SOFIA
Singularities of Feynman Integrals Automatized

Miguel Correia,[1] **Mathieu Giroux,**[1] **Sebastian Mizera**[2,3,4]

[1] *Department of Physics, McGill University*
Montréal, QC H3A 2T8, Canada

[2] *Department of Physics, Princeton University*
Princeton, NJ 08544, USA

[3] *Princeton Center for Theoretical Science, Princeton University*
Princeton, NJ 08544, USA

[4] *Institute for Advanced Study*
Einstein Drive, Princeton, NJ 08540, USA

`miguel.ribeirocorreia@mcgill.ca`, `mathieu.giroux2@mail.mcgill.ca`, `smizera@ias.edu`

Abstract

We introduce `SOFIA`, a Mathematica package that automatizes the computation of singularities of Feynman integrals, based on new theoretical understanding of their analytic structure. Given a Feynman diagram, `SOFIA` generates a list of potential singularities along with a candidate symbol alphabet. The package also provides a comprehensive set of tools for analyzing the analytic properties of Feynman integrals and related objects, such as cosmological and energy correlators. We showcase its capabilities by reproducing known results and predicting singularities and symbol alphabets of Feynman integrals at and beyond the high-precision frontier.

v1.0.0

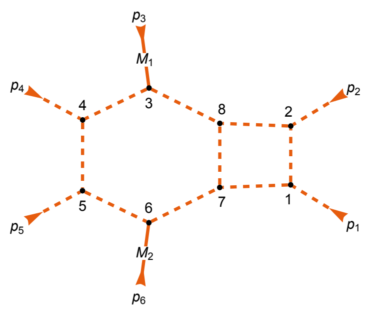
Contents

1 Introduction

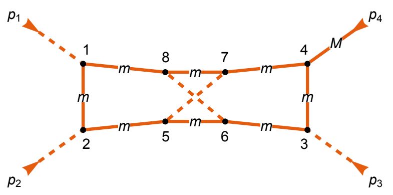
High-precision predictions for particle colliders rely more than ever on the computation of multi-loop Feynman integrals. For this reason, the study of Feynman integrals and their analytic structure has become a major area of research in theoretical particle physics (see, e.g., [1, 2] for reviews). A central challenge in these computations is the reliable prediction of Feynman integral singularities as functions of kinematic parameters, without the need for a direct evaluation of the integral. In this paper, we tackle this problem by introducing `SOFIA` (Singularities of Feynman Integrals Automatized), a MATHEMATICA package designed for this purpose, which we release for public use:

`SOFIA GitHub repository` [3].

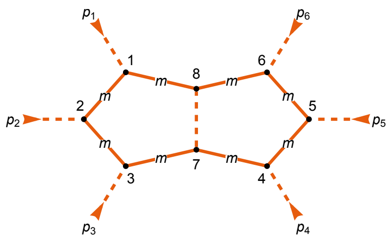
Differential equations. The most common application is in computing Feynman integrals using the method of differential equations [4]. In the simplest cases, these equations



(a) **135 singularities** (10m)
369 letters (84m)
`SolverBound -> 1000`



(b) **125 singularities** (11m)
237 letters (70m)
`SolverBound -> 100`



(c) **275 singularities** (10m)
? letters
`SolverBound -> 500`

Figure 1: Examples of cutting-edge diagrams and with timings (in minutes) to obtained singularities and letters. See [3] for expressions. Dashed lines denote massless particles and otherwise mass labels are displayed along the edges. The choice of `SolverBound` is determined by the maximum value for which the letters could be subsequently obtained. For diagram (c), although the singularity analysis finished in a reasonable time, the extraction of letters did not complete (hence the "?") for any reasonable choice of `SolverBound`. For diagram (a), we used a machine with two Intel Xeon Gold CPUs and 512GB memory, while for (b) and (c) we used a MacBook Pro with an M3 chip and 16GB memory.

take the form

$$\mathrm{d}\vec{I} = \sum_i \boldsymbol{\Omega}_i \,\mathrm{d}\log(W_i)\, \vec{I}\,, \tag{1.1}$$

where $\vec{I}$ is a vector of basis Feynman integrals with d refers to the differential in the kinematic variables. On the right-hand side, $\boldsymbol{\Omega}_i$ are matrices with entries rational in the kinematics and the spacetime dimension D. In fact, one often looks for bases in which all $\boldsymbol{\Omega}_i$ are linear in the dimensional-regularization parameter $\varepsilon = \frac{4-\mathrm{D}}{2}$ [5]. In this case, the resulting Feynman integrals $\vec{I}$ are often expressible in terms of multiple polylogarithms. The arguments of the logarithms, W_i, are known as the *letters* [6]. Being able to predict these letters is currently one of the main bottlenecks in computations of polylogarithmic integrals: once W_i's are known, one can fit the coefficients $\boldsymbol{\Omega}_i$ and consequently compute $\vec{I}$ using the differential equation together with some boundary conditions.

Many caveats and extra complications can arise in this story, but the motivation remains the same: We want to determine the set of all possible W_i's known as the *alphabet*. This application is the main motivation for our work.

Local behavior near singularities. Since problems involving analytic structure are notoriously subtle, let us be more precise about the type of questions that we aim to address. A *singularity* refers to a pole or a branch point in the (complexified) kinematic space. Let us collectively denote the kinematic variables as $\vec{s} \in \mathbb{C}^{\mathfrak{s}}$: they consist of the $\mathfrak{s}$ Mandelstam invariants and masses involved in the problem. While non-perturbatively scattering amplitudes might have a much richer singularity structure, in perturbation theory it is expected that singularities of the type

$$I_a \sim f_j^A \log^B f_j \,, \tag{1.2}$$

where $f_j = f_j(\vec{s})$ is a polynomial and A, B are rational numbers. For example, $\sim \sqrt{f_j} \log f_j$ is one possibility. Note that this refers to a *local* behavior of the amplitude near the locus $f_j = 0$. It is much simpler than the *global* behavior which has to be compatible with (1.2) for a list of all possible f_j's. Indeed, it is known that Feynman integrals are expressed in terms of complicated functions such as multi-polylogarithms, elliptic functions, periods of higher-genus Riemann surfaces and Calabi–Yau manifolds, etc. [7, 8] even though their local behavior (1.2) remains simple. The goal of singularity analysis is to exploit this simplicity.

The connection between the local behavior near singularities (1.2) and the global structure of differential equations (1.1) is very simple. It must be that zeros, poles, and singularities of W_i's are contained in the list of singularities f_j. Indeed, our goal will be to first compute a (possibly overcomplete) list of f_j's. Then, we will reconstruct a list of possible W_i's compatible with the aforementioned constraints.

Physical interpretation. In the literature, the aforementioned singularities $f_j = 0$ are also interchangeably known as *Landau singularities* or *anomalous thresholds* [9–12]. The equation $f_j = 0$ is a condition for "billiard ball scattering": a network of classical on-shell particles following their classical, though possibly complexified, paths. The simplest case corresponds to positive energies and real angles [13], i.e., the physical region. But the interpretation extends to the rest of the physical sheet, and in fact any sheet, after relaxing positivity and reality conditions; see, e.g., [14].

Recent progress. There has been an explosion of recent activity in the topic of determining computing singularities of Feynman integrals. Let us mention a couple of algorithms that have been implemented as parts of public packages and can handle state-of-the-art Feynman integrals. `HyperInt` implements a reduction algorithm that finds a superset of Landau singularities [15]. `PLD.jl` implements an algorithm based on toric geometry that finds a computable subset of Landau singularities [16, 17]. Both of these approaches work in the Schwinger-like parametrizations of Feynman integrals. A database comparing the two codes can be found in [18]. The package `Baikovletter` attempts to find symbol alphabets using Baikov representations [19]. We refer the readers to Sec. 2.2 for more detailed comparison and the introduction of [17] for a more complete list of references.

Importantly, it was recently observed that applying singularity analysis to the elastic unitarity constraint (relating the imaginary part of a diagram to its "cut" sub-diagrams) yields a powerful recursion formula for the calculation of Landau singularities in the loop momentum space [20]. This method enables efficient computation of the leading Landau singularities in two-particle reducible diagrams, successfully reproducing known results from the database [18] while also making new predictions, including all-loop examples.

In this paper, we extend this approach to any Feynman diagram. Building on the aforementioned advanced computational techniques and additional theoretical understanding, we propose an algorithm that offers a systematic way to study the full set of Landau singularities in general Feynman diagrams.

New contributions. The main goal of `SOFIA` is to streamline the analysis of singularities with minimal user input. The package comes with a companion Mathematica notebook containing 35 examples, where known results are reproduced and new predictions are given. This paper serves as a manual for `SOFIA`.

The first main novelty comes from a new variant of *Baikov representations* [21,22] adapted to minimize the number of integration variables and polynomials involved in the singularity analysis. `SOFIA` implements an optimization technique that results in a minimal integral representation (according to a metric explained in Sec. 2.1) without any user input besides the Feynman diagram under consideration. We believe it could come useful in other aspects of Feynman analysis that rely on Baikov representations, such as integration-by-parts reduction or analyzing Feynman geometries [23–32]. We illustrate the latter application in Sec. 4.2. Likewise, we extend our analysis to eikonalized Feynman integrals, such as those in post-Minkowskian expansions with an example in Sec. 4.3.

The second key innovation is the integration of advanced polynomial system solving methods for singularity analysis into the optimized Baikov representation. In particular, we implement a variant of the *Fubini reduction algorithm* [33] that results in an upper bound on the singularity locus. We also implement a Mathematica interface to `PLD.jl`, which allows one to seamlessly perform a numerical and symbolic analysis on a given diagram. We emphasize that this algorithm can be applied to any Euler integral and give a couple of examples of applications to cosmological and energy correlators in Sec. 4.4.

The third and final significant novelty is the implementation of graph-theoretical tools to break down and speed up the computation of singularities. It has long been observed that the symmetries of a Feynman diagram lead to an identification between Schwinger parameters of the leading physical sheet singularity [34, 35] (see [36] for a modern application in the context of the S-matrix bootstrap). In this work, we extend this idea by relating different sets of subleading singularities through the construction and identification of subtopologies that are connected via symmetry transformations accounting for variations in their kinematics. This procedure effectively mitigates the factorial growth in the number of subtopologies that would otherwise need to be analyzed.

Finally, we use the recent implementation of the publicly available packge `Effortless` [37] to provide an end-to-end function that takes a diagram as the input and provides a candidate symbol alphabet as the output. By "candidate" we mean that the alphabet might be overcomplete. An example is provided in Sec. 4.1.

We believe that `SOFIA` establishes substantial new milestones in the singularity analysis of Feynman integrals, producing results that were previously unattainable. The ancillary files [3] contain several new predictions for singularities and alphabets, including those presented in Fig. 1, which exemplify how SOFIA can be applied to next-frontier examples, namely, $HZ + 2$ jets, H+jet, and multi-jets production through light- (for (a)) or heavy-quark (for (b) and (c)) loop-induced gluon fusion at the LHC.

We hope that SOFIA serves as a tool to explore new avenues and sheds light on the analytic structures of perturbative scattering amplitudes that were previously inaccessible.

Outline. We provide theoretical background in Sec. 2, including the discussion of different variants of the Baikov representation, singularity analysis of integrals, and converting from singularities to symbol alphabets. The manual of SOFIA is given in Sec. 3 with an outline of all the functions and options. Sec. 4 focuses on examples and various applications. Finally, future directions are discussed in Sec. 5.

2 Theoretical background

The purpose of this section is to introduce three concepts: the Baikov representations, singularity analysis, and symbol alphabets.

2.1 Baikov representations

SOFIA is based on the *Baikov representation* of Feynman integrals. In this section, we illustrate two variations of the Baikov representation: global and loop-by-loop. We will illustrate them directly on the simple example of the sunrise diagram. More complicated examples will be studied in Sec. 4, and a detailed derivation of the Baikov representation can be found in App. A. All the (loop-by-loop) Baikov representations given below can be generated with the SOFIA package described in Sec. 3.3.

Normalization. Throughout this paper and in the package, we use the following conventions for scalar Feynman integrals

$$I_{\vec{\nu}}(\vec{s}) = \frac{1}{(i\pi^{\mathrm{D}/2})^L} \int \frac{\mathrm{d}^{\mathrm{D}}\ell_1\, \mathrm{d}^{\mathrm{D}}\ell_2 \cdots \mathrm{d}^{\mathrm{D}}\ell_L}{(q_1^2 - m_1^2)^{\nu_1}(q_2^2 - m_2^2)^{\nu_2} \cdots (q_N^2 - m_N^2)^{\nu_N}}\,, \tag{2.1}$$

where D is the spacetime dimension, ℓ_a denote the loop momenta, and q_e and m_e are the momenta and masses of the internal edges. The ν_i's are the (integer) powers of the propagators. The list $\vec{s}$ collectively denotes the external kinematic variables such as Mandelstam invariants and masses.

General strategy. The idea of Baikov representation is to translate the integration over the loop momenta ℓ_a into that over Lorentz products of the form $\ell_a \cdot p_i$ and $\ell_a \cdot \ell_b$, where p_i denote the external momenta in all-incoming conventions. Schematically, we want to convert the integration measure

$$\mathrm{d}^{\mathrm{D}}\ell_1\,\mathrm{d}^{\mathrm{D}}\ell_2\cdots\mathrm{d}^{\mathrm{D}}\ell_L \quad\mapsto\quad \mathcal{J}\,\mathrm{d}(\ell_1\cdot p_1)\,\mathrm{d}(\ell_1\cdot p_2)\cdots\mathrm{d}(\ell_1\cdot\ell_1)\,\mathrm{d}(\ell_1\cdot\ell_2)\cdots\,. \tag{2.2}$$

The change of variables comes with a Jacobian which we called $\mathcal{J}$. We can go one step further after we notice that the new variables are related to inverse propagators $x_i = (\ell_a + \ldots + p_i + \ldots)^2 - m_i^2$, which are always quadratic in the loop momenta, by a linear transformation. Therefore, we can express

$$\mathrm{d}^{\mathrm{D}}\ell_1\,\mathrm{d}^{\mathrm{D}}\ell_2\cdots\mathrm{d}^{\mathrm{D}}\ell_L \quad\mapsto\quad \tilde{\mathcal{J}}\;\underbrace{\mathrm{d}x_1\,\mathrm{d}x_2\,\ldots\,\mathrm{d}x_E}_{\text{inverse propagators}}\;\underbrace{\mathrm{d}x_{E+1}\,\ldots\mathrm{d}x_{\mathcal{N}}}_{\text{ISPs}}\,, \tag{2.3}$$

with yet another Jacobian $\tilde{\mathcal{J}}$. The caveat is that the number of variables $\mathcal{N}$ is in general larger than the number of propagators E. Therefore, $x_1, \ldots x_E$ label inverse propagators, while $x_{E+1}, \ldots, x_{\mathcal{N}}$ are the so-called irreducible scalar products (ISPs).

Global vs. loop-by-loop approaches. There are a few ways to implement the change of variables (2.3). The general idea dates back to the S-matrix theory in the 1960s [38–40], although in the more recent literature on Feynman integrals parametrizations of this type are referred to as Baikov representations [21]. In the *global* Baikov representation, one transforms all variables at the same time. This results in the representation of the form

$$I(\vec{s}) = c\,\tilde{\mathcal{G}}(\vec{s})^{\tilde{\gamma}} \int_{\Gamma'} \mathcal{G}(\vec{x};\vec{s})^{\gamma}\,\frac{\mathrm{d}^{N'}\vec{x}}{x_1^{\nu_1}x_2^{\nu_2}\cdots x_E^{\nu_E}}\,, \tag{2.4}$$

where $\vec{x} = (x_1, \ldots, x_E, \ldots, x_{N'})$ are the integration variables with

$$\mathcal{N} \mapsto N' = \frac{L(L+1)}{2} + L\mathcal{E}\,, \tag{2.5}$$

where $\mathcal{E}$ is the number of linearly-independent external momenta and $\vec{s}$ collectively denotes the external kinematic variables such as Mandelstam invariants and masses. The exponents γ, $\tilde{\gamma}$, as well as the prefactor c are expressed in terms of the topology of the diagram (number of edges, etc). as well as the spacetime dimension D. The derivation is given in App. A and the explicit expressions for all the ingredients in (2.4) are spelled out in (A.24).

The central object in the global Baikov representation is $\mathcal{G}(\vec{x},\vec{s})$ which arises as the general Jacobian for the change of variables. It can be expressed as a Gram determinant of a certain matrix. The integration contour Γ' is determined by $\vec{x}$ such that $\mathcal{G}(\vec{x},\vec{s}) > 0$. Likewise, $\tilde{\mathcal{G}}(\vec{s})$ is another Gram determinant that depends only on external kinematics and hence acts as an overall normalization. The derivation in App. A makes it clear that one can include different powers of propagators as well as non-trivial numerators with minimal modifications. For completeness, we review these aspects in App. A (see also [41, 42]).

The alternatives proposed by Frellesvig and Papadopoulos are *loop-by-loop Baikov representations* [23]. The idea is simply to Baikov parametrize each loop of the diagram sequentially in a given order. For each loop, we look for a shift in the loop momentum ℓ_a so that fewer invariants—such as $\ell_a \cdot p_i$ and $\ell_a \cdot \ell_b$—appear before changing variables from the ℓ's to the x's. This procedure has the effect of decreasing the total number of integration variables $N \leqslant N'$. Typically, the minimal/optimal N, is [43]

$$\mathcal{N} \mapsto N = L + \sum_{a=1}^{L} \mathcal{E}_a \,, \tag{2.6}$$

where $\mathcal{E}_a$ denotes the number of linearly independent external momenta associated with loop ℓ_a. Note that in (2.6), N scales *linearly* with L, whereas in (2.5), N' exhibits a *quadratic* dependence.

Loop-by-loop representations are not unique, as they depend on the exact change of variables. Therefore, one can distinguish between different *variants*, depending on how such choices are being made. Below, we will describe our optimization procedure. In any case, all the integral representations takes the general form:

$$I(\vec{s}) = c' \int_{\Gamma'} \tilde{\mathcal{G}}_1^{\tilde{\gamma}_1} \mathcal{G}_1^{\gamma_1} \tilde{\mathcal{G}}_2^{\tilde{\gamma}_2} \mathcal{G}_2^{\gamma_2} \cdots \tilde{\mathcal{G}}_L^{\tilde{\gamma}_L} \mathcal{G}_L^{\gamma_L} \frac{\mathrm{d}^N \vec{x}}{x_1^{\nu_1} x_2^{\nu_2} \cdots x_E^{\nu_E}} \,. \tag{2.7}$$

Structurally, the only difference is that it features L pairs of Gram determinants $\tilde{\mathcal{G}}_a$, $\mathcal{G}_a$ for every loop $a = 1, 2, \ldots, L$, instead of just one. Note that the order in which we label the loop momenta $\ell_1, \ell_2, \ldots, \ell_L$ plays a role. The integration contour is given by the intersection of regions defined by positivity conditions on internal-external Gram determinant ratios: $\Gamma' = \bigcap_{l=1}^{L} \{\mathcal{G}_l / \tilde{\mathcal{G}}_l > 0\}$. Details of the derivation are given in App. A.

Generalized cuts. Due to complicated integration contours, Baikov representations are typically not used in the explicit evaluation of Feynman integrals. However, they are extremely useful for integral reduction, integration-by-parts identities, singularity analysis, and other applications that rely on the simplicity of the integrand. The main advantage comes from the application of generalized unitarity. Generalized cuts can be taken by simply replacing the integration contour with residues around the relevant propagators (similarly, unitarity or Cutkosky cuts can be taken after an extra modification of the integration contour). For example, the *maximal cut* is obtained by

$$\int \mathrm{d}^{\mathcal{N}} \vec{x} \quad \mapsto \quad \frac{1}{(2\pi i)^E} \oint_{x_1=0} \oint_{x_2=0} \cdots \oint_{x_E=0} \int \mathrm{d}^{\mathcal{N}-E} \vec{x} \,, \tag{2.8}$$

in either global or loop-by-loop Baikov approach (i.e., $\mathcal{N} = N \,, N'$). This replacement exists if the original integration domain, Γ or Γ', restricted to $x_1 = x_2 = \ldots = x_{\mathcal{N}} = 0$ is non-empty (more on that is discussed in Sec. 3.3). Note that after taking the cut, $\mathcal{N} - E$ variables still have to be integrated over.

Implementation. As mentioned above, Baikov representations are *not* unique. They depend on the order and labeling of the loop momenta ℓ_a, as well as additional shifts.

We implement a search algorithm that optimizes for the most efficient loop-by-loop Baikov representation without any user input. It proceeds in two steps. First, when possible, the loops are ordered automatically so that the number of external momenta $\mathcal{E}_a$ (associated with each loop momentum ℓ_a, for $1 \leqslant a \leqslant L$) increases from the smallest to the largest.[1] For example, for the diagram in Fig. 1a, the algorithm selects the box (with $\mathcal{E}_1 = 3$) as the first loop and the hexagon (with $\mathcal{E}_2 = 5$) as the second. Second, the translation invariance of the integration measure is exploited by shifting the loop momenta, i.e., $\ell_a \mapsto \ell'_a = \ell_a + \lambda_a$, to eliminate as many scalar products as possible. An explicit example of such shifts is provided in the sunrise example below.

Comparison with `BaikovPackage`. Recently, a semi-automatized Mathematica package `BaikovPackage` for generating standard and loop-by-loop Baikov representations of Feynman integrals was released [22]. Unlike `SOFIA`, which by default selects the loop-by-loop representation minimizing the number of integration variables N, `BaikovPackage` requires the user to make that choice and specify the order and labeling of the loop momenta ℓ_a. Consequently, the outputs may differ depending on these selections.

Example. Let us illustrate the differences between the global and loop-by-loop approaches on (arguably) the simplest example: the sunrise diagram. To make expressions shorter, we take all internal masses equal m and external momentum $p^2 = s$. In this case, we have only three propagators, $E = 3$, with:

$$x_1 = \ell_1^2 - m^2\,, \qquad x_2 = \ell_2^2 - m^2\,, \qquad x_3 = (p - \ell_1 - \ell_2)^2 - m^2\,. \tag{2.9}$$

As predicted by (2.5) with $L = 2$ and $\mathcal{E} = 1$, expanding these propagators out, the result depends on $N' = 5$ internal kinematic invariants: $\ell_1 \cdot p$, $\ell_2 \cdot p$, $\ell_1 \cdot \ell_2$, ℓ_1^2, ℓ_2^2, in addition to the external parameters $\vec{s} = (s, m^2)$. The 5 internal variables can be traded for $\vec{x} = (x_1, x_2, x_3, x_4, x_5)$ by a linear change of variables, where x_4 and x_5 are arbitrarily-chosen ISPs, say

$$x_4 = \ell_1 \cdot p\,, \qquad x_5 = \ell_2 \cdot p\,. \tag{2.10}$$

This procedure results in the global Baikov parametrization (2.4) with

$$\tilde{\mathcal{G}}(s, m^2) = s\,, \quad \mathcal{G}(\vec{x}, s, m^2) = \det \begin{bmatrix} m^2{+}x_1 & * & * \\ \frac{1}{2}\left(x_3{-}x_1{-}x_2{-}m^2{-}s\right){+}x_4{+}x_5 & m^2{+}x_2 & * \\ x_4 & x_5 & s \end{bmatrix}\,, \tag{2.11}$$

[1] This strategy is inspired by the recursive elastic unitarity approach of [20], where the selection and ordering of cuts play a crucial role in determining the complexity of the singularity analysis. In [20], this selection was performed manually, whereas in the present work, the analogous loop-ordering process is fully automated.

where the "$*$" denote symmetric entries.

In contrast, loop-by-loop representations first parametrize all invariants associated with the first loop momentum ℓ_1. They are $\ell_1 \cdot (\ell_2 - p)$ and ℓ_1^2. Notice that the former is counted as one variable: $\ell_1 \cdot \ell_2$ and $\ell_1 \cdot p$ appear in the expression for the Feynman integral only in the combination $\ell_1 \cdot (\ell_2 - p)$. Hence, integrating out the first loop momentum would give a function of $(\ell_2 - p)^2$, which is treated as external kinematics from this point of view. When it comes to parameterizing the second loop momentum ℓ_2, we are thus left with only two kinematic invariants: $(\ell_2 - p)^2$ and ℓ_2^2. Consequently, the total number of variables required is $N = 4$. This is consistent with (2.6) for $L = 2$ loops and $\mathcal{E}_1 = \mathcal{E}_2 = 1$. Notably, this approach reduces the total number of variables compared to the global Baikov parametrization, which requires $N' = 5$ variables.

Equivalently, we could have first relabeled $\ell_2' = \ell_2 - p$, which turns the inverse propagators into

$$x_1 = \ell_1^2 - m^2 \,, \qquad x_2 = (\ell_2' + p)^2 - m^2 \,, \qquad x_3 = (\ell_1 + \ell_2')^2 - m^2 \,. \tag{2.12}$$

This makes it clear that only four invariants are needed: ℓ_1, $\ell_2'^2$, $\ell_1 \cdot \ell_2'$, and $p \cdot \ell_2'$. Taking $x_4 = \ell_2 \cdot p$ as the only ISP, the resulting representation has

$$\begin{aligned}
\tilde{\mathcal{G}}_1 &= m^2 + s + x_2 - 2x_4, \qquad & \mathcal{G}_1 &= \det \begin{bmatrix} m^2 + x_1 & * \\ \frac{x_3 - x_1 - x_2 - s - m^2}{2} + x_4 & m^2 + s + x_2 - 2x_4 \end{bmatrix}, \\
\tilde{\mathcal{G}}_2 &= s, & \mathcal{G}_2 &= \det \begin{bmatrix} m^2 + x_2 & * \\ x_4 & s \end{bmatrix}.
\end{aligned} \tag{2.13}$$

We now proceed to explain how the singularity analysis is carried out in Baikov space.

2.2 Singularity analysis: Fubini reduction algorithm

As a mathematical problem, singularity analysis amounts to analyzing the topology of the union of hypersurfaces

$$\mathcal{U}(\vec{x}, \vec{s}) \equiv \{g_1(\vec{x}; \vec{s}) = 0\} \cup \{g_2(\vec{x}; \vec{s}) = 0\} \cup \cdots \cup \{g_k(\vec{x}; \vec{s}) = 0\} \,, \tag{2.14}$$

in the complex space $\mathbb{C}^N$. Here, $\vec{x} \in \mathbb{C}^N$ denotes the integration variables and $\vec{s}$ denotes the collection of all the parameters, such as the kinematic invariants and the masses. Each of the k polynomials $g_i(\vec{x}; \vec{s})$ describes a singularity of the integrand (such as a pole) or an integration boundary. A singularity can arise only if the topology of (2.14) changes, e.g., when a mass shell degenerates into a lightcone or two mass shells pinch.

Example. For instance, loop-by-loop Baikov representations have

$$(g_1, g_2, \ldots, g_{2L-1}, g_{2L}, g_{2L+1}, \ldots, g_{2L+E}) = (\mathcal{G}_1, \tilde{\mathcal{G}}_1, \ldots, \mathcal{G}_L, \tilde{\mathcal{G}}_L, x_1, \ldots, x_E) \,. \tag{2.15}$$

The first $2L$ polynomials are the internal and external Gram determinants, while the last E are the inverse propagators (and possibly ISPs). We do not need any extra polynomial for the integration boundaries, since they are already described in terms of the Gram determinants.

As another example, on the maximal cut, we need E fewer polynomials:

$$(g_1, g_2, \ldots, g_{2L-1}, g_{2L}) = (\mathcal{G}_1, \tilde{\mathcal{G}}_1, \ldots, \mathcal{G}_L, \tilde{\mathcal{G}}_L)\Big|_{x_1=x_2=\ldots=x_E=0} . \tag{2.16}$$

In either case, one can add $x_{E+1}, \ldots, x_N$ to the list if one allows for negative powers of ISPs.

Singularity analysis. A conventional way to phrase this problem is to compute the Jacobian of the union of hypersurfaces:

$$J = [\nabla g_1 \; \nabla g_2 \; \ldots \; \nabla g_k] \, , \tag{2.17}$$

where $\nabla = [\partial_{x_1} \; \partial_{x_2} \; \ldots \; \partial_{x_N}]^\intercal$, and ask for what values of $\vec{s}$ the rank of J changes; see, e.g., [44, Ch. 9]. In other words, we are asking for at least one null vector $\vec{\beta} = [\beta_1, \beta_2, \ldots, \beta_k]$ of J. (When applied to the loop-momentum representation (2.1) with $g_i = q_i^2 - m_i^2$, β's have the physical interpretation of Schwinger proper times). This requirement can be written as the following system of equations:

$$\beta_1 \nabla g_1(\vec{x}, \vec{s}) + \beta_2 \nabla g_2(\vec{x}, \vec{s}) + \ldots + \beta_k \nabla g_k(\vec{x}, \vec{s}) = 0 \, , \tag{2.18}$$

where $\vec{x} \in \mathcal{U}$. We call the codimension-1 part of the solution set $\mathcal{S}(\vec{s})$ and write it as

$$\mathcal{S}(\vec{s}) = \{f_1(\vec{s}) = 0\} \cup \{f_2(\vec{s}) = 0\} \cup \cdots \cup \{f_K(\vec{s}) = 0\} \, . \tag{2.19}$$

The goal of singularity analysis is to convert $\mathcal{U}$ into $\mathcal{S}$. We have essentially converted a topological problem into an algebraic one. It has a certain combinatorial aspect since we can classify the solutions based on the number of $\{g_i = 0\}$ surfaces that intersect at $\vec{x}$. The problem becomes even more difficult when $\mathcal{U}$ has a complicated topology since one has to take into account its non-normal crossings through blow-ups.

Singularity analysis of this kind applied to Feynman integrals dates back to the pioneering work of Bjorken [9], Landau [10], and Nakanishi [11] who wrote systems of polynomial equations (2.18) in various parametrizations. These are nowadays known as *Landau equations* and serve as a prototype for the analysis of singularities. This analysis was made rigorous through the work of Pham, Thom, Whitney, and others with the application of algebraic geometry tools to finite Feynman integrals; see [45] for a review. While mathematically rigorous, these methods tend to explode in computational complexity in anything but the simplest examples; see, e.g., [46] for a recent implementation in `Macaulay2`. Hence, recent resurgence of interest in this topic aims to make the singularity locus more computable.

Numerical approaches. Let us briefly outline the scope of numerical tools that are currently available for cutting-edge Feynman integrals. The JULIA package `PLD.jl` [16,17] uses techniques from toric and polyhedral geometry to give a computable subset of the singularity locus $\mathcal{S}$. It also implements the Euler characteristic test, which is another way to probe the topology of the space $\mathcal{U}$. If for a given set of parameters $\vec{s}_*$, the absolute value of the Euler characteristic of

$$(\mathbb{C}^*)^N \setminus \mathcal{U}(\vec{x}, \vec{s}) \,, \tag{2.20}$$

drops compared to its generic value, then $\vec{s}_*$ belongs to the singular locus. This test provides a quick way to determine whether a given $\vec{s}_*$ belongs to the singular locus $\mathcal{S}$. `SOFIA` provides an interface to both functionalities of `PLD.jl`.

However, the package `PLD.jl` is based on toric geometry and hence uses $\mathbb{C}^* = \mathbb{C} \setminus \{0\}$ instead of $\mathbb{C}$. This is appropriate for parametric representations of Feynman integrals, but in the Baikov representation it has the effect of introducing all $\{x_i = 0\}$ in the hypersurface union $\mathcal{U}$. This is equivalent to putting ISPs with negative exponents.

Analytical approaches. Let us now outline the available analytical approaches that lie at the heart of `SOFIA`. In contrast to the aforementioned numerical approach, the goal will be to analytically compute a superset of the singularity locus $\mathcal{S}$. In other words, we expect to encounter certain "fictitious" polynomials $f_i(\vec{s})$ that do not satisfy the original system of equations (2.18). However, as we explain below, our algorithm is designed to minimize their occurrence.

Let us describe the *Fubini reduction algorithm*, which is a variation of [47, Sec. 4.2]. It circumvents the need to solve systems of equations such as (2.18). Instead, it proceeds by eliminating one variable x_i at a time. We thus construct the sets of polynomials $\mathcal{L}$, $\mathcal{L}_{x_i}$, $\mathcal{L}_{x_i,x_j}$, etc. recursively, where the subscript refers to the variables we already eliminated. At each step, we define

$$[\mathcal{L}_{x_i,\ldots,x_j}]_{x_k} \equiv \bigcup_{g\,\in\,\mathcal{L}_{x_i,\ldots,x_j}} \{D_{x_k}(g),\ R_{x_k}(\infty, g)\} \ \cup \bigcup_{g,\hat{g}\,\in\,\mathcal{L}_{x_i,\ldots,x_j}} \{R_{x_k}(g,\hat{g})\}\,. \tag{2.21}$$

On the right-hand side, $D_{x_k}(g)$ refers to the *discriminant* of f with respect to x_k, and $R_{x_k}(g,\hat{g})$ is the *resultant* of g and $\hat{g}$ with respect to x_k. Writing the polynomial $g = c_0 + c_1 x_k + \ldots + c_d x_k^d$, the resultant $R_{x_k}(\infty, g) = c_d$ equals to the coefficient of the highest power. The set $[\mathcal{L}_{x_i,\ldots,x_j}]_{x_k}$ might consist of reducible polynomials or repeated entries. We hence define $\mathcal{L}_{x_i,\ldots,x_j,x_k} = \mathrm{irred}[\mathcal{L}_{x_i,\ldots,x_j}]_{x_k}$ containing only the unique irreducible polynomials.

The three contributions in (2.21) have the following interpretations. $D_{x_k}(g)$ corresponds to self-pinching of $g = 0$, $R_{x_k}(\infty, g)$ gives the pinch at infinity, while $R_{x_k}(g,\hat{g})$ is associated with a pinch between two curves $g = 0$ and $\hat{g} = 0$.

One strategy in computing the singularity locus would be therefore to start with

$$\mathcal{L} = \{g_1, g_2, \ldots, g_k\}\,. \tag{2.22}$$

and then perform the reduction in some order:

$$\mathcal{L} \mapsto \mathcal{L}_{x_1} \mapsto \mathcal{L}_{x_1,x_2} \mapsto \ldots \mapsto \mathcal{L}_{x_1,x_2,\ldots,x_N}\,. \tag{2.23}$$

The result $\mathcal{L}_{x_1,x_2,\ldots,x_N}$ consists of polynomials that depend on $\vec{s}$ only. In general, it is a (large) superset of the singularity locus $\mathcal{S}$. However, the result of the computation depends on the specific order of the variables we chose in the reduction.

We refer to the difference between $\mathcal{L}_{x_1,x_2,\ldots,x_N}$ and $\mathcal{S}$ as *fictitious* singularities. They are artifacts of the reduction procedure and have no physical meaning. In general, we are not concerned with catching a few fictitious singularities, since the goal is to find a superset of $\mathcal{S}$ anyway. However, it is beneficial to minimize their number.

The Fubini reduction algorithm instructs us to scan over all choices of $\mathcal{L}_{x_1,x_2,\ldots,x_N}$ and take the intersection of their results, that is

$$\mathcal{L}_{\{x_1,x_2,\ldots,x_N\}} = \bigcap_{\sigma\in S_N} \mathcal{L}_{\sigma(x_1,x_2,\ldots,x_N)}\,, \tag{2.24}$$

where S_N denotes the permutation set of N labels (two polynomials are considered equivalent for the purposes of intersection if their vanishing locus is the same). In practice, a lot of the intermediate results can be recycled between each permutation. Hence, the optimal recursion is

$$\mathcal{L}_{\{x_1,\ldots,x_{k-1},x_k\}} = \bigcap_{i=1}^{k} \mathrm{irred}[\mathcal{L}_{\{x_1,\ldots,x_{k-1},x_k\}\setminus x_i}]_{x_i}\,, \tag{2.25}$$

where each $[\mathcal{L}_{\{x_1,\ldots,x_{k-1},x_k\}\setminus x_i}]_{x_i}$ is defined as in (2.21). The central difference between (2.24) and (2.25) is that the former intersects $N!$ sets of proposed singularities at the very end, while the latter does so at the intermediate stages. This comes with enormous computational benefit and provides a slimmer superset of the singular locus $\mathcal{S}$, often coinciding with $\mathcal{S}$. The equation (2.25) is implemented in SOFIA as the default solver option `FastFubini`.

Brown [33] and Panzer [48] have also proposed more sophisticated versions of the Fubini algorithm known as *compatibility graph reduction*. They are implemented in the Maple package `HyperInt` [15]; see [17, App. A] for a review. We found that our implementation of this approach was slower than (2.25) in practical applications, and is therefore not shipped with v1.0.0 of SOFIA.

Other approaches. Other approaches to determining symbol alphabets based on Landau analysis exist; for example, utilizing momentum twistor techniques or Schubert analysis; see, e.g., [49–51] for recent developments and more exhaustive lists of references. Out of the algorithms implemented as computer codes, the MATHEMATICA package `Baikovletter` [19, 52, 53] uses Baikov representations to attempt constructing symbol letters. Instead of a reduction algorithm, it aims to return a list of candidate symbol letters directly. There are instances in which the resulting alphabet is not complete; see, e.g., [54, 55].

2.3 From singularities to letters

If a Feynman integral is sufficiently simple, it can admit a representation as a linear combination of multiple polylogarithms. In such cases, there is a great interest in knowing a priori the list of possible *symbol letters* W_i, which translate to arguments of these polylogarithms; see, e.g., [6, 56, 57]. The collection of all independent letters is known as the *symbol alphabet* $\mathbb{A}$.

Before we move on, we note that even after identifying that a family of Feynman integrals includes integrals evaluating to non-polylogarithmic functions, knowledge of $\mathbb{A}$ remains valuable. This is because integrals in subsectors typically remain polylogarithmic (see [24, 30, 58] for explicit examples).

Even letters. As reviewed in the introduction, letters are closely related to singularities. By definition, the zeros, poles, and singularities of letters are singularities of Feynman integrals. The simplest example is $\log(W_i)$, which has branch points at $W_i = 0$ and $W_i = \infty$. Hence the symbol alphabet must at least contain all the singularities themselves:

$$(W_1, W_2, \ldots, W_K) = (f_1, f_2, \ldots, f_K) = \mathbb{A}^{\text{even}}\,. \tag{2.26}$$

(One can also consider ratios of polynomials of the form f_j/f_k.) These are sometimes called *even* letters and certainly constitute a subset of the alphabet $\mathbb{A}^{\text{even}} \subseteq \mathbb{A}$. The question therefore becomes how to leverage this information to recover the remaining part of $\mathbb{A}$.

Odd letters. We start by assuming the remaining letters, which we call *odd*, take the general form

$$W_i = \frac{P_i + R_i\sqrt{Q_i}}{P_i - R_i\sqrt{Q_i}}\,, \tag{2.27}$$

where $P_i = P_i(\vec{s})$, $Q_i = Q_i(\vec{s})$, and $R_i = R_i(\vec{s})$ are some homogeneous polynomials in $\vec{s}$, such that the mass dimensions match. We assume that Q_i is square-free, so that it does not have any factors outside the square root. This is of course just an ansatz, and one can extend it to more complicated forms if necessary. We can now apply the same analysis as before. First of all, W_i itself has a singularity at $Q_i = 0$. Therefore, we must have

$$Q_i = c_i \prod_{j=1}^{K} f_j^{e_{ij}}\,, \tag{2.28}$$

where $c_i \in \mathbb{Q}$ is a rational number and $e_{ij} \in \mathbb{Z}_{\geqslant 0}$ are non-negative integer powers. Next, we need to check that zero and poles are singularities, which leads to a similar constraint:

$$(P_i + R_i\sqrt{Q_i})(P_i - R_i\sqrt{Q_i}) = P_i^2 - R_i^2 Q_i = c'_i \prod_{j=1}^{K} f_j^{e'_{ij}}\,, \tag{2.29}$$

for some new sets of constants $c'_i \in \mathbb{Q}$ and exponents $e'_{ij} \in \mathbb{Z}_{\geqslant 0}$.

Differential equation. As another perspective on the same problem, let us insert the odd letter ansatz (2.27) into the differential equation (1.1). The relevant term is

$$\mathrm{d}\log(W_i) = \frac{P_i R_i \mathrm{d}Q_i - 2(\mathrm{d}P_i R_i - P_i \mathrm{d}R_i)Q_i}{(P_i^2 - R_i^2 Q_i)\sqrt{Q_i}}\,. \tag{2.30}$$

Hence, indeed, the only singular points of the differential equation are located at the positions of Landau singularities $f_j = 0$ for all possible j.

Therefore, knowing the symbol alphabet is not essential for constructing an ansatz for the differential equation (1.1) based solely on the Landau singularities f_j. However, such approach would lead to an unwieldy polynomial in the numerator. The symbol alphabet provides a practical way to reduce this ansatz into a more manageable form.

`Effortless` implementation. There are a few different strategies to finding odd letters. To our knowledge, they were first explored in [59]. We will follow [60, Sec. 3.4], since this is the algorithm implemented in the publicly available package `Effortless` [37, 61]. The package assumes that a list of all candidate polynomials Q_i, R_i in (2.27) and constants c'_i from (2.29) are given. It then attempts to construct all polynomials $R_i^2 Q_i + c'_i \prod_{j=1}^{K} f_j^{e'_{ij}}$ that are a perfect square P_i^2. This can be done by scanning over all choices of e'_{ij}. There is only a finite number of choices, since the set of singularities f_j is finite and the exponents are limited because the whole expression has to have a uniform mass dimension. Repeating the same exercise for all choices of Q_i, R_i, and c'_i results in the proposed odd letters. Further checking for linear dependence, one obtains a candidate odd alphabet $\mathbb{A}^{\text{odd}}$ and hence also $\mathbb{A} = \mathbb{A}^{\text{even}} \cup \mathbb{A}^{\text{odd}}$.

In practice, we consider all Q_i of the form

$$\sqrt{Q_i} = \sqrt{g_a}\sqrt{h_b}\sqrt{f_k}\sqrt{f_l}\,, \tag{2.31}$$

for every pair $k, l = 1, 2, \ldots, K$ with $R_i = 1$ and $c'_i = \pm 4$ (the default option in `Ef-`

`fortless`), as well as a possible user-specified set of polynomials $(g_1, g_2, \ldots, g_A)$ and $(h_1, h_2, \ldots, h_B)$ with $a = 1, 2, \ldots, A$ and $b = 1, 2, \ldots, B$, see Sec. 3.3.1 for details. Hence, in total, there are $A \times B \times K^2$ possibilities checked. The odd letter ansatz in SOFIA v1.0.0 is formulated using the root structure defined in (2.31).

There is a handful of examples of Feynman integrals in which letters more complicated than the ansatz (2.27) exist, for example those with nested square roots [62–64]. These are currently not implemented in `Effortless`.

3 Manual

This section covers how to install SOFIA and its dependencies, summarizes the key functions, and provides complete documentation.

3.1 Installation

As mentioned above, SOFIA is hosted in a GitHub repository at [3], where a compressed file with the latest version can be downloaded or cloned. The package requires a working installation of Mathematica. It was developed and tested using the Mathematica versions 13.2–14.0 and can be loaded by running the command:

```
Get["path/to/SOFIA.m"]
```

Alternatively, one may add the path to the package to the variable `$Path`, in which case the loading of the package amounts to calling `Get["SOFIA.m"]`. Running this command by itself uses the default options:

```
SOFIAoptionFiniteFlow = False;
SOFIAoptionJulia = False;
```

When an Internet connection is available, the `Effortless` package is automatically loaded from GitHub [61], ensuring that SOFIA always uses the latest version. If needed, users can overwrite this path with `SOFIAoptionEffortlessPath = "path/to/Effortless.m"` to point to, say, a locally cloned copy. To speed up the alphabet reconstruction with `Effortless`, we recommend enabling `FiniteFlow` [65] by setting `SOFIAoptionFinite-Flow = True` before running `Get["SOFIA.m"]`. Similarly, the Julia dependencies can be activated by setting `SOFIAoptionJulia = True`, although this option is typically unnecessary (more on that below). A short guide for setting up an optional link between Mathematica and Julia can be found in App. B.

3.2 Summary for people with busy schedules

We illustrate how to use the main functionality of SOFIA, which converts a diagram into a proposed list of singularities or candidate symbol letters depending on the options used.

Input. The input diagram is specified through the list of its `edges` and `nodes`. The list `edges` consists of tuples `{{i,j},m}` specifying that a vertex `i` and `j` are connected through a propagator with mass `m`. The order does not matter. The `nodes` is a list of tuples `{i,M}` specifying that an external momentum p_i with $p_i^2 = \mathtt{M}^2$ is attached to the vertex `i`. The order *does* matter here since we call the external momenta $p_1, p_2, \ldots$ in the order of appearance.

As an example, the "parachute" diagram shown at the top of Fig. 2a corresponds to

```
edges = {{{1, 2}, m}, {{1, 2}, m}, {{1, 3}, m}, {{2, 3}, m}};
nodes = {{1, M1}, {2, M2}, {3, M3}};
```

The code automatically assigns independent Mandelstam invariants of the form `sij...` $= (p_i + p_j + \ldots)^2$ in the cyclic basis. For example, 6-particle Mandelstam invariants are `s12`, `s23`, `s34`, `s45`, `s56`, `s16`, `s123`, `s234`, `s345` (we work in D dimensions). Alternatively, one can run the command:

```
{edges, nodes} = FeynmanDraw
```

which opens a window in which one can draw the diagram directly. The output of this command is the pair `{edges, nodes}` with generic masses. One can visually confirm that the diagram is correctly specified by plotting it with:

```
FeynmanPlot[{edges, nodes}]
```

which reproduces the top of Fig. 2a.

Note that the input masses in `{edges, nodes}` typically follow the format `M`, $\mathtt{M}_{\mathtt{i}}$ (equivalent to `Subscript[M, i]`), `m`, or $\mathtt{m}_{\mathtt{i}}$. However, as long as the JULIA dependencies are not enabled, users can choose their own variable names. The only effect is that the mass variables will not be automatically homogenized in terms of mass squared in the output of the singularity analysis performed by the functions we now describe.

SOFIA command: Most important options and output format. With this input, running SOFIA amounts to:

```
SOFIA[{edges, nodes}]
```

This command produces a list of candidate singularities. For the parachute diagram with `edges` and `nodes` given above, this yields the list at the top of Fig. 2b. The detailed structure of the output—including variations produced under different options—will be discussed later in this section.

The function `SOFIA[]` comes with many options that can customize the computation depending on the specific application. The most important options (and their default values) are:

```
SOFIA[{edges, nodes}, FindLetters -> False
                    , MaxCut -> True
                    , IncludeSubtopologies -> True
                    , Symmetries -> True
                    , SolverBound -> 100]
```

The complete list of options is given in Sec. 3.3.

Constructing candidate alphabets. When the `FindLetters -> True` option is enabled, `SOFIA[]` first extracts the kinematic singularities and then uses the publicly available package `Effortless` [61] to generate all candidate letters, both even and odd. The structure of the output is then a nested list, with independent even and odd letters in the first and second entries, and square roots in the third entry, respectively:

```
{{Even letters}, {Odd letters}, {Square roots}}
```

In contrast, with `FindLetters -> False`, the code simply returns the list of candidate singularities without generating any associated letters. As emphasized above, `SOFIA` looks only for candidate singularities and a candidate symbol alphabet, meaning that it might find fictitious terms (in the sense of Sec. 2.2) that are not true singularities or true letters of a given Feynman integral. ✧

Maximal cuts and subtopologies. The option `MaxCut -> True`, enabled by default, computes the leading singularities, which we take to be the singularities on the maximal cut (2.8). When `IncludeSubtopologies -> True` is also enabled, the calculation iterates over all subtopologies, therefore computing also subleading singularities.

To restrict the computation to leading singularities only, one can set `IncludeSubtopologies -> False` while keeping `MaxCut -> True`. Conversely, setting `MaxCut -> False` disables cuts entirely, computing both leading and subleading singularities using the top topology alone. However, this significantly increases runtime due to the much larger number of Baikov variables that must be eliminated by the polynomial solver. In this case, it is recommended to set `IncludeSubtopologies -> False`, as subleading singularities are already being analyzed. ✧

Symmetries. When `IncludeSubtopologies -> True`, the role of the option `Symmetries -> True` is to reduce the total number of subtopologies analyzed, by identifying those that are symmetric under a map between kinematic scales. These include transformations between internal and external masses, and Mandelstam invariants of the subtopologies. After the minimal set of subtopologies is analyzed, the singularities of the remaining ones are obtained via replacement rules. Thus, the option `Symmetries -> True` should reproduce the results as if all diagrams were considered, but with significantly improved efficiency. This is particularly valuable in high-multiplicity cases (i.e., six or more external particles), where it generally speeds up the `SOFIA[]` command by *at least* a factor of 2 to 10 in the examples considered. In general, its effect on reducing runtime is expected to increase with the number of external legs and loops. The specifics of how the symmetry map between subtopologies is constructed using the `Symmetry[]` function is discussed later in Sec. 3.3.4 below.

Enabling `Symmetries -> reflections` additionally accelerates the generation of subtopologies for highly symmetric graphs, as explained in more detail below. However, because the performance gain is graph-specific, the default value is set to `True` instead of `reflections`. ✧

Bound in the number of monomials. The option `SolverBound -> n` ensures that only intermediate polynomials $\mathcal{L}_{x_i,\ldots,x_j}$ in (2.21) with at most `n` monomials are considered in the elimination process. This option essentially controls the accuracy and speed of the computation. The higher `SolverBound` is, the more accurate the result becomes, but it will also take longer runtime. We recommend starting with a low value such as `SolverBound -> 100`. If the function returns results within a reasonable time, one should continue increasing `SolverBound`, as this generates a more complete list of singularities and letters if it is not already complete. The safest value is `SolverBound -> Infinity`. One of the few examples considered that required a high `SolverBound` is the "rocket diagram" from [54, Fig. 3c] covered in the notebook `SOFIA_examples.nb` [3]. ✧

3.3 Documentation of functions

The purpose of this section is to provide a more comprehensive documentation of all the most important functions included in SOFIA and their options. All these functions are internally memorized to improve performance, and in Sec. 4 we provide concrete examples that illustrate their usage.

3.3.1 `SOFIA` command

The singularities and associated alphabets are computed by the central function `SOFIA[]`, whose high-level behavior and performance may depend on several user-specified options. Below, we list all its available options together with the default values:

```
SOFIA[{edges, nodes}, FindLetters -> False
                    , IncludeSubtopologies -> True
                    , IncludeISPs -> False
                    , MaxCut -> True
                    , LoopEdges -> Automatic
                    , ShowPossiblyDegenerate -> False
                    , Symmetries -> True
                    , ShowHistoryGraph -> False
                    , UnclogTime -> 10^17
                    , StartAtSubtopology -> 1
                    , EndAtSubtopology -> -1
, Substitutions -> {}                12
, Solver -> FastFubini               13
, SolverBound -> 100                 14
, FactorResult -> True               15
, PLDMethod -> sym                   16
, PLDHomogeneous -> true             17
, PLDHighPrecision -> false          18
, PLDCodimStart -> -1                19
, PLDFaceStart -> 1                  20
, PLDRunASingleFace -> false         21
, AddSing -> {}                      22
, DoubleRoots -> {}                  23
, AddLetters -> {}]                  24
```

Lines 1–12 define options related to the Baikov representation, graph specifications, and scanning over subtopologies.

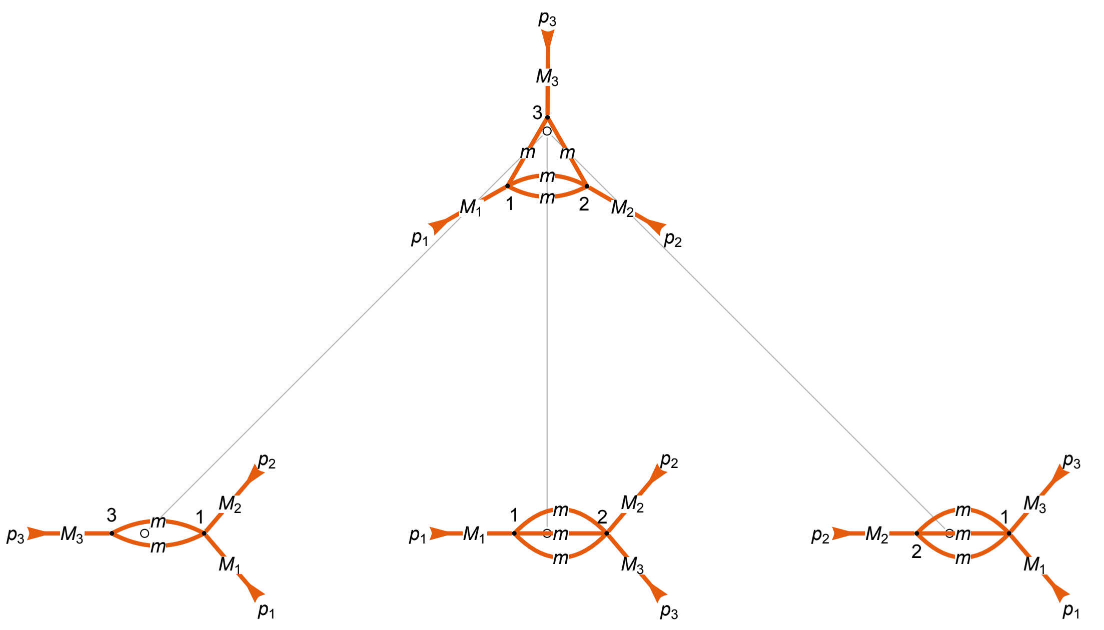
Baikov options. `LoopEdges` specifies which internal edges are assigned $\ell_1, \ldots, \ell_L$ before SOFIA searches for both an ordering and linear shifts in the ℓ's minimizing the number N of Baikov integration variables. For example, `LoopEdges -> {3,7}` means ℓ_1 and ℓ_2 are initially assigned to the 3rd and 7th edge in the list `edges`. If `LoopEdges -> Automatic`, this choice is automatically made by the package to minimize N. ✧

ISPs. Negative powers of ISPs are included in (2.15) if `IncludeISPs -> True` and are excluded otherwise. In other words, `IncludeISPs -> True` places ISPs in the denominators, while `IncludeISPs -> False` places them in the numerators, where they do not contribute to singularities. ✧

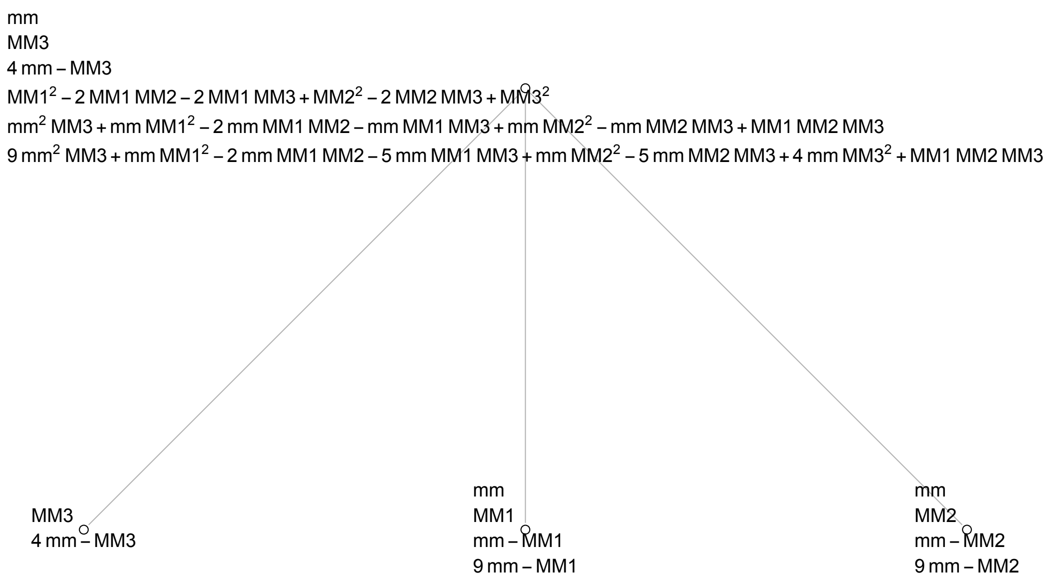
Skipping and targeting subtopologies. For examples that are still computationally expensive with `IncludeSubtopologies -> True` and `Symmetries -> True`, the user may want to set an upper bound on the time `T` (in seconds) taken by SOFIA to perform the singularity analysis. This can be done using the options `UnclogTime -> T`. The default value is the age of the Universe. If the analysis of a (sub)topology exceeds time `T`, the code will record the position of the clogging components, allowing the user to inspect them individually later and optimize performance by adjusting other options.

The options `StartAtSubtopology -> n` and `EndAtSubtopology -> m` allow the user to start and end the singularity analysis on the `n`-th and `m`-th subtopologies, respectively. ✧

Subtopology history graph. When `IncludeSubtopologies -> True`, the option `ShowHistoryGraph` can be used to print a top-bottom graph (laid out based on its spanning tree) whose nodes represent (sub)topologies, with edges connecting those related by edge contractions. If `ShowHistoryGraph -> True`, both the diagrams and the corresponding singularities are displayed at each node (see Figs. 2a and 2b, respectively); if `ShowHistoryGraph -> SingOnly`, only the singularities are shown.



(a)


mm
MM3
4 mm − MM3
MM1² − 2 MM1 MM2 − 2 MM1 MM3 + MM2² − 2 MM2 MM3 + MM3²
mm² MM3 + mm MM1² − 2 mm MM1 MM2 − mm MM1 MM3 + mm MM2² − mm MM2 MM3 + MM1 MM2 MM3
9 mm² MM3 + mm MM1² − 2 mm MM1 MM2 − 5 mm MM1 MM3 + mm MM2² − 5 mm MM2 MM3 + 4 mm MM3² + MM1 MM2 MM3
MM3
4 mm − MM3
mm
MM1
mm − MM1
9 mm − MM1
mm
MM2
mm − MM2
9 mm − MM2

(b)

Figure 2: Example output of `SOFIA[{edges, nodes}, ShowHistoryGraph -> True, SolverBound -> Infinity]` applied to the parachute diagram with `edges = {{{1,2},m}, {{1,2},m}, {{2,3},m}, {{1,3},m}}` and `nodes = {{1,M`$_1$`}, {2,M`$_2$`}, {3,M`$_3$`}}`. Panel (a) illustrates how diagrams are connected through edge contractions (with tadpoles removed), while panel (b) shows the corresponding candidate singularities. As explained in the main text, `SOFIA[]` uses the shorthands $\mathtt{mm} = \mathtt{m}^2$ and $\mathtt{MMi} = \mathtt{M}_\mathtt{i}^2$ in its outputs. Note that the singularities in (b) are in one-to-one correspondence with those in the database [18]—that is, there are no fictitious singularities in this example.

Numerical substitutions. Finally, `Substitutions` allows the user to substitute some of the kinematic invariants (e.g., `Substitutions -> {s12 -> 1}`). Substitutions are applied before the polynomial equations are solved. ✧

Degenerate cuts. As mentioned in the discussion following (2.8), computing the maximal cut of a valid loop-by-loop Baikov representation may result in a degenerate (e.g., measure zero) integration contour. `SOFIA[]` implements heuristics to detect such situations and prints a warning. Whenever this happens, an alternative choice of `LoopEdges` might be needed; see examples in [3]. In case of doubt, the warnings can be enabled by setting `ShowPossiblyDegenerate -> True`. ✧

Lines 13–21 give options that control the solving strategy.

Solver options. Within `SOFIA`, there are two available solvers: `FastFubini` (which implements the analytic procedure outlined in Sec. 2.2) and `momentumPLD` (which instead uses the numerical approach of Sec. 2.2). The former relies solely on MATHEMATICA functions, while the latter requires the working JULIA dependencies detailed in App. B. All the options `PLDname` relate specifically to `momentumPLD` (lines 13–18) are described under the same `name` as in [66]. Finally, the option `FactorResult -> False` can be used to skip the factorization into irreducible components of $\mathcal{S}(\vec{s})$ in (2.19), which, for large examples, can require both a long runtime and a large amount of memory.

We found that the native MATHEMATICA solver outperforms the JULIA solver on virtually all examples. Nevertheless, we keep the JULIA solver available as an option to, e.g., cross-check the results obtained from the Fubini reduction in cases where it is practical. ✧

Lines 22–24 set options for `Effortless`, which are relevant only when `FindLetters -> True` is enabled.

`Effortless` options. At times, users might want to supplement the singularities computed by the `SOFIA[]` command with additional ones. For example, if singularities of subtopologies are already known or anticipated and `IncludeSubtopologies -> False` is used. To do so, one can use the option `AddSing -> {p1,p2,...}` to append the specified polynomials to the singularities found by `SOFIA[]`.

By default (`DoubleRoots -> {}`), `SOFIA[]` uses $g_a = h_b = 1$ in (2.31) to construct the candidate odd alphabet $\mathbb{A}^{\text{odd}}$ from the ansatz in (2.27). The `DoubleRoots` option allows more flexibility. Specifying `DoubleRoots -> {{fm,..., fn},{fp,..., fq}}` forces `SOFIA[]` to consider in (2.31) the polynomials provided by the user (yet typically belonging to (2.26)): `{fm,..., fn}` for g_a and `{fp,..., fq}` for h_b. For instance, setting `Double-Roots -> {{f1, f2},{f1,f3,f4}}` ensures that $g_a = \texttt{f1}, \texttt{f2}$ and $h_b = \texttt{f1}, \texttt{f3}, \texttt{f4}, \texttt{f4}$ are considered in building the candidate $\mathbb{A}^{\text{odd}}$. Selecting `DoubleRoots -> {{fm, ..., fn}, All}` instructs `SOFIA[]` to assign the list `{fm,..., fn}` to g_a while using all f_i from (2.26) (which are internally constructed by `SOFIA[]`) for h_b.

In contrast, choosing `DoubleRoots -> All` directs `SOFIA[]` to apply all f_i from (2.26) to both g_a and h_b. In most cases, using the "sledgehammer" option `DoubleRoots -> All` comes at the cost of using significantly more computational resources. An example using the `DoubleRoots` option is discussed in Sec. 4.1.

The option `AddLetters -> {{Even letters},{Odd letters}}` allows the user to append additional even and odd letters to the list constructed by `Effortless`. The default value is the empty list `{}`. ✧

3.3.2 `SOFIADecomposeAlphabet` command

It is often useful to be able to check if two alphabets are equivalent. The command:

```
SOFIADecomposeAlphabet[alphabet1, alphabet2]
```

will check if (the differential of) each letter in the list `alphabet1` can be decomposed linearly in terms of that of `alphabet2`. The input format for the first entry `alphabet1` is a list of logarithms (for an explicit example, see `knownAlphabet` in Sec. 4.1). The second entry `alphabet2` should have the same format as the output of the `SOFIA[]` command when `FindLetters -> True`, that is `alphabet2 = {{Even letters}, {Odd letters}, {Square roots}}`. Consequently, one can directly set `alphabet2` = `SOFIA[...]`. This is useful to check whether the output of `SOFIA[]` spans an alphabet in the case that it is already known.

3.3.3 `SOFIABaikov` command

A loop-by-loop Baikov representation minimizing N' can be generated using:

```
1  SOFIABaikov[{edges, nodes}, LoopEdges -> Automatic
2                             , Dimension -> dim
3                             , MaxCut -> False
4                             , ShowXs -> False
5                             , ShowDetailedDiagram -> True]
```

The options in lines 1 and 3 work the same way as the options with the same names in the `SOFIA[]` command. To set the spacetime dimension, one uses `Dimension -> dim`. When restricting to an integer dimension (e.g., `dim` $= 4$), then either (2.7) or its maximal cut (2.8) might vanish. This can happen because the prefactors include Γ-functions that may diverge; see, e.g., (A.21), which could indicate a potential $\frac{\varepsilon}{\varepsilon}$ cancellation. In such cases, SOFIA will automatically display a warning and suggest to use dimensional regularization (e.g., `dim` $= 4 - 2\varepsilon$).

The Baikov variables, chosen internally by the code, can be printed in terms of loop and external momenta by setting `ShowXs -> True`. Enabling `ShowDetailedDiagram -> True` additionally displays these variables on the diagram specified by `{edges, nodes}`, along with the chosen internal edge momenta.

3.3.4 Graphical functions

SOFIA offers several functions that leverage graph-theoretical techniques to streamline and accelerate singularity analysis, which may also be useful for other purposes. In particular:

```
FeynmanDraw
FeynmanPlot[{edges,nodes}, EdgeLabels -> {},
                         , HighlightedEdges -> {}]
Subtopologies[{edges,nodes}, Reflections -> False]
Symmetry[{edges1,nodes1},{edges2,nodes2}]
SymmetryQuotient[ListOfDiagrams]
```

Drawing diagrams. The function `FeynmanDraw` opens a window that lets the user draw the desired Feynman diagram by clicking and dragging the pointer. It outputs a list of edges and nodes in the format `{edges,nodes}` with generic internal and external masses, which can be later adjusted by the user. By convention, the external momenta are all incoming and are assigned according to the ordering in `nodes`. The list `{edges,nodes}` completely specifies a Feynman diagram associated to (2.1), and is the main input of most functions in SOFIA. ✧

Plotting diagrams. A tuple of the form `{edges, nodes}` can be verified to represent the correct Feynman diagram by running `FeynmanPlot[{edges, nodes}]`. The options `EdgeLabels` and `HighlightedEdges` allow labeling each element of `edges` (for instance, with edge momenta or Baikov variables) and highlighting specific edges by specifying their positions in `edges`. By default, if no options are provided, internal edges are decorated with the masses from `edges` and external edges with the masses from `nodes`.

Generating subtopologies. Given a Feynman diagram represented as `{edges, nodes}`, the command `Subtopologies[{edges, nodes}]` generates a list of edges and nodes for each non-trivial subtopology of the original graph. It automatically removes contact diagrams, one-vertex reducible diagrams and tadpoles from the list of diagrams produced. For example, applying this command to any two-loop diagram will return a list of its one- and two-loop subtopologies, where the former arise from tadpole excisions—i.e., by removing internal edges of the form `{{i,i},_}` (see for example Fig. 2a).

The `Reflections -> True` option leverages `Symmetry[]` (discussed below) to concurrently eliminate subtopologies related by reflection transformations (i.e., symmetries that do not involve any kinematic replacement rules). This option is particularly effective when most subtopologies are reflection-symmetric. For example, for the triple-box diagram in `SOFIA_examples.nb` [3], this option is shown to improve the computation speed by approximately an order of magnitude. Note that behind the scenes, this is what the option `Symmetries -> reflections` is doing in the `SOFIA[]` command. ✧

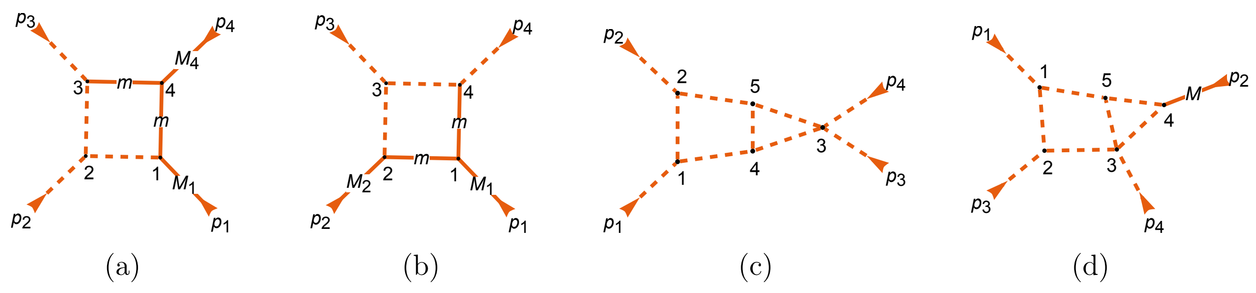


Figure 3: Diagrams related by the `Symmetry` function as described in the text. These diagrams were all generated by the `FeynmanPlot` function. The dashed lines denote massless particles and otherwise the mass label is displayed on the edges.

Symmetries. Given two Feynman diagrams represented as `{edges1, nodes1}` and `{edges2, nodes2}`, the command `Symmetry[{edges1,nodes1},{edges2,nodes2}]` identifies symmetry transformations relating them, accounting for masses and external momenta. If no symmetry exists, it returns an empty list: `{}`. Otherwise, it outputs `{{edges,nodes},rule}` where `{edges, nodes}` is one of the input diagrams and `rule` is the minimal replacement rule on the masses and Mandelstam variables required to relate it to the other diagram. In the above context, a reflection refers to a symmetry where `rule = {}`.

For example, Fig. 3 displays two pairs of diagrams (a-b and c-d) related by symmetry. Running `Symmetry[(a),(b)]`—with `(a)` and `(b)` representing the lists of edges and nodes of the diagrams of the same names—gives $\texttt{\{(a),\{}\mathtt{s}_{1,2}\to\mathtt{s}_{2,3},\mathtt{s}_{2,3}\to\mathtt{s}_{1,2},\mathtt{M}_1^2\to\mathtt{M}_2^2,\mathtt{M}_4^2\to\mathtt{M}_1^2\texttt{\}\}}$, which corresponds to a clockwise rotation of 90° relating diagram (a) to diagram (b).

Diagrams (c) and (d) have a less obvious relation. Running `Symmetry[(c),(d)]` outputs $\texttt{\{(d),\{}\mathtt{s}_{1,2}\to 0,\mathtt{s}_{2,3}\to 0,\mathtt{M}^2\to\mathtt{s}_{1,2}\texttt{\}\}}$ which effectively amounts to setting the momentum p_4 of diagram (d) to zero, $p_4 \to 0$. Indeed, in terms of Mandelstam variables, we have $p_4 \cdot p_1 = p_2 \cdot p_3 \propto s_{23} \to 0$ and $p_4 \cdot p_3 = p_1 \cdot p_2 \propto s_{12} \to 0$. In this case, `Symmetry[]` also correctly recognizes that the mass scale M^2 in diagram (d) plays the role of the Mandelstam invariant $s_{34} = s_{12}$ in diagram (c).

Note that the symmetry relation between diagrams is not necessarily bi-directional, as the example (d) $\to$ (c) demonstrates. We observe that `Symmetry[]` typically returns the diagram with the larger number of kinematic scales among the input diagrams. Furthermore, `Symmetry[]` can relate diagrams with different numbers of external edges. This includes cases where external momenta meet at a common vertex or where an external leg effectively "detaches" by setting its momentum exactly to zero.

In such cases, kinematic replacements can be subtle because the operations of specializing kinematics and computing singularities generally do not commute (a phenomenon analogous to that encountered when specializing kinematics and evaluating Feynman integrals; see [16,67,68]). To address this, we introduce a small parameter δ and extract the leading non-vanishing coefficient for each component of the singular locus $\mathcal{S}$ as $\delta \to 0$. See App. C for an explicit demonstration.

Before moving on, we introduce the higher-level command `SymmetryQuotient[L]`. This command takes a list of diagrams formatted as `L = {{edges1, nodes1}, {edges2, nodes2}, {edges3, nodes3}, ...}` and returns a minimal set of representative diagrams, along with the corresponding replacement rules for kinematic scales needed to reproduce all the diagrams in the input list. In the main command `SOFIA[]`, enabling the option `Symmetries -> True` internally applies `SymmetryQuotient` to the subtopologies generated by `Subtopologies`. ✧

3.3.5 Solvers

All the documented functions so far have been specifically designed for applications to Feynman integrals. However, the solvers used behind the scenes can handle a variety of other problems, some of which are explored in Sec. 4. They can be used independently of the earlier commands via:

```
SolvePolynomialSystem[polynomials, variables, Solver -> FastFubini
                                            , SolverBound -> 100
                                            , FactorResult -> True
                                            , PLDMethod -> sym
                                            , PLDHomogeneous -> true
                                            , PLDHighPrecision -> false
                                            , PLDCodimStart -> -1
                                            , PLDFaceStart -> 1
                                            , PLDRunASingleFace -> false]
```

Here, instead of the diagram information, the inputs consist of a set of `polynomials` and a set of `variables` to eliminate. The available options remain the same as above. In the case where `Solver -> momentumPLD`, the user must ensure that the variables to be eliminated are given as a text string in the form `variables = {x1, x2, x3, ..., xn}`. These variables are used to parametrize `polynomials` and are the default variables employed by the `SOFIA` wrapper for JULIA, i.e., they are predefined in the JULIA external evaluator and not using them would cause failure. While the `SOFIA` function ensures this automatically when called, this step must be handled manually when using `SolvePolynomialSystem`.

4 Detailed examples

We now demonstrate the explicit use of the functions introduced above with concrete examples that are not too large to fit within the text. Collectively, these examples took less than 30 seconds to run on a conventional laptop. A list of 30 additional examples, including many multi-scale and multi-loop ones, can be found in the MATHEMATICA notebook `SOFIA_examples.nb` [3].

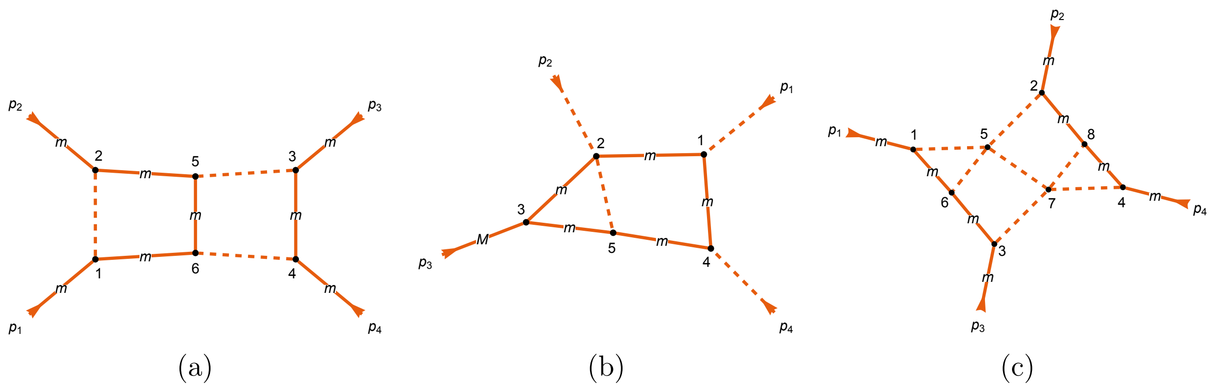

(a)
p_1 p_2 p_3 p_4 1 2 3 4 5 6 m
(b)
p_1 p_2 p_3 p_4 1 2 3 4 5 m M
(c)
p_1 p_2 p_3 p_4 1 2 3 4 5 6 7 8 m

Figure 4: Example diagrams considered in Sec. 4. These diagrams were all generated by the `FeynmanPlot` function. The dashed lines denote massless particles and otherwise the mass label is displayed on the edges.

4.1 Symbol alphabet

In this example, we illustrate how `SOFIA`, combined with the publicly available package `Effortless` [37], can be used to determine a (super)set of logarithmic letters, both even and odd, for a given Feynman integral.

We consider the diagram in Fig. 4a, which contributes to the t-channel planar Møller and Bhabha scattering amplitudes at $\mathcal{O}(e^6)$ in QED. The massive electrons and/or positrons are represented by solid lines, while the massless photon exchanges are depicted by dashed lines. This diagram was computed explicitly using differential equations in [69] and thus serves as an excellent crosscheck to the `SOFIA` implementation.

A candidate alphabet can be generated by defining the graph in terms of its nodes and edges and then running the `SOFIA[]` command as follows:

```
edges = {{{1,2},0}, {{2,5},m}, {{3,5},0}, {{3,4},m},
         {{4,6},0}, {{1,6},m}, {{5,6},m}};
nodes = {{1,m}, {2,m}, {3,m}, {4,m}};
extraRoots = {-s12, -s23, -s12-s23};
SOFIAalphabet = SOFIA[{edges, nodes}, FindLetters -> True
                                    , SolverBound -> Infinity
                                    , DoubleRoots -> {extraRoots, All}]
```

The output is a (super)set of 10 even and 42 odd candidate letters, which we collect in the alphabet `SOFIAalphabet[[1;;2]]` $= \{\{\tilde{W}_1, \ldots, \tilde{W}_{10}\}, \{\tilde{W}_{11}, \ldots, \tilde{W}_{52}\}\}$. We use the notation $\tilde{W}_i = \log(W_i)$. For example:

$$\tilde{W}_{31} = \text{Log}\left[\frac{\text{s12}\,(4\text{mm}^2+\text{mm}\,\text{s12}-\text{s12}\,\text{s23})+\text{r[[5]]}\,\text{r[[10]]}}{\text{s12}\,(4\text{mm}^2+\text{mm}\,\text{s12}-\text{s12}\,\text{s23})-\text{r[[5]]}\,\text{r[[10]]}}\right];$$
$$\tilde{W}_{32} = \text{Log}\left[\frac{\text{s12}\,(4\text{mm}-\text{s12})(3\text{mm}-\text{s23})+\text{r[[5]]}\,\text{r[[10]]}}{\text{s12}\,(4\text{mm}-\text{s12})(3\text{mm}-\text{s23})-\text{r[[5]]}\,\text{r[[10]]}}\right];$$
```
(* 50 more odd and even candidate letters *)
```

where the r[[i]]'s are square roots, which will be discussed below. The $\tilde{W}_i$ depend on $\mathfrak{s} = 3$ variables: s12, s23, and mm. We introduce mm = m^2 here because SOFIA internally homogenizes the output polynomials and thus automatically replaces the mass variables m with the square root of their squares $\sqrt{\texttt{mm}}$. In doing so, all the variables in the letters have the same mass dimension. In general, and as we will demonstrate explicitly below, not all of the letters in SOFIAalphabet are expected to appear in the differential equation (1.1).

It should also be noted that, in the example shown in Fig. 4a, if the odd letters are assumed to be constructible only from the square roots of individual candidate singularities—as is the case with the default value DoubleRoots -> {} turned on—the resulting alphabet would be incomplete.

To be more explicit, let us clearly distinguish the effect of working with the option DoubleRoots -> {extraRoots,All} instead of the default one. Running:

```
SOFIA[{edges, nodes}, (* same options as before *), DoubleRoots -> {}]
```

returns a set of odd letters that may depend on any of the 10 candidate square roots:

```
r0s = {Sqrt[mm],
       Sqrt[4 mm^3 - mm^2 s12 + 2 mm s12 s23 - s12 s23^2],
       (* 8 more single square roots *)};
```

found by SOFIA, but where each odd letter contains *at most* one square root. Now, keeping the same settings but with the replacement DoubleRoots -> {extraRoots, All}, SOFIA will (internally) add to this initial list of 10 allowed roots $10 \times 3 = 30$ double roots:

```
r = Flatten[{Map[Sqrt[extraRoots] #&, r0s], r0s}];
```

for a total of 40 (single *and* double) square roots to use in the construction of $\mathbb{A}^{\mathrm{odd}}$:

```
r[[5]] = Sqrt[-s12] Sqrt[4mm-s12];
r[[10]] = Sqrt[-s12] Sqrt[4 mm^3 - mm^2 s12 + 2 mm s12 s23 - s12 s23^2];
(* 38 more single and double square roots *)
```

As we shall show momentarily, r is sufficient for the SOFIA command to produce a complete basis of odd letters. (Note that the signs chosen in the DoubleRoots option above are purely conventional and ensure that the comparison below is performed using the root conventions of [69]).

At this stage, we can check the validity of SOFIAalphabet by comparing it against the alphabet knownAlphabet in [69, Eq. (2.8)]. This is done by running:

```
knownAlphabet = Log[{s12/mm, (s12 s23+r[[5]] r[[16]])/(s12 s23-r[[5]] r[[16]]), (* 14 more letters *)}];
SOFIADecomposeAlphabet[knownAlphabet, SOFIAalphabet]
```

In line 1 above, the syntax reflects that the roots defined in [69, Eq. (2.6)] were manually re-expressed, for comparison purposes, in terms of the roots `r[[i]]`. The `SOFIADecomposeAlphabet` command then checks whether it is possible to decompose each element of `knownAlphabet` in terms of `SOFIAalphabet` and returns the decomposition if it is. For example, the most complicated letter in `knownAlphabet` decomposes according to:

$$\mathrm{d}\log \frac{r_5 r_{10}\,(m^2{-}s_{12})\,{-}4m^6 s_{12}{-}3m^4 s_{12}^2{+}m^2 s_{12}^3{+}3m^2 s_{12}^2 s_{23}{-}s_{12}^3 s_{23}}{r_5 r_{10}\,(m^2{-}s_{12})\,{+}4m^6 s_{12}{+}3m^4 s_{12}^2{-}m^2 s_{12}^3{-}3m^2 s_{12}^2 s_{23}{+}s_{12}^3 s_{23}} = \mathrm{d}\tilde{W}_{31}{-}2\mathrm{d}\tilde{W}_{32}\,. \quad (4.1)$$

See the MATHEMATICA notebook `SOFIA_examples.nb` [3] for all the details of the complete decomposition, as well as how the roots r_i = `r[[i]]` in `r` and the letters $\tilde{W}_j$ in `SOFIAalphabet` are defined. In the same notebook, we show that the same procedure successfully reproduces other non-trivial alphabets, including those arising in the two-loop massless and one-mass pentagon functions [70–72] as well as the genuine two-loop massless six-particle letters [55, 73].

4.2 Feynman geometries

In the following example, we examine the diagram shown in Fig. 4b with a top-quark loop of mass m. This diagram is particularly interesting because it contributes, through the double-box containing it, to the $\mathcal{O}(g_s^4 m)$ amplitude for producing a Higgs boson of mass M and a jet via gluon fusion at the LHC.

An important early step in computing this amplitude is to determine the types of functions to which the contributing diagrams are expected to evaluate, i.e., the function space. This information is encoded in the underlying geometry of the associated master integrals. To reveal it, it suffices to carefully analyze the maximal cut of each master integral in four dimensions ($\varepsilon = 0$), which is conveniently done in the Baikov representation thanks to (2.8). This approach is reliable because the underlying geometry is characterized by the solution of the homogeneous part of the differential equation satisfied by the uncut integrals, which coincides with the differential equation governing the maximal cut integral [74, 75].

The maximal cut procedure *always* reduces Feynman integrals to periods over an algebraic variety. The geometry of this variety dictates the class of functions that appear after integration. For instance, a genus-0 (rational) curve gives rise to polylogarithms, a genus-1 (elliptic) curve to elliptic functions, and more intricate cases may involve higher-genus Riemann surfaces (see, e.g., [76]) or even higher-dimensional Calabi–Yau manifolds (see Sec.,4.3), pointing to increasingly rich—and still not fully understood—mathematical structures. Strictly speaking, this way, one can only place an upper bound on the complexity of a given Feynman geometry; verifying that it is not degenerate generally requires further analysis. (To our knowledge, there is currently no automated algorithm to *systematically* determine the precise function space of a Feynman integral, e.g., whether it is polylogarithmic, elliptic, or beyond. Even less is known at the amplitude level, where full cancellations among non-polylogarithmic sectors may occur; see, e.g., [77]).

For the diagram in Fig. 4b, we can use SOFIA to learn about its geometry. Computing its maximal cut amounts to running:

```
edges = {{{1,2},m}, {{2,3},m}, {{3,5},m}, {{4,5},m}, {{1,4},m}, {{2,5},0}};
nodes = {{1,0}, {2,0}, {3,M}, {4,0}};
SOFIABaikov[{edges, nodes}, MaxCut -> True, Dimension -> 4]
```

The result, obtained in a fraction of a second, is

$$\frac{-1}{256\pi^9}\int \frac{\mathrm{d}x_7}{\sqrt{\prod_{i=1}^4 (x_7 - r_i)}}\,, \qquad \text{where} \quad \begin{bmatrix} r_{1,2} = (m \pm M)^2 \\ r_{3,4} = m^2 + s_{12} \pm \frac{2m\sqrt{s_{12}s_{23}(s_{12}+s_{23}-M^2)}}{s_{23}} \end{bmatrix}\,. \tag{4.2}$$

The zero locus of the *irreducible* quartic in x_7, $P_4(x_7) = \prod_{i=1}^4 (x_7 - r_i)$, in the denominator *defines* the elliptic curve controlling the function space of this integral.

From (4.2), it is also easy to read off the leading singularities of the diagram in Fig. 4b. Indeed, these are given by the zeroes of the discriminant of $P_4(x_7)$ in x_7, i.e.,

$$\begin{aligned} D_{x_7}P_4 = m^2M^2 s_{12}s_{23}(M^2-s_{12})(M^2-s_{12}-s_{23})[16m^4(s_{12}+s_{23})^2+s_{23}^2(M^2-s_{12})^2 \\ -8m^2 s_{23}(M^2(s_{23}-s_{12})+s_{12}(s_{12}+s_{23}))]\,. \end{aligned} \tag{4.3}$$

These singularities can, of course, also be found automatically by running:

```
SOFIA[{edges, nodes}, MaxCut -> True]
```

One can verify that all the singularities in the ε-factorized differential equation from [78] are included in the resulting list available in [3].

4.3 Eikonal/Regge/heavy-mass physics

This example considers the eikonal/Regge scattering of two heavy masses—e.g., black holes or neutron stars, treated as point particles—via graviton exchange. In particular, we focus on the diagram shown in Fig. 4c, which contributes to the process at $\mathcal{O}(G_N^5)$. This example demonstrates two key points: first, how to take the eikonal limit of the Baikov representation; and second, how the latter turns out sufficient to uncover the underlying Calabi–Yau threefold geometry [43, 79].

In order to take the eikonal/Regge/heavy-mass expansion and keep the leading order term which captures the relevant classical physics [80], we first parametrize the momenta in Fig. 4c such that

$$p_1 = \overline{p}_1 - \frac{q}{2}, \qquad p_3 = -\overline{p}_1 - \frac{q}{2}, \qquad p_2 = \overline{p}_2 + \frac{q}{2}, \qquad p_4 = -\overline{p}_2 + \frac{q}{2}\,. \tag{4.4}$$

Recall that we use all-incoming conventions for the momenta p_i. The spacelike momentum transfer q satisfies the transverse condition $q \cdot \overline{p}_i \equiv 0$ for all i as and is normalized so that $q^2 \equiv -1$. Setting $\overline{p}_i = \overline{m} u_i$, with $u_i \cdot u_i \equiv 1$ and $u_1 \cdot u_2 \equiv y$, the loop-by-loop representation can be obtained in two steps.

First, we run the **SOFIABaikov** command on the diagram with the original kinematics. This can be done with:

```
edges = {{{1,5},0}, {{2,5},0}, {{5,6},0}, {{5,7},0}, {{7,8},0}, {{3,7},0},
         {{4,7},0}, {{1,6},m}, {{3,6},m}, {{2,8},m}, {{4,8},m}};
nodes = {{1,m}, {2,m}, {3,m}, {4,m}};
maxCut = SOFIABaikov[{edges, nodes}, MaxCut -> True
                                   , Dimension -> 4
                                   , ShowXs -> True]
```

The output **maxCut** is an expression for the maximal cut of this diagram. The option **ShowXs** prints also the definition of all the x_i's, including the leftover integration variables x_{12}, x_{13}, x_{14} we get after cutting all the $x_1, x_2, \ldots, x_{11}$. In particular, we find

$$x_{12} = \ell_4^2\,, \qquad x_{13} = \ell_3^2\,, \qquad x_{14} = \ell_4 \cdot p_1 = \overline{m}\,\underbrace{\ell_4 \cdot u_1}_{\hat{x}_{14}} + \ldots\,. \tag{4.5}$$

The assignment of the loop momenta was automatic, but the precise knowledge of which momentum is which does not affect the analysis. We find:

$$\texttt{maxCut} = \frac{1}{2^{15}\pi^{17}} \int \frac{\mathrm{d}x_{12}\,\mathrm{d}x_{13}\,\mathrm{d}x_{14}}{\sqrt{x_{12}(4m^2 - x_{12})}\sqrt{x_{13}(4m^2 - x_{13})}\sqrt{P_2}\sqrt{P_4}}\,, \tag{4.6}$$

where P_2 and P_4 are two polynomials with degrees 2 and 4 in the x_i's respectively.

Implementing the eikonal limit is a matter of substituting the kinematic variables and expanding in large $\overline{m} = \texttt{mb}$. It can be implemented with:

```
eikonal = {s1,2 -> 2 (1 + y) mb^2,
           s2,3 -> 2 (1 - y) mb^2,
           m -> Sqrt[mb^2 - 1/4],
           x14 -> x14h mb};
maxCut = Series[Applyd[maxCut/.eikonal, {x12, x13, x14h}]
                                    , {mb, ∞, 4}, Assumptions -> mb > 0]
```

The list **eikonal** defines the substitutions, and the **SOFIA** function **Applyd** implements the Jacobian for the change of variables to $(x_{12}, x_{13}, \hat{x}_{14}) = (\texttt{x12}, \texttt{x13}, \texttt{x14h})$. The leading order of this expansion gives:

$$\texttt{maxCut} = -\frac{1}{2^{19}\pi^{17}\overline{m}^4} \int \frac{\mathrm{d}x_{12}\,\mathrm{d}x_{13}\,\mathrm{d}\hat{x}_{14}}{\sqrt{x_{12}}\sqrt{x_{13}}\sqrt{\hat{P}_2}\sqrt{\hat{P}_4}} + \mathcal{O}(\overline{m}^{-5})\,, \tag{4.7}$$

where the polynomials are given by

$$\hat{P}_2 = 4\hat{x}_{14}^2 + (y^2-1)(1+x_{12})^2\,, \tag{4.8}$$

$$\hat{P}_4 = \hat{x}_{14}^2(1+x_{13})^2 - x_{12}x_{13}(1+x_{12}+x_{13})\,. \tag{4.9}$$

The curve $\hat{P}_4(x_{12}, x_{13}, \hat{x}_{14}) = 0$ defines a Calabi–Yau threefold (as a quartic polynomial in $\mathbb{CP}^3$ [81]), in agreement with the results of [43, 79].

The singularities of the resulting integral are simple to analyze. In SOFIA, it boils down to running:

```
SolvePolynomialSystem[{x12, x13, P̂2, P̂4}, {x12, x13, x14h}
                                          , SolverBound -> Infinity]
```

The result gives singularities at $y = 0$ and $y = \pm 1$, corresponding physically to the high-energy and static limits, respectively.

4.4 Beyond Feynman integrals

In this subsection, we illustrate how SOFIA can be used to study singularities of other classes of Feynman-like integrals.

4.4.1 Energy-energy correlators

As an example, we focus on the type of integrals arising in the computation of the four-point energy-energy correlator (EEC) in the collinear limit within $\mathcal{N} = 4$ super Yang-Mills theory [82]. Physically, the EEC captures meaningful correlations of energy flux after the emission of four final-state particles from an initial energetic parton and is now an active topic of research in both theoretical and experimental high-energy physics, see, e.g. [83, 84].

For four-point EEC, the relevant integrals to compute originate from a phase-space integral over a squared $1 \to 4$ amplitude [82, Eqs. (2-4)]. For our discussion, it is sufficient to quote the two families of integrals considered in [82]:

$$A_{a_1\dots a_7|a_8a_9a_{10}} \equiv \int_0^\infty \mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3\mathrm{d}x_4 \frac{\delta[1-(x_1+x_2+x_3+x_4)]}{s_{1234}^{a_1}s_{123}^{a_2}s_{234}^{a_3}x_{1234}^{a_4}x_{234}^{a_5}x_{123}^{a_6}x_{34}^{a_7}s_{12}^{a_8}s_{23}^{a_9}s_{34}^{a_{10}}}\,, \tag{4.10a}$$

$$B_{a_1\dots a_4|a_5a_6} \equiv \int_0^\infty \mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3 \frac{\delta[1-(x_1+x_2+x_3)]}{s_{123}^{a_1}x_{12}^{a_2}x_{23}^{a_3}x_{123}^{a_4}s_{12}^{a_5}s_{23}^{a_6}}\,, \tag{4.10b}$$

where $a_i \in \mathbb{Z}$ and the variables entering (4.10) are

$$s_{l_1l_2\cdots l_n} \equiv \sum_{1\leqslant i<j\leqslant n} x_{l_i}\,x_{l_j}\,(z_{l_i}-z_{l_j})\,(\bar{z}_{l_i}-\bar{z}_{l_j})\,, \qquad x_{l_1l_2\cdots l_n} \equiv \sum_{i=1}^{n} x_{l_i}\,, \tag{4.11a}$$

$$\text{with} \qquad (z_1,\bar{z}_1,z_2,\bar{z}_2,z_3,\bar{z}_3,z_4,\bar{z}_4) = \left(0,0,z,\bar{z},1,1,\frac{z-w}{1-w},\frac{\bar{z}-\bar{w}}{1-\bar{w}}\right)\,. \tag{4.11b}$$

We treat z, $\bar{z}$, w, $\bar{w}$ as independent variables. The parametrization of these families is overcomplete since, e.g., a_2 and a_3 in (4.10) cannot be simultaneously positive. This would introduce an overlapping pole $\frac{1}{x_{12}x_{23}}$ which is forbidden by a version of Steinmann relations. We thus find it simpler to perform the singularity analysis at the level of master integrals introduced in [82, App. B] (in the same way we typically do the analysis at the level of individual Feynman diagram topologies, as opposed to all integrals at a given loop order at the same time).

The complete analysis is given in [3]. Below, we illustrate the procedure on a simple master integral

$$B_1 \equiv |z_1 - z_2|^2 \int_0^\infty \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \frac{x_1 \delta[1-(x_1+x_2+x_3)]}{s_{123}\, x_{123}} \,. \tag{4.12}$$

As explained in Sec. 2.2, to find the singular locus of this master integral, we can simply apply the Fubini reduction algorithm to the system of polynomials

$$\mathcal{L} = \{s_{123}, x_{123}, x_1, x_2, x_3\} \,, \tag{4.13}$$

which explicitly can be inputted as:

```
L = {z zb (x1 x2 + x2 x3) + x2 x3 (1 - z - zb) + x1 x3,
     x1 + x2 + x3,
     x1,
     x2,
     x3};
```

Note that x_1 , x_2 and x_3 are included in $\mathcal{L}$ since $x_1 = x_2 = x_3 = 0$ appear in the integration boundaries of the B family in (4.10b). Within SOFIA, the singularities are computed by running:

```
SolvePolynomialSystem[L, {x1, x2, x3}, SolverBound -> Infinity]
```

The output is

$$\{z,\ \bar{z},\ 1-z,\ 1-\bar{z},\ z-\bar{z},\ 1-z\bar{z},\ z+\bar{z}-z\bar{z},\ z+\bar{z}-2z\bar{z}\} \,. \tag{4.14}$$

Looping this command over all the master integrals in [82, App. B] immediately reproduces the polynomial part of the alphabet quoted in that reference, namely

$$\begin{gathered}\{z\,,\overline{z}\,,1-z\,,1-\overline{z}\,,z-\overline{z}\,,w\overline{z}-\overline{w}z\,,w\overline{z}-\overline{w}\,,z\overline{w}-w\,,1-w-\overline{w}+z\overline{w}\,,\\ 1-w-\overline{w}+w\overline{z}\,,|z|^2-|w|^2\,,z-|w|^2\,,\overline{z}-|w|^2\}\cup(z\leftrightarrow w)\,.\end{gathered} \tag{4.15}$$

Notably, the analysis over the master integrals

$$\{A_{22}, A_{23}\} \equiv |z_2 - z_3|^4 \int_0^\infty \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \mathrm{d}x_4 \frac{\{x_2, x_3\}\delta[1-(x_1+x_2+x_3+x_4)]}{s_{123}\, s_{234}\, x_{1234}} \,, \tag{4.16}$$

returns a distinctly complicated letter (see `SOFIA_examples.nb` [3] for details):

$$L = w^4\overline{w}^2z^2 + 2w^3\overline{w}^3z^2 - 2w^3\overline{w}^2z^2 + w^2\overline{w}^4z^2 - 2w^2\overline{w}^3z^2 - 2w^2\overline{w}^2z^3 + (151 \text{ terms})\,. \tag{4.17}$$

It has an interesting geometric interpretation. As explained in [82], the solutions to the maximal cut conditions of the integrals in (4.16) (i.e., $s_{123} = s_{234} = x_{1234} = 0$) define a singular cubic curve after projection onto the (x_2, x_3)-plane. Its roots $a\,, b\,, c$, which give rise to the *non-integer* polynomial part of the alphabet listed in [82, Eq. (12)], satisfy

$$F = \left\{abc = -|z|^2|w|^2\,,\; a{+}b{+}c = 1{-}z{-}\overline{z}{-}w{-}\overline{w}\,,\; \frac{1}{a}{+}\frac{1}{b}{+}\frac{1}{c} = 1{-}\frac{1}{z}{-}\frac{1}{\overline{z}}{-}\frac{1}{w}{-}\frac{1}{\overline{w}}\right\}. \tag{4.18}$$

The discriminant $D_X V$ of the cubic polynomial V formed by these roots (given by Vièta's formula)

$$V = X^3 - F_2X^2 + F_1F_3X - F_1 = 0\,, \tag{4.19}$$

is precisely (4.17):

$$D_X V = [(a{-}b)(b{-}c)(c{-}a)]^2 = L\,. \tag{4.20}$$

4.4.2 Cosmological correlators

Similar singularity analysis can also be applied to cosmological correlators. Let us consider an example of a triangle diagram (also known as the one-loop three-site graph) for the conformally coupled scalar in Friedmann–Lemaître–Robertson–Walker cosmology. Following the notation of [85], the corresponding correlator expressed in Baikov representation is proportional to

$$\int \frac{y_{12}y_{23}y_{31}\mathrm{d}y_{12}\,\mathrm{d}y_{23}\,\mathrm{d}y_{31}}{q_{\mathfrak{g}_1}q_{\mathfrak{g}_2}q_{\mathfrak{g}_3}}\kappa^{\chi}\left[\frac{1}{q_{\mathcal{G}_{12}}}\left(\frac{1}{q_{\mathfrak{g}_{23}}}+\frac{1}{q_{\mathfrak{g}_{31}}}\right)+\text{cyclic}\right], \tag{4.21}$$

where

$$\begin{aligned} q_{\mathfrak{g}_j} &= x_j + y_{j-1,j} + y_{j,j+1}\,, && (4.22a)\\ q_{\mathfrak{g}_{j,j+1}} &= x_j + x_{j+1} + y_{j-1,j} + y_{j+1,j+2}\,, && (4.22b)\\ q_{\mathcal{G}_{j,j+1}} &= x_1 + x_2 + x_3 + y_{j,j+1}\,, && (4.22c) \end{aligned}$$

with the indices defined modulo 3. In addition, κ is proportional to the Cayley–Menger determinant

$$\kappa \propto \det \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & y_{12}^2 & y_{23}^2 & y_{31}^2 \\ 1 & y_{12}^2 & 0 & x_2^2 & x_1^2 \\ 1 & y_{23}^2 & x_2^2 & 0 & x_3^2 \\ 1 & y_{31}^2 & x_1^2 & x_3^2 & 0 \end{bmatrix}, \tag{4.23}$$

and χ is a generic parameter related to the type of cosmology. Physically, the x_i's are variables measuring energies of the external particles.

The singularity analysis of this integral is straightforward. We obtain it by considering all 6 terms in the square brackets of (4.21) separately. We find 37 possible singularities, for example:

$$L = x_1^4 + 2x_1^3 x_2 - x_1^2 x_3(2x_2 + x_3) - 2x_1(x_2 - x_3)(x_2 + x_3)^2 + 2x_3(x_2 + x_3)^3 \,, \tag{4.24}$$

and record the rest in `SOFIA_examples.nb` [3].

To our knowledge, the singularity set for the triangle integral was never obtained before. Partial results exist [86] and we have checked that our singularity set contains them.

5 Future directions

In this work, we introduced the package SOFIA, whose main purpose is to automate the computations of singularities of Feynman integrals. It combines a new theoretical understanding of Baikov representations with advanced polynomial reduction techniques and graph-theoretical tools.

This first implementation already pushes beyond the current state of the art in many cases. In particular, the ancillary file [3] includes several new predictions, including those of Fig. 1. These results highlight both the strength and versatility of SOFIA, leading to results that were previously beyond reach.

This progress also opens up a number of future directions in automatizing the determination of the analytic structure of scattering amplitudes and other objects of interest, such as cosmological or energy correlators.

Nature of singularities. One of the most pressing questions is the algorithmic determination of the nature of singularities, i.e., the exponents A and B in the local behavior $f_j^A \log^B f_j$ as $f_j \to 0$. These exponents depend on the spacetime dimension and the geometry of the pinch manifold. In the case where the masses are sufficiently generic, this geometry is simple to describe, and consequently a closed-form formula for the exponents A and B is known [10]; see, e.g., [45,87] for reviews. In "non-generic" situations, which are often the ones of physical interest, more work has to be done in analyzing the pinch manifold. Performing this analysis algorithmically would provide a significant improvement over the capabilities of SOFIA, especially given that it would allow one to distinguish between square-root and logarithmic letters, thus optimizing the size of the proposed symbol alphabet.

Sheet structure. A related question concerns the Riemann sheet structure of Feynman integrals and scattering amplitudes. The simplest version of this question is to determine which singularities lie on the *physical sheet* or in the *physical region*. This question is usually addressed by either determining whether the loop momenta along the pinch manifold are physical, or whether the associated Schwinger parameters are non-negative. We have not attempted to address these questions since the Fubini reduction algorithm circumvents the need to solve the Landau equations directly. Therefore, it does not directly reconstruct the pinch manifold. Similarly, the discontinuity (monodromy) structure of Feynman integrals is more difficult to study in Baikov-like representations, as has been recognized for a long time [39, 88], since the integration contour itself depends on the kinematics. One possible direction is to keep track of the genealogy of singularities in the reduction algorithm, see [33, Sec. 6], which is closely related to the hierarchical structure; see, e.g., [89].

The questions about the sheet structure have their avatar in the symbol properties for polylogarithmic integrals. For example, presence on the physical sheet corresponds to the first-entry condition and the discontinuity structure is encoded in symbol adjacency. See [90, 91] for recent work on the connection to Landau analysis. We believe that the same information is encoded in the commutation relations of the matrices $\boldsymbol{\Omega}_i$ in (1.1) and boundary conditions even for non-polylogarithmic integrals, which means being able to predict it independently would put constraints on the $\boldsymbol{\Omega}_i$'s. See [71, 92] for explicit examples in the context of extended Steinmann relations.

Finding the alphabet. To find the symbol alphabet we wrapped in `SOFIA` the publicly available package `Effortless` [37]. More work is needed in the algorithmic determination of alphabets, which is currently the main bottleneck in `SOFIA` as can be seen in the timings indicated in Fig. 1. We have already mentioned that this procedure would benefit from determining algorithmically which letters are square-root vs. logarithmic, though this information can be provided through the option `AddLetters`, as explained in Sec. 3.3.1. In addition, more conceptual work is needed to understand how general letters can be, including the maximum multiplicity of square roots that can be expected and which singularities are allowed to appear as root arguments. Here, we committed to the ansatz (2.31) with double square roots. More general ansätze can be implemented as needed.

Feynman geometries. We found the optimized loop-by-loop Baikov representation to be efficient at bounding the complexity of the geometries associated with Feynman integrals, e.g., whether they are expressible in terms of periods of elliptic curves or Calabi–Yau manifolds. In addition, `SOFIA` allows one to find the loci of parameters for which they become singular. One may imagine extending `SOFIA` to also analyze these geometries completely automatically.

Spacetime dimension. In the current implementation, `SOFIA` assumes that the spacetime dimension D is large enough so that all external momenta remain independent up to momentum conservation. It also assumes that D is generic, which is what is needed for dimensional regularization. This is why we only provide candidate singularities. Whenever one specializes the dimension D, the actual list of singularities might, and in general will, reduce. Therefore, another improvement on `SOFIA` would be to analyze such cases separately. Imposing a constraint on the external momenta is cumbersome because one has to work with constrained variables, or come up with a different set of kinematical data such as spinors. On the other hand, imposing a bound on the dimensionality of the loop momenta could potentially reduce the number of integration variables in Baikov representations.

Specializing the spacetime dimension may prove useful in other contexts, such as the S-matrix bootstrap [93], where understanding the singularity structure can be essential to make use of tools such as dispersion relations, particularly at higher points [14, 94] and beyond lightest-particle scattering [95, 96]. Another relevant application is the study of two-dimensional integrable models, in which anomalous thresholds manifest as Coleman–Thun poles [97, 98]. For example, an automated tool like `SOFIA` could help extend recent studies on perturbative integrability beyond one-loop [99, 100].

Bottleneck diagrams. We observe that the performance of `SOFIA` could be further optimized for Feynman diagrams that exhibit maximal connectivity—that is, diagrams for which the minimum number of vertex removals required to disconnect the graph into trees is maximized. This pattern becomes evident when monitoring the `SOFIA[]` command on sufficiently complicated diagrams with `IncludeSubtopologies -> True`: the (sub)topologies taking the highest runtime are those that are maximally, or nearly maximally, connected.

One main drawback of the loop-by-loop Baikov approach in these cases is that the resulting N in (2.6) typically remains high, because each $\mathcal{E}_a$ is as large as possible for maximally connected diagrams, regardless of our minimizing strategy for the choice of `LoopEdges`. This poses a computational challenge starting at three-loops, because maximally connected diagrams lead to systems with a large number ($\geqslant 5$) of variables to eliminate, even on the maximal cut. An example (with $N = 5$ on the maximal cut) is the symmetric "envelope diagram" from [67, Fig. 1e], whose singularities were previously calculated with `PLD.jl` in Schwinger parameter space for different masses assignments. Although the symbolic extraction of singularities for this diagram currently requires an impractical runtime in `SOFIA`, we were able to use it to successfully reproduce numerically all of its known singularities.

We emphasize that these diagrams remain equally challenging to solve with either `Solver -> FastFubini` or `Solver -> momentumPLD` in `SOFIA[]`. For the envelope, the difficulty in the latter arises because, on the max cut, one must still eliminate five Baikov variables (the x's) *and* five β's from (2.18) (with one β fixed to 1). In total, this requires eliminating nine variables in Baikov space, compared to only five in Schwinger space.

Fortunately, the stringent criteria of maximal vertex connectivity ensure that, at a given loop order and multiplicity, the removal of only a few vertices results in most diagrams disconnecting into trees. Consequently, the number of maximally connected diagrams remains combinatorially small, especially for processes of phenomenological relevance, e.g., in the Standard Model.

Finally, because the connectivity requirement naturally forces a uniformity in the distribution of vertices, maximally connected diagrams are generally highly symmetric. Fully exploiting this property, as demonstrated in [35, 36] for the physical sheet singularity, could plausibly lead to further improvements of SOFIA's performance on these graphs.

Acknowledgments. We thank Samuel Abreu, Giacomo Brunello, Simon Caron-Huot, Hjalte Frellesvig, Hofie Hannesdottir, Antonela Matijašić, Andrew McLeod, Ian Moult, Erik Panzer, Andrzej Pokraka, Simon Telen, and Simone Zoia for useful comments and discussions. In particular, we thank Antonela Matijašić and Julian Miczajka for sharing pre-release versions of Effortless with us. The work of M.C. and M.G. is supported by the National Science and Engineering Council of Canada (NSERC) and the Canada Research Chair program, reference number CRC-2022-00421. M.G. thanks the Institute for Advanced Study for its hospitality during a crucial stage of this work.

A Derivation of Baikov representations

Setup. We begin by expressing a generic Feynman integral I in a form that emphasizes the relevant structure for the derivation, particularly the integration measure:

$$I = \int \mathrm{d}^{\mathrm{D}}\ell_1\, \mathrm{d}^{\mathrm{D}}\ell_2 \cdots \mathrm{d}^{\mathrm{D}}\ell_L\, \mathcal{I}(\ell_1, \ell_2, \ldots, \ell_L)\,, \tag{A.1}$$

where the integrand $\mathcal{I}$ is a rational function of the loop and external momenta, ℓ_a and p_i respectively, as well as possible other variables such as masses.

Let us use the label q_i to collectively refer to the independent internal and external momenta, that is

$$(q_1, q_2, \ldots, q_M) = (p_1, p_2, \ldots, p_{\mathcal{E}}, \ell_L, \ldots, \ell_2, \ell_1)\,. \tag{A.2}$$

We only include $\mathcal{E}$ external momenta, where

$$\mathcal{E} = \min(\mathrm{D}_0, n-1)\,. \tag{A.3}$$

This is because there are at most $n-1$ independent p_i's due to momentum conservation. In addition, the number of independent p_i's cannot exceed the spacetime dimension D_0 (in dimensional regularization, $\mathrm{D} = \mathrm{D}_0 - 2\varepsilon$). The total number of such independent momenta q_i is therefore $M = L + \mathcal{E}$. They define a natural coordinate system, where we work in terms of the Lorentz-invariant scalar products: $X_{ij} = q_i \cdot q_j$.

Let us consider the integration measure $\mathrm{d}^{\mathrm{D}}\ell_a$ in the order $a = 1, 2, \ldots, L$. We can assume that all the momenta $\ell_1, \ell_2, \ldots, \ell_{a-1}$ have been handled and now we consider ℓ_a. The first key observation is that ℓ_a can be decomposed into two components:

$$\ell_a = \ell_{a\parallel} + \ell_{a\perp} \qquad (1 \leqslant a \leqslant L)\,, \tag{A.4}$$

where $\ell_{a\|}$ lies in the space spanned by $(q_1, \ldots, q_{M-a}) = (p_1, \ldots, p_{\mathcal{E}}, \ell_L, \ldots, \ell_{a+1})$, and $\ell_{a\perp}$ is perpendicular to this space. The integration measure then factorizes as

$$\mathrm{d}^{\mathrm{D}}\ell_a = \mathrm{d}^{M-a}\ell_{a\|}\, \mathrm{d}^{\mathrm{D}-M+a}\ell_{a\perp} \qquad (1 \leqslant a \leqslant L)\,. \tag{A.5}$$

Our next goal is to change the variables in the parallel components $\ell_{a\|}$ of the loop momenta to the scalar products X_{ij}. In order to do so, we first derive the following general result.

Key equation. Consider any vector v in an n-dimensional space and express it in terms of a (not necessarily orthonormal) basis $(q_1, q_2, \ldots, q_n)$:

$$v = \lambda_1\, q_1 + \lambda_2\, q_2 + \cdots + \lambda_n\, q_n\,. \tag{A.6}$$

In components, this reads

$$v^i = \sum_{a=1}^{n} \lambda_a\, q_a^i\,. \tag{A.7}$$

The differential volume element in the n-dimensional subspace is then given by

$$\mathrm{d}^n v = \left|\det \frac{\partial v^i}{\partial \lambda_a}\right| \mathrm{d}^n\lambda\,. \tag{A.8}$$

Since $\frac{\partial v^i}{\partial \lambda_a} = q_a^i$, the corresponding Jacobian matrix J has entries $J_{ia} = q_a^i$. Although the basis $(q_1, ..., q_n)$ is not assumed orthonormal, we can compute

$$(J^T J)_{ab} = q_a \cdot q_b \equiv G(q_1, q_2, \ldots, q_n)_{ab}\,, \tag{A.9}$$

where $G = G(q_1, q_2, \ldots, q_n)$ is called the Gram matrix. Noting that $\det\big(J^T J\big) = (\det J)^2$, we deduce $(\det J)^2 = \det G$ and hence: $\det J = \pm\sqrt{\det G}$. Thus, the change of variables yields

$$\mathrm{d}^n v = \sqrt{\det G}\; \mathrm{d}^n\lambda\,. \tag{A.10}$$

Note that the factor $\sqrt{\det G}$ is nothing but the volume of the parallelepiped spanned by the vectors $q_1, \ldots, q_n$. This factor accounts for the distortion of the unit hypercube in the λ-space when mapped into the momentum space by the basis $\{q_a\}$. For an orthonormal basis, $\det G = 1$, and the volume is unchanged. For a non-orthonormal basis, the hypercube is transformed into a parallelepiped with volume $\sqrt{\det G}$, which corrects the integration measure accordingly. Equipped with (A.10), we proceed with the derivation.

Parallel components. The parallel components are, by definition, linear combinations of the basis vectors $q_1, q_2, \ldots, q_{M-a}$:

$$\ell_{a\|} = \sum_{b=1}^{M-a} \lambda_{ab} q_b \qquad (1 \leqslant a \leqslant L)\,. \tag{A.11}$$

Under the projection onto this q-basis, the volume form transforms according to (A.10):

$$\mathrm{d}^{M-a}\ell_{a\|} = \sqrt{\det G}\prod_{b=1}^{M-a}\mathrm{d}\lambda_{ab}\,, \tag{A.12}$$

where now $G = G(q_1, q_2, \ldots, q_{M-a})$. Next, using (A.11), it is easy to see that the X's and the λ's are linearly related

$$X_{ak} = q_k \cdot \ell_{a\|} = \sum_{b=1}^{M-a}\lambda_{ab} q_b \cdot q_k \iff X = \lambda \cdot G\,, \tag{A.13}$$

where the right-hand side is a matrix equation. Thus,

$$\mathrm{d}^{M-a}\ell_{a\|} = \sqrt{\det G}\left|\det\frac{\partial X}{\partial\lambda}\right|^{-1}\prod_{b=1}^{M-a}\mathrm{d}X_{ab} = \frac{\prod_{b=1}^{M-a}\mathrm{d}X_{ab}}{\sqrt{\det G}} \qquad (1 \leqslant a \leqslant L)\,. \tag{A.14}$$

Perpendicular components. The perpendicular components $\ell_{a\perp}$ are integrated in spherical coordinates, where the volume element in an $(\mathrm{D}-M+a)$-dimensional space is

$$\mathrm{d}^{\mathrm{D}-M+a}\ell_{a\perp} = \ell_{a\perp}^{\mathrm{D}-M+a-1}\mathrm{d}\ell_{a\perp}\mathrm{d}\Omega_{\mathrm{D}-M+a}\,. \tag{A.15}$$

By integrating out the angular part, we obtain

$$\int\limits_{S_{\mathrm{D}-M+a}} \mathrm{d}^{\mathrm{D}-M+a}\ell_{a\perp} = \frac{1}{2}\Omega_{\mathrm{D}-M+a}\,\ell_{a\perp}^{\mathrm{D}-M+a-2}\,\mathrm{d}\ell_{a\perp}^2\,, \tag{A.16}$$

where

$$\Omega_n = \frac{2\pi^{n/2}}{\Gamma(n/2)}\,, \tag{A.17}$$

is the full n-dimensional solid angle. Choosing the $(q_1, \ldots, q_{M-a})$ such that $\ell_{a\perp}^2 = X_{aa}$, we substitute $\mathrm{d}\ell_{a\perp}^2 = \mathrm{d}X_{aa}$, leading to

$$\int\limits_{S_{\mathrm{D}-M+a}} \mathrm{d}^{\mathrm{D}-M+a}\ell_{a\perp} = \left[\frac{\det G(\ell_a, \ell_{a+1}, \ldots, p_{\mathcal{E}})}{\det G(\ell_{a+1}, \ldots, p_{\mathcal{E}})}\right]^{\frac{\mathrm{D}-M+a-2}{2}}\frac{\Omega_{\mathrm{D}-M+a}\,\mathrm{d}X_{aa}}{2}\,, \tag{A.18}$$

for $1 \leqslant a \leqslant L$. This formula has an insightful geometric interpretation. The loop momentum ℓ_a is decomposed into two parts: one lying within the subspace spanned by $(\ell_{a+1}, \ldots, \ell_L, p_1, \ldots, p_{\mathcal{E}})$ (base space) and the other orthogonal to it (ruled hypersurface). The Cartesian product of these two subspaces forms a parallelepiped. Just as the height of a three-dimensional cylinder can be described as the ratio of two volumes, the height/radius of the ruled hypersurface can similarly be expressed in terms of the volume ratio:

$$|\ell_{a\perp}| \equiv \sqrt{X_{aa}} = \sqrt{\frac{\det G(\ell_a, \ell_{a+1}, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})}{\det G(\ell_{a+1}, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})}} \,. \tag{A.19}$$

As in (A.10), the Gram determinant $\det G(q_1, \dots, q_{\mathrm{D}-M+a})$ gives the square of the volume of the parallelepiped spanned by the vectors $(q_i)_{i=1}^{\mathrm{D}-M+a}$. Hence, the numerator in (A.19) represents the volume of the entire parallelepiped formed by $(\ell_a, \ell_{a+1}, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})$ while the denominator represents the volume of the base spanned by $(\ell_{a+1}, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})$.

Finally, since the integration region for the radial coordinate $\mathrm{d}\ell_{a\perp}^2$ in (A.16) is $\ell_{a\perp}^2 > 0$, the integration region for the scalar measure in (A.18) is simply given by

$$\Gamma_a \equiv \left\{ \ell_{a\perp}^2 = \frac{\det G(\ell_a, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})}{\det G(\ell_{a+1}, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})} > 0 \right\} \qquad (1 \leqslant a \leqslant L)\,. \tag{A.20}$$

Final expression. After substituting the parallel (see (A.14)) and perpendicular (see (A.18)) volume elements for each loop momentum, all but the $a = 1$ and $a = L$ Gram determinants cancel, resulting in the Baikov representation of (A.1)

$$\begin{aligned} \int \mathrm{d}^{\mathrm{D}}\ell_1 \mathrm{d}^{\mathrm{D}}\ell_2 \dots \mathrm{d}^{\mathrm{D}}\ell_L\, \mathcal{I}(\ell_1, \dots, \ell_L) &= \frac{\pi^{\frac{L(1+2(\mathrm{D}+\mathcal{E})-L)}{4}}}{\prod_{l=1}^{L} \Gamma\left(\frac{\mathrm{D}-M+l}{2}\right)} \det G(p_1, \dots, p_{\mathcal{E}})^{(-\mathrm{D}+\mathcal{E}+1)/2} \\ &\times \int_\Gamma \prod_{i=1}^{L} \prod_{j=i}^{M} \mathrm{d}X_{ij} \det G(\ell_1, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}})^{(\mathrm{D}-M-1)/2} f(X_{ij})\,, \end{aligned} \tag{A.21}$$

where the multiple factors of $i\pi^{\mathrm{D}/2}$ in (2.1) were absorbed in the integrand f. Moreover, the integration domain Γ is given by imposing L conditions given in (A.20), such that $\Gamma \equiv \Gamma_1 \cap \Gamma_2 \cap \dots \cap \Gamma_L = \{\det G(\ell_1, \dots, \ell_L, p_1, \dots, p_{\mathcal{E}}) > 0\}$.[2]

Note that the number of integration variables X_{ij} in the measure product corresponds to the number of unique pairs (i, j), where i runs from 1 to L and j runs from i to M. This count evaluates to

$$\sum_{i=1}^{L} (M - i + 1) = \frac{L(L+1)}{2} + L\mathcal{E} \equiv N'\,, \tag{A.22}$$

[2] The last equality follows from the observation that, since $\det G(q_1, q_2, \dots, q_n) > 0$, the biggest Gram matrix is positive definite. A key property of positive definite matrices is that every principal submatrix is also positive definite. In particular, Sylvester's criterion tells us that all leading principal minors are positive, and in fact, this extends to any principal submatrix. Therefore, the submatrix corresponding to $q_2, q_3, \dots, q_n$ is positive definite, ensuring that $\det G(q_2, q_3, \dots, q_n) > 0$ and so on. In contrast, in the loop-by-loop case discussed in the main text, the multiple pairs of internal-external Gram matrices typically are not principal submatrices of one another. This occurs when they satisfy (A.21), but with Gram matrices of different sizes (see, e.g., (2.13)). In such cases, the integration contour remains the complete intersection of the corresponding Γ_i regions.

when $M = L + \mathcal{E}$. In the main text, we relabeled the N' integration variables as

$$(x_1, \ldots, x_{N'}) = \text{linear combinations of } \bigcup_{i=1}^{L} \{X_{ij} \mid i \leqslant j \leqslant M\}\,, \tag{A.23}$$

where the first E x's are propagators. Given this notation, let us finally note that, in Sec. 2.1, we denoted

$$\begin{aligned}
c &= \frac{2^{L-N}(-i)^L \pi^{(L-N)/2}}{\prod_{l=1}^{L} \Gamma\left(\frac{\mathrm{D}-M+l}{2}\right)}\,, && \text{(A.24a)}\\
\gamma &= (\mathrm{D} - M - 1)/2\,, && \text{(A.24b)}\\
\tilde{\gamma} &= (\mathcal{E} - \mathrm{D} + 1)/2\,, && \text{(A.24c)}\\
\mathcal{G}(\vec{x};\vec{s}) &= \det G(\ell_1, \ldots, \ell_L, p_1, \ldots, p_{\mathcal{E}})\,, && \text{(A.24d)}\\
\tilde{\mathcal{G}}(\vec{s}) &= \det G(p_1, \ldots, p_{\mathcal{E}})\,. && \text{(A.24e)}
\end{aligned}$$

This completes the derivation of the global Baikov representation in (2.4).

Loop-by-loop variants. In contrast, the loop-by-loop Baikov representations (2.7) are derived from the global Baikov formula (A.21) by first choosing an ordering for the loops in the diagram considered. For each loop, one identifies the set of linearly independent external momenta (these include momenta from loops to be integrated later). Then, the one-loop Baikov representation (i.e., (A.21) for $L = 1$) is applied to that loop momentum. Repeating this process for every remaining loop yields a representation that generally involves fewer integration variables than the global Baikov representation (see, e.g., the sunrise example discussed in Sec. 2.1). Note that this procedure is not unique—the chosen loop ordering and the selection of independent momenta can influence both the final number of integration variables N and the form of the integrand. The package SOFIA automatically looks for a choice of loop shifts and ordering that minimizes N.

B Interface with Julia

As mentioned in Sec. 3, SOFIA can optionally use the Julia package `PLD.jl` to extract the singular locus in (2.19). A working installation of this package can be linked to SOFIA with the option

```
SOFIAoptionJulia = True;
```

Running this command successfully for the first time requires a few steps. First, the user must ensure that `PLD.jl` and all its dependencies are correctly installed and running on their Julia interface. The installation procedure can be found in [66].

Next, the user needs to register the version of Julia in which `PLD.jl` runs (i.e., `v1.8.5`) as an external evaluator in Mathematica by running

```
RegisterExternalEvaluator["Julia", "/path/to/julia"]
```

where "`/path/to/julia`" is replaced by the actual path to the JULIA binary. The user can then verify that MATHEMATICA is communicating with JULIA correctly by running, e.g., `ExternalEvaluate["Julia", "1 + 1"]`. Only once this is done, one should be able to run `Get["SOFIA.m"]` with `SOFIAoptionJulia = True`; if the script is loaded successfully, a "`Checking status`" log should appear while loading without warnings or errors on the MATHEMATICA interface.

C Subtleties with kinematic replacements

As described in Sec. 3, in many cases the command `Symmetry[]` relates two diagrams with different numbers of scales. In such situations, some singularities might be missed if the kinematic replacement is not taken carefully.

To illustrate this point, we consider the two acnode diagrams defined by:

```
(* Acnode diagram #1 *)
edges1 = {{{1,2},m}, {{2,3},m}, {{3,4},0}, {{2,4},0}, {{1,3},m}};
nodes1 = {{1,M1}, {2,M}, {3,M}, {4,M}};
(* Acnode diagram #2 *)
edges2 = {{{1,2},0}, {{2,3},0}, {{3,4},0}, {{2,4},0}, {{1,3},0}};
nodes2 = {{1,0}, {2,M}, {3,M}, {4,M}};
```

We can plot them via:

```
FeynmanPlot/@{{edges1,nodes1},{edges2,nodes2}}
```

which outputs the diagrams in Fig. 5, with diagram (a) corresponding to `{edges1,nodes1}`, and diagram (b) corresponding to `{edges2,nodes2}`.

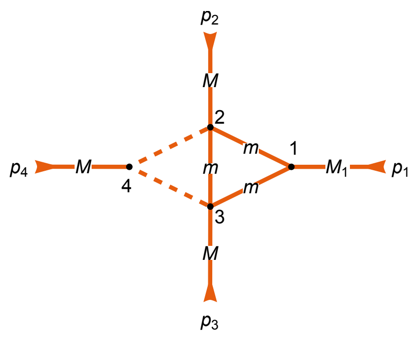
To find the symmetry relation we run:

```
Symmetry[{edges1,nodes1},{edges2,nodes2}]
```

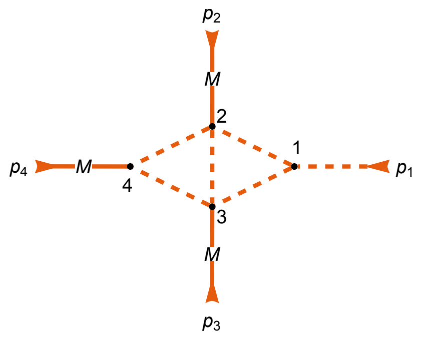
confirming that diagram `{edges1,nodes1}` is related to `{edges2,nodes2}` via the kinematic replacement r_0 =$\{\mathtt{M}_1^2 \to 0, \mathtt{m}^2 \to 0\}$.

Let us now see how this replacement rule applied naively is insufficient to reproduce all singularities. It is enough to perform the *leading* singularity analysis with `IncludeSubtopologies -> False` on both diagrams:

```
diags = {{edges1,nodes1},{edges2,nodes2}};
Do[singularitiesAcn[i] = SOFIA[diags[[i]], SolverBound -> 600
                                  , IncludeSubtopologies -> False]
                                  , {i,2}];
```



(a)



(b)

Figure 5: Diagrams related by the command `Symmetry[]` and whose kinematic replacement rule must be taken with care, as described in the text. These diagrams were generated by the `FeynmanPlot[]` function.

Applying the naive replacement rule r_0 directly to the eleven components of $\mathcal{S}$ obtained in `singularitiesAcn[1]` yields all eight components of $\mathcal{S}$ found for `singularitiesAcn[2]`, *except for one*:

$$\mathtt{MM}^2 - 3\,\mathtt{MM}\,\mathtt{s12} + \mathtt{s12}^2 + \mathtt{s12}\,\mathtt{s23}\,, \tag{C.1}$$

where, we recall, $\mathtt{MM} = \mathtt{M}^2$, $\mathtt{MM1} = \mathtt{M}_1^2$ and $\mathtt{sij} = \mathtt{s}_{ij}$.

However, by replacing r_0 with its "blown-up" version, $r_\delta = \{\mathtt{M}_1^2 \to \delta, \mathtt{m}^2 \to \delta\}$, expanding to leading order in δ, and factoring the result into irreducible components, we recover all the singularities in `singularitiesAcn[2]`, including (C.1). In particular, from the 11th component of $\mathcal{S}$ obtained for `singularitiesAcn[1]`, we have

$$3\,\mathtt{MM}^{12}\,\mathtt{MM1}\,\mathtt{mm} + (1048 \text{ terms}) = 0 \overset{r_\delta}{\longmapsto} \delta^2 \times (\text{C.1}) \times (2 \text{ terms})\,. \tag{C.2}$$

For all the details of this example, see the notebook `SOFIA_examples.nb` in [3]. The δ-procedure outlined above is implemented and executed automatically by SOFIA.

References

[1] S. Weinzierl, *Feynman Integrals: A Comprehensive Treatment for Students and Researchers*, UNITEXT for Physics, Springer (2022), 10.1007/978-3-030-99558-4, [2201.03593].

[2] S. Badger, J. Henn, J.C. Plefka and S. Zoia, *Scattering Amplitudes in Quantum Field Theory*, *Lect. Notes Phys.* **1021** (2024) pp. [2306.05976].

[3] SOFIA GitHub repository, 2025.

[4] A.V. Kotikov, *Differential equations method: New technique for massive Feynman diagrams calculation*, *Phys. Lett. B* **254** (1991) 158.

[5] J.M. Henn, *Multiloop integrals in dimensional regularization made simple*, *Phys. Rev. Lett.* **110** (2013) 251601 [1304.1806].

[6] A.B. Goncharov, M. Spradlin, C. Vergu and A. Volovich, *Classical Polylogarithms for Amplitudes and Wilson Loops*, *Phys. Rev. Lett.* **105** (2010) 151605 [1006.5703].

[7] F. Caola, W. Chen, C. Duhr, X. Liu, B. Mistlberger, F. Petriello et al., *The Path forward to N^3LO*, in *Snowmass 2021*, 3, 2022 [2203.06730].

[8] J.L. Bourjaily et al., *Functions Beyond Multiple Polylogarithms for Precision Collider Physics*, in *Snowmass 2021*, 3, 2022 [2203.07088].

[9] J.D. Bjorken, *Experimental tests of Quantum electrodynamics and spectral representations of Green's functions in perturbation theory*, Ph.D. thesis, Stanford U., 1959.

[10] L. Landau, *On analytic properties of vertex parts in quantum field theory*, *Nucl. Phys.* **13** (1960) 181.

[11] N. Nakanishi, *Ordinary and Anomalous Thresholds in Perturbation Theory*, *Prog. Theor. Phys.* **22** (1959) 128.

[12] R.E. Cutkosky, *Singularities and discontinuities of Feynman amplitudes*, *J. Math. Phys.* **1** (1960) 429.

[13] S. Coleman and R. Norton, *Singularities in the physical region*, *Nuovo Cim.* **38** (1965) 438.

[14] S. Caron-Huot, M. Giroux, H.S. Hannesdottir and S. Mizera, *Crossing beyond scattering amplitudes*, *JHEP* **04** (2024) 060 [2310.12199].

[15] E. Panzer, *Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals*, *Comput. Phys. Commun.* **188** (2015) 148 [1403.3385].

[16] C. Fevola, S. Mizera and S. Telen, *Landau Singularities Revisited: Computational Algebraic Geometry for Feynman Integrals*, *Phys. Rev. Lett.* **132** (2024) 101601 [2311.14669].

[17] C. Fevola, S. Mizera and S. Telen, *Principal Landau determinants*, *Comput. Phys. Commun.* **303** (2024) 109278 [2311.16219].

[18] Principal Landau Determinants MathRepo repository, 2023.

[19] X. Jiang, J. Liu, X. Xu and L.L. Yang, *Symbol letters of Feynman integrals from Gram determinants*, 2401.07632.

[20] S. Caron-Huot, M. Correia and M. Giroux, *Recursive Landau Analysis*, 2406.05241.

[21] P.A. Baikov, *Explicit solutions of the multiloop integral recurrence relations and its application*, Nucl. Instrum. Meth. A **389** (1997) 347 [hep-ph/9611449].

[22] H. Frellesvig, *The Loop-by-Loop Baikov Representation – Strategies and Implementation*, 2412.01804.

[23] H. Frellesvig and C.G. Papadopoulos, *Cuts of Feynman Integrals in Baikov representation*, JHEP **04** (2017) 083 [1701.07356].

[24] C. Bogner, S. Müller-Stach and S. Weinzierl, *The unequal mass sunrise integral expressed through iterated integrals on* $\overline{\mathcal{M}}_{1,3}$, Nucl. Phys. B **954** (2020) 114991 [1907.01251].

[25] H. Frellesvig, F. Gasparotto, M.K. Mandal, P. Mastrolia, L. Mattiazzi and S. Mizera, *Vector Space of Feynman Integrals and Multivariate Intersection Numbers*, Phys. Rev. Lett. **123** (2019) 201602 [1907.02000].

[26] J. Chen, X. Jiang, C. Ma, X. Xu and L.L. Yang, *Baikov representations, intersection theory, and canonical Feynman integrals*, JHEP **07** (2022) 066 [2202.08127].

[27] M. Giroux and A. Pokraka, *Loop-by-loop differential equations for dual (elliptic) Feynman integrals*, JHEP **03** (2023) 155 [2210.09898].

[28] S. Pögel, X. Wang and S. Weinzierl, *Bananas of equal mass: any loop, any order in the dimensional regularisation parameter*, JHEP **04** (2023) 117 [2212.08908].

[29] M. Delto, C. Duhr, L. Tancredi and Y.J. Zhu, *Two-Loop QED Corrections to the Scattering of Four Massive Leptons*, Phys. Rev. Lett. **132** (2024) 231904 [2311.06385].

[30] M. Giroux, A. Pokraka, F. Porkert and Y. Sohnle, *The soaring kite: a tale of two punctured tori*, JHEP **05** (2024) 239 [2401.14307].

[31] C. Duhr, F. Gasparotto, C. Nega, L. Tancredi and S. Weinzierl, *On the electron self-energy to three loops in QED*, JHEP **11** (2024) 020 [2408.05154].

[32] H. Frellesvig, R. Morales and M. Wilhelm, *Classifying post-Minkowskian geometries for gravitational waves via loop-by-loop Baikov*, JHEP **08** (2024) 243 [2405.17255].

[33] F.C.S. Brown, *On the periods of some Feynman integrals*, 0910.0114.

[34] V. Kolkunov, L. Okun and A. Rudik, *The singular points of some feynman diagrams*, JETP **11** (1960) 634.

[35] V. Kolkunov, L. Okun, A. Rudik and V. Sudakov, *Location of the nearest singularities of the* $\pi\pi$*-scattering amplitude*, JETP **12** (1961) 242.

[36] M. Correia, A. Sever and A. Zhiboedov, *Probing multi-particle unitarity with the Landau equations*, *SciPost Phys.* **13** (2022) 062 [2111.12100].

[37] A. Matijašić and J. Miczajka, *`Effortless`: Efficient generation of odd letters with multiple roots as leading singularities*, 25xx.xxxxx.

[38] S. Mandelstam, *Analytic properties of transition amplitudes in perturbation theory*, *Phys. Rev.* **115** (1959) 1741.

[39] V.N. Gribov and I.T. Dyatlov, *Analytic continuation of the three-particle unitarity condition. Simplest diagrams*, *Sov. Phys. JETP* **15** (1962) 140.

[40] M. Correia, A. Sever and A. Zhiboedov, *An analytical toolkit for the S-matrix bootstrap*, *JHEP* **03** (2021) 013 [2006.08221].

[41] R.N. Lee, *Space-time dimensionality D as complex variable: Calculating loop integrals using dimensional recurrence relation and analytical properties with respect to D*, *Nucl. Phys. B* **830** (2010) 474 [0911.0252].

[42] P. Mastrolia and S. Mizera, *Feynman Integrals and Intersection Theory*, *JHEP* **02** (2019) 139 [1810.03818].

[43] H. Frellesvig, R. Morales, S. Pögel, S. Weinzierl and M. Wilhelm, *Calabi-Yau Feynman integrals in gravity: ε-factorized form for apparent singularities*, 2412.12057.

[44] D.A. Cox, J. Little and D. O'Shea, *The dimension of a variety*, in *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, (Cham), pp. 469–538, Springer International Publishing (2015), DOI.

[45] F. Pham, *Singularities of integrals: Homology, hyperfunctions and microlocal analysis*, Universitext, Springer London (2011).

[46] M. Helmer, G. Papathanasiou and F. Tellander, *Landau Singularities from Whitney Stratifications*, 2402.14787.

[47] F. Brown, *The Massless higher-loop two-point function*, *Commun. Math. Phys.* **287** (2009) 925 [0804.1660].

[48] E. Panzer, *Feynman integrals and hyperlogarithms*, Ph.D. thesis, Humboldt U., 2015. 1506.07243. 10.18452/17157.

[49] R. Morales, A. Spiering, M. Wilhelm, Q. Yang and C. Zhang, *Bootstrapping Elliptic Feynman Integrals Using Schubert Analysis*, *Phys. Rev. Lett.* **131** (2023) 041601 [2212.09762].

[50] L. Lippstreu, M. Spradlin, A. Yelleshpur Srikant and A. Volovich, *Landau Singularities of the 7-Point Ziggurat II*, 2305.17069.

[51] S. He, X. Jiang, J. Liu and Q. Yang, *Landau-based Schubert analysis*, 2410.11423.

[52] X. Jiang and L.L. Yang, *Recursive structure of Baikov representations: Generics and application to symbology*, *Phys. Rev. D* **108** (2023) 076004 [2303.11657].

[53] X. Jiang, M. Lian and L.L. Yang, *Recursive structure of Baikov representations: The top-down reduction with intersection theory*, *Phys. Rev. D* **109** (2024) 076020 [2312.03453].

[54] S. Abreu, D. Chicherin, V. Sotnikov and S. Zoia, *Two-loop five-point two-mass planar integrals and double Lagrangian insertions in a Wilson loop*, *JHEP* **10** (2024) 167 [2408.05201].

[55] J. Henn, A. Matijašić, J. Miczajka, T. Peraro, Y. Xu and Y. Zhang, *Complete function space for planar two-loop six-particle scattering amplitudes*, 2501.01847.

[56] C. Duhr, H. Gangl and J.R. Rhodes, *From polygons and symbols to polylogarithmic functions*, *JHEP* **10** (2012) 075 [1110.0458].

[57] R.N. Lee, *Polylogarithmic functions with prescribed branching locus and linear relations between them*, 2407.12503.

[58] H. Müller and S. Weinzierl, *A Feynman integral depending on two elliptic curves*, *JHEP* **07** (2022) 101 [2205.04818].

[59] M. Heller, A. von Manteuffel and R.M. Schabinger, *Multiple polylogarithms with algebraic arguments and the two-loop EW-QCD Drell-Yan master integrals*, *Phys. Rev. D* **102** (2020) 016025 [1907.00491].

[60] A. Matijašić, *Singularity structure of Feynman integrals with applications to six-particle scattering processes*, Ph.D. thesis, Munich U., 2024. 10.5282/edoc.34154.

[61] Effortless GitHub repository, 2025.

[62] F. Febres Cordero, G. Figueiredo, M. Kraus, B. Page and L. Reina, *Two-loop master integrals for leading-color $pp \to t\bar{t}H$ amplitudes with a light-quark loop*, *JHEP* **07** (2024) 084 [2312.08131].

[63] S. Badger, M. Becchetti, N. Giraudo and S. Zoia, *Two-loop integrals for $t\bar{t}$+jet production at hadron colliders in the leading colour approximation*, *JHEP* **07** (2024) 073 [2404.12325].

[64] M. Becchetti, C. Dlapa and S. Zoia, *Canonical differential equations for the elliptic two-loop five-point integral family relevant to $t\bar{t}$+jet production at leading colour*, 2503.03603.

[65] T. Peraro, *FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs*, *JHEP* **07** (2019) 031 [1905.08019].

[66] C. Fevola, S. Mizera and S. Telen, "PLD tutorial." MathRepo, 2023.

[67] S. Mizera and S. Telen, *Landau discriminants*, *JHEP* **08** (2022) 200 [2109.08036].

[68] C. Dlapa, M. Helmer, G. Papathanasiou and F. Tellander, *Symbol alphabets from the Landau singular locus*, *JHEP* **10** (2023) 161 [2304.02629].

[69] C. Duhr, V.A. Smirnov and L. Tancredi, *Analytic results for two-loop planar master integrals for Bhabha scattering*, *JHEP* **09** (2021) 120 [2108.03828].

[70] D. Chicherin, J. Henn and V. Mitev, *Bootstrapping pentagon functions*, *JHEP* **05** (2018) 164 [1712.09610].

[71] S. Abreu, H. Ita, F. Moriello, B. Page, W. Tschernow and M. Zeng, *Two-Loop Integrals for Planar Five-Point One-Mass Processes*, *JHEP* **11** (2020) 117 [2005.04195].

[72] D. Chicherin, V. Sotnikov and S. Zoia, *Pentagon functions for one-mass planar scattering amplitudes*, *JHEP* **01** (2022) 096 [2110.10111].

[73] S. Abreu, P.F. Monni, B. Page and J. Usovitsch, *Planar Six-Point Feynman Integrals for Four-Dimensional Gauge Theories*, 2412.19884.

[74] A. Primo and L. Tancredi, *On the maximal cut of Feynman integrals and the solution of their differential equations*, *Nucl. Phys. B* **916** (2017) 94 [1610.08397].

[75] J. Bosma, M. Sogaard and Y. Zhang, *Maximal Cuts in Arbitrary Dimension*, *JHEP* **08** (2017) 051 [1704.04255].

[76] R. Marzucca, A.J. McLeod, B. Page, S. Pögel and S. Weinzierl, *Genus drop in hyperelliptic Feynman integrals*, *Phys. Rev. D* **109** (2024) L031901 [2307.11497].

[77] Z. Bern, E. Herrmann, R. Roiban, M.S. Ruf, A.V. Smirnov, V.A. Smirnov et al., *Amplitudes, supersymmetric black hole scattering at $\mathcal{O}(G^5)$, and loop integration*, *JHEP* **10** (2024) 023 [2406.01554].

[78] L. Görges, C. Nega, L. Tancredi and F.J. Wagner, *On a procedure to derive ε-factorised differential equations beyond polylogarithms*, *JHEP* **07** (2023) 206 [2305.14090].

[79] H. Frellesvig, R. Morales and M. Wilhelm, *Calabi-Yau Meets Gravity: A Calabi-Yau Threefold at Fifth Post-Minkowskian Order*, *Phys. Rev. Lett.* **132** (2024) 201602 [2312.11371].

[80] A. Brandhuber, G. Chen, G. Travaglini and C. Wen, *Classical gravitational scattering from a gauge-invariant double copy*, *JHEP* **10** (2021) 118 [2108.04216].

[81] T. Hubsch, *Calabi-Yau manifolds: A Bestiary for physicists*, World Scientific, Singapore (1994).

[82] D. Chicherin, I. Moult, E. Sokatchev, K. Yan and Y. Zhu, *Collinear limit of the four-point energy correlator in $\mathcal{N} = 4$ supersymmetric Yang-Mills theory*, *Phys. Rev. D* **110** (2024) L091901 [2401.06463].

[83] D.M. Hofman and J. Maldacena, *Conformal collider physics: Energy and charge correlations*, *JHEP* **05** (2008) 012 [0803.1467].

[84] A.J. Larkoski, I. Moult and B. Nachman, *Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning*, *Phys. Rept.* **841** (2020) 1 [1709.04464].

[85] P. Benincasa and F. Vazão, *The Asymptotic Structure of Cosmological Integrals*, 2402.06558.

[86] P. Benincasa, G. Brunello, M.K. Mandal, P. Mastrolia and F. Vazão, *On one-loop corrections to the Bunch-Davies wavefunction of the universe*, 2408.16386.

[87] H.S. Hannesdottir and S. Mizera, *What is the $i\varepsilon$ for the S-matrix?*, SpringerBriefs in Physics, Springer (1, 2023), 10.1007/978-3-031-18258-7, [2204.02988].

[88] J.N. Islam and Y. Kim, *Analytic property of three-body unitarity integral*, *Physical Review* **138** (1965) B1222.

[89] M. Berghoff and E. Panzer, *Hierarchies in relative Picard-Lefschetz theory*, 2212.06661.

[90] H.S. Hannesdottir, L. Lippstreu, A.J. McLeod and M. Polackova, *Minimal Cuts and Genealogical Constraints on Feynman Integrals*, 2406.05943.

[91] H. Hannesdottir, A. McLeod, M.D. Schwartz and C. Vergu, *The Landau Bootstrap*, 2410.02424.

[92] S. Abreu, H. Ita, B. Page and W. Tschernow, *Two-loop hexa-box integrals for non-planar five-point one-mass processes*, *JHEP* **03** (2022) 182 [2107.14180].

[93] M. Kruczenski, J. Penedones and B.C. van Rees, *Snowmass White Paper: S-matrix Bootstrap*, 2203.02421.

[94] A. Guerrieri, A. Homrich and P. Vieira, *Multiparticle Flux-Tube S-matrix Bootstrap*, *Phys. Rev. Lett.* **134** (2025) 041601 [2404.10812].

[95] A. Homrich, J.a. Penedones, J. Toledo, B.C. van Rees and P. Vieira, *The S-matrix Bootstrap IV: Multiple Amplitudes*, *JHEP* **11** (2019) 076 [1905.06905].

[96] M. Correia, *Nonperturbative anomalous thresholds*, *Phys. Rev. D* **110** (2024) 025012 [2212.06157].

[97] S.R. Coleman and H.J. Thun, *On the Prosaic Origin of the Double Poles in the Sine-Gordon S-matrix*, *Commun. Math. Phys.* **61** (1978) 31.

[98] H.W. Braden, E. Corrigan, P.E. Dorey and R. Sasaki, *Multiple poles and other features of affine Toda field theory*, *Nucl. Phys. B* **356** (1991) 469.

[99] P. Dorey and D. Polvara, *From tree- to loop-simplicity in affine Toda theories I: Landau singularities and their subleading coefficients*, *JHEP* **09** (2022) 220 [2206.09368].

[100] M. Fabri and D. Polvara, *One-loop integrability with shifting masses*, 2411.15080.