High-speed evaluation of loop integrals using lattice rules

E de Doncker[1], **A Almulihi**[1] **and F Yuasa**[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

E-mail:
`{elise.dedoncker, ahmedhassan.almulihi}@wmich.edu, fukuko.yuasa@kek.jp`

Abstract. Lattice rules are implemented in CUDA for many-core computations on GPUs. A high-speed evaluation of Feynman loop integrals is presented, based on lattice rules and suitable transformations. The accuracy and efficiency of the method are compared for higher order $\sin^m$-transformations. Extensive results are reported for classes of diagrams including 2-loop box and 3-loop self-energy diagrams with massive internal lines. The method is further combined with an extrapolation with respect to the dimensional regularization parameter.

1. Introduction

The goal of this work is to perform accurate loop integral computations, and to develop computer programs which evaluate multi-loop Feynman integrals in a fully numerical manner. Previously (e.g., in [1, 2, 3, 4]), we reported precise numerical results for 2-, 3- and 4-loop Feynman integrals using adaptive integration methods and extrapolation. In this paper we present a non-adaptive technique, applied to the calculation of 2- and 3-loop integrals, where the bulk of the computation, based on lattice rules with transformations by Sidi [5, 6], is executed on a GPU accelerator board. Lattice rules were used with Sidi's transformations in programs for 2- and 3-dimensional integration in (*r2d2lri* [7], *elrint3d* [8]). Lattice rules on GPU were recently applied to integrals resulting from sector decomposition [9].

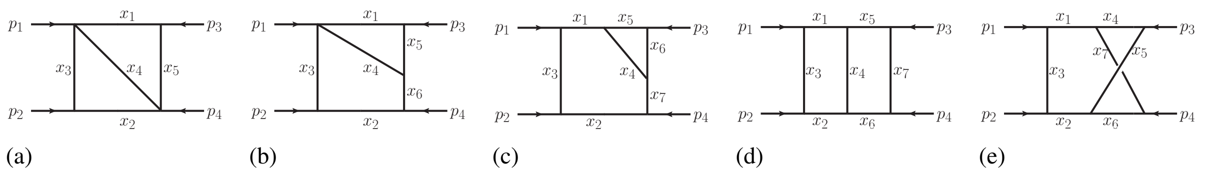
A Feynman integral with L loops and N internal lines is given by

$$\mathcal{F} = \frac{\Gamma\left(N - \frac{\nu L}{2}\right)}{(4\pi)^{\nu L/2}}(-1)^N \int_0^1 \prod_{r=1}^{N} dx_r\, \delta(1 - \sum x_r) \frac{C^{N-\nu(L+1)/2}}{(D - i\varrho C)^{N-\nu L/2}}, \tag{1}$$

where C and D are polynomials (arising from determinants) depending on the topology of the Feynman diagram and physical parameters, and $\nu = 4$ is the space-time dimension. However, in case the integral diverges with $\nu = 4$, the space-time dimension can be considered as a function of the dimensional regularization parameter, $\nu = \nu(\varepsilon)$, for a regularization where $\mathcal{F}$ is expanded as a series in ε. Ultra-violet (UV) and infra-red (IR) singularities give rise to integrand singularities on the boundaries of the domain. We let $\nu(\varepsilon) = 4 - 2\varepsilon$ or $\nu(\varepsilon) = 4 + 2\varepsilon$ for a UV or IR singularity, respectively. For a finite integral computation, the expansion can also serve at the basis of a convergence acceleration or extrapolation to the limit as $\varepsilon \to 0$. Furthermore, we set $\varrho = 0$ in Eq (1) unless D vanishes in the domain.

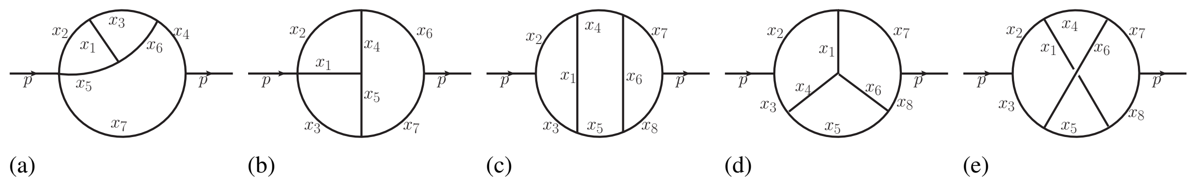
By eliminating the δ-function in Eq (1), the integral is re-written as $\mathcal{F} = (4\pi)^{-\nu L/2}\mathcal{F}_{L,N}$ with

$$\mathcal{F}_{L,N} = \Gamma(N - \frac{\nu L}{2})(-1)^N \int_{\mathcal{S}_{N-1}} \frac{C^{N-\nu(L+1)/2}}{(D - i\varrho C)^{N-\nu L/2}}\, d\mathbf{x}, \tag{2}$$



p_1 x_1 p_3 x_3 x_4 x_5 p_2 x_2 p_4 (a)
p_1 x_1 p_3 x_5 x_3 x_4 x_6 p_2 x_2 p_4 (b)
p_1 x_1 x_5 p_3 x_6 x_3 x_4 x_7 p_2 x_2 p_4 (c)
p_1 x_1 x_5 p_3 x_3 x_4 x_7 p_2 x_2 x_6 p_4 (d)
p_1 x_1 x_4 p_3 x_7 x_5 x_3 p_2 x_2 x_6 p_4 (e)

Figure 1. [2LB] Sample 2-loop box diagrams [10, 11]: (a) $N = 5$ (double-triangle), (b) $N = 6$ (tetragon-triangle), (c) $N = 7$ (pentagon-triangle), (d) $N = 7$ ladder, (e) $N = 7$ crossed box



x_2 x_3 x_4 x_1 x_6 p x_5 p x_7 (a)
x_2 x_6 x_4 x_1 p p x_5 x_3 x_7 (b)
x_2 x_4 x_7 x_1 x_6 p p x_3 x_5 x_8 (c)
x_2 x_7 x_1 p p x_4 x_6 x_3 x_8 x_5 (d)
x_2 x_4 x_7 x_1 x_6 p p x_3 x_8 x_5 (e)

Figure 2. [3LS] Sample 3-loop self-energy diagrams with massive internal lines, cf., Laporta [10]: (a) $N = 7$, (b) $N = 7$, (c) $N = 8$, (d) $N = 8$, (e) $N = 8$.

Table 1. Integrals $\mathcal{F}_{L,N}$

DIAG	L	N	$\mathcal{F}_{L,N}$	VALUE	DIAG	L	N	$\mathcal{F}_{L,N}$	VALUE
Fig 1(a)	2	5	$-\int_{\mathcal{S}_4} \frac{1}{C\,D}\, d\mathbf{x}$	0.9509235623171$^{(*)}$	Fig 2(a)	3	7	$-\int_{\mathcal{S}_6} \frac{1}{C\,D}\, d\mathbf{x}$	1.32644820827$^{(*)}$
Fig 1(b)	2	6	$\int_{\mathcal{S}_5} \frac{1}{D^2}\, d\mathbf{x}$	0.276209225359	Fig 2(b)	3	7	$-\int_{\mathcal{S}_6} \frac{1}{C\,D}\, d\mathbf{x}$	1.34139924145$^{(*)}$
Fig 1(c)	2	7	$-2\int_{\mathcal{S}_6} \frac{C}{D^3}\, d\mathbf{x}$	0.1723367907503$^{(*)}$	Fig 2(c)	3	8	$\int_{\mathcal{S}_7} \frac{1}{D^2}\, d\mathbf{x}$	0.2796089232826
Fig 1(d)	2	7	$-2\int_{\mathcal{S}_6} \frac{C}{D^3}\, d\mathbf{x}$	0.1036407209893$^{(*)}$	Fig 2(d)	3	8	$\int_{\mathcal{S}_7} \frac{1}{D^2}\, d\mathbf{x}$	0.1826272375392
Fig 1(e)	2	7	$-2\int_{\mathcal{S}_6} \frac{C}{D^3}\, d\mathbf{x}$	0.0853513981538$^{(*)}$	Fig 2(e)	3	8	$\int_{\mathcal{S}_7} \frac{1}{D^2}\, d\mathbf{x}$	0.14801330396

where the domain is the $d = (N-1)$-dimensional unit simplex, $\mathcal{S}_d = \{\mathbf{x} \in \mathcal{C}_\mathbf{d} \mid 0 \le \sum_{k=1}^{d} x_k \le 1\}$, and $\mathcal{C}_d = [0, 1]^d$ is the unit cube in $\mathbb{R}^d$. We will further transform $\mathcal{F}_{L,N}$ to an integral over $\mathcal{C}_d$.

For $\nu = 4$ and $\rho = 0$, the integrals $\mathcal{F}_{L,N}$ corresponding to the diagrams in Fig 1 and 2 are listed in Table 1. The numerical values are from [10] – except for the crossed box integral value, that resulted from an extensive computation with large numbers of integrand evaluations using the adaptive ParInt code in long double precision [3]. The functions C and D, given in Table A1 of Appendix A, are derived according to [12]. The cases indicated by $^{(*)}$ in Table 1 are taken with positive sign in subsequent test results. The computations assume unit masses and $p^2 = 1$.

2. Rank-1 lattice rules

A rank-1 lattice rule approximation of an integral $\mathcal{I}f = \int_{\mathcal{C}_d} f(\mathbf{x})\, d\mathbf{x}$ is represented by

$$\mathcal{Q}f = \frac{1}{n}\sum_{j=0}^{n-1} f(\{\frac{j}{n}\mathbf{z}\}) \tag{3}$$

where $\mathbf{z}$ is a generator vector with integer components $z \in \mathcal{Z}_n = \{1 \le z < n,\ gcd(z,n) = 1\}$. Here $\{\mathbf{x}\}$ denotes the vector in the half open cube $\mathcal{U}_d = [0,1)^d$ obtained by taking the fractional part of each component of $\mathbf{x}$. Classically n is prime (i.e., $\mathcal{Z}_n = \{1, 2, \ldots, n-1\}$); this is relaxed in [13, 14]. The generators used for the results in this paper are computed with the Fast CBC (component-by-component) algorithm by Nuyens and Cools [14, 15], which runs in $\mathcal{O}(d\,n\ \log n)$ time and $\mathcal{O}(n)$ space, and improves on the time of the CBC algorithm as introduced by Sloan and Reztsov [16].

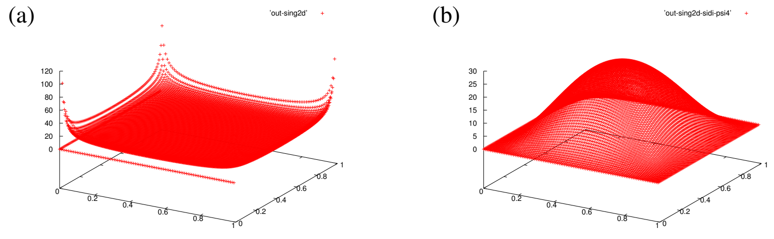


Figure 3. Ψ_4 transformation of $1/\sqrt{x_1 x_2 (1-x_1)(1-x_2)}$ over unit square: (a) without; (b) with transformation

Based on the theory of periodic functions, good lattice points (or $\mathbf{z}$ with optimal coefficients) as defined by Korobov [17, 18], and error bounds for lattice rule integration have been studied extensively in the literature (see, e.g., [13, 19, 20]), which further motivates the use of periodizing transformations.

3. Transformations

Consider the one-dimensional integral $If = \int_0^1 f(x)\,dx$ and a transformation $x = \varphi(t)$ such that $\varphi(0) = 0$, $\varphi(1) = 1$, and $\varphi'(t)$ has a number of derivatives that vanish at 0 and 1. Thus the transformed integrand in $If = \int_0^1 f(\varphi(t))\,\varphi'(t)\,dx$ can be extended periodically with period 1. Periodizing transformations include the transformations by Korobov [21], versions of the $\tanh$-transformation [22], the IMT [23, 24] and double exponential transformations [25]. For the $\tanh$ and double exponential type transformations one has to exercise considerable caution in their implementation, to avoid overflows and other run-time exceptions resulting from integrand evaluations near the boundaries.

In this paper we will rely on a class of transformations by Sidi [5, 6], $\varphi(t) = \Psi_m(t)$ given by

$$\Psi_m(t) = \frac{\theta_m(t)}{\theta_m(1)}, \qquad m = 1, 2, \ldots \tag{4}$$

with $\theta_m(t) = \int_0^t \sin^m(\pi u)\,du$. The transformations used for the work in this paper are listed below.

$\Psi_2(t) = t - \sin(2\pi t)/(2\pi), \quad \Psi_2'(t) = 2\sin^2(\pi t)$
$\Psi_4(t) = t + (-8\sin(2\pi t) + \sin(4\pi t))/(12\pi), \quad \Psi_4'(t) = \frac{8}{3}\sin^4(\pi t)$
$\Psi_6(t) = t - (45\sin(2\pi t) - 9\sin(4\pi t) + \sin(6\pi t))/(60\pi), \quad \Psi_6'(t) = \frac{16}{5}\sin^6(\pi t)$
$\Psi_8(t) = t + (-672\sin(2\pi t) + 168\sin(4\pi t) - 32\sin(6\pi t) + 3\sin(8\pi t))/(840\pi), \quad \Psi_8'(t) = \frac{128}{35}\sin^8(\pi t)$

Properties of $\Psi_m(t)$ are shown in [5, 6], also with respect to functions with end-point singularities. We will apply the transformations for periodization and also with the goal of dealing with integrand boundary singularities. Fig 3 illustrates the smoothening abilities of $\Psi_4(t)$, where the function $1/\sqrt{x_1 x_2 (1-x_1)(1-x_2)}$ is displayed before and after the transformation in both coordinate directions over the unit square. The jacobian "knocks out" the singular behavior in the transformed integrand.

4. Results

4.1. Lattice rule integration on GPU

The lattice rule integrations are implemented in CUDA C and executed on the *thor* cluster (of the Center for High Performance Computing and Big Data at WMU), where 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-20) GPU. For the CUDA program we used 64 blocks and 1024 threads per block on the GPU. The lattice generator vector is precomputed and is the only data communicated from the main program to the CUDA kernel, as a one-dimensional array of length d. The kernel sums the function values in Eq (3) over each block using the synchronization pattern for a reduction (see, e.g., [26]). A sample implementation of the integrand as a CUDA device function is given in Appendix C for the integral of Fig 1(b) in Table 1.

Table 2. Results for 2-loop box diagrams using no transformation or Ψ_2, Ψ_4, Ψ_6, and $(N-1)$-dim. lattice rule with 100M points. Listed are abs. accuracy (E_a) and execution times T in milliseconds (ms).

DIAG.		NO TRANS.		Ψ_2		Ψ_4		Ψ_6	
	N	E_a	T[ms]	E_a	T[ms]	E_a	T[ms]	E_a	T[ms]
2LB (a)	5	4.2 e-04	59	1.6 e-08	124	4.9 e-12	178	2.9 e-14	208
2LB (b)	6	1.6 e-04	58	3.0 e-07	154	3.6 e-10	224	6.2 e-11	258
2LB (c)	7	4.4 e-04	76	4.5 e-07	181	1.2 e-09	266	2.3 e-09	307
2LB (d)	7	4.0 e-05	76	3.7 e-11	181	6.6 e-11	266	1.1 e-11	310
2LB (e)	7	1.6 e-05	76	6.0 e-11	183	1.3 e-13	267	2.6 e-13	312

Table 3. Results for 3-loop self-energy diagrams using no transformation or Ψ_2, Ψ_4, Ψ_6, and $(N-1)$-dim. lattice rule with 100M points. Listed are abs. accuracy (E_a) and execution times T in milliseconds (ms).

DIAG.		NO TRANS.		Ψ_2		Ψ_4		Ψ_6	
	N	E_a	T[ms]	E_a	T[ms]	E_a	T[ms]	E_a	T[ms]
3LS (a)	7	5.7 e-02	77	3.4 e-04	184	1.0 e-05	267	2.5 e-07	309
3LS (b)	7	1.7 e-02	77	6.0 e-05	182	9.4 e-06	266	2.7 e-06	312
3LS (c)	8	2.8 e-03	87	1.2 e-05	214	2.4 e-06	316	5.3 e-07	366
3LS (d)	8	3.9 e-03	87	6.8 e-06	216	3.6 e-06	315	2.3 e-06	366
3LS (e)	8	1.7 e-03	86	2.6 e-06	218	1.2 e-07	315	6.9 e-08	367

Table 2 compares the accuracy and computation time for the 2-loop box diagrams of Fig 1(a-e), using a rank-1 lattice rule with 100,000,007 points generated for dimension $d = N - 1$, and Sidi's Ψ_2, Ψ_4, Ψ_6 transformations applied to the integrals; these are also compared to results obtained without a transformation (apart from mapping the unit simplex to the cube). The absolute accuracy E_a and the time in milliseconds are listed for each integration.

Similarly, Table 3 treats the 3-loop self-energy diagrams of Fig 2(a-e). These result in a more complicated structure and higher degree of the C and D polynomials, as well as difficult singularities on boundaries of the integration domain (cf., Table A1 in Appendix A). For example, D of 2(c) vanishes at $x_1 = x_2 = x_3 = 0$, and at $x_6 = x_7 = x_8 = 0$ (where $x_8 = 0$ corresponds to $\sum_{k=1}^{7} x_k = 1$ on $\mathcal{S}_7$). Results computed with Ψ_8 did not improve (or improved only slightly) on the results listed through Ψ_6. However, the accuracy improved significantly by the higher order transformations through Ψ_6.

Table 4 gives comparisons for lattice rules (LR) with 10M and 100M points, and adaptive integration (AD) with 100M points, using transformation Ψ_4 for the 2-loop box diagrams (2LB, Left), and Ψ_6 for the 3-loop self-energy diagrams (3LS, Right). The lattice rule with 100M points is clearly more accurate than the 10M rule, except for 3LS(d) where the accuracy is only marginally better (2.3 e-06 vs. 3.6 e-06).

For the sake of comparison, Table 4 also lists adaptive integration results with ParInt and 100M integrand evaluations, using 16 MPI processes on one 16-core node. The lattice integration method (LR) reaches considerably better accuracy with 100M evaluations and needs far less computation time than the adaptive integration (AD).

We further note that the (LR) GPU implementation achieves an excellent speedup (= the ratio of sequential to parallel time), which is around 456 for the crossed box integral 2LB (e) with 100M evaluations using the Ψ_4 transformation. The sequential time (around 122 seconds) was measured with *gettimeofday*, the parallel time (around 267.3 ms) with *cudaEventElapsedTime*.

4.2. Linear extrapolation

Extrapolation is tailored to an underlying asymptotic expansion, such as the expansions in [10] for loop integrals of the form $\mathcal{F}_{L,N}$ in Eq (2), with respect to the dimensional regularization parameter ε.

Table 4. Result comparisons for lattice rules (LR) with 10M and 100M points, and adaptive integration (AD) with 100M points, using transf. Ψ_4 for 2-loop box diagrams (2LB, Left), and tranf. Ψ_6 for 3-loop self-energy diagrams (3LS, Right). Listed are abs. accuracy (E_a) and execution times T in milliseconds (ms).

DIAG.	Ψ_4						DIAG.	Ψ_6					
2LB	LR 10M		LR 100M		Ad 100M		3LS	LR 10M		LR 100M		Ad 100M	
	E_a	T[ms]	E_a	T[ms]	E_a	T[ms]		E_a	T[ms]	E_a	T[ms]	E_a	T[ms]
2LB (a) :	5.7 e-09	18	4.9 e-12	178	2.2 e-11	6495	3LS (a) :	3.2 e-06	31	2.5 e-07	309	7.2 e-06	11782
2LB (b) :	3.5 e-08	23	3.6 e-10	224	2.4 e-08	7971	3LS (b) :	1.0 e-05	32	2.7 e-06	312	5.1 e-05	11757
2LB (c) :	9.9 e-07	27	1.2 e-09	266	3.2 e-06	9765	3LS (c) :	8.7 e-05	37	5.3 e-07	366	6.3 e-06	13497
2LB (d) :	3.5 e-08	27	6.6 e-11	266	2.3 e-07	9721	3LS (d) :	3.6 e-06	37	2.3 e-06	366	5.0 e-05	11707
2LB (e) :	3.2 e-09	27	1.3 e-13	267	1.9 e-07	9730	3LS (e) :	2.1 e-05	37	6.9 e-08	366	7.7 e-06	14000

Table 5. Results of linear extrapolation for loop integral of Fig 1(a), $\varepsilon_\ell = 2^{-\ell}$

ℓ	K	$I(\varepsilon)$	C_0	C_1	C_2	C_3
10	0	0.94969742599225604				
11	1	0.95030985357508291	0.9509222811579	-1.254251689629		
12	2	0.95061654757003522	0.9509235617007	-1.258185517020	2.685492832	
13	3	0.95077001482145529	0.9509235623195	-1.258189952916	2.694577546	-5.31585584
13	4	0.95084677853608013	0.9509235623197	-1.258189955234	2.694588624	-5.33530208
		Exact [10]:	0.9509235623171	-1.258189955235	2.694588643	-5.33534712

In general a linear extrapolation is performed for an underlying expansion

$$I(\varepsilon) \sim C_0\varphi_0(\varepsilon) + C_1\varphi_1(\varepsilon) + C_2\varphi_2(\varepsilon) + \ldots, \quad \text{as } \varepsilon \to 0$$

by creating an approximating sequence of $I(\varepsilon_\ell) \approx I(\varepsilon)$ such that

$$I(\varepsilon_\ell) \approx C_0\varphi_0(\varepsilon_\ell) + C_1\varphi_1(\varepsilon_\ell) + \ldots C_K\varphi_K(\varepsilon_\ell), \quad \ell = 1, \ldots, K+1,$$

and solving the resulting linear systems of orders $(K+1) \times (K+1)$, for increasing values of K and decreasing $\varepsilon = \varepsilon_\ell$. As an example, Table 5 shows extrapolated results for the leading term coefficients in the expansion of the 2-loop box integral of Fig 1(a). The functions $\varphi_k(\varepsilon) = \varepsilon^k,\ k \geq 0$, and $\varepsilon_\ell = 2^{-\ell}$.

5. Conclusions

We applied lattice rule integration to classes of 2-loop box and 3-loop self-energy Feynman diagrams, after a $\sin^m$-transformation, which was performed not only as a periodizing transformation but also to deal with integrand boundary singularities. We examined the effects of higher order transformations. The implementation in CUDA C was executed utilizing one Kepler 20 GPU for the integrand evaluations and summations over the blocks. The results were found far more accurate and efficient than parallel adaptive integration layered over MPI and using 16 CPU cores on one cluster node. The GPU lattice integration is promising for interaction cross section computations where thousands of integrals are needed. In future work it can be combined with other techniques, such as a suitable pre-partitioning of the domain.

Acknowledgments
We acknowledge the support from the National Science Foundation under Award Number 1126438, which funded the equipment used for the computations in this paper. This work is also funded by Grant-in-Aid for Scientific Research (17K05428) of JSPS, and the Large Scale Simulation Program No.16/17-21 of KEK. We further thank Dr. A. Sidi and Dr. S. Jones for pointing us to related papers.

Table A1. C and D functions in the integrands of Eq (2) and Table 1, for the diagrams in Fig 1 and 2

FIG	
1(a)	$C = x_{15}x_{234} + x_4x_{23}$
	$D = x_3^2x_4 + x_3x_4^2 + x_1^2x_{234} + x_3^2x_5 + 2x_3x_4x_5 + x_4^2x_5 + x_3x_5^2 + x_4x_5^2 - x_2^2x_{45}$
	$+x_2(x_4^2 + 2x_4x_5 + x_5^2 + x_3x_{45}) + x_1(x_2^2 + x_3^2 + x_4x_{45} + x_3(2x_4 + x_5) + x_2(x_{345} + x_4))$
1(b)	$D = (x_1^2 + x_5^2 + x_1x_5)x_{2346} + (x_2^2 + x_3^2 + x_6^2 + x_2x_3 + x_2x_6 + x_3x_6)x_{145} + x_4^2x_{12356}$
	$+3x_4x_5x_6 + 2x_1x_4x_{236} + 2x_4x_5x_{23}$
1(c)	$C = x_{1237}x_{456} + x_4x_{56}$
	$D = (x_1^2 + x_2^2 + x_3^2 + x_7^2 + x_1x_2 + x_{12}x_{37} + x_3x_7)x_{456} + x_4^2\, x_{123567}$
	$+(x_5^2 + x_6^2 + x_5x_6)x_{12347} + 3x_4(x_1x_5 + x_6x_7) + 2(x_{123}x_4x_6 + x_{237}x_4x_5)$
1(d)	$C = x_{1234}x_{4567} - x_4^2$
	$D = CM - x_2x_4x_5 - x_1x_4x_6 - x_{1234}x_5x_6 - x_3x_4x_7 - (x_{234}x_5 + x_1x_{45})x_7$
	$-(x_{134}x_6 + x_2x_{46})x_7 - x_1x_2x_{4567} - x_3(x_4x_5 + x_1x_{4567}) - x_3(x_4x_6 + x_2x_{4567})$
1(e)	$C = x_{12345}x_{12367} - x_{123}^2$
	$D = CM - x_2x_3x_4 - x_2x_3x_5 - x_2x_4x_5 - x_3x_4x_5 - x_2x_3x_6 - x_2x_4x_6 - 2x_3x_4x_6 - x_3x_5x_6$
	$-x_4x_5x_6 - x_2x_3x_7 - x_2x_4x_7 - x_3x_4x_7 - x_2x_5x_7 - x_3x_5x_7 - x_4x_5x_7 - x_2x_6x_7 - x_3x_6x_7$
	$-x_4x_6x_7 - x_5x_6x_7 - x_1(x_4x_5 + x_4x_6 + x_5x_6 + x_5x_7 + x_6x_7 + x_2x_{4567} + x_3x_{4567})$
2(a)	$C = -x_1^2x_{4567} + x_{125}x_{136}x_{4567} - x_{136}x_5^2 - 2x_1x_5x_6 - x_{125}x_6^2$
	$D = -x_1^2x_4^2 + x_{125}x_{136}x_4^2 - x_1^2x_{1567}x_{4567} + x_{125}x_{136}x_{1567}x_{4567} + x_{136}x_2^2x_{4567}$
	$+2x_1x_2x_3x_{4567} + x_{125}x_{4567}x_3^2 + 2x_{136}x_2x_4x_5 + 2x_1x_3x_4x_5 - x_{136}x_{1567}x_5^2 - x_3^2x_5^2$
	$+2x_1x_2x_4x_6 + 2x_{125}x_3x_4x_6 - 2x_1x_{1567}x_5x_6 + 2x_2x_3x_5x_6 - x_{125}x_{1567}x_6^2 - x_2^2x_6^2$
2(b)	$C = -x_{135}x_4^2 - x_1^2x_{4567} + x_{124}x_{135}x_{4567} - 2x_1x_4x_5 - x_{124}x_5^2$
	$D = -x_{13457}x_{135}x_4^2 - x_1^2x_{13457}x_{4567} + x_{124}x_{13457}x_{135}x_{4567} + x_{135}x_{4567}x_2^2 - 2x_1x_{13457}x_4x_5$
	$-x_{124}x_{13457}x_5^2 - x_2^2x_5^2 + 2x_{135}x_2x_4x_6 + 2x_1x_2x_5x_6 - x_1^2x_6^2 + x_{124}x_{135}x_6^2$
2(c)	$D = -x_{123}x_{13568}x_6^2 - x_2^2x_6^2 - x_1^2x_{13568}x_{678} + x_{123}x_{13568}x_{1456}x_{678} + x_{1456}x_2^2x_{678}$
	$+2x_1x_2x_4x_{678} + x_{123}x_4^2x_{678} + 2x_1x_2x_6x_7 + 2x_{123}x_4x_6x_7 - x_1^2x_7^2 + x_{123}x_{1456}x_7^2$
2(d)	$D = -x_{134568}x_{1678}x_4^2 - x_1^2x_{134568}x_{456} + x_{1234}x_{456}x_{134568}x_{1678} + x_{1678}x_2^2x_{456} - 2x_1x_{134568}x_4x_6$
	$-x_{1234}x_{134568}x_6^2 - x_2^2x_6^2 + 2x_1x_2x_{456}x_7 + 2x_2x_4x_6x_7 - x_4^2x_7^2 + x_{1234}x_{456}x_7^2$
2(e)	$D = -x_{13568}x_{2346}x_5^2 - x_{24}^2x_5^2 - x_{13568}x_{23}^2x_{5678} + x_{1235}x_{13568}x_{2346}x_{5678} + x_2^2x_{2346}x_{5678}$
	$-2x_2x_{23}x_{24}x_{5678} + x_{1235}x_{24}^2x_{5678} - 2x_{13568}x_{23}x_5x_6 - 2x_2x_{24}x_5x_6 - x_{1235}x_{13568}x_6^2 - x_2^2x_6^2$
	$-2x_2x_{2346}x_5x_7 + 2x_{23}x_{24}x_5x_7 - 2x_2x_{23}x_6x_7 + 2x_{1235}x_{24}x_6x_7 - x_{23}^2x_7^2 + x_{1235}x_{2346}x_7^2$

Table B1. Lattice rule generators

d	$\mathbf{z}$ components for $n = 10M$	$\mathbf{z}$ components for $n = 100M$
4	$(1, 2928962, 1859617, 3250721)$	$(1, 38278307, 43388112, 5988368)$
5	$(1, 3675449, 4456704, 3864450, 4934306)$	$(1, 38278307, 43388112, 48206750, 45950883)$
6	$(1, 2928962, 1859617, 4304839, 2689131, 2592686)$	$(1, 38278307, 34519177, 26711202, 33749025, 1281880)$
7	$(1, 2928962, 1859617, 4304839, 4474033, 273363, 1426318)$	$(1, 41883906, 22682973, 44229424, 29466837, 15518263, 42112409)$

Appendix A. C and D functions

The C and D functions of the loop integrals corresponding to the diagrams in Fig 1 and 2 are given in Table A1. We use the notation $x_{i_1i_2\ldots i_k} = x_{i_1} + x_{i_2} + \ldots + x_{i_k}$ for $1 \le i_k \le N$, and $M = \sum_r m_r^2 x_r$.

Appendix B. Lattice rule generators

Table B1 lists the components of the generator vector $\mathbf{z}$ in Eq (3) computed for dimensions $d = 4, 5, 6, 7$, with $n = 10,000,019$ and $n = 100,000,007$ points.

Appendix C.   Sample integrand function

The program section below is a sample implementation of the integrand of Fig 1(b) as a CUDA device function. The integration is over the 5-dimensional unit cube obtained by a transformation from the unit simplex $\mathcal{S}_5$. Note that $\texttt{pi12} = 12\pi$ and $\texttt{con} = (\frac{8}{3})^5$ (from the Jacobian of the Ψ_4 transformation according to Eq (4)).

```
#include <math.h>

#define pi 3.14159265358979323846
#define pi12 37.69911184307751886152
#define con 134.84773662551440329049

const int dim = 5;

__device__ double f1(double x[]) { //Integrand function for
                                   //Fig 1(b)
  double x1,x2,x3,x4,x5,x6;
  double arg[5],dd,f0,phip,pixi;
  int i;

  f0 = 0;
  phip = 1.0;
  for(i = 0; i < dim; i++) { //Sidi Psi_4 transformation
     pixi = pi*x[i];
     arg[i] = x[i]+(-8.0*sin(2*pixi)+sin(4*pixi))/pi12;
     phip = phip*pow(sin(pixi),4);
  }
  if(phip == 0) return(0);

  x1 = arg[0];               //Simplex to cube transformation
  x2 = (1.0-x1)*arg[1];
  x3 = (1.0-x1-x2)*arg[2];
  x4 = (1.0-x1-x2-x3)*arg[3];
  x5 = (1.0-x1-x2-x3-x4)*arg[4];
  x6 = 1.0-x1-x2-x3-x4-x5;

  phip = phip*(1.0-x1)*(1.0-x1-x2)*(1.0-x1-x2-x3)*(1.0-x1-x2-x3-x4);
  if(phip == 0) return(0);

  dd = (x1*x1+x5*x5+x1*x5)*(x2+x3+x4+x6)
       +(x2*x2+x3*x3+x6*x6+x2*x3+x2*x6+x3*x6)*(x1+x4+x5)
       +x4*x4*(x1+x2+x3+x5+x6)+3.0*x4*x5*x6+
       2.0*x1*x4*(x2+x3+x6)+2.0*x4*x5*(x2+x3);

  if(fabs(dd) > 0.0) f0 = con*phip/dd/dd;
  return f0;
}
```

References

[1] 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

[2] Kato K, de Doncker E, Ishikawa T, Kapenga J, Olagbemi O and Yuasa F 2016 *J. Physics: Conf. Ser.* **762** 012070, iopscience.iop.org/article/10.1088/1742-6596/762/1/012070

[3] de Doncker E, Yuasa F, Kato K, Ishikawa T, Kapenga J and Olagbemi O 2017 *J. Computer Phys. Communications* doi:10.1016/j.cpc.2017.11.001; Preprint at arXiv:[hep-ph] https://arxiv.org/abs/1702.04904

[4] de Doncker E and Yuasa F 2017 *Procedia Computer Science* **108** 1773–1782 https://doi.org/10.1016/j.procs.2017.05.253

[5] Sidi A 1993 *International Series of Numerical Mathematics* **112** 359–373

[6] Sidi A 2005 *Math. Comp.* **75** 327–343

[7] Robinson I and Hill M 2002 *Transactions on Mathematical Software* **28** 75–100 doi:10.1145/513001.513006
[8] Li T and Robinson 2010 *Transactions on Mathematical Software* **37** doi:10.1145/1824801.1824813
[9] Li Z, Wang J, Yan Q S and Zhao X 2016 *Chinese Physics C* **40** 033103 doi:10.1088/1674-1137/40/3/033103; Preprint at arXiv:1508.02512v1 [hep-ph]
[10] Laporta S 2000 *Int. J. Mod. Phys. A* **15** 5087–5159 arXiv:hep-ph/0102033v1
[11] de Doncker E and Yuasa F 2014 *Journal of Physics: Conf. Series (ACAT 2013)* **523** doi:10.1088/1742-6596/523/1/012052
[12] Yuasa F, de Doncker E, Hamaguchi N, Ishikawa T, Kato K, Kurihara Y and Shimizu Y 2012 *Journal Computer Physics Communications* **183** 2136–2144
[13] Niederreiter H 1978 *Monatshefte für Mathematik* **86** 203–219
[14] Nuyens D and Cools R 2006 *Journal of Complexity* **22** 4–28
[15] Nuyens D and Cools R 2006 *Math. Comp.* **75** 903–920
[16] Sloan I H and Reztsov A 2002 *Math. Comp.* **71** 263–273
[17] Korobov N M 1959 *Doklady Akademii Nauk SSSR* **124** 1207–1210 (Russian)
[18] Korobov N M 1960 *Doklady Akademii Nauk SSSR* **132** 1009–1012 (Russ.). Eng. trans. Soviet Math. Doklady, 1, 696-700
[19] Disney S A R and Sloan I H 1991 *Math. Comp.* **56** 257–266
[20] Sloan I and Joe S 1994 *Lattice Methods for Multiple Integration* (Oxford University Press, Oxford)
[21] Korobov N M 1963 *Number-Theoretic Methods in Approximate Analysis* (Fizmatgiz, Moscow (Russian))
[22] Sag T W and Szekeres G 1964 *Math. Comp.* **18** 245–253
[23] Iri M, Moriguti S and Takasawa Y 1970 *Kokyuroku of Res. Inst. for Math. Sci. Kyoto Univ.* **91** 82–118 (in Japanese); Engl. transl. in J. Comp. Appl. Math. 17 (1987), pp. 3-20
[24] Mori M 1978 *Publ. RIMS Kyoto Univ.* **14** 713–729
[25] Takahasi H and Mori M 1974 *Publications of the Research Institute for Mathematical Sciences* **9** 721–741
[26] Sanders J and Kandrot E 2010 *CUDA by Example: An Introduction to General-Purpose GPU Programming* (Pearson)