Eur. Phys. J. C (2011) 71: 1516
DOI 10.1140/epjc/s10052-010-1516-y

THE EUROPEAN
PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

Numerical evaluation of tensor Feynman integrals in Euclidean kinematics

J. Gluza[1,a], **K. Kajda**[1], **T. Riemann**[2,b], **V. Yundin**[2,c]

[1]Institute of Physics, University of Silesia, Uniwersytecka 4, 40007 Katowice, Poland
[2]Deutsches Elektronen-Synchrotron, DESY, Platanenallee 6, 15738 Zeuthen, Germany

Received: 26 October 2010 / Revised: 22 November 2010 / Published online: 6 January 2011

Abstract For the investigation of higher order Feynman integrals, potentially with tensor structure, it is highly desirable to have numerical methods and automated tools for dedicated, but sufficiently 'simple' numerical approaches. We elaborate two algorithms for this purpose which may be applied in the Euclidean kinematical region and in $d = 4 - 2\epsilon$ dimensions. One method uses Mellin–Barnes representations for the Feynman parameter representation of multi-loop Feynman integrals with arbitrary tensor rank. Our Mathematica package `AMBRE` has been extended for that purpose, and together with the packages `MB` (M. Czakon) or `MBresolve` (A.V. Smirnov and V.A. Smirnov) one may perform automatically a numerical evaluation of planar tensor Feynman integrals. Alternatively, one may apply sector decomposition to planar and non-planar multi-loop ϵ-expanded Feynman integrals with arbitrary tensor rank. We automatized the preparations of Feynman integrals for an immediate application of the package `sector_decomposition` (C. Bogner and S. Weinzierl) so that one has to give only a proper definition of propagators and numerators. The efficiency of the two implementations, based on Mellin–Barnes representations and sector decompositions, is compared. The computational packages are publicly available.

1 Introduction

One goal of present calculations in particle physics is reaching higher and higher precision in perturbation theory. Since Feynman's time we have rules allowing to build automatically the necessary mathematical objects. For a long term these were just the Feynman diagrams, but recently other approaches like the unitarity based perturbative approach get rising attention. The problem remains how to evaluate the complicated integrals originating in the perturbative picture of elementary particle interactions. Physics predicts these integrals to be defined in Minkowskian space-time. The integrals are multi-dimensional complex functions with a complicated singularity structure. Further, they have to be regularized by notions like d-dimensional space-time. In recent years ambitious projects appeared where techniques are used to calculate physical processes in highly automatized ways [1–3].

Here we focus on the calculation of Feynman integrals in *Euclidean space-time*. Though they are, in general, not the ultimate physical objects, their knowledge is very useful. First, if we know them analytically, analytic continuation gives a way to transform them to the Minkowskian region, if needed so. Moreover, if we solve, somehow, Feynman integrals involved in a given physical process analytically, we can use the knowledge in the Euclidean region to check these solutions numerically. This has been done in numerous cases at the 2-loop level (e.g. massive Bhabha scattering [4–9], QCD calculations [10–12]), but also at higher loop levels or in more general context [13–16]). For structural studies of quantum field theory, the numerical calculation of Feynman integrals in Euclidean space-time has also been proven to be useful, e.g. for the check of some conjectures in super-Yang–Mills theories [17, 18].

In the present article we describe two publicly available computational tools based on Mathematica which facilitate numerical calculations of the kind described.

The first tool is the extended version v.2.0 of the `AMBRE` program [19] which generates Mellin–Barnes (MB) representations for Feynman integrals. We discuss the construction of MB-integrals with numerators of arbitrary rank R for L-loop cases. We also shortly report on additional features of older versions (v.1.1 and v.1.2) which were released after publication of [20, 21]. We explicitely work out a variety of non-trivial numerical examples. For this purpose, `AMBRE`

[a] e-mail: janusz.gluza@us.edu.pl
[b] e-mail: tord.riemann@desy.de
[c] e-mail: Valery.Yundin@desy.de

is being combined with the Mathematica packages MB [22] and MBresolve [23].

The second tool is the Mathematica interface CSectors to the Ginac package sector_decomposition [24]. The package sector_decomposition uses the sector decomposition method to calculate general polynomial structures which are present in calculations of Feynman integrals. In the spirit of AMBRE, we perform in CSectors for a given Feynman integral the automatic calculation of the characteristic F and U polynomials, add some normalizing factors consisting of Gamma functions and, finally, use a general formula for multi-loop tensorial Feynman parameterisations. The result is a user-friendly interface to sector_decomposition for the specific purpose of tensor Feynman integral calculations. What remains to be done by the user of CSectors/sector_decomposition is writing in a proper way the definitions of propagators (plus numerators, if they are present). Further, optional algorithmic strategies have to be chosen which are part of the core program [24]. This is the stage of automatization reached also with AMBRE.

The article is organized as follows. In Sect. 2 we prepare expressions for the general multi-loop Feynman integral. Their evaluation based on Mellin–Barnes representations with AMBRE is described in Sect. 3. In Sect. 4 some details on sector decomposition and of using CSectors are discussed, and Sect. 5 contains numerical examples and few comparisons of the two approaches. It follows the Summary. In the Appendix we list the most important Mathematica functions of AMBRE and options for CSectors.

2 Definitions

Comprehensive overviews of the presentations of Feynman integrals may be found e.g. in [25, 26]. Here, we repeat some basic formulae in order to define our notations. The L-loop Feynman integral in $d=4-2\epsilon$ dimensions with N internal lines with momenta q_i and masses m_i, and E external legs with momenta p_e is defined here as follows:

$$G_L\big[T_R(k)\big] = \frac{1}{(i\pi^{d/2})^L} \times \int \frac{d^dk_1\cdots d^dk_L \;\; T_R(k)}{(q_1^2-m_1^2)^{\nu_1}\cdots(q_i^2-m_i^2)^{\nu_i}\cdots(q_N^2-m_N^2)^{\nu_N}}. \tag{2.1}$$

The numerator $T_R(k)$ is a tensor of rank R in the integration variables:

$$T_R(k) = k_{l_1}^{\mu_1}\cdots k_{l_R}^{\mu_R}, \tag{2.2}$$

and

$$D_i = q_i^2 - m_i^2 = \left[\sum_{l=1}^{L}\alpha_{il}k_l - P_i\right]^2 - m_i^2, \tag{2.3}$$

$$P_i = \sum_{e=1}^{E}\beta_{ie}p_e. \tag{2.4}$$

We allow for arbitrary indices ν_i, the powers of propagator functions D_i.

Next, the momentum integrals are replaced by Feynman parameter integrals:

$$\begin{aligned} G_L\big[T_R(k)\big] = {} & \frac{(-1)^{N_\nu}}{\Gamma(\nu_1)\cdots\Gamma(\nu_N)} \\ & \times \int \prod_{i=1}^{N} dx_i\, x_i^{\nu_i-1}\delta\left(1-\sum_{i=1}^{N}x_i\right) \\ & \times \frac{U^{N_\nu-\frac{d}{2}(L+1)-R}}{F^{N_\nu-\frac{d}{2}L}} \\ & \times \sum_{r=0}^{R}\frac{1}{(-2)^{\frac{r}{2}}}\Gamma(N_\nu - dL/2 - r/2) \\ & \times F^{\frac{r}{2}}\left\{\mathscr{A}_r^{[\mu_1,\ldots,\mu_r}\mathscr{P}_{R-r}^{\mu_{r+1},\ldots,\mu_R]}\right\}, \end{aligned} \tag{2.5}$$

where $N_\nu = \sum_{i=1}^{N}\nu_i$.

The two functions U and F are characteristics of the topology of the Feynman integral. One may derive them from

$$\mathscr{N} = \sum_{i=1}^{N} x_i\, D_i \;\equiv\; k(M_L)k - 2kQ + J, \tag{2.6}$$

where

$$(M_L)_{ll'} = \sum_{i=1}^{N}\alpha_{il}\alpha_{il'}x_i, \tag{2.7}$$

$$Q_l = \sum_{i=1}^{N}\alpha_{il}P_i x_i, \tag{2.8}$$

$$J = \sum_{i=1}^{N}(P_i^2 - m_i^2)x_i; \tag{2.9}$$

namely:

$$U_L(x) = \det(M_L), \tag{2.10}$$

$$F_L(x) = -\det(M_L)\; J + Q\tilde{M}_L Q, \tag{2.11}$$

with

$$\tilde{M}_L = \det(M_L)(M_L)^{-1}. \tag{2.12}$$

The object $\mathscr{A}_r\mathscr{P}_{R-r}$ contains the tensor structure due to its two elements:

$$\mathscr{A}_0 = 1, \tag{2.13}$$

$$\mathscr{A}_r = 0 \quad \text{for odd } r > 0, \tag{2.14}$$

$$\mathscr{A}_r^{\mu_1\cdots\mu_r} = \tilde{g}^{\mu_1\mu_2}\cdots\tilde{g}^{\mu_{r-1}\mu_r}, \tag{2.15}$$

and

$$\mathscr{P}_0 = 1, \tag{2.16}$$

$$\mathscr{P}_r^{\mu_1\cdots\mu_r} = \mathscr{P}^{\mu_1}\cdots\mathscr{P}^{\mu_r}, \tag{2.17}$$

where we left out in the notations the indices related to the loop numbering, because they are fixed by (2.2) when the Lorentz indices are defined:

$$\tilde{g}^{\mu_1\mu_2} \equiv \left(\tilde{M}_L^{-1}\right)_{l_1l_2} g^{\mu_1\mu_2}, \tag{2.18}$$

$$\mathscr{P}^{\mu_i} \equiv \sum_{l=1}^{L} \left(\tilde{M}_L\right)_{l_il}\, Q_l^{\mu_i}. \tag{2.19}$$

The product $\{\mathscr{A}_r^{[\mu_1,\ldots,\mu_r}\mathscr{P}_{R-r}^{\mu_{r+1},\ldots,\mu_R]}\}$ is completely symmetrized in its Lorentz indices; take as an example $\mathscr{A}_2\mathscr{P}_2$:

$$\begin{aligned}
&\mathscr{A}_2^{[\mu\nu}\mathscr{P}_2^{\lambda\rho]} \\
&\quad = \mathscr{A}_2^{\mu\nu}\mathscr{P}_2^{\lambda\rho} + \mathscr{A}_2^{\mu\lambda}\mathscr{P}_2^{\nu\rho} + \mathscr{A}_2^{\nu\lambda}\mathscr{P}_2^{\mu\rho} \\
&\qquad + \mathscr{A}_2^{\mu\rho}\mathscr{P}_2^{\nu\lambda} + \mathscr{A}_2^{\nu\rho}\mathscr{P}_2^{\mu\lambda} + \mathscr{A}_2^{\lambda\rho}\mathscr{P}_2^{\mu\nu},
\end{aligned} \tag{2.20}$$

and, more explicitely, to e.g. $T(k_1^{\mu_1}k_2^{\mu_2})$ correspond the terms:

$$\mathscr{A}_0\mathscr{P}_2^{\mu_1\mu_2} = P^{\mu_1}P^{\mu_2}, \tag{2.21}$$

and, with a different numerical factor (see (2.5)):

$$\mathscr{A}_2^{\mu_1\mu_2}\mathscr{P}_0^{\mu_1\mu_2} = \tilde{g}^{\mu_1\mu_2}. \tag{2.22}$$

For *one-loop integrals*, the rotation matrix M_1 in the loop momenta becomes trivial, $M_1 = \tilde{M}_1 = 1$, and so also $U_1 = \det(M_1) = 1$, and $F_1(x) = -J + Q^2$.[1] The (2.5) then becomes:

$$\begin{aligned}
G_1(T_R) = {}& \frac{(-1)^{N_\nu}}{\prod_{i=1}^N \Gamma(\nu_i)} \int \prod_{i=1}^N dx_i\, x_i^{\nu_i-1}\delta\left(1 - \sum_{j=1}^N x_j\right) \\
&\times \sum_{r=0}^R \frac{\Gamma(n - \frac{d+r}{2})}{(-2)^{\frac{r}{2}} F^{n-\frac{d+r}{2}}} \{\mathscr{A}_r\mathscr{P}_{R-r}\}^{[\mu_1,\ldots,\mu_R]}.
\end{aligned} \tag{2.23}$$

[1]Sometimes it is useful to rewrite $F_1(x) \to -(\sum x_i)J + Q^2$, which agrees under the integral in (2.5) with $F_1(x)$, but is now a bi-linear function of the x_i.

In case of e.g. $L = 1$, $R = 2$, we get for the sum in (2.5):

$$\begin{aligned}
T\left(k_1^{\mu_1}k_1^{\mu_2}\right) \to {}& \Gamma(N_\nu - d/2) \\
&\times \left[Q^{\mu_1}Q^{\mu_2} - \frac{1}{2(N_\nu - d/2 - 1)} F g^{\mu_1\mu_2}\right].
\end{aligned} \tag{2.24}$$

The general expressions as well as the examples agree with [19]. For one-loop tensors, equivalent expressions are also given in [27].

An important observation is that the Feynman parameter representations for arbitrary Feynman integrals would be just dependent on polynomials in the x_i, i.e. be sums of monomials in the x_i with integer exponents, if there were not the two types of terms $U(x)^{A(L,N,d,R)}$ and $F(x)^{B(L,N,d,r)}$. The functions $U(x)$ and $F(x)$ are such polynomials, but they have non-integer exponents. While, the additional terms arising from the *tensorial* structure of the L-loop Feynman integrals are polynomials in the x_i.[2] The function $U(x) = \sum_n m_n(x_i)$ depends only on monomials $m_n(x_i)$ and is positive semi-definite, while function $F(x) = \sum_{n'}[-s_{n'}]m_{n'}(x_i) + U(x)\sum_j^N x_j m_j^2$ depends also on the kinematical invariants and on the masses of the problem. For Euclidean kinematics, all the $[-s_{n'}] \geq 0$ and also $F(x)$ becomes positive semi-definite [26]. One typical example is the F-function (3.4).

The above formulae may be used for automated evaluations of specific Feynman integrals. For that purpose, one has to develop methods for their proper treatment, and two of them are worked out here.

3 Integrations using Mellin–Barnes representations

Iterated applications of Mellin–Barnes' formula

$$\begin{aligned}
&\frac{1}{[A(s_n)m_n(x) + B(s_{n'})m_{n'}(x)]^a} \\
&\quad = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} d\sigma \left[A(s_n)m_n(x)\right]^\sigma \left[B(s_{n'})m_{n'}(x)\right]^{-a-\sigma} \\
&\qquad \times \frac{\Gamma(a+\sigma)\Gamma(-\sigma)}{\Gamma(a)}
\end{aligned} \tag{3.1}$$

may be used to transform the x-integrand of (2.5) into a sequence of monomials in the x_i, allowing thus to perform the x-integrations applying

$$\int_0^1 \prod_{j=1}^N dx_j\, x_j^{\alpha_j-1}\delta\left(1 - \sum_{i=1}^N x_i\right) = \frac{\prod_{i=1}^N \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^N \alpha_i)}. \tag{3.2}$$

[2]Because (13) of [19] was not sufficiently simplified, this was not evident there for $L > 1$.

The integration contours have to be chosen such that in each integration step the corresponding left poles, typically $\Gamma(a+\sigma)$, are separated by the contour from the right poles, typically $\Gamma(-\sigma)$. One remains with the problem to evaluate multi-dimensional complex Mellin–Barnes integrals.

The publicly available package AMBRE prepares these Mellin–Barnes integrals. Since the first publication of AMBRE in [19], several features has been added to the package. In AMBREv.1.1, the MB-representations can be constructed only for tensor integrals where all momenta of the tensor $T(k_i)$ are multiplied by external momenta, i.e. are part of scalar products. Since AMBREv.1.2 it is foreseen to generate MB-representations for just tensor integrals. Additionally, some new options were added, consult for details on them the webpages [20, 21], where also appropriate examples are documented. One of the options allows to generate Feynman parameter representations without performing the x-integrations, leaving them to be performed by the preferred technique of the user.

Here we focus on AMBRE v.2.0 which generates *tensor* MB-integrals in a *fully* automatic way. Previous versions have the option Fauto which allows for manual manipulations on the F polynomial. Sometimes this is useful and helps to obtain a smaller dimensionality of MB-integrals (see e.g. [19] and the discussion for pentagons there). However, for many real processes with a large number of amplitudes, complete automatization is necessary. A second goal of the present version is a construction of MB-representations for tensorial planar n-loop tensors. Though non-planar topologies can be also obtained directly with the *loop-by-loop method* employed in AMBRE [28], results may come out wrong.[3] We recommend not to use AMBRE without independent checks for non-planar problems with several kinematical scales; they hopefully shall be treated more properly in the future.

In order to include tensor structures of Feynman integrals, we had to modify properly the iterative procedure of [19]. The best way to explain this might be an explicit example, see Fig. 1.

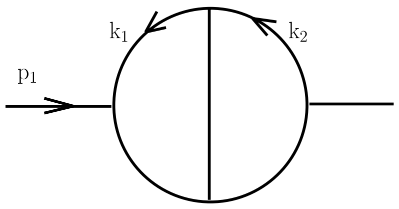

Fig. 1 A simple example for the discussion of the tensor algorithm of AMBRE

[3]We applied AMBRE successfully to e.g. the massless 2-loop vertex first studied in [29], which is a one-scale problem. A typical multi-scale problem is discussed in Sect. 3.2 of [30].

Here all propagators are assumed to be massless and the numerator is of rank $R=3$:

$$\int \frac{(k_1\cdot p)(k_1\cdot p)(k_2\cdot p)}{[k_1^2]^{n_1}[(k_2-k_1)^2]^{n_2}[(k_1+p)^2]^{n_3}[k_2^2]^{n_4}[(k_2+p)^2]^{n_5}} d^dk_1\, d^dk_2. \tag{3.3}$$

The calculation starts by working out the sub-loop integration over k_1, which leads to the following F polynomial:

$$F=\left[-k_2^2\right]x_1x_2+[-s]\,x_1x_3+\left[-(k_2+p)^2\right]x_2x_3. \tag{3.4}$$

According to (2.23), the generated tensor structure for the first sub-loop is an expression with five separate parts; we omit here a normalizing factor and end up with a table of five different integrals:

$$\begin{aligned}&P^{\mu_1}P^{\mu_2}\oplus\tilde{g}^{\mu_1\mu_2}\\ &\quad\to Q^{\mu_1}Q^{\mu_2}\oplus g^{\mu_1\mu_2}\\ &\quad\to \left(k_2^{\mu_1}x_2-p^{\mu_1}x_3\right)\left(k_2^{\mu_2}x_2-p^{\mu_2}x_3\right)\oplus g^{\mu_1\mu_2}\\ &\quad\to \left\{k_2^{\mu_1}k_2^{\mu_2}x_2^2,\ -k_2^{\mu_2}p^{\mu_1}x_2x_3,\ k_2^{\mu_1}p^{\mu_2}x_2x_3,\right.\\ &\qquad\left.p^{\mu_1}p^{\mu_2}x_3^2,\ g^{\mu_1\mu_2}\right\}.\end{aligned} \tag{3.5}$$

Evidently, we will have to perform the second sub-loop integration over k_2 separately for all the above parts because they have different tensor ranks. As a rule, the rank of a given integral in the next step will include higher rank tensors than the original one. The situation after the first momentum integration can be symbolized as follows:

$$\begin{aligned}&\int\frac{d^dk_2\ p_{1\mu_1}p_{1\mu_2}(k_2\cdot p)}{[k_2^2]^{-z_1}[(k_2+p)^2]^{-3+\epsilon+n_1+n_2+n_3+z_1+z_2}}\\ &\quad\times\left\{k_2^{\mu_1}k_2^{\mu_2}\times \mathrm{MB}_1,\ -k_2^{\mu_2}p^{\mu_1}\times\mathrm{MB}_2,\ k_2^{\mu_1}p^{\mu_2}\times\mathrm{MB}_3,\right.\\ &\qquad\left.p^{\mu_1}p^{\mu_2}\times\mathrm{MB}_4,\ g^{\mu_1\mu_2}\times\mathrm{MB}_5\right\},\end{aligned} \tag{3.6}$$

where the expressions MB_i stand for different Mellin–Barnes parts of the net integral. These parts are different due to different Feynman parameters in (3.5). As one explicit example we reproduce MB_1:

$$\begin{aligned}\mathrm{MB}_1={}&(-1)^{2-\epsilon-z_2}(-s)^{z_2}\Gamma(2-\epsilon-n_1-n_2-z_1)\\ &\times\Gamma(-z_1)\Gamma(4-\epsilon-n_1-n_3-z_2)\\ &\times\Gamma(-z_2)\Gamma(n_1+z_1+z_2)\\ &\times\Gamma(-2+\epsilon+n_1+n_2+n_3+z_1+z_2)\\ &/\left[\Gamma(n_1)\Gamma(n_2)\Gamma(6-2\epsilon-n_1-n_2-n_3)\Gamma(n_3)\right].\end{aligned} \tag{3.7}$$

Working with more than two loops, additional iterations can produce further 'fragmentations' of the expression.

In the program, all above steps are hidden to the user. The only action is to define the input object:

```
invariants = {p1^2->s};
MBrepr[{k1*p1,k1*p1,k2*p1},{PR[k1,0,n1]*PR[k2,0,n2]*
        PR[k2-k1,0,n3]*PR[k1+p1,0,n4]*PR[k2+p1,0,n5]},{k1,k2}]
```

The output is:

```
{-(((-1)^(n1+n2+n3+n4+n5)*(-s)^(4-2*eps-n1-n2-n3-n4-n5)*s^3
Gamma[2-eps-n1-n3-z1]*Gamma[-z1]*Gamma[5-eps-n2+z1]
  *Gamma[4-eps-n1-n4-z2]*Gamma[4-2*eps-n1-n3-n4-n5-z1-z2]
  *Gamma[-z2]*Gamma[-4+2*eps+n1+n2+n3+n4+n5+z2]*Gamma[n1+z1+z2]
  *Gamma[-2+eps+n1+n3+n4+z1+z2])/(Gamma[n1]*Gamma[n3]
  *Gamma[6-2*eps-n1-n3-n4]*Gamma[n4]*Gamma[n2-z1]
  *Gamma[9-3*eps-n1-n2-n3-n4-n5-z2]
  *Gamma[-2+eps+n1+n3+n4+n5+z1+z2])),...}
```
(3.8)

The dots stand for the remaining four MB-integrals. The complete expression can be found in the file `MB_SE510m.nb` at the webpage [20, 21].

4 Integrations by sector decomposition

The second approach—performing sector decompositions—transforms the x-integrand into a sequence of expressions where the singularities at $d = 4$, as a function of ϵ ($d = 4 - 2\epsilon$), are separated such that the arising expressions can be smoothly integrated. An appropriate algorithm for the automated computation of the ϵ series of multi-loop integrals has been formulated in [31, 32]. In [24], a publicly available program is described which calculates the Laurent expansion of the following type of parametric integrals in ϵ:

$$\int_{x_j\geq 0} d^n x\, \delta\left(1-\sum_{i=1}^{n} x_i\right)\left(\prod_{i=1}^{n} x_i^{a_i+\epsilon b_i}\right)\prod_{j=1}^{r}[P_j(x)]^{c_j+\epsilon d_j}. \tag{4.1}$$

The a_i, b_i, c_j and d_j are integers, and the P_j are polynomials in the variables $x_1, \ldots, x_n$. The program may handle a product of several polynomials, and it is not required that the polynomials are homogeneous. These features are important for applying (4.1) to our tensorial structures: it allows to calculate $G_L(T)$ (2.5), and some examples are given in [24] for *scalar* integrals.

Here we present an interface which directly calculates (2.5) using as the backbone the already programmed structure (4.1). What is expected by interface from the user is a proper definition of the integral (2.1) and of the kinematical invariants to be used. Formally, it has to be done in the following way:

```
invariants = {invariants as a rules};
DoSector[{numerator},{denominator},
         {internal momenta}][low, des];
```
(4.2)

The first three entries in "`DoSector`" and "`invariants`" in (4.3) have the same form as in `AMBRE`. The "`low`" is the leading singularity of the considered integral. Sometimes it is not easy to determine this power of singularity. In order to be save, we propose to start with one or two powers of ϵ less than being relevant, giving zeroes for these cases. The "`des`" is the highest power of ϵ to which we are going to calculate an integral. One has to be careful because it is not possible to calculate only one ϵ term if it is not the leading singularity: If we want to calculate an ϵ^n term, but if lower terms in ϵ are present, we must start from the leading term. This feature arises because the program calculates the whole Laurent series in ϵ, taking into account also the prefactor $\frac{(-1)^{N_\nu}}{\prod_{i=1}^{N}\Gamma(\nu_i)}$ in (2.5).

Certainly, the most important choice is that of a proper strategy, i.e. the construction of an iterative, terminating sequence of decompositions of the primary sectors (of integration region) into subsectors (of integration region). Primary sectors arise from resolving the δ-function in (2.5), and the integrals in sub-sectors have no ϵ-singularities in the transformed functions U, F. The package offers the terminating strategies A [33, 34], B [35], C [24], and also strategy X [24]. The recommended strategy is C, but when X is terminating, it may come out to be more efficient than C.

The user input for CSectors is extremely simple. Here we reproduce a (slightly shortened) massless 3-loop vertex example with rank $R = 3$; the topology is shown in Fig. 2:

Input:

```
************************ BEGIN *****************************
<< CSectors.m;
x=-11;
invariants={p1*p2->1/2*x,p1*p3->1/2*x,p2*p3->1/2*x,p1^2->x,
            p2^2->0,p3^2->0};
DoSectors[{k2*p2,k3*p1,k3*p1},{PR[k2+k3+p2,0,1]*PR[k1-k3,0,1]*
          PR[k3,0,1]*PR[k1+k2-p1+p2,0,1]*PR[k1,0,1]*PR[k2,0,3]},
          {k1,k2,k3}][-5,0];
************************ END ********************************
```
(4.3)

Output:

```
************************ BEGIN *****************************
CSectors by K.Kajda and V.Yundin ver:1.0
last modified 22 sep 2009

Using strategy C
U & F polynomials:

   U = x4 (x5 x6 + x3 (x5 + x6)) + x1 (x4 x5 + x2
       (x3 + x4 + x5) + x4 x6 + x5 x6 + x3 (x5 + x6)) + x2
       ((x4 + x5) x6 + x3 (x4 + x5 + x6))
   F = 11 (x3 (x2 + x4) + x1 (x3 + x4)) x5 x6

   Q11 = ...
   Q12 = ...
   Q21 = 121*x2*x3^2*x6^2/4
   Q22 = ...

Generating c++ source...Int11...Int12...Int21...Int22...done
Compiling source code...Int11...Int12...Int21...Int22...done
Running binary file.....Int11...Int12...Int21...Int22...done

InputForm=
{-7.622574999999999 - 0.0260407/eps^4
 + 0.04952700000000015/eps^3 - 0.4168788/eps^2 + 0.56955/eps,
 {1.3667737639753552, 2.85804370185272*^-6/eps^4,
  0.00009220935574625821/eps^3, 0.0004811810295896961/eps^2,
  0.006549529654501916/eps}}
************************ END ********************************
```
(4.4)

The functions F and U in (4.4) are calculated with Mathematica, in the same way as in AMBRE. The parameters Q11 … Q22 are polynomials in the x_i which arise due to the numerators in (2.5) for tensor integrals. These terms are select as positive semi-definite pieces Q11 … Q22 in order to satisfy the conditions of the sector decomposition algorithm.

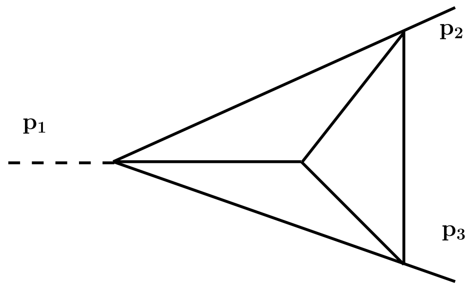

Fig. 2 Topology of the three-loop vertex exemplified in the text

The dots for `Q11`, `Q12`, `Q22` in (4.4) indicate expressions of some length and their explicit form can be found in file `output_mercedes` at the webpages [21, 36].

After "`Generating c++ source`" and "`Compiling source code`" in (4.4), the C++ files are running. They have the structure discussed in [24] and a user can inspect them by switching in the option "`TempFileDelete->False`" in (4.2). Other useful options are listed and described in the Appendix.

By default, the numerical errors of the results are also given. They are calculated by taking the combined error for all the integrals I_i calculated at a given term of the ϵ-expansion:

$$\Delta I_\epsilon = \sqrt{\sum_{i=1}^{A} (\Delta I_i)^2}. \tag{4.5}$$

As it is well known, some of integrals, especially massless ones, can be difficult to integrate numerically. Here the proper choice of a sector decomposition strategy may help [24]. In (4.4), the calculation has been done with optional strategy C. In order to control which ϵ-term is currently calculated, the default option `TempFileDelete->False` produces a log file. For the `Q11` term of the example, it is:

```
>> SE510m_num11.log <<
Order eps^(-2): 0 +/- 0
Order eps^(-1): 0 +/- 0
Order 1: -203.056 +/- 0.00948256
Order eps: 153.241 +/- 0.0721185
```

A list of all options can be obtained with the command `?Options` in Mathematica.

In order to test `CSectors` we have checked many topologies: multi-loop tadpoles and self-energies, three-loop vertices up to rank $R = 5$ and with double dots on propagators (corresponding to setting index $\nu = 3$, because a 'dot' raises the index by one), four-point functions up to rank $R = 4$, and some one-loop five- and six-point functions. For some higher rank tensors we have used Integration-by-Parts decompositions of the Feynman integrals using the computer algebra package `IdSolver`.[4] Some numerical examples of these tests can be found in the file `numerical_checks.nb` at the webpages [21, 36].

5 Numerical results

In Sect. 3 it has been shown how to define Feynman integrals and how to get numerical values for them at chosen kinematical points using the MB-method. For sector decomposition, the same has been discussed in Sect. 4. For scalar integrals it is straightforward to use, together with `AMBRE`, the `MB` package, and to perform the numerical integrations [22]. For tensor integrals, especially with loop order $L > 1$, we have usually many MB-integrals for which the command `MBrules` of `MB` has to be performed in order to find proper integration paths for the MB-integrations. Sometimes, depending on the degree of divergency of the original Feynman integral, this is not possible if only an analytical continuation in ϵ is done. Then, an analytical continuation in one of the indices, may be successful. This has also been automatized. If `MBrules` does not find a valid rule for some $\epsilon \neq 0$, then the power of the first propagator ν_1 is changed, e.g. if $\nu_1 = 1$, then $\nu_1 \to \nu_1 = 1 + \eta$ is applied. If again `MBrules` cannot find a valid rule, ν_2 is treated analogously, and the procedure can be continued until `MBrules` is successful.[5]

We introduced the auxiliary file `MBnum.m` to `MB.m` which realizes this procedure and the subsequent automatized analytic continuation, ϵ-expansion, and numerics. The file may be obtained from the webpages [20, 21].

If an already prepared MB-representation `repr` is available, e.g. in (3.8), it is enough to use the `MBnum` function (see Appendix for details), e.g.:

```
MBanalytic
  =MBnum[repr, -1, {s -> -11}, {n1 -> 1,
           n2 -> 1, n3 -> 1, n4 -> 1,
           n5 -> 1}, 2]
res=MBintegrate[MBanalytic,{s->-11}]
```

This gives the following numerical result (see also file `out_SE510m` at the webpages [20, 21]):

$$\begin{aligned} \text{SEnumMB} &= -7.5625/\epsilon^2 - 20.4506/\epsilon - 178.18 \pm 0.0171936 \\ &\quad + (18.3642 \pm 0.0248465)\epsilon \end{aligned} \tag{5.1}$$

[4] `IdSolver` is an unpublished C++ package performing Feynman integral reductions with the Laporta algorithm. J.G. thanks M. Czakon for the opportunity to use it.

[5] If it happens that an analytical continuation in only one additional parameter η is not sufficient, the program will stop with a proper remark.

The corresponding values returned by `CSectors` are (see also file `output_SE5l0m` at [21, 36]):

```
SEnumSD=
{-178.1927 - 7.56258/eps^2 - 20.4505/eps
  + 18.394000000000005*eps,
{0.011130644628528934, 0.00029563/eps^2,
0.00302109/eps, 0.06629380324170578*eps}
}
```

(5.2)

Another example is two-loop Bhabha scattering. For two-loop 4-point functions, sector decomposition needs a lot of RAM (typically up to few GB) and also of computing time. In such cases, often the numerical integrations are done faster using the MB-method. Some numerical results for the Bhabha Feynman integral with rank $R = 2$,

$$\mathrm{B1} = \int d^d k_1\, d^d k_2 \frac{(k_1 \cdot p_1)\,(k_2 \cdot p_2)}{(k_1^2 - m^2)(k_1 + p_1)^2[(k_1 + p_1 + p_2)^2 - m^2]} \times \frac{1}{(k_1 - k_2)^2 (k_2^2 - m^2)[(k_2 + p_1 + p_2)^2 - m^2](k_2 + p_1 + p_2 + p_4)^2}, \tag{5.3}$$

corresponding to the topology shown in Fig. 3, can be found in Table 1.

The Mandelstam variables are $s = (p_1 + p_2)^2$ and $t = (p_2 + p_4)^2$. The so-called reducible numerators of a tensor Feynman integral can be contracted with propagators. Apart from speeding up calculations, such contractions have been used to check the implementation of tensor structures into `AMBRE` and `CSectors`. In our example, the relation

$$(k_1 \cdot p_1) = \frac{1}{2}\left[(k_1 + p_1)^{-2} - \left(k_1^2 - m^2\right)^{-1} - 2m^2\right] \tag{5.4}$$

may be used to change the rank $R = 2$ tensor of (5.3) into three tensor integrals of rank $R = 1$ (in addition reducing the number of propagators by one for two of them). In this way, the numerical results given in Table 1 can be cross-checked.

Further, let us present a four-loop self-energy scalar diagram, Fig. 4. With `CSectors` it takes a few hours to calculate the constant term of the ϵ-expansion.[6] As it is a scalar diagram, we can make direct comparisons with `FIESTA2` [37], which is much faster but applies to scalar integrals only. For `AMBRE/MB` the complete calculation takes about two minutes only, which is comparable to using `FIESTA2`.

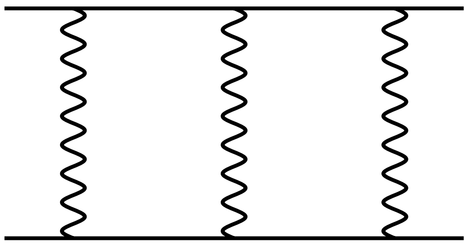

Fig. 3 The planar two-loop Bhabha topology B1

Table 1 Numerical values for the first terms of the ϵ expansion of the massless and massive Bhabha 2-loop double box with tensor rank $R = 2$, defined in (5.3). The topology is shown in Fig. 3. The package `CSectors` was used with strategy X for $m = 0$ and with strategy C for $m \neq 0$

	`AMBRE + MB`	`CSectors, X`	`AMBRE + MB`	`CSectors, C`
	$m = 0$		$m = 1$	
ϵ^0	0.65734(6)	0.659(2)	3.186(6)	3.174(2)
ϵ^{-1}	0.13921(8)	0.1396(5)	1.0383(1)	1.0381(3)
ϵ^{-2}	0.018095	0.01835(9)	0.28817(1)	0.28816(4)
ϵ^{-3}	0.104974	0.10500(2)	–	–
ϵ^{-4}	−0.0217857	−0.021785(3)	–	–
T $[s]$	368	26700 (7.5 h)	945	70220 (19.5 h)

$s = -5, \quad t = -7$

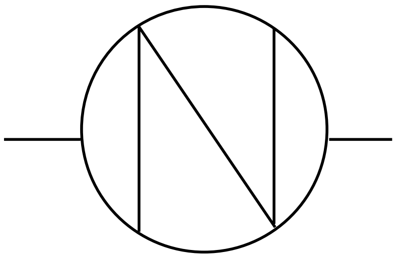

Fig. 4 Four-loop self-energy. For details of its calculation see the files `MB_SE4loop.m`, `MB_SE4loop.out` and `SD_SE4loop.sh`, `SD_SE4loop.out` at the webpages [20, 21, 36]

[6] More precisely, it takes eight ours both with strategies C and X. This and the other examples calculated in this paper have been run on a Xeon personal computer.

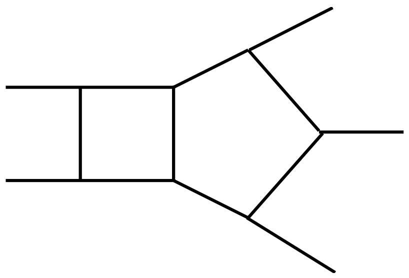

Fig. 5 The pentabox topology PB

Finally, as a more complicated case, let us take a pentabox of rank $R = 3$:

$$\mathrm{PB} = \int d^d l_1\, d^d l_2 \frac{(l_2\cdot k_1)\,(l_2\cdot k_2)\,(l_1\cdot k_5)}{l_1^2(l_1-k_1)^2(l_1-k_1-k_2)^2} \times \frac{1}{(l_1-k_1-k_2-k_3)^2 l_2^2 (l_2-k_5)^2(l_2-k_4-k_5)^2(l_1+l_2)^2}. \tag{5.5}$$

Its topology (called PB here) is shown in Fig. 5.

All external momenta k_i are incoming, $s_{ij} = (k_i + k_j)^2$, $s_{12} = -1, s_{23} = -2, s_{34} = -3, s_{45} = -4, s_{15} = -5$), and the numerical result is:

```
PB=
{-7.431348973217098+0.328125/eps^4
  +0.5340786885498234/eps^3
  -0.44570891627426246/eps^2
  - 3.1058689125651284/eps,                (5.6)
{0.012739775270198513
  +5.560657237081255*10^(-10)/eps^2
  +0.0003572744410650107/eps}}
```

Some caution must be paid to the higher dimensional MB integrals. They can give underestimated errors, taken from the CUBA error output. Whether this happens can be checked switching on the MB.m option `Debug`. The result in (5.6) has been obtained for the following set of `MB` parameters (see the file `MB_PBox.m` at [20, 21]):

```
SetOptions[MBintegrate, Verbose->False,
            PrecisionGoal->10,
            AccuracyGoal->24,
            MaxPoints -> 1000000];
```

As it can be seen in `MB_PBox.m`, the construction of the MB-representation for (5.5) starts with momentum l_2, leading to maximally nine dimensional MB-integrals. Applying Barnes lemmas reduces the integrals to at most seven-dimensional. Starting with the integration over l_1, we get maximally eleven-dimensional integrals. Altogether, the calculation takes about five hours: The generation of the MB-representations needs a couple of minutes, and the rest of the time is needed for analytic continuations and numerical integrations.

Additional instructive examples for using `AMBRE` and `CSectors` may be found at the webpages [20, 21, 36].

6 Summary

The `CSectors` package has been prepared as an interface to the sector decomposition package `sector_decomposition` for the automatic numerical evaluation of tensor Feynman integrals. The `AMBRE` package has been extended for the automatic treatment of tensor structures of multi-loop Feynman integrals.

For `CSectors`, the bottleneck is the numerical evaluation based on `sector_decomposition` and Ginac which, especially for higher rank tensors and higher loop orders, consumes a huge amount of RAM. However, for smaller problems, up to two loops and tensors of moderate rank, the program works well. For more complicated cases we recommend to use the Mellin–Barnes approach. Here there are many possibilities to optimize the way to get numerical results. This can be done at the level of construction of MB-representations (e.g. by a change of order of the integration over internal momenta), then there are different ways of analytic continuation, and finally the Mathematica package `barnesroutines` [38] can be used to try to reduce the dimensionality of the MB-integrals; examples for the latter can be found in [39] and at [20, 21, 36]).

For the near future it is planned to automatize the construction of MB-representations for non-planar Feynman integrals, what deserves to leave the loop-by-loop approach. Further, it is foreseen to build MB-representations for special forms of linear propagators which appear e.g. in heavy quark effective theory (HQET) or in calculations of the QCD static potential.

Acknowledgements The present work is supported in part by the European Community's Marie-Curie Research Training Networks MRTN-CT-2006-035505 'HEPTOOLS' and MRTN-CT-2006-035482 'FLAVIAnet', by the Polish Ministry of Science and High Education from budget for science for years 2010-2013, grant number N N202 102638, and by Sonderforschungsbereich/Transregio 9–03 of Deutsche Forschungsgemeinschaft 'Computergestützte Theoretische Teilchenphysik'.

Appendix

The instructive examples for using `AMBRE` and `CSectors` may be found at the webpages [20, 21, 36].

AMBRE (ver 1.X)

The appropriate loop integral is defined by:

(1) **A list of kinematic invariants**, e.g. **invariants** = **{p1** $*$ **p1** -> **s}**.

(2) **Fullintegral[numerator, propagators, internal momenta, options]**.

invariants must be defined before **Fullintegral**.

The arguments of **Fullintegral** are as follows:

- `numerator:` numerator which can be given in the contracted form, e.g. `{k1*p1,k2*p2}` or in the uncontracted form, e.g. `{k2[mu1],k2[mu2]}`.
- `propagators:` product of propagators of the form `PR[q,m,n1]` $\equiv (q^2 - m^2)^{n_1}$.
- `internal momenta:` list of internal momenta, e.g. `{k1,k2}`.

This version of AMBRE uses a semi-automatic approach when building Mellin–Barnes representation. That methodology is accomplished by the following two functions.

(3) **IntPart[iteration, options]**—prepares a subintegral for a given internal momentum by collecting the related numerator, propagators and integration momentum:

- `iteration:` iteration for which subintegral will be prepared. In practice `IntPart` function must be executed in specific order i.e. firstly `IntPart[1]` then `IntPart[2]` and so.
- `options:`
  - `Text:` it can have two boolean values `True` or `False`. Controls if additional text appears during calculations.

(4) **SubLoop[integral, options]**—determines for the selected subintegral the U and F polynomials and an M–B representation.

- `integral:` this argument must be left as it is.
- `options:`
  - `Text:` as in the `IntPart` function.
  - `Xintegration:` controls whether integration over Feynman parameters is performed or not.

(1)–(4) constitutes basic functions. An additional functions are:

(5) **Fauto[value]**—allows user specified modifications of the F polynomial `fupc`. Must be used after `IntPart` and before `SubLoop`.

- `value:` can be 0 or 1. For the first one user can modify F polynomial. For the latter this possibility is turned off.

(6) **BarnesLemma[representation, number, options]**—function tries to apply Barnes's first or second lemma to a given representation.

- `representation:` M–B representation to be checked.
- `number:` it has two possible values, 1 for the first Barnes lemma and 2 for the second lemma.
- `options:` there are the following two boolean options for this function
  - `Text:` displays or does not display an additional information.
  - `Shifts:` searches for the pairs of two integration variables $z_i + z_j$ and $z_i - z_j$ which, after application of the appropriate shift one of it is cancelled.

(7) **ARint[result, i_]**—in `version 1.2`, it displays the MB-representation number i for Feynman integrals with numerators.

AMBREnLOOP (ver 2.0)

The basic functions of `AMBREnLOOP` are:

(1) **MBrepr[numerator, propagators, internal momenta, options]**.
It returns M–B representation for a given loop diagram. All arguments, except `options`, have the same form as parameters in `version 1.X`.

- `options:`
  - `Text:` displays or not information text.
  - `OptimizedResult:` the final MB representation is written in such a way that Gamma functions are factorized. This option makes an output more condense. However sum of different Gamma functions for a given integral can cause problems if treated as it stands in analytic continuation (finding rules). Optionally it is switched off.
  - `BarnesLemma1:` turns on or off first Barnes lemma checking.
  - `BarnesLemma2:` does the same as above option but for second Barnes lemma

Intermediate output of a given subloop is displayed in a specific format using `INT` function:
`INT[numerator,representation,propagators1,propagators2]`

- `arguments:`
  - `numerator:` it is, for tensor integrals, of the form `{k2[mu1],k2[mu2]}`.
  - `representation:` Mellin–Barnes representation obtained during the calculation of the current subloop/iteration part. It is multiplied by Mellin–Barnes representations which were calculated in the previous step/iteration.
  - `propagators1:` propagators extracted out of the F-polynomial in the current iteration.

– `propagators2`: keeps propagators in the same form as propagators1 does. The propagators2 are independent of the present integration variable but undergo next iteration(s).

(2) **BarnesLemma[representation, number, powers of propagators, options]**—this function works as in the previous versions of `AMBRE`. The only difference is the extra parameter:

- `powers of propagators`:—here a list of powers of propagators which appear in an input loop integral must be given, e.g. `{n1,n2,n3}`.

CSectors

The appropriate loop integral is defined by:

(1) **A list of kinematic invariants**, e.g. **invariants** = **{p1** ∗ **p1 -> s}**.

(2) **DoSectors[numerator, propagators, internal momenta, options][min, max]**

The arguments of this function are exactly the same as in case of **MBrepr** in the `AMBREnLOOP` package. However, in contrast to the `AMBRE` package, numerator can be written only in the uncontracted form.

The package allows to modify its behaviour by adding additional options:

- `SetStrategy`: chooses one of the strategies available in sector decomposition libraries [24].
- `SourceName`: a prefixing for source, binary and log files; the option just allows to choose any name for the files connected with the calculation of a given integral.
- `TempFileDelete`: by default it is set to `TempFile-Delete->True`; when set to `False`, it does not delete C++ source and binary files as well as log file.
- `LogFile`: causes `CSectors` to create (or not) a log file, where numerical results for given integral and epsilon are stored. The default is `True`.
- `ShowErrors`: controls whether the errors of the numerical calculation will be displayed or not; errors are calculated using the function `#res.get_error()#`of `sector decomposition` [24].
- `IterationsLow,IterationsHigh,CallsLow, CallsHigh`: these are Monte-Carlo parameters, see [24] for a description.
- `compiler`: allows to choose another compiler; the default is `g++`.
- `cppflags, libs`: the paths to header and library files, required by `sector decomposition` libraries [24].
- `min, max`: indicates minimum and maximum of the Laurent series expansion in epsilon.

The default options can also be displayed by the command `Options[DoSectors]`.

References

1. Z. Bern et al., The NLO multileg working group: Summary report. arXiv:0803.0494
2. G. Ossola, C.G. Papadopoulos, R. Pittau, CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes. J. High Energy Phys. **03**, 042 (2008). arXiv:0711.3596. doi:10.1088/1126-6708/2008/03/042
3. M. Moretti, F. Piccinini, A.D. Polosa, A fully numerical approach to one-loop amplitudes. arXiv:0802.4171
4. G. Heinrich, V. Smirnov, Analytical evaluation of dimensionally regularized massive on-shell double boxes. Phys. Lett. B **598**, 55–66 (2004). arXiv:hep-ph/0406053
5. M. Czakon, J. Gluza, T. Riemann, A complete set of scalar master integrals for massive 2- loop Bhabha scattering: Where we are. Nucl. Phys. B, Proc. Suppl. **135**, 83–87 (2004). arXiv:hep-ph/0406203
6. M. Czakon, J. Gluza, T. Riemann, Master integrals for massive two-loop Bhabha scattering in QED. Phys. Rev. D **71**, 073009 (2005). arXiv:hep-ph/0412164
7. M. Czakon, J. Gluza, T. Riemann, The planar four-point master integrals for massive two-loop Bhabha scattering. Nucl. Phys. B **751**, 1–17 (2006). arXiv:hep-ph/0604101
8. S. Actis, M. Czakon, J. Gluza, T. Riemann, Virtual hadronic and leptonic contributions to Bhabha scattering. Phys. Rev. Lett. **100**, 131602 (2008). arXiv:0711.3847. doi:10.1103/PhysRevLett.100.131602
9. S. Actis, M. Czakon, J. Gluza, T. Riemann, Virtual hadronic and heavy-fermion $O(\alpha^2)$ corrections to Bhabha scattering. Phys. Rev. D **78**, 085019 (2008). arXiv:0807.4691. doi:10.1103/PhysRevD.78.085019
10. R. Bonciani, A. Ferroglia, T. Gehrmann, D. Maitre, C. Studerus, Two-loop fermionic corrections to heavy-quark pair production: the quark–antiquark channel. J. High Energy Phys. **07**, 129 (2008). arXiv:0806.2301
11. M. Beneke, T. Huber, X.Q. Li, Two-loop QCD correction to differential semi-leptonic $b \to u$ decays in the shape-function region. arXiv:0810.1230
12. Y. Kiyo, D. Seidel, M. Steinhauser, $\mathcal{O}(\alpha\alpha_s)$ corrections to the $\gamma t\bar{t}$ vertex at the top quark threshold. arXiv:0810.1597
13. Z. Bern, M. Czakon, L. Dixon, D. Kosower, V. Smirnov, The four-loop planar amplitude and cusp anomalous dimension in maximally supersymmetric Yang–Mills theory. Phys. Rev. D **75**, 085010 (2007). arXiv:hep-th/0610248
14. A. Smirnov, M. Tentyukov, Four loop massless propagators: a numerical evaluation of all master integrals. Nucl. Phys. B **837**, 40–49 (2010). arXiv:1004.1149. doi:10.1016/j.nuclphysb.2010.04.020
15. J. Gluza, K. Kajda, D.A. Kosower, Towards a basis for planar two-loop integrals. arXiv:1009.0472
16. D. Kosower, R. Roiban, C. Vergu, The six-point NMHV amplitude in maximally supersymmetric Yang–Mills theory. arXiv:1009.1376
17. Z. Bern, M. Czakon, D. Kosower, R. Roiban, V. Smirnov, Two-loop iteration of five-point $N = 4$ super-Yang–Mills amplitudes. Phys. Rev. Lett. **97**, 181601 (2006). arXiv:hep-th/0604074
18. J.M. Drummond, J. Henn, V.A. Smirnov, E. Sokatchev, Magic identities for conformal four-point integrals. J. High Energy Phys. **01**, 064 (2007). arXiv:hep-th/0607160
19. J. Gluza, K. Kajda, T. Riemann, AMBRE—a Mathematica package for the construction of Mellin–Barnes representations

for Feynman integrals. Comput. Phys. Commun. **177**, 879–893 (2007). arXiv:0704.2423. doi:10.1016/j.cpc.2007.07.001
20. Katowice, webpage http://www.us.edu.pl/~gluza/ambre
21. DESY, webpage http://theory-zeuthen.desy.de/
22. M. Czakon, Automatized analytic continuation of Mellin–Barnes integrals. Comput. Phys. Commun. **175**, 559–571 (2006). arXiv:hep-ph/0511200
23. A.V. Smirnov, V.A. Smirnov, On the resolution of singularities of multiple Mellin–Barnes integrals. Eur. Phys. J. C **62**, 445 (2009). arXiv:0901.0386
24. C. Bogner, S. Weinzierl, Resolution of singularities for multi-loop integrals. Comput. Phys. Commun. **178**, 596–610 (2008). arXiv:0709.4092. doi:10.1016/j.cpc.2007.11.012
25. V. Smirnov, *Evaluating Feynman Integrals*. Springer Tracts in Modern Physics, vol. 211 (Springer, Berlin, 2004)
26. G. Heinrich, Sector Decomposition. Int. J. Mod. Phys. A **23**, 1457–1486 (2008). arXiv:0803.4177. doi:10.1142/S0217751X08040263
27. C. Anastasiou, A. Daleo, Numerical evaluation of loop integrals. J. High Energy Phys. **10**, 031 (2006). arXiv:hep-ph/0511176
28. J. Gluza, K. Kajda, T. Riemann, V. Yundin, New results for loop integrals: AMBRE, CSectors, hexagon, PoS (ACAT08) 124. arXiv:0902.4830
29. R. Gonsalves, Dimensionally regularized two loop on-shell quark form-factor. Phys. Rev. D **28**, 1542 (1983)
30. M. Czakon, A. Mitov, S. Moch, Heavy-quark production in gluon fusion at two loops in QCD. Nucl. Phys. B **798**, 210–250 (2008). arXiv:0707.4139. doi:10.1016/j.nuclphysb.2008.02.001
31. T. Binoth, G. Heinrich, An automatized algorithm to compute infrared divergent multi-loop integrals. Nucl. Phys. B **585**, 741–759 (2000). arXiv:hep-ph/0004013v2
32. T. Binoth, G. Heinrich, Numerical evaluation of multi-loop integrals by sector decomposition. Nucl. Phys. B **680**, 375–388 (2004). arXiv:hep-ph/0305234
33. D. Zeillinger, Ph.D. thesis, Univ. Innsbruck (2005)
34. D. Zeillinger, Enseign. Math. **52**, 143 (2006)
35. M. Spivakovsky, Prog. Math. **36**, 419 (1983)
36. Katowice, webpage http://www.us.edu.pl/~gluza/csectors
37. A.V. Smirnov, V.A. Smirnov, M. Tentyukov, FIESTA 2: parallelizeable multiloop numerical calculations. arXiv:0912.0158
38. MB tools webpage http://projects.hepforge.org/mbtools
39. J. Gluza, K. Kajda, T. Riemann, V. Yundin, News on Ambre and CSectors. arXiv:1006.4728