MadGraph 4 at NLO

Valentin Hirschi

Contents

1 Introduction

MadGraph (MG) is a powerful tool to compute tree-level processes within very general user-defined models. It is soon to be upgraded to the fifth version, MG5, written in python. It will offer the user a better control on the processing and allow very general characteristics for the implemented model. Taking advantage of these new features, one of the main challenge of MG5 is to offer a computation of the different processes at Next-to-Leading-Order (NLO). This is a major improvement which introduces many new subtleties essentially related to the appearance of divergences in the one-loop diagrams and those where an unresolved particle is emitted.

Even when working at NLO, MadGraph formally stays a tree-diagram computer and uses external tools or modules to handle the two kinds of diagrams mentioned above. Many choices are available, but in the MadGraph case, CutTools[2] has been chosen to handle the one-loop diagrams, the *virtuals*, mainly because of its generality and MadFKS[5] (built within MG4) deals with the diagrams with an unresolved particle, the *reals*, using a innovating way of characterizing the divergences with what are called FKS pairs instead of the commonly used dipoles.

The MadGraph 4 framework, written in fortran, is not well-suited for organizing the computation involving these three quite independent codes and the upgrade to NLO requires some core changes in the structure of MadGraph. It is not clear yet what is exactly required to make such NLO computations possible, this is why it has been decided to compute some simple processes at NLO within the existing MadGraph 4 code to help understand what features are needed for the upcoming version MG5. In this context, a third-party program written in C++, NLOComp, coordinates the computation between MG4, CutTools and MadFKS.

The setup has been successfully tested (see Appendix 6.2) in the simplest case of e^+e^- to two jets and work is under progress for three jets. The plan of this note is as follows. In the first section, a general overview of the computation mechanism is given. The second section goes back to each step and individually describes them in details. The last section summarizes all the different requirements to perform NLO computations within MadGraph and how MG5 should ideally address them. A final word follows as a conclusion.

2 Description of the mechanism

2.1 Overview

The computations are split in four different parts rather independent from each other. First, the born cross-section is evaluated in the exact same manner as without the NLO upgrade. Then the NLO contribution is constituted of the reals and the interference terms between the virtuals and the born diagrams.

CutTools uses the Ossola Papadopoulos Pittau (OPP) method[1] to compute the virtuals in two steps : first the ϵ^{-2} and ϵ^{-1} poles computed by cutting the loop of the virtuals. The coefficient of these poles are then qualified as *cut-constructible*. From these cuts, a piece of the finite part of the amplitude can be derived, which is further referred to as R1, the cut-constructible contribution to the finite part of the amplitude (coming both from the rational part and the μ-logarithms part). Secondly the remaining of the rational part, called R2, is independently computed using tree-diagrams with modified Feynman rules.

MadFKS can then be treated as a self-contained black box which is fed with the virtual cross-section and adds the reals squared to give the total NLO cross-section.

2.2 Structure of the computation

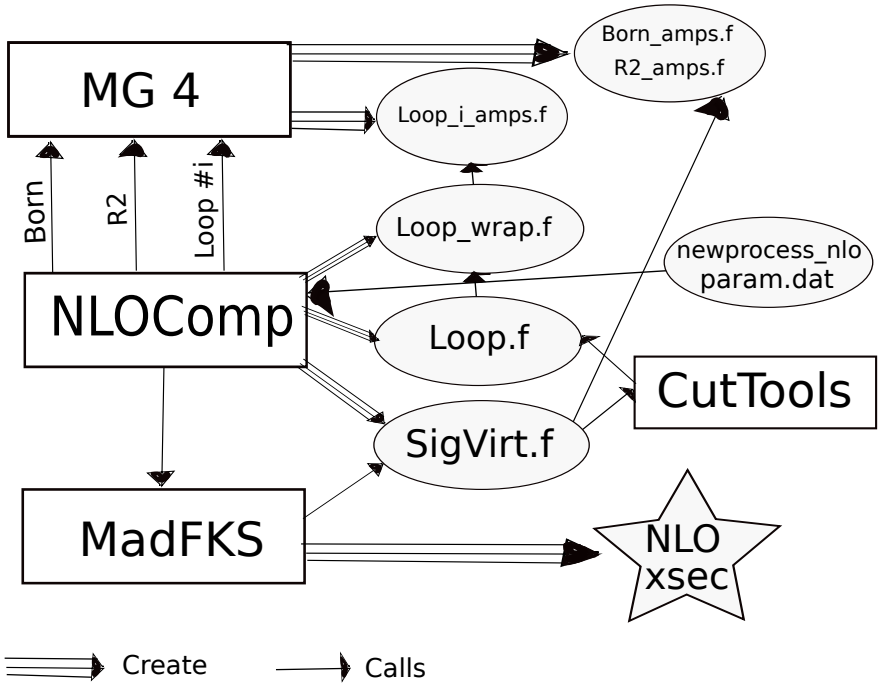

Figure 2.1: Scheme of NLO computations within MG4 framework

The diagram above 2.1 gives a good picture of the organization of the computation. The best way to explicit it is to follow its chronologic development. First NLOComp reads the `param.dat` card which contains all the information related to the NLO process to be computed (details in Appendix 6.1). It then builds a MadGraph 4 `param_card.dat` for the Born, the R2 part and for each of the particles running in the loop. There is a separate call to MG for each of them and the relevant files are copied from the MadGraph process folder to the NLOComp root folder. The core code of MadGraph has been modified so that it creates these four files providing the information and the code required by NLOComp:

- `<proc_name>_amps.f`

  This subroutine contains the HELAS calls for each diagram and is designed in such a way that it is possible to obtain the amplitude of a single specified diagram without computing the others (the standard MG optimization for the HELAS calls has been turned off). Also, the amplitudes given by this subroutine have the denominators of of the propagators of the loop lines (if any) taken out as CutTools requires. Note that here and below, `proc_name` refers to either `Born` , `R2` or `Loop_<i>` corresponding to each of the separate MG calls.

- `<proc_name>_colors.f`

  Give the full color information of the diagrams, essentially their components in the color-basis specified by MG. Note that because computing the color factor for the virtuals-born interference is unfeasible in MG4, this is entirely done within NLOComp.

- `<proc_name>_configs.f`

  Provides the topology information on the different diagrams of the process. This file is parsed by NLOComp which constructs an internal representation of them. For the tree-diagrams corresponding

to cut loops, MG generates each possible cut of the loops, so NLOComp has to select them to avoid any redundancy.

- `<proc_name>_diags.ps`

  Simply the tree-diagram draws automatically generated by MG which help diagnostics. Note that in MG5 the loop-diagrams will be effectively represented as loops and not tree cuts.

NLOComp parses the `<proc_name>_colors.f` and `<proc_name>_configs.f` to select the cut-loop diagrams (more details in 3.4) and computes the color factors for each pairing of a virtual with a born. It then creates the master routine in `SigVirt.f` which calls CutTools which in turn uses the `Loop_<i>_amps.f` files to return the amplitude of the loop diagrams. SigVirt adds to the cut-constructible part of the rational the R2 part computed in `R2_amps.f`. With the help of `Born_amps.f`, SigVirt can now square the virtuals against the born diagrams with the appropriate color factors and perform the helicity sum to give back the virtual cross-section σ_{virt} for a given phase-space point and renormalization scale.

As a final step, NLOComp creates the `proc_card.dat` for MadFKS and copies it along with `SigVirt.f` and the other relevant files into the appropriate MadFKS subdirectories. The user can now do the Monte-Carlo to compute the integrated NLO cross-section as usual within MadFKS but having now the virtual cross-section properly taken into account.

3 Technical details

3.1 Diagrams generation

3.1.1 Born diagrams

The borns are generated exactly as for leading order computations. The HELAS calls present in `Born_amps.f` are the same as those in the standard `matrix.f` file except that the optimization that prevent from reconstructing all of the external wave-functions for each diagram has been turned off so that it is possible to ask for the amplitude of a single given diagram without computing all the others. Note that the subroutine names in all `<proc_name>_amps.f` files are of the form `AMP<proc_name>(P,QP,NHEL,IC,DID,ANS)` where `proc_name` is a four-letter tag that can be specified in `proc_card.dat`. As for the arguments, here only `P` (External momenta), `NHEL` (Helicities), `DID` (Diagram ID) and `ANS` (Amplitude result) matter since `IC` is a dummy array in all cases and `QP` only serves for specifying the *complex* momenta for the cut loop amplitudes.

3.1.2 Loop diagrams

The OPP method used by CutTools is based on cutting[1] a loop diagram. The resulting tree-diagram is then computed many times for different complex momenta given to the loop line which has been cut, with the crucial modification that the denominators of the propagators of the loop lines must not be included. Solving a linear equations system, CutTools rewrites the integrand of the loop integral so that the result is factorized in terms of multi-point scalar master integrals easily computed in dedicated libraries.

MadGraph creates these cut loop tree diagrams by defining a new partner to each particle of the original model. These parters are given the same name but appended with a star. Since they carry complex momenta, they will sometimes be abusively called *complex particles* in what follows. The interactions involving these partners are a copy of the ones in the original model with the only constraint of including *exactly two* complex loop particles. This insures to build one-loop topologies. However, the HELAS subroutines associated to these vertices are modified to deal with the complex momenta of the partner particles and create currents with a propagator shortened of its denominator. Note that one only needs to define partners of particles which can run in loops[2].

[1] This pictural expression formally means to put on-shell at any given complex momenta one of the internal line.

[2] In the Standard Model case for example, the QED particles will not have partners since the diagrams where they run in a loop are subleading because of extra α_{QED} powers.

Here is an example with the gluon g. It will have a partner g^* with the following interactions:

$$g^*g^*g,\ g^*g^*g\ g,\ g^*q^*\bar{q} \text{ and } g^*\bar{q}^*q$$

Note that the update of an existing model working at tree level to the same model for NLO computations is not automated. This work has been done for the Standard Model, but one should in principle redo it for each new model submitted.

Specifying the process `In1 In2 > Loop1 Loop2 [OutS]` to MG creates the cut diagrams. `In1` and `In2` are the two incoming particles. `Loop1` and `Loop2` are the two same complex particles coming from the cut line and `[OutS]` symbolically denotes the list of external particles. It is important here to notice that the tree level diagrams generated by MG with this call will correspond to all the different possibilities of cutting the existing loop diagrams at a loop line of kind `Loop1/2`. So even though each loop diagram contributes once only, MG generates many associated cut diagrams (whose number is directly proportional to the number of loop lines). A selection must take place in order to chose only one tree cut diagram for each loop diagram, hence avoiding any redundancy. Of course, loop diagrams without any loop lines of the kind `Loop1/2` will never be generated with the process above. This is why it is necessary to run MadGraph with all the processes built with all the possible complex particles `Loop1/2`. The label i in `Loop_<i>_amps.f` runs over these processes. In the $e^+e^- > q\bar{q}$ case one would generate the loop diagrams by running MG with the processes:

$$e^+e^- > g^*g^*\ q\ \bar{q} \text{ and } e^+e^- > q^*\bar{q}^*\ q\ \bar{q}$$

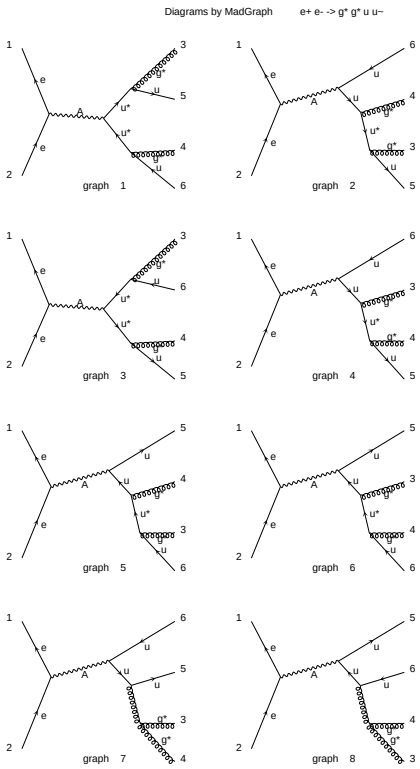

Figure 3.1: Tree cut diagrams generated from the process $e^+e^- > g^*g^*\ u\ \bar{u}$

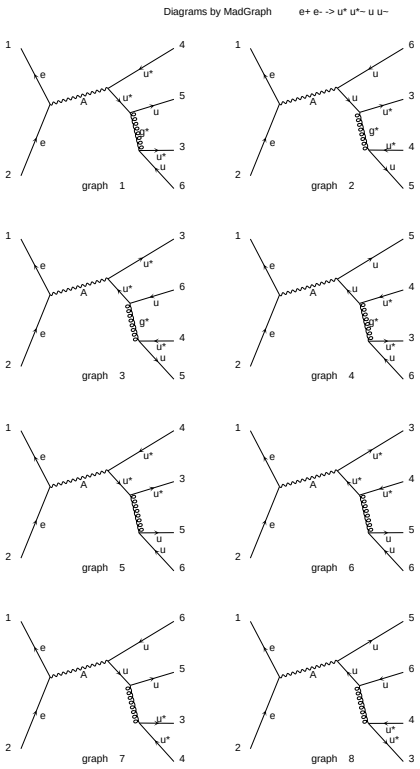

Figure 3.2: Tree cut diagrams generated from the process $e^+e^- > u^*\bar{u}^*\ u\ \bar{u}$

The first process generates eight diagrams (Fig 3.1): two corresponding to tadpoles, four for bubbles and the remaining two come from the cut of the triangle diagram. Here only one tree cut diagram is selected to include the relevant[3] triangle diagram. The second process Fig 3.2 also leads to another set of eight tree cut diagrams from which none will be selected since they are either irrelevant (tadpoles and bubbles) or already taken into account from the first process. Unfortunately the selection is not yet performed when the `Loop_<i>_amps.f` files are generated so all of the tree cut diagrams are written out even though `SigVirt.f` only uses the few selected ones. This inconvenience is of course only specific to the MG4 setup. The structure of the `Loop_<i>_amps.f` subroutine is very similar to the `Born_amps.f` one except that now the `QP` argument gives the complex momenta $-q$ of particle 3, `Loop_1`, and q of particle 4, `Loop_2` . The associated four-letter tag of these subroutines are `LP<i>`. Contrary to the born and R2 diagrams, the loop amplitudes are not self-contained in the `Loop_<i>_amps.f` files because the propagator of the cut loop line is not included. The cut line is split into two external complex lines, each with a definite pseudo on-shell wave-function. For instance, in diagram 1 in Fig 3.1 the gluon of the triangle diagram is split into the two complex gluons g^* of particles 3 and 4. Hence, the subroutine computing this diagram omits the corresponding propagator[4] $-ig^{\mu\nu}$ and asks for the external wave-function of particles 3 and 4. The most obvious way of reestablishing the propagator is to use the standard set of helicity states for vector spin-1 particles, denoted here $\{\epsilon_i^\mu\}$. Care is needed when doing this because the definition of these states must be extended to complex momenta. The subroutine without the cut propagator is called with each matching helicity state for particle 3 and 4. Provided that $\{\epsilon_i^\mu\}$ is a complete set, the full loop amplitude is simply the sum over all the outputs of these calls, thanks to the property

$$\sum_{i=1,2} \epsilon_i^{\mu *} \epsilon_i^\nu \to -g^{\mu\nu} \tag{3.1}$$

This solution works for gluons, but it encounters troubles for fermions because the summing relation over the standard set $\{u^s(q)\}$ reads[5]

$$\sum_{s=1,2} u^s(q)\bar{u}^s(q) = \not{q} + \sqrt{q^2}\mathbb{1} \tag{3.2}$$

The complex momentum q provided by CutTools does not need to be on-shell so that $\sqrt{q^2} \neq m$ in general. The quark propagator $i(\not{p} + m\mathbb{1})$ cannot be reproduced with this method.

Because of that, another workaround is used by NLOComp, both for the gluon and quark propagator. It uses the following trivial set $\{e^s = \delta_j^s\}$ for the external wave-functions of particle 3 and 4. Here s is the pseudo helicity index and j is the index of the representation of dimension N of the group to which the cut loop particle belongs. If we write $\mathcal{P}_{j_1 j_2}$ the propagator of this particle, $\mathcal{A}$ the full loop amplitude and $\mathcal{A}(v, w)$ the amplitude computed with v and w for the external wave-functions of particles 3 and 4, we have

$$\mathcal{A} = \sum_{s,t=1}^{N} \mathcal{P}_{st}\mathcal{A}(e^s, e^t) \tag{3.3}$$

The specific reconstruction of the gluon cut propagator efficiently illustrates the method. The trivial set takes the form

$$\{e^s\} = \{(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)\} \tag{3.4}$$

And the full loop amplitude is reconstructed with

$$\mathcal{A} = -i\mathcal{A}(e^1, e^1) + i\mathcal{A}(e^2, e^2) + i\mathcal{A}(e^3, e^3) + i\mathcal{A}(e^4, e^4) \tag{3.5}$$

[3]The diagrams with a bubble attached to each of the external outgoing line matter only when considering massive quarks. But, even in this case, the renormalization of the wave functions is performed analytically. So only bubbles attached to internal lines have to be considered, as explained in 3.4.

[4]Once again, the OPP method asks for the amplitudes of the loop diagrams computed for a certain loop momentum q and with all loop propagators shortened of their denominator.

[5]When using this relation or even 3.5, keep in mind that the components of spinors are Grassmann numbers flipping the sign of the result when commuted to be in the same order as in the summing relation.

This workaround is implemented in `Loop.f` which organizes the calls to the subroutines in `Loop_<i>_amps.f` and sums their outputs according to the rules above. An argument in the `Loop.f` subroutine set by NLOComp based on the nature of the cut particle, specifies which propagator to use. `Loop.f` is a generic file (*i.e.* not generated specifically for each process) and for now only includes two propagators, the massive vector and the massive spinor one.

So even though this method is very general, it lacks efficiency because it requires to compute the same amplitude at most N^2 times and is not generic since any new propagator has to be rewritten by hand even though it is already intrinsically present in the HELAS subroutines. This is however the best one can do within MG4 framework. Hopefully, MG5 will avoid reconstructing the missing propagator by somehow integrating it in the HELAS calls chain.

3.1.3 R2 diagrams

The R2 diagrams give the non cut-constructible part of the rational part of the loop amplitude. They are obtained from modified Feynman rules built from an analytical study[3] of the ϵ-dimensional part of the different loops affecting the propagators and vertices of the model at hand. If we exclude four-point vertices, all the R2 Feynman rules are proportional to the standard vertices and propagators. The special two-point R2 vertices (coming from the bubble diagrams) are viewed in this MG4 setup as modified propagators proportional[6] to the standard ones. In each of the R2 diagrams, there is *exactly one* such R2 special vertex or propagator at any place where there might be a relevant loop. So, for instance, you can put R2 propagators (or two-points vertices) only on internal lines.

In the actual setup, the R2 diagrams are generated by keeping the same particle content as the original model but adding R2-copies of the existing standard interactions. However, these R2 interactions are special because they come with different coupling constants and not only bring an additional order of their type (QCD, QED, ...) but also a R2 or R2p order. Each R2-vertex brings an order of R2 and each R2-propagator brings two orders of R2p, one for each vertex attached to its ends. MadGraph core code has been modified so that these order types labeled R2 and R2p now have a special status: MadGraph considers only diagrams with an order in R2 *or* R2p *exactly* equal to the target values specified in `proc_card.dat`. When asking for the born process and setting `R2=1` and `R2p=2`, MG generates the desired R2 diagrams. Note that the the four-point vertices are correctly implemented in the sense that they lead to the generation of the correct diagrams. However, the computation of R2 diagrams with four-point vertices is not possible with this framework because of the non trivial entanglement of the color and Lorentz structure of the associated Feynman rules.

The special R2 and R2p coupling constants are computed separately from an analytical study of the different loops affecting the vertices and propagators of the original model. Even though their number is finite and they can be computed once for all, this is still a consequent work and prevents the full automation of NLO computations from the raw model. This is the drawback for using CutTools and the OPP method which in return offers great generality.

The R2 subroutine's structure is exactly as for the born, except for the four-letter tag which is here `R2__`.

3.2 Ghost diagrams

The standard gauge choice for $SU(3)$ in the standard model is the *Feynman-'t Hooft* gauge in which the gluon propagator is conveniently $\frac{-ig^{\mu\nu}}{p^2}$. In non-abelian theories, the gauge-fixing term does not leave invariant the determinant of the path integral defining the gluon propagator. This effectively introduces additional states called *ghosts* which are anticommuting fields , scalars under Lorentz transformations. They only couple to the gauge fields, so the gluons in the $SU(3)$ case, and they never appear as on-shell external particles. This restricts they contribution to diagrams with a loop with only gluons attached to it. There are no such diagrams in most simple processes, like e^+e^- to two or three jets, so the ghost treatment has never been checked so far.

[6] This proportionality is not crucial, since it is easy to ask that the R2 interactions defined in `interactions.dat` use specific HELAS subroutines different from the standard ones.

Nevertheless, the necessary work has been carried out. Ghosts are implemented in MadGraph as fermions with only one degree of freedom so that the anticommuting behavior is correctly reproduced along with their scalar nature. This rather peculiar property for a particle required a modification of the core files of MadGraph. Their interactions lead to the correct contributing Feynman diagrams and the related HELAS subroutines have been written. So, as a matter of principle the whole procedure works but has never been tested.

3.3 Color factors

MadGraph 4 treats color internally and independently of the Lorentz structure of the amplitude implemented in the HELAS routines. It uses color-flow decomposition which provides a color basis on which the color structure of each diagram is projected. This method only uses color indices in the fundamental representation so that the color octets carry two such indices and have modified projectors including correction for the singlet-like gluon.

Each element of the color basis is then squared against each other to form the *color matrix* used for squaring the tree born diagrams. This method is very efficient since the size of the color matrix to be computed is $n \times n$ where n is the number of elements in the color basis. The four-gluon vertex is treated apart and is implemented as three copies of the same interaction, each one with a specific order of the color indices in the structure constant f^{abc} matching the order of the arguments in the HELAS calls (*i.e.* the associated Lorentz structure). Even if the color factors of diagrams with four-gluon vertex are correctly computed in NLOComp, the corresponding R2 diagrams cannot be handled so that the related processes can anyway not be computed in the actual setup.

In NLO computations, the situation is different because interference terms must be computed and the loop diagrams squared with the born ones have a different color basis. So for each process, MadGraph now creates a file `<proc_name>_colors.dat` which starts by specifying each of the color basis element as a string of delta-structures corresponding to a possible color-flow appearing in the diagrams for the process at hand. The second part of the file gives for each diagram the weight of each element in its projection on the color basis. From that information NLOComp can reconstruct the full color string of each diagram. For the loop diagrams, the color string is not complete until the loop is *closed*. If the loop has been cut at the level of a fermion, NLOComp adds to the color string the structure δ^a_b where a and b are the color indices carried by the complex fermionic particles 3 and 4. This closes the color loop and builds the trace. For a gluon cut, the projector is $\delta^{a_1}_{b_2}\delta^{b_1}_{a_2} - \frac{1}{N_c}\delta^{a_1}_{a_2}\delta^{b_1}_{b_2}$ with a_i and b_i being the two color indices in the fundamental representation carried by each gluon. Then NLOComp squares each of the color strings of the loop diagrams with each of the color strings of the borns to build the color matrix for the interference term computation. Note that the squaring operation also goes by including the same projectors as above for the color octets and triplets in the final states. While perfectly working, this method ruins all the efficiency gained by the use of color-flow decomposition since now the color matrix is $n \times m$ where n is the number of virtuals and m the number of born diagrams. The idealistic case would be to project both the loop and born diagrams on the same color basis which is in principle not hard to derive since it corresponds to the one of the real emission diagrams[7]. This optimization is not available in the MadGraph 4 setup, but hopefully will be within MG5.

3.4 Tree cut diagrams selection

Section 3.1.2 exposed how tree cut diagrams are generated within MG4 framework. Because of their redundancy, it appeared necessary to select a subset among these tree cut diagrams which has a one-to-one correspondence to each of the loop diagrams participating in the process. This subset is further referred to as the *loop basis*. NLOComp performs this selection based on an internal representation of the diagrams which is built on the files `<proc_name>_configs.dat` generated for each process by MadGraph[8]. These files contain a list of nodes for each diagram along with the PDG code of each internal and external particle.

[7] So, as a bonus NLOComp would not even need to recompute the color matrix.

[8] The original MG already create such a file, but it has been modified to include four-point interactions and also specify the PDG code of the external particles.

A node is simply a series of numbers (three or four in this case) corresponding to the ID tag of the lines attached together in a diagram. It is organized such that the integer ID of the external particles is positive and negative for the internal ones.

NLOComp parses the config files and constructs an internal representation of the diagrams suited for the selection of a loop basis. It first identifies the loop lines through which the complex loop momenta runs. This forms the *complex flow*. Then, NLOComp recursively builds all the structure branching out of the complex flow. These structures might be just one particle or even another tree organization. Also note that the contact point with the complex flow can be through one or more particles in the case of a four-point interaction. NLOComp is able to compare these structures and assign to the equivalent ones a unique integer tag. Each diagram can now be associated to a chain called the diagram tag. This tag is constructed by starting with the PDG code or representative letter of the cut line. Then, the next structure in a given direction is put in, followed by the PDG code of the particle next to it and so on, until the starting point is reached again. The rest of the selection algorithm is straight-forward; it takes out all diagrams whose tags are cyclic and/or mirror permutation of another one in the loop basis. Because of the specific orientation of the quantum number flow in a purely fermionic loop, the mirror symmetry must not be considered in those cases. This whole algorithm for the diagram selection is completely automated.

Here is an example with the triangle diagrams of Fig. 3.1 and Fig. 3.2. In these diagram, only three structures are identified:

- The one with external particles one, e^+, two, e^- and the virtual photon. Structure labelled A.
- The particle 5, quark up. Structure labelled B.
- The particle 6, antiquark up. Structure labelled C.

Chart 3.4 shows the tags obtained for the triangle diagrams when starting along the loop flow with particle 3.

Topology	g-cut Diag.	Tag	$\equiv$	Topology	q-cut Diag.	Tag	$\equiv$
$\triangle$	g1	$g^*\mathrm{B}u^*\mathrm{A}u^*\mathrm{C}g^*$	basis	$\triangle$	q1	$u^*\mathrm{C}g^*\mathrm{B}u^*\mathrm{A}u^*$	g1
$\triangle$	g3	$g^*\mathrm{C}u^*\mathrm{A}u^*\mathrm{B}g^*$	g1	$\triangle$	q3	$u^*\mathrm{A}u^*\mathrm{C}g^*\mathrm{B}u^*$	g1

Table 1: Tags of the triangle diagrams for $e^+e^- > u\bar{u}$ at NLO.

We see that once diagram g1 is added to the loop basis, then the other three tree cut diagrams generated with four complex lines are recognized as equivalent to g1 in agreement with the fact that there is only one triangle diagram contributing to $e^+e^- > u\bar{u}$ at NLO. As already discussed, the bubble and tadpole diagrams do not contribute and NLOComp can easily identify their topology by counting the number of their complex lines, so that the tadpoles diagram can be ignored. Some care is needed with the bubble diagrams, because they must be numerically taken into account if they are not wave-function renormalization (*i.e.* attached to an external line). So NLOComp considers only the bubble diagram with the two structures attached to it being constituted of more than one particle. In this $e^+e^- > u\bar{u}$ case, no tadpole nor bubble diagrams contributes so that NLOComp only keeps g1 in the loop basis.

To demonstrate the tagging method further, the analysis above is carried on with the other diagrams as well. In these diagrams, new structures appear which are nothing but the merging of the three fundamental ones A,B and C. They are denoted (AB), (BC), (AC), (ABC) or (ACB). Note however that NLOComp does not introduce these structures by reusing the smaller ones, but as independent new entities[9].

[9]In NLOComp, the structures are implemented as a list of lines (internal and external) each having a list of its parent lines. To compare two structures, NLOComp first makes sure that the number of lines matches and then that each line has the same parents. Thus, (BC)$\equiv$(CB) but (ABC)$\not\equiv$(ACB).

Topology	g-cut Diag.	Tag	$\equiv$	Topology	q-cut Diag.	Tag	$\equiv$
-O-	g2	$g^*\mathrm{B}u^*(\mathrm{AC})g^*$	basis	-O-	q2	$u^*(\mathrm{AC})g^*\mathrm{B}u^*$	g2
-O-	g4	$g^*(\mathrm{AC})u^*\mathrm{B}g^*$	g2	-O-	q4	$u^*\mathrm{C}g^*(\mathrm{AB})u^*$	g5
-O-	g5	$g^*\mathrm{C}u^*(\mathrm{AB})g^*$	basis	-O-	q5	$u^*(\mathrm{BC})u^*\mathrm{A}u^*$	basis
-O-	g6	$g^*(\mathrm{AB})u^*\mathrm{C}g^*$	g5	-O-	q6	$u^*\mathrm{A}u^*(\mathrm{BC})u^*$	q4
-O	g7	$g^*(\mathrm{ACB})g^*$	basis	-O	q7	$u^*(\mathrm{ACB})u^*$	basis
-O	g8	$g^*(\mathrm{ABC})g^*$	basis	-O	q8	$u^*(\mathrm{ABC})u^*$	basis

Table 2: Tags of the bubble and tadpole diagrams for $e^+e^- > u\bar{u}$ at NLO.

So, if we were selecting all non redundant cut tree diagrams, disregarding their relevance, the loop basis would be made of eight diagrams; one for the triangles, three for the bubbles and four for the tadpoles. Of course, as explained before, only the triangle diagram remains in the actual computation.

3.5 MadFKS

Once `SigVirt.f` is generated by NLOComp, it is copied along with the relevant fortran files into `<MG_Root_folder>/NLOComp_FKS/SubProcesses/SigVirt`. There is only one (!) line of all the MadFKS source files which needed to be changed: in `LesHouches.f` the call to the dummy subroutine for the virtuals-borns interference is changed to the call to SigVirt which in principle gives the right virtual cross-section for a given phase-space point and renormalization scale. Some minor changes in the makefiles and the `newprocess_fks` script were brought to link the CutTools library and to create the required hard links to the new source files in the FKS pairs subdirectories. Also, CutTools can only be compiled with `gfortran`[10], so all the other libraries and makefiles are switched to this compiler which showed to work fine for all of them. Unlike for tree-level, not all the libraries are automatically rebuilt, and notably not CutTools library `libcts`. So there are still some manual manipulations remaining when switching models or architectures. But this is not an obstacle to full automation in a final version.

Apart from these technical details MadFKS really works independently from the computation of the virtuals which has the great advantage of sand-boxing each of the two contributions to the total NLO cross-section, hence easing the debugging. However, it appears more and more certain that significant optimization can be obtained by not computing the full virtual cross-section and real emission separately. For instance, a promising line of thinking goes towards merging these two contributions already at the level of color-flux ordered amplitudes. This is anyway an optimization which requires radical changes mostly on the MadFKS structure. The MG5 setup would then easily adapt to MadFKS needs.

4 Requirements summary

Here are the different requirements and issues that MG5 must address to perform NLO Computations with CutTools. A ✗ means that the item is not yet implemented within MG5.

1. ✓ *Separately computing the amplitude of any diagram.*

   Necessary in order to square the right diagrams when building the interference term. Optimization can be turned on/off at will.

2. ✓ *Allow loop particles to carry complex momenta.*

   In MG5, all particles carry complex momentum by default.

3. ✗ *Diagram generation - Differentiate loop lines from those of the branches attached to the loop.*

   The loop particles will not be implemented as independent copies as in MG4, but instead a *loop* tag will be added to the *Leg* class of MG5. The recursive algorithm for the diagram generation should

[10]It must be possible to translate the code to make it compatible with another compiler.

be left untouched and generates all possible tree-cut diagrams. Only then a filter selects the loop basis, in a complete analogy to what NLOcomp does. It appears that constructing loop topologies by first generating many more tree diagrams and performing a selection among them might even be competitive with other standard methods used by *FeynArts* or *QGraph*.

4. ✗ *Select the tree cut diagrams to have a one-to-one matching with the relevant loop diagrams.*

   MG5 can bring much richer information on the generated diagrams. The selection algorithm described in 3.4 should be easily implemented along with the generation of `SigVirt.f` and related files.

5. ✗ *Allow the complex particles to have propagators with the denominator taken out.*

   This is again something which can be treated with the same workaround as used here, namely writing new HELAS subroutines. However, this is redundant since the full propagators are already coded in the existing HELAS subroutines and the idealistic behavior would be to somehow reuse them. This is really more a HELAS issue since MG5 has a very abstract internal representation of the wavefunctions creation chain which can be exported to any HELAS model. So one could easily create a HELAS model, where the presence of the denominators of propagators can be set by an argument. For many other reasons too, it would be wise to entirely rewrite the HELAS model.

6. ✗ *Ghost diagrams, gauge choice.*

   The MG4 implementation of ghosts conceptually proves that the impact of the gauge fixing terms on loops can really be effectively taken into account by introducing new scalar states. There is *a priori* no obstacle to the introduction of ghosts in MG5. It still needs some core changes of the actual setup to allow scalar particles to have fermion-like anticommuting behavior.

7. ✓ *R2 diagram generation, and the special two- and four-point vertices.*

   The two-point vertices are viewed as modified propagators in MG4 and even though working, this picture is not physically correct. MG5 being able to implement two-point vertices, this desiderata will be cleared in a proper manner. It is not satisfactory to have three generated diagrams for one four-leg vertices as it is done in MG4. The four-leg vertices have to be treated along with the three- and two-point vertices in a much more homogenous way. MG5 is designed for that and allow it without any problem.

8. ✓ *User defined color structure for any vertex.*

   Internal fixed treatment of colors had its time and user-defined color structure is now mandatory for complicated vertices like the R2 one with four gluons. For each vertex, MG5 allow a completely general list of Lorentz and Color structures associated.

9. ✗ *Reconstruction of the numerator of the propagator of the cut loop line.*

   In MG4, this issue is dealt in the worst manner and even though very general, it costs a lot in efficiency. The best way of handling this would be to incorporate the missing propagator in the HELAS calls chain hence avoiding to rewrite the already existing propagators. Some thinking showed that there is no better way to compute a loop amplitude than mimicking successive HELAS chains either by using tensor-like particles or by calling a maximum of four times a HELAS chain with standard particles and an additional two-point interaction giving the missing numerator. This last solution is preferable and should be investigated first.

The following items can bring significant optimizations but are not mandatory for NLO computation in MG5.

1. ✗ *Take out loops vanishing because of group factors.*

   Simple optimization, provided one can have access to the loop basis after color factor computation. This should not be a problem in MG5 and can even be implemented in the MG4 setup without any

problem. In MG5, it is even better to keep the diagram and associate zero weights to it so that it is automatically skipped when output.

2. ✓ *Computation of the color factors by projecting the virtuals and born diagrams on the same color basis.*

   This would be a major improvement since color treatment is the limiting factor for NLO computation of high multiplicities processes. MG5 can already create an appropriate loop basis from any set of diagrams.

3. ✗ *Summing color-ordered structures attached to a loop before calling CutTools to compute it.*

   What characterizes a loop for CutTools is only a set of momenta and masses for the loop lines. It is then possible to call CutTools with loop amplitudes computed with the sum of the matching color-ordered structures attached to the loop instead of calling CutTools for each structure within this sum.

   This might bring a very significant improvement for large multiplicities process for which there are many color ordered structures contributing to the same substructure attached to the loop. Recognizing these matching structures and organizing the computation with *smart* calls to CutTools is not straightforward and will only be implemented at a second stage as an optimization update. But let's keep in mind that for each loop computation, CutTools calls the loop amplitude up to more than 20 times so it is worth trying to spare any single call to CutTools.

5 Conclusion

This work on adapting MG4 for NLO computations started when it was already clear that the MadGraph python upgrade MG5 would take over. This released the burden of aiming at a final clean product and allowed any workaround to prove the feasibility of the concept and to help defining the requirements and obstacles to overcome within MG5.

This report first described these workarounds in sect. 3, then clearly defined the new requirements for NLO computation in sect. 4 and proved their feasibility with checks for the simplest processes in Append.6.2.

It is clear that the MG4 framework reached here its limits and with the unexpected fast progress of MG5, the efforts must now focus on exploiting our successes here to build a clean and powerful NLO upgrade within MadGraph 5.

6 Appendix

6.1 NLO `param.dat` card details

The `param.dat` card contains all the information required by NLOComp to compute the desired process. Here is a list of them. Note that a space must be put between each element in a given entry.

- `Process = e+ e- > u u~ g`

  Specify the desired process in the same manner as it is done in MG4. Note however, than it does not handle multiparticles definitions, neither particle exclusion nor multiprocesses.

- `Model = smNLO`

  Specify the model to be used. Note that one cannot expect a model folder working for the tree-level computations works as it stands with NLOComp. Many changes must be brought, related to the different issues discussed: R2 special Feynman rules, new particles definition carrying complex momenta and their associated vertex, ghost particles.

- `LoopParticles = u* g*`

  This entry sets what particles can possibly run in the loop of the virtuals. For particles having an antipartner, only specifiy the particle, the antiparticle is assumed. Only *star* particles (i.e. those which carry complex momenta) are meant to be specified here. In principle their order does not matter, however, during the diagram selection procedure, it will select the loop diagrams cut in the same order as entered here. For example if `u* g*` is used then the loops will be cut at a fermion line if possible and only at a gluon line if they do not involve any fermion. The opposite holds when using `g* u*` and since the result is independent of the cut, this option offers an efficient consistency check.

- `OutputBaseName = eejjj`

  Sets the name of the root folder. Will be of the form `NLO_<OutputBaseName>`.

- `MinimalLoopLines = 2`
  `MaximalLoopLines = -1`

  Sets an additional user-filter on the loop diagram taken into consideration. the -1 value turns the filter off, while setting `MinimalLoopLines` to 2 is customary to get rid of the tadpole diagrams which never contributes after UV renormalisation.

- `MinQEDOrder = -1`

  In order not to introduce sub-leading diagrams with extra powers of α_{QED}, MG builds only the diagram with the minimal QED order. When this option is set to -1, the minimal QED order is guessed by NLOComp from the number of external $SU(3)$ singlet states. If, for some reason, this value happens not to be accurate, then a user-defined value can be set here.

- `WaveFunctionRenormalization = OFF`

  The bubble diagrams appearing on the external real lines contribute to the wave-functions renormalization which is normally take care of analytically and only the bubble loops appearing in the internal virtual lines must be numerically taken into account. Setting this option to `OFF` selects only these diagrams.

6.2 Checks

The simplest process $e^+e^- > 2j$ or $3j$ have been used for checks. The first is simple enough to set and fix the basics of the NLO computing mechanism and the second covers more specific cases to show its generality. Note that the jets can only be composed of massless u-quarks or gluons. Extension to the massive case and sum over all jet particles is under progress. Some parameters are common to both processes and described here.

Name	Description	Value
M_Z	Mass of the Z boson	91.1880 GeV
Γ_Z	Width of the Z boson	2.44140 GeV
$\sin^2(\theta_W)$	Sine squared of the Weinberg angle	0.222247
$\alpha_S(M_Z)$	Strong coupling constant at M_Z	0.118000
$\alpha_W^{-1}(M_Z)$	Inverse of weak coupling constant at M_Z	132.507

Table 3: General parameters used for the checks.

6.2.1 The $e^+e^- > u\bar{u}$ process

This is the simplest process and it has the great advantage of having an analytical solution for the differential cross-section. Working in dimension $d = 4 - 2\epsilon$ gives

$$\sigma_{\text{virt}} = -\sigma_{\text{born}} \frac{\alpha_s}{2\pi} C_F \left(\frac{4\pi\mu^2}{-s}\right)^{\epsilon} \frac{\Gamma^2(1-\epsilon)\Gamma(1+\epsilon)}{\Gamma(1-2\epsilon)} \left(\frac{2}{\epsilon^2} + \frac{3}{\epsilon} + 8 + \mathcal{O}(\epsilon)\right) \tag{6.1}$$

Note that both in MadFKS and in CutTools, there is scalar integrals normalization factor taken out. So the expression above must be divided by it

$$\mathcal{N} = (4\pi)^{\epsilon} \frac{\Gamma^2(1-\epsilon)\Gamma(1+\epsilon)}{\Gamma(1-2\epsilon)} \tag{6.2}$$

The amplitudes are considered already summed over helicities. Two random phase-space (PS) points have been used for the checks. They are presented here with the four-momenta convention $p^\mu = (E, p_x, p_y, p_z)$.

$$PS_1 = \left\{ \begin{array}{cccccc} \text{Part. name} & & \text{Momentum} & & & \\ e^+ & (45.59400000, & 0.000000000, & 0.000000000, & 45.59400000) \\ e^- & (45.59400000, & 0.000000000, & 0.000000000, & -45.59400000) \\ u & (45.59400000, & 10.11496365, & 40.56323003, & -18.19683257) \\ \bar{u} & (45.59400000, & -10.11496365, & -40.56323003, & 18.19683257) \end{array} \right. \tag{6.3}$$

$$PS_2 = \left\{ \begin{array}{cccccc} \text{Part. name} & & \text{Momentum} & & & \\ e^+ & (455.9400000, & 0.000000000, & 0.000000000, & 455.9400000) \\ e^- & (455.9400000, & 0.000000000, & 0.000000000, & -455.9400000) \\ u & (455.9400000, & -153.9163943, & -319.8364079, & 286.1742465) \\ \bar{u} & (455.9400000, & 153.9163943, & 319.8364079, & -286.1742465) \end{array} \right. \tag{6.4}$$

Using these two PS points, and the renormalization scales $\mu = 91.188$ and $\mu = 911.88$ for each, the agreement between the MadGraph 4 setup (MG4) and the analytical solution above is very good. The relative error with it is printed under the *Err* tag. The coefficients indicated in 6.5 and 6.6 are the virtuals-borns interference cross-section *divided* by the born cross-section for the same phase-space point.

$\sigma_{\text{virt}}/\sigma_{\text{born}}(PS_1, \mu)$		Rational	ϵ^{-1} pole	ϵ^{-2} pole
$\mu = 91.188$	MG4	0.04681560038	−0.07512113314	−0.05008075543
	Err	$8.6 \cdot 10^{-13}$	$6.4 \cdot 10^{-14}$	$1.4 \cdot 10^{-16}$
$\mu = 911.88$	MG4	−0.8301761274	−0.3057515349	−0.05008075543
	Err	$7.5 \cdot 10^{-14}$	$1.6 \cdot 10^{-14}$	$1.4 \cdot 10^{-16}$

(6.5)

$\sigma_{\text{virt}}/\sigma_{\text{born}}(PS_2,\mu)$		Rational	ϵ^{-1} pole	ϵ^{-2} pole
$\mu = 91.188$	MG4	-0.1382849221	0.1555092686	-0.05008075543
	Err	$6.1 \cdot 10^{-12}$	$3.1 \cdot 10^{-12}$	$1.4 \cdot 10^{-16}$
$\mu = 911.88$	MG4	0.04681560038	-0.07512113314	-0.05008075543
	Err	$2.9 \cdot 10^{-11}$	$6.4 \cdot 10^{-12}$	$1.4 \cdot 10^{-16}$

(6.6)

The integrated result is also analytical and has a very simple expression which reduces to the famous α_S/π factor when substituting the actual value of the color factor C_F.

$$\sigma_{\text{tot}} = \sigma_{\text{born}} \left(1 + \frac{3}{4} C_F \frac{\alpha_S}{\pi} + \mathcal{O}(\alpha_S^2)\right) \tag{6.7}$$

With σ_{born} the leading order result given by

$$\sigma_{\text{born}} = \frac{16\pi\alpha_S^2}{9s} \tag{6.8}$$

The check of the integrated result was carried on for two beam energies, 45.591 GeV and 455.91 GeV and for two renormalization scales $\mu = 91.188$ GeV and $\mu = 911.88$ GeV. Of course, any cut in MadFKS is disabled.

$\sigma_{\text{tot}}(E_{\text{beam}},\mu)$ [pb]		$E_{\text{beam}} = 91.188$ GeV	$E_{\text{beam}} = 911.88$ GeV
$\mu = 91.188$ GeV	Mad_FKS	$15.48846452 \pm 0.39\%$	$0.1548847016 \pm 0.39\%$
	Err	0.23%	0.23%
$\mu = 911.88$ GeV	Mad_FKS	$15.40810809 \pm 0.38\%$	$0.1540810963 \pm 0.38\%$
	Err	0.33%	0.33%

(6.9)

The agreement is conclusive since it is within the Monte-Carlo statistical relative error, indicated in the chart 6.9 next to the evaluation. The similarity of the relative errors simply relies on the fact that a constant seed as been used for the random generator. A single check with higher statistics, not shown here, confirmed the agreement down to 0.01%.

The use of the second beam energy does not bring much information, only that the scaling property expected holds. It is however interesting to see that the use of a renormalization scale different from $\sqrt{s}$ does work and that the rational parts of the virtuals and the reals are consistently affected so that the final cross-section is left untouched. The success of this check is not general enough to claim that there is no hidden issue but it shows that the setup correctly processes NLO computations.

6.2.2 The $e^+e^- > \gamma > u\bar{u}g$ process

This check is under progress and this subsection describes its present status. Notice from the subsection's title that Z-mediator The two phase-space points considered are

$$PS_1 = \left\{ \begin{array}{cccccc} \text{Part. name} & & \text{Momentum} & & & \\ e^+ & (45.59400000, & 0.000000000, & 0.000000000, & 45.59400000) \\ e^- & (45.59400000, & 0.000000000, & 0.000000000, & -45.59400000) \\ u & (21.60320023, & -4.749855797, & -16.46340630, & -13.15649585) \\ \bar{u} & (33.08259569, & 31.97063231, & -6.581634440, & 5.386918910) \\ g & (36.50220408, & -27.22077652, & 23.04504074, & 7.769576940) \end{array} \right. \tag{6.10}$$

$$PS_2 = \left\{ \begin{array}{cccccc} \text{Part. name} & & \text{Momentum} & & & \\ e^+ & (45.59400000, & 0.000000000, & 0.000000000, & 45.59400000) \\ e^- & (45.59400000, & 0.000000000, & 0.000000000, & -45.59400000) \\ u & (44.18448866, & -10.12956818, & -25.27237152, & 34.79896728) \\ \bar{u} & (36.05057770, & 14.50624984, & 19.99089700, & -26.25979637) \\ g & (10.95293364, & -4.376681659, & 5.281474526, & 8.539170909) \end{array} \right. \tag{6.11}$$

The drastic difference between the two- and three-jet case is that there is no analytical solution for this case. There is however a semi-analytical[11] code for this process written by M. Seymour and S. Catani, called *Event2*, which serves as a reference result for the local cross-section computed by NLOComp.

The ϵ-poles are not provided by Event2. However, it is possible to use MadFKS to get the poles retrieved from the reals. These coefficients are entirely cut-constructible, hence easier to start with for crosschecks. The ϵ^2 poles is also independent of the phase-space point and the renormalization scale. From appendix B of [5], one gets this simple expression for the coefficient of the $1/\epsilon^2$ in the $e^+e^- > \gamma > u\bar{u}g$ process with $\mu^2 = s$.

$$\mathcal{M}_{\text{virt}} = -\frac{\alpha_S}{2\pi}\frac{17}{3}\frac{1}{\epsilon^2} + \mathcal{O}(\epsilon^{-1}) \tag{6.12}$$

The $1/\epsilon$ pole expression is more involved and also receive a contribution from the UV renormalization counterterms which CutTools does obviously not take into account. In the case at hand of massless QCD, there is no UV renormalization of masses and external wavefunctions but only the strong charge. Carlo Oleari [6] gives an expression for this renormalization which, in massless QCD, amounts to

$$\delta Z_{\alpha_s} = \frac{\alpha_s}{2\pi}\left(\frac{2}{3}T_F n_{lf} - \frac{11}{6}C_A\right)\frac{1}{\epsilon} \tag{6.13}$$

Where n_{lf} is the number of light colored fermions considered.

As shown in the chart 6.14, a good enough agreement is achieved. The renormalization scale is kept constant at $\mu = 91.188$ GeV. Note that the virtual cross-section is again *divided* by the born cross-section. Below the MadGraph 4 evalutation stands the relative error with respect to the *exact* result provided by Event2 for the finite part and MadFKS for the poles of the virtual amplitude.

$\sigma_{\text{virt}}/\sigma_{\text{born}}(PS_i)$		Rational		ϵ^{-1} pole	ϵ^{-2} pole
PS_1	MG4	0.2429869596		-0.2710919354	-0.1064216053
	Event2	$8.6 \cdot 10^{-12}$	MadFKS	$1.2 \cdot 10^{-14}$	$7.8 \cdot 10^{-16}$
PS_2	MG4	-0.4310416073		-0.4544163543	-0.1064216053
	Event2	$3.6 \cdot 10^{-13}$	MadFKS	$4.4 \cdot 10^{-13}$	$4.7 \cdot 10^{-16}$

The success of this computation proves the feasibility of the concept of using CutTools within MadGraph. The two types of different R2 vertices are used in this process and it involves all the subtleties that one can expect with QCD NLO corrections within the Standard Model. Contrary to other NLO computers, the massive case should be a rather straight-forward extension since CutTools can handle it without further complication. The loop basis selected by NLOComp is correct and the five purely fermionic loop give no contribution. This is expected because three of them have a vanishing color factor because of the trace on a single $SU(3)$ generator and the two triangles one have opposite amplitude because of Furry's theorem.

At this point, this MG4 framework has reached its goal of proving the feasibility and identifying the requirements for NLO computations with CutTools. So now, instead of pushing this implementation further to cover more general case, effort should be put into MG5.

[11] It pushes the analytical expression for the cross-section as far as possible and then uses library to numerically compute very precisely well-known integrals. This code is then process specific but has a very precise and reliable output which can be taken as a reference for cross-checks.

References

[1] G. Ossola, C. G. Papadopoulos and R. Pittau Nucl. Phys. B **763** (2007) 147
Reducing full one-loop amplitudes to scalar integrals at the integrand level
[arXiv:hep-ph/0609007]

[2] G. Ossola, C. G. Papadopoulos and R. Pittau JHEP **0803** (2008) 042
CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes
[arXiv:hep-ph/07113596.v2]

[3] P. Draggiotis, M.V. Garzelli, C.G. Papadopoulos and R. Pittau JHEP **0904** (2009) 072
Feynman Rules for the Rational Part of the QCD 1-loop amplitudes
[arXiv:hep-ph/09030356.v2]

[4] Rikkert Frederix, Stefano Frixione, Fabio Maltoni, Tim Stelzer JHEP **0910** (2009) 003
Automation of next-to-leading order computations in QCD: the FKS subtraction
[arXiv:hep-ph/09084272.v2]

[5] T. Stelzer and W. F. Long Comput.Phys.Commun. 81 (1994) 357-371
Automatic Generation of Tree Level Helicity Amplitudes
[arXiv:hep-ph/9401258.v1]

[6] Carlo Oleari PhD thesis (1998)
Next-to-Leading-Order Corrections to the Production of Heavy-Flavour Jets in e+e- Collisions
[arXiv:hep-ph/9802431v1]