Eur. Phys. J. C (2014) 74:3001
DOI 10.1140/epjc/s10052-014-3001-5

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

GoSam-2.0: a tool for automated one-loop calculations within the Standard Model and beyond

Gavin Cullen[1], **Hans van Deurzen**[2,a], **Nicolas Greiner**[2,b], **Gudrun Heinrich**[2,c], **Gionata Luisoni**[2,d], **Pierpaolo Mastrolia**[2,3,e], **Edoardo Mirabella**[2,f], **Giovanni Ossola**[4,5,g], **Tiziano Peraro**[2,h], **Johannes Schlenk**[2,i], **Johann Felix von Soden-Fraunhofen**[2,j], **Francesco Tramontano**[6,k]

[1] Deutsches Elektronen-Synchrotron DESY, Zeuthen, Germany
[2] Max-Planck-Institut für Physik, Munich, Germany
[3] Dipartimento di Fisica, Università di Padova, Padua, Italy
[4] New York City College of Technology, City University of New York, New York, USA
[5] The Graduate School and University Center, City University of New York, New York, USA
[6] Dipartimento di Scienze Fisiche, Università di Napoli and INFN, Sezione di Napoli, Naples, Italy

Received: 8 May 2014 / Accepted: 23 July 2014 / Published online: 12 August 2014

Abstract We present the version 2.0 of the program package GoSam for the automated calculation of one-loop amplitudes. GoSam is devised to compute one-loop QCD and/or electroweak corrections to multi-particle processes within and beyond the Standard Model. The new code contains improvements in the generation and in the reduction of the amplitudes, performs better in computing time and numerical accuracy, and has an extended range of applicability. The extended version of the "Binoth-Les-Houches-Accord" interface to Monte Carlo programs is also implemented. We give a detailed description of installation and usage of the code, and illustrate the new features in dedicated examples.

Contents

[a] e-mail: hdeurzen@mpp.mpg.de
[b] e-mail: greiner@mpp.mpg.de
[c] e-mail: gudrun@mpp.mpg.de
[d] e-mail: luisonig@mpp.mpg.de
[e] e-mail: Pierpaolo.Mastrolia@cern.ch
[f] e-mail: mirabell@mpp.mpg.de
[g] e-mail: gossola@citytech.cuny.edu
[h] e-mail: peraro@mpp.mpg.de
[i] e-mail: jschlenk@mpp.mpg.de
[j] e-mail: jfsoden@mpp.mpg.de
[k] e-mail: Francesco.Tramontano@cern.ch

1 Introduction

After the great achievement of discovering a new boson at the LHC [1,2], the primary goal is now to study its properties in detail, and to detect the slightest hints for possible extensions of the Standard Model. Certainly, precise theory predictions are indispensable to achieve this aim, which calls for calculations at next-to-leading order (NLO) accuracy and beyond.

NLO predictions nowadays should be considered as the standard for experimental data analysis. Ideally, matching NLO results to a parton shower and merging different jet multiplicities should be aimed for. However, this also requires fast and highly automated NLO tools to be available, to be compared to a vast amount of measurements, most of them dealing with multi-particle final states.

The development of automated NLO tools has seen tremendous progress in the past years, leading to public codes [3–7] which are able to produce multi-particle NLO predictions for user-defined processes, or to dedicated frameworks [8–15], which allowed to produce an impressive collection of NLO processes.

The integrand-reduction method [16–18] has changed our way of addressing the decomposition of amplitudes in terms of master integrals, whose coefficients can be determined by applying algebraic projections to polynomial functions.

The principle of an integrand-reduction method, which is valid at any order in perturbation theory [19–23], is the underlying multi-particle pole expansion for the integrand of any scattering amplitude, or, equivalently, a representation where the numerator of each Feynman integral is expressed as a combination of products of the corresponding denominators, with polynomial coefficients. These coefficients correspond to the residue of the integrand at the multiple-cut. Each residue is a multivariate polynomial in the *irreducible scalar products* formed by the loop momenta and either external momenta or polarization vectors constructed out of them.

GoSam is a code which was designed to maximally exploit both the integrand reduction for dimensionally regulated one-loop amplitudes [16,24], as implemented in Samurai [25], as well as improved tensor reduction methods as developed in [26,27]. The algebraic expression of the integrands are automatically generated by means of the Golem technology [26,28–30].

The polynomial structure of the multi-particle residues is a *qualitative* information that turns into a *quantitative* algorithm for decomposing arbitrary amplitudes in terms of master integrals at the integrand level. In fact, in the context of an integrand-reduction, any explicit integration procedure is replaced by a simpler operation like *polynomial fitting*, which in Samurai is implemented via Discrete Fourier Transform [31–33].

GoSam produces analytic expressions for the integrands. Because of this feature, it is suitable to be interfaced with a new library, called Ninja [34,35], implementing an ameliorated integrand-reduction method, where the decomposition in terms of master integrals is achieved by Laurent expansion through semi-analytic *polynomial divisions* [36]. With the new reduction algorithm, GoSam-2.0 can produce results for NLO virtual corrections that are more accurate and less time consuming than the ones provided by version 1.0.

In this paper we present the new version 2.0 of the program GoSam [6], which has been used already to produce a multitude of NLO predictions both within [35,37–47] and beyond [48,49] the Standard Model. The new version contains important improvements in speed, numerical robustness, range of applicability and user-friendliness. GoSam can be linked to different Monte Carlo programs via the Binoth-Les-Houches-Accord BLHA [50], where the extended version BLHA2 [51] is also implemented. The program can be downloaded from [52] http://gosam.hepforge.org. The structure of the paper is the following. In Sect. 2 we give a brief overview of the program structure. The new features of the program are presented in Sect. 3. Section 4 describes the installation and usage of GoSam, while in Sect. 5 we give examples illustrating some of the new features, before we conclude in Sect. 6. The appendices contain a commented example of an input card for convenience of the user, and some details about higher rank integrals.

2 Overview of the program

GoSam can be used either as a standalone code producing one-loop (and tree level) amplitudes, or it can be used as a *One Loop Provider* (OLP) in combination with a *Monte Carlo* (MC) program, where the interface is automated, based on the standards defined in [50,51]. The main workflow of GoSam is shown in Fig. 1 for the standalone version and in Fig. 2 for GoSam as an OLP within a Monte Carlo setup.

In the standalone version, the user will fill out a process *run card* which we call `process.in`, where the process is defined, together with some options. Then the code for the virtual amplitudes is generated by invoking `gosam.py process.in`.

After running the above command with an appropriate run card, all the files which are relevant for code generation will be created. The command `make source` will invoke `QGRAF` [53] and `FORM` [54,55] to generate the diagrams and algebaric expressions for the amplitudes, using also `spinney` [56] for the spinor algebra within `FORM` and `haggies` [57] for code generation. In version 2.0 of Go-

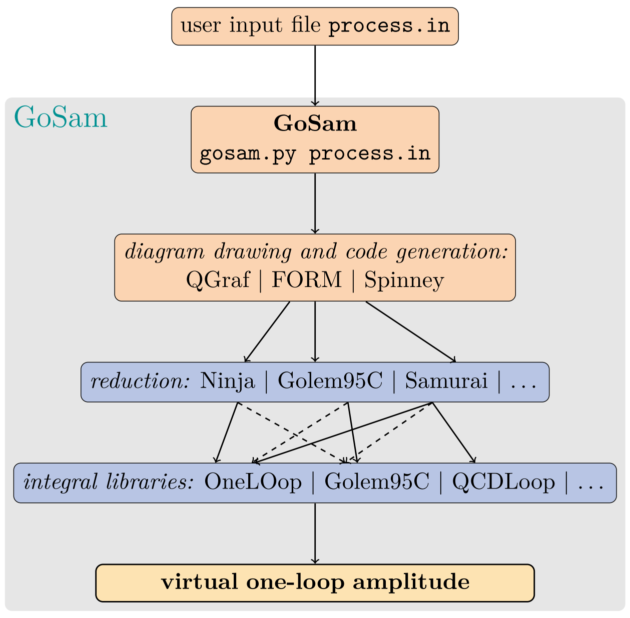
user input file process.in
GoSam
GoSam
gosam.py process.in
diagram drawing and code generation:
QGraf | FORM | Spinney
reduction: Ninja | Golem95C | Samurai | ...
integral libraries: OneLOop | Golem95C | QCDLoop | ...
virtual one-loop amplitude

Fig. 1 Basic workflow of GoSam

Sam, the production of optimized code however is largely relying on the new features of FORM version ≥4. The command make compile will finally compile the produced Fortran90 code.

In the OLP version, the information for the code generation is taken from the order file generated by the Monte Carlo program. Depending on the MC, the whole generation can be invoked automatically and steered by its setup. This is shown schematically in Fig. 2 and explained in more detail in Sect. 4.3.

The amplitudes are evaluated using D-dimensional reduction at integrand level [16,31,58], which is available through two different reduction procedures and libraries: Samurai [25,33] and Ninja [35,36]. Alternatively, tensorial reconstruction [27] is also available, based on the libraries Golem95C [30,59,60] and OneLOop [61].

It should be emphasized that all the reduction- and integral libraries used in GoSam-2.0 are included in the program package, and the installation script described in Sect. 4.1 will take care of compilation and linking, such that the user does not have to worry about installing them separately. Interfacing other tensor integral libraries, such as LoopTools [3,62], PJFRY [63,64] or Collier [65], should be straightforward, due to the modular structure of our setup.

More details about the reduction procedures implemented in GoSam will be given in Sect. 3.

3 New features

The version 2.0 of GoSam comes with several new features, which lead to an improvement in speed for both the generation and the evaluation of the amplitudes, more compact code, and more stable numerical evaluation. Further, the range of applicability of the code is extended, in particular to deal with effective theories and physics Beyond the Standard Model. We will describe some of the new features in more detail below.

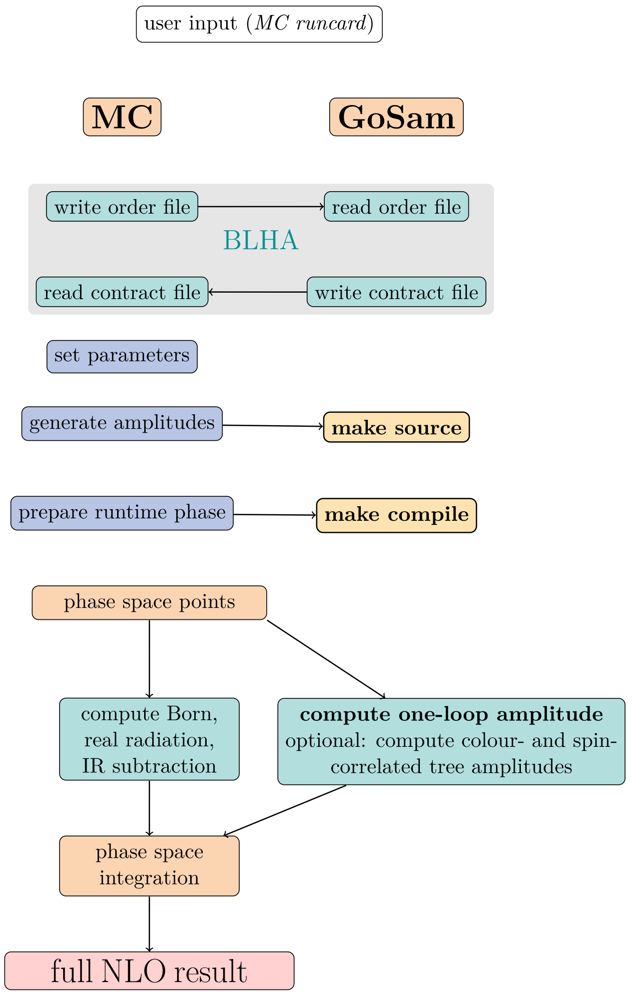

Fig. 2 Schematic setup for GoSam as an OLP in combination with a Monte Carlo program

3.1 Improvements in code generation

3.1.1 Producing optimised code with FORM *version 4*

While in version 1.0 of GoSam the Fortran code for the amplitudes was written using haggies [57], we now largely use the features provided by FORM version 4.x [55] to produce optimized code. This leads to more compact code and a speed-up in amplitude evaluation of about a factor of ten. The option to use haggies is still available by setting the extension noformopt.

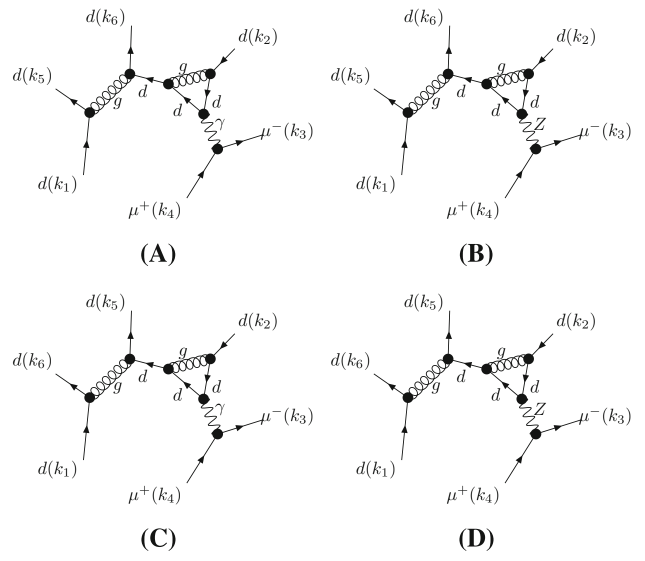

Fig. 3 Example of diagrams sharing a common tree part, which are summed when the `diagsum` option is set to `diagsum=true`

3.1.2 Grouping/summing of diagrams which share common subdiagrams

Already in the first release of GoSam, the diagrams were analyzed according to their kinematic matrix S_{ij} and grouped together before reduction. This lead to an important gain in efficiency, both with reduction based on integrand reduction methods, as well as with classical tensor reduction techniques. Details about the way diagrams are grouped can be found in [6]. This feature is still present when Samurai or Golem95C are used for computing the amplitudes.

In the new release an option called `diagsum` combines diagrams which differ only by a subdiagram into one "meta-diagram" to be processed as an entity. This allows one to further reduce the number of calls to the reduction program and therefore to increase the computational speed.

When the option `diagsum` is active, diagrams which differ only by a propagator external to the loop, as is the case e.g. for the $Z/\gamma^\star$ propagator in QCD corrections to the production of Z+jets, are summed together before being processed by `FORM`. Similarly, diagrams which differ only by an external tree part, but which share exactly the same set of loop propagators, are summed together prior the algebraic manipulation. An example is shown Fig. 3. Finally, diagrams which share the same set of propagators, but have different particles circulating in the loop, as shown in Fig. 4, are also summed into one "meta-diagram". The default setting for this option is `diagsum=true`.

3.1.3 Numerical polarisation vectors

The use of numerical polarisation vectors for massless gauge bosons (gluons, photons) is activated by default. This means that the various helicity configurations for the massless bosons will be evaluated numerically, based on a unique code containing generic polarisation vectors, rather than producing separate code for each helicity configuration. To switch off this default setting, for example if the user would like to optimize the choice of reference vectors for each helicity configuration, the option `polvec=explicit` should be given in the input file `process.in`.

3.2 Improvements in the reduction

The algebraic generation of the integrands in GoSam is tailored to the maximal exploitation of the D-dimensional integrand reduction algorithm.

In the previous version, GoSam-1.0, Samurai has been the default library for the amplitude decomposition in terms of master integrals. Within the original integrand reduction algorithm, implemented in Samurai, the determination of the unknown coefficients multiplying the master integrals requires: (i) to *sample* the numerator on a finite subset of the on-shell solutions; (ii) to *subtract* from the integrand the non-vanishing contributions coming from higher-point residues; and (iii) to *solve* the resulting linear system of equations. Gauss substitutions and the integrand subtractions enforce a *triangular* system solving strategy, which proceeds top-down, from the pentuple-cut to the single-cut. In this fashion, because of the integrand subtractions, the integrand which has to be evaluated numerically gets updated at any level, cut-by-cut, by the subtraction of the polynomial residues determined at the previous step. The sampling and the determination of the coefficients in Samurai proceeds with a projection technique based on the Discrete Fourier Transform [31,33].

In the new version GoSam-2.0, the amplitude decomposition is obtained by a new integrand-reduction method [36], implemented in the C++ code Ninja [34,35], which is the default reduction library.

In Ref. [36] an improved version of the integrand reduction method for one-loop amplitudes was presented. This method allows, whenever the analytic dependence of the integrand on the loop momentum is known, to extract the unknown coefficients of the residues by performing a Laurent expansion of the integrand with respect to one of the free loop components which are not constrained by the corresponding on-shell conditions.

Within the Laurent expansion approach, the system of equations for the coefficients becomes *diagonal*. In fact, in the asymptotic limit, both the integrand and the higher-point subtractions exhibit the same polynomial behaviour as the residue. Therefore one can identify the unknown coefficients with the ones of the expansion of the integrand, corrected by the contributions coming from higher-point residues. In other words, the subtractions of higher-point contributions are replaced by corrections at the coefficient level. Because

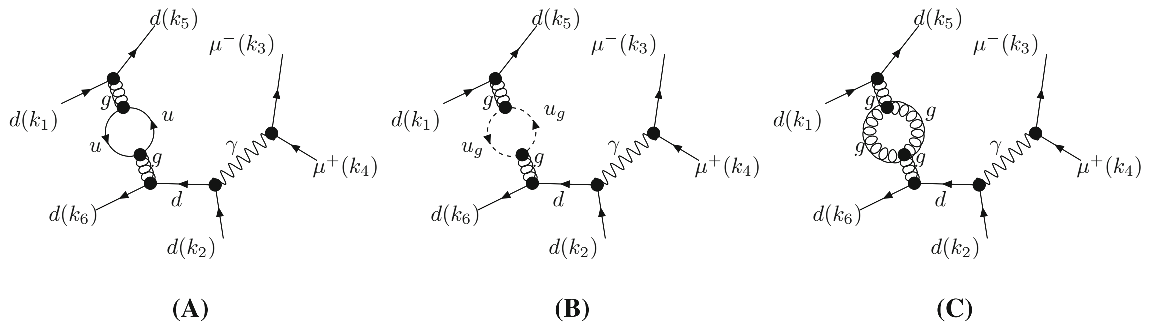

Fig. 4 Example of diagrams sharing a common loop propagator, but with different particle content in the loop, which are summed when the `diagsum` option is set to `diagsum=true`

of the universal structure of the residues, the parametric form of these corrections can be computed once and for all, in terms of a subset of the higher-point coefficients.

This novel D-dimensional unitarity-based algorithm is lighter than the original integrand reduction method, because less coefficients need to be determined, and turns out to be faster and numerically more accurate.

The integrand reduction via Laurent expansion has been implemented in the library NINJA, where the Laurent expansions of the integrands are performed by a *polynomial division* between some parametric expansions of the numerator and the uncut denominators. The expansions of the numerator, required by NINJA as input, are efficiently generated by GOSAM using `FORM`, after collecting the terms that do not depend on the loop momentum into global abbreviations.

NINJA and the new version of SAMURAI, as well as GOLEM95C, all distributed with the GOSAM-2.0 package, can deal with processes where the masses of the internal particles are complex, and where the rank r of the numerator of the integrands can exceed the number N of denominators by one unit, *i.e.* $r \leq N+1$, as it may happen e.g. in effective theories (see also Sect. 3.5.1).

3.2.1 The extension `derive`

The `derive` feature generates code to access the tensor coefficients of each diagram or group of diagrams individually. While it has been among the possible keywords for the `extensions` options in GOSAM-1.0 already, it now has been promoted to be used by default in the context of tensorial reconstruction [27]. It improves both the speed and the precision of tensorial reconstruction and makes connection to other reduction methods.

The idea behind it is to compute the numerator $\mathcal{N}(q)$ from a Taylor series

$$\mathcal{N}(q) = \mathcal{N}(0) + q^{\mu}\frac{\partial}{\partial q_{\mu}}\mathcal{N}(q)|_{q=0} + \frac{1}{2!}q^{\mu}q^{\nu}\frac{\partial}{\partial q_{\mu}}\frac{\partial}{\partial q_{\nu}}\mathcal{N}(q)|_{q=0} + \cdots \quad (1)$$

In this form one can read off a one-to-one correspondence between derivatives at $q = 0$ and the coefficients of the tensor integrals.

At a technical level, the files `helicity*/d*h*l1d.f90` contain the routines `derivative(`μ^2`, [`i_1`], [`i_2`],...)` and `reconstruct_d*(coeffs)`, where the latter is only generated in conjunction with the extension `golem95`, and `coeffs` is a type which comprises all coefficients of a diagram of a certain rank. The number of optional indices i_1, i_2, ...determine which derivative should be returned. The subroutine `reconstruct_d*` also takes into account the proper symmetrisation.

3.3 Electroweak scheme choice

Regularisation and renormalisation within the Standard Model can be performed using various schemes, which also may differ in the set of electroweak parameters considered as input parameters, while other electroweak parameters are derived ones. Within GOSAM-2.0, different schemes can be chosen in several different ways by setting appropriately the flag `model.options`, depending on whether the scheme might be changed after the generation of the code or not.

By default, when the flag is not set in the input card, GOSAM generates a code which uses $\mathrm{m_W}$, $\mathrm{m_Z}$ and α as input parameters, allowing however to change this in the generated code, by setting the variable `ewchoice` in the configuration file `common/config.f90` to the desired value. The user can choose among 8 different possibilities, which are listed in Table 1. When the electric charge e is set algebraically to one, the symbol for e will not be present in the generated amplitudes. This can be useful e.g. if the Monte Carlo generator used in combination with GOSAM will multiply the full result by the coupling constants at a later stage of the calculation. In scheme number 8, m_W is derived from a formula where both e and G_F enter. Thus, using $e = 1$ in combination with the standard value for G_F would lead to the wrong gauge boson masses. Therefore the scheme 8 cannot be used if e is set algebraically to one.

Table 1 Possible choices to select the electroweak scheme. To simplify the notation we write the sine of the weak mixing angle as sw. The lists of derived parameters contain only the parameters which are computed from the input parameters and used in the expressions for the amplitudes

ewchoice	Input parameters	Derived parameters
1	G_F, m_W, m_Z	e, sw
2	α, m_W, m_Z	e, sw
3	α, sw, m_Z	e, m_W
4	α, sw, G_F	e, m_W
5	α, G_F, m_Z	e, m_W, sw
6	e, m_W, m_Z	sw
7	e, sw, m_Z	m_W
8	e, sw, G_F	m_W, m_Z

The flag `model.options` in the input card allows one also to directly set the values of the different parameters appearing in the model. If the values of exactly three electroweak parameters are specified, GoSam automatically takes them as input parameters. In that case, in order to be able to switch among different schemes after code generation, the variable `ewchoose` also must be added to the `model.options` flag.

3.4 Stability tests and rescue system

Within the context of numerical and semi-numerical techniques, we should be able to assess in real time, for each phase space point, the level of precision of the corresponding one-loop matrix element. Whenever a phase space point is found in which the quality of the result falls below a certain threshold, either the point is discarded or the evaluation of the amplitude is repeated by means of a safer, albeit less efficient procedure. This procedure is traditionally called “rescue system”.

Apart from improvements in the stability of the reduction itself, which are provided by the new versions of Samurai and Golem95C, and in particular by the new reduction algorithm Ninja, the new version of GoSam also has a more refined rescue system as compared to version 1.0.

Looking at the literature, we observe that various techniques for detecting points with low precision have been implemented within the different automated tools for the evaluation of one-loop virtual corrections.

A first commonly used approach relies on the comparison between the numerical values of the infrared pole coefficients computed by the automated tool with their known analytic results dictated by the universal behaviour of the infrared singularities [66]. We refer to this as the *pole test*.

The main advantages of this method are its broad applicability to all amplitudes and the fact that it requires a negligible additional computation time. However, since not all integrals which appear in the reconstruction of the amplitude give a contribution to the double and single poles, this method often provides an overestimate of the precision, which might result in keeping phase space points whose finite part is less precise than what is predicted by the poles.

Different techniques have been proposed that target directly the precision of the finite part. Using the symmetry properties of scattering amplitudes under scaling of all physical scales, or alternatively the invariance under rotation of the momenta, we can build pairs of points that should provide identical results, both for the finite parts and for the poles, and use the difference between them as an estimator of the precision.

The *scaling test* [67], is based on the properties of scaling of scattering amplitudes when all physical scales (momenta, renormalisation scale, masses) are rescaled by a common multiplicative factor x. As shown in [67], this method provides a very good correlation between the estimated precision and the actual precision of the finite parts.

The *rotation test* [35] exploits the invariance of the scattering amplitudes under an azimuthal rotation about the beam axis, namely the direction of the initial colliding particles. Whenever the initial particles are not directed along the beam axis, one can perform a rotation of all particles by an arbitrary angle in the space of momenta. A validation of this technique, and the corresponding correlation plots, has been presented in [35].

While the *scaling* and the *rotation test* provide a more reliable estimate of the precision of the finite parts that enter in the phase space integration, their downside is that they require two evaluations of the same matrix element, therefore leading to a doubling in the computational time.

Additional methods have been proposed, within the context of integrand-reduction approaches, which target the relations between the coefficients before integration, namely the reconstructed algebraic expressions for the numerator function before integration, known as $\mathcal{N} = \mathcal{N}$ tests [18,25]. This kind of tests can be applied to the full amplitude (global $\mathcal{N} = \mathcal{N}$ test) or individually within each residue of individual cuts (local $\mathcal{N} = \mathcal{N}$ test). The drawback of this technique comes from the fact that the test is applied at the level of individual diagrams, rather than on the final result summed over all diagrams, making the construction of a rescue system quite cumbersome.

For the precision analysis contained in GoSam-2.0, and to set the trigger for the rescue system, we decided to employ a hybrid method, that takes advantage of the computational speed of the *pole test*, combined with the higher reliability of the *rotation test*. This hybrid method requires setting three different thresholds. After computing the matrix elements, GoSam-2.0 checks the precision δ_{pole} of the single pole with the *pole test*. Comparing the single pole $\mathcal{S}_{IR}$ that can be obtained from the general structure of infrared singularities and the one provided by GoSam-2.0, which we label $\mathcal{S}$, we define

$$\delta_{pole} = \left| \frac{\mathcal{S}_{IR} - \mathcal{S}}{\mathcal{S}_{IR}} \right| . \qquad (2)$$

The corresponding estimate of the number of correct digits in the result is provided by $P_{pole} = -\log_{10}(\delta_{pole})$. This step does not require any increase in computational time. The value of P_{pole} is then compared with two thresholds P_{high} and P_{low}.

If $P_{pole} > P_{high}$ the point is automatically accepted. Given the high quality of the computed pole, the finite part is very unlikely to be so poor that the point should be discarded.

If $P_{pole} < P_{low}$ the point is automatically discarded, or sent to the rescue system. If already the pole has a low precision, we can expect the finite part to be of the same level or worse.

In the intermediate region where $P_{high} > P_{pole} > P_{low}$, it is more difficult to determine the quality of the result solely based on the pole coefficients. Only in this case the point is recalculated using the *rotation test*, which requires additional computational time.

If we call the finite part of the amplitudes evaluated before and after the rotation $\mathcal{A}^{\text{fin}}$ and $\mathcal{A}^{\text{fin}}_{rot}$ respectively, we can define the error δ_{rot} estimated with the rotation as

$$\delta_{rot} = 2 \left| \frac{A^{\text{fin}}_{rot} - A^{\text{fin}}}{A^{\text{fin}}_{rot} + A^{\text{fin}}} \right| . \qquad (3)$$

and the corresponding estimate on the number of correct digits as $P_{rot} = -\log_{10}(\delta_{rot})$. P_{rot} provides a reliable estimate of the precision of the finite part [35], and can be compared with a threshold P_{set} to decide whether the point should be accepted or discarded.

The values of the three thresholds P_{high}, P_{low} and P_{set} can be chosen by the user, to adjust the selection mechanism to the fluctuations in precision which occur between different processes. In the input card, P_{high}, P_{low} and P_{set} correspond to `PSP_chk_th1`, `PSP_chk_th2` and `PSP_chk_th3`, respectively, see Appendix A. It is worth to notice that the *rotation test* can be bypassed simply by setting the initial thresholds $P_{high} = P_{low}$. In this case the selection is performed solely on the basis of the *pole test*.

3.5 New range of applicability

3.5.1 *Higher rank integrals*

The libraries Ninja, Golem95C and Samurai all support integrals with tensor ranks r exceeding the number of propagators N. Such integrals occur for example in effective theories (a prominent example is the effective coupling of gluons to the Higgs boson), or in calculations involving spin-two particles beyond the leading order. These extensions are described in detail in Refs. [33,36,60] and are contained in the distribution of GoSam-2.0. The additional integrals will be called automatically by GoSam if they occur in an amplitude, such that the user can calculate amplitudes involving higher rank integrals without additional effort. Ninja and Samurai provide higher rank integrals for rank $r = N + 1$, version 1.3 of Golem95C [60] provides higher rank integrals and the tensorial reconstruction routines up to $r = N + 1$ for $N \leq 6$, as well as form factors up to rank ten for $N \leq 4$. More details about the higher rank integrals are given in Appendix B.

3.5.2 *Production of colour-/spin correlated trees*

GoSam can also generate tree level amplitudes in a spin- and colour-correlated form. Colour correlated matrix elements are defined as

$$C_{ij} = \langle \mathcal{M} | \mathbf{T}_i \mathbf{T}_j | \mathcal{M} \rangle, \qquad (4)$$

and we define spin-correlated matrix elements as

$$S_{ij} = \langle \mathcal{M}, - | \mathbf{T}_i \mathbf{T}_j | \mathcal{M}, + \rangle. \qquad (5)$$

The spin-correlated matrix element (as well as the colour correlated matrix element) contains implicitly the sum over all non-specified helicities, while only the helicities with the indices i and j are fixed, i.e.

$$\begin{aligned} &\langle \mathcal{M}_{i,-} | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M}_{i,+} \rangle \\ &= \sum_{\lambda_1,\ldots,\lambda_{i-1},\lambda_{i+1},\ldots,\lambda_n} \langle \mathcal{M}_{\lambda_1,\ldots,\lambda_{i-1},-,\lambda_{i+1},\ldots,\lambda_n} \\ &\quad \times | \mathbf{T}_i \cdot \mathbf{T}_j | \mathcal{M}_{\lambda_1,\ldots,\lambda_{i-1},+,\lambda_{i+1},\ldots,\lambda_n} \rangle. \end{aligned} \qquad (6)$$

These matrix elements are particularly useful in combination with Monte Carlo programs which use these trees to build the dipole subtraction terms for the infrared divergent real radiation part. With these modified tree level matrix elements GoSam is able to generate all necessary building blocks for a complete NLO calculation.

Such a setup has been used successfully in combination with the framework of Herwig++/Matchbox [68–70].

3.5.3 *Support of complex masses*

The integral libraries contained in the GoSam package as well as the GoSam code itself fully support complex masses. The latter are needed for the treatment of unstable fermions and gauge bosons via the introduction of the corresponding decay width. A fully consistent treatment of complex W- and Z-boson masses requires the use of the complex mass scheme [71]. According to this scheme the boson masses become

$$m_V^2 \to \mu_V^2 = m_V^2 - i m_V \Gamma_V, \quad V = W, Z. \tag{7}$$

Gauge invariance requires that the definition of the weak mixing angle has to be modified accordingly:

$$\cos^2\theta_W = \frac{\mu_W^2}{\mu_Z^2}. \tag{8}$$

To make use of the complex mass scheme, we introduce two new model files, `sm_complex` and `smdiag_complex`, which contain the Standard Model with complex mass scheme, the first with a full CKM matrix, the latter with a diagonal unit matrix for the CKM matrix. An example dealing with a complex top quark mass is given in Sect. 5.

4 Installation and usage

4.1 Installation

The user can download the code either as a tar-ball or from a subversion repository at http://gosam.hepforge.org.

The installation of GoSam-2.0 is very simple when using the installation script. The latter can be downloaded by

```
wget http://gosam.hepforge.org/gosam-installer/gosam_installer.py
```

By default GoSam will be installed into a subfolder `./local` of the directory where the installation script is saved. A different path can be specified using the

```
--prefix=PATH_where_to_install
```

option. To run the script the user should execute the following commands

```
chmod +x gosam_installer.py
./gosam_installer.py [--prefix=...]
```

or

```
python gosam_installer.py
[--prefix=...]
```

Upon installation, the installer will ask some questions, which are described in more detail in the manual [52], which also can be downloaded from the webpage given above.

To use the default installation all the questions can be "answered" by pressing the `ENTER` key.

In particular, the installer will check if `QGRAF` [53] and `FORM` [54,55] already exist on the system. If they are not found, one can either press `ENTER` to have them installed by the script, or provide a path to the binary (tab-completion can be used). If they are found, their version is checked, and if needed the installation of a version which has been tested to run with GoSam is suggested.

As soon as all questions are answered, the main installation process will start. The components will be downloaded, built and installed. The whole procedure can take about 10–30 minutes.

At the end, a script `gosam_setup_env.sh` will be created in the `bin/` subdirectory of the install location, which will set (temporarily) all environment variables as soon as the script is sourced into a shell (with the command `source [path]/gosam_setup_env.sh`). The installer also gives a recommendation how these environment variables can be set permanently. The script can be used in all `tcsh`- and `bash`-compatible shells.

All files which have been installed are tracked in the `installer-log.ini` file. It is important to keep this file and the install script. They are needed to update and uninstall GoSam. For the default installation, internet access is required.

The program GoSam is designed to run in any modern Unix-like environment (Linux, Mac). The system requirements are `Python` ($\geq$ 2.6), a `Fortran` compiler (gfortran or ifort), a C/C++ compiler (gcc/icc), and (GNU) `make`. Compatibility with gcc versions 4.2.–4.9 as well as `clang` has been tested. By default, GoSam uses the gfortran/gcc compilers from the GNU Compiler Suite. To use an Intel compiler (ifort/icc), the `--intel` option can be used. Specific paths to the compilers can be provided using the `--fc`, `--cc`, `--cxx` options.

All further options can be listed by invoking the installation script with the flag `help`:

```
gosam_installer.py --help.
```

4.2 Using GoSam

We first start describing the use of GoSam in the standalone version. For the use in combination with a Monte Carlo program, based on the BLHA interface, we refer to Sect. 4.3.

In order to generate the matrix element for a given process the user should create a process specific setup file, `process.in`, which we call *process card*. An example process card for the process $e^+e^- \to t\bar{t}$ is given in Appendix A, where we explain each entry in detail.

It is recommended to generate and modify a template file for the process card. This can be done by invoking the shell command

```
gosam.py --template process.in
```

which generates the file `process.in` with some documentation for all defined options. The options are filled with the default values, and some paths are set by the installation script. User-defined options changing the default values can also be set in a global configuration file. The script will search in the GoSam-2.0 directory, in the user's home directory and in the current working directory for a file named '`.gosam`' or '`gosam.in`'.

In order to generate the Fortran code for the process specified in the input card one needs to invoke

```
gosam.py process.in
```

Structure of the generated code

The generated process directory will have the following subdirectory structure:

- `codegen`: This directory contains files which are only relevant for code generation. These files will therefore not be included in a tar-ball created with `make dist`.
- `common`: Contains `Fortran` files which are common to all helicity amplitudes and to the constructed matrix element code. The file `config.f90` contains some global settings, the file `model.f90` contains the definitions and settings for the model parameters. This directory is always compiled first.
- `doc`: Contains files which are necessary for creating `doc/process.pdf`, which displays all Feynman diagrams of the Born level and one-loop amplitude, together with colour and helicity info.
- `helicity*`: These directories contain all files for a specific helicity amplitude. The labeling of the helicities can be found in `doc/process.pdf`. Before invoking `make source`, this directory only contains the makefiles. After the full code generation, for each diagram three classes of files are created. The basic algebraic expressions for the individual one-loop diagrams are contained in the files `d*h*l1.txt` in an optimized format. The files `d*h*l1.prc` contain the expressions of the numerators as polynomials in the loop momentum. The corresponding `Fortran` files are `d*h*l1.f90` and `abbrevd*h*.f90`, where the latter contains the abbreviations.
  Files generated with the `derive` option (see Sect. 3.2.1) are named `d*h*l1d.*`, while the input files for NINJA (see Sect. 3.2) are named `d*h*l1*.*`. For more details we refer to the manual [52].
- `matrix`: Contains the code to combine the helicity amplitudes into a matrix element. Here one also finds the test program `test.f90`. The files in this directory are always compiled last.
- Further, there are some files in the main process directory, for example the Born/loop diagram files generated by `QGRAF`, called `diagrams-[0/1].hh`, or the model file `model.hh`.

Conventions

In the case of QCD corrections, the tree-level matrix element squared can be written as

$$|\mathcal{M}|^2_{\text{tree}} = \mathcal{A}_0^\dagger \mathcal{A}_0 = (g_s)^{2b} \cdot a_0. \tag{9}$$

The fully renormalised matrix element at one-loop level, i.e. the interference term between tree-level and one-loop amplitudes, can be written as

$$\begin{aligned}|\mathcal{M}|^2_{\text{1-loop}} &= \mathcal{A}_1^\dagger \mathcal{A}_0 + \mathcal{A}_0^\dagger \mathcal{A}_1 = 2 \cdot \Re(\mathcal{A}_0^\dagger \mathcal{A}_1)\\ &= |\mathcal{M}|^2_{\text{bare}} + |\mathcal{M}|^2_{\text{ct}, \delta m_Q} + |\mathcal{M}|^2_{\text{ct}, \alpha_s} + |\mathcal{M}|^2_{\text{wf, g}} + |\mathcal{M}|^2_{\text{wf, Q}}\\ &= \frac{\alpha_s(\mu)}{2\pi} \frac{(4\pi)^\varepsilon}{\Gamma(1-\varepsilon)} \cdot (g_s)^{2b} \cdot \left[c_0 + \frac{c_{-1}}{\varepsilon} + \frac{c_{-2}}{\varepsilon^2} + \mathcal{O}(\varepsilon)\right].\end{aligned} \tag{10}$$

In the default case the flag `nlo_prefactors` has the value zero, which means that a call to the subroutine `samplitude` returns an array consisting of the four numbers $(a_0, c_0, c_{-1}, c_{-2})$, in this order, with the prefactors extracted as given above. In the case of electroweak corrections `nlo_prefactors=0` has the same meaning, except that α_s is replaced by α. If the flag `nlo_prefactors` has the value one, a factor of $1/8\pi^2$ instead of $\alpha_s/2\pi$ (respectively $\alpha/2\pi$ in the EW case) has been extracted from the numerical result, while for `nlo_prefactors=2` all the prefactors are included in the numerical result.

The average over initial state colours and helicities is included in the default setup. In cases where the process is loop induced, i.e. the tree level amplitude is absent, the program returns the values for $\mathcal{A}_1^\dagger \mathcal{A}_1$, where a factor of

$$\left(\frac{\alpha_s(\mu)}{2\pi} \frac{(4\pi)^\varepsilon}{\Gamma(1-\varepsilon)}\right)^2$$

has been pulled out.

After UV renormalisation has been performed, only IR-singularities remain in the virtual matrix element. The coefficients of the latter can be checked using the routine `ir_subtractions`. This routine constructs the pole parts of the dipole subtraction terms and returns a vector of length two, containing the coefficients of the single and the double pole, which should be equal to (c_{-1}, c_{-2}).

4.3 Interfacing to Monte Carlo programs

The interface of GOSAM with a Monte Carlo event generator program is based on the Binoth-Les Houches Accord (BLHA) standards. GOSAM-2.0 supports both BLHA1 [50] and BLHA2 [51]. Certainly, a dedicated interface without using the BLHA is also possible, and such an interface with MADGRAPH/MADDIPOLE/MADEVENT [72–75] has been built and applied successfully in various phenomenological applications [40,43,45,48,49].

If GoSam is used as a One Loop Provider (OLP), the Monte Carlo program is steering the different stages of the calculation, in particular the phase space integration and the event generation, as illustrated in Fig. 2. Therefore, the user frontend will depend on the user interface of the Monte Carlo program. The latter will call GoSam at runtime to provide the corresponding value of the one-loop amplitude at the given phase space points.

A number of phenomenological results produced by combining GoSam with various Monte Carlo programs can be found in the literature, e.g in combination with Sherpa [39, 42–44,47], PowHeg [41], Herwig++/Matchbox [68].

Examples how to run GoSam with Sherpa can also be found on the Sherpa manual webpage [76] and on the GoSam process packages webpage [77]. For the interface with PowHeg, a detailed description can be found in the appendix of Ref. [41]. The interface with Herwig++/Matchbox is described in [68].

4.4 Using external model files

The GoSam-2.0 package comes with the built-in model files `sm`, `smdiag`, `smehc`, `sm_complex`, `smdiag_complex`, where the latter two are needed in the case of complex masses and couplings, see Sect. 3.5.3. The model files `smehc` contain the effective Higgs-gluon couplings.

Other models can be imported easily in the `UFO` (Universal FeynRules Output) [78] format. The model import in the `UFO` format can be used in the standalone as well as the OLP mode of GoSam, where both the BLHA1 and BLHA2 standards are supported for the syntax of the model import.

A model description in the `UFO` format consists of a `python` package which the user can either generate using FeynRules [79,80] or write himself and store in any directory. In order to import the model into GoSam one needs to set the `model` variable in the *process card* (line 5 in the example process card of Appendix A, specifying the keyword `FeynRules` in front of the path pointing to the `python` files defining the model. For example, if we assume that the model description is in the directory `$HOME/models /MSSM_UFO`, the *process card* should contain the line `model= FeynRules,$HOME/models/MSSM_UFO`.

The import of model files generated by `LanHEP` [81] is also supported. More details about the import from `LanHEP` are given in the GoSam-2.0 manual [52].

It should be pointed out that GoSam-2.0 provides automatic renormalisation only for QCD corrections. If external model files are used, as well as in the case of electroweak corrections, including the correct renormalisation is at the responsibility of the user.

The `examples` directory of the GoSam-2.0 package contains several examples for the import of model files, both in `UFO` and in `LanHEP` format. The subdirectory `examples/model` contains model files for the MSSM (as well as for the SM) in both `UFO` and `LanHEP` format. A concrete BSM example is discussed in Sect. 5.3.

5 Examples

5.1 $gg \to H{+}1$ jet in the heavy top mass limit

Recently GoSam was used to compute the virtual corrections for the production of a Higgs boson in association with 2 and 3 jets [39,43] in the infinite top-mass limit. As an example for this type of processes, where a special model file is needed, containing the Feynman rules for the effective vertices, which furthermore give rise to higher rank loop integrals, we consider here the process $gg \to H\,g$.

An example *process card* for the generation of this process, and a test routine comparing a phase space point with results from analytical amplitude representations is provided among the examples of the GoSam-2.0 distribution. In the following we will refer to that example to describe some feature of this process.

In order to compute amplitudes using the effective gluon-gluon-Higgs vertices, the model `smehc` has to be used. This model contains also the effective vertex for the Higgs boson decaying to two photons. When setting the powers of the strong coupling using the `order` flag, one has to remember that the effective vertex counts as two powers of the strong coupling. To compute the virtual corrections for $H + 1$ jet we therefore have to set `order=QCD, 3, 5`.

The inclusion of the effective gluon-gluon-Higgs coupling at NLO also requires corrections of the Wilson coefficient. At NLO the Wilson coefficient is given by [82]

$$C_1 = -\frac{\alpha_s}{3\pi}\left(1 + \frac{\alpha_s}{\pi}\frac{11}{4}\right), \tag{11}$$

where the effective Lagrangian is given by

$$\mathcal{L}_{\text{eff}} = -\frac{C_1}{4v} H G^{a,\mu\nu} G^{a}_{\mu\nu}. \tag{12}$$

The `smehc` model file also contains the effective vertex for the Higgs decay into a pair of photons via top- and W-loops. For the vertex factor we use the formula given by FeynRules [79]:

Table 2 Kinematic point used for $gg \to Hg$. The Higgs boson mass is set to $m_H = 125$ GeV

	E	p_x	p_y	p_z
g	298.17848024073913	0	0	298.17848024073913
g	298.17848024073913	0	0	−298.17848024073913
H	311.27885554899825	−282.56832327194081	−20.783785017815998	31.507187680134837
g	285.07810493248002	282.56832327194081	20.783785017815998	−31.507187680134837

Table 3 Result for Born and virtual amplitude including the QCD corrections to $gg \to Hg$. The renormalisation scale is set to $\mu = m_H = 125$ GeV

	Results for $gg \to Hg$ with the kinematic point from Table 2	
	GoSam result	MCFM result
a_0	$7.274563870476018 \times 10^{-4}$	$7.2745638706144032 \times 10^{-4}$
c_0/a_0	13.195495732443156	13.195495732443119
c_{-1}/a_0	12.160134391476801	12.160134391476900
c_{-2}/a_0	−8.9999999999999698	−9.0000000000000000

Table 4 Kinematic point used for $u\bar{d} \to \nu_e e^+ b\bar{b}$. The W-boson and top-quark mass and width are set to $m_W = 80.25$ GeV, $w_W = 0$, $m_t = 170.9$ GeV and $w_t = 1.5$ GeV

	E	p_x	p_y	p_z
u	250	0	0	250
$\bar{d}$	250	0	0	−250
ν_e	147.53211468467353	24.970405230567895	−18.431576028372117	144.23065114968881
e^+	108.70359662136400	103.25573902554709	−0.54846846595840537	33.976807664202191
b	194.06307653413651	−79.895963003674623	7.4858666717648710	−176.69486288452802
$\bar{b}$	49.701212159825850	−48.330181252440347	11.494177822565669	−1.5125959293629665

$$g_{\gamma\gamma H} = \frac{47e^2}{72\pi^2 v}\left(1 - \frac{14}{705}x_t^2 - \frac{2}{987}x_t^4 + \frac{33}{470}x_W^2 + \frac{57}{6580}x_W^4 + \frac{87}{65800}x_W^6 + \frac{41}{180950}x_W^8 + \frac{5}{119756}x_W^{10} + \frac{213}{26346320}x_W^{12}\right), \tag{13}$$

where $x_t = \frac{m_H}{m_t}$, $x_W = \frac{m_H}{m_W}$ and the corresponding effective Lagrangian is given by

$$\mathcal{L}_{\text{eff}} = -\frac{1}{4}g_{\gamma\gamma H} H F^{\mu\nu} F_{\mu\nu}. \tag{14}$$

Table 3 contains numerical results for $gg \to Hg$ at the phase space point shown in Table 2, where we have used $m_H = 125\,\text{GeV}$, $v^2 = \frac{1}{\sqrt{2}G_F}$, $G_F = 1.16639 \times 10^{-5}\,\text{GeV}^{-2}$.

5.2 Single top production

An example containing complex masses in the loop propagators is the so called s-channel single top quark production, where the top quark has a width $w_t = 1.5\,\text{GeV}$. In Table 5 we give numerical results for the subprocess $u\bar{d} \to \nu_e e^+ b\bar{b}$ at the phase space point given in Table 4. The b-quarks are taken to be massless and the comparison has been performed against the HELAC-NLO code [4].

Table 5 Result for Born and virtual amplitude including the QCD corrections to $u\bar{d} \to \nu_e e^+ b\bar{b}$. The renormalisation scale is set to $\mu = m_t = 170.9$ GeV

	Results for $u\bar{d} \to \nu_e e^+ b\bar{b}$ with the kinematic point from Table 4	
	GoSam result	HELAC-NLO result
a_0	$6.7779888808717541 \times 10^{-13}$	$6.7779888808718329 \times 10^{-13}$
c_0/a_0	8.8976474517729294	8.8976474517739739
c_{-1}/a_0	−4.9124524216371341	−4.9124524216370293
c_{-2}/a_0	−5.3333333333333393	−5.3333333333333073

5.3 Graviton production within models of large extra dimensions

As an example for the usage of GoSam with a model file different from the Standard Model we consider the QCD corrections to graviton production in ADD models [83,84] with large extra dimensions (LED). The corresponding model files in UFO [78] format, which we generated using FeynRules [79,80], are located in the subdirec-

Table 6 Kinematic point used for $u\bar{u} \to G \to \gamma\gamma$

	E	p_x	p_y	p_z
u	250	0	0	250
$\bar{u}$	250	0	0	−250
γ_1	250	218.30931500994714	−29.589212828575324	118.17580743990260
γ_2	250	−218.30931500994714	29.589212828575324	−118.17580743990260

Table 7 Result for the virtual amplitude including the QCD corrections to $u\bar{u} \to G \to \gamma\gamma$ within ADD models of large extra dimensions

	Results for $u\bar{u} \to G \to \gamma\gamma$ with the kinematic point from Table 6	
	GoSam result	Analytic result (Ref. [85])
a_0	$2.6456413225916027 \times 10^{-8}$	$2.6456413225916010 \times 10^{-8}$
c_0/a_0	1.1594725347858084	1.1594725347858106
c_{-1}/a_0	−4.0000000000000009	−4.0000000000000000
c_{-2}/a_0	−2.6666666666666661	−2.6666666666666666

tory `examples/model/LED_UFO`. To import new model files within the GoSam setup, the user should specify the path to the model file in the *process card*. In the given example, this already has been done, i.e. the *process card* contains the line `model=FeynRules,[gosampath]/examples/model/LED_UFO`.
The example process we included in the GoSam-2.0 distribution is $u\bar{u} \to G \to \gamma\gamma$, where G denotes a graviton, and the program calculates the virtual QCD corrections. Note that this example also involves integrals where the rank exceeds the number of propagators, due to the spin-2 nature of the graviton. Running `make test` in the subdirectory `examples/uu_graviton_yy` should produce the result shown in Table 7, using the phase space point given in Table 6. The full process, including an additional jet, has been calculated in [49], where we refer to for details about the parameter settings.

6 Conclusions

We have presented the program package GoSam-2.0, which is a highly automated tool to calculate one-loop multi-particle amplitudes. As the amplitudes at a first stage are produced in an algebraic form, the program offers a lot of flexibility concerning the particle content and the couplings, the choice of the reduction method and the treatment of the rational parts.

GoSam-2.0 can be used to calculate NLO QCD corrections both *within and Beyond* the Standard Model, as well as *electroweak* corrections, in combination with a Monte Carlo program providing the tree-level and NLO real radiation parts. The latter can be interfaced using the Binoth-Les-Houches-Accord, where both BLHA1 and BLHA2 standards are supported. The automated interface to various Monte Carlo programs also offers the possibility to produce parton showered events and to compare different shower Monte Carlo event generators at NLO level.

We also note that the structure of the code is favourable to be used as a building block for the one-loop virtual times singly unresolved real radiation part entering NNLO calculations.

GoSam-2.0 contains many important new features. The installation procedure is extremely simple: all dependencies are provided in one package, and an install script is building the whole package in a completely automated way. Setting up a process is also very user-friendly: the user only has to fill out a well documented text file, the *process card*, where the program automatically chooses appropriate default values for unspecified options.

Improvements in the code generation compared to version 1.0 lead to more compact and faster code. GoSam-2.0 also contains a new integrand reduction method, the integrand decomposition via Laurent expansion, implemented in the library Ninja, which leads to a considerable gain in stability and speed, in particular for amplitudes containing internal masses.

The range of applicability of GoSam also has been extended considerably. In particular, integrals where the rank exceeds the number of propagators (needed e.g. in effective theories) are fully supported, and propagators for spin-2 particles are implemented. The complex mass scheme is supported, including the complexification of the couplings, and several electroweak schemes can be chosen. Moreover, a new system for stability tests and the rescue of 'unstable' phase space points has been implemented. In addition, the program offers the possibility to produce spin-and colour correlated tree-level matrix elements. As a consequence, GoSam-2.0 can provide all the building blocks needed by modern Monte Carlo programs to construct a full NLO event generator, for QCD corrections both within and beyond the Standard Model, as well as electroweak corrections.

Therefore, to follow the strive for precision in the next phases of LHC data taking as well as at a future Linear Collider, not only regarding QCD corrections, GoSam-2.0 can serve as a highly valuable tool.

Acknowledgments We would like to thank the Herwig++ members J. Bellm, S. Gieseke, S. Plätzer, D. Rauch and C. Reuschle for fruitful interaction concerning the implementation of the BLHA2 interface. We are grateful to S. Pozzorini and C. Papadopoulos for comparisons and to P.-F. Monni for helpful discussions. We also would like to thank Joscha Reichel for collaboration on the BSM application of GoSam involving spin-2 particles. Finally, we are indebted to Thomas Reiter for setting the groundwork GoSam is based on. The work of G.C. was supported by DFG Sonderforschungsbereich Transregio 9, Computergestützte Theoretische Teilchenphysik. P.M., H.v.D., G.L. and T.P. are supported by the Alexander von Humboldt Foundation, in the framework of the Sofja Kovaleskaja Award Project "Advanced Mathematical Methods for Particle Physics", endowed by the German Federal Ministry of Education and Research. The work of G.O. was supported in part by the National Science Foundation under Grant PHY-0855489 and PHY-1068550.

Appendix A: Commented example of an input card

Here we give a commented example for the process $e^+e^- \to t\bar{t}$.

In the following it is assumed that the process $e^+e^- \to t\bar{t}$ should be calculated to order $\mathcal{O}(\alpha\alpha_s)$ (QCD corrections). We neglect the exchange of a Z or a Higgs boson and treat the electron as massless. The output directory is assumed to be in the relative path `eett`. A template file for a generic process card (called `eett.in` here) can be generated by invoking the shell command

```
gosam.py --template eett.in
```

The template file `eett.in` then should be edited by the user to define the process specifications. All lines starting with # are comments.

At this point we would like to emphasize that almost all specifications in the process card are options, which will take default values if they are not filled in by the user. The paths to the libraries will be inserted automatically by the install script. The only mandatory fields are the `in` and `out` particles, the perturbative order and the path where to store the process files. Therefore, a minimal process card can look like this:

Listing 1 `eett.in`

```
1  process_path=eett
2  in=    e+, e-
3  out=   t, t~
4  order= gs, 0, 2
```

In order to populate the process subdirectory specified under `process_path` with files for code generation one invokes

```
gosam.py eett.in
```

This will create the subdirectory structure described in Sect. 4.2.

In the following, we will give detailed comments to all the fields and options available in the process card for the example `eett.in`. (Please note that the line numbers on the left are only included for better readability and should *not* be included in your input file).

Listing 2 `eett.in`

```
1  process_name=eett
2  process_path=eett
3  in=    e+, e-
4  out=   t, t~
5  model= smdiag
6  model.options=ewchoose
7  order= gs, 0, 2
8  zero=me
9  one=gs,e
10 regularisation_scheme=dred
11 helicities=
12 qgraf.options=onshell,notadpole, nosnail
13 qgraf.verbatim=True=iprop[Z, 0, 0];\n\
14                true=iprop[H, 0, 0];
15 qgraf.verbatim.lo=
16 qgraf.verbatim.nlo=
17 polvec=numerical
18 diagsum=True
19 reduction_programs=ninja,golem95,samurai
20 extensions=shared
21 debug=nlo
22 select.lo=
23 select.nlo=
24 filter.lo=
25 filter.nlo=
26 filter.module=
27 renorm_beta=True
28 renorm_mqwf=True
29 renorm_decoupling=True
30 renorm_mqse=True
31 renorm_logs=True
32 renorm_gamma5=True
33 reduction_interoperation=-1
34 reduction_interoperation_rescue=-1
35 samurai_scalar=2
36 nlo_prefactors=0
37 PSP_check=True
38 PSP_rescue=True
39 PSP_verbosity=False
40 PSP_chk_th1=8
41 PSP_chk_th2=3
42 PSP_chk_th3=5
43 PSP_chk_kfactor=10000
44 reference-vectors=
45  abbrev.limit=0
46  templates=
47  qgraf.bin=qgraf
48  form.bin=form
49  form.threads=2
50  form.tempdir=/tmp
51  haggies.bin=
52  fc.bin=/usr/bin/gfortran
53  python.bin=python
54  ninja.fcflags=
55  ninja.ldflags=
56  samurai.fcflags=
57  samurai.ldflags=
58  golem95.fcflags=
59  golem95.ldflags=
60  r2=explicit
61  symmetries=family,generation
62  crossings=
```

The comments to the file `eett.in` are as follows.

1 Setting a process name is optional but recommended. All module names will be prefixed with the process name (e.g. `precision` → `eett_precision`). This will avoid name conflicts if at a later stage more than one matrix elements are linked into one executable.

2 The item `process_path` specifies the directory to which all generated files and directories are written. Specification of a process path is mandatory.

3–4 The items `in` and `out` specify the particles of the initial and final state. The particle names must be defined in the selected model file. As the model files usually define mnemonics for the particle names there might be several ways of specifying the same process. Instead of '`e+`' one could have written '`ep`' or '`positron`'. For a complete list of alternative particle names please refer to the documentation of the corresponding model file. Specifying `in` and `out` particles is mandatory.

5 The option `model` specifies which model files should be used in order to generate and evaluate the diagrams. How to import models in `UFO` or `LanHep` format is described in Sect. 4.4. The default for this field is `smdiag`, i.e. the built-in Standard Model file with a diagonal CKM matrix.

6 The option `model.options` can be used to pass options which are specific to a certain model. The default is `ewchoose`, which means that the electroweak scheme is selected automatically according to the given input parameters.

7 The item `order` is a comma separated list with three entries. The first entry specifies a symbol that denotes a coupling constant. In the Standard Model file `sm` the only two possibilities are '`gs`' for the strong coupling constant g_s and '`e`' for the electroweak coupling. The second number is the power of the chosen coupling constant for the tree-level diagrams and the third number specifies the power of that coupling constant for the one-loop diagrams. Note that the numbers refer to the powers in the diagrams of the amplitude rather than the squared amplitude. In the above example the string '`gs, 0, 2`' specifies that the tree-level diagrams should be of order g_s^0 and the one-loop diagrams should be of order g_s^2 and an unspecified power of e in both cases. If there is no tree level, i.e. the process is loop induced, the keyword `NONE` should be put as second item in the list, instead of the tree level power of the coupling.
The values of `order` are translated into a `vsum` constraint in the file `qgraf.dat`.
This field is mandatory.

8–9 The keywords `zero` and `one` specify a set of symbols that should be treated as zero (resp. one). These simplifications are applied at the symbolical level. Only symbols that appear in the `FORM` interface of the model file should be specified here (masses, couplings, CKM-matrix elements, etc). In the example we specify the electron mass '`me`' to be zero and we do not keep the coupling constants in the calculation explicitly ($g_s = e = 1$).
These options can be omitted.

10 The option `regularisation_scheme` allows to choose the dimensional regularisation scheme, in our example `dred` for dimensional reduction, which is the default. `cdr` for "conventional dimensional regularisation" is also possible.

11 `helicites`: a comma separated list of helicities to be calculated. An empty list means that all possible helicities should be generated. The characters correspond to particles 1, 2, ... from left to right.
Example: $e^+e^- \to \gamma\gamma$:
Only three helicity configurations are required; the other ones are either zero or can be obtained by symmetry transformations. This corresponds to
`helicities=+-++,+-+-,+--`
Multiple helicities can be encoded in patterns, which are expanded at the time of code generation. For more details we refer to the manual.

12 `qgraf.options=onshell,notadpole,nosnail`: a list of options which is passed to `QGRAF` via the 'options' line. Possible values (as of qgraf.3.1.1) are the following keywords: onepi, onshell, nosigma, nosnail, notadpole, floop, topol. In our example, it means that external lines are on-shell, i.e. do not contain selfenergy corrections, and that tadpole and snail diagrams are discarded. We refer to the `QGRAF` documentation for more details.

13–16 The value of the option `qgraf.verbatim` is passed verbatim to the file `qgraf.dat`. In our example we suppress the generation of diagrams containing Higgs and Z bosons. As these commands are passed verbatim to `QGRAF`, no mnemonic names are allowed here, e.g. the Higgs particle has to be denoted by '`H`' and cannot be replaced by '`h`'. For a complete list of available options, please consult the `QGRAF` manual. For a complete list of particle names we refer to the documentation of the corresponding model file.
These options can be omitted.

17 `polvec`: by default (`polvec=numerical`), numerical polarisation vectors are used for the massless gauge bosons, rather than producing separate code for each helicity (see Sect. 3.1.3). To switch off the use of numerical polarisation vectors, use `polvec=explicit`.

18 `diagsum`: if `True`, one-loop diagrams sharing some propagators are combined before the algebraic reduction. The default is `diagsum = True`.

19 The option `reduction_programs` allows to choose the amplitude reduction method. If several choices are given, the code is produced such that the reduction methods can be switched at runtime. The default is `ninja, golem95`.

20 `extensions`: this option contains a list of useful extensions to the core of the program, which operate at the code generation stage. The currently available extensions are

– `autotools`: use autotools to generate Makefiles
– `shared`: create shared libraries (i.e. dynamically linkable code rather than static libraries). This extension is enabled by default when using the `autotools` extension.
– `f77`: in combination with the BLHA interface it generates a file `olp_module.f90` linkable with Fortran77.
– `noformopt`: disables diagram optimization using `FORM`
– `gaugecheck`: modifies the massless gauge boson wave functions to allow for a check of gauge invariance for processes involving gluons or photons.
– `customspin2prop` allows to replace the propagator of spin-2 particles with a custom function (we refer to the manual for details).

In our example `shared` tells the program to build dynamic rather than static libraries.

21 `debug`: can take the values `lo`, `nlo`, `all`. It sets the level of information printed to the file `matrix/debug.xml` when running the test program.

22 `select.lo`: can be used to select/discard diagrams by their diagram numbers. It can contain a list of integer numbers, indicating leading order diagrams to be selected. If no list is given, all diagrams are selected. Otherwise, all diagrams whose numbers are not in the list will be discarded. The list may also contain ranges, with increments different from one, e.g. `select.lo=1,2,5:10:3, 50:53` is equivalent to `select.lo=1,2,5,8,50,51,52,53`, i.e. the 3 in `5:10:3` is the increment.

23 `select.nlo`: analogous to `select.lo`, for the one-loop diagrams.

24 `filter.lo`: a python function which provides a filter for tree diagrams.
Example: `filter.lo=lambda d: d.iprop (Z) == 1 and d.vertices(Z, U, Ubar) == 0` filters out diagrams containing exactly one Z propagator and no $Zu\bar{u}$ couplings.

25 `filter.nlo`: analogous to `filter.lo`, for the one-loop diagrams. For details we refer to the manual.

26 `filter.module`: a python file of predefined functions which can be used as filters.

27 `renorm_beta`: activates or disables beta function renormalisation. The default is `True`.

28 `renorm_mqwf`: activates or disables UV countertems coming from external massive quarks. The default is `True`.

29 `renorm_decoupling`: activates or disables UV counterterms coming from massive quark loops. The default is `True`.

30 `renorm_mqse`: activates or disables the UV counterterm coming from the massive quark propagators. The default is `True`.

31 `renorm_logs`: activates or disables the logarithmic finite terms associated with the UV counterterms. The default is `True`.

32 `renorm_gamma5`: activates finite renormalisation for axial couplings in the 't Hooft Veltman scheme (CDR). Implemented for QCD only, works only with the built-in model files. The default is `True`.

33 `reduction_interoperation`: denotes the reductuion libraries to be used. Possible values are: ninja, samurai, golem95 (listing all of them simultaneously is possible). A value of -1 lets GoSam decide. See `common/config.f90` for details.

34 `reduction_interoperation_rescue`: specifies the reduction library to be used to rescue 'unstable points'. A value of -1 lets GoSam decide.

35 `samurai_scalar`: integer which specifies the library Samurai chooses for the basis integrals. 1: QCDLoop, 2: OneLOop, 3: Golem95C. The default is 2.

36 `nlo_prefactors`: can take the integer values 0,1,2, which have the following meaning:

0 : a factor of $\alpha_{(s)}/(2\pi)$ is not included in the NLO result
1 : a factor of $1/(8\pi^2)$ is not included in the NLO result
2 : the NLO result includes all prefactors (see also manual).

Note, however, that the factor of $1/\Gamma(1-\epsilon)$ is not included in any of the cases. Please note also that `nlo_prefactors=0` is enforced in `test.f90` in order to recognize rational numbers for the pole coefficients. In the OLP interface mode (BLHA/BLHA2), the default is `nlo_prefactors=2`.

37 `PSP_check`: allows to switch the stability test of the full amplitude for each phase space point on or off. If `PSP_check` is set to `False`, the following flags concerning `PSP_rescue` and the various thresholds for the rescue system have no effect. Details about the stability tests are given in Sect. 3.4. Please note that this test only works for QCD with the built-in model files. The default is `PSP_check= True`.

38 `PSP_rescue`: activates the phase space point rescue system based on the estimated accuracy of the finite part. The accuracy is estimated using information on the single pole accuracy and the cancellation between the cut-constructible part and R_2. The default is `PSP_rescue= True`.

39 `PSP_verbosity`: sets the verbosity of the `PSP_check.verbosity = False` means no output, `verbosity = True` means that bad points are written to a file
`gs_badpts.log`. The default is `verbosity = False`.

40 `PSP_chk_th1`: an integer indicating the number of desired accurate digits of the single pole coefficient. For poles coefficients more precise than this threshold the finite part is not checked separately. Note that this works only for QCD, with the built-in model files. The default is 8.

41 `PSP_chk_th2`: threshold (number of accurate digits) to declare a phase space point as *bad point*, based on the precision of the pole coefficient. Points with precision less than this threshold are directly reprocessed with the rescue system (if available), or declared as unstable. According to the verbosity level set, such points are written to a file and not used when the code is interfaced to an external Monte Carlo program using the new BLHA2 standards. The default is 3.

42 `PSP_chk_th3`: threshold (number of accurate digits) to declare a phase space point as *bad point*, based on the precision of the finite part estimated with a *rotation*. According to the verbosity level set, such points are written to a file and not used when the code is interfaced to an external Monte Carlo program using the new BLHA2 standards. The default is 5.

43 `PSP_chk_kfactor`: threshold on the K-factor to declare a phase space point as *bad point*. According to the verbosity level set, such points are written to a file and not used when the code is interfaced to an external Monte Carlo program using the new BLHA2 standards. The default is 10000.

44 `reference-vectors`: comma separated list of reference vectors for massive fermions and vector bosons. If no reference vectors are assigned here, the program picks the reference vectors automatically. Each entry of the list has to be of the form $\langle index\rangle : \langle index\rangle$. Example:

```
in=g,u
out=t,W+
reference-vectors=1:2,3:4,4:3
```

In this example, the gluon (particle 1) takes the momentum k_2 as reference momentum for the polarisation vector. The massive top quark (particle 3) uses the light-cone projection l_4 of the W-boson as reference direction for its own momentum splitting. Similarly, the momentum of the W-boson is split into a direction l_4 and one along l_3.

45 `abbrev.limit`: maximum number of instructions per subroutine when calculating abbreviations. The default is 0, which means that no maximum is set.

46 `templates`: path pointing to the directory containing the template files for the process. If not set, the program uses the directory ⟨ gosam_path⟩/templates. The directory must contain a file called `template.xml`.

47 `qgraf.bin`: path to the `QGraf` executable. The default path will be set by the installation script.

48 `form.bin`: path to the `FORM` executable. The default path will be set by the installation script.

49 `form.threads`: the number of `FORM` threads when using `tform`, the parallel version of `FORM`. The default is 2.

50 `form.tempdir`: the temporary directory where `FORM` can store (large) intermediate files. the default is `/tmp`.

51 `haggies.bin`: path to the `haggies` executable. The default path will be set by the installation script.

52 `fc.bin`: path to the `Fortran` compiler. The default path will be set by the installation script.

53 `python.bin`: path to the `python` executable. The default path will be set by the installation script.

54 `ninja.fcflags`: compiler flags to compile with NINJA. The default will be set by the installation script.

55 `ninja.ldflags`: LDFLAGS required to link the NINJA library. The default will be set by the installation script.

56 `samurai.fcflags`: compiler flags to compile with SAMURAI. The default will be set by the installation script.

57 `samurai.ldflags`: LDFLAGS required to link the SAMURAI library. The default will be set by the installation script.

58 `golem95.fcflags`: compiler flags to compile with GOLEM95C. The default will be set by the installation script.

59 `golem95.ldflags`: LDFLAGS required to link the GOLEM95C library. The default will be set by the installation script.

60 `r2`: treatment of the rational part R_2. The possibilities are:

- `implicit`: μ^2 terms are kept in the numerator and reduced at runtime,
- `explicit`: μ^2 terms are reduced analytically,
- `off`: all μ^2 terms are set to zero.

The default is `r2=explicit`.

61 `symmetries`: this information is used when the list of helicity configurations is generated. An empty list means that all helicity configurations will be generated, even if some of them could be mapped onto each other. Possible values are:

- `flavour`: assumes that no flavour changing interactions are present. When calculating the list of helicities, fermions with PDG codess 1-6 are assumed not to mix.
- `family`: flavour changing only within families. When calculating the list of helicities, fermion lines with PDG codes 1-6 are assumed to mix only within families, i.e. a quark line connecting an up with a down quark would be considered, while up-bottom would be discarded.
- `lepton`: means for leptons what ‘flavour’ means for quarks.
- `generation`: means for leptons what ‘family’ means for quarks.
- $\langle n\rangle = \langle h\rangle$: restriction of particle helicities, e.g. `1=-`, `2=+` specifies helicities of particles 1 and 2.
- $\%\langle n\rangle = \langle h\rangle$ restriction by PDG code, e.g. %23=+- specifies the helicity of all Z-bosons to be ’+’ and ’-’ only (no ’0’ polarisation),
  $\%\langle n\rangle$ refers to both $+n$ and $-n$,
  $\%+\langle n\rangle$ refers to $+n$ only, $\%-\langle n\rangle$ refers to $-n$ only.

62 `crossings`: a list of crossed processes derived from this process. For each process in the list a module similar to `matrix.f90` is generated.

Example:
```
process_name=ddx_uux
in=1,-1
out=2,-2
crossings = dxd_uux: -1 1  →  2 -2,
ud_ud: 2 1→2 1
```

Appendix B: Higher rank integrals

Higher rank integrals are implemented in all reduction libraries included in GOSAM. NINJA and SAMURAI are based on integrand reduction, as described in Sect. 3.2, GOLEM95C provides tensor integrals, using a tensor reduction method and a basis of scalar integrals which has been designed to provide numerical stability in problematic phase space regions, for example in the limit of small Gram determinants.

In the following we briefly sketch the main features of the higher rank extensions for both approaches, more details can be found in [33,36,60].

B.1: Integrand reduction approach

If the rank r of a one-loop integrand is not larger than the number of propagators N, the respective integral can be written as the following combination of known master integrals

$$\begin{aligned}\mathcal{M}= &\sum_{\{i_1,i_2,i_3,i_4\}}\left\{c_0^{(i_1i_2i_3i_4)}I_{i_1i_2i_3i_4}+c_4^{(i_1i_2i_3i_4)}I_{i_1i_2i_3i_4}[\mu^4]\right\}\\ &+\sum_{\{i_1,i_2,i_3\}}\left\{c_0^{(i_1i_2i_3)}I_{i_1i_2i_3}+c_7^{(i_1i_2i_3)}I_{i_1i_2i_3}[\mu^2]\right\}\\ &+\sum_{\{i_1,i_2\}}\left\{c_0^{(i_1i_2)}I_{i_1i_2}+c_1^{(i_1i_2)}I_{i_1i_2}[(q+p_{i_1})\cdot e_2]\right.\\ &\left.+c_2^{(i_1i_2)}I_{i_1i_2}[((q+p_{i_1})\cdot e_2)^2]+c_9^{(i_1i_2)}I_{i_1i_2}[\mu^2]\right\}\\ &+\sum_{i_1}c_0^{(i_1)}I_{i_1},\end{aligned} \tag{15}$$

where

$$I_{i_1\cdots i_k}[\alpha]\equiv\int d^{4-2\epsilon}q\,\frac{\alpha}{D_{i_1}\cdots D_{i_k}},\qquad I_{i_1\cdots i_k}\equiv I_{i_1\cdots i_k}[1],$$
$$D_j=(q+p_j)^2-m_j^2 \tag{16}$$

with

$$\begin{aligned}I_{i_1i_2i_3i_4}[\mu^4]&=-\frac{1}{6}+\mathcal{O}(\epsilon)\\ I_{i_1i_2i_3}[\mu^2]&=\frac{1}{2}+\mathcal{O}(\epsilon)\\ I_{i_1i_2}[\mu^2]&=-\frac{1}{6}\left(p_{i_1}^2-3(m_{i_1}+m_{i_2})\right)+\mathcal{O}(\epsilon).\end{aligned} \tag{17}$$

In the case where $r = N + 1$ the integral is generalized as

$$\begin{aligned}\mathcal{M}^{(r=N+1)} &= \mathcal{M}^{(r=N)} + \sum_{\{i_1,i_2,i_3\}} c_{14}^{(i_1 i_2 i_3)} I_{i_1 i_2 i_3}[\mu^4] \\ &+ \sum_{\{i_1,i_2\}} \Big\{ c_{10}^{(i_1 i_2)} I_{i_1 i_2}[\mu^2 (q+p_{i_1})\cdot e_2)] \\ &+ c_{13}^{(i_1 i_2)} I_{i_1 i_2}[((q+p_{i_1})\cdot e_2)^3] \Big\} \\ &+ \sum_{i_1} \Big\{ c_{14} I_{i_1}[\mu^2] + c_{15}^{(i_1)} I_{i_1}[((q+p_{i_1}) \\ &\cdot e_3)((q+p_{i_1})\cdot e_4)] \Big\}. \qquad (18)\end{aligned}$$

The three integrals in Eq. (18) whose numerator is proportional to μ^2 are finite and contribute to the rational part of the amplitude. They have been computed in Ref. [29,36] and they read

$$\begin{aligned} I_{i_1 i_2 i_3}[\mu^4] &= \frac{i\pi^2}{6}\left(\frac{s_{i_2 i_1}+s_{i_3 i_2}+s_{i_1 i_3}}{4} \right. \\ &\left. - m_{i_1}^2 - m_{i_2}^2 - m_{i_3}^2\right) + \mathcal{O}(\epsilon) \qquad (19)\\ I_{i_1 i_2}[\mu^2 ((q+p_{i_1})\cdot e_2)] &= i\pi^2 \frac{((p_{i_2}-p_{i_1})\cdot e_2)}{12}\Big(s_{i_2 i_1} \\ &- 2 m_{i_1}^2 - 4 m_{i_2}^2\Big) + \mathcal{O}(\epsilon) \qquad (20)\\ I_{i_1}[\mu^2] &= \frac{i\pi^2 m_{i_1}^4}{2} + \mathcal{O}(\epsilon) \qquad (21)\end{aligned}$$

where $s_{ij} \equiv (p_i - p_j)^2$. The tadpole of rank 2 appearing in Eq. (18) can be written in terms of the scalar tadpole I_{i_1} as

$$\begin{aligned} &I_{i_1}[((q+p_{i_1})\cdot e_3)((q+p_{i_1})\cdot e_4)] \\ &= m_{i_1}^2 \frac{(e_3\cdot e_4)}{4}\left(I_{i_1} + \frac{i\pi^2 m_{i_1}^2}{2}\right) + \mathcal{O}(\epsilon). \qquad (22)\end{aligned}$$

Finally, since the vector e_2 can always be chosen to be massless, the bubble integral of rank 3 appearing in Eq. (18) is proportional to the form factor B_{111},

$$I_{i_1 i_2}[((q+p_{i_1})\cdot e_2)^3] = ((p_{i_2}-p_{i_1})\cdot e_2)^3 \, B_{111}(s_{i_2 i_1}, m_{i_1}^2, m_{i_2}^2). \qquad (23)$$

The latter can be computed using the formulas of Ref. [86], as a function of form factors of scalar integrals B_0. In the special case with $s_{i_2 i_1} = 0$ we use Eq. (A.6.2) and (A.6.3) of that reference. For the general case $s_{i_2 i_1} \neq 0$ we use instead the following formula [34]

$$\begin{aligned} B_{111}(s_{i_2 i_1}, m_{i_1}^2, m_{i_2}^2) &= \frac{1}{4 s_{i_2 i_1}^3}\Big\{ s_{i_2 i_1}\Big(m_{i_1}^2 I_{i_1} + I_{i_1}[\mu^2] \\ &- m_{i_2}^2 I_{i_2} - I_{i_2}[\mu^2] \\ &- 4 I_{i_1 i_2}[\mu^2((q+p_{i_1})\cdot(p_{i_2}-p_{i_1}))] \\ &- 4 m_{i_1}^2 I_{i_1 i_2}[(q+p_{i_1})\cdot(p_{i_2}-p_{i_1})]\Big) \\ &+ 4(m_{i_2}^2 - m_{i_1}^2 - s_{i_2 i_1}) \\ &\times I_{i_1 i_2}[((q+p_{i_1})\cdot(p_{i_2}-p_{i_1}))^2]\Big\}. \qquad (24)\end{aligned}$$

B.2: Tensor reduction approach

In the tensor reduction approach, the tensor integrals are written in terms of linear combinations of scalar *form factors* and all possible combinations of external momenta and metric tensors carrying the Lorentz structure. The form factors themselves are then reduced to a convenient set of basis integrals. It is well known that, due to the 4-dimensionality (resp. $D = 4-2\epsilon$ dimensionality in dimensional regularisation) of space-time, integrals with $N \geq 6$ can be reduced iteratively to 5-point integrals. Therefore form factors for $N \geq 6$ are never needed. The general form factor decomposition of an arbitrary tensor integral can be written as

$$\begin{aligned} I_N^{D,\mu_1\ldots\mu_r}(a_1,\ldots,a_r;S) &= \int \frac{d^D k}{i\pi^{D/2}} \frac{q_{a_1}^{\mu_1}\cdots q_{a_r}^{\mu_r}}{\prod_{j=1}^N (q_j^2 - m_j^2 + i\delta)} \\ &= \sum_{j_1,\ldots,j_r\in S} \left[\Delta^{\cdot}_{j_1\cdot}\cdots\Delta^{\cdot}_{j_r\cdot}\right]^{\{\mu_1\ldots\mu_r\}}_{\{a_1\ldots a_r\}} A^{N,r}_{j_1\ldots j_r}(S) \\ &+ \sum_{j_1,\ldots,j_{r-2}\in S} \left[g^{\cdot\cdot}\Delta^{\cdot}_{j_1\cdot}\cdots\Delta^{\cdot}_{j_{r-2}\cdot}\right]^{\{\mu_1\ldots\mu_r\}}_{\{a_1\ldots a_r\}} B^{N,r}_{j_1\ldots j_{r-2}}(S) \\ &+ \sum_{j_1,\ldots,j_{r-4}\in S} \left[g^{\cdot\cdot}g^{\cdot\cdot}\Delta^{\cdot}_{j_1\cdot}\cdots\Delta^{\cdot}_{j_{r-4}\cdot}\right]^{\{\mu_1\ldots\mu_r\}}_{\{a_1\ldots a_r\}} C^{N,r}_{j_1\ldots j_{r-4}}(S) \\ &+ \sum_{j_1,\ldots,j_{r-4}\in S} \left[g^{\cdot\cdot}g^{\cdot\cdot}g^{\cdot\cdot}\Delta^{\cdot}_{j_1\cdot}\cdots\Delta^{\cdot}_{j_{r-6}\cdot}\right]^{\{\mu_1\ldots\mu_r\}}_{\{a_1\ldots a_r\}} D^{N,r}_{j_1\ldots j_{r-6}}(S) \\ &+ \cdots, \qquad (25)\end{aligned}$$

where $\Delta_{ij}^{\mu} = r_i^{\mu} - r_j^{\mu}$ are differences of external momenta r, and $q_a = k + r_a$. The notation $[\cdots]^{\{\mu_1\cdots\mu_r\}}_{\{a_1\cdots a_r\}}$ stands for the distribution of the r Lorentz indices μ_i, and the momentum labels a_i, to the vectors $\Delta^{\mu_i}_{j\,a_i}$ and metric tensors in all distinguishable ways. Note that the choice $r_N = 0$, $a_i = N \; \forall i$ leads to the well known representation in terms of external momenta where the labels a_i are not necessary, but we prefer a completely shift invariant notation here.

S denotes an ordered set of propagator labels, corresponding to the momenta forming the kinematic matrix $\mathcal{S}$, defined by

$$\mathcal{S}_{ij} = (r_i - r_j)^2 - m_i^2 - m_j^2, \quad i, j \in \{1, \ldots, N\}.$$

We should point out that the form factors of type $D^{N,r}_{j_1 \ldots j_{r-6}}$ and beyond, i.e. form factors associated with three or more metric tensors, are not needed for integrals where the rank r does not exceed the number N of propagators, no matter what the value of N is, because integrals with $N \geq 6$ can be reduced algebraically to pentagons.

The program GOLEM95C reduces the form factors $A, \ldots, D$ internally to a set of basis integrals, i.e. the endpoints of the reduction (they do not form a basis in the mathematical sense, as some of them are not independent). The choice of the basis integrals can have important effects on the numerical stability in certain kinematic regions. Our reduction endpoints are 4-point functions in 6 dimensions I_4^6, which are IR and UV finite, 4-point functions in $D+4$ dimensions, and various 3-point, 2-point and 1-point functions. A special feature of GOLEM95C is that the algebraic reduction to scalar basis integrals is automatically replaced by a stable and fast one-dimensional numerical integration of parametric integrals corresponding to tensor rather than scalar integrals in kinematic situations where a further reduction would lead to spurious inverse Gram determinants tending to zero. This leads to improved numerical stability.

The extension of GOLEM95C to higher rank integrals [60] follows the reduction formalism as outlined in [26]. However, the extension of the formalism to rank six pentagons required some care, as the latter develop an UV divergence, and therefore $\mathcal{O}(\epsilon)$ terms occurring in the reduction need to be taken into account at intermediate stages.

The rational parts of all the integrals contained in GOLEM95C can be extracted separately, and analytic formulae for $r \leq N$ are provided in [87]. The results for those integrals which are relevant for the higher rank extension can be extracted from [29], where formulae for all possible rational parts are given in a general form. The ones which are relevant for the higher rank extension which have not been given above already are listed explicitly here, where the notation conventions are $k^\mu_{(D)} = \hat{k}^\mu_{(4)} + \tilde{k}^\mu_{(-2\epsilon)}$, $k^2_{(D)} = \hat{k}^2 + \tilde{k}^2$,

$$I_N^{D,\alpha;\mu_1 \ldots \mu_r}(a_1, \ldots a_r; S) \equiv \int \frac{\mathrm{d}^D k}{i\pi^{D/2}} \frac{\left(\tilde{k}^2\right)^\alpha \hat{q}^{\mu_1}_{a_1} \cdots \hat{q}^{\mu_r}_{a_r}}{\prod_{j=1}^N (q_j^2 - m_j^2 + i\delta)}, \tag{26}$$

with the results [60]

$$\begin{aligned} I_5^{D,3}(S) &= -\frac{1}{12} + \mathcal{O}(\epsilon), \\ I_5^{D,2;\mu_1\mu_2}(a_1, a_2; S) &= -\frac{1}{48}\, g^{\mu_1\mu_2} + \mathcal{O}(\epsilon), \\ I_5^{D,1;\mu_1\cdots\mu_4}(a_1, \ldots, a_4; S) &= -\frac{1}{96}\, \big[g^{\mu_1\mu_2} g^{\mu_3\mu_4} \\ &\quad + g^{\mu_1\mu_3} g^{\mu_2\mu_4} + g^{\mu_1\mu_4} g^{\mu_2\mu_3}\big] + \mathcal{O}(\epsilon), \\ \epsilon I_4^{D+6}(S) &= \frac{1}{240}\left(\sum_{i,j=1}^4 (\Delta_{ij}^2 \right. \\ &\quad \left. - m_i^2 - m_j^2) - 2\sum_{i=1}^4 m_i^2\right) + \mathcal{O}(\epsilon). \end{aligned} \tag{27}$$

References

1. ATLAS Collaboration, G. Aad et al., Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. Phys. Lett. B **716** 1–29 (2012). arXiv:1207.7214
2. CMS Collaboration, S. Chatrchyan et al., Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. Phys. Lett. B **716** 30–61 (2012). arXiv:1207.7235
3. T. Hahn, Feynman Diagram Calculations with FeynArts, FormCalc, and LoopTools. PoS **ACAT2010** 078 (2010). arXiv:1006.2231
4. G. Bevilacqua, M. Czakon, M. Garzelli, A. van Hameren, A. Kardos et al., HELAC-NLO. Comput. Phys. Commun. **184**, 986–997 (2013). arXiv:1110.1499
5. V. Hirschi, R. Frederix, S. Frixione, M.V. Garzelli, F. Maltoni et al., Automation of one-loop QCD corrections. JHEP **1105**, 044 (2011). arXiv:1103.0621
6. G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia et al., Automated one-loop calculations with GoSam. Eur. Phys. J. C **72**, 1889 (2012). arXiv:1111.2034
7. S. Badger, B. Biedermann, P. Uwer, V. Yundin, Numerical evaluation of virtual corrections to multi-jet production in massless QCD. Comput. Phys. Commun. **184**, 1981–1998 (2013). arXiv:1209.0100
8. C. Berger, Z. Bern, L. Dixon, F. Febres Cordero, D. Forde et al., An automated implementation of on-shell methods for one-loop amplitudes. Phys. Rev. D **78**, 036003 (2008). arXiv:0803.4180
9. A. Bredenstein, A. Denner, S. Dittmaier, S. Pozzorini, NLO QCD corrections to top anti-top bottom anti-bottom production at the LHC: 2. full hadronic results. JHEP **1003**, 021 (2010). arXiv:1001.4006
10. S. Alioli, P. Nason, C. Oleari, E. Re, A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX. JHEP **1006**, 043 (2010). arXiv:1002.2581
11. J.M. Campbell, R.K. Ellis, C. Williams, Vector boson pair production at the LHC. JHEP **1107**, 018 (2011). arXiv:1105.0020
12. K. Arnold, J. Bellm, G. Bozzi, M. Brieg, F. Campanario, et al., VBFNLO: A Parton Level Monte Carlo for Processes with Electroweak Bosons-Manual for Version 2.5.0. arXiv:1107.4038
13. S. Becker, D. Goetz, C. Reuschle, C. Schwan, S. Weinzierl, NLO results for five, six and seven jets in electron-positron annihilation. Phys. Rev. Lett. **108**, 032005 (2012). arXiv:1111.1733
14. F. Cascioli, P. Maierhofer, S. Pozzorini, Scattering amplitudes with open loops. Phys. Rev. Lett. **108**, 111601 (2012). arXiv:1111.5206
15. S. Actis, A. Denner, L. Hofer, A. Scharf, S. Uccirati, Recursive generation of one-loop amplitudes in the Standard Model. JHEP **1304**, 037 (2013). arXiv:1211.6316
16. G. Ossola, C.G. Papadopoulos, R. Pittau, Reducing full one-loop amplitudes to scalar integrals at the integrand level. Nucl. Phys. B **763**, 147–169 (2007). hep-ph/0609007
17. G. Ossola, C.G. Papadopoulos, R. Pittau, Numerical evaluation of six-photon amplitudes. JHEP **0707**, 085 (2007). arXiv:0704.1271

18. G. Ossola, C.G. Papadopoulos, R. Pittau, CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes. JHEP **03**, 042 (2008). arXiv:0711.3596
19. P. Mastrolia, G. Ossola, On the integrand-reduction method for two-loop scattering amplitudes. JHEP **1111**, 014 (2011). arXiv:1107.6041
20. S. Badger, H. Frellesvig, Y. Zhang, Hepta-cuts of two-loop scattering amplitudes. JHEP **1204**, 055 (2012). arXiv:1202.2019
21. Y. Zhang, Integrand-level reduction of loop amplitudes by computational algebraic geometry methods. JHEP **1209**, 042 (2012). arXiv:1205.5707
22. P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro, Scattering amplitudes from multivariate polynomial division. Phys. Lett. B **718**, 173–177 (2012). arXiv:1205.7087
23. P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro, Multiloop integrand reduction for dimensionally regulated amplitudes. Phys. Lett. B **727**, 532–535 (2013). arXiv:1307.5832
24. W.T. Giele, Z. Kunszt, K. Melnikov, Full one-loop amplitudes from tree amplitudes. JHEP **0804**, 049 (2008). arXiv:0801.2237
25. P. Mastrolia, G. Ossola, T. Reiter, F. Tramontano, Scattering amplitudes from unitarity-based reduction algorithm at the integrand-level. JHEP **1008**, 080 (2010). arXiv:1006.0710
26. T. Binoth, J.P. Guillet, G. Heinrich, E. Pilon, C. Schubert, An Algebraic/numerical formalism for one-loop multi-leg amplitudes. JHEP **0510**, 015 (2005). hep-ph/0504267
27. G. Heinrich, G. Ossola, T. Reiter, F. Tramontano, Tensorial reconstruction at the integrand level. JHEP **1010**, 105 (2010). arXiv:1008.2441
28. T. Binoth, J. Guillet, G. Heinrich, Reduction formalism for dimensionally regulated one loop N point integrals. Nucl. Phys. B **572**, 361–386 (2000). hep-ph/9911342
29. T. Reiter, Automated evaluation of one-loop six-point processes for the LHC. Ph.D. Thesis, The University of Edinburgh (2008). arXiv:0903.0947
30. G. Cullen, J. Guillet, G. Heinrich, T. Kleinschmidt, E. Pilon et al., Golem95C: a library for one-loop integrals with complex masses. Comput. Phys. Commun. **182**, 2276–2284 (2011). arXiv:1101.5595
31. P. Mastrolia, G. Ossola, C. Papadopoulos, R. Pittau, Optimizing the reduction of one-loop amplitudes. JHEP **0806**, 030 (2008). arXiv:0803.3964
32. P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro, H. van Deurzen, The integrand reduction of one- and two-loop scattering amplitudes. PoS **LL2012**, 028 (2012). arXiv:1209.5678
33. H. van Deurzen, Associated Higgs production at NLO with GoSam. Acta Phys. Polon. B **44**(11), 2223–2230 (2013)
34. T. Peraro, Ninja: automated integrand reduction via Laurent expansion for one-loop amplitudes. arXiv:1403.1229
35. H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, et al., Multi-leg one-loop massive amplitudes from integrand reduction via Laurent expansion. arXiv:1312.6678
36. P. Mastrolia, E. Mirabella, T. Peraro, Integrand reduction of one-loop scattering amplitudes through Laurent series expansion. JHEP **1206**, 095 (2012). arXiv:1203.0291
37. N. Greiner, A. Guffanti, T. Reiter, J. Reuter, NLO QCD corrections to the production of two bottom-antibottom pairs at the LHC. Phys. Rev. Lett. **107**, 102002 (2011). arXiv:1105.3624
38. N. Greiner, G. Heinrich, P. Mastrolia, G. Ossola, T. Reiter et al., NLO QCD corrections to the production of W+ W- plus two jets at the LHC. Phys. Lett. B **713**, 277–283 (2012). arXiv:1202.6004
39. H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella et al., NLO QCD corrections to the production of Higgs plus two jets at the LHC. Phys. Lett. B **721**, 74–81 (2013). arXiv:1301.0493
40. T. Gehrmann, N. Greiner, G. Heinrich, Photon isolation effects at NLO in $\gamma\gamma$ + jet final states in hadronic collisions. JHEP **1306**, 058 (2013). arXiv:1303.0824
41. G. Luisoni, P. Nason, C. Oleari, F. Tramontano, $HW^{\pm}$/HZ + 0 and 1 jet at NLO with the POWHEG BOX interfaced to GoSam and their merging within MiNLO. JHEP **1310**, 083 (2013). arXiv:1306.2542
42. S. Hoeche, J. Huang, G. Luisoni, M. Schoenherr, J. Winter, Zero and one jet combined NLO analysis of the top quark forward-backward asymmetry. Phys. Rev. D **88**, 014040 (2013). arXiv:1306.2703
43. G. Cullen, H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia et al., NLO QCD corrections to Higgs boson production plus three jets in gluon fusion. Phys. Rev. Lett. **111**, 131801 (2013). arXiv:1307.4737
44. H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola et al., NLO QCD corrections to Higgs boson production in association with a top quark pair and a jet. Phys. Rev. Lett. **111**, 171801 (2013). arXiv:1307.8437
45. T. Gehrmann, N. Greiner, G. Heinrich, Precise qcd predictions for the production of a photon pair in association with two jets. Phys. Rev. Lett. **111**, 222002 (2013). [arXiv:1308.3660]
46. M.J. Dolan, C. Englert, N. Greiner, M. Spannowsky, Further on up the road: $hhjj$ production at the LHC. Phys. Rev. Lett. **112**, 101802 (2014). arXiv:1310.1084
47. G. Heinrich, A. Maier, R. Nisius, J. Schlenk, J. Winter, NLO QCD corrections to WWbb production with leptonic decays in the light of top quark mass and asymmetry measurements. arXiv:1312.6659
48. G. Cullen, N. Greiner, G. Heinrich, Susy-QCD corrections to neutralino pair production in association with a jet. Eur. Phys. J. C **73**, 2388 (2013). arXiv:1212.5154
49. N. Greiner, G. Heinrich, J. Reichel, J.F. von Soden-Fraunhofen, NLO QCD corrections to diphoton plus jet production through graviton exchange. JHEP **1311**, 028 (2013). arXiv:1308.2194
50. T. Binoth, F. Boudjema, G. Dissertori, A. Lazopoulos, A. Denner et al., A proposal for a standard interface between Monte Carlo tools and one-loop programs. Comput. Phys. Commun. **181**, 1612–1622 (2010). arXiv:1001.1307
51. S. Alioli, S. Badger, J. Bellm, B. Biedermann, F. Boudjema et al., Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and one-loop programs. Comput. Phys. Commun. **185**, 560–571 (2014). arXiv:1308.3462
52. http://gosam.hepforge.org
53. P. Nogueira, Automatic Feynman graph generation. J. Comput. Phys. **105**, 279–289 (1993)
54. J. Vermaseren, New features of FORM, math-ph/0010025
55. J. Kuipers, T. Ueda, J. Vermaseren, J. Vollinga, FORM version 4.0. Comput. Phys. Commun. **184**, 1453–1467 (2013). arXiv:1203.6543
56. G. Cullen, M. Koch-Janusz, T. Reiter, Spinney: A Form Library for Helicity Spinors. Comput. Phys. Commun. **182**, 2368–2387 (2011). arXiv:1008.0803
57. T. Reiter, Optimising code generation with haggies. Comput. Phys. Commun. **181**, 1301–1331 (2010). arXiv:0907.3714
58. R. Ellis, W.T. Giele, Z. Kunszt, K. Melnikov, Masses, fermions and generalized D-dimensional unitarity. Nucl. Phys. B **822**, 270–282 (2009). arXiv:0806.3467
59. T. Binoth, J.-P. Guillet, G. Heinrich, E. Pilon, T. Reiter, Golem95: A Numerical program to calculate one-loop tensor integrals with up to six external legs. Comput. Phys. Commun. **180**, 2317–2330 (2009). arXiv:0810.0992
60. J. P. Guillet, G. Heinrich, J. von Soden-Fraunhofen, Tools for NLO automation: extension of the golem95C integral library. arXiv:1312.3887
61. A. van Hameren, OneLOop: for the evaluation of one-loop scalar functions. Comput. Phys. Commun. **182**, 2427–2438 (2011). arXiv:1007.4716
62. T. Hahn, M. Perez-Victoria, Automatized one loop calculations in four-dimensions and D-dimensions. Comput. Phys. Commun. **118**, 153–165 (1999). hep-ph/9807565

63. J. Fleischer, T. Riemann, A Complete algebraic reduction of one-loop tensor Feynman integrals. Phys. Rev. D **83**, 073004 (2011). arXiv:1009.4436
64. J. Fleischer, T. Riemann, V. Yundin, New developments in PJFry. PoS LL **2012**, 020 (2012). arXiv:1210.4095
65. S. Actis, A. Denner, L. Hofer, A. Scharf, S. Uccirati, EW and QCD one-loop amplitudes with RECOLA. arXiv:1311.6662
66. S. Catani, S. Dittmaier, Z. Trocsanyi, One loop singular behavior of QCD and SUSY QCD amplitudes with massive partons. Phys. Lett. B **500**, 149–160 (2001). hep-ph/0011222
67. S. Badger, B. Biedermann, P. Uwer, NGluon: A package to calculate one-loop multi-gluon amplitudes. Comput. Phys. Commun. **182**, 1674–1692 (2011). arXiv:1011.2900
68. Proceedings of the Les Houches 2013 workshop on Physics at TeV colliders (2014)
69. J. Bellm, S. Gieseke, D. Grellscheid, A. Papaefstathiou, S. Plätzer, et al., Herwig++ 2.7 Release Note. arXiv:1310.6877
70. S. Plätzer, S. Gieseke, Dipole showers and automated NLO matching in Herwig++. Eur. Phys. J. C **72**, 2187 (2012). arXiv:1109.6256
71. A. Denner, S. Dittmaier, M. Roth, L. Wieders, Electroweak corrections to charged-current $e^+e^- \rightarrow 4$ fermion processes: technical details and further results. Nucl. Phys. B **724**, 247–294 (2005). hep-ph/0505042
72. T. Stelzer, W. Long, Automatic generation of tree level helicity amplitudes. Comput. Phys. Commun. **81**, 357–371 (1994). hep-ph/9401258
73. R. Frederix, T. Gehrmann, N. Greiner, Automation of the dipole subtraction method in MadGraph/MadEvent. JHEP **0809**, 122 (2008). arXiv:0808.2128
74. R. Frederix, T. Gehrmann, N. Greiner, Integrated dipoles with MadDipole in the MadGraph framework. JHEP **1006**, 086 (2010). arXiv:1004.2905
75. J. Alwall, P. Demin, S. de Visscher, R. Frederix, M. Herquet et al., MadGraph/MadEvent v4: The New Web Generation. JHEP **0709**, 028 (2007). arXiv:0706.2334
76. https://sherpa.hepforge.org/doc/SHERPA-MC-2.1.0.html
77. http://gosam.hepforge.org/proc/
78. C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer et al., UFO—The Universal FeynRules Output. Comput. Phys. Commun. **183**, 1201–1214 (2012). arXiv:1108.2040
79. N.D. Christensen, C. Duhr, FeynRules—Feynman rules made easy. Comput. Phys. Commun. **180**, 1614–1641 (2009). arXiv:0806.4194
80. A. Alloul, N.D. Christensen, C. Degrande, C. Duhr, B. Fuks, FeynRules 2.0—a complete toolbox for tree-level phenomenology. arXiv:1310.1921
81. A. Semenov, LanHEP - a package for automatic generation of Feynman rules from the Lagrangian. Updated version 3.1, arXiv:1005.1909.
82. K. Chetyrkin, B.A. Kniehl, M. Steinhauser, Decoupling relations to O (alpha-s**3) and their connection to low-energy theorems. Nucl. Phys. B **510**, 61–87 (1998). hep-ph/9708255
83. N. Arkani-Hamed, S. Dimopoulos, G. Dvali, The Hierarchy problem and new dimensions at a millimeter. Phys. Lett. B **429**, 263–272 (1998). hep-ph/9803315
84. I. Antoniadis, N. Arkani-Hamed, S. Dimopoulos, G. Dvali, New dimensions at a millimeter to a Fermi and superstrings at a TeV. Phys. Lett. B **436**, 257–263 (1998). hep-ph/9804398
85. M. Kumar, P. Mathews, V. Ravindran, A. Tripathi, Direct photon pair production at the LHC to order α_s in TeV scale gravity models. Nucl. Phys.B **818**, 28–51 (2009). arXiv:0902.4894
86. R.G. Stuart, Algebraic reduction of one loop Feynman diagrams to scalar integrals. Comput. Phys. Commun. **48**, 367–389 (1988)
87. T. Binoth, J.P. Guillet, G. Heinrich, Algebraic evaluation of rational polynomials in one-loop amplitudes. JHEP **0702**, 013 (2007). hep-ph/0609054