`epsilon`: A tool to find a canonical basis of master integrals

Mario Prausa

Institute for Theoretical Particle Physics and Cosmology

RWTH Aachen University

52056 Aachen, Germany

`prausa@physik.rwth-aachen.de`

Abstract

In 2013, Henn proposed a special basis for a certain class of master integrals, which are expressible in terms of iterated integrals. In this basis, the master integrals obey a differential equation, where the right hand side is proportional to ϵ in $d = 4 - 2\epsilon$ space-time dimensions. An algorithmic approach to find such a basis was found by Lee. We present the tool `epsilon`, an efficient implementation of Lee's algorithm based on the `Fermat` computer algebra system as computational backend.

Program Summary

[1] R. N. Lee, JHEP 1504 (2015) 108 [arXiv:1411.0911 [hep-ph]].

arXiv:1701.00725v3 [hep-ph] 28 Jun 2017

1 Introduction

The perturbative treatment of quantum field theories leads quite naturally to the problem of evaluating a large number of multi-loop Feynman diagrams. After a tensor reduction the Feynman diagrams can be expressed in an even larger number of scalar Feynman integrals of the form

$$\int d^d l_1 \cdots \int d^d l_L \frac{1}{D_1^{n_1} \dots D_N^{n_N}} \,, \tag{1.1}$$

where L is the number of loops and $d = 4 - 2\epsilon$ the number of space-time dimensions in the context of dimensional regularization. The denominators D_i in (1.1) are usually of the form $p^2 - m^2$, where p is a linear combination of loop momenta and external momenta and m some mass.

A standard technique nowadays is the usage of integration-by-parts identities [1, 2] for the reduction of this large number of Feynman integrals to a rather small set of so-called master integrals. These identities provide linear dependences between various Feynman integrals, where the coefficients are rational functions in both the space-time dimension d and the kinematic variables of the problem.

Many methods were developed to solve these master integrals. For an overview see e.g. [3]. Among the most successful ones is the method of differential equations which is also based on integration-by-parts reductions[4–6]. Recently, significant progress was made in this method, when Henn conjectured the existence of a canonical basis for master integrals expressible in terms of iterated integrals [7]. In this basis the right hand side of the system of differential equations is proportional to $\epsilon = (4 - d)/2$. If the boundary conditions are known, the solution of the system of differential equations in an ϵ-series becomes trivial.

Two years ago, Lee proposed an algorithm to automate finding a canonical basis [8]. A first implementation for this algorithm was presented in [9, 10].

In this paper we present `epsilon`, a further implementation of Lee's algorithm based on the `Fermat`[11] computer algebra system. Our implementation utilizes the explicit dependence of the transformations used by Lee's algorithm on the kinematic variable to reduce the number of variables in intermediate steps. Another advantage of our implementation is the support of systems with singularities at complex points using `Fermat`'s polymod capability.

In Section 2 we introduce some definitions and explain implementation details. In Section 3 the installation procedure and the usage of `epsilon` is described. In Section 4 we give a non-trivial example of the usage based on a real three-loop computation.

2 Implementation details

2.1 Definitions

We consider a set of N master integrals $\vec{f}$ fulfilling an ordinary system of differential equations

$$\frac{\partial \vec{f}(x, \epsilon)}{\partial x} = \mathbb{M}(x, \epsilon) \vec{f}(x, \epsilon), \tag{2.1}$$

where x is a kinematic variable, $\mathbb{M}(x, \epsilon)$ is an $N \times N$-matrix and ϵ is a regulator in $d = 4 - 2\epsilon$ dimensions in the context of dimensional regularization. We restrict ourselves to the case

$$\mathbb{M}(x, \epsilon) = \sum_{x_j \in S} \sum_{k \geq 0} \frac{\mathbb{M}_k^{(x_j)}(\epsilon)}{(x - x_j)^{k+1}} + \sum_{k \geq 0} x^k \mathbb{M}_k(\epsilon), \tag{2.2}$$

where S is the set of all finite singularities and $\mathbb{M}_k^{(x_j)}$ and $\mathbb{M}_k(\epsilon)$ are independent of x. In particular, singularities x_j depending on ϵ are forbidden. In many physically relevant cases one can use a trial and error approach to find a basis of master integrals $\vec{f}$ fulfilling the restriction (2.2). The main strategy of our implementation is to keep the system always in the form of (2.2) since here the x-dependence is explicit.

A singularity $x_j < \infty$ has Poincaré rank p if $\mathbb{M}_p^{(x_j)} \neq 0$ and $\mathbb{M}_k^{(x_j)} = 0$ for $k > p$. In addition to the finite singularities, the system might also have a singularity at ∞. The Poincaré rank p of a singularity at ∞ is defined as the Poincaré rank of the singularity at $y = 0$ of the system $\mathbb{M}(1/y, \epsilon)/y^2$. So (2.2) has Poincaré rank $p > 0$ at ∞ if $\mathbb{M}_{p-1} \neq 0$ and $\mathbb{M}_k = 0$ for $k \geq p$, and Poincaré rank $p = 0$ at ∞ if all $\mathbb{M}_k = 0$ and $\sum_{x_j \in S} \mathbb{M}_0^{(x_j)} \neq 0$. If all $\mathbb{M}_k = 0$ and $\sum_{x_j \in S} \mathbb{M}_0^{(x_j)} = 0$, the system is not singular at ∞.

Let p be the Poincaré rank of a singularity $x_j < \infty$, then the generalized Poincaré rank (or Moser rank) [12] of this singularity is defined as $p + r/n - 1$, where $r = \text{rank} \, \mathbb{M}_p^{(x_j)}$ and n is the dimension of the system.

A system

$$\mathbb{M}(x, \epsilon) = \sum_{x_j \in S} \frac{\mathbb{M}_0^{(x_j)}(\epsilon)}{x - x_j}, \tag{2.3}$$

where all singularities have Poincaré rank zero is called Fuchsian, and a system

$$\mathbb{M}(x, \epsilon) = \epsilon \sum_{x_j \in S} \frac{\widehat{\mathbb{M}}_0^{(x_j)}}{x - x_j}, \tag{2.4}$$

3

where $\widehat{\mathbb{M}}_0^{(x_j)}$ is no longer a function of ϵ, is said to be in ϵ-form. A change of basis

$$\vec{g}(x, \epsilon) = \mathbb{T}^{-1}(x, \epsilon)\vec{f}$$

modifies the system (2.1) to

$$\frac{\partial \vec{g}(x, \epsilon)}{\partial x} = \widetilde{\mathbb{M}}(x, \epsilon)\vec{g}(x, \epsilon),$$

with

$$\widetilde{\mathbb{M}}(x, \epsilon) = \mathbb{T}^{-1}(x, \epsilon)\mathbb{M}(x, \epsilon)\mathbb{T}(x, \epsilon) - \mathbb{T}^{-1}(x, \epsilon)\frac{\partial}{\partial x}\mathbb{T}(x, \epsilon). \tag{2.5}$$

We assume the master integrals in $\vec{f}$ to be ordered in a way that a block-triangular structure of the system is obtained (for details see e.g. [8]). We will often make use of this block-triangular structure. Therefore we write

$$\mathbb{M} = \begin{pmatrix} \mathbb{A} & 0 & 0 \\ \mathbb{B} & \mathbb{C} & 0 \\ \mathbb{D} & \mathbb{E} & \mathbb{F} \end{pmatrix}, \tag{2.6}$$

and use the same indices as in (2.2) for the matrices $\mathbb{A}, \ldots, \mathbb{F}$ (e.g. $\mathbb{C}_k^{(x_j)}(\epsilon)$). The block $\mathbb{C}$ is called the active block as we apply Lee's algorithm to this block. As $\mathbb{A}$ to $\mathbb{F}$ are matrices as well, the definition of what we call the active block is more or less arbitrary as long as a block-triangular structure is obtained. But from a computational point of view a small dimension of the active block is preferable since this reduces the complexity of the resulting operations. In the following, the matrices $\mathbb{A}_k^{(x_j)}, \ldots, \mathbb{F}_k^{(x_j)}$ and $\mathbb{A}_k, \ldots, \mathbb{F}_k$ will be referred to as coefficient matrices.

2.2 Utilizing the explicit x-dependence

Lee's algorithm uses three types of transformations: balances, off-diagonal reductions and x-independent transformations.

We define balances as

$$\mathcal{B}(\mathbb{P}, x_1, x_2) = \overline{\mathbb{P}} + \frac{x - x_2}{x - x_1}\mathbb{P}, \tag{2.7a}$$

$$\mathcal{B}(\mathbb{P}, x_1, \infty) = \overline{\mathbb{P}} + \frac{1}{x - x_1}\mathbb{P}, \tag{2.7b}$$

$$\mathcal{B}(\mathbb{P}, \infty, x_2) = \overline{\mathbb{P}} + (x - x_2)\mathbb{P}, \tag{2.7c}$$

4

where $\mathbb{P}$ is a projector to be specified below, depending only on ϵ, $\overline{\mathbb{P}} = \mathbb{1} - \mathbb{P}$ and $x_1, x_2 < \infty$. In Lee's algorithm balances are applied to the active block $\mathbb{C}$ in order to reduce the generalized Poincaré rank of singular points and to normalize eigenvalues of Fuchsian singularities.

Off-diagonal reductions are used to reduce the block $\mathbb{B}$ to Fuchsian form after the blocks $\mathbb{A}$ and $\mathbb{C}$ were already reduced to ϵ-form. They are defined by

$$\mathcal{L}(x_1, k, \mathbb{G}) = \mathbb{1} + \frac{1}{(x - x_1)^k}\mathbb{G}, \tag{2.8a}$$

$$\mathcal{L}(\infty, k, \mathbb{G}) = \mathbb{1} + x^k\mathbb{G}, \tag{2.8b}$$

where

$$\mathbb{G} = \begin{pmatrix} 0 & 0 & 0 \\ \widehat{\mathbb{G}} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.8c}$$

The block $\widehat{\mathbb{G}}$ has the same boundaries in $\mathbb{G}$ as block $\mathbb{B}$ in (2.6). Note that $\mathbb{G}^2 = 0$.

In both types of transformations the x-dependence is explicit. Another type of transformation which is independent of x is used in the last step of Lee's algorithm to factor out ϵ.

Our goal is to use those three types in the transformation rule (2.5) without spoiling the form (2.2) or the block-triangular structure (2.6) of the system.

As a pedagogical example we consider the transformation of block $\mathbb{B}$ under a balance between two singularities x_1 and x_2, i.e.

$$\mathbb{T} = \mathcal{B}(\mathbb{P}, x_1, x_2) = \overline{\mathbb{P}} + \frac{x - x_2}{x - x_1}\mathbb{P}, \quad \mathbb{T}^{-1} = \mathcal{B}(\mathbb{P}, x_2, x_1) = \overline{\mathbb{P}} + \frac{x - x_1}{x - x_2}\mathbb{P}. \tag{2.9}$$

Since we want to apply Lee's algorithm to the active block we can restrict the form of the projector $\mathbb{P}$ to

$$\mathbb{P} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbb{Q} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.10}$$

where $\mathbb{Q}$ is a projector with the dimensions of the active block. Inserting (2.9) into (2.5)

we obtain

$$\widetilde{\mathbb{M}}(x,\epsilon) = \left[\overline{\mathbb{P}} + \frac{x-x_1}{x-x_2}\mathbb{P}\right]\mathbb{M}(x,\epsilon)\left[\overline{\mathbb{P}} + \frac{x-x_2}{x-x_1}\mathbb{P}\right] + \frac{x_2-x_1}{(x-x_1)(x-x_2)}\mathbb{P}$$

$$= \mathbb{M}(x,\epsilon) - \mathbb{P}\mathbb{M}(x,\epsilon)\overline{\mathbb{P}} - \overline{\mathbb{P}}\mathbb{M}(x,\epsilon)\mathbb{P} + \frac{x-x_2}{x-x_1}\overline{\mathbb{P}}\mathbb{M}(x,\epsilon)\mathbb{P} + \frac{x-x_1}{x-x_2}\mathbb{P}\mathbb{M}(x,\epsilon)\overline{\mathbb{P}}$$

$$+ \frac{x_2-x_1}{(x-x_1)(x-x_2)}\mathbb{P}$$

So block $\mathbb{B}$ in (2.6) transforms as

$$\widetilde{\mathbb{B}}(x,\epsilon) = \mathbb{B}(x,\epsilon) - \mathbb{Q}\mathbb{B}(x,\epsilon) + \frac{x-x_1}{x-x_2}\mathbb{Q}\mathbb{B}(x,\epsilon)\,.$$

Inserting the form (2.2) of $\mathbb{B}(x,\epsilon)$ leads to

$$\widetilde{\mathbb{B}}(x,\epsilon) = \mathbb{B}(x,\epsilon) - \mathbb{Q}\mathbb{B}(x,\epsilon) + \sum_{x_j\in S}\sum_{k\geq 0}\frac{(x-x_1)\mathbb{Q}\mathbb{B}_k^{(x_j)}(\epsilon)}{(x-x_2)(x-x_j)^{k+1}} + \sum_{k\geq 0}\frac{(x-x_1)x^k}{x-x_2}\mathbb{Q}\mathbb{B}_k(\epsilon)\,.$$

$$(2.11)$$

Using partial fractioning and the incomplete geometric series, we can show that

$$\sum_{k=0}^{\infty}\frac{a_k}{(x-x_2)(x-x_j)^{k+1}} = \frac{1}{x-x_2}\sum_{n\geq 0}\frac{a_n}{(x_2-x_j)^{n+1}}$$

$$(2.12a)$$

$$-\sum_{k\geq 0}\frac{1}{(x-x_j)^{k+1}}\sum_{n=0}^{\infty}\frac{a_{n+k}}{(x_2-x_j)^{n+1}}\,,$$

$$\sum_{k\geq 0}\frac{x^k}{x-x_2}a_k = \sum_{k\geq 0}x^k\sum_{n\geq 0}x_2^n\, a_{k+n+1} + \frac{1}{x-x_2}\sum_{n\geq 0}x_2^n\, a_n\,, \qquad (2.12b)$$

where identity (2.12a) only holds for $x_j \neq x_2$. Combining (2.11) and (2.12) yields

$$\widetilde{\mathbb{B}}(x,\epsilon) = \mathbb{B}(x,\epsilon) + \frac{x_2-x_1}{x-x_2}\sum_{x_j\in S\backslash\{x_2\}}\sum_{n\geq 0}\frac{\mathbb{Q}\mathbb{B}_n^{(x_j)}}{(x_2-x_j)^{n+1}} + \frac{x_2-x_1}{x-x_2}\sum_{n\geq 0}x_2^n\mathbb{Q}\mathbb{B}_n$$

$$+ \sum_{k\geq 1}\frac{(x_2-x_1)\mathbb{Q}\mathbb{B}_{k-1}^{(x_2)}(\epsilon)}{(x-x_2)^{k+1}} + \sum_{x_j\in S\backslash\{x_2\}}\sum_{k\geq 0}\frac{x_1-x_2}{(x-x_j)^{k+1}}\sum_{n\geq 0}\frac{\mathbb{Q}\mathbb{B}_{n+k}^{(x_j)}(\epsilon)}{(x_2-x_j)^{n+1}} \qquad (2.13)$$

$$+ (x_2-x_1)\sum_{k\geq 0}x^k\sum_{n\geq 0}x_2^n\mathbb{Q}\mathbb{B}_{k+n+1}(\epsilon)\,.$$

6

Hence, the transformation laws for the coefficient matrices can be found by comparing (2.13) with the structure of (2.2):

$$\widetilde{\mathbb{B}}_0^{(x_2)}(\epsilon) = \mathbb{B}_0^{(x_2)}(\epsilon) + \sum_{x_j \in S \setminus \{x_2\}} \sum_{n \geq 0} \frac{x_2 - x_1}{(x_2 - x_j)^{n+1}} \mathbb{Q} \mathbb{B}_n^{(x_j)}(\epsilon) + (x_2 - x_1) \sum_{n \geq 0} x_2^n \mathbb{Q} \mathbb{B}_n(\epsilon) \,,$$

$$\widetilde{\mathbb{B}}_{k>0}^{(x_2)}(\epsilon) = \mathbb{B}_k^{(x_2)}(\epsilon) + (x_2 - x_1) \mathbb{Q} \mathbb{B}_{k-1}^{(x_2)}(\epsilon) \,,$$

$$\widetilde{\mathbb{B}}_k^{(x_j \neq x_2)}(\epsilon) = \mathbb{B}_k^{(x_j)}(\epsilon) + \sum_{n \geq 0} \frac{x_1 - x_2}{(x_2 - x_j)^{n+1}} \mathbb{Q} \mathbb{B}_{n+k}^{(x_j)}(\epsilon) \,,$$

$$\widetilde{\mathbb{B}}_k(\epsilon) = (x_2 - x_1) \sum_{n \geq 0} x_2^n \mathbb{Q} \mathbb{B}_{k+n+1}(\epsilon) \,.$$

An advantage of this form over the original transformation (2.9) is that now all operations are independent of x. Therefore, the underlying computer algebra system has to deal with rational functions of one less variable. The form (2.2) remains unspoiled, i.e. it is not necessary to perform a partial fraction decomposition after the transformation.

All transformations in terms of the coefficient matrices are listed in appendix A.

2.3 Overview of Lee's algorithm

Three basic steps allow Lee's algorithm[8] to transform an ordinary system of differential equations into an ϵ-form (2.4) if they are applied to the whole system:

1. transformation of a system into Fuchsian form,

2. normalization of the eigenvectors of all matrix residues,

3. factorization of ϵ.

In order to make use of the block-triangular structure of the system (2.6) these three steps are applied only to the active block followed by a fourth step to transform the off-diagonal block $\mathbb{B}$ into Fuchsian form.

In this sub-section we briefly discuss all four steps. More details can be found in the original paper by Lee[8].

Fuchsification

The basic building blocks for the first part of Lee's algorithm, the transformation of the system to Fuchsian form, are the balances defined in (2.7). With the right choice of a

projector $\mathbb{P}$, it is possible to perform a so-called Moser reduction to strictly lower the generalized Poincaré rank of the singularity at x_1 [12]. In this discussion we restrict ourselves to the case where $x_1 < \infty$. A more general treatment is given in[8].

Let p be the Poincaré rank of the singularity at x_1 of the active block $\mathbb{C}$ and $\{u_k^\alpha\}$ with $\alpha = 0, \ldots, n_k$ be the set of $n_k + 1$ right generalized eigenvectors of $\mathbb{C}_p^{(x_1)}$ belonging to a Jordan block k in the Jordan decomposition of $\mathbb{C}_p^{(x_1)}$. The Jordan blocks are ordered by their size so that $n_i \geq n_{i+1}$. If $p > 0$, we assume all eigenvalues of $\mathbb{C}_p^{(x_1)}$ to be zero, or else no transformation to lower the generalized Poincaré rank exists. The right generalized eigenvectors fulfill

$$\mathbb{C}_p^{(x_1)} u_k^{(0)} = 0 \, , \quad \mathbb{C}_p^{(x_1)} u_k^{(\alpha+1)} = u_k^{(\alpha)} \, .$$

These relations are invariant under the transformation

$$u_k^{(\alpha)} \to u_k^{(\alpha)} + c u_l^{(\alpha)} \, , \tag{2.14}$$

where $\alpha = 0, \ldots, n_k$ and $k > l$. The left generalized eigenvectors $v_k^{(\alpha)}$ are related to the right generalized eigenvectors by

$$\left(v_1^{(n_1)}, \ldots, v_1^{(0)}, v_2^{(n_2)}, \ldots, v_2^{(0)}, \ldots \right) = \left[\left(u_1^{(0)}, \ldots, u_1^{(n_1)}, u_2^{(0)}, \ldots, u_2^{(n_2)}, \ldots \right)^{-1} \right]^\dagger \, , \tag{2.15}$$

and fulfill

$$v_k^{(0)\dagger} \mathbb{C}_p^{(x_1)} = 0 \, , \quad v_k^{(\alpha+1)\dagger} \mathbb{C}_p^{(x_1)} = v_k^{(\alpha)\dagger} \, .$$

The transformation (2.14) allows us to find a basis of eigenvectors which satisfies

$$v_j^{(0)\dagger} \mathbb{C}_{p-1}^{(x_1)} u_k^{(0)} = 0 \, , \tag{2.16}$$

for $j \notin R$ and $k \in R \cup \{k_0\}$, where R is some set of trivial Jordan blocks ($n_i = 0$ for $i \in R$) and k_0 is a non-trivial Jordan block ($n_{k_0} > 0$). An algorithm to find these generalized eigenvectors, together with the set R and k_0 is given in [8].

If in the definition of $\mathbb{P}$ (2.10) we use

$$\mathbb{Q} = \sum_{k \in R \cup \{k_0\}} u_k^{(0)} w_k^\dagger \, , \tag{2.17}$$

with $w_j^\dagger u_k^{(0)} = \delta_{jk}$ and then apply the balance (2.7a), the generalized Poincaré rank of the singularity at x_1 is decreased. To see this, we introduce the projector

$$\mathbb{Q}_1 = \sum_{k \in R \cup \{k_0\}} u_k^{(0)} v_k^{(n_k)\dagger} \, . \tag{2.18}$$

8

Since $v_j^{(n_j)\dagger}u_k^{(0)} = \delta_{jk}$, the projector $\mathbb{Q}_1$ also is of the form (2.17). From the definitions of $\mathbb{Q}$ and $\mathbb{Q}_1$ follows

$$\mathbb{Q}_1\mathbb{Q} = \mathbb{Q}, \quad \mathbb{Q}\mathbb{Q}_1 = \mathbb{Q}_1, \quad \mathbb{C}_p^{(x_1)}\mathbb{Q} = \mathbb{C}_p^{(x_1)}\mathbb{Q}_1 = 0.$$

Evaluating the transformation (A.1b) at $k = p$ leads to

$$\widetilde{\mathbb{C}}_p^{(x_1)} = \overline{\mathbb{Q}}\mathbb{C}_p^{(x_1)} + (x_1 - x_2)\overline{\mathbb{Q}}\mathbb{C}_{p-1}^{(x_1)}\mathbb{Q}$$
$$= (\overline{\mathbb{Q}} + \mathbb{Q}_1)\left[\overline{\mathbb{Q}}_1\mathbb{C}_p^{(x_1)} + (x_1 - x_2)\overline{\mathbb{Q}}_1\mathbb{C}_{p-1}^{(x_1)}\mathbb{Q}_1\right](\overline{\mathbb{Q}}_1 + \mathbb{Q})$$

As $(\overline{\mathbb{Q}} + \mathbb{Q}_1) = (\overline{\mathbb{Q}}_1 + \mathbb{Q})^{-1}$, the matrix rank of $\widetilde{\mathbb{C}}_p^{(x_1)}$ is given by

$$\operatorname{rank}\widetilde{\mathbb{C}}_p^{(x_1)} = \operatorname{rank}\widehat{\mathbb{C}}_p^{(x_1)}, \quad \widehat{\mathbb{C}}_p^{(x_1)} = \overline{\mathbb{Q}}_1\mathbb{C}_p^{(x_1)} + (x_1 - x_2)\overline{\mathbb{Q}}_1\mathbb{C}_{p-1}^{(x_1)}\mathbb{Q}_1,.$$

The argument that the matrix rank (and therefore the generalized Poincaré rank) of $\widetilde{\mathbb{C}}_p^{(x_1)}$ is lower than the matrix rank of $\mathbb{C}_p^{(x_1)}$ is as follows:

- All left eigenvectors $v_j^{(0)}$ of $\mathbb{C}_p^{(x_1)}$ with $j \in R$ are left eigenvectors of $\widehat{\mathbb{C}}_p^{(x_1)}$ as $v_j^{(0)\dagger}\overline{\mathbb{Q}}_1 = 0$.

- All left eigenvectors $v_j^{(0)}$ of $\mathbb{C}_p^{(x_1)}$ with $j \notin R$ are left eigenvectors of $\widehat{\mathbb{C}}_p^{(x_1)}$ as $v_j^{(0)\dagger}\overline{\mathbb{Q}}_1 = v_j^{(0)\dagger}$ and so

$$v_j^{(0)\dagger}\widehat{\mathbb{C}}_p^{(x_1)} = (x_1 - x_2)\sum_{k \in R \cup \{k_0\}} v_j^{(0)\dagger}\mathbb{C}_{p-1}^{(x_1)}u_k^{(0)}v_k^{(n_k)\dagger} = 0,$$

where we used (2.16).

- The vector $v_{k_0}^{(n_{k_0})}$ which is not a left eigenvector of $\mathbb{C}_p^{(x_1)}$ is an additional left eigenvector of $\widehat{\mathbb{C}}_p^{(x_1)}$ as $v_{k_0}^{(n_{k_0})\dagger}\overline{\mathbb{Q}}_1 = 0$.

So $\widehat{\mathbb{C}}_p^{(x_1)}$ has one eigenvector more than $\mathbb{C}_p^{(x_1)}$ and therefore has a lower matrix rank.

Unfortunately, the balance (2.7a) might also increase the Poincaré rank at x_2. Therefore, we have a closer look at (A.1d) evaluated at $k = q + 1$, where q is the Poincaré rank at the singularity x_2:

$$\widetilde{\mathbb{C}}_{q+1}^{(x_2)} = (x_2 - x_1)\mathbb{Q}\mathbb{C}_q^{(x_2)}\overline{\mathbb{Q}}.$$

If this expression vanishes, the Poincaré rank at x_2 is not increased. This is the case if the vectors $w_k^\dagger$ in (2.17) span a left-invariant space of $\mathbb{C}_q^{(x_2)}$.

9

In Lee's algorithm the above steps are used to decrease the Poincaré rank of a singularity x_1 as long as it is possible to find a projector (2.17) with $w_k^\dagger$ spanning a left-invariant space of some $\mathbb{C}_q^{(x_2)}$, where $x_2 \neq x_1$ is some other singular point. If no such projector exists, a regular point y is chosen and the projector $\mathbb{Q}_1$ defined by (2.18) is used in the balance. This of course creates a new (apparent) Fuchsian singularity at y. This way it is possible to reduce the system to an equivalent one with all singularities having Poincaré rank zero. Fortunately, the next step, the normalization of the eigenvalues, removes all apparent singularities introduced in this step.

Normalization

The second step of Lee's algorithm is the normalization of the eigenvalues of the matrix residues. Again, the main ingredients are the balances defined in (2.7). We assume all singularities to be Fuchsian now and the eigenvalues of the matrix residues to be of the form $a + b\epsilon$ with $a \in \mathbb{Z}$. This is a necessary condition for the existence of a transformation normalizing all eigenvalues. If this condition cannot be fulfilled a redefinition of the kinematic variable might be helpful. A normalized eigenvalue is an eigenvalue proportional to ϵ (i.e. with $a = 0$). The matrix residue at ∞ is equal to $-\sum_{x_j \in S} \mathbb{C}_0^{(x_j)}$. Therefore, the sum of all matrix residues (including the residue at ∞) vanishes and hence the sum of all eigenvalues of all matrix residues must vanish as well.

For brevity's sake, we will restrict our discussion to finite singularities $x_1, x_2 < \infty$; the generalized treatment can again be found in[8].

Let $\{u_k^{(\alpha)}\}$ with $\alpha = 0, \ldots, n_k$ be the set of $n_k + 1$ right generalized eigenvectors of $\mathbb{C}_0^{(x_1)}$ belonging to the Jordan block k:

$$\mathbb{C}_0^{(x_1)} u_k^{(0)} = \lambda_k u_k^{(0)}, \quad \mathbb{C}_0^{(x_1)} u_k^{(\alpha+1)} = \lambda_k u_k^{(\alpha+1)} + u_k^{(\alpha)}.$$

As in the fuchsification step, we define left generalized eigenvectors $v_k^{(\alpha)}$ via (2.15) which satisfy

$$v_k^{(0)\dagger} \mathbb{C}_0^{(x_1)} = \lambda_k v_k^{(0)\dagger}, \quad v_k^{(\alpha+1)\dagger} \mathbb{C}_0^{(x_1)} = \lambda_k v_k^{(\alpha+1)\dagger} + v_k^{(\alpha)\dagger}.$$

A balance (2.7a) with a right choice of a projector $\mathbb{P}$ can now be used to shift one eigenvalue of the matrix residue at x_1 up by one and/or one eigenvalue of the matrix residue at x_2 down by one. Let us consider a projector $\mathbb{P}$ defined by (2.10) with

$$\mathbb{Q} = u_k^{(0)} w^\dagger, \tag{2.19}$$

where $w^\dagger u_k^{(0)} = 1$. Also an additional projector

$$\mathbb{Q}_1 = u_k^{(0)} v_k^{(n_k)}$$

is useful for the discussion. The following relations involving these two projectors hold:

$$\mathbb{Q}\mathbb{Q}_1 = \mathbb{Q}_1\,,\quad \mathbb{Q}_1\mathbb{Q} = \mathbb{Q}\,,\quad \mathbb{C}_0^{(x_1)}\mathbb{Q} = \lambda_k\mathbb{Q}\,,\quad \mathbb{C}_0^{(x_1)}\mathbb{Q}_1 = \lambda_k\mathbb{Q}_1\,.$$

Let us now consider the transformation of $\mathbb{C}_0^{(x_1)}$ under a balance (2.7a) as given in (A.1a):

$$\widetilde{\mathbb{C}}_0^{(x_1)} = \mathbb{C}_0^{(x_1)} - \mathbb{Q}\mathbb{C}_0^{(x_1)}\overline{\mathbb{Q}} + \sum_{x_j\in S\backslash\{x_1\}} \frac{x_1 - x_2}{x_1 - x_j}\overline{\mathbb{Q}}\mathbb{C}_0^{(x_j)}\mathbb{Q} + \mathbb{Q}$$

$$= (\overline{\mathbb{Q}} + \mathbb{Q}_1)\widehat{\mathbb{C}}_0^{(x_1)}(\overline{\mathbb{Q}}_1 + \mathbb{Q})\,,$$

where

$$\widehat{\mathbb{C}}_0^{(x_1)} = \mathbb{C}_0^{(x_1)} - \mathbb{Q}_1\mathbb{C}_0^{(x_1)}\overline{\mathbb{Q}}_1 + \sum_{x_j\in S\backslash\{x_1\}} \frac{x_1 - x_2}{x_1 - x_j}\overline{\mathbb{Q}}_1\mathbb{C}_0^{(x_j)}\mathbb{Q}_1 + \mathbb{Q}_1\,.$$

Because of $(\overline{\mathbb{Q}}+\mathbb{Q}_1) = (\overline{\mathbb{Q}}_1+\mathbb{Q})^{-1}$, $\widetilde{\mathbb{C}}_0^{(x_1)}$ and $\widehat{\mathbb{C}}_0^{(x_1)}$ are related by a similarity transformation and thus have the same eigenvalues. To evaluate the eigenvalues of $\widetilde{\mathbb{C}}_0^{(x_1)}$ it is therefore sufficient to analyze the eigenvalues of $\widehat{\mathbb{C}}_0^{(x_1)}$ which are much simpler to determine. We consider $\widehat{\mathbb{C}}_0^{(x_1)}$ in the basis of the generalized eigenvectors of $\mathbb{C}_0^{(x_1)}$. In this basis $\mathbb{C}_0^{(x_1)}$ is in Jordan normal form. The second term $-\mathbb{Q}_1\mathbb{C}_0^{(x_1)}\overline{\mathbb{Q}}_1$ removes all elements from the row corresponding to $u_k^{(0)}$ but the diagonal one containing the eigenvalue to $u_k^{(0)}$. The last term $+\mathbb{Q}_1$ increases the diagonal element by one. The terms proportional to $\overline{\mathbb{Q}}_1\mathbb{C}_0^{(x_j)}\mathbb{Q}_1$ contribute to the non-diagonal elements of the column corresponding to $u_k^{(0)}$ (or $v_k^{(n_k)\dagger}$). Hence, the transformations can be summarized as

$$\begin{pmatrix} \ddots & & & & & & & & \\ & \lambda_{k-1} & & & & & & & \\ & & \lambda_k & 1 & & & & & \\ & & & \lambda_k & 1 & & & & \\ & & & & \ddots & \ddots & & & \\ & & & & & \lambda_k & 1 & & \\ & & & & & & \lambda_k & & \\ & & & & & & & \lambda_{k+1} & 1 \\ & & & & & & & & \ddots & \ddots \end{pmatrix} \rightarrow \begin{pmatrix} \ddots & & \vdots & & & & & & \\ & \lambda_{k-1} & * & & & & & & \\ & & \lambda_k+1 & 0 & & & & & \\ & & * & \lambda_k & 1 & & & & \\ & & \vdots & & \ddots & \ddots & & & \\ & & * & & & \lambda_k & 1 & & \\ & & * & & & & \lambda_k & & \\ & & * & & & & & \lambda_{k+1} & 1 \\ & & \vdots & & & & & & \ddots & \ddots \end{pmatrix}\,,$$

where $*$ stands for contributions from the $\overline{\mathbb{Q}}_1\mathbb{C}_0^{(x_j)}\mathbb{Q}_1$ terms. Calculating the characteristic polynomial by means of a Laplace expansion along the row corresponding to $u_k^{(0)}$ leads to the conclusion that all eigenvalues but one stay the same; only one eigenvalue λ_k is changed to $\lambda_k + 1$.

In the same way, a projector

$$\mathbb{Q}' = w v_k^{(0)\dagger} \tag{2.20}$$

with a left eigenvector $v_k^{(0)\dagger}$ of $\mathbb{C}_0^{(x_2)}$ and $v_k^{(0)\dagger} w = 1$ shifts one eigenvalue λ_k of $\mathbb{C}_0^{(x_2)}$ down by one.

A balance with a projector (2.19) could spoil the Fuchsian form of the system if it increases the Poincaré rank at any singularity. This is in principle possible at x_2. Evaluating (A.1d) at $k = 1$ leads to

$$\widetilde{\mathbb{C}}_1^{(x_2)} = (x_2 - x_1)\mathbb{Q}\mathbb{C}_0^{(x_2)}\overline{\mathbb{Q}}\,.$$

This vanishes if $w^\dagger$ is a left eigenvector of $\mathbb{C}_0^{(x_2)}$. In that case not only the Fuchsian form of the system is preserved but we also arrive at a projector of the form (2.20). So the projector of choice is

$$\mathbb{Q} = u_k^{(0)} v_l^{(0)\dagger}\,, \tag{2.21}$$

where $u_k^{(0)}$ is a right eigenvector of $\mathbb{C}_0^{(x_1)}$, $v_l^{(0)\dagger}$ is a left eigenvector of $\mathbb{C}_0^{(x_2)}$ and $v_l^{(0)\dagger} u_k^{(0)} = 1$. This projector, used in balance (2.7a), increases one eigenvalue λ_k of $\mathbb{C}_0^{(x_1)}$ by one and decreases one eigenvalue μ_l of $\mathbb{C}_0^{(x_2)}$ by one while conserving the Fuchsian form of the system.

In order to utilize the considerations above in an algorithmic approach, one first selects a singularity x_0 as 'fallback'. Then balances with projectors of the form (2.21) are used to 'mutually balance' eigenvalues between two singularities. Certainly, the eigenvalue to be increased should be negative and the eigenvalue to be decreased should be positive (for $\epsilon = 0$).

If no such balance exists, the eigenvalues will be balanced with the 'fallback singularity' x_0 regardless of the sign of the eigenvalue at x_0. This normalizes the eigenvalues at all singularities but x_0. To normalize even the eigenvalues at x_0, we balance one unnormalized eigenvalue with some regular point creating a new apparent singularity there and restart the algorithm. Hopefully, the unnormalized eigenvalue at this new apparent singularity can now be mutually balanced with another unnormalized eigenvalue at x_0.

ϵ-Factorization

In the next step we find an x-independent transformation $\mathbb{T}(\epsilon)$ to factor out ϵ. For an x-independent $\mathbb{T}$, (2.5) becomes a similarity transformation and does not change the eigenvalues of the system. This makes it necessary for the eigenvalues to be proportional to ϵ, i.e. the normalization step before must have been successful. We use the fact that

$\mathbb{T}^{-1}(\epsilon)[\mathbb{M}_0^{(x_j)}(\epsilon)/\epsilon]\mathbb{T}(\epsilon)$ should be independent of ϵ, so that the equation

$$\mathbb{T}^{-1}(\epsilon)\frac{\mathbb{M}_0^{(x_j)}(\epsilon)}{\epsilon}\mathbb{T}(\epsilon) = \mathbb{T}^{-1}(\mu)\frac{\mathbb{M}_0^{(x_j)}(\mu)}{\mu}\mathbb{T}(\mu)$$

holds for all $x_j \in S$. Multiplying this equation from the left by $\mathbb{T}(\epsilon)$ and from the right by $\mathbb{T}^{-1}(\mu)$, leads to

$$\frac{\mathbb{M}_0^{(x_j)}(\epsilon)}{\epsilon}\mathbb{T}(\epsilon,\mu) = \mathbb{T}(\epsilon,\mu)\frac{\mathbb{M}_0^{(x_j)}(\mu)}{\mu} \ , \tag{2.22}$$

where $\mathbb{T}(\epsilon,\mu) = \mathbb{T}(\epsilon)\mathbb{T}^{-1}(\mu)$. This linear system can be solved e.g. with Gaussian elimination and the constants should be fixed so that $\mathbb{T}(\epsilon,\mu_0)$ is an invertible matrix, where μ_0 is some arbitrary number. The transformation $\mathbb{T}(\epsilon,\mu_0)$ can now be used to factor out ϵ.

Fuchsification of off-diagonal blocks

Since the definition of the active block is somewhat arbitrary, we should in principle be able to transform all diagonal blocks to ϵ-form by redefining the active block and applying the three steps described above. However, we still need to reduce the off-diagonal block $\mathbb{B}$ to Fuchsian form. Again we will restrict the discussion to finite singularities $x_1 < \infty$ and assume the block $\mathbb{A}$ and $\mathbb{C}$ to be already in ϵ-form.

Let p be the Poincaré rank of the off-diagonal block $\mathbb{B}$ at the singularity x_1, i.e. $\mathbb{B}_p^{(x_1)} \neq 0$ and $\mathbb{B}_k^{(x_1)} = 0$ for $k > p$. The behavior of $\mathbb{B}_p^{(x_1)}$ under a transformation (2.8a) with $k = p$ is given by (A.3a):

$$\widetilde{\mathbb{B}}_p^{(x_1)} = \mathbb{B}_p^{(x_1)} + \mathbb{C}_0^{(x_1)}\widehat{\mathbb{G}} - \widehat{\mathbb{G}}\mathbb{A}_0^{(x_1)} + p\widehat{\mathbb{G}} \ .$$

In order to decrease the Poincaré rank, $\widehat{\mathbb{G}}$ has to be determined such that $\widetilde{\mathbb{B}}_p^{(x_1)}$ vanishes. This linear system of equations can be solved e.g. with Gaussian elimination.

Hence, with transformations of the form (2.8) it is possible to reduce all singularities of the off-diagonal block $\mathbb{B}$ to Fuchsian form.

3 Usage

3.1 Installation guide on Linux systems

Ensure that the dependencies

- Fermat (≥ 6.0) [11],

- GiNaC ($\geq 1.6.2$) [13] (for epsilon-prepare only)

are installed.

As a next step, libFermat has to be installed. libFermat, which was developed in connection with epsilon, is a C++ library designed to communicate with Fermat. Nevertheless, we decided to publish libFermat in a separate repository since it might be useful elsewhere. Internally, the communication is done with PStreams[14] which is included in the package. The source code of the most recent version of libFermat can be obtained via github using

```
git clone https://github.com/mprausa/libFermat.git
```

This will create a directory libFermat/ and clone the library into that location. Now inside this directory, run

```
cmake -DCMAKE_INSTALL_PREFIX=/path/to/install .
make
make install
```

where /path/to/install is your desired installation directory and defaults to /usr/local on a typical Linux system. The library is installed into the sub-directory lib and the header files into the sub-directory include of /path/to/install. If your choice is a global directory you will require root access for the last step make install. Remember to include the sub-directory lib of /path/to/install into the LD_LIBRARY_PATH environment variable if you are using a non-standard directory.

The next step is to install epsilon and epsilon-prepare. In principle the procedure is the same as for the installation of libFermat. First, obtain the most recent version of the source code with

```
git clone https://github.com/mprausa/epsilon.git
```

then change into the newly created directory epsilon/ and run

14

```
cmake -DCMAKE_INSTALL_PREFIX=/path/to/install .
make
make install
```

It is recommended to use the same `/path/to/install` as for `libFermat`, else the `cmake` step might require additional options to find `libFermat`. The programs `epsilon` and `epsilon-prepare` are installed into the sub-directory `bin` of the installation path. As before, `make install` might need `root` access depending on the installation prefix. For a non-standard installation prefix, the environment variable `PATH` should be adjusted to include the sub-directory `bin` of `/path/to/install` so that the programs `epsilon` and `epsilon-prepare` can be found by the shell.

It is also possible to build `epsilon` and `epsilon-prepare` individually. This can be done by changing into the corresponding sub-directory and running `cmake` and `make` from within there.

The `epsilon`-repository also offers a `Mathematica` package `EpsilonTools.m` found in the sub-directory `mma/`. Run

```
./install.sh
```

from within this sub-directory to install `EpsilonTools.m` into the `Applications/` directory of your Mathematica installation.

3.2   Input/Output format

`epsilon` uses its own file format to represent a system of differential equations of the form (2.2), where every line represents one coefficient matrix. A line starts with either '`A[`x_j`,k]:`' or '`B[k]:`' followed by a matrix. The matrix is stored as a list of rows, where each row is itself a list of matrix elements. Lists are enclosed in curly-braces and list entries are separated by commas. A line starting with '`A[`x_j`,k]:`' ('`B[k]:`') represents a matrix $\mathbb{M}_k^{(x_j)}$ ($\mathbb{M}_k$) in (2.2).

The name of the symbol used to represent ϵ is fixed to `ep`.

An example of an input file for `epsilon` is given in section 4.

3.3   Usage of `epsilon-prepare`

The tool `epsilon-prepare` is used to convert a matrix in `Mathematica` format (a list of lists) into `epsilon` input format (see section 3.2). If, for example, the file containing a

15

symbol	polymod	expression
i	$\mathtt{i}^2 + 1$	i
$\mathtt{r}N$	$\mathtt{r}N^2 - \mathtt{r}N + (1 + N)/4$	$\frac{1+i\sqrt{N}}{2}$
$\mathtt{q}N$	$\mathtt{q}N^2 - \mathtt{q}N + (1 - N)/4$	$\frac{1+\sqrt{N}}{2}$
$\mathtt{sqrt}N$	$\mathtt{sqrt}N^2 - N$	$\sqrt{N}$
$\mathtt{isqrt}N$	$\mathtt{isqrt}N^2 + N$	$i\sqrt{N}$

Table 1: Additional symbols used by `epsilon-prepare` to represent zeros of quadratic polynomials. N is a positive integer.

matrix in `Mathematica` format is called `matrix.m`, then the command

```
epsilon-prepare matrix.m > matrix.dat
```

is used to create a file `matrix.dat` in `epsilon` format. The matrix elements of the input matrix are expected to be rational functions in ϵ and x (represented by the symbols `ep` and `x`, respectively). The command `epsilon-prepare` performs a partial fraction decomposition over complex numbers in the variable x of the matrix elements and results in an expression of the form of (2.2).

To perform a partial fractioning of a rational function, the zeros of its denominator have to be determined. Therefore, `epsilon-prepare` applies `GiNaC`'s polynomial factorization algorithm to the denominator in order to factorize it into polynomials that are irreducible over the integers. In a second step, the zeros of all factors are found individually.

This step will fail, if the considered irreducible polynomial has a degree larger than two. An error is thrown as well if the system is not in form (2.2), i.e. a zero depends on the parameter ϵ.

The additional symbols listed in table 1 might be introduced by `epsilon-prepare`. See section 3.5 for how to make `epsilon` accept them.

3.4   Usage of `epsilon`

The general command syntax for the tool `epsilon` is

```
epsilon [OPTIONS] JOBS...
```

The path to the `Fermat` binary can be set inside the environment variable `FERMAT`. If this variable is not set explicitly, `epsilon` will look for a binary `fer64` inside the directories defined in the environment variable `PATH`. Options are set once at the start of `epsilon`.

Jobs are processed one by one in the same order they are defined on the command line. Some jobs will perform transformations to the system. These transformations are stored in a so-called internal transformation queue in RAM and can also be written to an external file.

Options:

- `--verbose`:   Enable verbose output.

  This option prints out all communication between `libFermat` and `Fermat` during a regular run. This is useful as a debug tool.

- `--timings`:   Enable timings.

  This option prints the elapsed real time after every job and the total time of the complete run.

- `--symbols` *symbols*:   Adjoin additional symbols to `Fermat`.

  This option adds the specified symbols to `Fermat`. The symbols defined in this option are the very first variables adjoint to `Fermat` followed by the symbol `ep` and further internally used symbols. *symbols* has to be a comma-separated list.

- `--echelon-fermat`:   Use the `Redrowech` function in `Fermat` to solve linear systems.

  At various points in the code linear systems of equations have to be solved. Use this option to choose the `Redrowech` function over our own implementation of Gaussian elimination.

Jobs:

- `--fermat` *file*:   Execute `Fermat` commands.

  This job reads *file* line by line and sends all non-empty lines to `Fermat`.

- `--load` *file start end*:   Load system.

  This job loads *file* and activates the block $\{start, end\}$. Hereby *file* is expected to be in the format specified in section 3.2

- `--write` *file*:   Write system.

  This job writes the system of differential equations to *file* using the format specified in section 3.2.

- `--queue` *file*:   Set external transformation queue.

  This job enables an external transformation queue. An external transformation queue is a file containing all transformations already performed by `epsilon` during a run. This is particularly useful in connection with the options `--load-queue` and `--replay` to restore an aborted run to the state after the last successful transformation.

- `--load-queue` *filename*:   Load transformation queue.

  This job loads an external transformation queue from *filename* into an internal transformation queue stored in RAM. This job does *not* apply the transformations stored in the file to the system.

- `--replay`:   Apply internal transformation queue.

  This job 'replays' the internal transformation queue, i.e. the transformations in the queue are applied to the system one by one. It should only be used immediately after `--load-queue`.

- `--export` *file*:   Export transformation matrix.

  This job computes a transformation matrix out of the transformations inside the internal transformation queue. The matrix is written in `Mathematica` format to *file*.

- `--block` *start* *end*:   Activate a block.

  This job activates the block $\{start, end\}$.

- `--fuchsify`:   Transform active block into Fuchsian form.

  This job reduces the active block to Fuchsian form (2.3). See section 2.3 for details.

- `--normalize`:   Normalize eigenvalues.

  This job normalizes the eigenvalues of all residue matrices making them proportional to ϵ. See section 2.3 for details.

- `--factorep`:   Factor out ϵ (auto detect μ).

  This job transforms the active block into ϵ-form (2.4) using the method described in section 2.3. The variable μ in (2.22) is left as an unknown and will be fixed only *after* the system is solved to ensure that the transformation is invertible.

- `--factorep-at` *mu*:   Factor out ϵ (with predefined μ).

  This job transforms the active block into ϵ-form (2.4) using the method described in section 2.3. In this variant, the variable μ in (2.22) is set to *mu* *before* the system is solved. This is faster than `--factorep` because `epsilon` has to deal with one less variable in the polynomials. Unfortunately, an unlucky choice of *mu* can hit a pole in the matrix elements of the system or one can end up with an uninvertible transformation. In both cases an error is thrown.

- `--left-fuchsify`:   Fuchsify off-diagonal block.

  This job is used to transform the block left of the active block (block $\mathbb{B}$ in (2.6)) to Fuchsian form. See section 2.3 for details.

- `--dyson` *file order type format*:   Generate Dyson operator.

  This job writes a Dyson operator $U(x, x_0)$ for the active block up to order *order* in ϵ to *file*. The active block has to be in ϵ-form. The Dyson operator fulfills

$$\frac{\partial}{\partial x} U(x, x_0) = \epsilon \sum_{x_j \in S} \frac{\widehat{\mathbb{C}}_0^{(x_j)}}{x - x_j} U(x, x_0) , \quad U(x_0, x_0) = \mathbb{1} .$$

  The option *type* specifies the type of multiple polylogarithms in the output and can be set to `GPL`, `HPL` or `HPLalt` for Goncharov polylogarithms[15] or harmonic polylogarithms in the "a"- or "m"-notation[16, 17], respectively. *format* should be `mma` or `form` to specify `Mathematica` or `FORM`[18] as output format.

3.5 Using field extensions

The `Fermat` computer algebra system works with multivariate polynomials over the ground ring $\mathbb{Z}$ and the corresponding quotient field, the rational functions over $\mathbb{Z}$. Fortunately, `Fermat` offers a way to extend the ground ring by setting so-called polymods.

A polymod $p(\xi)$ is a univariate polynomial in ξ, where ξ is one of the variables adjoint to `Fermat`. This instates a new quotient ring $\mathbb{Z}[\xi]/\langle p(\xi) \rangle$ as the ground ring, forcing

the condition $p(\xi) = 0$ onto the variable ξ. In other words every polynomial $q(\xi) \in \mathbb{Z}[\xi]$ encountered by `Fermat` is replaced immediately by the remainder of a polynomial division $q(\xi) \div p(\xi)$. For more details see the `Fermat` manual[11].

The polymod $i^2 + 1$ for example leads to the quotient ring $\mathbb{Z}[i]/\langle i^2 + 1\rangle$ which is equivalent to the Gaussian integers, a field extension of the integers by a number i with $i^2 = -1$. Also other complex numbers can be represented in `Fermat` using polymods. The complex numbers introduced via `epsilon-prepare` can be represented in `Fermat` using the polymods listed in table 1.

The syntax to set a polymod in `Fermat` is `&(P=`*polymod*`,1)`, where the variable of the polymod must be the first symbol adjoint to `Fermat` which does not have a polymod assigned yet. This `Fermat` command should be stored in a text file and can be read in by `epsilon` with the `--fermat` job.

For internal reasons `Fermat` runs very slow if more than one polymod is assigned to it. Therefore, in complicated cases with more than one complex number, it is useful to set a polymod only for the most frequent variable appearing in the block to work with next. As the order of symbols once set cannot be changed inside `Fermat`, the only way to change the 'active' polymod is by saving the system with `--write` and reload it into a new session after the symbols are adjoined in a different order. In our tests we were able to solve huge systems with up to three complex numbers applying this strategy.

3.6 Usage of `EpsilonTools.m`

The Mathematica package `EpsilonTools.m` provides functions which help to set up `epsilon` and to work with the `epsilon` input/output file format (see section 3.2). It is not essential in order to run `epsilon`. After a successful installation, the package can be loaded into a Mathematica session with

```
<<EpsilonTools'
```

`EpsilonTools.m` provides three functions:

- `EpsilonSymRules[`*expression*`]`

  This function scans *expression* for symbols of the form given in table 1 and compiles a list of rules for these symbols to their corresponding Mathematica expressions, e.g.

  ```
  {r3->(1+I*Sqrt[3])/2, q5->(1+Sqrt[5])/2, i->I}
  ```

- `EpsilonRead[`*`file`*`]`

  This function reads *`file`* into Mathematica where *`file`* is in the format described in section 3.2.

  Options:

  - `ReplaceSymbols` (default: `True`)

    If this option is set to `True`, all symbols introduced by `epsilon-prepare` will be replaced by their corresponding Mathematica expression.

  - `CheckFuchsian` (default: `False`)

    If this option is set to `True`, the function returns `$Failed` if the system in *`file`* is not in Fuchsian form.

  - `CheckEpsilon` (default: `False`)

    If this option is set to `True`, the function returns `$Failed` if the system in *`file`* is not in ϵ-form.

- `EpsilonBlocks[`*`M`*`]`

  This function scans for a block-triangular structure of the matrix *`M`* and returns a list of the boundaries of all diagonal blocks. The returned boundaries are in the form {*`start`*, *`end`*}. The values *`start`* and *`end`* can be used in the `--block` job of `epsilon`.

4   A physical example

As an example, we consider a set of three-loop master integrals $\{I_j\}$ with $j = 1, \ldots, 9$ in $d = 4 - 2\epsilon$ dimensions with internal lines of mass one or zero. A graphical representation of the master integrals except I_3 is contained in fig. 4.1. Further we define I_3 as

$$I_3 = \int d^d p \int d^d l \int d^d k \, \frac{(k-l)^2}{[(p+q_1)^2 - 1][l^2 - 1][(p-l+k)^2 - 1][(l-p+q_2)^2]} \, .$$

The kinematics is given by

$$q_1^2 = q_2^2 = 0 \,, \quad q_1 \cdot q_2 = -\frac{(1-x)^2}{2x} \, .$$

The vector $\vec{f} = (I_1, \ldots, I_9)$ obeys a differential equation

$$\frac{\partial}{\partial x} \vec{f} = \mathbb{M}(x, \epsilon) \vec{f},$$

Figure 4.1: Three-loop master integrals to be solved with `epsilon`. The not pictured integral I_3 has the same topology as I_2 but with an additional numerator. The thick (thin) lines are massive (massless). The thin external lines carry the momenta q_1 and q_2, while the double line carries the momentum $q_1 + q_2$.

where the 9×9-matrix $\mathbb{M}(x, \epsilon)$ has the structure

$$\mathbb{M}(x, \epsilon) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{pmatrix} . \tag{4.1}$$

The $*$ represents any non-zero entry and we point out the block-triangular structure of the system. First, this matrix should be stored in a `Mathematica` compatible file `matrix.m`. The next step is to convert this file to the format described in section 3.2 via

```
epsilon-prepare matrix.m > matrix.dat
```

The generated file `matrix.dat` should read

```
A[r3,0]:        {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[-1,0]:        {{0,0,0,0,0,0,0,0,0},{1-ep,5-6*ep,-6+ ...
A[-1,1]:        {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[-1,2]:        {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[1-r3,0]:      {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[1,0]:         {{0,0,0,0,0,0,0,0,0},{-1+ep,-11+10*ep ...
A[1,1]:         {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[1,2]:         {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[1,3]:         {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
A[1,4]:         {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
```

```
A[0,0]:                 {{0,0,0,0,0,0,0,0},{0,3-2*ep,0,0,0, ...
A[0,1]:                 {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
B[0]:                   {{0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0 ...
```

The arguments of `A` and `B` reveal singularities at $\{r_3, -1, 1 - r_3, 1, 0, \infty\}$ with Poincaré ranks $\{0, 2, 0, 4, 1, 1\}$, respectively. The symbol r_3 introduced by `epsilon-prepare` is a root of the polynomial $r_3^2 - r_3 + 1$ and is given by $r_3 = (1 + i\sqrt{3})/2$ (see table 1). Hence, we need a file `enable.r3.fer` to set a polynomial for `Fermat` to mod out with containing

```
&(P=r3^2-r3+1,1)
```

To understand the origin of r_3 we consider for example the partial fraction decomposition over the complex of $[\mathbb{M}(x, \epsilon)]_{51}$:

$$[\mathbb{M}(x, \epsilon)]_{51} = \frac{(\epsilon - 1)^2 (1 + x) \left(1 + 4x + x^2 + \epsilon (1 - 10x + x^2)\right)}{4\epsilon(2\epsilon - 1)(x - 1)^3 (x^2 - x + 1)}$$

$$= \frac{1}{x - r_3} \left\{ \frac{5 - 9\epsilon^3 + 23\epsilon^2 - 19\epsilon}{4(2\epsilon^2 - \epsilon)} \right\} + \frac{1}{x - (1 - r_3)} \left\{ \frac{5 - 9\epsilon^3 + 23\epsilon^2 - 19\epsilon}{4(2\epsilon^2 - \epsilon)} \right\}$$

$$+ \frac{1}{x - 1} \left\{ \frac{-5 + 9\epsilon^3 - 23\epsilon^2 + 19\epsilon}{2(2\epsilon^2 - \epsilon)} \right\} + \frac{1}{(x - 1)^2} \left\{ \frac{3 - 4\epsilon^3 + 11\epsilon^2 - 10\epsilon}{2(2\epsilon^2 - \epsilon)} \right\}$$

$$+ \frac{1}{(x - 1)^3} \left\{ \frac{3 - 4\epsilon^3 + 11\epsilon^2 - 10\epsilon}{2\epsilon^2 - \epsilon} \right\}.$$

The possible blocks to run `epsilon` with can be either read off from the matrix (4.1) or determined by the function `EpsilonBlocks` of the `Mathematica` package `EpsilonTools.m`. In this case, the possible blocks are

$$\{1, 1\}, \{2, 3\}, \{4, 4\}, \{5, 9\}.$$

Now we can run `epsilon` with the command

```
epsilon --timings --symbols r3 --load matrix.dat 1 1 \
  --queue out.queue \
  --fermat enable.r3.fer \
  --fuchsify --normalize --factorep-at -1 \
  --block 2 3 \
  --fuchsify --normalize --factorep-at -1 --left-fuchsify \
  --block 4 4 \
  --fuchsify --normalize --factorep-at -1 --left-fuchsify \
  --block 5 9 \
  --fuchsify --normalize --factorep-at  1 --left-fuchsify \
```

```
    --block 1 9 \
    --factorep-at -1 \
    --write epsilon.dat --export transformation.m
```

The `--load` job in the first line loads the file `matrix.dat` with an active block $\{1,1\}$; the `--block` jobs are used to change the active block. In all blocks except for block $\{5,9\}$ we factor out ϵ at $\mu = -1$. For this block $\mu = -1$ would lead to a singular system so we choose $\mu = 1$ instead. The final system is written into the file `epsilon.dat` and the corresponding transformation matrix into `transformation.m`. The whole reduction process takes about three minutes on an Intel Core i5-3320M CPU with 2.60 GHz. The package `EpsilonTools.m` offers the function `EpsilonRead` for reading the file `epsilon.dat` into Mathematica. The result is in ϵ-form (2.4), with

$$
\widehat{\mathbb{M}}_0^{(-1)} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{5} & -18 & \frac{36}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & -30 & 12 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{7790185}{4743936} & \frac{-406887355}{3162624} & \frac{13303239}{263552} & \frac{-3715319}{6325248} & \frac{-21}{58} & \frac{323}{58} & 1 & \frac{-121}{58} & \frac{263}{116} \\
\frac{222799291}{75902976} & \frac{-557897095}{12650496} & \frac{13303239}{2108416} & \frac{-85452337}{12650496} & \frac{-483}{116} & \frac{7429}{116} & \frac{23}{2} & \frac{-2783}{116} & \frac{6049}{232} \\
\frac{-19366519759}{2201186304} & \frac{71280791995}{366864384} & \frac{-2870982795}{61144064} & \frac{6728442709}{366864384} & \frac{38031}{3364} & \frac{-584953}{3364} & \frac{-1811}{58} & \frac{219131}{3364} & \frac{-476293}{6728} \\
\frac{90725693}{37951488} & \frac{2996225}{89088} & \frac{-27685119}{1054208} & \frac{-48299147}{6325248} & \frac{-273}{58} & \frac{4199}{58} & 13 & \frac{-1573}{58} & \frac{3419}{116} \\
\frac{-48778543}{37951488} & \frac{126440695}{6325248} & \frac{-3235923}{1054208} & \frac{18576595}{6325248} & \frac{105}{58} & \frac{-1615}{58} & -5 & \frac{605}{58} & \frac{-1315}{116}
\end{pmatrix},
$$

$$
\widehat{\mathbb{M}}_0^{(0)} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-2}{5} & 26 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-7}{9} & 50 & -23 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-30321797}{7115904} & \frac{356550775}{2371968} & \frac{-119849}{2059} & \frac{19535387}{4743936} & \frac{11137}{5568} & \frac{-51841}{1392} & \frac{-1225}{192} & \frac{35135}{2784} & \frac{-189319}{11136} \\
\frac{-686135525}{113854464} & \frac{223518385}{37951488} & \frac{29123307}{1054208} & \frac{529852429}{37951488} & \frac{419699}{44544} & \frac{-1585715}{11136} & \frac{-37835}{1536} & \frac{1147213}{22272} & \frac{-4875557}{89088} \\
\frac{49678009745}{3301779456} & \frac{-229880569165}{1100593152} & \frac{1437349057}{30572032} & \frac{-30031402873}{1100593152} & \frac{-25717127}{1291776} & \frac{91504967}{322944} & \frac{2219903}{44544} & \frac{-65705785}{645888} & \frac{274163633}{2583552} \\
\frac{-323472451}{56927232} & \frac{-2193835945}{18975744} & \frac{50456429}{527104} & \frac{344565875}{18975744} & \frac{262093}{22272} & \frac{-1020493}{5568} & \frac{-24181}{768} & \frac{746867}{11136} & \frac{-3104155}{44544} \\
\frac{158320529}{56927232} & \frac{402093395}{18975744} & \frac{-14262031}{527104} & \frac{-137227105}{18975744} & \frac{-101759}{22272} & \frac{404543}{5568} & \frac{9527}{768} & \frac{-290113}{11136} & \frac{1313657}{44544}
\end{pmatrix},
$$

$$\widehat{\mathbb{M}}_0^{(1)} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\[4pt]
\dfrac{1}{5} & -10 & \dfrac{24}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\[6pt]
\dfrac{4}{9} & -20 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\[6pt]
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\[4pt]
\dfrac{41108207}{14231808} & \dfrac{-636997435}{9487872} & \dfrac{21452971}{790656} & \dfrac{28883609}{18975744} & \dfrac{673}{116} & \dfrac{-1484}{87} & \dfrac{-13}{4} & \dfrac{1651}{174} & \dfrac{1785}{232} \\[6pt]
\dfrac{733595729}{227708928} & \dfrac{22172065}{4743936} & \dfrac{-1318339}{109056} & \dfrac{-334258861}{75902976} & \dfrac{-1507}{464} & \dfrac{5221}{87} & \dfrac{167}{16} & \dfrac{-14597}{696} & \dfrac{24933}{928} \\[6pt]
\dfrac{-41044567181}{6603558912} & \dfrac{2404770185}{137574144} & \dfrac{-2068713589}{91716096} & \dfrac{20579631337}{2201186304} & \dfrac{194887}{13456} & \dfrac{-331405}{2523} & \dfrac{-11707}{464} & \dfrac{952145}{20184} & \dfrac{-1183713}{26912} \\[6pt]
\dfrac{391067287}{113854464} & \dfrac{50935825}{1185984} & \dfrac{-71070457}{1581312} & \dfrac{-281285603}{37951488} & \dfrac{-1053}{232} & \dfrac{7870}{87} & \dfrac{121}{8} & \dfrac{-11539}{348} & \dfrac{17115}{464} \\[6pt]
\dfrac{-188522477}{113854464} & \dfrac{-28164515}{4743936} & \dfrac{18097199}{1581312} & \dfrac{43744885}{37951488} & \dfrac{-237}{232} & \dfrac{-1634}{87} & \dfrac{-23}{8} & \dfrac{1877}{348} & \dfrac{-7157}{464}
\end{pmatrix},$$

$$\widehat{\mathbb{M}}_0^{(r_3)} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\[2pt]
\dfrac{-119849}{889488} & \dfrac{1797735}{65888} & \dfrac{-838943}{49416} & \dfrac{-2996225}{1185984} & \dfrac{-2425}{696} & \dfrac{1355}{58} & \dfrac{97}{24} & \dfrac{-1053}{116} & \dfrac{5839}{1392} \\[6pt]
\dfrac{-3715319}{56927232} & \dfrac{55729785}{4216832} & \dfrac{-26007233}{3162624} & \dfrac{-92882975}{75902976} & \dfrac{-75175}{44544} & \dfrac{42005}{3712} & \dfrac{3007}{1536} & \dfrac{-32643}{7424} & \dfrac{181009}{89088} \\[6pt]
\dfrac{-26486629}{1650889728} & \dfrac{397299435}{122288128} & \dfrac{-185406403}{91716096} & \dfrac{-662165725}{2201186304} & \dfrac{-535925}{1291776} & \dfrac{299455}{107648} & \dfrac{21437}{44544} & \dfrac{-232713}{215296} & \dfrac{1290419}{2583552} \\[6pt]
\dfrac{-2037433}{28463616} & \dfrac{30561495}{2108416} & \dfrac{-14262031}{1581312} & \dfrac{-50935825}{37951488} & \dfrac{-41225}{22272} & \dfrac{23035}{1856} & \dfrac{1649}{768} & \dfrac{-17901}{3712} & \dfrac{99263}{44544} \\[6pt]
\dfrac{2277131}{28463616} & \dfrac{-34156965}{2108416} & \dfrac{15939917}{1581312} & \dfrac{56928275}{37951488} & \dfrac{46075}{22272} & \dfrac{-25745}{1856} & \dfrac{-1843}{768} & \dfrac{20007}{3712} & \dfrac{-110941}{44544}
\end{pmatrix},$$

$$\widehat{\mathbb{M}}_0^{(1-r_3)} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\[2pt]
\dfrac{-119849}{889488} & \dfrac{1797735}{65888} & \dfrac{-838943}{49416} & \dfrac{-2996225}{1185984} & \dfrac{-2425}{696} & \dfrac{1355}{58} & \dfrac{97}{24} & \dfrac{-1053}{116} & \dfrac{5839}{1392} \\[6pt]
\dfrac{-3715319}{56927232} & \dfrac{55729785}{4216832} & \dfrac{-26007233}{3162624} & \dfrac{-92882975}{75902976} & \dfrac{-75175}{44544} & \dfrac{42005}{3712} & \dfrac{3007}{1536} & \dfrac{-32643}{7424} & \dfrac{181009}{89088} \\[6pt]
\dfrac{-26486629}{1650889728} & \dfrac{397299435}{122288128} & \dfrac{-185406403}{91716096} & \dfrac{-662165725}{2201186304} & \dfrac{-535925}{1291776} & \dfrac{299455}{107648} & \dfrac{21437}{44544} & \dfrac{-232713}{215296} & \dfrac{1290419}{2583552} \\[6pt]
\dfrac{-2037433}{28463616} & \dfrac{30561495}{2108416} & \dfrac{-14262031}{1581312} & \dfrac{-50935825}{37951488} & \dfrac{-41225}{22272} & \dfrac{23035}{1856} & \dfrac{1649}{768} & \dfrac{-17901}{3712} & \dfrac{99263}{44544} \\[6pt]
\dfrac{2277131}{28463616} & \dfrac{-34156965}{2108416} & \dfrac{15939917}{1581312} & \dfrac{56928275}{37951488} & \dfrac{46075}{22272} & \dfrac{-25745}{1856} & \dfrac{-1843}{768} & \dfrac{20007}{3712} & \dfrac{-110941}{44544}
\end{pmatrix}.$$

5   Summary

In this paper we presented `epsilon`. The tool `epsilon` is an efficient implementation of an algorithm proposed by R.N. Lee to reduce a system of ordinary differential equations with rational coefficients to a canonical form, where the right hand side is proportional to ϵ.

In physically relevant situations, the small parameter ϵ usually is a regulator in dimensional regularization (e.g. in $d = 4 - 2\epsilon$ dimensions). We showed its applicability in

a three-loop example and demonstrated the possibility to reduce systems with complex singular points.

Acknowledgments

A   Transformations

A.1   Balances

The main building blocks of Lee's algorithm are balances. In this section we describe how they act on a system in the form (2.2) which is also assumed to be in block-triangular form (2.6).

The projector $\mathbb{P}$ is determined by its impact on the active block $\mathbb{C}$. Hence, it has to be of the form

$$\mathbb{P} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbb{Q} & 0 \\ 0 & 0 & 0 \end{pmatrix} .$$

Naturally, this also modifies the blocks $\mathbb{B}$ and $\mathbb{E}$. In this appendix we omit writing down the ϵ-dependencies explicitly.

First, we consider a balance between two singularities $x_1, x_2 < \infty$ (2.7a):

$$\mathbb{T} = \mathcal{B}(\mathbb{P}, x_1, x_2) , \quad \mathbb{T}^{-1} = \mathcal{B}(\mathbb{P}, x_2, x_1) .$$

The transformation (2.5) can now be written in terms of the coefficient matrices.

We find for the active block

$$\widetilde{\mathbb{C}}_0^{(x_1)} = \mathbb{C}_0^{(x_1)} - \sum_{n\geq 0} \frac{1}{(x_2 - x_1)^n} \mathbb{Q}\mathbb{C}_n^{(x_1)}\overline{\mathbb{Q}} + \sum_{x_j\in S\setminus\{x_1\}} \sum_{n\geq 0} \frac{x_1 - x_2}{(x_1 - x_j)^{n+1}}\overline{\mathbb{Q}}\mathbb{C}_n^{(x_j)}\mathbb{Q}$$
$$+ (x_1 - x_2) \sum_{n\geq 0} x_1^n \overline{\mathbb{Q}}\mathbb{C}_n\mathbb{Q} + \mathbb{Q}\,, \tag{A.1a}$$

$$\widetilde{\mathbb{C}}_{k>0}^{(x_1)} = \mathbb{C}_k^{(x_1)} + (x_1 - x_2)\overline{\mathbb{Q}}\mathbb{C}_{k-1}^{(x_1)}\mathbb{Q} - \sum_{n\geq 0} \frac{1}{(x_2 - x_1)^n}\mathbb{Q}\mathbb{C}_{n+k}^{(x_1)}\overline{\mathbb{Q}}\,, \tag{A.1b}$$

$$\widetilde{\mathbb{C}}_0^{(x_2)} = \mathbb{C}_0^{(x_2)} - \sum_{n\geq 0} \frac{1}{(x_1 - x_2)^n}\overline{\mathbb{Q}}\mathbb{C}_n^{(x_2)}\mathbb{Q} + \sum_{x_j\in S\setminus\{x_2\}} \sum_{n\geq 0} \frac{x_2 - x_1}{(x_2 - x_j)^{n+1}}\mathbb{Q}\mathbb{C}_n^{(x_j)}\overline{\mathbb{Q}}$$
$$+ (x_2 - x_1) \sum_{n\geq 0} x_2^n \mathbb{Q}\mathbb{C}_n\overline{\mathbb{Q}} - \mathbb{Q}\,, \tag{A.1c}$$

$$\widetilde{\mathbb{C}}_{k>0}^{(x_2)} = \mathbb{C}_k^{(x_2)} + (x_2 - x_1)\mathbb{Q}\mathbb{C}_{k-1}^{(x_2)}\overline{\mathbb{Q}} - \sum_{n\geq 0} \frac{1}{(x_1 - x_2)^n}\overline{\mathbb{Q}}\mathbb{C}_{n+k}^{(x_2)}\mathbb{Q}\,, \tag{A.1d}$$

$$\widetilde{\mathbb{C}}_k^{(x_j\neq x_1,x_2)} = \mathbb{C}_k^{(x_j)} + \sum_{n\geq 0} \frac{x_2 - x_1}{(x_1 - x_j)^{n+1}}\overline{\mathbb{Q}}\mathbb{C}_{n+k}^{(x_j)}\mathbb{Q} + \sum_{n\geq 0} \frac{x_1 - x_2}{(x_2 - x_j)^{n+1}}\mathbb{Q}\mathbb{C}_{n+k}^{(x_j)}\overline{\mathbb{Q}}\,, \tag{A.1e}$$

$$\widetilde{\mathbb{C}}_k = \mathbb{C}_k + (x_1 - x_2) \sum_{n\geq 0} \left\{ x_1^n\overline{\mathbb{Q}}\mathbb{C}_{k+n+1}\mathbb{Q} - x_2^n\mathbb{Q}\mathbb{C}_{k+n+1}\overline{\mathbb{Q}} \right\}\,, \tag{A.1f}$$

and for the off-diagonal blocks

$$\widetilde{\mathbb{B}}_0^{(x_2)} = \mathbb{B}_0^{(x_2)} + \sum_{x_j \in S \setminus \{x_2\}} \sum_{n \geq 0} \frac{x_2 - x_1}{(x_2 - x_j)^{n+1}} \mathbb{Q} \mathbb{B}_n^{(x_j)} + (x_2 - x_1) \sum_{n \geq 0} x_2^n \mathbb{Q} \mathbb{B}_n,$$

$$\widetilde{\mathbb{B}}_{k>0}^{(x_2)} = \mathbb{B}_k^{(x_2)} + (x_2 - x_1) \mathbb{Q} \mathbb{B}_{k-1}^{(x_2)},$$

$$\widetilde{\mathbb{B}}_k^{(x_j \neq x_2)} = \mathbb{B}_k^{(x_j)} + \sum_{n \geq 0} \frac{x_1 - x_2}{(x_2 - x_j)^{n+1}} \mathbb{Q} \mathbb{B}_{n+k}^{(x_j)},$$

$$\widetilde{\mathbb{B}}_k = \mathbb{B}_k + (x_2 - x_1) \sum_{n \geq 0} x_2^n \mathbb{Q} \mathbb{B}_{k+n+1},$$

$$\widetilde{\mathbb{E}}_0^{(x_1)} = \mathbb{E}_0^{(x_1)} + \sum_{x_j \in S \setminus \{x_1\}} \sum_{n \geq 0} \frac{x_1 - x_2}{(x_1 - x_j)^{n+1}} \mathbb{E}_n^{(x_j)} \mathbb{Q} + (x_1 - x_2) \sum_{n \geq 0} x_1^n \mathbb{E}_n \mathbb{Q},$$

$$\widetilde{\mathbb{E}}_{k>0}^{(x_1)} = \mathbb{E}_k^{(x_1)} + (x_1 - x_2) \mathbb{E}_{k-1}^{(x_1)} \mathbb{Q},$$

$$\widetilde{\mathbb{E}}_k^{(x_j \neq x_1)} = \mathbb{E}_k^{(x_j)} + \sum_{n \geq 0} \frac{x_2 - x_1}{(x_1 - x_j)^{n+1}} \mathbb{E}_{n+k}^{(x_j)} \mathbb{Q}$$

$$\widetilde{\mathbb{E}}_k = \mathbb{E}_k + (x_1 - x_2) \sum_{n \geq 0} x_1^n \mathbb{E}_{k+n+1} \mathbb{Q}.$$

All other blocks are unaffected.

Next, we consider the case $x_1 < \infty$, $x_2 = \infty$, i.e.

$$\mathbb{T} = \mathcal{B}(\mathbb{P}, x_1, \infty), \quad \mathbb{T}^{-1} = \mathcal{B}(\mathbb{P}, \infty, x_1).$$

In that case, the balances are given by (2.7b) and (2.7c). Here the active block transforms

as

$$\widetilde{\mathbb{C}}_0^{(x_1)} = \mathbb{C}_0^{(x_1)} - \overline{\mathbb{Q}}\mathbb{C}_0^{(x_1)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_0^{(x_1)}\overline{\mathbb{Q}} + \mathbb{Q}\mathbb{C}_1^{(x_1)}\overline{\mathbb{Q}}$$
$$+ \sum_{x_j \in S \setminus \{x_1\}} \sum_{n=0}^{\infty} \frac{1}{(x_1 - x_j)^{n+1}} \overline{\mathbb{Q}}\mathbb{C}_n^{(x_j)}\mathbb{Q} + \sum_{n=0}^{\infty} x_1^n \overline{\mathbb{Q}}\mathbb{C}_n\mathbb{Q} + \mathbb{Q}\,,$$

$$\widetilde{\mathbb{C}}_{k>0}^{(x_1)} = \mathbb{C}_k^{(x_1)} - \overline{\mathbb{Q}}\mathbb{C}_k^{(x_1)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_k^{(x_1)}\overline{\mathbb{Q}} + \mathbb{Q}\mathbb{C}_{k+1}^{(x_1)}\overline{\mathbb{Q}} + \overline{\mathbb{Q}}\mathbb{C}_{k-1}^{(x_1)}\mathbb{Q}\,,$$

$$\widetilde{\mathbb{C}}_k^{(x_j \neq x_1)} = \mathbb{C}_k^{(x_j)} - \overline{\mathbb{Q}}\mathbb{C}_k^{(x_j)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_k^{(x_j)}\overline{\mathbb{Q}} + \mathbb{Q}\mathbb{C}_{k+1}^{(x_j)}\overline{\mathbb{Q}} + (x_j - x_1)\mathbb{Q}\mathbb{C}_k^{(x_j)}\overline{\mathbb{Q}}$$
$$- \sum_{n=0}^{\infty} \frac{1}{(x_1 - x_j)^{n+1}} \overline{\mathbb{Q}}\mathbb{C}_{n+k}^{(x_j)}\mathbb{Q}\,,$$

$$\widetilde{\mathbb{C}}_0 = \mathbb{C}_0 - \overline{\mathbb{Q}}\mathbb{C}_0\mathbb{Q} - (1 + x_1)\mathbb{Q}\mathbb{C}_0\overline{\mathbb{Q}} + \sum_{x_j \in S} \mathbb{Q}\mathbb{C}_0^{(x_j)}\overline{\mathbb{Q}} + \sum_{n=0}^{\infty} x_1^n \overline{\mathbb{Q}}\mathbb{C}_{n+1}\mathbb{Q}\,,$$

$$\widetilde{\mathbb{C}}_{k>0} = \mathbb{C}_k - \overline{\mathbb{Q}}\mathbb{C}_k\mathbb{Q} - (1 + x_1)\mathbb{Q}\mathbb{C}_k\overline{\mathbb{Q}} + \mathbb{Q}\mathbb{C}_{k-1}\overline{\mathbb{Q}} + \sum_{n=0}^{\infty} x_1^n \overline{\mathbb{Q}}\mathbb{C}_{k+n+1}\mathbb{Q}\,,$$

and the blocks $\mathbb{B}$ and $\mathbb{E}$ as

$$\widetilde{\mathbb{B}}_k^{(x_j)} = \mathbb{B}_k^{(x_j)} - \mathbb{Q}\mathbb{B}_k^{(x_j)} + \mathbb{Q}\mathbb{B}_{k+1}^{(x_j)} + (x_j - x_1)\mathbb{Q}\mathbb{B}_k^{(x_j)}\,,$$
$$\widetilde{\mathbb{B}}_0 = \mathbb{B}_0 - (1 + x_1)\mathbb{Q}\mathbb{B}_0 + \sum_{x_j \in S} \mathbb{Q}\mathbb{B}_0^{(x_j)}\,,$$

$$\widetilde{\mathbb{B}}_{k>0} = \mathbb{B}_k - (1 + x_1)\mathbb{Q}\mathbb{B}_k + \mathbb{Q}\mathbb{B}_{k-1}\,,$$

$$\widetilde{\mathbb{E}}_0^{(x_1)} = \mathbb{E}_0^{(x_1)} - \mathbb{E}_0^{(x_1)}\mathbb{Q} + \sum_{x_j \in S \setminus \{x_1\}} \sum_{n \geq 0} \frac{1}{(x_1 - x_j)^{n+1}} \mathbb{E}_n^{(x_j)}\mathbb{Q} + \sum_{n \geq 0} x_1^n \mathbb{E}_n\mathbb{Q}\,,$$

$$\widetilde{\mathbb{E}}_{k>0}^{(x_1)} = \mathbb{E}_k^{(x_1)} - \mathbb{E}_k^{(x_1)}\mathbb{Q} + \mathbb{E}_{k-1}^{(x_1)}\mathbb{Q}\,,$$

$$\widetilde{\mathbb{E}}_k^{(x_j \neq x_1)} = \mathbb{E}_k^{(x_j)} - \mathbb{E}_k^{(x_j)}\mathbb{Q} - \sum_{n \geq 0} \frac{1}{(x_1 - x_j)^{n+1}} \mathbb{E}_{n+k}^{(x_j)}\mathbb{Q}\,,$$

$$\widetilde{\mathbb{E}}_k = \mathbb{E}_k - \mathbb{E}_k\mathbb{Q} + \sum_{n \geq 0} x_1^n \mathbb{E}_{k+n+1}\mathbb{Q}\,.$$

Finally, we consider the case $x_1 = \infty$, $x_2 < \infty$, i.e.

$$\mathbb{T} = \mathcal{B}(\mathbb{P}, \infty, x_2)\,, \quad \mathbb{T}^{-1} = \mathcal{B}(\mathbb{P}, x_2, \infty)\,.$$

This is similar to the previous case. The active block transforms to

$$\widetilde{\mathbb{C}}_0^{(x_2)} = \mathbb{C}_0^{(x_2)} - \overline{\mathbb{Q}}\mathbb{C}_0^{(x_2)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_0^{(x_2)}\overline{\mathbb{Q}} + \overline{\mathbb{Q}}\mathbb{C}_1^{(x_2)}\mathbb{Q}$$
$$+ \sum_{x_j \in S \setminus \{x_2\}} \sum_{n \geq 0} \frac{1}{(x_2 - x_j)^{n+1}} \mathbb{Q}\mathbb{C}_n^{(x_j)}\overline{\mathbb{Q}} + \sum_{n \geq 0} x_2^n \mathbb{Q}\mathbb{C}_n\overline{\mathbb{Q}} - \mathbb{Q},$$

$$\widetilde{\mathbb{C}}_{k>0}^{(x_2)} = \mathbb{C}_k^{(x_2)} - \overline{\mathbb{Q}}\mathbb{C}_k^{(x_2)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_k^{(x_2)}\overline{\mathbb{Q}} + \mathbb{Q}\mathbb{C}_{k-1}^{(x_2)}\overline{\mathbb{Q}} + \overline{\mathbb{Q}}\mathbb{C}_{k+1}^{(x_2)}\mathbb{Q},$$

$$\widetilde{\mathbb{C}}_k^{(x_j \neq x_2)} = \mathbb{C}_k^{(x_j)} - \overline{\mathbb{Q}}\mathbb{C}_k^{(x_j)}\mathbb{Q} - \mathbb{Q}\mathbb{C}_k^{(x_j)}\overline{\mathbb{Q}} + \overline{\mathbb{Q}}\mathbb{C}_{k+1}^{(x_j)}\mathbb{Q} + (x_j - x_2)\overline{\mathbb{Q}}\mathbb{C}_k^{(x_j)}\mathbb{Q}$$
$$- \sum_{n \geq 0} \frac{1}{(x_2 - x_j)^{n+1}} \mathbb{Q}\mathbb{C}_{n+k}^{(x_j)}\overline{\mathbb{Q}},$$

$$\widetilde{\mathbb{C}}_0 = \mathbb{C}_0 - \overline{\mathbb{Q}}\mathbb{C}_0\mathbb{Q} - \mathbb{Q}\mathbb{C}_0\overline{\mathbb{Q}} + \sum_{n \geq 0} x_2^n \mathbb{Q}\mathbb{C}_{n+1}\overline{\mathbb{Q}} - x_2\overline{\mathbb{Q}}\mathbb{C}_0\mathbb{Q} + \sum_{x_j \in S} \overline{\mathbb{Q}}\mathbb{C}_0^{(x_j)}\mathbb{Q},$$

$$\widetilde{\mathbb{C}}_{k>0} = \mathbb{C}_k - \overline{\mathbb{Q}}\mathbb{C}_k\mathbb{Q} - \mathbb{Q}\mathbb{C}_k\overline{\mathbb{Q}} + \sum_{n \geq 0} x_2^n \mathbb{Q}\mathbb{C}_{k+n+1}\overline{\mathbb{Q}} + \overline{\mathbb{Q}}\mathbb{C}_{k-1}\mathbb{Q} - x_2\overline{\mathbb{Q}}\mathbb{C}_k\mathbb{Q},$$

and the blocks $\mathbb{B}$ and $\mathbb{E}$ transform as

$$\widetilde{\mathbb{B}}_0^{(x_2)} = \mathbb{B}_0^{(x_2)} - \mathbb{Q}\mathbb{B}_0^{(x_2)} + \sum_{x_j \in S \setminus \{x_2\}} \sum_{n \geq 0} \frac{1}{(x_2 - x_j)^{n+1}} \mathbb{Q}\mathbb{B}_n^{(x_j)} + \sum_{n \geq 0} x_2^n \mathbb{Q}\mathbb{B}_n,$$

$$\widetilde{\mathbb{B}}_{k>0}^{(x_2)} = \mathbb{B}_k^{(x_2)} - \mathbb{Q}\mathbb{B}_k^{(x_2)} + \mathbb{Q}\mathbb{B}_{k-1}^{(x_2)},$$

$$\widetilde{\mathbb{B}}_k^{(x_j \neq x_2)} = \mathbb{B}_k^{(x_j)} - \mathbb{Q}\mathbb{B}_k^{(x_j)} - \sum_{n \geq 0} \frac{1}{(x_2 - x_j)^{n+1}} \mathbb{Q}\mathbb{B}_{n+k}^{(x_j)},$$

$$\widetilde{\mathbb{B}}_k = \mathbb{B}_k - \mathbb{Q}\mathbb{B}_k + \sum_{n \geq 0} x_2^n \mathbb{Q}\mathbb{B}_{k+n+1},$$

$$\widetilde{\mathbb{E}}_k^{(x_j)} = \mathbb{E}_k^{(x_j)} - \mathbb{E}_k^{(x_j)}\mathbb{Q} + \mathbb{E}_{k+1}^{(x_j)}\mathbb{Q} + (x_j - x_2)\mathbb{E}_k^{(x_j)}\mathbb{Q},$$

$$\widetilde{\mathbb{E}}_0 = \mathbb{E}_0 - (1 + x_2)\mathbb{E}_0\mathbb{Q} + \sum_{x_j \in S} \mathbb{E}_0^{(x_j)}\mathbb{Q},$$

$$\widetilde{\mathbb{E}}_{k>0} = \mathbb{E}_k - (1 + x_2)\mathbb{E}_k\mathbb{Q} + \mathbb{E}_{k-1}\mathbb{Q}.$$

A.2 Fuchsification of off-diagonal blocks

In this appendix we consider the transformation of the off-diagonal block $\mathbb{B}$ to Fuchsian form. We assume that the diagonal blocks $\mathbb{A}$ and $\mathbb{C}$ are already in ϵ-form. The transformation needed has the form

$$\mathbb{T} = \mathcal{L}(x_1, k, \mathbb{G}), \quad \mathbb{T}^{-1} = \mathcal{L}(x_1, k, -\mathbb{G}), \tag{A.2}$$

where we used the definitions (2.8).

In addition to block $\mathbb{B}$ only block $\mathbb{D}$ is influenced by this transformation, i.e. the blocks $\mathbb{A}$ and $\mathbb{C}$ are unaffected.

The transformation (2.5), with $\mathbb{T}$ defined as in (A.2) translates to rules for the coefficient matrices. For $x_1 < \infty$ we find

$$\widetilde{\mathbb{B}}_k^{(x_1)} = \mathbb{B}_k^{(x_1)} + \mathbb{C}_0^{(x_1)}\widehat{\mathbb{G}} - \widehat{\mathbb{G}}\mathbb{A}_0^{(x_1)} + k\widehat{\mathbb{G}}\,, \tag{A.3a}$$

$$\widetilde{\mathbb{B}}_{n<k}^{(x_1)} = \mathbb{B}_n^{(x_1)} - \sum_{x_j \in S\backslash\{x_1\}} \frac{\mathbb{C}_0^{(x_j)}\widehat{\mathbb{G}} - \widehat{\mathbb{G}}\mathbb{A}_0^{(x_j)}}{(x_j - x_1)^{k-n}}\,, \tag{A.3b}$$

$$\widetilde{\mathbb{B}}_{n>k}^{(x_1)} = \mathbb{B}_n^{(x_1)}\,, \tag{A.3c}$$

$$\widetilde{\mathbb{B}}_0^{(x_j \neq x_1)} = \mathbb{B}_0^{(x_j)} + \frac{\mathbb{C}_0^{(x_j)}\widehat{\mathbb{G}} - \widehat{\mathbb{G}}\mathbb{A}_0^{(x_j)}}{(x_j - x_1)^k}\,, \tag{A.3d}$$

$$\widetilde{\mathbb{B}}_{n>0}^{(x_j \neq x_1)} = \mathbb{B}_n^{(x_j)}\,, \tag{A.3e}$$

$$\widetilde{\mathbb{B}}_n = \mathbb{B}_n\,, \tag{A.3f}$$

and

$$\widetilde{\mathbb{D}}_{n<k}^{(x_1)} = \mathbb{D}_n^{(x_1)} - \sum_{x_j \in S\backslash\{x_1\}} \sum_{i\geq 0} (-1)^i \frac{\binom{k+i-n-1}{i}}{(x_j - x_1)^{k+i-n}} \mathbb{E}_i^{(x_j)}\widehat{\mathbb{G}}$$

$$+ \sum_{i\geq 0} x_1^i \binom{i+k-n-1}{i} \mathbb{E}_{i+k-n-1}\widehat{\mathbb{G}}\,,$$

$$\widetilde{\mathbb{D}}_{n\geq k}^{(x_1)} = \mathbb{D}_n^{(x_1)} + \mathbb{E}_{n-k}^{(x_1)}\widehat{\mathbb{G}}\,,$$

$$\widetilde{\mathbb{D}}_n^{(x_j \neq 1)} = \mathbb{D}_n^{(x_j)} + (-1)^k \sum_{i\geq 0} \frac{\binom{k+i-1}{i}}{(x_1 - x_j)^{k+i}} \mathbb{E}_{n+i}^{(x_j)}\widehat{\mathbb{G}}\,,$$

$$\widetilde{\mathbb{D}}_n = \mathbb{D}_n + \sum_{m\geq 0} (-1)^m \sum_{i\geq 0} x_1^{m+i} \binom{n+m}{n}\binom{i+n+m+k}{i} \mathbb{E}_{i+n+m+k}\widehat{\mathbb{G}}\,.$$

The case $x_1 = \infty$ leads to

$$\widetilde{\mathbb{B}}_{k-1} = \mathbb{B}_{k-1} + \sum_{x_j \in S} \left[\mathbb{C}_0^{(x_j)} \mathbb{G} - \widehat{\mathbb{G}} \mathbb{A}^{(x_j)} \right] - k \mathbb{G} \,,$$

$$\widetilde{\mathbb{B}}_{n<k-1} = \mathbb{B}_n + \sum_{x_j \in S} x_j^{k-n-1} \left[\mathbb{C}_0^{(x_j)} \widehat{\mathbb{G}} - \widehat{\mathbb{G}} \mathbb{A}_0^{(x_j)} \right] \,,$$

$$\widetilde{\mathbb{B}}_{n>k-1} = \mathbb{B}_n \,,$$

$$\widetilde{\mathbb{B}}_0^{(x_j)} = \mathbb{B}_0^{(x_j)} + x_j^k \left[\mathbb{C}_0^{(x_j)} \widehat{\mathbb{G}} - \widehat{\mathbb{G}} \mathbb{A}_0^{(x_j)} \right] \,,$$

$$\widetilde{\mathbb{B}}_{n>0}^{(x_j)} = \mathbb{B}_n^{(x_j)} \,,$$

and

$$\widetilde{\mathbb{D}}_n^{(x_j)} = \mathbb{D}_n^{(x_j)} + \sum_{i=0}^{k} x_j^i \binom{k}{k-i} \mathbb{E}_{n+k-i}^{(x_j)} \widehat{\mathbb{G}} \,,$$

$$\widetilde{\mathbb{D}}_{n<k} = \mathbb{D}_n + \sum_{x_j \in S} \sum_{m=0}^{k-n-1} \sum_{i=0}^{m} (-1)^{k-n-m-1} x_j^{k-n-i-1} \binom{k-m-1}{n} \binom{k}{k+i-m} \mathbb{E}_i^{(x_j)} \widehat{\mathbb{G}} \,,$$

$$\widetilde{\mathbb{D}}_{n \geq k} = \mathbb{D}_n + \mathbb{E}_{n-k} \widehat{\mathbb{G}} \,.$$

A.3 ϵ-Factorization

The transformation required in the ϵ-factorization is x-independent. Hence, every coefficient matrix in (2.2) transforms the same and the transformation rule (2.5) becomes a similarity transformation

$$\widetilde{\mathbb{M}}(x, \epsilon) = \mathbb{T}^{-1}(\epsilon) \mathbb{M}(x, \epsilon) \mathbb{T}(\epsilon) \,.$$

The matrices $\mathbb{T}(\epsilon)$ and $\mathbb{T}^{-1}(\epsilon)$ have the form

$$\mathbb{T}(\epsilon) = \begin{pmatrix} \mathbb{1} & 0 & 0 \\ 0 & \widehat{\mathbb{T}}(\epsilon) & 0 \\ 0 & 0 & \mathbb{1} \end{pmatrix} \,, \quad \mathbb{T}^{-1}(\epsilon) = \begin{pmatrix} \mathbb{1} & 0 & 0 \\ 0 & \widehat{\mathbb{T}}^{-1}(\epsilon) & 0 \\ 0 & 0 & \mathbb{1} \end{pmatrix} \,,$$

with $\widehat{\mathbb{T}}(\epsilon)$ and $\widehat{\mathbb{T}}^{-1}(\epsilon)$ corresponding to block $\mathbb{C}$. Using the block-triangular structure (2.6) yields

$$\begin{pmatrix} \widetilde{\mathbb{A}}(x, \epsilon) & 0 & 0 \\ \widetilde{\mathbb{B}}(x, \epsilon) & \widetilde{\mathbb{C}}(x, \epsilon) & 0 \\ \widetilde{\mathbb{D}}(x, \epsilon) & \widetilde{\mathbb{E}}(x, \epsilon) & \widetilde{\mathbb{F}}(x, \epsilon) \end{pmatrix} = \begin{pmatrix} \mathbb{A}(x, \epsilon) & 0 & 0 \\ \mathbb{T}^{-1}(\epsilon) \mathbb{B}(x, \epsilon) & \mathbb{T}^{-1}(\epsilon) \mathbb{C}(x, \epsilon) \mathbb{T}(\epsilon) & 0 \\ \mathbb{D}(x, \epsilon) & \mathbb{E}(x, \epsilon) \mathbb{T}(\epsilon) & \mathbb{F}(x, \epsilon) \end{pmatrix} \,.$$

Thus, besides block $\mathbb{C}$, block $\mathbb{B}$ and block $\mathbb{E}$ are influenced as well.

References

[1] F. V. Tkachov, *A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions*, Phys. Lett. **B100** (1981) 65–68.

[2] K. G. Chetyrkin and F. V. Tkachov, *Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops*, Nucl. Phys. **B192** (1981) 159–204.

[3] V. A. Smirnov, *Analytic tools for Feynman integrals*, Springer Tracts Mod. Phys. **250** (2012) 1–296.

[4] A. V. Kotikov, *Differential equations method: New technique for massive Feynman diagrams calculation*, Phys. Lett. **B254** (1991) 158–164.

[5] A. V. Kotikov, *Differential equations method: The Calculation of vertex type Feynman diagrams*, Phys. Lett. **B259** (1991) 314–322.

[6] A. V. Kotikov, *Differential equation method: The Calculation of N point Feynman diagrams*, Phys. Lett. **B267** (1991) 123–127. [Erratum: Phys. Lett.B295,409(1992)].

[7] J. M. Henn, *Multiloop integrals in dimensional regularization made simple*, Phys. Rev. Lett. **110** (2013) 251601, `arXiv:1304.1806 [hep-th]`.

[8] R. N. Lee, *Reducing differential equations for multiloop master integrals*, JHEP **04** (2015) 108, `arXiv:1411.0911 [hep-ph]`.

[9] O. Gituliar and V. Magerya, *Fuchsia and master integrals for splitting functions from differential equations in QCD*, PoS **LL2016** (2016) 030, `arXiv:1607.00759 [hep-ph]`.

[10] O. Gituliar and V. Magerya, *Fuchsia: a tool for reducing differential equations for Feynman master integrals to epsilon form*, `arXiv:1701.04269 [hep-ph]`.

[11] R. H. Lewis, "Computer Algebra System Fermat." `https://home.bway.net/lewis`. Accessed: 2016-12-19.

[12] J. Moser, *The Order of a Singularity in Fuchs' Theory*, Mathematische Zeitschrift **72** (1959/60) 379–398.

[13] C. W. Bauer, A. Frink, and R. Kreckel, *Introduction to the GiNaC framework for symbolic computation within the C++ programming language*, J. Symb. Comput. **33** (2000) 1, `arXiv:cs/0004015 [cs-sc]`.

[14] "PStreams." `http://pstreams.sourceforge.net`. Accessed: 2016-12-19.

[15] A. B. Goncharov, M. Spradlin, C. Vergu, and A. Volovich, *Classical Polylogarithms for Amplitudes and Wilson Loops*, *Phys. Rev. Lett.* **105** (2010) 151605, `arXiv:1006.5703 [hep-th]`.

[16] E. Remiddi and J. A. M. Vermaseren, *Harmonic polylogarithms*, *Int. J. Mod. Phys.* **A15** (2000) 725–754, `arXiv:hep-ph/9905237 [hep-ph]`.

[17] D. Maitre, *HPL, a mathematica implementation of the harmonic polylogarithms*, *Comput. Phys. Commun.* **174** (2006) 222–240, `arXiv:hep-ph/0507152 [hep-ph]`.

[18] J. A. M. Vermaseren, *New features of FORM*, `arXiv:math-ph/0010025 [math-ph]`.

[19] D. Binosi, J. Collins, C. Kaufhold, and L. Theussl, *JaxoDraw: A Graphical user interface for drawing Feynman diagrams. Version 2.0 release notes*, *Comput. Phys. Commun.* **180** (2009) 1709–1715, `arXiv:0811.4113 [hep-ph]`.

[20] J. A. M. Vermaseren, *Axodraw*, *Comput. Phys. Commun.* **83** (1994) 45–58.