Abstract

We give numerical integration results for Feynman loop diagrams such as those covered by Laporta [1] and by Baikov and Chetyrkin [2], and which may give rise to loop integrals with UV singularities. We explore automatic adaptive integration using multivariate techniques from the PARINT package for multivariate integration, as well as iterated integration with programs from the QUADPACK package, and a trapezoidal method based on a double exponential transformation. PARINT is layered over MPI (Message Passing Interface), and incorporates advanced parallel/distributed techniques including load balancing among processes that may be distributed over a cluster or a network/grid of nodes. Results are included for 2-loop vertex and box diagrams and for sets of 2-, 3- and 4-loop self-energy diagrams with or without UV terms. Numerical regularization of integrals with singular terms is achieved by linear and non-linear extrapolation methods.

Keywords: Feynman loop integrals, UV singularities, multivariate adaptive integration, numerical iterated integration, asymptotic expansions, extrapolation/convergence acceleration

1. Introduction

High energy physics collider experiments target the precise measurement of parameters in the standard model and beyond, and detection of any deviations of the experimental data from the theoretical predictions, leading to the study of new phenomena. In modern physics, there are three basic interactions acting on particles: weak, electromagnetic and strong interactions. When we consider a scattering process of elementary particles, the cross section reflects the dynamics that govern the motion of the particles, caused by the interaction.

All information on a particle interaction is contained in the amplitude according to the (Feynman) rules of Quantum Field Theory. Generally, with a given particle interaction, a large number of configurations (represented by Feynman diagrams) is associated. Each diagram represents one of the possible configurations of virtual processes, and it describes a part of the total amplitude. The square sum of the amplitudes delivers the probability or cross section of the process. Based on the Feynman rules, the amplitude can be obtained in an automatic manner: (i) determine the physics process (external momenta and order of perturbation); (ii) draw all Feynman diagrams relevant to the process; (iii) describe the contributions to the amplitude.

Feynman diagrams are constructed in such a way that the initial state particles are connected to the final state particles by propagators and vertices. Particles meet at vertices according to a coupling constant $g$, which indicates the strength of the interaction. The amplitude is expanded as a perturbation series in $g$, where the leading (lowest) order of approximation corresponds to the tree level of the Feynman diagrams. Higher orders require the evaluation
of loop diagrams, so that the computation of loop integrals is very important for the present and future high-energy experiments.

When few masses occur in the computation of loop integrals, analytic approaches are generally feasible. However, in the presence of a wide range of masses, analytic evaluation becomes very complicated or impossible. For one-loop integrals, explicit analytic methods have been established by many authors, but alternative approaches are compulsory for multi-loop integrals with a variety of masses and momenta. We propose a fully numerical approach based on multi-dimensional integration and extrapolation, and demonstrate results of the technique for multi-loop integrals with and without masses.

In the computation of loop integrals we have to handle singularities. Depending on the value of internal masses and external momenta, the integrand denominator may vanish in the interior of the integration domain. The term \( i\varphi \) (subtracted from \( V \)) in the denominator of the loop integral representation of Eq (4) is intended to prevent the integral from diverging if \( V \) vanishes in the domain. The idea of our numerical extrapolation approach is to consider \( \rho \) not as an infinitesimal small number for the analytic continuation but as a finite number, to make the integral non-singular. We choose a sequence of \( \rho \) values, \( \rho_\ell \to 0 \) (e.g., a geometric sequence), so that multi-dimensional integration yields consecutive \( I(\rho_\ell) \) corresponding to \( \rho_\ell \). The sequence of \( I(\rho_\ell) \) is extrapolated numerically to approximate the value of the loop integral in the limit as \( \rho_\ell \to 0 \). For physical kinematics where an imaginary part is present, it can be treated numerically as well as the real part, since the integrand is not singular for finite \( \rho_\ell \). In previous work we have demonstrated various loop integral computations using this type of method not only in the Euclidean but also in the physical region \([2,3,4,5,6,7,8,9]\).

For the infrared divergent case we have two prescriptions. One is to introduce a small fictitious mass for the massless particles and the other is to use dimensional regularization. We have shown results for several problem classes in \([10,5,11,12,9]\).

In this paper we concentrate on loop integrals with UV singularities, which satisfy asymptotic expansions in the dimensional regularization parameter \( \varepsilon \) (see Eq (4), where the space-time dimension \( n \) will be set to \( n = 4 - 2\varepsilon \) to account for UV singularity). Based on multi-dimensional integration and numerical extrapolation, we present a novel numerical regularization method for integrals with UV singularities, applied to 1-, 2-, 3- and 4-loop diagrams. We compare with results in the literature, including those of Laporta \([1]\) – whose method is based on the numerical solution of systems of difference equations, the sector decomposition approach by Smirnov and Tentyukov \([13]\), and the analytic results by Baikov and Chetyrkin \([2]\).

For a given integrand function \( f : D \subset \mathbb{R}^d \to \mathbb{R} \), a \( d \)-dimensional domain \( D \), and (absolute/relative) error tolerances \( t_a \) and \( t_r \). If it is found that Eq (2) cannot be achieved, an error indicator should be returned. In order to achieve the accuracy requirement, the actual error should not exceed the error estimate \( E_{a,f} \), and the error estimate should not exceed the weaker of the absolute and relative error tolerances (indicated by the maximum taken on the right of Eq (2)). When a relative or an absolute accuracy (only) needs to be satisfied we set \( t_a = 0 \) or \( t_r = 0 \), respectively. If both \( t_a \neq 0 \) and \( t_r \neq 0 \), the weaker of the two error tolerances is imposed; if \( t_a = t_r = 0 \) then the program will reach an abnormal termination. This type of accuracy requirement is based on \([1,4]\) and used extensively in QUADPACK \([1,5]\).

Known methods for parallelization of these procedures include:

(i) Parallelization on the rule or points level: typically in non-adaptive algorithms, e.g., for Monte-Carlo (MC) algorithms and composite rules using grid or lattice points. Then in \( I f = \int_D f \approx \sum_k w_k f(\vec{x}_k) \), the function evaluations \( f(\vec{x}_k) \) are performed in parallel.

(ii) Parallelization on the region level: in adaptive (region-partitioning) methods. These lead to task pool strategies, which may benefit from load balancing on distributed memory systems; or maintain a shared priority queue on shared memory systems.

(iii) We added multi-threading to iterated integration \([16,17,18]\): the inner integrals are independent and computed
in parallel. For example, over a subregion \( S = D_1 \times D_2 \) (with inner region \( D_2 \)) consider \( \int_{D_2} F(\vec{x}) \, d\vec{x} \approx \sum_{k} w_k \bar{F}(\vec{x}_k) \), with \( \bar{F}(\vec{x}_k) = \int_{D_1} f(\vec{\gamma}, \vec{x}) \, d\vec{\gamma} \). The integrations in the different coordinate directions can be performed adaptively, which we achieved with iterated versions of the 1D programs DQAGS or DQAGIE from QUADPACK \cite{25,26,27}.

We further apply numerical extrapolation techniques for convergence acceleration of a sequence of integrals with respect to a parameter \( \gamma \). For linear extrapolation, an asymptotic expansion of the form

\[
I(\gamma) \sim \sum_{k \geq 1} C_k \varphi_k(\gamma), \quad \text{as} \quad \gamma \to 0
\]

is assumed, where \( I(\gamma) \) represents the integral and the sequence of \( \varphi_k(\gamma) \) is known. If the structure of the expansion is unknown we resort to a non-linear extrapolation with the \( \varepsilon \)-algorithm \cite{19,20,21,22,23}.

This paper gives an overview of our recent work. Section 2 provides background and notations for multi-loop Feynman integrals and diagrams, and discusses the use of extrapolation or convergence acceleration. Section 3 describes iterated integration, the PARINT adaptive strategies, and the double exponential transformation method. Numerical results obtained for a set of 2-loop self-energy, vertex and box diagrams are discussed in Section 4; 3-loop massive and massive self-energy diagrams are covered in Section 5, and 4-loop massless self-energy diagrams in Section 6.

Results from parallel distributed computations were obtained on the thor cluster of the Center for High Performance Computing and Big Data at WMU, where we used 16-core cluster nodes with Intel(R) Xeon(R) E5-2670, 2.6 GHz dual processors and 128 GB of memory, and the cluster’s Infiniband interconnect for message passing via MPI. Some sample sequential and parallel results were collected from runs on Intel(R) Xeon(R) CPU E5-1660 3.30GHz, E5-2687W v3 3.10 GHz, and on a 2.6 GHz Intel(R) Core i7 Mac-Pro with 4 cores and 16 GB memory under OS X. For the inclusion of OpenMP \cite{24} multi-threading compiler directives in the iterated integration code (based on the Fortran version of QUADPACK), we used the (GNU) gfortran compiler and the Intel Fortran compiler, with the flags -fopenmp and -openmp, respectively. PARINT and its integrand functions were compiled with gec (mpice). Besides Intel processors, we used POWER7(R) 3.83 GHz on the KEKSC system A of the Computing Research Center at KEK (SR16000 model M1), with the HITACHI Fortran90 compiler that enables automatic parallelization with the flag -parallel.

2. Feynman loop integrals and extrapolation

2.1. General form of Feynman loop integrals

Higher-order corrections are required for accurate theoretical predictions of the cross section for particle interactions. Loop diagrams are taken into account, leading to the evaluation of loop integrals. The derivation of a closed analytic form is generally hard or impossible for higher-order loop integrals with arbitrary internal masses and external momenta. Thus we resort to numerical calculations.

A scalar \( L \)-loop integral with \( N \) internal lines can be represented in Feynman parameter space by

\[
I = I_N = (-1)^N \frac{\Gamma(N - \frac{d + 4}{2})}{(4\pi)^{d/2}} \int_0^1 \prod_{r=1}^N \left( 1 - \sum_{\ell} x_{r,\ell} \right) \frac{1}{U^{nL/2} (V - i\Omega)^{N-1} L^{d/2}},
\]

where

\[
V = M^2 - \frac{1}{U} W, \quad M^2 = m^2_{x_r}
\]

and \( m \) is the mass for the propagator associated with \( x_r \). Here \( U \) and \( W \) are polynomials determined by the topology of the corresponding diagram and physical parameters \( U = 1 \) for 1-loop \((L = 1)\) integrals, and \( n \) is the space-time dimension. We further denote

\[
I_N = \left( \frac{1}{(4\pi)^n L^{d/2}} \right) I = (-1)^N \frac{\Gamma(N - \frac{d + 4}{2})}{(4\pi)^{nL/2}} I,
\]

defining \( I \) and \( J \) as integrals with a factor different from that of \( I_N \), in order to draw comparisons with results in the literature. We sometimes also use the following notation for Feynman parameters,

\[
x_{jk\ldots} = x_j + x_k + \cdots.
\]
The integration in Eq (4) is taken over the \( N \)-dimensional unit cube. However, as a result of the \( \delta \)-function one of the \( x_i \) can be expressed in terms of the other ones in view of \( \sum_{j=1}^{N} x_j = 1 \), which reduces the integral dimension to \( N - 1 \) and the domain to the \( d = (N-1) \)-dimensional unit simplex

\[
S_d = \{ (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid \sum_{j=1}^{d} x_j \leq 1 \text{ and } x_j \geq 0 \}. \tag{6}
\]

When the behavior of a singularity of the integrand is moderate, we can carry out the integration within the unit simplex domain without variable transformation. For the numerical integration where a steeper singularity appears, the unit simplex domain of Eq (4) can be transformed to the \( (N - 1) \)-dimensional unit cube, using

\[
x_1 = \bar{x}_1 \\
x_2 = (1 - x_1) \bar{x}_2 \\
x_3 = (1 - x_1 - x_2) \bar{x}_3 \\
\vdots \\
x_{N-1} = (1 - x_1 - x_2 - \ldots - x_{N-2}) \bar{x}_{N-1}
\]

with Jacobian \((1 - x_1)(1 - x_1 - x_2)\ldots(1 - x_1 - x_2 - \ldots - x_{N-2})\), i.e.,

\[
\int_{0}^{1} dx_1 \int_{0}^{1-x_1} dx_2 \int_{0}^{1-x_1-x_2} \cdots \int_{0}^{1-x_1-x_2-\cdots-x_{N-2}} \frac{\text{d}x_{N-1}}{(1-x_1-x_2-\cdots-x_{N-2})} f(x_1, x_2, \ldots, x_{N-1})
\]

\[
= \int_{0}^{1} dx_1 \int_{0}^{1-x_1} dx_2 (1-x_1) (1-x_1-x_2) \cdots (1-x_1-x_2-\cdots-x_{N-2}) f(1-x_1, x_2, \ldots, 1-x_1-x_2-\cdots-x_{N-2}) \bar{x}_{N-1}.
\tag{8}
\]

We find that the approximations thus obtained are often more accurate than those generated with the multivariate double-exponential transformation by Takahasi and Mori \([25, 26, 27]\), which is given in Section 3.3. We show examples of its application in Sections 5.4 and 6.2. Furthermore, we introduce another type of variable transformations related to the topology of Feynman diagrams to increase the accuracy of the results for some integrals in Sections 4.1, 4.2, and 5.4. The integration domain is mapped to the unit cube. Unlike the first two transformations, we determine the latter using a heuristic approach.

Loop integrals are notorious for singularities due to vanishing denominators, which may lead to divergence (e.g., IR or UV divergence) of the integral. In the absence of IR and UV singularities, we set \( n = 4 + 2\varepsilon \) \((\text{cf., Section 2.2})\), and for UV singularities, \( n = 4 - 2\varepsilon \). We apply the regularization by a numerical extrapolation as \( \varepsilon \to 0 \) \((\text{cf., Section 2.2})\).

The term \( i \varphi \) prevents the integrand denominator in Eq (4) from vanishing in the interior of the domain, and can be used for regularization. A regularization to keep the integral from diverging was achieved by extrapolation as \( \varrho \to 0 \) in \([3, 4, 5, 6, 7, 8, 9]\). Results given in \([11]\) applied iterated integration with QUADPACK programs and a double extrapolation with respect to \( \varrho \) and \( \varepsilon \) to deal with interior as well as IR singularities.

However, even for finite integrals, setting \( \varepsilon = 0 \) or \( \varrho = 0 \) in the integrand of Eq (4) may not yield the desired accuracy and it may be advantageous to extrapolate as \( \varepsilon \to 0 \) or \( \varrho \to 0 \) \((\text{cf., Section 5.1})\).

2.2. Numerical extrapolation

For an extrapolation with respect to the dimensional regularization parameter \( \varepsilon \), the integral of Eq (4) is evaluated as a sequence of \( I(\varepsilon) \) for decreasing \( \varepsilon = \varepsilon_k \), which assumes an asymptotic expansion of the form of Eq (3) for \( \gamma = \varepsilon \). For example, the \( \varphi_k(\varepsilon) \) functions in Eq (3) may be integer powers of \( \varepsilon \), \( \varphi_k(\varepsilon) = \varepsilon^k \). Then for finite integrals, \( \kappa = 0 \) in Eq (3) and the integral is represented by \( C_0 \).

Linear extrapolation can be applied when the \( \varphi_k(\varepsilon) \) functions are known. In that case, \( I(\varepsilon) \) is approximated for decreasing values of \( \varepsilon = \varepsilon_k \), and Eq (3) is truncated after 2, 3, \ldots, \( v \) terms to form linear systems of increasing size in the \( C_k \) variables. This is a generalized form of Richardson extrapolation \([28, 22]\). If the integral approximation becomes harder with smaller \( \varepsilon \), we can use slowly decreasing sequences \( [\varepsilon_k] \), such as a geometric sequence with base 1/1.15. Another sequence of interest is based on the Bulirsch sequence \([b_k] : 1, 2, 3, 4, 6, 8, 12, 16, 24, \ldots \) \((\text{see 29})\); we employ \((1/b_k)_{k \geq k_0}\), from a starting index \( k_0 \) in the Bulirsch sequence. The stability of linear extrapolation using
geometric, harmonic and Bulirsch type sequences was studied by Lyness [30] with respect to the mesh ratio of composite rules. The condition of the system was found best for geometric and worse for the harmonic sequences, with the Bulirsch sequence behavior in between.

We resort to non-linear extrapolation when the structure of the asymptotic expansion is not known. In previous work we have made ample use of the $\epsilon$-algorithm [19, 21, 22, 23], which can be applied with geometric sequences of $\epsilon$. The extrapolation results given in this paper are achieved with a version of the $\epsilon$-algorithm code from QUADPACK [15]. In between calls, the implementation retains the last two lower diagonals of the triangular extrapolation table. When a new element $I(\varepsilon_j)$ of the input sequence is supplied, the algorithm calculates a new lower diagonal, together with an estimate or measure of the distance of each newly computed element from preceding neighboring elements. With the location of the “new” element in the table relative to $e_0$, $e_1$, $e_2$, $e_3$, pictured as: $e_0 \rightarrow e_1 \rightarrow e_2 \rightarrow e_3$ we have that $\text{new} = e_1 + 1/(1/e_1 - e_3) + 1/(e_2 - e_1) - 1/(e_1 - e_0)$, and the distance measure for the new element is set to $|e_2 - e_3| + |e_3 - e_0| + |e_2 - \text{new}|$. The new lower diagonal element with the smallest value of the distance measure is then returned as the result for this call to the extrapolation code. Note that the accuracy of the extrapolated result is generally limited by the accuracy of the input sequence.

For an extrapolation as $\varepsilon \to 0$ in Eq (4), the integral is approximated by a sequence of numerical results for $I(\varepsilon)$, $I(\varepsilon) \sim \sum_{k \geq x} C_k \varphi_k(\varepsilon)$ for $\varepsilon = \varepsilon_k$ is assumed, where the $\varphi_k(\varepsilon)$ functions are generally unknown, and we perform non-linear extrapolation with the $\epsilon$-algorithm.

### 3. Numerical Integration Methods

Though various integration methods may be applicable in our approach, we currently use three types of integration methods as presented in subsequent sections. These are: numerical iterated integration, parallel adaptive integration and double-exponential transformation methods.

#### 3.1. Numerical iterated integration

For iterated integration over a $d$-dimensional product region we express Eq (1) as

$$If = \int_{a_1}^{b_1} dx_1 \int_{a_2}^{b_2} dx_2 \ldots \int_{a_d}^{b_d} f(x_1, x_2, \ldots, x_d),$$

(9)

where the limits of integration are given by functions $\alpha_j = \alpha_j(x_1, x_2, \ldots, x_{j-1})$ and $\beta_j = \beta_j(x_1, x_2, \ldots, x_{j-1})$. In particular, the boundaries of the $d$-dimensional unit simplex $S_d$ given by Eq (6) are $\alpha_j = 0$ and $\beta_j = 1 - \sum_{k=1}^{j-1} x_k$.

For the numerical integration over the interval $[\alpha_j, \beta_j]$, $1 \leq j \leq d$ in Eq (9) we can apply, e.g., the 1D adaptive integration code DQAG from the QUADPACK package [15] in each coordinate direction, and select the $(K = 15)$-point Gauss-Kronrod rule pair via an input parameter, for the integral (and error) approximation on each subinterval. If an interval $[a, b]$ arises in the partitioning of $[\alpha_j, \beta_j]$, then the local integral approximation over $[a, b]$ is of the form

$$\int_{a}^{b} dx \, F(c_1, \ldots, c_{j-1}, x_j) = \sum_{k=1}^{K} w_k F(c_1, \ldots, c_{j-1}, x^{(k)}),$$

(10)

where the $w_k$ and $x^{(k)}$, $1 \leq k \leq K$, are the weights and abscissae of the local rule scaled to the interval $[a, b]$ and applied in the $x_j$-direction. For $j = 1$ this is the outer integration direction. The function evaluation

$$F(c_1, \ldots, c_{j-1}, x^{(k)}) = \int_{\alpha_{j-1}}^{b_{j-1}} dx_{j-1} \ldots \int_{a_{j-2}}^{b_{j-2}} dx_{j-2} f(c_1, \ldots, c_{j-1}, x^{(k)}, x_{j-1}, \ldots, x_d), \quad 1 \leq k \leq K,$$

(11)

is itself an integral in the $x_{j-1}, \ldots, x_d$-directions for $1 \leq j < d$, and is computed by the method(s) for the inner integrations. For $j = d$, Eq (11) is the evaluation of the integrand function

$$F(c_1, \ldots, c_{d-1}, x^{(k)}) = f(c_1, \ldots, c_{d-1}, x^{(k)}).$$
Note that successive coordinate directions may be combined into layers in the iterated integration scheme. Furthermore, the error incurred in any inner integration will contribute to the integration error in all of its subsequent outer integrations \[31, 32, 33\].

Since the \(F()\) evaluations on the right of Eq (10) are independent of one another they can be evaluated in parallel. Important benefits of this approach include that:

(i) the granularity of the parallel integration is large, especially when the inner integrals \(F()\) are of dimension \(\geq 2\);
(ii) the points where the function \(F\) is evaluated in parallel are the same as those of the sequential evaluation; i.e., apart from the order of the summation in Eq (10), the parallel calculation is essentially the same as the sequential one. This important property facilitates the debugging of parallel code. As another characteristic, the parallelization does not increase the total amount of computational work.

In addition, the memory required for the procedure is determined by (the sum of) the amounts of memory needed for the data pertaining to the subintervals incurred in each coordinate direction (corresponding to the length of the recursion stack for a recursive implementation). Consequently the total memory increases linearly as a function of the dimension \(d\). Note that successive coordinate directions may be combined into layers in the iterated integration scheme.

To achieve the multi-threading, OpenMP [24] compiler directives were inserted in the iterated integration code. For the Fortran version of QUADPACK we used the (GNU) gfortran compiler and the Intel Fortran compiler, with the flags \(-fopenmp\) and \(-openmp\), respectively.

3.2. PARINT package

Written in C and layered over MPI [34], the PARINT methods (parallel adaptive, quasi-Monte Carlo and Monte Carlo) are implemented as tools for automatic integration, where the user defines the integrand function and the domain, and specifies a relative and absolute error tolerance for the computation \((t_r, t_a)\), respectively. For PARINT the integrand is generally defined as a vector function with \(m\) components,

\[
\vec{f} : \mathcal{D} \subset \mathbb{R}^d \rightarrow \mathbb{R}^m,
\]

over a (finite) \(d\)-dimensional (hyper-rectangular or simplex) domain \(\mathcal{D}\). Denoting the exact integral by

\[
\int_{\mathcal{D}} \vec{f}(\vec{x}) \, d\vec{x},
\]

then the objective of Eq (2) is generalized to returning an approximation \(Q\vec{f}\) and absolute error estimate \(E_a\vec{f}\) such that

\[
\|Q\vec{f} - \int_{\mathcal{D}} \vec{f}(\vec{x}) \, d\vec{x}\| \leq \|E_a\vec{f}\| \leq \max\{t_a, t_r \|\int_{\mathcal{D}} \vec{f}(\vec{x}) \, d\vec{x}\|\}
\]

(in infinity norm). In order to satisfy the error criterion of Eq (14) the program tests throughout whether

\[
\|E_a\vec{f}\| \leq \max\{t_a, t_r \|Q\vec{f}\|\}
\]

is achieved. We used the vector function integration capability in [35] for a simultaneous computation of the entire entry sequence for extrapolation, obtained as the \(m\) components of the integral \(\int_{\mathcal{D}} \vec{f}(\vec{x}) \, d\vec{x}\).

The available cubature rules in PARINT (to compute the integral approximation over the domain or its subregions) include a set of rules for the \(d\)-dimensional cube [36, 37, 38], the 1D (Gauss-Kronrod) rules used in QUADPACK and a set of rules for the \(d\)-dimensional simplex [39, 40, 41]. Some results in this paper are computed over the \(d\)-dimensional simplex using iterated integration with Gauss-Kronrod rules. In other cases, multivariate rules of polynomial degree 7 or 9 are used over the \(d\)-dimensional unit cube. A formula is said to be of a particular polynomial degree \(k\) if it renders the exact value of the integral for integrands that are polynomials of degree \(\leq k\), and there are polynomials of degree \(k + 1\) for which the formula is not exact. The number of function evaluations per (sub)region is constant, and the total number of subregions generated, or the number of function evaluations in the course of the integration, is considered a measure of the computational effort.
3.2.1. **PARINT adaptive methods**

In the adaptive approach, the integration domain is divided initially among the workers. Each on its own part of the domain, the workers engage in an adaptive partitioning strategy similar to that of DQAGE from QUADPACK [15] and of DCUHRE [42] by successive bisections. The workers then each generate a local priority queue as a task pool of subregions.

The priority queue is implemented as a max-heap keyed with the estimated integration errors over the subregions, so that the subregion with the largest estimated error is stored in the root of the heap. If the user specifies a maximum size for the heap structure on the worker, the task pool is stored as a *deep* or *double-ended heap*, which allows deleting of the maximum as well as the minimum element efficiently, in order to maintain a constant size of the data structure once it reaches its maximum.

A task consists of the selection of the associated subregion and its subdivision (generating two children regions), integration over the children, deletion of the parent region (root of the heap) and insertion of the children into the heap (see Figure 1). The bisection of a region is performed perpendicularly to the coordinate direction in which the integrand is found to vary the most, according to 4th-order differences computed in each direction [42]. The subdivision procedure continues until the global error estimate falls below the tolerated error, or the total number of function evaluations exceeds the user-specified maximum.

3.2.2. **Load balancing**

For a regular integrand behavior and *p* MPI processes distributed evenly over homogeneous processors, the computational load would ideally decrease by a factor of about *p*. Otherwise it may be possible to improve the parallel time (and space) usage by load balancing, to attempt keeping the loads on the worker task pools balanced.

The receiver-initiated, scheduler based load balancing strategy in PARINT is an important mechanism of the distributed integration algorithm [43, 44, 45, 46]. The message passing is performed in a non-blocking and asynchronous manner, and permits overlapping of computation and communication, which benefits PARINT’s efficiency on a hybrid platform (multi-core and distributed) where multiple processes are assigned to each node. As a result of the asynchronous processing and message passing on MPI, PARINT executes on a hybrid platform by assigning multiple processes across the nodes. The user has the option of turning load balancing on or off, as well as allowing or dis-allowing the controller to also act as a worker.

3.2.3. **Use of PARINT**

PARINT can be invoked from the command line, or by calling the `pi_integrate()` function in a program for computing an integral of the form of Eq (13). A user guide is provided in [47]. The call sequence passes a pointer to the integrand function, typed as a pointer to a function that returns an integer, and where the parameters of the integrand function correspond to the integral dimension, argument vector `x`, number of component functions `nfuncs` (corresponding to `m` in Eq (12)) and the resulting component values of the function `f(x)`. Apart from `nfuncs`, further input parameters of `pi_integrate()` are: an integer identifying the cubature/quadrature rule to use, the maximum number of function evaluations allowed, the region type (hyper-rectangle or simplex) and specification. The output parameters are: the integral and error component approximations `result[]` and `error[]`, and a user-declared pointer to a status structure. The execution time is returned as part of the output printed by the PARINT `pi_print_result()` function.

When PARINT is used as a stand-alone executable, it uses the PARINT Plug-in Library (PPL) mechanism to specify integrand functions. The functions are written by the user, added to the library (along with related attributes), and then compiled using a PARINT-supplied compiler into plug-in modules (.ppl files). A single PPL file is loaded at runtime by the PARINT executable. Using a function library enables quick access to a predefined set of functions and lets PARINT users add and remove integrand functions dynamically without re-compiling the PARINT binary. Once these functions are stored in the library, they can be selected by name for integration.

For an execution on MPI, the MPI host file (myhostfile) contains lines of the form: `node_name slots=ppn` where `ppn` is the number of processes to be used on each participating node. A typical MPI run from the command line is invoked as: `mpiexec -n ppi xxx.ppl`. The script `xxx.ppl` contains the lines:

```
pi_integrate(0, 0, 11, 11, 0, x, result, error, status, &ppi);`
integrals as diagrams (a-d) as the sunrise-sunset

4.1. 2-loop self-energy integrals

We treat UV divergence by a dimensional regularization with \( \eta \) can be addressed by dimensional regularization with IR divergence occurs through a singularity arising when \( V \) vanishes at the boundaries of the domain. This problem can be addressed by dimensional regularization with \( n = 4 + 2\epsilon \), which we implemented numerically in [11, 12, 9] using an extrapolation as \( \epsilon \to 0 \) (\( \epsilon > 0 \)). It is assumed that the denominator does not vanish in the interior of the integration domain, so we can set \( \varrho = 0 \).

In this paper, we concentrate on UV divergence, which occurs when \( U \) vanishes at the boundaries. The \( \Gamma \)-function in Eq (16) contributes to UV divergence when \( N \leq 4 \). We treat UV divergence by a dimensional regularization with \( n = 4 - 2\epsilon \), implemented by a numerical extrapolation as \( \epsilon \to 0 \) after either iterated integration with DQAGS from QUADPACK, multivariate adaptive integration with PARINT or the DE formula.

3.3. Double-exponential transformation

The Double Exponential (tanh-sinh) formula, referred to here as DE formula in short, was proposed by Takahasi and Mori in 1974 [25, 26, 27]. It is an efficient method for the numerical approximation of an integral whose integrand is a holomorphic function with end-point singularities. This formula transforms the integration variable in \( \int_{-\infty}^{\infty} f(x) dx \) to \( x = \phi(t) = \frac{1}{2}(\tanh (\frac{\pi}{4}\sinh(t)) + 1) \). Then \( I = \int_{0}^{1} f(x) dx = \int_{-\infty}^{\infty} f(\phi(t)) \phi'(t) dt \) with \( \phi'(t) = \frac{\pi \cosh(t)}{4 \cosh^{2}(\frac{\pi}{4}\sinh(t))} \). After the transformation, the trapezoidal rule is applied leading to

\[
I^{\text{new}}_{\text{h}} = \sum_{k=0}^{N_{\text{evd}}} f(\phi(kh)) \phi'(kh), \quad (15)
\]

with mesh size \( h \) and \( N_{\text{evd}} = N_{-} + N_{+} + 1 \) function evaluations. A major issue in numerical integration with the DE formula is the treatment of overflows at large \( |t| \) and for large \( |N_{\text{evd}}| \), and it is sometimes necessary to evaluate the integrand using multi-precision arithmetic even though it takes more CPU time than double precision. This helps alleviating the loss of trailing digits in the evaluation of the integrand near the end-points.

For multi-dimensional loop integrals, we use the DE formula in a repeated integration scheme. Apart from our sequential implementation we also developed code for multi-core systems using a parallel library such as OpenMP or a compiler with auto-parallelization capabilities. For an execution in a multi-precision environment, a dedicated accelerator system consisting of multiple FPGA (Field Programmable Gate Array) boards was developed and its performance results were presented by Daisaka et al. [48].

4. 2-loop integrals with massive internal lines

In this section we calculate the integral \( I \) of Eq (5) for \( L = 2 \) and \( n = 4 - 2\epsilon \) according to

\[
I = (-1)^{N} \Gamma(N - 4 + 2\epsilon) \int_{0}^{1} \prod_{\alpha=1}^{N} dx_{\alpha} \delta(1 - \sum_{\alpha} x_{\alpha}) \frac{1}{U^{2-\epsilon}(V - i\varrho)^{N-4+2\epsilon}} \quad (16)
\]

IR divergence occurs through a singularity arising when \( V \) vanishes at the boundaries of the domain. This problem can be addressed by dimensional regularization with \( n = 4 + 2\epsilon \), which we implemented numerically in [11, 12, 9] using an extrapolation as \( \epsilon \to 0 \) (\( \epsilon > 0 \)). It is assumed that the denominator does not vanish in the interior of the integration domain, so we can set \( \varrho = 0 \).

In this paper, we concentrate on UV divergence, which occurs when \( U \) vanishes at the boundaries. The \( \Gamma \)-function in Eq (16) contributes to UV divergence when \( N \leq 4 \). We treat UV divergence by a dimensional regularization with \( n = 4 - 2\epsilon \), implemented by a numerical extrapolation as \( \epsilon \to 0 \) after either iterated integration with DQAGS from QUADPACK, multivariate adaptive integration with PARINT or the DE formula.

4.1. 2-loop self-energy integrals

Fig depicts 2-loop self-energy diagrams with \( N = 3, 4 \) and 5 internal lines. We refer to the 2-loop self-energy diagrams (a-d) as the sunrise-sunset, lemon, half-boiled egg and Magdeburg diagrams, and to the corresponding integrals as \( I_{u}^{S2} \), \( I_{b}^{S2} \), \( I_{c}^{S2} \) and \( I_{d}^{S2} \), respectively. As show in in Fig 2 the entering momentum is \( p \), and we denote \( s = p^{2} \).
Note that the value of \( \kappa \) we find that, if divergent as \( 1/\varepsilon \), we give the analytic formula for \( \kappa \) for the extrapolation, then the first coefficient converges to \( C_{\varepsilon-1} = 0 \).

Analytic results for the integrals have been derived by many authors. We use the following formulas for the functions \( U \) and \( V \) in Eq (16):

- for the sunrise-sunset diagram (Fig 2(a)), \( U = x_1 x_2 + x_2 x_3 + x_3 x_1, \quad W/s = x_1 x_2 x_3; \)
- for the lemon diagram (Fig 2(b)), we have \( U = x_1 x_3 x_4 + x_1 x_2, \quad W/s = x_4 (x_1 x_2 + x_2 x_3 + x_3 x_1); \)
- for the half-boiled egg diagram (Fig 2(c)), \( U = x_1 x_3 x_4 + x_1 x_2, \quad W/s = x_5 (x_1 x_2 + x_2 x_3 + x_3 x_1); \)
- for the Magdeburg diagram (Fig 2(d)), \( U = x_1 x_4 x_5 + x_3 x_1 x_2 + x_3 x_1 x_5 + x_3 (x_1 x_5 + x_2 x_4). \)

In the numerical evaluation, we take particular values for energy and masses, i.e., \( s = p^2 = 1 \), and all masses \( m_r = 1 \) in order to make comparisons with results in the references.

The integrals are expanded with respect to the dimensional regularization parameter \( \varepsilon \). The integrals \( I_2^a, I_2^b \) are divergent as \( 1/\varepsilon^2 \), which is the product of \( 1/\varepsilon \) from the \( \Gamma \)-function part and \( 1/\varepsilon \) from the integral part. The integral \( I_2^c \) is divergent as \( 1/\varepsilon \), which comes from the integral part. The integral \( I_2^d \) is finite. The expansions are of the form of Eq (17)

\[
S(\varepsilon) \sim \sum_{k \geq 0} C_k \varepsilon^k \quad \text{as} \quad \varepsilon \to 0,
\]

and we use linear extrapolation to approximate the coefficients of the leading terms. We multiply the integrals \( I_2^a, I_2^b \) and \( I_2^c \) of Eq (16) with the factor \( \Gamma(1 + \varepsilon)^{-2} \) for comparison with the results in Laporta (1), since the latter are computed with this factor. The half-boiled egg diagram is not covered in [1]. We give the analytic formula for \( J_2^c \) in Appendix A of this paper.

\[
I_2^a(\varepsilon) \Gamma(1 + \varepsilon)^{-2} = \sum_{k \geq 2} C_k \varepsilon^k = -1.5 \varepsilon^{-2} - 4.25 \varepsilon^{-1} - 7.375 - 17.22197253479 \varepsilon \ldots
\]

\[
I_2^b(\varepsilon) \Gamma(1 + \varepsilon)^{-2} = \sum_{k \geq 2} C_k \varepsilon^k = 0.5 \varepsilon^{-2} + 0.686206357658 \varepsilon^{-1} - 0.6868398873414
+ 1.486398391913 \varepsilon \ldots
\]

\[
J_2^c(\varepsilon) = \sum_{k \geq 2} C_k \varepsilon^k = 0.6045997880781 \varepsilon^{-1} - 0.175697002260 - 0.2977242542666 \varepsilon + 0.4140155361099 \varepsilon^2 \ldots
\]

\[
I_2^d(\varepsilon) \Gamma(1 + \varepsilon)^{-2} = \sum_{k \geq 0} C_k \varepsilon^k = 0.9236318265199 - 1.284921671848 \varepsilon + 2.689507626490 \varepsilon^2
- 5.338399227511 \varepsilon^3 \ldots
\]

Note that the value of \( \kappa \) in Eq (17) corresponds with the index of the first coefficient \( C_\kappa \) in the expansion. In that case we find that, if \( \kappa \) is replaced by \( \kappa - 1 \) for the extrapolation, then the first coefficient converges to \( C_{\kappa-1} = 0 \).
Table 1: Results UV sunrise-sunset integral, $I_B^2(1+e)^{-2}$ (on Mac Pro), rel. err. tol. $t_r = 10^{-9}$ (outer), $10^{-9}$ (inner), $T[x] = \text{Time (elapsed user time in s)}$, $e = 1.2 \times 10^{-4}$ (starting at 1.2\times 10^{-4}), $E_r = \text{outer integration estim. rel. error}$

| $b$   | $E_r$ | $T[s]$ | Res. $C_2$ | Res. $C_1$ | Res. $C_0$ | Res. $C_1$ |
|-------|-------|---------|-------------|-------------|-------------|-------------|
| 0     | 3.2e-10 | 0.015 |             |             |             |             |
| 1     | 4.7e-10 | 0.013 | -1.156740414 | -8.2070492 |             |             |
| 2     | 6.6e-10 | 0.013 | -1.603088961 | -2.1260993 | -20.5343    |             |
| 3     | 9.2e-10 | 0.013 | -1.476661131 | -4.9718825 | 0.60362     | -51.7769    |
| 4     | 4.0e-11 | 0.028 | -1.504342324 | -4.0584835 | -10.6252    | 8.7336      |
| 5     | 6.9e-11 | 0.029 | -1.499360223 | -4.2880409 | -6.4634     | -28.3731    |
| 6     | 1.3e-10 | 0.029 | -1.500078550 | -4.2438757 | -7.5734     | -13.7781    |
| 7     | 2.2e-10 | 0.028 | -1.499992106 | -4.2507887 | -7.3415     | -18.0041    |
| 8     | 3.6e-10 | 0.028 | -1.500000763 | -4.2499043 | -7.3801     | -17.0657    |
| 9     | 5.4e-10 | 0.029 | -1.499999896 | -4.2500174 | -7.3537     | -17.2650    |
| 10    | 7.8e-10 | 0.029 | -1.500000026 | -4.2499488 | -7.3544     | -17.2010    |

Table 2: Results UV lemon integral, $I_B^2(1+e)^{-2}$ (on Mac Pro), rel. err. tol. $t_r = 10^{-10}$ (outer), $5 \times 10^{-11}$ (inner two), $T[x] = \text{Time (elapsed user time in s)}$, $e = 1/b_2$ (starting at 1/4), $E_r = \text{outer integration estim. rel. error}$

| $b_2$  | $T[s]$ | $E_r$ | $C_2$ | $C_1$ | $C_0$ | $C_1$ |
|--------|---------|-------|-------|-------|-------|-------|
| 4      | 3.5e-11 | 0.36  |       |       |       |       |
| 6      | 8.8e-11 | 0.34  | 0.513822162587 | 0.5246760722 |
| 12     | 3.4e-12 | 0.41  | 0.500378829119 | 0.67218881764 | -0.518724512 | 0.52008986 |
| 16     | 1.5e-11 | 0.39  | 0.50048568101347 | 0.6836889795 | -0.643173699 | 1.08075772 |
| 24     | 4.7e-11 | 0.38  | 0.500037252535 | 0.6859187289 | -0.679195194 | 1.37495588 |
| 32     | 4.1e-11 | 0.43  | 0.5000002197177 | 0.66617780639 | -0.685884585 | 1.46545941 |
| 48     | 1.4e-11 | 0.44  | 0.5000000875318 | 0.6861930431 | -0.686757991 | 1.48373011 |
| 64     | 3.2e-11 | 0.31  | 0.5000000002937 | 0.6862065333 | -0.686834711 | 1.48641633 |
| 96     | 3.2e-11 | 0.31  | 0.500000000039 | 0.6682063534 | -0.686839782 | 1.4869673 |

Table 3: Results UV half-boiled egg integral, $I_B^2(1+e)^{-2}$ (on Mac Pro), rel. err. tol. $t_r = 10^{-12}$ (outer), $5 \times 10^{-13}$ (inner three), $T[x] = \text{Time (elapsed user time in s)}$, $e = 1/b_2$ (starting at 1/0), $E_r = \text{outer integration estim. rel. error}$

| $b_2$ | $E_r$ | $T[s]$ | $C_1$ | $C_0$ | $C_1$ | $C_2$ |
|-------|-------|---------|-------|-------|-------|-------|
| 1     | 4.2e-13 | 7.3   |       |       |       |       |
| 2     | 3.9e-14 | 11.4  | 0.612179532370 | -0.2689337928 |       |       |
| 3     | 2.6e-13 | 10.8  | 0.6219220162954 | -0.29816083105 | -0.01948967 |       |
| 4     | 5.2e-13 | 10.3  | 0.6084345649676 | -0.21723612309 | -0.128876997 | 0.08092471 |
| 6     | 1.1e-13 | 14.9  | 0.6058661595977 | -0.18355206939 | -0.246771815 | 0.24934498 |
| 8     | 7.9e-13 | 9.2   | 0.6046439346384 | -0.17679647604 | -0.286825256 | 0.35912347 |
| 12    | 9.6e-13 | 12.5  | 0.6046028724387 | -0.17581097725 | -0.286834926 | 0.41006691 |
| 16    | 2.3e-13 | 19.4  | 0.6045999612478 | -0.17570671438 | -0.297520457 | 0.41176667 |
| 24    | 8.7e-13 | 19.1  | 0.6045997948488 | -0.17569752162 | -0.297707921 | 0.41374216 |
| 32    | 7.7e-13 | 24.9  | 0.6045997882885 | -0.17569702364 | -0.297723238 | 0.41399119 |
| 48    | 4.1e-13 | 33.2  | 0.6045997880782 | -0.17569700033 | -0.297724241 | 0.41401488 |

$\epsilon = 10^{-9} \text{ (outer)}, 10^{-9} \text{ (inner)}$, $E_r = \text{outer integration estim. rel. error}$
Table 4: Results UV Magdeburg integral, $I_d^2 \Gamma(1 + \varepsilon)^{-2}$ (by PARINT on thor in long double precision), rel. err. tol. $t_c = 10^{-13}$, max. # evals = 1B, $T[s]$ = Time (elapsed user time in s), $\varepsilon = 2^{-t}$ (starting at 1.0), $E_r$ = estim. rel. error

| $\ell$ | $E_r$ | $T[s]$ | $C_0$ | $C_1$ | $C_2$ | $C_3$ |
|-------|-------|--------|-------|-------|-------|-------|
| 0     | 8.5e-14 | 0.8    | 0.69130084611470 | -0.142989490499 |
| 1     | 1.0e-13 | 19.7   | 0.84949643770104 | -0.617576265258 |
| 3     | 4.8e-13 | 6.9    | 0.90878784940610 | -1.032616144771 |
| 4     | 8.5e-13 | 6.6    | 0.92198476262012 | -1.230969848171 |
| 5     | 1.2e-12 | 6.5    | 0.92353497440505 | -1.278626506507 |
| 6     | 3.0e-12 | 6.6    | 0.9236889210499   | -1.284521126000 |
| 7     | 2.4e-12 | 6.6    | 0.92363178137723 | -1.284910070175 |
| 8     | 2.8e-12 | 6.6    | 0.923631826117066 | -1.284921492347 |
| 9     | 2.8e-12 | 6.6    | 0.92363182651847 | -1.284921670382 |
| 10    | 2.9e-12 | 6.6    | 0.92363182651995 | -1.284921671903 |
| 11    | 2.9e-12 | 6.6    | 0.92363182651990 | -1.284921671790 |
| 12    | 2.8e-12 | 6.6    | 0.92363182651992 | -1.284921671989 |
| 13    | 3.7e-12 | 6.6    | 0.92363182651991 | -1.284921671840 |
| 14    | 2.9e-12 | 6.6    | 0.92363182651991 | -1.284921671840 |

Integrals $P_d^2$, $C_0$, $C_1$, $C_2$ and $C_3$ are obtained using extrapolation. To evaluate $I_d^2$, we transform the variables as:

$$\begin{align*}
x_1 &= y_1 y_3 y_4, & x_2 &= y_1 y_3 y_4 m, \\
x_3 &= y_1 y_2, & x_4 &= y_1 y_3,
\end{align*}$$

with $y_{im} = 1 - y_i$ and Jacobian $y_1 y_2^{1-y_1} y_3^{y_1}$. The accuracy and time of the calculation of the integral sequence for the extrapolation in Table 4 are improved considerably by the transformation.

The first three integrals, an iterated integration is applied with DQAGS from QUADPACK, on a Mac Pro, 2.6 GHz Intel Core i7, with 16 GB memory, under OS X. The value of $E_r$ is (the absolute value of) the estimated relative error returned by the outer integration (not accounting for the inner integration error). It is listed for each integration, as well as the elapsed user time $T[s]$ (in seconds). The time for the extrapolation is negligible compared to that of the integration. We use a standard linear system solver to solve very small systems (of sizes $2 \times 2$ up to around 15 x 15 for the cases in this paper).

Table 4 illustrates an application of PARINT for the Magdeburg integral, on the thor system of the High Performance Computing and Big Data Center at WMU, with dual Intel Xeon E5-2670, 2.6 GHz processors, 16 cores and 128 GB of memory per node. For the distributed computation with PARINT, using 16 processes per node and 64 processes total, the MPI host file has four lines of the form $nx slots=16$ where $nx$ represents a selected node name. The running time is reported (in seconds) from PARINT, and comprises all computation not inclusive of the process spawning and PARINT initialization at the start of the program. The cubature rule of degree 9 is used for integration over the subregions (see Section 3.2), to an allowed maximum number of one billion (1B) integrand evaluations over all processes, and a requested accuracy of $t_r = 10^{-13}$ in long double precision. The total estimated relative error is denoted by $E_r$, in Table 4.

The extrapolation parameter for Tables 4 and 3 adheres to $\{1/\ell\}$ where $\ell$ is the Bulirsch sequence $25$ started at an early index. Tables 1 and 4 give results for a geometric sequence of the extrapolation parameter $\varepsilon = 2^{-t}$. The convergence results in Tables 1 and 4 are compared with with the expansion coefficients available from 1 (see Eqs 15-19, 21). Table 4 shows excellent approximations to the analytic result of Eq 20, derived in the Appendix.

Throughout the extrapolation we keep track of the difference with the previous result as a measure of convergence. Increases of the distance between successive extrapolation results are an indicator that the convergence is no longer improving and the procedure can be terminated.
sequence and linear system solver are shown in Table 5. Tables 6–7 show results for bases $1/1.2$ and $1/1.5$, respectively, achieved by PARI against the loss of significant digits near the boundaries. Both deliver very accurate results, with the final results in Table 6 slightly closer to the analytic values. The extrapolation in Table 7 converges somewhat faster. For the computation of $I_2$, we transform the variables as for $I_b$, and multiply the integrals with a factor $\Gamma(1+\varepsilon)^2$. The expansions from Laporta’s results [11] are:

\[
I_{2a}(\varepsilon) \Gamma(1+\varepsilon)^2 = \sum_{k\geq-2} C_k \varepsilon^k = 0.5 \varepsilon^{-2} + 0.6862006357658 \varepsilon^{-1} - 0.5916667014024
+ 1.356196533114 \varepsilon \ldots
\]

(23)

\[
I_{2b}(\varepsilon) \Gamma(1+\varepsilon)^2 = \sum_{k\geq-1} C_k \varepsilon^k = 0.671253105748 \varepsilon^{-1} - 0.08774519609257 + 0.7262375626947 \varepsilon
- 1.32112948587 \varepsilon^2 \ldots
\]

(24)

The numerical results for $I_{2a}$ obtained with iterated integration by DQAGS, and extrapolation using a Bulirsch sequence and linear system solver are shown in Table 5. Tables 6–7 show results for $I_{2b}$ with geometric sequences of base $1/1.2$ and $1/1.5$, respectively, achieved by PARINT using 64 processes on four 16-core nodes of the thor cluster. Both deliver very accurate results, with the final results in Table 6 slightly closer to the analytic values. The extrapolation in Table 7 converges somewhat faster. For the computation of $I_{2b}$, we transform the variables as for $I_{2a}$ in Section 4.1. This transformation maps the integration domain to the 4-dimensional unit cube and also guards against the loss of significant digits near the boundaries.
Approximation, relative error estimate \( E \) (64 processes) are given in Table 9, as well as the corresponding double precision results. Listed are: the integral for an increasing allowed maximum number of evaluations and increasingly strict (relative) error tolerance. For instance, the times of the \( N \) million \( t \) at \( p \) are expressed in seconds, as a function of the number of MPI processes \( N \rightarrow \) (starting at 1.0), \( E_\ell \) = estim. rel. error.

| \( \ell \) | \( E_\ell \) | \( T[s] \) | \( \text{Res. } C_{-1} \) | \( \text{Res. } C_{0} \) | \( \text{Res. } C_{1} \) | \( \text{Res. } C_{2} \) |
|---|---|---|---|---|---|---|
| 8 | 9.9e-14 | 0.7 | | | | |
| 9 | 5.7e-14 | 0.9 | 0.653547537693 | 0.1087344298 | | |
| 10 | 8.7e-14 | 1.3 | 0.667736720294 | -0.02549804326 | 0.148227487 | |
| 11 | 5.7e-14 | 1.7 | 0.670486635108 | -0.06852379156 | 0.536821659 -0.37763368 | |
| 12 | 9.8e-14 | 6.9 | 0.671129876266 | -0.9297974141 | 0.88633901 -0.8392737 | |
| 13 | 9.8e-14 | 8.4 | 0.671253102420 | -0.06875821868 | 0.707808436 -1.13401646 | |
| 14 | 9.6e-14 | 9.3 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |
| 15 | 9.4e-14 | 10.3 | 0.671252763777 | -0.08774518885 | 0.72623681 -1.32108845 | |
| 16 | 9.6e-14 | 11.2 | 0.671253102420 | -0.06875821868 | 0.707808436 -1.13401646 | |
| 17 | 9.4e-14 | 12.0 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |
| 18 | 1.0e-13 | 13.8 | 0.671253102420 | -0.06875821868 | 0.707808436 -1.13401646 | |
| 19 | 1.0e-13 | 14.7 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |
| 20 | 1.0e-13 | 15.6 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |
| 21 | 1.0e-13 | 16.5 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |
| 22 | 1.0e-13 | 17.4 | 0.671250129509 | -0.8757395495 | 0.148227487 -0.37763368 | |

The 2-loop box integrals according to Eq (24): 0.671253105748 -0.08774519609 0.726237563 -1.32 112949. The transformation of Eq (7) was used. The results for an increasing allowed maximum number of evaluations and increasingly strict (relative) error tolerance \( t_r \) (starting at 1.0), \( E_\ell \) = estim. rel. error.

| \( \ell \) | \( E_\ell \) | \( T[s] \) | \( \text{Res. } C_{-1} \) | \( \text{Res. } C_{0} \) | \( \text{Res. } C_{1} \) | \( \text{Res. } C_{2} \) |
|---|---|---|---|---|---|---|
| 0 | 1.0e-13 | 0.31 | 0.552646148416 | -0.03261093175 | | |
| 1 | 1.0e-13 | 0.77 | 0.645884613605 | -0.09301314549 | 0.139857698 | |
| 2 | 3.9e-14 | 0.44 | 0.659977890791 | -0.02607008786 | 0.240272298 -0.04756481 | |
| 3 | 4.9e-14 | 0.92 | 0.668110702931 | -0.00090901708 | 0.248597729 -0.0705818 | |
| 4 | 9.8e-14 | 4.32 | 0.670968634734 | -0.02729551667 | 0.588431949 -0.225263918 | |
| 5 | 1.2e-14 | 1.96 | 0.67115085836 | -0.8495659592 | 0.680218087 -0.98346458 | |
| 6 | 9.5e-14 | 13.9 | 0.67122836395 | -0.87718672678 | 0.715417521 -1.20334533 | |
| 7 | 1.0e-13 | 77.7 | 0.671250129509 | -0.87740895529 | 0.7262048133 -1.31636903 | |
| 8 | 9.3e-15 | 26.8 | 0.671253105451 | -0.8774988285 | 0.726228203 -1.32059155 | |
| 9 | 2.1e-13 | 78.3 | 0.671253105720 | -0.8774518885 | 0.726236881 -1.32108845 | |
| 10 | 4.9e-13 | 79.5 | 0.671253105451 | -0.8774988285 | 0.726228203 -1.32059155 | |

For the two-loop crossed box problem as an example, we ran PARINT in long double precision. The results for an increasing allowed maximum number of evaluations and increasingly strict (relative) error tolerance \( t_r \) (using 64 processes) are given in Table 8 as well as the corresponding double precision results. Listed are: the integral approximation, relative error estimate \( E_\ell \), number of function evaluations reached and time taken in long double precision.
Figure 4: 2-loop box diagrams with massive internal lines (finite diagrams) (a) double-triangle $N = 5$ (Laporta [1], Fig 4(c)), (b) tetragon-triangle $N = 6$ (Laporta [1], Fig 4(d)), (c) pentagon-triangle $N = 7$ (Laporta [1], Fig 4(g)), (d) ladder $N = 7$ (Laporta [1], Fig 4(h)), (e) crossed ladder $N = 7$ (Laporta [1], Fig 4(i)).

Table 8: Test specifications and range of times in Fig 5(a)-(d)

| DIAGRAM       | FIGURE/TIMING PLOT | $N$ | TOL  | MAX EVALS | $E_1$ [s] | $E_{64}$ [s] | SPEEDUP $S_p$ for $p = 64$ |
|---------------|--------------------|-----|------|-----------|-----------|-------------|-----------------------------|
| double triangle | Fig 4(a) / Fig 5(a) | 5   | $10^{-10}$ | 400M | 32.6 | 0.74 | 44.1 |
| crossed ladder | Fig 4(e) / Fig 5(a) | 7   | $10^{-7}$ | 300M | 27.6 | 0.49 | 56.3 |
| tetragon triangle | Fig 4(b) / Fig 5(b) | 6   | $10^{-9}$ | 3B  | 213.6 | 5.06 | 42.2 |
| ladder         | Fig 4(d) / Fig 5(b) | 7   | $10^{-8}$ | 2B  | 189.9 | 4.33 | 43.9 |
| pentagon triangle | Fig 4(c) / Fig 5(c) | 7   | $10^{-8}$ | 5B  | 507.9 | 8.83 | 57.5 |
| crossed ladder | Fig 4(e) / Fig 5(d) | 7   | $10^{-9}$ | 20B | 1892.5 | 34.6 | 54.7 |

precision, followed by the relative error estimate, number of function evaluations reached and running time in double precision. For a comparable number of function evaluations, the time using long doubles is slightly less than twice the time taken using doubles. The iflag parameter returns 0 when the requested accuracy is assumed to be achieved, and 1 otherwise. Reaching the maximum number of evaluations results in abnormal termination with iflag = 1. The integral approximation for the double computation is not listed in Table 7; it is consistent with the long double result within the estimated error (which appears to be over-estimated). Using doubles the program terminates abnormally for the requested relative accuracy of $t_r = 10^{-10}$. Normal termination is achieved in this case around 239B evaluations with long doubles.

Fig 5 shows PARINT timing plots for the diagrams of Fig 4, depicting a considerable time decrease as a function of the number of processes $p$. Plots with similar orders of the time are grouped in Fig 5(a) and in Fig 5(b).

Timing results obtained with parallel (multi-threaded) iterated integration were given in [18].

Table 9: PARINT long double and double results for crossed diagram Fig 4(e)

| $t_r$ | Max Evals | INTEGRAL APPROX | $E_4$ | #EVALS | INTEGRAL APPROX | $E_4$ | #EVALS |
|-------|-----------|-----------------|-------|--------|-----------------|-------|--------|
| $10^{-10}$ | 600M | 0.085351397948123 | 3.5e-07 | 1 | 6000009793 | 1.7 | 3.4e-07 | 1 | 600001115 | 0.95 |
| 1B | 0.085351397753978 | 1.7e-07 | 1 | 1000000141 | 2.9 | 1.6e-07 | 1 | 1000000141 | 1.6 |
| 2B | 0.085351398047799 | 2.9e-08 | 1 | 2000000443 | 6.0 | 2.9e-08 | 1 | 2000000765 | 3.3 |
| 6B | 0.085351398130559 | 5.6e-09 | 0 | 4164029999 | 14.3 | 8.0e-09 | 0 | 424355239 | 8.6 |
| 10B | 0.085351398134365 | 5.5e-09 | 1 | 10000001571 | 29.7 | 5.5e-09 | 1 | 1000000297 | 16.4 |
| 50B | 0.085351398152623 | 9.9e-10 | 0 | 35701321579 | 124.4 | 4.5e-10 | 0 | 979663859 | 16.0 |
| 100B | 0.085351398153315 | 5.5e-10 | 1 | 8000001137 | 240.2 | 5.5e-10 | 1 | 800000171 | 133.5 |
| 300B | 0.085351398157357 | 4.1e-10 | 1 | 1000000093 | 302.0 | 4.1e-10 | 1 | 1000000093 | 168.3 |
| 1400B | 0.085351398158798 | 9.1e-11 | 0 | 23885496513 | 642.3 | 3.1e-10 | 1 | 3000000279 | 587.2 |
5. 3-loop self-energy integrals

In this section we deal with the integral determined by Eq (5) for $L = 3$ and $n = 4 - 2\varepsilon$,

$$ I = (-1)^N \Gamma(N - 6 + 3\varepsilon) \int_0^1 \prod_{i=1}^N dx_i \delta(1 - \sum x_i) \frac{1}{U^{2x}(V - xq)^{N-6+3\varepsilon}}. $$  (25)

UV divergence occurs when $U$ vanishes at the boundaries. The $\Gamma$-function in Eq (25) contributes to UV divergence when $N \leq 6$. As shown in the figures, the entering momentum is $p$, and we denote $s = p^2$.

We calculate integrals adhering to Eq (25), denoted by $I^3_a$, $I^3_b$, $I^3_c$, $I^3_d$, for the diagrams of Fig 6(a)-(j), respectively. In the following four subsections, we consider massless/massive internal lines and UV finite/divergent cases.

5.1. 3-loop finite integrals with massless internal lines

The integrals $I^3_a$, $I^3_b$, $I^3_c$, $I^3_d$ are finite, with the $U$, $W$ functions given in Eqs (26)-(29) below.

\[ U = x_4 x_7 x_{123} x_{56} + x_{12} x_4 x_7 x_{56} + x_3 (x_{12} x_4 + x_5 x_6 - x_1 x_2 x_3) \]  
\[ W/s = x_4 x_7 x_{15} x_{23} x_6 + x_4 (x_{12} x_3 x_5 + x_{12} x_3 x_6) + x_7 (x_5 x_6 x_{123} + x_1 x_2 x_3) + x_3 (x_{12} x_5 x_6 + x_{12} x_3 x_6) \]  

\[ U = x_7 (x_{12} x_3 x_4 + x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

\[ U = x_7 (x_{12} x_{34} x_5 + x_3 x_5 x_6) + x_{13} x_2 x_4 x_5 + x_{12} x_3 x_4 \]  
\[ W/s = x_7 (x_{15} x_{26,23} + x_1 x_2 x_5 x_6 + x_1 x_2 x_4 + x_1 x_2 x_3) \]  

15
In this subsection we take $m = 0$ for the internal lines. In the absence of divergences we set $\rho = \varepsilon = 0$ in Eq. (25) for $I_a^3$, $I_b^3$, and show the results in Table 10. However, the integral $I_d^3$ is problematic with $\rho = \varepsilon = 0$ in view of integrand singularities, and we use the extrapolation method with either $\varepsilon = 0$ and in the limit as $\rho \to 0$, or with $\rho = 0$ and in the limit as $\varepsilon \to 0$. The analytic result is the same for the four diagrams (see [2]),

$$
I_a^3 = I_b^3 = I_c^3 = I_d^3 = 20 \zeta_5 = 20.738555102867 \ldots
$$

Figure 6: 3-loop self-energy diagrams with massive and massless internal lines (finite and UV-divergent diagrams), cf. Laporta [1], Baikov and Chetyrkin [2]: (a) $N = 7$, (b) $N = 7$, (c) $N = 8$, (d) $N = 8$, (e) $N = 7$, (f) $N = 8$, (g) $N = 4$, (h) $N = 6$, (i) $N = 7$, (j) $N = 7$

**PARINT** numerical results and timings were given in [49] for the 3-loop diagrams of Fig 6(a)-(d), from runs on 16-core nodes of the *Thor* cluster. For the integration over each subregion, the rule of degree 9 is used (see Section 3.2), which evaluates the 6D integrand at 453 points and the 7D integrand at 717 points. These integrals are transformed to the unit cube according to Eq (7). Table 10 lists the results obtained directly with $\rho = \varepsilon = 0$ for Fig 6(a)-(c) by PARINT using 48 processes: integral approximation, relative error estimate $E_n$, and time in seconds for various total numbers of function evaluations in double and long double precision. Fig 7 shows times and speedups as a function of the number of processes $p$, for a computation of the integrals of Fig 6(a)-(c) using 10B integrand evaluations in double precision. Denoting the time in seconds for $p$ processes by $T_p[x]$, the corresponding speedup given by $S_p = T_1/T_p$ is nearly optimal – which would coincide with the diagonal in the graph, or slightly superlinear ($> p$) over the given range of $p$.

With $\rho = 0$ the integrand has boundary singularities. For example, the integrals of Fig 6(a)-(d) have a zero denominator with $U = 0$ at $x_2 = 1$ and the other variables 0 (on the boundary of the unit simplex). Thus the integrand program codes test for zero denominators. However some of the computations overflow by integrand evaluations in
Table 10: PARINT accuracy and times with 48 procs. for loop integrals of Fig 6(a)-(c) with massless internal lines, using $\rho = 0$, and various numbers of function evaluations; $E_r$ = integration estim. rel. error

| Diagram | # FCN. EVALS. | DOUBLE PRECISION | LONG DOUBLE PRECISION |
|---------|----------------|------------------|-----------------------|
|         | INTEGRAL RESULT | REL. ERR. $E_r$ | TIME $T$ | INTEGRAL RESULT | REL. ERR. $E_r$ | TIME $T$ |
| Exact   | 20.73855510     | 20.73855510      | 16.0     |
| Fig 6(a) 5B | 20.73871652 2.21e-05 | 9.0 | 20.73871522 2.5e-05 | 16.0 |
|         | 10B | 20.73856839 3.50e-06 | 17.9    | 20.73856878 3.42e-06 | 32.1 |
|         | 25B | 20.73855535 3.79e-07 | 44.9    | 20.73855539 3.71e-07 | 80.5 |
|         | 50B | 20.73855508 9.07e-08 | 90.3    | 20.73855508 8.94e-08 | 161.1 |
|         | 75B | 20.73855507 4.26e-08 | 135     | 20.73855507 4.23e-08 | 242.1 |
|         | 100B | 20.73855508 2.59e-08 | 180.8   | 20.73855508 2.56e-08 | 323.2 |
| Fig 6(b) 5B | 20.73933800 3.69e-05 | 9.7 | 20.73933292 3.63e-05 | 17.5 |
|         | 10B | 20.73872210 6.64e-06 | 19.4    | 20.73872078 6.61e-06 | 35.1 |
|         | 25B | 20.73857098 8.32e-07 | 48.6    | 20.73857018 8.12e-07 | 87.9 |
|         | 50B | 20.73845716 1.96e-07 | 98.2    | 20.73845718 1.95e-07 | 175.9 |
|         | 75B | 20.73855576 4.26e-08 | 135.6   | 20.73855575 4.23e-08 | 242.1 |
|         | 100B | 20.73855540 2.59e-08 | 180.8   | 20.73855540 2.56e-08 | 323.2 |
| Fig 6(c) 5B | 20.73914961 1.19e-03 | 10.3 | 20.74196270 1.19e-03 | 19.6 |
|         | 10B | 20.73886434 3.64e-04 | 20.7    | 20.73880437 3.51e-04 | 39.2 |
|         | 25B | 20.73827908 5.04e-05 | 51.7    | 20.73827607 5.04e-05 | 98.2 |
|         | 50B | 20.73841802 1.09e-05 | 103.4   | 20.73842495 9.79e-06 | 196.7 |
|         | 75B | 20.73848662 3.66e-06 | 146.4   | 20.73848624 3.70e-06 | 295.0 |
|         | 100B | 20.73851402 1.96e-06 | 207.5   | 20.73851338 1.98e-06 | 393.9 |

Table 11: Integration with PARINT using 64 procs., max. # evals = 150B, $\rho = \rho_t = 2 - \ell, \ell = 20, 21, \ldots$ and extrapolation with $\epsilon$-algorithm for Fig 6(d) integral with massless internal lines

| Diagram | INTEGRAL [F11] | EXTRAPOLATION |
|---------|----------------|---------------|
|         | INTEGRAL $E_r$ | TIME $T$ | LAST | SELECTED |
| Eq (30) | 20.73855510 | 20.73855510 | 17.8 | 474.5 |

| $\ell$ | INTEGRAL $E_r$ | TIME $T$ | LAST | SELECTED |
|--------|----------------|-----------|------|----------|
| 20     | 19.69036128576084 1.44e-07 | 474.5     |     |          |
| 21     | 19.91633678795658 1.64e-07 | 474.6     |     |          |
| 22     | 20.0925613888053 1.84e-07 | 474.7     | 20.71685142 20.71685142 |
| 23     | 20.2303384092921 1.94e-07 | 474.7     | 20.72393791 20.72393791 |
| 24     | 20.33827222472266 2.16e-07 | 474.7     | 20.73801873 20.73801873 |
| 25     | 20.42997783943613 2.36e-07 | 474.7     | 20.73801873 20.73801873 |
| 26     | 20.4955227010659 2.53e-07 | 474.7     | 20.73815511 20.73815511 |
| 27     | 20.5419420864818 2.74e-07 | 474.7     | 20.73825979 20.73825979 |
| 28     | 20.58321345028519 2.95e-07 | 474.7     | 20.73811441 20.73811441 |
| 29     | 20.61575668045697 3.14e-07 | 474.7     | 20.74104946 20.74104946 |
| 30     | 20.64113527976811 3.34e-07 | 474.6     | 20.73845289 20.73833347 |
| 31     | 20.661712341606668 3.51e-07 | 474.7     | 20.73861767 20.73855952 |
| 32     | 20.67774379215522 4.16e-07 | 474.6     | 20.73855567 20.73847582 |

Eq (10) | 20.73855510 | 20.73855510 | 17.8
Table 12: Integration with PARINT using 64 procs., max. # evals = 100B, $\varepsilon = \varepsilon_\ell = 2^{-\ell}, \ell = 8, 9, \ldots$ and extrapolation with $\varepsilon$-algorithm for Fig 6(d) integral with massless internal lines

| $\ell$ | $I$ | $E_t$ | $T$ | Extrapolation | Selected |
|-------|-----|------|----|--------------|----------|
| 8     | 21.21987706233486 | 8.84e-08 | 648.5 | Eq (30) | 20.73855510 |
| 9     | 20.9772482739239 | 8.69e-08 | 648.0 | | |
| 10    | 20.8574366356065 | 8.61e-08 | 649.0 | 20.74044694 | 20.74044693 |
| 11    | 20.7978756374119 | 8.56e-08 | 647.7 | 20.73903010 | 20.73903010 |
| 12    | 20.7681590836464 | 8.53e-08 | 648.1 | 20.73855734 | 20.73855734 |
| 13    | 20.7533634262632 | 8.39e-08 | 648.3 | 20.73855626 | 20.73855626 |
| 14    | 20.7459576382081 | 8.29e-08 | 647.6 | 20.73855580 | 20.73855580 |
| 15    | 20.7422563903292 | 8.21e-08 | 647.6 | 20.73855592 | 20.73855592 |

Table 11 shows an extrapolation as $\varrho \to 0$ using the $\varepsilon$-algorithm of Wynn [19, 20, 21, 22, 23] (see Section 2.2). The $\varrho_\ell$ geometric sequence is computed with base 2, $\varrho_\ell = 2^{-\ell}$, and the integration is performed in long double precision using 150B evaluation points. The Selected column lists the element along the new lower diagonal that is presumed the best, based on its distance from the neighboring elements as computed by the $\varepsilon$-algorithm function from QUADPACK. The Last column lists the final (utmost right) element computed in the lower diagonal. Overall the $\varepsilon$-algorithm function from QUADPACK appears successful at selecting a competitive element as its result for the iteration.

For an extrapolation as $\varepsilon \to 0$ we set $\varrho = 0$, so that

$$I^3_d = \Gamma(2 + 3\varepsilon) \int \prod_{r=1}^{\delta} dx_r \delta(1 - \sum x_r) \frac{V^2 - \varrho^2}{U^2 + \varrho^2}.$$  \hspace{1cm} (31)
those of Table 11 likely by calling the pow function (in the C programming language) for each integrand evaluation, whereas the integrand of Eq (30) for the q extrapolation can be calculated using only multiplications, divisions, additions and subtractions.

5.2. 3-loop finite integrals with massive internal lines

For a set of 3-loop self-energy integrals with massive lines given in Fig 6 (a)-(f), corresponding numerical results and PARINT performance results are shown in Table 13. The U, W functions for (a)-(d) are given in the previous subsection and those for (e) and (f) are listed in Eqs (32)-(33) below.

\[
U = \sum_{k=1}^{6} \frac{1}{k!} \frac{\partial^k}{\partial \xi^k} \left[ f(x_1, x_2, x_3) \right]
\]

\[
W = \sum_{k=1}^{6} \frac{1}{k!} \frac{\partial^k}{\partial \xi^k} \left[ f(x_1, x_2, x_3) \right]
\]

In order to compare our integral approximations with Laporta’s Table 1, we set all masses \(m_p = 1\) and \(s = 1\), and furthermore divide the integral by \(\Gamma(1 + \epsilon)^3\). The integrals are transformed from the (unit) simplex to the (unit) cube according to the transformation of Eqs (7) and (8) and the integration is taken over the cube, using a basic integration rule of polynomial degree 9 (see Section 3.2) and a maximum total of 10B evaluations. The function evaluations are distributed over all the processes. The absolute tolerance is \(t_a = 5 \times 10^{-10}\) and the maximum number of integrand evaluations is \(10B = 10^{10}\) (which is reached in producing the results of Table 13).

The results in Table 13 are given for \(p = 1\) and for \(p = 64\) MPI processes. \(T_1\) is the time with one process and \(T_{64}\) is the parallel time on the thor cluster with \(p = 64\) processes, distributed over four 16-core, 2.6 GHz compute nodes and using the Infiniband interconnect for message passing via MPI. The speedup \(S_{64} = T_1 / T_{64}\) indicates good scalability of the parallel implementation (see also [52,53]). Note that superlinear speedups (S_{64}) are obtained in some cases, where the speedup exceeds the number of processes. This is partially due to the fact that the timing is done within PARINT after the processes are started. It may also be noted that the adaptive partitioning reaches somewhat more accuracy sequentially. Each process has its own priority queue, keyed with the absolute error estimates over their region. This may lead to unnecessary work by the processes locally, which increases with the number of processes.

Tables 14 and 17 are computed with consecutive calls to pi_integrate() in a loop and linear extrapolation, for the functions depicted in Fig 6 (b) and (f), respectively. The values of \(C_0, C_1\) and \(C_2\) are listed (\(\kappa = 0\) in Eq (3)). Results from extrapolation with the \(\epsilon\)-algorithm are shown in Table 15 for the diagram in Fig 6 (b). More extrapolations are needed with the \(\epsilon\)-algorithm than with linear extrapolation. In this case the latter is more accurate and efficient but utilizes knowledge of the structure of the asymptotic expansion, i.e., that \(\varphi(x) = x^\kappa\); this is not assumed for the non-linear extrapolation with the \(\epsilon\)-algorithm.

Tables 16 and 18 illustrate the vector function integration capability of PARINT to calculate the entry sequence for the extrapolation (as a vector integral result - see (13)) with one call to pi_integrate(). This procedure delivers excellent accuracy and efficiency. Note that the integration of the vector function in Table 16 took 2403.4 seconds, compared to the total time of 3766.6 seconds for the integrations listed in Table 12. With regard to Table 18 the time for integrating the vector function was 2180.2 seconds, vs. the total (sum) of 3426.5 seconds for the iterations in Table 17.

| 3-loop diag | N | Result Laporta | Result \(p = 1\) | Result \(p = 64\) | \(T_1[s]\) | \(T_{64}[s]\) | \(S_{64}\) |
|-------------|---|----------------|-----------------|-----------------|-------------|------------|--------|
| Fig 6 (a)   | 7 | 2.0025000411   | 2.0025000413    | 2.0025000412    | 879.3       | 13.4       | 65.6   |
| Fig 6 (b)   | 7 | 1.34139924145 | 1.34139924147   | 1.3413992416    | 1026.2      | 14.4       | 71.3   |
| Fig 6 (c)   | 8 | 0.2796089228   | 0.2796089227    | 0.2796089200    | 1019.7      | 15.9       | 64.1   |
| Fig 6 (d)   | 8 | 0.1480133036   | 0.1480133036    | 0.1480133026    | 976.6       | 16.4       | 59.5   |
| Fig 6 (e)   | 7 | 1.32644820827  | 1.3264482086    | 1.326448199     | 902.7       | 15.8       | 57.1   |
| Fig 6 (f)   | 8 | 0.1826272375   | 0.1826272372    | 0.1826272368    | 1018.3      | 15.8       | 64.4   |
| \( \ell \) | Integral of Fig 6(b) | Extrapola
tion | \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
|---|---|---|---|---|---|---|
| \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
| 0.1 | 0.101e-10 | 363.8 | 0.1846838362 | -0.0381571632 |
| 0.2 | 0.296e-10 | 953.6 | 0.1846838362 | -0.0381571632 |
| 0.3 | 0.396e-10 | 1353.8 | 0.1846838362 | -0.0381571632 |
| 0.4 | 0.496e-10 | 1753.8 | 0.1846838362 | -0.0381571632 |

Table 14: Integral and leading expansion coefficients using PARINT with 16 procs., and linear extrapolation for 3-loop integral of Fig 6(b). \( \varepsilon \) tol. \( t(s) = 10^{-12} \); max. # evals = 20B, \( \varepsilon = \varepsilon_L = 2^{-\ell} \), \( \ell = 3, 4, \ldots \), \( T(s) \) = elapsed time (s); \( \Delta C \) = integration estim. abs. error.

| \( \ell \) | Integral of Fig 6(f) | Extrapola
tion | \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
|---|---|---|---|---|---|---|
| \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
| 0.1 | 0.101e-10 | 363.8 | 0.1846838362 | -0.0381571632 |
| 0.2 | 0.296e-10 | 953.6 | 0.1846838362 | -0.0381571632 |
| 0.3 | 0.396e-10 | 1353.8 | 0.1846838362 | -0.0381571632 |
| 0.4 | 0.496e-10 | 1753.8 | 0.1846838362 | -0.0381571632 |

Table 15: Integral and leading expansion coefficients using PARINT with 16 procs., and extrapolation with e-algorithm for 3-loop integral of Fig 6(b). \( \varepsilon \) tol. \( t(s) = 10^{-12} \); max. # evals = 20B, \( \varepsilon = \varepsilon_L = 2^{-\ell} \), \( \ell = 3, 4, \ldots \), \( T(s) \) = elapsed time (s); \( \Delta C \) = integration estim. abs. error.

| \( \ell \) | Integral of Fig 6(b) | Extrapola
tion | \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
|---|---|---|---|---|---|---|
| \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
| 0.1 | 0.101e-10 | 363.8 | 0.1846838362 | -0.0381571632 |
| 0.2 | 0.296e-10 | 953.6 | 0.1846838362 | -0.0381571632 |
| 0.3 | 0.396e-10 | 1353.8 | 0.1846838362 | -0.0381571632 |
| 0.4 | 0.496e-10 | 1753.8 | 0.1846838362 | -0.0381571632 |

Table 16: Integral and leading expansion coefficients using one call to PARINT with vector function, 16 procs. and linear extrapolation for 3-loop integral of Fig 6(b). \( \varepsilon \) tol. \( t(s) = 10^{-12} \); max. # evals = 20B, \( \varepsilon = \varepsilon_L = 2^{-\ell} \), \( \ell = 3, 4, \ldots \); \( \Delta C \) = integration estim. abs. error.

| \( \ell \) | Integral of Fig 6(b) | Extrapola
tion | \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
|---|---|---|---|---|---|---|
| \( \varepsilon \) | \( t(s) \) | \( \Delta C_3 \) | \( \Delta C_2 \) |
| 0.1 | 0.101e-10 | 363.8 | 0.1846838362 | -0.0381571632 |
| 0.2 | 0.296e-10 | 953.6 | 0.1846838362 | -0.0381571632 |
| 0.3 | 0.396e-10 | 1353.8 | 0.1846838362 | -0.0381571632 |
| 0.4 | 0.496e-10 | 1753.8 | 0.1846838362 | -0.0381571632 |

Table 17: Integral and leading expansion coefficients using PARINT with 16 procs., and linear extrapolation for 3-loop integral of Fig 6(f). \( \varepsilon \) tol. \( t(s) = 10^{-12} \); max. # evals = 20B, \( \varepsilon = \varepsilon_L = 2^{-\ell} \), \( \ell = 3, 4, \ldots \), \( T(s) \) = elapsed time (s); \( \Delta C \) = integration estim. abs. error.
Table 18: Integral and leading expansion coefficients using one call to \textsc{ParInt} with vector function, 16 procs. and linear extrapolation for 3-loop integral of Fig 6(f), err. tol. \( t_\text{u} = 10^{-12}; \) max. # evals = 20B, \( \varepsilon = \varepsilon_\text{f} = 2^{-7}, \ell = 3, 4, \ldots; E_a = \text{integration estim. abs. error}

| \( \ell \) | \( E_a \) | \( \text{RESULT} \) | \( C_0 \) | \( C_1 \) | \( C_2 \) |
|---|---|---|---|---|---|
| 3 | 0.176693722956553 | 1.3c-e-09 | 0.181468366220 | -0.0381571515 | 0.1426206455 |
| 4 | 0.179083545593469 | 1.63e-09 | 0.182582919590 | -0.0648985351 | 0.1816992047 |
| 5 | 0.1806937908054 | 1.71e-09 | 0.182629521626 | -0.0674669083 | 0.1865403500 |
| 6 | 0.181617679222644 | 1.76e-09 | 0.182627317153 | -0.0674668729 | 0.1865403500 |
| 7 | 0.182366526276538 | 1.81e-09 | 0.182627337223 | -0.0674669083 | 0.1865403500 |
| 8 | 0.182496175679090 | 1.82e-09 | 0.182627337223 | -0.0674669083 | 0.1865403500 |
| 9 | 0.18256152924207 | 1.83e-09 | 0.182627337223 | -0.0674669083 | 0.1865403500 |

Table 19: Results UV 3-loop integral, \( n(\varepsilon)^3 I_{g}^{(3)} \) (on 4 nodes/64 procs thor cluster), abs. err. tol. \( t_\text{u} = 10^{-12}, T[s] = \text{Time (elapsed user time in s)}; \varepsilon = \varepsilon_\text{f} = 2^{-7}, \ell = 9, 8, \ldots , E_a = \text{integration estim. abs. error}

| \( \ell \) | \( E_a \) | \( T[s] \) | \( \text{RESULT} \) | \( C_0 \) | \( C_1 \) | \( C_2 \) |
|---|---|---|---|---|---|---|
| 8 | 2.3c-14 | 0.37 | 0.0277718241800302 | 0.166588879051 | 0.76450588 | 0.76450588 |
| 9 | 1.4c-13 | 0.65 | 0.0277718241800302 | 0.162037166892 | 0.76450588 | 0.76450588 |
| 10 | 1.6c-13 | 0.56 | 0.0277718241800302 | 0.162037166892 | 0.76450588 | 0.76450588 |
| 11 | 1.7c-13 | 1.01 | 0.0277718241800302 | 0.162037166892 | 0.76450588 | 0.76450588 |

5.3. 3-loop UV-divergent integrals with massless internal lines

This section handles the integral associated with the massless diagram of Fig 6(g) (the 3-loop sunrise-sunset diagram named \( P_3 \) in [2]), which has a 1/\( \varepsilon \) singularity in the dimensional regularization parameter, arising from the \( \Gamma \)-function factor \((\Gamma(N - 6 + 3\varepsilon)) \) in Eq (25).

The polynomials \( U \) and \( W \) for \( I_{g}^{(3)} \) are given by

\[
\begin{aligned}
U & = x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4 \\
W/s & = x_1 x_2 x_3 x_4
\end{aligned}
\]  
(34)

We take \( \varphi = 0 \) in Eq (25), and the numerical evaluation is done with \( s = 1 \). In order to compare the result to that of Baikov and Chetyrkin [2], we multiply with the factor \( n(\varepsilon)^4 \) where \( n(\varepsilon) \) is defined (in their footnote 11, p. 193) as

\[
n(\varepsilon) = \frac{\Gamma(2 - 2\varepsilon)}{\Gamma(1 + \varepsilon) \Gamma(1 - \varepsilon)^2},
\]
(35)

leading to the expansion

\[
n(\varepsilon)^3 I_{g}^{(3)} = \frac{1}{36\varepsilon} + \frac{35}{216\varepsilon} + \frac{991}{1296\varepsilon} + \ldots
\]
\[
= 0.027777777777 + 0.162037307307 + 0.764660493827 \varepsilon + \ldots
\]  
(36)

Based on integrations with \textsc{ParInt}, a maximum of 10B function evaluations and an absolute error tolerance of \( 10^{-12} \) (on the computation of the integral \( I_{g}^{(3)} / \Gamma(-2 + 3\varepsilon) \)), the results in Table 19 are produced using linear extrapolation. \textsc{ParInt} returns a 0 error flag for the integrals in the input sequence to the extrapolation, indicating that a successful termination is assumed according to Eqs. (3) or (14) for the requested accuracy.

5.4. 3-loop UV-divergent integrals with massive internal lines

In this subsection we calculate the integrals corresponding to massive diagrams of Fig 6(h)-(j). The integral \( I_{g}^{(3)} \) is divergent as \( 1/\varepsilon \), resulting from the \( \Gamma \)-function factor. On the other hand, the integrals \( I_{i}^{(3)} \) and \( I_{j}^{(3)} \) are divergent as...
Expansions for these integrals from [1] are:

\[
U = x_5 (x_{12}x_{34} + x_3x_6) + x_3x_4x_6 + x_1x_2x_6 + x_1x_4x_6 \]
\[
W/s = x_3 (x_5x_{12}x_{46} + x_1x_2x_{46} + x_2x_4x_6) 
\]  

(h) \[
\begin{align*}
I_1 &= x_6 (x_{12}x_{34} + x_3x_6) + x_5x_6x_{1234} \\
W/s &= x_3x_6 (x_1x_2x_3 + x_1x_3x_4 + x_2x_3x_4 + x_{13}x_{24}x_7) + x_5x_6x_{13}x_{24}
\end{align*}
\]  

(i) \[
\begin{align*}
U &= x_3x_6 (x_{12}x_{45} + x_{1245}x_6) + x_3x_7x_{126} \\
W/s &= x_3x_7 (x_1x_2x_4 + x_1x_2x_5 + x_1x_4x_5 + x_2x_4x_5 + x_{14}x_{25}x_6) \\
&+ x_3x_7 (x_1x_{25} + x_2x_5 + x_{25}x_6)
\end{align*}
\]  

(j) \[
\begin{align*}
U &= x_3x_7 (x_{12}x_{45} + x_{1245}x_6) + x_3x_7x_{126} \\
W/s &= x_3x_7 (x_1x_2x_4 + x_1x_2x_5 + x_1x_4x_5 + x_2x_4x_5 + x_{14}x_{25}x_6) \\
&+ x_3x_7 (x_1x_{25} + x_2x_5 + x_{25}x_6)
\end{align*}
\]  

Expansions for these integrals from [1] are:

\[
I_1^S (\varepsilon) = \sum \frac{C_i \varepsilon^k}{k!} = 2.404113806319 \varepsilon^{-1} - 9.7634244476 + 34.99888166 \varepsilon - 116.0420478 \varepsilon^2 \ldots \]  

(40)

\[
I_2^S (\varepsilon) = \sum \frac{C_i \varepsilon^k}{k!} = 0.923631826520 \varepsilon^{-1} - 2.4234916344 + 8.3813497101 \varepsilon - 26.99362122 \varepsilon^2 \ldots \]  

(41)

\[
I_3^S (\varepsilon) = \sum \frac{C_i \varepsilon^k}{k!} = 0.923631826520 \varepsilon^{-1} - 2.161697185 + 6.9295446853 \varepsilon - 21.50327838 \varepsilon^2 \ldots \]  

(42)

The evaluation is performed with \( s = 1 \), \( m_s = 1 \). Numerical results by PARINT on the thor cluster, for the asymptotic expansion coefficients of \( I_1^S (\varepsilon) \) in Eq (40), are listed in Table 20. A geometric sequence in base \( 2^{-1} \) is used for \( \varepsilon \).

For the computation of \( I_1^S \) and \( I_3^S \), the variables are transformed as:

\[
\begin{align*}
x_1 &= y_1 y_2 y_3 y_5, \\
x_2 &= y_1 y_2 y_4 y_5 m, \\
x_3 &= y_1 y_2 y_4 m y_6, \\
x_4 &= y_1 y_2 y_4 m y_6 m, \\
x_5 &= y_1 y_3, \\
x_6 &= y_1 y_3 m, \\
x_7 &= y_1 y_2 m 
\end{align*}
\]  

(43)

with \( y_{im} = 1 - y_i \) and Jacobian \( y_1 y_2 y_3 y_4 y_5 m \) for the former, and

\[
\begin{align*}
x_1 &= y_1 y_2 y_4, \\
x_2 &= y_1 y_2 y_4 m, \\
x_3 &= y_1 y_3 y_6, \\
x_4 &= y_1 y_2 y_5 y_6, \\
x_5 &= y_1 y_2 y_5 m, \\
x_6 &= y_1 y_3 y_6, \\
x_7 &= y_1 y_2 y_6 m 
\end{align*}
\]  

(44)

with \( y_{im} = 1 - y_i \) and Jacobian \( y_1 y_2 y_3 y_4 y_5 m \) for the latter. These variable transformations are beneficial to smoothen the integrand boundary singularities. The above two variable transformations can be adopted for both \( I_3^S \) and \( I_1^S \). However, the transformation (43) works better for \( I_1^S \) and (44) works better for \( I_3^S \).

Numerical results achieved with DE on Intel(R) Xeon(R) E5-2687W v3 3.10 GHz are shown in Tables 21 and 22. Using IEEE 754-2008 binary 128, extensive computation times are incurred (28 hours per iteration for Table 21 and 10 hours for Table 22), as a trade-off for high accuracy. Similar or slightly less accuracy but far shorter computation times (between 670 and 2540 seconds per iteration) are reported in [5], using PARINT in long double precision on 4 nodes and 16 MPI processes per node of the thor cluster.
Table 20: Results UV 3-loop integral of Fig [b] (on 4 nodes with 16 proc per node of i thor cluster), err. tol. $\epsilon_v = 10^{-12}$, $T[s] =$ Time (elapsed user time in s); $\epsilon = \epsilon_v = 2^{-\ell}$, $\ell = 2, 3, \ldots$ $E_r =$ integration estim. abs. error

| $\ell$ | $E_r$ | $T[s]$ | RESULT $C_{-1}$ | RESULT $C_0$ | RESULT $C_1$ | RESULT $C_2$ |
|-------|-------|---------|-----------------|----------------|----------------|----------------|
| 2     | 5.6e-10| 17.6    | 1.99863419875   | -3.5322810037 |
| 3     | 1.6e-09| 28.3    | 2.33027359814   | -7.4944882830 | 10.87234608 |
| 4     | 2.0e-09| 28.3    | 2.397223385974  | -9.3281362959 | 25.9019498999|
| 5     | 3.6e-09| 28.3    | 2.403788052525  | -9.7221078725 | 33.25460667   | -84.769999  |
| 7     | 7.0e-09| 28.3    | 2.404106076736  | -9.7634244078 | 34.83180676   | -110.0084  |
| 8     | 7.4e-09| 28.3    | 2.404113714590  | -9.7633771638 | 34.99091852   | -115.46366 |
| 9     | 7.7e-09| 28.3    | 2.404113805515  | -9.7634281383 | 34.99868012   | -116.01362 |
| 10    | 8.5e-09| 28.3    | 2.404113806117  | -9.7634444078 | 34.99888129   | -116.04239 |

Table 21: Results UV 3-loop diagram of Fig [c] with massive internal lines, using 36 threads on Intel(R) Xeon(R) E5-2687W v3 3.10GHz. DE is applied with mesh size $h = 0.1265988$ and number of evaluations $N_{eval} = 104$ (cf., Eq [15]). For extrapolation, $\epsilon = \epsilon_v = 1.15^{-\ell}$, $\ell = 17, 18, \ldots$

| $\ell$ | Result | $C_{-1}$ | Result | $C_0$ | Result | $C_1$ | Result |
|-------|--------|----------|--------|------|--------|------|--------|
| 17    | 1.8    | 0.89291935327 | -1.506520215 |
| 18    | 1.9    | 0.891526741612 | -2.187886833 | 0.45102458 |
| 19    | 2.0    | 0.92289914325 | -2.375404012 | 7.17894748 | -12.57809 |
| 21    | 1.9    | 0.92353815728 | -2.413584626 | 8.09798189 | 21.89098 |
| 22    | 2.0    | 0.92462147849 | -2.422338750 | 8.32777907 | -25.65197 |
| 23    | 1.9    | 0.92462082823 | -2.423351621 | 8.37298426 | -26.71586 |
| 24    | 1.8    | 0.9263174411 | -2.423477056 | 8.38025256 | -26.94683 |
| 25    | 1.9    | 0.9263182249 | -2.424390372 | 8.38122798 | -26.98707 |
| 26    | 1.9    | 0.9263182726 | -2.42491161 | 8.38131792 | -26.99177 |

Eq [43] 0.9263182652 -2.42491634 8.38149711 -26.99582

6. 4-loop self-energy integrals with massless internal lines

In this section we calculate the integral in Eq [5] for $L = 4$ and $n = 4 - 2\epsilon$,

$$I = (-1)^N \Gamma(N - 8 + 4\epsilon) \int_0^1 \prod_{i=1}^N dx_i \delta \left(1 - \sum_{i=1}^N x_i \right) \frac{1}{U^{2-\epsilon}(V - \zeta)^N - \epsilon^{4+4\epsilon}}. \quad (45)$$

UV divergence occurs when $U$ vanishes at the boundaries. The $\Gamma$-function in Eq [45] contributes to UV divergence when $N \leq 8$. As show in in figures, the entering momentum is $p$, and we denote $s = p^2$.

We address the integrals adhering to Eq [45] for the diagrams of Fig [b] which are denoted by $I_{s4}^4$, $I_{b4}^4$, $I_{e4}^4$, $I_{d4}^4$. We only consider the massless case, i.e., $m_r = 0$. In the numerical evaluation, we set the value $s = 1$.

6.1. 4-loop finite integrals

Let us consider the finite integrals $I_{s4}^4$, $I_{b4}^4$ corresponding to the diagrams of Fig [b] (a) and (b) (named $M_{44}$ and $M_{45}$ in Baikov and Chetyrkin [2]). The $W, U$ functions are given in Eqs [46]–[47].
Table 22: Results UV 3-loop diagram of Fig[34] with massive internal lines, using 36 threads on Intel(R) Xeon(R) E5-2687W v3 3.10GHz. DE is applied with mesh size $h = 0.1253191$ and number of evaluations $N_{eval} = 94$ (cf., Eq [15]). For extrapolation, $\varepsilon = \varepsilon_f = 1.15^{-l}$, $l = 10,11,\ldots$

| Integral Fig[34] | Extrapolation | $\ell$ | Result $C_1$ | Result $C_0$ | Result $C_1$ | Result $C_2$ |
|------------------|---------------|--------|--------------|--------------|--------------|--------------|
| 10               |               |        | 0.79879040550 | -0.6424371659 |              |              |
| 11               |               |        | 0.8778576859 | -1.3301603488 | 1.48816624   |              |
| 12               |               |        | 0.9017425166 | -1.756053047  | 3.46958790   | -0.0528730  |
| 13               |               |        | 0.91891701861| -1.9730679987 | 5.10026802   | -8.45486000 |
| 14               |               |        | 0.92233670919| -2.0663458474 | 6.10815188   | -13.847111  |
| 15               |               |        | 0.92331256399| -2.1096045295 | 6.61235817   | -17.726240  |
| 16               |               |        | 0.92356140680| -2.1120455643 | 6.82339433   | -19.916551  |
| 17               |               |        | 0.92361796674| -2.1151864782 | 6.89661152   | -20.933028  |
| 18               |               |        | 0.92362939667| -2.1196828755 | 6.92167297   | -21.326253  |
| 19               |               |        | 0.92363145072| -2.1161313491 | 6.92779243   | -21.455677  |
| 20               |               |        | 0.92363577660| -2.1161634494 | 6.92920424   | -21.492152  |
| 21               |               |        | 0.92363822245| -2.1161882958 | 6.92948625   | -21.501020  |
| 22               |               |        | 0.92363182806| -2.1161696940 | 6.92953519   | -21.502856  |
| 23               |               |        | 0.92363182875| -2.1161697224 | 6.92945359   | -21.503232  |

\begin{align*}
U &= x_7 x_8 x_9 x_{12} x_{13} + x_7 x_8 (x_{12} x_{34} + x_{34} x_6) + x_7 x_9 x_{126} x_{345} + x_8 x_9 x_{156} x_{234} \\
&+ x_7 x_3 x_{126} x_{34} + x_8 x_2 x_{156} x_{34} + x_9 (x_{162} x_{345} + x_{126} x_{34} x_5) + x_2 x_3 x_{156} x_3

\text{W/s} &= x_7 x_8 x_9 x_{12} x_{13} x_{456} + x_7 x_8 (x_{12} x_{456} + x_{24} x_{3}) x_9 (x_{12} x_{24} x_{356} + x_{124} x_{356}) + x_7 x_9 (x_{12} x_{456} + x_{123} x_{456}) \\
&+ x_8 x_9 (x_{12} x_{13} x_{456} + x_{123} x_{3}) x_6 x_7 x_5 (x_{12} x_{3} x_{46} + x_{123} x_{456}) + x_8 x_2 (x_{13} x_{456} + x_{13} x_{13} x_{456}) \\
&+ x_9 (x_{13} x_{2} x_{13} x_{456} + x_{123} x_{3} x_{456} + x_{123} x_{456}) + x_7 x_8 (x_{13} x_{3} x_{46} + x_{13} x_{13} x_{456})
\end{align*}

Since the corresponding integrals are finite, Eq [45] is evaluated with $\varphi = \varepsilon = 0$. These diagrams have $N = 9$ internal lines, leading to an 8-dimensional integral in the numerical evaluation. Table 22 lists the results for the integrals, obtained with PARINT executed on thor in double precision, using the cubature rule of degree 9 in 8 dimensions (see Section 3.3), which evaluates the function at 1105 points per subregion.

The analytic values given in [2] are:

\begin{align*}
I^S_a &= \frac{441 \zeta_5}{8} = 55.5852539156784, \\
I^S_b &= \frac{36 \zeta_5}{8} = 52.017868743610.
\end{align*}

$I^S_a$ and $I^S_b$ were evaluated numerically by Smirnov and Tentyukov using FIESTA [13]. The finite terms are given as $55.58537 \pm 0.00031$ and $52.0181 \pm 0.0003$, respectively, with 1.5 M samples.

6.2. 4-loop UV-divergent integrals

This section handles the integrals associated with the massless diagrams of Fig 8(c) and (d) (the 4-loop sunrise-sunset and Shimadzu diagrams named $M_{01}$ and $M_{56}$, respectively, in [2]), which have a UV singularity from the $\Gamma$-function factor in Eq [45]. We put $\varphi = 0$ in the integrals for Fig 8(c) and (d), and consider the expansions in $\varepsilon$.

Figure 8: 4-loop self-energy diagrams with massless internal lines, cf., Baikov and Chetyrkin [2]: (a) $N = 9$, (b) $N = 9$, (c) $N = 5$, (d) $N = 8$.
Table 23: PARINT accuracy and times (on 4 nodes/64 provs. thor cluster) for the loop integrals of the diagrams of Fig [8(a) and (b) (M44 and M45 in Baikov and Chetyrkin [2]) with ϑ = η = 0, using various numbers of function evaluations.

| Diagram | # FCN. EVALS | INTEGRAL RESULT | REL. ERR. EST. Eₚ | TIME T[s] |
|---------|--------------|-----------------|-------------------|-----------|
| Fig [8](a) | 100B | 55.594725 | 9.87e-04 | 185.1 |
| | 200B | 55.586822 | 3.53e-04 | 370.0 |
| | 300B | 55.585150 | 1.80e-04 | 554.3 |
| Eq (45) | | 55.585254 | | |
| Fig [8](b) | 100B | 52.026428 | 9.84e-04 | 239.5 |
| | 200B | 52.019118 | 3.63e-04 | 479.9 |
| | 275B | 52.017714 | 2.20e-04 | 658.8 |
| Eq (49) | | 52.017869 | | |

Table 24: Results UV 4-loop sunrise-sunset integral, Fig [8](c) (on 4 nodes/64 provs. thor cluster), err. tol. tₑ = 10⁻¹², T[s] = Time (elapsed user time in seconds); ϑ = η = 2⁻¹⁷, ℓ = 8, 9, . . . , Eₛ = integration estim. abs. error.

| INTEGRAL Iₚ¹⁴ | EXTRAPOLATION |
|---------------|---------------|
| ℓ | Eₛ | T[s] | RESULT C₁, | RESULT C₂ | RESULT C₃ |
| 8 | 3.9e-14 | 30.2 | -0.0031736179254977 | | |
| 9 | 3.9e-14 | 34.3 | -0.0031736115581839 | | |
| 10 | 3.6e-14 | 34.1 | -0.0031736115581839 | -0.016918557081 | -0.012276556 |
| 11 | 4.1e-14 | 50.8 | -0.0031736111099910 | -0.016927126297 | -0.011837812 |
| 12 | 4.2e-14 | 58.5 | -0.003173611111130 | -0.016927083216 | -0.011842959 |
| 13 | 5.6e-14 | 48.1 | -0.003173611111109 | -0.016927083381 | -0.011842916 |
| Eq (51) | -0.00317361111111 | -0.016927083333 | -0.011842930 |

For the integral Iₚ¹⁴ in Fig [8] the U, W functions are

(50)

The numerical results are compared with the expansion in Baikov and Chetyrkin [2],

(51)

Note that I = Iₚ¹⁴ of Eq (45) is multiplied with n(ε)₁⁴, where n(ε) is defined in Eq (35).

Based on integrations with PARINT, using a maximum of 10B function evaluations and an absolute error tolerance of 10⁻¹² (on the computation of the integral Iₚ¹⁴/T(−3 + 4ε)), the results in Table 24 are produced using linear extrapolation. PARINT returns a 0 error flag for the integrals constituting the input sequence to the extrapolation, indicating that a successful termination is assumed according to Eqs (2) or (14) for the requested accuracy.

For Iₚ¹⁴ of the N = 8 diagram shown in Fig [8](d) (Shimadzu, named M₅₆ in [2][3]), the U, W functions are

(52)
Table 25: Results UV 4-loop Shimadzu integral, \(I_{d}^{4}\), Fig 8(d), (on KEKSC 64 threads); \(\epsilon = \epsilon_{l} = 2^{-\ell}, \ell = 10, 11, \ldots\).

| \(\ell\) | RESULT \(C_{1}\) | EXTRAPOLATION | \(\ell\) | RESULT \(C_{0}\) | RESULT \(C_{1}\) |
|---|---|---|---|---|---|
| 10 | 5.18460577 | -2.47956688 | 5.18463921 | -2.58230627 | 70.1367604 |
| 11 | 5.18463921 | -2.58230627 | 70.1367604 |
| 12 | 5.18463922 | -2.58243393 | 70.3982056 |
| 13 | 5.18463923 | -2.58243413 | 70.3991764 |
| \(\text{Eq (53)}\) | 5.18463878 | -2.58243609 | 70.3991515 |

The expansion given in [2] is

\[
n(\epsilon)^{4}I_{d}^{4} = \frac{5\zeta_{5}}{\epsilon} - 5\zeta_{3} - 7\zeta_{3} + \frac{25}{2}\zeta_{6} + (35\zeta_{5} + 7\zeta_{3} - \frac{25}{2}\zeta_{6} - 21\zeta_{5}\epsilon \epsilon + \frac{127}{2}\zeta_{7}) \epsilon + \ldots \tag{53}
\]

This is \(I = I_{d}^{4}\) of Eq (45) multiplied with \(n(\epsilon)^{4}\), where \(n(\epsilon)\) is defined in Eq (35). The numerical result by FIESTA is shown in [13] and it is 5.184645 ± 0.000042.

The results by the DE formula (15) with \(N_{\text{eval}} = 49\) and mesh size \(h = 0.125\) in all dimensions, and linear extrapolation, are shown in Table 25. The starting \(\epsilon\) is \(2^{-10}\). The time required for each iteration using 64 threads on KEKSC System A, SR16000 Model M1 (POWER7(R) processor) is below 20 minutes.

7. Conclusions

In this paper we describe a fully numerical method for Feynman loop integrals, based on numerical multi-dimensional integration and linear or non-linear extrapolation. We use three categories of numerical integration methods, iterated integration with DQAGE (DQAGS) from QUADPACK, multivariate adaptive integration with PARINT, and the DE formula. For the numerical extrapolation, we employ nonlinear extrapolation with geometric sequences of the extrapolation parameter, and linear extrapolation with Bulirsch or geometric sequences. The main advantage of the method is its general applicability to multi-loop integrals with arbitrary physical masses and external momenta, without resorting to special problem formulations. Using dimensional regularization, both UV-divergent and finite terms are estimated. We have shown that the technique works well for sets of diagrams with up to four loops and up to four external lines, with or without UV-divergence, and the numerical results reveal excellent agreement with expansions in the literature [1, 2, 13].

We also demonstrate the effectiveness of variable transformations for some integrals. Regardless of whether or not variables are transformed, the formulation of the numerical method is not affected and it is only necessary to replace the integrand and Jacobian. The way to transform variables for a loop integral is not unique, and a general technique to find the most effective transformation is not currently known. However, the effectiveness of transformations can be assessed by examining the behavior of the integrations and the ensuing convergence of the extrapolation. The experience gained with the transformations in this paper will yield guidelines to construct a more general procedure, which will be studied in future work.

The computation time of the numerical multivariate integration increases with the number of internal lines, i.e., with the dimension of integration. For example, though we understand the importance of scattering processes with more external legs than five or six in two-loop order, this is beyond the scope of the current paper, and we plan on addressing these and related types of problems in future work.

Furthermore, while the examples shown here are limited to scalar loop integrals, they can easily be extended to more general cases with physical masses and external momenta, by including the associated numerator in the integrand (see [4]). The present work includes integrals with massive and massless internal particles. For massive particles, a mass value of one is assigned in order to compare results with the literature [1]. Massless cases are compared, e.g., with results in [2]. In view of the numerical nature of the methods there is in principle no limitation for the general
mass values, even though challenges may arise with respect to computing time vs. computational precision. We are helped in dealing with this trade-off by the evolution in computer architecture and the computational techniques.

Acknowledgments

We acknowledge the support from the National Science Foundation under Award Number 1126438, and the Center for High Performance Computing and Big Data at Western Michigan University. This work is further supported by Grant-in-Aid for Scientific Research (15H03668) of JSPS, and the Large Scale Simulation Program Nos. 15/16-06 and 16/17-21 of KEK.

Appendix A. Analytic method for Fig 2(c)

This Appendix derives the analytic result for the 2-loop half-boiled egg diagram of Fig 2(c).

By the variable transformation

$$x_1 = \tau (1 - \xi), \ x_2 = \tau \xi, \ x_3 = (1 - \tau) \tau' (1 - \xi'), \ x_4 = (1 - \tau) \tau' \xi', \ x_5 = (1 - \tau)(1 - \tau'),$$

the functions $U, V$ are given by

$$U = \tau F, \quad F = 1 - \tau + \tau \xi (1 - \xi)$$

$$W/s = \tau G, \quad G = (1 - \tau)(1 - \tau')((1 - \tau) \tau' + \tau \xi(1 - \xi))$$

and

$$J_{c2}^{S2} = \int_0^1 d\tau (1 - \tau)^2 \int_0^1 d\xi' \frac{1}{\tau^{1-3\epsilon} (\tau \xi')^{1+2\epsilon}} \int_0^1 d\tau' \frac{1}{\tau'^{1-3\epsilon} (\tau' \xi')^{1+2\epsilon}}$$

where we assume $m_3 = m_4$ to perform the $\xi'$-integration.

The integral is divergent at $\tau = 0$ to produce a $1/\epsilon$ singularity, and the separation of the singularity is done as follows. Let us denote $H = M^2 F - s G$, and let us use the suffix 0 for the function defined at $\tau = 0$, i.e., $F_0 = F(\tau = 0) = 1$ and $H_0 = H(\tau = 0) = [\tau m_3^2 + (1 - \tau') m_4^2] - s (1 - \tau') \tau'$. The integral is separated into two terms as

$$J_{c2}^{S2} = I_A + I_B$$

where the first term $I_A$ has a UV singularity. The integrands of $I_A$ and $I_B$ are given according to Eq (A.1) and

$$\frac{1}{\tau^{1-3\epsilon} F^{1+2\epsilon}} = \frac{1}{\tau^{2-\epsilon} F^{1-3\epsilon} H^{1+2\epsilon}} \left[ \frac{1}{\tau^{2-\epsilon} F^{1-3\epsilon} H^{1+2\epsilon}} + \frac{1}{F^{1-3\epsilon} H^{1+2\epsilon} - F_0^{1-3\epsilon} H_0^{1+2\epsilon}} \right] .$$

The divergent term $I_A$ is trivial in the $\xi$-integral and is calculated as

$$I_A = \int_0^1 d\tau \frac{(1 - \tau)^2}{\tau^{1-\epsilon}} \int_0^1 d\tau' \frac{1}{\tau'^{1-\epsilon}} = \left( \frac{1}{\epsilon} - \frac{3}{2} + \frac{7}{4} \epsilon - \frac{15}{8} \epsilon^2 \right) \left( I_A^{(0)} + I_A^{(1)} \epsilon + I_A^{(2)} \epsilon^2 + I_A^{(3)} \epsilon^3 \right)$$

(A.2)

and the non-divergent term $I_B$ is

$$I_B = \int_0^1 d\tau \int_0^1 d\xi \frac{(1 - \tau)^2}{\tau^{1-\epsilon}} \int_0^1 d\tau' \frac{1}{\tau'^{1-\epsilon}} \left( F^{1-3\epsilon} H^{1+2\epsilon} - F_0^{1-3\epsilon} H_0^{1+2\epsilon} \right)$$

$$= I_B^{(1)} + I_B^{(2)} \epsilon + I_B^{(3)} \epsilon^2$$

(A.3)

The terms $I_A^{(0)}$, $I_A^{(1)}$, $I_A^{(2)}$, $I_A^{(3)}$ and $I_B^{(1)}$, $I_B^{(2)}$, $I_B^{(3)}$ in Eqs (A.2) and (A.3) can be obtained by expanding the integrands in powers of $\epsilon$. We evaluated these terms numerically by DQAGE and DQAGS from QUADPACK and the DE formula, for the values $s = M^2 = 1$, yielding the expression in Eq (20).
References

[1] Laporta S 2000 Int. J. Mod. Phys. A 15 5087–5159 arXiv:hep-ph/0102033v1
[2] Baikov B A and Chetyrkin K G 2010 Nuclear Physics B 837 186–220
[3] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2004 Computer Physics Communications 159 145–156
[4] de Doncker E, Shimizu Y, Fujimoto J, Yuasa F, Cucos L and Van Voorst J 2004 Numerical Instruments and Methods in Physics Research A 539 269–273 hep-ph/0405098
[5] Yuasa F, de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T and Shimizu Y 2007 Precise numerical results of IR-vertex and box integration with extrapolation XI Adv. Comp. and Anal. Tech. in Phys. Res. PoS (ACAT07) 087, arXiv:0709.0777v2 [hep-ph]
[6] Yuasa F, Ishikawa T, Fujimoto J, Hamaguchi N, de Doncker E and Shimizu Y 2008 Numerical evaluation of Feynman integrals by a direct computation method XII Adv. Comp. and Anal. Tech. in Phys. Res. PoS (ACAT08) 222; arXiv:0904.2823
[7] de Doncker E, Fujimoto J, Kurihara Y, Hamaguchi N, Ishikawa T, Shimizu Y and Yuasa F 2010 Recursive box and vertex integrations for the one-loop hexagon reduction in the physical region XIII Adv. Comp. and Anal. Tech. in Phys. Res. PoS (ACAT10) 073
[8] Yuasa F, de Doncker E, Hamaguchi N, Ishikawa T, Kato K, Kurihara Y and Shimizu Y 2012 Journal of Computer Physics Communications 183 2136–2144
[9] de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T, Kurihara Y, Shimizu Y and Yuasa F 2011 Journal of Computational Science (JCoS) 3 102–112 doi:10.1016/j.jocs.2011.06.003
[10] de Doncker E, Li S, Fujimoto J, Shimizu Y and Yuasa F 2005 Springer Lecture Notes in Computer Science (LNCS) 3514 165–171
[11] de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T, Kurihara Y, Ljucovic M, Shimizu Y and Yuasa F 2010 Extrapolation algorithms for infrared divergent integrals arXiv:hep-ph/110.3587; PoS (CPP2010)011
[12] de Doncker E, Yuasa F and Kurihara Y 2012 Journal of Physics: Conf. Ser. 368
[13] Smirnov A V and Tentyukov M 2010 Nuclear Physics B 837 40–49
[14] de Boor C 1971 CADRE: An algorithm for numerical quadrature Mathematical Software ed Rice J R (Academic Press, New York) pp 417–449
[15] Piessens R, de Doncker E, Überhuber C W and Kahaner D K 1983 QUADPACK, A Subroutine Package for Automatic Integration (Springer Series in Computational Mathematics vol 1) (Springer-Verlag)
[16] de Doncker E and Yuasa F 2013 Journal of Physics: Conf. Series 410 012047 doi:10.1088/1742-6596/410/1/012047
[17] de Doncker E, Yuasa F and Assal R 2013 Journal of Physics: Conf. Ser. 454 doi:10.1088/1742-6596/454/1/012082
[18] de Doncker E and Yuasa F 2014 XV Adv. Comp. and Anal. Tech. in Phys. Res., Journal of Physics: Conference Series (ACAT 2013) 523 doi:10.1088/1742-6596/523/1/012052
[19] Shanks D 1955 J. Math. and Phys. 34 1–42
[20] Wynn P 1956 Mathematical Tables and Aids to Computing 10 91–96
[21] Sidi A 1996 Journal of Approximation Theory 86 21–40
[22] Sidi A 2003 Practical Extrapolation Methods - Theory and Applications (Cambridge University Press) ISBN 0-521-66159-5
[23] Sidi A 2011 Numer. Math. 119 725–764
[24] OpenMP http://www.openmp.org
[25] Takahasi H and Mori M 1974 Publications of the Research Institute for Mathematical Sciences 9 721–741
[26] Davis P J and Rabinowitz P 1984 Methods of Numerical Integration (Academic Press, New York)
[27] Sugihara M 1997 Methods of Numerical Integration (Academic Press, New York) pp 280–283
[28] Davis P J and Rabinowitz P 1984 Numer. Math.
[29] Brezinski C 1980 Numerische Mathematik
[30] Bulirsch R 1964 Numerische Mathematik
[31] Davis P J and Rabinowitz P 1984 Numer. Math.
[32] Brezinski C 1980 Numerische Mathematik
[33] Davis P J and Rabinowitz P 1984 Numer. Math.
[34] Open-MPI http://www.open-mpi.org
[35] de Doncker E, Yuasa F, Kapenga J and Olagbemi O 2015 The Journal of Physics: Conf. Series (JPCS) 640 doi:10.1088/1742-6596/640/1/012062
[36] Genz A and Malik A 1980 Journal of Computational and Applied Mathematics 6 295–302
[37] Genz A and Malik A 1983 SIAM J. Numer. Anal. 20 580–588
[38] Berntsen J, Espelid T O and Genz A 1991 ACM Trans. Math. Softw. 17 437–451
[39] Genz A 1990 An adaptive numerical integration algorithm for simplices Lecture Notes in Computer Science vol 507 ed Sherwani N A, de Doncker E and Kapenga J A pp 279–285
[40] Grundmann A and Möller H 1978 SIAM J. Numer. Anal. 15 282–290
[41] de Doncker E 1979 Math. Comp. 33 1003–1018
[42] Berntsen J, Espelid T O and Genz A 1991 ACM Trans. Math. Softw. 17 452–456 http://www.sci.wsu.edu/math/faculty/genz/homepage
[43] de Doncker E, Gupta A and Ealy P 1996 Lecture Notes in Computer Science, Springer-Verlag 1067 562–570
[44] de Doncker E, Kaugars K, Cucos L and Zanny R 2001 Current status of the ParInt package for parallel multivariante integration Proc. of Computational Particle Physics Symposium (CPP 2001) pp 110–119
[45] Achalla C, Kaugars K and de Doncker E 2003 Load balancing in distributed adaptive task partitioning Parallel and Distributed Computing and Systems pp 280–283
[46] Achalla C, de Doncker E, Kaugars K and Van Voorst J 2004 α-load balancing for parallel adaptive task partitioning Parallel and Distributed Computing and Systems
[47] ParInt http://www.cs.wmich.edu/parint

28
[48] Motoki S, Daisaka H, Nakasato N, Ishikawa T, Yuasa F, Fukushige T, Kawai A and Makino J 2015 *The Journal of Physics: Conf. Series* (JPCS) 608 doi:10.1088/1742-6596/608/1/012011

[49] de Doncker E, Yuasa F and Olagbemi F 2015 *Procedia Computer Science* 51 1333–1342 doi:10.1016/j.procs.2015.05.318

[50] de Doncker E, Yuasa F, Kato K, Ishikawa T and Olagbemi O 2015 *The Journal of Physics: Conf. Series* (JPCS) 608 doi:10.1088/1742-6596/608/1/012071

[51] de Doncker E, Yuasa F, Kato K and Ishikawa T 2016 Automatic numerical integration and extrapolation for Feynman loop integrals *Workshop on Computational Particle Physics (CPP2016)* Under review