Multi-threaded adaptive extrapolation procedure for Feynman loop integrals in the physical region

E de Doncker[1], **F Yuasa**[2] **and R Assaf**[1]

[1] Department of Computer Science, Western Michigan University, Kalamazoo MI 49008, U. S.
[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, rida.assaf@wmich.edu

Abstract. Feynman loop integrals appear in higher order corrections of interaction cross section calculations in perturbative quantum field theory. The integrals are computationally intensive especially in view of singularities which may occur within the integration domain. For the treatment of threshold and infrared singularities we developed techniques using iterated (repeated) adaptive integration and extrapolation. In this paper we describe a shared memory parallelization and its application to one- and two-loop problems, by multi-threading in the outer integrations of the iterated integral. The implementation is layered over OpenMP and retains the adaptive procedure of the sequential method exactly. We give performance results for loop integrals associated with various types of diagrams including one-loop box, pentagon, two-loop self-energy and two-loop vertex diagrams.

1. Introduction

Feynman loop integrals are generally affected by non-integrable singularities, through vanishing denominators in the interior and/or at the boundaries of the integration domain. In order to handle singularities inside the domain, the value for the integral is calculated by introducing a parameter $i\delta$ in the integrand denominator, effectively moving the singularity into the complex plane, and by taking the limit of the integral as $\delta \to 0$. We make use of numerical iterated integration and a procedure for convergence acceleration or extrapolation of a sequence of integral values as δ decreases.

A loop integral for a diagram with L loops and N propagators is given in Feynman parameter space as

$$\mathcal{I} = \frac{\Gamma\left(N - \frac{nL}{2}\right)}{(4\pi)^{nL/2}} (-1)^N \int_{\mathcal{D}} \prod_{j=1}^{N} dx_j \, \delta(1 - \sum x_j) \frac{C^{N-n(L+1)/2}}{(D - i\,\delta\, C)^{N-nL/2}}, \qquad (1)$$

where C and D are polynomials in $x_1, x_2, \ldots, x_N$, determined by the topology of the corresponding diagram and the physical parameters. Here $\mathcal{D}$ represents the N-dimensional unit hypercube integration domain, and the delta function $\delta(1 - \sum x_j)$ reduces the integration to the $(N-1)$-dimensional unit simplex.

Loop integrals are generally divergent when denominators vanish in the integration region. We assume $\mathcal{I}$ does not suffer from other divergences, such as infrared (IR) divergence, and exists in the limit as the parameter $\delta \to 0$. To calculate the limit numerically [1], we generate a sequence of $\mathcal{I} = \mathcal{I}(\delta)$ for (geometrically) decreasing values of δ, and apply convergence acceleration or extrapolation to the limit with the ε-algorithm [2, 3].

We have implemented the basic method in DCM (*Direct Computational Method*) for loop integrals. The technique can achieve high accuracies compared to, e.g., Monte Carlo integration, and it has been shown to successfully handle some singular problems which are problematic for other adaptive multivariate integration programs. The obtained accuracy helps controlling the accumulation of numerical error in large-scale calculations, where the precision of the end-results is fundamental for comparisons between theoretical and experimental results.

In previous work [4] we introduced a technique utilizing multi-threading for a parallel computation of the integrals. In this paper we add a parallelization on possibly multiple levels of the iterated integral, and focus further on testing the application to loop integrals. The implementation is layered over OpenMP [5] and retains the adaptive procedure of the sequential method exactly. Thus, each parallel execution performs the same subdivisions and function evaluations as the corresponding sequential run and can be regenerated; which is an important issue in the verification and debugging of parallel programs, and for avoiding useless parallel work. The parallelization is with respect to the function evaluations in the iterated method, where each function evaluation is itself an integral (except at the lowest level). This is important to guarantee a large enough granularity for the parallel procedure.

Concepts underlying automatic adaptive integration are reviewed in section 2 below. Their incorporation within an extrapolation strategy as implemented in DCM is outlined in section 3. section 4 describes our parallelization of iterated integration for a multi-threaded environment. Applications to the computation of Feynman loop integrals are given in section 5, with timing results using the OpenMP Application Program Interface (API).

2. Automatic adaptive integration

An *automatic* integration procedure can be considered as a *black-box* approach to produce (as outputs) an approximation $Q(f)$ to an integral

$$If = \int_{\mathcal{D}} f(\vec{x})\; d\vec{x} \tag{2}$$

and an error estimate $\mathcal{E}f$ of the actual error $Ef = |Qf - If|$, in order to satisfy an accuracy requirement of the form

$$|\,Qf - If\,| \;\leq\; \mathcal{E}f \;\leq\; \max\{\,t_a\,,\,t_r\,|If|\,\}, \tag{3}$$

where the integrand function f, region $\mathcal{D}$ and (absolute/relative) error tolerances t_a and t_r, respectively, are specified as part of the input.

A versatile type of algorithm to implement the black-box approach performs an adaptive partitioning of the integration region as shown in figure 1. At each step, a region is subdivided, integral and error estimates are computed over the subregions, and the overall result and error estimate are updated. Various strategies are possible for the selection of the region to be subdivided at each step. A *global adaptive* strategy maintains a priority queue on the subregion collection (e.g., a linked list or a heap), keyed with the local error estimates of the subregions, and selects the region with the highest (absolute) error estimate for subdivision at each step. As a result of the region selection in adaptive procedures, sample points tend to be concentrated in the vicinity of irregular behavior such as singularities, discontinuities, peaks or ridges and troughs of the integrand function.

Examples of adaptive integrators include the 1D adaptive programs of the Quadpack [6] package, the program Dcuhre [7] for multivariate integration over a cube, and programs in Cubpack [8] for adaptive integration over cubes and simplices.

```
Evaluate initial region and update results
Initialize priority queue with initial region
while (evaluation limit not reached and
       estimated error too large)
    Retrieve region from priority queue
    Split region into subregions
    Evaluate new subregions and update results
    Insert new subregions into priority queue
```

Figure 1. Adaptive Integration Meta-Algorithm

The local integral (over a subregion/interval) is approximated by a (cubature/quadrature) rule which is a linear combination of function values, of the form $\sum_{k=1}^{K} w_k f(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$. By evaluating more than one rule over the subregion, a local error estimate can be obtained as a function of the difference between local integral approximations. For example, the (1D) program DQAGE of the QUADPACK package uses a pair of approximations, given by an r-point Gauss rule and the interlacing $(2r+1)$-point Kronrod rule. The QUADPACK user has the option of choosing one of the pairs with $r = 7, 10, 15, 20, 25$ or 30. The Kronrod rule supplies the local integral approximation, and the Gauss rule (together with the Kronrod rule) serves to obtain the local error estimate. The r-point Gauss rule is of polynomial degree of accuracy $2r - 1$ (i.e., it integrates all polynomials of degrees $0, \cdots, 2r-1$ exactly, and there are polynomials of degree $2r$ which are not integrated exactly). The Kronrod rule is of polynomial degree $3r + 1$ if r is even, and of degree $3r + 2$ when r is odd (because of symmetry). The multivariate program DCUHRE applies an embedded sequence of cubature rules [9] for the local integral and error approximations over a subregion.

3. Integration and extrapolation strategy

The infinitesimal parameter $i\delta$ in the denominator of (1) prevents the integral from diverging and in that sense plays the role of a regulator. We consider the integral as a function of δ and construct a sequence of approximations to $\mathcal{I} = \mathcal{I}(\delta)$ for a decreasing sequence of δ, in order to perform an extrapolation to the limit as $\delta \to 0$. Methods for an extrapolation of the sequence rely on the existence of an asymptotic expansion $I(\delta) \sim I + a_1\varphi_1(\delta) + a_2\varphi_2(\delta) + \cdots$, as $\delta \to 0$. Linear or nonlinear extrapolation methods can be explored under certain conditions on the functions $\varphi_\ell(\delta)$ (see, e.g., [1]). DCM implements a nonlinear extrapolation or convergence acceleration with the ε-algorithm of Wynn [2, 3], which can be applied under more general conditions than linear extrapolation.

A schematic view of the program flow of DCM is given in figure 2 [10]. Note that $\delta = 0$ is valid in cases with unphysical kinematics, that is, where no divergence appears and no extrapolation is required, so the multi-dimensional integration only can be carried out.

For the numerical integration, we use iterated integration with the QUADPACK programs DQAGE or DQAGSE [6]. The extrapolation is applied to a sequence of $\mathcal{I}(\delta_j)$ computed for a geometric progression of δ_j, such as $c\,(1.2^{-j}), j \geq 0$ where c is a constant. We have used DCM for automatic integration without explicit information about the location or nature of the singularity, for various cases of one- and two-loop diagrams including 3-point (*vertex*), 4-point (*box*) [1, 11, 12, 13], 5-point (*pentagon*) and, after reductions, 6-point (*hexagon*) diagrams [14]; as well as two-loop *self-energy* [15], *double box*, *ladder* and *crossed vertex* [16, 17, 18, 19, 20, 21] diagrams with masses. The techniques were further incorporated for semi-automatic calculations

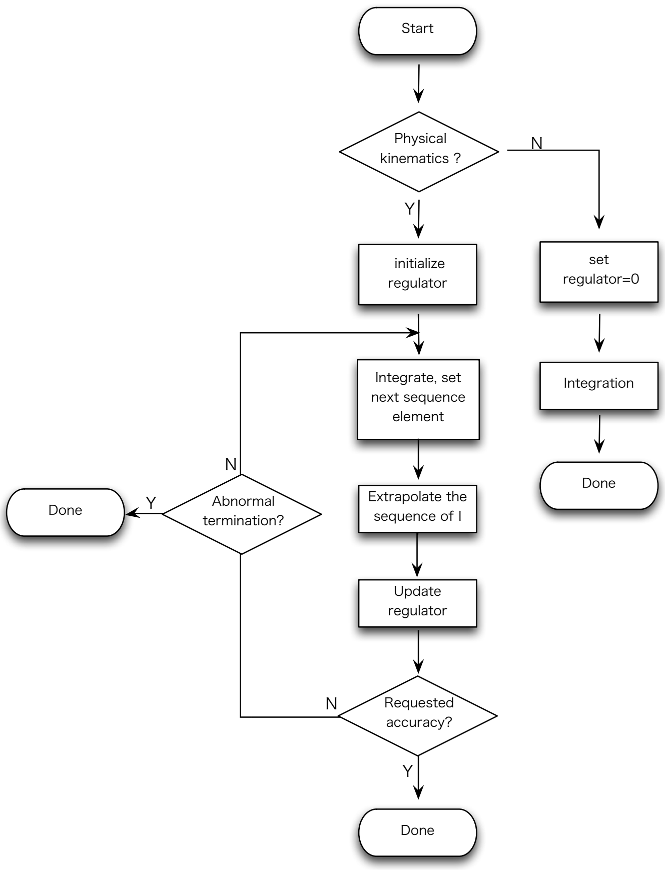

Figure 2. Program flow of DCM

of loop integrals with infrared divergences, in [18, 22].

4. Parallel numerical iterated integration

4.1. Iterated integration

In this paper we are dealing with iterated 1D integration over a finite d-dimensional product region, i.e., the integral If in (2) can be written as

$$If = \int_{\alpha_1}^{\beta_1} dx_1 \int_{\alpha_2}^{\beta_2} dx_2 \ldots \int_{\alpha_d}^{\beta_d} dx_d \; f(x_1, x_2, \ldots, x_d), \tag{4}$$

and the limits of integration may in general be functions, $\alpha_j = \alpha_j(x_1, x_2, \ldots, x_{j-1})$ and $\beta_j = \beta_j(x_1, x_2, \ldots, x_{j-1})$. The integration over the interval $[\alpha_j, \beta_j]$ will be performed with a 1D adaptive integration code, and different 1D integration programs may be applied in different directions $1 \le j \le d$.

If an interval $[a, b]$ arises in the subdivision of $[\alpha_j, \beta_j]$ for $1 \le j < d$, then the local integral approximation over $[a, b]$ is of the form

$$\int_a^b dx_j \; F(c_1, \ldots, c_{j-1}, x_j) \;\approx\; \sum_{k=1}^{K} w_k F(c_1, \ldots, c_{j-1}, x^{(k)}), \tag{5}$$

where the w_k and $x^{(k)}, 1 \leq k < K$, 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

$$F(c_1, \ldots, c_{j-1}, x^{(k)}) = \int_{\alpha_{j+1}}^{\beta_{j+1}} dx_{j+1} \ldots \int_{\alpha_d}^{\beta_d} dx_d \; f(c_1, \ldots, c_{j-1}, x^{(k)}, x_{j+1}, \ldots, x_d), \qquad 1 \leq k \leq K, \tag{6}$$

is itself an integral in the $x_{j+1}, \ldots, x_d$-directions, and is computed by the method(s) for the inner integrations. For $j = d$, (6) 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 the error incurred in the inner integration is subject to an error control condition of the form (3) and will contribute to the overall integration error. Work on the integration error interface is reported in [23, 24, 25].

Subsequently, in section 5, we give results obtained with 1D repeated integration by the program Dqage from Quadpack, where the local integration is performed with the (7, 15)- or the (10, 21)-points Gauss-Kronrod pairs. We will refer to the latter as GK15 and GK21, respectively.

The inner integrals given by (6), which appear in the local rule sum of (5), are independent and can thus be evaluated in parallel, by multiple threads. In previous work [4, 26] we reported results of this parallelization applied to the outer direction of integration using OpenMP. Important properties of this method include that: 1) the granularity of the parallel integration is large, especially when the inner integrals given by (6) are of dimension greater than two; 2) apart from possibly the order of the summation in the local rule evaluation, the parallel calculation is the same as the sequential evaluation.

Nesting of parallel constructs is also feasible, for a parallel evaluation of the rule in multiple coordinate directions of the iterated integral. The current implementation allows for a nested parallelization in the outer and the next to outer level (corresponding to the x_1 and x_2 directions in the iterated integral (4)); this can be extended to more or different levels. This feature is beneficial when the inner integral evaluations at the x_1-level are very time-consuming, so the work can be further distributed to new threads, on systems where a large number of threads are available.

5. Results for Feynman loop integrals

We implemented the methods of section 4 as a multi-threaded application layered over OpenMP [5], and report timing results on: a multi-core Intel workstation with Xeon X5680 @3.33GHz, 6-core (12 logical)-core CPU (*"minamivt005"*); the Hitachi SR16000/M1 with POWER7 @3.83GHz, 32-core processors at KEK; and the Intel cluster (*"thor"*) at WMU with Xeon E5-2670 processors, 16 cores per node.

The compilers used were: on *minamivt005*, the Intel *ifort* Fortran XE compiler with flag *-openmp*; the f90 compiler with flag *-omp* on SR16000; and the gfortran compiler with flag *-fopenmp* on *thor*. Furthermore we are working on a C version of the package, which is being tested with gcc *-fopenmp*.

In all cases the main loop in the rule routines Dqk15 and Dqk21 of Quadpack was parallelized (for the outer two integrations), by means of an OpenMP *omp parallel do* for the Gauss and Kronrod rule evaluations. On comparing loop schedules, we found that the *dynamic* loop schedule outperforms the default *static* schedule in selected test problems, on *minamivt005* and on *thor*. In the *static* schedule, the participating threads are assigned their loop iterations at the beginning of the loop, in a round-robin fashion; thus the assigned iterations are determined

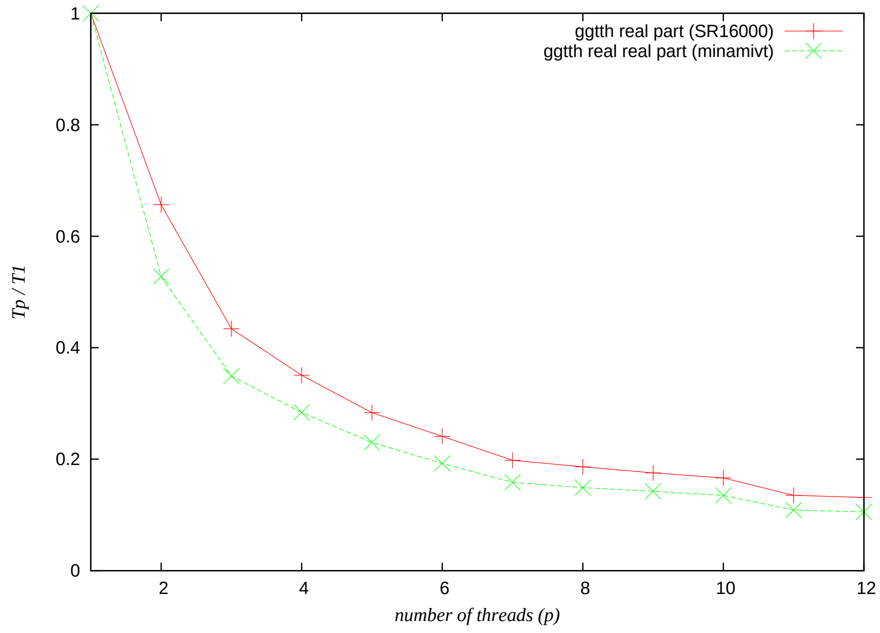

Figure 3. ρ_p vs. # threads p on SR16000 and *minamivt005* for $\gamma\gamma \to t\,\bar{t}\,H$ (1-loop pentagon)

only by the thread ID and the total number of iterations. The *dynamic* schedule adjusts the assignments at runtime. The threads can request more work at the completion of their previously assigned iterations, which is beneficial for load balancing in cases where the integrand behavior over the subregion is non-uniform. On SR16000 the *dynamic* schedule is not available, and neither is nested loop scheduling.

In [4] we reported timings for a one-loop non-scalar box problem, $M_4(f, g; \delta)$ from [27]. Presently we give timings for loop integral computations corresponding to the one-loop pentagon, two-loop self-energy, and two-loop ladder vertex diagrams and parameters specified below. The running times are total elapsed times spent in the integration code. The time for the extrapolation is negligible compared to the integration time.

For the one-loop pentagon diagram corresponding to the interaction $\gamma\gamma \to t\bar{t}H$, the integral (1) with $L = 1,\ N = 5,\ n = 4$, reduces to

$$\mathcal{I} = -2\int_0^1 dx \int_0^{1-x} dy \int_0^{1-x-y} dz \int_0^{1-x-y-u} du\ \frac{C}{(D - i\delta)^3}$$

via the simple transformation $x_1 = 1,\ x_2 = 1 - x - y - z - u,\ x_3 = y,\ x_4 = z,\ x_5 = u$ (and omitting the factor $\frac{1}{(4\pi)^{nL/2}}$). Here

$$\begin{aligned} D &= x_1 m_1^2 + x_2 m_2^2 + x_3 m_3^2 + x_4 m_4^2 + x_5 m_5^2 - x_1 x_2 s_1 - x_2 x_3 s_2 - x_3 x_4 s_3 \\ &\quad - x_4 x_5 s_4 - x_5 x_1 s_5 - x_1 x_3 s_{12} - x_2 x_4 s_{23} - x_3 x_5 s_{34} - x_4 x_1 s_{45} - x_5 x_2 s_{51}, \\ C &= 1, \end{aligned} \tag{7}$$

and the m_j are masses, the p_j are external momenta, $s_j = p_j^2$ and $s_{ij\cdots} = (p_i + p_j + \ldots)^2,\ 0 \le i, j \le 5$. Figure 3 displays the parallel running time T_p scaled by the sequential time T_1, i.e., $\rho_p = T_p/T_1$ as a function of the number of threads p, on *minamivt* and SR16000. The parallel time for the integration decreases significantly particularly on the Intel system (from $T_1 = 2726$ sec to $T_{12} = 287$ sec, so $\rho_{12} \approx 0.105$). The extrapolation converges to about 5 digits.

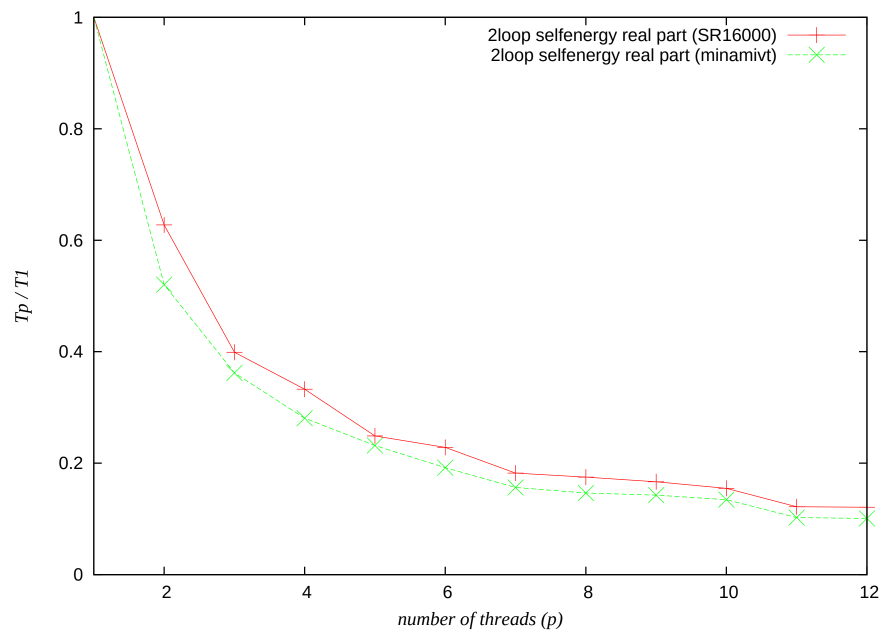

Figure 4. ρ_p vs. # threads p on SR16000 and *minamivt005* for 2-loop self-energy diagram

Two-loop self-energy integral results are reported for the integral

$$\mathcal{I} = \int_0^1 dx_1\ dx_2\ dx_3\ dx_4\ dx_5\ \delta(1 - \sum_{i=1}^{5} x_i)\ \frac{1}{CD},$$

with

$$\begin{aligned}
&D = -k^2(x_5(x_1+x_3)(x_2+x_4) + (x_1+x_2)x_3x_4 + (x_3+x_4)x_1x_2) + C\tilde{M}^2,\\
&C = (x_1+x_2+x_3+x_4)x_5 + (x_1+x_2)(x_3+x_4),\\
&\tilde{M}^2 = \sum_{1=1}^{5} x_i m_i^2.
\end{aligned}$$

The masses are $m_1 = m_2 = m_3 = m_4 = m_t$ (= 150 GeV, top quark mass), and $m_5 = m_Z$ (= 91.17 GeV, Z-boson mass). The integral is considered for an energy parameter $k^2 = 45000$ with $k^2/m_t^2 = 2$ and can be evaluated without extrapolation, in sequential time $T_1 \approx 27$ seconds on *minamivt005*. Figure 4 displays ρ_p as a function of the number of threads p on *minamivt* and SR16000.

The integral for the two-loop ladder vertex is (1) with $N = 6$, $n = 4$, and C, D given by

$$\begin{aligned}
&D = -C(x_1(p_1^2 - m_1^2) + x_2(p_2^2 - m_2^2) - x_3m_3^2 + x_4(p_1^2 - m_4^2) + x_5(p_2^2 - m_5^2) - x_6m_6^2)\\
&\quad + C_1(x_5^2p_2^2 + x_4^2p_1^2 - x_4x_5(p_3^2 - p_1^2 - p_2^2)) + C_2(x_2^2p_2^2 + x_1^2p_1^2 - x_1x_2(p_3^2 - p_1^2 - p_2^2))\\
&\quad + 2x_2x_3x_5p_2^2 + 2x_1x_3x_4p_1^2 - x_3(x_2x_4 + x_1x_5)(p_3^2 - p_1^2 - p_2^2),\\
&x_6 = 1 - x_1 - x_2 - x_3 - x_4 - x_5,\\
&C_1 = x_1 + x_2 + x_3,\\
&C_2 = 1 - x_1 - x_2,\\
&C = x_3(1 - x_1 - x_2 - x_3) + (x_1 + x_2)(1 - x_1 - x_2).
\end{aligned} \tag{8}$$

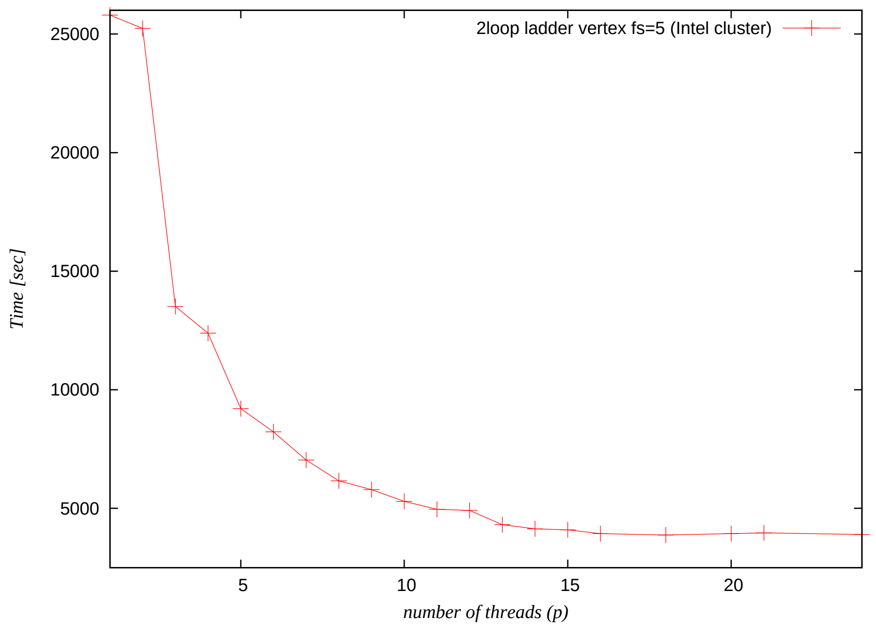

Figure 5. Time T_p vs. # threads p on WMU Intel cluster for 2-loop ladder vertex, $k^2/m_t^2 = 5$

The parameters are $m_1 = m_2 = m_4 = m_5 = m_t$ (= 150 GeV, top quark mass), and $m_3 = m_6 = m_Z$ (= 91.17 GeV, Z-boson mass). Figure 5 gives the running time T_p (on *thor*) for the real part integral (in seconds) as a function of the number of threads p, for parameter $k^2 = 112500$ with $k^2/m_t^2 = 5$ (denoted by *fs* = 5 in the key of figure 5). It is shown that the parallelization yields a spectacular savings in time for this compute-intensive problem (from $T_1 = 25800$ sec to $T_{24} = 3897$ sec).

Figure 6 gives corresponding times for the case $k^2/m_t^2 = 1$ which is part of the unphysical region for this problem; it is thus obtained without extrapolation by setting $\delta = 0$. The two curves in figure 6 represent the computations carried out with the GK15 and GK21 Gauss-Kronrod pairs in the x_1 dimension; the pair GK15 is used in the other dimensions. The (5D) integral is computed using the transformation derived in [1].

6. Concluding remarks

In addition to the lowest order or tree level, higher order corrections are required for an accurate theoretical prediction of the cross section of particle interactions, and for checking its agreement with the data observed at colliders. Feynman loop diagrams need to be taken into account, necessitating the calculation of *loop integrals*. While packages based on symbolic integration are available for one-loop integrals, symbolic reductions may lead to large sets of integrals, and analytic integration is not generally possible. The intensive nature of the direct computation performed by DCM motivates a parallelization of the individual problems, in particular for 3D and higher-dimensional integrals.

We have developed a multi-core parallelization on one or more function evaluation levels in the iterated integration procedure. In this paper we give timing results of the method implemented in OpenMP, which show good parallel performance for problems executed on a Xeon X5680, 6-(12 logical)-core CPU workstation, and furthermore on the (Hitachi) SR16000 system at KEK (Japan), and the Intel cluster (based on Xeon E5-2670 processors with 16 cores per node) at WMU. The Intel cluster at WMU and the SR16000 are suited for multi-threaded computations on each node and message passing between nodes. Future plans include hybrid (distributed

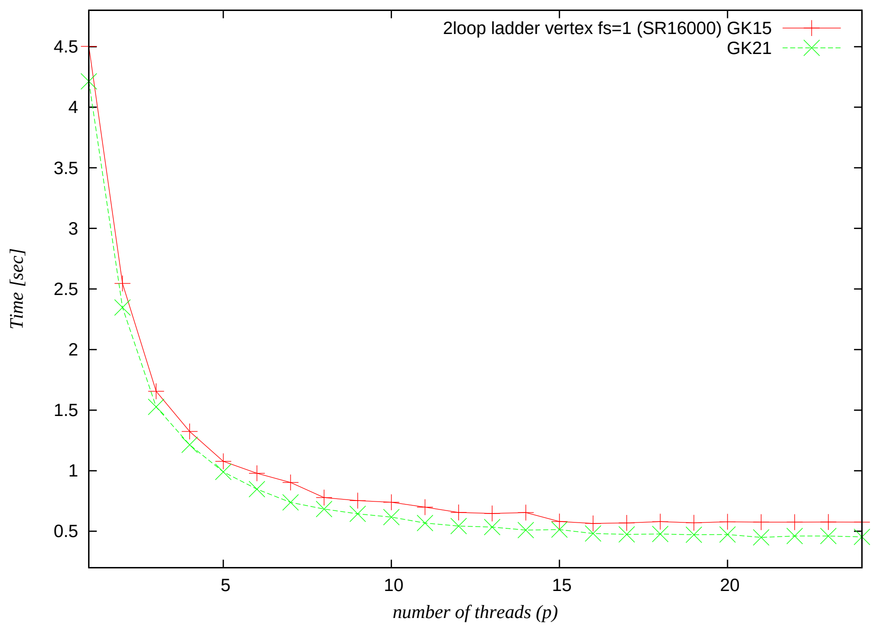

Figure 6. Time T_p vs. # threads p on SR16000 for 2-loop ladder vertex, $k^2/m_t^2 = 1$

shared memory) parallelizations using OpenMP and MPI.

Our future work will naturally involve implementations and testing for other and more computational intensive loop integrals.

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.12-07 of KEK. We also want to thank PhD student Vasilije Perovic at WMU, who generated a first version of the C code.

References
[1] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2004 *Computer Physics Communications* **159** 145–156
[2] Shanks D 1955 *J. Math. and Phys.* **34** 1–42
[3] Wynn P 1956 *Mathematical Tables and Aids to Computing* **10** 91–96
[4] de Doncker E and Yuasa F 2012 *IUPAP C20 Conference on Computational Physics (CCP 2011)* vol 402 (The Journal of Physics: Conference Series Dec. 2012)
[5] OpenMP web site, http://www.openmp.org
[6] 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 (Springer-Verlag)
[7] Berntsen J, Espelid T O and Genz A 1991 *ACM Trans. Math. Softw.* **17** 452–456
[8] Cools R and Haegemans A 2003 *ACM Transactions on Mathematical Software* **29** 287–296
[9] Berntsen J, Espelid T O and Genz A 1991 *ACM Trans. Math. Softw.* **17** 437–451
[10] de Doncker E and Yuasa F 2012 *Measurements in Quantum Mechanics* ISBN 978–953–51–0058–4
[11] de Doncker E, Shimizu Y, Fujimoto J, Yuasa F, Cucos L and Van Voorst J 2004 *Nuclear Instruments and Methods in Physics Research A* **539** 269–273 hep-ph/0405098
[12] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2007 *PAMM - Wiley InterScience Journal* **7**
[13] Yuasa F, de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T and Shimizu Y 2007 *XI Adv. Comp. and Anal. Tech. in Phys. Res.* PoS (ACAT07) 087, arXiv:0709.0777v2 [hep-ph]
[14] de Doncker E, Fujimoto J, Kurihara Y, Hamaguchi N, Ishikawa T, Shimizu Y and Yuasa F 2010 *XIV Adv. Comp. and Anal. Tech. in Phys. Res.* PoS (ACAT10) 073

[15] Yuasa F, Ishikawa T, Fujimoto J, Hamaguchi N, de Doncker E and Shimizu Y 2008 *XII Adv. Comp. and Anal. Tech. in Phys. Res.* PoS (ACAT08) 122; arXiv:0904.2823
[16] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2006 *LoopFest V, Stanford Linear Accelerator Center* http://www-conf.slac.stanford.edu/loopfestv/proc/present/DEDONCKER.pdf
[17] de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T, Kurihara Y, Shimizu Y and Yuasa F 2010 *Springer Lecture Notes in Computer Science (LNCS)* **6017** 139–154
[18] 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/1110.3587; PoS (CPP2010)011
[19] Yuasa F, Ishikawa T, Kurihara Y, Fujimoto J, Shimizu Y, Hamaguchi N, de Doncker E and Kato K 2010 Numerical approach to calculation of Feynman loop integrals arXiv:1109.4213v1 [hep-ph]; PoS (CPP2010)011
[20] de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T, Kurihara Y, Shimizu Y and Yuasa F 2011 *Journal of Computational Science (JoCS)* doi:10.1016/j.jocs.2011.06.003
[21] Yuasa F, de Doncker E, Hamaguchi N, Ishikawa T, Kato K, Kurihara Y and Shimizu Y 2012 *Journal Computer Physics Communications* **183** 2136–2144
[22] de Doncker E, Yuasa F and Kurihara Y 2012 *XV Adv. Comp. and Anal. Tech. in Phys. Res.* vol 368 (The Journal of Physics: Conference Series)
[23] Fritsch F N, Kahaner D K and Lyness J N 1981 *ACM TOMS* **7** 46–75
[24] Kahaner D, Moler C and Nash S 1988 *Numerical Methods and Software* (Prentice Hall)
[25] de Doncker E and Kaugars K 2010 *Procedia Computer Science* **1** 117–124
[26] de Doncker E and Yuasa F 2012 *Int. Conf. on Math. Modeling in Phys. Sciences*
[27] de Doncker E, Kaugars K, Cucos L and Zanny R 2001 *Proc. of Computational Particle Physics Symposium (CPP 2001)* pp 110–119