MG5aMC

In table (7), we compare the speed of the code between two versions of MG5aMC (2.8.1 and 2.9.0). 2.9.0 is the first version of the code containing the modification described in this paper. Let’s stress that every optimisation flag are set on their default value and therefore this is the speed-up that a user will observe without playing with any options.

The combined impact of all our optimisations is striking for VBF-like processes with a speed-up of more than 30,000 times faster for one process. While this process is very specific and probably not the most important one for many users, all the VBF processes show massive speed-up passing from hour long runs to a couple of minutes. Actually, in many cases, the previous version of the code had a lot of trouble in generating the requested number of events and sometimes event to correctly converge through the correct cross-section. All those problems are now solved with 2.9.0; the cross-section converges quickly and can generates events very efficiently.

Gain for the other processes is more modest. Firstly because the phase-space integration was much better handled to start with and secondly because those processes are less sensitive to t-channel diagrams on which we have focused. Nevertheless in combination with the helicity recycling, the code is, for processes heavily used at the LHC, around three times faster, a very valuable gain.

4 Conclusion

In order to evaluate an amplitude MG5aMC must sum it over all contributing helicity combinations. Before the work of this paper MG5aMC would calculate every wave function and amplitude separately for each helicity combination. We have successfully restructured the code so it will now only calculate a given wave function once and then reuse the output for all the different helicity combinations. We have also been able to split up the amplitude calculation such that part of it can be recycled across different helicity combinations. This restructuring of the code has also allowed us to avoid calculating diagrams which for a given helicity combination are null. All those optimisations mean that for complex processes with only colour-flow we can see a speed-up of the code of around 2x. At the other end of scale, simple processes dominated by fermions can see a speed-up of only a few percent.

Additionally, we have studied the efficiency issue of MG5aMC for VBF-like processes at high energy. We have identified that modifying the order of integration of the virtuality of t-channel particles and changing the multi-channel weight was highly valuable providing game-changing speed-up for such computation. Fixing a lot of the issue faced in the past for such processes.

Combining all those optimisations allows us to overshoot the target speed-up asked by the HSF community since we provide a code at least three times faster for the cpu intensive processes and typically much higher than that for VBF processes.

Optimisation is important for code which are massively used by LHC experiment and are run a non negligible amount of time on the grid/local cluster. We believe that this paper is a significant milestone providing significant speed improvement both in the time to evaluate a phase-space point and to the phase-space integrator. However this is certainly not the end of the road and this effort has to (and will) continue. First, the techniques developed of this paper needs to be applied to the NLO processes within MG5aMC. We do not expect any big difficulties in such porting and expect similar gain in speed. Second, they are still room to optimise the evaluation of the matrix element, even at Leading-Order: Work are in progress to have a tighter link to the hardware with investigation on a GPU port but also on the use of SIMD operation on CPU.

Appendix A: Manual

When generating a LO process within MG5aMC, the optimisation described in this paper will be activated by default. When activating the feature, you need to set the following variables in the configuration file. Most of those parameters are not present by default in that card since we consider them as advanced parameters. It is enough to add them in the file when needed. The only parameter present by default is the one allowing to choose the multi-channel strategy.

In table (8), we briefly present the various parameter that can be modified in that way. One should notice that the parameter “hel_recycling” is a switch that forbids to use the helicity recycling, therefore when set to False, the other parameters related to helicity recycling (prefixes with “hel_”) will be without any impact.

It is also possible to ask MG5aMC to generate a code without any possibility to use helicity recycling. This allows to have a code closer to the previous version and avoids to spend time to generate the new algo routine. To do this one needs to modify the “output” command and add the flag “--hel_recycling=False”. For example

```
1 generate p p > t t ~ 3j
2 output MYDIR --hel_recycling=False
```

Another new optional flag for the “output” command allows to control the ordering of the variable of integration corresponding to the invariant mass of t-channel propagators. For example:

```
1 generate p p > W+ W- j j QCD=0
2 output MYDIR --t_strategy=X
```

The possible values and the associate meaning is described below. A concrete example for the ordering of the Feynman diagram represented in Fig. 4 is also given to provide a simpler comparison:

- 0 [default] Automatically decide based on the diagram.
### Table 7: Integration timing (time to solution) on a mac-book pro (quad-core 2016) to generate 10k events. If in 2.8.1, it was not possible to generate 10 thousands events, the time is re-scaled accordingly (and the timing is set in bold). The second number presented in that case is the actual number of events that was generated.

| process | Option controlling helicity recycling | Option controlling phase-space integration | Option controlling advanced compilation flag |
|---------|--------------------------------------|-------------------------------------------|--------------------------------------------|
| pp → W⁺W⁻ jj | | | |
| pp → W⁺W⁻ j,j, W → ℓνℓ [gs = 0, 13 TeV] | | | |
| pp → W⁺W⁻ j,j, W → ℓνℓ [gs = 0, 100 TeV] | | | |
| ud → W⁺L⁺W⁻L⁻ ud, W⁺ → ð, W⁻ → τ⁻ντ [gs = 0] | | | |
| ud → W⁺L⁺W⁻L⁻ ud, W⁺ → ð, W⁻ → τ⁺ντ [gs = 0] | | | |
| μ⁺μ⁻ → hhh, µ⁻ [14 TeV] | | | |
| μ⁺μ⁻ → W⁺W⁻ μ⁺μ⁻ [13 TeV] | | | |
| μ⁺μ⁻ → W⁺W⁻ μ⁺μ⁻ [4 TeV] | | | |
| pp → W⁺[0 − 4]j | | | |
| pp → ℓl[j0 − 2]j | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
| pp → ℓl | | | |
generation of the Feynman diagram by MG5aMC, MG5aMC request Aloha to generate the HELAS function needed for the associated computation.

In the context of helicity recycling a new type of helicity routines has been introduced (Eq. (20-21)). Contrary to the other types of routines, MG5aMC does not know at generation time which of those functions will be effectively used. Consequently MG5aMC request Aloha to generate all possible choices (so in general three routines) such that any strategy can be picked if relevant. The implementation strategy is to ask Aloha to generate a standard internal wave-function routine but with a custom propagator. Consequently this was possible thanks to previous extension of Aloha adding support for custom propagators [54,55]. The definition for such propagators are:

\[
\text{scaler} : 1, \quad (B.1) \\
\text{fermion} : \delta_{ij}, \quad (B.2) \\
\text{vector} : \eta^{\mu\nu}. \quad (B.3)
\]

The reason of the presence of a metric term for the vector propagator allows to not include the metric in the final scalar product and therefore have a code which is easier for the compiler to optimise (giving the possibility to use some SIMD instructions) which can be critical since a large part of the computation is spent in such simple scalar product (\(\approx 20\%\) for \(gg \rightarrow Hgg\)).

Concerning the Aloha naming scheme convention, such a routine will have a suffix “PIN”. So for the following Lorentz structure (which correspond to a \(\gamma_5\) lorentz structure [56]):

\[
\text{FFV1 = Lorentz (name = 'FFV1P1N_2', } \\
\text{spins = [2, 2, 3], } \\
\text{structure = 'Gamma(3,2,1)' })
\]

The three new expression (20-21) will have the following name/definitions:

\[
\text{FFV1P1N}_1 : \gamma_\mu \psi^i_2 \phi^i_3, \quad (B.4) \\
\text{FFV1P1N}_2 : \psi^i_1 \gamma_\mu \phi^i_3, \quad (B.5) \\
\text{FFV1P1N}_3 : \psi^i_1 \gamma_5 \psi^i_2 \eta^{\mu\nu}. \quad (B.6)
\]

Acknowledgements The authors would like to thanks Fabio Maltoni, Mike Seymour, Richard Ruiz, Luca Mantani, Andrew Lisson, Andrea Valassi, Stefan Roiser and all MG5aMC authors (past and present) for useful discussions. We would also like to thank the Université catholique de Louvain staff for working around the limitations imposed by the Covid-19 pandemic. This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824093. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CECI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

References

1. T. Aarrestad, et al., in, 2021 Snowmass Summer Study, ed. by P. Canal, et al. (2020). DOI 10.5281/zenodo.4009114
2. J. Albrecht, et al., Comput. Softw. Big Sci. 3(1), 7 (2019). DOI 10.1007/s41781-018-0018-8
3. 2020 Update of the European Strategy for Particle Physics (CERN Council, Geneva, 2020). DOI 10.17181/ESU2020
4. J. Almejdorj, A. Lisson, C. Reuschle, M. Sjodahl, (2020)
5. F. Maltoni, K. Paul, T. Stelzer, S. Willenbrock, Phys. Rev. D 67, 014026 (2003). DOI 10.1103/PhysRevD.67.014026
6. F.A. Berends, W. Giele, Nucl. Phys. B 306, 759 (1988). DOI 10.1016/0550-3213(88)90442-7
7. F.A. Berends, W. Giele, H. Kuijf, Nucl. Phys. B 321, 39 (1989). DOI 10.1016/0550-3213(89)90242-3
8. C. Gao, S. Höche, J. Isaacson, C. Krause, H. Schulz, Phys. Rev. D 101(7), 076002 (2020). DOI 10.1103/PhysRevD.101.076002
9. J. Bendavid, (2017)
10. E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, S. Schumann, SciPost Phys. 8(4), 069 (2020). DOI 10.21468/SciPostPhys.8.4.069
11. M.D. Klimk, M. Perelstein, SciPost Phys. 9, 053 (2020). DOI 10.21468/SciPostPhys.9.4.053
12. M. Backes, A. Butter, T. Plehn, R. Winterhalder, (2020)
13. K. Hagiwara, J. Kanzaki, Q. Li, N. Okamura, T. Stelzer, Eur. Phys. J. C 73, 2608 (2013). DOI 10.1140/epjc/s10052-013-2608-2
14. K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater, T. Stelzer, Eur. Phys. J. C 70, 513 (2016). DOI 10.1140/epjc/s10052-016-4456-5
15. K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater, T. Stelzer, Eur. Phys. J. C 66, 477 (2010). DOI 10.1140/epjc/s10052-010-1276-8
16. D. Benjamin, J. Childers, S. Höche, T. LeCompte, T. Uram, J. Phys. Conf. Ser. 899(7), 072044 (2017). DOI 10.1088/1742-6596/899/7/072044
17. K.T. Matchev, P. Shyamandar, (2020)
18. B. Nachman, J. Thaler, Phys. Rev. D 102(7), 076004 (2020). DOI 10.1103/PhysRevD.102.076004
19. R. Frederix, S. Frixione, S. Prestel, P. Torrielli, JHEP 07, 238 (2020). DOI 10.1007/JHEP07(2020)238
20. O. Mattelaer, Eur. Phys. J. C 76(12), 674 (2016). DOI 10.1140/epjc/s10052-016-4533-7
21. J.R. Andersen, C. Gütschow, A. Maier, S. Prestel, Eur. Phys. J. C 80(11), 1007 (2020). DOI 10.1140/epjc/s10052-020-08548-w
22. H. Brooks, C.T. Preus, (2020)
23. J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.S. Shao, T. Stelzer, P. Torrielli, M. Zaro, JHEP 07, 079 (2014). DOI 10.1007/JHEP07(2014)079
24. J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, T. Stelzer, JHEP 06, 128 (2011). DOI 10.1007/JHEP06(2011)128
25. S. Frixione, P. Nason, C. Oleari, JHEP 11, 070 (2007). DOI 10.1088/1126-6708/2007/11/070
26. E. Bothmann, et al., SciPost Phys. 7(3), 034 (2019). DOI 10.21468/SciPostPhys.7.3.034
27. W. Kilian, T. Ohl, J. Reuter, Eur. Phys. J. C 71, 1742 (2011). DOI 10.1140/epjc/s10052-011-1742-y
28. H. Murayama, I. Watanabe, K. Hagiwara, (1992)
29. F. Maltoni, T. Stelzer, JHEP 02, 027 (2003). DOI 10.1088/1126-6708/2003/02/027
30. P. de Aquino, W. Link, F. Maltoni, O. Mattelaer, T. Stelzer, Comput. Phys. Commun. 183, 2254 (2012). DOI 10.1016/j.cpc.2012.05.004
31. A. Belyaev, N.D. Christensen, A. Pukhov, Comput. Phys. Commun. 184, 1729 (2013). DOI 10.1016/j.cpc.2013.01.014
32. P. De Causmaecker, R. Gastmans, W. Troost, T.T. Wu, Phys. Lett. B 105, 215 (1981). DOI 10.1016/0370-2693(81)91025-X
33. P. De Causmaecker, R. Gastmans, W. Troost, T.T. Wu, Nucl. Phys. B 206, 53 (1982). DOI 10.1016/0550-3213(82)90488-6
34. R. Gastmans, AIP Conf. Proc. 201, 58 (1990). DOI 10.1063/1.39098
35. R. Britto, F. Cachazo, B. Feng, Nucl. Phys. B 715, 499 (2005). DOI 10.1016/j.nuclphysb.2005.02.030
36. R. Britto, F. Cachazo, B. Feng, E. Witten, Phys. Rev. Lett. 94, 181602 (2005). DOI 10.1103/PhysRevLett.94.181602
37. M.D. Schwartz, Quantum Field Theory and the Standard Model (Cambridge University Press, 2014)
38. J.P. Ellis, Computer Physics Communications 210, 103–123 (2017). DOI 10.1016/j.cpc.2016.08.019. URL http://dx.doi.org/10.1016/j.cpc.2016.08.019
39. N. Nethercote, J. Seward, in Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation (Association for Computing Machinery, New York, NY, USA, 2007), PLDI ’07, p. 89–100. DOI 10.1145/1250734.1250746. URL https://doi.org/10.1145/1250734.1250746
40. J. Weidendorfer, in Tools for High Performance Computing, ed. by M. Resch, R. Keller, V. Himmel, B. Krammer, A. Schulz (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), pp. 93–113
41. D.E. Knuth, (1976)
42. S. Weinzierl, (2000)
43. G.P. Lepage, (2020)
44. J. Baglio, et al., (2014)
45. E. Byckling, K. Kajantie, Particle Kinematics: (Chapters I-VI, X) (University of Jyväskylä, Jyväskylä, Finland, 1971)
46. F. Maltoni, G. Ridolfi, M. Ubiali, JHEP 07, 022 (2012). DOI 10.1007/JHEP04(2013)095. [Erratum: JHEP 04, 095 (2013)]
47. K. Hagiwara, J. Kanazaki, K. Mawatari, Eur. Phys. J. C 80(6), 584 (2020). DOI 10.1140/epjc/s10052-020-8154-9
48. S. Catani, F. Krauss, R. Kuhn, B. Webber, JHEP 11, 063 (2001). DOI 10.1088/1126-6708/2001/11/063
49. V. Hirschi, O. Mattelaer, JHEP 10, 146 (2015). DOI 10.1007/JHEP10(2015)146
50. F. Caravaglios, M.L. Mangano, M. Moretti, R. Pittau, Nucl. Phys. B 539, 215 (1999). DOI 10.1016/S0550-3213(98)00739-1
51. J. Alwall, S. de Visscher, F. Maltoni, JHEP 02, 017 (2009). DOI 10.1088/1126-6708/2009/02/017
52. J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, X. Tian, in Data Parallel C++ (Apress, 2020), pp. 259–276. DOI 10.1007/978-1-4842-5574-2_11. URL https://doi.org/10.1007/978-1-4842-5574-2_11
53. N.D. Christensen, P. de Aquino, N. Deutschmann, C. Duhr, B. Fuks, C. Garcia-Cely, O. Mattelaer, K. Mawatari, B. Oexl, Y. Takea, Eur. Phys. J. C 73(10), 2580 (2013). DOI 10.1140/epjc/s10052-013-2580-x
54. D. Buarque Franzosi, O. Mattelaer, R. Ruiz, S. Shil, JHEP 04, 082 (2020). DOI 10.1007/JHEP04(2020)082
55. C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter, Comput. Phys. Commun. 183, 1201 (2012). DOI 10.1016/j.cpc.2012.01.022