Acceleration of Feynman loop integrals in high-energy physics on many core GPUs

F Yuasa¹, T Ishikawa¹, N Hamaguchi¹, T Koike² and N Nakasato³

¹ High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan
² Seikei University, Musashino, Tokyo 180-8633, Japan
³ University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan

E-mail: fukuko.yuasa@kek.jp

Abstract. The current and future colliders in high-energy physics require theorists to carry out a large scale computation for a precise comparison between experimental results and theoretical ones. In a perturbative approach several methods to evaluate Feynman loop integrals which appear in the theoretical calculation of cross-sections are well established in the one-loop level, however, more studies are necessary for higher-order levels. Direct Computation Method (DCM) is developed to evaluate multi-loop integrals. DCM is based on a combination of multi-dimensional numerical integration and extrapolation on a sequence of integrals. It is a fully numerical method and is applicable to a wide class of integrals with various physics parameters. The computation time depends on physics parameters and the topology of loop diagrams and it becomes longer for the two-loop integrals. In this paper we present our approach to the acceleration of the two-loop integrals by DCM on multiple GPU boards.

1. Introduction

Much of the success of the Standard Model in high-energy physics rests on the result of the various precision measurements[1, 2]. These precision measurements require the knowledge of higher order quantum corrections. In Feynman diagrammatic approach followed by most of the automated systems, the cross sections can be obtained by computing directly the squared of matrix elements or amplitudes. In order to include higher order corrections in the Standard model or beyond it is indispensable to handle evaluations of Feynman loop integrals.

The most general form of the scalar Feynman loop integral with $L$ loops and $n$ internal lines is given by

\[
\mathcal{I} = \lim_{\epsilon \to 0^+} \int \prod_{j=1}^{L} \frac{d^4 l_j}{i(2\pi)^4} \prod_{r=1}^{n} \frac{1}{D_r}.
\]

(1)

where $l_j$ is the $j$-th loop momentum and we put the space-time dimension 4. Here

\[
D_r = q_r^2 - m_r^2 + i\epsilon,
\]

(2)

is the inverse of Feynman propagator with the momentum $q_r$ and $m_r$ the mass of the $r$-th particle. Using the generalized Feynman’s identity,

\[
\prod_{r=1}^{n} \frac{1}{D_r} = \Gamma(n) \int_0^1 dx_r \frac{\delta(1 - \sum x_r)}{(\sum x_r D_r)^n},
\]

(3)
we find

\[ I = \left( \frac{1}{4\pi} \right)^{4L/2} \Gamma \left( \frac{n - 4L}{2} \right) \times I, \]  

where

\[ I = (-1)^n \int_0^1 \prod_{i=1}^n dx_i \delta(1 - \sum x_i) \frac{C^{n-4(L+1)/2}}{(D - iC)^{n-4L/2}}. \]  

Here \( D \) function is a polynomial in Feynman parameters \( \{x_i\} \) and it contains physics parameters such as external momenta and particle masses and \( C \) function is a polynomial in \( \{x_i\} \). Both functions are determined by the topology of Feynman diagram. The details on how to get them are summarized in [3].

In the case of modern theoretical calculations in high-energy physics, loop integrals are performed with new mathematical methods and are based on an intense use of computer power to get theoretical predictions of cross-sections for particle reactions. Depending on the number of loops, legs, and physics parameters, it may take long computation time. In fact it is reported that the two-loop box (four legs) with arbitrary masses requires long computation time[3]. To reduce this practical problem, current hardware trends such as many-core/multi-core computation can be available. Recently GPU boards have been used in scientific applications which require a huge computation time. Also in high-energy physics successful computations on GPU have been reported in [4, 5]. In this paper, we show our approach to reduce the computation time of loop integrals on multiple GPUs and show its performance.

The plan of this paper is as follows; we explain numerical techniques of our program for loop integrals in Section 2. The environment for the acceleration of the computation and the results of the performance test are shown in Section 3. Section 4 gives summary of this paper.

2. Numerical Techniques

One of the difficult points for the numerical computation of loop integrals is vanishing denominators in (5). Our program code for loop integrals is a combination of numerical multi-dimensional integration and extrapolation in order to reduce the difficulty. It consists of the following three steps:

(i) Let \( \epsilon \) in (5) be finite as \( \epsilon_l = \epsilon_0 / (\text{const.})^l \), with \( l = 0, 1, 2, \ldots \), where a starting \( \epsilon, \epsilon_0 \), and constant are positive numbers.

(ii) Evaluate the multi-dimensional integral \( I \) in (5) numerically. Thanks to \( \epsilon_l \) we get finite value corresponding to each \( l \). Repetition of integration gives a sequence of \( I(\epsilon_l), l = 0, 1, 2, \ldots \).

(iii) Extrapolate the sequence \( I(\epsilon_l) \) to the limit and determine \( I \).

We call above prescriptions Direct Computation Method (DCM). Please note that in the unphysical region extrapolation is not necessary and we can put \( \epsilon = 0 \) in (5) because singularities do not appear and the integration is not difficult in this region. On the other hand, singularities appear in the physical region and we should take the parameters for extrapolation, \( \epsilon_0 \) and constant, appropriately corresponding to physics parameters such as masses. In current DCM, the way to define them automatically is not included and we take them empirically.

In DCM, the integration routine plays a central role. The numerical result of the integral should be as precise as possible for each \( \epsilon_l \), so that the extrapolation of \( I(\epsilon_l) \) converges well. For multi-dimensional integration we can make use of any method when once it gives a good approximation. We have been using two different integration methods. One is an adaptive algorithm routine, DQAGE, in QUADPACK[6] and the other is Double Exponential Formulas[7], shortly DE. We call the combination of DQAGE and and the extrapolation technique DQ-DCM and call the combination of DE and the extrapolation technique DE-DCM.
We use Wynn’s ε algorithm[8, 9, 10] for the extrapolation and we usually repeat the integration 15 times. Since computation time for the extrapolation of the sequence is negligible small compared to that for the integration, we do not go further about the extrapolation.

Throughout this paper, we concentrate exclusively on the acceleration of DE-DCM. Let us explain about DE formulas briefly. An integral with the interval \((a, b)\) is given as

\[
I = \int_a^b f(x)dx
\]  \hspace{1cm} (6)

and with \(x = \phi(t)\), \(a = \phi(-\infty)\) and \(b = \phi(\infty)\).

\[
I = \int_{-\infty}^{\infty} g(t)dt,
\]  \hspace{1cm} (7)

where

\[
g(t) = f(\phi(t))\phi'(t).
\]  \hspace{1cm} (8)

Application of the trapezoidal rule with mesh size \(h\) gives

\[
I_h = h \sum_{k=-\infty}^{\infty} f(\phi(kh))\phi'(kh).
\]  \hspace{1cm} (9)

Here note the infinite sum should be truncated appropriately as

\[
I_h^N = h \sum_{k=-N_-}^{N_+} f(\phi(kh))\phi'(kh),
\]  \hspace{1cm} (10)

\[
N = N_- + N_+ + 1,
\]  \hspace{1cm} (11)

where \(N\) is the number of function evaluation. In DE-DCM we take the following transformation

\[
I = \int_0^1 f(x)dx,
\]  \hspace{1cm} (12)

\[
\phi(t) = \frac{1}{2}(1 + \tanh(\frac{\pi}{2}\sinh(t))).
\]  \hspace{1cm} (13)

Through our experiences in DE-DCM[11], we point out following things:

• DE is effective for singularities appearing in the boundary of the integration domain,
• When singularities appear in the integration domain, we may split the integration domain,
• DE has been developed for the one-dimensional integration, we apply it in an iterative way,
• The parallelization of DE is straightforward.

2.1. Two-loop self-energy integral

In this paper, we take two-loop self-energy integrals (two legs) as an example because there is a rich literature on the analytic methods or semi-numerical methods such as [12, 13, 14] and we can compare numerical results by DE-DCM to analytic ones.
\[ D = -s(x_5(x_1 + x_3)(x_2 + x_4) + (x_1 + x_2)x_3x_4 + (x_3 + x_4)x_1x_2) + C\tilde{M}^2, \]  
(14) 
\[ C = (x_1 + x_2 + x_3 + x_4)x_5 + (x_1 + x_2)(x_3 + x_4), \]  
(15) 
\[ \tilde{M}^2 = \sum_{i=1}^{5} x_i m_i^2. \]  
(16) 

Here, \( m_i \) is a mass of an internal particle indicated Feynman parameter \( x_i \) and \( s \) is a physics parameter.

**Figure 1.** Two-loop self-energy diagram with five internal lines. \( x_i \) is Feynman parameter related to each line and \( p \) a physics parameter which determines \( s \) in (14) as \( p^2 \) is \( s \).

**Figure 2.** Numerical results of real part of two-loop self-energy integrals with \( n = 1024 \) and \( h = 8.36181640625 \times 10^{-3} \). All filled circles are results after the extrapolation. The starting \( \epsilon \) for the extrapolation, \( \epsilon_0 \), is \( 1.2^{-15} \) and constant is 1.2. In this computation we did not split the integration domain.

The numerical results of the real part of the integrals for \( 4.0 \leq s \leq 10.0 \) GeV\(^2 \) in the physical region with mass parameters as \( m_1 = m_2 = m_5 = 10.0 \) GeV and \( m_3 = m_4 = 0.0 \) GeV are plotted in Fig.2. For positive \( s \), we repeated the integration to obtain \( I(\epsilon_l) \) for the extrapolation. For above physics parameter setting we don’t have to split the integration domain because the values of masses given are not so small compared to \( s \). All the computation shown here has been done in the double precision arithmetic because there is no infrared divergence in this case. For example, for the case \( s = 10.0 \) GeV\(^2 \), a numerical result is \( 0.26673229 \times 10^{-1} \pm 0.191 \times 10^{-8} \). The result using analytic formulas in [12] is \( 0.266732 \times 10^{-1} \) and the agreement is excellent.

3. Acceleration of Computation

In this section firstly we explain the software environment and the modification of DE-DCM program code for the acceleration. Then we show the hardware environment for the computation time measurement and results.
3.1. Software Environment
We use LSUMP compiler[15] which has been developed by Nakasato et al. [15]. In addition to the LSUMP, we use Goose[16] software package which enables us parallel programming with Goose directives on the SIMD hardware accelerator. Both are originally developed for astrophysical N-body simulations and several very effective N-body simulation codes are available on GPU or special-purpose computers. With the combination of LSUMP and Goose, we only have to define the inner-most computation loop in the program. In order to use LSUMP and Goose, we transform the variables so as to make integration domain as [0, 1] and then divide multiple computation loops for the multi-dimensional integration to double loops. Though the double precision and quadruple precision arithmetic are available with LSUMP[17] and Goose[18], all computation shown in this paper has been done in double precision arithmetic for the physics parameters given in Section 2.

3.2. Hardware Environment
We setup two kinds of hardware environment. One is a host computer (AMD Phenon II x6 1090T 3.2GHz) with two AMD/ATI HD5870 GPU boards and the other is a host computer (AMD FX-8150 3.6GHz) with four AMD/ATI HD6970 GPU boards. Parameters of each GPU board are shown in Table 1.

|                     | HD5870 | HD6970 |
|---------------------|--------|--------|
| Clock [MHz]          | 850    | 880    |
| number of cores      | 1600   | 1536   |
| SP FP [GFLOP]        | 2740   | 2732   |
| DP FP [GFLOP]        | 544    | 683    |

Table 2. Computation time for $s = 5.0$ GeV$^2$ with $N = 1024$ and $s = -1.0$ GeV$^2$ with $N = 2048$.

| $s$      | GPU | # of boards | time [sec] |
|----------|-----|-------------|------------|
| 5.0      | HD5870 | 1     | 242.313    |
|          | HD6870 | 2     | 122.251    |
| 5.0      | HD6970 | 1     | 221.059    |
|          | HD6970 | 2     | 110.715    |
|          | HD6970 | 3     | 74.519     |
|          | HD6970 | 4     | 55.638     |
| -1.0     | HD6970 | 4     | 882.288    |

3.3. Computation Time Measurement
We have measured the computation time using one to four GPU boards taking two-loop self-energy integral for $s = 5.0$ GeV$^2$ with $N = 1024$ and $h = 8.362 \times 10^{-3}$ as examples. The measured results are shown in Table 2. For positive $s$, in the physical region, we should repeat the integration to obtain the sequence $I(\epsilon_l)$ and extrapolate it to accomplish the loop integral. However, the values of computation time shown in Table 2 are time required for only one repetition for simplicity. Therefore the amount of time in total becomes 15 times longer in the practical computation. We also run the same program code compiled by Goose on a host computer using one core with same integration parameters for $s = 5.0$ GeV$^2$ and roughly speaking it takes about one day. For example on AMD Phenon II x6 1090T 3.2GHz, it took 79903.860 seconds.

In DE-DCM, the expected computation time for one repetition in both the unphysical region and the physical region is more or less the same and it depends on the value of $N$. For comparison
we also show the computation time for \( s = -1.0 \) GeV\(^2\) with \( N = 2048 \) and \( h = 4.181 \times 10^{-3} \) in Table 2 even though it is in the unphysical region. Please note that the computation time with \( N = 2048 \) is about 16 times longer compared to one with \( N = 1024 \), because the integration is 4-dimensional one.

4. Summary
We have been developing a fully numerical method for Feynman loop integrals, Direct Computation Method (DCM). DCM is based on the numerical integration and extrapolation method and it can handle the singularities due to the vanishing denominator in the expression of loop integrals. The numerical integration plays a central role in DCM and in this paper we used Double Exponential Formulas in an iterated way for the multi-dimensional integration. Since it is fully numerical, it can be applicable to multi-loop integrals with multi-leg and arbitrary physics parameters in principle. However, the computation time becomes longer for higher-order loops and reducing it is a crucial problem. In this paper we demonstrated the acceleration of the loop integrals taking two-loop self-energy integral as an example. In the performance test, it is clearly shown that the reduction of the computation time is excellent on multiple GPU boards. Concerning the further development we need to test various two-loop integrals with more legs and various physics parameters different from those in this paper.

Acknowledgments
We acknowledge the support from Grant-in-Aid for Scientific Research (No.24540292 and No. 23540328) of JSPS and the Large Scale Simulation Program No.12-07 of KEK. This work is supported in part by Strategic Programs for Innovative Research (SPIRE) Field 5 "The origin of matter and the universe". One of authors (F.Y.) wishes to thank Prof. E de Doncker for her continuous encouragement.

References
[1] D. Perret-Gallix, Computational Particle Physics for Event Simulation and Analysis, a talk in CCP2012.
[2] J. Blümlein, Why precision, arXiv:1205.4991v2 [hep-ph].
[3] F. Yuasa, E. de Doncker, N. Hamaguchi, T. Ishikawa, K. Kato, Y. Kurihara, J. Fujimoto, Y. Shinizui, Numerical computation of two-loop box diagrams with masses, Comput. Phys. Commun. 183 (2012) 2136.
[4] J. Kanzaki, Monte Carlo integration on GPU, DOI:10.1140/epjc/s10052-011-1559-8.
[5] M. Hayakawa, K.-I. Ishikawa, Y. Osaki, S. Takeda, S. Uno, N. Yamada, Improving many flavor QCD simulations using multiple GPUs, PoS Lattice2010:325,2010.
[6] R. Piessens, E. de Doncker, C. W. Ubelhuber and D. K. Kahaner, QUADPACK, A Subroutine Package for Automatic Integration, Springer Series in Computational Mathematics. Springer-Verlag, 1983.
[7] H. Takahasi and M. Mori, Double Exponential Formulas for Numerical Integration, RIMS Kyoto Univ. 9 (1974) 721.
[8] D. Shanks, Non-linear transformations of divergent and slowly convergent sequences, J. Math. Phys. 34 (1955) 1.
[9] P. Wynn, On a device for calculating the \( e_m(S_m) \) transformations, Mathematical Tables Aids to Computing 10 (1956) 91.
[10] P. Wynn, On the convergence and stability of the epsilon algorithm, SIAM J. Numer. Anal. 3 (1966) 91.
[11] F. Yuasa and N. Hamaguchi, Double Exponential Integration Method with algorithm for the Feynman Loop Integration Calculation [in Japanese], 2008-HPC-114 No.6 pp. 31-36.
[12] D.J. Broadhurst, J. Fleischer and O.V. Tarasov, Two-loop two-point functions with masses: asymptotic expansions and Taylor series, in any dimension, Z.Phys. C60 (1993) 287.
[13] J. Fujimoto, Y. Shimizu, K. Kato and Y. Oyanagi, Numerical Approach to Loop Integrals, in Proc. of the Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems in High Energy and Nuclear Physics, January 13-18, 1992, l’Agelonde.
[14] G. Passarino and S. Uccirati, Algebraic-numerical evaluation of Feynman diagrams: two-loop self-energies, Nucl. Phys. B629 (2002) 97.
[15] N. Nakasato and J. Makino, A Compiler for High Performance Computing With Many-Core Accelerators, in: IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1-9.
[16] A. Kawai, T. Fukushige, N. Nakasato and T. Narumi, *Goose and DS-CUDA* [in Japanese], "ENSEMBLE", The Bulletin of The Molecular Simulation Society of Japan, Vol.14, No. 2, (2012.04.30) 81-84.

[17] N. Nakasato, T. Ishikasa, J. Makino and F. Yuasa, *Fast Quad-Precision Operations On Many-Core Accelerators* [in Japanese], Vol2009-HPC-121 No.39.

[18] T. Koike, *Speed-up of numerical Feynman loop integration by graphics processing unit* [in Japanese], unpublished Masters thesis, Seikei University, Musashino, Tokyo, Japan, 2012.