`FeynGKZ`: A *Mathematica* package for solving Feynman integrals using GKZ hypergeometric systems ☆,☆☆

B. Ananthanarayan, Sumit Banik, Souvik Bera, Sudeepan Datta *

Centre for High Energy Physics, Indian Institute of Science, Bangalore, 560012, Karnataka, India

ARTICLE INFO

ABSTRACT

In the Lee-Pomeransky representation, Feynman integrals can be identified as a subset of Euler-Mellin integrals, which are known to satisfy Gel'fand-Kapranov-Zelevinsky (GKZ) system of partial differential equations. Here we present an automated package to derive the associated GKZ system for a given Feynman diagram and solve it in terms of hypergeometric functions using two equivalent algorithms, namely the triangulation method and the Gröbner deformation method. We present our code in the form of a *Mathematica* package `FeynGKZ.wl` which requires the softwares `polymake`, `Macaulay2` and `TOPCOM`, and the packages `AMBRE` and `Olsson.wl` as dependencies. As applications of the package, we find series solutions to the GKZ systems of several one-loop and two-loop Feynman integrals. These are included in the file `Examples.nb` that can be downloaded along with the package from GitHub.

Program summary

References

[1] Wolfram *Mathematica*; proprietary software.
[2] polymake, open-source software.
[3] Macaulay2, open-source software.
[4] TOPCOM, open-source software.
[5] AMBREv2.1.1.m, open-source *Mathematica* package.
[6] Olsson.wl, open-source *Mathematica* package.

---

1. Introduction

The evaluation of multi-loop multi-scale Feynman integrals at ever increasing levels of complexity, has been of great interest and importance as they are the cornerstone of high precision theoretical calculations to match the precision level of present and future colliders. In particular, analytical results for these are of great importance, even if in special cases, as they would offer interesting insights as well as useful cross-checks with the corresponding numerical results. For an overview on the different analytical and numerical techniques, see [1] and [2].

In dimensional regularization, Feynman integrals can be expressed in terms of multiple hypergeometric functions [3] which are widely known objects in mathematics literature. Several methods have been developed to solve Feynman integrals in terms of hypergeometric functions. One of the most popular being the Mellin-Barnes (MB) method [4–6]. Here we focus on another powerful method which is based on the work of Gel'fand, Kapranov, and Zelevinsky (GKZ) [7–10] which explores a system of partial differential equations (PDEs) whose solutions are hypergeometric functions and Euler-Mellin (EM) integrals. While studying the differential equations satisfied by Feynman integrals, Kalmykov and Kniehl were among the first to point out [11], based on the work of Beukers [12], that Feynman integrals could in all likelihood be solutions of an underlying GKZ system.

After the discovery of the Lee-Pomeransky (LP) representation [13] of the Feynman integral, the connection between Feynman integrals and GKZ systems is more straightforward. This has been explicitly shown in [14,15], where the associated $\mathcal{A}$-matrix of the GKZ system can be constructed from the G-polynomial of the LP representation. The GKZ system can also be derived from the PDE satisfied by the MB representation of the Feynman integral as discussed in [16–18]. The derived GKZ system can then be used to study different analytic properties of the original Feynman integral. For example, one can derive series solutions using two alternative algorithms as discussed in [14] and [15]. Alternatively, the GKZ framework is also useful to study the singular locus of Feynman integrals [19], vector spaces generated by Feynman integrals [20], Cohen-Macaulay property of Feynman integrals [21], matroids attached to Feynman integrals [22], etc. GKZ theory, along with ideas from studies of Calabi-Yau manifolds, has been useful in understanding the analytic structure of a certain class of amplitudes to all orders of perturbation [23,24]. It is also noted that the PDEs of the GKZ system is closely related to the PDEs appearing in the Yangian bootstrap approach [25]. Furthermore, due to the connection between GKZ systems and D-module theory, one can efficiently derive first-order differential equations [26] satisfied by a basis of master integrals of a Feynman integral. The realisation of parametric Feynman integrals as EM integrals has also of late provided a novel computational algorithm to numerically evaluate Euclidean Feynman integrals, employing methods inspired from tropical geometry [27].

Despite recent rapid progress in the theoretical side of these emerging topics, there are no publicly available automated computational tools for the study of GKZ systems associated to Feynman integrals. We take the first step in this direction by presenting the *Mathematica* package `FeynGKZ.wl` which can derive the GKZ system satisfied by a given Feynman integral. Subsequently, we find the series solutions of the derived GKZ system using two equivalent algorithms. One of these relies on Gröbner deformations (GD) which loosely speaking is a generalization of the method of Frobenius for ODEs in a single variable, to PDEs in several variables [28]. The other technique relies on taking suitable triangulations of the polytope defined by a certain point configuration. It may be noted that both methods yield solutions in terms of multivariate hypergeometric functions. `FeynGKZ.wl` uses `Macaulay2` and `TOPCOM` [29] for intermediate steps in the GD and triangulation method, respectively. It also uses `polymake` [30] to calculate the normalized volume of the Newton polytope associated to the $\mathcal{A}$-matrix of the GKZ system. The *Mathematica* packages `AMBRE` [31] and `Olsson.wl` [32] are used as well to derive MB representations for scalar Feynman integrals and to express double-sums in terms of known hypergeometric functions, respectively.

The outline of the paper is as follows. In section 2, we provide an overview of the GKZ hypergeometric systems. In section 3, we briefly recap scalar Feynman integrals and their LP representation. In section 4, we recall how to derive the GKZ system from the LP representation and find series solutions using either the GD or triangulation approach. In Section 5, we discuss the usage of the external modules of `FeynGKZ.wl`. In section 6 we demonstrate the package by taking a simple one-loop massive Feynman integral. Finally, in Section 7 we present our conclusions and remarks. In appendices A and B we provide the usage of some additional external modules and global variables of the package, respectively. In appendix C, we list the Feynman integrals we have demonstrated in the file `Examples.nb` which can be downloaded along with the package `FeynGKZ.wl` from https://github.com/anant-group/FeynGKZ.

2. Overview of GKZ hypergeometric systems

In this section, we provide a brief overview of the GKZ hypergeometric system. To keep the mathematical framework short, we provide only the notions required for the computational aspects. The full theory requires more rigorous mathematical ideas which are not required for our purpose. The interested reader will find those in the following [28,33–36]

2.1. Setting up the GKZ system

Let $\mathcal{A} = a_{ij}$ be a $(n + 1) \times N$ matrix with $n + 1 \leq N$, having rank $n + 1$ and all elements are integers. We assume that the vector $(1, 1, \ldots, 1)$ lies in the $\mathbb{Q}$-row span of $\mathcal{A}$ and the column vectors of $\mathcal{A}$ generate $\mathbb{Z}^{n+1}$ over $\mathbb{Z}$. The toric ideal associated with the matrix $\mathcal{A}$ is defined as,

$$I_{\mathcal{A}} := \langle \partial^u - \partial^v | \mathcal{A}u = \mathcal{A}v \; ; \; u, v \in \mathbb{N}^N \rangle \in \mathbb{C}[\partial] \tag{1}$$

where $\mathbb{C}[\partial] = \mathbb{C}[\partial_1, \ldots, \partial_N]$ is a commutative polynomial ring.

Given a column vector $\beta = (\beta_0, \ldots, \beta_n)^T \in \mathbb{C}^{n+1}$, the GKZ hypergeometric system characterised by matrix $\mathcal{A}$ and parameters β is defined as the left ideal $H_{\mathcal{A}}(\beta)$ generated by $I_{\mathcal{A}}$ and $\langle \mathcal{A} \cdot \theta - \beta \rangle$.

$$H_{\mathcal{A}}(\beta) = I_{\mathcal{A}} \cup \langle \mathcal{A} \cdot \theta - \beta \rangle \in \mathbb{C}[z_1, \ldots, z_N, \partial_1, \ldots, \partial_N] \tag{2}$$

where the vector $\theta = (\theta_1, \ldots, \theta_N)^T$ contains the Euler operators $\theta_i = z_i \partial_i$ as its elements. This is also known as the $\mathcal{A}$-hypergeometric ideal due to the presence of the matrix $\mathcal{A}$. A holomorphic function $\phi(z_1, \ldots, z_N)$ is called the $\mathcal{A}$-hypergeometric function if $H_{\mathcal{A}}(\beta) \bullet \phi = 0$. Here we are only interested in non-logarithmic series solutions of the above system.

2.2. EM integrals as solutions to GKZ hypergeometric systems

It is pertinent here to discuss the solutions to GKZ systems in terms of EM integrals, for we shall later see that a generalized version of Feynman integrals could be understood in terms of such solutions.

To define the EM integral, we consider a generic Laurent polynomial:

$$f(\alpha_1, ..., \alpha_n; z_1, ..., z_N) \equiv f_z(\alpha) := \sum_{j=1}^{N} z_j \prod_{i=1}^{n} \alpha_i^{a_{ij}}; \quad a_{ij} \in \mathbb{Z} \tag{3}$$

We can now define an EM integral as a generalization of the Mellin transform of the rational function $1/f$:

$$I_{f_z}(\nu, \nu_0) = \int\limits_{\mathbb{R}_n^+} d^n\alpha \, \alpha^{\nu-1} f_z(\alpha)^{-\nu_0} \tag{4}$$

where, $d^n\alpha = \prod_{i=1}^{n} d\alpha_i$, and $\alpha^{\nu-1} = \prod_{i=1}^{n} \alpha_i^{\nu_i-1}$.

Analytic and convergence properties of such integrals have been detailed in [37,14,15] by considering the Newton polytope corresponding to the Laurent polynomial, namely, in Theorems 2.2 and 2.4 in [37].

It has been shown in [37] (Theorem 4.2) that for generic z_j, the EM integral I_{f_z} with parameters $(\nu, \nu_0) = (\nu_1, ..., \nu_n, \nu_0)$ could be understood as a solution to a GKZ hypergeometric system of the form Eq. (2) with the following data:

$$\begin{aligned}
&\mathcal{A} = \{a_{ij}; i \in \{1, ..., n+1\}, j \in \{1, ..., N\}\} | a_{ij} = 1; i = 1\} \\
&\beta = -(\nu_0, \nu)^T = -\underline{\nu}
\end{aligned} \tag{5}$$

where, $\underline{\nu} = (\nu_0, \nu)^T$.

Thus, the GKZ system associated with $f_z(\alpha)$ takes the following form in terms of the parameter vector $\underline{\nu}$:

$$H_{\mathcal{A}}(\underline{\nu}) = I_{\mathcal{A}} \cup \langle \mathcal{A} \cdot \theta + \underline{\nu} \rangle \tag{6}$$

3. Scalar Feynman integrals and the Lee-Pomeransky representation

In the momentum-space representation, scalar Feynman integrals take the following form:

$$I_\Gamma(\nu, D) = \int \prod_{r=1}^{l} \frac{d^D k_r}{i\pi^{\frac{D}{2}}} \frac{1}{\prod_{j=1}^{n}(-q_j^2 + m_j^2)^{\nu_j}} \tag{7}$$

where, l is the number of loops, D is the space-time dimension, $\nu = (\nu_1, ..., \nu_n)$ are the propagator powers. k_r-s and q_j-s are the loop-momenta and internal-momenta for the Feynman graph Γ.

Introducing the *Feynman parameters* $\alpha = \{\alpha_i : i \in \{1, ..., n\}\}$ for all the internal edges, and using the notation $\omega = \nu - \frac{lD}{2}$, a parametric representation for the integral above is obtained:

$$I_\Gamma(\nu, D) = \Gamma(\omega) \Big(\prod_{i=1}^{n} \int\limits_{\alpha_i=0}^{1} \frac{d\alpha_i \, \alpha_i^{\nu_i-1}}{\Gamma(\nu_i)} \Big) \frac{\delta(1 - \sum_{j=1}^{n} \alpha_j)}{F(\alpha)^\omega U(\alpha)^{\frac{D}{2}-\omega}} \tag{8}$$

This expression is dubbed the *Feynman parametric representation* for the integral. $U(\alpha)$ and $F(\alpha)$ are called the first and second Symanzik polynomials, respectively.

There is another alternative representation which involves a single polynomial $G(\alpha) = U(\alpha) + F(\alpha)$,

$$\begin{aligned}
I_\Gamma(\nu, D) &= \frac{\Gamma(\frac{D}{2})}{\Gamma(\frac{D}{2}-\omega)} \Big(\prod_{i=1}^{n} \int\limits_{\alpha_i=0}^{\infty} \frac{d\alpha_i \, \alpha_i^{\nu_i-1}}{\Gamma(\nu_i)} \Big) G(\alpha)^{-\frac{d}{2}} \\
&= \frac{\Gamma(\frac{D}{2})}{\Gamma(\frac{D}{2}-\omega)\Gamma(\nu)} \int\limits_{\mathbb{R}_+^n} d\alpha \, \alpha^{\nu-1} G(\alpha)^{-\frac{d}{2}}
\end{aligned} \tag{9}$$

where, we use the following notation:

$$d\alpha = \prod_{i=1}^{n} d\alpha_i \quad, \qquad \alpha^{\nu-1} = \prod_{i=1}^{n} \alpha_i^{\nu_i-1} \quad, \qquad \Gamma(\nu) = \prod_{i=1}^{n} \Gamma(\nu_i) \tag{10}$$

This representation is called the *Lee-Pomeransky representation* [13] for the Feynman integral, and the polynomial $G(\alpha)$ is called the *Lee-Pomeransky polynomial*. Hereafter, we shall call this polynomial the *G-polynomial*.

For further details on the different representations of Feynman integrals, we refer the interested reader to [2].

4. Solution methodology

4.1. The toric G-polynomial and the $\mathcal{A}$-matrix

Polynomials with generic coefficients of its monomials are called *toric polynomials*. It has been shown in [14] that a toric version of the G-polynomial defines a GKZ-system. This toric version is expressed as follows:

$$G_z(\alpha) = \sum_{a_j \in A} z_j \alpha^{a_j} = \sum_{j=1}^{N} z_j \prod_{i=1}^{n} \alpha_i^{a_{ij}} \tag{11}$$

where, N is the number of monomials, z_j are generic coefficients that contain information about the kinematics in the non-generic limit, and are complex numbers $z_j \in (\mathbb{C} \backslash 0)^N$. A is a finite set with N pairwise distinct column vectors $a_j = (a_{1j}, ..., a_{nj})^T \in \mathbb{Z}_{\geq 0}^n$. As such, A is a matrix of order $n \times N$.

The matrix A defines a configuration of N points in affine space $\mathbb{Z}^n$. The convex hull of these points defines a convex polytope:

$$P = \text{Conv}(A) := \left\{ \sum_{j=1}^{N} k_j a_j \,\middle|\, k \in \mathbb{R}_{\geq 0}^N, \sum_{j=1}^{N} k_j = 1 \right\} \tag{12}$$

The toric G-polynomial is a special case of a Laurent polynomial where only non-negative exponents appear. Therefore by Eq. (5), we can write the associated $\mathcal{A}$-matrix as follows:

$$\mathcal{A} = \begin{pmatrix} 1 \\ A \end{pmatrix} = \begin{pmatrix} 1 & 1 & ... & 1 \\ a_1 & a_2 & ... & a_N \end{pmatrix} \in \mathbb{Z}_{\geq 0}^{(n+1) \times N} \tag{13}$$

where, $a_i \in \mathbb{Z}_{\geq 0}^n$.

This condition ensures that the columns in $\mathcal{A}$ are the homogeneous coordinates of the point configuration defined by A. This is the convention used in TOPCOM [29]. More details about this homogeneity condition can be found in [38].

The Newton polytope associated to the toric G-polynomial is defined as the convex hull of the exponent vectors:

$$\Delta_{G_z} := \text{Conv}(\{a_j = (a_{1j}, ..., a_{nj})^T | z_j \neq 0\}, j \in \{1, ..., N\}) \tag{14}$$

For details about convex polytopes in general and the Newton polytope in particular, we refer the reader to [15,2].

From this discussion, we can see that $\Delta_{G_z} = \text{Conv}(A)$.

4.2. The triangulation method

4.2.1. Triangulations of Δ_{G_z}

A triangulation of Δ_{G_z} is its subdivision into simplices $\{\sigma_1, ..., \sigma_r\}$, such that union of all simplices gives the full polytope, and the intersection of two distinct simplices is either empty or a proper face of both. Here, $\sigma_i \subset \{1, ..., N\}$ is an index set, $i \in \{1, ..., r\}$, and $\overline{\sigma} = \{1, ..., N\} \backslash \sigma$ is the complement of σ.

Regular or *coherent* triangulations $T(h) = \{\sigma_1, ..., \sigma_r\}$ are those for which there exists a height vector $h \in \mathbb{R}^N$, such that for every simplex σ_i of such a triangulation, there exists another vector $p_i \in \mathbb{R}^{n+1}$ satisfying

$$\begin{aligned} p_i \cdot a_j &= h_j \quad \text{for} \quad j \in \sigma_i \\ p_i \cdot a_j &< h_j \quad \text{for} \quad j \notin \sigma_i \end{aligned} \tag{15}$$

It has been shown in [38] that a regular triangulation can always be obtained, given an arbitrary convex polytope.

As mentioned earlier in section 2.1, we consider the rank of $\mathcal{A} \in \mathbb{Z}^{(n+1) \times N}$ to be $n+1$. Let δ be the greatest common divisor of all $(n+1) \times (n+1)$ minors of $\mathcal{A}$. Then the *normalized volume* of σ is given by $\text{vol}_0(\sigma) = |\det \mathcal{A}_\sigma| / \delta$ where, $\mathcal{A}_\sigma$ denotes the submatrix formed by retaining the columns indexed by σ and dropping the others. This definition could be found in [36,39].

Unimodular triangulations $\{\sigma_1, ..., \sigma_r\}$ are those for which, $\text{vol}_0(\sigma_i) = 1 \,\forall\, i \in \{1, ..., r\}$.

4.2.2. Generalized Feynman integrals

Employing the notion of the toric G-polynomial with generic coefficients $z_j \in (\mathbb{C} \backslash 0)^N$ such that $Re(z_j) > 0$, we can consider a generalized Feynman integral as the meromorphic continuation of the integral

$$I_{G_z}(\nu, \nu_0) = \Gamma(\nu_0) \int_{\mathbb{R}_+^n} d\alpha \, \alpha^{\nu-1} G_z(\alpha)^{-\nu_0} \tag{16}$$

defined on $\underline{\nu} = (\nu_0, \nu) \in \mathbb{C}^{n+1}$, with $\nu_0 = \frac{D}{2}$.

From the discussion in [15], another representation of $I_{G_z}(\nu, \nu_0)$ could be obtained as a multi-fold MB integral:

$$I_{G_z}(\nu, \nu_0) = \frac{z_\sigma^{-\mathcal{A}_\sigma^{-1} \underline{\nu}}}{\text{vol}_0(\sigma)} \int_\gamma \frac{dt}{(2\pi i)^r} \Gamma(t) \Gamma(\mathcal{A}_\sigma^{-1} \underline{\nu} - \mathcal{A}_\sigma^{-1} \mathcal{A}_{\overline{\sigma}} t) z_{\overline{\sigma}}^{-t} z_\sigma^{\mathcal{A}_\sigma^{-1} \mathcal{A}_{\overline{\sigma}} t} \tag{17}$$

From the above, a special case follows where $N = n + 1$. If there exists some region $R \subseteq \mathbb{C}^{n+1}$ such that $I_{G_z}(\nu, \nu_0)$ is absolutely convergent for $\underline{\nu} \in R$, then $\det \mathcal{A}_\sigma \neq 0$, and the Feynman integral takes a simple form:

$$I_{G_z}(\nu, \nu_0) = \frac{\Gamma(\mathcal{A}_\sigma^{-1}\underline{\nu})}{\mathrm{vol}_0(\sigma)} z^{-\mathcal{A}_\sigma^{-1}\underline{\nu}} \tag{18}$$

where, owing to the square matrix form, σ contains all the columns, that is, $\mathcal{A}_\sigma = \mathcal{A}$. Thus, $\mathrm{vol}_0(\sigma) = |\det \mathcal{A}|/\delta$.

4.2.3. Generalized Feynman integrals as GKZ hypergeometric functions

Comparing Eq. (11) with Eq. (3) and Eq. (16) with Eq. (4), we can see that generalized Feynman integrals could be understood as EM integrals. Thus, $I_{G_z}(\nu, \nu_0)$ satisfies the GKZ hypergeometric system $H_{\mathcal{A}}(\underline{\nu})$ with the associated data as specified in (5), where $\nu_0 = \frac{d}{2}$. For a formal proof, we refer the reader to [15] (Theorem 3.1).

A basis for the solution space of $H_{\mathcal{A}}(\underline{\nu})$ could be obtained using the regular triangulations of Δ_{G_z}, with the restriction that the parameter vector $\underline{\nu}$ be very generic. Details of this procedure could be found in [9] (Theorem 2.17) and [40].

Elements of this basis are called the Γ-series. The number of such elements is given by rank $H_{\mathcal{A}}(\underline{\nu}) = \mathrm{vol}_0(\mathrm{Conv}(A))$. Since GKZ hypergeometric systems are holonomic, it is ensured that the dimensionality of the solution space (and hence the number of such basis elements) is finite.

For unimodular regular triangulations, each simplex of the triangulation can lead to only one such series, and hence, we can label the Γ-series by their corresponding simplices. It can be shown that in the unimodular case, the Γ-series associated to some simplex $\sigma \in T$ takes the following form:

$$\Phi_\sigma(\underline{\nu}, z) = z_\sigma^{-\mathcal{A}_\sigma^{-1}\underline{\nu}} \sum_{\lambda \in \mathbb{N}_0^{|\bar{\sigma}|}} \frac{(\mathcal{A}_\sigma^{-1}\underline{\nu})_{\mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda}}{\lambda!} \frac{z_{\bar{\sigma}}^\lambda}{(-z_\sigma)^{\mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda}} \tag{19}$$

Using the Knudsen-Mumford theorem [41,42], it could be argued [15] that by adding monomials to the toric G-polynomial such that $G_z \rightarrow G'_{z'}$ (or equivalently, by scaling the underlying Newton polytope Δ_{G_z} with some integer scale-factor such that $\Delta_{G_z} \rightarrow \Delta_{G'_{z'}}$), an unimodular triangulation can always be obtained. As a result, it would be enough to consider only those regular triangulations of Δ_{G_z} that happen to be unimodular as well, in order to construct a basis for the solution space of $H_{\mathcal{A}}(\underline{\nu})$.

Starting with the case $\underline{\nu} \in (\mathbb{C} \setminus \mathbb{Z})^{n+1}$, we can write the corresponding generalized Feynman integral as a linear combination of the Γ-series:

$$I_{G_z}(\nu, \nu_0) = \sum_{\sigma \in T} P_\sigma(\underline{\nu}) \Phi_\sigma(\underline{\nu}, z) \tag{20}$$

for $z_j \in \mathbb{C}^N$ such that $Re(z_j) \geq 0$ wherever the Γ-series converge. The prefactors $P_\sigma(\underline{\nu})$ are meromorphic functions in the parameter vector, and are to be computed in order to obtain a series representation for the Feynman integral.

One simple way to compute these prefactors is to exploit subtriangulations of the triangulation T. By a series of arguments, the authors in [15] show that one such trivial subtriangulation could be considered to be one of the simplices of T. In this case, using Eq. (18), it could be shown that these prefactors are given by $P_\sigma(\underline{\nu}) = \Gamma(\mathcal{A}_\sigma^{-1}\underline{\nu})$, since due to the unimodularity condition, $\mathrm{vol}_0(\sigma) = 1$.

As a result, using the above and Eq. (19), we can write the generalized Feynman integral as follows:

$$I_{G_z}(\nu, \nu_0) = \Gamma(\mathcal{A}_\sigma^{-1}\underline{\nu}) \sum_{\sigma \in T} z_\sigma^{-\mathcal{A}_\sigma^{-1}\underline{\nu}} \sum_{\lambda \in \mathbb{N}_0^{|\bar{\sigma}|}} \frac{(\mathcal{A}_\sigma^{-1}\underline{\nu})_{\mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda}}{\lambda!} \frac{z_{\bar{\sigma}}^\lambda}{(-z_\sigma)^{\mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda}}$$

$$\implies I_{G_z}(\nu, \nu_0) = \sum_{\sigma \in T} z_\sigma^{-\mathcal{A}_\sigma^{-1}\underline{\nu}} \sum_{\lambda \in \mathbb{N}_0^{|\bar{\sigma}|}} \frac{\Gamma(\mathcal{A}_\sigma^{-1}\underline{\nu} + \mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda)}{\lambda!} \frac{z_{\bar{\sigma}}^\lambda}{(-z_\sigma)^{\mathcal{A}_\sigma^{-1}\mathcal{A}_{\bar{\sigma}}\lambda}} \tag{21}$$

where the series have a common convergence domain.

Eq. (18) and Eq. (21) comprise the main equations that go into the geometric solution side of our package. Note that these equations provide us with solutions only for the generalized Feynman integral. Suitable limits for the variables z_j need to be set in order to get back to the original Feynman integral, and the process has been discussed at length in [15].

4.3. The GD method

Here we summarize the key steps of the GD method, which is based on the algorithm proposed by Saito, Sturmfels and Takayama (SST) [28]. Let D_N be a Weyl algebra of N variables over a field $\mathbb{K}$. It is generated by the variables $z_1, \ldots, z_N, \partial_1, \ldots, \partial_N$ where all the z_i's commute among themselves and so do the ∂_i's, but $[\partial_i, z_j] = \delta_{ij}$. An element p that belongs to the Weyl algebra can be written in the canonical form, where all the ∂_i are brought to the right by the use of commutation relations.

$$p = \sum_{i,j \in \mathbb{N}^N} c_{ij} z^i \partial^j \tag{22}$$

Here z^i for $i \in \mathbb{N}^N$ implies $z_1^{i_1} \ldots z_N^{i_N}$ and the same for ∂^j. For a vector $w \in \mathbb{R}^N$,

$$\text{in}_{(-w,w)}(p) := \left\{ \sum_{r,s \in \mathbb{N}^N} c_{rs} z^r \partial^s \mid -\langle w, r \rangle + \langle w, s \rangle \text{ is maximum} \right\} \tag{23}$$

where $\langle \_, \_ \rangle$ is the usual scalar product in $\mathbb{R}^N$.

For an ideal $I \subset D_N$, the Gröbner deformation of I with respect to the weight w is defined as

$$\text{in}_{(-w,w)}(I) := \langle \text{in}_{(-w,w)}(p) \mid p \in I \rangle$$

The initial ideal is the Gröbner deformation of the toric ideal $I_{\mathcal{A}}$ associated with matrix $\mathcal{A}$ with respect to a generic weight vector.

Let M be a monomial ideal in $\mathbb{K}[\partial]$. A standard pair associated with it is a pair (∂^a, σ) where $a \in \mathbb{N}^N$ and $\sigma \subset \{1, \ldots, N\}$ satisfying

- $a_i = 0$ for all $i \in \sigma$
- for all choices of integers $b_j \geq 0$, $\partial^a \prod_{j \in \sigma} \partial_j^{b_j} \notin M$
- for all $l \notin \sigma, \exists\, b_j \geq 0 : \partial^a \partial_l^{b_l} \prod_{j \in \sigma} \partial_j^{b_j} \in M$

Let $\mathcal{S}(M)$ be the set of all the standard pairs of M then the decomposition of M into irreducible monomial ideals can be found as

$$M = \bigcap_{(\partial^a, \sigma) \in \mathcal{S}(M)} \langle \partial_i^{a_i+1} : i \notin \sigma \rangle$$

From the standard pair one can compute the indicial ideal as [33]

$$\text{ind}_w(I_{\mathcal{A}}) = \bigcap_{(\partial^a, \sigma) \in \mathcal{S}(\text{in}_w(I_{\mathcal{A}}))} \left\langle \left(\theta_j - a_j \right), j \notin \sigma \right\rangle$$

and fake indicial ideal as

$$\text{find}_w(H_{\mathcal{A}}(\beta)) = \bigcap_{(\partial^a, \sigma) \in \mathcal{S}(\text{in}_w(I_{\mathcal{A}}))} \left\langle \left(\theta_j - a_j \right), j \notin \sigma \right\rangle + \langle \mathcal{A} \cdot \theta - \beta \rangle$$

The roots of the fake indicial ideal are called fake exponents of the GKZ system for a given weight. We denote the fake exponents by v.

Let us denote the kernel of the matrix $\mathcal{A}$ by

$$L := \ker_{\mathbb{Z}}(\mathcal{A}) \tag{24}$$

For $u \in L$, one can write $u = u_+ - u_-$, where u_+ and u_- contains non-negative entries.

Following [28], we define the falling factorials for a vector $v \in \mathbb{K}^N$,

$$[v]_{u_-} = \prod_{i:u_i<0} \prod_{j=1}^{-u_i} (v_i - j + 1)$$

$$[u + v]_{u_+} = \prod_{i:u_i>0} \prod_{j=1}^{u_i} (u_i + j)$$

In what follows, we assume that the vector v does not contain negative integers so that

$$[u + v]_{u_+} \neq 0 \qquad \text{for all } u \in \mathbb{Z}^N$$

The series solution of the GKZ system can be written as [28]

$$\phi_v := \sum_{u \in L} \frac{[v]_{u_-}}{[u + v]_{u_+}} z^{u+v}$$

This sums up the general strategy to find the canonical series solutions of a GKZ system $H_{\mathcal{A}}(\beta)$.

5. Documentation and usage

In this section we describe the usage of the external modules of the package `FeynGKZ.wl`. There are a total of nine external modules that the user has access to. Here, we enlist only six of them that participate in creating the basic workflow of our package, namely, deriving the GKZ system for a given Feynman integral and solving the system in terms of hypergeometric functions. The remaining three modules with additional functionality have been described in appendix A.

For a given Feynman integral, our first step is to find the $\mathcal{A}$-matrix associated to its GKZ system. This is done by using the `FindAMatrix` command. This command can accept two different input patterns of the Feynman integral: the momentum representation or the Schwinger parametric representation.

`FindAMatrix[{MomentumRep, LoopMomenta, InvariantList, Dim, Prefactor}]`

or,

`FindAMatrix[{{U, F, PropagatorPowers}, LoopNumber, InvariantList, Dim, Prefactor}]`

Below, we provide details about the input arguments for `FindAMatrix`.

- `MomentumRep`: A list containing information about all the propagators in the Feynman integral. It consists of multiple sub-lists, each of which has the following pattern:
  `{Propagator-Momentum, Propagator-Mass, Propagator-Power}`.
- `U`: The first Symanzik polynomial corresponding to the given Feynman integral.
- `F`: The second Symanzik polynomial corresponding to the given Feynman integral.
- `PropagatorPowers`: A list containing all the powers of the propagators of the Feynman integral.
- `LoopMomenta`: A list containing the loop-momenta for the given Feynman integral.
- `LoopNumber`: The number of loops appearing in the given Feynman integral. It must be an positive integer.
- `InvariantList`: A list of kinematic substitutions.
- `Dim`: the space-time dimension.
- `Prefactor`: An overall factor that is independent of the loop-momenta.
- Options:
  - `UseMB`: This option accepts a boolean value. Its default value is `False`. When set `True`, `FindAMatrix` calls AM-BREv2.1.1.m to obtain the MB-representation corresponding to the given Feynman integral, and uses it to compute the $\mathcal{A}$-matrix.

Upon a successful run with `UseMB` $\rightarrow$ `False`, `FindAMatrix` prints the U, F and G polynomials, the $\mathcal{A}$-matrix, the normalised Euclidean volume of the polytope defined by the $\mathcal{A}$-matrix as calculated by `polymake`, and the total time taken by the command to run.

The option `UseMB` $\rightarrow$ `True` works only when the Feynman integral is given in its momentum representation. When run with `UseMB` $\rightarrow$ `True`, AMBRE is called to derive the MB representation, which is subsequently used to derive the $\mathcal{A}$-matrix following an algorithm similar to [16]. This option is still at an experimental stage as it works only when the derived MB representation is in a certain form.

The next step is to compute either the set of all unimodular regular triangulations, or the set of all square-free initial ideals, for the GKZ system defined by the $\mathcal{A}$-matrix obtained in the previous step. One may do this by using either of the following two external modules:

`FindTriangulations[FindAMatrixOut]`

This external module computes unimodular regular triangulations corresponding to the $\mathcal{A}$-matrix generated by `FindAMatrix` using TOPCOM. Below, we provide details about the input arguments for `FindTriangulations`.

- `FindAMatrixOut`: Result returned by `FindAMatrix`.
- Options:
  - `MaxRegularTriangs`: This option can take either of the two values: `All`, or some positive integer. Its default value is `All`. When set to `All`, `FindTriangulations` tries to find all possible regular triangulations. When set to a positive integer n, `FindTriangulations` stops after obtaining upto n regular triangulations and then scans for unimodular regular triangulations.
  - `PrintRegularTriangs`: This option accepts a boolean value. By default, this option is set to `False`. When set to `True`, `FindTriangulations` prints the regular triangulations in addition to the unimodular regular triangulations.
  - `RunInParallel`: This option accepts a boolean value. By default, this option is set to `False`. When set to `True`, Mathematica's native parallelization features are used.
  - `Docker`: This option accepts a boolean value. By default, this option is set to `False`. When set to `True`, one can call TOPCOM from a Docker container having the name `topcom`, to obtain the regular triangulations. This option might come in handy, should one fail to install TOPCOM natively on their system.

Upon a successful run with all options set to their default values, `FindTriangulations` prints all the unimodular regular triangulations, and the overall time consumed at this stage of the computation.

FindInitialIdeals[FindAMatrixOut]

This external module generates all possible square-free initial ideals corresponding to the $\mathcal{A}$-matrix generated by FindAMatrix. Below, we provide details about its input arguments.

- FindAMatrixOut: Result returned by FindAMatrix.

Executing FindInitialIdeals prints all possible square-free initial ideals associated with the given $\mathcal{A}$-matrix, and the overall time consumed. This module does the following:

- Use Macaulay2 to obtain the toric ideals associated with the $\mathcal{A}$-matrix.
- Use the gfanInterface package of Macaulay2 to find all possible initial ideals.
- Filter for all possible square-free initial ideals.

We now construct the series solutions for a given choice of unimodular regular triangulation or square-free initial ideal generated by FindTriangulations or FindInitialIdeals, respectively.

SeriesRepresentation[Triangulations, TriangNum]

or,

SeriesRepresentation[Ideals, IdealNum]

where, the input arguments are as follows:

- Triangulations: Result returned by FindTriangulations.
- TriangNum: Serial number of the unimodular regular triangulation for which the corresponding series solution is to be derived
- Ideals: result returned by FindInitialIdeals.
- IdealNum: Serial number of the square-free initial ideal for which the corresponding series solution is to be derived.
- Options:
  - SubstituteScales: This option accepts a boolean value. By default, this option is set to True. When set to True, all the generic coefficients z_j, are substituted with their appropriate kinematic counterparts while constructing the series solution for the given non-generic Feynman integral. When False, this substitution is not done.
  - ParameterValue: This option accepts a list of parameter substitutions.
    By default, this option is assigned an empty list.

Running SeriesRepresentation with the result returned from FindTriangulation or FindInitialIdeals will first print the triangulation number TriangNum or the ideal number IdealNum, respectively. It will then print the number of summation variables involved in the series solution, the scale substitutions done internally to construct the non-generic limit result for SubstituteScales → True. For SubstituteScales → False, it simply prints the generic scales in terms of z_j. Finally, it prints the summands of the Γ-series terms constituting the series solution and the total time taken.

Subsequently, one may wish to check if the solution returned from SeriesRepresentation can be written in terms of known hypergeometric functions. This functionality has been implemented in the external module GetClosedForm. For single-fold series, it relies on *Mathematica*'s native capabilities to obtain a closed-form expression. However, for double-sums, these capabilities are usually not sufficient, and therefore, we use the package Olsson.wl [32] to try to obtain a closed-form expression for the series solution. Below, we specify the syntax of this module:

GetClosedForm[MySeries]

which has the following input arguments:

- MySeries: Result returned by SeriesRepresentation or GroebnerDeformation.
- Options:
  - ParameterValue: This option accepts a list of parameter substitutions.
    By default, this option is assigned an empty list.

This module prints closed-form expressions if found, else it prints the original series, in the same order as in SeriesRepresentation.

Beyond double-sums, it is not straightforward to write Γ-series terms as known hypergeometric functions. Therefore, in the present version GetClosedForm can yield closed form result upto two-fold series. In fact, even in the two-fold case, one might not necessarily be always able to express these terms as hypergeometric closed-forms.
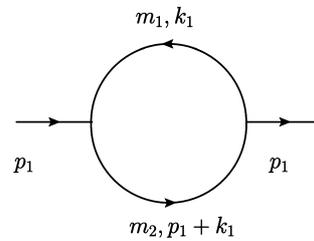
Fig. 1. The two-mass bubble diagram.

The final module `NumericalSum` is meant for numerical summation of the series solution we obtain from `SeriesRepresentation`. It also works for the series solution derived from the module `GroebnerDeformation` which is described in appendix A.1.

`NumericalSum[MySeries, SubstitutionRules, SumLim]`

with the following input arguments:

- `MySeries`: Result returned by `SeriesRepresentation` or `GroebnerDeformation`.
- `SubstitutionRules`: This a list to substitute numerical values to parameters and scales of `MySeries` before numerical summation.
- `SumLim`: The upper-limit for summing over the summation variables. The same upper-limit is maintained for each summation variable.
- Options:
  - `NumericalPrecision`: Can be used for arbitrary precision calculations. By default, this option is set to `MachinePrecision`.
  - `RunInParallel`: This option accepts a boolean value. Its default value is `False`. When set to `True` for cases with a rather complicated `MySeries` or a large `SumLim`, running `NumericalSum` may achieve a non-trivial gain in speed.

Running `NumericalSum` for a given `MySeries`, `SubstitutionRules` and `SumLim` would print the numerical result, along with the net time taken for the module's execution.

6. Example: two-mass bubble

In this section, we provide an explicit demonstration of the typical Feynman integral evaluation procedure using `FeynGKZ.wl`. We consider the Feynman integral corresponding to the two-mass bubble diagram (Fig. 1),

$$I_\Gamma(\nu_1, \nu_2, D; p_1^2) = \int \frac{d^D k_1}{i\pi^{\frac{D}{2}}} \frac{1}{(-k_1^2 + m_1^2)^{\nu_1}(-(p_1+k_1)^2 + m_2^2)^{\nu_2}} \tag{25}$$

with two unequal masses m_1 and m_2, and external momentum p_1.

We start with specifying the paths to `FeynGKZ.wl`, the `polymake` executable and the `points2triangs` executable from TOPCOM. They are stored as strings in the predefined variables `FeynGKZPath`, `PolyMakePath` and `TOPCOMPath`, respectively. Internally, these variables have been set to the default values `NotebookDirectory[]`, `"/usr/bin"` and `"/usr/local/bin"`, respectively. For our case here, we shall assume that they have been installed in the following non-standard locations.

```
In[1]:=   FeynGKZPath="/home/user/gkz/math/" ;
          PolyMakePath="/home/user/polymake/bin" ;
          TOPCOMPath="/home/user/topcom/bin" ;
```

Once this has been done, one can load the package as follows.

```
In[2]:=   Get[FeynGKZPath<>"FeynGKZ.wl"] ;
```

```
Prints ⇒   FeynGKZ 1.0 - a Mathematica package for solving
               Feynman Integrals using GKZ hypergeometric systems
           Authors: B. Ananthanarayan, Sumit Banik, Souvik Bera, Sudeepan Datta
           Last updated: 2nd November, 2022
```

Then we specify the momentum representation of the Feynman integral in Eq. (25) as follows,

```
In[3]:=    MomentumRep ={{k_1, m_1, a_1}, {p_1 + k_1, m_2, a_2}};
           LoopMomenta ={k_1};
           InvariantList ={p_1^2 → −s};
           Dim = 4 − 2ϵ;
           Prefactor = 1;
```

Next, we use the module `FindAMatrix` to compute the $\mathcal{A}$-matrix associated with the GKZ system satisfied by this Feynman integral.

```
In[4]:=    FindAMatrixOut = FindAMatrix[{MomentumRep, LoopMomenta,
            InvariantList, Dim, Prefactor}, UseMB → False];
```

```
Prints ⇒   The Symanzik polynomials → U = x_1 + x_2
            , F = m_1^2 x_1^2 + s x_1 x_2 + m_1^2 x_1 x_2 + m_2^2 x_1 x_2 + m_2^2 x_2^2
           The Lee-Pomeransky polynomial → G =
            x_1 + m_1^2 x_1^2 + x_2 + s x_1 x_2 + m_1^2 x_1 x_2 + m_2^2 x_1 x_2 + m_2^2 x_2^2
                                          ⎛ 1  1  1  1  1 ⎞
           The associated A−matrix →     ⎜ 2  1  1  0  0 ⎟ ,  which has codim = 2.
                                          ⎝ 0  1  0  2  1 ⎠
           Normalized Volume of the associated Newton Polytope → 3
           Time Taken 1.50005 seconds
```

The codimension (`codim`) of $\mathcal{A}$ is defined as the number of columns minus the number of rows.

After this, we can calculate the unimodular regular triangulations of the point configuration defined by the $\mathcal{A}$-matrix using `FindTriangulations` as follows:

```
In[5]:=    Triangulations = FindTriangulations[FindAMatrixOut];
```

```
Prints ⇒   Finding all regular triangulations ...
           Found 5 Regular Triangulations, out of which 3 are Unimodular
           The 3 Unimodular Regular Triangulations →
           1 :: {{1,2,3}, {2,3,4}, {3,4,5}}
           2 :: {{1,2,3}, {2,4,5}, {2,3,5}}
           3 :: {{2,4,5}, {1,3,5}, {1,2,5}}
           Time Taken 0.126965 seconds
```

Alternatively, we can also calculate all possible square-free initial ideals from the $\mathcal{A}$-matrix using `FindIntialIdeals`,

```
In[6]:=    InitialIdeals = FindInitialIdeals[FindAMatrixOut];
```

```
Prints ⇒   There are 3 possible square-free initial ideals :
           1 :: {t_2 t_5,  t_1 t_5,  t_1 t_4}
           2 :: {t_1 t_5,  t_1 t_4,  t_3 t_4}
           3 :: {t_1 t_4,  t_3 t_4,  t_2 t_3}
           Time Taken 3.38268 seconds
```

Next, one can find the Γ-series solutions for the non-generic Feynman integral either for a given choice of square-free initial ideal or unimodular regular triangulation. As described earlier in the documentation, this is done using `SeriesRepresentation`. We derive the series solution corresponding to the second unimodular regular triangulation as follows,

```
In[7]:=   SeriesSolution = SeriesRepresentation[Triangulations,2];
```

Prints $\Rightarrow$ Unimodular Triangulation $\rightarrow$ 2
Number of summation variables $\rightarrow$ 2
Non-generic limit $\rightarrow$ $\{z_1 \rightarrow m_1^2,\ z_2 \rightarrow s+m_1^2+m_2^2,\ z_3 \rightarrow 1,\ z_4 \rightarrow m_2^2,\ z_5 \rightarrow 1\}$
The series solution is the sum of following 3 terms.

Term 1 ::

$$\Bigg(\Bigg(\Big((-1)^{-n_1-n_2}\, \text{Gamma}[-2+\epsilon+a_1-n_1-n_2]\, \text{Gamma}[4-2\epsilon-a_1-a_2+n_2]$$

$$\text{Gamma}[a_2+2n_1+n_2]\,(m_1^2)^{2-\epsilon-a_1}\left(\frac{m_1^2 m_2^2}{(s+m_1^2+m_2^2)^2}\right)^{n_1}\left(\frac{m_1^2}{s+m_1^2+m_2^2}\right)^{n_2}$$

$$(s+m_1^2+m_2^2)^{-a_2}\Big)\Big/\Big(\text{Gamma}[a_1]\,\text{Gamma}[4-2\epsilon-a_1-a_2]\,\text{Gamma}[a_2]$$

$$\text{Gamma}[1+n_1]\,\text{Gamma}[1+n_2]\Big)\Bigg)$$

Term 2 ::

$$\Bigg(\Bigg(\Big((-1)^{-n_1-n_2}\, \text{Gamma}[-2+\epsilon+a_2-n_1-n_2]\, \text{Gamma}[4-2\epsilon-a_1-a_2+n_2]$$

$$\text{Gamma}[a_1+2n_1+n_2]\,(m_2^2)^{2-\epsilon-a_2}\left(\frac{m_1^2 m_2^2}{(s+m_1^2+m_2^2)^2}\right)^{n_1}\left(\frac{m_2^2}{s+m_1^2+m_2^2}\right)^{n_2}$$

$$(s+m_1^2+m_2^2)^{-a_1}\Big)\Big/\Big(\text{Gamma}[a_1]\,\text{Gamma}[4-2\epsilon-a_1-a_2]\,\text{Gamma}[a_2]$$

$$\text{Gamma}[1+n_1]\,\text{Gamma}[1+n_2]\Big)\Bigg)$$

Term 3 ::

$$\Bigg(\Bigg(\Big((-1)^{-n_1-n_2}\, \text{Gamma}[2-\epsilon-a_2+n_1-n_2]\, \text{Gamma}[2-\epsilon-a_1-n_1+n_2]$$

$$\text{Gamma}[-2+\epsilon+a_1+a_2+n_1+n_2]\left(\frac{m_1^2}{s+m_1^2+m_2^2}\right)^{n_1}\left(\frac{m_2^2}{s+m_1^2+m_2^2}\right)^{n_2}$$

$$(s+m_1^2+m_2^2)^{2-\epsilon-a_1-a_2}\Big)\Big/\Big(\text{Gamma}[a_1]\,\text{Gamma}[4-2\epsilon-a_1-a_2]$$

$$\text{Gamma}[a_2]\,\text{Gamma}[1+n_1]\,\text{Gamma}[1+n_2]\Big)\Bigg)$$

Time Taken 0.066558 seconds

where n_1 and n_2 are the summation indices, each of which runs from 0 to ∞. Let us now try to express these results in terms of known hypergeometric functions using the module `GetClosedForm` which internally uses `Olsson.wl` [32] for this step.

```
In[8]:=   GetClosedForm[SeriesSolution];
```

Prints $\Rightarrow$ Closed form found with Olsson!

Term 1 ::

$$\frac{1}{\text{Gamma}[a_1]}\,\text{Gamma}[-2+\epsilon+a_1]$$

$$\text{H3}\left[a_2, 4-2\epsilon-a_1-a_2, 3-\epsilon-a_1, \frac{m_1^2 m_2^2}{(s+m_1^2+m_2^2)^2}, \frac{m_1^2}{s+m_1^2+m_2^2}\right]$$

$$m_1^4\,(m_1^2)^{-\epsilon-a_1}\,(s+m_1^2+m_2^2)^{-a_2}$$

Term 2 ::

$$\frac{1}{\text{Gamma}[a_2]}\,\text{Gamma}[-2+\epsilon+a_2]$$

$$\text{H3}\left[a_1, 4-2\epsilon-a_1-a_2, 3-\epsilon-a_2, \frac{m_1^2 m_2^2}{(s+m_1^2+m_2^2)^2}, \frac{m_2^2}{s+m_1^2+m_2^2}\right]$$

$$m_2^4\,(m_2^2)^{-\epsilon-a_2}\,(s+m_1^2+m_2^2)^{-a_1}$$

Term 3 ::

$$\Bigg(\Bigg(\text{G1}\Big[-2+\epsilon+a_1+a_2, 2-\epsilon-a_1, 2-\epsilon-a_2, -\frac{m_2^2}{s+m_1^2+m_2^2}$$

$$, -\frac{m_1^2}{s+m_1^2+m_2^2}\Big]\,\text{Gamma}[2-\epsilon-a_1]\,\text{Gamma}[2-\epsilon-a_2]$$

$$\text{Gamma}[-2+\epsilon+a_1+a_2]\,(s+m_1^2+m_2^2)^{2-\epsilon-a_1-a_2}\Bigg)\Big/\Big(\text{Gamma}[a_1]$$

$$\text{Gamma}[4-2\epsilon-a_1-a_2]\,\text{Gamma}[a_2]\Big)\Bigg)$$

Time Taken 0.05827 seconds

where, `H3` and `G1` are the double variable Horn H_3 and G_1 functions [3].

We can also numerically evaluate the result returned in `Out[7]`, which has been stored in the variable `SeriesSolution`.

```
In[9]:=   SumLim = 30;
          ParameterSub = {ε → 0.001, a₁ → 1, a₂ → 1, s → 10, m₁ → 0.4, m₂ → 0.3};
          NumericalSum[SeriesSolution, ParameterSub, SumLim];

Prints ⇒  Numerical result = 997.382
          Time Taken 0.222572 seconds
```

7. Discussions and conclusions

In this work, we have considered two very active and interconnected fields of research: Feynman integrals and multivariate hypergeometric functions. While the former is of great importance to the particle physics community as they present challenges at the frontier with consequences to precision tests of the Standard Model, the latter is a rich field in mathematics which has been studied for over a century and has found numerous applications in several domains of physics. The connection between these two diverse fields of research was first pointed out in late 1960s by Regge [44]. Since then, considerable efforts have been made to study them in parallel, and our work in this paper is motivated by these considerations.

One systematic approach to study multivariate hypergeometric functions came from GKZ, in the form of what we know today as the theory of GKZ hypergeometric systems. Multivariate hypergeometric series are solutions to the partial differential equations of the GKZ system. The mechanism to find a basis of solutions for the GKZ system was advanced by the original authors themselves in the triangulation scheme. On the other hand, SST introduced the GD method of finding solutions to a GKZ system, which is inspired by the method of Frobenius for solving ordinary differential equations. Its connection to Feynman integrals has become explicit in the recent years due to development of the LP representation for counting the number of master integrals for any given family of Feynman integrals [13].

To recall, the GD and the triangulation methods, were recently used to evaluate Feynman integrals in [14] and [15] respectively. In both the aforementioned articles, the LP polynomial is used to construct the $\mathcal{A}$-matrix. Alternatively, one can start from the MB representation and derive the associated GKZ system for a given Feynman integral, as advocated in [16–18]. Although, the methods were algorithmic, no explicit computer programs were publicly available.

We have presented in this work a working code automating the above methods for solving GKZ systems that are associated with Feynman integrals. Our package is based on *Mathematica*, which relies on `Macaulay2`, `polymake` and `TOPCOM`, for the GD and triangulation methods, respectively. We have described in detail these methods and their implementation with adequate usage documentation, and have also made our code available in the public domain. We believe that our implementation would also help the user to appreciate the underlying equivalence of both the methods, particularly in the setting of Feynman integral evaluation. To this end, we have maintained the following principles in our implementation:

- Enabling sufficient flexibility in the usage, so that the user can take an informed decision about the method they would feel at ease to proceed with in a particular circumstance - which would also help them in ensuring that the results obtained from both the methods are always indeed the same.
- Provision of two commands that could be used to explicitly test the equivalence of unimodular regular triangulations and square-free initial ideals. The details of these commands have been provided in appendix A.2.

We have also automated the part where we can obtain the $\mathcal{A}$-matrix from the MB representation. Our algorithm is inspired by considerations in [16] and its rigorous validity is a subject of ongoing work.

An alternative method to derive MB representations is proposed in [43], which can yield lower-fold MB integrals compared to `AMBRE` in certain cases. This may yield some improvements to our current implementation of the method for obtaining solutions to the underlying GKZ system, and is a topic for future investigation.

We have tested our package analytically as well as numerically using `FIESTA` [45] by evaluating several familiar yet reasonably non-trivial Feynman integrals for which results mostly exist in the literature. One of them includes the non-trivial two-loop self-energy with four propagators presented recently in [18]. Several other examples can be found in the *Mathematica* notebook `Examples.nb` which can be downloaded along with the package from GitHub.

It may be noted here, the time taken by the command `FindTriangulations` to find all the unimodular regular triangulations is typically lesser than the same for the command `FindInitialIdeals` that is used to find all the square-free initial ideals. In this sense, the command `points2triangs` of `TOPCOM` works faster than the `gfan` command of `Macaulay2`. There is one more advantage of the triangulation method. The process of finding the unimodular regular triangulations can be stopped midway, after yielding the required number of triangulations. This is not possible in the `gfan` command of `Macaulay2`. Even so, the results coming from the GD method provide a useful crosscheck of the results obtained through the triangulation method.

`FeynGKZ.wl` yields several series solutions in terms of multivariate hypergeometric functions that are analytic continuations of each other, each being valid in its own convergence region. These solutions are valid for generic values of the dimension and propagator powers. The evaluation of these solutions for integer powers of propagators where logarithmic terms appear, could be carried out by performing a suitable limit-taking procedure manually. We hope to automate this in a future version of the package. This is however not a restriction in the MB approach [6], where one uses multivariate residues to evaluate the integer propagator case and obtain logarithmic series solutions. Therefore, in future, it will be useful to study a cross-fertilization of the two techniques.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix A. Other external modules - documentation

In this section, we provide the documentation for the three remaining external modules that were not covered in the main text.

A.1. The `GroebnerDeformation` external module

This external module provides an alternative method to construct the Γ-series solutions, employing the SST algorithm. Its syntax is as follows:

`GroebnerDeformation[FindAMatrixOut]`

with the following input arguments:

- `FindAMatrixOut`: result returned by `FindAMatrix`.
- Options:
  - `InitialIdeal`: Serial number of the square-free initial ideal for which the corresponding series solution is to be derived. Its default value is `False`.
  - `Weight`: This option accepts a list of non-negative integers, whose length is same as the number of columns of the $\mathcal{A}$-matrix. Its default value is `False`.
  - `AllInitialIdeals`: This option accepts a boolean value. Its default value is `False`. When set to `True`, this module yields series solutions for all the possible square-free initial ideals sequentially.
  - `ParameterValue`: This option accepts a list of parameter substitutions. By default, this option is assigned an empty list.

Note: the options `InitialIdeal` and `Weight` cannot be used simultaneously. One of them must be set to `False` while using the other. If the given weight vector is generic, the series solution for the corresponding initial ideal is obtained, otherwise the module prints an error message.

A.2. The `TriangulationToIdeal` and `IdealToTriangulation` external modules

These external modules are useful in mapping unimodular regular triangulations to the corresponding square-free initial ideals, and vice-versa. They rely on `Macaulay2` to function as intended.

Below, we describe the syntax for each of these external modules:

`TriangulationToIdeal[Triangulations, TriangNum]`

where, the input arguments are as follows:

- `Triangulations`: Result returned by `FindTriangulations`.
- `TriangNum`: Serial number of the unimodular regular triangulation for which the corresponding square-free initial ideal is to be derived.

---

and,

```
IdealToTriangulation[Ideals, IdealNum]
```

where, the input arguments are as follows:

- `Ideals`: Result returned by `FindInitialIdeals`.
- `IdealNum`: Serial number of the square-free initial ideal for which the corresponding unimodular regular triangulation is to be derived.

Appendix B. Global variables used in `FeynGKZ.wl`

Below, we enlist the global variables that have been used in our package. The user can modify some of them as per their convenience, while others are not meant for such usage.

- **FeynGKZPath**
  Set this variable such that it points to the directory containing `FeynGKZ.wl` and its dependencies `AMBREv2.1.1.m` and `Olsson.wl`.
  Note: the path must be passed as a string.
  By default, it points to `NotebookDirectory[]`.
- **PolyMakePath**
  Set this variable such that it points to the directory containing the `polymake` executable.
  Note: the path must be passed as a string.
  By default, it points to `/usr/bin`.
- **TOPCOMPath**
  Set this variable such that it points to the directory containing the `points2triangs` executable from `TOPCOM`.
  Note: the path must be passed as a string.
  By default, it points to `/usr/local/bin`.
- **Dim**
  Space-time dimension of the Feynman integral.
- **n**
  Summation indices.
- **z**
  Generic coefficients of the toric G-polynomial.

Appendix C. Feynman integrals in `Example.nb`

In this appendix, we provide the list of the Feynman integrals solved in `Examples.nb` using `FeynGKZ.wl`. The list is organised in terms of the codimension of the associated $\mathcal{A}$-matrices. All the integrals are evaluated in the $d = 4 - 2\epsilon$ dimension.

1. **Codim** $= 0$
   - *Massless Bubble*: Here, the massless one-loop bubble with off-shell external momentum is considered. We find the solution as a single term using the triangulation method.
   - *On-Shell Massless Triangle*: Here, the one-loop triangle integral with massless propagators and external momenta $p_1^2 = 0$, $p_2^2 = 0$ and $p_1 \cdot p_2 = -\frac{s}{2}$ is considered. We find the solution as a single term using the triangulation method.
2. **Codim** $= 1$
   - *One-Mass Bubble*: Here, the one-loop bubble with one massive propagator with off-shell external momentum is considered. The GD method is used to find the series solution.
   - *On-Shell Massless Box*: Here, the one-loop box integral with massless propagators and on-shell external momenta $p_1^2 = p_2^2 = p_3^2 = 0$ is considered. The triangulation method is used to find the series solution.
3. **Codim** $= 2$
   - *Off-Shell Massless Triangle*: Here, the one-loop triangle integral with massless propagators and off-shell external momenta is considered. The GD method is used to find the series solution.
   - *Two-Mass Bubble*: Here, the one-loop bubble with two massive propagators and off-shell external momentum is considered. The triangulation method is used to find the series solution.
   - *On-shell massive sunset*: Here, the two-loop sunset diagram with three massive propagators and on-shell external momentum $p^2 = 0$ is considered. The triangulation method is used to find the series solution.
4. **Codim** $= 3$
   - *Three-Mass Sunset*: Here, the two-loop sunset diagram with three massive propagators and off-shell external momentum is considered. The triangulation method is used to find the series solution.
5. **Codim** $= 4$
   - *Two-Loop Self-Energy with Four Propagators*: Here, the two-loop self energy integral with four propagators is considered. The triangulation method is used to find the series solution.

6. **Codim** = 5
   - *Off-Shell Massive Triangle*: Here, the one-loop triangle integral with three massive propagators and off-shell external momenta is considered. The triangulation method is used to find the series solution.

These examples were considered in [14–16,18] using the GKZ approach. The solutions of these Feynman integrals obtained from our package provide a crosscheck on the results presented in the literature and match numerically with FIESTA [45].

Appendix D. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2023.108699.

References

[1] V.A. Smirnov, Analytic Tools for Feynman Integrals, 2012.
[2] S. Weinzierl, Feynman integrals, arXiv:2201.03593 [hep-th], 2022.
[3] H.M. Srivastava, P.W. Karlsson, Multiple Gaussian Hypergeometric Series, 1985.
[4] V.A. Smirnov, Phys. Lett. B 460 (1999) 397–404, arXiv:hep-ph/9905323.
[5] J.B. Tausk, Phys. Lett. B 469 (1999) 225–234, arXiv:hep-ph/9909506.
[6] B. Ananthanarayan, S. Banik, S. Friot, S. Ghosh, Phys. Rev. Lett. 127 (15) (2021) 151601, arXiv:2012.15108 [hep-th].
[7] I. Gelfand, M. Graev, A. Zelevinsky, Dokl. Akad. Nauk SSSR 295 (1) (1987) 14–19.
[8] I. Gelfand, M.M. Kapranov, A.V. Zelevinsky, Discriminants, Resultants, and Multidimensional Determinants, 1994.
[9] I.M. Gelfand, M.M. Kapranov, A. Zelevinsky, Hypergeometric Functions, Toric Varieties and Newton Polyhehra, 1991.
[10] I. Gelfand, M.M. Kapranov, A. Zelevinsky, Adv. Math. 84 (1990) 255–271.
[11] M.Y. Kalmykov, B.A. Kniehl, Phys. Lett. B 714 (2012) 103–109, arXiv:1205.1697 [hep-th].
[12] F. Beukers, J. Reine Angew. Math. 2016 (2011) 183–206.
[13] R.N. Lee, A.A. Pomeransky, J. High Energy Phys. 11 (2013) 165, arXiv:1308.6676 [hep-ph].
[14] L. de la Cruz, J. High Energy Phys. 12 (2019) 123, arXiv:1907.00507 [math-ph].
[15] R.P. Klausen, J. High Energy Phys. 04 (2020) 121, arXiv:1910.08651 [hep-th].
[16] T.-F. Feng, C.-H. Chang, J.-B. Chen, H.-B. Zhang, Nucl. Phys. B 953 (2020) 114952, arXiv:1912.01726 [hep-th].
[17] T.-F. Feng, H.-B. Zhang, C.-H. Chang, Feynman integrals of Grassmannians, arXiv:2206.04224 [hep-th], 2022.
[18] T.-F. Feng, H.-B. Zhang, Y.-Q. Dong, Y. Zhou, GKZ-system of the 2-loop self energy with 4 propagators, arXiv:2209.15194 [hep-th], 2022.
[19] R.P. Klausen, J. High Energy Phys. 02 (2022) 004, arXiv:2109.07584 [hep-th].
[20] D. Agostini, C. Fevola, A.-L. Sattelberger, S. Telen, Vector spaces of generalized Euler integrals, arXiv:2208.08967 [math.AG], 2022.
[21] F. Tellander, M. Helmer, Cohen-Macaulay property of Feynman integrals, arXiv:2108.01410 [hep-th], 2021.
[22] U. Walther, On Feynman graphs, matroids, and GKZ-systems, arXiv:2206.05378 [math-ph], 2022.
[23] A. Klemm, C. Nega, R. Safari, J. High Energy Phys. 04 (2020) 088, arXiv:1912.06201 [hep-th].
[24] K. Bönisch, F. Fischbach, A. Klemm, C. Nega, R. Safari, J. High Energy Phys. 05 (2021) 066, arXiv:2008.10574 [hep-th].
[25] F. Loebbert, D. Müller, H. Münkler, Phys. Rev. D 101 (6) (2020) 066006, arXiv:1912.05561 [hep-th].
[26] V. Chestnov, F. Gasparotto, M.K. Mandal, P. Mastrolia, S.J. Matsubara-Heo, H.J. Munch, N. Takayama, Macaulay matrix for Feynman integrals: linear relations and intersection numbers, arXiv:2204.12983 [hep-th], 2022.
[27] M. Borinsky, Tropical Monte Carlo quadrature for Feynman integrals, arXiv:2008.12310 [math-ph], 2020.
[28] M. Saito, B. Sturmfels, N. Takayama, Gröbner Deformations of Hypergeometric Differential Equations, vol. 6, Springer Science & Business Media, 2013.
[29] J. Rambau, in: Proceedings of the International Congress of Mathematical Software, 2002.
[30] E. Gawrilow, M. Joswig, polymake: a Framework for Analyzing Convex Polytopes, 2000.
[31] J. Gluza, K. Kajda, T. Riemann, Comput. Phys. Commun. 177 (2007) 879–893, arXiv:0704.2423 [hep-ph].
[32] B. Ananthanarayan, S. Bera, S. Friot, T. Pathak, Olsson.wl: a *Mathematica* package for the computation of linear transformations of multivariable hypergeometric functions, arXiv:2201.01189 [cs.MS], 2021.
[33] E. Cattani, Three Lectures on Hypergeometric Functions, 2006.
[34] J. Stienstra, in: Instanbul 2005: CIMPA Summer School on Arithmetic and Geometry Around Hypergeometric Functions, 2005, arXiv:math/0511351.
[35] F. Beukers, Notes on A-Hypergeometric Functions, 2011.
[36] T. Hibi (Ed.), Gröbner Bases, Springer, Japan, 2013.
[37] C. Berkesch, J. Forsgård, M. Passare, Euler–Mellin Integrals and A-Hypergeometric Functions, 2011.
[38] J. Loera, J. Rambau, F. Santos, Triangulations. Structures for Algorithms and Applications. Vol. 25, 2010.
[39] R.R. Thomas, Lectures in Geometric Combinatorics. Vol. 33. Student Mathematical Library, American Mathematical Society, 2006.
[40] I.M. Gelfand, M.I. Graev, V. Retakh, Russ. Math. Surv. 47 (1992) 1–88.
[41] F.F. Knudsen, in: Toroidal Embeddings I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1973, pp. 109–164.
[42] W. Bruns, J. Gubeladze, Polytopes, Rings, and K-Theory, Springer Monographs in Mathematics, Springer, New York, 2009.
[43] M. Prausa, Eur. Phys. J. C 77 (9) (2017) 594, arXiv:1706.09852 [hep-ph].
[44] T. Regge, in: Battelle Rencontres, 1968, pp. 433–458.
[45] A.V. Smirnov, N.D. Shapurov, L.I. Vysotsky, Comput. Phys. Commun. 277 (2022) 108386, arXiv:2110.11660 [hep-ph].