Published for SISSA by Springer

Received: *April 14, 2011*
Revised: *June 2, 2011*
Accepted: *June 13, 2011*
Published: *June 28, 2011*

MadGraph 5: going beyond

Johan Alwall,[a] **Michel Herquet,**[b] **Fabio Maltoni,**[c] **Olivier Mattelaer**[c] **and Tim Stelzer**[d]

[a] *Theoretical Physics Department, Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, U.S.A.*

[b] *Nikhef Theory Group, Kruislaan 409, 1098 SJ Amsterdam, The Netherlands*

[c] *Centre for Cosmology, Particle Physics and Phenomenology (CP3), Université Catholique de Louvain, Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium*

[d] *Department of Physics, University of Illinois at Urbana-Champaign, 1110 West Green Street, Urbana, IL 61801 U.S.A.*

Abstract: MadGraph 5 is the new version of the MadGraph matrix element generator, written in the Python programming language. It implements a number of new, efficient algorithms that provide improved performance and functionality in all aspects of the program. It features a new user interface, several new output formats including C++ process libraries for Pythia 8, and full compatibility with FeynRules for new physics models implementation, allowing for event generation for any model that can be written in the form of a Lagrangian. MadGraph 5 builds on the same philosophy as the previous versions, and its design allows it to be used as a collaborative platform where theoretical, phenomenological and simulation projects can be developed and then distributed to the high-energy community. We describe the ideas and the most important developments of the code and illustrate its capabilities through a few simple phenomenological examples.

Keywords: QCD Phenomenology

JHEP06(2011)128

doi:10.1007/JHEP06(2011)128

Contents

JHEP06(2011)128

1 Introduction

Identifying the fundamental building blocks of matter and describing their interactions from first principles is the goal not only of current accelerator based experiments, such as those operating at Tevatron and the LHC, but also of many other experiments, including flavour and neutrino experiments, and dark matter detection experiments in underground laboratories or satellites.

Discoveries from these experiments as well as their interpretation will rely on our ability to perform accurate simulations for both the signals and their backgrounds. At the LHC, for instance, extracting physics from the data will present several significant challenges. First, proton-proton collisions at very high energies produce final states that involve a large number of jets, heavy-flavour quarks, leptons and missing energy, providing an overwhelming background to many new physics searches. Second, even in the presence of a clear "anomaly" with respect to the Standard Model prediction, its interpretation in terms of an underlying phenomena or theory could be extremely difficult. Tools that are able to make precise predictions for wide classes of Beyond the Standard Model (BSM) physics, as well as those that help in building up an effective field theory from the data, will be employed. Before one or a few candidate theories can be selected, accurate measurements of the corresponding parameters (masses, couplings, spin, charges) will be needed. Production rates and/or branching ratio measurements, for example, will provide constraints only if we are able to connect them to the fundamental parameters of a model through an accurate calculation, at least at next-to-leading order in perturbative QCD.

In this context, there is no doubt that Monte Carlo simulations play a key role at each stage of the exploration of the TeV scale, i.e., from the discovery and identification of BSM physics, to the measurement of its properties. The realization of the need for better simulation tools for the LHC has spurred an intense activity in recent years, that has resulted in several important advances in the field.

General purpose matrix-element based event generators, such as CompHEP/CalcHEP [1–3], MadGraph/ MadEvent [4–6], Sherpa [7] have been available for several years now. More recently highly efficiencient multiparton techniques which go beyond usual Feynman diagrams have been introduced [8–11], and implemented in publicly available codes, such as Whizard [12, 13], Alpgen [14], Helac [15] and Comix [16]. As a result, the problem of automatically generating tree-level matrix elements (and then cross sections and events) for a very large class of renormalizable models has been solved. The recent introduction of FeynRules [17] has provided a new method for implementing new physics models as well as setting a new standard in terms of validation and availability [18, 19]. Communication between FeynRules and

JHEP06(2011)128

matrix element programs is being standardized via the new Universal FeynRules Output format, the UFO [20].

A connected effort is being made in the automation of NLO computations. The generation of the real corrections with the appropriate subtractions has been achieved in an automatic way [21–26]. For virtual corrections, several new algorithms for numerical calculation of loop amplitudes have been proposed (see, e.g., [27] for a review) and some of them successfully applied to the computation of SM processes of physical interest [28–32]. Very recently, CutTools [33] has been successfully interfaced with MadGraph. The resulting tool, MadLoop [34] interfaced to MadFKS [26], allows a fully automatic calculation of infrared-safe observables at NLO in QCD for a wide range of processes in the Standard Model.

Last but not least, an accurate simulation of a hadronic collision requires a careful integration of the matrix-element hard process, with the full parton showering and hadronization infrastructure [35–37]. Here again, significant progress has been made in the development of merging algorithms, such as CKKW and MLM merging [38–44], and in their comparison [45, 46], with applications to SM [41, 47, 48] and to BSM [49] processes. A breakthrough in merging fixed order calculations and parton showers was achieved in refs. [50, 51], where it is shown how to correctly interface an NLO computation to avoid double counting and delivered the first event generator at NLO, MC@NLO. More recently, another method along the same lines, dubbed Powheg, has been proposed [52] and applied to a variety of processes at the LHC through the PowhegBox general implementation [53].

The new version of MadGraph has been designed to support and advance the lines of development mentioned above, with three main objectives:

1. Lagrangian-based BSM physics via FeynRules for any renormalizable or effective theory.

2. Full automation and optimization of NLO computations in the SM and beyond.

3. Merging to showering/hadronization codes for complete event simulation at LO (via CKKW and MLM methods) and at NLO (via MC@NLO and Powheg), as well as the combination of the two ("CKKW at NLO").

MadGraph 5 is open source software written in Python and features a collaborative development structure. It can generate matrix elements at the tree-level for any Lagrangian based model (renormalizable or effective) implemented in FeynRules via the UFO interface, and automatic generation of the corresponding helicity amplitude subroutines via the ALOHA package [54]. With respect to MadGraph 4, a significant improvement in efficiency has been attained, and the possibilities for tree-level matrix element generation (and diagram plotting) have been extended, including optimization of the MadEvent output and reorganization of multi-jet final state subprocesses. It features a wide set of flexible output formats in Fortran, C++, and Python, and dedicated matrix element output for Pythia 8 [55].

This work documents and describes the general philosophy of the new MadGraph version, as well as some of the most important improvements in the code. The paper is

JHEP06(2011)128

structured as follows: In section 2 we give a general overview on the code structure and of the algorithms employed. In dedicated subsections we describe the diagram generation algorithm, the fermion-flow algorithm, the colour algebra module, and the generation of decay chains. We then present the available output formats, the new multiprocesses optimization in MadEvent and process library generation for Pythia 8 as well as the new diagram drawing algorithm. Model inheritance from FeynRules via the UFO and ALOHA is presented in section 4 together with a comprehensive list of available models and the suite of model tests that can be performed at the process level. The following section describes the validation checks that have been performed for SM, MSSM, HEFT and RS processes, as well as some key indicators for the performance improvements in speed compared to previous versions of MadGraph/MadEvent. Section 6 provides a selective set of examples of applications. We leave our conclusions and the discussion of the outlook to the last section. Technical appendices follow, where the interested reader can find more details and examples.

2 Overview and algorithms

MadGraph [4] is a tool for automatically generating matrix elements for High Energy Physics processes, such as decays and $2 \to n$ scatterings. First, the user specifies a process in terms of initial and final state particles (allowing for a number of refined criteria, including forced or forbidden s-channel resonances, excluded internal particles, and forced decay chains of final state particles). Multiparticle labels can be used to specify all possible processes involving a range of particles. As a result, MadGraph generates all Feynman diagrams for the process, and outputs the computer code necessary to evaluate the matrix element at a given phase space point. The matrix element evaluation is done using calls to helicity wavefunctions and amplitudes, as were first implemented in the HELAS package [56]. This implementation is efficient because it naturally allows helicity wavefunctions corresponding to identical subdiagrams to be reused across diagrams. MadGraph also produces pictorial output of the Feynman diagrams for the process in question. The computer code produced by MadGraph can then be used for cross section or decay width calculations and event generation, e.g. using the MadEvent package [5], which is included with MadGraph 5.

While previous versions of MadGraph were written in Fortran 77, MadGraph 5 is written in Python. This object oriented computer language allows for completely new algorithms, and removes many of the restrictions that were inherent in the Fortran versions. As a result, both MadGraph 5 and the code it produces run significantly faster than previous versions. Even more important however, is that the structure of the new implementation greatly facilitates selective use of modules and additions of new features. In this paper we will discuss a few such additions, such as implementation of new colour representations (colour sextets and ϵ^{ijk}), implementation of multi-fermion vertices and addition of new output formats, including output in C++ and Python.

JHEP06(2011)128

2.1 Diagram generation

The diagram generation algorithm used in MadGraph 5 is faster, more efficient and produces better optimized code than earlier MadGraph versions.

In MadGraph 4 diagrams generation is based on the following algorithm:

1. Generate all topologies with appropriate number of external legs.
2. Assign particles to external legs.
3. Identify vertices with at most one unassigned line.
4. Check to see if there is any interaction in the model that will accomodate the assigned lines.
5. If yes,
   (a) if a vertex had an unassigned line, assign to it the appropriate particle ID from the interaction, then check next vertex.
   (b) if all lines in the vertex are known, diagram is complete. Write it to file.
6. If no, diagram fails, try next topology.

This algorithm is straightforward to implement, but the time requirement grows quickly with the number of external particles since every topology must be explicitly checked, even if only a small fraction contribute to viable diagrams.

The algorithm in MadGraph 5 eliminates this inefficiency by making use of the model information to effectively only construct topologies that will yield valid diagrams. Furthermore, it recursively generates all of the diagrams in parallel. This ensures that any combination of external legs (a,b), (a,b,c) etc. that is common to multiple diagrams will be recognized as such, allowing for optimal recycling of already calculated subdiagrams in the resulting helicity amplitude code. It also removes restrictions on the number of particles in an interaction vertex, paving the way for implementation of higher-dimensional effective interactions with 5 or more fields.

The algorithm, presented below, is based on recursively creating sub-diagrams from the diagrams by merging legs, with the crucial addition of a flag, `from_group`, which is used to indicate whether a given particle results from a merging of particles (i.e., it is connected to a given set of particles) in the previous step (`True`), or if it is simply copied from the previous step (`False`). This flag helps ensure that no diagrams are double-counted by the algorithm.

1. Given the model, generate two hash maps (called dictionaries in Python), containing information about the interactions in the model. The first dictionary (called `Vertices`) maps all combinations of n particles to all n-point interactions combining these particles, and maps all pairs particle-antiparticle to "`0`". The second dictionary (called `Currents`) maps, for all n-point interactions, $n-1$ particles to all combinations of resulting particles for the interactions.

JHEP06(2011)128

1st iteration	Groupings	After replacements	Result
$e^-, e^+, u, \bar{u}, g$	$(e^-, e^+), u, \bar{u}, g$	$(\gamma), u, \bar{u}, g$	Failed (only 1 FG=`True`)
		$(Z), u, \bar{u}, g$	Failed (only 1 FG=`True`)
	$e^-, e^+, (u, \bar{u}), g$	$e^-, e^+, (\gamma), g$	Failed (only 1 FG=`True`)
		$e^-, e^+, (Z), g$	Failed (only 1 FG=`True`)
		$e^-, e^+, (g), g$	Failed (only 1 FG=`True`)
	$e^-, e^+, (u, g), \bar{u}$	$e^-, e^+, (u), \bar{u}$	Failed (only 1 FG=`True`)
	$e^-, e^+, u, (\bar{u}, g)$	$e^-, e^+, u, (\bar{u})$	Failed (only 1 FG=`True`)
	$(e^-, e^+),\ (u, \bar{u}), g$	$(\gamma),\ (\gamma), g$	Failed (no vertex)
		$(\gamma),\ (Z), g$	Failed (no vertex)
		$(\gamma),\ (g), g$	Failed (no vertex)
		$(Z),\ (\gamma), g$	Failed (no vertex)
		$(Z),\ (Z), g$	Failed (no vertex)
		$(Z),\ (g), g$	Failed (no vertex)
	$(e^-, e^+),\ (u, g), \bar{u}$	$(\gamma), (u), \bar{u}$	Diagram 1
		$(Z), (u), \bar{u}$	Diagram 2
	$(e^-, e^+), u, (\bar{u}, g)$	$(\gamma), u, (\bar{u})$	Diagram 3
		$(Z), u, (\bar{u})$	Diagram 4

Table 1. Tabel to illustrate the steps of the diagram generation algorithm. See text for explanations. `from_group` has been abbreviated as FG.

2. Flip particle/anti particle status for incoming particles in the process (i.e., consider all the particles outgoing). Set the flag `from_group` = `True` for all external particles.

3. If there is an entry in the `Vertices` dictionary combining all external particles, create the combination [(1,2,3,4,...)] if **at least two** particles have `from_group` = `True`.

4. Create all allowed groupings of particles with **at least one** `from_group`=`True` present in the `Currents` dictionary.

5. Set `from_group`=`True` for the newly combined particles, and `False` for any particle that has not been combined in this iteration. Repeat from **3** for the reduced set of external particles.

6. Stop algorithm when at most 2 external particles remain.

As a simple, yet complete example, let us consider the process $e^+e^- \to u\bar{u}g$ in the standard model. The procedure is illustrated in table. 1, and described in detail below. The relevant interactions are $(e^+e^-\gamma)$, (e^+e^-Z), $(u\bar{u}\gamma)$, $(u\bar{u}Z)$, and $(u\bar{u}g)$.

First iteration: after flipping the particle/antiparticle identities for the initial state, we have the external particles $e^-, e^+, u, \bar{u}, g$.

1. No grouping $(e^-, e^+, u, \bar{u}, g)$ is possible.

JHEP06(2011)128

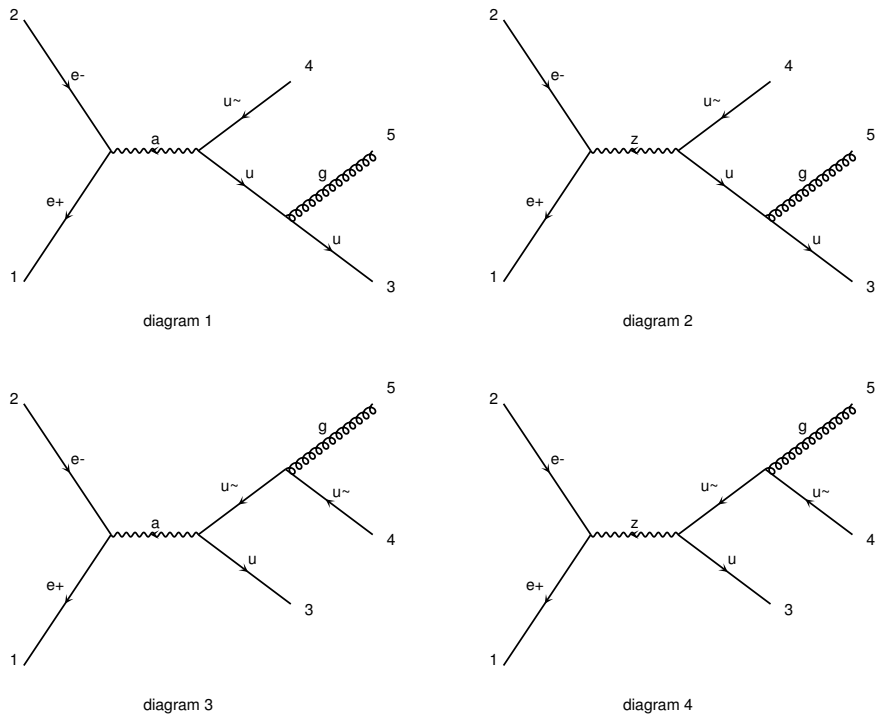

Figure 1. Diagrams for the process $e^+e^- \to u\bar{u}g$. Note that $\bar{u}$ is denoted by "u~" and γ by "a" in the diagrams.

2. Create all possible particle groupings (see table 1):
   $[(e^-, e^+), u, \bar{u}, g]$, $[e^-, e^+, (u, g), \bar{u}]$, $[e^-, e^+, u, (\bar{u}, g)]$,
   $[(e^-, e^+), (u, \bar{u}), g]$, $[(e^-, e^+), (u, g), \bar{u}]$, $[(e^-, e^+), u, (\bar{u}, g)]$
   and replace the grouped particles with the resulting particles from the interactions:
   $[(\gamma), u, \bar{u}, g]$, $[(Z), u, \bar{u}, g]$, $[e^-, e^+, (\gamma), g]$, $[e^-, e^+, (Z), g]$, $[e^-, e^+, (g), g]$,
   $[e^-, e^+, (u), \bar{u}]$, $[e^-, e^+, u, (\bar{u})]$, $[(\gamma), (\gamma), g]$, $[(Z), (\gamma), g]$, $[(\gamma), (Z), g]$, $[(Z), (Z), g]$,
   $[(\gamma), (g), g]$, $[(Z), (g), g]$, $[(\gamma), (u), \bar{u}]$, $[(Z), (u), \bar{u}]$, $[(\gamma), u, (\bar{u})]$, $[(Z), u, (\bar{u})]$. Note that only the particles in parentheses now have `from_group` = `True`.

Second iteration: the resulting reduced sets with four particles all have only one `from_group` = `True`, and can therefore not give valid diagrams. We therefore ignore these and focus on the reduced sets with three particles.

3. The combinations allowed by the interactions are:
   $((\gamma), (u), \bar{u})$, $((Z), (u), \bar{u})$, $((\gamma), u, (\bar{u}))$ and $((Z), u, (\bar{u}))$.

4. Any further combination in the `Currents` dictionary will result in an external state with only one `from_group` = `True`, which can not give any diagrams.

5. The iteration stops, since no external particles are left.

The resulting diagrams are found in figure 1.

JHEP06(2011)128

2.2 Helicity amplitude call generation and fermion number violation

The code for matrix element evaluation generated by MadGraph (previous versions as well as MadGraph 5) is written in terms of successive calls to a helicity amplitude function library (originally the HELAS library, in MadGraph 5 either HELAS or helicity amplitude functions automatically generated by ALOHA [54]). A helicity wavefunction is generated for each external leg in a diagram, and these wavefunctions are combined into new wavefunctions corresponding to the propagators in the diagram by successive helicity wavefunction calls. The final vertex corresponds to a helicity amplitude call which returns the value of the amplitude corresponding to this diagram. In this procedure, wavefunctions corresponding to identical subdiagrams contributing to different diagrams can be reused between the diagrams, leading to a considerable optimization, effectively giving up to a factor hundred fewer wavefunction calls than naively expected by the number of Feynman diagrams.

In the presence of Majorana particles or fermion number violating vertices, special care is needed to allow for all possible contractions of fermions. MadGraph 5, like its predecessors [6, 57], uses the 4-spinor Feynman rules for fermion number violation developed in [58]. In this formulation, a fermion flow is defined for each fermion line in a diagram, and fermion number violation (due to fermion flow clashes) is taken into account using special charge conjugate versions of the Γ^μ matrices. With this modification, the fermion lines meeting at the vertex should be treated as if they had a single fermion flow. The flow therefore needs to be inverted along one of the lines, resulting in a continuous fermion flow.

In earlier versions of MadGraph, this fermion flow was implemented by creating a charge-conjugate particle for each fermion in the model. In order to check for diagrams with clashing arrows it was necessary to check each topology with both the regular fermion, and its charge conjugate. This would increase the time for generating code by a factor of $2^{N_{\text{fermions}}-1}$.

In MadGraph 5, the diagrams resulting from the diagram generation are independent of the fermion flow, and the definition of the flow is postponed to the time of helicity amplitude call generation.

Fermions with positive PDG code ("particles") are tentatively assigned to be incoming (outgoing) if they are in the initial (final) state, and vice versa for fermions with negative PDG code ("antiparticles"). If a fermion flow clash is detected at the meeting of two fermion lines (i.e., the two fermions have the same "incoming/outgoing" status), the fermion flow direction of one of the lines has to be inverted, and charge conjugate vertices have to be introduced starting from the position of the vertex or Majorana particle line responsible for the fermion flow clash.

The procedure is the following: Fermion lines involved in the clash are traversed, looking for Majorana particles. If a Majorana particle is found along one of the lines, the "incoming/outgoing" status and particle/antiparticle id is reversed for all particles up to (and including) the last Majorana particle along the line. For particles beyond the Majorana particle along the same fermion flow line, a flag `fermionflow` is set to -1. The other line is left unchanged. If no Majorana particle is found along either of the lines

JHEP06(2011)128

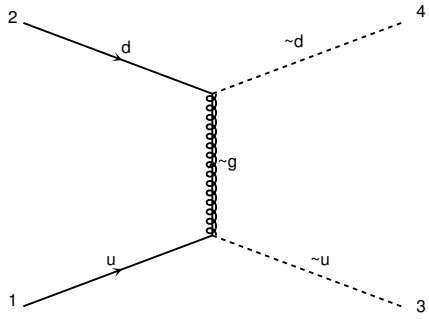

Figure 2. Diagram for the fermion flow violating process $ud \to \tilde{u}\tilde{d}$.

(which is the case when the clash is due to a fermion flow violating vertex), all fermions along the first leg have their `fermionflow` flag is set to -1.

A helicity amplitude or wavefunction with any fermion with negative `fermionflow` flag is required to use the charge conjugate version of the amplitude of wavefunction, in accordance to the Feynman rules in [58]. For an external leg, the incoming/outgoing status is reversed.

For multifermion vertices, the fermions are grouped in pairs, each constituting a fermion line (see section 6.2). Each fermion line then has its own charge conjugate, which can lead to multiple charge conjugate flags for a single helicity amplitude or wavefunction.

As an explicit example of both helicity amplitude call generation and the treatment of fermion flow violation, we take the fermion number violating process $ud \to \tilde{u}\tilde{d}$ with t-channel exchange of a Majorana gluino (see figure 2). The diagram is represented internally (with tentative incoming/outgoing fermion state given in parentheses) as

$$(u(\text{in}), \tilde{u} \to \tilde{g}(\text{in})), (\tilde{g}(\text{in}), d(\text{in}), \tilde{d})$$

The $u\tilde{u} \to \tilde{g}$ vertex would result in an incoming spin $\frac{1}{2}$ helicity wavefunction with incoming flow, and the $\tilde{g}d\tilde{d}$ vertex correspond to a helicity amplitude call. However, a fermion flow clash is detected in the last vertex, since there are two incoming fermions. The two fermion lines coming from this vertex (initiated by the $\tilde{g}$ and d respectively) are now traversed, starting with the gluino. Since the gluino is a Majorana particle (and there are no other Majorana particles along the line), its flow is reversed to "outgoing". The u, which is beyond the gluino along the line, gets the flag `fermionflow` set to -1. This indicates that the u wavefunction should be that of an outgoing external spin $\frac{1}{2}$ particle, and the $u\tilde{u} \to \tilde{g}$ wavefunction should be conjugated (since one of the particles in this wavefunction has `fermionflow` flag -1). The $\tilde{g}d\tilde{d}$ helicity amplitude does not use the conjugated version, since none of the involved wavefunctions have negative `fermionflow` flag.

2.3 Colour algebra

In this section we discuss how the computation of colour coefficients in scattering amplitudes of states which are charged under $\text{SU}(3)_C$ is performed. There are two main motivations to dedicate special attention to this aspect.

JHEP06(2011)128

First, multiparton amplitudes, i.e., amplitudes with many external quarks and gluons, are phenomenologically very important as they correspond to the leading order approximation of multi-jet production (by themselves or in association with other particles) in high energy collisions and especially at hadron colliders. These processes are the major backgrounds to many new-physics signals, so an accurate description of these final states is essential. In this case, not only the complexity of colour computation grows factorially but also the amount of information to be stored grows at the same rate. Efficiency and improved algorithms are therefore necessary.

Second, new-physics models possibly feature states in exotic colour representations or non-standard colour structures in interaction vertices, which need to be taken into account. A generic interaction vertex may also have a complicated structure with several colour factors in front of different Lorentz structures.

The needs above require both flexibility and efficiency of the colour algebra module, a challenge that is not easy to meet.

Prior versions of MadGraph hard-coded three colour structures δ_{ij}, T^a_{ij} and f^{abc} as well as identities for summing over the colour indices. The number of colours was explicitly set to 3 in these identities. Numeric values for the colour matrix where computed by summing over the colour indices using integer operations on the numerator and denominator separately. The resulting colour matrix was exact, but lacked flexibility. The colour structure of the interaction was inferred based on the colour of the participating particles. Models that had new colour interactions required the user to explicitly code new colour structures, which required detailed knowledge of the MadGraph code, significantly restricting the types of new interactions that could be implemented.

The solution adopted in MadGraph 5 is to have all the colour objects and their algebra coded in a symbolic way. This approach has many advantages, the most important one being complete flexibility.

For example, one important aspect in having efficient colour computations is that of the choice of the colour basis. Several possibilities have been proposed in the literature [59–61] with the aim to make the computation of colour symbolically and/or numerically efficient at the amplitude level. All these choices can be easily adopted in our implementation as the colour algebra is dealt with symbolically and several different basis and representations can be present at the same time and used as needed. The colour algebra is realized through objects, such as the Gell-Mann matrices or structure functions, whose products and combinations can be easily simplified by algebraic reduction rules, among which

$$(T^a)^i_j (T^a)^k_l = \frac{1}{2}\left(\delta^i_l \delta^k_j - \frac{1}{N_c}\delta^i_j \delta^k_k\right) \tag{2.1}$$

plays a central role. Other objects, such as, δ^{ab}, f^{abc} and d^{abc} are defined in terms of linear combinations of traces of Gell-Mann matrices, and any colour factor can be easily simplified in a recursive way.

Our default algorithm goes as follows. The colour factor corresponding to each diagram is linearly decomposed over a complete and orthogonal (in the large N_c limit) basis which is constructed at the same time. This allows the automatic organization of the full amplitude

JHEP06(2011)128

into gauge invariant subamplitudes $\mathcal{A}_i$ (normally called dual or colour-ordered amplitudes), each de facto corresponding to a given colour flow i, so that

$$\mathcal{M} = \sum_i \mathcal{C}_i \mathcal{A}_i \,. \tag{2.2}$$

The form above is the basis for further manipulations, as it can be now used in different ways depending on the complexity of the calculation itself. In the case of a limited number of partons in the amplitude, it is squared by analytically computing the colour matrix $\mathcal{C}_{ij} = \sum_{\text{colours}} \mathcal{C}_i \mathcal{C}_j^*$, which is then stored in memory and written in the output file. This is the default approach followed in this version of MadGraph 5 where all diagrams are computed and the colour ordered amplitudes obtained. At this point we note that in the presence of identical external particles, typically gluons, many of the entries of the colour matrix are equal due to simple symmetry properties. Such symmetries are efficiently exploited to reduce the effective number of computations needed to determine the full $\mathcal{C}_{ij}$. This allows MadGraph to calculate amplitudes with up to 7 gluons. Beyond 7 gluons, explicitly calculating the amplitude associated with each Feynman diagram is not viable and recursive relations need to be employed [62, 63]. These allow one to calculate the $\mathcal{A}_i$ directly and prove that their complexity is only polynomial. The problem of the factorial growth therefore remains only for the colour sum. Several techniques have been proposed, the most commonly employed is the idea of recursively computing $\mathcal{M}$ at fixed colour for the external states and then randomly choose colour configurations [10, 14, 15]. Another possibility, which is born out of eq. 2.2, is to organize the calculation as an expansion in $1/N_c$. This is the approach that is currently under investigation to calculate multiparton amplitudes at the tree and loop level and also to combine real and virtual corrections in NLO computations, following the approach of ref. [64].

The other main advantage of treating the colour algebra symbolically is a straightforward implementation of new colour structures. As an example, we here give a detailed description of the necessary algebra for colour sextets and colour triplet ϵ tensors. Higher dimensionality colour representations can be implemented in a similar way.

The $\epsilon^{ijk}(\bar{\epsilon}_{ijk})$ object is the totally antisymmetric tensor of three colour triplet (antitriplet) indices. The algebraic relations needed for ϵ and $\bar{\epsilon}$ are:

$$\epsilon^{ijk}\bar{\epsilon}_{ilm} = \delta^j_l \delta^k_m - \delta^j_m \delta^k_l$$

where $(\epsilon^{ijk})^* = \bar{\epsilon}_{ijk}$. For colour sextets, we need three new colour objects $(K_6)^A_{ij}$, $(\overline{K_6})^{ij}_A$ and the sextet representation $(T^a_6)^A_B$. K_6 ($\overline{K_6}$) is the symmetric tensor contracting a colour sextet (antisextet) index and two colour antitriplet (triplet) indices (i.e., the Clebsch-Gordan coefficient), while the T_6 describes the interaction of a gluon with a sextet. As in the colour triplet implementation described above, the sextet delta function is denoted as a two-index δ^m_n. The complete set of needed algebraic relations for these objects can be found in the appendix of ref. [65]. They can be easily reproduced by starting from

$$(K_6)^A_{ij}(\overline{K_6})^{kl}_A = \frac{1}{2}(\delta^l_i \delta^k_j + \delta^k_i \delta^l_j).$$

JHEP06(2011)128

Together with fundamental anti-commutation relation, the colour matrix and colour flows of any diagram involving colour sextet particles can be calculated.

As a final remark, we note that a small technical complication arises if the parton level events are passed to a parton shower. In this case the writing of the event in the Les Houches Accord (LHA) at leading-N_C colour strings is needed. The ϵ tensor can be handled by inserting new colour labels in the event and even though an apparent violation of the colour flow arises, this can be interpreted correctly by the parton-shower (see ref. [66]). For colour sextets no convention has been established. A simple solution is to note that in the planar "double line" colour flow notation, a colour sextet is contracted with a K_6, to give the equivalent pair of triplet lines. This can then be written into the LHA event, by using a negative antitriplet label for the second triplet, and vice versa for a second antitriplet. A parton shower program reading the event must then treat the continued colour flow, keeping track of colour sextets and antisextets appropriately.

2.4 Decay chains

Whereas decay chains in MadGraph 4 were generated in the same way as regular process generation (by dressing topologies with particle and interaction information to create diagrams), decay chains in MadGraph 5 are defined using successive chains of processes. The core process can have a number of decay processes defined, and each decay process can again have a number of decay processes defined, and so on. This treatment allows for quick and efficient generation of decay chains of virtually unlimited length.

The helicity amplitude calls for the individual processes (core process and each decay process) are generated separately, which allows for very efficient treatment of multiprocesses, where multiple processes have the same decaying final state. As a simple example consider $pp \to W^+$ with $W^+ \to l^+\nu_l$, which gives the core processes $u\bar{d} \to W^+$, $\bar{d}u \to W^+$, $c\bar{s} \to W^+$ and $\bar{s}c \to W^+$, and the decay processes $W^+ \to e^+\nu_e$, $W^+ \to \mu^+\nu_\mu$ and $W^+ \to \tau^+\nu_\tau$. The total number of subprocesses is obtained by combining the four core processes with the three decay processes. With the helicity amplitudes for each of the core processes and each of the decay processes already generated, creating the subprocesses only amounts to replacing the final state wavefunction corresponding to the W^+ by the wavefunctions corresponding to the decays into $e^+\nu_e$, $\mu^+\nu_\mu$ and $\tau^+\nu_\tau$ respectively.

The procedure gets slightly more complicated when decay processes have multiple diagrams- in this case, the diagrams for the core process need to be multiplied together with the diagrams for the decay.

The resulting matrix elements contain full spin correlations and Breit-Wigner effects, but are not valid far from the mass peak, where non-resonant diagrams might give significant interference effects. The tails of the Breit-Wigner distributions for the specified decaying particles are therefore cut off in MadEvent, using the run parameter `bwcutoff`, at $M \pm \Gamma * $`bwcutoff` (by default set to 15).

Process generation, as well as event generation with MadEvent, has been successfully tested up to 14 final state particles at the time of writing this paper.

JHEP06(2011)128

3 Outputs

Previous versions of MadGraph could only output matrix elements in Fortran 77. The modularized design of MadGraph 5 allows easy implementation of outputs in any language or user desired format. The UFO output and ALOHA implementations allow for the same flexibility in the output of models and helicity amplitude routines.

At present, matrix element output is available in Fortran 77, C++ and Python. The Fortran 77 output is in the form of MadEvent directory output (see section 3.1), or standalone matrix element evaluation in the form of MadGraph standalone directory output. For C++, presently available output formats are standalone matrix element evaluation output, and dedicated output for Pythia 8 (see section 3.2). The Python matrix element code output is currently used internally in MadGraph 5 to perform model consistency checks during process generation as described in section 4.3 below. Implementation of other output formats, either in any of the currently supported languages or different ones, can be easily done based on the existing output format implementations.

Finally, one of most useful outputs of MadGraph has always been the drawings of the Feynman diagrams. Current new features (e.g., generation of processes with long decay chains) as well as planned ones (e.g., generation of loop diagrams) required a new algorithm to be implemented. This is described in the last subsection.

3.1 Multiprocess generation and MadEvent event generation

Simultaneous generation of multiple processes (e.g. $pp \to jj$, using multiparticle labels such as $p/j = g/d/u/s/c/\bar{d}/\bar{u}/\bar{s}/\bar{c}$) has been considerably optimized in MadGraph 5, using improved algorithms both for process generation and for event generation.

On the process generation side, the time spent attempting to generate processes which have no diagrams (such as $u\bar{d} \to gg$) is minimized on one hand by checking any conserved quantum numbers, such as electrical charge, provided by the model, and on the other hand in a model-independent manner by keeping a memory of failed processes, and ignoring any process which corresponds to a crossing of such a failed process. This is done before applying crossing symmetry breaking conditions such as required or forbidden s-channel propagators. Furthermore, processes with mirrored initial state (such as $gu \to \gamma u$ and $ug \to \gamma u$) are automatically recognized and combined into a single process. To further speed up the generation, diagrams from crossed processes are reused in the diagram generation (so that the diagrams for, e.g., $gg \to u\bar{u}$, $ug \to ug$ and $u\bar{u} \to gg$ are generated only once, and then reused with leg numbers replaced as needed).

The organization of subprocess directories for multiprocess event generation in MadEvent has been revamped. While MadEvent 4 was already combining processes with identical matrix elements (such as $gg \to u\bar{u}$ and $gg \to d\bar{d}$), MadGraph 5 combines all processes with the same spin, colour and mass of external particles into a single subprocess directory. The diagrams of these subprocesses are matched to combined integration channels, so that a single integration channel will perform single diagram enhanced phase space integration [5] for all subprocesses with the corresponding diagram. Diagrams whose pole structure differ only by permutations of the final state momenta are also combined,

JHEP06(2011)128

Process	Subproc. dirs.		Channels		Directory size		Event gen. time	
	MG 4	MG 5	MG 4	MG 5	MG 4	MG 5	MG 4	MG 5
$pp \to W^+j$	6	2	12	4	79 MB	35 MB	3:15 min	1:55 min
$pp \to W^+jj$	41	4	138	24	438 MB	64 MB	9:15 min	4:19 min
$pp \to W^+jjj$	73	5	1164	120	842 MB	110 MB	21:41 min*	8:14 min*
$pp \to W^+jjjj$	296	7	15029	609	3.8 GB	352 MB	2:54 h*	46:50 min*
$pp \to W^+jjjjj$	-	8	-	2976	-	1.5 GB	-	11:39 h*
$pp \to l^+l^-j$	12	2	48	8	149 MB	44 MB	21:46 min	3:00 min
$pp \to l^+l^-jj$	54	4	586	48	612 MB	83 MB	2:40 h	11:52 min
$pp \to l^+l^-jjj$	86	5	5408	240	1.2 GB	151 MB	49:18 min*	16:38 min*
$pp \to l^+l^-jjjj$	235	7	65472	1218	5.3 GB	662 MB	7:16 h*	2:45 h*
$pp \to t\bar{t}$	3	2	5	3	49 MB	39 MB	2:39 min	1:55 min
$pp \to t\bar{t}j$	7	3	45	17	97 MB	56 MB	10:24 min	3:52 min
$pp \to t\bar{t}jj$	22	5	417	103	274 MB	98 MB	1:50 h	32:37 min
$pp \to t\bar{t}jjj$	34	6	3816	545	620 MB	209 MB	2:45 h*	23:15 min*

Table 2. Number of subprocess directories, number of integration channels for the initial run ("survey") of the event generation, size of the directory after one run generating 10,000 events, and run times for generating 10,000 events, comparing MadGraph/MadEvent 4 output ("MG 4") with grouped subprocess output ("MG 5"). For all processes, $p = j = g/u/\bar{u}/c/\bar{c}/d/\bar{d}/s/\bar{s}$, $l^\pm = e^\pm/\mu^\pm$. The run times for 0-, 1- and 2-jet processes are for a Sony VAIO TZ laptop with 1.06 GHz Intel Core Duo CPU running Ubuntu 9.04, gFortran 4.3 and Python 2.6, while the 3-, 4- and 5-jet run times (marked by *) are for a 128-core computer cluster with Intel Xeon 2.50GHz CPUs. $pp \to W^+ + 5j$ is not possible to run with MadGraph/MadEvent 4.

to further minimize the number of integration channels. This means that for a process like $pp \to l^+l^- + 3j$, there will be only 5 subprocess directories, corresponding to the subprocess groups $gg \to l^+l^-gqq$, $gq \to l^+l^-ggq$, $gq \to l^+l^-qqq$, $qq \to l^+l^-ggg$, and $qq \to l^+l^-gqq$, as compared to 86 directories in MadGraph 4 (see table 2).

Taken together with the identification of initial state mirror processes, the number of integration channels for the initial cross section determination run ("survey") is significantly reduced, see table 2. Also for the subsequent event generation run ("refine"), the number of integration channels is usually reduced. The required disk space is reduced by a factor corresponding to the reduction in number of integration channels.

To further improve parallel running of the resulting configurations, we have implemented the ability to split up channels with large contribution to the cross section into multiple sub-channels, each generating a fraction of the events for the channel in question. This results in shorter generation times and more equal work load for jobs submitted to a cluster, as well as considerably more stable unweighting efficiency for the integration. Many further measures to speed up and improve the stability of the generation have also been taken, including a new initial guess for the shapes of the integration variables, leading to considerably better stability of the results. The resulting reduction in run times for a few sample processes are also given in table 2.

JHEP06(2011)128

3.2 Matrix element libraries for PYTHIA 8

PYTHIA is one of the most widely used multipurpose event generators, which includes matrix element evaluation, parton showering, hadronization, particle decays and underlying events in a single framework. Matrix elements for PYTHIA have historically been implemented by hand. The most recent implementation of PYTHIA, the C++ version PYTHIA 8, allows matrix elements for $2 \to 1$, $2 \to 2$ and $2 \to 3$ processes to be provided by external programs. The flexibility in output formats in MADGRAPH 5 has allowed us to implement dedicated matrix element output for PYTHIA 8, thereby effectively removing the need for implementation of any matrix elements for PYTHIA by hand.

Let us now describe the main features of this new implementation for the readers interested in more technical details.

The new matrix elements are given in the form of classes inheriting from the internal base class `SigmaProcess`. Such process classes need to implement a number of member functions, providing PYTHIA with information about the process (initial states, external particle masses, s-channel resonances, etc.), as well as functions to evaluate the matrix elements for all included subprocesses and select final-state particle id's and colour flow for each event. During event generation, PYTHIA 8 calls the matrix element classes with given momenta for the external particles. Starting from v.8.150, PYTHIA also makes model parameters for new models (read in using the Les Houches interface for BSM model parameters [67]) available to the resulting process class through an instance of the class `SusyLesHouches`, allowing for the implementation and simulation of processes in any new physics model.

The resulting matrix elements are treated by PYTHIA in exactly the same way as the processes available in the PYTHIA core code. MADGRAPH 5 also provides classes for evaluation of all model parameters needed for the matrix element evaluation as well as the helicity amplitudes used by the matrix elements. Standard model parameters are extracted from internal PYTHIA parameters, while new physics parameters are read in from BSM-LHA files. Just as in the regular MADEVENT output, any model parameters based on the strong coupling constant are recalculated event by event to use the running value of α_s (as provided by PYTHIA), while fixed model parameters are initialized at the time of process initialization.

The PYTHIA 8 output of MADGRAPH 5 is a library called `Processes_`*`model_name`*, which is automatically created in a new directory under the PYTHIA 8 base directory. This library contains source code files for all generated processes, model parameters, and helicity amplitudes, as well as a makefile to compile the library and place it in the `lib` directory of PYTHIA. As an extra help to the inexperienced user, an example main program file is also created in the `examples` directory together with a dedicated makefile, which shows how to generate events from the implemented processes. These main program files can be edited, compiled, and run directly from the MADGRAPH 5 command line by running the `launch` command, or compiled and run externally.

If multiple processes are generated in MADGRAPH 5, those processes will automatically be arranged in different process classes according to the initial and final states. This

JHEP06(2011)128

ordering uses the same machinery as the new organization of MadEvent subdirectories described in section 3.1 above. In order for Pythia to perform event generation, it needs all subprocesses inside a given process to have the same spin and mass of external particles. This combination of subprocesses into single classes allows for further optimization of the matrix element calculation, in that the helicity wavefunctions and amplitudes can be calculated once and for all, and then be reused between all the different subprocesses in a process class.

As a cross check, we have compared the cross section results for the automatically generated processes with internal Pythia processes for a large variety of processes in the Standard Model and the MSSM, with perfect agreement.

3.3 Diagram drawing

In MadGraph 4 the amplitude generation and diagram drawing is limited to handling three and four point vertices. In addition, the diagram drawing is based on a length minimization procedure, whose convergence is not a priori guaranteed and sometimes creates lines with zero length. To overcome these limitations, a completely new algorithm has been implemented in MadGraph 5. The basic idea is to associate to each vertex a level (i.e., to organise the verteces in classes each one characterized by a "distance" from the left end of the diagram) defined by the following simple rules:

- The vertices associate to initial particles are always at level one.
- All vertices attached to a t-channel propagator are set at level one.
- The difference of level between the endpoints of an s-channel propagator equals one.

As a boundary condition, all external lines are associated to a level at the start of the procedure: initial state particles are set to level zero, while all final state particle's levels are set to the maximal possible one plus unity.[1]

The position of the vertices are then computed such that all vertices at a given level lie equally spaced on the same vertical line.

The main challenge of this method consists in avoiding line crossing between propagators/final state particles. In most situations, a simple re-ordering of the vertices at each level is enough to avoid line crossings. The order should be such as vertices connected to the first vertex of the previous level comes first, then the ones linked to the second one and so on. If a line connects two non-adjacent levels, which happens for final state particles, this methods needs an additional trick. Then, the intermediate level needs a special configuration — i.e., not equally spaced — in order to avoid any possible line crossing. This is dealt with by adding a fake vertex (which is not drawn) to the intermediate level. Since we keep the "equally spaced" rule with the fake vertex, this creates the appropriate gap. The resulting algorithm is very fast and efficiently generates clean diagrams with any number of external particles.

[1] Options are available that allow to modify the level of the external partons edges, resulting in different-looking graphs.

JHEP06(2011)128

Model Class	Model name	Description/Comments	In MG5
Standard Model	sm	Several restrictions/simplifications available	✓
	heft	Top and W loops for the Higgs through dim-5 operators	✓
SM extensions	4Gen	Fourth Generation model with full CKM4	✓
	SMScalars	Extra $O(n)$ scalar sector	✓
	Hidden	Hidden Abelian Higgs Model	
	Hill	Hill Model	
	2HDM	The general Two Higgs-Doublet Model	✓
	TripletDiquarks	SM plus triplet diquarks	✓*
	SextetDiquarks	SM plus sextet diquarks	✓*
SUSY models	mssm	Minimal Supersymmetric extension of the SM	✓
	nmssm	Next-to-Minimal MSSM	✓
	rmssm	R-symmetric MSSM	
	rpvmssm	R parity violating MSSM	
Extra-Dim models	3-site	Minimal Higgless Model (3-site Model)	
	MUED	Minimal UED	
	LED	Large Extra Dimensions	
	RS	Randall-Sundrum	✓
	HEIDI	Compact HEIDI	
EFT's	ChiPT	Chiral perturbation theory	
	SILH	Strongly Interacting Light Higgs	
	MWT	Technicolor	

Table 3. Selection of models that are currently available in FeynRules and can be used in MadGraph 5. The last column indicate model which are present by default in the current release of MadGraph 5. An asterisk (*) indicates that the model in the MadGraph 5 library is a simplified version of the complete model.

4 Models

The MadGraph 5 library of models is built upon that of FeynRules [17] and written in the UFO format [20]. Backward compatibility with the current models of MadGraph 4 is supported as long as no particles with spin 3/2 or higher are present in the model.

The set of currently publicly available models from the FeynRules wiki page is shown in the table 3. Several models are currently available in the MadGraph 5 release - these are marked with a "✓" in the table. Besides the models in the table, also some additional models used for examples in this paper are included in the release, e.g., the four-fermion interaction models used in section 6.2.

4.1 Inheriting models from FeynRules: UFO and ALOHA

Any local quantum field theory can be identified by:

- A set of particles and their quantum numbers (spin, charges, etc.).
- A set of parameters (masses, coupling constants, etc.).
- A set of interactions among the different particles.

The most efficient, reliable and compact way to encode that information is by directly writing a Lagrangian with (matter and interaction) fields carrying the desired quantum

JHEP06(2011)128

numbers and by using well-known text-book rules to extract the Feynman vertices. This is what FeynRules does in a fully automatic way. Once the vertices are obtained, the issue of passing this information to a matrix element generator like MadGraph arises. For example, for MadGraph 4, FeynRules writes the output files exactly in the format needed by the code. This procedure, however, in addition to being very heavy to maintain for FeynRules developers, has the drawback that it suffers from the same intrinsic limitations of the MadGraph 4 model format itself.

The purpose of the Universal FeynRules Output (UFO) [20] is to overcome possible limitations due to specific matrix element generators and translate *all* the information about a given particle physics model into a Python module that can easily be linked to any existing code. This output is complete and independent of the matrix element generator, allowing full flexibility and improvements. It saves the model information in an abstract (generator-independent) way in terms of Python objects and classes, which in case of MadGraph 5 can be directly linked to the code.

Once the information of a model is available in the UFO, it can be used by ALOHA to automatically write the HELAS library corresponding to the corresponding Feynman rules. ALOHA, which is written in Python, produces the complete set of routines (wavefunctions and amplitudes) that are needed for the computation of Feynman diagrams at leading as well as at higher orders. The representation is language independent and outputs in Fortran, C++, Python are currently available.

In so doing all the intrinsic limitations regarding the possibility of generating arbitrary new physics model process of MadGraph 4 are overcome. As already explained above, MadGraph 4 is based on the HELAS library that encodes a limited set of Lorentz structures. While extensions are possible and have been done for several important cases (such as spin-2 [68] and spin-3/2 [69] particles), they entail a tedious work of writing and testing all new routines by hand. ALOHA via the UFO fully automates this procedure. In addition, complicated interactions that feature non-factorizable colour and Lorentz structures, such as those showing up in the counterterms and R_2 vertices at NLO [70], which cannot be handled by MadGraph 4, are now fully supported.

4.2 Model restriction files

In phenomenology applications, it is often convenient to fix some parameters at some given values (e.g., masses / CKM parameters / couplings, etc.). Such restrictions might allow important gains both in terms of speed and also in size of the generated matrix element code. MadGraph 5 allows to restrict a given UFO model based on a numerical evaluation of all couplings with parameters from a BSM-LHA file [67] (note that analytical model restrictions can also be performed directly inside FeynRules).

From a given BSM-LHA file (that we call a *restriction file*), MadGraph can evaluate the value of the internal parameters and of the couplings. The model is then modified according to the following rules:

- All vanishing parameters are removed from the model, and all parameters with value equal to unity are fixed to this value.

JHEP06(2011)128

- All parameters belonging to the same `LHA` block with identical values are represented by a single parameter.
- All couplings with identical values are represented by a single coupling.
- All interactions linked to a vanishing coupling are removed from the list of interactions.

These steps allow MadGraph 5 to optimise the matrix element output, and the output of multi-process generation (see section 3.1).

In order to avoid the user setting the parameters of the model in a way which is inconsistent with the restricted model, we also modify the param_card associated with the model, removing all parameters that have been fixed by the restriction. The default value for all remaining parameters is set to that given in the restriction file. Note that care is needed when the user designs the restriction file, to ensure that the result corresponds to what it is expected, and that the resulting param_card includes all desidered parameters.

Model restrictions are used for several models present in MadGraph 5, including SM and MSSM. By default when a model is loaded, MadGraph 5 applies the restriction defined in the file `restrict_default.txt` in the corresponding model directory. For the Standard Model, the default restriction sets the CKM matrix to be diagonal and sets the mass of the first and second generation fermions to zero. The user can bypass the default restriction by adding "`-full`" to the model name when importing the model, or apply a different restriction file by adding "-restriction" to the model name, corresponding to a restriction file `restrict_`*`restriction`*`.dat`.

4.3 Consistency checks for processes and models

For new model implementations, either from FeynRules [17] or by directly providing the UFO model format used in MadGraph 5, it is crucial to be able to check the consistency of the new models. To this end, MadGraph 5 features a series of consistency checks for processes:

1. Helicity amplitude calls and the helicity amplitude implementations are checked by calculating the specified processes with multiple permutations of the external particles in the diagram generation in a given phase space point, checking that the value of the matrix element is identical for the different permutations. This efficiently checks several aspects of the model implementation: the relation between the Lorentz and colour structures, the implementation of the related helicity amplitudes with different wavefunction decomposition, and the effects of fermion flow violation and charge conjugation.

2. Gauge invariance is checked by calculating the matrix element in a random phase space point with the wavefunction of an external massless vector boson replaced by its momentum (for processes with external massless vector bosons). If gauge invariance is satisfied, the resulting matrix element is zero (within numerical precision).

JHEP06(2011)128

3. Invariance of the matrix element by Lorentz transformations is checked by comparing the matrix element value before and after a series of Lorentz boosts.

We have found that altogether, these checks provide a powerful way to validate model implementations in MadGraph 5.

5 Validation and speed benchmarks

5.1 Validation

Once the checks based on symmetries, gauge invariance, and Lorentz invariance described above are satisfied, one can perform the "physics" validation by comparing results for the evaluation of the matrix element in given points of the phase space and/or integrated cross sections with other generators or analytical calculations. To validate both MadGraph 5 and the models provided with it, we have extensively compared squared matrix elements computed point-by-point in the phase-space with those obtained in MadGraph 4.

Since MadGraph 5 supports different input model formats (the MadGraph 4 format models and the new UFO models) and is able to create output in different languages (Fortran 77, C++), we have compared the MadGraph 4 results with MadGraph 5 in the following three configurations:

- Using the UFO model as input and choosing to export the matrix element in Fortran 77. (In this context the helicity amplitude routines are created by ALOHA).
- Using the UFO model as input and choosing to export the matrix element in C++. (In this context the helicity amplitude routines are created by ALOHA).
- Using the MadGraph 4 model as input (therefore the only output available is Fortran 77 and we use the HELAS package).

A summary of the checks performed is presented in table 4. Those processes (more than three thousand in total) are all in perfect agreement between all three configurations and the MadGraph 4 value.

In addition to the squared matrix element tests, we have also performed a systematic comparison at the cross-section level for $2 \to 2$ and $2 \to 3$ both in the SM and in the MSSM. This comparison was done with the Feynrules web validation interface [71] with respect to MadGraph 4, CompHEP/CalcHEP, and Whizard.

5.2 Speed benchmarks — Process generation

It can be interesting to compare the time required for the generation of complete MadEvent directories for different processes in MadGraph 4 and MadGraph 5. Note that this is the time for the generation of the matrix element output code and additional files needed for phase space integration and event generation, not the time for evaluating the matrix element values using the generated code. A time comparison for evaluation the generated matrix element is given in the next section.

JHEP06(2011)128

model	process class	information	number of processes
SM	$AA \to AA$	Only first generation for fermions	249
SM	$AA \to AA$	No first generation of fermions	589
SM	$BB \to BBB$		86
SM	$BB \to BF_{1,2}F_{1,2}$		46
SM	$F_{1,2}F_{1,2} \to BF_{1,2}F_{1,2}$		40
SM	$F_{1,2}F_{1,2} \to BF_{1,2}F_{1,2}$		216
SM	$BB \to BBBB$		55
MSSM	$PP \to \chi^0\chi^0$		50
MSSM	$PP \to \chi^+\chi^-/\tilde{g}\tilde{g}$		26
MSSM	$PP \to \tilde{L}\tilde{L}$		55
MSSM	$PP \to \tilde{Q}_{1,2}\tilde{Q}_{1,2}$		188
MSSM	$PP \to \tilde{Q}_3\tilde{Q}_3$		71
MSSM	$\tilde{Q}_{1,2}\tilde{Q}_{1,2} \to \tilde{L}\tilde{L}$		208
MSSM	$\tilde{Q}_{1,2}\tilde{Q}_{1,2} \to \tilde{Q}_{1,2}\tilde{Q}_{1,2}$		285
MSSM	$PP \to \tilde{Q}_{1,2}\tilde{Q}_{1,2}V$	only SM vector bosons included	564
MSSM	$VV \to V\chi^+\chi^-$		71
MSSM	$PP \to L^+L^-\chi^0\chi^0$		200
MSSM	$VV \to VV\chi^+\chi^-$		177
HEFT	$BB \to BB$	including the CP-odd Higgs	62
HEFT	$gg \to H + ng$	$n = 1, 2, 3, 4$	4
RS	$AA \to AA$	Only first generation for fermions	362
RS	$F_3F_3 \to AA$		248
RS	$F_{1,3}F_{1,3} \to F_{1,3}F_{1,3}B$		452

Table 4. Classes of processes used to compare the output of MadGraph 5 with MadGraph 4. The different letters designate classes of particles: A contains all the particles of the model; F_i contains the i^{th} generation of fermions of the model (No indices means all generations allowed); $L^\pm$ contains all the leptons of the model; V contains all the vector bosons of the model; B contains all the bosons of the model (i.e., the vector and the scalar particles); χ^0 contains all the neutralinos; $\chi^\pm$ contains all the charginos; $\tilde{Q}_i$ contains the i^{th} generation of squarks; $\tilde{L}$ contains the full set of sleptons.

Table 5 shows the time needed for a number of example processes, including multi-processes, high-multiplicity final state processes, and decay chain processes. As can be seen from the table, the main gains in speed are in complicated processes: processes with a large number of subprocesses due to large multiplicities of multiparticle labels (such as $pp \to jjje^+e^-$), processes with many external particles (such as $gg \to 5g$ or $e^+e^- \to 6e$), and in decay chain processes, where the speedup can be several orders of magnitude with respect to previous versions of MadGraph.

5.3 Speed benchmarks — Matrix element evaluation

MadGraph 5 is not only faster than its predecessors in generating code for complicated processes, the produced matrix element code is also faster and more compact. This is

JHEP06(2011)128

Process	MadGraph 4	MadGraph 5	Subprocesses	Diagrams
$pp \to jjj$	29.0 s	25.8 s	34	307
$pp \to jjl^+l^-$	341 s	103 s	108	1216
$pp \to jjje^+e^-$	1150 s	134 s	141	9012
$u\bar{u} \to e^+e^-e^+e^-e^+e^-$	772 s	242 s	1	3474
$gg \to ggggg$	2788 s	1050 s	1	7245
$pp \to jj(W^+ \to l^+\nu_l)$	146 s	25.7 s	82	304
$pp \to t\bar{t}$+full decays	5640 s	15.7 s	27	45
$pp \to \tilde{q}/\tilde{g}\,\tilde{q}/\tilde{g}$	222 s	107 s	313	475
7 particle decay chain	383 s	13.9 s	1	6
$gg \to (\tilde{g} \to u\bar{u}\tilde{\chi}_1^0)(\tilde{g} \to u\bar{u}\tilde{\chi}_1^0)$	70 s	13.9 s	1	48
$pp \to (\tilde{g} \to jj\tilde{\chi}_1^0)(\tilde{g} \to jj\tilde{\chi}_1^0)$	—	251 s	144	11008

Table 5. Time for generation of complete MadEvent directories (with the exception of $gg \to 5g$, for which a Fortran standalone directory was generated) for a selection of processes, for MadGraph 4 and MadGraph 5. All processes have $p = j = g/u/\bar{u}/c/\bar{c}/d/\bar{d}/s/\bar{s}$, $l^\pm = e^\pm/\mu^\pm/\tau^\pm$, $\nu_l = \nu_e/\nu_\mu/\nu_\tau$ and $\bar{\nu}_l = \bar{\nu}_e/\bar{\nu}_\mu/\bar{\nu}_\tau$. $\tilde{q}/\tilde{g}$ in the table corresponds to $\tilde{d}^{(*)}_{l/r}/\tilde{u}^{(*)}_{l/r}/\tilde{s}^{(*)}_{l/r}/\tilde{c}^{(*)}_{l/r}/\tilde{g}$. For $t\bar{t}$+full decays (meaning $pp \to (t \to b\,q/l^+\,\bar{q}/\nu_l)(\bar{t} \to \bar{b}\,q/l^-\,\bar{q}/\bar{\nu}_l)$), the MadGraph 4 process generation was split up in 12 different process definitions to reduce the number of failed process attempts. The "seven particle decay chain" was $gg \to (\tilde{g} \to u(\bar{\tilde{u}}_l \to \bar{u}(\tilde{\chi}_2^0 \to Z\tilde{\chi}_1^0)))(\tilde{g} \to ud\tilde{\chi}_1^-)$. The number of subprocesses and diagrams are quoted after combination of subprocesses with identical matrix elements. All processes are generated with maximal number of QCD vertices. All numbers are for a Sony VAIO TZ laptop with 1.06 GHz Intel Core Duo CPU running Ubuntu 9.04, gFortran 4.3 and Python 2.6.

thanks to the new diagram generation algorithm, which allows for improved recycling of subdiagram wavefunctions between different diagrams, reducing the number of helicity wavefunction calls (as discussed in sections 2.1 and 2.2). Table 6 shows a comparison of the number of function calls and run time for matrix element evaluation using HELAS and ALOHA routines, relative to MadGraph 4.

From the table, we see how the improved wavefunction call optimisation translates to considerably improved run times, especially for complicated processes.

6 BSM example applications

6.1 Non-standard colour structures: ϵ^{ijk} and colour sextets

The phenomenology of diquark resonances has recently become popular, since these particles could be among the first new physics particles to be observed at the proton on proton collider LHC [65]. Such diquarks have the quantum numbers of two valence quarks, and must therefore be either colour sextets or colour anti-triplets. In the latter case, the coupling to quarks is through a completely antisymmetric colour triplet ϵ tensor, which is the only way to contract three colour triplet indices. The triplet ϵ tensor is also important in the formulation of R-parity violation supersymmetric models, where e.g. a scalar quark can decay into a pair of Standard Model antiquarks. As discussed in section 2.3, both the sextet colour algebra and the ϵ tensor have been implemented in MadGraph 5.

JHEP06(2011)128

Process	Function calls		Run time relative to MG 4	
	MG 4	MG 5	MG 5+HELAS	MG 5+ALOHA
$u\bar{u} \to e^+e^-$	8	8	1.0	1.1
$u\bar{u} \to e^+e^-e^+e^-$	110	80	0.52	1.4
$u\bar{u} \to e^+e^-e^+e^-e^+e^-$	6668	3775	0.33	0.57
$gg \to gg$	13	13	0.90	0.81
$gg \to ggg$	86	78	0.94	0.94
$gg \to gggg$	811	621	0.99	0.66
$u\bar{u} \to d\bar{d}$	6	6	1.0	1.0
$u\bar{u} \to d\bar{d}g$	16	16	1.0	1.2
$u\bar{u} \to d\bar{d}gg$	85	67	0.74	0.86
$u\bar{u} \to d\bar{d}ggg$	748	515	0.68	0.52
$u\bar{u} \to u\bar{u}gg$	160	116	0.67	0.70
$u\bar{u} \to u\bar{u}ggg$	1468	960	0.48	0.36
$u\bar{u} \to d\bar{d}d\bar{d}$	42	33	0.99	1.2
$u\bar{u} \to d\bar{d}d\bar{d}g$	310	197	0.61	0.74
$u\bar{u} \to d\bar{d}d\bar{d}gg$	3372	1876	0.24	0.19
$u\bar{u} \to d\bar{d}d\bar{d}d\bar{d}$	1370	753	0.18	0.19

Table 6. Number of helicity function calls and run time ratio to MadGraph 4 for matrix element evaluation of matrix element code produced by MadGraph 4 and MadGraph 5 using HELAS, and MadGraph 5 using ALOHA routines. For MadGraph 4, the HELAS library has been used. Note that in the $u\bar{u} \to q\bar{q} + X$ process generations, only QCD interactions have been allowed (`QED=0`). The number of function calls for MadGraph 5 does not depend on whether HELAS or ALOHA is used.

As an example of phenomenology using these implementations, we show in figure 3a the cross sections for different species of colour sextet and antitriplet scalar diquarks D at LHC with 7 TeV c.m. energy. We have included colour sextet diquarks coupling to $uu/cc/tt$, $dd/ss/bb$ and $ud/cs/tb$, and colour antitriplet diquarks coupling to $ud/cs/\ tb$. Note that due to the antisymmetry of the ϵ^{ijk} colour coupling of colour triplet diquarks to quarks, colour triplet diquarks can only couple to off-diagonal flavour quark combinations. The $Dqq^{(\prime)}$ Yukawa coupling constants have been set to 10^{-2} in the figure. Note the factor 2 between the $pp \to D$ production cross sections for sextet and triplet diquarks (for identical Yukawa couplings), due to the different colour factors.

In figure 3b, we show the effect of jet matching between matrix elements and parton showers for charge $+\frac{4}{3}$ sextet diquark production at 7 TeV LHC. p_T distributions for the radiated jets are compared between matched production with MadGraph (using the k_T-MLM matching scheme that is default in MadGraph, with matrix elements for $pp \to D + 0, 1, 2$ jets) and Pythia 6.4 with p_T-ordered showers, and just using the leading order process $pp \to D$ with p_T-ordered Pythia default settings. The mass of the diquark is 500 GeV. It is clear that matching is necessary for a precise description of high-p_T jet radiation in association with diquark production.

JHEP06(2011)128

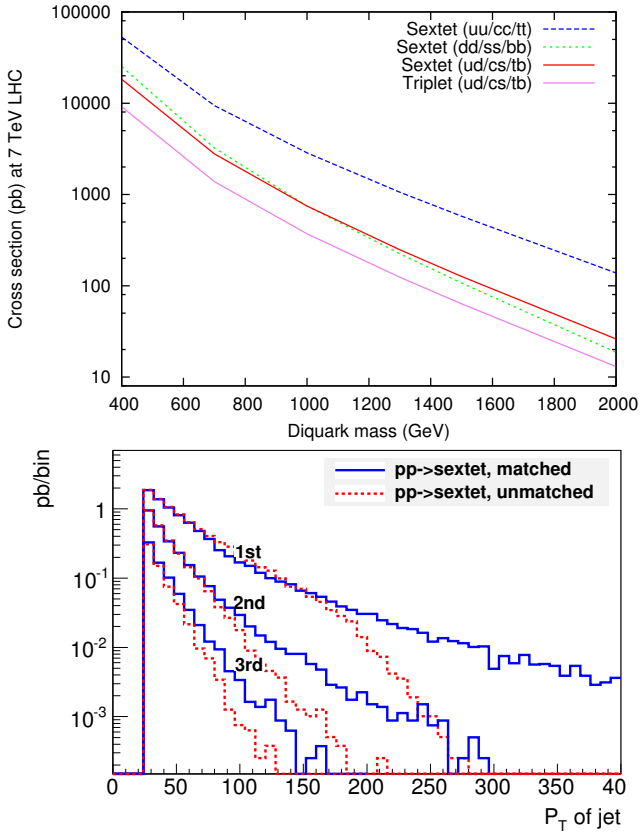

Figure 3. Upper: Cross sections for different types of diquark resonances at 7 TeV LHC. See text for details. Lower: Comparison of p_T for radiated jets between single colour sextet diquark production with jet matching (using MadGraph and Pythia) and without jet matching (using Pythia parton showers for the leading order process $pp \to D$ only). See text for details.

6.2 4-fermion vertices: $uu \to tt$

With the possibility of specifying vertices with arbitrary number of particles, a particular difficulty arises when a vertex has more than two fermions, in which case it is necessary to define the fermion flow in an unambiguous way. The convention chosen by the MadGraph 5 and FeynRules authors is that the fermion flow is defined by the position of the particle in the interaction, with the order being $IOIO\ldots$ where I stands for "incoming" and O stands for "outgoing" fermion. Any number of bosons can be added after the fermions.

This means that, from the MadGraph 5 point of view, the interactions $u\bar{t}u\bar{t}$ and $uu\bar{t}\bar{t}$ are treated in different ways - in the former case, the fermion flows go between u and $\bar{t}$, while in the latter case we have fermion number violating flows $u \to u$ and $\bar{t} \to \bar{t}$. Such fermion number violating multi-fermion vertices are readily treated by the algorithm described in section 2.2, by the use of conjugate Γ matrices for each fermion number violating fermion line.

Of particular interest are four-fermion vertices, which are a common feature of effective theory formulations for physics beyond the Standard Model, and have recently been studied

JHEP06(2011)128

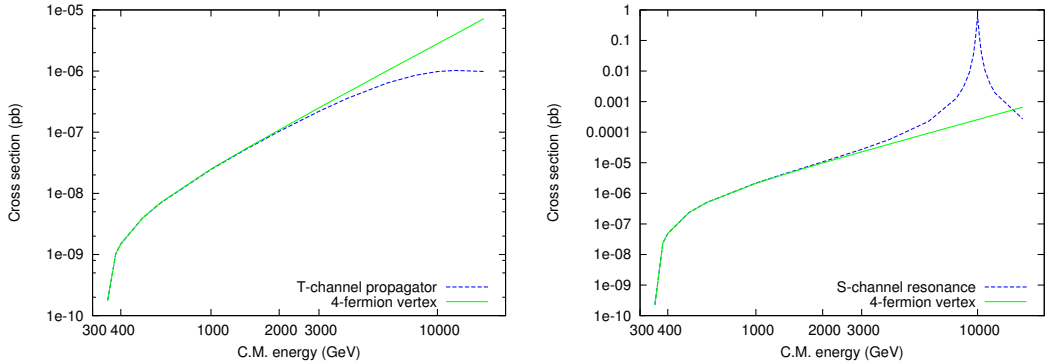

Figure 4. Cross section for $uu \to tt$ as a function of (fixed) beam energy, comparing 4-fermion implementations and the corresponding implementations with explicit propagators. Left: t-channel scalar exchange. Right: s-channel scalar exchange. For both cases, the mass of the propagator is set to 10 TeV.

in the context of top-quark LHC phenomenology [72–74] We are here presenting two examples of four-fermion vertices leading to the process $uu \to tt$ [75], one which corresponds to the exchange of a heavy s-channel propagator (in this case a scalar colour sextet diquark) and one which corresponds to a heavy t-channel propagator (a neutral colour singlet flavour changing scalar). In both cases, we use a mass of 10 TeV for the propagators, and represent the four-fermion vertices by integrating out the appropriate scalar propagator.

In figure 4 we compare the full theories, including the explicit propagators, with the 4-fermion vertex versions of the theories. The figure shows $uu \to tt$ cross sections for the two scenarios as a function of fixed center of mass energy, for some particular coupling values; to compare with a particular collider, the cross sections need to be convoluted with the uu parton luminosity. The sudden turn-on of the cross section is due to the kinematical suppression from the final-state top quark mass.

As expected, the four fermion vertex formulation agrees with the explicit propagator formulation up to a c.m. energy of about 1/10 of the propagator mass. The explicit t-channel propagator makes the cross section level off as the energy gets close to the mass and the exchange momentum term in the denominator of the propagator starts dominating over the mass term, while the explicit s-channel propagator displays the usual Breit-Wigner peak as the energy gets close to the mass. In this case, the width of s-channel propagator is $\Gamma_S = 200$ GeV.

6.3 n-particle vertices: $H + 4g$

MadGraph 5 allows vertices with any number of external particles. Such vertices frequently appear in effective field theories, where non-renormalizable operators are included with some appropriate scale suppression. One of the most phenomenologically important effective field theories for high-energy physics is the addition to the Standard Model of effective couplings between the Higgs boson and gluons through a top quark loop, with the top mass taken much larger than the Higgs boson mass.

JHEP06(2011)128

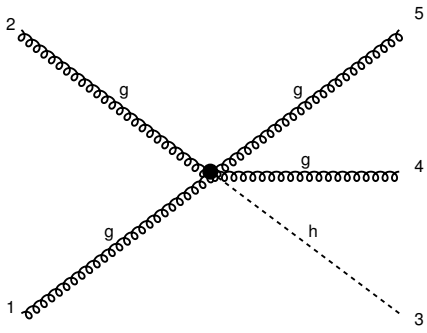

Figure 5. 5-particle diagram by MadGraph 5. This is one of 38 diagrams from the process $gg \to Hgg$ in the Standard Model with effective couplings of the Higgs boson to gluons through a top loop.

While the simplest vertex we can write down in this theory is Hgg, the non-Abelian nature of QCD require us to include two additional operators in the Lagrangian, $Hggg$ and $Hgggg$, coupling the Higgs directly to three and four gluons respectively. While previous versions of MadGraph had to split up the five-particle $Hgggg$ vertex using an auxiliary non-propagating tensor particle, MadGraph 5, in conjunction with ALOHA, can directly handle this vertex in exactly the same way that it handles vertices with lower multiplicity. The corresponding diagram, from the process $gg \to Hgg$, is found in figure 5.

Note that for consistency, only one effective Higgs-gluon coupling vertex can be present in a given diagram. This is made possible by specifying a separate coupling order, `HIG`, for these vertices. Any process generation in this model should therefore be specified with a maximum order `HIG=1`.

The new formulation of Higgs boson couplings to gluons has been thoroughly checked against the implementation in MadGraph 4, and is one of the models included in the MadGraph 5 package.

The same issue happens for a graviton interacting with four gauge bosons. A full validation of the MadGraph 5 implementation the FeynRules RS model against the MadGraph 4 [68] implementation has been performed.

6.4 Chromo-magnetic operator

Recent measurements in top quark pair production performed both at Tevatron and LHC offer an ideal ground to search for new physics effects [74]. If such new physics is at scales higher than those explored at the current colliders it can be efficiently modeled by an effective theory with a non-renormalizable operator suppressed by a energy-scale Λ. In the case of the top quark, only one operator of dimension 6 exists that is not a 4-fermion operator, the so called *chromo-magnetic* operator:

$$\mathcal{L} = \frac{(H\bar{Q})\sigma^{\mu\nu}T^A t G^A_{\mu\nu}}{\Lambda^2} + h.c.,$$

where Λ represents the cutoff of the effective theory. Adding such a term to the Lagrangian of the standard model leads to additional interactions, see figure 6. For consistency, any

JHEP06(2011)128

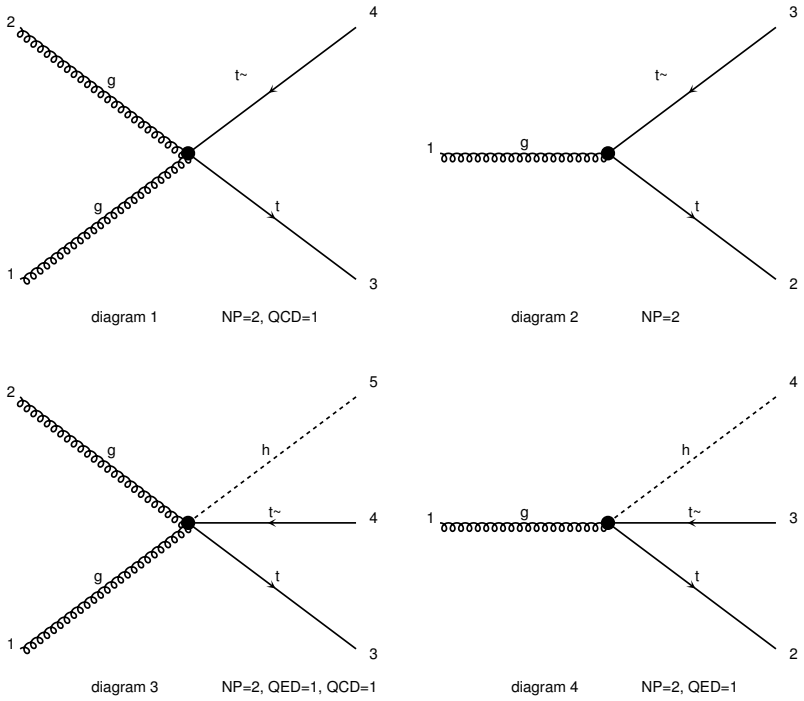

Figure 6. Interactions induced by the chromomagnetic operator.

matrix element should be computed up to order Λ^{-2} and therefore requiring one and only one effective coupling to enter at the squared matrix element level. This can be obtained by defining a third type of coupling (in addition to QED and QCD) associated with the new interactions.

Again the automation of the HELAS routines and the possibility for MadGraph 5 to deal with vertices with arbitrary number of legs, allows a straightforward treatment of such operators. The validation of this model has been performed by comparing the MadGraph 5 results to those obtained in a private (and not fully automatic) implementation of this model inside the MadGraph 4 framework. In addition to automatically providing the matrix element for any process involving the new operators, the evaluation of the cross sections are approximatively four times faster than the previous version.

7 Conclusions and outlook

The complete rewriting of the MadGraph matrix-element generator in Python has allowed us to build on the twenty years of experience gained with the Fortran version and push the code as well as its functionalities to a new level. The resulting code, modular in structure and with embedded robustness and sanity checks, is naturally organised as a collaborative platform. Any sufficiently skilled user can exploit, modify and extend the functionalities of the current version. The code is available via a major open-source project development hosting service using the Bazaar version control system.

JHEP06(2011)128

The new code structure and new functionalities open the way to developments in three main directions: BSM, NLO, and merging with shower/hadronization codes. The direct link to the FeynRules model database will allow quick and robust implementation not only of new physics models but also of any type of 1-loop counterterms, essential ingredients to achieve NLO automatic computations in the SM and beyond. Work in all these directions is in progress.

Acknowledgements

It is a great pleasure for us to thank all the people who, directly or indirectly, help and support our efforts and services to the high-energy community and all our users for their continuous and patient feedback. In particular, for the help in the extensive testing of the new version, we thank Alexis Kalogeropoulos; for the validation of new physics models (and much more) we thank the FeynRules core authors (Neil Christensen, Claude Duhr, Benjamin Fuks) and associates (Priscila de Aquino, Celine Degrande); for our cluster management we are grateful to Vincent Boucher, Jerôme de Favereau, Pavel Demin, and Larry Nelson; for the great physics work and the fun, we in particular thank our colleagues and collaborators: Pierre Artoisenet, Simon de Visscher, Rikkert Frederix, Stefano Frixione, Nicolas Greiner, Kaoru Hagiwara, Junichi Kanzaki, Valentin Hirschi, Qiang Li, Kentarou Mawatari, Roberto Pittau, Tilman Plehn, Marco Zaro. This work is partially supported by the Belgian Federal Office for Scientific, Technical and Cultural Affairs through the 'Interuniversity Attraction Pole Program - Belgium Science Policy' P6/11-P and by the IISN "MadGraph" convention 4.4511.10.

A Installation and online web version

MadGraph 5 can be either used directly online on the web or locally. The code can be downloaded from the web page `https://launchpad.net/madgraph5`. In order to run MadGraph 5 locally, Python 2.6 or higher (but not 3.x) must be installed. The package does not require any compilation or configuration; after unpacking, simply launch the main script:

```
tar -xzpvf MadGraph5_v1.x.x.tar.gz
cd MadGraph5_v1_x_x
./bin/mg5
```

To learn how to use MadGraph 5, enter `tutorial` in the command line interface. The full list of presently available commands are described in appendix B.

MadGraph 5 can also be used on the web (after free registration). We currently have three public clusters:

- `http://madgraph.phys.ucl.ac.be/`
- `http://madgraph.hep.uiuc.edu/`

JHEP06(2011)128

- `http://madgraph.roma2.infn.it/`

At this stage, MadEvent (Fortran) output can be generated online, and downloaded as a standalone process directory. In the near future, also other output format will be available online. Based on a user request, we also grant access to run event generation directly on one of the clusters. In that case, the user can also choose to directly pass the events through Pythia 6 for hadronization, and use a fast detector simulation, either Delphes [76] or PGS [77].

B Command line user interface

The command line interface for MadGraph 5 is built on the Python module `cmd`. This module allows for a flexible treatment of user input and support of features such as tab completion, command history (accessed by the `up` key), help texts, and access to shell commands from inside the command line interface. Using the command line interface, the user can conveniently access the full functionality of MadGraph 5, including importing models, generating processes, drawing Feynman diagrams, generating output in all available output formats (at present, MadEvent, Pythia 8 and standalone process and model output in Fortran or C++), performing model checks, and launching event generation in previously created process directories.

The command line interface is straightforward to extend, and more functionality is continuously added based on user requests and code development.

The syntax for process generation in MadGraph 5 is very similar to that for MadGraph 4, with a few exceptions reflecting the extended functionality of MadGraph 5 — most notably, spaces are needed between particle names, since there is no longer any limit on particle name length, and the syntax for generating decay chains is modified to accommodate the greater flexibility in decay chain generation. Some syntax examples are given in table 7. However, the interface also has the ability to read process cards written for MadGraph 4.

The user can start the command line interface by running `bin/mg5` from the MadGraph 5 directory. If the name of a file containing MadGraph 5 commands is given as argument, then the commands in the file are performed. Such a file can be generated from the series of commands used in a session by the `history` command, and is automatically placed in the `Cards/` directory when a MadEvent directory is created. Process generation can also be done as in MadGraph 4, by running `bin/newprocess_mg5` in a copy of the Template directory with an appropriate `proc_card.dat` or `proc_card_mg5.dat` placed in the `Cards` directory.

We list here the presently available commands in the command interface (in alphabetical order).

- `add process`: Add and generate diagrams for a process, keeping previously generated processes.

JHEP06(2011)128

Syntax	Description
`p p > l+ l-`	Generate the (valid) processes $u\bar{u} \to e^+e^-$, $d\bar{d} \to \mu^+\mu^-$ etc.
`p p > j j j`	(No specification of orders.) Only allow diagram with maximal `QCD`/minimal `QED` order.
`p p > j j j QED=2`	Allow up to 2 `QED` vertices (and unlimited QCD vertices) in diagrams.
`p p > z > l+ l- j`	Include only diagrams with an s-channel Z.
`l+ l- > z | a > l+ l-`	Include only diagrams with an s-channel Z or an s-channel γ.
`b b~ > t t~ / g`	Exclude any diagrams with a g as internal propagator.
`p p > b w+ t~ $ t`	Exclude any diagrams with an s-channel t propagator.
`p p > t t~, t > b w+, \ (t~ > b~ w-, w- > l- vl~)`	Generate a decay chain with t, $\bar{t}$ and W^- required to be near-onshell. Note the use of parentheses to specify decays within a decay chain.

Table 7. Some examples of process generation syntax in the MadGraph 5 command line interface. The main differences w.r.t. MadGraph 4 are that spaces are needed between particle names, that by default minimal `QED` coupling order is assumed (if there are only `QED` and `QCD` couplings in the model), and furthermore the decay chain syntax which now allows full specification of all decay processes including coupling orders, required and excluded particles, etc.

- `check`: Run model consistency checks for specified processes. Available checks are: process permutation checks, gauge invariance check, and Lorentz invariance check (see section 4.3).
- `define`: Define a multiparticle label used for implicit summing over processes. Some commonly used multiparticle labels (`p`, `j`, `l+`, `l-`, `vl`, `vl~`) are automatically defined when a model is imported.
- `display`: Display particles, interactions, defined multiparticle labels, generated processes, generated diagrams, or the results of process checks.
- `generate`: Generate diagrams for a process, replacing any previously generated processes.
- `history`: List the history of previous commands to the screen or to a file. The resulting file can be used to repeat a sequence of commands using the `import` command.
- `import`: Import a model (either in the UFO format or MadGraph 4 format) or a process card (in MadGraph 5 or MadGraph 4 format).
- `launch`: Launch event generation or matrix element evaluation for created process directories.
- `load`: Load a process or model previously saved to file using the `save` command.

JHEP06(2011)128

- `output`: Output process files for matrix element integration. Presently available output formats are: MadEvent format (default), stand-alone Fortran format, Pythia 8 format, and stand-alone C++ format.

- `save`: Save a process or model to file, using the Python "pickle" format.

- `set`: Modify settings, including turning on/off subprocess grouping (see section 3.1).

- `tutorial`: Start a short tutorial showing how to use the most common commands.

Note that the `help` command gives the full list of available commands. Typing `help` *`command`* gives information about each command. The built-in commands `quit` and `exit` (or pressing `control-D`) quits MadGraph 5, and `shell` (or starting the line with a "`!`") allows to access any shell command.

Finally, the interactive interface has a tutorial mode which allows to quickly learn the most common commands used in the interface.

C Process generation examples

This appendix presents examples of the series of commands required to generate the code corresponding to the square matrix element of various process. Those command can be either copied-pasted directly in the interactive interface (accessed by running `./bin/mg5`) or written in a text file executed by MadGraph 5 (executed by `./bin/mg5` *`command_file`*).

C.1 Top-quark pair production

The first example shows how to evaluate a cross-section (and how to generate partonic events) for top-quark pair production in the Standard Model:

```
generate p p > t t~ QED=2 QCD=2
output MyOutputDir
launch
```

By default, MadGraph 5 imports the standard model. Therefore, no specific command is needed for that. As stated above, the process definition syntax is slightly different from the MadGraph 4 one. In MadGraph 5, spaces between particles names are mandatory. As in MadGraph 4, it is possible to specify coupling orders. If the coupling orders are not specified, then MadGraph 5 guesses which interactions to allow based on the following rules:

- If the orders defined in the model are `QED` and `QCD` only: The strong coupling is assumed to be dominant over the electroweak couplings, and the `QED` order is therefore set to its minimal possible value, while putting `QCD` to its maximal possible value. This provides the dominant contribution to the cross section, without the (negligible) sub-leading diagrams with additional QED couplings.

JHEP06(2011)128

- If additional orders are present besides `QED` and `QCD`, then we allow any coupling order for any coupling, (if given couplings are preferred, the user needs to supply maximum orders in the process definition).

The computation of the cross-section and the generation of partonic events is done with the command `launch`. Different options for cross section calculation and event generation or matrix element evaluation can be found either with the `help launch` command or in the file MyOutputDir/README.

C.2 Stop pair production

This example shows how to evaluate the value of the square matrix element for a given point in phase-space. This is often used for testing the code and/or to interface MadGraph with an external program.

```
import model mssm
generate p p > t1 t1~
add process p p > t2 t2~
output standalone
launch
```

First, the `mssm` model is imported. Then, the example shows how to create multiple subprocesses. The `generate` command clears all previously generated processes, while the `add process` keeps all previous processes and adds the new process to the set. If no output directory is given to the `output` command, the output will be placed in an automatically named directory `PROC_mssm_0`.

C.3 Slepton pair production

In this example, we show how to generate all slepton pair production matrix elements for use in Pythia 8 (See section 3.2). To generate the needed subprocesses, we could use the `add process` command. However, given that the number of sub-processes is quite large, this is not very convenient. A much more handy solution is to use a multi-particle label, similar to `p/j` for proton/jet:

```
import model mssm
define sl- = el- mul- ta1- er- mur- ta2-
define sl+ = el+ mul+ ta1+ er+ mur+ ta2+
generate p p > sl+ sl-
output pythia8
launch
```

In order to run event generation from the process, you will need to have Pythia 8 installed on your computer. If the path to the Pythia 8 main directory is not given in the `output` command, MadGraph 5 will look for it in the default location (./pythia8). The default location can be modified by editing the file `./input/mg5_configuration.txt`.

JHEP06(2011)128

C.4 W^+jj production

In this simple example we show how to create eps files containing the Feynman diagrams for a set of processes:

```
generate p p > W+ j j
display diagrams ./
```

The '.eps' files will be written in the output directory. Note that unless a coupling order is specified, MadGraph 5 will minimize the number of allowed `QED` orders, as explained in section C.1 above; here `QED=1`. If you want the full expansion in α_{em}, you will need to specify where to stop the expansion, i.e.,

```
generate p p > W+ j j QED=3
display diagrams ./
```

C.5 Graviton-jet production

In this example, we show how to evaluate the squared matrix element in C++ for a more exotic model. We will use the specific case of the graviton production with one additional jet in the Randall-Sundrum model [78, 79]. The list of command is the following:

```
import model RS
generate p p > y j
output standalone_cpp
launch
```

In the model implementation, the name for the graviton is `y`. In order to know the name of all particles present in the model, you can use the command `display particles` and in order to get more information about a specific particle you can enter `display particles y`. In this examples they are three different coupling order labels `QED` / `QCD` / `QTD`; the latter is linked to the graviton sector of the theory. Since they are three coupling orders, MadGraph 5 is not able to guess a suitable hierarchy between those order and set by default all orders to their maximal possible value.

C.6 Gluino decay

In this example we evaluate the partial decay width for gluino decay into $u\bar{u}\chi_1^0$ through a left-handed squark:

```
generate go > ul > u~ u n1
output madevent
launch
```

The particle(s) between the two > are requested to be present in all diagrams as s-channel propagators. Note that this condition does not imply that the particle is strictly on-shell. Also note that the result of the decay width calculation is given in GeV.

JHEP06(2011)128

C.7 Top-pair production with one leptonic decay

In MadGraph 5, it is possible to specify the decay of a (nearly) on-shell particle (see section 2.4). The syntax follows the logic of first stating the core process and then indicating the decay(s). If sub-decays are requested, they should be enclosed between parenthesis:

```
generate p p > t t~ QED=0, \
         (t > W+ b,   W+ > j j), \
         (t~ > w- b\,, W- > l- vl~)
output madevent
launch
```

The full process can be written on one line, or using the line continuation symbol \ to divide lines.

D The test suite

MadGraph 5 features a test suite that allows to test the installation as well as any further development of the code. During code development, this test suite is extremely important in order to avoid bugs, ensure the stability of the package and allow a robust multi-developer approach. In this respect, it is considered mandatory that the implementation of any new functionality is accompanied by related tests. General advice is that the test suite should be as important and developed as the code itself and should be split into three different levels:

Unit tests:
Each part of the code (class or function) is tested by a series of dedicated tests. The purpose is to fully check the behaviour of the routine / class, i.e., check the output in some specific case, check the behavior of the code in case of wrong input, etc. Currently MadGraph 5 includes more than 400 independent unittests.

Acceptance tests:
This part simulates the instructions entered by the user and checks that all modules are correctly interfaced. This corresponds to check that a series of examples are correctly running and provide the expected results.

Parallel tests:
These tests check that the output of MadGraph 5 are the same of those obtained by MadGraph 4. These checks are made for different models (SM/ MSSM/ HEFT) and using both UFO and version 4 models.

In order to efficiently run these tests, we have implement a script which automatically detects all the tests available. The tests can be filtered by name or file, in order to run only a subset of tests. A similar test module is now present by default in Python 2.7. However, since MadGraph 5 is by design also compatible with Python 2.6, a dedicated test code is included in the distribution. In order to run the different test suites, the user can type (respectively for unittest / acceptance tests / parallel tests):

JHEP06(2011)128

```
./tests/tests_manager.py
./tests/tests_manager.py -p A
./tests/tests_manager.py -p P
```


References

[1] A. Pukhov et al., *CompHEP: a package for evaluation of Feynman diagrams and integration over multi-particle phase space. User's manual for version 33*, hep-ph/9908288 [SPIRES].

[2] CompHEP collaboration, E. Boos et al., *CompHEP* 4.4*: automatic computations from lagrangians to events*, *Nucl. Instrum. Meth.* **A 534** (2004) 250 [hep-ph/0403113] [SPIRES].

[3] A. Pukhov, *Calchep* 2.3*: MSSM, structure functions, event generation,* 1 *and generation of matrix elements for other packages*, hep-ph/0412191 [SPIRES].

[4] T. Stelzer and W.F. Long, *Automatic generation of tree level helicity amplitudes*, *Comput. Phys. Commun.* **81** (1994) 357 [hep-ph/9401258] [SPIRES].

[5] F. Maltoni and T. Stelzer, *MadEvent: automatic event generation with MadGraph*, *JHEP* **02** (2003) 027 [hep-ph/0208156] [SPIRES].

[6] J. Alwall et al., *MadGraph/MadEvent v4: the new web generation*, *JHEP* **09** (2007) 028 [arXiv:0706.2334] [SPIRES].

[7] T. Gleisberg et al., *SHERPA* 1*.alpha, a proof-of-concept version*, *JHEP* **02** (2004) 056 [hep-ph/0311263] [SPIRES].

[8] F. Caravaglios and M. Moretti, *An algorithm to compute Born scattering amplitudes without Feynman graphs*, *Phys. Lett.* **B 358** (1995) 332 [hep-ph/9507237] [SPIRES].

[9] P. Draggiotis, R.H.P. Kleiss and C.G. Papadopoulos, *On the computation of multigluon amplitudes*, *Phys. Lett.* **B 439** (1998) 157 [hep-ph/9807207] [SPIRES].

[10] C. Duhr, S. Hoeche and F. Maltoni, *Color-dressed recursive relations for multi-parton amplitudes*, *JHEP* **08** (2006) 062 [hep-ph/0607057] [SPIRES].

[11] M. Moretti, T. Ohl and J. Reuter, *O'Mega: an optimizing matrix element generator*, hep-ph/0102195 [SPIRES].

[12] W. Kilian, T. Ohl and J. Reuter, *WHIZARD: simulating multi-particle processes at LHC and ILC*, arXiv:0708.4233 [SPIRES].

[13] W. Kilian, *WHIZARD* 1.0*: a generic Monte-Carlo integration and event generation package for multi-particle processes. Manual*, LC-TOOL-2001-039.

[14] M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau and A.D. Polosa, *ALPGEN, a generator for hard multiparton processes in hadronic collisions*, *JHEP* **07** (2003) 001 [hep-ph/0206293] [SPIRES].

[15] C.G. Papadopoulos and M. Worek, *HELAC: a Monte Carlo generator for multi-jet processes*, hep-ph/0606320 [SPIRES].

JHEP06(2011)128

[16] T. Gleisberg and S. Hoeche, *Comix, a new matrix element generator*, *JHEP* **12** (2008) 039 [arXiv:0808.3674] [SPIRES].

[17] N.D. Christensen and C. Duhr, *FeynRules — Feynman rules made easy*, *Comput. Phys. Commun.* **180** (2009) 1614 [arXiv:0806.4194] [SPIRES].

[18] N.D. Christensen et al., *A comprehensive approach to new physics simulations*, *Eur. Phys. J.* **C 71** (2011) 1541 [arXiv:0906.2474] [SPIRES].

[19] C. Duhr and B. Fuks, *A superspace module for the FeynRules package*, arXiv:1102.4191 [SPIRES].

[20] C. Degrande et al., *UFO — The Universal FeynRules Output*.

[21] T. Gleisberg and F. Krauss, *Automating dipole subtraction for QCD NLO calculations*, *Eur. Phys. J.* **C 53** (2008) 501 [arXiv:0709.2881] [SPIRES].

[22] M.H. Seymour and C. Tevlin, *TeVJet: a general framework for the calculation of jet observables in NLO QCD*, arXiv:0803.2231 [SPIRES].

[23] K. Hasegawa, S. Moch and P. Uwer, *Automating dipole subtraction*, *Nucl. Phys. Proc. Suppl.* **183** (2008) 268 [arXiv:0807.3701] [SPIRES].

[24] R. Frederix, T. Gehrmann and N. Greiner, *Automation of the dipole subtraction method in MadGraph/MadEvent*, *JHEP* **09** (2008) 122 [arXiv:0808.2128] [SPIRES].

[25] M. Czakon, C.G. Papadopoulos and M. Worek, *Polarizing the dipoles*, *JHEP* **08** (2009) 085 [arXiv:0905.0883] [SPIRES].

[26] R. Frederix, S. Frixione, F. Maltoni and T. Stelzer, *Automation of next-to-leading order computations in QCD: the FKS subtraction*, *JHEP* **10** (2009) 003 [arXiv:0908.4272] [SPIRES].

[27] G. Zanderighi, *Recent theoretical progress in perturbative QCD*, arXiv:0810.3524 [SPIRES].

[28] R.K. Ellis, K. Melnikov and G. Zanderighi, *Generalized unitarity at work: first NLO QCD results for hadronic $W+3$ jet production*, *JHEP* **04** (2009) 077 [arXiv:0901.4101] [SPIRES].

[29] C.F. Berger et al., *Precise predictions for $W+3$ jet production at hadron colliders*, *Phys. Rev. Lett.* **102** (2009) 222001 [arXiv:0902.2760] [SPIRES].

[30] A. van Hameren, C.G. Papadopoulos and R. Pittau, *Automated one-loop calculations: a proof of concept*, *JHEP* **09** (2009) 106 [arXiv:0903.4665] [SPIRES].

[31] C.F. Berger et al., *Next-to-Leading Order QCD predictions for $Z, \gamma^* + 3$-jet distributions at the Tevatron*, *Phys. Rev.* **D 82** (2010) 074002 [arXiv:1004.1659] [SPIRES].

[32] C.F. Berger et al., *Precise predictions for $W+4$ jet production at the Large Hadron Collider*, *Phys. Rev. Lett.* **106** (2011) 092001 [arXiv:1009.2338] [SPIRES].

[33] G. Ossola, C.G. Papadopoulos and R. Pittau, *CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes*, *JHEP* **03** (2008) 042 [arXiv:0711.3596] [SPIRES].

[34] V. Hirschi et al., *Automation of one-loop QCD corrections*, *JHEP* **05** (2011) 044 [arXiv:1103.0621] [SPIRES].

[35] T. Sjöstrand, S. Mrenna and P.Z. Skands, *PYTHIA 6.4 physics and manual*, *JHEP* **05** (2006) 026 [hep-ph/0603175] [SPIRES].

JHEP06(2011)128

[36] G. Corcella et al., *HERWIG 6.5: an event generator for Hadron Emission Reactions With Interfering Gluons (including supersymmetric processes)*, *JHEP* **01** (2001) 010 [hep-ph/0011363] [SPIRES].

[37] T. Gleisberg et al., *Event generation with SHERPA 1.1*, *JHEP* **02** (2009) 007 [arXiv:0811.4622] [SPIRES].

[38] S. Catani, F. Krauss, R. Kuhn and B.R. Webber, *QCD matrix elements + parton showers*, *JHEP* **11** (2001) 063 [hep-ph/0109231] [SPIRES].

[39] F. Krauss, *Matrix elements and parton showers in hadronic interactions*, *JHEP* **08** (2002) 015 [hep-ph/0205283] [SPIRES].

[40] S. Mrenna and P. Richardson, *Matching matrix elements and parton showers with HERWIG and PYTHIA*, *JHEP* **05** (2004) 040 [hep-ph/0312274] [SPIRES].

[41] M.L. Mangano, M. Moretti, F. Piccinini and M. Treccani, *Matching matrix elements and shower evolution for top-quark production in hadronic collisions*, *JHEP* **01** (2007) 013 [hep-ph/0611129] [SPIRES].

[42] L. Lönnblad, *Correcting the colour-dipole cascade model with fixed order matrix elements*, *JHEP* **05** (2002) 046 [hep-ph/0112284] [SPIRES].

[43] N. Lavesson and L. Lönnblad, *W + jets matrix elements and the dipole cascade*, *JHEP* **07** (2005) 054 [hep-ph/0503293] [SPIRES].

[44] S. Hoeche, F. Krauss, S. Schumann and F. Siegert, *QCD matrix elements and truncated showers*, *JHEP* **05** (2009) 053 [arXiv:0903.1219] [SPIRES].

[45] S. Hoeche et al., *Matching parton showers and matrix elements*, hep-ph/0602031 [SPIRES].

[46] J. Alwall et al., *Comparative study of various algorithms for the merging of parton showers and matrix elements in hadronic collisions*, *Eur. Phys. J.* **C 53** (2008) 473 [arXiv:0706.2569] [SPIRES].

[47] F. Krauss, A. Schalicke, S. Schumann and G. Soff, *Simulating W/Z + jets production at the Tevatron*, *Phys. Rev.* **D 70** (2004) 114009 [hep-ph/0409106] [SPIRES].

[48] C. Englert, T. Plehn, P. Schichtel and S. Schumann, *Jets plus missing energy with an autofocus*, *Phys. Rev.* **D 83** (2011) 095009 [arXiv:1102.4615] [SPIRES].

[49] J. Alwall, S. de Visscher and F. Maltoni, *QCD radiation in the production of heavy colored particles at the LHC*, *JHEP* **02** (2009) 017 [arXiv:0810.5350] [SPIRES].

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

[51] S. Frixione, P. Nason and B.R. Webber, *Matching NLO QCD and parton showers in heavy flavour production*, *JHEP* **08** (2003) 007 [hep-ph/0305252] [SPIRES].

[52] P. Nason, *A new method for combining NLO QCD with shower Monte Carlo algorithms*, *JHEP* **11** (2004) 040 [hep-ph/0409146] [SPIRES].

[53] S. Alioli, P. Nason, C. Oleari and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, *JHEP* **06** (2010) 043 [arXiv:1002.2581] [SPIRES].

[54] J. Alwall et al., *Aloha — Automatic helas routines for helicity amplitude calculations in any quantum field theory.*

JHEP06(2011)128

[55] T. Sjöstrand, S. Mrenna and P.Z. Skands, *A brief introduction to PYTHIA* 8.1, *Comput. Phys. Commun.* **178** (2008) 852 [arXiv:0710.3820] [SPIRES].

[56] H. Murayama, I. Watanabe and K. Hagiwara, *HELAS: HELicity amplitude subroutines for Feynman diagram evaluations*, KEK-91-11.

[57] G.C. Cho et al., *Weak boson fusion production of supersymmetric particles at the LHC*, *Phys. Rev.* **D 73** (2006) 054002 [hep-ph/0601063] [SPIRES].

[58] A. Denner, H. Eck, O. Hahn and J. Kublbeck, *Feynman rules for fermion number violating interactions*, *Nucl. Phys.* **B 387** (1992) 467 [SPIRES].

[59] M.L. Mangano and S.J. Parke, *Multi-parton amplitudes in gauge theories*, *Phys. Rept.* **200** (1991) 301 [hep-th/0509223] [SPIRES].

[60] V. Del Duca, L.J. Dixon and F. Maltoni, *New color decompositions for gauge amplitudes at tree and loop level*, *Nucl. Phys.* **B 571** (2000) 51 [hep-ph/9910563] [SPIRES].

[61] F. Maltoni, K. Paul, T. Stelzer and S. Willenbrock, *Color-flow decomposition of QCD amplitudes*, *Phys. Rev.* **D 67** (2003) 014026 [hep-ph/0209271] [SPIRES].

[62] F.A. Berends and W.T. Giele, *Recursive calculations for processes with n gluons*, *Nucl. Phys.* **B 306** (1988) 759 [SPIRES].

[63] R. Britto, F. Cachazo and B. Feng, *New recursion relations for tree amplitudes of gluons*, *Nucl. Phys.* **B 715** (2005) 499 [hep-th/0412308] [SPIRES].

[64] S. Frixione, *Colourful FKS subtraction*, arXiv:1106.0155 [SPIRES].

[65] T. Han, I. Lewis and T. McElmurry, *QCD corrections to scalar diquark production at hadron colliders*, *JHEP* **01** (2010) 123 [arXiv:0909.2666] [SPIRES].

[66] J. Alwall et al., *A standard format for Les Houches event files*, *Comput. Phys. Commun.* **176** (2007) 300 [hep-ph/0609017] [SPIRES].

[67] J. Alwall et al., *A Les Houches interface for BSM generators*, arXiv:0712.3311 [SPIRES].

[68] K. Hagiwara, J. Kanzaki, Q. Li and K. Mawatari, *HELAS and MadGraph/MadEvent with spin-2 particles*, *Eur. Phys. J.* **C 56** (2008) 435 [arXiv:0805.2554] [SPIRES].

[69] K. Hagiwara, K. Mawatari and Y. Takaesu, *HELAS and MadGraph with spin-3/2 particles*, *Eur. Phys. J.* **C 71** (2011) 1529 [arXiv:1010.4255] [SPIRES].

[70] P. Draggiotis, M.V. Garzelli, C.G. Papadopoulos and R. Pittau, *Feynman rules for the rational part of the QCD 1-loop amplitudes*, *JHEP* **04** (2009) 072 [arXiv:0903.0356] [SPIRES].

[71] N. D. Christensen and C. Speckner, *Automated validation of FeynRules models.*

[72] C. Zhang and S. Willenbrock, *Effective-field-theory approach to top-quark production and decay*, *Phys. Rev.* **D 83** (2011) 034006 [arXiv:1008.3869] [SPIRES].

[73] J.A. Aguilar-Saavedra, *Effective four-fermion operators in top physics: a roadmap*, *Nucl. Phys.* **B 843** (2011) 638 [arXiv:1008.3562] [SPIRES].

[74] C. Degrande, J.-M. Gerard, C. Grojean, F. Maltoni and G. Servant, *Non-resonant new physics in top pair production at hadron colliders*, *JHEP* **03** (2011) 125 [arXiv:1010.6304] [SPIRES].

JHEP06(2011)128

[75] C. Degrande, J.-M. Gerard, C. Grojean, F. Maltoni and G. Servant, *An effective approach to same sign top pair production at the LHC and the forward-backward asymmetry at the Tevatron*, arXiv:1104.1798 [SPIRES].

[76] S. Ovyn, X. Rouby and V. Lemaitre, *Delphes, a framework for fast simulation of a generic collider experiment*, arXiv:0903.2225 [SPIRES].

[77] J. Conway, *Pretty Good Simulator*, http://www.physics.ucdavis.edu/~conway/research/software/pgs/pgs.html

[78] L. Randall and R. Sundrum, *An alternative to compactification*, *Phys. Rev. Lett.* **83** (1999) 4690 [hep-th/9906064] [SPIRES].

[79] L. Randall and R. Sundrum, *A large mass hierarchy from a small extra dimension*, *Phys. Rev. Lett.* **83** (1999) 3370 [hep-ph/9905221] [SPIRES].

JHEP06(2011)128