Numerical multi-loop integration on heterogeneous many-core processors

E de Doncker[1], F Yuasa[2], A Almulihi[1], N Nakasato[3], H Daisaka[4], T Ishikawa[2]

[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
[3] University of Aizu, Aizu-wakamatsu, Fukushima, 965-8580, Japan
[4] Hitotsubashi University, 2-1, Naka, Kunitachi, Tokyo, 186-0801, Japan

E-mail: elise.dedoncker@wmich.edu, fukuko.yuasa@kek.jp, ahmedhassan.almulihi@wmich.edu, nakasato@u-aizu.ac.jp, daisaka@phys.science.hit-u.ac.jp, tadashi.ishikawa@kek.jp

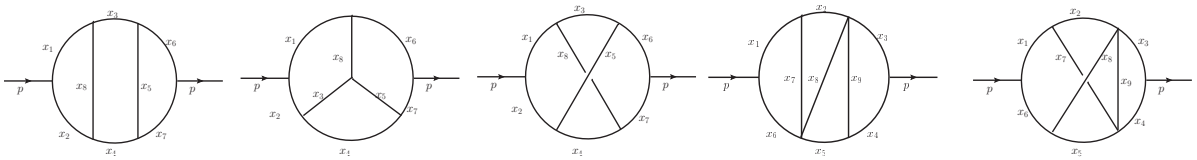
Abstract. We report on multi-loop integral computations executed on a PEZY/Exascaler large-scale (immersion cooling) computing system. The programming model requires a host program written in C++ with a PZCL (OpenCL-like) kernel. However the kernel can be generated by the Goose compiler interface, which allows parallelizing program loops according to compiler directives. As an advantage, the executable derived from a program instrumented with Goose pragmas can be run on multiple devices and multiple nodes without changes to the program. We use lattice rules and lattice copy (composite) rules on PEZY to approximate integrals for multi-loop self-energy diagrams with and without masses. GPU results are also given and the performnce on the different architectures is compared.

1. Introduction

Higher order corrections are required for accurate theoretical predictions of the cross-section for particle interactions. Loop diagrams are taken into account, leading to the evaluation of loop integrals, for which analytic integration is not generally possible. The goal is to perform accurate loop integral computations, and to develop computer programs to evaluate multi-loop Feynman integrals numerically/directly. Previously in [1, 2, 3], we reported precise numerical results for 2-, 3- and 4-loop Feynman integrals using adaptive multi-dimensional integration and linear extrapolation. In [4, 5] we handled 2- and 3-loop integrals with (transformed) lattice rules on GPUs. We further considered diagrams with massless internal lines [6], where dimensional regularization was treated numerically with a linear extrapolation as the dimensional regularization parameter tends to zero. Other authors evaluated shifted lattice rules (originally proposed in [7]) on GPUs for integrals resulting from sector decomposition [8], which was also implemented in pySecDec [9]. In the present work we use composite lattice rules for 3- and 4-loop (finite) integrals, and focus on the parallel performance on systems with GPU or PEZY accelerators.

The L-loop integral with N internal lines can be represented by $\mathcal{I} = (4\pi)^{-\nu L/2}\mathcal{F}$ with

$$\mathcal{F} = \Gamma(N - \frac{\nu L}{2})\,(-1)^N \int_{\mathcal{C}_N} \prod_{r=1}^{N} dx_r\, \delta(1 - \sum x_r) \frac{C^{N-\nu(L+1)/2}}{(D - i\varrho C)^{N-\nu L/2}}$$
$$= \Gamma(N - \frac{\nu L}{2})\,(-1)^N \int_{\mathcal{S}_{N-1}} \prod_{r=1}^{N-1} dx_r\, \frac{C^{N-\nu(L+1)/2}}{(D - i\varrho C)^{N-\nu L/2}} \qquad (1)$$



(a) (b) (c) (d) (e)

Figure 1. Sample 3-loop self-energy diagrams with masses [10] / massless [11]: (a) Diagram $(2t/L_0)$ with $N = 8$, (b) Diagram $(2u)$ with $N = 8$, (c) Diagram $(2v/N_0)$ with $N = 8$; 4-loop self-energy diagrams with massless internal lines [11]: (d) Diagram *M44* with $N = 9$, (e) Diagram *M45* with $N = 9$.

where C and D are polynomials determined by the topology of the corresponding diagram and physical parameters, ν is the space-time dimension (i.e., $\nu = 4$ unless used for regularization), and $\varrho = 0$ unless D vanishes in the interior of the domain. The domain $\mathcal{C}_N$ is the N-dimensional unit cube in the first integral of (1), and $\mathcal{S}_{N-1}$ is the $(N-1)$-dimensional unit simplex in the second integral, $\mathcal{S}_d = \{\mathbf{x} \in \mathcal{C}_d \mid \sum_{j=1}^{d} x_j \leq 1\}$.

For the cases presented here, D has integrable singularities on the boundaries of the domain. The diagrams are given in Fig 1, including three 3-loop self-energy diagrams $l(2t), l(2u), l(2v)$ with $N = 8$ and considered with masses in [10] (Laporta Fig 2) and massless (L_0 and N_0) in [11], and two massless 4-loop diagrams, *M44* and *M45* with $N = 9$ from [11] (Baikov and Chetyrkin).

2. Integration

2.1. Lattice rules

For the integration we transform the integral over $\mathcal{S}_d$ of (1) to $\mathcal{C}_d$ for dimension $d = N-1$, and use rank-1 lattice rules, given by $Qf = \frac{1}{n}\sum_{j=0}^{n-1} f(\{\frac{j}{n}\mathbf{z}\})$ (see, e.g., [12] for a comprehensive treatment). Here $\mathbf{z}$ is an integer *generator* vector of length d with components $z \in \mathcal{Z}_n = \{1 \leq z < n \mid gcd(z, n) = 1\}$, where $\{\mathbf{x}\}$ denotes the vector obtained by taking the fractional part of each component of $\mathbf{x}$. Classically n is prime [13, 14] (i.e., $\mathcal{Z}_n = \{1, 2, \ldots, n-1\}$); extensions for non-prime n have been given [15, 16]. We precompute the generators $\mathbf{z}$ for various numbers of points n using the component by component (CBC) algorithm [17, 16]. The CBC algorithm runs in $\mathcal{O}(d\,n\,\log(n))$ time and $\mathcal{O}(n)$ space.

2.2. Composite/embedded lattice rules

An embedded sequence is given in [12], where each rule of the sequence applies a scaled rank-1 rule Q_0 to the subregions obtained by subdividing $\mathcal{C}_d$ into m equal parts in each of r coordinate directions,

$$Q_r f = \frac{1}{m^r n} \sum_{k_r=0}^{m-1} \cdots \sum_{k_1=0}^{m-1} \sum_{j=0}^{n-1} f\left(\{\frac{j}{n}\mathbf{z} + \frac{1}{m}(k_1, \ldots, k_r, 0, \ldots, 0)\}\right), \quad 0 \leq r \leq d \tag{2}$$

for n and m relatively prime.The points of Q_r are included (*embedded*) in Q_{r+1} for $0 \leq r < d$. Q_r has $m^r n$ points and is of rank r for $1 \leq r \leq d$, and Q_d is the m^d-copy rule of Q_0. An error estimate for Q_d is calculated using a sequence of rules $Q^{(i)},\ 1 \leq i \leq d$, of order $m^{d-1}n$ and embedded in Q_d [12].

2.3. Transformation

Lattice rules enjoy favorable convergence properties for smooth, 1-periodic functions. In view of the singular behavior of the integrand of (1) at the boundaries of the domain, we use a $\sin^m$ transformation [18, 19] of fairly high order ($m = 6$) to also smoothen the singularity. For an integral $\mathcal{I}f = \int_0^1 f(x)\,dx$, this is $x = \Psi_m(t) = \theta_m(t)/\theta_m(1),\ \ m = 1, 2, \ldots,$ with $\theta_m(t) = \int_0^t \sin^m(\pi u)\,du$, so that $\Psi_6(t) = t - (45\sin(2\pi t) - 9\sin(4\pi t) + \sin(6\pi t))/(60\pi)$, and $\Psi_6'(t) = \frac{16}{5}\sin^6(\pi t)$.

Table 1. $(m = 1, 2)$ LR results for 3-loop massive self-energy diagrams on GPU

DIAGRAM	N	# PTS n	m	RESULT	ABS.ERR.	REL.ERR.	TIME [s]
Fig 1(a) $(2t)$	8	400M	1	0.2796089827126	5.94 e-08	2.12 e-07	1.0
			2	0.2796089232826	2.52 e-14	9.01 e-14	127.8
		5^{13} $(7D)$	1	0.2796089226167	6.66 e-10	2.38 e-09	3.0
			2	0.2796089232824	1.63 e-13	5.83 e-13	390.1
Exact:				0.2796089232826			
Fig 1(b) $(2u)$	8	400M	1	0.1826272683315	3.08 e-08	1.69 e-07	1.0
			2	0.1826272375394	2.18 e-13	1.19 e-12	128.1
		5^{13} $(7D)$	1	0.1826272372834	2.56 e-10	1.40 e-09	3.1
			2	0.1826272375391	1.46 e-13	7.99 e-13	391.0
Exact:				0.1826272375392			
Fig 1(c) $(2v)$	8	400M	1	0.1480133458323	4.19 e-08	2.83 e-07	1.6
			2	0.1480133039588	2.37 e-13	1.60 e-12	199.4
		5^{13} $(7D)$	1	0.1480133033037	6.55 e-10	4.43 e-09	4.8
			2	0.1480133039581	3.46 e-13	2.34 e-12	608.1
		5^{13} $(15 \to 7D)$	1	0.1480133034035	5.55 e-10	3.75 e-09	4.8
			2	0.1480133039583	8.97 e-14	6.06 e-13	608.6
Exact:				0.1480133039584			

Table 2. $(m = 1, 2)$ LR results for 4-loop massless self-energy diagrams on GPU

DIAGRAM	N	# PTS n	m	RESULT	ABS.ERR.	REL.ERR.	TIME [s]
Fig 1(d) (*M44*)	9	100M	1	55.657754	7.25 e-02	1.30 e-03	0.4
			2	55.586092	8.38 e-04	1.51 e-05	113.4
		400M	1	55.600351	1.51 e-02	2.72 e-03	1.8
			2	55.585416	1.62 e-04	2.91 e-06	452.9
		5^{13} $(15 \to 8D)$	1	55.577700	7.55 e-03	1.36 e-04	5.4
			2	55.585135	1.19 e-04	2.14 e-06	1382
Exact:				55.585254			
Fig 1(e) (*M45*)	9	100M	1	52.058688	4.08 e-02	7.84 e-04	0.5
			2	52.018436	5.67 e-04	1.09 e-05	114.9
		400M	1	52.014437	3.43 e-03	6.59 e-05	1.8
			2	52.017790	7.92 e-05	1.52 e-06	459.4
		5^{13} $(15 \to 8D)$	1	52.012072	5.80 e-03	1.11 e-04	5.5
			2	52.017807	6.17 e-05	1.19 e-06	1402
Exact:				52.017869			

3. GPU results

For the GPU results, the program is implemented in CUDA C and executed on the *thor* cluster of the Center for High Performance Computing and Big Data at WMU. The host process runs on a node with Intel Xeon E5-2670, 2.6 GHz dual CPU, and the lattice rule is evaluated on a Kepler-20m GPU, using 64 blocks and 512 or 1024 threads per block on the GPU. The GPU has 2496 CUDA cores and 4800 MBytes of global memory. The lattice generator vector $\mathbf{z}$ is precomputed and communicated (as a one-dimensional array of length d) from the main program to the CUDA kernel.

Table 1 shows results for lattice rules ($m = 1$) with $n =$ 100M (million), 400M and 5^{13} points, and for their composite versions with $m = 2$. Two rules were generated (using Lattice Builder [20]) with 5^{13} points, one $(7D)$ for dimension $d = 7$ (which is the dimension of the $2t, 2u$ and $2v$ integrals) and one $(15 \to 7D)$ for dimension 15, which is projected to $d = 7$. The methods yield high accuracy for these problems, and further provide a significant improvement from the $m = 1$ to the $m = 2$ applications, which comes with a price to pay in the number of function evaluations and hence the execution times.

Table 3. $(m = 2)$ 400M results for 3-loop massive self-energy diagram $(2u)$ on Suiren2; Abs. err. = 4.28e-11; Loop blocks of size 128×32; [Compare to GPU: 128.1 s]

DIAGRAM	N	#PTS. n	TIMES [S] [ON SUIREN2], NXTY: X NODES, Y TASKS			
			1 node	2 nodes	4 nodes	8 nodes
Fig 1(b) $(2u)$	8	400M	n1t1: 140.8			
			n1t2: 71.6	n2t1: 73.7		
			n1t4: 36.7	n2t2: 37.0	n4t1: 36.6	
			n1t8: 19.2	n2t4: 19.0	n4t2: 18.9	n8t1: 18.8
				n2t8: 10.0	n4t4: 9.9	n8t2: 9.8
					n4t8: 5.7	n8t4: 5.5
						n8t8: 3.5

Notwithstanding the much more complicated integrals, Table 2 reveals similar trends for the 4-loop massless diagrams. The 5^{13}-point rule $(15 \to 8D)$ is projected from dimension 15 to $d = 8$.

4. PEZY results

The PEZY results are obtained on the Suiren2 system based on the MIMD manycore PEZY-SC2 accelerator [21] at the High Energy Accelerator Research Organization (KEK) in Tsukuba, Japan. The system is manufactured by PEZY Computing/Exascaler Inc. [22] and uses direct liquid immersion cooling for power efficiency. Power consumption is listed as 47.40 kW, and power efficiency is 16.759 GFlops/Watts. The machine at KEK is a smaller configuration of the Shoubu system B, a ZettaScaler-2.2 supercomputer at the Advanced Center for Computing and Communication, RIKEN, Japan, which ranks first on the Nov. 2018 list of the Green 500. Suiren2 ranked second after Shoubu in Nov. 2017.

The configuration at KEK has 6 tanks, two bricks (high density server boards) per tank, four Xeon D-1571 16C 1.3GHz nodes per brick, 8 PEZY-SC2 boards per node. This makes for a total of $6 \times 2 \times 4 = 48$ nodes and $48 \times 6 = 384$ PEZY-SC2 boards. The node interconnect is Infiniband EDR. Each PEZY-SC2 board has 64 GB of memory associated with it. Half of the nodes further have 64 GB, the other half 32 GB of memory, so that the total amount of memory is $(384 \times 64 + 24 \times 32 + 24 \times 64)$ GB $= 26,880$ GB. With 1984 cores per board and 16 cores per Xeon D-1571 host processor, the total number of cores is $(16 + 8 \times 1984) \times 48 = 762,624$. For the Linpack performance of Suiren2, Rmax is 798 TFlop/s with Nmax of 1,238,016, and the theoretical peak (Rpeak) is 1,082.6 TFlop/s (see [23]).

The programming model for PEZY-SC2 requires a kernel in PZCL, an OpenCL-like language, and a host program in C++. However, we generated the kernels via the Goose compiler interface [24, 25], which takes a program instrumented with Goose compiler directives to parallelize the code. A pragma for the composite rule computation with $m = 2$ is illustrated below, where n is the number of points in the lattice rule, `dim` is the dimension (d), `p_dim`$= 2^d$ and the outer loop runs over loop blocks of size $2^d \times 32$. The precision labeled `double-double` was overridden by the Goose compiler option –dd-double that indicates the calculation within the section is done in double precision.

```
const int dim = 8;
const int p_dim = 1 << dim;
long long int nj = p_dim*32;
long long int ni = p_dim*(long long int)n/nj;
#pragma goose parallel for loopcounter(i,j) precision("double-double")
for (i = 0; i < ni; i++) {
   ...
   for(j = 0; j < nj; j++) {
     ...
   }
   ...
}
```

Table 4. $(m = 2)$ 400M results for 3-loop massless self-energy diagram L_0 on Suiren2; Abs. err. = 8.98e-07, Rel. err. = 4.33e-08; Loop blocks of size 128×32; [Compare to GPU: 128.0 s]

DIAGRAM	N	#PTS. n	TIMES [S] ON SUIREN2 nXTY: X NODES, Y TASKS			
			1 node	2 nodes	4 nodes	8 nodes
Fig 1(a) (L_0)	8	400M	n1t1: 143.0			
			n1t2: 71.5	n2t1: 71.5		
			n1t4: 36.7	n2t2: 36.6	n4t1: 36.6	
			n1t8: 19.0	n2t4: 18.9	n4t2: 18.9	n8t1: 18.7
				n2t8: 10.1	n4t4: 9.9	n8t2: 9.8
					n4t8: 5.7	n8t4: 5.5
						n8t8: 3.5

Table 5. $(m = 2)$ 5^{13} pts. $(7D)$ results for 3-loop massless self-energy diagram L_0 on Suiren2; Abs. err. = 1.71e-07, Rel. err. = 8.22e-09; Loop blocks of size 128×32; [Compare to GPU: 390.6 s]

DIAGRAM	N	#PTS. n	TIMES [S] ON SUIREN2, NXTY: X NODES, Y TASKS			
			1 node	2 nodes	4 nodes	8 nodes
Fig 1(a) (L_0)	8	5^{13}	n1t1: 432.7			
			n1t2: 217.8	n2t1: 217.7		
			n1t4: 111.7	n2t2: 111.4	n4t1: 112.0	
			n1t8: 58.3	n2t4: 57.6	n4t2: 56.6	n8t1: 56.2
				n2t8: 30.7	n4t4: 29.7	n8t2: 29.0
					n4t8: 16.7	n8t4: 15.9
						n8t8: 9.9

Tables 3-6 display Suiren2 timings for some of the diagrams of interest. The times are labeled as nXtY indicating the number of nodes (X) and number of tasks per node (Y) used. Table 3 handles the $l(2u)$-diagram using the composite $(m = 2)$ lattice of the 400M-point rule. Results for the massless 3-loop diagram L_0 are included in Tables 4 and 5 for 400M and 5^{13} points, repectively.

Table 6. $(m = 2)$ results for 4-loop massless self-energy diagram *M45* on Suiren2; Loop blocks of size 128×32; 400M pts.: Abs. err. = 2.57e-05, Rel. err. = 4.94e-07, [Compare to 400M pts. GPU time: 459.5 s]; 5^{13} pts.: Abs. err. = 1.02e-06, Rel. err. = 1.95e-08, [Compare to 5^{13} pts. GPU time: 1402 s]

DIAGRAM	N	#PTS. n	TIMES [S] ON SUIREN2, NXTY: X NODES, Y TASKS			
			1 node	2 nodes	4 nodes	8 nodes
Fig 1(e) (*M45*)	9	400M	n1t1: 332.5			
			n1t8: 43.3			
				n2t8: 22.4		
					n4t8: 11.9	
						n8t8: 6.7
Fig 1(e) (*M45*)	9	5^{13}				
				n2t8: 68.6		
					n4t8: 35.8	
						n8t8: 19.5

Table 6 gives timings for the 4-loop massless diagram *M45* for 400M and 5^{13} points. Note the comparisons of the Suiren2 with the GPU times (listed in the Table captions). The Suiren2 times are significantly lower, e.g., for n8t8 (6.7s vs. 459.5s for 400M points, and 19.5s vs. 1402s for 5^{13} points).

5. Conclusions

A parallel composite/embedded lattice rule method is presented for the calculation of Feynman loop integrals, using a periodizing transformation that deals with singularities at the boundaries of the integration domain. The algorithm is implemented in CUDA C for GPUs, and in C++ with Goose compiler directives for the PEZY-SC2 Exascaler accelerator. Test results are given for classes of 3- and 4-loop self-energy loop diagrams, with or without masses, and using simple $(m = 1)$ and composite $(m = 2)$ lattice rules. The accuracy improves considerably from $(m = 1)$ to $(m = 2)$ and supersedes that of adaptive parallel integration with the ParInt package [26] for these problems. The execution times are decreased considerably on the Exascalar system, using multiple PEZY accelerators per node and multiple nodes. The programs incorporate Goose and MPI, and run unchanged on different configurations.

Acknowledgments

The cluster used for the GPU computations in this paper was funded by the National Science Foundation under Award Number 1126438. We also acknowledge the Grant-in-Aid support for Scientific Research (17K05428) of JSPS. This work is furhter supported in part by the Large Scale Computational Sciences with Heterogeneous Many-Core Computers, Grant-in-Aid for High Performance Computing with General Purpose Computers from MEXT (Ministry of Education, Culture, Sports, Science and Technology-Japan).

References

[1] de Doncker E, Yuasa F, Kato K, Ishikawa T, Kapenga J and Olagbemi O 2018 *Computer Physics Communications* **224** 164–185 https://doi.org/10.1016/j.cpc.2017.11.001
[2] Kato K, de Doncker E, Ishikawa T, Kapenga J, Olagbemi O and Yuasa F 2016 *J. Physics: Conf. Series (JPCS), IOP Series* **762** 012070, iopscience.iop.org/article/10.1088/1742-6596/762/1/012070
[3] 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
[4] de Doncker E, Almulihi A and Yuasa F 2018 *The Journal of Physics: Conf. Series (JPCS), IOP Series* **1085** http://iopscience.iop.org/article/10.1088/1742-6596/1085/5/052005
[5] de Doncker E, Almulihi A and Yuasa F 2018 *The Journal of Physics: Conf. Series (JPCS), IOP Series* **1136** doi:10.1088/1742-6596/1136/1/012002
[6] de Doncker E, Yuasa F and Almulihi A 2019 Efficient GPU integration for multi-loop Feynman diagrams with massless internal lines To appear
[7] Cranley R and Patterson T N L 1976 *SIAM J. Numer. Anal.* **13** 904–914
[8] Li Z, Wang J, Yan Q S and Zhao X 2015 *Chinese Physics C* **40** doi:0.1088/1674-1137/40/3/033103
[9] Borowka S, Heinrich G, Jahn S, Jones S P, Kerner M and Schlenk J 2018 A GPU compatible quasi-Monte Carlo integrator interfaced to pySecDec arXiv:1811.11720v1 [hep-ph], https://arxiv.org/abs/1811.11720
[10] Laporta S 2000 *Int. J. Mod. Phys. A* **15** 5087–5159 arXiv:hep-ph/0102033v1
[11] Baikov B A and Chetyrkin K G 2010 *Nuclear Physics B* **837** 186–220
[12] Sloan I and Joe S 1994 *Lattice Methods for Multiple Integration* (Oxford University Press, Oxford)
[13] Korobov N M 1959 *Doklady Akademii Nauk SSSR* **124** 1207–1210 (Russian)
[14] Korobov N M 1960 *Doklady Akademii Nauk SSSR* **132** 1009–1012 (Russ.). Eng. trans. Soviet Math. Doklady, 1, 696-700
[15] Niederreiter H 1978 *Monatshefte für Mathematik* **86** 203–219
[16] Nuyens D and Cools R 2006 *Journal of Complexity* **22** 4–28
[17] Nuyens D and Cools R 2006 *Math. Comp.* **75** 903–920
[18] Sidi A 1993 *International Series of Numerical Mathematics* **112** 359–373
[19] Sidi A 2005 *Math. Comp.* **75** 327–343
[20] L' Equyer P and Munger D 2016 *ACM Trans. Math. Softw.* **42** 15:1–30 doi: http://dx.doi.org/10.1145/2754929
[21] Torii S and Ishikawa H 2017 ZettaScaler: Liquid immersion cooling manycore based supercomputer Tech. rep. ExaScaler Inc., PEZY Computing K. K.
[22] PEZY Computing/Exascaler Inc. http://www.exascaler.co.jp/
[23] Suiren2, High Energy Accelerator Research Organization /KEK https://www.top500.org/system/179164
[24] 2010 Goose software package user's guide - for goose version 1.3.3 k & F Computing Research Co. (in English)
[25] 2014 Goose - GRAPE9-MPx - for goose version 1.5.0 k & F Computing Research Co. (in Japanese)
[26] de Doncker E, Kaugars K, Cucos L and Zanny R 2001 Current status of the ParInt package for parallel multivariate integration *Proc. of Computational Particle Physics Symposium (CPP 2001)* pp 110–119