Calculating the five-loop QED contribution to the electron anomalous magnetic moment: Graphs without lepton loops

Sergey Volkov*

SINP MSU, Moscow 119234, Russia and DLNP JINR, Dubna 141980, Russia

(Received 19 September 2019; published 8 November 2019)

This paper describes a computation of a part of the QED contribution to the electron anomalous magnetic moment that was performed by the author with the help of a supercomputer. The computed part includes all 5-loop QED Feynman graphs without lepton loops. The calculation has led to the result $A_1^{(10)}[\text{no lepton loops}] = 6.793(90)$ that is slightly different than the value 7.668(159) presented by T. Aoyama, T. Kinoshita, and M. Nio in 2018. The discrepancy is about 4.8σ. The computation gives the first independent check for that value. A shift in the fine-structure constant prediction is revealed in the paper. The developed calculation method is based on (a) a subtraction procedure for removing all ultraviolet and infrared divergences in Feynman parametric space before integration; (b) a nonadaptive Monte Carlo integration that uses the probability density functions that are constructed for each Feynman graph individually using its combinatorial structure. The method is described briefly in the paper (with the corresponding references to the previous papers). The values for the contributions of nine gauge-invariant classes splitting the whole set are presented in the paper. Moreover, the whole set of all 5-loop graphs without lepton loops is split into 807 subsets for comparison (in the future) of the calculated values with the values obtained by other methods. These detailed results are presented in the supplemental materials. Also, the supplemental materials contain the contribution values for each of 3213 individual Feynman graphs. An "oscillating" nature of these values is discussed. A realization of the numerical integration on the graphics accelerator NVidia Tesla V100 (as a part of the supercomputer "Govorun" from JINR, Dubna) is described with technical details such as pseudorandom generators, calculation speed, code sizes and structure, prevention of round-off errors and overflows, etc.

DOI: 10.1103/PhysRevD.100.096004

I. INTRODUCTION

The most precise measurement of the electron anomalous magnetic moment (AMM) gave the result

$$a_e[\text{expt}] = 0.00115965218073(28). \tag{1}$$

This result was presented by Gabrielse research group at Harvard in Ref. [1]. All theoretical predictions for a_e must satisfy this "quality standard" for the precision. The "mainstream" Standard Model prediction uses the following expression:

$$a_e = a_e(\text{QED}) + a_e(\text{hadronic}) + a_e(\text{electroweak}),$$

$$a_e(\text{QED}) = \sum_{n\geq 1}\left(\frac{\alpha}{\pi}\right)^n a_e^{2n},$$

$$a_e^{2n} = A_1^{(2n)} + A_2^{(2n)}(m_e/m_\mu) + A_2^{(2n)}(m_e/m_\tau) + A_3^{(2n)}(m_e/m_\mu, m_e/m_\tau),$$

where m_e, m_μ, m_τ are the masses of the electron, muon and tau-lepton, respectively. The universal QED terms $A_1^{(2n)}(\alpha/\pi)^n$ form the most significant contribution to the value. The coefficient values

$$A_1^{(2)} = 0.5, \qquad A_1^{(4)} = -0.328478965579\ldots$$

were presented in Refs. [2,3] and Refs. [4,5], respectively. The value of $A_1^{(6)}$ was being calculated in 1970-x by different groups of scientists using numerical integration; see Refs. [6–9]. The most accurate value $A_1^{(6)} = 1.195 \pm 0.026$ for that era was obtained in 1974 by T. Kinoshita and P. Cvitanović. The uncertainty is caused by the statistical error of the Monte Carlo integration. A work of analytical calculation of $A_1^{(6)}$ with the help of computers was started at the same time. The final value

*volkoff_sergey@mail.ru, sergey.volkov.1811@gmail.com

$$A_1^{(6)} = 1.181241456\ldots$$

was obtained by S. Laporta and E. Remiddi in 1996; see Ref. [10]. That value was a product of efforts of many researchers; see, for example, Refs. [11–26]. First numerical estimations for $A_1^{(8)}$ were obtained by T. Kinoshita and W. B. Lindquist in 1981 and published in Ref. [27]. The most accurate value presented by T. Kinoshita's team

$$A_1^{(8)} = -1.91298(84)$$

was published in 2015 in Ref. [28]. That value was obtained by Monte Carlo integration. S. Laporta's semianalytical result

$$A_1^{(8)} = -1.9122457649\ldots$$

was obtained in 2017 and published in Ref. [29]. These two calculations of $A_1^{(8)}$ are in good agreement as well as other independent calculations of this value from Refs. [30,31], and for Feynman graphs without lepton loops from Ref. [32].

The full calculation of $A_1^{(10)}$ was performed only by T. Kinoshita's team using Monte Carlo integration. The most precise value was obtained in 2019 by T. Aoyama, T. Kinoshita, M. Nio and was published in Ref. [33]:

$$A_1^{(10)}[\mathrm{AKN}] = 6.737(159). \tag{2}$$

A special place is occupied by the contribution of Feynman graphs without lepton loops to $A_1^{(10)}$. This set contains 3213 Feynman graphs[1] and forms a gauge-invariant class. This contribution is the most complicated one for both Monte Carlo integration and analytical calculations. For example, the uncertainty in (2) is entirely determined by that contribution. Also, it is the contribution that suffered the most from found mistakes and corrections; see Ref. [34]. The value

$$A_1^{(10)}[\text{no lepton loops, AKN}] = 7.668(159) \tag{3}$$

can be obtained by using (2) and the value of the remaining part that can be extracted from Ref. [34]. By 2019, there were no independent calculations of $A_1^{(10)}$[no lepton loops].

We recalculated this contribution with the help of the supercomputer "Govorun" (JINR, Dubna, Russia). 40000 GPU-hours of Monte Carlo integration on NVidia Tesla V100 that were spread over several months have led to the result

$$A_1^{(10)}[\text{no lepton loops, Volkov}] = 6.793(90), \tag{4}$$

where the uncertainty corresponds to 1σ limits. It is in good agreement with the preliminary value 6.782(113) published in Ref. [35]. The discrepancy between this result and (3) is approximately 4.8σ. This means that the values are probably different. The reason of this difference is unknown. Section V contains some considerations about the reliability of the result. In addition, it is important that this result can be checked by parts; see the detailed explanation in Sec. V.

Combining (4) with the value of the residual part of $A_1^{(10)}$ from Ref. [34], we obtain

$$A_1^{(10)}[\mathrm{Volkov} + \mathrm{AKN}] = 5.862(90). \tag{5}$$

Taking the known and double-checked values for $A_2^{(2n)}$, $n \le 5$, $A_3^{(2n)}$, $n \le 4$, $a_e(\mathrm{hadronic}) + a_e(\mathrm{electroweak})$ (see a review in Ref. [33]) and the measured value of α from Ref. [36] based on a measurement of the cesium atom mass relative to the Planck constant

$$\alpha^{-1}(\mathrm{Cs}) = 137.035999046(27), \tag{6}$$

we obtain

$$a_e[\mathrm{theory}, \alpha(\mathrm{Cs}), \mathrm{Volkov}] = 0.001159652181547(6)(12)(229),$$

where the first uncertainty comes from (4), the second one from the hadronic and electroweak corrections, and the last one from the uncertainty of α. The usage of (2) will give

$$a_e[\mathrm{theory}, \alpha(\mathrm{Cs}), \mathrm{AKN}] = 0.001159652181606(11)(12)(229)$$

instead. If we will use the a_e prediction with (5) and the measured value (1) for improving α, we obtain

$$\alpha^{-1}[a_e, \mathrm{Volkov}] = 137.0359991427(7)(14)(331), \tag{7}$$

where the uncertainties come from (4), the hadronic and electroweak corrections, (1), correspondingly. The discrepancy with (6) is approximately 2.27σ. The corresponding value obtained from (2) is

$$\alpha^{-1}[a_e, \mathrm{AKN}] = 137.0359991496(13)(14)(330), \tag{8}$$

with the discrepancy 2.43σ relative to (6). If we take

$$\alpha^{-1}(\mathrm{Rb}) = 137.035998996(85) \tag{9}$$

obtained from the measurement of the rubidium atom mass relative to the Planck constant (Ref. [37]) combined with

[1]Graphs that are obtained from each other by changing arrow directions are regarded as one.

the improved values of some constants from CODATA-2014 (Ref. [38]), we obtain

$$a_e[\text{theory}, \alpha(\text{Rb}), \text{Volkov}] = 0.001159652181969(6)(12)(720),$$

$$a_e[\text{theory}, \alpha(\text{Rb}), \text{AKN}] = 0.001159652182037(11)(12)(720).$$

The values (7) and (8) have the discrepancies 1.61σ and 1.69σ relative to (9). This means that the discrepancy between (4) and (3) affects α and a_e slightly. However, this discrepancy can become significant in the future, when the precision of the measurements will be increased. Also, if both calculations have mistakes, then this can be sensible even at the current level of precision. Thus, an additional independent calculation is required.

There is no universal method that makes it possible to calculate 5-loop QED contributions in a realistic time frame. Firstly, the existing universal IR divergence control methods like those that are based on the dimensional regularization lead to enormous amounts of symbolic manipulations. And secondly, the universal integration routines demonstrate a very slow convergence on the obtained integrals.

To make the 5-loop calculations practically feasible it is required to remove all ultraviolet (UV) and infrared (IR) divergences before integration and to avoid any ε-like regularizations. All UV divergences in Feynman integrals can be removed by the direct subtraction on the mass shell using a forestlike formula like Zimmermann's forest formula.[2] However, an analogous method for removing IR divergences has not been invented yet. The anomalous magnetic moment is free from IR divergences: the IR divergences corresponding to soft virtual photons are compensated by the IR divergences connected with the on-shell renormalization; see notes in Ref. [42]. But, unfortunately, direct methods lead to an emergence of IR divergences in individual Feynman graphs. Different authors use different homemade divergence subtraction procedures that work in some cases; see Refs. [6,8,33,43]. A relatively simple subtraction procedure giving finite Feynman parametric integrals was developed for our calculations. It was presented firstly in Ref. [44] and is briefly described in Sec. II.

The 5-loop calculations lead to Feynman parametric integrals with 13 variables. At this time, the only way to evaluate such integrals numerically is to use Monte Carlo integration. Unfortunately, Feynman parametric integrands after divergence subtraction are unbounded and have a very complicated asymptotic behavior near boundaries. The universal adaptive Monte Carlo integration routines like VEGAS can, in principle, work with unbounded functions and functions having a steep landscape. However, these routines are suited for functions with a certain shape. This becomes critical for large numbers of variables. For example, VEGAS uses the probability density functions of the form

$$f_1(x_1) \cdot f_2(x_2) \cdot \ldots \cdot f_n(x_n)$$

and tries to fit the functions f_j to make the convergence as fast as possible.[3] Unfortunately, this approximation does not work fine for Feynman parametric integrals with large numbers of variables. A nonadaptive[4] method that uses some *a priori* knowledge about the Feynman parametric integrands behavior was developed for our calculations. The method that is briefly described in Sec. III works only for graphs without lepton loops. The first version of this method was presented in Ref. [42].

The developed Monte Carlo integration method allows us to reduce the needed number of samples substantially. However, in the 5-loop case, for evaluating 3213 Feynman graphs a supercomputer is still required. Modern graphics processors (GPUs) are more suitable for performing many uniform sequences of arithmetic operations in parallel than usual processors. The Monte Carlo integration was performed on GPUs NVidia Tesla V100 as a part of the supercomputer[5] "Govorun" from JINR (Dubna, Russia). The realization is described in Sec. IV with some programming details. Section V contains the results of the calculations, a discussion about these results, the description of the supplemental materials [45], and some technical information about the computation including the GPU performance, arithmetic precision statistics and so on.

II. DIVERGENCE ELIMINATION

The developed subtraction procedure is based on a forest formula with linear operators that are applied to the Feynman amplitudes of UV divergent subgraphs. This is similar to the Zimmermann forest formula. The difference is only in the choice of the linear operators used and in the way of combining them. Let us recapitulate the advantages of the developed procedure:

[2]The Zimmermann forest formula was first published in Ref. [39] and Ref. [40]. However, the historic name is connected with Ref. [41].

[3]The Monte Carlo integration error usually behaves as $\sigma \sim C/\sqrt{N}$, where N is the number of samples. However, it is very important to make C as small as possible.

[4]Except the intergraph adaptivity described in Sec. IV C and the adjustment of six constants (15) that was performed once for the 4-loop graphs.

[5]The GPU part of the supercomputer "Govorun" has 40 GPUs NVidia Tesla V100. The peak performance of the GPU part is 300 TFlops for double precision. The peak performance of the whole supercomputer (including the CPU part) is 500 TFlops.

(i) The procedure is fully automated for any order of the perturbation series.[6]
(ii) The method is beautiful and is relatively simple for realization on computers.
(iii) The subtraction is equivalent to the on-shell renormalization: for obtaining the final result we should only sum up the contributions of all Feynman graphs after subtraction. Thus, no residual renormalizations are required.
(iv) Feynman parameters can be used directly, without any additional tricks.

There are the following types of UV-divergent subgraphs[7] in QED Feynman graphs without lepton loops: *electron self-energy subgraphs* ($N_e = 2,\ N_\gamma = 0$) and *vertexlike* subgraphs ($N_e = 2,\ N_\gamma = 1$), where by N_e and N_γ we denote the number of external electron and photon lines in the subgraph.

Two subgraphs are said to overlap if they are not contained one inside the other, and the intersection of their sets of lines is not empty.

A set of subgraphs of a graph is called a *forest* if any two elements of this set do not overlap.

For a vertexlike graph G by $\mathfrak{F}[G]$ we denote the set of all forests F that consist of UV-divergent subgraphs of G and satisfy the condition $G \in F$. By $\mathfrak{T}[G]$ we denote the set of all vertexlike subgraphs G' of G such that G' contains the vertex that is incident[8] to the external photon line of G.[9]

We work in the system of units, in which $\hbar = c = 1$, the factors of 4π appear in the fine-structure constant: $\alpha = e^2/(4\pi)$, the tensor $g_{\mu\nu}$ is defined by

$$g_{\mu\nu} = g^{\mu\nu} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

the Dirac gamma-matrices satisfy the condition $\gamma^\mu\gamma^\nu + \gamma^\nu\gamma^\mu = 2g^{\mu\nu}$.

The following linear operators are used for the subtraction:

(1) A is the projector of the AMM. This operator is applied to the Feynman amplitudes of vertexlike subgraphs. See the definition in Refs. [42,44].
(2) The definition of the operator U depends on the type of UV-divergent subgraph to which the operator is applied:
   (a) If $\Sigma(p)$ is the Feynman amplitude that corresponds to an electron self-energy subgraph,
   $$\Sigma(p) = u(p^2) + v(p^2)\hat{p},$$
   then, by definition,[10]
   $$U\Sigma(p) = u(m^2) + v(m^2)\hat{p},$$
   where m is the mass of the electron, $\hat{p} = p^\mu\gamma_\mu$.
   (b) If $\Gamma_\mu(p,q)$ is the Feynman amplitude corresponding to a vertexlike subgraph,
   $$\Gamma_\mu(p,0) = a(p^2)\gamma_\mu + b(p^2)p_\mu + c(p^2)\hat{p}p_\mu + d(p^2)(\hat{p}\gamma_\mu - \gamma_\mu\hat{p}), \tag{10}$$
   then, by definition,
   $$U\Gamma_\mu = a(m^2)\gamma_\mu.$$
(3) L is the operator that is used in the standard subtractive on-shell renormalization of vertexlike subgraphs. If $\Gamma_\mu(p,q)$ is the Feynman amplitude that corresponds to a vertexlike subgraph, (10) is satisfied, then, by definition,
$$L\Gamma_\mu = [a(m^2) + mb(m^2) + m^2c(m^2)]\gamma_\mu.$$

Let f_G be the unrenormalized Feynman amplitude that corresponds to a vertexlike graph G. Let us write the symbolic definition

$$\tilde{f}_G = \mathcal{R}_G^{\boldsymbol{new}} f_G,$$

where

$$\mathcal{R}_G^{\boldsymbol{new}} = \sum_{\substack{F=\{G_1,\ldots,G_n\}\in\mathfrak{F}[G] \\ G'\in\mathfrak{T}[G]\cap F}} (-1)^{n-1} M_{G_1}^{G'} M_{G_2}^{G'} \ldots M_{G_n}^{G'},$$

$$M_{G''}^{G'} = \begin{cases} A_{G'}, & \text{if } G' = G'', \\ U_{G''}, & \text{if } G'' \notin \mathfrak{T}[G], \text{ or } G'' \not\subseteq G', \\ L_{G''}, & \text{if } G'' \in \mathfrak{T}[G], G' \not\subseteq G'', G'' \neq G, \\ (L_{G''} - U_{G''}), & \text{if } G'' = G, G' \neq G. \end{cases}$$

In this notation, the subscript of an operator symbol denotes the subgraph to which this operator is applied.

[6]The method must work for all Feynman graphs contributing to $A_1^{(2n)}$ including the ones containing lepton loops; see Ref. [44]. However, a rigorous mathematical proof for this fact is not developed even for graphs without lepton loops.

[7]We consider only such subgraphs that are strongly connected and contain all lines that join the vertexes of the given subgraph.

[8]We say that a line l and a vertex v are *incident* if v is one of the endpoints of l.

[9]In particular, $G \in \mathfrak{T}[G]$.

[10]Note that it differs from the standard on-shell renormalization.

The coefficient before γ_μ in $\tilde{f}_G$ is the contribution of G to a_e.

For example, for the graph G from Fig. 3 we will have the following operator expression:

$$[A_G(1-U_{bcdefghij})-(L_G-U_G)A_{bcdefghij}] \times(1-U_{cd})(1-U_{fghi})(1-U_{fgh}-U_{ghi}). \quad (11)$$

Here the subscripts mean the subgraphs to which the operators are applied (denoted by the enumeration of the vertexes). The expression means that we should remove brackets, and for each term we should transform the Feynman amplitudes of the subgraphs using the corresponding operators from the inner subgraphs to the outer ones. The transformation is applied in Feynman parametric space before integration. This can be explained easily using the approach to Feynman parameters based on the transferring from Schwinger parameters; see Ref. [44].

The operators U are designed for removing UV divergences in the way similar to the Zimmermann forest formula and Bogoliubov's R-operation. In contrast to the usual for QED operator L the operators U do not generate additional IR divergences. The multiplier in the square brackets in (11) corresponds to elimination of the IR divergences that correspond to soft virtual photons on the external electron lines and the UV divergences connected with the subgraphs to which the operators are applied. Also, the "overall" UV and IR divergences are removed by the magnetic moment projector A as well as it works in the 1-loop case; see [43,44]. It is important that the operator U applied to self-energy subgraphs extracts the self-mass part completely. This allows us to avoid IR divergences of power type; see Discussion in Ref. [44]. The cancellation of divergences is described in detail[11] in terms of Feynman parameters in Ref. [44]; see also additional comments in Ref. [32].

The equivalence of the subtraction procedure and the direct subtraction on the mass shell is proved in a combinatorial way in Ref. [44], Appendix B. For proving this equivalence we use the fact that the operator U preserves the Ward identity; see Ref. [44]. It is easy to see this equivalence in the 2-loop case; see Sec. 3 of Ref. [44]. Let us note that we do not use the operator of QED on-shell renormalization of electron self-energy subgraphs; the Ward identity helps us in this case too. For a detailed explanation of the developed method, see Ref. [44] and some additional explanations in Refs. [32,42].

III. MONTE CARLO INTEGRATION

A. Probability density functions

After removing divergences the contribution of each Feynman graph to $A_1^{(2n)}$ is represented as an integral of the form

$$\int_{z_1,\ldots,z_M>0} I(z_1,\ldots,z_M)\delta(z_1+\cdots+z_M-1)dz_1\ldots dz_M, \quad (12)$$

where $M=3n-1$ (see[12]), z_j are the Feynman parameters. For each graph we calculate the $(3n-2)$-dimensional integral directly; we do not use any additional reductions.

We propose to split all the integration area into the Hepp sectors (see Ref. [46]) that are simply orders on the Feynman parameters:

$$z_{j_1}\geq z_{j_2}\geq\ldots\geq z_{j_M}.$$

We use the probability density functions of the form

$$g(\underline{z})=C_1g_1(\underline{z})+C_2g_2(\underline{z})+C_3g_3(\underline{z})+C_4g_4(\underline{z}), \quad (13)$$

where $\underline{z}=(z_1,\ldots,z_M)$,

$$g_1(\underline{z})=C\cdot\frac{\prod_{l=2}^{M}(z_{j_l}/z_{j_{l-1}})^{\mathrm{Deg}(\{j_l,j_{l+1},\ldots,j_M\})}}{z_1\cdot z_2\cdot\ldots\cdot z_M}, \quad (14)$$

C_1, C_2, C_3, C_4 are some constants (see Sec. IV), $\mathrm{Deg}(s)$ are positive real numbers for each set s of internal lines[13] of the graph (except the empty and full sets), C is the normalization constant defined by

$$\int_{z_1,\ldots,z_M>0} g_1(z_1,\ldots,z_M)\delta(z_1+\cdots+z_M-1)dz_1\ldots dz_M=1.$$

The stabilization functions g_2, g_3, g_4 are defined in Ref. [32]; an additional constant D is used for defining g_3.

Functions of the form (14) were first used for approximating the behavior of parametric integrals by E. Speer; see Ref. [47].

The main problem in this approach is that for good Monte Carlo convergence the values Deg must be adjusted very accurately. Speer's lemma (Ref. [47]) states that in some simple cases, when we do not have UV divergent subgraphs and we do not consider the infrared behavior, we may take the ultraviolet degree of divergence (with the sign minus) of s as $\mathrm{Deg}(s)$ and use (14) as an upper bound for $|I(\underline{z})|$. A good upper bound can play the role of a good probability density function for Monte Carlo integration; see Ref. [42]. However, in the real case we should use more complicated formulas for obtaining $\mathrm{Deg}(s)$. These formulas were developed for

[11] Although not completely rigorously.

[12] We use a trick for reducing the number from $3n$ to $3n-1$; see [42].

[13] If we use the trick for reducing the number of variables by one, we consider two electron lines that adjoin the external photon line as one line.

our calculations.[14] The first version of the method was presented in Ref. [42]. We use an improved version from Ref. [32]. The algorithm of obtaining $\mathrm{Deg}(s)$ uses six constants $C_{\mathrm{bigF}} > 0$, $C_{\mathrm{bigZ}} > 0$, C_{add}, C_{subI}, C_{subSE}, C_{subO} that should be chosen by hand. For the 5-loop case we use the same values as we used for the 4-loop, 3-loop, and 2-loop cases in Ref. [32]:

$$C_{\mathrm{bigZ}} = 0.256, \qquad C_{\mathrm{bigF}} = 0.839, \qquad C_{\mathrm{add}} = 0.786,$$
$$C_{\mathrm{subI}} = 0.2, \qquad C_{\mathrm{subSE}} = 0, \qquad C_{\mathrm{subO}} = 0.2. \tag{15}$$

These values were obtained by numerical experiments with 4-loop graphs. For clarity and completeness note that some of the values $\mathrm{Deg}(s)$, obtained by the method, less than 1 and even sometimes less than $1/3$, in contrast to integer numbers in Speer's lemma (Ref. [47]).

The terms $C_j g_j(\underline{z})$, $j = 2, 3, 4$ in (13) are added for insurance: they cannot slow down the Monte Carlo convergence speed significantly, but they can (in principle) prevent from occasional emergence of gigantic contributions of some samples; see Ref. [32].

The algorithm of fast random sample generation is described in Ref. [42].

B. Obtaining the value and uncertainty

If the random samples $\underline{z}_1, \ldots, \underline{z}_N$ are generated with the probability density function $g(\underline{z})$, then the integral value is approximated as

$$\frac{1}{N}\sum_{j=1}^{N}\frac{I(\underline{z}_j)}{g(\underline{z}_j)}. \tag{16}$$

For approximating the standard deviation σ we can use the formula

$$\sigma^2 = \frac{\sum_{j=1}^{N} y_j^2}{N^2} - \frac{(\sum_{j=1}^{N} y_j)^2}{N^3}, \tag{17}$$

where $y_j = I(\underline{z}_j)/g(\underline{z}_j)$. However, in practice this formula often leads to an underestimation of the standard deviation. The reason is that the real σ^2 is the mean value of the right part of (17), but using (17) we will rather obtain something near the median of that value that is often less than the mean value. Taking into account this difference is especially important when we integrate unbounded functions. Because of this, we use an improved value $\sigma_\uparrow$ as σ instead of (17). The algorithm of obtaining $\sigma_\uparrow$ based on heuristic predictions is described in Ref. [32]. For the 5-loop case we use exactly the same method. The value defined by (17) we denote by $\sigma_\downarrow$. A large value of $\sigma_\uparrow/\sigma_\downarrow$ indicates that the obtained integral value is suspicious, but no guarantees are possible for Monte Carlo integration. We use $\sigma_\uparrow$ for all intervals in the paper.

[14]However, a rigorous mathematical proof that the expressions of this form can be used as upper bounds for $I(\underline{z})$ has not been obtained yet. The assurance is based on numerical experiments.

IV. REALIZATION

A. Evaluation of the integrands with GPUs

The code for all 3213 integrands was generated automatically. The D programming language was used for the code generator; see Ref. [48]. The generated code was written in C++[15] with CUDA; see Ref. [49]. The code generation took about one month on two CPU cores of a personal computer.

Numerical subtraction of divergences under the integral sign can cause round-off errors. We use interval arithmetic (IA) for controlling them. In interval arithmetic we work not with numbers, but with intervals of numbers. NVidia GPUs support all necessary operations for the realization of interval arithmetic. However, arithmetic operations with intervals are slow, and we developed a fast modification of interval arithmetic that was called "*eliminated interval arithmetic*" (EIA). The main idea of EIA is that in some cases we can replace a large sequence of interval arithmetic operations by the analogous sequence of operations on the centers of the intervals and estimate the radius of the final interval by a relatively simple formula. The intervals obtained by EIA are wider than the ones obtained by IA, but both of them are reliable. EIA is described in detail in Ref. [32].

The integrals for all Feynman graphs are calculated simultaneously; see Sec. IV C. At the stage of initialization, we evaluate approximately 10^8 random points for each Feynman graph with the machine double-precision IA taking the nearest to zero point of each interval. After initialization, when we evaluate the value of $I(\underline{z})/g(\underline{z})$ from (16) at some point $\underline{z}$, we first calculate it using EIA. The obtained interval $[y^-; y^+]$ is accepted if[16]

$$y^+ - y^- \le \frac{1}{4}\sigma_{\downarrow,j}\cdot\frac{\sqrt{\sum_l (\sigma_{\downarrow,l})^2}}{\sum_l \sigma_{\downarrow,l}}, \tag{18}$$

where the summations go over all contributing Feynman graphs, j is the number of the current graph, $\sigma_{\downarrow,l}$ is the value of $\sigma_\downarrow$ calculated for the integral corresponding to the graph with the number l. This formula guarantees that the total

[15]We did not use any substantial improvement of C++ over C like object oriented programming for the generated code. But some little improvements were used, so we must call it "C++", not "C".

[16]This criteria differs from the previous one from Ref. [32]. The previous criteria was erroneous: it did not take into account that the mean value of the round-off error is not zero. However, that error did not significantly affect the result.

round-off error (summed over all graphs) does not exceed $C\sigma_{\downarrow}$ for some constant C. Also, it satisfies the natural demand that larger round-off errors are possible for graphs with larger $\sigma_{\downarrow,l}$. If the interval was not accepted, it is recalculated using IA with increased precisions until it is accepted: machine double precision, 128-bit-mantissa precision, 192-bit-mantissa precision, 256-bit-mantissa precision. If all precisions failed, then the contribution is supposed to be zero. EIA fails approximately on one in five samples. However, the integrand evaluation in EIA is approximately 6.5 times faster than in the double-precision IA; see Sec. V. Thus, the usage of EIA significantly improves the performance.

The Monte Carlo samples are generated and performed by blocks. Each block contains approximately 10^9 samples pertaining to a single Feynman graph. The block scheduling algorithm is described in Sec. IV C. The samples are processed on a GPU in 20480 parallel threads.[17] Each thread processes some set of the block samples sequentially. Branching is not allowed in the execution of a code for GPU, so the samples requiring increased precision are collected and then processed in the subsequent GPU calls.

We use a handmade library for arbitrary precision arithmetic. The 128-bit-mantissa arithmetic is realized using the GPU register memory.[18] The greater precisions are realized with the global GPU memory. The usage of the register memory improves the performance by approximately 10 times.[19] Nevertheless, the increased precision calculations occupy a considerable part of the calculation time; see Sec. V.

For each integrand we generate program codes for three precisions separately: EIA, double-precision IA, and arbitrary-precision IA. This leads to a relatively large code. The total size of the integrands code is 400 GB in the not compiled form and 500 GB in the compiled form.

The calculation of some integrand values requires millions of arithmetic operations. However, both compilers and optimizers do not like big functions. We split the calculation of each integrand into several CUDA kernels.[20] Each CUDA kernel contains approximately 3000 arithmetic operations for the EIA code, 2000 operations for the double-precision IA code, and 1000 operations for the arbitrary-precision IA code. The arbitrary-precision integrand code is also split into several files: approximately 50 CUDA kernels per file. The choice of the function sizes is a compromise: the performance of small functions suffers from memory transfer delays, but a big function size leads to a badly optimized[21] and slowly compiled code.

We use the techniques for prevention of occasional emergence of very large values that are described in [42] (with little modifications and adaptation for GPU parallelism).

When we calculate $I(\underline{z})/g(\underline{z})$, it is often the case that machine double precision is not enough for storing $g(\underline{z})$. The machine double precision allows values up to 2^{1025}. This situation is due to a large number of variables and a closeness of some values of $\mathrm{Deg}(s)$ from (14) to zero. It is not obvious from the beginning that these points can be ignored; see Sec. V. To solve this problem, we store $g(\underline{z})$ as $x \cdot 2^j$, where $0.5 \le x < 1$ is stored with machine double precision, j is stored as 32-bit integer.

B. Compilation of the integrands code

The integrands code was compiled with the NVidia Compiler `nvcc` into shared libraries that are linked dynamically with the integrator. The compiler is a relatively slow one, and 400 GB of code requires a lot of time for compilation. Like the integration, this compilation was performed on the supercomputer "Govorun" from JINR (Dubna, Russia). The processors Intel Xeon Gold 6154 with 18 cores were mostly used for this work. The compilation operation was organized using the MPI[22] protocol with parallel processes that run `nvcc`: two processes per CPU core. The total compilation time amounted to about 120 CPU-hours.

C. Monte Carlo integration: Details

The Monte Carlo integrator was written in C++ with CUDA. The integration was performed on several GPUs NVidia Tesla V100 of the supercomputer "Govorun" from JINR (Dubna, Russia). Most of the time from 2 to 16 GPUs were occupied for the integration. The interdevice parallelism was organized using the MPI protocol.

The controlling part of the integrator generates the numbers of Feynman graphs to obtain a next block of samples. The number j of a Feynman graph is generated randomly. The probabilities p_j of taking the graph j are chosen to make the convergence as fast as possible. Let us describe the method of obtaining p_j. Put

$$C_j = \sigma_{\uparrow,j}\sqrt{N_j},$$

where N_j is the number of samples that have already been processed for the graph j. By t_j we denote the average

[17] 80 blocks of 256 threads; see [49].

[18] The register memory is the fastest kind of memory in NVidia GPUs.

[19] However, the computation shows a more significant gap; See Sec. V. That is because there are very few points that require 192-bit-mantissa and more precision, and the GPU parallelism can not be exploited for all its worth on these points.

[20] A CUDA kernel is a GPU function that is called from the CPU part; see [49].

[21] We are not sure that we understand the behavior of the NVidia optimizer. For example, increasing the CUDA kernel size from 2000 arithmetic operations to 3000 ones sometimes slows down the integrand evaluation speed twice.

[22] Message passing interface.

time required for evaluation of one integrand value for the graph j. The total time that is needed for evaluation of N samples is approximately

$$t = N\sum_j p_j t_j.$$

The total standard deviation can be estimated as

$$\sigma^2 = \frac{1}{N}\sum_j \frac{(C_j)^2}{p_j} = \frac{1}{t}\left(\sum_j \frac{(C_j)^2}{p_j}\right)\left(\sum_j p_j t_j\right) = \frac{1}{t}\left(\sum_j \frac{(C_j)^2 t_j}{q_j}\right)\left(\sum_j q_j\right),$$

where $q_j = p_j t_j$. The minimum point satisfies the equation

$$\left(\frac{\partial}{\partial q_i} - \frac{\partial}{\partial q_l}\right)\left(\sum_j \frac{(C_j)^2 t_j}{q_j}\right) = 0$$

for any i, l. Using this, we obtain

$$q_j = CC_j\sqrt{t_j},$$

where C is some constant, or

$$p_j = \frac{C_j/\sqrt{t_j}}{\sum_l (C_l/\sqrt{t_l})}.$$

We use these probabilities for random generation of the graph numbers with a little modification for stabilization: a little more attention is being given to the graphs j with big $\sigma_{\uparrow,j}/\sigma_{\downarrow,j}$.

After integration, the total standard deviations (upper and lower) are obtained by

$$(\sigma_\uparrow)^2 = \sum_j (\sigma_{\uparrow,j})^2, \qquad (\sigma_\downarrow)^2 = \sum_j (\sigma_{\downarrow,j})^2. \quad (19)$$

V. RESULTS AND THE TECHNICAL INFORMATION

For reliability, two calculations were performed with different pseudorandom generators, with different choices of the constants C_2, C_3, C_4 from (13) and the constant D that is used for defining g_3 from (13); see Ref. [32].

(i) `Calc 1`: the generator `MRG32k3a` from the NVidia CURAND library,

$$C_2 = 0.03, \quad C_3 = 0.035, \quad C_4 = 0.035, \quad D = 0.75.$$

(ii) `Calc 2`: the generator `Philox_4x32_10` from the NVidia CURAND library,

$$C_2 = 0.03, \quad C_3 = 0.01, \quad C_4 = 0.06, \quad D = 0.75.$$

We use the value

$$C_1 = 1 - C_2 - C_3 - C_4$$

for all calculations.

The calculations have led to the results

$$A_1^{(10)}[\text{no lepton loops, Calc 1}] = 6.74(13),$$

$$A_1^{(10)}[\text{no lepton loops, Calc 2}] = 6.84(12).$$

The results were first statistically combined graph-by-graph and then were summed using (19). These operations are not commutative. Thus, some of the results may look strange.[23]

The supplemental materials contain the results for all 3213 Feynman graphs for both calculations [45].

Table I contains the results for nine gauge-invariant classes (k, m, m') splitting the set of all 5-loop Feynman graphs without lepton loops. By definition, (k, m, m') is the set of all Feynman graphs such that m and m' are the quantities of internal photon lines to the left and to the right from the external photon line (or vice versa), k is the quantity of photons with the ends on the opposite sides of it. In this table, N_{diag} and N_{total} are the number of Feynman graphs and the total number of Monte Carlo samples generated for this class.

It was observed by different researchers that the contributions of gauge-invariant classes are relatively small in absolute value, but the contributions of individual Feynman graphs are relatively large and often significantly greater than the class contributions. This occurs regardless of the divergence elimination method used. Table I demonstrates this fact: the sums and maximums of the graph contribution absolute values are included to the table. Some of the individual graph contributions are 10 times greater than the total contribution. However, this "oscillating" nature does not emerge at the level of Feynman parameters. The table demonstrates this too: if the graph contributions are obtained by (12), then the values of

$$\int_{z_1,\ldots,z_M>0} |I(z_1,\ldots,z_M)|\delta(z_1+\cdots+z_M-1)dz_1\ldots dz_M$$

are greater than the contribution absolute values only a little; the sums are given in the table. These values are useful for understanding what accuracy can potentially be reached by Monte Carlo integration methods with these integrands. The values for the individual graphs are presented in the supplemental materials. The Feynman

[23]For example, in Table I some average values are not in the interval of the source values.

TABLE I. Contributions of the gauge invariant classes (k,m,m') to $A_1^{(10)}$; here, $a_i = \int I_i(\underline{z})d\underline{z}$ is the contribution of the i-th graph to the value, I_i is the corresponding Feynman parametric integrand.

Class	Calc 1	Calc 2	Value $= \sum_i a_i$	$\sum_i \lvert a_i\rvert$	$\max_i \lvert a_i\rvert$	$\sum_i \int \lvert I_i(\underline{z})\rvert d\underline{z}$	N_{diag}	N_{total}
(1,4,0)	6.158(49)	6.184(45)	6.157(33)	1219.8	11.8	2521.8	706	43×10^{12}
(2,3,0)	−0.746(63)	−0.763(59)	−0.754(42)	3076.8	46.2	4871.0	706	73×10^{12}
(1,3,1)	0.854(50)	0.972(45)	0.970(33)	3170.1	67.5	3749.9	148	31×10^{12}
(3,2,0)	−0.399(51)	−0.402(47)	−0.403(34)	2593.5	54.9	3783.4	558	56×10^{12}
(2,2,1)	−2.133(53)	−2.197(50)	−2.165(36)	3318.1	85.0	4563.6	370	48×10^{12}
(4,1,0)	−1.028(31)	−0.991(29)	−1.011(21)	1199.3	56.7	1758.2	336	27×10^{12}
(1,2,2)	0.312(30)	0.315(28)	0.315(20)	1338.5	68.7	1515.3	55	11×10^{12}
(3,1,1)	2.628(35)	2.630(33)	2.625(24)	1437.3	63.5	2013.9	261	26×10^{12}
(5,0,0)	1.0929(94)	1.0898(87)	1.0902(62)	137.0	19.3	209.8	73	39×10^{11}

graphs with the maximal absolute values of the contributions are presented in Fig. 1 for each class (k,m,m').

It is very important to check the obtained values independently. However, the amount of computations is huge is this case. Thus, an ability to check the values by parts using different methods would be very useful. We have a splitting of the whole set of graphs into 807 subsets for which the developed subtraction procedure is equivalent to the direct subtraction on the mass shell in Feynman gauge. For each set the equivalence can be proved combinatorially using the Ward identity for individual graphs; see Ref. [32]. The splitting is presented in the supplemental materials. It was generated automatically. Each set in this splitting is contained in some gauge-invariant class (k,m,m'). There are many sets containing only one graph. The largest set contains 706 graphs: it is the class (1,4,0). We do not know if it is possible to divide this class. An analogous splitting and a comparison with known analytical results is presented in Ref. [44] for the 3-loop case and in Ref. [32] for the 2-loop and 3-loop cases without lepton loops. For the 4-loop case without lepton loops an analogous splitting is presented in Ref. [32], but without a comparison (because no one presented the 4-loop results in the form that is applicable for the comparison).

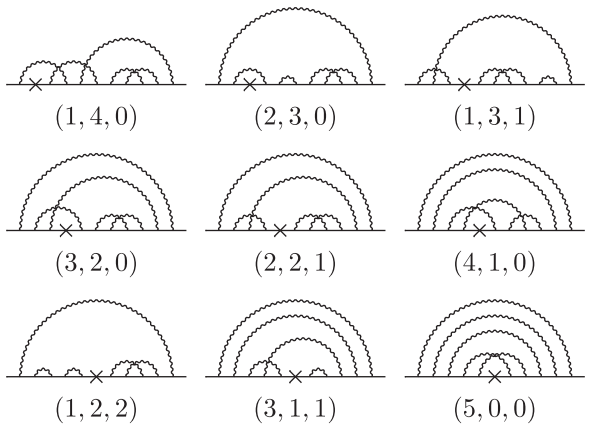

FIG. 1. Graphs from the gauge-invariant classes (k,m,n) with the maximal absolute values of the contributions.

The graph sets from the splitting smooth the peaks of the individual graph contributions as well as the gauge-invariant sets.[24] However, this "smoothing" is not so prominent: some of the set contributions are many times greater than the total contribution (in absolute value). The set with the maximum contribution (in absolute value) is depicted in Fig. 2. This contribution equals 42.0700(50).

Table II contains the dependence of the total calculated value and the error on the number of Monte Carlo samples for `Calc 2`.

Table III contains some technical information about the calculations `Calc 1` and `Calc 2`. The fields of the table have the following meaning:

(i) Value is the obtained value for $A_1^{(10)}$[no lepton loops] with the uncertainty $\sigma_\uparrow$; see Sec. III B and Ref. [32];
(ii) $\sigma_\uparrow/\sigma_\downarrow$ is the relation between the improved standard deviation and the conventional one; see Sec. III B and Ref. [32];
(iii) N_{total} is the total quantity of Monte Carlo samples;
(iv) $N_{\text{EIA}}^{\text{fail}}$ is the quantity of samples for which eliminated interval arithmetic failed; see Sec. IV A and Ref. [32];
(v) $\triangle_{\text{EIA}}^{\text{fail}}$ is the contribution of that samples;
(vi) $N_{\text{IA}}^{\text{fail}}$ is the quantity of samples for which direct double-precision interval arithmetic failed;
(vii) $\triangle_{\text{IA}}^{\text{fail}}$ is the contribution of that samples;
(viii) N_{128}^{fail}, N_{192}^{fail}, N_{256}^{fail} are the quantities of samples for which the interval arithmetic based on numbers with 128-bit, 192-bit, 256-bit mantissa failed;
(ix) $\triangle_{128}^{\text{fail}}$, $\triangle_{192}^{\text{fail}}$ are the contributions of that samples;

[24]It should be noted that this smoothing is not a general principle: for example, the sum of n independent random numbers with the mean values 0 and the quadratic means a have the quadratic mean $a \cdot \sqrt{n}$.

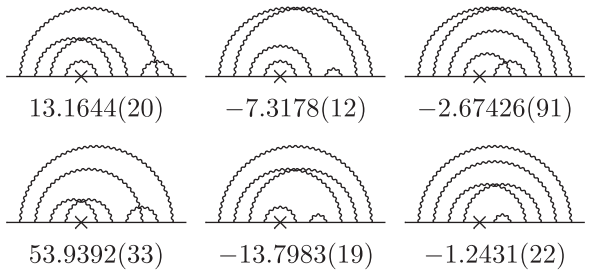

FIG. 2. The set with the maximum contribution (in absolute value) from the splitting for comparison with the direct subtraction on the mass shell: nonoriented Feynman graphs and their contributions to $A_1^{(10)}$.

TABLE II. Dependence of the value and the estimated error on the number of Monte Carlo samples N_{total}: $A_1^{(10)}$ [no lepton loops], Calc 2.

N_{total}	Value	$\sigma_\uparrow/\sigma_\downarrow$
5×10^{11}	9(13)	2.40
10^{12}	10.2(8.9)	2.45
2×10^{12}	11.2(5.4)	2.42
5×10^{12}	9.4(2.6)	2.25
10^{13}	7.9(1.4)	2.10
2×10^{13}	7.21(53)	1.67
5×10^{13}	6.88(24)	1.38

(x) $N^{\text{dens}}_{\text{out of double}}$ is the quantity of samples for which machine double precision was not enough for storing the probability density; see Sec. IV A;
(xi) $\triangle^{\text{dens}}_{\text{out of double}}$ is the contribution of those samples;
(xii) GFlops = billions floating point number operations per second (during the evaluation of the integrands); GIntervals = billions interval operations per second (in the sense of interval arithmetic); M = millions.

It is easy to see that in EIA one arithmetic operation on intervals takes approximately one operation on numbers. This is due to the fact that the biggest part of the EIA calculation is occupied by the operations on the centers of the intervals. However, in IA one interval operation takes approximately five operations on numbers. Also, the speed of the number operations for IA is by 1.6 times less than for EIA. This is because most of the operations in IA require

TABLE III. Technical information about the calculations.

	Calc 1	Calc 2
Value	6.74(13)	6.84(12)
$\sigma_\uparrow/\sigma_\downarrow$	1.31	1.31
N_{total}	15×10^{13}	17×10^{13}
$N^{\text{fail}}_{\text{EIA}}$	34×10^{12}	39×10^{12}
$N^{\text{fail}}_{\text{IA}}$	38×10^{10}	42×10^{10}
N^{fail}_{128}	67×10^{6}	73×10^{6}
N^{fail}_{192}	10787	2453
N^{fail}_{256}	8669	0
$N^{\text{dens}}_{\text{out of double}}$	11×10^{5}	13×10^{5}
$\triangle^{\text{fail}}_{\text{EIA}}$	4	5
$\triangle^{\text{fail}}_{\text{IA}}$	0.9	3
$\triangle^{\text{fail}}_{128}$	−0.07	−0.07
$\triangle^{\text{fail}}_{192}$	−0.002	-3×10^{-6}
$\triangle^{\text{dens}}_{\text{out of double}}$	-6×10^{-13}	6×10^{-10}
Total calculation time, GPU-hours	19515	20341
Share in the time: double-precision EIA	21.6%	23.3%
Share in the time: double-precision IA	35.5%	34.5%
Share in the time: 128-bit-mantissa IA	28.1%	29.1%
Share in the time: 192-bit and 256-bit-mantissa IA	11.4%	10.0%
Share in the time: sample generation	1.8%	1.5%
Share in the time: other operations	1.7%	1.7%
GPU speed: double-precision EIA, GFlop/s	2221.88	2227.99
GPU speed: double-precision EIA, GInterval/s	1962.13	1965.60
GPU speed: double-precision IA, GFlop/s	1358.63	1505.30
GPU speed: double-precision IA, GInterval/s	274.13	303.31
GPU speed: 128-bit-mantissa IA, GFlop/s	13.47	13.48
GPU speed: 128-bit-mantissa IA, GInterval/s	2.54	2.53
GPU speed: 192-bit and 256-bit-mantissa IA, MFlop/s	4.21	4.65
GPU speed: 192-bit and 256-bit-mantissa IA, MInterval/s	0.74	0.80

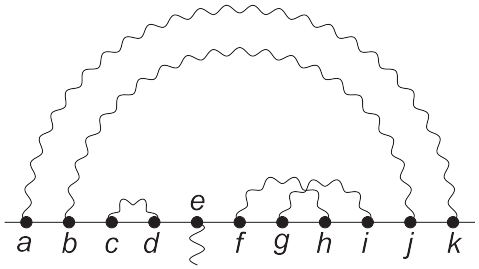

FIG. 3. The graph with the maximum (in absolute value) contribution of the Monte Carlo samples for which eliminated interval arithmetic failed. See Sec. V for the detailed information.

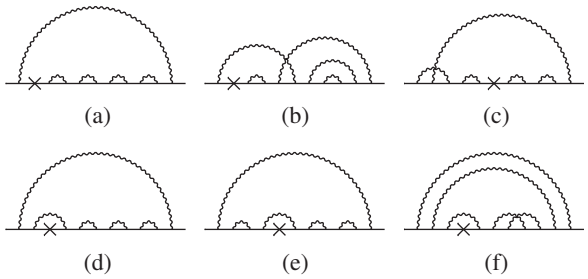

FIG. 4. The extreme graphs of different kinds: (a) best Monte Carlo integration convergence quality; (b) worst Monte Carlo integration convergence quality; (c,d,e) maximal (in absolute value) contribution of the samples for which the interval arithmetic with numbers of double precision, 128-bit mantissa, 192-bit mantissa failed; (f) maximal (in absolute value) contribution of the samples for which double precision was not enough for storing the probability density.

specifying a rounding mode,[25] but the operations on the centers of intervals in EIA do not require it.

`Calc 1` suffered from some errors that cause an emergence of anomalous points that have contributions to N^{fail}_{192}, N^{fail}_{256}, $N^{\text{dens}}_{\text{out of double}}$; see Table III. We can not perform the full recalculation because this requires a lot of time. However, those points do not have a significant impact on the results; the table confirms this fact. Those errors were corrected in `Calc 2`.

Table III demonstrates that the points requiring an increased precision have a significant contribution to the result. For example, $\triangle^{\text{fail}}_{\text{EIA}}$ and $\triangle^{\text{fail}}_{\text{IA}}$ are at the level of the total contribution, $\triangle^{\text{fail}}_{128}$ is at the level of the uncertainty. Also, the table shows that contributions are unstable due to an "oscillating" character of the individual graph contributions, a floating character of the interval acceptance criteria (18), and a difference in the probability density functions. In addition, the table shows that the contribution $\triangle^{\text{dens}}_{\text{out of double}}$ is insignificant. However, this contribution is too far from the boundaries of machine double precision like 2^{-1025} (on a logarithmic scale). Thus, there may be situations, where such contributions will be significant. This fact demonstrates that universal Monte Carlo integration routines can work poorly for many-loop Feynman parametric integrals.

An analogous information for the individual Feynman graphs is contained in the supplemental materials. The graphs with the maximal contributions to $\triangle^{\text{fail}}_{\text{EIA}}$, $\triangle^{\text{fail}}_{\text{IA}}$, $\triangle^{\text{fail}}_{128}$, $\triangle^{\text{fail}}_{192}$, $\triangle^{\text{dens}}_{\text{out of double}}$ are shown in Fig. 3 and Fig. 4(c–f). The corresponding contributions (for `Calc 2`) are

$$67.1, \quad 26.3, \quad 0.15, \quad 3.1 \times 10^{-5}, \quad 5.9 \times 10^{-10}.$$

The Monte Carlo integration convergence quality for a given graph j can be estimated as

$$\frac{\sigma_{\uparrow,j} \cdot \sqrt{N_j}}{\int |I_j(\underline{z})| d\underline{z}},$$

where N_j is the number of Monte Carlo samples for the j-th graph, I_j is the corresponding Feynman parametric integrand. Less values correspond to a better quality. The graphs with the best and the worst quality are shown in Fig. 4(a,b). The corresponding values (for `Calc 2`) are

$$16.2, \quad 525.9.$$

These values demonstrate that even in the best case the Monte Carlo integration works not ideally due to large dimensionality. However, this is acceptable and requires a relatively small amount of the supercomputer time for integration.

VI. CONCLUSION

A numerical calculation of the total contribution of the 5-loop QED Feynman graphs without lepton loops to the corresponding coefficient of the electron anomalous magnetic moment expansion in α was performed. The calculation is based on a specific method of reduction of the problem to Feynman parametric integrals and on Monte Carlo integration using a supercomputer. Usage of some mathematical considerations about the integrands behavior provided us an ability to reduce the amount of the needed supercomputer power and time significantly.

This calculation provides the first independent check of the value obtained by T. Kinoshita's team that is presented in Ref. [33]. However, the discrepancy of about 4.8σ between the results was discovered. On the one hand, this discrepancy does not significantly affect the known values of a_e and α. But on the other hand, it requires an additional independent calculation and can affect the physics in the future.

The results of the calculation are presented in detail. This detailed presentation gives us an ability to check the results by parts using other methods. The contribution values of nine gauge-invariant classes splitting the whole set are

[25]However, this difference in the speed was not discovered in the calculations on NVidia Tesla K80 from Ref. [32] despite the fact that the difference was discovered during the preliminary tests.

presented for the first time (except the preliminary values in Ref. [35]).

For reliability, two different Monte Carlo integrations with different pseudorandom generators were performed. The results of these calculations agree with each other, and they were statistically combined in the final result.

A cancellation of an "oscillating" nature of the individual Feynman graph contributions in the gauge-invariant classes confirms that the results are correct. This "oscillating" nature is described in detail. However, there is no mathematical foundation for this cancellation at the current moment of time. Also, it is surprising that we have only an intergraph oscillation, but not in Feynman parametric space for one graph.

The technical information that is presented in the paper will be useful for the scientists that are going to perform many-loop calculations in quantum field theory or another computations using supercomputers and graphics accelerators. Also, the provided information about the Monte Carlo integration will be useful for developers of Monte Carlo integrators.

In closing, let us recapitulate some problems that still remain open:

(1) To perform an independent calculation of the 5-loop contribution of the graphs with lepton loops, to check the value from Ref. [34];
(2) To prove rigorously (or disprove) that the developed subtraction procedure (Ref. [44]) leads to finite integrals for each suitable Feynman graph;
(3) To substantiate rigorously the developed Monte Carlo integration method (Ref. [42]) and to extend it to the graphs with lepton loops;
(4) To explain why the "oscillating" nature of the individual Feynman graph contributions is cancelled in the gauge-invariant classes.

ACKNOWLEDGMENTS

The author thanks Andrey Kataev for helpful recommendations, Lidia Kalinovskaya for her help in organizational issues, and Predrag Cvitanović for the ideas about gauge-invariant classes. Also, the author thanks the Laboratory of Information Technologies of JINR (Dubna, Russia) for providing an access to the supercomputer "Govorun" and the organizers of the conference ACAT-2019 (Saas Fee, Switzerland) for providing an ability to present the preliminary results at the conference without financial problems.

[1] D. Hanneke, S. F. Hoogerheide, and G. Gabrielse, Phys. Rev. A **83**, 052122 (2011).
[2] J. Schwinger, Phys. Rev. **73**, 416 (1948).
[3] J. Schwinger, Phys. Rev. **76**, 790 (1949).
[4] A. Petermann, Helv. Phys. Acta **30**, 407 (1957).
[5] C. Sommerfield, Phys. Rev. **107**, 328 (1957).
[6] R. Carroll and Y. Yao, Phys. Lett. **48B**, 125 (1974).
[7] R. Carroll, Phys. Rev. D **12**, 2344 (1975).
[8] M. J. Levine and J. Wright, Phys. Rev. D **8**, 3171 (1973).
[9] P. Cvitanović and T. Kinoshita, Phys. Rev. D **10**, 4007 (1974).
[10] S. Laporta and E. Remiddi, Phys. Lett. B **379**, 283 (1996).
[11] J. A. Mignaco and E. Remiddi, Nuovo Cimento A **60**, 519 (1969).
[12] R. Barbieri, M. Caffo, and E. Remiddi, Lett. Nuovo Cimento **5**, 769 (1972).
[13] D. Billi, M. Caffo, and E. Remiddi, Lett. Nuovo Cimento **4**, 657 (1972).
[14] R. Barbieri and E. Remiddi, Phys. Lett. **49B**, 468 (1974).
[15] R. Barbieri, M. Caffo, and E. Remiddi, Lett. Nuovo Cimento **9**, 690 (1974).
[16] M. J. Levine and R. Roskies, Phys. Rev. D **9**, 421 (1974).
[17] K. A. Milton, W. Tsai, and L. L. DeRaad, Jr., Phys. Rev. D **9**, 1809 (1974).
[18] L. L. DeRaad, Jr., K. A. Milton, and W. Tsai, Phys. Rev. D **9**, 1814 (1974).
[19] R. Barbieri, M. Caffo, and E. Remiddi, Phys. Lett. **57B**, 460 (1975).
[20] M. J. Levine, R. C. Perisho, and R. Roskies, Phys. Rev. D **13**, 997 (1976).
[21] M. J. Levine and R. Roskies, Phys. Rev. D **14**, 2191 (1976).
[22] R. Barbieri, M. Caffo, and E. Remiddi, S. Turrini, and D. Oury, Nucl. Phys. **B144**, 329 (1978).
[23] M. J. Levine, E. Remiddi, and R. Roskies, Phys. Rev. D **20**, 2068 (1979).
[24] S. Laporta and E. Remiddi, Phys. Lett. B **265**, 182 (1991).
[25] S. Laporta, Phys. Rev. D **47**, 4793 (1993).
[26] S. Laporta, Phys. Lett. B **343**, 421 (1995).
[27] T. Kinoshita and W. B. Lindquist, Phys. Rev. Lett. **47**, 1573 (1981).
[28] T. Aoyama, M. Hayakawa, T. Kinoshita, and M. Nio, Phys. Rev. D **91**, 033006 (2015).
[29] S. Laporta, Phys. Lett. B **772**, 232 (2017).
[30] P. Marquard, A. V. Smirnov, V. A. Smirnov, M. Steinhauser, and D. Wellmann, EPJ Web Conf. **218**, 01004 (2019).
[31] F. Rappl, Dissertationsreihe der Fakultät für Physik der Universität Regensburg 49, Ph.D. Dissertation, Universität Regensburg, 2016.
[32] S. Volkov, Phys. Rev. D **98**, 076018 (2018).
[33] T. Aoyama, T. Kinoshita, and M. Nio, Atoms **7**, 28 (2019).
[34] T. Aoyama, T. Kinoshita, and M. Nio, Phys. Rev. D **97**, 036001 (2018).

[35] S. Volkov, arXiv:1905.08007.
[36] R. H. Parker, C. Yu, W. Zhong, B. Estey, and H. Muller, Science **360**, 191 (2018).
[37] R. Bouchendira, P. Clade, S. Guellati-Khelifa, F. Nez, and F. Biraben, Phys. Rev. Lett. **106**, 080801 (2011).
[38] P. J. Mohr, D. B. Newell, and B. N. Taylor, Rev. Mod. Phys. **88**, 035009 (2016).
[39] V. A. Scherbina, *Catalogue of Deposited Papers* (VINITI, Moscow, 1964), p. 38 (in Russian).
[40] O. I. Zavialov and B. M. Stepanov, Yad. Fys. **1**, 922 (1965) (in Russian).
[41] W. Zimmermann, Commun. Math. Phys. **15**, 208 (1969).
[42] S. Volkov, Phys. Rev. D **96**, 096018 (2017).
[43] P. Cvitanović and T. Kinoshita, Phys. Rev. D **10**, 3991 (1974).
[44] S. Volkov, Zh. Eksp. Teor. Fiz. **149**, 1164 (2016) [J. Exp. Theor. Phys. **122**, 1008 (2016)].
[45] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevD.100.096004 for detailed results of the computation.
[46] K. Hepp, Commun. Math. Phys. **2**, 301 (1966).
[47] E. Speer, J. Math. Phys. (N.Y.) **9**, 1404 (1968).
[48] A. Alexandrescu, *The D Programming Language* (Addison-Wesley Professional, Reading, 2010).
[49] CUDA C Programming Guide, NVIDIA Developer Documentation.