Automated one-loop calculations with GoSAM

Gavin Cullen, Nicolas Greiner, Gudrun Heinrich, Gionata Luisoni, Pierpaolo Mastrolia, Giovanni Ossola, Thomas Reiter, Francesco Tramontano

1 DESY, Zeuthen, Germany
2 School of Physics and Astronomy, The University of Edinburgh, Edinburgh, UK
3 Max-Planck-Institut für Physik, München, Germany
4 Department of Physics, University of Illinois at Urbana-Champaign, Urbana, USA
5 Institute for Particle Physics Phenomenology, University of Durham, Durham, UK
6 Dipartimento di Fisica, Università di Padova, Padova, Italy
7 New York City College of Technology, City University of New York, New York, USA
8 The Graduate School and University Center, City University of New York, New York, USA
9 CERN, Geneva, Switzerland

Abstract We present the program package GoSAM which is designed for the automated calculation of one-loop amplitudes for multi-particle processes in renormalisable quantum field theories. The amplitudes, which are generated in terms of Feynman diagrams, can be reduced using either D-dimensional integrand-level decomposition or tensor reduction. GoSAM can be used to calculate one-loop QCD and/or electroweak corrections to Standard Model processes and offers the flexibility to link model files for theories beyond the Standard Model. A standard interface to programs calculating real radiation is also implemented. We demonstrate the flexibility of the program by presenting examples of processes with up to six external legs attached to the loop.

Contents

1 Introduction ........................................ 2
2 Overview and algorithms ............................ 3
  2.1 Overview ......................................... 3
  2.2 Generation and organisation of the diagrams 3
  2.3 Algebraic processing .............................. 4
  2.4 Code generation ................................. 7
  2.5 Conventions of the amplitudes ................. 8
3 Requirements and installation .................... 10
  3.1 Requirements ................................... 10
  3.2 Download and installation ..................... 10
4 Using GoSAM ....................................... 11
  4.1 Setting up a simple process .................... 11
  4.2 Interfacing the code ............................ 14
  4.3 Using the BLHA interface ....................... 15
  4.4 Using external model files ..................... 15
5 Sample calculations and benchmarks ............... 16
  5.1 pp → W′ + j with SHERPA ..................... 16
  5.2 pp → W± + j, EW corrections ................. 17
  5.3 γγ → γγ .......................................... 18
  5.4 pp → χ10χ10 in the MSSM ..................... 18
  5.5 e+e− → e+e− .................................. 19
  5.6 pp → t ¯tH ....................................... 20
  5.7 gg → t ¯tZ ....................................... 20
  5.8 pp → b ¯b b ¯b + X ................................. 21
  5.9 pp → t ¯tb ¯b + X ................................. 21
  5.10 pp → W+W−b ¯b ................................ 22
  5.11 ud → W+ggg .................................. 22
  5.12 ud → W+ (→ νe e+ b ¯b ) (massive b-quark) .......... 23
6 Conclusions ......................................... 23

Acknowledgements ..................................... 24
Appendix A: Examples included in the release ....... 24
  A.1 How to run the examples ....................... 25
  A.2 e+e− → u ¯u ..................................... 25
  A.3 e+e− → t ¯t ....................................... 25
  A.4 u ¯u → d ¯d ....................................... 25
  A.5 gg → gg ......................................... 26
  A.6 gg → gZ .......................................... 26
1 Introduction

The Standard Model is currently being re-discovered at the LHC, and new exclusion limits on Beyond the Standard Model particles—and on the Higgs boson—are being delivered by the experimental collaborations with an impressive speed. Higher order corrections play an important role in obtaining bounds on the Higgs boson and New Physics. In particular, the exclusion limits for the Higgs boson would look very different if we only had leading order tools at hand. Further, it will be very important to have precise theory predictions to constrain model parameters once a signal of New Physics has been established. Therefore it is of major importance to provide tools for next-to-leading order (NLO) predictions which are largely automated, such that signal and background rates for a multitude of processes can be estimated reliably.

The need for an automation of NLO calculations has been noticed some time ago and lead to public programs like FeynArts [1] and QGraf [2] for diagram generation and FormCalc/LoopTools [3] and GRACE [4] for the automated calculation of NLO corrections, primarily in the electroweak sector. However, the calculation of one-loop amplitudes with more than four external legs were still tedious case-by-case calculations. Only very recently, conceptual and technical advances in multi-leg one-loop calculations allowed the calculation of six-point [5–24] and even seven-point [25, 26] processes at all, and opened the door to the possibility of an automated generation and evaluation of multi-leg one-loop amplitudes. As a consequence, already existing excellent public tools, each containing a collection of hard-coded individual processes, like e.g. MCFM [27, 28], VBFNLO [29, 30], MC@NLO [31, 32], POWHEG-Box [33, 34], POWHEL [35–37], can be flanked by flexible automated tools such that basically any process which may turn out to be important for the comparison of LHC findings to theory can be evaluated at NLO accuracy.

We have recently experienced major advances in the activity of constructing packages for fully automated one-loop calculations, see e.g. [38–43]. The concepts that lead to these advances have been recently reviewed in [44]. Among the most important developments are the integrand-reduction technique [45, 46] and the generalized n-dimensional unitarity [47]. Their main outcome is a numerical reconstruction of a representation of the tensor structure of any one-loop integrand where the multi-particle pole configuration is manifest. As a consequence, decomposing one-loop amplitudes in terms of basic integrals becomes equivalent to reconstructing the polynomial forms of the residues to all multi-particle cuts. Within this algorithm, the integrand of a given scattering amplitude, carrying complete and explicit information on the chosen dimensional-regularisation scheme, is the only input required to accomplish the task of its evaluation. In fact, the integration is substituted by a much simpler operation, namely by polynomial fitting, which requires the sampling of the integrand on the solutions of generalised on-shell conditions.

In this article, we present the program package GoSAM which allows the automated calculation of one-loop amplitudes for multi-particle processes. Amplitudes are expressed in terms of Feynman diagrams, where the integrand is generated analytically using QGRAF [2], FORM [48], spinney [49] and haggies [50]. The individual program tasks are steered via python scripts, while the user only needs to edit an “input card” to specify the details of the process to be calculated, and launch the generation of the source code and its compilation, without having to worry about internal details of the code generation.

The program offers the option to use different reduction techniques: either the unitarity-based integrand reduction as implemented in SAMURAI [40] or traditional tensor reduction as implemented in Golem95C [51, 52] interfaced through tensorial reconstruction at the integrand level [53], or a combination of both. It can be used to calculate one-loop corrections within both QCD and electroweak theory. Beyond the Standard Model theories can be interfaced using FeynRules [54] or LanHEP [55]. The Binnoth-Les Houches-interface [56] to programs providing the real radiation contributions is also included.

The advantage of generating analytic expressions for the integrand of each diagram gives the user the flexibility to organize the computation according to his own efficiency preferences. For instance, the computing algorithm can proceed either diagram-by-diagram or by grouping diagrams that share a common set of denominators (suitable for a unitarity-based reduction), and it can deal with the evaluation of the rational terms either on the same footing as the rest of the amplitude, or through an independent routine which evaluates them analytically. These options and the other features of GoSAM will be discussed in detail in the following.

In Sect. 2, after giving an overview on the diagram generation and on processing gauge-group and Lorentz algebra, we discuss the code generation and the reduction strategies.
The installation requirements are given in Sect. 3, while Sect. 4 describes the usage of GoSAM, containing all the set-up options which can be activated by editing the input card. In Sect. 5 we show results for processes of various complexity. The release of GoSAM is accompanied by the generated code for some example processes, listed in Appendix A.

2 Overview and algorithms

2.1 Overview

GoSAM produces, in a fully automated way, all the code required to perform the calculation of one-loop matrix elements. There are three main steps in the process of constructing the code: the generation of all contributing diagrams within a process directory, the generation of the Fortran code, and finally compiling and linking the generated code. These steps are self-contained in the sense that after each step all the files contained in the process directory could be transferred to a different machine where the next step will be carried out.

In the following sections we focus on the algorithms that are employed for the construction of the code to produce and evaluate matrix elements.

The first step (setting up a process directory), which consists in the generation of some general source files and the generation of the diagrams, is described in Sect. 2.2. The second step (generating the fortran code) is carried out by means of advanced algorithms for algebraic manipulation and code optimization which are presented in Sects. 2.3 and 2.4. The third step (compilation and linking) is not specific to our code generation, therefore will not be described here.

The practical procedures to be followed by the user in generating the code will be given in Sect. 4, which can be considered a short version of the user manual.

2.2 Generation and organisation of the diagrams

For the diagram generation both at tree level and one-loop level we employ the program QGRAF [2]. This program already offers several ways of excluding unwanted diagrams, for example by requesting a certain number of propagators or vertices of a certain type or by specifying topological properties such as the presence of tadpoles or on-shell propagators. Although QGRAF is a very reliable and fast generator, we extend its possibilities by adding another level of analysing and filtering over diagrams by means of Python. This gives several advantages: first of all, the possibilities offered by QGRAF are not always sufficient to distinguish certain classes of diagrams (see examples in Fig. 1); secondly, QGRAF cannot handle the sign for diagrams with Majorana fermions in a reliable way; finally, in order to fully optimize the reduction, we want to classify and group diagrams according to the sets of their propagators.

Within our framework, QGRAF generates three sets of output files: an expression for each diagram to be processed with FORM [48], Python code for drawing all diagrams, and Python code for computing the properties of each diagram. The information about the model for QGRAF is either read from the built-in Standard Model file or is generated from a user defined LanHEP [55] or Universal FeynRules Output (UFO) [54] file.

The Python program automatically performs several operations:

– diagrams whose color factor turns out to be zero are dropped automatically;
– the fermion flow is determined and used to compute an overall sign for each diagram, which is relevant in the presence of Majorana fermions;
– the number of propagators containing the loop momentum, i.e. the loop size of the diagram, the tensor rank and the kinematic invariants of the associated loop integral are computed;
– diagrams with an associated vanishing loop integral (see Fig. 1(a)) are detected and flagged for the diagram selection;
– all propagators and vertices are classified for the diagram selection; diagrams containing massive quark self-energy insertions or closed massless quark loops are specially flagged.

Any one-loop diagram can be written in the form

\[ \mathcal{D} = \int \frac{d^n q}{i \pi^{n/2}} \frac{\mathcal{N}(q)}{\prod_{i=1}^{N} ((q + r_i)^2 - m_i^2 + i\delta)}, \] (1)
where the numerator is a polynomial of tensor rank $r$.

\[ \mathcal{N}(q) = C_0 + C_1^\mu q_\mu + \cdots + C_r^{\mu_1\cdots\mu_r} q_{\mu_1} \cdots q_{\mu_r}, \]  

and the $N \times N$ kinematic matrix is defined as

\[ S_{ij} = (r_i - r_j)^2 - m_i^2 - m_j^2. \]  

All masses can be either real or complex. Important information about the integrals that will appear in the reduction of each one-loop diagram is contained in the tensor rank $r$ of the loop integral and its kinematic matrix $S_{ij}$.

We define a preorder relation on one-loop diagrams, such that $D_1 \preceq D_2$ if their associated matrices $S(D_1)$ and $S(D_2)$ are related by a finite (not necessarily unique) chain of transformations

\[ S(D_2) \xrightarrow{T_1} S' \xrightarrow{T_2} \cdots \xrightarrow{T_n} S(D_1), \]

where each transformation is one of the following:

- the identity,
- the simultaneous permutation of rows and columns,
- the simultaneous deletion of the row and column with the same index, which corresponds to pinching the corresponding propagator in the diagram.

The relation “$\preceq$” can be read as “appears in the reduction of”. Our algorithm groups the one-loop diagrams $D_1, \ldots, D_D$ of a process into subsets $V_1, \ldots, V_n$ such that

- $V_1, \ldots, V_n$ form a partition of $\{D_1, \ldots, D_D\}$ and
- each cell $V_i$ contains a maximum element max $V_i \in V_i$, such that $D \preceq \max V_i$, $\forall D \in V_i$.

The partitioning procedure provides an important gain in efficiency, because while carrying out the tensor reduction for the diagram max $V_i$, all other diagrams in the same cell $V_i$ are reduced with virtually no additional computational cost. The gain in efficiency can be observed when reducing the diagram using the OPP method [45] and its implementations in CutTools [57] and SAMURAI [40], as well as in classical tensor reduction methods as implemented e.g. in Go-lem95C [51, 52], PJFRY [58] and LoopTools [3, 59].

In order to draw the diagrams, we first compute an ordering of the external legs which allows for a planar embedding of the graph. Such ordering can always be found for a tree or a one-loop graph since non-planar graphs only start to appear in diagrams with two or more loops. After the legs have been assigned to the vertices of a regular polygon, we use our own implementation of the algorithms described in [60] for fixing the coordinates of the remaining vertices; the algorithm has been extended to determine an appealing layout also for graphs containing tadpoles. Starting from these coordinates and using the package Axodraw [61], GoSAM generates a \LaTeX file that contains graphical representations of all diagrams.

2.3 Algebraic processing

2.3.1 Color algebra

In the models used by GoSAM, we allow one unbroken gauge group $SU(N_\text{C})$ to be treated implicitly; any additional gauge group, broken or unbroken, needs to be expanded explicitly. Any particle of the model may be charged under the $SU(N_\text{C})$ group in the trivial, (anti-)fundamental or adjoint representation. Other representations are currently not implemented.

For a given process we project each Feynman diagram onto a color basis consisting of strings of generators $T_{A_1}^{(1)} T_{A_2}^{(2)} \cdots T_{A_p}^{(p)}$ and Kronecker deltas $\delta_{ij}$ but no contractions of adjoint indices and no structure constants $f_{ABC}$. Considering, for example, the process

\[ u(1) + \bar{u}(2) \rightarrow Z(3) + g(4) + g(5) \]

GoSAM finds the color basis

\[ |c_1\rangle = q_{1i_1}^{(1)} g_{i_2}^{(2)} A_{A_1}^{A_2} \delta_{ij_1} \delta_{ij_2}, \]

\[ |c_2\rangle = q_{i_1}^{(1)} g_{j_2}^{(2)} g_{i_3}^{(3)} (T^{A_1} T^{A_2}) \delta_{ij_2}, \]

\[ |c_3\rangle = q_{i_1}^{(1)} g_{i_2}^{(2)} A_{A_1}^{A_2} \delta_{ij_1} \delta_{ij_2} \delta_{ij_3}, \]

where $q_{i\bullet}$ and $g_{i\bullet}$ are the color parts of the quark and gluon wave functions respectively. The dimension of this color basis for $N_\text{g}$ external gluons and $N_{q\bar{q}}$ quark-antiquark pairs is given by [62]:

\[ d(N_\text{g}, N_{q\bar{q}}) = \sum_{i=0}^{N_\text{g}} (-1)^i \binom{N_\text{g}}{i} (N_\text{g} + N_{q\bar{q}} - i)!. \]  

It should be noted that the color basis constructed in this way is not a basis in the mathematical sense, as one can find linear relations between the vectors $|c_i\rangle$ once the number of external partons is large enough.

Any Feynman diagram can be reduced to the form

\[ D = \sum_{i=1}^{k} C_i |c_i\rangle \]

for the process specific color basis $|c_1\rangle, \ldots, |c_k\rangle$ by applying the following set of relations:

\[ T_{ij}^{A B} T_{kl}^{C} = T_{R} \left( \delta_{ij} \delta_{kl} - \frac{1}{N_\text{C}} \delta_{ij} \delta_{kl} \right), \]

\[ f_{ABC} = \frac{1}{i T_{R}} (T_{ij}^{A} T_{jk}^{B} T_{ki}^{C} - T_{ij}^{A} T_{jk}^{C} T_{ki}^{B}). \]
The same set of simplifications is used to compute the matrices \( \langle c_i | c_j \rangle \) and \( \langle c_i | T_I \cdot T_J | c_j \rangle \). The former is needed for squaring the matrix element, whereas the latter is used to provide color correlated Born matrix elements which we use for checking the IR poles of the virtual amplitude and also to provide the relevant information for parton showers like POWHEG [33, 34, 63]. For the above example, GoSAM obtains

\[
\langle c_i | c_j \rangle = T_R C_F \begin{pmatrix}
(N_C^2 - 1) & -1 & N_C \\
-1 & (N_C^2 - 1) & N_C \\
N_C & N_C & N_C^2
\end{pmatrix}.
\]

(9)

Similarly, the program computes the matrices \( \langle c_i | T_I \cdot T_J | c_j \rangle \) for all pairs of partons \( I \) and \( J \).

If \( M^{(0)} \) denotes the tree-level matrix element of the process and we have

\[
M^{(0)} = \sum_{j=1}^{k} C_{j}^{(0)} | c_j \rangle,
\]

(10)

then the square of the tree level amplitude can be written as

\[
|M^{(0)}|^2 = \sum_{i,j=1}^{k} (C_{i}^{(0)})^* C_{j}^{(0)} \langle c_i | c_j \rangle.
\]

(11)

For the interference term between leading and next-to-leading order we use a slightly different philosophy. First of all we note that it is sufficient to focus on a single group \( V_\alpha \) as defined in Sect. 2.2,

\[
(M^{(1)})^\dagger M^{(0)} + h.c.
\]

\[
= \sum \int \frac{d^4 q}{i\pi^n n^2} \frac{N_\alpha(q)}{\prod_{l=1}^{n} [(q + r_l)^2 - m_l^2 + i\delta]} + h.c.
\]

(12)

In order to reduce the complexity at the level of the reduction, we perform the contraction with the tree-level already at the integrand level,

\[
N_\alpha(q) = \sum_{i,j=1}^{k} \langle c_i | c_j \rangle (C_{i}^{(0)})^* C_{j}^{(1)}(q),
\]

(13)

where \( C_{j}^{(1)} \) is formed by the sum over the corresponding coefficients of all diagrams \( D \in V_\alpha \).

2.3.2 Lorentz algebra

In this section we discuss the algorithms used by GoSAM to transform the coefficients \( C_{j}^{(0)} \) and \( C_{j}^{(1)}(q) \), as defined in the previous section, such that the result is suitable for efficient numerical evaluation. One of the major goals is to split the \( n \)-dimensional algebra \( (n = 4 - 2\varepsilon) \) into strictly four-dimensional objects and symbols representing the higher-dimensional remainder.

In GoSAM we have implemented the ’t Hooft-Veltman scheme (HV) and dimensional reduction (DRED). In both schemes all external vectors (momenta and polarisation vectors) are kept in four dimensions. Internal vectors, however, are kept in the \( n \)-dimensional vector space. We adopt the conventions used in [49], where \( \hat{k} \) denotes the four-dimensional projection of an in general \( n \)-dimensional vector \( k \). The \( (n-4) \)-dimensional orthogonal projection is denoted as \( \bar{k} \).

\[
q^2 = \hat{q}^2 + \bar{q}^2 = \hat{q}^2 - \mu^2.
\]

(14)

We also introduce suitable projectors by splitting the metric tensor

\[
g^{\mu\nu} = \hat{g}^{\mu\nu} + \bar{g}^{\mu\nu}, \quad \hat{g}^{\mu\nu} \bar{g}_{\mu\nu} = 0,
\]

\[\hat{g}_{\mu}^{\mu} = 4, \quad \bar{g}_{\mu}^{\mu} = n - 4.
\]

(15)

In the following, we describe the ’t Hooft algebra in detail. For DRED, the only differences are that the numerator algebra is performed in four dimensions for both external and internal vectors (i.e. \( q \equiv \hat{q} \)) and that in the very end all appearances of \( q^2 \) are replaced by \( \hat{q}^2 - \mu^2 \).

Wave functions and propagators GoSAM contains a library of representations of wave functions and propagators up to spin two. The exact form of the interaction vertices is taken from the model files.

The representation of all wave functions with non-trivial spin is based on massless spinors. Each massive external vector \( p_i \) is replaced by its light-cone projection \( l_i \) with respect to a lightlike reference vector \( k \),

\[
p_i^{\mu} = l_i^{\mu} + \frac{p_i^{2}}{2p_i \cdot k} k^{\mu}.
\]

(16)

For spin \( 1/2 \) particles we use the assignment of wave functions as shown in Table 1; here, we quote the definition of the massive spinors from [49] assuming the splitting of (16):

\[
|p^\pm\rangle = |l\rangle \pm \frac{\sqrt{p^2}}{|k\rangle} |k\rangle, \quad |p^\mp\rangle = |l\rangle \pm \frac{\sqrt{p^2}}{|k\rangle} |k\rangle.
\]

(17a)

\[
|p^\pm\rangle = |l\rangle \pm \frac{\sqrt{p^2}}{|k\rangle} |k\rangle, \quad |p^\mp\rangle = |l\rangle \pm \frac{\sqrt{p^2}}{|k\rangle} |k\rangle.
\]

(17b)

\[3\] Processes with particles of spin \( 3/2 \) and spin 2 have not been tested extensively. Furthermore, these processes can lead to integrals where the rank is higher than the loop size, which at the moment are neither implemented in SAMURAI nor in Golem95C.
In order to preserve the condition that for any loop integral the tensor rank does not exceed the number of loop propagators we fix all gauge boson propagators to be in Feynman gauge. Their wave functions are constructed as \[64\]

\[
\epsilon_\mu(p, +1) = \frac{\langle q|\gamma_\mu|p^b\rangle}{\sqrt[2]{2(p^b q)}}, \quad \epsilon_\mu(p, -1) = \frac{\langle q|\gamma_\mu|p^b\rangle}{\sqrt[2]{2(p^b q)}},
\]

where \( p^b = p \) in the massless case and \( p^b = l \) according to \(16\) in the massive case. In the latter case the third polarisation is defined as

\[
\epsilon_\mu(p, 0) = \frac{1}{\sqrt{2}}(2p^b_\mu - p_\mu).
\]

The wave functions and propagators for spin \( \frac{3}{2} \) and spin 2 particles correspond to those in \[65\].

**Simplifications** Once all wave functions and propagators have been substituted by the above definitions and all vertices have been replaced by their corresponding expressions from the model file then all vector-like quantities and all metric tensors are split into their four-dimensional and their orthogonal part. As we use the \('t\) Hooft algebra, \( \gamma_5 \) is defined as a purely four-dimensional object, \( \gamma_5 = i\epsilon_{\mu\nu\rho\sigma}\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma \). By applying the usual anti-commutation relations for Dirac matrices we may always separate the four-dimensional and \((n - 4)\)-dimensional parts of Dirac traces, as we can use the fact that \[49, 62\]

\[
\text{tr}(1) \cdot \text{tr}(\hat{\gamma}_\mu_1 \cdots \hat{\gamma}_\mu_n | \hat{\gamma}_\mu_{n+1} \cdots \hat{\gamma}_\mu_{n+p}) = \text{tr}(\hat{\gamma}_\mu_1 \cdots \hat{\gamma}_\mu_n) \cdot \text{tr}(\hat{\gamma}_\mu_{n+1} \cdots \hat{\gamma}_\mu_{n+p}).
\]

The same logic applies to open spinor lines such as \[49\]

\[
\text{tr}(1) \cdot \langle k_1|\hat{\gamma}_\mu_1 \cdots \hat{\gamma}_\mu_n | \hat{\gamma}_\mu_{n+1} \cdots \hat{\gamma}_\mu_{n+p} | k_2\rangle = \langle k_1|\hat{\gamma}_\mu_1 \cdots \hat{\gamma}_\mu_n | k_2\rangle \cdot \text{tr}(\hat{\gamma}_\mu_{n+1} \cdots \hat{\gamma}_\mu_{n+p}).
\]

While the \((n - 4)\)-dimensional traces are reduced completely to products of \((n - 4)\)-dimensional metric tensors \( g^{\mu_1} \), the four-dimensional part is treated such that the number of terms in the resulting expression is kept as small as possible. Any spinor line or trace is broken up at any position where a light-like vector appears. Furthermore, Chisholm identities are used to resolve Lorentz contractions between both Dirac traces and open spinor lines. If any traces remain we use the built-in trace algorithm of \textsc{form} \[48\].

In the final result we can always avoid the explicit appearance of Levi-Civita tensors, noticing that any such tensor is contracted with at least one light-like vector\(^4\) \( \hat{k}_\mu \), and we can replace

\[
\hat{k}_\mu \epsilon_{\mu\nu\rho\sigma} = - \frac{i}{4} \left[ \langle k|\hat{\gamma}_\nu \hat{\gamma}_\rho \hat{\gamma}_\sigma | k \rangle - \langle k|\hat{\gamma}_\nu \hat{\gamma}_\rho \hat{\sigma} | k \rangle \right).
\]

Hence, the kinematic part of the numerator, at the end of our simplification algorithm, is expressed entirely in terms of:

- spinor products of the form \( \langle k_1|k_j \rangle, [k_i k_j] \) or \( [k_i|\hat{\gamma}_\mu|k_j] \cdot \hat{\gamma}_\mu \),
- dot products \( \hat{k}_i \cdot \hat{k}_j \) or \( \hat{k}_i \cdot \hat{\gamma} \),
- constants of the Lagrangian such as masses, widths and coupling constants,
- the symbols \( \mu^2 = \hat{q}^2 - q^2 \) and \( \epsilon = \frac{n - 4}{2}4\)

**Treatment of \( R_2 \) rational terms** In our representation for the numerator of one-loop diagrams, terms containing the symbols \( \mu^2 \) or \( \epsilon \) can lead to a so-called \( R_2 \) term \[66\], which contributes to the rational part of the amplitude. In general, there are two ways of splitting the numerator function:

\[
\mathcal{N}(\hat{q}, \mu^2, \epsilon) = \mathcal{N}_0(\hat{q}, \mu^2) + \epsilon \mathcal{N}_1(\hat{q}, \mu^2)
\]

\[
+ \epsilon^2 \mathcal{N}_2(\hat{q}, \mu^2)
\]

or, alternatively,

\[
\mathcal{N}(\hat{q}, \mu^2, \epsilon) = \hat{\mathcal{N}}(\hat{q}) + \hat{\mathcal{N}}(\hat{q}, \mu^2, \epsilon).
\]

It should be noted that in \(23a\) the terms \( \mathcal{N}_1 \) and \( \mathcal{N}_2 \) do not arise in \textsc{dred}, where only terms containing \( \mu^2 \) contribute to \( \mathcal{R}_2 \). Instead of relying on the construction of \( R_2 \)

\(^4\)Any external massive vector at this point has been replaced by a pair of light-like ones. Contractions between two Levi-Civita symbols can be resolved to products of metric tensors.
from specialized Feynman rules [67–70], we generate the $R_2$ part along with all other contributions without the need to separate the different parts. For efficiency reasons, however, we provide an implicit and an explicit construction of the $R_2$ terms.

The implicit construction uses the splitting of (23a) and treats all three numerator functions $N_i$ on equal grounds. Each of the three terms is reduced separately in a numerical reduction and the Laurent series of the three results are added up taking into account the powers of $\epsilon$.

The explicit construction of $R_2$ is based on the assumption that each term in $\tilde{N}$ in (23b) contains at least one power of $\mu^2$ or $\epsilon$. The expressions for those integrals are relatively simple and known explicitly. Hence, the part of the amplitude which originates from $\tilde{N}$ is computed analytically whereas the purely four-dimensional part $\hat{N}$ is passed to the numerical reduction.

2.4 Code generation

2.4.1 Abbreviation system

To prepare the numerator functions of the one-loop diagrams for their numerical evaluation, we separate the symbol $\mu^2$ and dot products involving the momentum $\hat{q}$ from all other factors. All subexpressions which do not depend on either $\hat{q}$ or $\mu^2$ are substituted by abbreviation symbols, which are evaluated only once per phase space point. Each of the two parts is then processed using haggies [50], which generates optimized Fortran code for their numerical evaluation. For each diagram we generate an interface to SAMURAI [40], Golem95C [52] and/or PJFRY [58]. The two latter codes are interfaced using tensorial reconstruction at the integrand level [53].

2.4.2 Reduction strategies

In the implementation of GoSAM, great emphasis has been put on maintaining flexibility with respect to the reduction algorithm that the user decides to use. On the one hand, this is important because the best choice of the reduction method in terms of speed and numerical stability can strongly depend on the specific process. On the other hand, we tried to keep the code flexible to allow further extensions to new reduction libraries, such that GoSAM can be used as a laboratory for interfacing future methods with a realistic environment.

Our standard choice for the reduction is SAMURAI, which provides a very fast and stable reduction in a large part of the phase space. Furthermore, SAMURAI reports to the client code if the quality of the reconstruction of the numerator suffices the numerical requirements (for details we refer to [40]). In GoSAM we use this information to trigger an alternative reduction with either Golem95C [52] or PJFRY [58] whenever these reconstruction tests fail, as shown in Fig. 2. The reduction algorithms implemented in these libraries extend to phase space regions of small Gram determinants and therefore cover most cases in which on-shell methods cannot operate sufficiently well. This combination of on-shell techniques and traditional tensor reduction is achieved using tensorial reconstruction at the integrand level [53], which also provides the possibility of running on-shell methods with a reconstructed numerator. In addition to solving the problem of numerical instabilities, in some cases this option can reduce the computational cost of the reduction. Since the reconstructed numerator is typically of a form where kinematics and loop momentum dependence are already separated, the use of a reconstructed numerator tends to be faster than the original procedure, in particular in cases with a large number of legs and low rank.

The flowchart in Fig. 2 summarizes all possible reduction strategies which are currently implemented. The strategy in use is selected by assigning the variable reduction_interoperation in the generated Fortran code. The availability of the branches is determined during code generation by activating (at least one of) the extensions (samurai, golem95, pjfry) in the input card.

**Fig. 2** Reduction strategies currently implemented in GoSAM: the reduction algorithm is chosen by setting the variable reduction_interoperation in the generated Fortran code and can be modified at run time. 0: SAMURAI only; 1: Golem95C only; 2: SAMURAI with rescue option (Golem95C); 3: SAMURAI with numerator from tensorial reconstruction; 4: same as 3 but with rescue option (Golem95C). 11, 12 and 14 are the same as 1, 2, 3 (respectively) with the difference that PJFRY is used instead of Golem95C.
Switching between active branches is possible at run time.

In detail, the possible choices for the variable reduction_interopera_tion are the following:

0 the numerators of the one-loop diagrams are reduced by SAMURAI, no rescue system is used in case the reconstruction test fails;
1 the tensor coefficients of the numerators are reconstructed using the tensorial reconstruction at the integrand level, the numerator is expressed in terms of tensor integral form factors which are evaluated using Golem95C;
2 the numerators are reduced by SAMURAI; whenever the reconstruction test fails, numerators are reduced using the option 1 as a backup method;
3 tensorial reconstruction is used to compute the tensor coefficients; SAMURAI is employed for the reduction of the reconstructed numerator, no rescue system is used;
4 as in option 3, SAMURAI is used to reduce the reconstructed numerator, Golem95C is used as backup option;
11 same as 1 but PJFRY is used instead of Golem95C;
12 same as 2 but PJFRY is used instead of Golem95C;
14 same as 4 but PJFRY is used instead of Golem95C.

It is difficult to make a statement about the “optimal” reduction method because this depends on the process under consideration. For multi-leg processes, e.g. $b\bar{b}b\bar{b}$ production, we found that SAMURAI is clearly superior to tensor reduction in what concerns timings and size of the code. Concerning points which need a special treatment, we did not make extensive studies using traditional tensor reduction only, but one can certainly say that the combination of SAMURAI and tensorial reconstruction seems to be optimal in what concerns the avoidance of numerical instabilities due to inverse Gram determinants.

2.5 Conventions of the amplitudes

In this section we briefly discuss the conventions chosen for the results returned by GoSAM. Depending on the actual setup for a given process, in particular if an imported model file is used, conventions may be slightly different. Here we restrict the discussion to the case where the user wants to compute QCD corrections to a process and in the setup files he has put $g_s = 1$. In this case, the tree-level matrix element squared can be written as

$$|M|_{\text{tree}}^2 = A_0^\dagger A_0 = (g_s)^{2b} \cdot a_0. \quad (24)$$

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

$$|M|_{\text{1-loop}}^2 = A_1^\dagger A_0 + A_0^\dagger A_1 = 2 \cdot \Re(A_0^\dagger A_1)$$

$$= |M|_{\text{bare}}^2 + |M|_{\text{ct,bare}}^2 + |M|_{\text{ct,as}}^2 + |M|_{\text{wt,f}}^2 + |M|_{\text{wt,Q}}^2$$

$$= \frac{\alpha_s(\mu)}{2\pi} \left[ \frac{4\pi}{\Gamma(1-\varepsilon)} \cdot (g_s)^{2b} \cdot \left[ c_0 + \frac{c-1}{\varepsilon} + \frac{c-2}{\varepsilon^2} + \mathcal{O}(\varepsilon) \right] \right]. \quad (25)$$

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.

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 $A_1^\dagger A_1$ where a factor

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

has been pulled out.

After all UV-renormalisation contributions have been taken into account correctly, only IR-singularities remain, which can be computed using the routine ir_subtraction. This routine returns a vector of length two, containing the coefficients of the single and the double pole, which should be equal to $(c_{-1}, c_{-2})$ and therefore can be used as a check of the result.

**Ultraviolet renormalisation in QCD** For UV-renormalisation we use the MS scheme for the gluon and all massless quarks, whereas a subtraction at zero momentum is chosen for massive quarks [71]. Currently, counterterms are only provided for QCD corrections. In the case of electroweak corrections only unrenormalised results can be produced automatically.

For computations involving loop propagators for massive fermions, we introduced the automatic generation of a mass counter term needed for the on-shell renormalisation of the massive particle. Here, we exploit the fact that such a counter term is strictly related to the massive fermion self energy bubble diagrams (see Fig. 3). As described in Sect. 2.2, the program GoSAM analyzes all generated diagrams. In that step also self-energy insertions of massive

---

Fig. 3 Feynman diagram of a massive quark self energy in QCD. For this type of diagram GoSAM automatically generates UV-counterterms.
quarks are detected, where we make the replacement
\[
\frac{(q + f + m) \cdot g^{\mu \nu}}{[(q + r)^2 - m^2]q^2} \rightarrow \frac{(q + f + m) \cdot g^{\mu \nu}}{[(q + r)^2 - m^2]q^2} + \frac{m}{4} \left[ 6q \cdot r + 3(q^2 - m^2) \right] m^2 + \frac{3(4 + 1_{1HV}) \mu^2}{r^2 - 3m^2} \right] \frac{g^{\mu \nu}}{[(q + r)^2 - m^2]q^2}.
\]

The symbol $1_{1HV}$ is one in the 't Hooft Veltman scheme and zero in DRED.

Performing the integral, contracting the expression with the QCD vertices at both sides and multiplying the missing factor of $(2\pi)^{-1}$ we retrieve the expression for the mass counter-term,
\[
\frac{\delta m}{m} = \frac{\alpha_s}{2\pi} \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)} \frac{C_F}{2} \left( \frac{\mu^2}{m^2} \right)^\epsilon \left[ \frac{3}{\epsilon} + 5 - 1_{1HV} \right].
\]

Furthermore, the renormalisation of $\alpha_s$ leads to a term of the form
\[
|M|_{\text{ct}, \alpha_s}^2 = b \cdot \frac{\alpha_s}{2\pi} \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)} |M|_{\text{free}}^2 \cdot \left[ -\frac{\beta_0}{\epsilon} + \frac{2T_R}{3\epsilon} \sum_{q=N_f+1}^{N_f+N_{hQ}} \left( \frac{\mu^2}{m_q^2} \right)^\epsilon + \frac{C_A}{6} (1 - 1_{1HV}) \right],
\]

with $\beta_0 = (11C_A - 4T_R N_f)/6$, $N_f$ being the number of light quark flavours, $N_{hQ}$ the number of heavy flavours, and $b$ is the power of the coupling in the Born amplitude as defined in (24). The last term of (28) provides the finite renormalisation needed to compensate the scheme dependence of $\alpha_s$,
\[
\alpha_s^{\text{DR}} = \alpha_s^{\text{MS}} \left( 1 + \frac{C_A}{6} \frac{\alpha_s^{\text{MS}}}{2\pi} \right).
\]

A further contribution consists of the wave-function renormalisation of massive external quark lines. If we denote the set of external massive quark lines by $Q_h = \{Q_1(m_1), \ldots, Q_p(m_p)\}$ we obtain
\[
|M|_{\text{wf}, Q}^2 = -\frac{\alpha_s}{2\pi} \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)} \frac{C_F}{2} \left[ \frac{3}{\epsilon} + 5 - 1_{1HV} \right] \cdot |M|_{\text{free}}^2.
\]

Finally, also the wave function of the gluon receives a contribution from the presence of heavy quarks in closed fermion loops. If $N_g$ is the number of external gluon lines, this contribution can be written as
\[
|M|_{\text{wf}, g}^2 = -\frac{\alpha_s}{2\pi} \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)} \frac{2T_R}{3\epsilon} \left[ \frac{N_f + N_{hQ}}{\mu^2} \right]^\epsilon \cdot |M|_{\text{free}}^2.
\]

At the level of the generated Fortran code the presence of these contributions can be controlled by a set of variables defined in the module config.f90. The variable renormalisation can be set to 0, 1, or 2. If renormalisation=0, none of the counterterms are present. If renormalisation=2 only $|M|_{\text{ct}, \delta Q}^2$ is included, which is the counterterm stemming from all terms of the type of (27) contributing to the amplitude.

In the case where renormalisation=1 a more fine-grained control over the counterterms is possible.

The default settings for renormalisation=1 are true for all the renorm options listed above.

**Finite renormalisation of $\gamma_5$ in QCD** In the 't Hooft Veltman scheme, a finite renormalisation term for $\gamma_5$ is required beyond tree level. The relevant terms are generated only if $fr5$ is added in the input card to the list of extensions before code generation. Currently, the automatic generation of this finite contribution is not performed if model files different from the built-in model files are used. In agreement with [72] and [73] we replace the axial component at each vertex,
\[
\gamma^\mu \gamma_5 \rightarrow \frac{1}{2} Z_{\text{axial}} (\gamma^\mu \gamma_5 - \gamma_5 \gamma^\mu),
\]

with
\[
Z_{\text{axial}} = 1 - 2 \frac{\alpha_s}{2\pi} C_F \cdot 1_{1HV}.
\]

Once it is generated, this contribution can be switched on and off at run-time through the variable renorm_gamma5, which is defined in the module config.f90.
Conversion between the schemes  In GoSAM we have implemented two different schemes, the ’t Hooft Veltman scheme and dimensional reduction. By default, the former is used, while the latter can be activated by adding the extension \texttt{dred}. If a QCD computation has been done in dimensional reduction the result can be converted back to the ’t Hooft Veltman scheme by adding a contribution for each external massless parton,

\begin{equation}
\langle M^{\text{CDR}} \rangle^2_{1\text{-loop}} = \langle M^{\text{DR}} \rangle^2_{1\text{-loop}} - \frac{\alpha_s}{2\pi}\langle M^{\text{DR}} \rangle^2_{\text{tree}} \sum_{I=1}^{N_{\text{ext}}} \gamma_I^{\text{DR}},
\end{equation}

with \( \gamma_q^{\text{DR}} = \gamma_{\bar{q}}^{\text{DR}} = C_F / 2 \) and \( \gamma_g^{\text{DR}} = C_A / 6 \). This conversion can be switched on by setting \texttt{convert\_to\_cdr} to \texttt{true} in the module \texttt{config.f90}. At one-loop level, the ’t Hooft Veltman scheme and conventional dimensional regularisation (CDR) are equivalent in the sense that \( \gamma_I^{\text{tHV}} = 0 \) for all partons.

3 Requirements and installation

3.1 Requirements

The program GoSAM is designed to run in any modern Linux/Unix environment; we expect that Python (\( \geq 2.6 \)), Java (\( \geq 1.5 \)) and Make are installed on the system. Furthermore, a Fortran 95 compiler is required in order to compile the generated code. Some Fortran 2003 features are used if one wants to make use of the Les Houches interface [56]. We have tried all examples using \texttt{gfortran} versions 4.1 and 4.5.

On top of a standard Linux environment, the programs \texttt{FORM} [48], version \( \geq 3.3 \) (newer than Aug. 11, 2010) and \texttt{QGRAF} [2] need to be installed on the system. Whereas \texttt{spinney} [49] and \texttt{haggies} [50] are part of GoSAM and are not required to be installed separately, at least one of the libraries \texttt{SAMURAI} [40] and \texttt{Golem95C} [52] needs to be present at compile time of the generated code. Optionally, \texttt{PJFRY} [58] can be used on top of \texttt{Golem95C}.

3.2 Download and installation

\texttt{QGRAF}  The program can be downloaded as Fortran source code from

\url{http://cfif.ist.utl.pt/~paulo/qgraf.html}.

After unpacking the tar-ball, a single \texttt{Fortran77} file needs to be compiled.

\texttt{FORM}  The program is available at

\url{http://www.nikhef.nl/~form/}

both as a compiled binary for many platforms and as a tar-ball. The build process, if built from the source files, is controlled by Autotools.

\texttt{SAMURAI and Golem95C}  These libraries are available as tar-balls and from subversion repositories at

\url{http://projects.hepforge.org/samurai/}

and

\url{http://projects.hepforge.org/golem/95/}

respectively. For the user’s convenience we have prepared a package containing \texttt{SAMURAI} and \texttt{Golem95C} together with the integral libraries \texttt{OneLoop} [74], \texttt{QCD-Loop} [75] and \texttt{FF} [59]. The package \texttt{gosam-contrib-1.0.tar.gz} containing all these libraries is available for download from:

\url{http://projects.hepforge.org/gosam/}

GoSAM  The user can download the code either as a tar-ball or from the subversion repository at

\url{http://projects.hepforge.org/gosam/}.

The build process and installation of GoSAM is controlled by Python Distutils, while the build process for the libraries \texttt{SAMURAI} and \texttt{Golem95C} is controlled by Autotools.

Therefore the installation proceeds in two steps:

1. For all components which use Autotools, the following sequence of commands installs them under the user defined directory \texttt{MYPATH}.

\begin{verbatim}
./configure --prefix=MYPATH
make FC=gfortran F77=gfortran
make install # or sudo make install
\end{verbatim}

If the \texttt{configure} script is not present, the user needs to run \texttt{sh ./autogen.sh} first.

2. For GoSAM which is built using Distutils, the user needs to run

\begin{verbatim}
python setup.py install \
--prefix MYPATH
\end{verbatim}

If \texttt{MYPATH} is different from the system default (e.g. \texttt{/usr/bin}), the environment variables \texttt{PATH}, \texttt{LD_LIBRARY_PATH} and \texttt{PYTHONPATH} might have to be set accordingly. For more details we direct the user to the GoSAM reference manual and to the documentation of the beforementioned programs.
4 Using GoSAM

4.1 Setting up a simple process

GoSAM is a very flexible program and comes with a wide range of configuration options. Not all of these options are relevant for simple processes and often the user can leave most of the settings at their default values. In order to generate the code for a process, one needs to prepare an input file, which will be called process card in the following, which contains

- process specific information, such as a list of initial and final state particles, their helicities (optional) and the order of the coupling constants;
- scheme specific information and approximations, such as the regularisation and renormalisation schemes, the underlying model, masses and widths which are set to zero, the selection of subsets of diagrams; the latter might be process dependent;
- system specific information, such as paths to programs and libraries or compiler options;
- optional information for optimisations which control the code generation.

In the following we explain how to set up the required files for the process \( q\bar{q} \rightarrow gZ^0 \rightarrow g e^-e^+ \). The example computes the QCD corrections for the \( u\bar{u} \) initial state, where \( m_e = 0 \) and \( N_f = 5 \) massless quarks are assumed. For our example, we follow an approach where we keep the different types of information in separate files—process.rc, scheme.rc and system.rc—and use GoSAM to produce a process card for this process based on these files. This is not required—one could also produce and edit the process card directly—it is however more convenient to store system specific information into a separate, re-usable file, and it makes the code generation more transparent.

**Process specific information** The following listing contains the information which is specific to the process. The syntax of process cards requires that no blank character is left between the equals sign and the property name. Commentary can be added to any line, marked by the ‘#’ character. Line continuation is achieved using a backslash at the end of a line.\(^5\)

**Listing 1** File ‘process.rc’

```plaintext
1 process_path=qggz
2 in=u,u-
3 out=g,e-,e+
4 helicities=+-+-,+-+-+,+-++,+--+
5 order=QCD,1,3
```

The first line defines the (relative) path to the directory where the process files will be generated. GoSAM expects that this directory has already been created. Lines 2 and 3 define the initial and final state of the process in terms of field names, which are defined in the model file. Besides the field names one can also use PDG codes [76, 77] instead. Hence, the following lines would be equivalent to lines 2 and 3 in Listing 1:

```
2 in=2,-2
3 out=21,11,-11
```

Line 4 describes the helicity amplitudes which should be generated. If no helicities are specified, the program defaults to the generation of all possible helicity configurations, some of which may turn out to be zero. The different helicity amplitudes are separated by commas; within one helicity amplitude there is one character (usually ‘+’, ‘-’ and ‘0’) per external particle from the left to the right. In the above example for the reaction

\[
u(k_1, \lambda_1)\bar{u}(k_2, \lambda_2) \rightarrow g(k_3, \lambda_3)e^-(k_4, \lambda_4)e^+(k_5, \lambda_5)
\]

we have the following assignments:

| Helicity | \( \lambda_1 \) | \( \lambda_2 \) | \( \lambda_3 \) | \( \lambda_4 \) | \( \lambda_5 \) |
|----------|------------------|------------------|------------------|------------------|------------------|
| 0        | +                | -                | +                | -                | +                |
| 1        | +                | -                | +                | -                | +                |
| 2        | +                | +                | -                | +                | -                |
| 3        | -                | +                | -                | +                | -                |

With the above value for helicities we generate all non-vanishing helicities for the partons but keep the lepton helicities fixed. In more complicated examples this way of listing all helicities explicitly can be very tedious. Therefore, we introduced the option to generate sets of helicities using square brackets. For example, if the gluon helicity is replaced by [+-], the bracket is expanded automatically to take the values +, -.

4 **helicities=+-[+-]+, -+[+-]+**

A further syntactical reduction can be achieved for the quarks. The current expansion of a square bracket and its opposite value can be assigned to a pair of variables as in \( [x\gamma++] \). If the bracket expands to ‘+’ then \( x \) is assigned ‘+’ and \( \gamma \) is assigned the opposite sign, i.e. ‘-’. If the bracket expands to ‘-’ the assignments are \( x=- \) and \( \gamma=+ \). Hence, the helicity states of a massless quark anti-quark pair are generated by \([qQ=++]Q\) and the selection of helicities in our example can be abbreviated to

4 **helicities=[qQ=++]Q[++]**

which is equivalent to the version of this line in Listing 1.

\(^5\)The line numbers are just for reference and should not be included in the actual files.
Finally, the order (power) of the coupling constants has to be specified. Line 5 contains a keyword for the type of coupling (QCD or QED), the order of this coupling constant in the unsquared tree level amplitude (in our example: 1) and the order of the coupling constant in the unsquared one-loop amplitude (in our example: 3). One can also use GoSAM to generate the tree level only, by giving only the power of the tree level amplitude:

```
5 order=QCD,1
```

Conversely, GoSAM will generate the virtual amplitude squared for processes where no tree level is present if the tree level order is replaced by the keyword NONE.

```
5 order=QCD,NONE,3
```

Up to now, the file would generate all 8 tree level and 180 one-loop diagrams contributing to the process \( \bar{u}u \rightarrow g\epsilon^{-}\epsilon^{+} \), regardless of the intermediate states. Nevertheless, what we intended to generate were only those diagrams where the electron pair comes from the decay of a \( Z \rightarrow \epsilon^{-}\epsilon^{+} \). GoSAM offers two ways of achieving this diagram selection, either by passing a condition to QGRAF or by applying a filter written in Python. The first option would be specified by the option qgraf.verbatim, which copies the argument of the option to the QGRAF input file in verbatim. The following filter demands the appearance of exactly one \( Z \)-propagator, leaving us with 2 tree-level and 45 one-loop diagrams:

```
6 qgraf.verbatim= true=iprop[Z,1,1];
```

The alternative solution is the application of a Python filter using the options filter.lo for tree level and filter.nlo for one-loop diagrams. The current example requires the two lines

```
6 filter.lo= IPROP([Z])==1
7 filter.nlo= IPROP([Z])==1
```

### Scheme specific information
For our example we put all scheme specific definitions in the file scheme.rc. It contains the choice of a suitable regularisation scheme and fixes what types of UV counterterms are included in the final result.

Listing 2  File 'scheme.rc'
```
1 extensions=dred
2 qgraf.options=onshell
3 zero=mU,mD,mC,mS,mB,me,\wT
4 one=gs
```

In Listing 2, line 1 selects dimensional reduction as a regularisation scheme. If \( \text{dred} \) is not specified in the list of extensions, GoSAM works in the ’t Hooft Veltman scheme by default. Line 2 removes all on-shell bubbles on external legs. This is, on the one hand, required to be consistent with the renormalisation scheme. On the other hand, those diagrams would lead to zero denominators at the algebraic level. In line 3 all light quark masses, the mass of the electron and the width of the top quark are set to zero. Further, as a convention rather than a scheme, the strong coupling \( g_{s} \) is set to one in line 4, which means that \( g_{s} \) will not occur in the algebraic expressions, assuming that the user will multiply the final result by his desired value for the strong coupling. If the option one=gs is not used, the default value contained in the file common/model.f90 will be used. This default value of course can be changed by the user.

### System specific information
In order to adapt the code generation to the system environment, GoSAM needs to find a way of determining all relevant paths and options for the programs and libraries used during generation, compilation and linking of the code. Those settings are fixed in the file system.rc in our example.

Listing 3  File 'system.rc'
```
1 system.extensions=samurai,golem95
2 samurai.fcflags= \n3 -I${PREFIX}/include/samurai
4 samurai.ldflags= \n5 -L${PREFIX}/lib -lsamurai
6 samurai.version=2.1.1
7 golem95.fcflags= \n8 -I${PREFIX}/include/golem95
9 golem95.ldflags= \n10 -L${PREFIX}/lib -lgolem95
11 form.bin=${PREFIX}/bin/tform
12 qgraf.bin=${PREFIX}/bin/qgraf
13 fc.bin=gfortran
```

Generating the code  After having prepared the input files correctly we need to collect the information distributed over the three files process.rc, scheme.rc and system.rc in one input file, which we will call gosam.in here. The corresponding command is:
```
gosam.py --template gosam.in \ 
--merge process.rc \ 
--merge scheme.rc --merge system.rc
```

---

\(^{6}\) In this example we assume that the user has defined an environment variable PREFIX.
The generated file can be processed with `gosam.py` directly but requires the process directory to be present.

```
mkdir qqgz
gosam.py gosam.in
cd qqgz
```

All further steps are controlled by the generated make files; in order to generate and compile all files relevant for the matrix element one needs to invoke

```
make compile
```

The generated code can be tested with the program `matrix/test.f90`. The following sequence of commands will compile and run the example program.

```
cd matrix
make test.exe
./test.exe
```

The last lines of the program output should look as follows:

```
# LO: 0.3450350717601E-06
# NLO, finite part -10.77604823456547
# NLO, single pole -19.98478948141949
# NLO, double pole -5.666666665861926
# IR, single pole -19.98478948439310
# IR, double pole -5.666666666666666
```

The printed numbers are, in this order, $a_0$, $c_0/a_0$, $c_{-1}/a_0$, $c_{-2}/a_0$ and the pole parts calculated from the infrared insertion operator [78, 79].

One can generate a visual representation of all generated diagrams using the command

```
make doc
```

which generates the file `doc/process.ps` using a Python implementation of the algorithm described in [60] and the LaTeX package AXODRAW [61].

### 4.1.1 Further options

GoSam provides a range of options which influence the code generation, the compilation and the numerical evaluation of the amplitude. Giving an exhaustive list of all options would be far beyond the scope of this article and the interested user is referred to the reference manual. Nonetheless, we would like to point out some of GoSam’s capabilities by presenting the corresponding options.

---

**Generating the $R_2$ term** When setting up a process the user can specify if and how the $R_2$ term of the amplitude should be generated by setting the variable `r2` in the setup file.

```
r2=explicit
```

Possible options for `r2` are `implicit`, which is the default, `explicit`, `off` and `only`. The keyword `implicit` means that the $R_2$ term is generated along with the four-dimensional numerator as a function in terms of $\hat{q}$, $\mu^2$ and $\epsilon$ and is reduced at runtime by sampling different values for $\mu^2$. This is the slowest but also the most general option. Using the keyword `explicit` carries out the reduction of terms containing $\mu^2$ or $\epsilon$ during code generation (see Appendix B). The keyword `off` puts the $R_2$ term to zero which is useful if the user wants to provide his own calculation for these terms. Conversely, using `r2=only` discards everything but the $R_2$ term (reducing it as in the case `explicit`) and puts GoSam in the position of providing $R_2$ terms for external codes which work entirely in four dimensions.

**Diagram selection** GoSam offers a two-fold way of selecting and discarding diagrams. One can either influence the way QGRAF generates diagrams or apply filters to the diagrams after they have been generated by QGRAF or combine the two methods. Let us assume that in the above example we want to remove the third generation of quarks completely. Hence, all closed quark loops would be massless and therefore the second generation is just an exact copy of the first one. We can therefore restrict the generation of closed quark loops to up and down quarks. GoSam has a filter precisely for this purpose, which takes the field names of the flavours to be generated as arguments.

```
filter.nlo=NFGEN(U,D)
```

This filter can be combined with the already existing filter selecting only diagrams containing a $Z$-propagator using the `AND` function:

```
filter.nlo=AND( NFGEN(U,D), \n                IPROP([[Z]]) == 1 )
```

A further feature of the code generated by GoSam is the possibility of selecting diagrams at runtime. For example, we would like to distinguish at runtime three different gauge invariant sets of diagrams at one-loop level:

1. diagrams with a closed quark loop where the $Z$ is attached to the loop;
2. diagrams with a closed quark loop where the $Z$ is emitted from the external quark line;
3. diagrams without a closed quark loop.

---

7The actual numbers depend on the random number generator of the system because the phase space point is generated randomly; however, the pole parts should agree between the matrix element and the infrared insertion operator given that the matrix element is fully renormalised.
In order to provide the code for a diagram selection at run-time one simply replaces the above filter by a list of filters as follows

```
filter.nlo=[
  AND( NFGEN(U,D), IPROP([Z]) == 1, \ 
  NF, LOOPVERTICES([Z],_,_) == 1), \ 
  AND( NFGEN(U,D), IPROP([Z]) == 1, \ 
  NF, LOOPVERTICES([Z],_,_) == 0), \ 
  AND( NFGEN(U,D), IPROP([Z]) == 1, \ 
  NOT(NF)) ]
```

The two new filters in use are NF which selects closed quark loops only and LOOPVERTICES which counts the number of vertices attached to the loop with the given sets of fields running through the vertex (where _ replaces any field). In the Fortran files one can access the diagram selection through the routine update_flags. The three selection criteria are stored in a derived data type virt_flags which has fields eval_0,...,eval_2, in general ranging from zero to the length of the list given in filter.nlo.

4.2 Interfacing the code

The matrix element code generated by GoSAM provides several routines to transparently access partial or full results of the amplitude calculation. Here, we only present a minimal set of routines which can be used to obtain the set of coefficients \([a_0, c_0, c_{-1}, c_{-2}]\) for a given scale and a given set of external momenta. The routines, which can be accessed through the modules matrix8 are defined as follows:

- **initgolem** This subroutine must be called once before the first matrix element evaluation. It initializes all dependent model parameters and calls the initialisation routines of the reduction libraries.

  ```fortran
  subroutine initgolem(init.libs)
  use config, only: ki
  logical, optional, &
  intent(in) :: init.libs
  end subroutine
  end interface
  ```

  The optional argument init.libs can usually be omitted. It should be used only when several initialisation calls become necessary, but the reduction libraries and loop libraries should be initialized only once. All model parameters are accessible as global variables in the module model and should be modified (if at all) before calling initgolem.

- **samplitude** This subroutine starts the actual calculation of the amplitude for a given phase space point.

  ```fortran
  subroutine samplitude &
  (vecs,scale2,amp,ok,h)
  use config, only: ki
  use kinematics, only: num_legs
  real(ki), dimension(num_legs,4),
  intent(in) :: vecs
  double precision, &
  intent(in) :: scale2
  double precision, &
  intent(out) :: amp
  logical, optional, &
  intent(out) :: ok
  integer, optional, &
  intent(in) :: h
  end subroutine
  end interface
  ```

Additional extensions Some of GoSAM’s functionality is available through the extensions variable. On top of the already presented options for selecting a regularisation scheme (by adding the option dred) or for activating interfaces to several different reduction libraries (samurai, golem95, pjfry) the user can also add the following options:

- **fr5** adds and activates the relevant code for the computation of the finite renormalisation of \(\gamma_5\) required in the ’t Hooft Veltman scheme as described in (32).

- **powhegbox** generates routines for the computation of the color and spin correlated Born matrix elements as required by POWHEG [34].

- **autotools** uses make files which use Autoconf and Automake for compilation of the matrix element.

- **gaugecheck** replaces the polarisation vectors of external vector fields by

  \[ \epsilon^\mu(k_i) \rightarrow \epsilon^\mu(k_i) + z_i k_i^\mu \]  

  (35)

  where the variable \(z_i\) defaults to zero and is accessible in the Fortran code through the symbol gaugeiz.
The first mandatory arguments of this routine are the external momenta vecs, where vecs(i,:) contains the momentum of the i-th particle as a vector \([E_i, p_i^x, p_i^y, p_i^z]\), and we use in-out kinematics, i.e. \(p_1 + p_2 \rightarrow p_3 + \cdots + p_N\). Maximal numerical stability is achieved if the beam axis is chosen along the z-axis. The second argument, scale2 = \(\mu^2\), is the square of the renormalisation scale. As a third argument the routine expects a vector which accepts the result in the format \([a_0, c_{-1}, c_{-2}]\) with the coefficients being defined in (24) and (25). The optional argument ok may be used in order to report the outcome of the reconstruction tests in samurai if no rescue method has been chosen (see Sect. 2.4.2). The last argument allows one to select a single helicity subamplitude; the index \(h\) runs from zero to the number of helicities minus one. The labelling of the helicities is documented for each process in the file doc/process.ps.

**exitgolem** This routine should be called once after the last amplitude evaluation in the program. It closes all open log files and gracefully terminates the reduction and loop libraries.

```fortran
subroutine exitgolem(exit_libs)
use config, only: ki
logical, optional, &
intent(in) :: exit_libs
end subroutine
end interface
```

The optional argument exit_libs should only be set if multiple calls to this routine (e.g. for different matrix elements) are necessary and the dependent libraries should be terminated only once.

A small program which computes the amplitude for a set of phase space points is automatically generated with the amplitude code in the file test.f90 in the subdirectory matrix. The script config.sh in the process directory returns suitable compilation and linking options for the generated matrix element code.

### 4.3 Using the BLHA interface

The so-called **Binoth Les Houches Accord** (BLHA) [56] defines an interface for a standardized communication between one-loop programs (OLP) and Monte Carlo (MC) tools. The communication between the two sides is split into two main phases: an initialisation phase and a runtime phase. During initialisation the two programs establish an agreement by exchanging a set of files and typically initiate the code generation. The OLP runtime code is then linked to the MC program and, during the runtime phase, called through a well-defined set of routines providing NLO results for the phase space points generated by the MC. According to this standard, it is the responsibility of the MC program to provide results for the Born matrix element, for the real emission and for a suitable set of infrared subtraction terms. A schematic overview on this procedure is shown in Fig. 4.

GoSAm can act as an OLP in the framework of the BLHA. In the simplest case, the MC writes an order file—in this example it is called `olp_order.lh`—and invokes the script `gosam.py` as follows:

```bash
gosam.py --olp olp_order.lh
```

Further, GoSAM specific options can be passed either in a file or directly at the command line. One can, for example, use autotools for the compilation by modifying the above line as follows.

```bash
gosam.py --olp olp_order.lh \ extensions=autotools
```

The contract file is given the extension `.olc` by default and would be `olp_order.olc` in this example. Alternatively, the name can be altered using the `-o` option.

If successful, the invocation of `gosam.py` generates a set of files which can be compiled as before with a generated make file. The BLHA routines are defined in the Fortran module `olp_module` but can also be accessed from C programs. The routines OLP_Start and OLP_EvalSubProcess are defined exactly as in the BLHA proposal [56]. For convenience, we extended the interface by the functions OLP_Finalize(), which terminates all reduction libraries, and OLP_Option(char*, int*), which can be used to pass non-standard options at runtime. For example, a valid call in C to adjust the Higgs mass would be

```c
int ierr;
OLP_Options("mH=146.78", &ierr);
```

A value of one in ierr indicates that the setting was successful. A value of zero indicates an error.

### 4.4 Using external model files

With a few modifications in the process description files, GoSAM can immediately make use of model files generated by either FeynRules [80] in the UFO format [54] or by LanHEP [55]. In both cases, the following limitations and differences with respect to the default model files, sm and smdiag, apply:

---

9 A header file is provided in `olp.h`. 
As usual, particles can be specified by their PDG code. The field names, as used by QGRAF, are $\text{part}_i$ and $\text{ant}_i$ for the particles with the PDG code $i$ and $-i$ respectively. For example, the $W^+$ and the $W^-$ boson would be called $\text{part}_24$ and $\text{ant}_24$.

All model parameters are prefixed by the letters $\text{mdl}$ in order to avoid name clashes with existing variable names in the matrix element code.

The variable $\text{model.options}$ and the extension $\text{fr5}$ are not guaranteed to work with models other than the built-in models.

Details about the allowed names for the table columns are described in the GoSAM reference manual. Precompiled MSSM_UFO and MSSM_LHEP files can also be found in the subdirectory examples/model.

5 Sample calculations and benchmarks

The codes produced by GoSAM have been tested on several processes. In this section we describe some examples of applications. Additional results, whose corresponding code is also included in the official distribution of the program, will be reported in Appendix B.

5.1 $pp \rightarrow W^- + j$ with SHERPA

In Sect. 4.3 the BLHA interface of GoSAM was presented. This interface allows one to link the program to a Monte Carlo event generator, which is, in general, responsible for supplying the missing ingredients for a complete NLO calculation of a physical cross section. Among the different general purpose Monte Carlo event generators, SHERPA [81] is one of those which offers these tools: computing the LO cross section, the real corrections with both the subtraction terms and the corresponding integrated counterparts [82–84]. Furthermore, SHERPA offers the possibility to match a NLO calculation with a parton shower [85, 86]. Using the BLHA interface, we linked GoSAM with SHERPA to compute the physical cross section for $W^- + 1$ jet at NLO.

The first steps to perform this linking is to write a SHERPA input card for the desired process. Instructions and many examples on how to write this can be found in the on-line...
Running the code for the first time will produce an order file OLF_order.lh which contains all the necessary information for GoSam, to produce the desired code for the loop part of the process. This includes a list of all partonic subprocesses needed. In parallel to the production of the needed SHERPA libraries with the provided script, one can at this point run the gosam.py command with the flag -olp and the correct path to the order file as explained in Sect. 4.3. Further options may be specified. Among them it is useful to have a second, GoSam-specific, input card with all the important GoSam options. Since, at the end, SHERPA needs to be linked to a dynamic library, it is convenient to run GoSam with the autotools extension, which allows the direct creation of both static and dynamic libraries, together with the test routine test. The gosam.py script creates all the files needed for interfacing GoSam with the Monte Carlo event generator together with the code for the one-loop computation of all needed subprocesses, and a makefile to run them. The different parton-level subprocesses are contained in different subdirectories. At this point the user simply has to run the makefile to generate and compile the code. Once the one-loop part of the code is ready, the produced shared library must be added to the list of needed libraries in the SHERPA input card as follows.

\[
\text{SHERPA\_LDADD = LHOLE golem\_olp;}
\]

With this operation the generation of the code is completed. The evaluation of the process and the physical analysis can then be performed at the user’s discretion following the advice given in the SHERPA on-line documentation [87].

We tested the BLHA interface by computing \( W^- + 1 \) jet and producing distributions for several typical observables. In Figs. 5(a) and 5(b) the inclusive transverse momentum and rapidity of the jets is shown. These distributions were compared with similar ones produced using the program MCFM [27, 28], and perfect agreement was found.

### 5.2 \( pp \rightarrow W^\pm + j \), EW corrections

As a first example of an electroweak calculation, we computed the virtual one-loop corrections to \( u\bar{d} \rightarrow Wg \). A complete analytical calculation for this process was presented in Ref. [88].

| parameters |
|------------|------------|------------|------------|
| \( M_Z \)  | 91.1876    | \( M_W \)  | 80.419     |
| \( \cos \theta_w \) | 0.88156596117995394232 |

For the kinematic point given in Table 2 and the above parameters we obtain the following result:

\[
\begin{align*}
\alpha_0 & = 2.812364835883295 \\
\alpha_0/\mu & \text{ unren.} = -94.52525523327047 \\
\alpha_{-1}/\mu & \text{ unren.} = 17.8424023696827 \\
\alpha_{-2}/\mu & \text{ unren.} = -0.5555555555555560 \\
\end{align*}
\]

The poles have been renormalized using (49)–(64) in Sects. 3.3 and 3.4 of [88]. Our result is agreement with (67), (70) of Ref. [88] and with Ref. [89] for the infrared divergences that remain after renormalisation.

Fig. 5 NLO calculation of \( W^- + 1 \) jet production at LHC using GoSam interfaced with SHERPA via the BHIA interface. The comparison to MCFM is also shown.

(a) Inclusive transverse momentum of the leading jet for \( W^- + 1 \) jet production at LHC with \( \sqrt{s} = 7 \) TeV.

(b) Inclusive pseudorapidity of the leading jet for \( W^- + 1 \) jet production at LHC with \( \sqrt{s} = 7 \) TeV.
5.3 $\gamma \gamma \rightarrow \gamma \gamma$

The process $\gamma \gamma \rightarrow \gamma \gamma$ in the Standard Model first arises at the one-loop order, and proceeds through a closed loop of fermions and $W$ bosons. Of the 16 helicity amplitudes contributing to it, only three are independent and their analytic expressions can be found in [90]. The pure QED contribution, involving a fermion loop, is contained in samurai-1.0 [40] and will not be repeated here. Instead, we show the results of the $W$-loop contribution to the independent helicity amplitudes, as an example of EW corrections that can be handled with GoSam.

With the above parameters and the kinematics of Table 3 we obtain the following results.

result $\gamma \gamma \rightarrow \gamma \gamma$ (EW)

| $\mathcal{M}$ | GoSam (dred) | Refs. [90] |
|--------------|--------------|-------------|
| $\mathcal{M}^{++++}$ | 12.02541904626610 | 12.025419045962 |
| $\mathcal{M}^{++---}$ | 7.3804060434329961 | 7.3804060437434 |
| $\mathcal{M}^{+=-+}$ | 982.7804939723322 | 982.78049397093 |

5.4 $pp \rightarrow \chi_1^0 \chi_1^0$ in the MSSM

As an example for the usage of GoSam with a model file different from the Standard Model we calculated the QCD corrections to neutralino pair production in the MSSM. The model file has been imported via the interface UFO (Universal FeynRules Output) [54] which facilitates the import of Feynman rules generated by FeynRules [80] to programs generating one-loop amplitudes. To import such files within the GoSam setup, all the user has to do is to give the path to the corresponding model file in the input card.

For this example, we combined the one-loop amplitude with the real radiation corrections to obtain results for differential cross sections. A calculation of neutralino pair production for the LHC presenting total cross sections at NLO is given in [91].

For the infrared subtraction terms the program MadDipole [92, 93] is used, the real emission part is calculated using MadGraph/MadEvent [94]. The virtual matrix element is renormalized in the $\overline{MS}$ scheme, while massive particles are treated in the on-shell scheme. The renormalisation terms specific to the massive MSSM particles have been added manually.

In Fig. 6 we show the differential cross section for the $m_{\chi_1^0 \chi_1^0}$ invariant mass, where we employed a jet veto to suppress large contributions from the channel $gg \rightarrow \chi_1^0 \chi_1^0 q$ which opens up at order $\alpha^2 \alpha_s$, but for large $p_T^{\text{jet}}$ belongs to the distinct process of neutralino pair plus one hard jet production at leading order. We used $N_f = 5$ massless quark flavours and the MSTW08 [95] parton distribution functions. For the SUSY parameters we use the modified benchmarks point SPS1amod suggested in [96], and we use $\sqrt{s} = 7$ TeV.

For reference, we also give the result for the unrenormalised amplitude at one specific phase space point for $u\bar{u} \rightarrow \chi_1^0 \chi_1^0$ in the DRED scheme, using the following parameters and momenta:

### Table 2 Kinematic point used in $pp \rightarrow W^\pm + j$, EW

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $u$ | 500   | 0     | 0     | 500   |
| $d$ | 500   | 0     | 0     | 500   |
| $W$ | 503.23360778049988 | 110.20691318538486 | 441.95397288433196 | −198.26237811718670 |
| $g$ | 496.76639221950012 | −110.20691318538488 | −441.95397288433202 | 198.26237811718664 |

### Table 3 Kinematic point used in $\gamma \gamma \rightarrow \gamma \gamma$

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $\gamma$ | 500   | 0     | 0     | 500   |
| $\gamma$ | 500   | 0     | 0     | −500  |
| $\gamma$ | 500   | 436.6186300198938284 | −59.1784256571505765 | 236.3516148798047425 |
| $\gamma$ | 500   | −436.6186300198938284 | 59.1784256571505765 | −236.3516148798047425 |
parameters

\[ M_Z = 91.1876 \quad \Gamma_Z = 0 \]
\[ M_W = 79.829013 \quad \sin^2 \theta_W = 1 - \frac{M_W^2}{M_Z^2} \]
\[ \mu = 0 \quad N_f = 0 \]
\[ g_s = 1 \quad \alpha_s^{-1} = 127.934 \]
\[ M_{\tilde{g}} = 96.680866 \quad M_{\tilde{f}} = 607.713704 \]
\[ M_{\tilde{u}} = 561.119014 \quad M_{\tilde{u}_R} = 549.259265 \]
\[ M_{\tilde{t}} = 549.259265 \quad M_{\tilde{t}_R} = 549.259265 \]
\[ M_{\tilde{b}} = 399.960116 \quad M_{\tilde{b}_R} = 399.960116 \]

All widths have been set to zero; for further settings we refer to the model parameter files contained in the subdirectory examples/model/MSSM_UFO. We have checked that the pole terms of the renormalised amplitude cancel with the infrared poles from MadDipole. For the phase space point given in Table 4 we obtain the following numbers.

| GoSAM result | ee → eeγ |
|--------------|----------|
| a_0          | 0.8680577964234597 \times 10^{-3} |
| c_0/a_0      | -3.9136615197871 |
| c_1/a_0      | 13.4374663711899 |
| c_2/a_0      | 2.6666666666667 |

5.5 e⁺e⁻ → e⁺e⁻γ

As an example of a QED calculation, we compared the virtual QED corrections for the process e⁺e⁻ → e⁺e⁻γ with the results provided in [97]. The results compared in the table are the bare unrenormalised amplitudes in the 't Hooft Veltman scheme. No counterterms or subtraction terms have been added to the result.

parameters

\[ \sqrt{s} = 1.0 \quad \alpha = 7.2973525376 \times 10^{-3} \]
\[ \mu = \sqrt{s} \quad m_e = 0.51099891 \times 10^{-3} \]

Using the parameters given above and the kinematics of Table 5 we obtain the following results.

| GoSAM | Ref. [97] |
|-------|-----------|
| a_0   | 0.7586101468103619 |
| c_0/a_0 | 0.5005827938274877 |
| c_1/a_0 | 0.047450640708029 |
| c_2/a_0 | 0.0474506427003504 |

Table 4 Kinematic point used in pp → χ₁⁺χ₀⁻ in the MSSM

| E   | p_x | p_y | p_z |
|-----|-----|-----|-----|
| u   | 1000| 0   | 0   | 1000|
| u̅  | 1000| 0   | 0   | -1000|
| χ₁⁺ | 1000| 42.3752677206678996 | 115.0009952646289548 | 987.7401101322898285|
| χ₀⁻ | 1000| -42.3752677206678996 | -115.0009952646289548 | -987.7401101322898285|

Table 5 Kinematic point used in e⁺e⁻ → e⁺e⁻γ

| E   | p_x | p_y | p_z |
|-----|-----|-----|-----|
| e⁺ (in) | 0.5 | 0   | 0.4999997388800458 | 0 |
| e⁻ (in) | 0.5 | 0   | -0.4999997388800458 | 0 |
| e⁺ (out) | 0.1780937847558600 | 0.05006809884093004 | 0.1133477415216646 |
| e⁻ (out) | 0.3563944406457374 | 0.1832142729949070 | 0.3043534176228102 |
| γ    | 0.4655117745984024 | 0.156521724517891 | 0.133146171539769 | 0.411701159144748 |
5.6 \( pp \rightarrow t\bar{t}H \)

This process has been compared with the results given in [39]. The partonic subprocesses \( u\bar{u} \rightarrow t\bar{t}H \) and \( gg \rightarrow t\bar{t}H \) where computed both in the 't Hooft Veltman scheme and in dimensional reduction and the fully renormalised results were successfully compared as an internal consistency check. Apart from wave function renormalisation and mass counterterms, Yukawa coupling renormalisation is also needed here. Yukawa coupling counterterms are in this case equal to the wave function counterterms. The Yukawa top mass is set equal to its pole mass.

| \( \sqrt{s} \) | 500.0 | \( N_f \) | 5 |
| \( \mu \) | \( m_t \) | \( \alpha_s \) | 0.1076395107858145 |
| \( m_t \) | 172.6 | \( \alpha_s \) | 246.21835258713082 |
| \( m_H \) | 130 | \( \alpha_s \) |

The kinematics used to obtain the results below is given in Table 6. The results are given in the 't Hooft Veltman scheme, and are fully renormalised.

result \( u\bar{u} \rightarrow t\bar{t}H \)

| \( a_0 \cdot 10^5 \) | 2.200409364806190 | 2.200409364806190 |
| \( c_0/a_0 \) | \(-15.29615178164782\) | \(-15.29615178164782\) |
| \( c_{-1}/a_0 \) | \(-1.640361500121837\) | \(-1.640361500121837\) |
| \( c_{-2}/a_0 \) | \(-2.666666666666666\) | \(-2.666666666666666\) |

result \( gg \rightarrow t\bar{t}H \)

| \( a_0 \cdot 10^6 \) | 6.127399805961155 | 6.127399805961155 |
| \( c_0/a_0 \) | 9.0066809536410272 | 9.0066809536410272 |
| \( c_{-1}/a_0 \) | 2.9863477301662056 | 2.9863477301662056 |
| \( c_{-2}/a_0 \) | \(-6.00000000000004\) | \(-6.00000000000004\) |

On an Intel Core i7 950 at 3GHz the evaluation of a single phase space point took 44 ms in the \( u\bar{u} \) channel and 223 ms in the \( gg \) channel. The code was compiled with gfortran without optimisations.

5.7 \( gg \rightarrow t\bar{t}Z \)

This amplitude, fully renormalised, has been compared with the results given in [37].

| \( g_s \) | 1 |
| \( m_t \) | 170.9 |
| \( \alpha_{em} \) | 0.1076395107858145 |
| \( \alpha_s \) | 0.1076395107858145 |

The kinematics used to obtain the results below is given in Table 7.

result \( gg \rightarrow t\bar{t}Z \)

| \( a_0 \cdot 10^6 \) | 0.1531395190212139 | 0.1531395190212139 |
| \( c_0/a_0 \) | \(-204.9208290898557\) | \(-204.9208290898557\) |
| \( c_{-1}/a_0 \) | \(-204.920829867328\) | \(-204.920829867328\) |
| \( c_{-2}/a_0 \) | \(-5.999999999999997\) | \(-6.00000000000003\) |

Table 6 Kinematic point used in \( pp \rightarrow t\bar{t}H \)

| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( u/g \) | 250.0 | 0.0 | 0.0 | 250.0 |
| \( \bar{u}/g \) | 250.0 | 0.0 | 0.0 | 250.0 |
| \( H \) | 15.133871809486299 | 136.35582793693018 |
| \( t \) | 20.889486679044587 | 181.47665951104506 |
| \( \bar{t} \) | 36.023358488530903 | 182.16751255202476 |

Table 7 Kinematic point used for \( gg \rightarrow t\bar{t}Z \)

| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( g \) | 7000.0 | 0.0 | 0.0 | 7000.0 |
| \( g \) | 7000.0 | 0.0 | 0.0 | 7000.0 |
| \( t \) | 806.937261988771772 | 6270.1855170414337 |
| \( \bar{t} \) | 5306.337428745517 | 6925.5258180925930 |
| \( Z \) | 474.9390792714622 | 804.28866486597315 |
The evaluation of a single phase space point took 1433 ms on a 2 GHz processor. The code was compiled with gfortran -O2.

5.8 \( pp \to b\bar{b}b + X \)

A detailed discussion of this process can be found in [98, 99]. In this section we focus on the parts that are relevant in the context of the virtual corrections. In particular we compared our result to the one given in [38], which is the fully renormalised amplitude including the mass counterterms for the top-quark contribution.

The results below are obtained for the phase space point of Table 8 using the above parameters.

| result \( gg \to b\bar{b}b \) | GoSAM | Ref. [38] |
|-------------------------------|--------|-----------|
| \( a_0 \cdot 10^9 \)         | 5.753293428094349 | 5.753293428094391 |
| \( c_{0/a_0} \)             | -22.19223384585620 | -22.19223384564902 |
| \( c_{-1/a_0} \)           | -20.89828996870689 | -20.89828996857439 |
| \( c_{-2/a_0} \)           | -8.0000000000000199 | -8.000000000000037 |

On an Intel Xeon E7340 the running time for the calculation of a single phase space point was 19.6 s for the gluon initiated channel and 440 ms for the quark channel.

5.9 \( pp \to t\bar{t}b\bar{b} + X \)

This process has been compared with the results given in [38]. We have set up the process both in the 't Hooft Veltman scheme and in dimensional reduction and successfully compared the fully renormalised results as an internal consistency check. The results below are given in the 't Hooft Veltman scheme, and only the counterterms for \(|M|^2_{\text{ct,sm}}\) are included.

| result \( u\bar{u} \to b\bar{b}b \) | GoSAM | Ref. [38] |
|-------------------------------|--------|-----------|
| \( a_0 \cdot 10^9 \)         | 5.753293428094349 | 5.753293428094391 |
| \( c_{0/a_0} \)             | -22.19223384585620 | -22.19223384564902 |
| \( c_{-1/a_0} \)           | -20.89828996870689 | -20.89828996857439 |
| \( c_{-2/a_0} \)           | -8.0000000000000199 | -8.000000000000037 |

Using the above parameters and the phase space point of Table 9 we obtain the following results.

| Table 8 Kinematic point used in \( pp \to b\bar{b}b \) |
|-------------------------------|--------|-----------|
| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( u/g \) | 250.0 | 0.0 | 0.0 | 250.0 |
| \( \bar{u}/g \) | 250.0 | 0.0 | 0.0 | -250.0 |
| \( b \) | 147.5321146846735 | 24.97040523056789 | -18.43157602837212 | 144.2306511496888 |
| \( \bar{b} \) | 108.7035966213640 | 103.2557390255471 | -0.5486486459584054 | 33.97680766420219 |
| \( b \) | 194.0630765341365 | -79.89596300367462 | 7.485866671764871 | -176.6948628845280 |
| \( \bar{b} \) | 49.70121215982584 | -48.33018125244035 | 11.49417782256567 | -1.512595929362970 |

| Table 9 Kinematic point used in \( pp \to t\bar{t}b\bar{b} \) |
|-------------------------------|--------|-----------|
| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( u/g \) | 250.0 | 0.0 | 0.0 | 250.0 |
| \( \bar{u}/g \) | 250.0 | 0.0 | 0.0 | -250.0 |
| \( t \) | 190.1845561691092 | 12.99421901255723 | -9.591511769543683 | 75.05543670827210 |
| \( \bar{t} \) | 108.0642163285034 | 110.4537390255471 | -0.5486486459584054 | 33.97680766420219 |
| \( b \) | 194.0630765341365 | -79.89596300367462 | 7.485866671764871 | -176.6948628845280 |
| \( \bar{b} \) | 25.86375471407044 | -45.15029102876678 | 5.981395280396083 | -0.7871319108423604 |
5.10 pp → W+W−b̄b̄

The subprocesses ūu → W+W−b̄b̄ and gg → W+W−b̄b̄ have been calculated both in [38] and [39]. Accordingly, the results below are given in the 't Hooft Veltman scheme, where only the counterterms for |M|2 in the gg channel. The code was compiled without optimisations.

### 5.11 ūd → W+gg

The amplitude ūd → W+gg is an important channel in the calculation of the process pp → W+ + 3 jets. The QCD corrections to this process have been presented in Refs. [6–9].

The subprocess with one quark pair and three gluons consists of more than 1500 Feynman diagrams. We have computed the amplitude including the leptonic decay of the W-boson and compared our result to [38].

### Table 10 Kinematic point used in pp → W+W−b̄b̄

| E  | px | py | pz |
|----|----|----|----|
| u/g | 250.0 | 0.0 | 0.0 | 250.0 |
| i/g | 250.0 | 0.0 | 0.0 | 250.0 |
| W+ | 154.8819879118765 | 22.40377113462118 | -16.53704884550758 | 129.4056091248114 |
| W− | 126.4095336206695 | 92.64238702192333 | -0.497098751607141 | 30.48443210132545 |
| b  | 174.1159068988160 | -71.68369328357026 | 6.716416578342183 | -158.5329205583284 |
| ̄b | 44.59257156863792 | -43.36246487297426 | 10.31272528177322 | -1.35712066754454 |

Furthermore, the values for the dependent parameters are $\cos^2 \theta_W = M_W^2 / M_Z^2$ and $\alpha = \sqrt{2} G_{F} M_W^2 \sin^2 \theta_W / \pi$. For the phase space point of Table 11 we obtain the numbers below.
Table 11 Kinematic point used in $u\bar{d} \rightarrow W^+ ggg$

|   | $E$  | $p_x$ | $p_y$ | $p_z$ |
|---|------|-------|-------|-------|
| $u$ | 250.0 | 0.0   | 0.0   | 250.0 |
| $\bar{d}$ | 250.0 | 0.0   | 0.0   | -250.0 |
| $W^+$ | 162.5391101447744 | 23.90724239064912 | -17.6468163854432 | 138.0897548661186 |
| $g$ | 104.0753274555388 | 98.859428123863483 | -0.5251163702879512 | 32.53017998659339 |
| $g$ | 185.8004627300822 | -76.49423931754684 | 7.167141557113385 | -169.1717405928078 |
| $g$ | 47.58508783676686 | 11.00479118171890 | -1.448194259904179 |

On an Intel Core 2 i5 Laptop at 2.0 GHz the evaluation of a single phase space point took about 2.5 s for $u\bar{d} \rightarrow e^+\nu_e ggg$ and about 7.5 s for on-shell W’s without decay. The code was compiled with gfortran -O2.

5.12 $u\bar{d} \rightarrow W^+(\rightarrow \nu_e e^+)b\bar{b}$ (massive $b$-quark)

The process $u\bar{d} \rightarrow W^+b\bar{b}$, with an on-shell $W$-boson, has been studied in [100], while the effects of the $W$-decay have been recently accounted for in [101], and implemented within MCFM. We consider the latter process, and compare the renormalised amplitude evaluated by MCFM. The $b$-quark is treated as massive in all diagrams except in the vacuum-polarisation like contributions.

Using the above parameters and the kinematics given in Table 12 we obtain the following results.

Table 12 Kinematic point used in $u\bar{d} \rightarrow W^+b\bar{b}$

|   | $E$  | $p_x$ | $p_y$ | $p_z$ |
|---|------|-------|-------|-------|
| $u$ | 76.084349979114506 | 0.0   | 0.0   | 76.084349979114506 |
| $\bar{d}$ | -998.0331337409114 | 0.0   | 0.0   | -998.0331337409114 |
| $\nu_e$ | 955.01676368653477 | 955.01676368653477 | 50.025808060592873 | 17.060211586132972 |
| $e^+$ | 194.2279012023398 | 194.2279012023398 | 4.3588877692445251 | 39.063065018596490 |
| $b$ | 468.23544715890415 | 468.23544715890415 | 208.221739968401815 | 40.62579518424117 |
| $\bar{b}$ | -360.8008778946474 | -360.8008778946474 | -262.60643579391922 | -96.749061789153586 |
terms of Feynman diagrams within the dimensional regularisation scheme, and optionally the calculation can be carried out either in the ’t Hooft Veltman or in the dimensional reduction variant. The user can choose among different libraries for the master integrals, and the setup is such that other libraries can be linked easily.

The calculation of the rational terms is very modular and can proceed either along with the same numerical reduction as the rest of the amplitude, or independently, before any reduction, by using analytic information on the integrals which can potentially give rise to a rational part. In the current version of the code, UV-renormalisation counterterms are provided for QCD corrections only. Further improvements concerning the full automatisation of electroweak corrections are planned.

Different systems to detect and rescue numerical instabilities are implemented, and the user can switch between them without having to re-generate the source code. Due to a careful organisation of the calculation both at the code generation stage and at the reduction stage, the runtimes for multi-particle amplitudes are very satisfactory. Moreover, the GoSAM generator can also produce codes for processes that include intermediate states with complex masses.

Within the context of the automated matching of Monte Carlo programs to NLO virtual amplitudes, GoSAM can be used as a module to produce differential cross sections for multi-particle processes which can be compared directly to experiment. Therefore we believe that GoSAM can contribute to the goal of using NLO tools as a standard framework for the LHC data analysis at the TeV scale.

Acknowledgements We would like to thank the SHERPA collaboration for the support, in particular Jennifer Archibald, Frank Krauss and Marek Schönherr. We also would like to thank Rikwert Frederix, Adam Kardos, Stefano Pozzorini, Zoltan Trocsanyi, Thomas Schutzmeier and Christian Sturm for their input to various comparisons and clarifying discussions, and Edoardo Mirabella for important feedback on Samurai. G.C. and G.L. were supported by the British Science and Technology Facilities Council (STFC). The work of G.C. was supported by DFG Sonderforschungsbereich Transregio 9, Computergestützte Theoretische Teilchenphysik. N.G. was supported in part by the U.S. Department of Energy under contract No. DE-FG02-91ER40677. P.M. and T.R. were 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-1068850. The research of F.T. is supported by Marie-Curie-IF, project: “SAMURAI-Apps”. We also acknowledge the support of the Research Executive Agency (REA) of the European Union under the Grant Agreement number PITN-GA-2010-264564 (LHCPhenoNet).

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

Appendix A: Examples included in the release

In the following we give results for the processes listed in the examples directory. Unless stated otherwise, we assume that the coupling constants (\(\alpha_s\) and \(g_s\) in the standard model) have been set to one in the input card. The conventions for the returned numbers \((a_0, c_0, c_{-1}, c_{-2})\) are as stated in Sect. 2.5. Dimensionful parameters are understood to be in powers of \(\text{GeV}\).

As an illustration of the potential of GoSAM, we display in Table 13 the timings required by a wide list of benchmark processes. The first value provided in the table is the time required for the code generation (Generation, given in seconds): we remind the reader that this operation only needs to be performed once per process. The second value is the timing for the full calculation of the amplitude at one phase-space point (Evaluation, in milliseconds). Results are obtained with an Intel(R) Core(TM) i7 CPU 950 @ 3.07 GHz.

Table 13 Time required for code generation and calculation of one phase-space point. The results were obtained with an Intel(R) Core(TM) i7 CPU 950 @ 3.07 GHz. The time for the evaluation of a phase space point is taken as the average of the time obtained from the evaluation of 100 random points generated using RAMBO [102], where the code was compiled using gfortran without any optimisation options. The generation of the \(R_\perp\) term was set to explicit.

| Process               | Generation [s] | Evaluation [ms] |
|-----------------------|----------------|-----------------|
| \(bg \to Hb\)         | 236            | 2.49            |
| \(d\bar{d} \to t\bar{t}\) | 341            | 4.71            |
| \(d\bar{d} \to t\bar{t}\) (DRED) | 324            | 4.05            |
| \(dg \to dg\)         | 398            | 3.08            |
| \(dg \to dg\) (DRED)  | 402            | 3.28            |
| \(e^+ e^- \to t\bar{t}\) | 221            | 1.27            |
| \(e^+ e^- \to t\bar{t}\) (LanHEP) | 180            | 1.27            |
| \(gg \to gg\)         | 525            | 1.69            |
| \(gg \to gg\) (DRED)  | 428            | 1.66            |
| \(gg \to gg\) (LanHep) | 1022           | 1.70            |
| \(gg \to gZ\)         | 529            | 15.18           |
| \(gg \to t\bar{t}\)   | 1132           | 24.65           |
| \(gg \to t\bar{t}\) (DRED) | 957            | 30.13           |
| \(gg \to t\bar{t}\) (UFO) | 1225           | 29.45           |
| \(H \to \gamma \gamma\) | 140            | 0.24            |
| \(gb \to e^- \bar{\nu}_t t\) | 337            | 2.89            |
| \(ud \to e^- \bar{\nu}_e\) | 71             | 0.09            |
| \(ud \to e^- \bar{\nu}_g\) | 154            | 1.15            |
| \(u\bar{u} \to d\bar{d}\) | 186            | 2.06            |
| \(\bar{u}d \to W^+ W^+ \bar{\ell}_s\) | 1295           | 17.37           |
| \(\gamma \gamma \to \gamma \gamma\) | 597            | 6.08            |
A.1 How to run the examples

The example directories only define the system independent part of the setup. All settings which are defined in the file system.rc (see Sect. 4) must be put either in a file called $HOME/.gosam or in the file setup.in in the GoSAM examples/ directory. A script runtests.sh is provided to generate, compile and run the test programs. The names of the directories respectively examples to be run should be specified at the command line, e.g.

```
./runtests.sh eeuu bghb
```

If the script is invoked without arguments it will loop over all subdirectories. A second script, summarize.sh, can be used in order to collect the test results and print a summary to the screen. The command

```
./summarize.sh
```

will produce an output like the following one.

```
+ bghb (succeeded)
+ eeuu (succeeded)
grep: ./ddtt/...: No such file ...
```

The examples $e^+e^- \rightarrow t\bar{t}$ have an explicit dependence on the Golem95C library and will therefore fail if the extension golem95 is not added.

A.2 $e^+e^- \rightarrow u\bar{u}$

The following parameters and momenta have been used to produce the numerical result:

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $e^+$ | $E$ | 0 | 0 | $E$ |
| $e^-$ | $E$ | 0 | 0 | $-E$ |
| $u$ | $E$ | $E \sin \theta \sin \phi$ | $E \sin \theta \cos \phi$ | $E \cos \theta$ |
| $\bar{u}$ | $E$ | $-E \sin \theta \sin \phi$ | $-E \sin \theta \cos \phi$ | $-E \cos \theta$ |

A.3 $e^+e^- \rightarrow t\bar{t}$

This example has been produced twice: once with the default model file and once with a model file imported from LanHEP [55]. Thus it also can serve as an example of how to import model parameters from LanHEP. The result is given in dimensional reduction, and no renormalisation terms are included.

| $M_Z$ | 91.1876 | $\Gamma_Z$ | 2.4952 |
| $M_W$ | $\cos \theta_w \ M_Z$ | $\sin \theta_w$ | 0.47303762 |
| $m_t$ | 172.5 | $\mu^2$ | $m_t^2$ |

A.4 $u\pi \rightarrow d\bar{d}$

This example has been produced twice: once in the ’t Hooft Veltman (HV) scheme and once with dimensional reduction (DRED). Only the result in the HV scheme will be

```
Table 14 Kinematic point used in $e^+e^- \rightarrow t\bar{t}$

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $e^+$ | 74.7646520969852 | 0 | 0 | 74.7646520969852 |
| $e^-$ | 6067.88254935176 | 0 | 0 | $-6067.88254935176$ |
| $t$ | 5867.13826404309 | 16.794697430656 | 169.437140279981 | $-5862.12966020487$ |
| $\bar{t}$ | 275.508937405653 | $-16.794697430656$ | $-169.437140279981$ | $-130.988237049907$ |
```
listed below, for the DRED calculation see the directory uudd_dred.

parameters

\[ \mu = 91.188 \quad N_f = 2 \]

Using the above parameters and the phase space point of Table 15 we obtain the following numbers.

result \( u\bar{u} \rightarrow d\bar{d} \)

| GoSAM(HV) | Ref. [103] |
|----------------|-------------|
| \( a_0 \) | 0.28535063700913421 | 0.28535063700913416 |
| \( c_0/a_0 \) | -2.7940629929270155 | -2.7940629929268876 |
| \( c_{-1}/a_0 \) | -6.4881359148866604 | -6.4881359148866391 |
| \( c_{-2}/a_0 \) | -5.3333333333333333 | -5.3333333333333333 |

A.5 \( gg \rightarrow gg \)

This example has been produced both with the default model file and with a model file imported from LanHEP. Further, it has been calculated in the ’t Hooft Veltman scheme and in the dimensional reduction scheme. Only the results in the ’tHooft Veltman scheme are listed below, for further details please see the subdirectories gggg_dred and gggg_lhep. The result is for the helicity configuration \( g(+)g(+) \rightarrow g(-)g(-) \), and pure Yang-Mills theory, i.e. fermion loops are not included.

parameters

\[ \mu^2 = 442 \quad N_f = 0 \]

\[ \alpha_s = 0.13 \]

Evaluating the amplitude for above parameters and the phase space point given in Table 16 we obtain the following results.

result \( gg \rightarrow gg \)

| GoSAM(HV) | Ref. [104] |
|----------------|-------------|
| \( a_0 \) | 14.120983050796795 | 14.120983050796804 |
| \( c_0/a_0 \) | -124.0247557942351 | -124.02475579423495 |
| \( c_{-1}/a_0 \) | 55.003597347101078 | 55.003597347101035 |
| \( c_{-2}/a_0 \) | -12.0000000000000000 | -12 |

A.6 \( gg \rightarrow gZ \)

As this process has no tree level amplitude, the result is for the one-loop amplitude squared.

parameters

\[ \mu^2 = 442 \quad s_{12} = 0.13 \quad \alpha_s = 1 \quad M_Z = 91.1876 \quad \Gamma_Z = 0 \quad \sin^2 \theta_W = 0.4808222 \quad M_W = 80.41891 \quad \cos^2 \theta_W = 0.4808222 \]

With the above parameters and the kinematics given in Table 17 we obtain the following result.

result \( gg \rightarrow gZ \)

\[ |\mathcal{M}|^2_{1\text{-loop}} = 0.1075742599502829 \quad 0.10757425995048300 \]

A.7 \( d\bar{d} \rightarrow t\bar{t} \)

This example has been calculated in the ’t Hooft Veltman scheme and in the dimensional reduction scheme. Only the

| Table 15 | Kinematic point used in \( u\bar{u} \rightarrow d\bar{d} \) |
|----------------|-------------|
| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( u \) | 102.6289752320661 | 0 | 0 | 102.6289752320661 |
| \( \bar{u} \) | 102.6289752320661 | 0 | 0 | -102.6289752320661 |
| \( d \) | 102.6289752320661 | -85.98802977488269 | -12.11018104528534 | 54.70017191625945 |
| \( \bar{d} \) | 102.6289752320661 | 85.98802977488269 | 12.11018104528534 | -54.70017191625945 |

| Table 16 | Kinematic point used in \( gg \rightarrow gg \) |
|----------------|-------------|
| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
| \( p_1 \) | 220.9501779577791 | 0 | 0 | 220.9501779577791 |
| \( p_2 \) | 220.9501779577791 | 0 | 0 | -220.9501779577791 |
| \( p_3 \) | 220.9501779577791 | 119.9098300357375 | 183.049135511419 | -30.5548585967430 |
| \( p_4 \) | 220.9501779577791 | -119.9098300357375 | -183.049135511419 | 30.5548585967430 |
results in the ’t Hooft Veltman scheme are listed below, for the renormalised amplitude with \( N_f = 5 \) and the top mass renormalised on-shell.

For further details please see the subdirectories \( \dd t t \) and \( \dd t t _d \_r e d \).

| parameters |
|------------|
| \( m_t \) | 172.5 |
| \( \mu^2 \) | \( N_f \) |
| \( \alpha_s \) | 1 |
| \( m^2_t \) | 5 |

With the above parameters and the kinematics given in Table 18 we obtain the following results.

Table 18  Kinematic point used in \( d \bar{d} \rightarrow t \bar{t} \)

| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
|------|------|------|------|
| \( d \) | 74.7646520969852 | 0 | 0 | 74.7646520969852 |
| \( \bar{d} \) | 6067.88254935176 | 0 | 0 | -6067.88254935176 |
| \( t \) | 5867.13826404309 | 16.7946967430656 | 169.437140279981 | -5862.12966020487 |
| \( \bar{t} \) | 275.508937405653 | -16.7946967430656 | -169.437140279981 | -130.988237049907 |

Table 19  Kinematic point used in \( gg \rightarrow t \bar{t} \)

| \( E \) | \( p_x \) | \( p_y \) | \( p_z \) |
|------|------|------|------|
| \( g \) | 137.84795086008967 | 0 | 0 | 137.84795086008967 |
| \( g \) | 3161.1731634194916 | 0 | 0 | -3161.1731634194916 |
| \( t \) | 3058.6441209877348 | 16.445287185144903 | 165.91204201912493 | -3049.2945357402382 |
| \( \bar{t} \) | 240.37699329184659 | -16.445287185144903 | -165.91204201912493 | 25.969323180836145 |

A.8  \( gg \rightarrow t \bar{t} \)

The result is for the renormalised amplitude in the HV scheme.

A.9  \( bg \rightarrow Hb \)

For this process the mass of the b-quark is set to zero. However, in order to have a coupling between the b-quark and the Higgs boson, the following Yukawa coupling is implemented in the model file:

\[ L_{yuk} = Y_{Hb} \bar{\psi}_L \psi_R \phi, \; Y_{Hb} = \frac{\bar{m}_b(\mu)}{v}. \]
With the above parameters and the kinematics given in Table 20 we obtain the following results.

### Table 20 Kinematic point used in $b g \rightarrow H b$

| $E$  | $p_x$ | $p_y$ | $p_z$ |
|------|-------|-------|-------|
| $b$  | 250   | 0     | 0     | 250   |
| $g$  | 250   | 0     | 0     | −250  |
| $H$  | 264.4 | −83.84841332241601 | −86.85350630148753 | −202.3197272300720 |
| $b$  | 235.6 | 83.84841332241601  | 86.85350630148753  | 202.3197272300720   |

### A.10 $H \rightarrow \gamma\gamma$

The decay width $\Gamma_{H\rightarrow\gamma\gamma}$ of this loop induced process is known analytically at lowest order. For comparison we used the equations including the top loop and the bosonic contribution given in [108, 109]. The decay width can be expressed as

$$\Gamma_{H\rightarrow\gamma\gamma} = \frac{G_F \alpha^2 m_H^3}{128\sqrt{2}\pi^3} \hat{F}(\tau_W, \tau_t),$$

where $\tau_i = m_H^2/(4m_i^2)$ for $i = W, t$.

### Table 21 Kinematic point used in $u d \rightarrow e^- \bar{\nu}_e$

| $E$  | $p_x$ | $p_y$ | $p_z$ |
|------|-------|-------|-------|
| $u$  | 100   | 0     | 0     | 100   |
| $\bar{d}$ | 100 | 0     | 0     | −100  |
| $e^-$ | 100   | 75.541566535633046 | 30.240603423558878 | −58.12897410002661 |
| $\bar{\nu}_e$ | 100 | −75.541566535633046 | −30.240603423558878 | 58.12897410002661 |

### A.11 $u d \rightarrow e^- \bar{\nu}_e$

This example has been calculated in the ’t Hooft Veltman scheme and in the dimensional reduction scheme. Only the results in the ’t Hooft Veltman scheme are listed below, for the renormalised amplitude. In addition to a calculation with the default model file, calculations using LanHEP [55] and UFO [54] are also contained in the examples directory.

### A.12 $u d \rightarrow e^- \bar{\nu}_e g$

We list the renormalised amplitude in the HV scheme.
parameters

\begin{align*}
M_W &= 80.398 \\
\Gamma_W &= 2.1054 \\
\sin \theta_W &= 0.4808222 \\
N_f &= 5 \\
\mu^2 &= 512 \\
M_Z & \frac{M_W}{\cos \theta_W} \quad V_{ud} = 0.97419 \\
\mu^2 &= 512
\end{align*}

With the above parameters and the kinematics given in Table 22 we obtain the following results.

result $u \bar{d} \to e^- \bar{\nu}_e g$

| GoSAM(HV) | Ref. [39] |
|----------------|----------------|
| $a_0 \cdot 10^7$ | 2.3898509625435832 | 2.3898509625435922 |
| $c_0/a_0$ | -8.6052919370147745 | -8.6052919368774248 |
| $c_{-1}/a_0$ | -18.722010655600936 | -18.72201065557121 |
| $c_{-2}/a_0$ | -5.66666666666667 | -5.66666666666667 |

A.13 $g b \to e^- \bar{\nu}_e t$

We list the renormalised result in the dimensional reduction scheme.

parameters

\begin{align*}
M_W &= 80.4190 \\
\Gamma_W &= 2.04760 \\
M_Z & = 91.1876 \\
\Gamma_Z &= 2.49520 \\
m_t &= 171.2 \\
\Gamma_t &= 0 \\
\mu &= 71.2 \\
\mu^2 &= 0.30794906326863203
\end{align*}

With the above parameters and the kinematics given in Table 23 we obtain the following results.

result $g b \to e^- \bar{\nu}_e t$

| GoSAM | Ref. [27, 106] (MCFM) |
|----------------|----------------|
| $a_0 \cdot 10^7$ | 8.52301540675800134 | 8.52301540708130106 |
| $c_0/a_0$ | -79.879718568538991 | -79.879718569273024 |
| $c_{-1}/a_0$ | 26.57018548790770 | 26.570185487963364 |
| $c_{-2}/a_0$ | -4.33333333333401 | -4.3333333331689596 |

A.14 $\bar{u} \bar{d} \to W^+ W^+ \bar{\tau}_s$

Results are given for the unrenormalised amplitude in the dimensional reduction scheme.

parameters

\begin{align*}
\mu &= 80 \\
N_f &= 5
\end{align*}

With the above parameters and the kinematics given in Table 24 we obtain the following results.

result $\bar{u} \bar{d} \to W^+ W^+ \bar{\tau}_s$

| GoSAM | Ref. [20, v3] |
|----------------|----------------|
| $a_0$ | 23.35965167118 | 23.35965 |
| $c_0/a_0$ | 13.6255429251954 | 13.62554 |
| $c_{-1}/a_0$ | -5.3333333333333 | -5.33333 |

Table 22 Kinematic point used in $u \bar{d} \to e^- \bar{\nu}_e g$

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $u$ | 500   | 0     | 0     |
| $\bar{d}$ | 500 | 0     | 0     |
| $e^-$ | 483.244841094218 | -86.311221869418 | 147.629518147233 | -451.975082051212 |
| $\bar{\nu}_e$ | 279.253730247231 | 6.62401666401929 | -5.5083951102529 | 279.119009435087 |
| $g$ | 237.501788658551 | 79.6872052053988 | -142.0487863208 | 172.856072616124 |

Table 23 Kinematic point used in $g b \to e^- \bar{\nu}_e t$

| $E$ | $p_x$ | $p_y$ | $p_z$ |
|-----|-------|-------|-------|
| $g$ | 1187.7086110647201 | 0 | 0 | 1187.7086110647201 |
| $b$ | 2897.148136260289 | 0 | 0 | 1187.7086110647201 |
| $e^-$ | 2293.043558834492 | 629.81047833131981 | 258.58120146220904 | -2189.6399870328105 |
| $\bar{\nu}_e$ | 509.48956356743611 | 144.72113807954338 | 19.883624374757 | -488.098411670514 |
| $t$ | 1282.3236278741238 | -774.53161641086319 | -278.46456389968404 | 968.29887350775562 |
Appendix B: Explicit reduction of $R_2$ rational terms

In this Appendix we list all integrals which give rise to $R_2$ terms as we use these expressions in their explicit construction. We use the definition

$$I_{\nu}^{n,\alpha_1 \ldots \alpha_k}(S) = \int \frac{\mu_2^2 e^{\nu} q \cdot \hat{q}^{\alpha_1} \ldots \hat{q}^{\alpha_k} (\mu^2)^2 \alpha}{i \pi^{n/2}} D_1 \ldots D_N,$$  \hspace{1cm} (B.1)

with

$$D_I = (q + r_i)^2 - m_I^2$$

and

$$S_{ij} = (r_i - r_j)^2 - m_i^2 - m_j^2.$$  \hspace{1cm} (B.2)

The integrals up to $O(\varepsilon)$ are

$$\varepsilon \cdot I_3^{n,0,(\mu_1 \mu_2 \mu_3)}(S) = -\frac{1}{12} \sum_{i=1}^{3} [\delta^{* \gamma \gamma} \hat{r}_i^{\mu_1} \hat{r}_i^{\mu_2} \hat{r}_i^{\mu_3}],$$ \hspace{1cm} (B.12)

$$I_4^{n,1,\mu_1 \mu_2}(S) = \frac{1}{12} \delta^{\mu_1 \mu_2},$$ \hspace{1cm} (B.13)

$$I_4^{n,2}(S) = -\frac{1}{6},$$ \hspace{1cm} (B.14)

$$\varepsilon \cdot I_4^{n,0;\mu_1 \mu_2 \mu_3 \mu_4}(S) = \frac{1}{4!} [\delta^{* \gamma \gamma} \delta^{* \gamma \gamma} \delta^{* \gamma \gamma} \delta^{* \gamma \gamma}] \hat{r}_1^{\mu_1} \hat{r}_2^{\mu_2} \hat{r}_3^{\mu_3} \hat{r}_4^{\mu_4}.$$ \hspace{1cm} (B.15)

References

1. T. Hahn, Comput. Phys. Commun. 140, 418 (2001). doi:10.1016/S0010-4655(01)00290-9
2. P. Nogueira, J. Comput. Phys. 105, 279 (1993). doi:10.1006/jcph.1993.1074
3. T. Hahn, M. Perez-Victoria, Comput. Phys. Commun. 118, 153 (1999). doi:10.1016/S0010-4655(99)00173-8
4. G. Belanger, F. Boudjema, J. Fujimoto, T. Ishikawa, T. Kaneko et al., Phys. Rep. 430, 117 (2006). doi:10.1016/j.physrep.2006.02.001
5. A. Denner, S. Dittmaier, M. Roth, L. Wieders, Nucl. Phys. B 724, 247 (2005)
6. C. Berger, Z. Bern, L.J. Dixon, F. Febres Cordero, D. Forde et al., Phys. Rev. D 80, 074036 (2009). doi:10.1103/PhysRevD.80.074036
7. C. Berger, Z. Bern, L.J. Dixon, F. Febres Cordero, D. Forde et al., Phys. Rev. Lett. 102, 222001 (2009). doi:10.1103/PhysRevLett.102.222001
8. R. Ellis, K. Melnikov, G. Zanderighi, Phys. Rev. D 80, 094002 (2009). doi:10.1103/PhysRevD.80.094002
9. K. Melnikov, G. Zanderighi, Phys. Rev. D 81, 074025 (2010). doi:10.1103/PhysRevD.81.074025
10. C. Berger, Z. Bern, L.J. Dixon, F. Cordero, D. Forde et al., Phys. Rev. D 82, 074002 (2010). doi:10.1103/PhysRevD.82.074002
11. J.M. Campbell, R. Ellis, C. Williams, Phys. Rev. D 81, 074023 (2010). doi:10.1103/PhysRevD.81.074023
12. A. Breidenstein, A. Denner, S. Dittmaier, S. Pozzorini, Phys. Rev. Lett. 103, 012002 (2009). doi:10.1103/PhysRevLett.103.012002
13. A. Breidenstein, A. Denner, S. Dittmaier, S. Pozzorini, J. High Energy Phys. 1003, 021 (2010). doi:10.1007/JHEP03(2010)021
14. G. Bevilacqua, M. Czakon, C. Papadopoulos, R. Pittau, M. Worek, J. High Energy Phys. 0909, 109 (2009). doi:10.1088/1126-6708/2009/09/109
34. S. Alioli, P. Nason, C. Oleari, E. Re, J. High Energy Phys. 1006
33. S. Frixione, P. Nason, C. Oleari, J. High Energy Phys.
32. S. Frixione, B.R. Webber, J. High Energy Phys.
31. S. Frixione, B.R. Webber, C.D. White
29. K. Arnold, M. Bahr, G. Bozzi, F. Campanario, C. Englert
28. J.M. Campbell, R. Ellis, C. Williams, J. High Energy Phys.
27. J.M. Campbell, R. Ellis, Phys. Rev. D
25. C. Berger, Z. Bern, L.J. Dixon, F. Cordero, D. Forde
22. F. Campanario, C. Englert, M. Rauch, D. Zeppenfeld, Phys. Lett.
45. G. Ossola, C.G. Papadopoulos, R. Pittau, Nucl. Phys. B
44. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 07, 085 (2007). doi:10.1088/1126-6708/2007/07/085
43. L. Reina, T. Schutzmeier (2011). arXiv:1104.4388 [hep-ph]
42. G. Cullen, N. Greiner, A. Guftanti, J.P. Guillet, G. Heinrich et al., Nucl. Phys. Proc. Suppl. 205–206, 67 (2010). doi:10.1016/j.nuclphysbps.2010.08.021
41. K. Arnold, J. Bellm, B.R. Webber, J. High Energy Phys. 2006, 029 (2002)
40. S. Frixione, F. Stoeckl, P. Torrielli, J. Vermaseren, T. Reiter, Ph.D. thesis, 2009
39. V. Hirschi, R. Frederix, S. Frixione, M.V. Garzelli, F. Maltoni et al., J. High Energy Phys. 1105, 044 (2011). doi:10.1007/JHEP05(2011)044
38. A. Kardos, C. Papadopoulos, Z. Trocsanyi, Phys. Lett. B 705, 76 (2011). doi:10.1016/j.physletb.2011.09.080
37. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
36. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0904, 072 (2009). doi:10.1088/1126-6708/2009/04/072
35. A. Kardos, C. Papadopoulos, Z. Trocsanyi, J. High Energy Phys. 0909, 106 (2009). doi:10.1126/6708/2009/11/070
34. V. Hirschi, R. Frederix, S. Frixione, M.V. Garzelli, F. Maltoni et al., J. High Energy Phys. 1105, 044 (2011). doi:10.1007/JHEP05(2011)044
33. S. Alioli, P. Nason, C. Oleari, E. Re, J. High Energy Phys. 1006, 043 (2010). doi:10.1007/JHEP06(2010)043
32. A. Kardos, C. Papadopoulos, Z. Trocsanyi, Phys. Lett. B 705, 76 (2011). doi:10.1016/j.physletb.2011.09.080
31. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
30. S. Frixione, F. Stoeckl, P. Torrielli, B.R. Webber, C.D. White (2010). arXiv:1010.0819 [hep-ph]
29. S. Frixione, P. Nason, C. Oleari, J. High Energy Phys. 0711, 070 (2007). doi:10.1126/6708/2007/11/070
28. S. Alioli, P. Nason, C. Oleari, E. Re, J. High Energy Phys. 1006, 043 (2010). doi:10.1007/JHEP06(2010)043
27. A. Kardos, C. Papadopoulos, Z. Trocsanyi, Phys. Lett. B 705, 76 (2011). doi:10.1016/j.physletb.2011.09.080
26. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 1001, 040 (2010). doi:10.1007/JHEP01(2010)040
25. M. Garzelli, I. Malamos, R. Pittau, J. High Energy Phys. 1101, 029 (2009). doi:10.1088/1126-6708/2009/04/072
24. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
23. P. Draggiotis, M. Garzelli, C. Papadopoulos, R. Pittau, J. High Energy Phys. 0904, 072 (2009). doi:10.1126/6708/2009/04/072
22. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0909, 106 (2009). doi:10.1126/6708/2009/11/070
21. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
20. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
19. V. Hirschi, R. Frederix, S. Frixione, M.V. Garzelli, F. Maltoni et al., J. High Energy Phys. 1105, 044 (2011). doi:10.1007/JHEP05(2011)044
18. A. Kardos, C. Papadopoulos, Z. Trocsanyi, Phys. Lett. B 705, 76 (2011). doi:10.1016/j.physletb.2011.09.080
17. A. Kardos, C. Papadopoulos, Z. Trocsanyi, Europhys. Lett. 96, 11001 (2011). doi:10.1290/0295-5075/96/11001
16. G. Ossola, C.G. Papadopoulos, R. Pittau, J. High Energy Phys. 0805, 004 (2008). doi:10.1088/1126-6708/2008/05/004
15. G. Bevilacqua, M. Czakon, C. Papadopoulos, M. Worek, Phys. Rev. Lett. 104, 162002 (2010). doi:10.1103/PhysRevLett.104.162002
76. G. Yost et al., Phys. Lett. B 204, 1 (1988). doi:10.1016/0370-2693(88)90505-9
77. C. Caso et al., Eur. Phys. J. C 3, 1 (1998). doi:10.1007/s10052-998-0104-x
78. S. Catani, M. Seymour, Nucl. Phys. B 485, 291 (1997). doi:10.1016/S0550-3213(96)00589-5
79. S. Catani, S. Dittmaier, Z. Trocsanyi, Phys. Lett. B 500, 149 (2001). doi:10.1016/S0370-2693(01)00065-X
80. N.D. Christensen, C. Duhr, Comput. Phys. Commun. 180, 1614 (2009). doi:10.1016/j.cpc.2009.02.018
81. T. Gleisberg, S. Hoehe, F. Krauss, M. Schonherr, S. Schumann et al., J. High Energy Phys. 0902, 007 (2009). doi:10.1088/1126-6708/2009/02/007
82. F. Krauss, R. Kuhn, G. Soff, J. High Energy Phys. 0202, 044 (2002).
83. T. Gleisberg, F. Krauss, Eur. Phys. J. C 53, 501 (2008). doi:10.1140/epjc/s10052-007-0495-0
84. M. Schonherr, F. Krauss, J. High Energy Phys. 0812, 018 (2008). doi:10.1088/1126-6708/2008/12/018
85. S. Hoehe, F. Krauss, M. Schonherr, F. Siegert, J. High Energy Phys. 1104, 024 (2011). doi:10.1007/JHEP04(2011)024
86. S. Hoehe, F. Krauss, M. Schonherr, F. Siegert (2011)
87. http://projects.hepforge.org/sherpa/, 2011
88. J.H. Kuhn, A. Kulesza, S. Pozzorini, M. Schulze, Nucl. Phys. B 797, 27 (2008). doi:10.1016/j.nuclphysb.2007.12.029
89. T. Gehrmann, N. Greiner, J. High Energy Phys. 12, 050 (2010). doi:10.1007/JHEP12(2010)050
90. G. Gounaris, P. Porfyriadis, F. Renard, Eur. Phys. J. C 9, 673 (1999). doi:10.1016/S0944-8260(99)00079-9
91. W. Beenakker et al., Phys. Rev. Lett. 83, 3780 (1999). doi:10.1103/PhysRevLett.83.3780
92. R. Frederix, T. Gehrmann, N. Greiner, J. High Energy Phys. 0809, 122 (2008). doi:10.1088/1126-6708/2008/09/122
93. R. Frederix, T. Gehrmann, N. Greiner, J. High Energy Phys. 1006, 086 (2010). doi:10.1007/JHEP06(2010)086
94. J. Alwall, P. Demin, S. de Visscher, R. Frederix, M. Herquet et al., J. High Energy Phys. 0709, 028 (2007). doi:10.1088/1126-6708/2007/09/028
95. A.D. Martin, W.J. Stirling, R.S. Thorne, G. Watt, Eur. Phys. J. C 63, 189 (2009). doi:10.1140/epjc/s10052-009-1072-5
96. B. Feigl, H. Rzehak, D. Zeppenfeld (2011)
97. S. Actis, P. Mastrolia, G. Ossola, Phys. Lett. B 682, 419 (2010). doi:10.1016/j.physletb.2009.11.035
98. N. Greiner, A. Guffanti, J.P. Guillet, T. Reiter, J. Reuter, PoS DIS2010, 156 (2010)
99. T. Binoth, G. Cullen, N. Greiner, A. Guffanti, J.P. Guillet et al., PoS RADCOR2009, 026 (2010)
100. F. Febres Cordero, L. Reina, D. Wackeroth, Phys. Rev. D 74, 034007 (2006). doi:10.1103/PhysRevD.74.034007
101. S. Badger, J.M. Campbell, R. Ellis, J. High Energy Phys. 1103, 027 (2011). doi:10.1007/JHEP03(2011)027
102. R. Kleiss, W. Stirling, S. Ellis, Comput. Phys. Commun. 40, 359 (1986). doi:10.1016/0010-4655(86)90119-0
103. R. Ellis, J. Sexton, Nucl. Phys. B 269, 445 (1986). doi:10.1016/0550-3213(86)90232-4
104. T. Binoth, J. Guillet, G. Heinrich, J. High Energy Phys. 0702, 013 (2007). doi:10.1088/1126-6708/2007/02/013
105. J. van der Bij, E. Glover, Nucl. Phys. B 313, 237 (1989). doi:10.1016/0550-3213(89)90317-9
106. J.M. Campbell, R.K. Ellis, Phys. Rev. D 62, 114012 (2000). doi:10.1103/PhysRevD.62.114012
107. J.M. Campbell, R. Ellis, F. Maltoni, S. Willenbrock, Phys. Rev. D 67, 095002 (2003). doi:10.1103/PhysRevD.67.095002
108. U. Aglietti, R. Bonciani, G. Degrassi, A. Vicini, Phys. Lett. B 595, 432 (2004). doi:10.1016/j.physletb.2004.06.063
109. R. Harlander, P. Kant, J. High Energy Phys. 0512, 015 (2005). doi:10.1088/1126-6708/2005/12/015