

Edinburgh 2008/40  
LAPTH-1277/08  
IPPP/08/73, DCPT/08/146  
Nikhef-2008-26

# Golem95: a numerical program to calculate one-loop tensor integrals with up to six external legs

T.Binoth <sup>a</sup>, J.-Ph.Guillet <sup>b</sup>, G.Heinrich <sup>c</sup>, E.Pilon <sup>b</sup>, T.Reiter <sup>d</sup>

<sup>a</sup>*School of Physics, The University of Edinburgh, Edinburgh EH9 3JZ, UK*

<sup>b</sup>*LAPTH, Université de Savoie and CNRS, Annecy-le-Vieux, France*

<sup>c</sup>*Institute for Particle Physics Phenomenology, University of Durham,  
Durham, DH1 3LE, UK*

<sup>d</sup>*NIKHEF, Kruislaan 409, 1098 SJ Amsterdam, The Netherlands*

---

## Abstract

We present a program for the numerical evaluation of form factors entering the calculation of one-loop amplitudes with up to six external legs. The program is written in Fortran95 and performs the reduction to a certain set of basis integrals numerically, using a formalism where inverse Gram determinants can be avoided. It can be used to calculate one-loop amplitudes with massless internal particles in a fast and numerically stable way.

PACS: 12.38.Bx

*Key words:* NLO Computations, One-Loop Diagrams, QCD, Hadron Colliders

---

## PROGRAM SUMMARY

*Manuscript Title:* Golem95: a numerical program to calculate one-loop diagrams with up to six external legs

*Authors:* T. Binoth, J.-Ph. Guillet, G. Heinrich, E. Pilon, T. Reiter

*Program Title:* golem95\_v1.0

*Journal Reference:*

*Catalogue identifier:*

*Licensing provisions:* none

*Programming language:* Fortran95

*Computer:* Any computer with a Fortran95 compiler

*Operating system:* Linux, Unix

*RAM:* RAM used per form factor is insignificant, even for a rank six six-point form factor

*Number of processors used:* one

*Keywords:* NLO Computations, One-Loop Diagrams, Tensor Reduction

*PACS:* 12.38.Bx

*Classification:* 4.4 Feynman diagrams, 11.1 High Energy Physics Computing

*External routines/libraries:* perl

*Nature of problem:* Evaluation of one-loop multi-leg tensor integrals occurring in the calculation of next-to-leading order corrections to scattering amplitudes in elementary particle physics.

*Solution method:* Tensor integrals are represented in terms of form factors and a set of basic building blocks (“basis integrals”). The reduction to the basis integrals is performed numerically, thus avoiding the generation of large algebraic expressions.

*Restrictions:* The current version contains basis integrals for massless internal particles only. Basis integrals for massive internal particles will be included in a future version.

*Running time:* Depends on the nature of the problem. A rank 6 six-point form factor at a randomly chosen kinematic point takes 0.13 seconds on an Intel Core 2 Q9450 2.66 GHz processor.

# LONG WRITE-UP

## 1 Introduction

Collider experiments at the TeV scale, in particular the LHC experiments, are expected to shed light on the mechanism of electroweak symmetry breaking and to open up new horizons concerning our understanding of elementary particle interactions. In order to achieve these goals, expected signal as well as background rates should be well under control, which implies that a multitude of scattering processes should be known at next-to-leading order (NLO) accuracy.

Over the last years, enormous efforts have been made to calculate NLO corrections, in QCD as well as in the electroweak sector. For a review see e.g. [1]. These calculations in general involve two parts, the treatment of extra real emission and the calculation of virtual corrections, i.e. one-loop amplitudes. While the calculation of one-loop amplitudes with up to four external particles has reached a quite mature state meanwhile, and automated tools have been developed already some time ago [2,3,4,5,6], the calculation of processes with five or more external legs required and boosted new developments in various directions, for recent developments see e.g. [1,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23].

Initially, NLO calculations have mostly been done on a process-by-process basis, but fortunately we are moving towards automation also for multi-particle processes, as can be seen from the tools which have been constructed recently. For the automated calculation of one-loop amplitudes with more than four external legs, there are the publicly available programs `FormCalc/LoopTools` [4,6] which recently have been extended to 5-point processes [24,25] and the program `CutTools` [26] which is based on a numerical unitarity formalism [11,15,19,20,22]. Further, there are the programs `BlackHat` [27] and `Rocket` [28], relying also on cutting techniques. Concerning the generation of subtraction terms for real radiation, automated tools have become publicly available recently as well [29,30,31,32]. Integral libraries for massive [2,25] as well as infrared divergent [33] scalar integrals also exist.

As already mentioned, a public program for the reduction of tensor integrals so far is available only for infrared-finite integrals and for up to five external legs [4,25]. In this paper we present a program for the numerical reduction of tensor integrals with up to six external legs. In the present version, we focus mainly on massless QCD applications, i.e. processes with massless internal particles. The master integrals are implemented in the code to be valid in all kinematic regions. Infrared divergences are regulated dimensionally, i.e. the loop momenta live in  $n = 4 - 2\epsilon$  dimensions. The output for a specific kinematic

point is a set of six numbers representing the real and imaginary parts of the coefficients of the Laurent series in  $\epsilon$ , i.e. the coefficients of the  $1/\epsilon^2$ ,  $1/\epsilon$  poles and the finite part.

The reduction formalism is valid for massless as well as massive external and internal particles. However, the basis integrals for processes involving *internal* massive particles will be implemented in a forthcoming version. We would like to emphasize that the program can be used not only for tensor reduction, but also to calculate basis integrals, with or without Feynman parameters in the numerator, and therefore is also of interest for calculations where the integral coefficients have been determined by unitarity cut techniques: `golem95` can be used as a library for master integrals.

The paper is organised as follows. In section 2, we review shortly the theoretical background. Section 3 contains a brief summary of the software structure, while section 4 contains a detailed description of the individual components of the program. The installation instructions are given in section 5, and section 6 contains the descriptions of three different test runs: the calculation of a form factor for a rank five five-point function, the calculation of all form factors for rank one six-point functions in one go, and finally the calculation of a full amplitude: the helicity amplitudes for light-by-light scattering. We give an outlook on future versions in section 7 and explain technical details in appendices A.1 and A.2. The code comes with a number of demonstration programs, demonstrating for example the behaviour near a scattering singularity, or the relation to LoopTools notation. All these demonstration programs are listed in Appendix A.3.

## 2 Theoretical background

The program is an implementation of the formalism developed in Ref. [7]. Here we will summarize only its main features relating to the `golem95` program, for further details we refer to [7].

### 2.1 Form Factors

A general one-loop tensor integral of rank  $r$  can be written as

$$I_N^{n, \mu_1 \dots \mu_r}(a_1, \dots, a_r) = \int \frac{d^n k}{i \pi^{n/2}} \frac{q_{a_1}^{\mu_1} \dots q_{a_r}^{\mu_r}}{(q_1^2 - m_1^2 + i\delta) \dots (q_N^2 - m_N^2 + i\delta)} \quad (1)$$

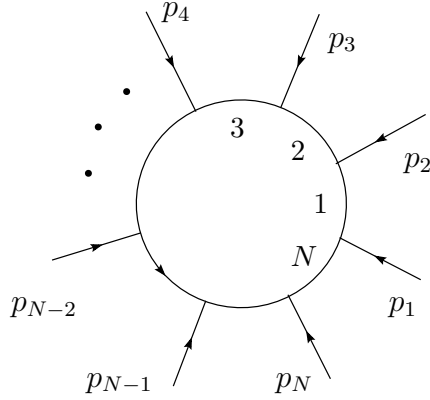


Fig. 1. General  $N$ -point one-loop graph with momentum and propagator labelling.

where  $q_a = k + r_a$ , and  $r_a$  is a combination of external momenta. For the diagram in Fig. 1,  $r_i = \sum_{j=1}^i p_j$ . Our method is defined in  $n = 4 - 2\epsilon$  dimensions and thus is applicable to general scattering processes with arbitrary propagator masses. Taking integrals of the form (1), i.e. with  $q_a^\mu$  instead of just  $k^\mu$  in the numerator, as building blocks has two advantages: first, combinations of loop and external momenta appear naturally in Feynman rules, second, it allows for a formulation of the tensor reduction which manifestly maintains the invariance of the integral under a shift  $k \rightarrow k + r_0$  in the loop momentum. Such a shift can be absorbed into a redefinition of the  $r_j$ ,  $r_j \rightarrow r_j - r_0$ . By setting  $a_1, \dots, a_r = N$ , and using momentum conservation to set  $r_N = 0$ , we can always retrieve the commonly used form

$$I_N^{n, \mu_1 \dots \mu_r}(N, \dots, N) = \int \frac{d^n k}{i \pi^{n/2}} \frac{k^{\mu_1} \dots k^{\mu_r}}{(q_1^2 - m_1^2 + i\delta) \dots (q_N^2 - m_N^2 + i\delta)}. \quad (2)$$

The Lorentz structure of the integral (1) is carried by tensor products of the metric  $g^{\mu\nu}$  and the difference vectors

$$\Delta_{ij}^\mu = r_i^\mu - r_j^\mu, \quad (3)$$

which are shift invariant. Therefore, tensor integrals are expressible by linear combinations of such Lorentz tensors and *form factors*  $A_{l_1 \dots l_r}^{N,r}$ ,  $B_{l_1 \dots l_r}^{N,r}$ ,  $C_{\dots}^{N,r}$ , defined by

$$\begin{aligned} I_N^{n, \mu_1 \dots \mu_r}(a_1, \dots, a_r; S) = & \sum_{j_1 \dots j_r \in S} \left[ \Delta_{j_1 \dots j_r}^\cdot \right]_{\{a_1 \dots a_r\}}^{\{\mu_1 \dots \mu_r\}} A_{j_1 \dots j_r}^{N,r}(S) \\ & + \sum_{j_1 \dots j_{r-2} \in S} \left[ g^\cdot \Delta_{j_1 \dots j_{r-2}}^\cdot \right]_{\{a_1 \dots a_r\}}^{\{\mu_1 \dots \mu_r\}} B_{j_1 \dots j_{r-2}}^{N,r}(S) \end{aligned}$$

$$+ \sum_{j_1 \dots j_{r-4} \in S} \left[ g \dots g \Delta_{j_1} \dots \Delta_{j_{r-4}} \right]_{\{a_1 \dots a_r\}}^{\{\mu_1 \dots \mu_r\}} C_{j_1 \dots j_{r-4}}^{N,r}(S) \quad (4)$$

where  $[\dots]_{\{a_1 \dots a_r\}}^{\{\mu_1 \dots \mu_r\}}$  denotes the distribution of the  $r$  Lorentz indices  $\mu_i$ , and momentum labels  $a_i$  to the vectors  $\Delta_{j_{a_i}}^{\mu_i}$  in all distinguishable ways.  $S$  denotes an ordered set of propagator labels, related to the kinematic matrix  $\mathcal{S}$ , defined by

$$\mathcal{S}_{ij} = (r_i - r_j)^2 - m_i^2 - m_j^2 \quad ; \quad i, j \in \{1, \dots, N\} . \quad (5)$$

There is a one-to-one correspondence between  $\mathcal{S}_{ij}$  and the set  $S = \{1, \dots, N\}$ . We recall that standard form factor representations can be simply obtained by replacing  $a_j = N$  for all  $j$ , together with  $r_N = 0$ . This also shows that the form factors do *not* depend on the introduction of the difference vectors  $\Delta_{ij}^\mu$ . The form factors are shift invariant by themselves. Therefore the program `golem95` can be used without ever introducing difference vectors, if the user prefers not to do so.

Due to the fact that for  $N \geq 5$ , four linearly independent external vectors form a basis of Minkowski space, the tensor reduction for  $N \geq 6$  can be done in such a way that only form factors for  $N \leq 5$  are needed. Therefore, the Lorentz structure of  $(N > 5)$ -point rank  $r$  tensor integrals does not require the introduction of additional factors of  $g^{\mu\nu}$  as compared to the  $N = 5$  case, only additional external vectors appear. We note that for  $N = 5$ , one could already express the metric by external momenta, but this would introduce inverse Gram determinants. In [34], it is shown that all tensor five-point functions can be reduced to some basis integrals without generating higher dimensional five-point functions. In [7], a formal proof of this fact can be found, as well as a reduction method which avoids both inverse Gram determinants and spurious higher dimensional five-point functions. A method where inverse Gram determinants in the reduction from five-point to four-point integrals are absent is also presented in Ref. [9].

The form factors are linear combinations of reduction coefficients and basis<sup>1</sup> integrals, where our basis integrals are not necessarily scalar integrals, as explained in section 2.3. The reduction coefficients are derived from the kinematic matrices  $\mathcal{S}$ , where we define

$$b_i = \sum_{k \in S} \mathcal{S}_{ki}^{-1} , \quad B = \sum_{i \in S} b_i . \quad (6)$$

---

<sup>1</sup> We call them “basis integrals ” because they are the endpoints of our reduction, although they do not form a basis in the mathematical sense.

The quantity  $B$  is related to the Gram determinant by

$$B \det \mathcal{S} = (-1)^{N+1} \det G . \quad (7)$$

The form factors are all given explicitly in [7]. As an example, a rank two pentagon integral is represented as

$$\begin{aligned} I_5^{n, \mu_1 \mu_2}(a_1, a_2; S) &= \sum_{l_1, l_2 \in S} \Delta_{l_1 a_1}^{\mu_1} \Delta_{l_2 a_2}^{\mu_2} A_{l_1 l_2}^{5,2}(S) + g^{\mu_1 \mu_2} B^{5,2}(S) \\ B^{5,2}(S) &= -\frac{1}{2} \sum_{j \in S} b_j I_4^{n+2}(S \setminus \{j\}) \\ A_{l_1 l_2}^{5,2}(S) &= \sum_{j \in S} \left( \mathcal{S}_{j l_1}^{-1} b_{l_2} + \mathcal{S}_{j l_2}^{-1} b_{l_1} - 2 \mathcal{S}_{l_1 l_2}^{-1} b_j + b_j \mathcal{S}_{l_1 l_2}^{\{j\}-1} \right) I_4^{n+2}(S \setminus \{j\}) \\ &\quad + \frac{1}{2} \sum_{j \in S} \sum_{k \in S \setminus \{j\}} \left[ \mathcal{S}_{j l_2}^{-1} \mathcal{S}_{k l_1}^{\{j\}-1} + \mathcal{S}_{j l_1}^{-1} \mathcal{S}_{k l_2}^{\{j\}-1} \right] I_3^n(S \setminus \{j, k\}) \end{aligned} \quad (8)$$

and it is the form factors like  $A_{l_1 l_2}^{5,2}(S), B^{5,2}(S)$  which are implemented in `golem95`.

The program `golem95` can be used for amplitude calculations in several ways: One approach, which aims to avoid tensor integrals of high rank, is to cancel the reducible numerators of an expression before interfacing to `golem95` to calculate the irreducible tensor integrals. However, as all form factors for maximal rank (in a renormalisable gauge) are implemented, the expression for an amplitude can also be interfaced to `golem95` without performing any cancellations between numerators and propagators. This has the advantage that the algebraic manipulations to do are minimal, and that even the Dirac traces, which often appear as coefficients of the form factors, can be done numerically.

## 2.2 Feynman parameter representations

The `golem95` program uses the fact that tensor integrals are related to Feynman parameter integrals with Feynman parameters in the numerator. The basic object is the set  $S$ , containing the labels of the propagators which define the integral. A scalar integral, after Feynman parametrisation, can be written as

$$I_N^n(S) = (-1)^N \Gamma\left(N - \frac{n}{2}\right) \int \prod_{i=1}^N dz_i \delta\left(1 - \sum_{l=1}^N z_l\right) \left(R^2\right)^{\frac{n}{2}-N}$$

$$R^2 = -\frac{1}{2} \sum_{i,j=1}^N z_i \mathcal{S}_{ij} z_j - i\delta . \quad (9)$$

In general, a one-loop  $N$ -point amplitude will contain  $N$ -point integrals as well as  $(N-1), (N-2), \dots, (N-M)$ -point integrals with tree graphs attached to some of the external legs of the loop integral. The latter are characterised by the omission (“pinch”) of some propagators (say  $j_1, \dots, j_m$ ) of the “maximal” one loop  $N$ -point graph, and therefore correspond to a subset of  $S$  where certain propagator labels are missing,  $S \setminus \{j_1, \dots, j_m\}$ . The program `golem95` is based on this concept of sets characterising the integrals.

The general relation between tensor integrals and parameter integrals with Feynman parameters in the numerator is well known [35,36,37,38]

$$I_N^{n, \mu_1 \dots \mu_r} (a_1, \dots, a_r; S) = (-1)^r \sum_{m=0}^{\lfloor r/2 \rfloor} \left(-\frac{1}{2}\right)^m \sum_{j_1 \dots j_{r-2m}=1}^N \left[ (g^{\cdot\cdot})^{\otimes m} \Delta_{j_1} \dots \Delta_{j_{r-2m}} \right]_{\{a_1 \dots a_r\}}^{\{\mu_1 \dots \mu_r\}} I_N^{n+2m} (j_1 \dots, j_{r-2m}; S) , \quad (10)$$

where  $I_N^{n+2m} (j_1 \dots, j_{r-2m}; S)$  is an integral with Feynman parameters in the numerator.  $\lfloor r/2 \rfloor$  stands for the nearest integer less or equal to  $r/2$  and the symbol  $\otimes m$  indicates that  $m$  powers of the metric tensor are present. Feynman parameter integrals corresponding to diagrams where propagators  $j_1, \dots, j_m$  are pinched with respect to the “maximal” topology can be defined as

$$I_N^n (j_1, \dots, j_r; S \setminus \{l_1, \dots, l_m\}) = (-1)^N \Gamma(N - \frac{n}{2}) \int \prod_{i=1}^N dz_i \delta(1 - \sum_{k=1}^N z_k) \delta(z_{l_1}) \dots \delta(z_{l_m}) z_{j_1} \dots z_{j_r} (R^2)^{n/2-N} . \quad (11)$$

### 2.3 Basis integrals

The basis integrals, i.e. the endpoints of our reduction, are 4-point functions in 6 dimensions  $I_4^6$ , which are IR and UV finite, UV divergent 4-point functions in  $n+4$  dimensions, and various 2-point and 3-point functions, some of the latter with Feynman parameters in the numerator. This provides us with a very convenient separation of IR/UV divergences, as the IR poles are exclusively contained in the triangle functions. Explicitly, our reduction basis is given by integrals of the type

$$\begin{aligned}
I_3^n(j_1, \dots, j_r) &= -\Gamma\left(3 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^3 dz_i \delta\left(1 - \sum_{l=1}^3 z_l\right) \frac{z_{j_1} \dots z_{j_r}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{3-n/2}}, \\
I_3^{n+2}(j_1) &= -\Gamma\left(2 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^3 dz_i \delta\left(1 - \sum_{l=1}^3 z_l\right) \frac{z_{j_1}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{2-n/2}}, \\
I_4^{n+2}(j_1, \dots, j_r) &= \Gamma\left(3 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^4 dz_i \delta\left(1 - \sum_{l=1}^4 z_l\right) \frac{z_{j_1} \dots z_{j_r}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{3-n/2}}, \\
I_4^{n+4}(j_1) &= \Gamma\left(2 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^4 dz_i \delta\left(1 - \sum_{l=1}^4 z_l\right) \frac{z_{j_1}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{2-n/2}},
\end{aligned} \tag{12}$$

where  $r^{\max} = 3$ , as well as  $I_3^n, I_3^{n+2}, I_4^{n+2}, I_4^{n+4}$  with no Feynman parameters in the numerator, and two-point functions.

Note that  $I_3^{n+2}$  and  $I_4^{n+4}$  are UV divergent, while  $I_3^n$  can be IR divergent. In the code, the integrals are represented as arrays containing the coefficients of their Laurent expansion in  $\epsilon = (4 - n)/2$ .

Further reduction of these integrals to scalar basis integrals (i.e. integrals with no Feynman parameters in the numerator) introduces factors of  $1/B$ , i.e. inverse Gram determinants. A particular feature of `golem95` is the fact that the above integrals are *not* reduced to scalar basis integrals in cases where  $B$  becomes small, thus avoiding problems with small inverse determinants. In these cases, the above integrals are evaluated numerically. As  $B = (-1)^{N+1} \det(G)/\det(\mathcal{S})$  is a dimensionful quantity, the switch to the numerical evaluation of the basis integrals is implemented such that the value of the dimensionless parameter  $\hat{B}$  is tested, where

$$\hat{B} = B \times (\text{largest entry of } \mathcal{S}). \tag{13}$$

If  $\hat{B} > \hat{B}^{\text{cut}}$ , the reduction is performed, else the program switches to the direct numerical evaluation of the integral. The default value is  $\hat{B}^{\text{cut}} = 0.005$ . A major improvement with respect to the numerical evaluation method used in [7] is the following: while in [7] the numerical evaluation of box integrals was based on three-dimensional parameter representations, we use a certain one-dimensional parameter representation here, obtained after performing two integrations analytically, as outlined in Appendix A.2. In this way one can use deterministic integration routines, leading to a fast and precise numerical evaluation. This has been done for box integrals with up to three off-shell legs and all triangle integrals. The relative error to be achieved in the numerical integration has been set to the default value  $10^{-8}$ . If this precision has not been reached, the program will write a message to the file `error.txt`. In

some cases, calculating in double precision Fortran may not be sufficient. The code is designed such that it can be compiled in quadruple precision as well.

We would like to emphasize that the program also can be used as a library for master integrals with massless internal particles. For example, the scalar box integrals in  $n$  dimensions, with up to four off-shell external legs, can be calculated by just calling the form factor  $A^{4,0}$ . Depending on the kinematics, the program will call the appropriate box type automatically. The scalar box integrals in  $n + 2$  and  $n + 4$  dimensions are related to the form factors by  $B^{4,2} = -I_4^{n+2}/2$ ,  $C^{4,4} = I_4^{n+4}/4$ , analogous for  $N = 3$ .

### 3 Overview of the software structure

The structure of the `golem95` program is the following: There are four main directories:

- (1) **src**: the source files of the program
- (2) **demos**: some programs for demonstration
- (3) **doc**: documentation which has been created with `robodoc` [39]
- (4) **test**: supplements the demonstration programs, containing files to produce form factors with user-defined kinematics. The user can specify the rank, numerator, numerical point etc. via a steering file.

### 4 Description of the individual software components

Here we give a short summary of the contents of the individual modules. A detailed description of the usage, dependencies and output of each module is given at the beginning of each file of the program and can also be read in *html* format by loading the file `masterindex.html` from the subdirectory `doc` into the browser and following the various links.

The program is written in Fortran95 and is downwards compatible to Fortran90.

The directory `src` contains the subdirectories

- **form\_factor**: contains five modules to compute the form factors for two-point to six-point functions:  
`form_factor_2p.f90`, `form_factor_3p.f90`, `form_factor_4p.f90`,  
`form_factor_5p.f90`, `form_factor_6p.f90`.
- **integrals**: contains the subdirectories `four_point`, `three_point`, `two_point`.

- four\_point:** contains six modules to compute the four-point functions with  $p_i^2 \neq 0$  holding for four, three, two, one or none of the external legs: `function_4p1m.f90`, `function_4p2m_opp.f90`, `function_4p2m_adj.f90`, `function_4p3m.f90`, `function_4p4m.f90`, `generic_function_4p.f90`.
- three\_point:** contains six modules to compute the three-point functions with three, two or one external legs off-shell: `function_3p1m.f90`, `function_3p2m.f90`, `function_3p3m.f90`, `generic_function_3p.f90`, `mod_h0.f90`, `mod_hf.f90`, `mod_he.f90`.
- two\_point:** contains one module to compute the two-point functions: `generic_function_2p.f90`.
- **kinematic:** contains two modules to compute the matrix  $\mathcal{S}$  and its inverse and to compute the reduction coefficients  $b_i$ : `matrice_s.f90`, `inverse_matrice.f90`.  
The definition of  $\hat{B}$  (see eq. (13)) is contained in `matrice_s.f90`.
  - **module:** contains auxiliary functions/subroutines and the definition of some default parameters:  
The file `parametre.f90` contains the parameters defining the switch between the reduction down to scalar basis integrals (which are implemented in analytic form) versus the numerical evaluation of integrals (with or without Feynman parameters in the numerator), as explained in section 2.3. The default value for  $\hat{B}$  for three-point as well as four-point functions has been set to 0.005. The other default parameters for the numerical integration are also fixed in `parametre.f90`. Further, there is a switch to calculate the rational parts of amplitudes only. The default is `tot` to calculate the complete form factors. If `tot` is replaced by `rat`, only the rational parts will be calculated.  
The auxiliary functions will not all be listed here, we only point to the most important ones:
    - Polylogarithms and other special functions are defined in `z_log.f90`, `z_dilog.f90`, `kronecker.f90`, `constante.f90`.
    - `spinor.f90` contains functions to compute scalar products of four-momenta, spinorial products and totally antisymmetric epsilon tensors.
    - The files `preci_double.f90` and `preci_quad.f90` are needed for the switch between double precision and quadruple precision. The default is double precision. If quadruple precision should be used, one has to define `$precision = "quadruple"` in the file `configure.pl`. Note that quadruple precision is at present only supported by the `ifort` compiler.
    - The file `form_factor_type.f90` defines a type `form_factor` such that form factors, which are arrays of three complex numbers, can be involved in algebraic manipulations.
    - `cache.f90` is used to reserve memory in order to store results for three-point or four-point functions which already have been computed.
  - **numerical:** contains two modules for the numerical integration : `mod_adapt_gauss.f90`, `mod_numeric.f90`.

Concerning the numerical integration, the following features should be pointed out:

- The user can change the integration method for the numerical integration of the one-dimensional parameter integrals by changing the module `numerical_evaluation` in the file `mod_numeric.f90` in the directory `src/numerical`.
- The values for the cuts defining the switch to a one-dimensional numerical integration of the basis integrals are given in `parametre.f90` and can be changed easily by the user.  
*Note:* If the user wants to change the default values defined in `parametre.f90`, it is *not* necessary to recompile the library. If the desired values are defined in the main program, the default values will be overwritten. The command `use parametre` still has to be included in the header of the main program.
- For boxes with 4 off-shell external legs, the expressions for one-dimensional numerical integrations are not worked out in this version. Here the program will always reduce numerically to scalar basis integrals, irrespective of the size of the Gram determinants.

## 5 Installation instructions

The program can be downloaded as a `.tar.gz` archive from the following URL: [http://lappweb.in2p3.fr/lapth/Golem/golem95\\_v1.0.tar.gz](http://lappweb.in2p3.fr/lapth/Golem/golem95_v1.0.tar.gz). The installation instructions given below also can be found in the `Readme` file coming with the code.

To install the `golem95` library, type the following commands:

```
./configure.pl [--install_path=mypath] [--compiler=mycompiler]
make
make install
```

Please note that `mypath` must be the absolute path of the directory where you would like the library to be installed. If no option for `install_path` is given, a subdirectory of the current directory with the name `libgolem` will be created and the library will be installed in this subdirectory.

For example, if you want to put the library into the directory

`/home/myname/lib/libgolem` and use the compiler `g95`, then type:

```
./configure.pl --install_path=/home/myname/lib/libgolem --compiler=g95
make
make install
```

The directory `/home/myname/lib/libgolem` will then contain a collection of files of type `.mod` plus a file named `libgolem.a` which is the `golem95` library.

If no option for the compiler is specified, the installation script will search for fortran 95 compilers installed on your system and will take the first matching compiler found.

The program has been tested with the GNU compilers `g95` and `gfortran`, the intel compiler `ifort`, the dec compiler `f95`, the NAG compiler `f95` and the portland compiler `pgf95`.

## 6 Test run description

The program comes with several demonstration programs located in the subdirectory `demos`. A list of all options contained in the `demos` directory is given in Appendix A.3. We will describe some selected examples in the following.

### 6.1 Rank five five-point form factor

As an example for a test run, we first describe the calculation of a form factor for a rank five 5-point integral,  $A_{j_1 \dots j_5}^{5,5}$ . We choose  $j_i = i$ , i.e.  $z_1 \dots z_5$  in the numerator. Further, we choose the following numerical point (in terms of entries of the kinematic matrix  $\mathcal{S}$  containing the invariants):

$$\mathcal{S} = \begin{pmatrix} 0 & p_2^2 & s_{23} & s_{51} & p_1^2 \\ p_2^2 & 0 & p_3^2 & s_{34} & s_{12} \\ s_{23} & p_3^2 & 0 & p_4^2 & s_{45} \\ s_{51} & s_{34} & p_4^2 & 0 & p_5^2 \\ p_1^2 & s_{12} & s_{45} & p_5^2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -3 & -4 & 0 \\ 0 & 0 & 0 & 6 & 15 \\ -3 & 0 & 0 & 0 & 2 \\ -4 & 6 & 0 & 0 & 0 \\ 0 & 15 & 2 & 0 & 0 \end{pmatrix} \quad (14)$$

These values are already implemented in the file `demo_5point.f90` in the subdirectory `demos`. All the user has to do is the following:

- go to the subdirectory `demos`
- type “perl configure.pl”. The shell will prompt for the choice of the demo to be run:

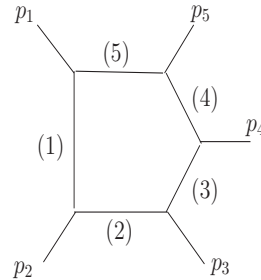
Choose which demo program you want to run:

- 1) three-point functions
- 2) four-point functions

- 3) five-point functions
- 4) six-point functions
- 5) 4-photon helicity amplitudes
- 6) numerical stability demo:  $\det G \rightarrow 0$
- 7) numerical stability demo:  $\det S \rightarrow 0$
- 8) Golem  $\leftrightarrow$  LoopTools conventions
- Choosing option 3 will produce the following output:  
you have chosen option 3: five-point functions  
The Makefile has been created  
Please run:  
make  
./comp.exe
- Running “make” will produce the executable `comp.exe` where the `demo*.f90` files matching the choice above will be compiled automatically. Running `comp.exe` will prompt for the rank of the form factor to be calculated:  
Choose what the program should compute:  
0) form factor for five-point function, rank 0  
1) form factor for five-point function, rank 3 ( $z_1*z_2*z_4$ )  
2) form factor for five-point function, rank 5 ( $z_1*z_2*z_3*z_4*z_5$ )  
3) form factor for diagram with propagator 3 pinched, rank 0  
4) form factor for diagram with propagators 1 and 4 pinched, rank0
- Choosing option 2 will produce the result which will be written to the file `test5point.txt` and looks as follows:

The kinematics is:

$$\begin{aligned}
p_1 + p_2 + p_3 + p_4 + p_5 &= 0 \\
S(1, 3) &= (p_2 + p_3)^2 = -3. \\
S(2, 4) &= (p_3 + p_4)^2 = 6. \\
S(2, 5) &= (p_1 + p_2)^2 = 15. \\
S(3, 5) &= (p_4 + p_5)^2 = 2. \\
S(1, 4) &= (p_1 + p_5)^2 = -4. \\
S(1, 2) &= p_2^2 = 0. \\
S(2, 3) &= p_3^2 = 0. \\
S(3, 4) &= p_4^2 = 0. \\
S(4, 5) &= p_5^2 = 0. \\
S(1, 5) &= p_1^2 = 0.
\end{aligned}$$



A factor  $\Gamma(1 + \epsilon)\Gamma(1 - \epsilon)^2/\Gamma(1 - 2\epsilon) (4\pi \mu^2)^\epsilon$  is factored out from the result.

```

result=      1/ε2 * (0.0000000000E+00 + I* 0.0000000000E+00)
            + 1/ε * (0.0000000000E+00 + I* 0.0000000000E+00)
            + (-.8615520644E-04 + I* 0.1230709464E-03)

CPU time=   7.9990000000000001E-003

```

We recall that we use the integral measure as in eq. (1). The factor  $r_\Gamma = \Gamma(1 + \epsilon)\Gamma(1 - \epsilon)^2/\Gamma(1 - 2\epsilon) (4\pi\mu^2)^\epsilon$  has been extracted from the integrals to comply with the conventions of ref. [7] and the  $\overline{\text{MS}}$  subtraction scheme. Note that it may be advantageous to call `golem95` with rescaled, dimensionless invariants (e.g.  $s_{ij}/\mu^2$ ), for example in cases where most of the invariants have very small numerical values in all kinematic regions.

## 6.2 Calculating all possible numerators at once

In the previous example, the numerical point (and the type of numerator for tensor integrals) has been fixed in the demo programs. If the user would like to give the the numerical point and the Feynman parameters in the numerator as an *input*, he can use the file `param.input` in the subdirectory `test`. This setup also allows to calculate all possible numerators for a certain rank in one go. A typical example looks as follows:

Assume we would like to calculate the form factors for rank one six-point functions for all possible numerators  $z_j, j = 1 \dots 6$ , for the following numerical point ( $p_i = (E_i, x_i, y_i, z_i)$ ):

$$\begin{aligned}
p_1 &= (0.5, 0., 0., 0.5) \\
p_2 &= (0.5, 0., 0., -0.5) \\
p_3 &= (-0.19178191, -0.12741180, -0.08262477, -0.11713105) \\
p_4 &= (-0.33662712, 0.06648281, 0.31893785, 0.08471424) \\
p_5 &= (-0.21604814, 0.20363139, -0.04415762, -0.05710657) \\
p_6 &= -\sum_{i=1}^5 p_i
\end{aligned}$$

To use these momenta, go to the subdirectory `test` and edit<sup>2</sup> the file `momenta.dat`, writing each component of the above momenta into a single line. To calculate an  $N$ -point function, the program will use the first  $N$  momenta found in `momenta.dat` (respectively the momenta file specified in `param.input`).

<sup>2</sup> Alternatively, random momenta can be generated using the program `mom_rambo.f`, adapted from [40] and also contained in the subdirectory `test`.

For  $N = 5$  and  $N = 6$ , it is important that momentum conservation, i.e.  $\sum_{i=1}^N p_i = 0$ , is fulfilled because momentum conservation has been assumed to hold true in the reduction.

To generate results, the user only has to do the following:

- edit the file `param.input` to choose the number of legs, rank and numerator. If only a particular numerator should be calculated, give the labels of Feynman parameters, else put `all` into the numerator field,
- type `perl maketest.pl`.

The program will automatically compile the corresponding functions and run the executable. The following output will be produced:

- (1) a separate output file called `N[nb of legs][rank]_[pt].out` for each individual numerator ([pt] denotes the “label” of a particular numerical point, which can be chosen by the user to distinguish results for different numerical points)
- (2) a file called `N[nb of legs][rank]_[pt].numbers`, where all form factors that have been calculated for a particular rank and number of legs and numerical point are *appended*. For example, if the option `all` has been chosen to calculate all possible combinations of Feynman parameters in the numerator, this file will contain the results for all these numerators. The format is such that it can be read by `Mathematica`, to allow direct comparisons to results obtained from algebraic programs. If the result is  $P_2/\epsilon^2 + P_1/\epsilon + P_0$  for a rank  $r$   $N$ -point form factor of type  $A$ , the output will be a list  $a_N[j_1, \dots, j_r] = \{\mathcal{R}e[P_2], \mathcal{I}m[P_2], \mathcal{R}e[P_1], \mathcal{I}m[P_1], \mathcal{R}e[P_0], \mathcal{I}m[P_0]\}$ .

For example, for rank one six-point functions at the numerical point given above (“pt1”), having chosen `all` in `param.input` to calculate all six possible numerators, the program produces seven output files: `N6rank1zi_pt1.out` for  $i = 1 \dots 6$  and the file `N6rank1_pt1.numbers`. While the files `N6rank1zi_pt1.out` contain, in addition to the result for the particular numerator, also the kinematic point and CPU time information, the file `N6rank1_pt1.numbers` just lists the results. Note that for  $N \geq 5$ , individual form factors are not uniquely defined because the metric tensor  $g_{\mu\nu}$  can be expressed by external momenta, such that individual terms can be shifted between form factors of type  $A$ ,  $B$  or  $C$ .

### 6.3 Calculation of the 4-photon helicity amplitudes

In order to show how `golem95` can be embedded into the calculation of full one-loop amplitudes, we give here the calculation of the light-by-light scattering amplitude in massless QED as a pedagogical example.

This amplitude is defined by six Feynman diagrams where the four photons are attached to a closed fermion loop in all possible ways. Diagrams which differ by the charge flow only lead to the same value, which leaves us with three different topologies defined by the photon orderings 1243, 1234 and 1324, respectively. Each diagram is IR finite and UV divergent. The UV divergence only cancels in the sum of the diagrams. The results for the three independent helicity amplitudes  $++++$ ,  $+++-$ ,  $+-+-$  are well known, see for example [41,42]. For completeness we list the analytic formulae, omitting the irrelevant phases

$$\begin{aligned} \mathcal{A}^{++++} &= 8 \quad , \quad \mathcal{A}^{+++-} = -8 \quad , \\ \mathcal{A}^{+-+-} &= -8 \left[ 1 + \frac{t-u}{s} \log\left(\frac{t}{u}\right) + \frac{t^2+u^2}{2s^2} \left( \log\left(\frac{t}{u}\right)^2 + \pi^2 \right) \right] \quad . \end{aligned} \quad (15)$$

The analytic expressions in terms of form factors and Mandelstam invariants which are given in the demo program `demo_4photon.f90` were obtained as follows. After working out the trace of gamma matrices one finds for each graph a polynomial in scalar products of polarization vectors,  $\varepsilon_j$ , external momenta  $p_j$  and the  $D = 4 - 2\epsilon$  dimensional loop momentum  $k$ . All reducible scalar products, i.e. those which can be written in terms of inverse propagators, were cancelled directly. The remaining expressions, containing only irreducible scalar products, are proportional to tensor integrals which are transformed to form factors using eq. (4). Each form factor now has scalar coefficients containing polarisation vectors and external momenta, i.e.  $\varepsilon_i \cdot \varepsilon_j$ ,  $\varepsilon_i \cdot p_j$ ,  $s = 2p_1 \cdot p_2$ ,  $t = 2p_2 \cdot p_3$  and  $u = 2p_1 \cdot p_3$ , where we defined all external momenta as incoming. Using spinor helicity methods one can map these coefficients to polynomials in the Mandelstam variables  $s$ ,  $t$  and  $u = -s - t$ . Choosing reference momenta  $p_2$ ,  $p_1$ ,  $p_4$ ,  $p_3$  for the polarization vectors  $\varepsilon_1$ ,  $\varepsilon_2$ ,  $\varepsilon_3$ ,  $\varepsilon_4$  respectively, one easily can show the following relations [43] relevant for the  $++++$  amplitude

$$\begin{aligned} \varepsilon_1^+ \cdot \varepsilon_1^+ &= -\frac{2s}{tu} \varepsilon_1^+ \cdot p_3 \varepsilon_2^+ \cdot p_3 \quad , \quad \varepsilon_1^+ \cdot \varepsilon_3^+ = \frac{2}{t} \varepsilon_1^+ \cdot p_4 \varepsilon_3^+ \cdot p_1 \quad , \\ \varepsilon_1^+ \cdot \varepsilon_4^+ &= \frac{2}{u} \varepsilon_1^+ \cdot p_3 \varepsilon_4^+ \cdot p_1 \quad , \quad \varepsilon_2^+ \cdot \varepsilon_3^+ = \frac{2s}{u} \varepsilon_2^+ \cdot p_4 \varepsilon_3^+ \cdot p_2 \quad , \\ \varepsilon_2^+ \cdot \varepsilon_4^+ &= \frac{2s}{t} \varepsilon_2^+ \cdot p_3 \varepsilon_4^+ \cdot p_2 \quad , \quad \varepsilon_3^+ \cdot \varepsilon_4^+ = -\frac{2s}{tu} \varepsilon_3^+ \cdot p_1 \varepsilon_4^+ \cdot p_1 \quad , \\ \varepsilon_1^+ \cdot p_3 \varepsilon_2^+ \cdot p_3 \varepsilon_3^+ \cdot p_1 \varepsilon_4^+ \cdot p_1 &= \left(\frac{tu}{2s}\right)^2 \frac{[21][43]}{\langle 12 \rangle \langle 34 \rangle} \quad . \end{aligned} \quad (16)$$

The phase factor in the last line is irrelevant for observables and thus can be dropped. All coefficients of the form factors of the  $++++$  amplitude are now rational polynomials in  $s$ ,  $t$  and  $u = -s - t$ . For the  $+++-$  amplitude one needs instead of eq. (16)

$$\begin{aligned}
\varepsilon_1^+ \cdot \varepsilon_4^- &= \frac{2}{t} \varepsilon_1^+ \cdot p_4 \varepsilon_4^+ \cdot p_1, \quad \varepsilon_2^+ \cdot \varepsilon_4^- = \frac{2}{u} \varepsilon_2^+ \cdot p_4 \varepsilon_4^+ \cdot p_2, \\
\varepsilon_3^+ \cdot \varepsilon_4^- &= 0, \\
\varepsilon_1^+ \cdot p_3 \varepsilon_2^+ \cdot p_3 \varepsilon_3^+ \cdot p_1 \varepsilon_4^- \cdot p_1 &= \left(\frac{tu}{2s}\right)^2 \frac{[21]\langle 14 \rangle [31]}{\langle 12 \rangle [41] \langle 13 \rangle}
\end{aligned} \tag{17}$$

and for the  $++--$  amplitude

$$\begin{aligned}
\varepsilon_1^+ \cdot \varepsilon_3^- &= \frac{2}{u} \varepsilon_1^+ \cdot p_3 \varepsilon_3^+ \cdot p_1, \quad \varepsilon_1^+ \cdot \varepsilon_4^- = \frac{2}{t} \varepsilon_1^+ \cdot p_4 \varepsilon_4^+ \cdot p_1, \\
\varepsilon_2^+ \cdot \varepsilon_3^- &= \frac{2}{t} \varepsilon_2^+ \cdot p_3 \varepsilon_3^+ \cdot p_2, \quad \varepsilon_3^- \cdot \varepsilon_4^- = -\frac{2s}{tu} \varepsilon_3^- \cdot p_1 \varepsilon_4^- \cdot p_1, \\
\varepsilon_1^+ \cdot p_3 \varepsilon_2^+ \cdot p_3 \varepsilon_3^- \cdot p_1 \varepsilon_4^- \cdot p_1 &= \left(\frac{tu}{2s}\right)^2 \frac{[21]\langle 34 \rangle}{\langle 12 \rangle [34]}.
\end{aligned} \tag{18}$$

These relations define the different coefficients of the form factors present in the file `demo_4photon.f90` which evaluates the four photon amplitude<sup>3</sup>. The form factors for each momentum ordering have to be evaluated only once. We have compared our numerical result with the well-known results for these amplitudes and find perfect agreement.

The program `demo_4photon.f90` can be used as a guideline how to express any amplitude with massless loops in terms of form factors and scalar coefficients before evaluating it with `golem95`.

## 7 Conclusions and Outlook

We have presented the Fortran 95 program `golem95` for the numerical evaluation of tensor integrals up to rank six six-point functions. The program is based on a form factor representation of tensor integrals and performs the reduction to a certain set of basis integrals numerically. The basis integrals are implemented in analytic form. If during the reduction process an inverse determinant becomes small, the program switches to a numerical evaluation of the (tensor-)integral without further reduction, thus avoiding small denominators. The numerical evaluation is based on one-dimensional parameter integral representations for most of the basis integrals, allowing for a fast and precise numerical integration.

---

<sup>3</sup> We note that the three helicity amplitudes can be evaluated in a much simpler way. By applying spinor helicity methods at an earlier stage, one can achieve a representation without any tensor four-point function. The given representation should only illustrate a generic form factor representation of an amplitude.

The results are given as a set of three complex numbers representing the coefficients of the Laurent series in the dimensional regularisation parameter  $\epsilon$ , i.e. the coefficients of the  $1/\epsilon^2$ ,  $1/\epsilon$  poles and the finite part.

The program can also be used as a library for master integrals (including infrared divergent ones), as the form factors with no Feynman parameter labels directly correspond to scalar integrals. In the current version, master integrals with massive *internal* particles are not implemented yet. They will be available in a forthcoming version. There is no restriction on the number of massive *external* legs.

A future version will also combine the `golem95` code for the form factor evaluation with a code for the generation of amplitudes, thus moving towards a full automatisisation of the calculation of one-loop amplitudes.

## Acknowledgements

We would like to thank A. Guffanti and G. Sanguinetti for collaboration at an earlier stage of this work. TB, GH and TR would like to thank the LAPTH for hospitality while part of this work was carried out. This research was supported by the UK Science and Technology Facilities Council (STFC) and the Scottish Universities Physics Alliance (SUPA).

## A Appendices

### A.1 Landau singularities

Besides the spurious appearance of powers of inverse Gram determinants caused by the decomposition on scalar integrals in the reduction process, which can be avoided e.g. with the method advocated here and in [7], another source of problems in the numerical evaluation of scattering amplitudes may be caused by the occurrence of *actual* kinematic singularities, the so-called Landau singularities [44]. The latter may appear in some diagrams contributing to the considered amplitude whenever the determinant of the kinematic matrix  $\mathcal{S}$  associated with these diagrams - or with reduced diagrams obtained by one or several pinches - vanishes. Typical cases of such singularities are threshold singularities in loop calculations with internal and external masses, collinear and infra-red singularities with massless internal and external lines. Another type is the one of scattering singularities [1,45], for which  $(\det G) \rightarrow 0$

and  $\det S$  becomes proportional to  $(\det G)^2$ , such that both vanish simultaneously. The occurrence of these particular cases of vanishing Gram determinants should not be confused with the spurious ones. Individual diagrams lead to infinities at such kinematic configurations, where a mass singularity and a scattering singularity coincide.

In addition, it should be noted that for a given diagram, the reduction algorithm breaks down<sup>4</sup> at such a scattering singularity, as inferred from the relation  $B \propto \det G / \det S \rightarrow \infty$ . As one combines the diagrams to scattering amplitudes, gauge cancellations may occur analytically, which in general reduces the degree of singularity as compared to individual diagrams, or even make the singularity bounded, as e.g. observed in the 6 photon amplitudes<sup>5</sup> [1,47]. On the other hand, the numerical combination of the singularities from separate diagrams is expected to be problematic, and leads to instabilities even in cases of expected finiteness. Note that this problem is known to all methods based on the reduction of diagrams, so it is not specific to our reduction formalism. We note that the problem of large numerical cancellations is inherent to any method based on the reduction to scalar master integrals like  $I_3^n, I_4^n$ , as the latter may become linearly dependent near such singularities.

Depending on the inclusiveness of the observable to be calculated, and the degree of the singularity, possible cures could be to resort to multiple precision in some vicinity of the kinematic singularities, and/or place a hole in the phase space around the singularity together with a smooth interpolation over it. Certainly, in specific cases, when the observable would be controlled by the singularity, such methods would be inadequate.

## A.2 One-dimensional integral representations

In this appendix we will derive representations of IR finite box-and triangle integrals as one-dimensional Feynman parameter integrals. These representations have the advantage that they can be integrated numerically in a very fast and precise way using deterministic numerical integration routines. This approach is similar to the one in [48] where one parameter integration has been carried out analytically. The program switches to this numerical evalu-

---

<sup>4</sup> In the example of double parton scattering for  $2(\text{massless}) \rightarrow 2$  massive legs with no internal mass, it can be checked that for the four leg diagram with two opposite masses, the equations determining  $B$  and  $b_4$  in the notations of ref. [7] have no solution, because  $(\delta v) \cdot H \cdot (\delta v) = 0$ , hence the equation for  $B$  becomes  $0 \times B = 1$  which has obviously no solution. For a discussion of the reduction in exceptional kinematic configurations see also [46].

<sup>5</sup> *Singularity* means *non-analyticity*; the latter can be either infinite - integrable or not - or bounded.

ation if  $\hat{B}$  becomes smaller than a value defined in `module/parametre.f90` (the default is 0.005).

### A.2.1 Four-point functions

Our starting point are the higher dimensional four-point functions  $I_4^{n+2}, I_4^{n+4}$  given by

$$I_4^{n+2}(j_1, \dots, j_r) = \Gamma\left(3 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^4 dz_i \delta\left(1 - \sum_{l=1}^4 z_l\right) \frac{z_{j_1} \dots z_{j_r}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{3-n/2}},$$

$$I_4^{n+4}(j_1) = \Gamma\left(2 - \frac{n}{2}\right) \int_0^1 \prod_{i=1}^4 dz_i \delta\left(1 - \sum_{l=1}^4 z_l\right) \frac{z_{j_1}}{\left(-\frac{1}{2} z \cdot \mathcal{S} \cdot z - i\delta\right)^{2-n/2}},$$

The reduction of these integrals to integrals with no Feynman parameters in the numerator introduces inverse Gram determinants. Therefore it can be advantageous to evaluate these integrals without further reduction. To do so, we proceed as follows:

First, to get rid of the  $\delta$  distribution, we make the following change of variables:

$$z_1 = w(1-x), \quad z_2 = wxyz, \quad z_3 = wxy(1-z), \quad z_4 = wx(1-y) \quad (\text{A.1})$$

Now, instead of computing directly the three-dimensional integrals numerically as proposed in [7], we perform analytically the integration over  $x$  and  $y$  and integrate numerically over the leftover variable  $z$ , using an adaptive Gauss-Kronrod method [49].

For the cases treated in the `golem95` library (no internal masses), the  $x$  and  $y$  integration for the six- and eight-dimensional four-point functions can be computed using two basis integrals:

$$\int_0^1 dx \frac{x^n}{A+Bx} = J(n, A, B), \quad (\text{A.2})$$

$$\int_0^1 dx x^n \ln(A+Bx) = K(n, A, B) \quad (\text{A.3})$$

which obey to the following relations:

$$J(n, A, B) = \frac{1}{nB} - \frac{A}{B} J(n-1, A, B) \quad (\text{A.4})$$

$$J(0, A, B) = \frac{\ln(A+B) - \ln(A)}{B} \quad (\text{A.5})$$

$$K(n, A, B) = \frac{(A+B) \ln(A+B) - n A K(n-1, A, B)}{(n+1) B} - \frac{1}{(n+1)^2} \quad (\text{A.6})$$

$$K(0, A, B) = \frac{(A+B) \ln(A+B) - A \ln(A) - B}{B} \quad (\text{A.7})$$

Here we assume that  $B \neq 0$ ; if  $B = 0$  the integrations are trivial. When the two first integrations have been done, we are left with the  $z$  integration.

To explain how we proceed, we treat the case of the six-dimensional three-mass four-point function as an example. After integration over  $x$  and  $y$ , we are left with the following structure:

$$I = \frac{-h \ln(h) + e \ln(e)}{f g} + \frac{h \ln(h) - c \ln(c)}{f d} \quad (\text{A.8})$$

with

$$\begin{aligned} c &= z \mathcal{S}_{12} + (1-z) \mathcal{S}_{13} \\ f &= z (\mathcal{S}_{24} - \mathcal{S}_{12}) + (1-z) (\mathcal{S}_{34} - \mathcal{S}_{13}) \\ g &= z (1-z) \mathcal{S}_{23} - z \mathcal{S}_{24} - (1-z) \mathcal{S}_{34} \\ d &= z (1-z) \mathcal{S}_{23} - z \mathcal{S}_{12} - (1-z) \mathcal{S}_{13} \\ e &= z \mathcal{S}_{24} + (1-z) \mathcal{S}_{34} \\ h &= z (1-z) \mathcal{S}_{23} \end{aligned} \quad (\text{A.9})$$

where  $\mathcal{S}_{ij}$  are the  $\mathcal{S}$ -matrix elements, they must be understood as  $\mathcal{S}_{ij} + i \delta$ .

The first thing to note is that  $I$  has no poles. All six-dimensional four-point functions are infrared finite, and the UV pole of the eight-dimensional four-point functions is contained in the overall  $\Gamma$ -function. Indeed, it is easy to see that:  $g = h - e$ ,  $d = h - c$  and  $f = e - c$ , so when  $g \rightarrow 0$  or  $d \rightarrow 0$  or  $f \rightarrow 0$ , the numerator of  $I$  goes to zero. To compute the  $z$  integral numerically, we use a contour deformation: we complexify the  $z$  variable

$$z = u - i \epsilon g(u) \quad (\text{A.10})$$

i.e. we have to compute the following integrals:

$$\int_0^1 dz f(z) = \int_0^1 du C f(u - i \epsilon g(u)) \quad (\text{A.11})$$

where  $C$  is the jacobian of the transformation :  $C = 1 - i \epsilon dg/du$  and  $\epsilon = \pm 1$ . The function  $g$  has the following properties:  $g(0) = g(1) = 0$  and  $g(u) > 0$  for  $u \in [0, 1]$ . For practical applications, we took  $g(u) = u(1 - u)$ . As the numerator of  $I$  contains some logarithms, some care has to be taken in order to avoid a clash between the cut of the logarithm and the contour. To analyse that, let us consider the following example:

$$E = \int_0^1 dz \frac{\ln(a + bz + i s_1 \lambda)}{c + dz + i s_2 \lambda} \quad (\text{A.12})$$

with  $a, b, c$  and  $d \in \mathbb{R}$ ,  $s_1, s_2 = \pm 1$  and  $\lambda > 0$ . Making the change of variable (A.10) , we get:

$$E = \int_0^1 du C \frac{\ln(a + bu + i (s_1 \lambda - b \epsilon g(u)))}{c + du + i (s_2 \lambda - d \epsilon g(u))} \quad (\text{A.13})$$

By choosing  $\epsilon = -s_1 \text{sign}(b)$ , the imaginary part of the argument of the logarithm will be constant and have the sign of  $s_1$ . This choice of  $\epsilon$  defines the contour but the important point is that by varying  $u$  (walking on the contour) the cut of the logarithm is never crossed. The pole is located at:

$$z_0 = -\frac{c}{d} - i \frac{s_2}{d} \lambda \quad (\text{A.14})$$

Using the Cauchy theorem, we arrive at the following relation:

$$\int_0^1 dz f(z) = \int_0^1 du C \frac{\ln(a + bu + i (s_1 \lambda - b \epsilon g(u)))}{c + du + i (s_2 \lambda - d \epsilon g(u))} - 2i\pi R \epsilon \Theta\left(1 + \frac{c}{d}\right) \Theta\left(-\frac{c}{d}\right) \delta_{\text{sign}(s_2/d)}^{\text{sign}(\epsilon)} \quad (\text{A.15})$$

where  $R$  is the residue of  $f(z)$  at  $z = z_0$ . This is the way we proceed to compute numerically the two terms of eq. (A.8). We compute the two terms separately despite the fact that each term has a pole when  $f \rightarrow 0$  while the sum does not, because they contain two kinds of logarithms ( $\ln(e)$  and  $\ln(c)$ ), and there is no reason that the choice for  $\epsilon$  for one term prevents the contour from crossing the cut for the other term.

For the case where there are Feynman parameters in the numerator, everything works like the preceding example: we always split the integrand of the  $z$  integration into two pieces (each piece having more terms than the scalar case) by separating the two kinds of logarithms. For the other types of four-point functions, we proceed in an analogous way.

### A.2.2 Three-mass three-point functions

In the case of the three-point functions with three off-shell legs, after making a change of variables of type (A.1), we are left with two-dimensional integrals. One parameter is integrated out analytically using (A.2), the remaining integral is computed numerically, using the same techniques as for the four-point functions.

### A.2.3 Two-mass three-point functions

The two mass three-point functions are written in terms of functions  $H_i$  [7], which are defined such that in the numerically problematic case where  $X \rightarrow Y$ , their evaluation is numerically stable. The functions  $H_0$ ,  $H_1$ ,  $H_2$ ,  $H_3$  and  $H_4$  are given by:

$$H_0(X, \alpha) = \frac{\bar{X}^\alpha}{X} \quad (\text{A.16})$$

$$H_1(X, Y, \alpha) = \frac{\bar{X}^\alpha - \bar{Y}^\alpha}{X - Y} \quad (\text{A.17})$$

$$H_2(X, Y, \alpha) = \frac{\bar{Y}^\alpha}{Y - X} + \frac{1}{1 + \alpha} \frac{\bar{Y}^{1+\alpha} - \bar{X}^{1+\alpha}}{(Y - X)^2} \quad (\text{A.18})$$

$$H_3(X, Y, \alpha) = \frac{\bar{Y}^\alpha}{Y - X} + \frac{2}{1 + \alpha} \frac{\bar{Y}^{1+\alpha}}{(Y - X)^2} + \frac{2}{(1 + \alpha)(2 + \alpha)} \frac{\bar{Y}^{2+\alpha} - \bar{X}^{2+\alpha}}{(Y - X)^3} \quad (\text{A.19})$$

$$H_4(X, Y, \alpha) = \frac{\bar{Y}^\alpha}{Y - X} + \frac{3}{1 + \alpha} \frac{\bar{Y}^{1+\alpha}}{(Y - X)^2} + \frac{6}{(1 + \alpha)(2 + \alpha)} \frac{\bar{Y}^{2+\alpha}}{(Y - X)^3} + \frac{6}{(1 + \alpha)(2 + \alpha)(3 + \alpha)} \frac{\bar{Y}^{3+\alpha} - \bar{X}^{3+\alpha}}{(Y - X)^4} \quad (\text{A.20})$$

$$\bar{X} = -X - i\delta$$

For each function  $H_i(X, Y, \epsilon)$ , one can define

$$H_i(X, Y, \epsilon) = \epsilon H_{E_i}(X, Y) + \frac{\epsilon^2}{2} H_{F_i}(X, Y), \quad (\text{A.21})$$

and one can show that

$$H_{E_n}(X, Y) = \int_0^1 dz z^{(n-1)} \frac{1}{z\bar{X} + (1-z)\bar{Y}} \quad (\text{A.22})$$

$$H_{F_n}(X, Y) = \int_0^1 dz z^{(n-1)} \frac{\ln(z \bar{X} + (1-z) \bar{Y})}{z \bar{X} + (1-z) \bar{Y}} \quad (\text{A.23})$$

From this definition, it is easy to show that

$$H_{E_n}(X, Y) = \frac{1}{X - Y} \left( \frac{1}{n-1} - Y H_{E_{n-1}}(X, Y) \right) \quad (\text{A.24})$$

The equations (A.22) and (A.23) are used to compute numerically the functions  $H_{E_n}$  and  $H_{F_n}$ .

### A.3 Contents of the demonstration programs

The demo programs calculate the following examples, listed also in the file `DemoContents` in the subdirectory `demos`:

- (1) three-point functions
- (2) four-point functions
- (3) five-point functions
- (4) six-point functions
- (5) calculation of 4-photon helicity amplitudes
- (6) numerical stability demo:  $\det G \rightarrow 0$
- (7) numerical stability demo:  $\det S \rightarrow 0$
- (8) Golem  $\leftrightarrow$  LoopTools conventions

The items above contain the following options:

- Three-point functions:

- (1) one off-shell leg
- (2) two off-shell legs
- (3) three off-shell legs

For each of the three options above, one can choose to calculate:

- (a) scalar three-point function in  $n$  dimensions
- (b) three-point function in  $n$  dimensions with one Feynman parameter  $(z_1)$  in the numerator
- (c) three-point function in  $n$  dimensions with two Feynman parameters  $(z_1 z_2)$
- (d) three-point function in  $n$  dimensions with three Feynman parameters  $(z_1^2 z_3)$
- (e) scalar three-point function in  $n+2$  dimensions
- (f) three-point function in  $n+2$  dimensions with one Feynman parameter  $(z_2)$

- Four-point functions:

- (1) no off-shell leg
- (2) one off-shell leg
- (3) two opposite off-shell legs
- (4) two adjacent off-shell legs
- (5) three off-shell legs
- (6) four off-shell legs

For each of the five options above, one can choose to calculate:

- (a) scalar four-point function in  $n$  dimensions
  - (b) four-point function in  $n$  dimensions with one Feynman parameter ( $z_1$ )
  - (c) four-point function in  $n$  dimensions with two Feynman parameters ( $z_1 z_4$ )
  - (d) four-point function in  $n$  dimensions with three Feynman parameters ( $z_1^2 z_3$ )
  - (e) four-point function in  $n$  dimensions with four Feynman parameters ( $z_1 z_2 z_3 z_4$ )
  - (f) scalar four-point function in  $n+2$  dimensions
  - (g) four-point function in  $n+2$  dimensions with two Feynman parameters ( $z_1 z_2$ )
  - (h) scalar four-point function in  $n+4$  dimensions
- Five-point functions:
    - (1) form factor for five-point function, rank 0
    - (2) form factor for five-point function, rank 3 ( $z_1 z_2 z_4$  in numerator)
    - (3) form factor for five-point function, rank 5 ( $z_1 z_2 z_3 z_4 z_5$  in numerator)
    - (4) form factor for a pinched 5-point diagram (propagator 3 missing), rank 0
    - (5) form factor for a doubly pinched 5-point diagram (propagators 1 and 4 missing), rank 0
  - Six-point functions:
    - (1) form factor for six-point function, rank 0
    - (2) form factor for six-point function, rank 4 ( $z_1^2 z_2 z_3$  in numerator)
    - (3) form factor A5 for pinched diagram, propagator 3 missing, rank 0
    - (4) form factor for double pinched diagram, propagators 2,5 missing, rank 0
    - (5) form factor for triple pinched diagram, propagators 2,4,6 missing, rank 0
  - Calculation of 4-photon helicity amplitudes:
 

the purpose of this example is to demonstrate how to use `golem95` for the calculation of full amplitudes. It calculates three different helicity configurations of the on-shell 4-photon amplitude for a certain kinematic point.
  - Numerical stability demo:  $\det G \rightarrow 0$ :
 

calculates a rank three four-point function (in 6 dimensions) in a region where  $|B| = \det G / \det S$  becomes small, i.e. where a representation based on the reduction to scalar integrals would fail. The Feynman parameters in the numerator are  $z_1 z_2^2$ . The example follows closely the one described in section 7.2 of [7] and is also described in the `golem95` manuscript: The program makes 30 iterations where  $B = -\det G / \det S$  becomes smaller in each iteration. The results for real and imaginary parts of  $I_4^6(z_1 z_2^2)$  are written to the file `demo_detG.dat` as a function of  $x$ , where  $|B| \propto x^2$  for small

*x.* The files `plotDetG_Re.gp` and `plotDetG_Im.gp` can be used to plot the result with gnuplot by `load 'plotDetG_Re/Im.gp'` . One can see from the plots that The file `demo_detG.txt` contains the details of the kinematics for each iteration.

- Numerical stability demo:  $\det S \rightarrow 0$ :  
tests the rank 5 five-point tensor coefficient  $A^{5,5}(1, 1, 1, 1, 1)$  with respect to its behaviour when a sub-determinant  $\det S \sim (\det G)^2 \rightarrow 0$ . The results for real and imaginary parts of the  $\epsilon^0$  part of  $A^{5,5}$  are written to the file `demo_a55_dets_sing.dat` as a function of the transverse momentum of particle 5 and can be plotted with gnuplot by `load 'plot_demo_A55.gp'`.
- Relation between Golem output and LoopTools format:  
produces Golem output for four-point functions up to rank four and gives the relation to LoopTools conventions. If LoopTools is linked, the lines containing the call of LoopTools functions can be uncommented to produce LoopTools output in parallel.

## References

- [1] Z. Bern et al. The NLO multileg working group: summary report. *0803.0494 [hep-ph]*, 2008.
- [2] G. J. van Oldenborgh and J. A. M. Vermaseren. New Algorithms for One Loop Integrals. *Z. Phys.*, C46:425–438, 1990.
- [3] R. Mertig, M. Bohm, and Ansgar Denner. FEYN CALC: Computer algebraic calculation of Feynman amplitudes. *Comput. Phys. Commun.*, 64:345–359, 1991.
- [4] T. Hahn and M. Perez-Victoria. Automatized one-loop calculations in four and D dimensions. *Comput. Phys. Commun.*, 118:153–165, 1999.
- [5] F. Yuasa et al. Automatic computation of cross sections in HEP: Status of GRACE system. *Prog. Theor. Phys. Suppl.*, 138:18–23, 2000.
- [6] Thomas Hahn. Generating Feynman diagrams and amplitudes with FeynArts 3. *Comput. Phys. Commun.*, 140:418–431, 2001.
- [7] T. Binoth, J. Ph. Guillet, G. Heinrich, E. Pilon, and C. Schubert. An algebraic/numerical formalism for one-loop multi-leg amplitudes. *JHEP*, 10:015, 2005.
- [8] Ansgar Denner, S. Dittmaier, M. Roth, and L. H. Wieders. Electroweak corrections to charged-current e+e- to 4 fermion processes: Technical details and further results. *Nucl. Phys.*, B724:247–294, 2005.
- [9] Ansgar Denner and S. Dittmaier. Reduction schemes for one-loop tensor integrals. *Nucl. Phys.*, B734:62–115, 2006.
- [10] R. Keith Ellis, W. T. Giele, and G. Zanderighi. Semi-numerical evaluation of one-loop corrections. *Phys. Rev.*, D73:014027, 2006.
- [11] Giovanni Ossola, Costas G. Papadopoulos, and Roberto Pittau. Reducing full one-loop amplitudes to scalar integrals at the integrand level. *Nucl. Phys.*, B763:147–169, 2007.
- [12] T. Binoth, J. Ph. Guillet, and G. Heinrich. Algebraic evaluation of rational polynomials in one-loop amplitudes. *JHEP*, 02:013, 2007.
- [13] Charalampos Anastasiou, Ruth Britto, Bo Feng, Zoltan Kunszt, and Pierpaolo Mastrolia. Unitarity cuts and reduction to master integrals in d dimensions for one-loop amplitudes. *JHEP*, 03:111, 2007.
- [14] Zvi Bern, Lance J. Dixon, and David A. Kosower. On-Shell Methods in Perturbative QCD. *Annals Phys.*, 322:1587–1634, 2007.
- [15] R. K. Ellis, W. T. Giele, and Z. Kunszt. A Numerical Unitarity Formalism for Evaluating One-Loop Amplitudes. *JHEP*, 03:003, 2008.

- [16] William B. Kilgore. One-loop Integral Coefficients from Generalized Unitarity. *0711.5015 [hep-ph]*, 2007.
- [17] Ruth Britto, Bo Feng, and Pierpaolo Mastrolia. Closed-Form Decomposition of One-Loop Massive Amplitudes. *Phys. Rev.*, D78:025031, 2008.
- [18] Ruth Britto, Bo Feng, and Gang Yang. Complete One-Loop Amplitudes With Massless Propagators. *JHEP*, 09:089, 2008.
- [19] Walter T. Giele, Zoltan Kunszt, and Kirill Melnikov. Full one-loop amplitudes from tree amplitudes. *JHEP*, 04:049, 2008.
- [20] P. Mastrolia, G. Ossola, C. G. Papadopoulos, and R. Pittau. Optimizing the Reduction of One-Loop Amplitudes. *JHEP*, 06:030, 2008.
- [21] Stefano Catani, Tanju Gleisberg, Frank Krauss, German Rodrigo, and Jan-Christopher Winter. From loops to trees by-passing Feynman’s theorem. *0804.3170 [hep-ph]*, 2008.
- [22] R. Keith Ellis, Walter T. Giele, Zoltan Kunszt, and Kirill Melnikov. Masses, fermions and generalized D-dimensional unitarity. *0806.3467 [hep-ph]*, 2008.
- [23] E. W. Nigel Glover, Pierpaolo Mastrolia, and Ciaran Williams. One-loop phi-MHV amplitudes using the unitarity bootstrap: the general helicity case. *JHEP*, 08:017, 2008.
- [24] T. Hahn and J. I. Illana. Excursions into FeynArts and FormCalc. *Nucl. Phys. Proc. Suppl.*, 160:101–105, 2006.
- [25] Thomas Hahn and Michael Rauch. News from FormCalc and LoopTools. *Nucl. Phys. Proc. Suppl.*, 157:236–240, 2006.
- [26] Giovanni Ossola, Costas G. Papadopoulos, and Roberto Pittau. CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes. *JHEP*, 03:042, 2008.
- [27] C. F. Berger et al. An Automated Implementation of On-Shell Methods for One- Loop Amplitudes. *Phys. Rev.*, D78:036003, 2008.
- [28] W. T. Giele and G. Zanderighi. On the Numerical Evaluation of One-Loop Amplitudes: the Gluonic Case. *0805.2152 [hep-ph]*, 2008.
- [29] Tanju Gleisberg and Frank Krauss. Automating dipole subtraction for QCD NLO calculations. *Eur. Phys. J.*, C53:501–523, 2008.
- [30] Michael H. Seymour and Christopher Tevlin. TeVJet: A general framework for the calculation of jet observables in NLO QCD. *0803.2231 [hep-ph]*, 2008.
- [31] K. Hasegawa, S. Moch, and P. Uwer. Automating dipole subtraction. *0807.3701 [hep-ph]*, 2008.
- [32] Rikkert Frederix, Thomas Gehrmann, and Nicolas Greiner. Automation of the Dipole Subtraction Method in MadGraph/MadEvent. *0808.2128 [hep-ph]*, 2008.

- [33] R. Keith Ellis and Giulia Zanderighi. Scalar one-loop integrals for QCD. *JHEP*, 02:002, 2008.
- [34] Zvi Bern, Lance J. Dixon, and David A. Kosower. Dimensionally regulated pentagon integrals. *Nucl. Phys.*, B412:751–816, 1994.
- [35] Andrei I. Davydychev. A Simple formula for reducing Feynman diagrams to scalar integrals. *Phys. Lett.*, B263:107–111, 1991.
- [36] O. V. Tarasov. Connection between Feynman integrals having different values of the space-time dimension. *Phys. Rev.*, D54:6479–6490, 1996.
- [37] Zvi Bern, Lance J. Dixon, and David A. Kosower. Dimensionally regulated one loop integrals. *Phys. Lett.*, B302:299–308, 1993.
- [38] T. Binoth, J. P. Guillet, and G. Heinrich. Reduction formalism for dimensionally regulated one-loop N-point integrals. *Nucl. Phys.*, B572:361–386, 2000.
- [39] <http://www.xs4all.nl/~rfsber/robo/robodoc.html>.
- [40] R. Kleiss, W. James Stirling, and S. D. Ellis. A new Monte Carlo treatment of multiparticle phase space at high energies. *Comput. Phys. Commun.*, 40:359, 1986.
- [41] T. Binoth, E. W. Nigel Glover, P. Marquard, and J. J. van der Bij. Two-loop corrections to light-by-light scattering in supersymmetric QED. *JHEP*, 05:060, 2002.
- [42] Christophe Bernicot. Light-light amplitude from generalized unitarity in massive QED. *0804.0749 [hep-ph]*, 2008.
- [43] T. Binoth, J. P. Guillet, and F. Mahmoudi. A compact representation of the gamma gamma g g g to 0 amplitude. *JHEP*, 02:057, 2004.
- [44] R. J. Eden, P. V. Landshoff, David I. Olive, and J. C. Polkinghorne. *The Analytic S-Matrix*. Cambridge University Press, 1966.
- [45] Zoltan Nagy and Davison E. Soper. Numerical integration of one-loop Feynman diagrams for N- photon amplitudes. *Phys. Rev.*, D74:093006, 2006.
- [46] G. Duplancic and B. Nizic. Reduction method for dimensionally regulated one-loop N- point Feynman integrals. *Eur. Phys. J.*, C35:105–118, 2004.
- [47] C. Bernicot and J. Ph. Guillet. Six-Photon Amplitudes in Scalar QED. *JHEP*, 01:059, 2008.
- [48] T. Binoth, G. Heinrich, and N. Kauer. A numerical evaluation of the scalar hexagon integral in the physical region. *Nucl. Phys.*, B654:277–300, 2003.
- [49] W.H. Press, S.A. Teukolsky, W.T. Vetterlin, and B.P. Flannery. *Numerical Recipes*. Cambridge University Press, 3rd edition, 2007.