

Notes on the calculation of loop integrals

Christoph Sieg¹

Niels Bohr International Academy
Niels Bohr Institute
Blegdamsvej 17, 1662 Copenhagen, Denmark

¹ csieg@nbi.dk

Contents

1	Useful relations in $D = 2(\lambda + 1)$ dimensions	1
2	G-functions	2
2.1	Integrals without numerators	2
2.2	Integrals with numerators	4
3	Gegenbauer polynomial x-space technique	5
3.1	Introduction to the technique	5
3.2	Implementation in Mathematica	12
3.2.1	Radial integrations over domains	12
3.2.2	Evaluation of radial integrals by list manipulations	13
3.2.3	Evaluation of the integrand by list manipulations	14
3.2.4	Simplifying the output from the radial integration routines	16
3.2.5	Harmonic series by list manipulations	16

1 Useful relations in $D = 2(\lambda + 1)$ dimensions

The loop integrals are computed by using dimensional regularization, i.e. we set the spacetime dimension to

$$D = 2(\lambda + 1) , \quad \lambda = 1 - \varepsilon . \quad (1.1)$$

Also, the Wick rotation to Euclidean space is always understood. Here, we discuss the Euclidean versions of the integrals. The transformation of a propagator with weight α between x - and p -space reads

$$\begin{aligned} \frac{1}{k^{2\alpha}} &= \frac{\Gamma(\lambda + 1 - \alpha)}{\Gamma(\alpha)\pi^{\lambda+1}} \int \frac{d^D x e^{2ik \cdot x}}{x^{2(\lambda+1-\alpha)}} , \\ \frac{k^\mu}{k^{2\alpha}} &= -i \frac{\Gamma(\lambda + 2 - \alpha)}{\Gamma(\alpha)\pi^{\lambda+1}} \int \frac{d^D x e^{2ik \cdot x} x^\mu}{x^{2(\lambda+2-\alpha)}} . \end{aligned} \quad (1.2)$$

We normalize a propagator-type L -loop integral in momentum space as

$$I_L(p) = \int \frac{d^D k_1}{(2\pi)^D} \cdots \frac{d^D k_L}{(2\pi)^D} \frac{1}{\Pi_1 \dots \Pi_P} , \quad (1.3)$$

where the number of propagators is P , and the denominator Π of each propagator is a quadratic polynomial of a subset of the loop momenta k and possibly also the external momentum p .

This integral is transformed to an integral over points in x -space by inserting (1.2), and then integrating over the L loop momenta k , which gives L δ -functions, each with a prefactor π^D . We then obtain in $D = 2(\lambda+1)$ dimensions for propagators which all carry weight factors $\alpha = 1$ and which do not have non-trivial numerators

$$I_L = \frac{\Gamma(\lambda)^P}{(2^{2L}\pi^P)^{\lambda+1}} \int \frac{d^D x_1 \dots d^D x_{P-L} e^{2ip \cdot (x_{\text{out}} - x_{\text{in}})}}{X_1 \dots X_P} , \quad (1.4)$$

where the denominator X of each propagator in x -space only depends on the distance of two points, i.e. it is a function of only two coordinates.

2 G -functions

The G -functions are e.g. used in [1, 2].

2.1 Integrals without numerators

The relations (1.2) can be used to directly obtain the expressions for certain integrals. For example, a general one-loop integral with weights α and β for the two propagators is expressed as

$$\begin{aligned} \text{---} \text{---} \text{---} \text{---} &= \frac{1}{(2\pi)^D} \int \frac{d^D k}{k^{2\alpha}(k-p)^{2\beta}} = \frac{\Gamma(\lambda+1-\alpha)\Gamma(\lambda+1-\beta)}{(4\pi^2)^{\lambda+1}\Gamma(\alpha)\Gamma(\beta)} \int \frac{d^D x e^{2ip \cdot x}}{x^{2\lambda+2-\alpha-\beta}} \\ &= \frac{\Gamma(\lambda+1-\alpha)\Gamma(\lambda+1-\beta)\Gamma(\alpha+\beta-\lambda-1)}{(4\pi)^{\lambda+1}\Gamma(\alpha)\Gamma(\beta)\Gamma(2\lambda+2-\alpha-\beta)} \frac{1}{p^{2(\alpha+\beta-\lambda-1)}} . \end{aligned} \quad (2.1)$$

This relation is represented by

$$\text{---} \text{---} \text{---} \text{---} = G(\alpha, \beta) \frac{\alpha+\beta-\lambda-1}{p^{2(\alpha+\beta-\lambda-1)}} , \quad (2.2)$$

where the G -function is given by

$$G(\alpha, \beta) = \frac{\Gamma(\lambda+1-\alpha)\Gamma(\lambda+1-\beta)\Gamma(\alpha+\beta-\lambda-1)}{(4\pi)^{\lambda+1}\Gamma(\alpha)\Gamma(\beta)\Gamma(2\lambda+2-\alpha-\beta)} . \quad (2.3)$$

The G -function can now be used to compute the simplest scalar one-loop integral as

$$\text{---} \text{---} \text{---} \text{---} = G(1, 1) \frac{1-\lambda}{p^{2(1-\lambda)}} . \quad (2.4)$$

This integral is logarithmically UV divergent in four dimensions. The divergence is regularized by reducing the dimension as in (1.1), introducing the parameter ε . With the properties

$$\Gamma(1+x) = x\Gamma(x) , \quad \Gamma(n) = (n-1)! , \quad (2.5)$$

and the series expansion for the Γ -functions

$$\ln \Gamma(1-x) = \gamma x + \sum_{n=2}^{\infty} \frac{1}{n} \zeta(n) x^n , \quad \zeta(n) = \sum_{k=1}^{\infty} \frac{1}{k^n} , \quad (2.6)$$

we can then obtain the pole part of the above expression as

$$\boxed{\text{---} \text{---} \text{---} \text{---}} = \frac{1}{(4\pi)^2 \varepsilon} , \quad (2.7)$$

where the box drawn around the integral denotes the overall UV divergence of that integral given by the simple $\frac{1}{\varepsilon}$ pole in this case.

We can use the G -functions to obtain integrals at higher order. For example, we find

$$\begin{aligned}
\text{triangle with bubble} &= G(1, 1) \text{triangle}^{1-\lambda} = G(1, 1) \text{bubble}^{2-\lambda} = G(1, 1)G(2-\lambda, 1) \frac{1}{p^{2-2\lambda}}, \\
\text{rectangle with bubble} &= G(1, 1) \text{triangle with bubble}^{2-\lambda} = G(1, 1)G(2-\lambda, 1) \text{bubble}^{3-2\lambda} \\
&= G(1, 1)G(2-\lambda, 1)G(3-2\lambda, 1) \frac{1}{p^{3-3\lambda}}.
\end{aligned} \tag{2.8}$$

The above integrals are also logarithmically UV divergent. The expansion of the first integral in terms of ε yields

$$\text{triangle with bubble} = \frac{1}{(4\pi)^4} \left[\frac{1}{2\varepsilon^2} + \frac{1}{\varepsilon} \left(\frac{5}{2} - \gamma + \ln \frac{4\pi\mu^2}{p^2} \right) + \mathcal{O}(\varepsilon^0) \right]. \tag{2.9}$$

We see that the simple $\frac{1}{\varepsilon}$ pole of the UV divergence depends on the logarithm of the external momentum p .¹ The cancellation of this divergence hence seem to require a counterterm which is non-local. However, this is not the case. The presence of higher order poles in ε indicates the presence of subdivergences. The above integral contains a one-loop subdivergence which is removed by a corresponding one-loop counterterm. The counterterm at two loops should thus be read-off from the overall UV divergence of the integral, i.e. from the UV divergence which remains after the one-loop subdivergence has been subtracted. We hence have to compute

$$\text{triangle with bubble} - \boxed{\text{triangle with bubble}} - \text{bubble} = \frac{1}{(4\pi)^4} \left[-\frac{1}{2\varepsilon^2} + \frac{1}{2\varepsilon} + \mathcal{O}(\varepsilon^0) \right]. \tag{2.10}$$

From now on we will understand that a box drawn around a loop integral denotes the overall UV divergence of this integral, given by the pole part after all subdivergences have been subtracted. We hence write

$$\boxed{\text{triangle with bubble}} = \frac{1}{(4\pi)^4} \left[-\frac{1}{2\varepsilon^2} + \frac{1}{2\varepsilon} \right]. \tag{2.11}$$

The overall UV divergence of the three-loop integral in (2.8) is found by extracting the pole part of the expression

$$\text{rectangle with bubble} - \boxed{\text{triangle with bubble}} - \text{triangle with bubble} - \boxed{\text{triangle with bubble}} - \text{bubble} \tag{2.12}$$

The result then reads

$$\boxed{\text{rectangle with bubble}} = \frac{1}{(4\pi)^6} \left[\frac{1}{6\varepsilon^3} - \frac{1}{2\varepsilon^2} + \frac{2}{3\varepsilon} \right] \tag{2.13}$$

¹The mass μ comes from a factor $\mu^{4\varepsilon}$ in front of the dimensionally regularized integral which has to be added to preserve the total mass dimension.

2.2 Integrals with numerators

Very often one is confronted with scalar products of momenta in the numerators of loop integrals. If two contracted momenta are attached to different lines of a cubic vertex, the integral can be simplified by completing the square. I.e. one rewrites the scalar product as linear combination of the squares of those momentum combinations, which appear in the denominator. E.g. the scalar product of momenta p and q can be rewritten as

$$p \cdot q = \frac{1}{2}(\pm(p \pm q)^2 \mp p^2 \mp q^2) . \quad (2.14)$$

We can then reexamine the simple scalar integral (2.2), adding a momentum vector in the numerator with Lorentz index μ . Since the final result can only depend on the external momentum, it must be proportional to p^μ . We indicate the momentum in the numerator by an arrow at that line which carries the momentum. We do not write Lorentz indices explicitly. This yields

$$\begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} = G_1(\alpha, \beta) \xrightarrow{\alpha+\beta-\lambda-1} , \quad (2.15)$$

where the function G_1 can be obtained by completing the square with the external momentum as follows

$$\begin{aligned} \begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} &= \int \frac{d^D k}{(2\pi)^D} \frac{k \cdot p}{k^{2\alpha}(k-p)^{2\beta}} = \frac{1}{2} \int \frac{d^D k}{(2\pi)^D} \frac{-(k-p)^2 + k^2 + p^2}{k^{2\alpha}(k-p)^{2\beta}} \\ &= \frac{1}{2} \left(- \begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta-1 \\ \curvearrowleft \end{array} + \begin{array}{c} \alpha-1 \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} + \begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} \right) . \end{aligned} \quad (2.16)$$

On the l.h.s. it is thereby understood that the momenta indicated by arrows of the same type are contracted. Comparing with (2.15), and using (2.2), we read off

$$G_1(\alpha, \beta) = \frac{1}{2}(-G(\alpha, \beta - 1) + G(\alpha - 1, \beta) + G(\alpha, \beta)) . \quad (2.17)$$

If the numerator contains the scalar product of the momenta of the two propagators in the loop, we can use momentum conservation to obtain

$$\begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} = \begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowright \end{array} - \begin{array}{c} \alpha-1 \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} . \quad (2.18)$$

We hence immediately obtain

$$\begin{array}{c} \alpha \\ \curvearrowright \\ \text{---} \text{---} \\ \beta \\ \curvearrowleft \end{array} = G_2(\alpha, \beta) \xrightarrow{\alpha+\beta-\lambda-2} , \quad (2.19)$$

where

$$G_2(\alpha, \beta) = \frac{1}{2}(-G(\alpha, \beta - 1) - G(\alpha - 1, \beta) + G(\alpha, \beta)) . \quad (2.20)$$

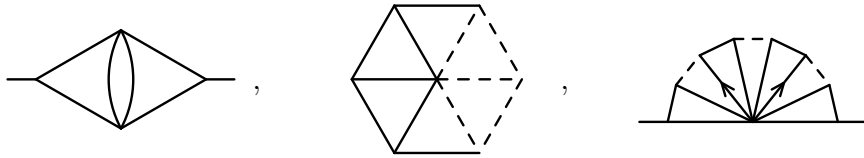


Figure 1: Loop integrals which cannot be solved by contracting bubbles

As an example we use the above given G -functions to immediately write down the result for certain class of integrals. We find

$$\begin{aligned}
 \begin{array}{c} 3 \\ \diagup \quad \diagdown \\ 2 \quad \quad L \\ \diagdown \quad \diagup \\ 1 \end{array} &= G_1(2, 1)G_1(3 - \lambda, 1) \dots G_1(L - (L - 2)\lambda, 1)G_2(L + 1 - (L - 1)\lambda, 1) \\
 &= \frac{1}{(4\pi)^{2L}} \left(-\frac{1}{L\varepsilon} + \mathcal{O}(\varepsilon^0) \right) .
 \end{aligned} \tag{2.21}$$

The types of integrals which can be directly solved in terms of G -functions are only a very small subclass of integrals. For example, integrals of the form given in Fig. 1 cannot be directly transformed into G -functions, they are not solvable by ‘contracting bubbles’. In the first and third graph, external momentum has to flow through the attached external lines. This is necessary to regulate the appearing IR divergences, which in four dimensions appear when two propagators without any numerators are attached to each other in a loop, and momentum is running only through these two legs of the vertex, where they meet. The second graph has no IR divergence, the external momenta can be rearranged in any convenient form or be put to zero when one is only interested in its UV divergence. It can, however not be solved in terms of G -functions for any configuration of its external momenta.

To solve the above types of integrals, the so-called Gegenbauer polynomial x -space technique proves to be very useful. It is described in the following section.

3 Gegenbauer polynomial x -space technique

The Gegenbauer polynomial x -space was introduced in [3]. Integrals with scalar products of momenta in the numerator are discussed in [4]. The treatment for the multiple appearance of a certain combination of the momenta in the numerator is described in [5].

G -functions are used to compute integrals which contain massless propagators. See [1] for some more details.

3.1 Introduction to the technique

The Gegenbauer polynomial x -space technique (GPXT) is based on an expansion of the propagators in the Wick-rotated loop integral in terms of Gegenbauer polynomials. Gegenbauer polynomials $C_n^{(\lambda)}$ are one set of orthogonal polynomials, which appear as generalizations

of the Legendre polynomials of ordinary 2-dimensional spherical harmonics to dimension $D = 2\lambda + 1$. Their definition in terms of a generating function as

$$\frac{1}{(1 - 2xt + t^2)^\lambda} = \sum_{n=0}^{\infty} C_n^{(\lambda)}(x)t^n \quad (3.1)$$

allows us immediately to reexpress the x -space propagators in terms of Gegenbauer polynomials. This yields

$$\frac{1}{(x_1 - x_2)^{2\lambda}} = \frac{1}{\max_{12}^\lambda} \sum_{n=0}^{\infty} C_n^\lambda(\hat{x}_1 \cdot \hat{x}_2) \left(\frac{\min_{12}}{\max_{12}} \right)^{\frac{n}{2}} . \quad (3.2)$$

where we denote $r_i = x_i^2$ in Euclidean space and \hat{x}_i is the corresponding unit vector in the direction of x_i . We have also abbreviated

$$\min_{ij} = \min(r_i, r_j) , \quad \max_{ij} = \max(r_i, r_j) . \quad (3.3)$$

In the coordinates r, \hat{x} the x -space integration measure reads

$$d^D x = \frac{1}{2} \Omega_{D-1} r^\lambda dr d\hat{x} , \quad (3.4)$$

where the volume of the $(D - 1)$ -dimensional unit sphere is given by

$$\Omega_{D-1} = \frac{2\pi^{\frac{D}{2}}}{\Gamma(\frac{D}{2})} . \quad (3.5)$$

The transformation of the integral measure in the L -loop integral (1.4) then leads to the normalization factor

$$N_\lambda(L, P) = \frac{\Gamma(\lambda)^P}{(2^{2L} \pi^P)^{\lambda+1}} \frac{1}{2^{P-L}} \Omega_{D-1}^{P-L} \quad (3.6)$$

in front of that integral. In the physical $D = 4$ dimensions where $\lambda = 1$ the factor simplifies to

$$N_1(L, P) = \frac{1}{(4\pi)^{2L}} . \quad (3.7)$$

In (1.4) there remains the exponential function. It can also be expanded in terms of series involving Bessel functions. Here, however we will not do so. For the renormalization problem we are only interested in the pole part of a given logarithmically UV divergent loop integral. In the x -space integral the UV divergence is located when all coordinates are small, i.e. the exponential factor from the Fourier transformation of the momentum integral is constant and can be removed. This corresponds to setting to zero all external momenta. In general this will spoil the final result by mixing it up with IR divergences which come from the integration region where certain coordinates are very large. To avoid this, we introduce a simple cutoff R as upper bound for the radial integrations.

From now on by the integral or by its graphical representation we always mean an expression which has the same UV divergence as the original momentum integral, but might differ from it by finite parts due to rearrangements of external momenta or due to applying the above described cutoff-procedure.

Inserting the expansion (3.2) of the propagators, the loop integral then also contains several infinite sums, running over the degrees in the product of Gegenbauer polynomials. The number of these sums depends on the number of propagators X_i , which contain coordinate differences. Since the value of the integral (1.4) does not change by shifting all coordinates by the same vector, we can transform a convenient point of the integral to the origin in the D -dimensional space. This point is called the root vertex. All propagators which are directly connected to it only depend on the respective radial coordinate r_i of the second vertex at x_i they are attached to. They do not yield a series expansion in terms of Gegenbauer polynomials. The most convenient choice of the root vertex in most cases is hence that vertex to which the maximum number of propagators is directly attached. For integrals that come from the renormalization of a composite operator with many of its elementary fields involved in interactions, the root vertex will very often be the composite operator itself.

As an example, we discuss the first and the second integral at four loops in Fig. 1. For the first integral we choose the downer central vertex as the root vertex, while for the second integral we choose the central vertex as the root vertex. The expansions in terms of Gegenbauer polynomials then read

$$\begin{aligned}
& \begin{array}{c} x_2 \\ \diagup \quad \diagdown \\ x_1 \quad \quad x_3 \\ \diagdown \quad \diagup \\ x_2 \end{array} = N_\lambda(3, 6) \sum_{n_1, n_2=0}^{\infty} A_\lambda(n_1, n_2) R_\lambda(n_1, n_2) , \\
& A_\lambda(n_1, n_2) = \int d\hat{x}_1 d\hat{x}_2 d\hat{x}_3 C_{n_1}^{(\lambda)}(\hat{x}_1 \cdot \hat{x}_2) C_{n_2}^{(\lambda)}(\hat{x}_2 \cdot \hat{x}_3) , \\
& R_\lambda(n_1, n_2) = \int_0^R \frac{dr_1 dr_2 dr_3 r_2^{-\lambda}}{(\max_{12} \max_{23})^\lambda} \left(\frac{\min_{12}}{\max_{12}} \right)^{\frac{n_1}{2}} \left(\frac{\min_{23}}{\max_{23}} \right)^{\frac{n_2}{2}} , \\
& \begin{array}{c} x_2 \\ \diagup \quad \diagdown \\ x_1 \quad \quad x_3 \\ \diagdown \quad \diagup \\ x_4 \end{array} = N_\lambda(4, 8) \sum_{n_1, \dots, n_4=0}^{\infty} A_\lambda(n_1, n_2, n_3, n_4) R_\lambda(n_1, n_2, n_3, n_4) , \\
& A_\lambda(n_1, n_2, n_3, n_4) = \int d\hat{x}_1 \dots d\hat{x}_4 C_{n_1}^{(\lambda)}(\hat{x}_1 \cdot \hat{x}_2) C_{n_2}^{(\lambda)}(\hat{x}_2 \cdot \hat{x}_3) C_{n_3}^{(\lambda)}(\hat{x}_3 \cdot \hat{x}_4) C_{n_4}^{(\lambda)}(\hat{x}_4 \cdot \hat{x}_1) , \\
& R_\lambda(n_1, n_2, n_3, n_4) = \int_0^R \frac{dr_1 \dots dr_4}{(\max_{12} \max_{23} \max_{34} \max_{41})^\lambda} \\
& \quad \left(\frac{\min_{12}}{\max_{12}} \right)^{\frac{n_1}{2}} \left(\frac{\min_{23}}{\max_{23}} \right)^{\frac{n_2}{2}} \left(\frac{\min_{34}}{\max_{34}} \right)^{\frac{n_3}{2}} \left(\frac{\min_{41}}{\max_{41}} \right)^{\frac{n_4}{2}} .
\end{aligned} \tag{3.8}$$

Having chosen the root vertex, we can now draw independent graphs for the radial and for the angular integrations. The angular graph is obtained by erasing from the original x -space representation of the loop integral the root vertex and all the propagators which are directly attached to it. To the remaining lines one then attaches its degree, i.e. the summation index of the expansion (3.2) of the respective propagator. This means, in the angular graph the Gegenbauer polynomials are represented by

$$C_n^{(\lambda)}(\hat{x} \cdot \hat{y}) = x \overset{n}{\text{---}} y \tag{3.9}$$

For the above examples, the angular graphs respectively become

$$A_\lambda(n_1, n_2) = \hat{x}_1 \begin{array}{c} \nearrow^{n_1} \hat{x}_2 \\ \searrow^{n_2} \end{array} \hat{x}_3, \quad A_\lambda(n_1, n_2, n_3, n_4) = \hat{x}_1 \begin{array}{c} \nearrow^{n_1} \hat{x}_2 \\ \searrow^{n_2} \\ \nwarrow^{n_4} \hat{x}_4 \\ \nearrow^{n_3} \end{array} \hat{x}_3. \quad (3.10)$$

The angular integrations can now be performed easily. For this, we exploit the orthogonality property of the Gegenbauer polynomials, which in terms of the \hat{x}_i becomes

$$\int d\hat{x} C_m^\lambda(\hat{x}_1 \cdot \hat{x}) C_n^\lambda(\hat{x} \cdot \hat{x}_2) = \frac{\lambda}{n + \lambda} \delta_{mn} C_n^\lambda(\hat{x}_1 \cdot \hat{x}_2), \quad (3.11)$$

when a product of Gegenbauer polynomials is integrated over a common angular vector \hat{x} . It has the graphical representation

$$\hat{x}_1 \begin{array}{c} \xrightarrow{m} \hat{x} \\ \xrightarrow{n} \end{array} \hat{x}_2 = \frac{\lambda}{n + \lambda} \delta_{mn} \hat{x}_1 \xrightarrow{n} \hat{x}_2, \quad \hat{x} \circlearrowleft^n = C_n^{(\lambda)}(1) = \frac{\Gamma(n + 2\lambda)}{n! \Gamma(2\lambda)}. \quad (3.12)$$

With the above rules, we immediately find for the angular graphs

$$A_\lambda(n_1, n_2) = \delta_{0n_1} \delta_{n_1 n_2}, \quad A_\lambda(n_1, n_2, n_3, n_4) = \left(\frac{\lambda}{n_1 + \lambda} \right)^3 C_{n_1}^{(\lambda)}(1) \delta_{n_1 n_2} \delta_{n_2 n_3} \delta_{n_3 n_4}. \quad (3.13)$$

The graph of radial integrations is basically the graphical presentation of the loop integral itself, with the degrees n and the x -space weights of the propagators attached to the lines. For the first integral the angular integration cancels all summations, and for the second integral the four summations associated to the four propagators are reduced to a single one. We can hence write

$$R_\lambda \equiv R_\lambda(0, 0) = r_1 \begin{array}{c} \nearrow^0 r_2 \\ \searrow^0 \\ \nwarrow^0 \\ \nearrow^0 \end{array} r_3, \quad R_\lambda(n_1) \equiv R_\lambda(n_1, n_1, n_1, n_1) = r_1 \begin{array}{c} \nearrow^{n_1} r_2 \\ \searrow^0 \\ \nwarrow^0 \\ \nearrow^{n_1} \end{array} r_3, \quad (3.14)$$

where the x -space weights of the propagators directly attached to the root vertex are the original weights, but shifted by $-\lambda$ due to the factors r_i^λ in the numerator, stemming from the transformation of the integral measures (3.4). The weights other propagators which are present also in the angular graphs are the powers of the function min. The powers of the corresponding function max is obtained by shifting them by $-\lambda$.

The first 3-dimensional radial integral can be solved by splitting the volume, i.e. the cube with edge length R into $3! = 6$ integration domains, defined by the orderings $0 \leq r_1 \leq r_2 \leq r_3 \leq R$ and permutations thereof. Due to the symmetry of the integrand under reflections

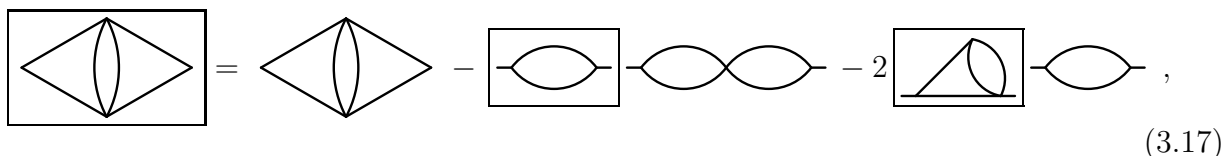
which exchange r_1 and r_3 , we have only three independent domains $0 \leq r_1 \leq r_2 \leq r_3 \leq R$, $0 \leq r_2 \leq r_1 \leq r_3 \leq R$ and $0 \leq r_1 \leq r_3 \leq r_1 \leq R$. We hence find for the radial integral

$$\begin{aligned}
R_\lambda &= \int_0^R \frac{dr_1 dr_2 dr_3 r_2^{-\lambda}}{(\max_{12} \max_{23})^\lambda} \\
&= 2 \left(\int_0^R dr_3 \int_0^{r_3} dr_2 \int_0^{r_2} dr_1 r_2^{-2\lambda} r_3^{-\lambda} + \int_0^R dr_3 \int_0^{r_3} dr_1 \int_0^{r_1} dr_2 r_1^{-\lambda} r_2^{-\lambda} r_3^{-\lambda} \right. \\
&\quad \left. + \int_0^R dr_2 \int_0^{r_2} dr_3 \int_0^{r_3} dr_1 r_2^{-3\lambda} \right) \\
&= \frac{2}{3-3\lambda} \left(\frac{1}{(2-2\lambda)} + \frac{1}{(2-2\lambda)(1-\lambda)} + \frac{1}{2} \right) R^{3-3\lambda}.
\end{aligned} \tag{3.15}$$

Setting $\lambda = 1 - \varepsilon$, we obtain the expansion

$$R_\lambda = \frac{1}{\varepsilon^3} + \frac{1}{\varepsilon^2} \left(\frac{1}{3} + \ln R \right) + \frac{1}{\varepsilon} \left(\frac{1}{3} + \ln R + \frac{3}{2} \ln^2 R \right). \tag{3.16}$$

As expected, the presence of all higher order poles indicates the presence of one- and two-loop subdivergences, and the dependence on $\ln R$ reminds us that we have to subtract these subdivergences to obtain the final result. We therefore have to compute



$$\boxed{\text{diamond}} = \text{diamond} - \boxed{\text{line}} - \boxed{\text{two lines}} - 2 \boxed{\text{triangle}} , \tag{3.17}$$

where as before a box drawn around an integral denotes the overall UV divergence of that integral. For the integrals which multiply these pole parts, we have to use the same scheme that we used for the integral itself, i.e. we have to compute them with GPXT, dropping the exponential and introducing the IR regulator R . The one-loop integral is found as

$$\text{line} = \frac{\Gamma(\lambda)^2}{(2^2 \pi^2)^{1+\lambda}} \int \frac{d^D x_1}{x_1^{4\lambda}} = N_\lambda(1, 2) \int_0^R dr_1 r_1^{-\lambda} = N_\lambda(1, 2) \frac{1}{1-\lambda} R^{1-\lambda}. \tag{3.18}$$

Its pole part coincides with the result we obtain by using G -functions. The two-loop integral reads

$$I_2 = \text{triangle} = N_\lambda(2, 4) \int_0^R \frac{dr_1 dr_2 r_2^{-\lambda}}{\max_{12}^\lambda} = N_\lambda(2, 4) \frac{2-\lambda}{2(1-\lambda)^2} R^{2-2\lambda}. \tag{3.19}$$

It is easy to check that the overall UV divergence again coincides with the result found via G -functions. With the above results, we then find for the overall UV divergence of the three-loop integral

$$\boxed{\text{diamond}} = \frac{1}{(4\pi)^6} \left(\frac{1}{3\varepsilon^3} - \frac{2}{3\varepsilon^2} + \frac{1}{3\varepsilon} \right). \tag{3.20}$$

The second 4-dimensional radial integral requires a splitting into $4! = 24$ integration domains, defined by the orderings $0 \leq r_1 \leq r_2 \leq r_3 \leq r_4 \leq R$ and permutations thereof. Due

to the symmetry of the integrand under cyclic permutations of the r_i , and under reflections (forming the 8-dimensional dihedral group D_4), we can divide out these symmetries and thus only have to consider 3 e.g. the three independent domains $0 \leq r_1 \leq r_2 \leq r_3 \leq r_4 \leq R$, $0 \leq r_2 \leq r_1 \leq r_3 \leq r_4 \leq R$ and $0 \leq r_1 \leq r_3 \leq r_2 \leq r_4 \leq R$. We hence find

$$\begin{aligned}
R_\lambda(n) &= \int_0^R \frac{dr_1 \dots dr_4}{(\max_{12} \max_{23} \max_{34} \max_{41})^\lambda} \left(\frac{\min_{12} \min_{23} \min_{34} \min_{41}}{\max_{12} \max_{23} \max_{34} \max_{41}} \right)^{\frac{n}{2}} \\
&= 8 \left(\int_0^R dr_4 \int_0^{r_4} dr_3 \int_0^{r_3} dr_2 \int_0^{r_2} dr_1 r_1^n r_2^{-\lambda} r_3^{-\lambda} r_4^{-n-2\lambda} \right. \\
&\quad + \int_0^R dr_4 \int_0^{r_4} dr_3 \int_0^{r_3} dr_1 \int_0^{r_1} dr_2 r_1^{-\lambda} r_2^n r_3^{-\lambda} r_4^{-n-2\lambda} \\
&\quad \left. + \int_0^R dr_4 \int_0^{r_4} dr_2 \int_0^{r_2} dr_3 \int_0^{r_3} dr_1 r_1^n r_2^{-n-2\lambda} r_3^n r_4^{-n-2\lambda} \right) \\
&= \frac{8}{(4-4\lambda)(n+3-2\lambda)(n+1)} \left(\frac{2}{(n+2-\lambda)} + \frac{1}{(2n+2)} \right) R^{4-4\lambda}.
\end{aligned} \tag{3.21}$$

It is immediately visible that the above expression contains a simple pole for $\lambda = 1 - \varepsilon$. Which is given by $R_\lambda(n) = \frac{5}{(n+1)^3 \varepsilon}$. Inserting also the result from the angular integrations, we find

$$\begin{aligned}
\left[\text{Diagram: A square with a diamond inside, divided into four triangles by a vertical and horizontal line.} \right] &= N_\lambda(4, 8) \sum_{n=0}^{\infty} \frac{5}{(n+1)^5 \varepsilon} = \frac{1}{(4\pi)^8} \frac{5}{\varepsilon} \zeta(5).
\end{aligned} \tag{3.22}$$

Recall that the above equality has to be understood only for the pole parts. The terms in a further expansion in ε differ, because we have explicitly simplified the computational effort by dropping the exponential factor and introducing the IR regulator scale R .

We have seen an example, in which the angular graph consists of a simple loop. This is equivalent to the statement that each integral over an angular coordinate contains maximally two Gegenbauer polynomials, which depend on the respective angular direction. The method can be generalized to cases in which the angular graph contains vertices which can be resolved by contraction of bubbles. This statement appears to be similar to the statement made in the context of G -functions. However, it is fulfilled by a larger class of integrals. The condition for an integral to be solvable in terms of G -functions, which is that its graphical representation only contains ‘contractible bubbles’, has to be fulfilled for the *complete* graph. For GPXT the statement only has to be fulfilled for the *angular* graph, and it is sufficient to find one choice for the root vertex for which the angular graph fulfills this condition.

The contraction of bubbles in the angular graph involves a reexpression of two Gegenbauer polynomials with the same weight and the same argument as a Clebsch-Gordan decomposition in terms of a sum over single Gegenbauer polynomials. The relevant expression reads

$$C_m^{(\lambda)}(x) C_n^{(\lambda)}(x) = \sum_{\substack{i=|m-n| \\ \frac{i+m+n}{2} \in \mathbb{N}}}^{m+n} D_\lambda(m, n, i) C_i^{(\lambda)}(x) \tag{3.23}$$

where the coefficients are given by

$$D_\lambda(m, n, i) = \frac{i!(i + \lambda)\Gamma(\frac{m+n+i}{2} + 2\lambda)}{\Gamma(\lambda)^2\Gamma(\frac{m+n+i}{2} + \lambda + 1)\Gamma(i + 2\lambda)} \frac{\Gamma(\frac{-m+n+i}{2} + \lambda)\Gamma(\frac{m-n+i}{2} + \lambda)\Gamma(\frac{m+n-i}{2} + \lambda)}{\Gamma(\frac{-m+n+i}{2} + 1)\Gamma(\frac{m-n+i}{2} + 1)\Gamma(\frac{m+n-i}{2} + 1)} . \quad (3.24)$$

Its graphical representation is

$$\hat{x}_1 \begin{array}{c} \text{---} m \text{---} \\ \text{---} n \text{---} \end{array} \hat{x}_2 = D_\lambda(m, n, i) \hat{x}_1 \text{---} i \text{---} \hat{x}_2 . \quad (3.25)$$

Such bubbles in angular graphs appear not only due to the presence of bubbles in the integral. They also appear, if scalar products of the momenta are present in the numerator. According to (1.2), in x -space these products translate to scalar products of the coordinates. They can be reexpressed in terms of Gegenbauer polynomials as

$$x_1 \cdot x_2 = \frac{1}{2\lambda}(r_1 r_2)^{\frac{1}{2}} C_1^{(\lambda)}(\hat{x}_1 \cdot \hat{x}_2) . \quad (3.26)$$

The above expression can be used for a single scalar products and also if all momentum factors within multiple scalar products correspond to distinct propagators, i.e. that not more than a single arrow is associated to each line².

In principle one could try to directly compute any integral with non-trivial numerators via GPXT. In many cases, however this might not be the most efficient way. As can be seen from (1.2), the presence of scalar products in the numerator changes the weight of the corresponding propagator in x -space. Its expansion (3.2) in terms of Gegenbauer polynomials then yields a Gegenbauer polynomial with a weight that differs from the one of the other Gegenbauer polynomials, and hence the orthogonality relation (3.11) does not hold. To proceed one must expand the Gegenbauer polynomial of different weight in terms Gegenbauer polynomials with the required weight.

This difficulty can be avoided, if momentum conservation at the vertices is used to first shift the numerator momenta to those propagators, which are directly connected to the root vertex and are hence not expanded in terms of Gegenbauer polynomials. The integral of interest is then expressed as a linear combination of integrals. As an example, we consider an integral without subdivergences and decompose it as

$$\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} + \frac{1}{2} \left(- \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} + \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} - \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} + \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \right) , \quad (3.27)$$

such that the central vertex becomes the root vertex. Apart from the third integral, all integrals on the r.h.s. have one-, two- and three-loop subdivergences, but it is not necessary to subtract these to find the result for the combination. They must cancel among each other. This means, however, that one has to compute all the integrals in the same scheme, e.g. using the IR cutoff-regularization when applying GPXT.

²Tensor products of equal momenta are mapped to traceless products in x -space. Their presence requires more care [5].

The angular and radial graphs for the first integral read

$$\begin{aligned}
A_\lambda(n, m) &= \text{Diagram} = \left(\frac{\lambda}{n+\lambda}\right)^4 D_\lambda(m, 1, n) C_n^{(\lambda)}(1), \\
R_\lambda(n, m) &= \text{Diagram} = \int_0^R \frac{dr_1 \dots dr_5 r_1^\lambda r_2^{-\frac{1}{2}} r_3^{-\frac{1}{2}}}{(\max_{12} \max_{23} \max_{34} \max_{45} \max_{51})^\lambda} \\
&\quad \left(\frac{\min_{23}}{\max_{23}}\right)^{\frac{m}{2}} \left(\frac{\min_{12}}{\max_{12}} \frac{\min_{34}}{\max_{34}} \frac{\min_{45}}{\max_{45}} \frac{\min_{51}}{\max_{51}}\right)^{\frac{n}{2}},
\end{aligned} \tag{3.28}$$

where the integral is then given by

$$\begin{aligned}
\text{Diagram} &= -\frac{\lambda}{2} N_\lambda(4, 9) \sum_{n=0}^{\infty} \sum_{\substack{m=|n-1| \\ m \neq n}}^{n+1} A_\lambda(n, m) R_\lambda(n, m) \\
&= -\frac{\lambda}{2} N_\lambda(4, 9) \left[R_\lambda(0, 1) + \sum_{n=1}^{\infty} \sum_{\delta=\pm 1} A_\lambda(n, n+\delta) R_\lambda(n, n+\delta) \right],
\end{aligned} \tag{3.29}$$

and we have used that $A_\lambda(0, 1) = 1$. All higher order poles are contained in the first term $R_\lambda(0, 1)$, while the infinite sum only leads to simple $\frac{1}{\epsilon}$ poles.

Computing also the other integrals, and combining them as in (3.27) we then find

$$\text{Diagram} = \frac{1}{(4\pi)^8} \zeta(3). \tag{3.30}$$

3.2 Implementation in Mathematica

First of all, we have to define the normalization of an x -space integral as $\text{NLP}[\lambda_ , L_ , P_]$, the Gegenbauer polynomial evaluated at unit argument as $\text{Cs}[\lambda_ , j_]$, and the Clebsch-Gordan coefficients $\text{Ds}[\lambda_ , l_ , m_ , n_]$ for the contraction of bubbles in the angular graphs.

The main task is to implement the radial integrations of GPXT. This is discussed in the following.

3.2.1 Radial integrations over domains

For GPXT we have to solve integrals of piecewise defined monomials over a cube with edge length R . The corresponding expressions are of the form

$$I = \int_0^R dr_1 \dots dr_n I(r_1, \dots, r_n), \tag{3.31}$$

where $I(r_1, \dots, r_n)$ is a monomial of the r_i , $i = 1, \dots, n$, and the exponents of the r_i depend on the ordering of the real numbers r_i on the positive real axis. The above expression can be explicitly evaluated by first splitting the integration into $n!$ domains, given by $r_1 \leq r_2 \leq \dots \leq r_n$ and permutations thereof. We therefore first define a list of the n radial variables r_1, \dots, r_n . We fuse a variable a with a variable i by defining

```
vfuse[a_, i_] := ToExpression[ToString[a] <> ToString[i]]
```

and then set up a list $\{r_1, r_2, \dots, r_n\}$ as

```
r1[n_] := Table[vfuse[r, i], {i, 1, n}]
```

A list of lists of the $n!$ permutations of the original list r_1 is then generated as

```
pl[n_] := Permutations[r1[n]]
```

For the integration itself, we define a function

```
RIV1[r1_, integr_] := Block[{order, al, it},
  order = r1 /. {List -> Less};
  al = Join[r1, {R}];
  it = Sequence @@ Reverse[
    Table[{al[[i]], 0, al[[i+1]]}, {i, 1, Length[r1]}]];
  Integrate[FullSimplify[integr, order], it]]
```

Its first argument r_1 should be one of the lists with entries r_1, \dots, r_n . Its second argument is the integrand. First, we assign to `order` the ordering of the entries in the list, the first entry is associated to the smallest integration variable in the ordering relation. The list `al` is the original r_1 with element `R` appended to the end. This list is used to define the second argument of the integration routine `Integrate`, which requires for each integration a list of the form $\{\text{integr_var}, \text{lower_bdy}, \text{upper_bdy}\}$. The first argument of `Integrate` contains the integrand which with `FullSimplify` is simplified due to the ordering `order`.

It is then applied as

```
RIV1[#, Integrand[Sequence @@ r1[n]]] & /@ pl[n]
```

to the list of permutations `pl`. It takes the list and transforms it into an ordering relation, with which the integrand is then evaluated and the integrations are carried out.

3.2.2 Evaluation of radial integrals by list manipulations

The above procedure is a possible solution, but too slow to compute integrals at high loop order. We should refrain from using the integration procedure of `Mathematica` and exploit the fact that the integrands are monomials in the arguments. This allows us to simulate the integrations by the much faster list manipulations. In the integration domain $r_1 \leq r_2 \leq \dots \leq r_n$, where the integrand in (3.31) is evaluated to $I(r_1, \dots, r_n) = r_1^{e_1} \dots r_n^{e_n}$, the integration yields

$$\int_0^R dr_n \int_0^{r_n} dr_{n-1} \dots \int_0^{r_2} dr_1 r_1^{e_1} \dots r_n^{e_n} = \frac{1}{(e_1 + 1)(e_1 + e_2 + 2) \dots (e_1 + \dots + e_n + n)} R^{e_1 + \dots + e_n + n}. \quad (3.32)$$

We see, that for a realization of the integration in `mathematica` we should first obtain the list of exponentials e_i in a given integration domain from a given integrand. We therefore define the following function

```
Exordlist[r1_, integr_] := Block[{ordint, order},
  order = r1 /. {List -> Less};
  ordint = FullSimplify[integr, order];
  Thread[Exponent[PowerExpand[ordint], r1]]]
```

The integrand `integr`, evaluated with the ordering `order` is assigned to the variable `ordint`. It is then power expanded to ensure that all factors are separate. The function `Exponent` extracts the power with which its second argument appears in its first argument. In our case the second argument is the list of radial variables, since `Thread` evaluates this function with each entry of that list as argument and places the result in that list at the position of that entry. Hence the individual exponents appear in the output list at the same positions as their individual bases appear in `r1`.

The list of the exponents e_i , $i = 1, \dots, n$ called `exlist` is now transformed by the following function to simulate the integrations

```
Intop[exlist_] := Block[{intex},
  intex = Simplify[Drop[FoldList[#1 + #2 + 1 &, 0, exlist], 1]];
  (intex /. List -> Times)-1 Rintex[[ -1 ]]]
```

The command `FoldList` applies the function in its first argument to the list `exlist` in the following way: `FoldList` takes its second argument and writes it into an output list. The first argument of `FoldList` is then evaluated with its first argument taken from the output list and its second argument taken from the list which is the third argument of `FoldList`. The positions in both lists are thereby the same, and the result is written into the output list at the subsequent position. In our particular case, the output list of `FoldList` starts with 0 and then contains the n factors in the denominator of (3.32). We drop the unwanted first entry from the output list. In a second step, the remaining entries are then multiplied, the result is inverted, and the appropriate factor from the upper boundary of the last integration is added. The integration routine is then defined as

```
Riv2[r1_, integr_] := Intop[Exordlist[r1, integr]]
```

It is applied to the list of permutations `pl` as

```
Riv2[#, Integrand[Sequence @@ r1[n]]] & /@ pl[n]
```

3.2.3 Evaluation of the integrand by list manipulations

It also turns out that the evaluation `FullSimplify[integr, order]` used in `Exordlist` is very time-consuming. To improve the performance, we transform the integrand before its evaluation, using self defined functions based on lists. The integrand that appears in (3.31) is of the form

$$I(r_1, \dots, r_n) = \prod_{i=1}^n r_i^{a_i} \prod_{j \neq i}^n \max_{r_i, r_j}^{b_{ij}} \min_{r_i, r_j}^{c_{ij}} \quad (3.33)$$

The extraction of the list of exponents e_i , $i = 1, \dots, n$ for a given ordering of the r_i is made more efficient by defining the following functions as substitutes

```

ExFac[r1_, r_, ex_] :=
  ReplacePart[Table[0, {i, 1, Length[r1]}], Position[r1, r][[1]] -> ex]

ExMax[r1_, r1_, r2_, ex_] := Block[{mp},
  mp = Max[Position[r1, r1], Position[r1, r2]];
  ReplacePart[Table[0, {i, 1, Length[r1]}], mp -> ex]]

```

and a corresponding function `ExMin`. The function `ExfFac` returns a list of the same length as its first entry `r1` which contains the value of `ex` at the same position at which `r` appears in `r1`. This simulates the extraction of the exponentials a_i from the factors $r_i^{a_i}$ in (3.33). The function `ExMax` compares the positions of its second and its third argument in the list `r1` and returns a list with the same length as `r1` which at the larger of the two positions contains the entry `ex` and zeroes in all other entries. This simulates the extraction of the exponentials b_{ij} from the factors $\max_{r_i r_j}^{b_{ij}}$ in (3.33). The list of exponentials c_{ij} which appear as $\min_{ij}^{c_{ij}}$ is obtained by the corresponding function `ExMin`.

Since we want to give the integrand as an input in terms of standard functions of *Mathematica*, we define the transformation from this input form to our preferred form as

```

trafo[r1_, integr_] :=
  Plus @@ (List @@ PowerExpand[integr]
    //.{Max[a_, b_]^{ex_:1} -> ExMax[r1, a, b, ex],
      Min[a_, b_]^{ex_:1} -> ExMin[r1, a, b, ex]}
    //.(r_?(MemberQ[r1, #]&))^{ex_:1} -> ExFac[r1, r, ex]))

```

The transformation `trafo` first transforms the products of individual factors $r_i^{a_i}$, $\max_{r_i r_j}^{b_{ij}}$, $\min_{ij}^{c_{ij}}$ into a list of these factors. It then replaces each of these factors by the respective functions `ExfFac`, `ExMax` and `ExMin`. Factors which are not powers of the functions `Max` or `Min` are tested whether their bases are given by the variables r_i that occur in the list `r1`. If so, they are replaced by `ExfFac`. As a result, we obtain a list of lists, each of length of `r1`, which contain the exponentials e_i at the position at which r_i occurs in `r1`. As a last step, we have to add-up all these list to consider the multiplication of the factors in (3.33). This is achieved by applying `Plus` to the obtained list. It is important to stress that the argument `integr` must only contain factors which match one of the replacement rules. A factor that does not match one of the replacement rules is not transformed via `ExfFac`, `ExMax` or `ExMin` into a list of length `Length[r1]`. Hence, when the list elements are summed, it is mistakenly added to each of the entries of the resulting list.

The integration over all domains is given by applying the combination

```

RIv3[r1_, integr_] := Intop[trafo[r1, integr]]

```

to the list of permutations `pl` as

```

RIv3[#, Integrand[Sequence @@ r1[n]]] & /@ pl[n]

```


3.2.4 Simplifying the output from the radial integration routines

As a result of the integration procedures RIV2 and RIV3 applied to an n -dimensional radial integral, we obtain a list of length $n!$ which contains the result of the integration for each ordering of the radial variables. An effective procedure to sum up all contributions, thereby also simplifying the result, is to look for entries which have the same denominator and sum them up first. This procedure is carried out by the following function

```
cdensort[expr_, binop_:Plus] :=
Block[{ilist, denlist, numlist, reddenlist},
  ilist = Factor[List @@ expr];
  denlist = Denominator[ilist];
  numlist = Numerator[ilist];
  reddenlist = Union[denlist];
  Head[expr] @@
  Map[Factor[Tr[Extract[numlist, Position[denlist, #]], binop, 1]] /# &,
    reddenlist]]
```

The first argument is the expression `expr` to be simplified, the second argument is the binary operator `binop`, which should be used to combine similar objects. If the function is called with one argument, the default `Plus` is substituted as its second argument. First, `expr` is transformed into a `List` called `ilist`, with elements which are factorized by `Factor`. Then separate lists `denlist` and `numlist` of respectively the denominators and the numerators of the elements in `ilist` are generated. The list `reddenlist` of different numerators is obtained by applying `Union` to the list of denominators. Then, the function `Position[denlist, #]` applied to `reddenlist` extracts the list of positions, at which each denominator in `reddenlist` appears in `denlist`. The resulting list is then used by `Extract` to obtain the list of numerators at the very same positions, i.e. the numerators of fractions with the same denominators. The trace operation applied to the result with binary operation `binop` then combines the numerators. The result is divided by the respective denominator taken from `reddenlist`. Finally, the original `Head` of `expr` is restored.

We use the above function to sum up expressions with the same denominator. The complete expression for the integration of an integrand `Integrand[r1, ..., rn]` over the n radial coordinates is then defined as

```
RInt[n_] :=
  Tr[cdensort[RIV3[#, Integrand[Sequence @@ r1[n]]] & /@ p1[n]]]
```

3.2.5 Harmonic series by list manipulations

Summing up the expressions of the form (3.32), the final result will be a rational function of two polynomials of order $n = P - L$ in α different summation indices i_1, \dots, i_α , where α corresponds to the number of loops in the angular graph. Combining the expression for the radial integrations with the one for the angular integrations and expanding in powers of λ , the order of the polynomials might increase, but the form of the expressions remains the same.

The examples discussed here will be a special case, in which the number of sums with infinity as upper boundary is restricted to one. There might be additional sums over finite

intervals which originate from scalar products of loop momenta in the numerator. The problem of summing up the result then reduces to summing the series of a finite number of rational functions of polynomials which depend on a single summation index. Furthermore, the denominator polynomial is a product of factors $(a + i)^e$, where a and e are integers.

Mathematica evaluates these summations automatically when the routine `Sum` is used. For rational functions of polynomials of high order which appear at high loop order the built-in routine becomes too slow. We hence use a self-written routine for these kind of summations. It reads

```
hsum[f_, i_] :=
Block[{g, arg, arg2, j, den, denl, ex, exp, denpl, fl, cl, c, numpoly,
      numansatz, num0, num1, sol, arglist, hsl, dl, res, Sdiv},
  g = f;
  arg = Together[g];
  arg2 = Together[g] /. i -> j;
  den = Factor[Numerator[Denominator[arg] / Denominator[arg2]]];
  If[den === 1,
    If[g === 0, 0, Print["Warning: sum does not converge"];
      
$$\frac{1}{\text{Denominator}[g]} S_0[\text{Numerator}[g]]$$
,
    denl = List @@ (ex[1, 1] den /. (a_ + i)e_ -> ex[a + i, e])
      /. ex -> List;
    denpl[l_] := Table[(1[[1]])-k, {k, 1, 1[[2]]}];
    fl = Flatten[Map[denpl, denl]];
    cl = Table[vfuse[c, k], {k, 1, Length[fl]}];
    numpoly = CoefficientList[Together[den g], i];
    numansatz = CoefficientList[Numerator[Together[Tr[fl cl]]], i];
    num0 = Drop[numansatz, Length[numpoly]];
    num1 = Take[numansatz, Length[numpoly]];
    sol = Solve[{num0 == 0, num1 == numpoly}, cl];
    arglist = Select[fl cl /. sol[[1]], # != 0 &];
    hsl = arglist //. (a_ : 0 + i)e_ /; e_ ≤ -1 :> (S-e[∞] - Sum[je, j, 1, a]);
    dl = (arglist //. (a_ : 0 + i)e_ /; e_ ≤ -1 :> 0) S0[∞];
    res = Simplify[Tr[hsl + dl] /. Se_ /; e_ ≥ 2 [∞] :> Zeta[e]];
    If[Coefficient[res //. Se_[∞] -> Sdiv, Sdiv] != 0,
      Print["Warning: Sum does not converge"]];
    res]]
```

The function depends on two arguments, the first is the rational function f of two polynomials in the second variable i , which selects the summation index. The variable `den` contains the denominator with all factors which do not depend on the summation index i canceled. Hence, `den===1` is `True` if the denominator is independent of i . This is tested first. Since `den===1` is `True` also for `den=Denominator[0]`, another test as to distinguish whether the numerator (or the rational function $g=f$) is zero or non-zero. It returns 0 in the former case or prints a warning message and returns the result in terms of the divergent sum $S_0(\infty)$ in the latter case. If the denominator `den` depends on the summation index, first a list of denominators is generated by replacing factors of the form $(a + i)^e$ by an auxiliary function

`ex[a + i, e]`. A factor `ex[1, 1]` is added as a representation for the constant monomial $(i^0)^1$. The expression is then transformed into a list of lists which contain as the first entry the base and as a second entry the exponent (the power) with which the base occurs as a factor in the denominator of the original expression. Partial fraction decomposition requires that in an ansatz the fractions with all powers of a factor in the denominator are present, up to the power with which it appears in the denominator. The function `denpl` generates a list of all the fractions associated with a base and its power occurring in the denominator. The function `denpl` is then applied to each individual element of the list `denl` by using `Map`. This generates a list containing all elements of a basis for the partial fraction decomposition called `fl`. A coefficient list `c1` for this basis is then generated. The coefficients are then determined by comparing the polynomial in the numerator of the original expression with the polynomial in the numerator which is found when the linear combination of individual fractions of the ansatz is expressed as a single fraction with a common numerator. The lists which contain the coefficients of the polynomials in `i` occurring in the original expression and in the ansatz are called `numpoly` and `numansatz`, respectively. In general, the list `numansatz` will be longer than the coefficient list `numpoly`, since the numerator polynomial of the original expression might have lower degree. The list `numansatz` is hence split into two lists `num0` and `num1`. The first contains the coefficients of the subpolynomial of the ansatz that matches the degree of the polynomial of the original expression. The second contains the remaining coefficients multiplying powers of `i` that exceed the degree of the polynomial of the original expression. The coefficients of the coefficient list `c1` are now found by solving the system of equations, given by the vanishing of all coefficients in `num0` and by identification of the coefficients in `num1` with the ones in `numpoly`. The solution is then used to set up the list `arglist` which contains all basis fractions multiplied by their respective coefficient. Zeros are dropped from this list. Each element in `arglist` contains a factor of the form $(a + i)^{-e}$. The final summation can be cast into the form

$$\sum_{i=1}^{\infty} \frac{1}{(a+i)^e} = S_e(\infty) - S_e(a) , \quad S_e(n) = \sum_{i=1}^n \frac{1}{i^e} . \quad (3.34)$$

Summation is thus simulated by replacing the corresponding structures in `arglist` by the r.h.s. of the first of the above equations. The result is the list `hs1`. In a second list `ds1` the same structures are set to zero to test for any constant parts in the result. They are multiplied by $S_0(\infty)$. The harmonic sums $S_e(n)$ converge in the limit $n \rightarrow \infty$ for $e \geq 2$, and in this case are replaced by their limits, i.e. by the functions $\zeta(e)$. Hence `Res` gives the final result, and it is convergent if no further $S_e(\infty)$ are present, which then necessarily have $e \leq 1$. This is tested in the last step, which prints a warning message if the sum is divergent.

References

- [1] D. I. Kazakov, “Calculation of Feynman integrals by the method of ‘uniqueness’,” *Theor. Math. Phys.* **58** (1984) 223–230.
- [2] D. I. Kazakov, “Analytical Methods for multiloop calculations: two lectures on the method of uniqueness,” JINR-E2-84-410.
- [3] K. G. Chetyrkin, A. L. Kataev, and F. V. Tkachov, “New Approach to Evaluation of Multiloop Feynman Integrals: The Gegenbauer Polynomial x Space Technique,” *Nucl. Phys.* **B174** (1980) 345–377.
- [4] A. V. Kotikov, “The Gegenbauer Polynomial Technique: the evaluation of a class of Feynman diagrams,” *Phys. Lett.* **B375** (1996) 240–248, [hep-ph/9512270](#).
- [5] F. Fiamberti, A. Santambrogio, C. Sieg, and D. Zanon, “Anomalous dimension with wrapping at four loops in $\mathcal{N} = 4$ SYM,” [0806.2095](#).