Published for SISSA by Springer

Received: *June 24, 2025*
Accepted: *September 1, 2025*
Published: *October 2, 2025*

CALICO: Computing Annihilators from Linear Identities Constraining (differential) Operators

Giuseppe Bertolini[a], **Gaia Fontana**[b] **and Tiziano Peraro**[a]

[a] *Dipartimento di Fisica e Astronomia, Universitá di Bologna e INFN, Sezione di Bologna, via Irnerio 46, I-40126 Bologna, Italy*
[b] *Physik-Institut, Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland*

E-mail: giuseppe.bertolini@studio.unibo.it, gaia.fontana@physik.uzh.ch, tiziano.peraro@unibo.it

Abstract: We elaborate on the method of parametric annihilators for deriving integral relations. Parametric annihilators are differential operators that annihilate multivalued integration kernels appearing in suitable integral representations of special functions. We illustrate this approach in a way that applies to a broad variety of integral representations. We describe a method for computing parametric annihilators based on efficient linear solvers and use them to derive relations between a wide class of special functions related to important problems in high-energy physics. We also formulate a similar method for deriving differential equations satisfied by the independent integrals within an integral family. We show applications to several classes of special functions, including hypergeometric functions, loop integrals in various representations (including Baikov, loop-by-loop Baikov, Lee-Pomeransky and Schwinger representations) and duals of loop integrals. We finally present the public Mathematica package CALICO for computing parametric annihilators and its usage in several examples of high relevance in theoretical particle physics.

Keywords: Automation, Higher-Order Perturbative Calculations, Scattering Amplitudes, Differential and Algebraic Geometry

ArXiv ePrint: 2506.13653

https://doi.org/10.1007/JHEP10(2025)018

JHEP10(2025)018

Contents

1 Introduction **2**

2 Integral families and parametric annihilators **3**
- 2.1 Basic definitions and notation 3
- 2.2 Integral identities via annihilators 5
- 2.3 Connection with syzygy-based approaches 6

3 Differential equations **8**
- 3.1 Differential equations via recurrence relations 8
- 3.2 Differential equations via differential operators 9

4 Computing annihilators via linear constraints **10**
- 4.1 From annihilators to syzygies 10
- 4.2 Solving syzygy equations via linear constraints 11
- 4.3 Filtering out solutions 12
- 4.4 Computing other differential operators 13
- 4.5 Finite-field methods and implementation details 14

5 Hypergeometric functions **15**

6 Loop integrals **18**

7 Baikov representations **20**
- 7.1 Standard Baikov representation 20
- 7.2 Loop-by-loop Baikov representation 20

8 Duals of loop integrals **22**
- 8.1 Reduction of dual integrals via annihilators 24
- 8.2 Computation of connection matrices 25

9 Loop integrals as twisted Mellin moments **28**
- 9.1 Lee-Pomeransky representation 29
- 9.2 Schwinger representation 30
- 9.3 Examples 30

10 The CALICO Mathematica package **34**

11 Conclusions and outlook **42**

A Representations of loop integrals **43**

JHEP10(2025)018

1 Introduction

Loop integrals are crucial ingredients for our theoretical understanding of quantum field theory. They contribute to precise phenomenological predictions about the fundamental interactions of nature, which are essential to match the uncertainty of modern experiments. They are also used for new approaches to classical theories — including gravity — which are increasingly important due to recent improvements in experimental observations. Due to the rich mathematical structure they exhibit, they are also an interesting subject for mathematical studies, which in turn we may exploit to improve our capability of making higher-order theoretical predictions in perturbation theory.

The set of loop integrals which contribute to a theoretical prediction are generally not independent. They satisfy linear relations, such as symmetry relations and, most importantly, *Integration By Parts* (*IBP*) identities [1, 2], which can be found via purely algebraic means. By finding and solving such relations, this set of integrals is reduced to a linear combination of an independent subset of them, called *master integrals* (*MIs*). The reduction to master integrals is an essential ingredient of most higher-loop theoretical predictions. Besides drastically reducing the number of integrals which need to be computed, it casts amplitudes in a form where many of its properties and symmetries are manifest, and it enables the method of differential equations [3, 4] for computing the master integrals themselves.

IBPs are traditionally found using the momentum-space representation of loop integrals. However, many other representations of these integrals are known, including the Feynman, Schwinger, Baikov [5] (and its loop-by-loop variant [6]) and Lee-Pomeransky [7] representations. Different representations offer different tradeoffs, hence their usefulness is highly dependent on the problem at hand.

The Baikov representation, for instance, is used — among other things — in modern approaches to IBP reduction, combined with syzygy techniques, for finding simpler identities among loop integrals which do not contain higher-powers of denominators [8]. The Lee-Pomeransky representation has also been used in the context of integral reduction, most notably using the approach of *parametric annihilators* [5, 9, 10]. The latter proved to be extremely efficient in certain contexts, e.g. for reducing high powers of denominators (see e.g. [11]) which often appear in integrals having desirable properties, such as quasi-finiteness [12, 13] or a pure functional form [14]. More recently, integral reduction in the Feynman representation has also be considered [15].

The method of parametric annihilator has been extensively described in [10] for *twisted Mellin integrals* and in particular for the Lee-Pomeransky parametrization of loop integrals. It consists in finding *differential operators* in the integration variables which annihilate the multivalued integration kernel appearing in such integral representation. These operators are thus used to find linear identities obeyed by loop integrals.

The main goals of this work are the following. We review and elaborate on the method of *parametric annihilators* for finding integral identities, with a focus on parametric representations of loop integrals. We formulate the method in more general terms, extending its range of applications to different integral representations — including Schwinger and loop-by-loop Baikov which had not been used in this context. In particular, we find the Schwinger representation yields very efficient systems of identities in some cases. We also illustrate a similar

JHEP10(2025)018

technique for finding differential equations satisfied by the master integrals. We provide a description of these methods in a language that is accessible to the scientific community of theoretical particle physics and phenomenology, in particular the one focusing on higher-order perturbative corrections, avoiding mathematical language that is not common knowledge in these fields. We describe and provide an implementation of annihilators and other important differential operators based on modern linear solvers which rely on cutting edge finite-field techniques [16, 17]. This implementation is distributed as a MATHEMATICA package named CALICO, which we believe can be a useful tool for studying a broad variety of integrals, including loop integrals in various parametric representations. Finally, we introduce a new way of characterizing and finding relations between duals of loop integrals [18–20] — which play a crucial role in the study of loops within the framework of intersection theory [21, 22] — compatible with the approach discussed in this paper.

The paper is structured as follows. In section 2, we set some notation and describe the method of parametric annihilators in a way that covers a wide class of functions having a suitable integral representation. We also clarify the connection between this approach and others based on syzygy equations. In section 3, we describe two methods for deriving differential equations, the second of which also relies on finding suitable differential operators in the integration variables. In section 4, we illustrate a method for computing annihilators by solving linear constraints, which is suitable for applying modern linear solvers and finite-field methods. In sections 6 to 9, we describe several applications of this approach, including different representations of loop integrals — namely the standard and loop-by-loop Baikov, Lee-Pomeransky and Schwinger representations — as well as duals of loop integrals. In section 10, we describe the basic usage of the CALICO package. Finally, in section 11 we draw our conclusions and in appendix A we provide more details on the definitions of the polynomials which appear in the integral representations we use.

2 Integral families and parametric annihilators

In this section we set up our notation and review the main concepts we use in this paper, namely parametric annihilators and their usage in finding linear identities within integral families.

2.1 Basic definitions and notation

We use a multi-index notation. Given an n-dimensional multi-index $\alpha = (\alpha_1, \ldots, \alpha_n)$ and a list of n variables $\mathbf{z} = (z_1, \ldots, z_n)$, we define the monomial

$$\mathbf{z}^\alpha = \prod_{j=1}^{n} z_j^{\alpha_j} \tag{2.1}$$

and its total degree

$$|\alpha| = \sum_{j=1}^{n} \alpha_j. \tag{2.2}$$

When this does not cause ambiguity, we define the derivative operator as

$$\partial_j \equiv \frac{\partial}{\partial z_j}. \tag{2.3}$$

JHEP10(2025)018

We consider an *integral family*, whose elements are linear combinations of integrals of the type

$$I_\alpha = \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z})\, u(\mathbf{z}), \tag{2.4}$$

where $u(\mathbf{z})$ is a *multivalued function* common to all integrals, also known as *twist*. In most of our applications, u takes the form

$$u(\mathbf{z}) = \prod_j B_j(\mathbf{z})^{\gamma_j}, \tag{2.5}$$

where $B_j(\mathbf{z})$ are polynomials and γ_j are *generic exponents*, which typically depend on regulators. Other forms are however possible, such as

$$u(\mathbf{z}) = \exp F(\mathbf{z}) \prod_j B_j(\mathbf{z})^{\gamma_j}, \tag{2.6}$$

where $F(z)$ is a polynomial or a rational function.

The functions $\varphi_\alpha(\mathbf{z})$ in eq. (2.4) are identified by the multi-index α and contain n *rational factors* raised to the *integer exponents* $\alpha = (\alpha_1, \ldots, \alpha_n)$. In most cases φ_α is a rational function. We assume the family of functions φ_α to be closed under differentiation and multiplication by monomials. In other words, both $\partial_j\varphi_\alpha(\mathbf{z})$ and $\mathbf{z}^\beta\varphi_\alpha(\mathbf{z})$ are linear combinations of functions of the same form as $\varphi_\alpha(\mathbf{z})$. Explicit examples are given in later sections.

The integration domain never varies in our applications and is thus left implicit in this section. We also understand that u and φ_α may also depend on additional free parameters (other than regulators). In the context of loop integrals, for instance, these are generally kinematic variables. If x is one of these free parameters, we assume that $\partial_x\varphi_\alpha(\mathbf{z})$ either vanishes or belongs to the same space of functions φ_α. The integrals I_α are thus *special functions* of the regulators and these parameters.

In this paper, we are interested in finding and solving *linear relations* satisfied by integrals having the general form in eq. (2.4). An important ingredient, in this context, are *Integration by parts identities* (*IBPs*), which read

$$\int \mathrm{d}^n\mathbf{z}\, \big(\partial_j\varphi_\alpha(\mathbf{z})\big)\, u(\mathbf{z}) + \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z})\, \big(\partial_j \log u(\mathbf{z})\big)\, u(\mathbf{z}) = \int \mathrm{d}^n\mathbf{z}\, \partial_j\Big(\varphi_\alpha(\mathbf{z})\, u(\mathbf{z})\Big). \tag{2.7}$$

The right-hand side (r.h.s.) is a boundary term. Even in cases where the boundary term does not vanish, it can be rewritten as an integral in fewer integration variables and thus belonging to a simpler integral family. In all our applications, the boundary term can be set to zero, hence it will be neglected in the remainder of this paper. By carrying out derivatives explicitly on the left-hand side (l.h.s.) of the previous equation, we obtain non-trivial integral identities. However, in most applications, the term $\partial_j \log u(\mathbf{z})$ contains denominator factors which cannot be absorbed in the family of functions $\varphi_\alpha(\mathbf{z})$, hence the l.h.s. is generally not a relation within the original integral family. When $u(\mathbf{z})$ has the form in (2.5), as an example, the term $\partial_j \log u(\mathbf{z})$ generates rational functions with $B_j(\mathbf{z})$ as denominator factor or, equivalently, integrals with shifted exponents $\gamma_j \to \gamma_j - 1$, also known as *shifted integrals* (or dimensionally shifted integrals in the context of loop integrals). In principle, we can

JHEP10(2025)018

deal with these by writing down *recurrence relations* [23–25], which relate integrals with shifted exponents. It is generally more convenient, however, to write down identities for integrals within the same family and without shifted exponents. In the following, we review how to achieve this by using the method of parametric annihilators. We will expand on recurrence relations again in section 3.

2.2 Integral identities via annihilators

In this section we review the method of *parametric annihilators* (henceforth simply annihilators, for brevity). These are operators that annihilate integration kernels of special functions, including various parametric representations of loop integrals, which can be used to derive linear relations between them (see e.g. [5, 9, 10]).

An *annihilator* of order o of $u(\mathbf{z})$ is an operator of the form

$$\hat{A} = c_0(\mathbf{z}) + \sum_j c_j(\mathbf{z})\,\partial_j + \sum_{j_1\leq j_2} c_{j_1 j_2}(\mathbf{z})\,\partial_{j_1}\partial_{j_2} + \cdots + \sum_{j_1\leq\cdots\leq j_o} c_{j_1\cdots j_o}(\mathbf{z})\,\partial_{j_1}\cdots\partial_{j_o} \tag{2.8}$$

with *polynomials* $c_{j_1 j_2\ldots}$ in $\mathbf{z}$, such that

$$\hat{A}\,u(\mathbf{z}) = 0. \tag{2.9}$$

For any annihilator $\hat{A}$, we have infinitely many integral identities

$$\int \mathrm{d}^n\mathbf{z}\,\varphi_\alpha(\mathbf{z})\,\hat{A}\,u(\mathbf{z}) = 0 \tag{2.10}$$

for any multi-index α. We thus obtain non-trivial identities within the integral family in eq. (2.4) by integrating by parts derivatives. More explicitly, neglecting boundary terms, we obtain

$$\int u\,(\varphi_\alpha c_0) - \sum_j \int u\,(\partial_j c_j \varphi_\alpha) + \cdots + (-1)^o \sum_{j_1\leq\cdots\leq j_o} \int u\,\partial_{j_1}\cdots\partial_{j_o}(c_{j_1\cdots j_o}\varphi_\alpha) = 0, \tag{2.11}$$

where we omitted the integration measure and the $\mathbf{z}$ dependence for brevity. Under our assumptions, all integrals in eq. (2.11) belong to the integral family in (2.4).

In general, for each annihilator $\hat{A}$, we use eq. (2.11) to write a *template identity*, which is an identity valid for generic, symbolic exponents α. At a later stage, identities valid for specific integrals are obtained from the template identities, by replacing the symbolic α with a list of integer exponents. A multi-index of integer exponents α, or the function φ_α and integral I_α it identifies, is called *seed* or *seed integral* in this context. A common strategy consists in applying each template identity to a large number of seed integrals and thus obtain a linear system of equations.[1] Additional relations, such as *symmetry* relations, may also exist. These are changes of integration variables which map integrals into linear combinations of integrals of the same family. In applications where more than one integral family is considered, symmetry relations relating integrals of different families may also exist.

[1] In the context of loop integrals, this is known as the Laporta algorithm [26].

JHEP10(2025)018

By solving this system of linear relations, one can reduce all integrals within a family (or several families) to a linear combination of a smaller subset of independent integrals, called *master integrals* (*MIs*)

$$I_\alpha = \sum_{\beta \in \text{MIs}} c_{\alpha\beta} I_\beta, \tag{2.12}$$

where, with a slight abuse of notation in the summation, we confuse the (master) integrals with the multi-index that identifies them. The reduction to MIs is a crucial step in many theoretical and phenomenological studies in particle physics and other fields, including classical gravity and cosmology.

It should be noted that, whenever a system is solved, an *order* must be introduced between the unknowns, which determines which unknowns are substituted with higher priority. In this context, such an order is expressed in terms of the so-called *weight*, which can be regarded as an estimate of the complexity of the integrals [26]. While solving the system, more complex integrals (i.e. with higher weight) are eliminated in favour of simpler integrals (i.e. with lower weight). The list of MIs also depends on this weight, although their number does not. For a choice of weight used in several modern calculations see e.g. [27].

We now review a few basic properties of annihilators. Linear combinations of annihilators are obviously also annihilators. Moreover, if $\hat{A}$ is an annihilator of $u(\mathbf{z})$ then

$$\mathbf{z}^\beta \hat{A} \qquad \text{and} \qquad \partial_j \hat{A} \tag{2.13}$$

are also annihilators. Using the language of algebraic geometry, this means than the set of annihilators of $u(\mathbf{z})$ is a D-module. In the second expression, we understand that the derivative ∂_j acts on everything on its right, namely both the polynomial coefficients inside $\hat{A}$ and the function $\hat{A}$ is applied to. However, we can easily see that these two annihilators are not needed, since

$$\begin{aligned} \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\mathbf{z}^\beta \hat{A}\, u(\mathbf{z})\right) &= \int \mathrm{d}^n\mathbf{z} \left(\mathbf{z}^\beta \varphi_\alpha(\mathbf{z})\right) \hat{A}\, u(\mathbf{z}) \\ \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\partial_j \hat{A}\, u(\mathbf{z})\right) &= -\int \mathrm{d}^n\mathbf{z} \left(\partial_j \varphi_\alpha(\mathbf{z})\right) \hat{A}\, u(\mathbf{z}). \end{aligned} \tag{2.14}$$

Since, under our assumptions, the space of functions $\varphi_\alpha(\mathbf{z})$ is closed under monomial multiplication and differentiation, all the identities we can obtain from $\mathbf{z}^\beta \hat{A}$ and $\partial_j \hat{A}$ with seed $\varphi_\alpha(\mathbf{z})$ can also be obtained as identities generated by $\hat{A}$ applied to a suitable set of seeds which span $\mathbf{z}^\beta \varphi_\alpha(\mathbf{z})$ and $\partial_j \varphi_\alpha(\mathbf{z})$. Hence, we are generally interested in finding a set of *generators* that is minimal, namely a set of annihilators $\hat{A}$ that are independent modulo linear combinations of operators obtained as in eq. (2.13).

2.3 Connection with syzygy-based approaches

Before we further elaborate on the method of annihilators and related techniques, we make an observation about the relation between this and another method that is commonly used in the literature to avoid the appearance of shifted integrals. The latter relies on syzygy equations.

For simplicity, we focus on a twist of the form

$$u(\mathbf{z}) = B(\mathbf{z})^\gamma, \tag{2.15}$$

JHEP10(2025)018

i.e. defined by just one polynomial $B(\mathbf{z})$. The most general IBP relation one can write for an integral of the corresponding family reads

$$\begin{aligned} 0 &= \sum_{j=1}^{n} \int \mathrm{d}^n\mathbf{z}\, \partial_j \Big(a_j(\mathbf{z}) \varphi_\alpha(\mathbf{z}) B(\mathbf{z})^\gamma \Big) \\ &= \int \mathrm{d}^n\mathbf{z} \left[B(\mathbf{z})^\gamma \sum_{j=1}^{n} \partial_j \Big(\varphi_\alpha(\mathbf{z}) a_j(\mathbf{z}) \Big) + B(\mathbf{z})^\gamma \varphi_\alpha(\mathbf{z}) \Big(\gamma \frac{1}{B(\mathbf{z})} \sum_{j=1}^{n} a_j(\mathbf{z}) \partial_j B(\mathbf{z}) \Big) \right], \end{aligned} \tag{2.16}$$

where $a_j(\mathbf{z})$ are polynomials in the integration variables. The first term in the second equality, under our assumptions, is a linear combination of integrals within our family, since the space of functions φ_α is closed under differentiation and monomial multiplication. The second term, instead, is proportional to a shifted integral, with $\gamma \to \gamma - 1$, due to the denominator $1/B(\mathbf{z})$. The denominator however simplifies if a special choice of $a_j(\mathbf{z})$ is made, namely such that

$$\sum_{j=1}^{n} a_j(\mathbf{z}) \partial_j B(\mathbf{z}) = b(\mathbf{z})\, B(\mathbf{z}) \tag{2.17}$$

for some polynomial $b(\mathbf{z})$. This is a *syzygy equation* for the unknown polynomials $a_j(\mathbf{z})$ and $b(\mathbf{z})$, which can be solved with methods of algebraic geometry (we will further elaborate on this in section 4). After plugging a solution of eq. (2.17) into eq. (2.16), we obtain and identity that is free of shifted integrals

$$\int \mathrm{d}^n\mathbf{z}\, B(\mathbf{z})^\gamma \left[\sum_{j=1}^{n} \partial_j \Big(\varphi_\alpha(\mathbf{z}) a_j(\mathbf{z}) \Big) + \gamma \varphi_\alpha(\mathbf{z}) b(\mathbf{z}) \right] = 0. \tag{2.18}$$

This method has been used with the Baikov representation [8] and the Lee-Pomeranski representation [28] of loop integrals, both of which have the form in eq. (2.15), although one can easily generalize it to a more complex twist.

The strategy we just reviewed is equivalent to the method of parametric annihilators when the latter is restricted to first-order operators [29]. First, consider a first-order operator ($o = 1$) of the form in eq. (2.8). Multiplying eq. (2.9) by $B(\mathbf{z})^{1-\gamma}$, for our choice of twist, we obtain that the condition the annihilator must satisfy is equivalent to the following relation for its polynomial coefficients $c_j(\mathbf{z})$

$$c_0(\mathbf{z})\, B(\mathbf{z}) + \gamma \sum_{j=1}^{n} c_j(\mathbf{z})\, \partial_j B(\mathbf{z}) = 0. \tag{2.19}$$

By comparing this relation with eq. (2.17) we immediately see that there is a one-to-one correspondence between first-order annihilators and syzygy solutions of (2.17), via the identifications

$$b(\mathbf{z}) = -c_0(\mathbf{z}), \qquad a_j(\mathbf{z}) = \gamma\, c_j(\mathbf{z}). \tag{2.20}$$

Second, by inserting this relation into eq. (2.18), we obtain the same first-order integral identity we get from eq. (2.11) by setting $o = 1$. This shows that the syzygy method yields the same integral identities as first-order annihilators.

JHEP10(2025)018

Additional syzygy equations can be used to further constrain the form of the identities that are generated, e.g. such that certain kinds of integrals do not appear (see e.g. [8, 30–32]). These additional constraints can be applied to both the solutions of eq. (2.17) and the ones of eq. (2.9) regardless of the order o of the annihilator.

Hence, the method of parametric annihilators *generalizes* the one based on syzygy equations. In cases where first-order annihilators are sufficient for the solution of a problem, the two strategies are equivalent. This seems to be the case when the standard Baikov representation is used, as we are not aware of counterexamples. When second-order or higher-order annihilators are needed (examples in the Lee-Pomeransky representation have already been reported [11] and more, for various representations, will be given in this work), the method of annihilators superseeds the one based on syzygy equations. The two methods may become equivalent, if the syzygy method is generalized to include higher-order derivatives in eq. (2.16), but this goes beyond the way it is commonly formulated.

3 Differential equations

As we already mentioned, integrals in the form of eq. (2.4) typically also depend on additional free parameters. A very effective approach for both studying the analytic structure of these integrals and evaluating them either analytically or numerically is the method of *differential equations* (*DEs*) [3, 4].

Let x be a free parameter the integrals depend on. By reducing the derivative of MIs with respect to x to MIs, we can write a system of differential equations satisfied by the MIs themselves

$$\partial_x I_\alpha = \sum_{\beta \in \text{MIs}} M_{\alpha\beta} I_\beta, \qquad \text{for } \alpha \in \text{MIs}. \tag{3.1}$$

In the following, we describe two strategies for deriving DEs. The first is based on *recurrence relations*, while the second is based on building suitable *differential operators* in the integration variables.

3.1 Differential equations via recurrence relations

The most straightforward way of computing DEs for integrals of the form in eq. (2.4) is by differentiating directly under the integral sign, which leads to the need of recurrence relations. For simplicity, we consider a twist u of the form of eq. (2.15), but generalizations to other forms of u are straightforward. By differentiating at the integrand level we obtain

$$\partial_x I_\alpha = \int \mathrm{d}^n\mathbf{z} \left(\partial_x \varphi_\alpha(\mathbf{z})\right) B(\mathbf{z})^\gamma + \gamma \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\partial_x B(\mathbf{z})\right) B(\mathbf{z})^{\gamma-1}. \tag{3.2}$$

Under the assumptions outlined in section 2.1, the first term of the r.h.s. is an integral belonging to the original integral family. The second term, however, belongs to an integral family of *shifted* integrals, i.e. a family obtained from the original one by shifting exponents, namely $\gamma \to \gamma - 1$. In order to reduce the r.h.s. to a linear combination of integrals in the original family, with unshifted exponents, we need to derive a recurrence relation that shifts back the exponents from $\gamma - 1$ to γ. While in some cases there are closed formulas

JHEP10(2025)018

for such recurrence relation [23–25], in the most general case a direct relation of this form is not available. Deriving a recurrence relation for the opposite shift, namely from $\gamma+1$ to γ, is however straightforward. Indeed, it simply consists in multiplying the function φ_α by the polynomial B:

$$I_\alpha^{(\gamma+1)} = \int \mathrm{d}^n\mathbf{z}\left(B(\mathbf{z})\varphi_\alpha(\mathbf{z})\right) B(\mathbf{z})^\gamma, \tag{3.3}$$

where $I_\alpha^{(\gamma')} \equiv I_\alpha|_{\gamma\to\gamma'}$. By writing such a relation for all MIs and reducing the r.h.s. of it back to MIs we obtain a matrix $R^{(+)}$ which realizes the recurrence

$$I_\alpha^{(\gamma+1)} = \sum_{\beta\in\mathrm{MIs}} R^{(+)}_{\alpha\beta}(\gamma)\, I_\beta, \qquad \text{for } \alpha\in\mathrm{MIs}. \tag{3.4}$$

By inverting $R^{(+)}$ and shifting $\gamma\to\gamma-1$ we obtain the recurrence relation we seek

$$I_\alpha^{(\gamma-1)} = \sum_{\beta\in\mathrm{MIs}} R^{(-)}_{\alpha\beta}(\gamma)\, I_\beta, \qquad \text{for } \alpha\in\mathrm{MIs}, \tag{3.5}$$

with

$$R^{(-)}_{\alpha\beta}(\gamma) = \left[R^{(+)}(\gamma-1)\right]^{-1}_{\alpha\beta}. \tag{3.6}$$

We thus reduce the second term in eq. (3.2) to shifted MIs with exponent $\gamma-1$ and then use recurrence relation (3.5) to rewrite the shifted MIs in terms of unshifted MIs. We thus obtain a DE for the (unshifted) MIs of the original family, as in eq. (3.1).

While the strategy we just outlined is viable, the appearance of shifted integrals in intermediate stages and the need of finding and using recurrence relations make it quite involved. In the following, we present a method where shifted integrals do not appear at any stage of the calculation.

3.2 Differential equations via differential operators

A more elegant method of deriving differential equations, that better fits the general approach we follow in this paper, consists in deriving an operator $\hat{O}_x$ that realizes differentiation with respect to x. In other words, we find an operator $\hat{O}_x$ of order o

$$\hat{O}_x = c_0^{(x)}(\mathbf{z}) + \sum_j c_j^{(x)}(\mathbf{z})\,\partial_j + \sum_{j_1\le j_2} c_{j_1j_2}^{(x)}(\mathbf{z})\,\partial_{j_1}\partial_{j_2} + \cdots + \sum_{j_1\le\cdots\le j_o} c_{j_1\cdots j_o}^{(x)}(\mathbf{z})\,\partial_{j_1}\cdots\partial_{j_o} \tag{3.7}$$

with polynomials $c^{(x)}_{j_1j_2\ldots}$, such that

$$\hat{O}_x\, u(\mathbf{z}) = \partial_x\, u(\mathbf{z}). \tag{3.8}$$

Once such an operator is found, since it contains only derivatives with respect to the integration variables, differentiation with respect to x can be obtained by integrating by parts all derivatives in $\hat{O}_x$, namely

$$\begin{aligned}\partial_x I_\alpha &= \int (\hat{O}_x u)\,\varphi_\alpha + \int u\,(\partial_x\varphi_\alpha)\\ &= \int u\,(\varphi_\alpha c_0^{(x)}) - \sum_j \int u\,(\partial_j c_j^{(x)}\varphi_\alpha) + \cdots + (-1)^o \sum_{j_1\le\cdots\le j_o} \int u\,\partial_{j_1}\cdots\partial_{j_o}(c^{(x)}_{j_1\cdots j_o}\varphi_\alpha)\\ &\quad + \int u\,(\partial_x\varphi_\alpha).\end{aligned} \tag{3.9}$$

JHEP10(2025)018

Under our assumptions (see section 2.1), all integrals on the r.h.s. of the last equality belong to our integral family and can thus be reduced to MIs. This reduction yields the DEs in eq. (3.1). This is our preferred method for computing DEs in the context of this paper.

It is worth observing that the operator $\hat{O}_x$ satisfying eq. (3.8) is not unique, but the difference between two operators satisfying eq. (3.8) is a parametric annihilator. Hence, we can say that $\hat{O}_x$ is unique modulo the addition of parametric annihilators of u.

4 Computing annihilators via linear constraints

In this section we illustrate a method for computing parametric annihilators up to a certain order and polynomial degree. The method has been implemented in the CALICO package. It consists in two steps. The first translates the annihilator equation in eq. (2.9) into a syzygy equation for the polynomials $c_{j_1\cdots j_k}(\mathbf{z})$ in eq. (2.8). The second step finds syzygy solutions by constraining a polynomial ansatz.

4.1 From annihilators to syzygies

A syzygy equation is an equation of the form

$$\mathbf{f}(\mathbf{z}) \cdot \mathbf{g}(\mathbf{z}) = 0 \tag{4.1}$$

where

$$\mathbf{f}(\mathbf{z}) = \{f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})\} \tag{4.2}$$

is a list of *known polynomials* and

$$\mathbf{g}(\mathbf{z}) = \{g_1(\mathbf{z}), \ldots, g_n(\mathbf{z})\} \tag{4.3}$$

is a list of *unknown polynomials* to be found. If $\mathbf{g}^{(k)}(\mathbf{z})$ are a set of solutions of eq. (4.1), then

$$\mathbf{g}(\mathbf{z}) = \sum_k p_k(\mathbf{z})\, \mathbf{g}^{(k)}(\mathbf{z}), \tag{4.4}$$

where $p_k(\mathbf{z})$ are arbitrary polynomials, is also a solution. In the language of algebraic geometry, we say that syzygy solutions form a *module*. We generally wish to find a set of *generators* $\{\mathbf{g}^{(k)}\mathbf{z})\}_k$ that is minimal, i.e. that are independent of each other with respect to combinations of the form in eq. (4.4). Finding a complete set of generators is however often unnecessary. We can, indeed, generally limit ourselves to a subset of generators which is sufficient for the solution of a certain problem, for instance all independent generators up to a maximum degree.

One can easily use eq. (2.9) to identify polynomials $c_{j_1,\ldots,j_k}(\mathbf{z})$ of an annihilator of order o (see eq. (2.8)) as the unknown polynomials $\mathbf{g}(\mathbf{z})$ of a suitable syzygy equation. We first rewrite eq. (2.9) as

$$\frac{1}{u(\mathbf{z})}\, \hat{A}\, u(\mathbf{z}) = 0. \tag{4.5}$$

After inserting eq. (2.8), with symbolic unknown polynomials $c_{j_1,\ldots,j_k}(\mathbf{z})$, into the previous equation and simplifying out non-rational terms, we collect the resulting rational expression

JHEP10(2025)018

on the l.h.s. under a common denominator. By imposing the vanishing of the numerator of such expression we obtain a syzygy equation for the polynomials $c_{j_1,\ldots,j_k}(\mathbf{z})$.

The strategy can be clarified by means of an explicit example. Consider *first-order annihilators* (i.e. with $o = 1$) of a twist having the form in eq. (2.5), which is the most common in our applications. Inserting eq. (2.8) into eq. (4.5) we obtain

$$c_0(\mathbf{z}) + \sum_{j=1}^{n} c_j(\mathbf{z}) \sum_k \gamma_k \frac{\partial_j B_k(\mathbf{z})}{B_k(\mathbf{z})} = 0, \tag{4.6}$$

which is a rational equation. After collecting under common denominator and setting the numerator to zero we obtain

$$c_0(\mathbf{z}) \prod_k B_k(\mathbf{z}) + \sum_{j=1}^{n} c_j(\mathbf{z}) \sum_k \gamma_k \left(\partial_j B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}) = 0, \tag{4.7}$$

which has now the form of eq. (4.1) by identifying

$$\begin{aligned} \mathbf{f}(\mathbf{z}) &= \left\{ \prod_k B_k(\mathbf{z}), \sum_k \gamma_k \left(\partial_{z_1} B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}), \ldots, \sum_k \gamma_k \left(\partial_{z_n} B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}) \right\} \\ \mathbf{g}(\mathbf{z}) &= \{c_0(\mathbf{z}), c_1(\mathbf{z}), \ldots, c_n(\mathbf{z})\}. \end{aligned} \tag{4.8}$$

The same strategy can be applied to higher-order annihilators and annihilators for twists having a different form, e.g. having an exponential factor as well.

4.2 Solving syzygy equations via linear constraints

Syzygy equations are often solved via techniques of computational algebraic geometry, such as Gröbner basis techniques. However, they can also easily be reduced into a problem of linear algebra by turning them into linear constraints for the coefficients of a polynomial ansatz. We prefer the latter strategy, as it allows us to exploit efficient sparse linear solvers based on finite-field techniques and functional reconstruction routines. Moreover, as already mentioned, we are generally interested to find solutions up to a maximal degree, which in turn can be easily achieved by writing an ansatz having this property.

The strategy of finding syzygy solutions via linear constraints is well known and has already been used in the context of loop integrals (see e.g. [33]). Here, we give a brief review of it. Given a syzygy equation of the form in eq. (4.1), we make an ansatz for its solution[2]

$$\mathbf{g}(\mathbf{z}) = \sum_j \sum_\alpha c_{j\alpha}\, \mathbf{z}^\alpha\, \hat{e}_j \tag{4.9}$$

with $\hat{e}_j$ being the unit vector in the j-th direction. The second sum is over all the multidimensional exponents α such that $|\alpha| \leq d$ for some maximum degree d.

[2]In our ansatz the coefficients $c_{j\alpha}$ are rational functions in the external parameters x and regulators. In principle, without loss of generality, one could also make a polynomial ansatz with respect to these additional parameters, in terms of rational unknown numerical coefficients (see e.g. [34]). We however found that our strategy is generally more efficient when combined with finite-field techniques and functional reconstruction algorithms, besides being slightly easier to generalize to polynomial decomposition problems (see eq. (4.12)).

JHEP10(2025)018

After inserting this ansatz into eq. (4.1) we impose that the coefficient of each monomial in $\mathbf{z}$ on the l.h.s. of the equation vanishes. This yields a linear system of equations for the coefficients $c_{j\alpha}$. Each independent solution of this system yields a syzygy solution. Since the linear system is homogeneous in the unknowns $c_{j\alpha}$, we can distinguish two cases. The first is when there's no solution but the trivial one, $c_{j\alpha} = 0$ for all j, α. In this case there's no non-trivial syzygy solution compatible with the ansatz. In the second case, the system has a non-trivial solution which constraints a subset of the coefficients $c_{j\alpha}$ — which we call *dependent* coefficients — to be specific linear combinations of a second subset — which we call *independent coefficients*. More explicitly

$$c_{j\alpha} = \sum_{k,\beta} a_{jk\alpha\beta}\, c_{k\beta}, \tag{4.10}$$

where sum over k and β on the r.h.s. runs over the pairs of indexes and exponents which identify the independent coefficients $c_{k\beta}$, while $a_{jk\alpha\beta}$ are rational functions of the free parameters of the problem.

Hence, we obtain a list of syzygy solutions by substituting eq. (4.10) into the ansatz (4.9) and setting all independent coefficients but one to zero and the remaining coefficient to an arbitrary non-zero value (typically 1), for all possible choices of the non-zero independent coefficient.

Since we are solving an underdetermined system, the list of independent coefficients and thus the explicit form of the solution depend on the order of the unknowns, namely on which unknowns are eliminated with higher priority. Unknown coefficients $c_{j\alpha}$ which compare lower are eliminated with lower priority and are thus preferred to be in the independent subset. Notice that, via the procedure described before, terms which correspond to independent coefficients appear less frequently in the syzygy solutions, since in all but one they are set to zero. There are two criteria that determine the order of the coefficients that we use. The first criterion is based on the *monomial order*. In particular we have $c_{j\alpha} < c_{k\beta}$ if $\mathbf{z}^\alpha > \mathbf{z}^\beta$. This means that, monomials that compare higher generally appear less frequently in the syzygy solutions. At the time of writing, the CALICO package uses the *degree reverse lexicographic order* to compare monomials. The second criterion is the *position*, namely $c_{j\alpha} < c_{k\beta}$ if $j > k$, which means that terms in the rightmost positions of the solutions appear less frequently than terms in the leftmost positions, in the syzygy solutions. By default CALICO uses a *term over position order*, meaning the first criterion takes higher priority, while the second is used as a tie-breaker. The opposite, namely the *position over term order*, can also be optionally used.

The syzygy solutions found as we just described are not guaranteed to be independent modulo combinations of the form in eq. (4.4), except for the case where p_j are all constants. In order to obtain a minimal set of generators for the syzygy solutions, it is thus convenient to *filter out* some of them, by restricting the ansatz in eq. (4.9). This is discussed in the next subsection.

4.3 Filtering out solutions

As explained in section 2.2, we are interested in finding a minimal set of generators of parametric annihilators, from which other annihilators can be derived via monomial multiplication,

JHEP10(2025)018

differentiation with respect to the variables $\mathbf{z}$ and combinations thereof. Annihilators are, in turn, mapped into solutions of syzygy equations, which we find by turning them into linear constraints applied to the coefficients of an ansatz. In order to remove, from these, solutions that are not independent upon monomial multiplication and differentiation, we generally want to further constrain the initial ansatz in (4.9).

We generally find annihilators in multiple steps. In each step, we compute annihilators of a specific order o and degree d. The latter is identified as the degree of the polynomials $c_{j_1 j_2 \ldots}(\mathbf{z})$ in $\mathbf{z}$. We proceed from lower to higher orders o. Furthermore, annihilators of the same order are computed from lower to higher degrees d in the variables $\mathbf{z}$. When finding annihilators of order o and maximal degree d, we generally want to exclude solutions which can be found by

- differentiating annihilators of lower orders,
- multiplying solutions of the same order but lower degree by a monomial $\mathbf{z}^\alpha$

and linear combinations of the above, since they are generated by solutions found in previous steps. Before finding annihilators of order o and degree d, we thus generate a list of annihilators of the same order and maximal degree generated by those already known, as we outlined in the two points above. Each o-th order annihilator, via the method described in the previous subsection, can be mapped into a solution of a suitable syzygy equation. After this mapping, it takes the form in eq. (4.9), now with *known* coefficients $c_{j\alpha}$. Each syzygy solution is, in turn, in a one-to-one correspondence to a linear equation, namely

$$\sum_{j,\alpha} c_{j\alpha}\, \mathbf{z}^\alpha\, \hat{e}_j \quad \leftrightarrow \quad \sum_{j,\alpha} c_{j\alpha}\, y_{\alpha j} = 0, \tag{4.11}$$

where $y_{\alpha j}$ are the unknowns of this equation. By solving the system of equations generated this way from all the known solutions of order o and maximal degree d, for instance using Gaussian elimination, we are — via the mapping above — effectively taking linear combinations of the syzygy solutions. For all dependent unknowns $y_{\alpha j}$ of the solutions, we thus set to zero the corresponding coefficient $c_{\alpha j}$ in the ansatz in eq. (4.9). This guarantees that the new syzygy solutions, found as described in the previous subsection, are independent of those which are already known. The order we use for the unknowns $y_{j\alpha}$ in this system is reversed compared to the one described above for the $c_{j\alpha}$.

4.4 Computing other differential operators

Besides annihilators, in this paper we are also interested in other differential operators, such as those that are equivalent to differentiation with respect to external parameters, as explained in section 3. Their calculation is similar to the one of annihilators, with some important differences that we quickly illustrate below.

Consider, for instance, the differential operator $\hat{O}_x$ defined in eq. (3.8). Similarly to the case of annihilators, we insert the general form of an operator of order o given in eq. (3.7) into eq. (3.8), divide by $u(\mathbf{z})$ and put everything under common denominator to obtain a polynomial equation for the polynomial coefficients $c^{(x)}_{j_1 \cdots j_k}(\mathbf{z})$. This has a form that is similar to the one of a syzygy equation, except that it has a non-vanishing right-hand side

$$\mathbf{f}(\mathbf{z}) \cdot \mathbf{g}(\mathbf{z}) = h(\mathbf{z}), \tag{4.12}$$

JHEP10(2025)018

where $\mathbf{f}(\mathbf{z})$ and $\mathbf{g}(\mathbf{z})$ are lists of known and unknown polynomials respectively, as above, while $h(\mathbf{z})$ on the r.h.s. is also a known polynomial. Note that we generally need only one solution for this equation. Any other solution can be obtained by adding to it a solution of the syzygy equation (4.1).

Once again, we clarify this by means of an example. Consider a twist of the form in eq. (2.5). Following the steps outlined above for a first-order operator $\hat{O}_x$, we obtain

$$c_0^{(x)}(\mathbf{z})\prod_k B_k(\mathbf{z}) + \sum_{j=1}^{n} c_j^{(x)}(\mathbf{z}) \sum_k \gamma_k \left(\partial_j B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}) = \sum_k \gamma_k \left(\partial_x B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}), \quad (4.13)$$

which has the form in (4.12), identifying

$$\begin{aligned}
\mathbf{f}(\mathbf{z}) &= \left\{ \prod_k B_k(\mathbf{z}), \sum_k \gamma_k \left(\partial_{z_1} B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}), \ldots, \sum_k \gamma_k \left(\partial_{z_n} B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}) \right\} \\
\mathbf{g}(\mathbf{z}) &= \{c_0^{(x)}(\mathbf{z}), c_1(\mathbf{z}), \ldots, c_n^{(x)}(\mathbf{z})\} \\
h(\mathbf{z}) &= \sum_k \gamma_k \left(\partial_x B_k(\mathbf{z})\right) \prod_{l\neq k} B_l(\mathbf{z}).
\end{aligned} \quad (4.14)$$

Eq. (4.12) can be mapped into a problem of linear algebra, following a similar strategy to the one we described in the previous subsections. This has also been implemented in the CALICO package. We still make an ansatz of the form in eq. (4.9). If solutions of the corresponding syzygy equation (4.1) are known, they can be used to constrain the ansatz as already described for syzygy equations. This is not strictly necessary, because only one solution is needed, but it can improve performance. By inserting the ansatz into eq. (4.12) and matching the coefficients of each monomial $\mathbf{z}^\alpha$ appearing on either side of the equation, we obtain a linear system of identities for the coefficients $c_{j\alpha}$. This time, however, the system is not homogeneous in the unknowns, hence it may have no solution. If the system is impossible, then there's no solution compatible with the ansatz and we may proceed by making a more general ansatz with a higher maximum degree d or searching for an higher-order form of the operator $\hat{O}_x$. If there is at least one solution, we set to zero any independent coefficient $c_{j\alpha}$ that has not been constrained by the system of equations, thus obtaining the solution we seek. Once one solution has been found, the algorithm successfully terminates.

4.5 Finite-field methods and implementation details

We now provide additional details about an implementation of the methods we outlined, which we publish with this work as the Mathematica package CALICO, whose usage is described in more details in section 10. CALICO relies on the FiniteFlow [17, 27] program for solving linear systems. The latter uses a numerical sparse solver over finite fields to efficiently solve the system for numerical values of the input parameters, combined with functional and rational reconstruction techniques to recover the full analytic form of the solution.

The strategy we described starts with an analytic preparation which converts equations (2.9) and (3.8) into polynomial equations of the form in (4.1) and (4.12) respectively. This is done using a Computer Algebra System (CAS), in our case Mathematica. The main non-trivial optimization we optionally make here is avoiding the use of the explicit

JHEP10(2025)018

expressions of the polynomials which define the twist until after the equation has been cast into a polynomial form (see the function `CATTwist` described in section 10).

After eq. (4.1) or (4.12) has been obtained for a given o-th order operator, we need to make an ansatz for $\mathbf{g}(\mathbf{z})$, as in eq. (4.9) and generate a linear system for the unknown coefficients $c_{j\alpha}$ appearing in it. If implemented naively, this step can easily become a bottleneck due to the size of the expressions involved, especially when using a CAS such as Mathematica that is not optimized for handling large expressions. Communication between Mathematica and FiniteFlow can also cause significant overhead when many large expressions are present. These issues are mitigated by exploiting the general form of the equations as well as the features available in FiniteFlow, such as the possibility of combing core algorithms into more complex ones by defining them via computational graphs.

The *known* polynomials appearing in our equations can be written as

$$\begin{aligned} \mathbf{f}(\mathbf{z}) &= \sum_{j\alpha} b_{j\alpha}\, \mathbf{z}^{\alpha}\, \hat{e}_j \\ h(\mathbf{z}) &= \sum_{j\alpha} b_{0\alpha}\, \mathbf{z}^{\alpha}, \end{aligned} \tag{4.15}$$

where $b_{j\alpha}$ are *known* rational functions (typically polynomials) of the free parameters of the problem. We notice that, when these expressions are inserted in equations (4.1) or (4.12), different coefficients $b_{j\alpha}$ always multiply either different unknowns $c_{j\alpha}$ or different monomials in $\mathbf{z}$ which correspond to different equations, hence they never mix. Indeed, each entry of the matrix that defines the system of equations for the coefficients $c_{j\alpha}$ coincides with one of the coefficients $b_{j\alpha}$. We thus exploit this by first loading the coefficients $b_{j\alpha}$ in a computational graph of FiniteFlow (after removing duplicate entries) and then using this to define the system for the unknowns $c_{j\alpha}$ via list manipulations, whose performance is vastly superior to algebraic manipulations. These manipulations are meant to identify which coefficient $b_{j\alpha}$ corresponds to each entry of the matrix which defines the system, without performing any symbolic algebra. Moreover, the same set of coefficients $b_{j\alpha}$ is reused for all choices of the maximum degree of the ansatz for $\mathbf{g}(\mathbf{z})$. This greatly increases the performance, especially when many free parameters or many integration variables are present.

In our implementation, the maximum order and degree of annihilators to be found is specified by the user. In realistic applications, one would need to adjust these until either no more annihilator is found or one is convinced that the ones which have been found are sufficient for the reduction to master integrals (this is analogous to how one normally chooses the seeds).

Finally, the filtering described in 4.3 is performed numerically, i.e. by replacing all free parameters with arbitrary numerical values while solving the corresponding system, since the only information we need from it is a list of coefficients and corresponding monomials to exclude from the ansatz.

5 Hypergeometric functions

Families of integrals in the general form we defined in section 2.1 are found in all sorts of applications. They include classes of special functions, integrals representing expectations values of operators in quantum mechanics, correlation functions in quantum field theory,

JHEP10(2025)018

and loop integrals (see e.g. ref. [35]). In this section we present some application of the formalism we implemented to hypergeometric functions.

Function ${}_2F_1$. As a simple application, consider the family of univariate integrals

$$\begin{aligned} I_\alpha &= \int_0^1 \mathrm{d}z\, \varphi_\alpha(z)\, u(z) \\ \varphi_\alpha(z) &= z^\alpha, \qquad u(z) = z^{b_2-1}\,(1-z)^{b_3-b_2-1}\,(1-x\,z)^{-b_1}, \end{aligned} \tag{5.1}$$

which is related to the hypergeometric function ${}_2F_1$ by

$$I_\alpha = \frac{\Gamma(b_2+\alpha)\Gamma(b_3-b_2)}{\Gamma(b_3+\alpha)}\, {}_2F_1(b_1,\, b_2+\alpha,\, b_3+\alpha;\, x). \tag{5.2}$$

The parameters b_i are analytic regulators, while x is a free parameter. Since the integral is univariate, α here is an integer exponent rather than a list of exponents.

We first find annihilators of the twist $u(z)$. Using the algorithm described above, we find that first order annihilators are generated by

$$\hat{A} = c_0(z) + c_1(z)\,\partial_z \tag{5.3}$$

with

$$\begin{aligned} c_0(z) &= 1 - b_2 + (b_3 - 2 + (b_2 - b_1 - 1)\,x)\,z + (2 + b_1 - b_3)\,x\,z^2 \\ c_1(z) &= z - (1+x)\,z^2 + x\,z^3. \end{aligned} \tag{5.4}$$

Using this annihilator, eq. (2.11) yields

$$-(\alpha+b_2)I_\alpha + (\alpha + b_3 + (1+\alpha-b_1+b_2)x)I_{\alpha+1} + (b_1 - b_3 - 1 - \alpha)\,x\,I_{\alpha+2} = 0, \tag{5.5}$$

which can be solved to express, say, $I_{\alpha+2}$ in terms of $I_{\alpha+1}$ and I_α, or alternatively I_α in terms of $I_{\alpha+1}$ and $I_{\alpha+2}$. By using this relation for several values of α, it is straightforward to see that this family has two MIs, which we can choose as I_0 and I_1. As an example of such a reduction identity, we have

$$I_2 = \frac{b_2}{(b_1-b_3-1)\,x}\, I_0 + \frac{(b_1-b_2-1)\,x - b_3}{(b_1-b_3-1)\,x}\, I_1. \tag{5.6}$$

The integrals depend on the free parameter x and we can derive differential equations with respect to it. In order to do that, as explained in section 3, we first compute the corresponding differential operator $\hat{O}_x$, which we find to be

$$\hat{O}_x = c_0^{(x)}(z) + c_1^{(x)}(z)\,\partial_z \tag{5.7}$$

with

$$\begin{aligned} c_0^{(x)}(z) &= \frac{(b_2-1) + (2+b_1-b_3)\,z}{1-x} \\ c_1^{(x)}(z) &= \frac{z^2 - z}{1-x}. \end{aligned} \tag{5.8}$$

JHEP10(2025)018

Using eq. (3.9) we thus obtain

$$\partial_x I_\alpha = \frac{\alpha + b_2}{1-x}\, I_\alpha + \frac{b_1 - b_3 - \alpha}{1-x}\, I_{\alpha+1}. \tag{5.9}$$

Applying this to both MIs I_0, I_1 and using eq. (5.6) to reduce I_2 to MIs, we obtain a system of differential equations satisfied by the MIs

$$\partial_x \begin{pmatrix} I_1 \\ I_0 \end{pmatrix} = \begin{pmatrix} \frac{b_1\, x - b_3}{(1-x)\, x} & \frac{b_2}{(1-x)\, x} \\ \frac{b_1 - b_3}{1-x} & \frac{b_2}{1-x} \end{pmatrix} \cdot \begin{pmatrix} I_1 \\ I_0 \end{pmatrix}. \tag{5.10}$$

Alternatively, we can use this relation to treat $\partial_x I_0$ as a new master integral which replaces I_1. This change of basis yields[3] a second-order differential equation satisfied by I_0, namely

$$x\,(1-x)\,\frac{\partial^2}{\partial x^2}\, I_0 = \Big((b_1 + b_2 + 1)\, x - b_3\Big)\, \partial_x\, I_0 + \Big(b_1\, b_2\Big)\, I_0, \tag{5.11}$$

which is the well-known differential equation satisfied by the hypergeometric function ${}_2F_1(b_1, b_2, b_3; x)$.

Functions ${}_{n+1}F_n$. As a generalization of the previous example, we consider the integral family defined as in eq. (2.4) with

$$\varphi_\alpha(\mathbf{z}) = \mathbf{z}^\alpha, \qquad u(\mathbf{z}) = \left(1 - x \prod_{j=1}^n z_j\right)^{-a_{n+1}} \prod_{j=1}^n \left[z_j^{a_j - 1} (1 - z_j)^{b_j - a_j - 1}\right], \tag{5.12}$$

where the integration contour is $z_j \in (0,1)$ for $j = 1, \ldots, n$, while a_j and b_j are analytic regulators and x a free parameter.

These integrals are related to the hypergeometric functions ${}_{n+1}F_n$ by

$$\begin{aligned} I_\alpha = &\prod_{j=1} \left[\frac{\Gamma(a_j + \alpha_j)\Gamma(b_j - a_j)}{\Gamma(b_j + \alpha_j)}\right] \\ &\times {}_{n+1}F_n(a_1 + \alpha_1, \ldots, a_n + \alpha_n, a_{n+1}; b_1 + \alpha_1, \ldots, b_n + \alpha_n; x). \end{aligned} \tag{5.13}$$

We empirically find that, for various choices of n, one or more first-order parametric annihilators of $u(\mathbf{z})$ of degree 3 and $n+2$ exist. Higher-order generators of annihilators also exist for $n > 1$ but, even though their generators are independent, the identities they generate are not independent of the ones obtained from first-order annihilators — as we empirically verified they do not add independent constraints among the integrals.

Using eq. (2.11) we generate identities satisfied by the integrals I_α. The solution of the linear system yields $n+1$ independent master integrals, which we may choose as

$$\{I_{0\cdots 0}\} \cup \{I_{\hat{e}_j}\}_{j=1}^n. \tag{5.14}$$

[3] Given a system of equations $\partial_x G_i = \sum_j M_{ij}\, G_j$ and a new basis $\tilde{G}_i = \sum_j T_{ij}\, G_j$, the latter satisfies the new differential equation $\partial_x \tilde{G}_i = \sum_j \tilde{M}_{ij}\, \tilde{G}_j$ with $\tilde{M} = (\partial_x T) \cdot T^{-1} + T \cdot M \cdot T^{-1}$. In our example, the new basis is $(\partial_x I_0, I_0)$ where $\partial_x I_0$ can be read from eq. (5.10). The new first order DE satisfied by $(\partial_x I_0, I_0)$ is equivalent to a second-order DE satisfied by I_0.

JHEP10(2025)018

We are also able to find a first-order differential operator $\hat{O}_x$ of polynomial degree $n+1$ in $\mathbf{z}$ which implements differentiation with respect to the parameter x, defined as in equations (3.7) and (3.8). With this, we find differential equations for the $n+1$ MIs. These identities have been checked against numerical evaluations of the hypergeometric functions. Similarly to the previous case, this can also be cast as a $(n+1)$-th order differential equation for the master integral $I_{0\cdots0}$.

6 Loop integrals

As already stated, the formalism we described can be applied to a wide variety of problems. The main focus of our work, however, is the study of loop integrals in dimensional regularization. These are commonly organized into families. A family of loop integrals includes linear combinations of integrals that, in momentum space, have the form

$$J_\alpha = J_{\alpha_1\cdots\alpha_n} = \int \prod_{j=1}^{\ell} \frac{\mathrm{d}^d k_j}{i\pi^{d/2}} \frac{1}{D_1^{\alpha_1}\dots D_n^{\alpha_n}}, \tag{6.1}$$

for any multi-index of integers α. The denominators D_j in the integrands are functions of the ℓ loop momenta $k_1,\dots,k_\ell$ and the e linearly independent external momenta $p_1,\dots,p_e$. These integrals generally contribute to scattering matrix elements or Green functions with $e+1$ external momenta, one of which is fixed as a linear combination of the others by momentum conservation. The integral is performed in a generic number d of dimensions and it is an analytic function of d.

Each loop integral family is thus defined by the set of generalized denominators D_j. For each integral J_α, we split these into two disjoint subsets: *proper denominators* are D_j with $\alpha_j > 0$, while *irreducible scalar products (ISPs)* are D_j with $\alpha_j \leq 0$ (i.e. contributing to the numerator of the integrand). The denominators D_j typically have the quadratic form

$$D_j = l_j^2 - m_j^2, \tag{6.2}$$

but one can also use the bi-linear form

$$D_j = l_j \cdot v_j - m_j^2, \tag{6.3}$$

where m_j are internal masses, l_j are linear combinations of loop momenta and external momenta, while v_j are linear combinations of external momenta only. For loop integrals appearing in scattering amplitudes or Green functions in Quantum Field Theory (QFT), the bi-linear form is only allowed for ISPs.

When computing loop integrals or deriving linear relations for them, it is useful to partition them into subsets called *sectors*. We define sectors

$$S_\alpha = S_{\alpha_1\cdots\alpha_n}, \qquad \text{with } \alpha_j = 0, 1 \tag{6.4}$$

such that

$$J_\alpha \in S_{\Theta(\alpha_1-1/2)\cdots\Theta(\alpha_n-1/2)} \tag{6.5}$$

JHEP10(2025)018

with Θ being the Heaviside step function. Each sector can also be identified by its *corner integral*, that is the only integral J_α of the sector with $\alpha_j \in \{0, 1\}$ for all j. Given two sectors S_α and S_β, we say that S_α is a *subsector* of S_β if $\alpha \neq \beta$ and $\alpha_j \leq \beta_j$ for all j.

It is also common to define, for each integral J_α, the number

$$t = \sum_{j=1}^{n} \Theta(\alpha_j - 1/2), \tag{6.6}$$

which is the same for integrals belonging to the same sector, as well as the degree or *rank* s of the numerator

$$s = -\sum_{j=1}^{n} \alpha_j \Theta(-\alpha_j) \tag{6.7}$$

and the numbers of *dots*, that is the sum of the powers of denominators in excess with respect to those of the corner integral of the same sector, namely

$$u = \sum_{j=1}^{n} (\alpha_j - 1)\Theta(\alpha_j - 1/2). \tag{6.8}$$

These numbers are often used to characterize the complexity of an integral.

In an integral family, we identify one or more *top sectors*. These are a minimal set of sectors such that all integrals of interest, for solving a particular problem, either belong to those sectors or to their subsectors. In the examples illustrated in this paper, we focus on families with just one top sector.

Linear identities among loop integrals are often found using IBPs in momentum space [1, 2], which read

$$\int \prod_{j=1}^{\ell} \frac{\mathrm{d}^d k_j}{i\pi^{d/2}} \frac{\partial}{\partial k_m^\mu} \frac{v^\mu}{D_1^{\alpha_1} \dots D_n^{\alpha_n}} = 0, \quad \text{with } v^\mu = k_i^\mu, p_i^\mu. \tag{6.9}$$

By carrying out the derivatives explicitly, we obtain a non-trivial linear combination of integrals that vanish. These integrals, however, generally belong to the same family in eq. (6.1) only if we are able to rewrite all scalar products of the form $k_i \cdot k_j$ and $k_i \cdot p_j$ as linear combinations of generalized denominators. This implies that the number n of denominators must be

$$n = \ell e + \frac{\ell(\ell + 1)}{2}, \tag{6.10}$$

which we can achieve by defining a suitable number of ISPs and adding them to the list of denominators that appear in the loop propagators of the amplitude or Green function that is being computed. In the applications we will describe later, however, we will see that, when using parametric representations of loop integrals combined with annihilators, we are able to find integral identities without having to introduce a full set of ISPs. This can drastically simplify the problem in some applications, by allowing a number n of generalized denominators that is *lower* that the one in eq. (6.10).

JHEP10(2025)018

7 Baikov representations

7.1 Standard Baikov representation

The first parametric representation of loop integrals we discuss is the Baikov representation [5], where loop integrals take the form in eq. (2.4), namely

$$J_\alpha = K\, I_\alpha, \tag{7.1}$$

with

$$I_\alpha = \int \mathrm{d}^n\mathbf{z}\, \frac{1}{\mathbf{z}^\alpha}\, B(\mathbf{z})^\gamma \tag{7.2}$$

with $\gamma = \frac{d-\ell-e-1}{2}$. This is equivalent to the identifications

$$\varphi_\alpha = \frac{1}{\mathbf{z}^\alpha}, \qquad u(\mathbf{z}) = B(\mathbf{z})^\gamma \tag{7.3}$$

in eq. (2.4). The prefactor K is the same for all integrals within a family, hence completely irrelevant when finding linear relations. However, it generally contains a factor $B_0^{\gamma_0}$ — with γ_0 a function of d and B_0 independent of $\mathbf{z}$ but dependent on the external kinematic variables — which must be taken into account when deriving DEs. The integration contour is not important for the purposes of this paper, except for the property that boundary terms in IBPs can be set to zero.

The standard Baikov representation requires a full set of ISPs, such that the number n of generalized denominators, equal to the number of integration variables in eq. (2.4), is given by eq. (6.10). Identities between integrals in the Baikov representation have already been extensively discussed in the literature, mostly using syzygy approaches, which we showed in section 2.3 to be equivalent to using identities generated by first-order annihilators. Since a complete set of syzygy solutions for this parametrization is known in closed form [36], we conclude that first-order annihilators (just like syzygy solutions) are generated by $n+1$ generators, all of which have degree 1. We are not aware of examples where higher-order annihilators are needed for a complete reduction to master integrals in this representation.

We also recall that, as already stated in section 2.3, additional syzygy equations or constraints are often used to exclude integrals with higher power of denominators from the identities. This strategy can be applied within both approaches but its discussion is outside the purposes of this work.

7.2 Loop-by-loop Baikov representation

A closely related representation of loop integrals is the *loop-by-loop Baikov parametrization* [6]. This consists in applying the Baikov parametrization one loop at the time. The two main differences with the standard Baikov parametrization are:

- the twist $u(\mathbf{z})$, up to a common overall factor, takes the form in eq. (2.5), more precisely it is the product of $2\ell-1$ polynomials $B_j(\mathbf{z})$ raised to exponents γ_j that depend on the number of space-time dimensions d;
- the number n of integration variables is generally lower than the one in eq. (6.10), hence fewer ISPs are generally needed to define a family.

JHEP10(2025)018

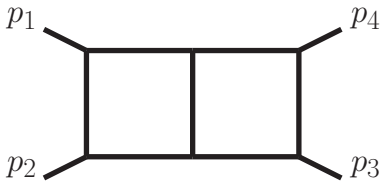


Figure 1. Double-box.

Via the method of annihilators, we can derive identities in the loop-by-loop Baikov parametrization of loop integrals, thus having fewer ISPs and integration variables with respect to the standard Baikov parametrization or the momentum representation.

Example. We consider a massless double-box integral family. Its top sector has 7 proper denominators and corresponds to the diagram depicted[4] in figure 1. When IBPs in momentum representation are used, 2 additional generalized denominators are needed, to match eq. (6.10). In the loop-by-loop Baikov representation, instead, only 1 additional generalized denominator is required. The full list of $n = 8$ generalized denominators can be chosen as in eq. (6.2) with $m_j = 0$ and

$$\begin{aligned} &l_1 = k_1, && l_2 = k_1 + p_1, && l_3 = k_1 + p_1 + p_2, \\ &l_4 = k_1 + k_2, && l_5 = k_2, && l_6 = k_2 - p_1 - p_2 - p_3, \\ &l_7 = k_2 - p_1 - p_2, && l_8 = k_1 + p_1 + p_2 + p_3. \end{aligned} \tag{7.4}$$

For simplicity, we consider massless external legs, $p_j^2 = 0$. This implies there are two independent invariants, which we may choose as

$$s_{12} = (p_1 + p_2)^2, \qquad s_{23} = (p_2 + p_3)^2. \tag{7.5}$$

We consider integrals of the form in eq. (6.1) with

$$\alpha_j \leq 0, \qquad \text{for } j > 7. \tag{7.6}$$

By integrating out the k_2-dependent loop first and k_1 second,[5] the loop-by-loop Baikov representation yields a twist of the form

$$u(\mathbf{z}) = C(d)\, B_0^{\frac{4-d}{2}}\, B_1(\mathbf{z})^{\frac{d-5}{2}}\, B_2(\mathbf{z})^{\frac{4-d}{2}}\, B_3(\mathbf{z})^{\frac{d-5}{2}}, \tag{7.7}$$

where B_0 is independent of $\mathbf{z}$, hence it is irrelevant when finding linear relations, but it depends on the invariants s_{ij}, therefore it needs to be taken into account for deriving differential equations. The explicit expressions of $B_j(\mathbf{z})$ are reported in the examples within the public

[4]The conventions for the graphs depicted in this paper are: black single lines correspond to massless particles (on-shell if external), external double lines are off-shell external momenta, coloured lines correspond to massive particles (having the same mass, unless explicitly stated otherwise).

[5]We may also chose to integrate k_1 first, but in this case the generalized denominator D_8 needs to be replaced with D_9, with $l_9 = k_2 - p_1$. Note that both D_8 and D_9 are instead needed for representations satisfying eq. (6.10), i.e. requiring a total of 9 generalized denominators for our example.

JHEP10(2025)018

repository of the CALICO package. The factor $C(d)$, instead, only depends on d and is thus irrelevant for both applications.

Using CALICO, we find 8 first-order annihilators of degree 1 and 1 first-order annihilator of degree 2. We find that these, combined with some symmetry relations, are sufficient for a complete reduction to master integrals. We also empirically found that this twist admits 9 independent second-order annihilators of degree 1, which however are not required for the reduction to master integrals.

For this example, we used symmetry relations found with the help of the private package FFIntRed. For this purpose, we used the momentum representation of this integral family and searched for shifts of loop momenta which map generalized denominators into each other — modulo permutations of external momenta which do not change the two invariants s_{12} and s_{23}. It was sufficient to generate these symmetries up to rank $s = 1$. These supplementary identities generally introduce an additional scalar product $k_2 \cdot p_1$ which is not a linear combination of the 8 generalized denominators. This is, however, easily removed by taking suitable linear combinations of symmetry relations. We thus simply generated symmetry relations in momentum space involving integrals with rank $s = 0, 1$, and used Gauss elimination to remove from them integrals having the unwanted scalar product $k_2 \cdot p_1$.

The reduction yields 8 independent master integrals, in agreement with the traditional Laporta algorithm in momentum space. A possible choice of them is

$$\{I_{1111111-1}, I_{11111110}, I_{11010110}, I_{11110100}, I_{10101010}, I_{10110100}, I_{01010100}, I_{10010010}\}. \tag{7.8}$$

We also found differential operators $\hat{O}_{s_{12}}$ and $\hat{O}_{s_{23}}$ as first-order operators of degree 1. Using this, we take derivatives of the master integrals (cfr. eq. (3.9)) which, combined with the linear relations above, yield a system of differential equations for the master integrals.

8 Duals of loop integrals

In this section, we apply the formalism of parametric annihilators to *duals of loop integrals*. Duals of loop integrals play a crucial role within the framework of *intersection theory*. The latter defines scalar products, called *intersection numbers*, between integrals having a form analogous to the one in eq. (2.4), including — as we have also reviewed in this paper — various parametrizations of loop integrals. A review of intersection theory is outside the scope of this paper. We refer the reader to [21, 22, 37–40] and references therein. In the following, after some motivation, we recall only the main details that are relevant to our application of parametric annihilators to duals of loop integrals.

The definition of intersection numbers, similarly to any other scalar product, relies on the one of a *dual* vector space. While dual integrals are often studied in the context of intersection theory, there are several reasons to appreciate an alternative method to compute the linear relations they satisfy. We give a quick review of some of them. Modern methods for computing intersection numbers involve deriving differential equations satisfied by dual integrals in fewer integration variables. While these differential equations can be (and usually are) derived using intersection numbers, our approach offers an alternative to that. Even when such differential equations are computed via intersection numbers, one can still avoid

JHEP10(2025)018

the appearance of shifted integrals by expressing derivatives via differential operators as in section 3, potentially simplifying their calculation. Moreover, once the *metric*, i.e. a set intersection numbers between MIs and dual MIs, has been computed — possibly choosing the MIs in a way that makes this calculation as easy as possible — any other intersection number is uniquely fixed by reductions of regular integrals and dual integrals. While the intersection numbers themselves are often used for reductions, they are also interesting mathematical objects by themselves and can give important insights about certain classes of integrals (see e.g. [41–43]). Hence, having an alternative method for computing them, after the metric has been fixed, can be highly beneficial. Finally, this method enables many checks, including consistency checks between intersection numbers and reduction identities of dual integrals.

We now briefly review how dual integrals can be defined, following the approach of ref. [20]. For properly regulated integrals, the dual space can be defined by replacing $u(\mathbf{z}) \to 1/u(\mathbf{z})$ in eq. (2.4) — which is equivalent to $\gamma \to -\gamma$ in the Baikov parametrization (7.2). Unfortunately this is problematic for loop integrals, since it yields impossible systems of equations to be solved during the calculation of intersection numbers. When using the Baikov parametrization, this is due to singularities at $z_j \to 0$ in $\varphi_\alpha(\mathbf{z})$ that are not properly regulated by the twist $u(\mathbf{z})$.

In the following, up to a relabeling of the integration variables, we will assume that $(z_1, \ldots, z_m)$ are proper denominators of the top sector, while $(z_{m+1}, \ldots, z_n)$ are its ISPs. This implies that $\alpha_j \leq 0$ for $j > m$. A possible solution [38] to the issue above consists in replacing the twist $u(\mathbf{z})$ with

$$u(\mathbf{z}) \to u_\rho(\mathbf{z}) = u(\mathbf{z})\, \mathbf{z}^\rho, \tag{8.1}$$

where

$$\rho = (\rho_1, \ldots, \rho_m, 0, \ldots, 0) \tag{8.2}$$

is a list of additional regulators for the proper denominators. While intersection numbers are singular in the limit $\rho_j \to 0$, the coefficients of a reduction to MIs are finite. However, since this limit can only be taken at the very end, the need of additional regulators constitutes a major bottleneck and drawback of this approach.

In references [18, 20], different approaches to defining the space of duals of loop integrals were presented. These sidestep the need of regulators. Here we focus on the one proposed in [20], which exploits our freedom of choice of dual loop integrals to effectively avoid the need of performing algebraic operations using regulators. More precisely, it consists in choosing dual integrals I_α of the form

$$I_\alpha = \prod_{j=1}^{m} \rho_j^{\Theta(\alpha_j - 1/2)} \int \mathrm{d}^n\mathbf{z}\, \frac{1}{\mathbf{z}^{\alpha-\rho}}\, B(\mathbf{z})^{-\gamma}. \tag{8.3}$$

with ρ defined in (8.2) and systematically work on the leading coefficients of the $\rho_j \to 0$ limit that yield a finite contribution to intersection numbers (see below for more details). In other words, dual integrals are multiplied by a factor ρ_j if $\alpha_j > 0$. Ref. [44] shows (as a formal proof in the univariate case and heuristically in the multivariate one) that the approaches of references [18] and [20] yield equivalent results for intersection numbers, despite being significantly different in their formulation and computationally.

JHEP10(2025)018

While, in the context of intersection theory, the factors $z_j^{\rho_j}$ are commonly absorbed into the definition of $u(\mathbf{z})$, for the purposes of this paper it is more convenient to define

$$\varphi_\alpha(\mathbf{z}) = \prod_{j=1}^{m} \rho_j^{\Theta(\alpha_j - 1/2)} \frac{1}{\mathbf{z}^{\alpha-\rho}}, \qquad u(\mathbf{z}) = B(\mathbf{z})^{-\gamma} \tag{8.4}$$

for duals of loop integrals in the Baikov parametrization. We can similarly define duals of integrals in the loop-by-loop Baikov parametrization, with the only difference that $u(\mathbf{z})$ will take the form in eq. (2.5) with opposite exponents $\gamma_j \to -\gamma_j$ with respect to loop integrals. When computing intersection numbers, if φ_α is such that $\alpha_j > 0$, then intersection numbers involving the dual integral I_α have a $1/\rho_j$ pole that simplifies their ρ_j prefactor yielding a finite result. Hence, when $\rho_j \to 0$, we can "effectively" regard a factor $z_j^{-\alpha_j}$ of the integrand as being $\mathcal{O}(1/\rho_j)$ if $\alpha_j > 0$ and $\mathcal{O}(1)$ if $\alpha_j \leq 0$. Our prescription of working on the leading terms of the $\rho_j \to 0$ limit that yield finite contributions to intersection numbers effectively amounts to the following rules

$$\rho_j^2 \frac{1}{\mathbf{z}^{\alpha-\rho}} u(\mathbf{z}) \to 0 \quad \forall \alpha, \qquad \rho_j \frac{1}{\mathbf{z}^{\alpha-\rho}} u(\mathbf{z}) \to 0 \quad \text{if } \alpha_j \leq 0, \tag{8.5}$$

which can be implemented as substitutions at the integrand level when dealing with dual integrals. For a more comprehensive justification of these rules we refer to section 4 of [20].

8.1 Reduction of dual integrals via annihilators

Applying the annihilator approach to dual integrals is straightforward. First, annihilators of $u(\mathbf{z}) = B(\mathbf{z})^{-\gamma}$ can obviously be computed using the methods described in the previous sections. In practice, one can use the same annihilators of loop integrals in the Baikov representation and simply replace $\gamma \to -\gamma$ (or, equivalently, $d \to 2 + 2\ell + 2e - d$) in their expression. Once a set of annihilators is known, identities are found, as before, applying eq. (2.11). The latter requires knowing the behaviour of φ_α under multiplication by monomials and differentiation. Our rules in eq. (8.5) imply that, for monomial multiplication, with an exponent $\beta_j \geq 0$,

$$z_j^{\beta_j}\, \varphi_\alpha(\mathbf{z}) = \begin{cases} \varphi_{\alpha-\beta_j\hat{e}_j}(\mathbf{z}) & \text{if } \alpha_j > \beta_j \text{ or } \alpha_j \leq 0 \\ 0 & \text{if } 0 < \alpha_j \leq \beta_j. \end{cases} \tag{8.6}$$

This shows that φ_α *cuts* the propagators j such that $\alpha_j > 0$. Differentiation instead works with the following rules:

$$\partial_j\, \varphi_\alpha(\mathbf{z}) = \begin{cases} -(\alpha_j - \delta_{\alpha_j 0})\, \varphi_{\alpha+\hat{e}_j}(\mathbf{z}) & \text{if } j \leq m \\ -\alpha_j\, \varphi_{\alpha+\hat{e}_j}(\mathbf{z}) & \text{if } j > m. \end{cases} \tag{8.7}$$

Hence derivatives follow the normal rules (as if regulators ρ were not present) except for the case of zero exponents of regulated integration variables. If $\alpha_j = 0$ for $j \leq m$, then ∂_j generates the denominator $1/z_j$.

We stress that, in practice, we only need to implement the rules in eq. (8.6) and (8.7), hence regulators never explicitly appear in our calculation. Of course, besides reduction identities, one can also derive differential equations for dual integrals, following section 3.

JHEP10(2025)018

From equations (8.6) and (8.7) it follows that, as already noted in the literature [20], reduction identities and differential equations for dual integrals have a block triangular structure that is transposed with respect to the one of loop integrals. For this reason, we find that a good choice of weight for duals of loop integrals would assign a lower weight to integrals with higher values of t, i.e. with more proper denominators, which is the opposite of what is commonly done for loop integrals.

We tested this approach on several examples and successfully compared against reductions performed using intersection numbers, with the algorithm of ref. [20].

8.2 Computation of connection matrices

An important ingredient for computing multivariate intersection numbers using the recursive method in [37, 38] are the so-called connection matrices. We rewrite eq. (2.4) as

$$I_\alpha = \int \mathrm{d}z_{k+1} \cdots \mathrm{d}z_n \, I_\alpha^{(k)} \tag{8.8}$$

with

$$I_\alpha^{(k)} \equiv \int \mathrm{d}z_1 \cdots \mathrm{d}z_k \, \varphi_\alpha(\mathbf{z}) \, u(\mathbf{z}). \tag{8.9}$$

The k-fold integral $I_\alpha^{(k)}$ has a parametric dependence on $z_{k+1}, \ldots, z_n$. Hence, one can find a basis of master integrals for $I_\alpha^{(k)}$ and derive differential equation with respect to z_{k+1}

$$\frac{\mathrm{d}}{\mathrm{d}z_{k+1}} I_\alpha^{(k)} = \sum_{\beta \in \mathrm{MIs}} \Omega_{\alpha\beta}^{(k)} \, I_\beta^{(k)}. \tag{8.10}$$

The calculation of the *connection matrix* $\Omega_{\alpha\beta}^{(k)}$ for *duals* of Feynman integrals, with $k = 1, \ldots, n-1$, is a key ingredient of the recursive method for computing $(k+1)$-variate intersection numbers in terms of k-variate intersection numbers.

The connection matrices can be computed with the formalism described in this paper, simply by treating $z_{k+1}, \ldots, z_n$ as free parameters, hence following section 3 to compute differential equation with respect to them. In particular, following this strategy, differential operators $\hat{O}_{z_{k+1}}$ are written in terms of derivatives w.r.t. the first k integration variables, which are thus turned into linear combinations of dual integrals using eq. (8.6) and (8.7).

A simple example. We describe a simple example for computing a connection matrix for the dual integrals of the one-loop massive bubble family with internal mass m^2 depicted in figure 2. The family is defined by the proper denominators

$$D_1 = k^2 - m^2, \qquad D_2 = (k-p)^2 - m^2. \tag{8.11}$$

It has one independent external leg, p, that satisfies

$$p^2 = s. \tag{8.12}$$

The Baikov representation for dual integrals reads

$$I_\alpha = \int \mathrm{d}^2\mathbf{z} \, \varphi_\alpha(\mathbf{z}) \, u(\mathbf{z}), \tag{8.13}$$

JHEP10(2025)018

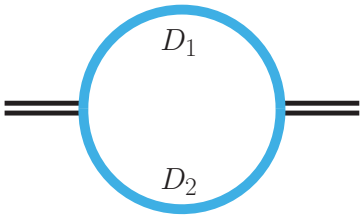


Figure 2. Massive one-loop bubble.

with $\alpha = (\alpha_1, \alpha_2)$ and

$$\varphi_\alpha(\mathbf{z}) = \rho_1^{\Theta(\alpha_1-1/2)}\, \rho_2^{\Theta(\alpha_2-1/2)}\, \frac{1}{z_1^{\alpha_1}\, z_2^{\alpha_2}} \tag{8.14}$$

$$u(\mathbf{z}) = \Big(4m^2 s - s^2 + 2s(z_1+z_2) - (z_1-z_2)^2\Big)^{\frac{3-d}{2}}. \tag{8.15}$$

The recursive algorithm to calculate intersection numbers requires the variables to be ordered. We choose the ordering where z_1 is the innermost layer. We can rewrite the bubble integral family in the iterated form

$$I_\alpha = \rho_2^{\Theta(\alpha_2-1/2)} \int \mathrm{d}z_2\, \frac{1}{z_2^{\alpha_2}}\, I^{(1)}_{\alpha_1}, \tag{8.16}$$

$$I^{(1)}_{\alpha_1} = \int \mathrm{d}z_1\, u(\mathbf{z})\, \varphi_{\alpha_1}(z_1). \tag{8.17}$$

where the integral $I^{(1)}_{\alpha_1}$ has a parametric dependence on z_2 and

$$\varphi_{\alpha_1}(z_1) = \rho_1^{\Theta(\alpha_1-1/2)}\, \frac{1}{z_1^{\alpha_1-\rho_1}}. \tag{8.18}$$

We aim to compute the connection matrix $\Omega^{(1)}$ associated to the differential equation satisfied by the 1−fold integrals $I^{(1)}_{\alpha_1}$:

$$\frac{\mathrm{d}}{\mathrm{d}z_2} I^{(1)}_{\alpha_1} = \sum_{\beta \in \mathrm{MIs}} \Omega^{(1)}_{\alpha_1\beta}\, I^{(1)}_\beta. \tag{8.19}$$

The matrix $\Omega^{(1)}$ is needed to express intersection numbers for the two-fold integrals in eq. (8.16) in terms of intersection numbers for the one-fold integrals in (8.17).

As a first step, we compute annihilators for the twist containing derivatives with respect to z_1 only, since it is the only integration variable for the innermost layer we are considering. We find one first-order annihilator

$$\hat{A} = c_0(\mathbf{z}) + c_1(\mathbf{z})\, \partial_{z_1} \tag{8.20}$$

with the following expression for the polynomial coefficients $c_j(\mathbf{z})$

$$\begin{aligned} c_0(\mathbf{z}) &= (3-d)(s - z_1 + z_2), \\ c_1(\mathbf{z}) &= -4m^2 s + s^2 - 2s(z_1+z_2) + (z_1-z_2)^2. \end{aligned} \tag{8.21}$$

Integral identities are thus found applying eq. (2.11), which in this case takes the form

$$\int \mathrm{d}z_1 u(\mathbf{z}) \Big(c_0(\mathbf{z})\, \varphi_{\alpha_1} - \partial_{z_1} \left(c_1(\mathbf{z})\, \varphi_{\alpha_1} \right) \Big) = 0. \tag{8.22}$$

The terms in parentheses are computed by applying the rules in equations (8.6) and (8.7). As an example, for the choice $\alpha_1 = 1$ the equation reads

$$\left(-4m^2 s + s^2 - 2sz_2 + z_2^2 \right) I_2^{(1)} - (d-3)(s+z_2) I_1^{(1)} = 0, \tag{8.23}$$

and can be solved to rewrite for $I_2^{(1)}$ in terms of $I_1^{(1)}$. Using this and other identities valid for various integer values of α_1, we are able to reduce all integrals $I_{\alpha_1}^{(1)}$ to two master integrals, which can be chosen as

$$\{ I_1^{(1)}, I_0^{(1)} \}. \tag{8.24}$$

To obtain the connection matrix $\Omega^{(1)}$, we derive the differential operator $\hat{O}_{z_2}$, which realizes the differentiation with respect to z_2 for the twist $u(\mathbf{z})$ as in eq. (3.8). Once again, since we are considering the innermost integration in z_1, the operator must be polynomial in z_1 and can only contain derivatives with respect to z_1. On the other hand, it may have a rational dependence on z_2, m^2 and d. We find the first-order operator

$$\hat{O}_{z_2} = c_0^{(z_2)}(\mathbf{z}) + c_1^{(z_2)}(\mathbf{z})\, \partial_{z_1} \tag{8.25}$$

with

$$\begin{aligned} c_0^{(z_2)}(\mathbf{z}) &= -\frac{d-3}{2(m^2+z_2)} \\ c_1^{(z_2)}(\mathbf{z}) &= -\frac{2m^2 - s + z_1 + z_2}{2(m^2+z_2)}. \end{aligned} \tag{8.26}$$

In particular, applying it to the master integrals $I_0^{(1)}$ and $I_1^{(1)}$, using (3.9) and again equations (8.6) and (8.7), we obtain

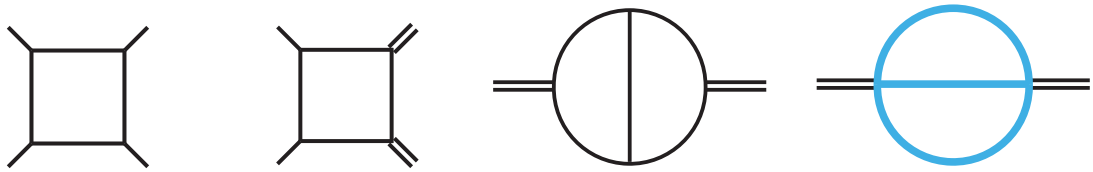
$$\begin{aligned} \partial_{z_2} I_0^{(1)} &= \frac{(4-d) I_0^{(1)}}{2(m^2+z_2)} + \frac{(2m^2 - s + z_2) I_1^{(1)}}{2(m^2+z_2)}, \\ \partial_{z_2} I_1^{(1)} &= \frac{(3-d) I_1^{(1)}}{2(m^2+z_2)} + \frac{(-2m^2 + s - z_2) I_2^{(1)}}{2(m^2+z_2)}. \end{aligned} \tag{8.27}$$

By inserting the reduction of $I_2^{(1)}$ to master integrals, found by solving (8.23), we finally obtain the system of first order differential equation in eq. (8.19) where the connection matrix reads

$$\Omega^{(1)} = \begin{pmatrix} \frac{(d-3)(s-z_2)}{(s-z_2)^2 - 4m^2 s} & 0 \\ \frac{2m^2 - s + z_2}{2(m^2+z_2)} & -\frac{d-4}{2(m^2+z_2)} \end{pmatrix}. \tag{8.28}$$

Additional examples. We tested this approach on all the connection matrices for all the integral families with non-trivial (regulated) denominators appearing in the applications of ref. [20]. The diagrams corresponding to their top sectors are depicted in figure 3. These tests are published with the CALICO program.

JHEP10(2025)018



(a) One-loop box (b) One-loop hard box (c) Kite (d) Equal-mass sunrise

Figure 3. Integral families with regulated denominators considered in ref. [20], for which connection matrices of inner integration layers have been checked against the method proposed in this paper.

9 Loop integrals as twisted Mellin moments

In this paper, we define *twisted Mellin moments* to be families of integrals of the form in eq. (2.4) with integration contour $z_i \in (0, \infty)$ and

$$\varphi_\alpha = \left(\prod_{j=1}^{n} \frac{1}{\Gamma(\alpha_j)} \right) \mathbf{z}^{\alpha-1}, \tag{9.1}$$

where $\alpha - 1 = (\alpha_1 - 1, \ldots, \alpha_n - 1)$. Several parametric representations of loop integrals (see eq. (6.1)) have this form, where we identify

$$J_\alpha \propto I_\alpha, \tag{9.2}$$

where the proportionality factor depends on the specific representation and on the exponents α.

We note that, while the space of functions φ_α is closed under monomial multiplication and differentiation, hence satisfying our assumptions, the $\Gamma(\alpha_j)$ factors need to be adjusted accordingly when converting monomials into functions φ_α. In particular, a derivative of the twist can be converted into

$$\int_0^\infty \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z})\, \partial_{z_j} u(\mathbf{z}) = -\int_0^\infty \mathrm{d}^n\mathbf{z}\, \varphi_{\alpha-\hat{e}_j}(\mathbf{z})\, u(\mathbf{z}), \tag{9.3}$$

where, after an IBP, we used $\Gamma(\alpha_j) = (\alpha_j - 1)\,\Gamma(\alpha_j - 1)$. It would naively appear that, in the IBP we just used, we neglected a boundary term at $z_j = 0$ in the special case $\alpha_j = 1$. However, one can check that the equality one finds assuming $\alpha_j > 1$ (such that the boundary term does not contribute) is analytic in α and can be continued[6] to $\alpha_j = 1$ and, in fact, also to $\alpha_j \leq 0$. Indeed, the dimensionally regulated integrals J_α can be analytically continued in the exponents α_j and, for generic d, have non-singular limits for integer values of α_j. This is also true for the integrals I_α we just defined, since the proportionality factor in eq. (9.2) is non-singular in these limits, as we will see below. We can thus write template equations from annihilators for generic exponents, neglecting boundary terms in IBPs, and then analytically continue them to any value of α_j, including zero or negative integers.

[6]For the special case $\alpha_j = 1$, one can easily check that only the boundary term at $z_j = 0$ contributes to the IBP identity. However, in parametric representations of Feynman integrals, setting $\alpha_j = 1$ and $z_j = 0$ in the whole integrand (including the twist) corresponds to removing the j-th denominator, which in turn corresponds to setting the exponent $\alpha_j = 0$ in eq. (6.1). Hence, eq. (9.3) is still valid in the limit $\alpha_j \to 1$, once written in terms of J_α.

JHEP10(2025)018

9.1 Lee-Pomeransky representation

The Lee-Pomeransky representation of loop integrals is related to the twisted Mellin moments with the special choice of twist

$$u(\mathbf{z}) = G(\mathbf{z})^{-d/2} \tag{9.4}$$

where d is a regulator. The loop integrals in eq. (6.1) can be written in this form, up to a prefactor

$$J_\alpha = (-1)^{|\alpha|} \frac{\Gamma\left(\frac{d}{2}\right)}{\Gamma\left(\frac{(\ell+1)d}{2} - |\alpha|\right)} I_\alpha, \tag{9.5}$$

by identifying d with the space-time dimension and the polynomial $G(\mathbf{z})$ as the sum of the two Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$ (see appendix A for a definition),

$$G(\mathbf{z}) = \mathcal{U}(\mathbf{z}) + \mathcal{F}(\mathbf{z}). \tag{9.6}$$

This is known as the Lee-Pomeransky parametrization of loop integrals [7]. Parametric annihilators in this integral representation have already been extensively discussed in [10].

In this section, as well as in the CALICO program presented in section 10, we deal with Mellin moments of the form in eq. (9.1). Converting a reduction of the form in eq. (2.12) for Mellin integrals into one for the corresponding loop integrals

$$J_\alpha = \sum_{\beta \in \text{MIs}} \tilde{c}_{\alpha\beta}\, J_\beta \tag{9.7}$$

is straightforward by identifying

$$\tilde{c}_{\alpha\beta} = (-1)^{|\beta|-|\alpha|} \frac{\Gamma\left(\frac{(\ell+1)d}{2} - |\beta|\right)}{\Gamma\left(\frac{(\ell+1)d}{2} - |\alpha|\right)} c_{\alpha\beta}. \tag{9.8}$$

Note that, because both $|\alpha|$ and $|\beta|$ are integers, the arguments of the two Gamma functions in the ratio above differ by an integer number, hence it is always a rational function of d. The latter can be found by recursively applying the relation $\Gamma(t+1) = t\,\Gamma(t)$ to shift the argument of one of the two Γ functions until they simplify yielding simple rational factors. In practice, we find more convenient to apply this relation directly at the level of the template equations, to translate the identities in terms of loop integrals before they are solved.

The Lee-Pomeransky parametrization of loop integrals has a number of advantages compared to other representations, such as Baikov's or momentum space. First, it only involves n integration variables, where n is the number of loop denominators, with no need of introducing ISPs — although the latter can also be added if required. Moreover, the degree of the polynomial G is always $\deg G = \ell + 1$ at ℓ loops (see e.g. appendix A). For comparison, the degree of the Baikov polynomial is $\deg B = \min(\ell + e, 2\ell)$, where e is the number of independent external momenta, hence $\deg B \geq \deg G$ for $e \geq 1$. This makes this method particularly efficient in some contexts, as already observed in the literature (see e.g. [11]). Its main drawback is that, in the presence of ISPs, the number of independent annihilators is comparatively large and it is thus less convenient than other representations.

JHEP10(2025)018

9.2 Schwinger representation

The Schwinger representation of loop integrals is related to twisted Mellin moments with twist

$$u(\mathbf{z}) = \exp\left[-\mathcal{F}(\mathbf{z})/\mathcal{U}(\mathbf{z})\right]\mathcal{U}(\mathbf{z})^{-d/2} \tag{9.9}$$

and the integral I_α corresponds to the loop integral in eq. (6.1) up to a sign

$$J_\alpha = (-1)^{|\alpha|}\, I_\alpha. \tag{9.10}$$

As before, d is the space-time dimension, while $\mathcal{U}$ and $\mathcal{F}$ are the Symanzik polynomials.

The Schwinger representation shares many properties with the Lee-Pomeransky representation, including the fact that it consistently works without the need of ISPs — although, in both cases, they can be included when needed. Moreover, we empirically find that annihilators generally have lower degree in this representation, as compared to Lee-Pomeransky. When a full set of ISPs is used, for instance, first-order annihilators in the Schwinger representation all have degree one. However, due to a more complex twist, which also includes an exponential factor, finding such annihilators can sometimes be harder, depending on the specific problem. Moreover, we empirically find that we need to use second-order annihilators in this representation more often than in others, which can significantly add complexity to a calculation. On the other hand, template identities are generally fewer and simpler in this representation, compared to Lee-Pomeransky or others, hence they often yield simpler systems of equations and more efficient reductions to master integrals.

9.3 Examples

The methods described in this work have been checked in several non-trivial examples using both the Schwinger and the Lee-Pomeransky parametrization. We tested reductions both with and without including a full set of ISPs in the definition of the integral families. The relations we derive agree with those obtained with the Laporta method in momentum space.

In particular, by using annihilators obtained by defining, for each sector, a family of twisted Mellin moments with only its proper denominators and no ISP, one can efficiently obtain reductions with very high powers of denominators [11]. These can be, in turn, useful because some of these integrals have good properties such as (quasi-)finiteness [12, 13] or uniform trascendental weight [14].

Several non-trivial examples have been checked via a preliminary interface between the CALICO package (presented in this work) and the private FFIntRed package for integral reduction, which we plan to share in a future work. One of them is the reduction of all integrals with trivial numerators (i.e. $s = 1$, defined in section 6) and up to 3 dots ($u = 3$) for the integral family in figure 4. The latter contributes to top-pair plus Higgs production at hadron colliders at two loops accuracy. We computed the annihilators numerically and used them for generating a linear system solving the integral relations, combined with symmetry relations found with the FFIntRed package. We tested this, on a modern laptop, using both the Lee-Pomeransky and the Schwinger parametrization. For the latter, computing the annihilators and figuring out the correct seeding — which turned out had to include seeds with $s = 1$ — was more complicated, also due to the need of using some second-order

JHEP10(2025)018

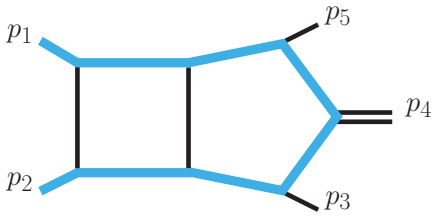


Figure 4. Pentabox contributing to top-pair plus Higgs production.

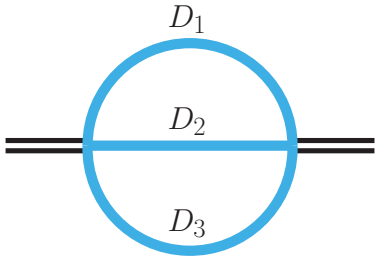


Figure 5. Two-loop sunrise with different internal masses.

annihilators. However, after the system had been generated and independent equations had been filtered out by the FiniteFlow solver, this method combined with the Schwinger parametrization yielded an extremely efficient system of equations. This contained about 20'000 equations and could be solved numerically in a few hundredths of a second on a laptop, which is orders of magnitude more efficient than what we were able to obtain using other approaches or integral representations.

While we leave a detailed description of complex examples of this kind to future works, the core ideas behind this approach are also illustrated in the two simple applications we discuss in detail below. An implementation of them can be found in the public repository of the CALICO program.

The two-loop sunrise with different masses. As a first simple example, consider the two-loop sunrise with different masses (depicted in figure 5), i.e. the family of loop integrals defined by the denominators

$$\begin{aligned} D_1 &= k_1^2 - m_1^2 \\ D_2 &= k_2^2 - m_2^2 \\ D_3 &= (k_1 + k_2 - p)^2 - m_3^2. \end{aligned} \tag{9.11}$$

These integrals depend on four invariants, namely the three squared masses m_j^2 and

$$s \equiv p^2. \tag{9.12}$$

We immediately observe that, in order to produce integral identities via IBPs in momentum space for this family, two ISPs are needed. However, using annihilators in the Lee-Pomeranski or Schwinger parametrization of loop integrals, ISPs are not needed.

JHEP10(2025)018

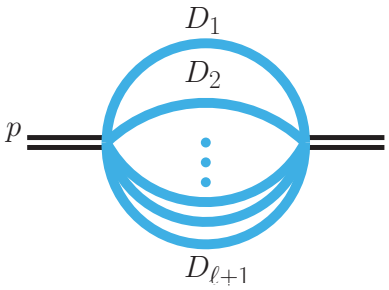


Figure 6. ℓ-loop banana graph.

Using the Lee-Pomeranski representation, we find 3 generators for first-order annihilators of degree 2 and 6 of degree 3. Using the Schwinger representation, we find instead 5 first-order generators, as well as one second-order generator, all of which have degree 2. This second-order annihilator is needed to obtain a complete reduction to master integrals in the Schwinger representation, which would yield one additional master integral otherwise. As explained, each generator yields a template equation for the integral family, using eq. (2.11).

We further observe that this family has one top sector and 3 non-zero subsectors. The proper denominators of these subsectors factorize into products of one-loop vacuum integrals (a.k.a. tadpole integrals). In principle, we can limit ourselves to using the same template identities defined above, considering seeds with zero or negative exponents as well. However, for the subsectors, we also consider identities obtained from annihilators of one-loop vacuum integrals identified by the denominator

$$D_0 = k^2 - m^2 \tag{9.13}$$

with m replaced by any of the m_j. For this, we obtain one first-order generator, of degree 2 for Lee-Pomeranski and degree 1 for Schwinger. After replacing $m \to m_j$ with $j = 1, 2, 3$ we thus obtain 3 more template identities valid for subsectors. Of course, this is of no importance in this simple example, but the same principle applied to more complex examples may substantially decrease the complexity of a reduction.

For all sectors we use seed integrals with trivial numerators and find 7 MIs in the whole family, which can be chosen as

$$\{J_{112}, J_{121}, J_{211}, J_{111}, J_{011}, J_{101}, J_{110}\}. \tag{9.14}$$

Differential operators $\hat{O}_s$ and $\hat{O}_{m_j^2}$ are found following section 3, which we use to find DEs satisfied by the MIs. These have been checked against a calculation that uses the Laporta algorithm.

Equal-mass ℓ-loop banana integrals. The ℓ-loop banana integral family with internal mass m is defined by the set of denominators

$$\begin{aligned} D_j &= k_j^2 - m^2 \quad \text{for } j = 1, \ldots, \ell \\ D_{\ell+1} &= (k_1 + \cdots + k_\ell - p)^2 - m^2. \end{aligned} \tag{9.15}$$

This family depends on two invariants, namely m^2 and s, with the latter defined as in eq. (9.12). These integrals are of great interest due to their analytic structure. For $\ell \geq 2$, they are indeed among the simplest non-factorizable ℓ-loop integrals which cannot be expressed in terms of generalized polylogarithms.

Despite their apparent algebraic simplicity, deriving differential equations for these integrals can actually become challenging as the number of loops ℓ grows, when using IBPs in momentum space. This is due to the need of adding a large number of ISPs (namely $(\ell+2)(\ell-1)/2$), which grows quadratically in ℓ, to have a complete set of generalized denominators. As an example, at six loops we would need 20 ISPs. The representations of loop integrals reviewed in this section, however, do not require introducing ISPs and can work just the $\ell+1$ denominators in eq. (9.15), drastically simplifying the problem. We tested this examples for several choices of the loop order ℓ, finding the general patterns which we describe below. We recall that alternative efficient algorithms for computing the Picard-Fuchs operator of ℓ-loop banana graphs exist (see e.g. [45–47]) but the one we use here is much more general and not limited to this integral family.

In the Lee-Pomeransky representation we find first-order annihilators up to degree $\ell+1$. In the Schwinger parametrization we find first-order annihilators up to degree ℓ, as well as one second-order annihilator of degree 2 which is required for a complete reduction. Besides the top sector, we have $\ell+1$ subsectors, whose integrals with trivial (i.e. constant) numerators factorize as products of ℓ one-loop integrals with one denominator as in eq. (9.13).

This integral family also has a clear symmetry. Integrals $\mathcal{I}_\alpha$ are symmetric under any permutation of the indexes $\alpha_1, \ldots, \alpha_n$ (where here $n = \ell+1$). We can simply implement this by sorting the indexes identifying an integral, say, from higher to lower. By doing this, all the integrals are mapped to the two sectors $S_{1\cdots1}$ and $S_{1\cdots10}$.

By combining these symmetries with identities generated using annihilators and seeds with trivial numerator, we find $\ell+1$ master integrals, that can be chosen as

$$\{\mathcal{I}_{2\cdots211}, \mathcal{I}_{2\cdots2111}, \ldots, \mathcal{I}_{21\cdots1}, \mathcal{I}_{1\cdots1}, \mathcal{I}_{1\cdots10}\}. \tag{9.16}$$

Differential operators $\hat{O}_s$ and $\hat{O}_{m^2}$ are also found as first-order operators with maximum degree 2 (or 1 for $\ell = 1$). Combined with the identities above, we thus easily find DEs satisfied by the MIs. We successfully checked these against a calculation that uses the Laporta algorithm in momentum space up to $\ell = 5$ loops, although the approach described here is orders of magnitude more efficient, since it does not need ISPs. Indeed, even at $\ell = 6$ loops we are able to find annihilators and template identities in a couple of minutes on a modern laptop, after which deriving the DEs (including the generation of the system of equations, the derivative operators and the analytic reconstruction of the matrices) takes no more than a few seconds.

Although this is a relatively simple example, it shows that for specific applications the approach of annihilators can outperform state-of-art alternatives, as already seen in the literature (see e.g. [11]).

JHEP10(2025)018

10 The CALICO Mathematica package

In this section we give a general description of the CALICO program and its features. The package is open source and it is available at

https://github.com/fontana-g/calico

where several examples with detailed comments can also be found. Here we describe the core features of the package and we refer to the builtin examples, and comments therein, for more details about using it in non-trivial applications. These include most of the examples illustrated in the previous sections.

The package depends on the FiniteFlow [27] program,[7] which is used for solving linear identities constraining the coefficients of annihilators and other differential operators (see also section 4.5).

We load the package with

```
<<CALICO
```

All public symbols exported by the package are prefixed by `CAT` (as in "Computing AnnihilaTors") to avoid conflicts with other packages or builtin symbols.

The most important function of the package is arguably `CATAnnihilator`, which is used as follows

```
ann = CATAnnihilator[u(z),{z_1,...,z_n},d_max,o_max];
```

where $u(\mathbf{z})$ is the twist, while $d_{\max}$ and $o_{\max}$ are the maximum degree and order of the generators of the annihilators to be found. The output of the function contains the coefficients $c_{j_1 j_2 \ldots}(\mathbf{z})$ which define the annihilators as in eq. (2.8). Its exact form is reported later, but it is unlikely to be important for most users, since we already provide functions which use the output `ann` to produce integral identities for many types of integral families. However, it is worth mentioning that

```
Length[ann[[o,d+1]]]
```

returns the number of generators of order o and degree d which have been found. The annihilators returned by this function are, by default, cast as polynomials in both $\mathbf{z}$ and in the free parameters appearing in the twist. This yields integral identities that are easier to manipulate, but in principle annihilators only need to be polynomials in $\mathbf{z}$. Casting them in the default form however requires additional algebraic manipulations, such as computing the least common multiple of polynomials in the free parameters. The option `"PolynomialInParameters"->False` can be specified, saving some algebraic manipulations, if annihilators are not needed to be cast in this form. We also clarify that CALICO does

[7] CALICO's performance can benefit from new features of an upcoming release of FiniteFlow which, at the time of writing, are available in the experimental branch `exp` of the public FiniteFlow repository.

JHEP10(2025)018

not provide a way of choosing $d_{\max}$ and $o_{\max}$. In most applications, one may wish to adjust these values either until no more annihilators are found beyond a certain o and d (see also the previous code block), or until one can check via other means that the generators that have been found are sufficient to solve a certain problem.

Sometimes, one may need to compute annihilators in multiple steps. As an example, after computing `ann` as before, we may wish to increase $d_{\max}$ or $o_{\max}$ to seek additional solutions, without recomputing again the generators that are already in `ann`. This can be achieved using

```
annextra = CATAnnihilator[u(z),{z1,...,zn},d'max,o'max,
                          "KnownSolutions"->ann];
```

after which `annextra` will contain generators up to degree $d'_{\max}$ and order $o'_{\max}$ that are independent of those in `ann`. A similar call can also be used to exclude solutions generated by operators `ann` that are known or defined by any other mean. The options `"MinDegree"->`$d_{\min}$ and `"MinOrder"->`$o_{\min}$ are available to specify the minimum degree and order at which the search for a solution is started (by default they are 0 and 1 respectively). The two solutions can thus be merged using

```
ann = CATAnnihilatorMerge[ann,annextra];
```

after which `ann` will include generators previously contained either `ann` or `annextra`.

The other main function of the package is `CATDiffOperator`, which is used to compute differential operators in the integration variables that are equivalent to differentiation of the twist with respect to an external free parameter. If $x_1, \ldots, x_m$ are external parameters, then

```
diffop = CATDiffOperator[u(z),{z1,...,zn},{x1,...,xm},dmax,omax,
                         "KnownAnnihilatorSolutions"->ann];
```

returns a list of m elements having, at position i, analytic data about the differential operator $\hat{O}_{x_i}$ defined as in eq. (3.8), or `CATImpossible` if no such operator was found (in such case one may wish to try again with higher values of $d_{\max}$ or $o_{\max}$). The inputs other than x_i have the same meaning as before. The option `"KnownAnnihilatorSolutions"` in the second line is not required but it can improve performance, since CALICO will use known annihilators `ann` of the twist $u(\mathbf{z})$ to trim the ansatz for the differential operators.

Both functions we just described use the functional reconstruction techniques of FiniteFlow to reconstruct analytic results from numerical evaluations over finite fields. The user can specify the maximum total degree (in the free parameters, including the regulators) and the maximum number of prime fields to use for the reconstruction via the options `"MaxRecDegree"` and `"MaxRecPrimes"` respectively. Similarly, the option `"NThreads"` can be specified to control the number of threads used in multivariate reconstructions. The option `"Substitutions"` takes instead a list of substitutions in the free parameters that is performed right before solving the syzygy or polynomial decomposition equations. It may be used e.g. to set one dimensional variable to one (later recovering its dependence via

JHEP10(2025)018

dimensional analysis), which improves performance of reconstruction algorithms, or to test the calculation for rational numerical values of the free parameters.

The calculation of annihilators and differential operators might take a substantial amount of time for very complex applications. CALICO can optionally print information about the ongoing calculation. This is done when the verbosity is set to `True`, which is achieved calling

```
CATVerbose[True];
```

before the call to these functions.

The twist $u(\mathbf{z})$ taken as input by the functions above can be passed just by using its *analytic expression*, but CALICO has many functions for building twists for a wide variety of integral families. These cast the twist into a form that is optimized for the analytic manipulations performed internally by these functions, as briefly mentioned in section 4.5. The wrapper function `CATTwist` is used to signal this optimized form to CALICO. Users can also build twists in such a form, as described later when more advanced use cases are discussed.

We will now give an overview of basic functions that can be used to

- define the twist $u(\mathbf{z})$ and the polynomials it depends on;
- convert annihilators into template equations, for various forms of φ_α;
- generate lists of seeds, to which the template equations can be applied;
- convert differential operators into derivatives of integrals, as functions of their exponents, for various forms of φ_α.

We first review the functions we provide for building polynomials and twists related to various integral families.

For the Baikov parametrization

```
{baikov, zs} = CATBaikovPoly[
  {{l1, m1^2}, ..., {ln, mn^2}}, (* generalized denominators *)
  {k1, ..., kℓ}, (* loop momenta *)
  {p1, ..., pe}, (* linearly independent external momenta *)
  replacements,
  z
]
```

sets `baikov` to the Baikov polynomial and `zs` to the list of integration variables

$$\texttt{\{z[1],...,z[}n\texttt{]\}}.$$

The pairs $\{l_j, m_j^2\}$ identify the generalized denominators in momentum space, as in eq. (6.2), with l_j being linear combinations of the ℓ loop momenta k_i and the e independent external momenta p_i (excluding the one fixed by momentum conservation which must not appear in the l_j). Bilinear generalized denominators, as in eq. (6.3), may be specified using $\{l_j, v_j, m_j^2\}$ instead. The input `replacements` should be a list of rules of the form $p_i p_j \to \ldots$ converting

JHEP10(2025)018

scalar products (specified as normal products in the rules) of external momenta into invariants. The corresponding twist is obtained using[8]

```
CATBaikov[baikov, d, ℓ, e, zs]
```

with d being the number of space-time dimensions (typically a symbol d or $4-2\epsilon$), ℓ the number of loops and e the number of independent external momenta. Similarly, the twist for dual loop integrals in the Baikov representation is obtained with

```
CATBaikovDual[baikov, d, ℓ, e, zs]
```

whose inputs have the same meaning.

For the integral parametrizations discussed in section 9, we need the Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$, as well as $G = \mathcal{U} + \mathcal{F}$. In CALICO we compute them using

```
{u, f, g, zs} = CATUFGPolys[
  {{l1, m1^2}, ..., {ln, mn^2}},
  {k1, ..., kℓ},
  replacements,
  z
];
```

where the inputs have the same meaning as above. This sets the symbols `u`, `f` and `g` to $\mathcal{U}(\mathbf{z})$, $\mathcal{F}(\mathbf{z})$ and $G(\mathbf{z})$ respectively and assigns the list of integration variables to `zs`, as before. The twist is returned by the call

```
CATLP[g, d, zs]
```

for the Lee-Pomeransky representation and

```
CATSchwinger[u, f, d, zs]
```

for the Schwinger representation.

Another function, that can define the twist in a wide variety of cases, is

```
CATMultiBaikov[{B1(z), B2(z), ...}, {γ1, γ2, ...}, z];
```

to define a twist with the form in eq. (2.5) or

```
CATMultiBaikov[{B1(z), B2(z), ...}, {γ1, γ2, ...}, F(z), z]
```

[8]Note that `CATBaikov` only depends on the Baikov polynomial, hence it does not include the external prefactor, which depends on the kinematic invariants and is needed to derive the differential operator $\hat{O}_x$. For this purpose, one should use `CATMultiBaikov` instead, including all kinematic-dependent factors.

JHEP10(2025)018

to define a twist with the form in eq. (2.6) if $F(\mathbf{z})$ is a polynomial. This form covers, in principle, all cases illustrated in this paper except for Schwinger parametrization, although we generally prefer dedicated routines for most of them. `CATMultiBaikov` is however the function we use for hypergeometric functions and the loop-by-loop Baikov parametrization. For the latter, CALICO does not provide functions for computing the polynomials $B_j(\mathbf{z})$ and the exponents γ_j but this functionality is already available in the public BaikovPackage [48]. In the builtin example that uses this parametrization, we explicitly show how the input for CALICO can be generated using BaikovPackage.

Once the twist has been computed, it can be used to compute parametric annihilators and differential operators $\hat{O}_x$ as already discussed above. We thus typically need to turn these into integral identities. This translation effectively implements equations (2.11) and (3.9), which depend on the specific form of the integrand φ_α. CALICO has several functions which perform this translation for different forms of φ_α. For this purpose, the package uses the symbolic function

```
CATInt[F,{α1,...,αn}]
```

to represent either φ_α or the corresponding integral I_α, identified by the multi-index of exponents $\alpha = (\alpha_1, \ldots, \alpha_n)$. The first argument F can be any symbol or expression (even a string) and it is used to identify the integral family, so that integrals belonging to more than one family can co-exist in an expression. The function `CATInt` also automatically performs the substitutions[9]

$$\texttt{CATInt[}F\texttt{,}\alpha\texttt{]}\ \texttt{CATInt[}F\texttt{,}\beta\texttt{]} \to \texttt{CATInt[}F\texttt{,}\alpha+\beta\texttt{]}$$
$$\texttt{CATInt[}F\texttt{,}\alpha\texttt{]}^n \to \texttt{CATInt[}F\texttt{,}n\alpha\texttt{]}$$

that are consistent with α_j being exponents of rational factors of the integrands φ_α. In the following, we describe CALICO's functions that convert annihilators `ann` and differential operators $\hat{O}_x$, denoted as `diffop` as before, into integral identities of a family F. Moreover, $\mathbf{z}$ corresponds to the list of integration variables, while $\mathbf{x} = \{x_1, \ldots, x_m\}$ is a list of free parameters for which we need differential operators $\hat{O}_{x_j}$.

The simplest case is when φ_α is just a (Laurent) monomial

$$\varphi_\alpha(\mathbf{z}) = \mathbf{z}^\alpha. \tag{10.1}$$

In such case, template integral identities can be obtained from annihilators using

```
tmpids = CATMonIdsFromAnnihilators[F,ann,z];
```

The returned value `tmpids` is an anonymous function [49] of the exponents such that, for each seed α

[9] After these substitutions are performed, prefactors might need to be manually adjusted, depending on the explicit form of φ_α (see e.g. eq. (8.4) and (9.1)). This is only relevant when users implement their own identities, since CALICO already takes care of adjusting prefactors in the template identities it generates.

JHEP10(2025)018

```
tmpids[α1,...,αn]
```

returns a list of linear combinations of integrals which vanish, namely the l.h.s. of eq. (2.11) for each generator. Similarly, derivatives of integrals can be generated with

```
deriv = CATMonIdsFromDiffOperators[F, diffop, z];
```

whose return value `deriv` is also an anonymous function of the exponents. For each seed α

$$\texttt{deriv[}\alpha_1\texttt{,...,}\alpha_n\texttt{]} = \{\partial_{x_1}\mathcal{I}_\alpha, \ldots, \partial_{x_m}\mathcal{I}_\alpha\},$$

with each entry obtained as the r.h.s. of eq. (3.9).

Completely analogous functions are available for different forms of φ_α. Since they have the same behaviour and take the same inputs as the ones we just described, we simply list them here.

- For inverses of (Laurent) monomials

$$\varphi_\alpha(\mathbf{z}) = \mathbf{z}^{-\alpha}, \tag{10.2}$$

  used e.g. in the Baikov representation, we have

```
tmpids = CATInvMonIdsFromAnnihilators[...];
deriv  = CATInvMonIdsFromDiffOperators[...];
```

- For *duals* of loop integrals, with φ_α defined as in section 8, i.e. implementing eq. (8.6) and (8.7), we need to first specify which denominators are regulated (i.e. *can* appear with positive exponent) using

```
CATDualRegulated[F] = {σ1,...,σn};
```

  where $\sigma_j = 1$ ($\sigma_j = 0$) if the j-th denominator is regulated (not regulated). Then, the identities are found with

```
tmpids = CATDualInvMonIdsFromAnnihilators[...];
deriv  = CATDualInvMonIdsFromDiffOperators[...];
```

- For Mellin moments with φ_α defined as in eq. (9.1) we have

```
tmpids = CATMellinIdsFromAnnihilators[...];
deriv  = CATMellinIdsFromDiffOperators[...];
```


JHEP10(2025)018

The Mellin integrals defined in eq. (9.1), while closely related to loop integrals, generally differ from them by an α-dependent prefactor. It is generally convenient to convert the template identities and rewrite them directly in terms of loop integrals. We achieve this in CALICO by using, instead of the code snippet of the previous block,

```
tmpids = CATSchwingerIdsFromAnnihilators[F,ann,z];
deriv  = CATSchwingerIdsFromDiffOperators[F,diffop,z];
```

for the Schwinger representation and

```
tmpids = CATLPIdsFromAnnihilators[F,ann,z,ℓ,d];
deriv  = CATLPIdsFromDiffOperators[F,diffop,z,ℓ,d];
```

for the Lee-Pomeransky representation. Note that, for the latter, the two functions also take, as input, the number of loops ℓ and space-time dimensions d, since the prefactors to be adjusted depend on them.

Once template identities are available, we typically want to apply them to a list of seeds to generate systems of identities to be solved, as explained below eq. (2.11). Generating seeds and the corresponding system is up to the user. A typical strategy involves generating, for all sectors S_β (with $\beta_j = 0, 1$, defined as in section 6) all seeds up to a certain numerator degree s (or rank) and a certain number of dots u. CALICO can help to obtain this list via

```
CATGenerateSeeds[β,{u_min,u_max},{s_min,s_max}]
```

which generates all seeds α such that $I_\alpha \in S_\beta$ and $u_{\min} \leq u \leq u_{\max}$, $s_{\min} \leq s \leq s_{\max}$.

Once the system of identities is generated from the template identities, we need to choose an ordering, or a weight (see section 2.2), in order to solve it. It is generally useful to list all integrals appearing in an expression (for instance, a system of equations), sorted by weight. This is achieved with

```
CATIntCases[expression,weight]
```

which returns a list of integrals, identified by the symbolic function `CATInt`, sorted by the specified weight as if using

```
SortBy["list of integrals in the expression",weight]
```

with `weight` being a suitable function. CALICO includes several implementations for the function `weight` tailored to loop integrals. Here we list them with the most important criteria they use:

- `CATIntMellinDefaultWeight` eliminates numerators, i.e. negative exponents, of loop integrals, and it is more conveniently used with the representations of loop integrals as twisted Mellin transforms, described in section 9;

JHEP10(2025)018

- `CATIntInvMonDefaultWeight` is conveniently used with Baikov representations (see section 7.1) and prefers integrals with lower u, or lower t as a tie-breaker;

- `CATIntDualInvMonDefaultWeight` is conveniently used with duals of loop integrals in Baikov representations (see section 8) and still prefers integrals with lower u, but prefers higher values of t as a tie-breaker.

These functions sort higher-weight integrals on the left, as required when using the linear solver of FiniteFlow — although any other solver or definition of weight can be used.

As already stated, the integrals in a family may also obey additional relations, such as symmetry relations, which cannot be found by the methods described here and thus have to be found and implemented by the user.

We refer to the builtin examples for a description of how to combine all these ingredients in several applications.

Finally, CALICO also exposes functions for solving syzygy equations, since this can be useful for a much broader set of problems, both within and outside physics. Generators of syzygy solutions for eq. (4.1) up to a maximum degree $d_{\max}$ can be obtained using

$$\mathrm{g} \;=\; \mathrm{CATSyz}\left[\left\{f_1(\mathbf{z}),\ldots,f_k(\mathbf{z})\right\},\left\{z_1,\ldots,z_n\right\},d_{\max}\right]$$

which can also take most of the options of `CATAnnihilator`. The returned value `g` is a list which contains, at position $d+1$, all the generators of degree d that have been found. Each generator is in turn a list of polynomials $\mathbf{g}(\mathbf{z})$ that satisfy the syzygy equation.

Similarly, a solution $\mathbf{g}(\mathbf{z})$ for the polynomial decomposition in eq. (4.12) can be obtained using

$$\mathrm{g} \;=\; \mathrm{CATPolyDec}\left[\left\{f_1(\mathbf{z}),\ldots,f_k(\mathbf{z})\right\},\left\{h_1(\mathbf{z}),\ldots,h_m(\mathbf{z})\right\},\left\{z_1,\ldots,z_n\right\},d_{\max}\right]$$

which returns a list of length m, whose i-th entry is a solution of eq. (4.12) for $h(\mathbf{z}) = h_i(\mathbf{z})$, or `CATImpossible` if no such solution was found.

Advanced usage. Users who wish to implement custom identities that are not already supported by the package may need to understand the form of the output of the function `CATAnnihilator`. The format is

$$\begin{Bmatrix}
\left\{A_{10},A_{11},\ldots,A_{1d_{\max}}\right\},\\
\left\{A_{20},A_{21},\ldots,A_{1d_{\max}}\right\},\\
\ldots,\\
\left\{A_{o_{\max}0},A_{o_{\max}1},\ldots,A_{o_{\max}d_{\max}}\right\}
\end{Bmatrix}$$

where A_{od} is the *list* of generators of order o and degree d which have been found. Each element $A_{od}[[k]]$ of A_{od} is itself a list containing the coefficients $c_{j_1 j_2 \ldots}(\mathbf{z})$ which define the annihilator

JHEP10(2025)018

as in eq. (2.8). More precisely, the i-th element of the list $A_{od}[[k]]$ is the polynomial which multiplies the differential operator

$$\prod_{j=1}^{n} (\partial_j)^{a_{ij}} \tag{10.3}$$

where $\{a_{i1}, \ldots, a_{in}\}$ is the i-th element of the list returned by

```
Join@@Table[CATExponentList[o,n],{o,0,o_max}]
```

Similarly, more advanced use cases may require a better understanding of the output of `CATDiffOperator`. As already stated, this is a list of length m, where m is the number of free parameters which we need differential operators for. Each entry of this list corresponds to the solution for the i-th parameter and contains either the symbol `CATImpossible`, to signal no solution was found, or a rule of the form

```
o -> {c_0^(x)(z), c_1^(x)(z), ...}
```

where o is the order of the operator that has been found and the list on the right contains the coefficients $c^{(x)}_{j_1 j_2 \ldots}(\mathbf{z})$ that define the operator as in eq. (3.7). In particular, the i-th element of such a list is the polynomial that multiplies the differential operator defined as in eq. (10.3).

As already stated, the twist $u(\mathbf{z})$ taken as input by the functions above can be passed just by using its analytic expression. It is, however, sometimes convenient to use symbolic polynomials, inside the twist, in a first stage of the analytic preparation performed by such functions, then substitute their analytic expressions only at a later stage. For this purpose, one can specify the twist in the following form

```
CATTwist[u(z,x),{B_1->..., B_2->..., ...}]
```

where $\mathbf{z}$ are the integration variables and $\mathbf{x}$ are the list of free parameters, while $u(\mathbf{z}, \mathbf{x})$ is an expression for the twist where all polynomials that define it are replaced by *symbolic functions* of the form $B_i(\mathbf{z}, \mathbf{x})$. The second argument is a replacement rule or a list of them that is used to replace the polynomial functions B_i with their explicit expression. Note that this replacement will be done on expressions that depend on both the polynomials $B_i(\mathbf{z}, \mathbf{x})$ and their (first or higher-order) derivatives with respect to z_j — and also with respect to x_j when computing operators $\hat{O}_{x_j}$. Hence, the replacement should involve the *heads* B_i rather than the functions evaluated at specific arguments $(\mathbf{z}, \mathbf{x})$. All twists computed by CALICO's functions are in this form, hence we suggest examining any of those for explicit examples.

11 Conclusions and outlook

In this paper we reviewed and expanded on the method of parametric annihilators for finding linear relations among functions having a suitable integral representation. We illustrated this approach in a way that applies to a broad class of integral families. These include loop integrals

JHEP10(2025)018

in several parametric representations. We applied it to examples involving hypergeometric functions, loop integrals in the Lee-Pomeransky or Baikov representation (for which this or similar approaches had already been formulated [8, 10]), as well as to the loop-by-loop Baikov and Schwinger representations which, until now, had not been systematically used for the purpose of integral reduction. Building on similar principles, we illustrated a method to derive differential equations for the independent master integrals. We also showed that our formulation applies to duals of loop integrals, which play a crucial role in intersection theory.

We described algorithms for computing annihilators and the differential operators needed for differential equations which exploit modern sparse linear solvers and finite-field techniques. We released an implementation of it, in the public Mathematica package CALICO. At the time of writing, the CALICO package can be useful for theoretical studies involving various representations of loop integrals and special functions.

In the future, on top of implementing additional optimizations, we plan to release an interface between it and the FFIntRed package (also to be published) for the systematic generation of identities needed for the reduction of loop integrals contributing to a process. CALICO numerically computes and analytically reconstructs the analytic dependence of differential operators on external parameters using nodes of FiniteFlow's computational graphs. In the future, we plan to add an interface which enables using these nodes in a custom graph to generate identities. This would sidestep the need to reconstruct complex template identities for multiscale processes, while still benefitting from this method for computing linear relations.

We believe the techniques and applications illustrated in this paper, as we as the CALICO package we released, could become highly beneficial to numerous future applications, including studies that rely on the strengths of several representations of loop integrals.

Acknowledgments

We thank Vsevolod Chestnov, Thomas Gehrmann, Pierpaolo Mastrolia, Kay Schönwald, Simone Zoia and Lorenzo Tancredi for feedback on our work and on the draft. The work of GF received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement 101019620 (ERC Advanced Grant TOPUP) and from the UZH Candoc scheme (Candoc grant Nifty LooPS). The work of TP received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme grant agreement 101040760 (ERC Starting Grant FFHiggsTop).

A Representations of loop integrals

In this appendix, we review the definition of the polynomials appearing in the twist $u(\mathbf{z})$ of the representations of loop integrals used in the paper. In the following, we consider an $\ell-$loop Feynman integral with $e+1$ external legs — out of which only e are independent because of momentum conservation. Loop integrals are defined as in eq. (6.1), with n generalized denominators with the form in equations (6.2) or (6.3). In the following $\mathbf{z} = (z_1, \ldots, z_n)$

JHEP10(2025)018

is the list of integration variables that these polynomials depend on, in the parametric representations we consider.

Baikov polynomial. The Baikov representation results from a change of variables from the d-dimensional loop momenta to the generalized denominators. By inverting equations (6.2) and (6.3), we rewrite every scalar product of the form $k_i \cdot k_j$ or $k_i \cdot p_j$ as a linear combination of generalized denominators

$$\begin{aligned} k_i \cdot k_j &= d_{ij0} + \sum_{m=1}^{n} d_{ijm}\, D_m \\ k_i \cdot p_j &= e_{ij0} + \sum_{m=1}^{n} e_{ijm}\, D_m, \end{aligned} \tag{A.1}$$

where d_{ijm} and e_{ijm} are functions of external invariants. This inversion requires a full set of n generalized denominators, with n given in eq. (6.10). The Baikov polynomial is a Gram determinant. We recall that, given a list of vectors $\{v_j\}_{j=1}^{m}$, their Gram determinant is defined as

$$\mathrm{Gram}(v_1, \ldots, v_m) = \det V, \qquad \text{with } V_{ij} \equiv 2\, v_i \cdot v_j. \tag{A.2}$$

The Baikov polynomial $B(\mathbf{z})$ is given by the Gram determinant of the loop momenta k_j and external momenta p_j, after rewriting each scalar product involving loop momenta as a linear combination of generalized denominators (as in eq. (A.1)) and replacing each denominator D_j with the variable z_j. More explicitly,

$$B(\mathbf{z}) = \mathrm{Gram}(k_1, \ldots k_\ell, p_1, \ldots, p_e)\Big|_{\substack{k_i\cdot k_j \,\to\, d_{ij0} + \sum_{m=1}^{n} d_{ijm}\, z_m \\ k_i\cdot p_j \,\to\, e_{ij0} + \sum_{m=1}^{n} e_{ijm}\, z_m}}.$$

This polynomial enters the Baikov representation in equations (7.1) and (7.2).

We also add that the proportionality factor K in eq. (7.1) is proportional to another Gram determinant, namely

$$K = C(d)\, \mathrm{Gram}(p_1, \ldots, p_e)^{\frac{-d+e+1}{2}}. \tag{A.3}$$

In the standard Baikov representation, this prefactor is irrelevant in linear integral identities, since it is the same for all integrals within a family. In the loop-by-loop Baikov representation [6] — which consists in applying the Baikov parametrization one loop at the time — the external legs of a subloop generally depend on the other loop momenta. The Gram determinant in the last relation thus contributes to the twist in this case, except for the one corresponding to the last loop integration that is rewritten in this parametric form. Hence, in the loop-by-loop Baikov representation, we obtain a twist $u(\mathbf{z})$ with the form in eq. (2.5) with $2\ell - 1$ polynomials $B_j(\mathbf{z})$ and exponents γ_j, where the latter are linear functions of d. The external prefactor, which we wrote as $B_0^{\gamma_0}$ in the example in eq. (7.7), must however be included when finding DEs, e.g. via differential operators $\hat{O}_x$ as described in section 3, since it depends on the external invariants x.

JHEP10(2025)018

Symanzik polynomials. The first and second Symanzik polynomials, respectively denoted as $\mathcal{U}(\mathbf{z})$ and $\mathcal{F}(\mathbf{z})$, appear in the twists of both Lee-Pomeranski (9.4) and Schwinger (9.9) representations. They also appear in the well-known Feynman parametrization. We briefly recall how they can be computed algebraically. Given the structure of the generalized denominators D_j, the sum of the denominators D_j weighted by the variables z_j has the following dependence on the loop momenta,

$$\sum_{j=1}^{n} z_j D_j = \sum_{i,j=1}^{n} A_{ij} (k_i \cdot k_j) + 2 \sum_{j=1}^{\ell} (B_j \cdot k_j) + C, \tag{A.4}$$

which defines the symmetric matrix A_{ij}, the Lorentz vectors B_j^μ and the scalar C, all of which have a linear dependence on $\mathbf{z}$. The vector B_j^μ and the scalar C also depend on the external kinematics. The polynomials $\mathcal{U}(\mathbf{z})$ and $\mathcal{F}(\mathbf{z})$ are thus obtained as

$$\mathcal{U}(\mathbf{z}) = \det A, \tag{A.5}$$

$$\mathcal{F}(\mathbf{z}) = \mathcal{U}(\mathbf{z}) \left(\sum_{i,j=1}^{n} A_{ij}^{-1} (B_i \cdot B_j) - C \right), \tag{A.6}$$

with A^{-1} being the inverse of the matrix A. Note that $\det A \times A^{-1}$ is a polynomial function of A_{ij}, hence $\mathcal{F}(\mathbf{z})$ is also polynomial in $\mathbf{z}$.

Data Availability Statement. This article has no associated data or the data will not be deposited.

Code Availability Statement. This article has associated code in a code repository.

References

[1] F.V. Tkachov, *A theorem on analytical calculability of 4-loop renormalization group functions*, *Phys. Lett. B* **100** (1981) 65 [INSPIRE].

[2] K.G. Chetyrkin and F.V. Tkachov, *Integration by parts: the algorithm to calculate β-functions in 4 loops*, *Nucl. Phys. B* **192** (1981) 159 [INSPIRE].

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

[4] T. Gehrmann and E. Remiddi, *Differential equations for two-loop four-point functions*, *Nucl. Phys. B* **580** (2000) 485 [hep-ph/9912329] [INSPIRE].

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

[6] H. Frellesvig and C.G. Papadopoulos, *Cuts of Feynman integrals in Baikov representation*, *JHEP* **04** (2017) 083 [arXiv:1701.07356] [INSPIRE].

JHEP10(2025)018

[7] R.N. Lee and A.A. Pomeransky, *Critical points and number of master integrals*, *JHEP* **11** (2013) 165 [arXiv:1308.6676] [INSPIRE].

[8] K.J. Larsen and Y. Zhang, *Integration-by-parts reductions from unitarity cuts and algebraic geometry*, *Phys. Rev. D* **93** (2016) 041701 [arXiv:1511.01071] [INSPIRE].

[9] R.N. Lee, *Modern techniques of multiloop calculations*, in the proceedings of the *49th Rencontres de Moriond on QCD and high energy interactions*, (2014) [arXiv:1405.5616] [INSPIRE].

[10] T. Bitoun, C. Bogner, R.P. Klausen and E. Panzer, *Feynman integral relations from parametric annihilators*, *Lett. Math. Phys.* **109** (2019) 497 [arXiv:1712.09215] [INSPIRE].

[11] A. von Manteuffel and R.M. Schabinger, *Quark and gluon form factors in four loop QCD: The N_f^2 and $N_{q\gamma}N_f$ contributions*, *Phys. Rev. D* **99** (2019) 094014 [arXiv:1902.08208] [INSPIRE].

[12] E. Panzer, *On hyperlogarithms and Feynman integrals with divergences and many scales*, *JHEP* **03** (2014) 071 [arXiv:1401.4361] [INSPIRE].

[13] A. von Manteuffel, E. Panzer and R.M. Schabinger, *A quasi-finite basis for multi-loop Feynman integrals*, *JHEP* **02** (2015) 120 [arXiv:1411.7392] [INSPIRE].

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

[15] W. Chen, *Reduction of Feynman integrals in the parametric representation*, *JHEP* **02** (2020) 115 [arXiv:1902.10387] [INSPIRE].

[16] A. von Manteuffel and R.M. Schabinger, *A novel approach to integration by parts reduction*, *Phys. Lett. B* **744** (2015) 101 [arXiv:1406.4513] [INSPIRE].

[17] T. Peraro, *Scattering amplitudes over finite fields and multivariate functional reconstruction*, *JHEP* **12** (2016) 030 [arXiv:1608.01902] [INSPIRE].

[18] S. Caron-Huot and A. Pokraka, *Duals of Feynman integrals. Part I. Differential equations*, *JHEP* **12** (2021) 045 [arXiv:2104.06898] [INSPIRE].

[19] S. Caron-Huot and A. Pokraka, *Duals of Feynman integrals. Part II. Generalized unitarity*, *JHEP* **04** (2022) 078 [arXiv:2112.00055] [INSPIRE].

[20] G. Fontana and T. Peraro, *Reduction to master integrals via intersection numbers and polynomial expansions*, *JHEP* **08** (2023) 175 [arXiv:2304.14336] [INSPIRE].

[21] S. Mizera, *Scattering amplitudes from intersection theory*, *Phys. Rev. Lett.* **120** (2018) 141602 [arXiv:1711.00469] [INSPIRE].

[22] P. Mastrolia and S. Mizera, *Feynman integrals and intersection theory*, *JHEP* **02** (2019) 139 [arXiv:1810.03818] [INSPIRE].

[23] S.E. Derkachov, J. Honkonen and Y.M. Pis'mak, *Three-loop calculation of the random walk problem: an application of dimensional transformation and the uniqueness method*, *J. Phys. A:Math. Gen.* **23** (1990) 5563 [INSPIRE].

[24] O.V. Tarasov, *Connection between Feynman integrals having different values of the space-time dimension*, *Phys. Rev. D* **54** (1996) 6479 [hep-th/9606018] [INSPIRE].

[25] R.N. Lee, *Calculating multiloop integrals using dimensional recurrence relation and D-analyticity*, *Nucl. Phys. B Proc. Suppl.* **205-206** (2010) 135 [arXiv:1007.2256] [INSPIRE].

[26] S. Laporta, *High-precision calculation of multiloop Feynman integrals by difference equations*, *Int. J. Mod. Phys. A* **15** (2000) 5087 [hep-ph/0102033] [INSPIRE].

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

JHEP10(2025)018

[28] R.D. Sameshima, *On different parametrizations of Feynman integrals*, Ph.D. thesis, CUNY, New York, NY, U.S.A. (2019) [INSPIRE].

[29] G. Bertolini, *Identità per integrali di Feynman in rappresentazioni parametriche* (in Italian), thesis, Università di Bologna, Bologna, Italy (2024).

[30] J. Gluza, K. Kajda and D.A. Kosower, *Towards a basis for planar two-loop integrals*, *Phys. Rev. D* **83** (2011) 045012 [arXiv:1009.0472] [INSPIRE].

[31] H. Ita, *Two-loop integrand decomposition into master integrals and surface terms*, *Phys. Rev. D* **94** (2016) 116015 [arXiv:1510.05626] [INSPIRE].

[32] Z. Wu et al., *NeatIBP 1.0, a package generating small-size integration-by-parts relations for Feynman integrals*, *Comput. Phys. Commun.* **295** (2024) 108999 [arXiv:2305.08783] [INSPIRE].

[33] R.M. Schabinger, *A new algorithm for the generation of unitarity-compatible integration by parts relations*, *JHEP* **01** (2012) 077 [arXiv:1111.4220] [INSPIRE].

[34] D. Bendle et al., *Integration-by-parts reductions of Feynman integrals using Singular and GPI-space*, *JHEP* **02** (2020) 079 [arXiv:1908.04301] [INSPIRE].

[35] S.L. Cacciatori and P. Mastrolia, *Intersection numbers in quantum mechanics and field theory*, arXiv:2211.03729 [INSPIRE].

[36] J. Böhm et al., *Complete sets of logarithmic vector fields for integration-by-parts identities of Feynman integrals*, *Phys. Rev. D* **98** (2018) 025023 [arXiv:1712.09737] [INSPIRE].

[37] S. Mizera, *Aspects of scattering amplitudes and moduli space localization*, Ph.D. thesis, Inst. Advanced Study, Princeton, NJ, U.S.A. (2020) [arXiv:1906.02099] [INSPIRE].

[38] H. Frellesvig et al., *Vector space of Feynman integrals and multivariate intersection numbers*, *Phys. Rev. Lett.* **123** (2019) 201602 [arXiv:1907.02000] [INSPIRE].

[39] S. Weinzierl, *On the computation of intersection numbers for twisted cocycles*, *J. Math. Phys.* **62** (2021) 072301 [arXiv:2002.01930] [INSPIRE].

[40] G. Fontana, *Rational algorithms for the decomposition of Feynman integrals via intersection theory*, M.Sc. thesis, Università di Bologna, Bologna, Italy (2022) [INSPIRE].

[41] C. Duhr and F. Porkert, *Feynman integrals in two dimensions and single-valued hypergeometric functions*, *JHEP* **02** (2024) 179 [arXiv:2309.12772] [INSPIRE].

[42] C. Duhr, F. Porkert, C. Semper and S.F. Stawinski, *Twisted Riemann bilinear relations and Feynman integrals*, *JHEP* **03** (2025) 019 [arXiv:2407.17175] [INSPIRE].

[43] C. Duhr, F. Porkert, C. Semper and S.F. Stawinski, *Self-duality from twisted cohomology*, *JHEP* **03** (2025) 053 [arXiv:2408.04904] [INSPIRE].

[44] G. Brunello et al., *Intersection numbers, polynomial division and relative cohomology*, *JHEP* **09** (2024) 015 [arXiv:2401.01897] [INSPIRE].

[45] P. Vanhove, *The physics and the mixed Hodge structure of Feynman integrals*, *Proc. Symp. Pure Math.* **88** (2014) 161 [arXiv:1401.6438] [INSPIRE].

[46] K. Bönisch et al., *Feynman integrals in dimensional regularization and extensions of Calabi-Yau motives*, *JHEP* **09** (2022) 156 [arXiv:2108.05310] [INSPIRE].

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

[48] H. Frellesvig, *The loop-by-loop Baikov representation — strategies and implementation*, *JHEP* **04** (2025) 111 [arXiv:2412.01804] [INSPIRE].

[49] Wolfram Research Inc., *Wolfram language documentation*, https://reference.wolfram.com/language/ref/Function.html.

JHEP10(2025)018