doi 10.26089/NumMet.2024s03

Feynman integral reduction: balanced reconstruction of sparse rational functions and implementation on supercomputers in a co-design approach

Alexander V. Smirnov
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: 0000-0002-7779-6735, e-mail: asmirnov@srcc.msu.ru

Mao Zeng
Higgs Centre for Theoretical Physics, Edinburgh, United Kingdom
ORCID: 0000-0002-4741-4038, e-mail: mao.zeng@ed.ac.uk

Abstract: Integration-by-parts (IBP) reduction is one of the essential steps in evaluating Feynman integrals. A modern approach to IBP reduction uses modular arithmetic evaluations at the specific numerical values of parameters with subsequent reconstruction of the analytic rational coefficients. Due to the large number of sample points needed, problems at the frontier of science require an application of supercomputers. In this article, we present a rational function reconstruction method that fully takes advantage of sparsity, combining the balanced reconstruction method and the Zippel method. Additionally, to improve the efficiency of the finite-field IBP reduction runs, at each run several numerical probes are computed simultaneously, which allows to decrease the resource overhead. We describe what performance issues one encounters on the way to an efficient implementation on supercomputers, and how one should co-design the algorithm and the supercomputer infrastructure. We present characteristic examples of IBP-reduction in the case of massless two-loop four- and five-point Feynman diagrams using a development version of FIRE and give illustrative examples mimicking the reduction of coefficients appearing in scattering amplitudes for post-Minkowskian gravitational binary dynamics.

Keywords: Feynman integrals, computer algebra, calculation optimization, supercomputer co-design.

Acknowledgements: The work of Mao Zeng is supported in part by the U.K. Royal Society through Grant URF\R1\20109. The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [1]. We are grateful to the supercomputer team support, especially to I. D. Fateev for being ready to upgrade and tune the supercomputer options for our needs.

For citation: A. V. Smirnov, Mao Zeng, “Feynman integral reduction: co-design of balanced reconstruction algorithm and supercomputers,” Numerical Methods and Programming. 2024. Special Issue. 30–45. doi 10.26089/NumMet.2024s03.

Редукция интегралов Фейнмана: суперкомпьютерный кодизайн алгоритма балансированного восстановления разреженных рациональных функций

А. В. Смирнов
Московский государственный университет имени М. В. Ломоносова,
Научно-исследовательский вычислительный центр, Москва, Российская Федерация
ORCID: 0000-0002-7779-6735, e-mail: asmirnov@srcc.msu.ru

Мао Цзэн
Центр теоретической физики Хиггса,
Университет Эдинбурга, Эдинбург, Великобритания
ORCID: 0000-0002-4741-4038, e-mail: mao.zeng@ed.ac.uk

Аннотация: Редукция с использованием интегрирования по частям (IBP) является одним из существенных этапов при вычислении интегралов Фейнмана. Современный подход к IBP-редукции использует модулярную арифметику при конкретных числовых значениях параметров в пробных точках с последующей реконструкцией аналитических рациональных коэффициентов. Задачи, возникающие на переднем крае науки, требуют применения суперкомпьютеров из-за большого количества необходимых проб. В этой статье мы представляем алгоритм рациональной реконструкции, который в полной мере использует преимущества разреженности, объединяя сбалансированный алгоритм реконструкции и алгоритм Зиппеля. Кроме того, для повышения эффективности редукции в модулярной арифметике при каждом запуске одновременно вычисляется несколько числовых проб, что позволяет сокращать потребляемые ресурсы. Мы описываем, какие проблемы появляются на пути к эффективной реализации на суперкомпьютерах и как следует совместно проектировать алгоритмы и соответствующую суперкомпьютерную инфраструктуру. Представлены характерные примеры IBP-редукции в случае безмассовых двухпетлевых четырехточечных и пятиточечных диаграмм Фейнмана с использованием частной версии FIRE, а также показательные примеры редукции, имитирующие редукцию коэффициентов, появляющихся в амплитуде рассеяния в рамках постминковской гравитационной бинарной динамики.

Ключевые слова: Интегралы Фейнмана, компьютерная алгебра, оптимизация вычислений, суперкомпьютерный кодизайн.

Благодарности: Работа Мао Цзэна частично поддержана Королевским Обществом Великобритании URF\R1\20109. Исследования проводились с использованием оборудования коллективного пользования вычислительных ресурсов НРС Московского государственного университета имени М. В. Ломоносова [1]. Мы благодарны команде поддержки суперкомпьютерного центра, особенно И. Д. Фатееву за готовность модернизировать и настроить возможности суперкомпьютера под наши задачи.

Для цитирования: Смирнов А.В., Цзэн М., Редукция интегралов Фейнмана: суперкомпьютерный кодизайн алгоритма балансированного восстановления // Специальный выпуск. 30–45. doi 10.26089/NumMet.2024s01.

1. Introduction. Feynman integral reduction is one of the essential steps in evaluating Feynman integrals, and it is based on the integration-by-part (IBP) relations [2, 3] that allow reducing required integrals to a small subset of so-called master integrals.

From the mathematical point of view, IBP reduction is the solving of a huge sparse system of linear equations with polynomial coefficients. This statement is a serious simplification of the process, because mathematically the number of unknowns (Feynman integrals with particular indices) and the number of relations (IBP relations) is infinite, but it is known that the basis of the vector space of Feynman integrals of a particular family is finite-dimensional [4]. Nevertheless, when approaching a particular reduction problem, the complexity of the system might differ significantly depending on the choice of equations to be used — while knowing the requested unknowns to be found, one can choose differently the bases of what they are reduced to the master integrals and the intermediate relations to use. In this paper we will assume that this choice is already made, so the reduction comes down to solving a huge sparse linear system.

The conventional approach consists of solving this system directly, i.e. over the field of rational functions of some variables, the masses and kinematic invariants. The number of these variables can vary from 1 (only the space-time dimension) to 5–6 in complicated cases. To solve such a system, the Gaussian elimination method could be used, but on each step one needs to simplify the rational coefficients, bringing each of them to a common denominator and canceling the greatest common divisor. Avoiding such a step would lead to “hidden zeros” when a function is identically equal to zero but looks like a complex expression, or even if one could avoid hidden zeros, this would lead to uncontrolled growth of intermediate coefficients. There is a number of public programs performing the conventional Feynman integral reduction [3–10], some of them also use the modular approach we are going to discuss. The public programs of Refs. [11, 12] focus on the modular approach exclusively.

The simplification of intermediate coefficients is a time-consuming step taking almost all the time in the conventional approach. A proper choice of the simplification library as well as some other tricks aim to make this step faster, but in this paper we are going to discuss another approach. One of the most promising methods of Feynman integral reduction is based on modular arithmetic and the subsequent reconstruction of analytic coefficients [13–30]. If one fixes specific values of variables, the calculations will be performed over the field of rational numbers. To prevent their growth, one more step is made and one takes the projection of coefficients into a finite field modulo some large prime number. Now the coefficients can fit into machine-sized arithmetic (normally up to 2^{64}) and solving such a system becomes immensely faster compared with the system over rational functions. Note that the prime choice around 2^{64} is made to fit calculations into machine-sized arithmetic. Making them significantly smaller does not speed reduction any more, but leads to a need of more primes for a successful reconstruction.

Obviously, solving a substituted system once is not enough. In order to reconstruct the analytical coefficients, one needs to solve it many times for different variable values and different primes, and the number of “sampling points” might be a large number (in our works we encountered values exceeding a million). Thus, the modular approach to Feynman integral reduction changes the class of the problem. The conventional approach for a complicated reduction requires a server with multiple processors, a large amount of RAM and a preferably non-interrupted run that might take a few months. The modular approach is designed for the use of supercomputers where multiple small tasks can be distributed over a set of a many less powerful nodes (typically with less RAM per node).

When using supercomputers, there are usually problems that would not appear when developing for servers, even large ones. For each supercomputer one knows its peak performance as well as its behaviour on some benchmarks such as the LINPACK (while the LINPACK benchmark measures the floating point operations and we work with integers, the modern processors normally have a similar factor to convert one kind of operations to another in terms of CPU clock). But is it possible to achieve such a performance for a particular problem? The gap between the theoretical performance and real performance can become enormous, as the loss of performance occurs on each step starting from sequational implementation. Certainly the algorithm should be optimal, but the usual starting point is to optimize it for sequential execution. When moving to a server with multiple cores, one considers parallelization of some time-consuming parts. But the prevalent approach to parallelization assumes shared memory, and its efficiency is limited by the number of cores that can exist on single server. Supercomputers are different — the number of cores accessed might be large and exceed a thousand, but at the same time one does not get shared memory across nodes. Designing an algorithm for such a situation requires special skills and the risk to get large overhead costs for something not initially considered problematic. Furthermore, a time-consuming algorithm at the frontier of science should adapt to a particular supercomputer to efficiently utilize its specific characteristics. And ideally, the supercomputer infrastructure should also be adaptable, the supercomputer support staff should react to user problems arising in large-scale computations and adapt to them, leading to a co-design between the supercomputer and applications. Note that there are different definitions of the co-design term, starting from the most strict one meaning initially designing supercomputers with care of algorithms that are going to be evaluated on them, but obviously this generally impossible for most applications. However, a more relaxed approach is to treat existing supercomputers as an object that can be changed during algorithm design, for example with new software installation and tuning of the system.

In this paper, we are going to describe a new algorithm for multivariate rational function reconstruction which fully exploits the polynomial degree bounds in individual variables and the sparsity of the rational functions. We present details of its implementation as part of the version 7 of FIRE (currently in development) and the problems (and their resolutions) encountered on supercomputers. While the code described in this paper is still private, we assume that the algorithm and the presented ideas should be useful to for other scientists working with Feynman integral reduction and even outside this field. For readers from other fields, Appendix 1 contains a toy example which illustrates the main problem of Feynman integrals and their linear algebraic relations known as IBP identities.

2. The reconstruction algorithm.

2.1. Classical Newton and Thiele methods. Let us briefly overview the current status in reconstruction of rational functions. Not to repeat some of the formulas, we refer readers to details in the paper [13] or Refs. [25, 27]. The simplest case is univariate polynomial reconstruction, where a simple Newton interpolation polynomial formula does the job, as there is a well-defined formula for the coefficients based on the values $a_1, \ldots, a_N$ of the polynomial in N points $y_1, \ldots y_N$

$$\begin{aligned} f_N(x) &= \text{Newton}_x[f(x), N] \\ &\equiv a_1 + (x - y_1)\Big[a_2 + (x - y_2)\big[a_3 + (x - y_3)\left[a_4 + \ldots\right]\big]\Big] . \end{aligned} \tag{1}$$

For multivariate polynomial reconstruction one can proceed variable by variable, i.e. when reconstructing from n variables to $n+1$ variables one takes a needed number of reconstructed functions $f(x_1, x_2, \ldots, x_n, x_{n+1,i})$ for different values of $x_{n+1,i}$ of x_{n+1} and runs the univariate Newton reconstruction. In the notation $x_{n+1,i}$, the first subscript index $n + 1$ refers to the position of the variable in the function f, and the second index i specifies the i-th number among an arbitrarily chosen sequence of numerical values. This is not an optimal approach for sparse polynomials, and we are going to discuss it later.

When passing from polynomials to rational functions, the univariate case is still simple enough, since there is a replacement for the Newton formula, the so-called Thiele formula, looking very similar to the previous one

$$\begin{aligned} f_T(x) &= \text{Thiele}_x[f(x), T] \\ &\equiv b_0 + (x - y_1)\left[b_1 + (x - y_2)\left[b_2 + (x - y_3)\left[b_4 + \ldots\right]^{-1}\right]^{-1}\right]^{-1} . \end{aligned} \tag{2}$$

Again, the coefficients are well-defined. The main complexity is that the Thiele formula is a continued fraction and evaluating it algebraically is much more complicated than expanding the Newton polynomial. While this still works in the univariate case, the obvious generalization to the multivariate case with an attempt to run Thiele reconstruction iteratively is inefficient due to the algebraic complexity.

Thus, there exist a number of multivariate rational reconstruction methods, one of which is the balanced Newton reconstruction method proposed by one of the present authors and collaborators [13], which mitigates some of the inefficiencies of the homogeneous reconstruction approach suggested in [25][1].

All reconstruction algorithms mentioned in this paper are valid over both the field of rational numbers and finite fields modulo primes. Despite presenting some simple examples with rational numbers, all our practical implementations of the algorithms exclusively use finite fields modulo prime numbers close to 2^{64}. After running the same reconstruction algorithm for multiple prime numbers, rational numbers will be reconstructed from the

[1]Another alternative to the homogeneous reconstruction approach has recently appeared in Ref. [30] which essentially packs the exponents of variables into a single exponent with a radix numeral system.

finite-field images. For this, we use Monagon's maximal quotient rational reconstruction algorithm [14] which is an improved version of Wang's algorithm [31]. Here we will not discuss this topic again in the rest of the paper and will focus on the reconstruction of rational functions over finite fields. Also, we only consider generally applicable algorithms and do not exploit physics insights on the analytic structures of specific results [32–34] or relations between more than one rational function to be reconstructed [35].

Let us now turn to the main idea of the balanced Newton reconstruction method, as it will be important for the explanation of the new algorithm.

2.2. Balanced Newton method. Suppose we need to reconstruct a bivariate rational function $f(x,y) = n(x,y)/d(x,y)$ and we know its values for some $y = y_i$, being a univariate rational function in the remaining variable x, $n'(x,y_i)/d'(x,y_i)$ (here and later we use this notation for other functions and not the derivative). To make the numerator and denominator of every rational function unambiguous under simultaneous scalings, we require any nontrivial polynomial common divisor to br canceled and furthermore that the "leading monomial" of the denominator to have a coefficient equal to one. The leading monomial is the highest-degree monomial with ties broken by e.g. lexicographic orderings in the variables involved. Take an example,

$$\textit{example:}\ f(x,y) = \frac{n(x,y)}{d(x,y)} = \frac{xy+2}{xy-2x+4}, \quad n(x,y) = xy+2, \quad d(x,y) = xy-2x+4\,, \tag{3}$$

where the leading monomial xy of the denominator $d(x,y)$ is normalized by having a unit coefficient. At $y = y_1 = 4$, the rational function becomes

$$\textit{example:}\ f(x,y_1=4) = \frac{4x+2}{2x+4} = \frac{2x+1}{x+2} = \frac{n'(x,y_1)}{d'(x,y_1)}, \quad n'(x,y_1) = 2x+1, \quad d'(x,y_1) = x+2\,,$$

where after substituting $y_1 = 4$, we have canceled a factor of 2 between the numerator and denominator to restore the normalization convention that the leading monomial x in the denominator $d'(x,y_1)$ has a unit coefficient. One would like to perform Newton reconstruction in the variable y to obtain the bivariate polynomial $n(x,y)$ from the univariate polynomials $n'(x,y_i)$ known at sufficiently large number of values of y_i, but this would not be possible because $n(x,y_i)$ is generally not equal to $n'(x,y_i)$, since some factors between the numerator and denominator are canceled according to the normalization convention. We denote the cancellation with

$$n'(x,y_i) = n(x,y_i)\cdot c(y_i),$$
$$d'(x,y_i) = d(x,y_i)\cdot c(y_i),$$

where $c(y_i)$ is a rational number for each i. For the specific example of Eq. (3), we need

$$\textit{example:}\quad c(y_i) = \frac{1}{y_i-2}$$

to satisfy the normalization convention. Now to run Newton reconstruction, one needs to derive $n(x,y_i)$ from $n'(x,y_i)$ getting rid of $c(y_i)$ or, in other words, "balancing" the numerator and denominator. First of all, one takes $n'(x,y_i)/n'(x_0,y_i)$ for some particular x_0. We have

$$\frac{n'(x,y_i)}{n'(x_0,y_i)} = \frac{n(x,y_i)\cdot c(y_i)}{n(x_0,y_i)\cdot c(y_i)} = \frac{n(x,y_i)}{n(x_0,y_i)}.$$

The factor $c(y_i)$ has vanished, but there is an unknown constant (for each i) in the denominator. Therefore we need to take one more function. Substituting $x \to x_0$ for one particular value of x, we obtain $f(x_0,y) = n''(x_0,y)/d''(x_0,y)$. Again some factor $C = C(x_0)$ might get canceled to normalize the leading monomial of the denominator (now considered as a polynomial in y) to have a unit coefficient, so we have

$$n''(x_0,y) = n(x_0,y)\cdot C,$$
$$d''(x_0,y) = d(x_0,y)\cdot C$$

for some constant C. For the example of Eq. (3), we get

$$\textit{example:}\quad C = \frac{1}{x_0}\,.$$

Now we can complete the balancing taking

$$\frac{n'(x, y_i) \cdot n''(x_0, y_i)}{n'(x_0, y_i)} = \frac{n(x, y_i) \cdot c(y_i) \cdot n(x_0, y_i) \cdot C}{n(x_0, y_i) \cdot c(y_i)} = n(x, y_i) \cdot C\,. \tag{4}$$

The coefficient C on the right-hand side is a constant, since x_0 is considered a constant. The analogous formula works for denominators, also rescaled by the same constant C. Thus, knowing the numerators and denominators (according to our normalization convention) for a sequence y_i and one value of x_0, one can balance the numerator and denominator and run two separate Newton reconstructions to obtain the rescaled numerator $n(x, y)\cdot C$ and the rescaled denominator $d(x, y)\cdot C$. Computing the ratio, we succeed in reconstructing the bivariate rational function $n(x, y)/d(x, y)$.

This procedure works for more than two variables too, but we will skip this discussion here.

As was demonstrated in our paper [13], the balanced Newton method is a useful addition to the dense reconstruction methods since it lacks some problems of homogeneous methods and requires less sampling points. But as was noted by reviewers and fixed in the final version of our paper, this approach is still dominated by the Zippel method ([16], see description in section 2.4) designed for the sparse multivariate polynomial reconstruction. So now let us remind how the Zippel method works for polynomials and then present our balanced Zippel algorithm for multivariate rational function.

2.3. Explicit Example for Balanced Newton Reconstruction. Readers who are thoroughly familiar with the algorithm presented in Section 2.2 can directly jump to Section 2.4. Otherwise, this subsection presents a full example of reconstructing the bivariate rational function in Eq. (3) considered as a *black box* function. For example, the black box may feed the input values of (x, y) to a complicated algorithm, e.g. involving solving a large linear system, before returning the result. As it may be very computationally demanding to run the black box algorithm symbolically to produce the analytic form in Eq. (3), we only use the black box for calculations with rational numerical values of (x, y), called *probes*, before reconstructing the rational function Eq. (3) after obtaining results for sufficiently many probes.

We arbitrary set $y_i = 3 + i$ and $x_0 = 3$. First, we reconstruct $f(x, y_1 = 4)$ by the Thiele interpolation formula (2). We evaluate $f(x, y_1 = 4)$ at a sequence of different values of x. After each evaluation, we apply Eq. (2) to obtain a new candidate rational function in x. When a new evaluation agrees with the candidate function, the procedure has *converged*, and the candidate function is taken as the true function. The sequence of evaluations is shown in Table 1.

Therefore, we reconstruct

$$f(x, y_1 = 4) = \frac{n'(x, y_1)}{d'(x, y_1)}, \quad n'(x, y_1) = 2x + 1, \quad d'(x, y_1) = x + 2\,,$$

following the normalization convention. Similarly, for $y_2 = 5$, Thiele interpolation over the same set of x values produces

$$f(x, y_2) = \frac{n'(x, y_2)}{d'(x, y_2)}$$

with

$$n'(x, y_2) = (5/3)x + 2/3, \quad d'(x, y_3) = x + 4/3\,.$$

If we fix $x = x_0 = 3$ and evaluate $f(x_0, y)$ at different values of $y = 4, 5, 6, 7$, the last evaluation agrees with the candidate function produced after the preceding evaluation and gives

$$f(x_0 = 3, y) = \frac{n''(x_0, y)}{d''(x_0, y)}, \quad n''(x_0, y) = y + 2/3, \quad d''(x_0, y) = y - 2/3\,.$$

Table 1. Steps in Thiele reconstruction of $f(x, y_1 = 4)$ for the black box function Eq. (3).

x	$f(x,4)$	Converged?	New Candidate function
3	7/5	N/A	7/5
4	3/2	No	$x + 11/10$
5	11/7	No	$(2x + 1)/(x + 2)$
6	13/8	Yes	N/A

According to Eq. (4), n'' and d'' are different from n and d, respectively, by a constant factor. Therefore, we know that n and d are linear polynomials in y[2]. Then it suffices to obtain Eq. (4) at two different values of y to complete the reconstruction, as will become clear shortly. Having obtained the univariate rational functions, $n'(x, y_i)$ for two constant values of y_i and $n''(x_0, y)$ for a constant x_0, we are ready to evaluate the left-hand side of Eq. (4) for different y_i. This gives

$$\frac{n'(x,y_i)\cdot n''(x_0,y_i)}{n'(x_0,y_i)} = \begin{cases}(4/3)x+(2/3), & y_1=4,\\ (5/3)x+(2/3), & y_1=5.\end{cases} \tag{5}$$

Applying the Newton interpolation formula Eq. (1) to the constant term and the coefficient of x in Eq. (5), we recover the polynomial dependence on y,

$$\frac{n'(x,y)\cdot n''(x_0,y)}{n'(x_0,y)} = (y/3)x + 2/3 = n(x,y)\cdot C\,.$$

A similar procedure for the denominators yields

$$\frac{d'(x,y)\cdot d''(x_0,y)}{d'(x_0,y)} = (y/3-2/3)x + 4/3 = d(x,y)\cdot C\,.$$

Taking the ratios of the two bivariate polynomial expressions above, we cancel the unknown factor C and succeed in reconstructing the analytic rational function given by the original expression Eq. (3). The used probe points (x, y), not including repeatedly utilized ones as the black box calculation can be cached, involve the following ones,

$$\begin{gathered}(3,4),\ (4,4),\ (5,4),\ (6,4),\\ (3,5),\ (4,5),\ (5,5),\ (6,5),\\ (3,6),\ (3,7),\end{gathered}$$

with 10 probes used in total.

2.4. Zippel method. The Zippel algorithm for polynomials works in the following way. Let us suppose that we have already reconstructed a polynomial $f(x_1,\ldots,x_{k-1},c_0,\ldots)$, where c_0 is some constant value of x_k and the remaining variables also have some fixed values. We would like to run the Newton reconstruction for x_k, therefore we need similar reconstructed polynomials for other values of x_k. One might work in a traditional manner with Newton reconstructions for previous variables, but we want to exploit the sparsity of the polynomial, i.e. the absence of certain monomials in the variables $(x_1,\ldots,x_{k-1})$ under a certain bound on the monomial degrees.

Suppose we take another value c_i of x_k. We consider the already existing polynomial $f(x_1,\ldots,x_{k-1},c_0,\ldots)$ as a skeleton, take all its non-zero monomials and assume that the set of nonzero monomials will remain the same for the yet unknown $f(x_1,\ldots,x_{k-1},c_i,\ldots)$. Here we will leave aside the probability for this assumption to remain valid and how the values of the constants should be chosen. Within this assumption we can write

$$f(x_1,\ldots,x_{k-1},c_i,\ldots) = a_1\cdot x_1^{p_{1,1}}\ldots {x_{k-1}}^{p_{k-1,1}} + \ldots + a_z\cdot x_1^{p_{1,z}}\ldots {x_{k-1}}^{p_{k-1,z}}$$

for some $z = z(k-1)$, where a_i are unknown constant coefficients and p_i are exponents taken from the skeleton.

This is a linear system for a_i and, thus, knowing the values of $f(x_1,\ldots,x_{k-1},c_i,\ldots)$ for z different sets of $\{x_1,\ldots,x_{k-1}\}$ (sampling points) we can solve the system. However, solving this system has a complexity $O(t^3)$ and can grow fast with the number of variables. Thus, the Zippel algorithm for polynomials suggests a specific set of sampling points, i.e. $y_1,\ldots y_{k-1}, y_1^2,\ldots y_{k-1}^2,\ldots, y_1^z,\ldots y_{k-1}^z$. In this case the systems turns into a Vandermonde system which can be solved with complexity $O(z^2)$. The Zippel method consists in solving this system leading to the knowledge of $f(x_1,\ldots,x_{k-1},c_i,\ldots)$ for different i, followed by univariate Newton reconstruction in x_k to obtain $f(x_1,\ldots,x_{k-1},x_k,\ldots)$. Proceeding iteratively, now we know the nonzero monomials in the variables $(x_1,\ldots,x_k)$, and the Zippel algorithm for polynomials can be used to reconstruct the polynomials in these k variables at other numerical values of the remaining constants denoted by the dots. Note that the last variable will be only reconstructed using the Newton method, and we usually choose the last variable to be the spacetime dimension d, as the involved polynomials are typically dense in d.

[2] In principle, the polynomial degree can change if there are accidental cancellations at $x = x_0$, which in practice can be avoided by choosing a large random value of x_0.

2.5. Balanced Zippel method. The known approach in literature to apply the Zippel method to rational functions [27, 28] is based on the homogeneous approach [25], but we are going to suggest a new method which combines the Zippel method with the balanced reconstruction method [13]. As with the polynomial Zippel or balanced Newton methods, we are going to work variable by variable.

Suppose we need to reconstruct a rational function

$$f(x_1,\ldots,x_{k-1},x_k,\ldots)=\frac{n(x_1,\ldots,x_{k-1},x_k,\ldots)}{d(x_1,\ldots,x_{k-1},x_k,\ldots)}, \tag{6}$$

where the dots at the end denote some fixed values of the remaining variables. And suppose we have already reconstructed function

$$f(x_1,\ldots,x_{k-1},c_0,\ldots)=\frac{n'(x_1,\ldots,x_{k-1},c_0,\ldots)}{d'(x_1,\ldots,x_{k-1},c_0,\ldots)}$$

for some fixed value c_0 of x_k. Since some factor can be canceled out after the substitution, we write

$$\begin{aligned} n'(x_1,\ldots,x_{k-1},c_0,\ldots)&=n(x_1,\ldots,x_{k-1},c_0,\ldots)\cdot C\,,\\ d'(x_1,\ldots,x_{k-1},c_0,\ldots)&=d(x_1,\ldots,x_{k-1},c_0,\ldots)\cdot C \end{aligned}$$

for a constant C. Let us calculate the maximal number $z(k-1)$ of non-zero monomials in $n'(x_1,\ldots,x_{k-1},c_0,\ldots)$ and $d'(x_1,\ldots,x_{k-1},c_0,\ldots)$ and assume that for other c_i the set of nonzero monomials remains the same. Also, we calculate the number $t(k)$ of sampling points needed for Thiele reconstruction of the function in the variable x_k for fixed values of other variables.

Now one needs to evaluate the unknown function in $z(k-1)\cdot t(k)$ sampling points $f(y_1^i,\ldots,y_{k-1}^i,c_0^j)$ for some fixed y, with i from 1 to $z(k-1)$ and j from 1 to $t(k)$.

Then, the Thiele reconstruction is performed for each i, resulting in a set of univarite rational functions in x_k:

$$f(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)=\frac{n_i(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)}{d_i(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)}\,.$$

Since some factor can be cancelled, we have

$$\begin{aligned} n_i(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)&=n(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)\cdot h_i\,,\\ d_i(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)&=d(y_1^i,\ldots,y_{k-1}^i,x_k,\ldots)\cdot h_i \end{aligned}$$

for some constant (but dependent on i) coefficient h_i.

Now let us create a balanced expression for the numerator for each of the needed j. We can write

$$\begin{aligned} &\frac{n_i(y_1^i,\ldots,y_{k-1}^i,c_0^j,\ldots)\cdot n'(y_1^i,\ldots,y_{k-1}^i,c_0,\ldots)}{n_i(y_1^i,\ldots,y_{k-1}^i,c_0,\ldots)}=\\ &\qquad=\frac{n(y_1^i,\ldots,y_{k-1}^i,c_0^j,\ldots)\cdot h_i\cdot n(y_1^i,\ldots,y_{k-1}^i,c_0,\ldots)\cdot C}{n(y_1^i,\ldots,y_{k-1}^i,c_0,\ldots)\cdot h_i}=n(y_1^i,\ldots,y_{k-1}^i,c_0^j,\ldots)\cdot C. \end{aligned}$$

This is our unknown numerator multiplied by a constant factor, so now we can run the Zippel polynomial reconstruction for j, obtaining $n(x_1,\ldots,x_{k-1},c_0^j,\ldots)\cdot C$, and then, perform a Newton reconstruction obtaining $n(x_1,\ldots,x_{k-1},x_k,\ldots)\cdot C$. The same procedure is applied to the denominators. After this, the constants get canceled in the ratio and give the desired rational function in $(x_1,\ldots,x_k)$, as in Eq. (6).

It should be also noted that the space-time dimension d can be placed as the last variable, and with a proper master integral choice [36, 37], the denominators get factorized into a polynomial in d and a polynomial in variables other than d. The former polynomial can be found by reconstructing the analytic dependence on other variables at a random numerical value of d, while the latter polynomial can be found by reconstructing the analytic dependence on d at random numerical values of other variables. When the denominator is fully known, the original black box rational function can be multiplied by the denominator to give the value of the numerator for each probe calculation, and the Thiele reconstruction for the last step can be replaced with Newton reconstruction, thereby halving the number of needed sampling points. We refer to this as the *balanced Zippel method with separation*, since the d dependence is separated in the denominator.

3. Benchmark for the number of probe points. Before discussing details of implementations, we present a benchmark for the number of probe points needed for reconstructing a particular rational function. It depends only on the algorithm itself, not how it is implemented in software. We consider the bivariate rational function

$$\frac{(d+13)^{30}\,(y^2+9)^7+1}{(d-4)^{29}\,(y^2-1)^5}\,. \tag{7}$$

This function is designed to mimic the typical coefficients encountered in the IBP reduction in the 4-loop calculation for post-Minkowskian gravitational two-body dynamics in Ref. [38] by the present authors and collaborators. The function depends on a kinematic parameter y and the spacetime dimension parameter d. Although actual IBP reduction coefficients are more complicated, the above test function Eq. (7) preserves the essential features that affect the number of probes needed for reconstruction, including the polynomial degrees of the numerator and denominator in each of the variables, the factorized dependence of the denominator on y and d, and the sparsity pattern in y.[3] Table 2 compares the number of probes needed in the balanced Zippel algorithm presented in this paper, with or without exploiting the factorized dependence of the denominators on d (separation), the number of probes needed in the balanced Newton algorithm, and the number of probes needed in the homogeneous scaling algorithm of Ref. [25].

Table 2. Number of probes (per prime number) needed to reconstruct the bivariate rational function of Eq. (7) using various rational function reconstruction algorithms.

Algorithm	Number of probes per prime
Balanced Zippel Method with Separation	306
Balanced Zippel Method	509
Balanced Newton Method	957
Homogeneous Scaling Method	1424

The number of probes per prime number is presented. The number of prime numbers needed for reconstruction is independent of the rational function reconstruction algorithm, and for this problem, we need 3 prime numbers to obtain a candidate reconstruction result, plus a 4th prime number and a random value of (y, d) used to verify the correctness of the result. Looking at Table 2 from the last row to the first row, we can see that as sparsity information is taken into account, first by switching on the balanced algorithm, then by switching on the Zippel algorithm, and finally by using the factorized nature of denominators, the number of probe points keeps decreasing, eventually reaching less than one quarter of the number required by the crudest algorithm.

4. Sequential implementation. One might invent an optimal algorithm, but it means nothing without a proper implementation. Before proceeding to parallelization, one should consider an optimized sequential implementation, since any type of parallelization makes the codebase more complex, and one should first understand which parts of the algorithm need this parallelization — there is no reason to implement a parallel version of something that takes a small part of total time.

Within the FIRE framework we have a possibility to run reduction for different sampling points at the same time for more than 5 years [6] with MPI parallelization on a supercomputer, so currently we focus on the reconstruction algorithm that comes after the IBP runs.

While implementing a sequential version of the balanced Zippel method and other reconstruction methods, we obviously had to choose a library to work with rational functions (we consider the x86-64 as our primary architecture, but the code can also work on ARM). First of all, we wrote a version using our FUEL framework [39] which allowed us to use different libraries and compare their speed. Since we were interested in the case of 5–6 variables, the only libraries that scale well to such a number of variables turned out to be competitors, specifically FLINT [40] and Symbolica [41]. While the performance turned out to be similar, FLINT has a significant advantage being completely open-source and free for use.

[3] In the problem of Ref. [38], all IBP coefficients have either an even parity or an odd parity under $y \to -y$. Also, the polynomial degrees in d and y (in either the numerator or denominator) are uncorrelated; for example, monomials with a higher power of d do not have a lower maximum power of y, as in the test function Eq. (7).

Preliminary tests integrating reconstruction into the FIRE framework showed that the reconstruction takes a significant part of the total time (the other part being IBP runs), thus we decided to optimize our approach by removing the middle layer FUEL and working directly with FLINT objects. A number of optimization steps were performed to reach a satisfactory performance with properly chosen FLINT structures. Since the reconstruction step can be easily separated from the other part of the reduction, it could be profiled well with Valgrind and Google Perf.

As a result, we obtained a stand-alone binary that accepts a needed number of FIRE tables and runs the reconstruction producing a table with reconstructed coefficients. We are aware of the FireFly library [27, 28] for modular and rational reconstruction, but we could not adapt it to our needs. First of all, we invented a new algorithm that would require a FireFly modification. But perhaps even more importantly, we have a different ideology — the FireFly reconstruction itself can be the main command and it can request other program to create sampling points running reduction, possibly with MPI. However, in our approach, we start from reduction and decide what reconstruction to call based on reduction results. The reconstruction algorithms might be called on individual nodes, so the reconstruction cannot be the master MPI task for us.

The basic FIRE algorithm to obtain the result with the modular approach can be described in the following way:

Algorithm 1. Modular approach to reduction

```
1:  **input** Required reduction information, variables list $x_1, \dots, x_n$
2:  Fix initial variables values $y_1, \dots, y_n$ and first large prime $p_1$
3:  **for** $k = 1 \dots n$ **do**
4:    **for** $j = 1 \dots \infty$ **do**
5:      Run reduction for $y_1, \dots, y_k^j, \dots, y_n$
6:      Try Thiele reconstruction by $y_k$ modular $p_1$
7:      **if** reconstructed **goto** 9
8:    **end for**
9:    record the number of sampling points $t_j$ needed for Thiele reconstruction by $x_j$
10: **end for**
11: **for** $p = p_1 \dots$ **do**
12:   Seed $t_1$ sampling points $y_1^j, y_2, \dots, y_n$ for $j = 1..t_1$ and run reductions
13:   Thiele-reconstruct by $x_1$ modular $p$ (these two steps can be skipped for $p_1$ since obtained ealier)
14:   **for** $k = 2 \dots n$ **do**
15:     Having a reconstructed function by $x_1 \dots x_{k-1}$ record the maximum number $z_{k-1}$ of nonzero monomials in
        its numerator and denominator
16:     Run reduction for $y_1^i, \dots, y_{k-1}, y_k^j, y_{k+1} \dots, y_n$ for $i = 1 \dots z(k-1)$ and $j = 1 \dots t(k)$ modular $p$
17:     Run Thiele reconstruction by $x_k$ for $i = 1 \dots z(k-1)$
18:     Run balanced Zippel reconstruction to obtain reconstructed function by $x_1 \dots x_k$
19:   **end for**
20:   **if** the reconstruction can be performed from modular fields to rational numbers **goto** 22
21: **end for**
22: **output** the reconstructed function
23: **stop**
```

5. Conventional parallelization. Each parallelization attempt (currently we consider x86-64 or ARM architecture with multiple cores and shared-memory) should first detect which parts of the algorithm really need this parallelization (normally time-consuming parts) and which ones can be parallelized (depends on the algorithm, should be a separate study). We have analyzed time-consuming parts of the algorithm 1 and located two major blocks: the reduction for particular sampling points and the reconstruction algorithm.

5.1. Optimization of the reduction algorithm. The reduction algorithm for particular sampling points has been described in previous papers and is generally solving a sparse linear system of equations, in this case over a finite field. As an output, this algorithm saves a file to disk with “tables” in the FIRE format for reduction results. The table files have a prefix indicating the values of variables and the prime modulus. Thus, the reconstruction can be called later as a separate process relying on tables saved to disk earlier.

While the discussion of optimal methods to solve such systems stands outside the scope of this article, we have also measured the performance of individual reduction and noticed that the used time can be split into two categories — the arithmetic operations with coefficients and what can be considered as overhead costs organizing the process of solving. The time for these categories turned out to be comparable with each other (which was not clear in advance), therefore we have decided to develop a version of FIRE working not with one sampling point (variable values) but with a set of sampling points. This approach has allowed us to reduce the overhead costs, since they remain constant no matter how many sampling points we solve together. For example, for 16 simultaneous sampling points, the reduction time turns out to be only about 4 times longer.

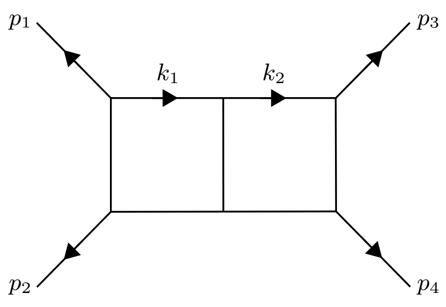

Figure 1. The double box diagram.

We test an easy problem of the reduction of a double box integral with a rank-2 numerator (i.e. degree 2 in the irreducible scalar products), $(k_2 + p_1)^2(k_1 - p_3)^2$, as well as harder problems of the reduction of integrals with higher-rank numerators. The double box diagram is shown in Fig. 1. The kinematic variables are

$$(p_1 + p_2)^2 = s, \quad (p_1 + p_3)^2 = t\,.$$

The two irreducible scalar products are chosen as

$$(k_2 + p_1)^2, \quad (k_1 - p_3)^2\,.$$

The results are shown in Table 3.

5.2. Reconstruction optimization. The reconstruction algorithm reads the required table files and runs the reconstruction, then saves the result to another file. There are two obvious sources for conventional parallelization that we implemented.

First of all, reading tables can be performed in parallel. Second, tables normally contain a number of coefficients that need reconstruction, so reconstructing different coefficients can be performed independently. Both steps are parallelized with a simple OpenMP approach adding only a few `pragma omp` directives to the C++ code. While being simple, this method is quite efficient with an almost linear performance gain.

6. Advancing to supercomputers. While with the conventional parallelism it is hard to predict the complexities to be encountered before an implementation, with supercomputers it is almost impossible to predict them. For the modular IBP reduction part, the first thing we had to implement was an efficient MPI distribution of reduction jobs. While being more or less straightforward, this requires the ability to handle a number of situations that can happen if

- one of reduction jobs crashes due to a node failure;
- the job is killed due do an MPI failure;
- the whole job is killed due to a time limit of the cluster's job scheduler;
- some of the table files became corrupted due to one of those failures.

When dealing with massive MPI jobs, one has to keep in mind that any strange event that can occur will eventually happen. Since jobs get killed due to hardware and software problems not related to your code, they will get eventually killed at any possible place and the algorithm has to be able to recover from this place when restarted later.

The reconstruction turned out to be a lengthy step too when performed on one node while the reductions are performed on multiple nodes. To overcome this complication, we had to implement an MPI version of

Table 3. Benchmark results of IBP reductions for double box integrals in the modular approach for integrals with different numerator powers. FIRE7p performs IBP reduction for one sampling point at a time, and FIRE7mp performs IBP reduction for a number (chosen to be 64) of sampling points simultaneously.

Numerator power	FIRE7p time	FIRE7p time × 64	FIRE7mp with 64 values	Efficiency gain
2	4.2	268.8	110.2	2.43
4	5.2	332.8	122.3	2.72
6	8.6	550.4	144.2	3.81
8	19.4	1241.6	214.0	5.80

the balanced Zippel reconstruction algorithm. Initially we tried to read the files with tables on one node and distribute coefficients that need to be reconstructed to other nodes, but this led to a dramatic performance degradation due to heavy MPI communication.

Thus, we had to switch to an approach where multiple nodes participate in reading tables, reconstructing a part of the data they read, and saving the tables. In our first implementation, all nodes read all the tables files, and it worked perfectly until approaching a million sampling points.

6.1. Filesystem overload. This is one of the cases where we had to adapt to a particular supercomputer or at least to supercomputers using the same filesystem type and version. The Lomonosov-2 supercomputer [1] uses the OSS Lustre distributed filesystem (`https://wiki.lustre.org/Main_Page`) which appears to be heavily loaded when the number of files in a folder becomes large enough (an order of 50 thousand files or more). On the other hand, it works well when files are put into subfolders. When loading the filesystem heavily we even had crashes, and the communication with the support staff showed that the filesystem produced error messages like `layout.c:2121:__req_capsule_get()) @@@ Wrong buffer for field 'niobuf_inline' (7 of 7) in format 'LDLM_INTENT_OPEN',...`. Partially the problem was solved when we switched to reduction in multiple sampling points at the same time, which decreased the number of tables on disk, but the challenge came back for problems of a larger scale.

A possible co-design solution here would be to upgrade the filesytem to a version 2.14 of Lustre, but that would require us to stop the calculation on the whole supercomputer for a couple of weeks. Therefore we switched to adapting the code to change the way files are stored. If the total number of the files is not essential, we can split the table files into subfolders, so that depending on the maximal variable exponent in table names and the subfolder size limit, the table is placed in a corresponding subfolder.

Also we had to change the format of tables files to allow MPI to handle them without overloading the filesystem. The reason is that MPI has a nice set of functions allowing parallel file read, where various nodes read different parts of a file. In this case the file reading internally is handled by the MPI communication with special filesystem API that does not exist on personal computers or clusters without a distributed filesystem. Thus reorganizing the files, so that different coefficients could be read by different nodes, allowed us to use these functions and decrease the filesystem load.

These two steps allowed us to overcome the filesystem limitations without upgrading it to a new version.

6.2. Bugs in MPI and Singularity. When exceeding a million values needed for balanced Zippel reconstruction on the Lomonosov-2 supercomputer, we again encountered crashes happening during reconstruction. While the previous crashes happened during table loading, the new ones took place at some random moments after tables had been succesfully loaded. The stack trace has always shown a problem in MPI_Barrier like

```
libc.so.6(+0x36280) [0x7ffff4f21280]
mca_btl_openib.so(+0x1546d) [0x7fffe937346d]
libopen-pal.so.40(opal_progress+0x2c) [0x7ffff45875dc]
mca_pml_ob1.so(mca_pml_ob1_recv+0x2a5)[0x7fffe27301c5]
libmpi.so.40(ompi_coll_base_barrier_intra_tree+0x17e)[0x7ffff5ab9f0e]
libmpi.so.40(MPI_Barrier+0xa8)[0x7ffff5a6f278]
```

with following trace inside FIRE. A discussion with the support staff has led to a conclusion that there seemed to be a rare bug in the version of libc installed on the supercomputer, but it could be updated without stopping the whole supercomputer for a long period.

The solution we came together was to install the Singularity [42] modules on the supercomputer, enabling the use of a virtualization container layer to host FIRE builds inside the container. Thus, FIRE could use a newer version of libc. The modules were installed and it allowed us to proceed to larger numbers of sample points for Zippel reconstruction.

6.3. Limits and benchmarks. The Zippel reconstruction still remains one of the most problematic points for the whole process. We already had to optimize it in a way that each coefficient could be reconstructed on a separate node, and each node used OpenMP parallelization to engage all cores to handle this reconstruction in parallel. This way, we could run a reconstruction job requesting around 1.7 million points for the final Zippel call. This took about 16 hours using 210 cores (15 nodes with 14 cores per each). However, the problem with the Zippel algorithm is that it is quadratic in complexity by the number of points, and the reconstruction job currently needs around 5 million points and does not fit in the time limit for a job on the Lomonosov-2 supercomputer.

While we cannot reveal the details of this calculation, we are going to provide some supercomputer benchmarks on an already well-known (but still complex) reduction problem (see e.g. Ref. [32]) of a penta-box integral with a rank-5 numerator. The penta-box diagram is shown in Fig. 2. There are five external kinematic variables:

$$(p_1+p_2)^2 = s_{12}, \quad (p_2+p_3)^2 = s_{23}, \quad (p_3+p_4)^2 = s_{34}, \quad (p_4+p_5)^2 = s_{45}, \quad (p_5+p_1)^2 = s_{51}\,.$$

The irreducible scalar products are chosen to be

$$(k_1+p_3)^2, \quad (k_1+p_5)^2, \quad (k_2+p_1)^2$$

and the actual numerator of the reduced integral is

$$[(k_1+p_5)^2]^3[(k_2+p_1)^2]^2\,.$$

The finite-field version of FIRE reduces the above integral to 62 master integrals (using only IBP identities and ignoring symmetry identities). For the purpose of the demonstration, we split the master integrals into levels (the number of positive indices, i.e. uncanceled propagators) resulting in levels from 3 to 8. When targeting master integrals at a particular level, all other master integrals are set to zero in the IBP reduction process. While this reduction could be handled as a whole, splitting into levels is a good demonstration of how the complexity grows when trying to obtain coefficients of master integrals at lower levels, i.e. with fewer propagators. We ran the reduction on 210 cores split into 15 nodes with 14 cores in each node. We used the test queue for fast job submissions having a 15 minutes time limit. Some of the levels had to be restarted multiple times after exceeding the time limit. (The algorithm is able to restart from the point the previous run was interrupted.) We used 64 points (in kinematic and dimension values and the modulus) per FIRE7mp run. The results are shown in Table 4.

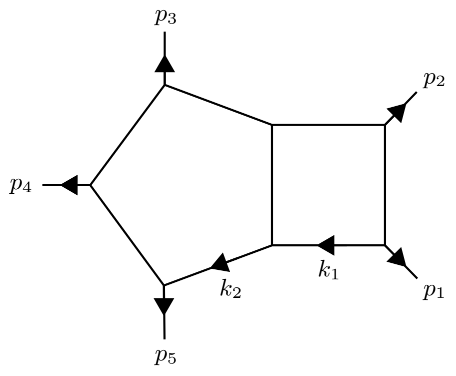

Figure 2. The penta-box diagram.

While for these tests the Zippel time is still quite small, the quadratic growth of speed will become a problem at larger numbers, requiring further work in future.

7. Conclusion. We have described a new algorithm of the balanced Zippel reconstruction and how it fits in the general reduction scheme of Feynman integrals in the modular approach. We have used this algorithm and its implementation as an example to demonstrate a co-design approach of an algorithm and supercomputers. We plan to further improve the performance with all sources of parallelism including the vectorization of instructions. For the Zippel algorithm we also plan a GPU implementation. We intend to publish a paper describing a public release of the new FIRE 7 next year.

Table 4. Benchmark results for the pentabox example using FIRE7mp which simultaneously calculates 64 numerical probes in each run, working on 210 cores. The column *Level* gives the number of propagators in every master integral that are not set to zero in the run. *Time* gives the wallclock time in seconds. *Primes* represents the number of primes needed for reconstruction. *Runs* gives the number of FIRE7mp runs, each of which produces 64 numerical probes. *Zippel* provides the size of the Vandermonde matrix involved in the last Zippel reconstruction step for reconstructing analytic dependence on kinematic variables at each numerical value of the spacetime dimension. The last column compares the IO time and the actual computation time during Zippel reconstruction, in seconds.

Level	Time	Primes	Runs	Zippel	Zippel load / reconstruction time
8	19	1	94	324	0.97 / 0.01
7	29	1	395	1724	1.29 / 0.03
6	107	1	1348	5447	1.79 / 0.91
5	1689	2	8792	16896	3.40 / 7.60
4	4358	2	26886	27443	3.51 / 21.49
3	14968	2	29129	49158	8.41 / 68.70

Appendix 1
Toy example for Feynman integrals and integration-by-parts identities

While the paper presents a general algorithm for reconstructing rational functions from numerical black box evaluations, our primary application is the simplification of Feynman integrals by IBP identities. For readers without prior knowledge in the latter topic, this section presents a toy example of a family of Feynman integrals and linear algebraic relations, known as integration-by-parts (IBP) identities, between these integrals, to give a glimpse of the problem.

The toy example is the following family of integrals I_ν, parameterized by a positive integer parameter ν, in D dimensional Euclidean space

$$\int d^D\boldsymbol{k}\,\frac{1}{(\boldsymbol{k}^2+m^2)^\nu}\,, \tag{8}$$

where $\boldsymbol{k}^2 \equiv \sum_{i=1}^{D} \boldsymbol{k}_i\boldsymbol{k}_i$. Such integrals are commonly referred to as "tadpole integrals" in the physics literature. IBP identities follow from the fact that the integral of a total derivative is zero,

$$0 = \int d^D\boldsymbol{k}\,\frac{\partial}{\partial \boldsymbol{k}_i}\frac{\boldsymbol{k}_i}{(\boldsymbol{k}^2+m^2)^\nu}\,, \tag{9}$$

where the repeated index i is implicitly summed from 1 to D in the "Einstein summation convention". The boundary terms vanish in *dimensional regularization* commonly used in theoretical physics; we refer readers to the literature regarding the related subtleties. Evaluation of Eq. (9) and application of partial fractioning gives for any positive integer ν

$$0 = (D-2\nu)I_\nu + 2m^2\nu I_{\nu+1}\,.$$

It is a set of linear algebraic identities relating I_ν with different values of ν. In this simple case, it is a set of recursion relations which allows any I_ν to be expressed in terms of I_1, the *master integral*, multiplied by a coefficient that is a rational function of the spacetime dimension D and the kinematic parameter m^2. Dimensional regularization allows us to consider generic non-integer values of D which prevents possible singularities in solving the recursion relations.

More complicated Feynman integrals have more distinct denominators than those appearing in the RHS of Eq. (8). For example, the family of "bubble integrals" in Euclidean space is defined as

$$\int d^D\boldsymbol{k}\,\frac{1}{(\boldsymbol{k}^2+m^2)^{\nu_1}((\boldsymbol{k}+\boldsymbol{p})^2+m^2)^{\nu_2}}$$

for a fixed vector $\boldsymbol{k}$, known as the external momentum in physics, and integers ν_1 and ν_2, at least one of which must be positive. The IBP identities in this case give relations between such integrals with different values of (ν_1, ν_2) with the same $\boldsymbol{k}$. The solutions of the IBP identities are then functions of the spacetime dimension D and kinematic parameters m^2 and $\boldsymbol{p}^2$. For Feynman integrals involved in state-of-the-art research, there can be nearly 10 or even more than 20 different distinct denominators, and the integral family is parametrized by a corresponding number of distinct integers. A large number of IBP identities is generally needed. It is often impractical to identify a recursion structure to directly solve the identities in a simple manner; rather, we solve all identities together as a large linear system, which is known as the Laporta algorithm [3]. This is computationally intensive, and the solutions can contain very complicated rational functions. The challenges motivate our study of efficient algorithms for reconstructing multivariate rational functions from numerical finite-field valuations, in this case the numerical solution of the linear system of IBP identities.

References

1. Vl. V. Voevodin, A. S. Antonov, D. A. Nikitenko, et al., “Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community,” Supercomput. Front. Innov. **6** (2), 4–11 (2019). doi 10.14529/jsfi190201.
2. K. G. Chetyrkin and F. V. Tkachov, “Integration by Parts: The Algorithm to Calculate β Functions in 4 Loops,” Nucl. Phys. B **192** (1), 159–204 (1981). doi 10.1016/0550-3213(81)90199-1.
3. S. Laporta, “High-Precision Calculation of Multiloop Feynman Integrals by Difference Equations,” Int. J. Mod. Phys. A **15** (32), 5087–5159 (2000). doi 10.1142/S0217751X00002159.
4. A. V. Smirnov and A. V. Petukhov, “The Number of Master Integrals is Finite,” Lett. Math. Phys. **97** (1), 37–44 (2011). doi 10.1007/s11005-010-0450-0.
5. C. Anastasiou and A. Lazopoulos, “Automatic Integral Reduction for Higher Order Perturbative Calculations,” J. High Energy Phys. No. 7, Article Number 046 (2004). doi 10.1088/1126-6708/2004/07/046.
6. A. V. Smirnov and F. S. Chukharev, “FIRE6: Feynman Integral REduction with Modular Arithmetic,” Comput. Phys. Commun. **247**, Article Number 106877 (2020). doi 10.1016/j.cpc.2019.106877.
7. C. Studerus, “Reduze — Feynman Integral Reduction in C++,” Comput. Phys. Commun. **181** (7), 1293–1300 (2010). doi 10.1016/j.cpc.2010.03.012.
8. P. Maierhöfer, J. Usovitsch, and P. Uwer, “Kira — A Feynman Integral Reduction Program,” Comput. Phys. Commun. **230**, 99–112 (2018). doi 10.1016/j.cpc.2018.04.012.
9. J. Klappert, F. Lange, P. Maierhöfer, and J. Usovitsch, “Integral Reduction with Kira 2.0 and Finite Field Methods,” Comput. Phys. Commun. **266**, Article Number 108024 (2021). doi 10.1016/j.cpc.2021.108024.
10. R. N. Lee, “LiteRed 1.4: A Powerful Tool for Reduction of Multiloop Integrals,” J. Phys. Conf. Ser. **523**, Article Number 012059 (2014). doi 10.1088/1742-6596/523/1/012059.
11. Z. Wu, J. Boehm, R. Ma, et al., “NeatIBP 1.0, a Package Generating Small-Size Integration-by-Parts Relations for Feynman Integrals,” Comput. Phys. Commun. **295**, Article Number 108999 (2024). doi 10.1016/j.cpc.2023.108999.
12. X. Guan, X. Liu, Y.-Q. Ma, and W.-H. Wu, “Blade: A Package for Block-Triangular Form Improved Feynman Integrals Decomposition,” arXiv:2405.14621 [hep-ph]. doi 10.48550/arXiv.2405.14621.
13. A. V. Belitsky, A. V. Smirnov, and R. V. Yakovlev, “Balancing Act: Multivariate Rational Reconstruction for IBP,” Nucl. Phys. B **993**, Article Number 116253 (2023). doi 10.1016/j.nuclphysb.2023.116253.
14. M. Monagan, “Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction,” Proc. Int. Symp. on Symbolic and Algebraic Computation (2004). 243–249. doi 10.1145/1005285.1005321.
15. M. Ben-Or and P. Tiwari, “A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation,” Proc. 20 Ann. ACM Symp. on Theory of Computing (1988). 301–309. doi 10.1145/62212.62241.
16. R. Zippel, “Interpolating Polynomials from Their Values,” J. Symbolic Comput. **9** (3), 375–403 (1990). doi 10.1016/S0747-7171(08)80018-1.
17. D. Grigoriev, M. Karpinski, and M. F. Singer, “Computational Complexity of Sparse Rational Interpolation,” SIAM J. Comput. **23** (1), 1–11 (1994). doi 10.1137/S0097539791194069.
18. E. Kaltofen, “Greatest Common Divisors of Polynomials Given by Straight-Line Programs,” J. ACM **35** (1), 231–264 (1988). doi 10.1145/42267.45069.
19. E. Kaltofen and B. M. Trager, “Computing with Polynomials Given by Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators,” J. Symbolic Comput. **9** (3), 301–320 (1990). doi 10.1016/S0747-7171(08)80015-6.
20. E. Kaltofen and Z. Yang, “On Exact and Approximate Interpolation of Sparse Rational Functions,” Proc. Int. Symp. on Symbolic Algebraic Computation (2007). 203–210. doi 10.1145/1277548.1277577.
21. J. de Kleine, M. Monagan, and A. Wittkopf, “Algorithms for the Non-monic Case of the Sparse Modular GCD Algorithm,” Proc. Int. Symp. on Symbolic Algebraic Computation (2005). 124–131. doi 10.1145/1073884.1073903.
22. A. Diaz and E. Kaltofen, “FOXBOX: A System for Manipulating Symbolic Objects in Black Box Representation,” Proc. Int. Symp. on Symbolic Algebraic Computation (1998). 30–37. doi 10.1145/281508.281538.
23. E. Kaltofen, W.-s. Lee, and A. A. Lobo, “Early Termination in Ben-Or/Tiwari Sparse Interpolation and a Hybrid of Zippel’s Algorithm,” Proc. Int. Symp. on Symbolic Algebraic Computation (2000). 192–201. doi 10.1145/345542.345629.

24. Q.-L. Huang and X.-S. Gao, "Sparse Polynomial Interpolation with Finitely Many Values for the Coefficients," in *Lecture Notes in Computer Science* (Springer, Cham, 2017), Vol. 10490, pp. 196–209. doi 10.1007/978-3-319-66320-3_15.
25. T. Peraro, "Scattering Amplitudes over Finite Fields and Multivariate Functional Reconstruction," J. High Energy Phys. No. 12, Article Number 030 (2016). doi 10.1007/JHEP12(2016)030.
26. T. Peraro, "FiniteFlow: Multivariate Functional Reconstruction Using Finite Fields and Dataflow Graphs," J. High Energy Phys. No. 7, Article Number 031 (2019). doi 10.1007/JHEP07(2019)031.
27. J. Klappert and F. Lange, "Reconstructing Rational Functions with FireFly," Comput. Phys. Commun. **247**, Article Number 106951 (2020). doi 10.1016/j.cpc.2019.106951.
28. J. Klappert, S. Y. Klein, and F. Lange, "Interpolation of Dense and Sparse Rational Functions and Other Improvements in FireFly," Comput. Phys. Commun. **264**, Article Number 107968 (2021). doi 10.1016/j.cpc.2021.107968.
29. M. S. Floater and K. Hormann, "Barycentric Rational Interpolation with no Poles and High Rates of Approximation," Numer. Math. **107** (2), 315–331 (2007). doi 10.1007/s00211-007-0093-y.
30. A. Maier, "Scaling up to Multivariate Rational Function Reconstruction," arXiv:2409.08757v1 [hep-ph]. doi 10.48550/arXiv.2409.08757.
31. P. S. Wang, "A p-Adic Algorithm for Univariate Partial Fractions," Proc. Fourth ACM Symposium on Symbolic and Algebraic Computation (1981). 212–217. doi 10.1145/800206.806398.
32. S. Abreu, J. Dormans, F. Febres Cordero, et al., "Analytic Form of Planar Two-Loop Five-Gluon Scattering Amplitudes in QCD," Phys. Rev. Lett. **122** (8), Article Number 082002 (2019). doi 10.1103/PhysRevLett.122.082002.
33. G. De Laurentis and B. Page, "Ansätze for Scattering Amplitudes from p-Adic Numbers and Algebraic Geometry," J. High Energy Phys. **2022**, Article Number 140 (2022). doi 10.1007/JHEP12(2022)140.
34. H. A. Chawdhry, "p-Adic Reconstruction of Rational Functions in Multiloop Amplitudes," Phys. Rev. D **110** (5), Article Number 056028 (2024). doi 10.1103/PhysRevD.110.056028.
35. X. Liu, "Reconstruction of Rational Functions Made Simple," Phys. Lett. B **850**, Article Number 138491 (2024). doi 10.1016/j.physletb.2024.138491.
36. A. V. Smirnov and V. A. Smirnov, "How to Choose Master Integrals," Nucl. Phys. B **960**, Article Number 115213 (2020). doi 10.1016/j.nuclphysb.2020.115213.
37. J. Usovitsch, "Factorization of Denominators in Integration-by-Parts Reductions," arXiv:2002.08173v2 [hep-ph]. doi 10.48550/arXiv.2002.08173.
38. Z. Bern, E. Herrmann, R. Roiban, et al., "Amplitudes, Supersymmetric Black Hole Scattering at $\mathcal{O}(G^5)$, and Loop Integration," arXiv:2406.01554v1[hep-th]. doi 10.48550/arXiv.2406.01554.
39. K. S. Mokrov, A. V. Smirnov, and M. Zeng, "Rational Function Simplification for Integration-by-Parts Reduction and Beyond," Numerical Methods and Programming **24** (4), 352–367 (2023). doi 10.26089/NumMet.v24r425.
40. FLINT: Fast Library for Number Theory. 2023. Version 3.0.0. https://flintlib.org.
41. B. Ruijl, *Symbolica: Modern Computer Algebra*. https://symbolica.io.
42. G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific Containers for Mobility of Compute," PLoS ONE **12** (5), Article Number e0177459 (2017). doi 10.1371/journal.pone.0177459.

Received
October 8, 2024

Accepted for publication
November 21, 2024

Information about the authors

Alexander V. Smirnov — Dr. Sci., Laboratory head; Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, building 4, 119234, Moscow, Russia.

Mao Zeng — Ph.D., Royal Society University Research Fellow; Higgs Centre for Theoretical Physics, University of Edinburgh, James Clark Maxwell Building, Peter Guthrie Tait Road, EH9 3FD, Edinburgh, United Kingdom.