Computer Physics Communications 222 (2018) 295–312

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Algorithmic transformation of multi-loop master integrals to a canonical basis with *CANONICA*☆

Christoph Meyer

Institut für Physik, Humboldt-Universität zu Berlin, 12489 Berlin, Germany

ARTICLE INFO

Article history:
Received 2 June 2017
Received in revised form 31 August 2017
Accepted 14 September 2017
Available online 28 September 2017

Keywords:
Feynman integrals
Differential equations
Canonical form

ABSTRACT

The integration of differential equations of Feynman integrals can be greatly facilitated by using a canonical basis. This paper presents the Mathematica package *CANONICA*, which implements a recently developed algorithm to automatize the transformation to a canonical basis. This represents the first publicly available implementation suitable for differential equations depending on multiple scales. In addition to the presentation of the package, this paper extends the description of some aspects of the algorithm, including a proof of the uniqueness of canonical forms up to constant transformations.

Program summary
Program Title: CANONICA
Program Files doi: http://dx.doi.org/10.17632/fmwnmmhn77.1
Licensing provisions: GNU General Public License version 3
Programming language: Wolfram Mathematica, version 10 or higher
Nature of problem: Computation of a rational basis transformation of master integrals leading to a canonical form of the corresponding differential equation.
Solution method: The transformation law is expanded in the dimensional regulator. The resulting differential equations for the expansion coefficients of the transformation are solved with a rational ansatz.

1. Introduction

The calculation of higher order corrections to the cross-sections measured at the LHC is crucial in order to improve the understanding of both the background reactions as well as the signal processes. The current state of the art are NNLO QCD corrections to $2 \to 2$ processes involving a limited number of mass scales. A major challenge in these computations is the evaluation of the occurring Feynman integrals. While the calculation of Feynman integrals can be attempted with numerous approaches, the method of differential equations [1–3] has been particularly successful in the recent years [4–38]. This success is due to the observation [4] that the differential equation can be simplified significantly by turning to a so-called canonical basis of master integrals. The differential equation of a canonical basis of master integrals can easily be integrated in terms of iterated integrals such as multiple polylogarithms [39,40].

It is well known that Feynman integrals exist [32,41–49], which do not evaluate to this class of functions. These integrals generally satisfy differential equations of higher order. The solutions of the homogeneous part of these equations have been shown to be constructible by evaluating unitarity cuts [50–54].[1] Some integrals, which exceed the class of multiple polylogarithms, have recently been shown to be iterated integrals of modular forms [56]. However, the concept of a canonical basis has not yet been extended to integrals of this kind.

The class of Feynman integrals, which do admit a canonical basis, is still large and contains many integrals of phenomenological interest. It is therefore desirable to automate the calculation of these integrals as much as possible. The systematic application [57,58] of integration by parts relations [59,60] to reduce all scalar integrals to a finite number of master integrals has been automated in a variety of publicly available tools [61–69]. This leaves the process of choosing a canonical basis as the next step to be automated. A number of different methods [4,8,10,11,14,30,70–73] have been proposed to construct such a basis or, equivalently, the transformation from a given basis to a canonical basis. Until now, only implementations of the algorithm presented in Ref. [70] are publicly available [74–76]. However, this

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

E-mail address: christoph.meyer@physik.hu-berlin.de.

[1] In fact, unitarity cuts have also been used to derive differential equations of Feynman integrals [55].

https://doi.org/10.1016/j.cpc.2017.09.014
0010-4655/

algorithm is restricted to ordinary differential equations, which are not sufficient to describe the full functional dependence of Feynman integrals depending on multiple dimensionless scales. A wide class of phenomenologically relevant integrals is thus not covered.

This paper aims to overcome this restriction by introducing an implementation of the algorithm in Ref. [72], which is applicable to multi-scale problems. The accompanying `Mathematica` package *CANONICA* allows to calculate a rational transformation to a canonical basis for a given differential equation. In addition, the package provides some supplemental functionality for handling differential equations of Feynman integrals.

The description of the algorithm in Ref. [72] is extended in the present paper by a detailed account of the construction of the set of rational functions used for the ansatz. Moreover, the occurrence of non-linear polynomial equations in the parameters of the ansatz is addressed with a procedure to extract all relevant information by solving only linear equations, while maintaining all of the algorithms generality. The latter relies on the uniqueness of canonical forms up to constant transformations. While it is trivial to show that a constant transformation of a given canonical form leads again to a canonical form, it is not obvious that all possible canonical forms can be obtained in this way, which will be proven in this paper.

The paper is organized as follows. In Section 2 the algorithm in Ref. [72] is briefly reviewed and a description of the procedure for the generation of the ansatz is given. Furthermore, the uniqueness of canonical forms up to constant transformations is proven in this section. Building on this result, the treatment of non-linear polynomial equations in the parameters of the ansatz is discussed. Section 3 introduces the *CANONICA* package by outlining the installation and the contents of the package, which is followed by a few examples illustrating the usage of the main features of *CANONICA*. Furthermore, an overview over the hierarchy of the main public functions is given. The conclusions are drawn in section 4. A brief description of all functions and options provided by the package is contained in Appendices A and B. The global variables and protected symbols of the package are listed in Appendix C.

2. Algorithm

This section briefly reviews the algorithm introduced in Ref. [72] and presents more details on some aspects of the algorithm. In particular, the procedure used by *CANONICA* to generate an ansatz is described in detail. Furthermore, canonical forms are proven to be unique up to constant transformations. This result allows to attribute the occurrence of non-linear equations in the parameters of the ansatz precisely to this ambiguity. On this basis, a procedure to calculate the transformation by solving only linear equations is outlined.

2.1. Preliminaries

Let $\vec{f}(\epsilon, \{x_j\})$ denote the m-dimensional vector of master integrals, which depends on the dimensional regulator ϵ and a set $\{x_j\}$ of dimensionless invariants. By taking the total derivative of the vector of master integrals with respect to the invariants and expressing the result as a linear combination of master integrals, a coupled system of differential equations is obtained:

$$\mathrm{d}\vec{f}(\epsilon, \{x_j\}) = a(\epsilon, \{x_j\})\vec{f}(\epsilon, \{x_j\}), \tag{1}$$

with

$$a(\epsilon, \{x_j\}) = \sum_{i=1}^{M} a_i(\epsilon, \{x_j\})\mathrm{d}x_i. \tag{2}$$

Here $a_i(\epsilon, \{x_j\})$ denote $m \times m$ matrices of rational functions in the invariants and ϵ. Using the linear independence of the master integrals over the field of rational functions in the invariants and taking the exterior derivative of Eq. (1) implies the following integrability condition:

$$\mathrm{d}a - a \wedge a = 0, \tag{3}$$

which is a valuable consistency check for differential equations in several variables. Transforming the basis of master integrals with an invertible transformation T,

$$\vec{f} = T(\epsilon, \{x_j\})\vec{f}', \tag{4}$$

as suggested in Ref. [4], leads to the following transformation law for $a(\epsilon, \{x_j\})$:

$$a' = T^{-1}aT - T^{-1}\mathrm{d}T. \tag{5}$$

It has been observed [4] that with an appropriate change of the basis of master integrals, the differential equation can often be cast in the following form:

$$a'(\epsilon, \{x_j\}) = \epsilon \mathrm{d}\tilde{A}(\epsilon, \{x_j\}) = \epsilon \sum_{l=1}^{N} \tilde{A}_l \mathrm{d}\log(L_l(\{x_j\})), \tag{6}$$

where $\tilde{A}_l$ denote constant $m \times m$ matrices and the functions $L_l(\{x_j\})$ are called *letters*. The above form of the differential equation is called *canonical-* or *ϵ-form*. In this form, the integration of the differential equation in terms of iterated integrals is reduced to a merely combinatorial task (c.f. e.g., [6,9–12]).

2.2. Review of the algorithm

In this section the algorithm presented in Ref. [72] is briefly reviewed. Throughout this section, the existence of a rational transformation of the differential equation into ϵ-form is assumed. The purpose of the algorithm is to compute this transformation for a given differential equation, provided it exists. Any transformation to a canonical basis has to satisfy the following equation:

$$\epsilon \mathrm{d}\tilde{A} = T^{-1} a T - T^{-1} \mathrm{d}T, \tag{7}$$

for some $\mathrm{d}\tilde{A}$ of the form in Eq. (6). The resulting differential form $\mathrm{d}\tilde{A}$ is unknown and thus has to be determined as well. In Ref. [72] it has been proven that the determinant of the transformation is fixed up to a rational function $F(\epsilon)$ by the trace of the differential form a

$$\mathrm{Tr}[a] = \epsilon X(\{x_j\}) + Y(\epsilon, \{x_j\}), \tag{8}$$

where $X(\{x_j\})$ denotes the sum of dlog-terms with coefficients proportional to ϵ and $Y(\epsilon, \{x_j\})$ denotes the sum of dlog-terms with constant coefficient. Then the determinant is given by

$$\det(T) = F(\epsilon) \exp\left(\int_\gamma Y(\epsilon, \{x_j\}) \right), \tag{9}$$

and the trace of the resulting ϵ-form is determined by

$$\mathrm{Tr}[\mathrm{d}\tilde{A}] = X(\{x_j\}). \tag{10}$$

For invertible transformations T, Eq. (7) can equivalently be written as

$$\mathrm{d}T - aT + \epsilon T \mathrm{d}\tilde{A} = 0. \tag{11}$$

The basic idea of the algorithm is to expand this equation in ϵ and solve for the expansion coefficients of the transformation with a rational ansatz. However, the expansion of T may not be finite and therefore an additional step has to be taken, which is reviewed in the following. As $a(\epsilon, \{x_j\})$ is required to be rational in both the invariants and ϵ, a polynomial $h(\epsilon, \{x_j\})$ exists such that $\hat{a} = ah$ has a finite Taylor expansion in ϵ

$$\hat{a} = \sum_{k=0}^{k_{\max}} \epsilon^k \hat{a}^{(k)}. \tag{12}$$

In order to fix $h(\epsilon, \{x_j\})$ up to an irrelevant constant factor, $h(\epsilon, \{x_j\})$ is required to satisfy the above condition with the smallest possible number of irreducible factors. The expansion of h is denoted as follows:

$$h(\epsilon, \{x_j\}) = \sum_{l=l_{\min}}^{l_{\max}} \epsilon^l h^{(l)}(\{x_j\}), \quad l_{\min} \geq 0. \tag{13}$$

Rewriting Eq. (11) in terms of $\hat{T} = Th$ yields the following equation:

$$-\hat{T}\mathrm{d}h + h\mathrm{d}\hat{T} - \hat{a}\hat{T} + \epsilon h \hat{T} \mathrm{d}\tilde{A} = 0. \tag{14}$$

It has been shown in Ref. [72] that Eq. (14) will have a solution for $\hat{T}$ with finite expansion, if Eq. (11) has a rational solution for T. This allows to expand $\hat{T}$ in ϵ

$$\hat{T} = \sum_{n=l_{\min}}^{n_{\max}} \epsilon^n \hat{T}^{(n)}, \tag{15}$$

and solve Eq. (14) order by order in ϵ for finitely many coefficients $\hat{T}^{(n)}$. The equations at each order are solved by making an ansatz for $\hat{T}^{(n)}$ in terms of rational functions of the invariants $r_k(\{x_j\})$

$$\hat{T}^{(n)} = \sum_{k=1}^{|\mathcal{R}_T|} \tau_k^{(n)} r_k(\{x_j\}), \tag{16}$$

$$\mathcal{R}_T = \left\{ r_1(\{x_j\}), \ldots, r_{|\mathcal{R}_T|}(\{x_j\}) \right\}, \tag{17}$$

where $\tau_k^{(n)}$ denote the unknown $m \times m$ matrices independent of the invariants and the regulator, which are to be determined by the algorithm. More details on the choice of the set of rational functions $\mathcal{R}_T$ are given in Section 2.3. For the unknown $\tilde{A}$ an ansatz of the following form can be used:

$$\tilde{A} = \sum_{l=1}^{N} \alpha_l \log(L_l(\{x_j\})), \tag{18}$$

where α_l are considered to be unknown $m \times m$ matrices independent of the invariants and the regulator. The set of polynomials $L_l(\{x_j\})$ is taken to be the set of irreducible denominator factors of the differential form $a(\epsilon, \{x_j\})$ with trivial dependence on the regulator. In Section 2.4, this set is shown to contain all letters of the resulting canonical form.

Inserting the ansatz in the expansion of Eq. (14) and requiring the resulting equations to hold for all non-singular values of the invariants implies polynomial equations in the parameters of the ansatz. For more details on the solution of these equations, see Section 2.7.

It is well known that the differential form $a(\epsilon, \{x_j\})$ can be cast in a block-triangular form. This allows to split the computation of the transformation into a recursion over the sectors of the differential equation, which leads to significant performance improvements. With regard to the recursion step, consider a differential equation where all previous sectors have already been transformed into ϵ-form. The first part of the recursion step is to transform the diagonal block of the next sector into ϵ-form with the part of the algorithm described above. After this step, the differential equation is in the following form:

$$a_l = \left(\begin{array}{c|c} \epsilon\tilde{c} & 0 \\ \hline b & \epsilon\tilde{e} \end{array}\right), \tag{19}$$

where $\tilde{c}$ and $\tilde{e}$ are in dlog-form. It has been shown in Ref. [72] that the transformation of a differential equation in this form can be split into two parts. First, the off-diagonal block b is transformed into dlog-form with a transformation of the form

$$t_D = \left(\begin{array}{c|c} \mathbb{I} & 0 \\ \hline D & \mathbb{I} \end{array}\right), \tag{20}$$

which is determined by a differential equation for D

$$\mathrm{d}D - \epsilon(\tilde{e}D - D\tilde{c}) = b - b'. \tag{21}$$

Here b' is an unknown quantity, which is required to be in dlog-form. The above equation is solved by first multiplying D by appropriate factors to render its expansion finite. The expansion coefficients are then determined by making an ansatz in terms of a set of rational functions $\mathcal{R}_D$. For more details, see [72] and Section 2.5, which describes how the set $\mathcal{R}_D$ is constructed.

The last part of the recursion step is to employ a procedure proposed in Ref. [70] to compute a rational transformation in the regulator, which transforms the full differential equation into ϵ-form.

2.3. Ansatz for diagonal blocks

In this section, the choice of the set $\mathcal{R}_T$ of rational functions in the ansatz is discussed. The basic compromise with the ansatz is to choose it large enough to encompass the solution and as small as possible in order to keep the resulting number of equations small and therefore allow the algorithm to perform well. The goal of this section is to present a procedure to generate a finite set of rational functions for a given differential form $a(\epsilon, \{x_j\})$, which can then be used as an ansatz. The first step towards this goal is to determine the set of possible denominator factors of the transformation to a canonical basis. A natural guess is to consider the set of irreducible denominator factors of $\hat{a}$, which is proven in the following to contain all possible factors.

It is useful to first define some notation. Let $f(\{x_j\})$ be an irreducible polynomial and $S(\epsilon, \{x_j\})$ some matrix-valued rational differential form or function. Then, the notation

$$S \sim \frac{1}{f^n} \tag{22}$$

indicates[2] $n \in \mathbb{N}$ to be the maximal number for which S can be written as

$$S = R\frac{1}{f^n}. \tag{23}$$

Here R is required to be nonzero and not to be the product of f and a quantity, which is finite on the set of all zeros of f. The set $\mathcal{I}(S)$ of irreducible denominator factors of S is then given by those factors f with $S \sim 1/f^k$ and $k \geq 1$.

Claim 1. *Each irreducible denominator factor $f(\{x_j\})$ of a rational solution $\hat{T}$ of* Eq. (14) *is an irreducible denominator factor of $\hat{a}$.*

The assertion in the claim is equivalent to $\mathcal{I}(\hat{T}) \subseteq \mathcal{I}(\hat{a})$, which will be proven by showing that

$$\hat{T} \sim \frac{1}{f^n}, \quad n \geq 1 \tag{24}$$

implies

$$\hat{a} \sim \frac{1}{f^k}, \quad k \geq 1. \tag{25}$$

To this end, Eq. (24) is assumed to hold. It is instructive to rearrange the terms in Eq. (14)

$$-\hat{T}\mathrm{d}h + h(\mathrm{d}\hat{T} + \epsilon\hat{T}\mathrm{d}\tilde{A}) = \hat{a}\hat{T}. \tag{26}$$

Since $\mathrm{d}h$ is polynomial, the term $\hat{T}\mathrm{d}h$ behaves as

$$\hat{T}\mathrm{d}h \sim \frac{1}{f^k}, \quad k \leq n. \tag{27}$$

[2] Throughout this paper, the number 0 is understood to be included in the natural numbers.

Given Eq. (24), there must be a lowest order $s \in \mathbb{Z}$ in the ϵ-expansion of $\hat{T}$ with

$$\hat{T}^{(s)} \sim \frac{1}{f^n}, \tag{28}$$

and consequently

$$\hat{T}^{(s-1)} \sim \frac{1}{f^k}, \quad k < n. \tag{29}$$

The derivative raises the power of f by one, which implies

$$\mathrm{d}\hat{T}^{(s)} \sim \frac{1}{f^{n+1}}. \tag{30}$$

Note that $\mathrm{d}\tilde{A}$ is in dlog-form, and therefore

$$\mathrm{d}\tilde{A} \sim \frac{1}{f^k}, \quad k \leq 1. \tag{31}$$

Taking both Eq. (29) and Eq. (31) into account, it follows

$$\hat{T}^{(s-1)}\mathrm{d}\tilde{A} \sim \frac{1}{f^k}, \quad k \leq n. \tag{32}$$

This implies for the order s of the expansion of $(\mathrm{d}\hat{T} + \epsilon\hat{T}\mathrm{d}\tilde{A})$

$$\mathrm{d}\hat{T}^{(s)} + \hat{T}^{(s-1)}\mathrm{d}\tilde{A} \sim \frac{1}{f^{n+1}}, \tag{33}$$

which also holds for the full expression $(\mathrm{d}\hat{T} + \epsilon\hat{T}\mathrm{d}\tilde{A})$. Due to the minimality requirement, h does not contain any irreducible factors independent of ϵ and therefore the multiplication of the term in brackets with h in Eq. (26) cannot cancel any power of f, because f is independent of ϵ. Since Eq. (27) shows that the term $\hat{T}\mathrm{d}h$ is of lower order in f than $h(\mathrm{d}\hat{T} + \epsilon\hat{T}\mathrm{d}\tilde{A})$, the whole left-hand side of Eq. (26) is of order $1/f^{n+1}$ and consequently the right-hand side as well:

$$\hat{a}\hat{T} \sim \frac{1}{f^{n+1}}. \tag{34}$$

Since $\hat{T}$ is only of order $1/f^n$, it can be concluded

$$\hat{a} \sim \frac{1}{f^k}, \quad k \geq 1, \tag{35}$$

which proves the claim. Thus, the ansatz can without loss of generality be restricted to the set

$$\mathcal{Q} = \left\{ \frac{x_1^{p_1} \cdots x_M^{p_M}}{f_1^{q_1} \cdots f_U^{q_U}} \;\middle|\; p_1, \ldots, p_M, q_1, \ldots, q_U \in \mathbb{N} \right\} \tag{36}$$

of rational functions with the denominator factors drawn from the set $\mathcal{I}(\hat{a}) = \{f_1, \ldots, f_U\}$ of irreducible denominator factors of $\hat{a}$.

As was argued in Ref. [72], rational functions may be decomposed in terms of a class of simpler rational functions, called Leinartas functions [77,78]. Let $\mathcal{L}(\mathcal{Q})$ denote a basis of the K-span of $\mathcal{Q}$ in terms of Leinartas functions. While $\mathcal{L}(\mathcal{Q})$ is guaranteed to contain the correct ansatz, it is still an infinite set. Therefore, a constructive procedure is needed to generate a finite subset of $\mathcal{L}(\mathcal{Q})$ for a given $a(\epsilon, \{x_j\})$. This procedure should be inexpensive to compute while yielding a correct ansatz for most practical examples. Since the procedure outlined in the following is not proven to generate a correct ansatz, it is important to be able to systematically enlarge the ansatz in a way that is guaranteed to eventually encompass the solution.

The strategy to define a finite subset of $\mathcal{L}(\mathcal{Q})$ is to set restrictions on the powers of the invariants in the numerator as well as on the powers of the denominator factors.

While the powers of those factors occurring in $a(\epsilon, \{x_j\})$ may be suspected to be a good indicator for the powers in the transformation, the following simple example demonstrates this to be false. Consider the differential equation

$$a(\epsilon, \{x\}) = \left(-\frac{\alpha}{x} + \frac{\epsilon}{x}\right)\mathrm{d}x, \quad \alpha \in \mathbb{Z}, \tag{37}$$

which contains the factor x with the negative power one. However, for any given integer α, the rational transformation to the canonical form reads

$$T(\epsilon, \{x\}) = \frac{1}{x^\alpha}. \tag{38}$$

Consequently, the transformation can contain any power of the factor x, while the power of the same factor in the differential equation remains fixed. A much better predictor is given by the determinant of the transformation, which in the 1-dimensional example above is identical to the transformation itself and therefore always yields the correct power of the factor x. For higher dimensional differential equations, the determinant does not fix the transformation but still carries information on the powers of the irreducible denominator factors of the transformation. Let the determinant of the transformation read

$$\det(T) = F(\epsilon, \{x_j\}) \prod_{i=1}^{U} f_i^{-\lambda_i}, \quad \lambda_i \in \mathbb{Z},\ f_i \in \mathcal{I}(\hat{a}), \tag{39}$$

where $F(\epsilon, \{x_j\})$ denotes the product of all irreducible factors with a non-trivial dependence on the regulator. Then, for each factor f_i with $\lambda_i > 0$, there has to be a component T_{jl} of the transformation satisfying

$$T_{jl} \sim \frac{1}{f_i^k}, \quad k \geq \left\lceil \frac{\lambda_i}{m} \right\rceil, \tag{40}$$

where m denotes the dimension of the differential equation. Thus, the determinant sets lower bounds on the maximal powers of the denominator factors in the transformation, which have to be taken into account in the construction of the ansatz.

In the following, a finite subset of $\mathcal{Q}$ will be constructed, which then leads to a finite subset of $\mathcal{L}(\mathcal{Q})$ by taking a basis of its K-span in terms of Leinartas functions. The powers λ_i are used to define a set of denominators:

$$\mathcal{D}(\delta_D) = \left\{ \frac{1}{f_{i_1}^{p_{i_1}} \cdots f_{i_M}^{p_{i_M}}} \;\middle|\; f_{i_j} \in \mathcal{I}(\hat{a}), 0 \leq p_i \leq \Theta(\lambda_i)\lambda_i + \delta_D, i_j \neq i_k \text{ for } j \neq k \right\}, \tag{41}$$

which has been restricted to at most M denominator factors with M denoting the number of invariants. Any higher number of polynomials in M invariants is algebraically dependent and therefore reducible in terms of Leinartas functions with M or less denominator polynomials. The parameter $\delta_D \in \mathbb{N}$ has been introduced to define a way to enlarge the set of $\mathcal{D}(\delta_D)$ systematically. The default value is going to be $\delta_D = 0$. The lower bounds in Eq. (40) are satisfied for all allowed values of δ_D. For the numerators, consider the set of all possible monomials up to a fixed bound on their total degree

$$\mathcal{N}(\delta_N) = \left\{ x_1^{\nu_1} \cdots x_M^{\nu_M} \;\middle|\; \nu_1, \ldots, \nu_M \in \mathbb{N}, \; \sum_{i=1}^{M} \nu_i \leq 3 + \delta_N \right\}, \tag{42}$$

where the parameter δ_N has been introduced to control the highest total degree of the monomials in $\mathcal{N}(\delta_N)$. For the default value $\delta_N = 0$ the highest total degree of the numerator monomials is three. This choice is made based on practical examples and is intended to make the default value $\delta_N = 0$ work for most cases and at the same time yield a rather small ansatz. Furthermore, it has proven useful to also include the following sets of monomials:

$$\mathcal{N}_{\text{det}} = \{\text{numerator monomials of } \det(T)\}, \tag{43}$$

$$\mathcal{N}_a = \left\{\text{numerator monomials of the } \hat{a}^{(k)}(\{x_j\})\right\}, \tag{44}$$

in order to capture the correct ansatz in more cases already with the default value $\delta_N = 0$. Usually, the inclusion of $\mathcal{N}_{\text{det}}$ and $\mathcal{N}_a$ does not significantly enlarge the ansatz, while making the default value work for more examples. Finally, the ansatz $\mathcal{R}_T$ is obtained by computing a basis of Leinartas functions of the K-span of the set of rational functions drawing their numerators and denominators from the sets defined above:

$$\mathcal{R}_T(\delta_D, \delta_N) = \mathcal{L}\left(\left\{ \frac{p}{f} \;\middle|\; f \in \mathcal{D}(\delta_D), \;\; p \in \mathcal{N}(\delta_N) \cup \mathcal{N}_a \cup \mathcal{N}_{\text{det}} \right\} \right). \tag{45}$$

The set $\mathcal{R}_T(\delta_D, \delta_N)$ is finite and contains all elements of $\mathcal{L}(\mathcal{Q})$ necessary to represent the elements of $\mathcal{Q}$ with denominators from $\mathcal{D}(\delta_D)$ and numerators from $\mathcal{N}(\delta_N)$. Therefore, by increasing the values of δ_D and δ_N, the set $\mathcal{R}_T(\delta_D, \delta_N)$ can be systematically extended to the whole set of $\mathcal{L}(\mathcal{Q})$, which contains the correct ansatz. While the correct ansatz is necessarily contained in $\mathcal{L}(\mathcal{Q})$, the choice of the finite subset $\mathcal{R}_T \subset \mathcal{L}(\mathcal{Q})$ presented here is a heuristic procedure. However, the knowledge of upper bounds on δ_D and δ_N would be enough to turn the algorithm into a computable criterion for the existence of a rational transformation transforming a given differential equation into canonical form.

2.4. Ansatz for the resulting canonical form

The ansatz for the resulting canonical form in Eq. (18) requires the knowledge of a set of polynomials in the invariants that encompasses the set of letters of the resulting canonical form. In this section, these letters will be shown to be a subset of the set $\mathcal{I}(a)$ of irreducible denominator factors of the original differential equation with trivial dependence on the regulator. Consider the transformation law Eq. (7)

$$\epsilon \mathrm{d}\tilde{A} = T^{-1}(aT - \mathrm{d}T). \tag{46}$$

Since the transformation T is rational, the derivative does not alter the set of its denominator factors $\mathcal{I}(\mathrm{d}T) = \mathcal{I}(T)$. Claim 1 implies $\mathcal{I}(T) \subseteq \mathcal{I}(a)$ and thus

$$\mathcal{I}(aT - \mathrm{d}T) \subseteq \mathcal{I}(a). \tag{47}$$

The denominator factors of T^{-1} can be deduced by writing the inverse as

$$T^{-1} = \det(T)^{-1}\mathrm{adj}(T). \tag{48}$$

The cofactors in the adjugate of T are a sum of products of components of T, which implies

$$\mathcal{I}(\mathrm{adj}(T)) \subseteq \mathcal{I}(T) \subseteq \mathcal{I}(a). \tag{49}$$

Due to Eq. (9) and Eq. (8), the determinant of T can be written in the form

$$\det(T) = F(\epsilon, \{x_j\}) \prod_{i=1}^{U} f_i(\{x_j\})^{-\lambda_i}, \quad \lambda_i \in \mathbb{Z}, \quad f_i \in \mathcal{I}(a), \tag{50}$$

which leads to

$$\mathcal{I}(\det(T)^{-1}) \subseteq \mathcal{I}(a). \tag{51}$$

Thus, the irreducible denominator factors of the right-hand side of Eq. (46) have been shown to be a subset of $\mathcal{I}(a)$, hence the same holds for those of the left-hand side

$$\mathcal{I}(\mathrm{d}\tilde{A}) \subseteq \mathcal{I}(a). \tag{52}$$

Since $\mathcal{I}(\mathrm{d}\tilde{A})$ is equal to the set of letters of $\tilde{A}$, this allows to restrict the set of polynomials in the ansatz in Eq. (18) to the set $\mathcal{I}(a)$.

2.5. Ansatz for off-diagonal blocks

The transformation t_D in Eq. (20), which transforms the off-diagonal blocks into dlog-form, is determined by Eq. (21). A rational solution of this equation for D is computed by making a rational ansatz. In this section, the set of rational functions $\mathcal{R}_D$ to be used for the ansatz is constructed. First, the set of irreducible denominator factors of D will be shown to be a subset of the irreducible denominator factors of b. Here and in the following, these factors are assumed not to depend on the regulator unless stated otherwise. The argument proceeds similarly to the one in the proof of claim 1. In this case it will even be possible to derive upper bounds on the powers of the irreducible denominator factors of D. In a second step, these global upper bounds will be refined to upper bounds for the individual components of D, which reduces the number of rational functions in the ansatz considerably.

The set of possible irreducible denominator factors occurring in a rational solution D of

$$\mathrm{d}D - \epsilon(\tilde{e}D - D\tilde{c}) = b - b' \tag{53}$$

can be determined from the denominator factors of b. In order to demonstrate this, assume

$$D \sim \frac{1}{f^n}, \quad n \geq 1, \tag{54}$$

where the same notation as in Section 2.3 is used. Then, there exists a lowest order $s \in \mathbb{Z}$ in the expansion of D with

$$D^{(s)} \sim \frac{1}{f^n} \tag{55}$$

and therefore

$$D^{(s-1)} \sim \frac{1}{f^k}, \quad k < n. \tag{56}$$

The derivative raises the order of f by one

$$\mathrm{d}D^{(s)} \sim \frac{1}{f^{n+1}}. \tag{57}$$

Since $\tilde{e}$ and $\tilde{c}$ are in dlog-form and thus at most of order $1/f$, it follows

$$\mathrm{d}D^{(s)} - (\tilde{e}D^{(s-1)} - D^{(s-1)}\tilde{c}) \sim \frac{1}{f^{n+1}}, \tag{58}$$

which in turn implies the left-hand side and therefore also the right-hand side of Eq. (53) to be of order $1/f^{n+1}$

$$b - b' \sim \frac{1}{f^{n+1}}. \tag{59}$$

As b' is in dlog-form and consequently at most of order $1/f$, it can be concluded

$$b \sim \frac{1}{f^{n+1}}. \tag{60}$$

This result allows to extract upper bounds on the order of the irreducible denominator factors of D from a given b. Let $\mathcal{I}(b) = \{f_1, \ldots, f_U\}$ denote the set of irreducible denominator factors of b and λ_i the order of the denominator factor f_i

$$b \sim \frac{1}{f_i^{\lambda_i}}, \quad i = 1, \ldots, U. \tag{61}$$

According to the argument above, the upper bounds μ_i

$$D \sim \frac{1}{f_i^{k_i}}, \quad 0 \leq k_i \leq \mu_i, \quad i = 1, \ldots, U, \tag{62}$$

are given by

$$\mu_i = \lambda_i - 1, \quad i = 1, \ldots, U. \tag{63}$$

Rather than using these bounds to make an ansatz, it is beneficial to reduce the combinatorics of the ansatz by refining the above bounds. The idea is to infer bounds on the powers of the denominator factors of individual components of the solution D rather than for all components at once. Assume

$$D_{ij} \sim \frac{1}{f^n}, \quad n \geq 1, \tag{64}$$

and let $s \in \mathbb{Z}$ denote the lowest order in the expansion of D_{ij} with

$$D_{ij}^{(s)} \sim 1/f^n. \tag{65}$$

The derivative raises the power by one

$$\mathrm{d}D_{ij}^{(s)} \sim \frac{1}{f^{n+1}}. \tag{66}$$

Consider a component of the order s in the expansion of Eq. (53)

$$\mathrm{d}D_{ij}^{(s)} - (\tilde{e}D^{(s-1)} - D^{(s-1)}\tilde{c})_{ij} = b_{ij}^{(s)} - b_{ij}^{\prime(s)}. \tag{67}$$

Since b' is in dlog-form, this term cannot cancel the order $1/f^{n+1}$ of the derivative term. Therefore at least one of the following cases must be true:

case 1:

$$b_{ij}^{(s)} \sim \frac{1}{f^k}, \quad k \geq n+1, \tag{68}$$

case 2:

$$(\tilde{e}D^{(s-1)} - D^{(s-1)}\tilde{c})_{ij} \sim \frac{1}{f^k}, \quad k \geq n+1. \tag{69}$$

In case 2, there has to be at least one index α with either

$$\tilde{e}_{i\alpha} \sim \frac{1}{f^1} \quad \text{and} \quad D_{\alpha j}^{(s-1)} \sim \frac{1}{f^k}, \quad k \geq n \tag{70}$$

or

$$\tilde{e}_{i\alpha} \sim \frac{1}{f^0} \quad \text{and} \quad D_{\alpha j}^{(s-1)} \sim \frac{1}{f^k}, \quad k \geq n+1, \tag{71}$$

or there exists at least one index β with either

$$\tilde{c}_{\beta j} \sim \frac{1}{f^1} \quad \text{and} \quad D_{i\beta}^{(s-1)} \sim \frac{1}{f^k}, \quad k \geq n \tag{72}$$

or

$$\tilde{c}_{\beta j} \sim \frac{1}{f^0} \quad \text{and} \quad D_{i\beta}^{(s-1)} \sim \frac{1}{f^k}, \quad k \geq n+1. \tag{73}$$

So far, the assumption $D_{ij}^{(s)} \sim 1/f^n$ has been demonstrated to imply either $b_{ij}^{(s)} \sim 1/f^k$ for some $k \geq n+1$ (case 1) or that some other component of $D^{(s-1)}$ is of order $1/f^k$ with $k \geq n$ (case 2). In case 2, the whole argument can be applied again to the respective components of $D^{(s-1)}$. This can be repeated until either the lowest order in the expansion is reached and therefore case 2 is not possible anymore or at some point only case 1 is possible due to the structure of $\tilde{e}$ and $\tilde{c}$. Thus, all possible chains of this argument necessarily end with case 1. Since $\tilde{e}$, $\tilde{c}$ and b are known prior to the computation of D, the chains can be followed backwards in order to derive upper bounds on the powers of the denominator factors of the components of D. The idea is to consider all chains at once and start at the last step by reversing case 1 for all components of D. Using the powers of the denominator factors of b

$$b_{ij} \sim \frac{1}{f_k^{\lambda_{k,ij}}}, \tag{74}$$

case 1 is reversed for all components by setting the upper bounds $\mu_{k,ij}$ of D_{ij} on f_k

$$D_{ij} \sim \frac{1}{f_k^p}, \quad 0 \leq p \leq \mu_{k,ij}, \tag{75}$$

to

$$\mu_{k,ij} = \lambda_{k,ij} - 1, \quad \forall k, i, j. \tag{76}$$

It can then be deduced from $\tilde{e}$ and $\tilde{c}$ for each component which other components could have implied the current bounds via case 2. For instance, if there exists an α with

$$\tilde{e}_{i\alpha} \sim \frac{1}{f^1}, \tag{77}$$

and the current bound for the order in $1/f$ of $D_{\alpha j}$ is n, case 2 is reversed by setting the bound on the order of D_{ij} to n as well, unless it is already higher. At each step it is checked for all components D_{ij}, whether there is an $\tilde{e}_{i\alpha}$ as in Eq. (70) or Eq. (71) or a $\tilde{c}_{\beta j}$ as in Eq. (72) or Eq. (73). If this is the case, the bounds are updated accordingly. This is repeated until the bounds do not change anymore, and therefore they incorporate all possible cases. Algorithm 1 summarizes the procedure. Since the values of the bounds $\mu_{k,ij}$ can only increase during each iteration in algorithm 1 and the overall bounds μ_k given in Eq. (63) are upper bounds on the bounds of the components

$$\mu_{k,ij} \leq \mu_k, \quad \forall k, i, j, \tag{78}$$

it is clear that the algorithm terminates after a finite number of steps. Using the bounds computed with Algorithm 1, the following sets of rational functions can be defined:

$$\mathcal{R}_{ij}(\delta_N) = \left\{ \frac{p}{f_1^{q_1} \dots f_U^{q_U}} \;\middle|\; p \in \mathcal{N}(\delta_N),\ 0 \leq q_k \leq \mu_{k,ij}\ \forall k \right\}, \tag{79}$$

with

$$\mathcal{N}(\delta_N) = \left\{ x_1^{\nu_1} \cdots x_M^{\nu_M} \;\middle|\; \nu_1, \dots, \nu_M \in \mathbb{N},\ \sum_{i=1}^{M} \nu_i \leq 3 + \delta_N \right\}. \tag{80}$$

Input: $\{\lambda_{k,ij}\}, \tilde{e}, \tilde{c}$
Output: Set of upper bounds $\mu_{k,ij}$ with $D_{ij} \sim 1/f^k$ and $0 \leq k \leq \mu_{k,ij}$
$\mu_{k,ij} = \lambda_{k,ij} - 1$.
repeat
  foreach k, i, j, α, β **do**
    if $\tilde{e}_{i\alpha} \sim \frac{1}{f^0}$ **then** $\mu_{k,ij} = \max\left(\mu_{k,ij}, \mu_{k,\alpha j} - 1\right)$;
    if $\tilde{e}_{i\alpha} \sim \frac{1}{f^1}$ **then** $\mu_{k,ij} = \max\left(\mu_{k,ij}, \mu_{k,\alpha j}\right)$;
    if $\tilde{c}_{\beta j} \sim \frac{1}{f^0}$ **then** $\mu_{k,ij} = \max\left(\mu_{k,ij}, \mu_{k,i\beta} - 1\right)$;
    if $\tilde{c}_{\beta j} \sim \frac{1}{f^1}$ **then** $\mu_{k,ij} = \max\left(\mu_{k,ij}, \mu_{k,i\beta}\right)$;
  end
until *Bounds μ do not change anymore*;
return $\{\mu_{k,ij}\}$

Algorithm 1: Determination of upper bounds on the powers of the denominator factors of the components of D.

The above argument shows that for high enough δ_N, the component D_{ij} is an element of the K-span of $\mathcal{R}_{ij}(\delta_N)$. For the ansatz, a basis of Leinartas functions of the K-span of the union of all $\mathcal{R}_{ij}(\delta_N)$ is taken

$$\mathcal{R}_D(\delta_N) = \mathcal{L}\left(\bigcup_{i,j} \mathcal{R}_{ij}(\delta_N)\right). \tag{81}$$

It would be more efficient to make a different ansatz for each component of D using $\mathcal{L}(\mathcal{R}_{ij}(\delta_N))$. However, this functionality will only be included in a future version of *CANONICA*.

2.6. On the uniqueness of canonical bases

The application of a constant invertible transformation C to a differential equation in canonical form obviously preserves the canonical form

$$a' = \epsilon \sum_{l=1}^{N} \left(C^{-1} \tilde{A}_l C\right) \mathrm{d}\log(L_l). \tag{82}$$

This raises the question whether *all* canonical forms can be obtained in this way. The following claim shows that indeed every canonical form can be obtained by a constant transformation from any other canonical form. In this sense the canonical form of a given differential equation is unique up to constant transformations.

Claim 2. *Let $a(\epsilon, \{x_j\})$ be a differential equation of Feynman integrals and $T_1(\epsilon, \{x_j\})$ and $T_2(\epsilon, \{x_j\})$ be invertible rational transformations, which transform it into the canonical forms $\epsilon \mathrm{d}\tilde{A}_1(\{x_j\})$ and $\epsilon \mathrm{d}\tilde{A}_2(\{x_j\})$, respectively. Then there exists a constant invertible transformation C transforming $\epsilon \mathrm{d}\tilde{A}_1(\{x_j\})$ into $\epsilon \mathrm{d}\tilde{A}_2(\{x_j\})$.*

Consider the transformation $T = T_1^{-1} T_2$, which transforms $\epsilon \mathrm{d}\tilde{A}_1$ to $\epsilon \mathrm{d}\tilde{A}_2$. First, the transformation T has to be shown to be independent of the invariants. The corresponding transformation law reads

$$\epsilon \mathrm{d}\tilde{A}_2 = T^{-1} \epsilon \mathrm{d}\tilde{A}_1 T - T^{-1} \mathrm{d}T. \tag{83}$$

It is instructive to rewrite this equation:

$$\mathrm{d}T = \epsilon \left(\mathrm{d}\tilde{A}_1 T - T \mathrm{d}\tilde{A}_2 \right) \tag{84}$$

$$= \epsilon \sum_{l=1}^{N} \left(\tilde{A}_{1l} T - T \tilde{A}_{2l} \right) \mathrm{d}\log(L_l). \tag{85}$$

The summation over the letters is meant to run over the union of the sets of letters of the two canonical forms, since it is a priori not clear that they both have exactly the same set of letters. The letters are assumed to be irreducible polynomials and the union is meant to remove all scalar multiples of letters as well. Since the transformation law is invariant under the multiplication of T with any rational function $g(\epsilon)$, the ϵ-expansion of T can be assumed to start at the order ϵ^0. Then the first order in the expansion of the above equation reads

$$\mathrm{d}T^{(0)} = 0 \tag{86}$$

and therefore $T^{(0)}$ has to be constant. At any order $n > 0$ the expansion of the above equation is given by

$$\mathrm{d}T^{(n)} = \sum_{l=1}^{N} \left(\tilde{A}_{1l} T^{(n-1)} - T^{(n-1)} \tilde{A}_{2l} \right) \mathrm{d}\log(L_l). \tag{87}$$

Assuming $T^{(n-1)}$ to be constant, this equation can easily be integrated

$$T^{(n)} = \sum_{l=1}^{N} \left(\tilde{A}_{1l} T^{(n-1)} - T^{(n-1)} \tilde{A}_{2l} \right) \log(L_l) + const. \tag{88}$$

Since T_1 and T_2 are assumed to be rational in ϵ and the invariants, the same holds for T and therefore the coefficients of its ϵ-expansion have to be rational as well. This implies

$$\tilde{A}_{1l} T^{(n-1)} - T^{(n-1)} \tilde{A}_{2l} = 0, \quad \forall\, l \tag{89}$$

and consequently $T^{(n)}$ has to be constant. By induction, these arguments imply that all coefficients of the ϵ-expansion of T are constant and therefore $T = T(\epsilon)$. As T is independent of the invariants, the transformation law Eq. (83) has the form

$$\mathrm{d}\tilde{A}_2 = T(\epsilon)^{-1} \mathrm{d}\tilde{A}_1 T(\epsilon). \tag{90}$$

It can be concluded that $T(\epsilon)$ transforms $\mathrm{d}\tilde{A}_1$ to $\mathrm{d}\tilde{A}_2$ for all non-singular values of ϵ, because the left-hand side does not depend on ϵ. Upon choosing such a value ϵ_0, a constant invertible transformation $C = T(\epsilon_0)$ is obtained, which concludes the proof of the claim. The same argument also holds for the more general case of an algebraic dependence of T_1 and T_2 on ϵ and the invariants. Altogether, canonical forms have been shown to be unique modulo $GL(m, K)$ transformations.

This result explains the origin of the non-linear parameter equations, which are treated in Section 2.7. Moreover, the above result can be utilized for the comparison of two different canonical forms of the same problem, provided they are expressed in the same set of invariants. In this situation, claim 2 asserts the existence of a constant transformation relating the two canonical forms. This can be tested by checking whether the following system of linear equations

$$C\tilde{A}_{2l} = \tilde{A}_{1l} C, \quad l = 1, \ldots, N, \tag{91}$$

has a non-singular solution for the components of C.

The uniqueness of the canonical form also manifests itself in Eq. (21), which governs the transformation of the off-diagonal blocks into dlog-form. In the following, the rational solution of this equation is proven to be unique up to the addition of terms depending solely on the regulator. In practice, this result allows to exclude terms with trivial dependence on the invariants from the ansatz without losing generality.

For a given b, let D and b' satisfy Eq. (21)

$$\mathrm{d}D - \epsilon(\tilde{e}D - D\tilde{c}) = b - b', \tag{92}$$

with b' understood to be in dlog-form. Adding a term $C(\epsilon)$ to the solution $\tilde{D} = D + C(\epsilon)$ solves the same equation

$$\mathrm{d}\tilde{D} - \epsilon(\tilde{e}\tilde{D} - \tilde{D}\tilde{c}) = b - \tilde{b}', \tag{93}$$

with

$$\tilde{b}' = b' + \epsilon(\tilde{e}C(\epsilon) - C(\epsilon)\tilde{c}), \tag{94}$$

which is also in dlog-form, since $\tilde{e}$ and $\tilde{c}$ are in dlog-form. This argument establishes the freedom to add terms independent of the invariants to a solution of Eq. (21). The following argument proves this to be the only possible relation between two solutions of Eq. (92). Let D_1 and D_2 satisfy Eq. (92) for a given b.

$$\mathrm{d}D_1 - \epsilon(\tilde{e}D_1 - D_1\tilde{c}) = b - b'_1, \tag{95}$$

$$\mathrm{d}D_2 - \epsilon(\tilde{e}D_2 - D_2\tilde{c}) = b - b'_2. \tag{96}$$

Then the difference $\bar{D} = D_1 - D_2$ satisfies

$$\mathrm{d}\bar{D} - \epsilon(\tilde{e}\bar{D} - \bar{D}\tilde{c}) = b'_2 - b'_1. \tag{97}$$

Let $\hat{\bar{D}} = \bar{D}\epsilon^{\tau}$ be defined such that the expansion of $\hat{\bar{D}}$ starts at the constant order. The equation for $\hat{\bar{D}}$ reads

$$\mathrm{d}\hat{\bar{D}} - \epsilon(\tilde{e}\hat{\bar{D}} - \hat{\bar{D}}\tilde{c}) = B, \tag{98}$$

with $B = \epsilon^{\tau}(b'_2 - b'_1)$, which is in dlog-form. The first order in the expansion of Eq. (98) reads

$$\mathrm{d}\hat{\bar{D}}^{(0)} = \sum_{l=1}^{N} B_l^{(0)} \mathrm{d}\log(L_l), \tag{99}$$

which integrates to

$$\hat{\bar{D}}^{(0)} = \sum_{l=1}^{N} B_l^{(0)} \log(L_l) + const. \tag{100}$$

As D_1 and D_2 are assumed to be rational, $\hat{\bar{D}}$ has to be rational as well and therefore

$$B_l^{(0)} = 0, \quad l = 1, \ldots, N, \tag{101}$$

which implies that $\hat{\bar{D}}^{(0)}$ is constant. Consider the expansion of Eq. (98) at some order $n > 0$

$$\mathrm{d}\hat{\bar{D}}^{(n)} = \left(\tilde{e}\hat{\bar{D}}^{(n-1)} - \hat{\bar{D}}^{(n-1)}\tilde{c}\right) + \sum_{l=1}^{N} B_l^{(n)} \mathrm{d}\log(L_l). \tag{102}$$

The right-hand side is in dlog-form for constant $\hat{\bar{D}}^{(n-1)}$ and therefore $\hat{\bar{D}}^{(n)}$ can only be rational if it is constant as well. This proves by induction that $\hat{\bar{D}}$ is independent of the invariants. Consequently, the difference of the solutions $\bar{D} = \hat{\bar{D}}\epsilon^{-\tau}$ has to be independent of the invariants as well. Altogether, the argument establishes the uniqueness of a rational solution for D of Eq. (21) up the addition of terms that are independent of the invariants. This fact can be used in practice to fix this freedom without losing generality.

2.7. Treatment of non-linear parameter equations

In the course of applying the algorithm, the ansatz in Eq. (16) and Eq. (18) is inserted in the expansion of Eq. (14). By requiring the resulting equations to hold for all allowed values of the invariants, a system of equations in the unknown parameters is obtained at each order of the expansion. Due to the term $\epsilon T \mathrm{d}\tilde{A}$ in Eq. (11), these equations can be non-linear. Instead of directly solving these non-linear equations, it will be shown in the following how they can be reduced to linear equations by imposing appropriate constraints.

In Section 2.6, it has been shown that the resulting canonical form is uniquely fixed up to an invertible constant transformation. Exactly this ambiguity leads to the non-linear equations, because if $\mathrm{d}\tilde{A}$ was fixed, the term $\epsilon T \mathrm{d}\tilde{A}$ would not generate non-linear equations. Therefore, the non-linear equations can be turned into linear equations by fixing the degrees of freedom in the ansatz corresponding to a subsequent invertible constant transformation. In order to fix these degrees of freedom directly, they would have to be disentangled from those which are determined by the equations in the parameters. Since this would require a parameterization of the solution set of the non-linear equations, which is essentially equivalent to solving them, a more indirect approach of fixing the freedom is taken in the following.

To this end, suppose for the moment that the parameters of the ansatz can be separated in those which are fixed by the parameter equations $\{\tau\}$ and those which correspond to the remaining freedom $\{\tau'\}$. Let $T(\epsilon, \{x_j\}, \{\tau\}, \{\tau'\})$ be a solution of Eq. (11), provided the parameters $\{\tau\}$ solve the parameter equations. According to the proof of claim 2, this transformation can be thought of as the product of some fixed transformation $T_1(\epsilon, \{x_j\}, \{\tau\})$, which transforms the original differential equation to some canonical form $\epsilon \mathrm{d}\tilde{A}_1$, and a transformation $C(\epsilon, \{\tau'\})$ parameterizing the transformation of $\mathrm{d}\tilde{A}_1$ to any other possible canonical form $\epsilon \mathrm{d}\tilde{A}_2(\tau')$

$$T(\epsilon, \{x_j\}, \{\tau\}, \{\tau'\}) = T_1(\epsilon, \{x_j\}, \{\tau\}) C(\epsilon, \{\tau'\}), \tag{103}$$

$$\mathrm{d}\tilde{A}_2(\{\tau'\}) = C(\epsilon, \{\tau'\})^{-1} \mathrm{d}\tilde{A}_1 C(\epsilon, \{\tau'\}). \tag{104}$$

It should be noted that $\tilde{A}_2$ does in general only depend on a subset of the parameters $\{\tau'\}$, because some parameters can correspond to a non-trivial ϵ-dependence of $C(\epsilon, \{\tau'\})$. As mentioned above, the goal is to fix the resulting differential equation $\mathrm{d}\tilde{A}_2$ by fixing the corresponding parameters of $\{\tau'\}$. This can be achieved by demanding $C(\epsilon, \{\tau'\})$ to equal some fixed constant invertible transformation at some non-singular value $\epsilon = \epsilon_0$. Since the left-hand side of Eq. (104) does not depend on ϵ, this completely fixes $\mathrm{d}\tilde{A}_2$ irrespective of the particular value of ϵ_0. However, fixing $C(\epsilon, \{\tau'\})$ directly would require the computation of the factorization in Eq. (103), which is only possible if the separation of the parameters into the sets $\{\tau\}$ and $\{\tau'\}$ is known. Instead, $C(\epsilon, \{\tau'\})$ can be fixed indirectly by demanding

$$T(\epsilon_0, \{x_{0j}\}, \{\tau\}, \{\tau'\}) = \mathbb{I} \tag{105}$$

to hold at some non-singular point $\{x\} = \{x_{j0}\}, \epsilon = \epsilon_0$. This is equivalent to fixing $C(\epsilon, \{\tau'\})$ as follows:

$$C(\epsilon_0, \{\tau'\}) = T_1(\epsilon_0, \{x_{0j}\}, \{\tau\})^{-1}. \tag{106}$$

The constraints given by Eq. (105) can be imposed without being able to separate the parameters into $\{\tau\}$ and $\{\tau'\}$. Moreover, these constraints are linear in both $\{\tau\}$ and $\{\tau'\}$, since the ansatz in Eq. (16) is linear in all parameters. Therefore, the additional constraints in Eq. (105) can be used to completely fix the resulting canonical form, which turns the non-linear parameter equations into linear equations.

Recall that the parameter equations are generated order by order in the expansion of Eq. (14) and at each order it is tested whether the series terminates at the current order. The constraints in Eq. (105) can only be imposed if the full $T(\epsilon, \{x_j\}, \{\tau\}, \{\tau'\})$ is known. Thus, the computation must have reached the order at which the series terminates. However, non-linear equations can already occur at lower orders in the expansion, i.e. before Eq. (105) can be imposed to turn them into linear equations. The strategy described in the following overcomes this point by essentially just solving the linear equations at each order and keeping the non-linear equations until the constraints can be imposed.

At each order in the expansion of the transformation, the linear equations are solved first and then their solution is inserted into the non-linear ones, which possibly turns some of them into linear equations. These newly generated linear equations can again be solved. This procedure is iterated until no further linear equations are generated. The remaining non-linear equations are kept unsolved. It is then tested whether the series terminates at the current order by generating the additional parameter equations corresponding to this assumption (cf. Ref. [72]). The linear equations of these additional equations are then iteratively solved as described above, while the previously obtained still unsolved non-linear equations are taken into account as well. If it turns out during this iteration that the system has no solution, the algorithm proceeds with the next order in the expansion of the transformation. If this is not the case and some non-linear equations remain at the end of the iteration, Eq. (105) is imposed. If the series does terminate at the current order, the additional constraints will turn the remaining non-linear equations into linear ones, which then determine the transformation. If either non-linear equations remain or the linear ones have no solution, it can be concluded that the series does not terminate at the current order and the algorithm proceeds with the next order in the expansion.

Altogether, this procedure allows to compute a transformation to a canonical form by only solving linear equations at each order without sacrificing the generality of the algorithm.

3. The *CANONICA* package

This section introduces the *CANONICA* package, which implements the algorithm proposed in Ref. [72]. After describing the installation and the contents of the package, the main functionality is illustrated with short usage examples. A detailed description of all functions and options of *CANONICA* can be found in the interactive manual `./manual.nb`.

3.1. Installation

CANONICA is a `Mathematica` package and requires an installation of version 10 or higher of `Mathematica`. The *CANONICA* repository can be copied to the local directory with

```
git clone https://github.com/christophmeyer/CANONICA.git
```

Alternatively, an archive file can be downloaded at

```
https://github.com/christophmeyer/CANONICA/archive/v1.0.tar.gz
```

which may be extracted with

```
tar -xvzf CANONICA-1.0.tar.gz
```

There is no further installation necessary, in particular, there are no dependencies other than `Mathematica`. In a `Mathematica` session, the package can be loaded by

```
Get["CANONICA.m"];
```

provided the file `CANONICA.m` is placed either in the current working directory or in one of the search paths. If this is not the case, `Get` either has to be called with the full path of the file `CANONICA.m`, or its location has to be added to the list of Mathematica's search paths, which is stored in the global variable `$Path`, by e.g.

```
AppendTo[$Path,"/path/to/CANONICA/src/"]
```

Changes to `$Path` can be made permanent by adding them to the initialization file `init.m`.

3.2. Files of the package

The root directory of the *CANONICA* package contains the following files and directories.

`./src/CANONICA.m` Contains all of the source code of the program, in particular, all function definitions as well as short usage messages for the public functions and options.

`./manual.nb` An interactive manual in the `Mathematica` notebook format explaining the usage of all functions and options with short examples.

`./examples` Several examples are provided in this directory. The directory of each example contains a .m file with the corresponding differential equation and a .nb notebook file illustrating the application of *CANONICA* to this example. The calculation of the full transformation can also be run in terminal mode with the script `RunExample.m`. The script is started by calling

```
math -run "<<RunExample.m"
```

or

```
math -script RunExample.m
```

Some basic information about the examples, such as the master integrals and the definition of the kinematic invariants, is provided in the `./examples/examples.pdf` file.

`./LICENSE` A copy of the third version of the GNU General Public License.

`./README` A README file providing basic information on the package.

3.3. Usage examples

In this section, the main features and their usage are illustrated with short examples. A similar, but more extensive account of this functionality can be found in the manual notebook file, which comes with the package.

The most common input required by *CANONICA* is a differential equation of the form in Eq. (1), which is determined by the differential form $a(\epsilon, \{x_j\})$. Consider the following example:

$$a(\epsilon, \{x, y\}) = \begin{pmatrix} -\frac{2+\epsilon}{x} & 0 \\ 0 & -\frac{1+\epsilon}{x} \end{pmatrix} \mathrm{d}x + \begin{pmatrix} 0 & 0 \\ \frac{(-1+\epsilon)x}{(-1+y)y} & \frac{1-\epsilon(1+y)}{(-1+y)y} \end{pmatrix} \mathrm{d}y, \tag{107}$$

depending on the invariants x and y. The differential form is represented in *CANONICA* as a list of the matrix-valued coefficients of the differentials of the invariants. The dimensional regulator ϵ has to be denoted by the protected symbol `eps`. For the above example, the input reads

```
a = {
     {{-(2+eps)/x, 0}, {0, -(1+eps)/x}}
     ,
     {{0, 0}, {((-1+eps)x)/((-1+y)y), (1-eps(1+y))/((-1+y)y)}}
    };
```

The order of the coefficient matrices has to be specified by a list of the corresponding invariants.

```
invariants = {x, y};
```

The algorithm to compute a transformation of a differential equation to canonical form as outlined in Section 2.2 and in Ref. [72] is implemented in the function `TransformDiagonalBlock`. As the name suggests, this function is intended to be used for calculating the transformation of diagonal blocks of differential equations. For the example above, the function is called as follows:

```
res=TransformDiagonalBlock[a, invariants]
```

which returns the output

```
{
    {{(1-2eps)/x^2, (1-2eps)/x^2}, {(1-eps)/x, (1-eps)/(xy)}}
    ,
    {
    {{-(eps/x), 0}, {0, -(eps/x)}}
    ,
    {{-eps/(-1+y),-eps/(y-y^2)}, {eps/(-1+y),eps/(y-y^2)}}
    }
   }
```

The output of `TransformDiagonalBlock` is a list with two entries. The first contains the transformation and the second contains the resulting differential equation in ϵ-form. The resulting differential equation is of course redundant, since it can be computed by applying the transformation to the original differential equation. However, the resulting differential equation is generated anyway in the course of the computation of the transformation and applying the transformation can be a costly operation in itself for larger differential equations.

The application of a transformation to a differential equation, according to the transformation law Eq. (5), is implemented in the function `TransformDE`, which for some transformation

```
trafo = res[[1]]
```

is called as

```
TransformDE[a, invariants, trafo]
```

and returns the resulting differential equation. In order to apply Eq. (5), the inverse of the transformation needs to be computed, which can consume significant computation time for larger matrices, when done with the build-in `Mathematica` command. However, the transformations usually exhibit a block-triangular structure, which is exploited by `TransformDE` leading to a considerably better performance.

The function `TransformDiagonalBlock` is in principle applicable to differential equations of any size. However, the performance can be improved significantly by splitting the computation according to the block-triangular structure of the differential equation and performing the computation in a recursion over the sectors of the differential equation. The main function in *CANONICA* for this purpose is `RecursivelyTransformSectors`. In addition to the two arguments related to the differential equation itself, this function expects an argument that defines the boundaries of the diagonal blocks. The differential equation in the example above actually splits into two blocks of dimension one, and in this case the boundaries read

```
boundaries = {{1, 1}, {2, 2}};
```

Each entry of the boundaries list corresponds to one diagonal block, which is specified by the position of its lowest and highest integral. Instead of using `TransformDiagonalBlock` to transform `a` into ϵ-form all at once, the following command

```
RecursivelyTransformSectors[a, invariants, boundaries, {1, 2}]
```

computes the transformation in a recursion over the sectors, as described in Section 2.2. The last argument determines the sectors at which the computation starts and ends. The output is of the same format as described above for `TransformDiagonalBlock`. If some lower sectors have already been transformed into ϵ-form and the computation should therefore not start at the first sector, the differential equation of the lower-sectors in ϵ-form and the transformation leading to it need to be provided as two additional arguments.

CANONICA also has functionality to extract the boundaries of the diagonal blocks from the differential equation. The function `SectorBoundariesFromDE` extracts the most fine grained boundaries compatible with the differential equation. For instance, in the example above

```
SectorBoundariesFromDE[a]
```

returns

```
{{1, 1}, {2, 2}}
```

The boundaries obtained in this way may be too fine for the algorithm to find the solution, since the solution space could be constrained too much by splitting the transformation into smaller blocks. It is safer to choose the boundaries according to the sector-ids of the integrals, which in general yields coarser grained boundaries. For a given list of integrals specified by their propagator powers

```
masterIntegrals={Int["T1", {0, 1, 0, 1, 0, 1, 0, 0, 0}],
                 Int["T1", {0, 1, 0, 1, 1, 1, 0, 0, 0}],
                 Int["T1", {0, 0, 1, 1, 1, 1, 0, 0, 0}],
                 Int["T1", {0, 1, 1, 1, 1, 1, 0, 0, 0}],
                 Int["T1", {1, 1, 0, 0, 0, 0, 1, 0, 0}],
                 Int["T1", {1, 1, -1, 0, 0, 0, 1, 0, 0}]};
```

the boundaries for the corresponding differential equation can be computed with

```
SectorBoundariesFromID[masterIntegrals]
```

provided the integrals are ordered with respect to their sector-id. `SectorBoundariesFromID` then returns the sector boundaries derived from the sector-ids of the integrals in the above format

```
{{1, 1}, {2, 2}, {3, 3}, {4, 4}, {5, 6}}
```

While the main function of *CANONICA* is `RecursivelyTransformSectors`, it is in some cases useful to be able to perform only certain steps of the algorithm. For this reason, there is a hierarchy of functions available in *CANONICA* allowing to break the calculation of the transformation into smaller steps. The hierarchy of these lower-level functions is illustrated in Fig. 1. For more information on specific functions, see the manual notebook included in the package.

3.4. Tests and limitations

CANONICA has been successfully tested on a variety of non-trivial single- and multi-scale problems, some of which are included as examples in the package. All tests have been performed with the `Mathematica` versions 10 and 11 on a Linux operating system. The limitations of *CANONICA* are mostly limitations of the algorithm itself. In particular, the algorithm is limited to differential equations for which a rational transformation to a canonical form exists. However, it is well known that rational differential equations may require non-rational transformations to attain a canonical form. The following example illustrates this behavior:

$$a(\epsilon, \{x\}) = \left(\frac{1}{2x} + \frac{\epsilon}{x}\right) \mathrm{d}x, \tag{108}$$

where the transformation to a canonical form is given by

$$T(\epsilon, \{x\}) = \sqrt{x}. \tag{109}$$

In this situation it is often possible to render the transformation rational with a change of coordinates. For instance, in the above example, the differential form transforms under the change of variables

$$x = y^2 \tag{110}$$

to

$$a(\epsilon, \{y\}) = \left(\frac{1}{y} + \frac{2\epsilon}{y}\right) \mathrm{d}y, \tag{111}$$

which has the following rational transformation to a canonical form

$$T(\epsilon, \{y\}) = y. \tag{112}$$

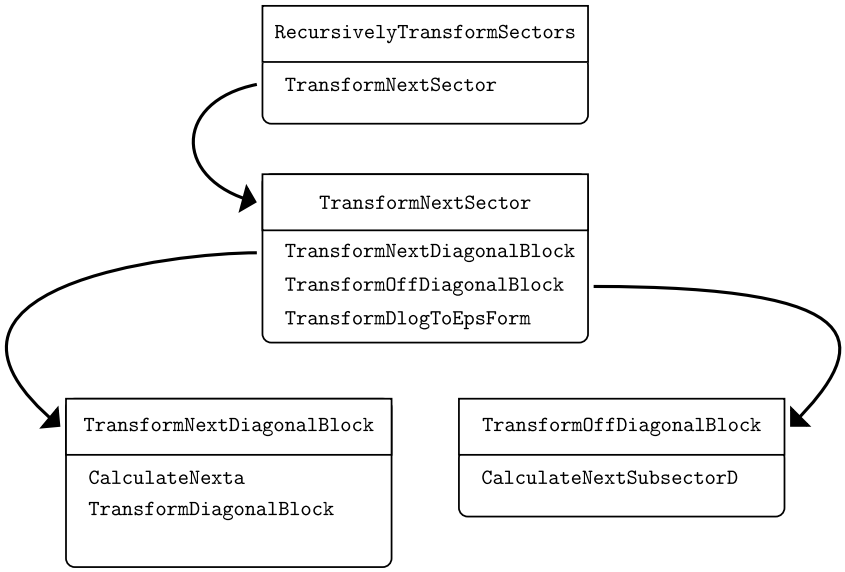

Fig. 1. Hierarchy of the main functions in *CANONICA*. Each block lists the public functions called by the function in the blocks title.

While a change of coordinates can remove non-rational letters in more complicated examples as well [6,9,11,12], this has neither been proven to be always possible, nor is a general method to construct such coordinate changes known. In fact, the existence of such a procedure appears to be unlikely, given that the number of independent roots can largely outgrow the number of variables in a problem [32].

In addition to this limitation of the algorithm, the calculations are in practice limited by the size of the available memory. This imposes limits on the size of the systems of linear parameter equations. The main factors determining the sizes of these systems of linear equations are the size of the differential equation itself and the size of the ansatz. Thus, run time and memory consumption of *CANONICA* are highly problem dependent. For instance, the most complicated example that is provided with the package is a two-loop double box topology depending on three dimensionless scales. It has a run time of about 20 min and a memory consumption of less than 8 GB.

4. Conclusion

The description of the algorithm in Ref. [72] has been extended in the present paper by providing details on the choice of an ansatz for both the diagonal blocks and the off-diagonal blocks. In both cases it has been shown that the irreducible denominator factors can be chosen from the set of irreducible denominator factors occurring in the differential equation. The ansatz for the off-diagonal blocks has been further restricted by proving upper bounds on the powers of the denominator factors in the solution.

Furthermore, canonical forms have been proven to be unique up to constant transformations, which allowed to attribute the occurrence of non-linear equations in the parameters to the freedom in the choice of this constant transformation. By fixing this freedom in a specific way, it has been argued that only linear equations need to be solved in the course of applying the algorithm.

The main focus of this publication has been the presentation of the `Mathematica` package *CANONICA*, which implements the aforementioned algorithm and allows to compute rational transformations of differential equations into canonical form. This represents the first publicly available implementation of an algorithm applicable to problems depending on multiple scales. *CANONICA* has been successfully tested on a number of state of the art multi-scale problems, including previously unknown integral topologies. *CANONICA* may thus provide a valuable contribution to the ongoing efforts to automatize multi-loop calculations.

Acknowledgments

The author would like to thank Peter Uwer for useful discussions and comments on the manuscript. This research was supported by the German Research Foundation (DFG) via the Research Training Group 1504.

Appendix A. List of functions provided by *CANONICA*

CalculateDlogForm: `CalculateDlogForm[a, invariants, alphabet]` returns a list of matrices of the same dimensions as `a`, where each matrix is the matrix-residue of one of the letters. The ordering is the same as the one in `alphabet`. Returns `False` if `a` cannot be cast in a dlog-form with the given `alphabet`.

CalculateNexta: `CalculateNexta[aFull, invariants, sectorBoundaries, trafoPrevious, aPrevious]` applies `trafoPrevious` to `aFull` and returns the differential equation of the next sector. `aPrevious` is used to recycle the transformation of lower sectors.

CalculateNextSubsectorD: `CalculateNextSubsectorD[a, invariants, sectorBoundaries, previousD]` computes the D_k of the next sector, prepends it to `previousD` and returns the result. The ansatz to be used can be specified with the optional argument `userProvidedAnsatz`. If no ansatz is provided, an ansatz is generated automatically. The size of the automatically generated ansatz can be controlled with the option `DDeltaNumeratorDegree`.

CheckDlogForm: `CheckDlogForm[a, invariants, alphabet]` tests whether the differential equation `a` is in dlog-form for the given `alphabet`. Returns either `True` or `False`.

CheckEpsForm: `CheckEpsForm[a, invariants, alphabet]` tests whether the differential equation `a` is in ϵ-form with the given `alphabet`. Returns either `True` or `False`.

CheckIntegrability: `CheckIntegrability[a, invariants]` tests whether `a` satisfies the integrability condition $\mathrm{d}a - a \wedge a = 0$ and returns either `True` or `False`.

CheckSectorBoundaries: `CheckSectorBoundaries[a, sectorBoundaries]` tests whether the `sectorBoundaries` are compatible with `a` and returns either `True` or `False`.

ExtractDiagonalBlock: `ExtractDiagonalBlock[a, boundaries]` returns the diagonal block of the differential equation `a` specified by the `boundaries` argument. `boundaries` is expected to be of the format `{nLowest, nHighest}`, where `nLowest` and `nHighest` are positive integers indicating the lowest and highest integrals of the diagonal block, respectively.

ExtractIrreducibles: `ExtractIrreducibles[a]` returns the irreducible denominator factors of `a` that do not depend on the regulator. The option `AllowEpsDependence->True` allows the irreducible factors to depend on both the invariants and the regulator.

FindAnsatzSubsectorD: `FindAnsatzSubsectorD[a, invariants, sectorBoundaries, previousD]` takes a differential equation `a`, which is required to be in ϵ-form except for the off-diagonal block of the highest sector. Needs to be provided with all previous D_k in the argument `previousD` and computes the ansatz $\mathcal{R}_D$ for the computation of the next D_k. Takes the option `DDeltaNumeratorDegree` to enlarge the ansatz. For more details, see Section 2.5.

FindAnsatzT: `FindAnsatzT[a, invariants]` takes a differential equation `a` in the `invariants` and computes an ansatz $\mathcal{R}_T$ as described in Section 2.3. The ansatz can be enlarged with the options `TDeltaNumeratorDegree` and `TDeltaDenominatorDegree`.

FindConstantNormalization: `FindConstantNormalization[invariants, trafoPrevious, aPrevious]` calculates a constant diagonal transformation to minimize the number of prime factors present in the matrix-residues. The transformation is composed with `trafoPrevious` and returned together with the resulting differential equation.

FindEpsDependentNormalization: `FindEpsDependentNormalization[a, invariants]` calculates a diagonal transformation depending only on the dimensional regulator in order to attempt to minimize the number of orders that need to be calculated in a subsequent determination of the transformation to a canonical form. Returns the transformation together with the resulting differential equation.

RecursivelyTransformSectors: `RecursivelyTransformSectors[aFull, invariants, sectorBoundaries, {nSecStart, nSecStop}]` calculates a rational transformation of `aFull` to a canonical form in a recursion over the sectors of the differential equation, which have to be specified by `sectorBoundaries`. The arguments `nSecStart` and `nSecStop` set the first and the last sector to be computed, respectively. If `nSecStart` is greater than one, the result of the calculation for the sectors lower than `nSecStart` needs to be provided in the additional arguments `trafoPrevious` and `aPrevious`. `RecursivelyTransformSectors` returns the transformation of `aFull` to a canonical form for the sectors up to `nSecStop` and the resulting differential equation. The ansaetze for the individual blocks are generated automatically. The sizes of the ansaetze for the diagonal blocks can be controlled with the options `TDeltaNumeratorDegree` and `TDeltaDenominatorDegree`. Similarly, the sizes of the ansaetze for the off-diagonal blocks are controlled by the option `DDeltaNumeratorDegree`.

SectorBoundariesFromDE: `SectorBoundariesFromDE[a]` returns the most fine grained sector boundaries compatible with `a`.

SectorBoundariesFromID: `SectorBoundariesFromID[masterIntegrals]` takes a list of `masterIntegrals`, which need to be ordered by their sector-id and returns the sector boundaries computed from the sector-ids.

TransformDE: `TransformDE[a, invariants, t]` applies the transformation `t` to the differential equation `a`. Returns $a' = t^{-1}at - t^{-1}\mathrm{d}t$. The option `SimplifyResult->False` deactivates the simplification of the result.

TransformDiagonalBlock: `TransformDiagonalBlock[a, invariants]` calculates a rational transformation to transform `a` into canonical form and returns the transformation together with the resulting differential equation. With the optional argument `userProvidedAnsatz`, the user can specify the ansatz to be used. If no ansatz is provided, an ansatz is generated automatically. The size of the automatically generated ansatz can be controlled with the options `TDeltaNumeratorDegree` and `TDeltaDenominatorDegree`.

TransformDlogToEpsForm: `TransformDlogToEpsForm[invariants, sectorBoundaries, trafoPrevious, aPrevious]` computes a transformation depending only on the regulator in order to transform `aPrevious` from dlog-form into canonical form (cf. Ref. [70]). The transformation is composed with `trafoPrevious` and returned together with the resulting differential equation. Per default, the transformation is demanded to be in a block-triangular form induced by `sectorBoundaries`. This condition can be dropped with the option `EnforceBlockTriangular->False`.

TransformNextDiagonalBlock: `TransformNextDiagonalBlock[aFull, invariants, sectorBoundaries, trafoPrevious, aPrevious]` calls `TransformDiagonalBlock` to compute the transformation of the next diagonal block into canonical form and composes it with `trafoPrevious`. Returns the composed transformation together with the resulting differential equation. With the optional argument `userProvidedAnsatz`, the user can specify the ansatz to be used. If no ansatz is provided, an ansatz is generated automatically. The size of the automatically generated ansatz can be controlled with the options `TDeltaNumeratorDegree` and `TDeltaDenominatorDegree`.

TransformNextSector: `TransformNextSector[aFull, invariants, sectorBoundaries, trafoPrevious, aPrevious]` transforms the next sector into canonical form, composes the calculated transformation with `trafoPrevious` and returns it together with the resulting differential equation. With the optional argument `userProvidedAnsatz`, the user can specify the ansatz to be used for the diagonal block. If no ansatz is provided, an ansatz is generated automatically. The size of the automatically generated ansatz for the diagonal block can be controlled with the options `TDeltaNumeratorDegree` and `TDeltaDenominatorDegree`. Similarly, the sizes of the ansaetze for the off-diagonal blocks are controlled by the option `DDeltaNumeratorDegree`.

TransformOffDiagonalBlock: `TransformOffDiagonalBlock[invariants, sectorBoundaries, trafoPrevious, aPrevious]` assumes `aPrevious` to be in canonical form except for the highest sector of which only the diagonal block is assumed to be in canonical form. Computes a transformation to transform the off-diagonal block of the highest sector into dlog-form. This transformation is composed with `trafoPrevious` and returned together with the resulting differential equation. Proceeds in a recursion over sectors, which can be resumed at an intermediate step by providing all previous D_k (cf. Ref. [72]) in the optional argument `userProvidedD`. The sizes of the automatically generated ansaetze for the off-diagonal blocks are controlled by the option `DDeltaNumeratorDegree`.

Appendix B. List of options

AllowEpsDependence: `AllowEpsDependence` is an option of `ExtractIrreducibles` controlling whether irreducible factors depending on both the invariants and the regulator are returned as well. The default value is `False`.

DDeltaNumeratorDegree: `DDeltaNumeratorDegree` is an option controlling the numerator powers of the rational functions in the ansatz used for the computation of D for the transformation of off-diagonal blocks. The default value is `0`. For more details, see Section 2.5. `DDeltaNumeratorDegree` is an option of the following functions: `CalculateNextSubsectorD`, `FindAnsatzSubsectorD`, `RecursivelyTransformSectors`, `TransformNextSector`, `TransformOffDiagonalBlock`.

EnforceBlockTriangular: `EnforceBlockTriangular` is an option of `TransformDlogToEpsForm` controlling whether the resulting transformation is demanded to be in the block-triangular form induced by the `sectorBoundaries` argument. The default value is `True`.

FinalConstantNormalization: `FinalConstantNormalization` is an option of `RecursivelyTransformSectors` controlling whether `FindConstantNormalization` is invoked after all sectors have been transformed into canonical form in order to simplify the resulting canonical form. The default value is `False`.

PreRescale: `PreRescale` is an option of `TransformDiagonalBlock` controlling whether `FindEpsDependentNormalization` is called prior to the main computation in order to attempt to minimize the number of orders that need to be calculated in a subsequent determination of the transformation to a canonical form. The default value is `True`.

SimplifyResult: `SimplifyResult` is an option of `TransformDE` controlling whether the resulting differential equation is simplified. The default value is `True`.

TDeltaDenominatorDegree: `TDeltaDenominatorDegree` is an option controlling the denominator powers of the rational functions in the ansatz used for the computation of the transformation of diagonal blocks. The default value is `0`. For more details, see Section 2.3. `TDeltaDenominatorDegree` is an option of the following functions: `FindAnsatzT`, `RecursivelyTransformSectors`, `TransformNextDiagonalBlock`, `TransformNextSector`.

TDeltaNumeratorDegree: `TDeltaNumeratorDegree` is an option controlling the numerator powers of the rational functions in the ansatz used for the computation of the transformation of diagonal blocks. The default value is `0`. For more details, see Section 2.3. `TDeltaNumeratorDegree` is an option of the following functions: `FindAnsatzT`, `RecursivelyTransformSectors`, `TransformNextDiagonalBlock`, `TransformNextSector`.

VerbosityLevel: `VerbosityLevel` is an option controlling the verbosity of several main functions. Takes integer values from `0` to `12` with a value of `12` resulting in the most detailed output about the current state of the computation and a value of `0` suppressing all output but warnings about inconsistent inputs. The default value is `10`. The following functions accept the `VerbosityLevel` option: `CalculateNextSubsectorD`, `FindConstantNormalization`, `RecursivelyTransformSectors`, `TransformDiagonalBlock`, `TransformDlogToEpsForm`, `TransformNextDiagonalBlock`, `TransformNextSector`, `TransformOffDiagonalBlock`.

Appendix C. List of global variables and protected symbols

$ComputeParallel: `$ComputeParallel` is a global variable that needs to be set to `True` to enable parallel computations. The number of kernels to be used is controlled by `$NParallelKernels`.

$NParallelKernels: `$NParallelKernels` is a global variable setting the number of parallel kernels to be used. `$NParallelKernels` has no effect if `$ComputeParallel` is not set to `True`. If `$ComputeParallel` is `True` and `$NParallelKernels` is not assigned a value, then all available kernels are used for the computation.

eps: `eps` is a protected symbol representing the dimensional regulator.

References

[1] A.V. Kotikov, Phys. Lett. B 254 (1991) 158–164. http://dx.doi.org/10.1016/0370-2693(91)90413-K.
[2] E. Remiddi, Nuovo Cimento A 110 (1997) 1435–1452 arXiv:hep-th/9711188.
[3] T. Gehrmann, E. Remiddi, Nuclear Phys. B 580 (2000) 485–518. http://dx.doi.org/10.1016/S0550-3213(00)00223-6. arXiv:hep-ph/9912329.
[4] J.M. Henn, Phys. Rev. Lett. 110 (25) (2013) 251601. http://dx.doi.org/10.1103/PhysRevLett.110.251601. arXiv:1304.1806.
[5] J.M. Henn, A.V. Smirnov, V.A. Smirnov, J. High Energy Phys. 07 (2013) 128. http://dx.doi.org/10.1007/JHEP07(2013)128. arXiv:1306.2799.
[6] J.M. Henn, V.A. Smirnov, J. High Energy Phys. 11 (2013) 041. http://dx.doi.org/10.1007/JHEP11(2013)041. arXiv:1307.4083.
[7] J.M. Henn, A.V. Smirnov, V.A. Smirnov, J. High Energy Phys. 03 (2014) 088. http://dx.doi.org/10.1007/JHEP03(2014)088. arXiv:1312.2588.
[8] M. Argeri, S.D. Vita, P. Mastrolia, E. Mirabella, J. Schlenk, U. Schubert, L. Tancredi, J. High Energy Phys. 03 (2014) 082. http://dx.doi.org/10.1007/JHEP03(2014)082. arXiv:1401.2979.

[9] J.M. Henn, K. Melnikov, V.A. Smirnov, J. High Energy Phys. 05 (2014) 090. http://dx.doi.org/10.1007/JHEP05(2014)090. arXiv:1402.7078, http://dx.doi.org/10.1007/s13130-014-8200-x.
[10] S. Caron-Huot, J.M. Henn, J. High Energy Phys. 06 (2014) 114. http://dx.doi.org/10.1007/JHEP06(2014)114. arXiv:1404.2922.
[11] T. Gehrmann, A. von Manteuffel, L. Tancredi, E. Weihs, J. High Energy Phys. 06 (2014) 032. http://dx.doi.org/10.1007/JHEP06(2014)032. arXiv:1404.4853.
[12] F. Caola, J.M. Henn, K. Melnikov, V.A. Smirnov, J. High Energy Phys. 09 (2014) 043. http://dx.doi.org/10.1007/JHEP09(2014)043. arXiv:1404.5590.
[13] Y. Li, A. von Manteuffel, R.M. Schabinger, H.X. Zhu, Phys. Rev. D 90 (5) (2014) 053006. http://dx.doi.org/10.1103/PhysRevD.90.053006. arXiv:1404.5839.
[14] M. Höschele, J. Hoff, T. Ueda, J. High Energy Phys. 09 (2014) 116. http://dx.doi.org/10.1007/JHEP09(2014)116. arXiv:1407.4049.
[15] S.D. Vita, P. Mastrolia, U. Schubert, V. Yundin, J. High Energy Phys. 09 (2014) 148. http://dx.doi.org/10.1007/JHEP09(2014)148. arXiv:1408.3107.
[16] A. von Manteuffel, R.M. Schabinger, H.X. Zhu, Phys. Rev. D 92 (4) (2015) 045034. http://dx.doi.org/10.1103/PhysRevD.92.045034. arXiv:1408.5134.
[17] A. Grozin, J.M. Henn, G.P. Korchemsky, P. Marquard, Phys. Rev. Lett. 114 (6) (2015) 062006. http://dx.doi.org/10.1103/PhysRevLett.114.062006. arXiv:1409.0023.
[18] G. Bell, T. Huber, J. High Energy Phys. 12 (2014) 129. http://dx.doi.org/10.1007/JHEP12(2014)129. arXiv:1410.2804.
[19] T. Huber, S. Kränkl, J. High Energy Phys. 04 (2015) 140. http://dx.doi.org/10.1007/JHEP04(2015)140. arXiv:1503.00735.
[20] T. Gehrmann, A. von Manteuffel, L. Tancredi, J. High Energy Phys. 09 (2015) 128. http://dx.doi.org/10.1007/JHEP09(2015)128. arXiv:1503.04812.
[21] T. Gehrmann, S. Guns, D. Kara, J. High Energy Phys. 09 (2015) 038. http://dx.doi.org/10.1007/JHEP09(2015)038. arXiv:1505.00561.
[22] R. Bonciani, V. De. Duca, H. Frellesvig, J.M. Henn, F. Moriello, V.A. Smirnov, J. High Energy Phys. 08 (2015) 108. http://dx.doi.org/10.1007/JHEP08(2015)108. arXiv:1505.00567.
[23] C. Anzai, A. Hasselhuhn, M. Höschele, J. Hoff, W. Kilgore, M. Steinhauser, T. Ueda, J. High Energy Phys. 07 (2015) 140. http://dx.doi.org/10.1007/JHEP07(2015)140. arXiv:1506.02674.
[24] A. Grozin, J.M. Henn, G.P. Korchemsky, P. Marquard, J. High Energy Phys. 01 (2016) 140. http://dx.doi.org/10.1007/JHEP01(2016)140. arXiv:1510.07803.
[25] T. Gehrmann, J.M. Henn, N.A. Lo Presti, Phys. Rev. Lett. 116 (6) (2016) 062001. http://dx.doi.org/10.1103/PhysRevLett.116.062001. arXiv:1511.05409, [Erratum: Phys. Rev. Lett. 116(18) (2016) 189903, http://dx.doi.org/10.1103/PhysRevLett.116.189903].
[26] O. Gituliar, J. High Energy Phys. 02 (2016) 017. http://dx.doi.org/10.1007/JHEP02(2016)017. arXiv:1512.02045.
[27] R.N. Lee, K.T. Mingulov, Phys. Lett. B 757 (2016) 207–210. http://dx.doi.org/10.1016/j.physletb.2016.03.083. arXiv:1602.02463.
[28] J.M. Henn, A.V. Smirnov, V.A. Smirnov, M. Steinhauser, J. High Energy Phys. 05 (2016) 066. http://dx.doi.org/10.1007/JHEP05(2016)066. arXiv:1604.03126.
[29] R. Bonciani, S. Di Vita, P. Mastrolia, U. Schubert, J. High Energy Phys. 09 (2016) 091. http://dx.doi.org/10.1007/JHEP09(2016)091. arXiv:1604.08581.
[30] B. Eden, V.A. Smirnov, J. High Energy Phys. 10 (2016) 115. http://dx.doi.org/10.1007/JHEP10(2016)115. arXiv:1607.06427.
[31] R.N. Lee, V.A. Smirnov, J. High Energy Phys. 10 (2016) 089. http://dx.doi.org/10.1007/JHEP10(2016)089. arXiv:1608.02605.
[32] R. Bonciani, V. Del Duca, H. Frellesvig, J.M. Henn, F. Moriello, V.A. Smirnov, J. High Energy Phys. 12 (2016) 096. http://dx.doi.org/10.1007/JHEP12(2016)096. arXiv:1609.06685.
[33] M. Bonetti, K. Melnikov, L. Tancredi, Nuclear Phys. B 916 (2017) 709–726. http://dx.doi.org/10.1016/j.nuclphysb.2017.01.020. arXiv:1610.05497.
[34] J.M. Henn, A.V. Smirnov, V.A. Smirnov, J. High Energy Phys. 12 (2016) 144. http://dx.doi.org/10.1007/JHEP12(2016)144. arXiv:1611.06523.
[35] J. Henn, A.V. Smirnov, V.A. Smirnov, M. Steinhauser, R.N. Lee, J. High Energy Phys. 03 (2017) 139. http://dx.doi.org/10.1007/JHEP03(2017)139. arXiv:1612.04389.
[36] S. Di Vita, P. Mastrolia, A. Primo, U. Schubert, J. High Energy Phys. 04 (2017) 008. http://dx.doi.org/10.1007/JHEP04(2017)008. arXiv:1702.07331.
[37] R.H. Boels, T. Huber, G. Yang, The four-loop non-planar cusp anomalous dimension in N = 4 SYM, arXiv:1705.03444.
[38] R.N. Lee, A.V. Smirnov, V.A. Smirnov, M. Steinhauser, Phys. Rev. D 96 (1) (2017) 014008. http://dx.doi.org/10.1103/PhysRevD.96.014008. arXiv:1705.06862.
[39] K.-T. Chen, Bull. Amer. Math. Soc. 83 (1977) 831–879. http://dx.doi.org/10.1090/S0002-9904-1977-14320-6.
[40] A.B. Goncharov, Math. Res. Lett. 5 (1998) 497–516. http://dx.doi.org/10.4310/MRL.1998.v5.n4.a7. arXiv:1105.2076.
[41] M. Caffo, H. Czyz, S. Laporta, E. Remiddi, Nuovo Cimento A 111 (1998) 365–389 arXiv:hep-th/9805118.
[42] S. Laporta, E. Remiddi, Nuclear Phys. B 704 (2005) 349–386. http://dx.doi.org/10.1016/j.nuclphysb.2004.10.044. arXiv:hep-ph/0406160.
[43] S. Bloch, P. Vanhove, J. Number Theory 148 (2015) 328–364. http://dx.doi.org/10.1016/j.jnt.2014.09.032. arXiv:1309.5865.
[44] L. Adams, C. Bogner, S. Weinzierl, J. Math. Phys. 55 (10) (2014) 102301. http://dx.doi.org/10.1063/1.4896563. arXiv:1405.5640.
[45] S. Bloch, M. Kerr, P. Vanhove, Compos. Math. 151 (2015) 2329–2375. http://dx.doi.org/10.1112/S0010437X15007472. arXiv:1406.2664.
[46] L. Adams, C. Bogner, S. Weinzierl, J. Math. Phys. 56 (7) (2015) 072303. http://dx.doi.org/10.1063/1.4926985. arXiv:1504.03255.
[47] S. Bloch, M. Kerr, P. Vanhove, Local mirror symmetry and the sunset Feynman integral, arXiv:1601.08181.
[48] E. Remiddi, L. Tancredi, Nuclear Phys. B 907 (2016) 400–444. http://dx.doi.org/10.1016/j.nuclphysb.2016.04.013. arXiv:1602.01481.
[49] L. Adams, C. Bogner, A. Schweitzer, S. Weinzierl, J. Math. Phys. 57 (2016) 122302. http://dx.doi.org/10.1063/1.4969060. arXiv:1607.01571.
[50] A. Primo, L. Tancredi, Nuclear Phys. B 916 (2017) 94–116. http://dx.doi.org/10.1016/j.nuclphysb.2016.12.021. arXiv:1610.08397.
[51] H. Frellesvig, C.G. Papadopoulos, J. High Energy Phys. 04 (2017) 083. http://dx.doi.org/10.1007/JHEP04(2017)083. arXiv:1701.07356.
[52] J. Bosma, M. Sogaard, Y. Zhang, Maximal Cuts in Arbitrary Dimension, arxiv.org/abs/1704.04255.
[53] A. Primo, L. Tancredi, Maximal cuts and differential equations for Feynman integrals. An application to the three-loop massive banana graph, arxiv.org/abs/1704.05465.
[54] M. Harley, F. Moriello, R.M. Schabinger, Baikov-Lee Representations Of Cut Feynman Integrals, arXiv:1705.03478.
[55] M. Zeng, Differential equations on unitarity cut surfaces, arxiv.org/abs/1702.02355.
[56] L. Adams, S. Weinzierl, Feynman integrals and iterated integrals of modular forms, arXiv:1704.08895.
[57] S. Laporta, Internat. J. Modern Phys. A 15 (2000) 5087–5159. http://dx.doi.org/10.1016/S0217-751X(00)00215-7. arXiv:hep-ph/0102033.
[58] R.N. Lee, J. Phys. Conf. Ser. 523 (2014) 012059. http://dx.doi.org/10.1088/1742-6596/523/1/012059. arXiv:1310.1145.
[59] F.V. Tkachov, Phys. Lett. B 100 (1981) 65–68. http://dx.doi.org/10.1016/0370-2693(81)90288-4.
[60] K.G. Chetyrkin, F.V. Tkachov, Nuclear Phys. B192 (1981) 159–204. http://dx.doi.org/10.1016/0550-3213(81)90199-1.
[61] C. Anastasiou, A. Lazopoulos, J. High Energy Phys. 07 (2004) 046. http://dx.doi.org/10.1088/1126-6708/2004/07/046. arXiv:hep-ph/0404258.
[62] C. Studerus, Comput. Phys. Comm. 181 (2010) 1293–1300. http://dx.doi.org/10.1016/j.cpc.2010.03.012. arXiv:0912.2546.
[63] A. von Manteuffel, C. Studerus, Reduze 2 - Distributed Feynman Integral Reduction, arXiv:1201.4330.
[64] R.N. Lee, Presenting LiteRed: A tool for the Loop InTEgrals REDuction, arxiv.org/abs/1212.2685.
[65] A.V. Smirnov, V.A. Smirnov, Comput. Phys. Comm. 184 (2013) 2820–2827. http://dx.doi.org/10.1016/j.cpc.2013.06.016. arXiv:1302.5885.
[66] A.V. Smirnov, Comput. Phys. Comm. 189 (2015) 182–191. http://dx.doi.org/10.1016/j.cpc.2014.11.024. arXiv:1408.2372.
[67] K.J. Larsen, Y. Zhang, Phys. Rev. D 93 (4) (2016) 041701. http://dx.doi.org/10.1103/PhysRevD.93.041701. arXiv:1511.01071.
[68] A. Georgoudis, K.J. Larsen, Y. Zhang, Azurite: An algebraic geometry based package for finding bases of loop integrals, arxiv.org/abs/1612.04252.
[69] P. Maierhoefer, J. Usovitsch, P. Uwer, Kira - A Feynman Integral Reduction Program, arXiv:1705.05610.
[70] R.N. Lee, J. High Energy Phys. 1504 (2015) 108. http://dx.doi.org/10.1007/JHEP04(2015)108. arXiv:1411.0911.
[71] J.M. Henn, J. Phys. A 48 (2015) 153001. http://dx.doi.org/10.1088/1751-8113/48/15/153001. arXiv:1412.2296.
[72] C. Meyer, J. High Energy Phys. 04 (2017) 006. http://dx.doi.org/10.1007/JHEP04(2017)006. arXiv:1611.01087.
[73] L. Adams, E. Chaubey, S. Weinzierl, Phys. Rev. Lett. 118 (14) (2017) 141602. http://dx.doi.org/10.1103/PhysRevLett.118.141602. arXiv:1702.04279.
[74] O. Gituliar, V. Magerya, Fuchsia and master integrals for splitting functions from differential equations in QCD, in: 13th DESY Workshop on Elementary Particle Physics: Loops and Legs in Quantum Field Theory (LL2016) Leipzig, Germany, April 24–29, 2016, 2016, arXiv:1607.00759.
[75] M. Prausa, epsilon: A tool to find a canonical basis of master integrals, arxiv.org/abs/1701.00725.
[76] O. Gituliar, V. Magerya, Fuchsia: A tool for reducing differential equations for Feynman master integrals to epsilon form, arxiv.org/abs/1701.04269.
[77] E.K. Leinartas, Izv. Vyssh. Uchebn. Zaved. Mat. 22(10).
[78] A. Raichev, Leinartas's partial fraction decomposition, arXiv e-prints arxiv.org/abs/1206.4740.