Adaptive Integration for 3-loop Feynman Diagrams with Massless Propagators

Elise de Doncker¹, Fukuko Yuasa², and Fola Olagbemi³

¹ Western Michigan University, Kalamazoo MI 49008, U.S.A.
   elise.dedoncker@wmich.edu
   omofolakunmiel.olagbemi@wmich.edu
² High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki, 305-0801, Japan
   fukuko.yuasa@kek.jp

Abstract
We apply multivariate adaptive integration to problems arising from self-energy Feynman loop diagrams with massless internal lines. Results are obtained with the PARINT software package, which is layered over MPI (Message Passing Interface) and incorporates advanced parallel computation techniques such as load balancing among processes that may be distributed over a network of nodes. To solve the problems numerically we introduce a parameter $\varrho$ in a factor of the integrand function. Some problem categories allow setting $\varrho = 0$; other cases require an extrapolation as $\varrho \to 0$. Furthermore we apply extrapolation with respect to the dimensional regularization parameter by setting the dimension $n = 4 - 2\varepsilon$ and extrapolating as $\varepsilon \to 0$. Timing results with ParInt show near optimal parallel speedups for the problems at hand.

Keywords: Automatic numerical integration, massless Feynman diagrams, parallel computation

1 Introduction

In recent years we set out to demonstrate the capabilities of automatic, adaptive integration software for the numerical evaluation of Feynman integrals with multiple loops. This has included iterated or repeated quadrature in successive coordinate directions with 1D programs from the QUADPACK package [PdDUK83], which proved successful for various types of singular behavior such as infrared (IR) and ultraviolet (UV) singularities [dDYK12, dDYK+14], as well as multivariate adaptive integration (cubature) with the PARINT package for massive 3-loop diagrams. The method is based on assigning a sequence of finite values to an infinitesimal parameter representing a regulator in the loop integral, and extrapolating numerically as the parameter tends to zero. This paper shows an extension of the domain of applicability to a set of 3-loop integrals with massless internal lines discussed by Baikov and Chetyrkin [BC10], in particular, the scalar 3-loop finite self-energy integrals $L_0, N_0, N_1$, and $N_2$.

We use the adaptive code from the PARINT package [ZdDKC], which is automatic in the sense that the user specifies the required accuracy via absolute and relative error tolerances $t_a$ and $t_r$, respectively,

doi:10.1016/j.procs.2015.05.318
and the algorithm attempts to achieve the accuracy within the given maximum number of function evaluations, or terminates with an error flag. PARINT is written in C and layered over MPI (Message Passing Interface [OMI]) for execution by parallel distributed processes, and can be invoked from the command line or called from a program.

Section 2 gives a general expression for Feynman loop integrals and introduces the problem at hand. The automatic integration (cubature) package PARINT is described in Section 3, with emphasis on the parallel adaptive techniques including load balancing. Section 4 gives some background on nonlinear extrapolation with the $\epsilon$-algorithm and its implementation. The numerical results presented in Section 5 are obtained either by setting the parameter $\rho = 0$, or by a nonlinear extrapolation as $\rho \to 0$, or by an extrapolation with respect to the dimensional regularization parameter $\epsilon$, introduced by setting the dimension $n = 4 - 2\epsilon$.

2 Background

An $L$-loop integral with $N$ internal lines can be represented in Feynman parameter space by

$$I = \frac{\Gamma(N - \frac{nL}{2})}{(4\pi)^{nL/2}} (-1)^N \int_0^1 \prod_{r=1}^{N} dx_r \delta(1 - \sum x_r) \frac{C^{N-n(L+1)/2}}{(D - i\rho C)^{N-nL/2}}, \quad (1)$$

where $C$ and $D$ are polynomials determined by the topology of the corresponding diagram and physical parameters. The integration in Eq (1) is taken over the $N$-dimensional unit cube. However, as a result of the $\delta$-function one of the $x_r$ can be expressed in terms of the other ones, which reduces the integral dimension to $N - 1$ and the domain to the $d = (N - 1)$-dimensional unit simplex of the form

$$S_d = \{ (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid \sum_{j=1}^{d} x_j \leq 1 \text{ and } x_j \geq 0 \}. \quad (2)$$

In the absence of IR and UV singularities we have $n = 4$. For dimensional regularization in case of IR singularities we set $n = 4 + 2\epsilon$, and for UV singularities $n = 4 - 2\epsilon$. We applied the regularization by a numerical extrapolation as $\epsilon \to 0$ in [dDYK12] and [dDYK+14], respectively.

The term $i\rho C$ prevents the denominator from vanishing in the integration domain and can be used for regularization. A regularization to keep the integral from diverging was achieved by extrapolation as $\rho \to 0$ in [dDSFY04, dDSF+04, YdDF+07, YIF+08, dDFK+10, YdDH+12, dDFH+11].

As in [YdDH+12] (where $L = 2$) we set $C = 1$ in the denominator of Eq (1) for a regularization as $\rho \to 0$. Multiplying the numerator and denominator by $(D + i\rho)^{N-nL/2} = (D + i\rho)^{N-6}$ (using $L = 3, n = 4$) and omitting the factor $\Gamma \left( N - \frac{nL}{2} \right) / (4\pi)^{nL/2}$ then leads to the integral

$$I_N = (-1)^N \prod_{j=1}^{N-1} \int_0^1 dx_j \frac{C^{N-8}(D + i\rho)^N}{(D^2 + \rho^2)^{N-6}}$$

$$= (-1)^N \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \ldots \int_0^{1-x_1-\ldots-x_{N-2}} dx_{N-1} \frac{C^{N-8}(D + i\rho)^{N-6}}{(D^2 + \rho^2)^{N-6}} \quad (3)$$

over the $(N - 1)$-dimensional simplex $S_{N-1}$ of Eq (2). For the diagrams of Figure 1(a) and (b) the number of internal lines is $N = 7$ and Eq (3) becomes

$$I_7 = \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \ldots \int_0^{1-x_1-\ldots-x_5} dx_6 \frac{(D + i\rho)}{C(D^2 + \rho^2)} \quad (4)$$
The integral for the diagrams of Figure 1(c) and (d) is of the form
\[ I_8 = \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \ldots \int_0^{1-x_1-\ldots-x_6} dx_7 \frac{(D + i\varrho)^2}{(D^2 + \varrho^2)^2}. \] (5)

For the numerical integration we apply cubature rules over the \( d = (N - 1) \)-dimensional unit cube
\[ C_d = \{ (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid 0 \leq x_j \leq 1 \} \] (6)
using the transformation (from the \( d \)-dimensional simplex to the cube),
\[ (x_1, \ldots, x_d) \to (x(1), (1-x_1)x(2), \ldots, (1-x_1-\ldots-x_{d-1})x(d)) \] (7)
with Jacobian \((1-x_1)(1-x_1-x_2)\ldots(1-x_1-\ldots-x_{d-1})\). The variables \( x(i), 1 \leq i \leq d \), on the right of Eq (7) vary between 0 and 1 for integration over the cube \( C_{N-1} \) given in Eq (6).

3 ParInt Adaptive Cubature

Written in C and layered over MPI [OM], 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 (\( t_r \) and \( t_a \), respectively). The integrand is generally defined as a vector function over the domain \( \mathcal{D} \). Denoting the (exact) integral by
\[ \mathcal{I}\tilde{f} = \int_{\mathcal{D}} \tilde{f}(\vec{x}) \, d\vec{x}, \]
it is then the objective to return an approximation \( Q\tilde{f} \) and absolute error estimate \( E_a\tilde{f} \) such that
\[ || Q\tilde{f} - \mathcal{I}\tilde{f} || \leq || E_a\tilde{f} || \leq \max\{ t_a, t_r || \mathcal{I}\tilde{f} || \} \] (8)

or indicate with an error flag that the requested accuracy cannot be reached. In order to satisfy the error criterion of Eq (8) the program tests throughout whether \( || E_a\tilde{f} || \leq \max\{ t_a, t_r || Q\tilde{f} || \} \) is achieved.
(in infinity norm). Since in this paper the integrand is a scalar-valued function \( f : D \to \mathbb{R} \), we denote
the absolute error estimate (in absolute value) by \( E_a \) and the corresponding relative error estimate by
\( E_r = E_a/|Q| \), assuming \( Q \neq 0 \). 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.

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 \([GM80, GM83, BEG91a]\), the 1D
(Gauss-Kronrod) rules used in QUADPACK and a set of rules for the \( d \)-dimensional simplex \([Gen90,
GM78, dD79]\). The results in Section 5 below are computed over the \( d \)-dimensional unit cube, with a
rule of polynomial degree 9. 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 rule of degree 9 gives better results for the
problems tested than the degree-7 rule that is also available in PARINT for integration over the cube. 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.

Optionally the PARINT installation can be configured to use long doubles instead of doubles.

### 3.1 PARINT Adaptive Methods

![Figure 2: Adaptive integration meta-algorithm](image)

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 \([PdDÜK83]\) and of DCUHRE \([BEG91b]\) by successive bisections. The workers then each gen-
erate a local priority queue of subregions as a task pool. 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 struc-
ture on the worker, the task pool is stored as a deep or double-ended heap, which allows deleting 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 chil-
dren regions), integration over the children, deletion of the parent region (root of the heap) and insertion
of the children into the heap (see Figure 3.1). The bisection of a region is performed perpendicularly
to the coordinate direction in which the integrand is found to vary the most, according to fourth-order
differences computed in each direction \([BEG91b]\).

### 3.2 Load Balancing

For a regular integrand behavior and \( p \) MPI processes distributed evenly over homogeneous processors,
the computational load decreases ideally by a factor of about \( p \). Otherwise the parallel time (and space)
requirements may be improved 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 mecha-
nism of the distributed integration algorithm \([dDGE96, dDKCZ01, AKdD03, AdDKVV04]\). The mes-
sage passing is performed in a non-blocking and asynchronous manner, and permits overlapping of computations and communications, which benefits PARINT’s efficiency on a hybrid platform (multicore and distributed, possibly different types of processors) where multiple processes are assigned to each node. We note that some of the tests in Section 5 were also run with load balancing turned off, which gave inferior results.

4 Numerical Extrapolation

Even though the integrals in Section 2 are finite, we perform extrapolation or convergence acceleration to obtain a desired accuracy. We apply nonlinear extrapolation with the $\epsilon$-algorithm [Sha55, Wyn56, Sid96, Sid11]. A sequence of integral values $I(\epsilon\ell)$ is generated using a geometric progression, $\epsilon\ell = 1/2^\ell$, which tends to zero rapidly with increasing $\ell$.

The extrapolation results given here are achieved with a version of the $\epsilon$-algorithm code from QUADPACK [PdDUK83]. In between calls, the implementation retains the last two lower diagonals of the triangular extrapolation table. When a new element $I(\epsilon\ell)$ 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:

$$e_0 \quad e_1 \quad new \quad e_2 \quad e_3$$

we have that

$$new = e_1 + 1/(1/(e_1 - e_3) + 1/(e_2 - e_1) - 1/(e_1 - e_0)),$$

and the distance measure for the new element is set to $|e_2 - e_1| + |e_1 - e_0| + |e_2 - new|$. 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.

5 Numerical Results and Parallel Performance

Results are given below for the diagrams in Fig 1(a) and (b), represented by the (real part) integral of Eq (4), and for the diagrams of Fig 1(c) and (d), adhering to Eq (5):

$$\Re I_7 = \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \ldots \int_0^{1-x_1-\ldots-x_5} dx_6 \frac{D}{C(D^2 + \varrho^2)}$$

and

$$\Re I_8 = \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \ldots \int_0^{1-x_1-\ldots-x_6} dx_7 \frac{D^2 - \varrho^2}{(D^2 + \varrho^2)^2}.$$  

The integrals are further transformed to the unit cube according to Eq (7). For the integration over each subregion the rule of degree 9 is applied, which evaluates the 6D integrand at 453 points and the 7D integrand at 717 points. The exact integral for the four diagrams is $20\zeta_5 \approx 20.738555102867$ (see [BC10]). For the numerical approximations and timings we use the thor cluster of the High Performance Computing and Big Data Center at WMU. The tests are run on 16-core cluster nodes with Intel Xeon E5-2670, 2.6GHz processors and 128GB of memory each, and using the Infiniband interconnect for message passing via MPI.

5.1 Setting $\varrho = 0$

Results can be obtained with PARINT for the $N_1, N_2$ and $L_0$ integrals directly by setting $\varrho = 0$ in Eqs (9) and (10). Table 1 lists the results with 48 processes: integral approximation, relative error
Table 1: Accuracy and times with 48 procs. for $N_2$, $N_1$ and $L_0$ loop integrals with $\varrho = 0$, using various numbers of function evaluations. Also given is the relative error estimate $E_r$. 

For $\varrho = 0$ the integrand of Eq (9) has a zero denominator when $C = 0$ or $D = 0$, and Eq (10) has a zero denominator when $D = 0$, e.g., when $x_3 = 0$ and $x_1 + x_2 + x_4 + x_5 + x_6 = 1$, or when $x_2 = 1$, $x_1 = x_3 = x_4 = x_5 = x_6 = 0$. Furthermore, $C = 0$, for example at $x_2 = 1$, $x_1 = x_3 = x_4 = x_5 = x_6 = 0$, and at $x_3 = x_5 = x_6 = 0$, $x_1 + x_4 + x_5 = 1$ (on the boundaries of the unit simplex). For a discussion of vanishing $C$ and $D$ on the boundaries of the integration domain see also [BW10]. The $N_0$ integral has $D = 0$ at $x_2 = 1$, $x_1 = x_3 = x_4 = x_5 = x_6 = x_7 = 0$. Thus our integrand codes test for zero denominators before performing the divisions. However some of the computations overflow by integrand evaluations in the vicinity of the singularities, which is found to occur for the $N_0$ integral in double precision around $1B = 10^9$ evaluations and higher.

### 5.2 Parallel Performance

Fig 3 shows times and speedups as a function of the number of processes $p$, for a computation of the $N_1$, $N_2$ and $L_0$ integrals using $10B$ integrand evaluations in double precision, with $\varrho = 0$ in Eqs (9) and (10). PARINT is invoked by a call to the $pi\_integrate()$ function, and the times are returned as part of the output printed by the PARINT $pi\_print\_results()$ function. Denoting the time in seconds for $p$ processes by $T_p(s)$, the corresponding speedup given by $S_p = T_1(s)/T_p(s)$ is nearly optimal, which would coincide with the diagonal in the graph, or slightly superlinear ($>p$) over the given range of $p$. 

---

<table>
<thead>
<tr>
<th>Diagram</th>
<th># Fcn. Eval.</th>
<th>Integral Result</th>
<th>Rel. Err. Est. $E_r$</th>
<th>Time $T(s)$</th>
<th>Integral Result</th>
<th>Rel. Err. Est. $E_r$</th>
<th>Time $T(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact:</td>
<td>20.7385510</td>
<td>20.7385510</td>
<td></td>
<td></td>
<td>20.7385510</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_2$</td>
<td>5B</td>
<td>20.7393800</td>
<td>3.69e-05</td>
<td>9.7</td>
<td>20.7393292</td>
<td>3.63e-05</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>10B</td>
<td>20.7387220</td>
<td>6.64e-06</td>
<td>19.4</td>
<td>20.73872078</td>
<td>6.61e-06</td>
<td>35.1</td>
</tr>
<tr>
<td></td>
<td>25B</td>
<td>20.73857098</td>
<td>8.32e-07</td>
<td>48.6</td>
<td>20.73857018</td>
<td>8.12e-07</td>
<td>87.9</td>
</tr>
<tr>
<td></td>
<td>50B</td>
<td>20.73855716</td>
<td>1.96e-07</td>
<td>98.2</td>
<td>20.73855718</td>
<td>1.95e-07</td>
<td>175.9</td>
</tr>
<tr>
<td></td>
<td>75B</td>
<td>20.73855576</td>
<td>9.28e-08</td>
<td>146.4</td>
<td>20.73855575</td>
<td>9.12e-08</td>
<td>264.4</td>
</tr>
<tr>
<td></td>
<td>100B</td>
<td>20.73855540</td>
<td>5.68e-08</td>
<td>196.7</td>
<td>20.73855540</td>
<td>5.43e-08</td>
<td>352.6</td>
</tr>
<tr>
<td>$N_1$</td>
<td>5B</td>
<td>20.73871652</td>
<td>2.21e-05</td>
<td>9.0</td>
<td>20.73871522</td>
<td>2.14e-05</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>10B</td>
<td>20.73856839</td>
<td>3.50e-06</td>
<td>17.9</td>
<td>20.73856878</td>
<td>3.42e-06</td>
<td>32.1</td>
</tr>
<tr>
<td></td>
<td>25B</td>
<td>20.73855535</td>
<td>3.79e-07</td>
<td>44.9</td>
<td>20.73855539</td>
<td>3.71e-07</td>
<td>80.5</td>
</tr>
<tr>
<td></td>
<td>50B</td>
<td>20.73855508</td>
<td>9.07e-08</td>
<td>90.3</td>
<td>20.73855508</td>
<td>8.94e-08</td>
<td>161.1</td>
</tr>
<tr>
<td></td>
<td>75B</td>
<td>20.73855507</td>
<td>4.26e-08</td>
<td>135.6</td>
<td>20.73855507</td>
<td>4.23e-08</td>
<td>242.1</td>
</tr>
<tr>
<td></td>
<td>100B</td>
<td>20.73855508</td>
<td>2.59e-08</td>
<td>180.8</td>
<td>20.73855508</td>
<td>2.56e-08</td>
<td>323.2</td>
</tr>
<tr>
<td>$L_0$</td>
<td>5B</td>
<td>20.74194961</td>
<td>1.19e-03</td>
<td>10.3</td>
<td>20.74196270</td>
<td>1.19e-03</td>
<td>19.6</td>
</tr>
<tr>
<td></td>
<td>10B</td>
<td>20.73866434</td>
<td>3.64e-04</td>
<td>20.7</td>
<td>20.73860437</td>
<td>3.51e-04</td>
<td>39.2</td>
</tr>
<tr>
<td></td>
<td>25B</td>
<td>20.73827908</td>
<td>5.04e-05</td>
<td>51.7</td>
<td>20.73827067</td>
<td>5.04e-05</td>
<td>98.2</td>
</tr>
<tr>
<td></td>
<td>50B</td>
<td>20.73841802</td>
<td>1.09e-05</td>
<td>103.4</td>
<td>20.73842955</td>
<td>9.79e-06</td>
<td>196.7</td>
</tr>
<tr>
<td></td>
<td>75B</td>
<td>20.73848662</td>
<td>3.66e-06</td>
<td>146.4</td>
<td>20.73848624</td>
<td>3.70e-06</td>
<td>295.0</td>
</tr>
<tr>
<td></td>
<td>100B</td>
<td>20.73851402</td>
<td>1.96e-06</td>
<td>207.5</td>
<td>20.73851338</td>
<td>1.98e-06</td>
<td>393.9</td>
</tr>
</tbody>
</table>
Figure 3: Parallel performance for computation of the $N_1$, $N_2$ and $L_0$ integrals of Eqs (9) and (10) using 10B evaluations: (a) Computation times (in seconds), (b) Speedup

5.3 Extrapolation Results as $\varrho \to 0$

The $N_0$ integral cannot be obtained for large numbers of integrand evaluations by setting $\varrho = 0$, as the computation overflows by evaluating the function in the vicinity of the singularity. Extrapolation alleviates the need to set $\varrho = 0$. Table 2 shows an extrapolation as $\varrho \to 0$ using the $\epsilon$-algorithm of Wynn [Sha55, Wyn56, Sid96, Sid11] (see Section 4). The $\varrho_\ell$ 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 $\epsilon$-algorithm function from QUADPACK.
### Table 2: Integration with PARINT using 64 procs., max. # evals = 150B, \( \rho_\ell = 2^{-\ell} \), and extrapolation with \( \epsilon \)-algorithm for \( N_0 \) integral

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>( \text{INTEGRAL} )</th>
<th>( \nu )</th>
<th>( E_\nu )</th>
<th>( T(s) )</th>
<th>( \text{EXTRAPOLATION} )</th>
<th>( \text{LAST} )</th>
<th>( \text{Selected} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>19.69036128576084</td>
<td>1.44e-07</td>
<td>474.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>19.91633676759658</td>
<td>1.64e-07</td>
<td>474.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>20.09256513888053</td>
<td>1.84e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>20.23033840929211</td>
<td>1.94e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>20.33827222472266</td>
<td>2.16e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>20.42297783943613</td>
<td>2.36e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>20.48955227010659</td>
<td>2.53e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>20.54194208640818</td>
<td>2.74e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>20.58321345028519</td>
<td>2.95e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>20.61575568045697</td>
<td>3.14e-07</td>
<td>474.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>20.64143527978811</td>
<td>3.34e-07</td>
<td>474.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>20.66171281606668</td>
<td>3.51e-07</td>
<td>474.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>20.6774379215522</td>
<td>4.16e-07</td>
<td>474.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Exact:** 20.73855510 20.73855510

### Table 3: Integration with PARINT using 64 procs., max. # evals = 100B, \( \epsilon_\ell = 2^{-\ell} \), and extrapolation with \( \epsilon \)-algorithm for \( N_2 \) integral

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>( \text{INTEGRAL} )</th>
<th>( \nu )</th>
<th>( E_\nu )</th>
<th>( T(s) )</th>
<th>( \text{EXTRAPOLATION} )</th>
<th>( \text{LAST} )</th>
<th>( \text{Selected} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>21.21987706233486</td>
<td>8.84e-08</td>
<td>648.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>20.97727482739239</td>
<td>8.69e-08</td>
<td>648.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>20.85743468356065</td>
<td>8.61e-08</td>
<td>649.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>20.79787566374119</td>
<td>8.56e-08</td>
<td>647.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>20.76818590083646</td>
<td>8.53e-08</td>
<td>648.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>20.75363432626232</td>
<td>8.39e-08</td>
<td>648.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>20.74595780282081</td>
<td>8.29e-08</td>
<td>647.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20.74225639032920</td>
<td>8.21e-08</td>
<td>647.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Exact:** 20.73855510 20.73855510

The *Last* column lists the final (utmost right) element computed in the lower diagonal. Overall the \( \epsilon \)-algorithm function from QUADPACK appears successful at selecting a competitive element as its result.

### 5.4 Extrapolation with respect to the Dimensional Regularization Parameter

The dimensional regularization parameter \( \epsilon \) is introduced in Eq (1) by assigning \( n = 4 - 2\epsilon \). Baikov and Chetyrkin [BC10] produce asymptotic expansions in integer powers of \( \epsilon \) for 3- and 4-loop integrals arising from diagrams with massless propagators. We illustrate extrapolation with respect to \( \epsilon \) in Table 3 for the integral of the \( N_2 \)-diagram, using 100B evaluations for the integrations in long double precision. The integrals are computed with \( \rho = 0 \); thus the integrand retains a singular behavior at \( C = 0 \) or \( D = 0 \). The extrapolation converges faster than that with respect to \( \rho \) in Table 2. The times are larger compared to those of Table 2, likely by calling the C-language `pow` function for the exponentiations in each integrand evaluation, whereas the integrand of Eq (10) for the \( \rho \) extrapolation can be calculated using only multiplications, divisions, additions and subtractions.
6 Concluding Remarks

This paper reports on the calculation of Feynman integrals of 3-loop self-energy diagrams with massless propagators. We introduce a parameter $\varrho$ in the factor $D$ of the integrand, as well as a dimensional regularization parameter $\varepsilon$ in the exponents of the functions $C$ and $D$. The integrals are finite and the computation succeeds for some problems by setting $\varrho = 0$ (and $\varepsilon = 0$). In case this procedure does not yield the desired accuracy, we perform a nonlinear extrapolation with the $\epsilon$-algorithm on a sequence of integral approximations. Test results are given for extrapolations with respect to the $\varrho$ parameter, as well as the dimensional regularization parameter $\varepsilon$. All computations are numerical and the integrals are evaluated with the parallel/distributed multivariate cubature package PARINT. Parallel speedup results indicate near optimal speedups.

Apart from the reductions in [BC10], other approaches include the Mellin-Barnes representation and sector decomposition. However, our numerical computations (with or without extrapolation) have been achieved without resorting to significant manipulation of the integrals, and thus hold promise for future applications to other complicated Feynman loop integrals.

Acknowledgments

We thank Dr. J. Fujimoto and Dr. T. Ueda for helpful comments. We further acknowledge the support from the National Science Foundation under Award Number 1126438, and from Grant-in-Aid for Scientific Research (24540292) of JSPS, and the Large Scale Simulation Program No.13/14-13 of KEK.

References


