Journal of Physics: Conference Series **640** (2015) 012062 doi:10.1088/1742-6596/640/1/012062

Scalable Software for Multivariate Integration on Hybrid Platforms

E de Doncker[1], **F Yuasa**[2], **J Kapenga**[1] **and O Olagbemi**

[1] Department of Computer Science, Western Michigan University, Kalamazoo MI 49008, U S A
[2] High Energy Accelerator Research Organization (KEK), Oho 1-1, Tsukuba, Ibaraki, 305-0801, Japan

E-mail:
elise.dedoncker@wmich.edu, fukuko.yuasa@kek.jp, john.kapenga@wmich.edu,

Abstract. The paper describes the software infrastructure of the PARINT package for multivariate numerical integration, layered over a hybrid parallel environment with distributed memory computations (on MPI). The parallel problem distribution is typically performed on the region level in the adaptive partitioning procedure. Our objective has been to provide the end-user with state of the art problem solving power packaged as portable software. We will give test results of the multivariate PARINT engine, with significant speedups for a set of 3-loop Feynman integrals. An extrapolation with respect to the dimensional regularization parameter (ε) is applied to sequences of multivariate PARINT results $Q(\varepsilon)$ to obtain the leading asymptotic expansion coefficients as $\varepsilon \to 0$. This paper further introduces a novel method for a parallel computation of the $Q(\varepsilon)$ sequence as the components of the integral of a vector function.

1. Introduction

The PARINT parallel integration package supports the computation of multivariate integrals over hyper-rectangular and simplex regions. The algorithms and tools in the parallel integration engine include:
(i) an adaptive region subdivision algorithm, equipped with a load balancing strategy to handle localized integrand difficulties such as peaks and singularities;
(ii) a non-adaptive Quasi-Monte Carlo (QMC) method based on Korobov lattice rules;
(iii) Monte Carlo (MC) methods for high dimensions and/or erratic integrands;
(iv) a user interface based on the PARINT plugin compiler which pre-processes the user problem specification and integrand function to be linked with the ParInt executable;
(v) 1D rules corresponding to those in QUADPACK [1], which can be used for repeated integration in successive coordinate directions.
Repeated integration with QUADPACK programs has been proven extremely effective for applications to Feynman loop integrals with severe singularities in high-energy physics [2]. However, repeated or iterated integration has an impeding drawback with respect to the expense of the method for higher-dimensional problems (say, dimensions ≥ 6). In a sequential setting, multivariate adaptive integration is considered suitable for moderate dimensionality (say, up to 10) while possibly dealing with mildly irregular integrand behavior, and for higher dimensions with well-behaved functions. These limits can be moved up considerably in a parallel environment by allowing finer subdivisions with higher numbers of integrand evaluations.

Written in C and layered over MPI [3], the PARINT methods 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 defined as a vector function over the domain $\mathcal{D}$. Denoting the (exact) integral by

$$I\vec{f} = \int_{\mathcal{D}} \vec{f}(\vec{x})\, d\vec{x}, \tag{1}$$

it is then the objective to return an approximation $Q\vec{f}$ and absolute error estimate $E_a\vec{f}$ such that $|| Q\vec{f} - I\vec{f} || \leq || E_a\vec{f} || \leq \max\{ t_a, t_r|| I\vec{f} || \}$, or indicate with an error flag that the requested accuracy cannot be achieved. When a relative accuracy (only) needs to be satisfied we set $t_a = 0$ (and vice-versa). If both $t_a \neq 0$ and $t_r \neq 0$, the weaker of the two error tolerances is imposed; if both $t_a = t_r = 0$ then the program will reach an abnormal termination, typically when the maximum number of function evaluations is reached. Optionally the PARINT installation can be configured to use long doubles instead of doubles.

This paper focuses on the distributed adaptive algorithm in the PARINT multivariate integration package combined with extrapolation techniques as addressed in Section 2, and presents test results for Feynman loop integrals in Section 3.

2. Adaptive integration and extrapolation

2.1. PARINT *parallel adaptive integration*

Layered over MPI [3] for distributed computing, the PARINT adaptive code assigns the roles of controller and worker processes for adaptive region partitioning (see also [4] and [5] for an explanation of the algorithm). 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 in QUADPACK [1], DCUHRE [6] and HALF [7] by successive bisections, thus generating a local priority queue of subregions as a task pool. The priority queue is implemented as a max-heap keyed with the estimated integration error over the subregions, so that the subregion with the largest estimated error is stored in the root of the heap, or as a *deap* or *double-ended heap*, to allow for efficient deletions of the minimum as well as the maximum element in case the user specifies a maximum size to be retained for the heap structure.

An important mechanism of the distributed integration algorithm is the load balancing strategy, which is receiver-initiated and geared to keeping the loads on the worker task pools balanced, particularly for problems with irregular integrand behavior. The message passing is performed in a non-blocking and asynchronous manner, and permits overlapping of computation and communication. As a result of the asynchronous processing and message passing on MPI, PARINT executes on a hybrid platform (multi-core and distributed) by assigning multiple processes to each node. The parallelization may, however, lead to *breaking loss* or extra work performed at the end of the computation, due to the time elapsed while workers continue to generate subregions after the termination message has been issued but before receiving it. This may increase the total amount of work (compared to the sequential work), particularly when the number of processes is large.

2.2. 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 (1). A user guide is provided in [8]. 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 $\vec{x}$, number of component functions `nfuncs` and the resulting values of the function $\vec{f}(x)$. Further parameters of `pi_integrate()` are (on input): `nfuncs`, an integer identifying the cubature/quarature rule to use, the maximum number of function evaluations allowed, the region type (hyper-rectangle or simplex) and specification; and (on output): the integral and error approximations `result[]` and `error[]`, and a user-declared pointer to a status structure.

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 may be of the form

```
mpirun -np 64 --hostfile myhostfile ./parint -f fcn -lf 10000000 -ea 0.0 -er 5.0e-10
```

For example, with 4 nodes listed in `myhostfile` and `ppn = 16`, a total of 64 processes is requested. The integrand function of this run is named `fcn` in the user's library; the maximum number of function evaluations is 10000000, and the absolute and relative error tolerances are 0 and 5.0e-10, respectively.

2.3. Use of extrapolation

We apply numerical extrapolation to integrals with an asymptotic expansion in the dimensional regularization parameter ε, of the form

$$S(\varepsilon) \sim \sum_{k \geq K} C_k \varphi_k(\varepsilon) \qquad \text{as } \varepsilon \to 0. \tag{2}$$

For example, the $\varphi_k(\varepsilon)$ functions may be integer powers of ε, $\varphi_k(\varepsilon) = \varepsilon^k, K \geq k$, with $K = 0$ for finite integrals. If the $\varphi_k(\varepsilon)$ functions are known, we can apply a linear extrapolation by approximating $S(\varepsilon)$ for decreasing values of $\varepsilon = \varepsilon_\ell$, and truncating Eq (2) after $2, 3, \ldots, \nu$ terms to form linear systems of increasing size to be solved for the C_k variables.

When the $\varphi_k(\varepsilon)$ functions are unknown we perform a non-linear numerical extrapolation with the ϵ-algorithm [9, 10, 11, 12]. In that case we generate a sequence of integral approximations for $S(\varepsilon_\ell)$, using a geometric progression of ε_ℓ that tends to zero with increasing ℓ (see also [13, 2]), and employ a version of the ϵ-algorithm code (DQEXT) from QUADPACK. In between calls, the DQEXT implementation retains the last two lower diagonals of the triangular extrapolation table. When a new element of the input sequence is provided, 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: $\begin{matrix} & & e_0 & \\ e_3 & & e_1 & \textit{new} \\ & & e_2 & \end{matrix}$ we assign $\textit{new} = e_1 + 1/(1/(e_1 - e_3) + 1/(e_2 - e_1) - 1/(e_1 - e_0))$, and set the distance measure for the *new* element as $|e_2 - e_1| + |e_1 - e_0| + |e_2 - \textit{new}|$. The new lower diagonal element with the smallest value of the distance measure is returned as the result for this call to the extrapolation code.

3. Results for Feynman loop integrals

One prominent application of multivariate integration is to the computation of Feynman loop integrals, which contribute higher order corrections for accurate theoretical predictions of the cross-section for particle interactions. The integral corresponding to an n-dimensional scalar Feynman diagram with L loops and N internal lines can be represented in Feynman parameter space as

$$I = \frac{\Gamma\left(N - \frac{nL}{2}\right)}{(4\pi)^{nL/2}} (-1)^N \int_0^1 \prod_{j=1}^{N} dx_j\, \delta(1 - \sum x_i) \frac{C^{N-n(L+1)/2}}{(D - i\varrho C)^{N-nL/2}}. \tag{3}$$

Here the functions C and D are polynomials determined by the topology of the corresponding Feynman diagram [14]. After removing the δ-function in (3) and eliminating one of the

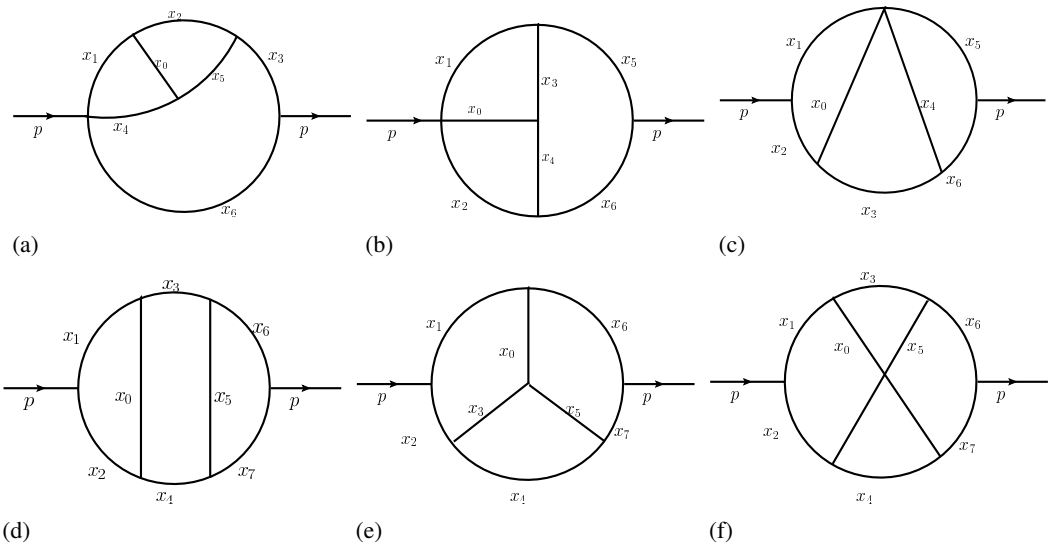

Figure 1. 3-loop diagrams (a) $N = 7$ (Laporta[15] Fig 2(q)), (b) $N = 7$ (Laporta[15] Fig 2(r)), (c) $N = 7$ (Laporta[15] Fig 2(s)), (d) $N = 8$ (Laporta[15] Fig 2(t)), (e) $N = 8$ (Laporta[15] Fig 2(u), (f) $N = 8$ (Laporta[15] Fig 2(v))

variables in view of $\sum_{j=1}^{N} x_j = 1$, the integration domain is a d-dimensional simplex $\mathcal{S}_d = \{ (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid \sum_{j=1}^{d} x_j \leq 1 \text{ and } x_j \geq 0 \}$, with $d = N - 1$. The term $i\varrho C$ prevents the integral in (3) from diverging if D vanishes in the interior of the domain. As this does not happen for the problems under consideration in this paper we can set $\varrho = 0$. We set $n = 4 - 2\varepsilon$ to compute the leading order coefficients in an asymptotic expansion of the integral with respect to the dimensional regularization parameter ε.

Test results are presented below for a set of 3-loop self-energy diagrams from Laporta [15] (given in Fig 1). In order to compare our integral approximations with Laporta's we set all masses $m_r = 1$ and $s = 1$, and furthermore divide the integral by $\Gamma^3(1 + \varepsilon)$. For the numerical approximations and timings we use the HPCC (High Performance Computation Center) cluster at WMU. The tests are run on 16-core cluster nodes with Intel Xeon E5-2670, 2.6GHz processors and 128GB of memory, and using the Infiniband interconnect for message passing via MPI. Tables 1-4 show various types of results obtained by calling the `pi_integrate()` function and running the executable with 16 processes on the HPCC cluster. The integrals are transformed from the (unit) simplex to the (unit) cube and the integration is taken over the cube, using the basic integration rule of polynomial degree 9 [6]. Tables 1

Table 1. Integral and leading order expansion coefficients using PARINT with 16 procs., and linear extrapolation for 3-loop integral of Laporta Fig 2(r), err. tol. $t_a = 10^{-12}$; max. # evals = 20B, $T(s)$ = elapsed time (s); E_a = integration estim. abs. error

	INTEGRAL FIG 2(r)			EXTRAPOLATION		
ε_ℓ	INTEGRAL	E_a	T(S)	RESULT C_0	RESULT C_1	RESULT C_2
2^{-3}	0.89462319318517	6.33e-10	356.9			
2^{-4}	1.07605987265074	8.56e-10	426.3	1.257496552116	-2.9029868714	
2^{-5}	1.19524813881849	1.15e-09	426.2	1.333416355943	-4.7250621633	9.7177349
2^{-6}	1.26445377191768	1.34e-09	426.2	1.341017173944	-5.1507079713	16.5280678
2^{-7}	1.30188593759114	1.46e-09	426.2	1.341390110905	-5.1954604067	18.1988254
2^{-8}	1.32137252564963	1.52e-09	426.2	1.341399132800	-5.1976978366	18.3778198
2^{-9}	1.33131707386056	1.55e-09	426.2	1.341399240859	-5.1977522983	18.3868241
2^{-10}	1.33634079003748	1.57e-09	426.2	1.341399241503	-5.1977529523	18.3870438
2^{-11}	1.33886565298449	1.58e-09	426.2	1.341399241505	-5.1977529584	18.3870480
			Eq (2)*, Laporta [15]:*	1.341399241447	-5.1977529559	18.3870466

Table 2. Integral and leading order expansion coefficients using PARINT with 16 procs., and extrapolation with ϵ-algorithm for 3-loop integral of Laporta Fig 2(r), err. tol. $t_a = 10^{-12}$; max. # evals = 20B, $T(s)$ = elapsed time (s); E_a = integration estim. abs. error

	INTEGRAL FIG 2(r)			EXTRAPOLATION		
ε_ℓ	INTEGRAL	E_a	T(S)	RESULT C_0	RESULT C_1	RESULT C_2
2^{-3}	0.89462319318517	6.33e-10	356.9			
2^{-4}	1.07605987265074	8.56e-10	426.3			
2^{-5}	1.19524813881849	1.15e-09	426.2	1.423460265674	-2.0676022044	-7.9521322
2^{-6}	1.26445377191768	1.34e-09	426.2	1.360275447540	-3.2163693548	-23.710490
2^{-7}	1.30188593759114	1.46e-09	426.2	1.339480501116	-6.2931480198	5.5420653
2^{-8}	1.32137252564963	1.52e-09	426.2	1.341163816983	-5.4086365404	9.8856275
2^{-9}	1.33131707386056	1.55e-09	426.2	1.341410985041	-5.1746466051	25.0634683
2^{-10}	1.33634079003748	1.57e-09	426.2	1.341399965444	-5.1950222482	19.5001958
2^{-11}	1.33886565298449	1.58e-09	426.2	1.341399223875	-5.5197895049	18.6327973
2^{-12}	1.34013135392416	1.58e-09	426.2	1.341399240952	-5.1977615908	18.3726055
2^{-13}	1.34076502405465	1.59e-09	426.2	1.341399241506	-5.1977527356	18.3878520
		Eq (2)*, Laporta [15]:*		1.341399241447	-5.1977529559	18.3870466

Table 3. Integral and leading order expansion coefficients using PARINT with 16 procs., and linear extrapolation for 3-loop integral of Laporta Fig 2(u), err. tol. $t_a = 10^{-12}$; max. # evals = 20B, $T(s)$ = elapsed time (s); E_a = integration estim. abs. error

	INTEGRAL FIG 2(u)			EXTRAPOLATION		
ε_ℓ	INTEGRAL	E_a	T(S)	RESULT C_0	RESULT C_1	RESULT C_2
2^{-3}	0.176698722960541	1.01e-09	363.8			
2^{-4}	0.179083545661235	9.17e-10	437.2	0.181468368362	-0.0381571632	
2^{-5}	0.180693790881676	9.24e-10	437.3	0.182582592016	-0.0648985309	0.14262063
2^{-6}	0.181617679292608	9.34E-10	437.2	0.182626195317	-0.0673403158	0.18168919
2^{-7}	0.182111420125171	9.79e-10	437.7	0.182627225039	-0.0674638824	0.18630234
2^{-8}	0.18236652627441	9.84e-10	437.6	0.182627237225	-0.0674669046	0.18654412
2^{-9}	0.182496175680214	9.86e-10	437.7	0.182627237219	-0.0674669014	0.18654358
2^{-10}	0.182561529243301	9.86e-10	437.8	0.182627237221	-0.0674669038	0.18654439
		Eq (2)*, Laporta [15]:*		0.182627237539	-0.0674669097	0.18654624

Table 4. Integral and leading order expansion coefficients using one call to PARINT with vector function, 16 procs. and linear extrapolation for 3-loop integral of Laporta Fig 2(u), err. tol. $t_a = 10^{-12}$; max. # evals = 20B, $T(s)$ = elapsed time (s); E_a = integration estim. abs. error

	INTEGRAL FIG 2(u)		EXTRAPOLATION		
ε_ℓ	INTEGRAL	E_a	RESULT C_0	RESULT C_1	RESULT C_2
2^{-3}	0.176698722966533	1.57e-09			
2^{-4}	0.179083545593469	1.63e-09	0.181468368220	-0.0381571620	
2^{-5}	0.1806937908054	1.71e-09	0.182582591950	-0.0648985315	0.1426206455
2^{-6}	0.181617679222644	1.76e-09	0.182626195261	-0.0673403170	0.1816892047
2^{-7}	0.182111420130547	1.80e-09	0.182627225208	-0.0674639106	0.1863033669
2^{-8}	0.182366526276538	1.81e-09	0.182627237153	-0.0674668729	0.1865403500
2^{-9}	0.182496175679009	1.82e-09	0.182627237223	-0.0674669081	0.1865461701
2^{-10}	0.182561529242407	1.83e-09	0.182627237223	-0.0674669083	0.1865462415
		Eq (2)*, Laporta [15]:*	0.182627237539	-0.0674669097	0.1865462421

Table 5. Parallel performance of PARINT (on MPI) for 3-loop integrals (C_0) of Fig 1, abs. tolerance $t_a = 5 \times 10^{-10}$, and max. number of evaluations = 10B

3-loop diag.	N	Result Laporta [15]	Result $p = 1$	Result $p = 64$	$T_1[s]$	$T_{64}[s]$	S_{64}
Fig 1 (a)	7	1.32644820827	1.326448206	1.32644819	902.7	15.8	57.1
Fig 1 (b)	7	1.34139924145	1.34139924147	1.3413992416	1026.2	14.4	71.3
Fig 1 (c)	7	2.00250004111	2.00250004113	2.0025000412	879.3	13.4	65.6
Fig 1 (d)	8	0.27960892328	0.2796089227	0.279608920	1019.7	15.9	64.1
Fig 1 (e)	8	0.18262723754	0.1826272372	0.1826272368	1018.3	15.8	64.4
Fig 1 (f)	8	0.14801330396	0.1480133036	0.1480133026	976.6	16.4	59.5

and 3 are computed with consecutive calls to `pi_integrate()` in a loop and linear extrapolation, for the functions of (Laporta) Fig 2(r) with $N = 7$, and Fig 2(u) with $N = 8$, respectively (depicted in Fig 1). The values of C_0, C_1 and C_2 are listed ($K = 0$ in Eq (2)). The abbreviation "B" refers to *billion* with regard to the number of integrand evaluations, and $T(s)$ represents the integration time in seconds. Table 2 is obtained similarly for Fig 2(r), but with non-linear extrapolation by the ϵ-algorithm.

The results in Table 4 for Fig 2(u) are computed with one call to `pi_integrate()` from the main program, integrating a vector function that contains the ε-dependence in the definition of its components. The integration time for the vector function (at 2180 seconds) is considerably less than the total integration time through the loop in Table 3, with comparable or slightly better accuracy for the same problem. If less accuracy is needed overall, the computation will take less time with both methods. The extrapolations in Tables 1-4 show good agreement with the values of the Eq (2) coefficients from Laporta [15]. Table 5 presents timing results on the HPCC cluster, corresponding to the six diagrams in Fig 1. Since the integrals with kinematics specified by Laporta [15] are finite and only the integral needs to be computed, we set $\varepsilon = 0$ in the integrand codes. The speedup $S_p = T_1/T_p$ (where T_p is the integration time incurred with p processes) indicates good scalability of the parallel implementation (see also [16]). For a total of $p = 64$ MPI processes on 4 cluster nodes, 16 procs are spawned per node.

4. Concluding remarks

In this paper we explored *distributed, automatic* integration with the PARINT multivariate integration package, in particular its adaptive integration engine. We demonstrated its accuracy and parallel scalability for the computation of the leading order coefficients in an asymptotic expansion with respect to the dimensional regularization parameter ε, as $\varepsilon \to 0$, for a class of 3-loop self-energy integrals from [15]. This procedure relies on obtaining approximations of a sequence of integrals as $\varepsilon \to 0$, and an extrapolation for convergence acceleration of the sequence. In view of the compute-intensive nature for moderate integral dimensions, the parallelization has been proven beneficial for obtaining considerable accuracy.

Acknowledgments

We acknowledge the support from the National Science Foundation under Award Number 1126438. This work is further supported by Grant-in-Aid for Scientific Research (24540292) of JSPS, and the Large Scale Simulation Program No.13/14-13 of KEK.

References

[1] 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)
[2] de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T, Kurihara Y, Shimizu Y and Yuasa F 2011 *Journal of Computational Science (JoCS)* **3** 102–112 doi:10.1016/j.jocs.2011.06.003
[3] Open-MPI http://www.open-mpi.org
[4] de Doncker E, Kaugars K, Cucos L and Zanny R 2001 *Proc. of Comp. Particle Physics Symp. (CPP 2001)* pp 110–119
[5] 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
[6] Berntsen J, Espelid T O and Genz A 1991 *ACM Trans. Math. Softw.* **17** 437–451
[7] De Ridder L and Van Dooren P 1976 *Journal of Computational and Applied Mathematics* **2** 207–210
[8] ParInt http://www.cs.wmich.edu/parint
[9] Shanks D 1955 *J. Math. and Phys.* **34** 1–42
[10] Wynn P 1956 *Mathematical Tables and Aids to Computing* **10** 91–96
[11] Sidi A 1996 *Journal of Approximation Theory* **86** 21–40
[12] Sidi A 2011 *Numer. Math.* **119** 725–764
[13] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2004 *Computer Physics Communications* **159** 145–156
[14] Nakanishi N 1971 *Graph Theory and Feynman Integrals* (Gordon and Breach, New York)
[15] Laporta S 2000 *Int. J. Mod. Phys. A* **15** 5087–5159 arXiv:hep-ph/0102033v1
[16] de Doncker E, Yuasa F, Kato K, Ishikawa T and Olagbemi O *XVI Adv. Comp. and Anal. Tech. in Phys. Res., Journal of Physics: Conference Series (ACAT 2014)* Accepted