Difference Equations for 5-loop Massive Vacuum Integrals Using Syzygies

Master's thesis

submitted to the

Faculty of Physics,

Bielefeld University

by

Thomas Luthe

Supervisor & 1st Assessor:	Prof. Dr. York Schröder
2nd Assessor:	Dr. Cedric Studerus

Bielefeld, November 19th, 2012

Abstract

In many precision calculations within the Standard Model of particle physics so-called Feynman integrals over loop momenta need to be calculated. Of these integrals, I consider the subclass of fully massive vacuum integrals with a single mass scale up to 5 loops. I have implemented in C++ a recently developed novel approach [1] to the well known integration by parts relations, which I use to generate linear equations in these integrals that do not exhibit so-called raised exponents. The resulting system of equations is subjected to a Laporta algorithm to obtain difference equations for a basis of master integrals. Many of the difference equations at the 5-loop level have been generated in this way, some of which were previously unknown. Furthermore, I am able to for the first time present a full set of orders for the difference equations of the 5-loop master integrals in the class of integrals considered.

Contents

1 Introduction and motivation

In many areas of physics advances in experimental procedures create a need for constant improvement of theoretical predictions to match the increasing precision. A common approach in Quantum Field Theory (QFT) for the calculation of observables is the expansion in a small coupling parameter. Within this framework, higher precision is achieved by increasing the order of the coupling parameter up to which the expansion is performed. Higher-order terms require the calculation of so-called Feynman integrals with a larger number of loop momenta which need to be integrated. One of the major problems that this perturbative approach faces is that the number and complexity of integrals increase significantly with the number of loops. This creates a need for automation of calculations, which in turn requires algorithms that are general enough to apply to a wide range of integrals. One such algorithm will be the cornerstone of this thesis.

The first step in a higher-loop order calculation is usually to identify and classify all Feynman integrals that will appear. The typically very large number of integrals one encounters is then reduced by expressing all integrals in terms of comparatively few master integrals. There are already several computer programs published [2–4] which accomplish this task via Laporta's algorithm [5]. As a final step, the remaining master integrals need to be evaluated, either analytically or numerically. One approach to this last step is the use of so-called difference equations [5], which roughly can be thought of as a discrete version of differential equations for the master integrals. In this thesis, I will focus on the generation of such difference equations for a specific class of integrals using a new method building on the ideas of Gluza, Kajda and Kosower [1] as well as Schabinger [6]. The class of integrals I consider is that of fully massive vacuum integrals with a single mass scale, also called tadpoles, which appear in several calculations in perturbative QFT (see e.g. [7–9]). These integrals are already known up to 4 loops [5, 10, 11], while work on the 5-loop level has recently begun [12]. I will continue this work by attempting to generate the difference equations at 5 loops.

The thesis is structured as follows: I will first give some motivation as to why massive vacuum integrals are worthwhile studying at high loop levels by showing how they emerge from the calculation of the beta-function of Quantum Chromodynamics. Sections 2 and 3 cover the classification of integrals and the fundamentals of a reduction to master integrals as well as difference equations. In section 4, I will present in detail the ideas of Gluza, Kajda, Kosower and Schabinger along with my own modifications and extensions of the algorithm. The particulars of the implementation of the algorithm in C++ will be discussed in section 5. Section 6 will then cover the application to the massive vacuum integrals including results as well as some runtimes of the code and current limitations. Some of the latter will be addressed in section 7, where I list potential future improvements of the code and algorithm before making some final remarks in section 8.

1.1 The β-function of QCD

There are many precision calculations in particle physics from which massive vacuum Feynman integrals emerge, one of the most famous of which is the β-function of Quantum Chromodynamics (QCD). In the following I will show some of the steps needed to arrive at the vacuum integrals for this particular example.

Within the Standard Model of Particle Physics (SM) the strong interactions are described by Yang-Mills theory [13], which is a non-Abelian gauge theory with the symmetry group $SU(N_c)$ ($N_c = 3$ for QCD). The QCD Lagrangian is thus given by

$$\mathcal{L}_{\mathrm{QCD}} = \sum_{f=1}^{N_f} \sum_{i,j=1}^{N_c} \bar{\Psi}_{f,i} \left[i\gamma^\mu D_\mu - m_f\right]_{ij} \Psi_{f,j} - \frac{1}{4} F^a_{\mu\nu} F^{a\mu\nu}. \tag{1.1}$$

The field strength tensor $F^a_{\mu\nu}$ reads

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu, \tag{1.2}$$

and D_μ is the covariant derivative:

$$[D_\mu]_{ij} = \delta_{ij}\partial_\mu - ig A^a_\mu T^a_{ij}. \tag{1.3}$$

The quarks are represented by the spinors $\Psi_{f,j}$, where f is the flavour index and $j = 1, \ldots, N_c$ the colour index, while gluons are denoted as A^a_μ where $a = 1, \ldots, N_c^2 - 1$. g is the coupling constant of the strong interaction and the matrices T^a represent the generators of $SU(N_c)$ and fulfil

$$\left[T^a, T^b\right] = i f^{abc} T^c, \tag{1.4}$$

where the f^{abc} are called structure constants. After introducing a gauge fixing term via the Faddeev-Popov method [14] the Lagrangian reads

$$\mathcal{L}_{\mathrm{QCD,GF}} = \mathcal{L}_{\mathrm{QCD}} - \frac{1}{2\xi}\left(\partial^\mu A^a_\mu\right)^2 - \bar{c}^a \partial^\mu \left(g f^{abc} A^b_\mu + \delta^{ac}\partial_\mu\right) c^b, \tag{1.5}$$

where ξ is the gauge parameter and one has at the same time introduced the unphysical ghost fields c^a via the Faddeev-Popov determinant. If one starts to calculate amplitudes to some loop level n at this point, one will encounter ultraviolet (UV) divergences that need to first be regularised and then removed via renormalisation. In the following, I will assume that dimensional regularisation [15] is chosen, although other regularisation schemes such as momentum cutoffs are also possible. In dimensional regularisation the space-time dimension is (typically) chosen as $d = 4 - 2\epsilon$ and the UV divergences then appear as $\frac{1}{\epsilon}$ poles. Since QCD is a renormalisable theory [16], it is then always possible to remove these divergences perturbatively up to n loops by rescaling the quantities that appear in the Lagrangian:

$$\begin{aligned}
\Phi_B &= \sqrt{Z_\Phi}\Phi_R, \quad &\Phi \in \{\Psi, A, c\}, \\
\Phi_B &= Z_\Phi \Phi_R, \quad &\Phi \in \{m, \xi\}, \\
g_B &= Z_g g_R \mu^\epsilon.
\end{aligned} \tag{1.6}$$

Here the subscript B denotes bare quantities as they appear in Eq. (1.5) and R denotes the renormalised quantities. One introduces the arbitrary mass scale μ, because in dimensional regularisation g_B has the mass dimension ϵ and one would like a dimensionless coupling parameter g_R to expand terms in. The Z_Φ are called renormalisation constants (RCs) and are typically expressed as

$$Z_\Phi = 1 + \delta Z_\Phi. \tag{1.7}$$

The Lagrangian can then be split into

$$\mathcal{L} = \mathcal{L}_R + \mathcal{L}_{CT}, \tag{1.8}$$

where $\mathcal{L}_R$ is of the same form as Eq. (1.5) and $\mathcal{L}_{CT}$ consists of all so-called counterterms containing the δZ_Φ. If all UV divergences up to n loops are to be removed, the δZ_Φ need to be of the form

$$\delta Z_\Phi = \sum_{l=1}^{n} \sum_{k=1}^{l} Z_{\Phi,lk} \frac{h^l}{\epsilon^k}, \tag{1.9}$$

where $h = \frac{g_R^2}{16\pi^2}$. One then has to require that the UV divergences up to n loops are cancelled from all amplitudes. This fixes the values for all $Z_{\Phi,lk}$ in terms of the divergent parts of the amplitudes only. This means that the RCs are independent of the finite parts of the respective Feynman integrals, which will become important later on. In principle, one can also add to the RCs finite terms of the form $Z_{\Phi,l0}h^l$. These can be chosen freely and in this section I will consider the minimal subtraction scheme (MS) where $Z_{\Phi,l0} = 0$.

The beta-function is then defined as

$$\beta(h) = \mu^2 \frac{\partial h}{\partial \mu^2} \tag{1.10}$$

and describes the running of the coupling with the mass scale. Rewriting the last formula in Eq. (1.6) in terms of h and $Z_h \equiv Z_g^2$ as

$$h_B = h\mu^{2\epsilon} Z_h, \tag{1.11}$$

applying the operator $(\mu^{2-2\epsilon}/Z_h)\frac{\partial}{\partial\mu^2}$ and using the fact that the bare coupling does not depend on μ one finds:

$$\begin{aligned} 0 &= \beta(h) + \epsilon h + \frac{h}{Z_h}\mu^2 \frac{\partial Z_h}{\partial \mu^2} \\ &= \beta(h) + \epsilon h + \frac{h}{Z_h}\left(\left(\mu^2 \frac{\partial h}{\partial \mu^2}\right)\frac{\partial Z_h}{\partial h} + \left(\mu^2 \frac{\partial \xi}{\partial \mu^2}\right)\underbrace{\frac{\partial Z_h}{\partial \xi}}_{=0}\right) \\ &= \beta(h) + \epsilon h + h\beta(h)\frac{\partial \ln(Z_h)}{\partial h} \quad . \end{aligned} \tag{1.12}$$

Solving for the beta-function and expanding in h one obtains:

$$\begin{aligned}\beta(h) &= \frac{-\epsilon h}{1 + h\frac{\partial \ln(Z_h)}{\partial h}} \\ &= -\epsilon h \sum_{i=0}^{\infty} (-1)^i \left(h \frac{\partial}{\partial h} \sum_{j=1}^{\infty} \frac{(-1)^{j+1}}{j} \left[\delta Z_h\right]^j \right)^i \\ &= -\epsilon h \sum_{i=0}^{\infty} (-1)^i \left(h \frac{\partial}{\partial h} \sum_{j=1}^{\infty} \frac{(-1)^{j+1}}{j} \left[\sum_{l=1}^{\infty} \sum_{k=1}^{l} Z_{h,lk} \frac{h^l}{\epsilon^k} \right]^j \right)^i \quad .\end{aligned} \tag{1.13}$$

Taking $\epsilon \to 0$ and demanding that the limit exists fixes all $Z_{h,lk}$ in terms of the coefficients $Z_{h,l1}$ of the $1/\epsilon$ poles and from the above expression only the terms with $i = j = k = 1$ remain, therefore

$$\beta(h) = h^2 \frac{\partial}{\partial h} \sum_{l=1}^{\infty} Z_{h,l1} h^l = \sum_{l=0}^{\infty} h^{l+2} (l+1) Z_{h,(l-1)1} \equiv - \sum_{l=0}^{\infty} \beta_l h^{l+2}. \tag{1.14}$$

The β-function of QCD played an important role in establishing the acceptance of Yang-Mills theory as a description of the strong interaction, since its first order correctly predicted asymptotic freedom [17]. Since then, 2-loop [18,19], 3-loop [20,21] and 4-loop [22,23] corrections have been calculated. As described above, this requires determining the RCs Z_Φ such that all UV divergences are cancelled. To fix the Z_Φ one needs to calculate several (but not all) of the self-energies and vertices of the theory to the desired loop level, which is where Feynman integrals come into play. Unlike the vacuum diagrams considered in this thesis, these integrals still contain external momenta. One can however use the fact that the Z_Φ only depend on the UV-divergent parts of these integrals and extract these parts in terms of vacuum diagrams via an appropriate expansion, as I will show in section 1.2.

1.2 Divergences and vacuum integrals

As the example of the QCD β-function in section 1.1 shows, it is sometimes sufficient to calculate the UV-divergent part of certain integrals. There are several ways in which the UV divergences can be extracted from the integrals in terms of rewriting them as massive vacuum integrals (see e.g. [24, 25]). I will portray in the following the one given in [7]. It relies on the fact that for mass-independent renormalisation schemes, such as MS or the modified minimal subtraction scheme $\overline{\text{MS}}$, all UV counterterms are simply polynomial in both momenta and masses [26]. I will further assume that all subdivergences have already been removed and a Euclidean metric is used. One can then expand an integrand via the exact propagator decomposition

$$\frac{1}{(k+p)^2 + M^2} = \frac{1}{k^2 + m^2} - \frac{2k \cdot p + p^2 + M^2 - m^2}{k^2 + m^2} \frac{1}{(k+p)^2 + M^2}. \tag{1.15}$$

Here k and p are linear combinations of loop and external momenta respectively. M is the particle mass for the propagator and m is an artificial mass introduced to avoid spurious infrared (IR) divergences the decomposition would otherwise cause. The same m is chosen for all propagators expanded in this way. The right hand side of Eq. (1.15) consists of one part which is independent of all external momenta as well as masses except the artificial one and another part which leads to a lower degree of divergence than the original propagator. Since the latter is again contained in the second part of the decomposition, one can repeatedly apply Eq. (1.15) until the last term does not contribute to the UV divergences one is interested in any more and can thus be dropped. All powers of external momenta in the numerators can be factored out using e.g. tensor integral projection methods (see e.g. [27]). In the end one is left with fully massive scalar integrals depending only on loop momenta and a single mass scale m, which is exactly the class of integrals considered in this thesis. Since the use of a mass-independent regularisation scheme ensures that the UV counterterms are polynomial in all masses, the artificial mass m can safely be set to zero at that point (but not before the integrals have been evaluated).

2 Conventions and notations

Most of the conventions in this thesis will be chosen such that they match those of J. Möller's dissertation [12], since he focusses on the same set of difference equations, albeit using a very different strategy in obtaining them.

2.1 Integrals and sectors

Throughout this thesis I will assume a Euclidean metric for all momentum vectors unless otherwise stated. Integrals that are expressed using the Minkowski metric will have to be Wick-rotated [28] for comparison of results. The integration measure for loop momentum integrations will be written as

$$\int_{k_1,\dots,k_n} \equiv \int \frac{\mathrm{d}^d k_1}{(2\pi)^d} \cdots \int \frac{\mathrm{d}^d k_n}{(2\pi)^d}. \tag{2.1}$$

Using these conventions, a generic scalar n-loop Feynman integral in d dimensions with loop momenta k_i and n_e external momenta p_i can be written in the form

$$F\left(p_1,\dots,p_{n_e},\{m_i\},d\right) \equiv \int_{k_1,\dots,k_n} \frac{N}{D_1^{a_1}\dots D_t^{a_t}}\ , \tag{2.2}$$

where the $D_i = d_i^2 + m_i^2$ are the denominators of the propagators raised to some powers a_i, which are typically small integers. The d_i are linear combinations of loop (and external) momenta with possible coefficients -1, 0 or 1. The numerator N can contain a product of scalar products of momenta including at least one loop momentum. Any Lorentz structure an integral might have can always be projected out to arrive at scalar integrals (see e.g. [27]). In the following, the D_i will be referred to as *propagators* rather than the more accurate *inverse propagators* to improve readability.

Analytic solutions are known only for a few of the easier Feynman integrals, such as the one-loop massive vacuum integral with an arbitrary propagator exponent z:

$$\begin{aligned} I(z) \equiv \int_k \frac{1}{(k^2+m^2)^z} &= m^{d-2z} \int \frac{\mathrm{d}^d\mathbf{k}}{(2\pi)^d} \frac{1}{(\mathbf{k}^2+1)^z} \\ &= m^{d-2z} \frac{1}{(2\pi)^d} \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \underbrace{\int_0^\infty \mathrm{d}|k| \frac{|k|^{d-1}}{(|k|^2+1)^z}}_{=\frac{\Gamma(d/2)\Gamma(z-d/2)}{2\Gamma(z)}} \\ &= m^{d-2z} \frac{\Gamma\left(z-\frac{d}{2}\right)}{2^d\pi^{\frac{d}{2}}\Gamma\left(z\right)} \end{aligned} \tag{2.3}$$

More complicated integrals often need to be evaluated numerically (see e.g. [29,30]). With higher loop orders the integrals not only grow more complex, but the number of different integrals that appear in calculations also rises significantly. It is thus essential to have a unified classification of all integrals one might encounter in a calculation. For this thesis I will consider the set of all fully massive n-loop vacuum integrals with $n \leq 5$ and a single mass scale m.

2.2 Vacuum tadpoles

In physical applications only integrals that can be drawn as a graph will appear. Since the maximum number of new lines that can be added to a graph by increasing its number of loops by one is 3, it is straightforward to see that for a single vacuum integral the maximum number $t_{max}^{(n)}$ of different propagators is given by

$$t_{max}^{(n)} = \begin{cases} 1 & , \quad n = 1, \\ 3 \cdot (n-1) & , \quad n \geq 2, \end{cases} \tag{2.4}$$

while the number $N_s^{(n)}$ of possible scalar products of loop momenta reads

$$N_s^{(n)} = \frac{n \cdot (n+1)}{2}. \tag{2.5}$$

While for $n = 1, 2, 3$ we have $t_{max}^{(n)} = N_s^{(n)}$, starting at 4 loops the number of possible scalar products exceeds the maximum number of propagators for a given integral. Since different graphs clearly need different choices of propagators to be represented, the number of propagators one needs to be able to construct integrals for all possible graphs is at least $N_s^{(n)}$. In this thesis, I will therefore use so-called auxiliary topologies A_n with $m \equiv N_s^{(n)}$ propagators D_i, out of which all possible n-loop diagrams can be constructed. The specific choices are given in Table 6.1 in section 6.1. Since I only consider vacuum integrals in this thesis, all scalar products of momenta can be expressed as linear combinations of the D_i and masses, which means the numerator N in Eq. (2.2) can be assumed to be 1. Once an auxiliary topology is chosen, any integral can be uniquely represented by its propagator exponents a_i:

$$I(a_1, \ldots, a_m) \equiv \int_{k_1,\ldots,k_n} \frac{1}{D_1^{a_1} \ldots D_m^{a_m}} \tag{2.6}$$

Since $a_i \in \mathbb{Z}$, each integral corresponds to a point in the space $\mathbb{Z}^m$ and vice versa. To further classify integrals, I will introduce some definitions:

$$\begin{aligned} t &\equiv \text{Number of positive } a_i, \\ r &\equiv \text{Sum of positive } a_i, \\ s &\equiv \text{Sum of the absolute values of negative } a_i, \\ ID &\equiv \sum_{i=1}^{m} 2^{m-i-1} \cdot \Theta(a_i), \text{ with } \Theta(x) \equiv \begin{cases} 1 & , \quad x > 0 \\ 0 & , \quad x \leq 0 \end{cases} . \end{aligned} \tag{2.7}$$

Each ID has an infinite number of integrals associated with it, the set of which is called a sector. With an auxiliary topology of m propagators one can form $\binom{m}{t}$ sectors with t different propagators, yielding

$$\sum_{t=0}^{m} \binom{m}{t} = 2^m \tag{2.8}$$

sectors in total. As can already be seen from $t_{max}^{(n)} < N_s^{(n)}$ for $n \geq 4$, not all of these sectors will appear in calculations. It is therefore useful to classify the sectors further. A sector will be called

- a physical sector, if a graph can be drawn with the respective momenta and the sector contains non-vanishing integrals,
- a trivial zero sector, if its number of propagators t is smaller than the number of loops n, since all integrals of that form vanish in dimensional regularisation,
- a non-trivial zero sector, if $t \geq n$, but all integrals of the sector are zero anyway in dimensional regularisation,
- a trivial antisector, if $t > t_{max}$, since no graph can be drawn for such a sector,
- a non-trivial antisector, if $t \leq t_{max}$, but nevertheless no graph can be drawn for the respective momenta.

Only the physical sectors have to be taken into account for the calculation of difference equations. If one is interested in integrals containing external momenta instead of vacuum integrals, the same classification as described in this section can be used, if one introduces additional (auxiliary) propagators to deal with the higher number of scalar products of momenta that may appear in such integrals (see e.g. [12]).

2.3 Sector shifts and symmetries

It is possible to reduce the number of relevant sectors even further by considering momentum transformations of the form

$$k_j \to \sum_{i=1}^{n} R_{ji} k_i \ , \quad \det(R) \neq 0. \tag{2.9}$$

The propagators of the auxiliary topology A_n will be transformed accordingly, with two possible results:

$$D_i \xrightarrow{(2.9)} \begin{cases} D_i' = D_j & \in A_n \\ D_i' = \sum_{a=1}^{m} c_a \cdot D_a + c \cdot m^2 & \notin A_n \end{cases} \tag{2.10}$$

For a sector ID where the propagators with positive exponents a_i are given by the set $P_{ID} = \{D_{i_1}, \ldots, D_{i_t}\}$, only those transformations which fulfil

$$D_{i_j}' \in A_n \quad \text{for } j = 1, \ldots, t \tag{2.11}$$

will be considered, since all other transformations lead to integrals with sums of propagators in the denominator. The former can then be subdivided into two classes: A transformation of the form given in Eq. (2.9) for a sector ID with P_{ID} as above is called

- a sector symmetry, if $D_{i_j}' \in P_{ID} \ \forall \ j$. Note that this is always a one-to-one mapping $D_{i_j} \to D_{i_{j'}}$, since $\det(R) \neq 0$ means that the transformation is reversible.
- a sector shift, if $\exists \ j$ with $D_{i_j} \to D_a \in A_n \setminus P_{ID}$. The resulting integral(s) in this case will belong to a different sector ID'.

In either case the remaining D_i in the numerator (those with negative a_i) may be shifted to a sum of elements of A_n and masses according to Eq. (2.10). To be able to keep up the notation introduced in Eq. (2.6), one then has to expand the product of all such sums and split the integral into a sum of integrals. It is possible that some of the resulting integrals have a lower number $t' < t$ of different propagators in the denominator than the original integral due to cancellations. The corresponding sectors are then subsectors of the sector ID in case of a sector symmetry or ID' in case of a sector shift. More precisely, a sector ID_{sub} is called a subsector of sector ID, if $P_{ID_{sub}} \subsetneq P_{ID}$. I will also use the term for all sectors the subsectors can be mapped to using sector shifts. In terms of the corresponding graphs, removing an element of P_{ID} can be thought of as contracting and removing the line representing that element, effectively merging the two vertices at its end into one.

In general, the number of physical sectors is greater than the number of different graphs one can draw, since different sectors can describe the same graph. I will call the complete set of sectors which describes one graph a *topology*. All sectors in a topology are connected via sector shifts, which means it is sufficient to choose one representative sector for the topology and use the sector shifts to map all other sectors to the chosen one. Doing so effectively reduces the number of sectors that need to be considered during the calculations from the set of all physical sectors to the comparatively small set of representatives. In this thesis, I will always choose as the representative sector of a topology the one with the highest ID.

It should be noted that for all sector symmetries and most sector shifts the momentum transformation matrix R fulfils $|\det(R)| = 1$. Starting at 4 loops there are also some sector shifts with $|\det(R)| \neq 1$. For the choice of the auxiliary topology A_4 which is given in Table 6.1, there is only a single example of this. The sector shift relating the sector with $ID = 537$ to sector 960 is given by

$$R = \frac{1}{2}\begin{pmatrix} 2 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{pmatrix}, \quad |\det(R)| = \frac{1}{2}. \tag{2.12}$$

In this case the transformation introduces an additional factor 2^{-d}, which is not present if $|\det R| = 1$:

$$\underbrace{I(a_1, 0, 0, 0, 0, a_6, a_7, 0, 0, a_{10})}_{ID=537} = 2^{-d}\,\underbrace{I(a_1, a_6, a_7, a_{10}, 0, 0, 0, 0, 0, 0)}_{ID=960} \tag{2.13}$$

In practice one does not have to worry about these factors for appropriately chosen auxiliary topologies, because all physical sectors which are related to the representative sectors via a sector shift with $|\det(R)| \neq 1$ turn out to appear only as subsectors of antisectors, which are not present in the calculations to begin with.

2.4 Example: A_2

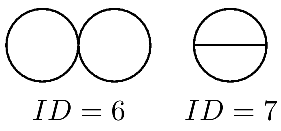

Figure 2.1: 2-loop graphs and representative sectors.

The conventions and notations are best illustrated by looking at the massive vacuum 2-loop integrals, where we have $t_{max}^{(2)} = N_s^{(2)} = 3$. There are two independent graphs, the trivial one with two disconnected 1-loop parts and the so-called sunset topology, which are given in Fig. 2.1. The auxiliary topology will be chosen as

$$A_2 = \left\{ k_1^2 + m^2, k_2^2 + m^2, (k_1 - k_2)^2 + m^2 \right\} , \tag{2.14}$$

so that any integral can be written in the form

$$I(a_1, a_2, a_3) \equiv \int_{k_1,k_2} \frac{1}{(k_1^2 + m^2)^{a_1} (k_2^2 + m^2)^{a_2} \left((k_1 - k_2)^2 + m^2 \right)^{a_3}} . \tag{2.15}$$

The auxiliary topology thus has $2^3 = 8$ sectors in total. Of these, the sectors with IDs 0, 1, 2 and 4 are trivial zero sectors, since they have at most one propagator in the denominator. Sectors 3 and 5 with $t = 2$ can be shifted to sector 6 via

$$\begin{aligned} & k_1 \to k_1 + k_2 \quad \Rightarrow \quad D_3 \to D_1 \\ \text{and} \quad & k_2 \to k_1 + k_2 \quad \Rightarrow \quad D_3 \to D_2 \end{aligned}$$

respectively. The only remaining sectors for which difference equations have to be calculated are thus sectors 6 and 7. A complete set of the relations between the sectors is given in Fig. 2.2.

As can be seen from Fig. 2.1, both sector 6 and 7 are fully symmetric under any permutation of their propagator exponents. The explicit symmetry transformations including the identity transformation are listed in Tables 2.1 and 2.2.

Sector 6				
R	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$
$D_1 \to$	D_1	D_1	D_2	D_2
$D_2 \to$	D_2	D_2	D_1	D_1
$D_3 \to$	D_3	$2D_1 + 2D_2 - D_3 - 3m^2$	D_3	$2D_1 + 2D_2 - D_3 - 3m^2$

Table 2.1: The 4 sector symmetry transformations of sector 6.

Sector 7						
R	$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$	$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 1\\-1 & 1\end{pmatrix}$	$\begin{pmatrix}1 & -1\\1 & 0\end{pmatrix}$	$\begin{pmatrix}1 & 0\\1 & -1\end{pmatrix}$	$\begin{pmatrix}1 & -1\\0 & -1\end{pmatrix}$
$D_1 \to$	D_1	D_2	D_2	D_3	D_1	D_3
$D_2 \to$	D_2	D_1	D_3	D_1	D_3	D_2
$D_3 \to$	D_3	D_3	D_1	D_2	D_2	D_1

Table 2.2: The 6 sector symmetry transformations of sector 7.

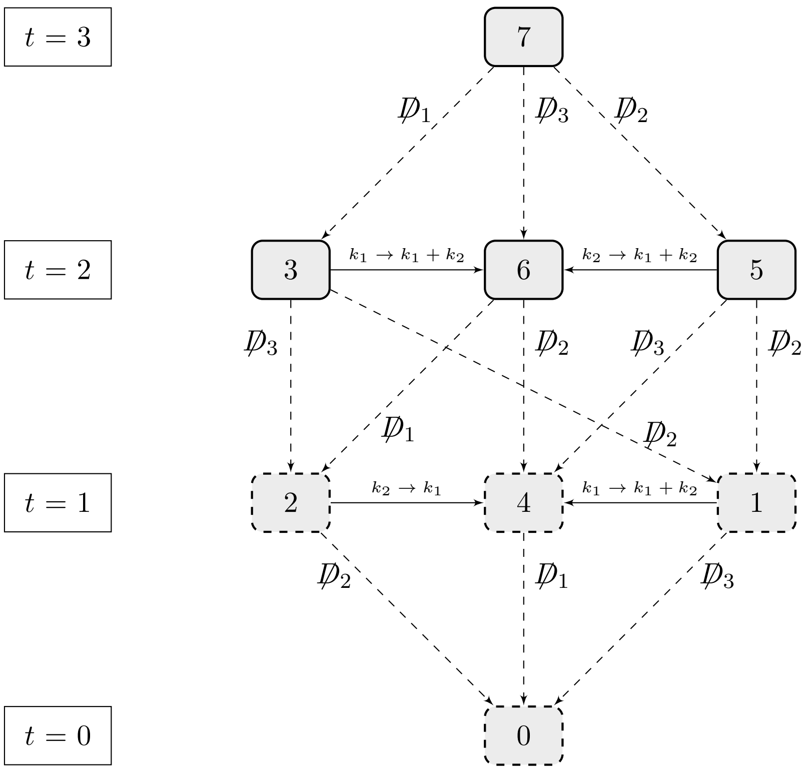

Figure 2.2: Overview of 2-loop vacuum integral sectors. Solid boxes represent physical sectors, dashed boxes zero sectors. Solid arrows signify sector shifts via the specified momentum transformations and dashed arrows point to subsectors arrived at by cancelling the specified propagator.

3 Reduction and difference equations

Even after applying the sector shifts and symmetries introduced in section 2.3, the number of integrals that can appear in calculations is still very large. This is due to the fact that each sector still contains an infinite number of possible r and s values and different distributions of the respective positive and negative powers. It is therefore highly desirable to find additional relations between the remaining integrals, thus reducing the number of integrals that in the end need to be explicitly evaluated either analytically or numerically. As it turns out, the integrals in a sector can be reduced to a basis of very few so-called master integrals plus the master integrals of the corresponding subsectors. For standard Feynman integrals this basis has been proven to be finite in general [31]. The master integrals are typically chosen such that $s = 0$ (meaning no propagators in the numerator) and r is minimal.

In practice usually only a finite subset of integrals, where the propagator exponents a_i are small integers, is reduced to the master integrals, since one is only interested in integrals appearing in actual calculations. A complete reduction of a sector would require the a_i to be kept as symbols throughout the whole reduction rather than substituting integer values, which complicates matters greatly. Examples of complete reductions include results at the 2-loop level [32] as well as the 3-loop level [33], but so far there is no automatable approach to this which is both fast and generic enough for a broad range of calculations.

While a completely symbolic reduction is quite difficult, keeping only a single exponent as a symbol is much easier in comparison and can yield very useful results, so-called difference equations for the master integrals. A difference equation for an integral $I(\{a_i\})$ in a propagator position x, where $a_x > 0$, is defined as a linear equation of the form

$$\sum_{j=j_0}^{R+j_0} c_j\left(d, z, \{m_i\}, \{p_i \cdot p_k\}\right) I\left(a_1, \ldots, a_{x-1}, z+j, a_{x+1}, \ldots, a_m\right) = S\left(d, z, \{m_i\}, \{p_i \cdot p_k\}\right). \tag{3.1}$$

Here R is the order of the difference equation and the c_j are rational functions of the space-time dimension d, the symbolic propagator exponent z, the mass scales appearing in the problem and scalar products of external momenta. On the right hand side S is a linear combination of integrals similar to those on the left side, but from subsectors of $I(\{a_i\})$, with coefficients of the same kind as the c_j. The offset j_0 can be absorbed by shifting z and will therefore be assumed to be 0 from here on.

One example of a difference equation is easy to obtain by recalling the analytic result for the massive one-loop vacuum integral in Eq. (2.3):

$$I(z) = m^{d-2z} \frac{\Gamma\left(z - \frac{d}{2}\right)}{2^d \pi^{\frac{d}{2}} \Gamma(z)}. \tag{3.2}$$

From this one can see that

$$I(z+1) = \frac{2z-d}{2zm^2} I(z), \tag{3.3}$$

which can be written in the form of Eq. (3.1) and thus is a difference equation of order 1. In this simple case it also provides a prescription for a complete reduction of the sector, since it can be used to write $I(z)$ in terms of the master integral $I(1)$ for all integer values $z > 1$ [1]. In a more general setup, a difference equation could potentially be used to reduce integrals by lowering a single exponent at position x to at most R, but only for a fixed set of remaining exponents a_i. This has very limited value for the reduction of a sector, since it usually only covers a small part of the integrals one is interested in. It does however provide a means to improve the numeric calculation of master integrals and can in some cases even yield analytic results for the integrals [34].

The basis of master integrals is typically chosen such that $s = 0$ and r is minimal, meaning the non-zero exponents a_i have small integer values. These integrals can be difficult to evaluate numerically, since the integrand converges slowly for large loop momenta. This behaviour changes to a much more rapid convergence when one of the exponents is large, which can be exploited using factorial series. The approach is described in detail in [34], but the general idea is to calculate integrals with one large exponent numerically to the required precision and then map the results down to the actual master integrals using difference equations.

For a master integral with t different propagators one could in principle generate t difference equations by choosing all possible relevant positions x, each of which could be used to evaluate the integral numerically via factorial series. In case the topology in question exhibits sector symmetries, the number of independent difference equations is reduced, since propagator positions, whose exponents can be interchanged via symmetries, can be thought of as equivalent in regard to difference equations. While a single difference equation is sufficient for numerical evaluation of one integral in sector ID, the difference equations for all possible (non-equivalent) positions x might still be needed, since the respective integrals with z at that position might appear in the right hand side S of Eq. (3.1) for sectors ID' which have ID as a subsector. Additionally, difference equations for different positions x may have different orders R. This is a relevant factor, since the difficulty of the numerical evaluation rises significantly with the order of the difference equation [12].

It should be noted that difference equations for master integrals in general are not constrained to the form in Eq. (3.1). R.N. Lee proposed [36] to use the space-time dimension d rather than a propagator exponent z as the variable of a difference equation. To generate these equations he makes use of the dimensional recurrence relations suggested by O.V. Tarasov [35] (see section 3.1). In this thesis, I will however focus on the difference equations defined in Eq. (3.1).

[1] In fact, if one solves Eq. (3.3) for $I(z)$, one also finds that $I(z) = 0$ for all integers $z \leq 0$.

3.1 Generating equations

There are several ways beyond the sector shifts and symmetries in which additional linear equations between integrals can be generated:

- *Integration By Parts* (IBP) relations [37]
  The most common approach to generate the necessary equations for the reduction employs equations of the form
  $$\begin{aligned} 0 = \int_{k_1,\ldots,k_n} \frac{\partial}{\partial k_i^\mu} \frac{q_j^\mu}{D_1^{a_1}\ldots D_m^{a_m}}\ , \quad i = 1,\ldots,n\ , \quad q_j \in M, \\ M = \{k_1,\ldots,k_n,p_1,\ldots,p_{n_e}\}\ . \end{aligned} \tag{3.4}$$
  For the one-loop massive vacuum integral in Eq. (2.3) this yields
  $$\begin{aligned} 0 &= \int_k \frac{\partial}{\partial k^\mu}\frac{k^\mu}{(k^2+m^2)^z} = dI(z) - z\int_k \frac{k^2}{(k^2+m^2)^{z+1}} \\ &= dI(z) - 2zI(z) + 2zI(z+1) \\ \Rightarrow I(z+1) &= \frac{2z-d}{2zm^2}I(z), \end{aligned} \tag{3.5}$$
  which is in full agreement with the result obtained earlier (Eq. (3.3)).
- *Lorentz-invariance* (LI) identities [38]:
  Since the integral in Eq. (2.2) is a Lorentz scalar, it is invariant under an infinitesimal Lorentz transformation of the external momenta
  $$p^\mu \to p^\mu + \delta p^\mu = p^\mu + \delta\epsilon^\mu{}_\nu p^\nu, \quad \epsilon^\mu{}_\nu = -\epsilon^\nu{}_\mu. \tag{3.6}$$
  One therefore obtains
  $$\begin{aligned} F(p_1,\ldots,p_{n_e}) &= F(p_1+\delta p_1,\ldots,p_{n_e}+\delta p_{n_e}) \\ &= F(p_1,\ldots,p_{n_e}) + \left(\sum_{i=1}^{n_e}\delta p_i^\mu \frac{\partial}{\partial p_i^\mu}\right) F(p_1,\ldots,p_{n_e}) \end{aligned} \tag{3.7}$$
  and subsequently
  $$\epsilon^\mu{}_\nu \left(\sum_{i=1}^{n_e} p_i^\nu \frac{\partial}{\partial p_i^\mu}\right) F(p_1,\ldots,p_{n_e}) = 0. \tag{3.8}$$
  Since this is valid for all $\epsilon^\mu{}_\nu = -\epsilon^\nu{}_\mu$, one can drop the $\epsilon^\mu{}_\nu$ for an antisymmetrisation, yielding
  $$\left(\sum_{i=1}^{n_e}\left(p_i^\nu\frac{\partial}{\partial p_i^\mu} - p_i^\mu\frac{\partial}{\partial p_i^\nu}\right)\right) F(p_1,\ldots,p_{n_e}) = 0. \tag{3.9}$$
  Upon contraction with $(p_a^\mu p_b^\nu - p_a^\nu p_b^\mu)$ one obtains
  $$p_a^\mu p_b^\nu \left(\sum_{i=1}^{n_e} p_i^\nu \frac{\partial}{\partial p_i^\mu}\right) F(p_1,\ldots,p_{n_e}) \equiv O_{ab}^{\mathrm{LI}} F(p_1,\ldots,p_{n_e}) = 0. \tag{3.10}$$

If taken to act directly on the integrand, the operators O_{ab}^{LI} can yield non-trivial relations between integrals. In this thesis LI identities cannot be used, since I only consider integrals with no external momenta. Furthermore, it can be shown [39] that all LI identities can be expressed as linear combinations of IBP identities.

- *Mass derivatives:*
  If one rewrites all masses that appear in an integral in terms of a single mass scale such that $D_i = d_i^2 + x_i m^2$ for all massive propagators, the integral can be written as m raised to the mass dimension of the integral times a numerical factor:

$$I(a_1,\ldots,a_m) = m^{nd-2\sum\limits_i a_i} X. \tag{3.11}$$

  Applying $\frac{\partial}{\partial m^2}$ to both forms will result in the non-trivial equation

$$-\sum_i a_i x_i I(a_1,\ldots,a_i+1,\ldots,a_m) = \left(\frac{nd}{2} - \sum_i a_i\right)\frac{1}{m^2} I(a_1,\ldots,a_m). \tag{3.12}$$

  This is easily demonstrated for the one-loop massive vacuum case:

$$\int_k \frac{1}{(k^2+m^2)^z} \equiv I(z) \equiv m^{d-2z} X \tag{3.13}$$

  Since X is independent of m one obtains

$$\overset{\frac{\partial}{\partial m^2}}{\Longrightarrow} \quad -zI(z+1) = \left(\frac{d}{2}-z\right) m^{d-2z-2} X = \left(\frac{d}{2}-z\right)\frac{1}{m^2} I(z) \tag{3.14}$$

  and thus the same result as in Eq. (3.3):

$$I(z+1) = \frac{2z-d}{2zm^2} I(z). \tag{3.15}$$

  I expect that the relations obtained by these mass derivatives are already contained in the IBPs, but could not find a general proof of this in the literature.

- *Space-time dimensional relations*
  These relations connect integrals with different values of the space-time dimension d. They will find no application in this thesis and I will therefore only provide a short sketch of the idea following the one given in [12]. For an in-depth discussion I refer to the original paper [35].
  Using the Schwinger parametrisation

$$\frac{1}{(D+i\epsilon)^a} = \frac{i^{-a}}{\Gamma(a)} \int_0^\infty \mathrm{d}\alpha\ \alpha^{a-1} e^{i\alpha(D+i\epsilon)} \tag{3.16}$$

  on all propagators of a d-dimensional integral $G^{(d)}$ turns the loop integrations into Gaussian integrals which can be solved to yield

$$G^{(d)}\left(\{s_i\},\{m_i\}\right) = i^n \left(\frac{\pi}{i}\right)^{\frac{nd}{2}} \prod_{j=1}^{N_d} \frac{i^{-a_j}}{\Gamma(a_j)} \int_0^\infty \frac{\mathrm{d}\alpha_j\ \alpha_j^{a_j-1}}{(\mathcal{U}(\alpha))^{d/2}} e^{i\left(\frac{\mathcal{F}(\{s_i\},\alpha)}{\mathcal{U}(\alpha)} + \sum\limits_{l=1}^{N_d} \alpha_l (m_l^2+i\epsilon)\right)}. \tag{3.17}$$

Here N_d is the number of internal lines, $\{s_i\}$ a set of scalar invariants formed from external momenta and $\mathcal{U}$ and $\mathcal{F}$ are the so-called graph polynomials (see e.g. [40]) of a diagram corresponding to G. Next, one constructs the differential operator $\mathcal{U}(\partial)$ by replacing $\alpha_j \to \frac{\partial}{\partial m_j^2}$ in $\mathcal{U}(\alpha)$ with the additional assumption that all masses m_j are different. The actual physical masses can be inserted into calculations once all mass derivatives have been executed. Acting with $\mathcal{U}(\partial)$ on Eq. (3.17) and using

$$\mathcal{U}(\partial)e^{i\sum \alpha_l m_l^2} = \mathcal{U}(\alpha)i^n e^{i\sum \alpha_l m_l^2} \tag{3.18}$$

one finds

$$G^{(d-2)}\left(\{s_i\},\{m_i\}\right) = \frac{1}{\pi^n}\mathcal{U}(\partial)G^{(d)}\left(\{s_i\},\{m_i\}\right). \tag{3.19}$$

This method can also be extended to tensor integrals [12]. So far there are no cases known where space-time dimensional relations yield any additional information over the IBP relations for d-dimensional integrals, but they can be used to reduce the number of indices needed for an integral reduction and thus e.g. simplify symmetrisation [12].

This thesis will focus entirely on IBP relations (IBPs) with the addition of sector shifts and symmetries. The latter are actually contained in the IBPs [39], but are useful to impose on the system from the beginning anyway, since they can dramatically decrease the number of integrals that need to be considered and hence calculation time.

The integrals $I(a_1, \ldots, a_m)$ to which equations (3.4) are applied will be called *seed integrals.* For each seed integral $n(n+n_e)$ equations can be generated. These will generally contain integrals with so-called raised propagators with an exponent $a_i + 1$ due to the derivative hitting a propagator, as well as lowered propagators with exponent $a_i - 1$, or both. To relate a given integral to simpler[2] integrals, it is therefore often necessary to apply the equations (3.4) to multiple seed integrals and solve the linear system of equations. In practice, all possible integrals of a sector with predefined maximum values of r and s are used as seed integrals to reduce as many integrals as possible in the resulting system of equations. There are several programs which automate this process, such as AIR [2], FIRE [3] or Reduze (2) [4]. Automation becomes necessary, because for integrals more complicated than the one-loop example above the number and complexity of equations needed for a reduction rises significantly. In a sector with t propagators defined in an auxiliary topology with m propagators, the number of integrals with set values (r, s) is given by

$$N(m,t,r,s) = \underbrace{\binom{r-1}{r-t}}_{\substack{\#\text{ distributions of}\\ \text{positive powers}}} \cdot \underbrace{\binom{m-t+s-1}{s}}_{\substack{\#\text{ distributions of}\\ \text{negative powers}}}, \tag{3.20}$$

with $r \geq t > 0$, $s \equiv 0$ for $t = m$ and $\binom{-1}{0} := 1$. With up to $n(n+n_e)$ equations per integral, r and s thus have to be restricted to small values, since these combinatorics cause the size of the system of equations to quickly exceed the limits of either software or hardware.

[2]A definition of the difficulty of an integral will be given in section 3.2. For now it is sufficient to know that for any two non-identical integrals, one is considered simpler than the other.

There are however several ways to decrease the size of the system of equations without discarding any information. The sector symmetries discussed in section 2.3 can be used to effectively reduce $N(m,t,r,s)$ by mapping many of the integrals to linear combinations of others in the same sector or subsectors. Furthermore, the number of distinct integrals in the system of equations grows approximately with the volume V of $\mathbb{Z}^m$-space the seed integrals occupy, while the number of equations grows as $n(n+n_e)V$. For large V this means that the system of equations is vastly overdetermined. To reduce the number of redundant equations per seed integral, closer inspection of the IBP operators

$$O_{ij} = \frac{\partial}{\partial k_i^\mu} q_j^\mu \tag{3.21}$$

is needed. They turn out to form an algebra with the commutation relation

$$[O_{ik}, O_{jl}] = d\left(\delta_{il} O_{jk} - \delta_{jk} O_{il}\right). \tag{3.22}$$

R.N. Lee therefore proposes [39] to only use the subset

$$\begin{aligned}
&\frac{\partial}{\partial k_i} \cdot k_{i+1}, \quad i = 1, \dots, n, \quad k_{n+1} \equiv k_1, \\
&\frac{\partial}{\partial k_1} \cdot p_j, \quad\;\; j = 1, \dots, n_e \\
&\sum_{i=1}^{n} \frac{\partial}{\partial k_i} \cdot k_i,
\end{aligned} \tag{3.23}$$

which forms a multiplicative basis of the algebra, meaning all other operators can be written as commutators of the basis elements. In a complete (symbolic) reduction of a sector using only the above set of operators would reduce the number of equations per seed integral to $n + n_e + 1$. While still leading to an overdetermined system of equations, for $n \geq 2$ this would be a noticeable improvement over $n(n + n_e)$ equations per seed integral. When inserting integer values for the a_i, the benefit turns out to be much smaller. If an operator O_{ij} can be expressed by a single commutator of two of the basis elements in Eq. (3.23), to ensure that the information gained by acting with O_{ij} upon a seed integral $I(\{a_i\})$ is redundant would require acting both basis elements not only on $I(\{a_i\})$, but also on all possible integrals $I(\{a_i'\})$ which may be contained in the IBPs resulting from acting either basis element on $I(\{a_i\})$. Since an IBP can contain raised exponents a_i, lowered exponents or both, this includes integrals within the range $(r' \in \{r, r \pm 1\}, s' \in \{s, s \pm 1\})$, where r and s are the sum of positive and negative exponents of $I(\{a_i\})$ respectively. The values $r - 1$ and $s - 1$ would likely not pose a problem, since the corresponding seed integrals would be considered simpler than $I(\{a_i\})$ (see section 3.2) and therefore already be included in the system of equations. However, if r or s happen to already be the respective maximum r_{max} or s_{max} considered for the reduction, either the integrals with $r + 1$ and/or $s + 1$ would have to be added as seed integrals, which is undesirable due to the combinatorics in Eq. (3.20), or the operator O_{ij} has to be explicitly included for the seed integrals at the (r, s) boundaries. This greatly diminishes the advantage of needing less equations, since it is exactly the maximum values of (r, s) that provide the most seed integrals.

For greater numbers of loops n the advantages are reduced even further. Inspecting the set of operators in Eq. (3.23) closely, one finds that the number of recursive commutators needed to construct a specific operator O_{ij} is given by

$$\begin{aligned} \frac{\partial}{\partial k_i} \cdot k_j &: \quad (n+j-i-1) \bmod n \\ \frac{\partial}{\partial k_i} \cdot p_j &: \quad (n-i+1) \bmod n. \end{aligned} \tag{3.24}$$

This means that for some IBP operators the representations in the multiplicative basis contain products of up to n basis elements. As a consequence, these operators can only be safely cast aside for seed integrals with (r, s)-values up to $(r_{max} - n + 1, s_{max} - n + 1)$, which for large values n is usually a negligible portion of the whole set of seed integrals.

3.2 Solving the system of equations

Once a sufficient number of linear equations has been generated using the methods described in section 3.1, the resulting system of equations has to be brought into the desired form. For a general reduction substituting all exponents with integers this means expressing as many integrals as possible via the master integrals. If a basis of master integrals is not yet known, a minimal set of integrals with which to express all others should emerge in the reduction to take on this role. When searching for a difference equation for a master integral in sector ID, one aims to find a non-trivial equation of the form in Eq. (3.1), which means eliminating all integrals in sector ID which differ in any exponent other than the symbolic one from the master integral as well as all integrals from subsectors which differ in non-symbolic exponents from their respective master integrals.

In either case, a common approach is the Laporta algorithm [34], which is essentially a Gauss algorithm, but subject to additional considerations. Since one is interested in finding the "simplest" master integral basis, a unique, deterministic ordering prescription for the difficulty of integrals is needed. Experience shows that the runtime for the system of equations is very sensitive to this choice of ordering. One possible choice could be, in decreasing order of priority:

Out of a set of integrals, the integral which is considered to be the simplest is the one with

1. the least amount t of different propagators in the denominator,
2. the least sum $r + s$ of absolute values of propagator exponents,
3. the lowest power a_1. If there is more than one integral with the same lowest a_1, pick the one with the lowest a_2, failing that go on to a_3, a_4, ...

It is of course possible to give far more refined ordering prescriptions including criteria like the maximum or minimum exponent, as long as the ordering stays unique. The effectiveness of a prescription is difficult to predict and may vary with the set of integrals considered. To solve the system of equations for difference equations, where the of sum exponents still contains a symbol, I will use a different ordering prescription than the above, which will be given in section 5.2.

Once an ordering of integrals is established, the equations in the system are considered successively. For each new equation, all integrals which have already been reduced to simpler integrals are replaced with the respective expressions. Since, as seen in section 3.1, the system of equations is usually vastly overdetermined, many equations will turn out to be trivial at this point. If this is not the case, the equation is solved for the most difficult integral remaining and can then be used to cancel this integral from further equations as well as those already previously considered. The order in which equations are added to the system in this way is a critical factor for determining the runtime of the algorithm. A good approach is to start from the simplest equations (where difficulty is determined by the most difficult integral according to the ordering prescription) and work towards the more difficult ones, since this means that for the latter many integrals can already be reduced to only a few simpler ones, thus reducing the complexity of equations. In case the equations are not known before starting this algorithm, but instead generated (via IBPs) as needed, a good indicator for the difficulty of an equation is the difficulty of the seed integral.

For a general reduction to master integrals, one usually uses all equations up to chosen values r_{max} and s_{max} for the seed integrals. When trying to find a difference equation, it can happen that one of order R will appear fairly early in the algorithm, in which case the remaining equations do not necessarily need to be considered. It is however possible that one would find a difference equation of lower order $R' < R$ later on in the algorithm, which would simplify the numeric evaluation via factorial series. Additionally, it may be useful to continue to reduce more integrals of the sector, in case they appear in calculations for higher sectors.

4 Syzygy algorithm

The goal of a reduction as described in section 3 is to express integrals in terms of simpler integrals, where simpler usually means lower values for the sum of positive integers r. Besides leading to overdetermined systems of equations, the plain IBP approach described in section 3.1 poses another problem in this regard. IBP relations often contain integrals with one exponent a_i raised by one with respect to the seed integral, which are caused by the derivative acting on a propagator. These integrals are then typically rated as more difficult than the seed integral by the ordering prescription. If one is not interested in reducing these integrals as well, they then have to first be eliminated from the system of equations to get to the results one is looking for, thus causing additional calculations.

In 2010, Gluza, Kajda and Kosower (GKK in the following) proposed an approach which prohibits the occurrence of integrals with raised propagator exponents in the IBPs altogether [1]. This is very appealing in the context of difference equations, since Eq. (3.1) does not permit raised exponents (other than the symbolic one) with respect to the integral $I(\{a_i\})$. GKK also give further reasons to avoid raised exponents in a more general context, such as the possible introduction of worse infrared divergences by the corresponding integrals [1]. They enforce their objective that each raised propagator is immediately cancelled by requiring

$$\sum_{i=1}^{n} \sum_{j=1}^{n+n_e} \alpha_{i,j} \frac{\partial D_c}{\partial k_i^\mu} q_j^\mu \propto D_c \ \forall \ c \in C \ , \tag{4.1}$$

thus constraining the coefficients $\alpha_{i,j}$. Furthermore, they allow these coefficients to have a mass dimension and depend on scalar products of momenta as well as all masses of the integral with the following constraints: The $\alpha_{i,j}$ must be

1. polynomials in all scalar products which contain at least one loop momentum,
2. rational functions in all scalar products of external momenta and masses.

C is the set of indices of exponents one does not wish to be raised, e.g. the positive exponents excluding the symbolic one for difference equations. Raising negative exponents generally leads to simpler integrals and thus does not need to be prohibited. By requiring Eq. (4.1), the effort of eliminating the integrals with raised propagators is shifted to finding appropriate sets of coefficients $\alpha_{i,j}$ by generalising the IBP operators to linear combinations of derivatives with possibly dimensionful coefficients. The modified IBP relations, which I will refer to in the following as GKK relations, then read

$$0 = \int_{k_1,\ldots,k_n} \sum_{i=1}^{n} \sum_{j=1}^{n+n_e} \frac{\partial}{\partial k_i^\mu} \frac{\alpha_{i,j} q_j^\mu}{D_1^{a_1} \ldots D_m^{a_m}} \quad . \tag{4.2}$$

Finding a complete set of all possible sets $\{\alpha_{i,j}\}$ turns out to be non-trivial. It is obvious from Eq. (4.1) that there is an infinite number of possible sets of coefficients $\alpha_{i,j}$, since every solution can simply be multiplied with an arbitrary scalar product and still remain a solution. Because all scalar products can be expressed via the propagators D_i, Eq. (4.2) shows that this multiplication is equivalent to including seed integrals with lower propagator exponents. An algorithm to find viable coefficients $\alpha_{i,j}$ should therefore avoid such factorisable solutions, since they carry no additional information. It is then sufficient to obtain a *basis* for the sets $\alpha_{i,j}$. Here, a basis is defined as a minimal set $B = \{\alpha^{(k)}\}$, where $\alpha^{(k)} = (\alpha^{(k)}_{0,0}, \ldots, \alpha^{(k)}_{n,n+n_e})$, with the property that each set $\alpha = (\alpha_{0,0}, \ldots, \alpha_{n,n+n_e})$ allowed by Eq. (4.1) can be written as a linear combination of the elements of the basis, where the coefficients of those elements have the same constraints as the $\alpha_{i,j}$. Unlike the complete set of all possible α, a basis defined in this way is finite [1]. In section 4.1, I will present a possible algorithm to find such a basis, but first I want to address two important questions that arise at this point:

(I) *Can this approach be more efficient than generating equations via pure IBPs?*

To answer this question definitely, one needs to look at algorithms for the generation of the $\alpha_{i,j}$ and their implementation. I will do so in sections 4.1 and 5 respectively. There are however some general points to be made. The coefficients $\alpha_{i,j}$ are not dependent on the actual exponents of the seed integral, only the set C of indices whose exponents must not be raised. They can therefore be used to generate equations without raised exponents for all seed integrals which do not differ in the propagator positions in C. For unaltered IBPs on the other hand, different seed integrals would lead to different integrals with raised exponents, which would all have to be eliminated independently. This advantage of the GKK approach is independent of the algorithm for finding the $\alpha_{i,j}$ and grows with the number of seed integrals required for the problem. Furthermore, expressions in the calculation of the $\alpha_{i,j}$ are easier to deal with than coefficients in the Laporta algorithm, since the former depend neither on the space-time dimension d nor the symbolic exponent z. As a possible drawback, it should be noted that, due to the GKK operators being linear combinations of derivatives, the number of terms in the initial equations (4.2) can be rather large compared to the plain IBP relations. To minimise this effect, it is important to choose the basis elements $\alpha^{(k)}$ with as few entries as possible.

(II) *Does one lose information compared to the IBP approach when generating equations in this way?*

I will first consider a scenario where the following two assumptions hold true:

1. All seed integrals differ only in the non-positive exponents.
2. No further knowledge of integrals (such as symmetries) beyond acting with the IBP/GKK operators on the seed integrals is added to the system.

If, in this scenario, one considers a pure IBP approach with the unaltered operators O_{ij} acting on the set of seed integrals, the system of equations can be solved and divided into two parts: The first part consists of equations that have been solved for an integral with one exponent raised, and the second part consists of the equations from which all integrals with raised exponents have been eliminated by linear combinations. The information in the former part is obviously not contained in the GKK approach by design, because one is not interested in it. The latter part in this scenario however can be reproduced completely, if one finds a basis of all possible sets $\alpha_{i,j}$ and acts with them on the same set of seed integrals. This can be seen as follows:

The only way to cancel an integral with a raised propagator at a position $y \in C$ from the IBP approach in this scenario is by linear combinations of the equations. The coefficients in these linear combinations are at most rational functions of scalar products of external momenta and masses, but do not depend on the space-time dimension d or the symbolic exponent z. This is due to the fact that the dimension d only appears in terms where the derivative hits q_j^μ (hence no raised propagator) and z can only be found in coefficients where the symbolic exponent at position x may be raised, but $x \notin C$. The coefficients in the linear combinations needed to cancel the integrals with raised propagators at positions $y \in C$ are therefore allowed as expressions for the $\alpha_{i,j}$. One can then conclude that a complete basis of the sets $\{\alpha_{i,j}\}$ reproduces all of these linear combinations and thus no information is lost in this scenario except the information which is inseparably linked to integrals with raised exponents. Furthermore, since the $\alpha_{i,j}$ may also contain scalar products of loop momenta which can be written as a sum of propagators D_i, some of the information in the GKK equations (4.2) would require the derivatives in the pure IBP approach acting on additional seed integrals with lower exponents. For the same set of seed integrals the GKK approach can therefore even contain additional information over the system of equations generated by plain IBPs.

However, in practice the two above assumptions usually do not hold. In the IBP approach the set of seed integrals typically contains integrals with different values for the sum of positive exponents r. For the GKK operators this makes no sense, since it would introduce the raised exponents one is trying to avoid already in the seed integrals. If one can find an equation in the IBP approach which contains no raised propagators compared to an integral I one is interested in, but can only be generated if seed integrals with raised propagators compared to I are considered, the information that equation holds may be lost to the GKK approach. It can be argued that not a large fraction of the overall obtainable information is lost in this way, since most of the information can already be generated from seed integrals without raised propagators. Additionally, since the equations obtained from the IBP operators are vastly overdetermined (see section 3.1) and this also translates to the GKK operators, the loss of some equations does not automatically imply that all of their information is lost to the complete system of equations.

The second assumption, that one puts no further knowledge than the IBPs into the Laporta algorithm, does not hold in general either, since the information one has about symmetries and which sectors are zero sectors is typically added manually to the system of equations. This presents us with two additional ways other than linear combinations in which propagators with raised exponents can vanish from an equation. The corresponding integrals may either vanish exactly in dimensional regularisation, or be transformed via a sector symmetry into an integral with a different raised propagator, which might then cancel another term with the same integral. Neither of these possibilities is covered by the GKK approach as described in [1], since Eq. (4.1) prohibits any raised propagators from the beginning and information about symmetries and zero sectors is only added to the system after equations have been generated. Even though the sector symmetries are always contained in the complete set of IBPs [39], there is no guarantee that they will turn up in the system for a finite set of seed integrals. It should therefore be possible to lose information when replacing IBP relations with the unmodified GKK ones for the same seed integrals. In section 4.3, I will therefore propose a way of fixing this problem (to some extent) by relaxing the constraints of Eq. (4.1) and introducing symmetry and zero sector information already at that point.

4.1 Syzygies via linear algebra

To find the coefficients $\alpha_{i,j}$, it is convenient to rewrite Eq. (4.1) as

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{n+n_e} \alpha_{i,j} \frac{\partial D_c}{\partial k_i^\mu} q_j^\mu + \beta_c D_c \quad \forall\, c \in C\,, \tag{4.3}$$

which can then be expressed as a matrix equation

$$\alpha \cdot E = 0. \tag{4.4}$$

Here α and E are given by

$$\begin{aligned} \alpha &= (\alpha_{1,1}, \ldots, \alpha_{1,n+n_e}, \ldots, \alpha_{n,1}, \ldots, \alpha_{n,n+n_e}, \beta_{c_1}, \ldots, \beta_{c_\gamma})\,, \\ E &= \begin{pmatrix} \frac{\partial D_{c_1}}{\partial k_1^\mu} q_1^\mu & \cdots & \frac{\partial D_{c_\gamma}}{\partial k_1^\mu} q_1^\mu \\ \vdots & \ddots & \vdots \\ \frac{\partial D_{c_1}}{\partial k_1^\mu} q_{n+n_e}^\mu & \cdots & \frac{\partial D_{c_\gamma}}{\partial k_1^\mu} q_{n+n_e}^\mu \\ \vdots & & \vdots \\ \frac{\partial D_{c_1}}{\partial k_n^\mu} q_1^\mu & \cdots & \frac{\partial D_{c_\gamma}}{\partial k_n^\mu} q_1^\mu \\ \vdots & \ddots & \vdots \\ \frac{\partial D_{c_1}}{\partial k_n^\mu} q_{n+n_e}^\mu & \cdots & \frac{\partial D_{c_\gamma}}{\partial k_n^\mu} q_{n+n_e}^\mu \\ D_{c_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & D_{c_\gamma} \end{pmatrix}, \end{aligned} \tag{4.5}$$

where $\gamma \equiv |C|$ and the number of rows in E will be called $\rho = n(n + n_e) + \gamma$. The entries of E are all polynomials of scalar products of momenta and squared masses and thus have homogeneous mass dimension 2. This means that the solutions α, which in mathematics are also known as *syzygies*, also have homogeneous mass dimension. GKK already suggested several algorithms for finding a complete basis of these syzygies in their paper [1]. All of them require the use of Gröbner bases, which are usually difficult to compute [41]. A different algorithm was therefore proposed by Schabinger [6], who is building on the work of Cabarcas and Ding [42]. The algorithm is based entirely on basic linear algebra and thus completely avoids Gröbner bases. I have implemented a modified version of this algorithm in a C++ program, which I will discuss in section 5. In this section, I will outline the algorithm proposed by Schabinger, albeit with slight alterations, and in section 4.3 suggest modifications that will solve most of the problem of possible loss of information in the GKK approach due to the missing information about symmetries and zero sectors.

Schabinger starts by choosing a basis

$$X = \{x_1, \ldots, x_\xi\} \tag{4.6}$$

which contains all scalar products that include at least one loop momentum. Additionally, one mass scale m^2 is added to the basis. All other kinematic invariants w_i of mass dimension 2 can then be expressed as the dimensionless ratios $\chi_i = w_i/m^2$ times m^2. In the following, I will call the degree Δ of an expression its mass dimension divided by 2, which is simply the sum of powers of x_i it contains.

The basic idea of the algorithm is then to find solutions α incrementally in their degree Δ, starting at either $\Delta = 0$ or $\Delta = 1$ and working up to a predefined Δ_{max}. The elements of α are homogeneous polynomials in the x_i, where the coefficients are rational functions of the χ_i. After finding the solutions for degree Δ, they are "mapped" to degree $\Delta + 1$ by multiplying them with all x_i in turn, so that the algorithm may avoid finding those factorisable syzygies anew at the higher degree. Choosing the initial value of Δ as either 0 or 1 only makes a difference where syzygies of degree 0 can be found, which is only the case for a few very simple integrals. For the cutoff Δ_{max} Schabinger points out [6] that there is no known way of determining from mathematics alone whether there will be any further non-factorisable new syzygies beyond a given degree, but that physics can provide a cutoff by considering the allowed mass dimensions in the numerator of an integral. In the case of difference equations this does not hold however, since all integrals contain a symbolic propagator exponent which makes the numeric value of the mass dimension of the integral indeterminable. Δ_{max} therefore has to be chosen manually with regard to the difficulty of the master integral considered and computational limits. The highest value required in this thesis was $\Delta_{max} = 4$ (for details see section 6). Since there is no guarantee that the basis for the syzygies does not contain elements beyond the chosen cutoff, one should note at this point that using the Schabinger algorithm may therefore result in a loss of information that would be contained in a complete basis for the syzygies generated e.g. via Gröbner bases. In practice, this was not an issue so far, since one is primarily interested in finding a difference equation and any information beyond that is not necessarily required.

To write out the algorithm in detail, I first have to introduce some further definitions. For each degree Δ one constructs the set

$$M_\Delta = \left\{ X_1^{(\Delta)}, \ldots, X_{\frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!}}^{(\Delta)} \right\} \tag{4.7}$$

of all possible monomials of that degree built from the basis elements in X. This means that $M_0 = \{1\}$, $M_1 = X$, $M_2 = \{x_1^2, x_1x_2, \ldots, x_\xi^2\}, \ldots$ It can be seen from basic combinatorics that there are $\frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!}$ such monomials. Schabinger then introduces a set of dummy variables

$$T = \{t_1, \ldots, t_\gamma\} \tag{4.8}$$

which fulfil $t_i t_j = 0$ for $i \neq j$ and will only serve to distinguish which propagator an expression is related to in the following. It is not necessary to implement these, but they simplify the task of illustrating the algorithm. One can use them to define the vector

$$P_0 = E \cdot \mathbf{t} \tag{4.9}$$

which holds the same information as the matrix E, but with the column index of an element now represented by the t_i. Building on this, Schabinger then defines further vectors P_Δ as the outer product of P_0 with M_Δ:

$$P_\Delta = \left(P_{0,1}X_1^{(\Delta)}, \ldots, P_{0,1}X_{\frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!}}^{(\Delta)}, \ldots, P_{0,\rho}X_1^{(\Delta)}, \ldots, P_{0,\rho}X_{\frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!}}^{(\Delta)} \right). \tag{4.10}$$

Since P_0 has ρ elements, P_Δ will have

$$|P_\Delta| \equiv \rho \frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!} \tag{4.11}$$

elements. The P_Δ have the advantage that a syzygy α of degree Δ, which has to fulfil

$$\alpha \cdot P_0 = 0, \tag{4.12}$$

can now be expressed instead as a vector $\sigma(\alpha)$ of degree 0 which fulfils

$$\sigma(\alpha) \cdot P_\Delta = 0. \tag{4.13}$$

In the following, I will distinguish α from $\sigma(\alpha)$ by calling the former a *syzygy* and the latter a *syzygy vector*, even though both are technically vectors. The information about which monomial in X a single term in an element of α carried is now encoded in the indices of the syzygy vector $\sigma(\alpha)$. To reconstruct α from $\sigma(\alpha)$, one has to first partition $\sigma(\alpha)$ into vectors z_i of length $\frac{(\Delta+\xi-1)!}{\Delta!(\xi-1)!}$: $\sigma(\alpha) = (z_1, \ldots, z_\rho)$. Each z_i then corresponds to one element of α:

$$\alpha = (z_1 \cdot M_\Delta, \ldots, z_\rho \cdot M_\Delta). \tag{4.14}$$

Internally, the algorithm will work with the $\sigma(\alpha)$. This is beneficial because for most operations it takes the variables x_i completely out of the picture in favour of indices (which are simpler to deal with), but comes at the cost of significantly increasing the sizes of vectors and matrices. As a result, many matrix quantities will also be sparse, which should be taken into account in an implementation (see section 5.1).

One last important quantity that will appear in the algorithm remains to be defined. The set B_Δ will hold all independent syzygy vectors $\sigma(\alpha)$ found at degree Δ as well as those found at lower degrees mapped to degree Δ. Now that the stage is set, Fig. 4.1 will give an overview of the algorithm as proposed by Schabinger [6].

It should be noted that Fig. 4.1 is primarily designed to illustrate the basic idea of the algorithm and by no means portrays an optimised solution. The implementation I will describe in section 5.1 will differ in several aspects. Some of the steps shown also warrant further discussion:

- *Subroutine 1* maps syzygies to higher degrees. This can be understood as multiplying a syzygy α with each element of the basis X in turn. The results will be syzygies with their degree raised by one. Since all of them are by construction factorisable syzygies which hold no additional information over α, they are only used to prevent the algorithm from finding linear combinations of them again. For efficiency reasons the mapping is not an actual multiplication of the syzygy α with x_i, but rather the equivalent mapping of indices for the syzygy vector $\sigma(\alpha)$.
- *Subroutine 2* is where the syzygies, or more precisely the syzygy vectors $\sigma(\alpha)$, are actually found. The positions of the pivot columns of factorisable syzygies in B_Δ mapped up from lower degrees are set to zero in the vectors y to prevent the same syzygies being found again by this subroutine. In the dot product $y \cdot P_\Delta$ the coefficients of each distinct monomial in the x_i and t_i have to vanish individually, resulting in a number of constraints which can be expressed as a matrix equation $Q \cdot y = 0$. The result D is then a basis for the null space of Q.

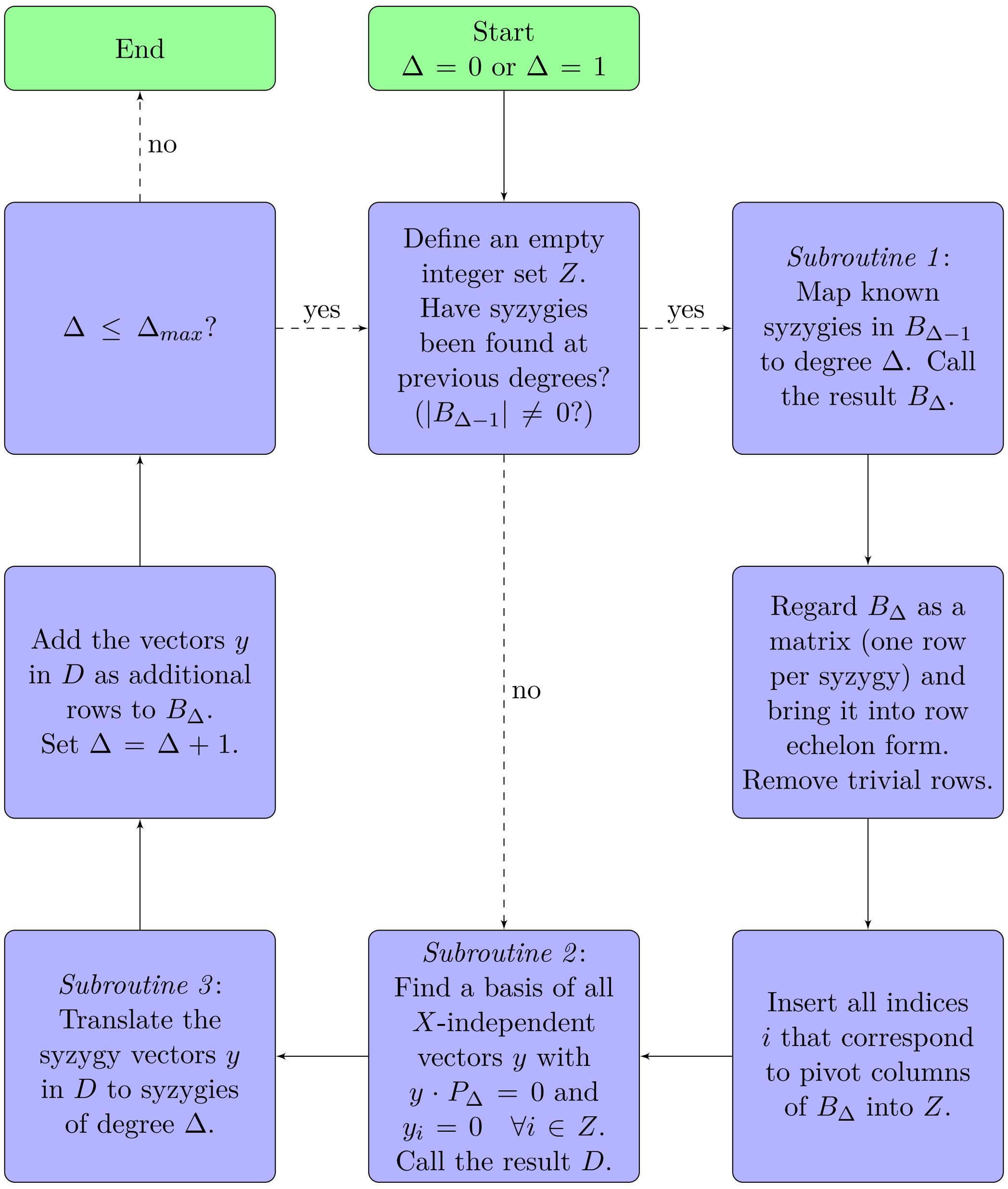
End
Start
$\Delta = 0$ or $\Delta = 1$
no
$\Delta \leq \Delta_{max}$?
yes
Define an empty integer set Z. Have syzygies been found at previous degrees? ($|B_{\Delta-1}| \neq 0$?)
yes
Subroutine 1: Map known syzygies in $B_{\Delta-1}$ to degree Δ. Call the result B_Δ.
Add the vectors y in D as additional rows to B_Δ. Set $\Delta = \Delta + 1$.
no
Regard B_Δ as a matrix (one row per syzygy) and bring it into row echelon form. Remove trivial rows.
Subroutine 3: Translate the syzygy vectors y in D to syzygies of degree Δ.
Subroutine 2: Find a basis of all X-independent vectors y with $y \cdot P_\Delta = 0$ and $y_i = 0 \quad \forall i \in Z$. Call the result D.
Insert all indices i that correspond to pivot columns of B_Δ into Z.

Figure 4.1: Flow chart of Schabinger's syzygy algorithm [6].

- *Subsection 3* produces the final results of the algorithm by mapping the syzygy vectors y to syzygies α of degree Δ as described in Eq. (4.14). Under some circumstances (see section 5.1) it can be useful to omit this step and generate the GKK relations for integrals directly from the syzygy vectors y.

4.2 Example

I will demonstrate the steps of Schabingers algorithm using a simple example, where I choose

$$P_0 = (x_1 + x_2, x_1, x_1 - x_2), \tag{4.15}$$

$X = \{x_1, x_2\}$ and $\Delta_{max} = 1$. A P_0 defined as in Eq. (4.9) would also contain a variable t_i in each term, but this is not needed for illustration purposes. Starting at $\Delta = 0$ the algorithm jumps directly into subroutine 2, since no syzygies could have been found yet. Requiring that $y \cdot P_0 = 0$ where $y = (y_1, y_2, y_3)$ yields

$$\begin{aligned} & 0 = y_1 + y_2 + y_3 \\ \wedge \quad & 0 = y_1 - y_3 \end{aligned} \tag{4.16}$$

and thus only one independent syzygy vector which I will choose as $y = (1, -2, 1)$. The work of subroutine 3 in this case is trivial since for $\Delta = 0$ the syzygy vector y is identical to the corresponding syzygy α. Continuing at $\Delta = 1$ leads to the first pass through the right branch of Fig. 4.1, because B_0 now contains the syzygy vector already found. Subroutine 1 then maps that vector to $\Delta = 1$, resulting in

$$B_1 = \{(1, 0, -2, 0, 1, 0), (0, 1, 0, -2, 0, 1)\}. \tag{4.17}$$

These vectors correspond to the factorisable syzygies $(x_1, -2x_1, x_1)$ and $(x_2, -2x_2, x_2)$. Since B_1 written as a matrix is already in row echelon form, one can directly read of the pivot columns as 1 and 2, which will be needed as input in subroutine 2. P_1 can be generated as the outer product of P_0 and $M_1 = (x_1, x_2)$:

$$P_1 = (x_1^2 + x_1 x_2, x_1 x_2 + x_2^2, x_1^2, x_1 x_2, x_1^2 - x_1 x_2, x_1 x_2 - x_2^2). \tag{4.18}$$

Subroutine 2 then searches for vectors y with $y \cdot P_1 = 0$ where $y = (0, 0, y_3, y_4, y_5, y_6)$. Setting elements 1 and 2 to zero prevents the algorithm from finding the solutions in B_1 again. One gets the constraints

$$\begin{aligned} & 0 = y_3 + y_5 \\ \wedge \quad & 0 = y_4 - y_5 + y_6 \\ \wedge \quad & 0 = y_6, \end{aligned} \tag{4.19}$$

which again allows one independent solution, chosen as $y = (0, 0, 1, -1, -1, 0)$. This vector is then added to B_1 and subroutine 3 uses it to construct the corresponding syzygy $\alpha = (0, x_1 - x_2, -x_1)$.

4.3 Symmetry and zero integral extensions

As already mentioned at the beginning of section 4, it might be possible to lose information when using the GKK syzygy approach, because the information about symmetries and zero sectors only enters the system after equations have been generated. The algorithm of Schabinger (section 4.1) offers an easily accessible way of solving this problem. In this section, I will first propose an extension to this algorithm, which introduces to it the symmetry information, and then a similar extension for the zero sectors. A discussion of the usefulness of these extensions can be found in section 6.2.

4.3.1 Symmetry extension: A simple example

To start with I will consider the 2-loop massive vacuum sector with $ID = 7$. This sector is already known from section 2.4 and its integrals are expressed as

$$I(a_1, a_2, a_3), \tag{4.20}$$

where all 3 exponents are positive and the integrals are symmetric under any permutation of the a_i. If one is looking for a difference equation for the master integral $I(1,1,1)$ at position $x = 1$, the symmetry effectively reduces to $a_2 \leftrightarrow a_3$, since one wants the symbolic exponent to stay in place. The matrix E (Eq. (4.5)) in the GKK approach then reads

$$E = \begin{pmatrix} \frac{\partial D_2}{\partial k_1^\mu} k_1^\mu & \frac{\partial D_3}{\partial k_1^\mu} k_1^\mu \\ \frac{\partial D_2}{\partial k_1^\mu} k_2^\mu & \frac{\partial D_3}{\partial k_1^\mu} k_2^\mu \\ \frac{\partial D_2}{\partial k_2^\mu} k_1^\mu & \frac{\partial D_3}{\partial k_2^\mu} k_1^\mu \\ \frac{\partial D_2}{\partial k_2^\mu} k_2^\mu & \frac{\partial D_3}{\partial k_2^\mu} k_2^\mu \\ D_2 & 0 \\ 0 & D_3 \end{pmatrix} . \tag{4.21}$$

Note that D_1 does not appear because a raised symbolic exponent at position 1 is perfectly acceptable in a difference equation. To introduce the information about the symmetry $a_2 \leftrightarrow a_3$, one can add additional rows to E in the form

$$E_{sym} = \begin{pmatrix} s_1 & 0 \\ 0 & s_1 \end{pmatrix} , \tag{4.22}$$

where s_1 is a dummy variable of mass dimension 0. This effectively replaces the GKK constraint for the $\alpha_{i,j}$ (Eq. (4.1)) with the less restrictive

$$\sum_{i=1}^{n} \sum_{j=1}^{n+n_e} \alpha_{i,j} \frac{\partial D_c}{\partial k_i^\mu} q_j^\mu = \xi_c D_c + S_c, \quad c = 2, 3 \tag{4.23}$$

For $S_c \not\propto D_c$ (in fact, S_c may not depend on D_c at all, see below) the second term on the right hand side of Eq. (4.23) will produce integrals with a raised exponent a_c. To ensure that such integrals cancel in the complete relation, one has to also enforce

$$S_2 + S_3 = 0. \tag{4.24}$$

This is sufficient because the integrals will then only differ in the position of the raised exponents and are thus identical upon application of the symmetry. It is assumed here and in the following that the exponents involved in the symmetry are identical in the seed integral. The constraint in Eq. (4.24) can be implemented in subroutine 2 of the Schabinger algorithm. Where in section 4.1 for the dot product $y \cdot P_\Delta$ the coefficient of each monomial in the x_i and t_i had to vanish, there will now be two constraints. The coefficients for each such monomial still have to vanish individually, but only after substituting the dummy variable(s) s_i to 1, since one does not want the solutions to depend on it/them. Additionally, the coefficients of all different monomials in the x_i and s_i have to vanish upon substituting $t_i = 1$, thus enforcing Eq. (4.24) by connecting the coefficients for different propagators.

There are several complications that I skipped over in this brief summary of the symmetry extension. I already mentioned that the S_c may not depend on the propagator D_c. If there was any term in S_2 (and therefore also in S_3) that contained D_2 but not D_3, that term would cause an integral with raised exponent a_3, but the raised exponent a_2 would be cancelled immediately by the factor D_2. As a result, the two integrals would not be identical upon application of the symmetries and therefore cannot cancel each other. One would be left with one integral with raised exponent a_3. This could in theory be avoided by having the term depend on D_2 and D_3 in the same way, but such solutions would be identical to ones already found with the original algorithm, since they do not contain any raised propagators to begin with. The conclusion one should draw is therefore to restrict the S_c from depending on D_c right from the start. Since all quantities with a mass dimension are expressed using the basis elements $x_i \in X$ and the scalar products Schabinger chose as the x_i depend on the D_i as linear combinations, imposing this restriction does not seem to be straightforward. The most natural solution is to change the basis X such that the x_i are chosen as the propagators rather than scalar products:

$$X_{sym} = \{D_1, \dots, D_m, m^2\}. \tag{4.25}$$

One can then customise this basis for each index position p in P_0 that corresponds to a row of the symmetry extension part of E. In the example above this leads to

$$\begin{aligned} X_{sym,p} &= \{D_1, D_2, D_3, m^2\}, \quad p = 1, \dots 6, \\ X_{sym,7} &= \{D_1, D_3, m^2\}, \\ X_{sym,8} &= \{D_1, D_2, m^2\}. \end{aligned} \tag{4.26}$$

In this simple example one could also remove D_3 from $X_{sym,7}$ and D_2 from $X_{sym,8}$, since S_2 and S_3 can only contain the same kinds of terms anyway, but for more complex symmetries this is not allowed (see below). As a consequence of having different bases $X_{sym,p}$, one then also has to introduce separate $M_{\Delta,p}$ for the indices p which are based on the $X_{sym,p}$ and the definition of the P_Δ changes from being the outer product of P_0 and M_Δ to

$$P_\Delta = \Big(P_{0,1} M_{\Delta,1}, \dots, P_{0,\rho} M_{\Delta,\rho}, P_{0,\rho+1} M_{\Delta+1,\rho+1}, \dots, P_{0,\rho_{sym}} M_{\Delta+1,\rho_{sym}} \Big), \tag{4.27}$$

where P_Δ is to be understood as a flattened vector and ρ_{sym} is the original number ρ of rows in E plus the number of rows in E_{sym}. For the symmetry extension positions one has to choose $M_{\Delta+1,p}$ with the degree raised by one compared to the original algorithm, because E_{sym} has degree 0 instead of 1. With the above definition of P_Δ and the additional constraints in subroutine 2 one can ensure that the S_c cannot depend on the respective propagators D_c and fulfil $\sum_c S_c = 0$.

Another potential problem when dealing with a sector symmetry of a sector ID is the occurrence of propagators $D_i \notin P_{ID}$. As seen in section 2.3, a sector symmetry is a one-to-one mapping $P_{ID} \to P_{ID}$ for the propagators with positive exponents, but any propagators with negative exponents may also be transformed by the symmetry. If this is the case, the integrals with raised exponents that arise from the symmetry extension might no longer be identical after applying the symmetry. This can be demonstrated on the 2-loop massive vacuum sector ID 6 which contains integrals of the form $I(a_1 > 0, a_2 > 0, a_3 \leq 0)$ and has a sector symmetry which translates to $a_1 \leftrightarrow a_2$. Using the syzygy algorithm with the symmetry extension as described above on the seed integral $I(1,1,-1)$, one might encounter the integrals $I(2,1,-1)$ and $I(1,2,-1)$. By applying a symmetry transformation, the first two exponents can be made identical in both integrals so that they cancel each other. In this simple case this works, because D_3 is left unchanged by the symmetry. If D_3 would transform to anything else upon applying the symmetry, the resulting integrals with raised exponents would not be identical and therefore remain in the equation. For each symmetry one thus has to determine which of the $D_i \notin P_{ID}$ remain unchanged by the symmetry transformation and prevent all others from occurring in the integrals. This means removing the unwanted propagators from the bases $X_{sym,p}$ corresponding to the rows of E_{sym}, but also rejecting seed integrals that contain them for all syzygies that have non-zero elements at the symmetry extension positions. It should be noted that in some cases these constraints may lead to a loss of information, since it is possible that e.g. two propagators D_3 and D_4 with the same negative exponent would be mapped as $D_3 \leftrightarrow D_4$ by the symmetry transformation, in which case the symmetry extension would theoretically still work, even though the above prescription prohibits this scenario.

One last complication of the symmetry extension concerns subroutine 1. Usually this would map the previously found syzygies by multiplying them with each element of the basis X in turn, but with the symmetry extension one introduces different bases $X_{sym,p}$ for each position p in P_0. Subroutine 1 therefore has to check for each such multiplication whether it is allowed by the choice of basis at each position p in the range $[\rho+1, \rho_{sym}]$. If a syzygy has a non-zero entry at that position, it may only be multiplied by elements of $X_{sym,p}$.

4.3.2 Generalisation to more complex symmetries

The discussion of the symmetry extension so far has been limited to the case where an integral is symmetric under the exchange of 2 propagator exponents. I will now give some examples that show how it can be generalised to work with more complex symmetries.

- *Case 1*: The integral exhibits more than one symmetry.
  I will assume for now that there are exactly two symmetries of the kind already shown above. If the integral is symmetric under $a_1 \leftrightarrow a_2$ as well as under $a_3 \leftrightarrow a_4$ independently, one can simply add the necessary rows for both symmetries to the

matrix E:

$$E_{sym} = \begin{pmatrix} s_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & s_1 & 0 & 0 & & 0 \\ 0 & 0 & s_2 & 0 & & 0 \\ 0 & 0 & 0 & s_2 & \cdots & 0 \end{pmatrix}, \tag{4.28}$$

where the columns to the right correspond to any further propagator positions in C. The bases $X_{sym,p}$ already have to be chosen depending on the position p in P_0, so it is no problem to remove D_i from the position corresponding to the i-th line in E_{sym} for $i = 1, 2, 3, 4$. The only further change is that subroutine 2 will now gain two extra sets of constraints instead of one, since $S_1 + S_2$ and $S_3 + S_4$ have to vanish independently. This method can be trivially generalised to any number of symmetries and works in the same way for the more complex symmetries discussed below.

- *Case 2*: The integral is symmetric under any permutation of 3 or more propagator exponents.
  Without loss of generality, I will assume a symmetry in exactly 3 exponents, a_1, a_2 and a_3. There are two possible approaches to this kind of symmetry. One could think of it as the combination of three separate symmetries $a_1 \leftrightarrow a_2$, $a_1 \leftrightarrow a_3$ and $a_2 \leftrightarrow a_3$ and implement these as seen in case 1. However, there might be syzygies where

  $$S_1 + S_2 + S_3 = 0 \tag{4.29}$$

  and none of the S_i can be made to vanish by linear combinations. Such syzygies would then be missed by this approach. Furthermore, the number of additional lines in E_{sym} would grow very quickly with the amount of propagators involved in the symmetry, which would increase the vector and matrix sizes of the calculation and thus the runtime of the algorithm. It is therefore sensible to instead consider an extension

  $$E_{sym} = \begin{pmatrix} s_1 & 0 & 0 & \cdots & 0 \\ 0 & s_1 & 0 & & 0 \\ 0 & 0 & s_1 & \cdots & 0 \end{pmatrix}, \tag{4.30}$$

  and enforce only Eq. (4.29) in subroutine 2. The bases $X_{sym,p}$ would then be chosen as

  $$\begin{aligned} X_{sym,\rho+1} &= \{D_2, D_3, \dots\}, \\ X_{sym,\rho+2} &= \{D_1, D_3, \dots\}, \\ X_{sym,\rho+3} &= \{D_1, D_2, \dots\}. \end{aligned} \tag{4.31}$$

  Here one can no longer remove e.g. D_3 from all bases as in the simple example above, since this would remove syzygies where $S_1 + S_2 = 0$ and S_1, S_2 depend on D_3.

- *Case 3*: The integral has a symmetry which requires the simultaneous exchange of more than one pair of exponents, e.g.

  $$\begin{pmatrix} a_1 \\ a_3 \end{pmatrix} \leftrightarrow \begin{pmatrix} a_2 \\ a_4 \end{pmatrix}. \tag{4.32}$$

This scenario is similar to case 1 insofar as one has to ensure that S_1+S_2 and S_3+S_4 vanish independently. This can again be achieved by the same

$$E_{sym} = \begin{pmatrix} s_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & s_1 & 0 & 0 & & 0 \\ 0 & 0 & s_2 & 0 & & 0 \\ 0 & 0 & 0 & s_2 & \cdots & 0 \end{pmatrix} . \tag{4.33}$$

The difference is that upon shifting e.g. a raised exponent from a_2 to a_1, a_3 and a_4 also swap positions. To ensure that the two resulting integrals are identical in those exponents as well, one has to require that S_1 be dependent on D_3 in exactly the same way as S_2 depends on D_4 and vice versa. This means that the above mentioned constraint $S_1 + S_2 = 0$ is not quite correct and has to be replaced with $S_1 + S_2|_{D_3 \leftrightarrow D_4} = 0$. While it is possible to achieve this, it requires careful mapping of the respective terms in subroutine 2. A somewhat simpler approach is the more restrictive constraint that both S_1 and S_2 may only depend on the product D_3D_4, which only requires removing any terms with different powers in those propagators from $M_{\Delta,\rho+1}$ and $M_{\Delta,\rho+2}$. Because this would lead to $S_2|_{D_3 \leftrightarrow D_4} = S_2$, no mapping would be required in subroutine 2. The simplicity of this approach comes with a possible loss of information by placing too strict a constraint on the syzygies. Additionally, subroutine 1 would not be allowed to map syzygies found in this way by multiplying them with either D_3 or D_4, but only with the product D_3D_4, which means those syzygies could only be mapped in increments of two in their degree Δ.

The above cases can be combined without restrictions to cover all possible sector symmetries. An interesting example of this is the 5-loop massive vacuum sector with ID 30239, which is symmetric under any permutation of

$$\{ \begin{pmatrix} a_1 \\ a_6 \\ a_3 \end{pmatrix}, \begin{pmatrix} a_{11} \\ a_2 \\ a_5 \end{pmatrix}, \begin{pmatrix} a_{13} \\ a_{15} \\ a_{12} \end{pmatrix} \} \tag{4.34}$$

as well as under

$$\begin{pmatrix} a_1 \\ a_2 \\ a_{13} \end{pmatrix} \leftrightarrow \begin{pmatrix} a_6 \\ a_{11} \\ a_{15} \end{pmatrix} \tag{4.35}$$

and therefore exhibits a combination of all three cases discussed above.

4.3.3 Zero integral extension

The information about which integrals vanish in dimensional regularisation can be introduced to the Schabinger algorithm in a way similar to the symmetry extension approach. The constraints on the $\alpha_{i,j}$ have to be relaxed to

$$\sum_{i=1}^{n} \sum_{j=1}^{n+n_e} \alpha_{i,j} \frac{\partial D_c}{\partial k_i^\mu} q_j^\mu = \xi_c D_c + Z_c, \quad c \in C, \tag{4.36}$$

which is achieved by extending the matrix E with

$$E_0 = \mathbb{1}_{|C|\times|C|} \tag{4.37}$$

as additional rows. Z_c must be chosen such that a seed integral vanishes if Z_c is multiplied to its integrand before integration. If one chooses the seed integral with the greatest exponents for this constraint, it will also hold for all other seed integrals. As in the symmetry extension, Z_c may not depend on D_c, since such terms would not cause raised exponents and therefore only lead to redundant syzygies. Consequently one has to also choose the D_i as the elements x_i of the basis X. The limitations on Z_c can then be enforced by choosing the $M_{\Delta,p}$ not as the set of all possible monomials of the x_i of degree Δ, but only those which do not depend on D_c and will cause the resulting integral to vanish. Once again the extension positions of the P_Δ will have to be built using the $M_{\Delta+1,p}$ instead of the $M_{\Delta,p}$, since the degrees of E and E_0 differ by one. Mapping syzygies to a higher degree in subroutine 1 can be done by multiplying them with any of the x_i (except those which would cancel a raised propagator), since this cannot cause vanishing integrals to not vanish any longer in dimensional regularisation.

The zero integral extension can be used in conjunction with the symmetry extension. It is however possible that this will lead to redundant syzygies being found, since some integrals with raised exponents might be cancelled after applying a symmetry as well as vanish in dimensional regularisation. In this case, both extensions would allow those integrals, but use different positions of the syzygies to do so.

5 Implementation: *SPADES*

For this thesis I have written a C++ program called *SPADES* (Some Program to Automatically find Difference Equations via Syzygies) which is designed to find difference equations for Feynman integrals. Due to the time constraints of writing a master's thesis, it is currently limited to vacuum integrals, but I plan to remove this limitation in the future. The program can roughly be divided into two parts: The syzygy part implements the algorithm described in section 4 including the symmetry and zero extensions. The resulting syzygies are used by the difference equation part of the program to generate a system of linear relations between integrals, which is then solved for a difference equation using a variation of the Laporta algorithm. For most of the algebraic manipulations in the syzygy part I make use of the library *GiNaC* [43], while the polynomial manipulations of the integral coefficients in the Laporta algorithm, which usually take up the bulk of the program's runtime, are performed by the external program *Fermat* [44]. The part of the code that is responsible for the communication with *Fermat* is an adaption from the equivalent part from the source code of *Reduze 2* [4] (© 2012, A. von Manteuffel, C. Studerus). Since the ideas of the syzygy algorithm and the Laporta algorithm have already been described in sections 4 and 3.2 respectively, I will use this section to highlight some of the more technical details in their implementations.

5.1 The syzygy algorithm in practice

To find the syzygies that can be used to generate a difference equation for a master integral I with the symbolic exponent at position x, the user is required to provide the following information:

- The auxiliary topology A_n,
- the positive exponents of I and the position x of the symbolic exponent,
- the maximum degree Δ_{max} up to which syzygies are to be searched,
- whether the symmetry and/or zero integral extensions should be used,
- the sector symmetries, if the symmetry extension is enabled,
- a list of non-trivial zero-sectors, if the zero integral extension is enabled.

The program then starts the preparations for the actual algorithm by constructing the matrix E. The set C of exponent positions that may not be raised is chosen as the set of positions of positive exponents excluding the symbolic one. If the symmetry extension is activated, the user-provided symmetries will be checked for viability before E_{sym} is constructed as described in section 4.3. This means removing or trimming symmetries the symbolic exponent is involved in, as well as checking that exponents in the input are identical where necessary. When the construction of E is completed, the scalar products and masses it contains would need to be expressed via the $x_i \in X$ and the χ_i. However, since I will consider only vacuum integrals with a single mass scale in this thesis, there are no ratios χ_i between masses or external scalar products necessary. This means that all coefficients of monomials in X will just be rational numbers, which simplifies calculations by a great deal, because no polynomial algebra will be necessary.

For the basis X the program offers two options. The x_i can either be chosen as the scalar products of momenta or as the propagators D_i. The mass scale m^2 is included in either case. Choosing to activate either the symmetry or zero integral extension will force the basis to be built out of the propagators. Experience shows that the algorithm takes roughly twice as long if the x_i are propagators instead of scalar products. In most cases this is not a severe problem, since usually the time spent on finding syzygies is small compared with the duration of the Laporta algorithm. Details on runtimes will be discussed in section 6.3.

Once the matrix E is expressed via the x_i and P_0 is obtained as the dot product of E with the dummy variables t_i, the $M_{\Delta,p}$ need to be constructed for $\Delta \leq \Delta_{max}$. For positions p corresponding to either the standard or symmetry rows of E, these are simply the lexicographically ordered set of all monomials of degree Δ which can be built out of the respective bases X_p for each position. For the positions of the zero integral extension each such monomial is tested as to whether it will cause the integral with the user-defined exponents to vanish when it is multiplied to the integrand, and only those monomials that do will be inserted into the corresponding $M_{\Delta,p}$. The P_Δ are then determined from the $M_{\Delta,p}$ as defined in section 4.1. At the same time the program creates a map for later use by subroutine 1, which holds information about which position in a syzygy vector for degree $\Delta+1$ each element in a syzygy vector for degree Δ is mapped to when multiplying with any of the x_i. This concludes the preparations and the program then moves on to the main algorithm.

Since Schabinger's algorithm is already described in Fig. 4.1, I will in the following not go through the implementation of each of its steps, but rather focus on those where I have made changes for efficiency as well as obstacles that come up in an implementation. As Schabinger already mentions in his paper [6], the main problem one is faced with in the algorithm is that the syzygy vectors $\sigma(\alpha)$ and the matrix quantities composed of them (B_Δ, the matrix Q in subroutine 2) grow very large. The number of elements of one such vector at degree Δ for an n-loop calculation with n_e independent external momenta, ξ elements in the basis X and γ exponents that may not be raised, without activating either of the extensions, is given by

$$\frac{(\Delta+\xi-1)!}{\Delta!\,(\xi-1)!}\,(n\,(n+n_e)+\gamma)\,. \tag{5.1}$$

For the 5-loop vacuum diagrams I will consider in section 6 at degree $\Delta = 4$ this is roughly $\mathcal{O}(10^5)$. The situation worsens if one also activates the symmetry or zero extensions, since they add additional rows to E and therefore increase the length of the syzygies. At the same time the vectors are very sparse, typically less than 1% of the elements are not zero. This has to be taken into account in the data structure of an implementation to improve the efficiency of the algorithm. In *SPADES* all syzygy vectors $\sigma(\alpha)$ are expressed via the class `sparseLst`, which is essentially a wrapper class for a sorted list of index-element pairs (using `std::map<unsigned int, GiNaC::ex>`), in which only the non-zero elements are stored. Added on top of that are algebraic capabilities such as vector addition. While this structure is slightly less efficient for accessing or modifying only a specific element than a static vector (logarithmic vs. constant time), storing only the $< 1\%$ non-trivial elements saves a lot of memory and is very efficient for frequently required index-independent operations such as multiplying the vector with a common factor. The matrices B_Δ and Q are stored as vectors of `sparseLst`s, where each `sparseLst` represents one row. A different example where the `sparseLst` structure is useful is subroutine 1 (see Listing 5.1), which maps the syzygy vectors to a higher degree. Instead of having to go through all elements to find the non-zero ones, it is sufficient to iterate over all the elements of the `sparseLst`.

Listing 5.1: Subroutine 1

```
void syzygyModule::subroutine1(const vector<sparseLst> & oldBd,
   vector<sparseLst> & bd, unsigned int delta) {
  //Get size of syzygy vectors of degree Delta
  unsigned int vecSize = getVecSize(delta);

  //Load mapping of positions. newIdx = m[oldIdx][pos of x_i]
  //(newIdx = -1 if map not allowed.)
  const vector<vector<int> > & m = getVecMap(delta-1);

  //actual mapping
  for(vector<sparseLst >::const_iterator syzV = oldBd.begin();
     syzV != oldBd.end(); ++syzV){
    //for each syzygy vector in oldBd = B_(Delta-1)
    for(unsigned int xPos = 0; xPos < xs.size(); ++xPos){
      //for each x_i
      bool allowedMap = true;
      sparseLst mappedVec(vecSize);
      for(map<unsigned int, ex>::const_iterator el = (*syzV).
         elemsBegin(); el != (*syzV).elemsEnd(); ++el){
        //for each element (*el) = (index, value) of syzV
        if(m[(*el).first][xPos] < 0){//map not allowed
          allowedMap = false;
          break;
        }
        else{
          mappedVec.set(m[(*el).first][xPos], (*el).second);
        }
      }//el
      if(allowedMap){
        //add mapped syzygy vector to B_Delta
        bd.push_back(mappedVec);
      }
    }//xPos
  }//syzV
}
```

The vectors resulting from subroutine 1 are always sparser than the original vectors, since they have the same amount of non-zero elements but a greater size. Nevertheless, they are not necessarily linearly independent. In fact, if a syzygy vector was found at degree Δ and inserted into B_Δ, a linear dependence is guaranteed for all $B_{\Delta'}$ with $\Delta' \geq \Delta + 2$, since all vectors are always mapped to the next degree using "multiplication" with all x_i. This means that for degrees that differ by at least 2, there are different orders of applying the x_i in which vectors might be mapped up with the same end result, all of which will be realised. To avoid this would not be straightforward, since the intermediate matrix $B_{\Delta+1}$ is transformed to a row echelon form (or a similar form, see below) by linear combinations of its rows, making a simple identification of which vector has already been mapped up using which x_i impossible. However, it might be worth investigating in the future whether a more sophisticated approach to the mapping in subroutine 1 will lead to less redundant vectors in $B_{\Delta'}$.

Schabinger in his algorithm introduced a step where the matrix B_Δ is brought into row echelon form after subroutine 1 to remove redundant rows as well as find the pivot columns, so that the corresponding positions could be set to zero while searching new syzygy vectors at degree Δ. It turns out that this constraint is too strict for the task one is trying to accomplish, since it requires finding the pivot columns from left to right, always choosing the leftmost column possible. To remove the degrees of freedom in subroutine 2 that would lead to the syzygy vectors that are already in B_Δ being found anew, it is sufficient to find any possible set of pivot columns, not necessarily the leftmost, and in an arbitrary order. This freedom of choice and the sparseness of the B_Δ can be used to replace the row echelon form step in Schabinger's algorithm with a faster algorithm for finding a set of pivot columns and removing redundant rows. In *SPADES* this is implemented in the method `lSolveSparse`.

In principle, `lSolveSparse` is just a Gauss algorithm, but with a considerable overhead of operations to determine an optimised order of cancelling elements. I have found that the runtime of the algorithm for the matrices one considers is extremely sensitive to this choice of order, which for larger matrices can make a difference of several orders of magnitude. This is due to the fact that unfortunate choices can result in very large fractions (with both numerator and denominator of e.g. $\mathcal{O}(10^{50})$) for the elements of the matrix in intermediate steps. It is therefore worthwhile to invest a considerable amount of computation time into avoiding such lengthy expressions. The best order I have found, which is implemented in `lSolveSparse`, is:

1. Choose as pivot column the column with the least number (but > 0) of entries in *unfinished* rows. If there is more than one, pick the leftmost.

2. Choose as pivot element in the pivot column the one in the *unfinished* row with the least number of entries. If there is more than one, pick the first.

3. Divide the row of the pivot element by the pivot element and mark the row as *finished*.

4. Remove all entries in the pivot column from *unfinished* rows by subtracting from the rows the row of the pivot element with an appropriate factor.

5. If there are any non-trivial *unfinished* rows left, go back to step 1, otherwise finish.

Steps 1 and 2 can possibly be further improved by not simply picking the leftmost column and first row with the least number of entries respectively, but applying additional analysis of the columns and rows in question. While it is easy to keep track of the number of elements in a row, as they are all saved in one `sparseLst`, accessing the same information for the columns requires more work. `lSolveSparse` keeps track of this with an auxiliary list for each column, which holds the indices of the rows with non-zero elements at that position and is updated whenever rows are subtracted from each other. Table 5.1 shows some runtimes for `lSolveSparse` as well `Redrowech`, which is the *Fermat* command for the row echelon form of a sparse matrix. Runtimes for the latter also include the time needed to transfer the matrix to and back from the *Fermat*-executable, but this is negligible compared to the overall time for the calculation. It is obvious from these results that the order of cancellations and the choice of pivot columns have a great effect on the runtime of the problem and I believe it might be possible that an even more sophisticated order may yield an improvement of another order of magnitude. Due to the time constraints of a master's thesis, I was unable to investigate this further, but for the calculations up to 5 loops discussed in section 6 the above described `lSolveSparse` was sufficiently fast.

n	Δ	Rows	Columns	Linearly indep. rows	Non-zero elements		`lSolve-Sparse` [sec]	*Fermat*: `Redrowech` [sec]
3	2	63	364	59	639	2.8%	< 0.01	0.01
3	2	42	364	42	842	5.5%	< 0.01	0.06
4	3	$5.5 \cdot 10^3$	$6.9 \cdot 10^3$	$3.2 \cdot 10^3$	$1.8 \cdot 10^4$	0.046%	0.03	0.5
3	3	539	$1.1 \cdot 10^3$	324	$5.9 \cdot 10^3$	1.0%	0.02	0.98
3	3	301	$1.2 \cdot 10^3$	217	$4.8 \cdot 10^3$	1.9%	0.03	3.5
4	2	594	$1.5 \cdot 10^3$	523	$7.6 \cdot 10^3$	0.88%	0.02	12
4	3	$4.4 \cdot 10^3$	$5.7 \cdot 10^3$	$2.2 \cdot 10^3$	$4.3 \cdot 10^4$	0.17%	0.11	624
4	3	$5.8 \cdot 10^3$	$6.0 \cdot 10^3$	$2.8 \cdot 10^3$	$6.6 \cdot 10^4$	0.19%	0.23	4905
5	3	$2.5 \cdot 10^4$	$2.2 \cdot 10^4$	$1.1 \cdot 10^4$	$3.3 \cdot 10^5$	0.059%	1.4	-
5	3	$3.4 \cdot 10^3$	$2.9 \cdot 10^4$	$2.1 \cdot 10^3$	$3.9 \cdot 10^5$	0.41%	3.5	-
4	4	$6.7 \cdot 10^3$	$2.7 \cdot 10^4$	$3.5 \cdot 10^3$	$1.5 \cdot 10^6$	0.83%	202	-
5	4	$2.7 \cdot 10^4$	$1.4 \cdot 10^5$	$1.4 \cdot 10^4$	$8.9 \cdot 10^6$	0.23%	1034	-

Table 5.1: Comparison of runtimes for `lSolveSparse` and the `Redrowech` command of *Fermat*. The matrices are taken from actual calculations that will be discussed in section 6.

The method `lSolveSparse` also allows the option to perform what in a standard Gauss elimination would be the elimination of the upper right triangle. If the option is chosen, the pivot elements are considered in reverse order of their selection and all other remaining non-zero elements in the respective pivot column are eliminated using the row of the pivot element. For the B_Δ this is not required, but I have found that a partial elimination can benefit the runtime of later steps in the algorithm, if it decreases the total number of non-zero elements in the matrix. All such elimination steps for the "upper right triangle" are therefore considered for all B_Δ with $\Delta \lneq \Delta_{max}$, but only applied if they decrease the number of elements in the target line.

Subroutine 2 also makes use of the algorithm in `lSolveSparse` to find a basis of the nullspace of the matrix Q, but with a maximal elimination of the "upper right triangle". All columns which were not chosen as pivot columns are then considered so-called parameter columns. The corresponding elements in the syzygy vectors y will be used as a parametrisation of the nullspace. The basis for the nullspace is then chosen as the canonical basis in these parameters with the elements corresponding to the pivot columns defined by the rows of the respective pivot elements. It follows then that in the set of solutions for each parameter position there is exactly one syzygy vector with a non-zero entry at that position, while for each pivot position there are as many such syzygy vectors as there are entries in parameter columns in the row of the corresponding pivot element. If the symmetry or zero integral extensions are activated, this distribution can make a significant difference, since as seen in section 4.3 syzygy vectors with entries at extension positions may generally not be mapped up to higher degrees in subroutine 1 by multiplication with some of the x_i. It is therefore sensible to minimise the number of syzygy vectors with non-zero entries at these positions. This can be achieved by limiting the choice of pivot columns to the non-extension positions. Experience shows that the performance loss due to this constraint is small, since there are usually far more non-extension positions in the syzygy vectors than extension ones.

Most of the work and time in Schabinger's algorithm go into the application of `lSolveSparse` to the $B_\triangle$ and Q. Which of the two takes up more time depends entirely on the integral in question. If many syzygies are found at low degrees, the $B_\triangle$ will generally be larger, which in turn leads to more pivot columns and thus more positions of the syzygy vectors in subroutine 2 set to zero and as a consequence an easier matrix Q. In some cases the $B_\triangle$ therefore require up to 90% of the time needed for the whole algorithm, while in others syzygies are found at higher degrees and subroutine 2 becomes the bottleneck. This behaviour makes it difficult to predict the runtime of the algorithm for a given integral. Further details about runtimes will be given in section 6.3. To prevent the program from having to spend the time on finding the syzygies every time one starts the Laporta algorithm part (e.g. with different sets of seed integrals), *SPADES* also contains an export/import function for the syzygies. Additionally, if the x_i are chosen as the D_i, it is convenient to skip subroutine 3 entirely and generate the relations between integrals directly from the syzygy vectors y.

5.2 Finding difference equations

After the syzygies have been found, the next step is to determine the seed integrals they will be applied to in order to generate the system of equations. In the following I will assume that one is looking for a difference equation in propagator position x for a master integral

$$J = I(a_1^{(J)}, \dots, a_m^{(J)}), \tag{5.2}$$

where

$$\begin{aligned} a_i^{(J)} &> 0 \quad \text{for } i \in P_J \subseteq \{1, \dots, m\}, \\ a_i^{(J)} &= 0 \quad \text{for } i \in \bar{P}_J \equiv \{1, \dots, m\} \setminus P_J, \\ x &\in P_J, \\ t_J &= |P_J|. \end{aligned} \tag{5.3}$$

It is useful at this point to make some changes to the way integrals are described in section 2, to account for the fact that there will always be one symbolic exponent. Instead of using the sum r of all positive exponents I will define r' as the sum of all positive exponents except the symbolic one. Assuming that $a_x^{(J)}$ has already been replaced with z in J one therefore obtains

$$r'_J = \sum_{i \neq x} a_i^{(J)}. \tag{5.4}$$

I will further introduce the parameter σ of an integral $I(\{a_i\})$ as

$$\sigma = \begin{cases} s = \sum_{i \in \bar{P}_J} |a_i| & \text{, if } a_i = a_i^{(J)} \; \forall i \in P_J \\ 0 & \text{, otherwise,} \end{cases} \tag{5.5}$$

which will be needed later on. The set of seed integrals that is used in *SPADES* can then be written as

$$\left\{ I(a_1, \dots, a_x = z, \dots, a_m) \middle| a_i = a_i^{(J)} \; \forall i \in P_J \setminus \{x\}, a_i \leq 0 \; \forall i \in \bar{P}_J, \sum_{i \in \bar{P}_J} |a_i| \leq s_{max} \right\}, \tag{5.6}$$

where the maximum sum of negative exponents s_{max} has to be chosen by the user. One could in principle also consider seed integrals with a shifted symbolic exponent $z + \delta z$, but the same effect will be achieved later on by shifting z in the Laporta algorithm. There are no seed integrals with $r' < r'_J$, since r' is never raised by applying the syzygies and most (ideally all) integrals with $r' < r'_J$ that would result from such seeds are assumed to already have been solved in previous calculations. In this context, I refer to an integral I in sector ID as *solved*, if an equation is known which expresses I only in terms of integrals which differ only in the symbolic exponent from the master integrals of sector ID and the equivalent integrals of subsectors of ID. The term *unsolved integrals* will then be used for integrals for which no such relation is known. During the Laporta algorithm the program tries to solve as many integrals with $r' = r'_J$ and $s > 0$ as possible to export them for later use in calculations for sectors which have the current sector as a subsector. Should any *unsolved integrals* with $r' < r'_J$ remain in the system of equations, *SPADES* attempts to solve and export them as well, but they are only considered a side-product of the calculation.

For efficiency reasons the syzygies are not applied to all seed integrals individually, but only to one generalised seed integral with symbolic a_i for $i \in \bar{P}_J$. The resulting equations are blueprints for the whole system of equations, which is then generated by substituting all allowed values for the remaining a_i. If the symmetry extension was enabled when finding the syzygies, one has to take into account that some of the syzygies with non-zero elements in the symmetry positions have the restriction that certain a_i in the seed integrals have to be zero, for reasons explained in section 4.3. It should be noted that for very large systems of equations generating the whole system before starting the Laporta algorithm might cause the memory requirements of the program to exceed acceptable limits. For all calculations considered in this thesis (fully massive vacuum integrals up to 5-loop), this was not a problem.

After the system of equations is generated, it is first checked whether any of the occurring integrals are in trivial or non-trivial zero-sectors and can thus be removed from the system. To minimise the number of the remaining integrals, the sector shifts and symmetries described in section 2.3 are used to bring the integrals into a canonical form. In *SPADES* this means that for each topology the sector with the highest possible ID is chosen and within this sector the program tries to shift the symbolic exponent as far to the left (meaning a lower propagator position) as possible. After that the highest numeric exponent(s) is/are shifted to the leftmost possible position(s), then the second highest and so forth. A database is created to avoid bringing an integral into canonical form twice. The exact procedure to canonicalise an integral I is then given by:

1. Check whether the canonical expression of I is already known. If so, return that expression, otherwise proceed with step 2.

2. Check whether I can be mapped by a sector shift to a sector with a higher ID than the current one. If this is not the case, proceed with step 3. Otherwise perform the shift to the sector with the highest possible ID. In the case that I contained negative exponents, the result may be a sum of integrals. Bring all of those integrals into canonical form first. Sum up coefficients of identical integrals in the resulting expression, if necessary. Return the result and save it as the canonical form of I.

3. Consider the set of all possible symmetry transformations for this sector including the identity transformation. Remove all transformations from the set except the ones that shift the symbolic exponent to the leftmost possible position. If more than one transformation is left in the set, build the weight vector for each of those transformations. This is a vector of integers with the same number of elements as the set of propagators, which is constructed as follows:
   Start with each element of the weight vector initialised to zero. Add each positive numeric exponent a_i to the position it is shifted to by the symmetry transformation. The propagators with negative exponents are in general shifted to a linear combination of propagators. From the propagator positions in that linear combination pick

the lowest possible position which is not occupied by a positive exponent in I and add the negative exponent to that position in the weight vector. Repeat this for all negative exponents in a_i.
A weight vector A is considered greater than a weight vector B, if $A_v > B_v$, where v is the lowest position in which the two vectors differ. From the set of remaining transformations then remove all but those with the greatest occurring weight vector. If the identity transformation still remains in that set at this point, continue with step 4. Otherwise, apply any one of the symmetry transformations remaining in the set, as they will all result in the same expression once the result is fully canonicalised. As in step 2, the result may be a sum of integrals and needs to be canonicalised first, then saved as the canonical result of I and returned.

4. The integral I is already in canonical form. Add this information to the database and return I.

The above procedure alone would still result in unnecessary work, since two integrals that only differ in the symbolic exponent (at the same position), e.g. z and $z-1$, would be canonicalised independently in exactly the same way. This is prevented by shifting the symbolic exponent to just z before the canonicalisation of an integral and applying the reverse shift to the result.

For the Laporta algorithm one needs an ordering prescription for the difficulty of integrals as discussed in section 3.2. In *SPADES* I have implemented the following: Of two non-identical integrals I_1 and I_2, define as more difficult

- the one with the greater number t of positive exponents. If $t_1 = t_2$,
- the one with the greater sum r' of positive numeric exponents. If $r'_1 = r'_2$,
- the one with the greater sum s of absolute values of negative exponents. If $s_1 = s_2$,
- the one with the greatest individual numeric exponent. If $\max(a_i^{(1)}) = \max(a_i^{(2)})$,
- the one with the lowest individual exponent. If $\min(a_i^{(1)}) = \min(a_i^{(2)})$,
- the one with the greater position x of the symbolic exponent. If $x_1 = x_2$,
- the one with greater exponent at the lowest (leftmost) position where the numeric exponents differ. If the numeric exponents are identical at each position,
- the one with the greater symbolic exponent.

This ordering prescription is however not applied to the set of all integrals. I will classify integrals into the following five types, which the prescription is then applied to individually:

- Type I: Integrals with $r' = r'_J$ and $s > 0$.
- Type II: *Unsolved integrals* with $r' < r'_J$ and $s > 0$.
- Type III: Integrals with $r' = r'_J$ and $s = 0$.
- Type IV: *Solved integrals* with $r' < r'_J$ and $s > 0$.
- Type V: Integrals with $r' < r'_J$ and $s = 0$.

Types I and II are the integrals that one needs to eliminate from an equation for it to become a difference equation, while type III needs to remain for it to be a difference equation for J. Type V integrals build the right side of Eq. (3.1) and integrals of type IV can be expressed as integrals of type V using results from previous calculations. This substitution of the integrals of type IV can optionally be done either before or after the Laporta algorithm. I have found that especially for sectors with many subsectors the expressions via linear combinations of integrals of type V can be rather large and one may be better of waiting with the substitutions until after the Laporta algorithm.

For each type, the program orders all integrals that may occur in the calculation using the ordering prescription. This enables one to write all possible equations as rows of a matrix, where each column represents one integral, sorted from left to right by type first and then in order of decreasing difficulty. The matrix representation is chosen because it replaces the integral objects in the equations with the much simpler integer indices. A further simplification is the substitution of the mass scale m^2 with 1 in the coefficients. This is possible because the mass dimensions of the coefficients can be completely recovered from the mass dimensions of the integrals and it reduces the number of variables in the coefficients by one. The remaining variables in the case of the integrals considered in this thesis are then only z and d. If the integrals would contain external momenta or more than one mass scale, the coefficients would also depend on the ratios χ_i. For efficiency reasons, the coefficients are saved from this point on only as strings, since all the polynomial algebra in the Laporta algorithm is handled by the external program *Fermat* [44]. The communication with *Fermat* is adapted from the one found in the source code of *Reduze 2* [4] (© 2012, A. von Manteuffel, C. Studerus). A pipe to a *Fermat* executable is established before the Laporta algorithm starts and single expressions are sent to this pipe as a string. *Fermat* then simplifies the terms, brings them into a canonical form and writes the results to the pipe, from which they are read again by *SPADES* as strings. A row of a matrix is then stored using a class similar to `sparseLst` with `std::string` instead of `GiNaC::ex` as the basic object for expressions. The matrix for the whole system of equations is actually stored internally as five separate matrices, one for each type, for reasons discussed below.

While each equation in the Laporta algorithm will be stored in the same rows of the five matrices throughout the calculation, equations will be grouped into five pools and may be in more than one pool at a time. All equations start out in Pool A, which contains the equations which have not yet been used. Pool B will hold the equations that are currently under consideration, while Pool C is a temporary pool which contains equations that will be shifted via $z \to z + 1$. Pool D holds all equations that have been solved for the most difficult integral with $s > 0$, while Pool E stores all difference equations found.

The user can define two additional parameters that will be used in the Laporta algorithm, δz_{max} and σ_{min}, with default values 0 and -1 respectively. δz_{max} is the maximum number of times an equation can be subjected to the shift $z \to z + 1$. The resulting equations can generate additional information. While they could also be obtained by allowing z-shifts for the seed integrals, it is more efficient to apply the shifts within the Laporta algorithm only to equations that have turned out to not be redundant. The second parameter σ_{min} governs the amount of equations in the system that are evaluated by the program. By default the Laporta algorithm starts with all equations where $\sigma = 0$ and works its way up to higher values of σ, until a difference equation is found, at which point the algorithm terminates. If σ_{min} is set to be non-negative, the program is forced to evaluate all equations with $\sigma \leq \sigma_{min}$, even though a difference equation may have been found earlier than that. This can be useful, since in some cases the order R of the first difference equation to be found is not the minimal order that can be achieved with the information in the system of equations. The parameter σ of an equation is to be understood as the maximum value occurring for the σ of the integrals in the equation. Similarly, the difficulty of an equation is defined as the difficulty of the most difficult integral it contains. As one last definition, |A|, |B|, etc. will refer to the number of integrals in the respective pool. With this setup, the variation of the Laporta algorithm used in *SPADES* can be illustrated in the flow chart of Fig. 5.1.

The elimination steps described in Fig. 5.1 are not initially executed for all five integral types, which is one of the reasons for splitting the system of equations into separate matrices. They are executed only for types I-III to begin with, while all steps taken are recorded in the order of execution. This includes cancellations by subtracting rows, dividing rows by coefficients and applying z-shifts. After the Laporta algorithm has finished, the program first uses all difference equations in Pool E to construct the difference equation with the lowest possible order R, while still recording all steps. This equation as well as all equations from Pool D that one wants to export as equations for *solved integrals* (in the sense explained above) are then entered into a set W. For the integral types IV and V the program then only executes those steps from the record that directly or indirectly affect the final versions of the equations in W, while all other steps are discarded. This means that for equations from which all integrals of types I-III can be eliminated, or which are in any other way unneeded, the coefficients of types IV and V are never even touched. This saves a lot of time, especially for sectors with many subsectors, in which case the integrals of types IV and V can easily far outnumber those of the other types.

After all types of integrals have caught up with the necessary steps, the integrals of type IV are substituted with the respective linear combinations of integrals of type V, if this has not already been done before the Laporta algorithm. In the difference equation z is shifted such that the lowest symbolic exponent in Eq. (3.1) is just z to have a standardised way of expressing a difference equation. The right hand side of the equation also requires further work to be standardised. The symbolic exponents of all subsector integrals of type V on that side can be shifted into the range $[z, z + R_i]$ using the difference equations of the respective subsectors with orders R_i. All equations for *solved integrals* in pool D that one wants to export for use in later calculations are shifted such that the symbolic exponent is simply z with no offset. The right hand sides of these equations are then standardised in the same way as that of the difference equation. Note that it is sufficient to only export one equation for a set of *solved integrals* which only differ in the symbolic exponent, thus saving further steps in the calculation for integral types IV and V, since only one of the equations is entered into the set W.

Experience shows that if the master integral J comes from a sector with many subsectors, the steps after the Laporta algorithm, namely executing the recorded steps for the last two types of integrals and bringing the equations into a standardised form, can take up much more time than the actual Laporta algorithm itself. If one is only interested in the information whether the generated system of equations contains a difference equation and its order, but not the complete difference equation, *SPADES* offers some options that

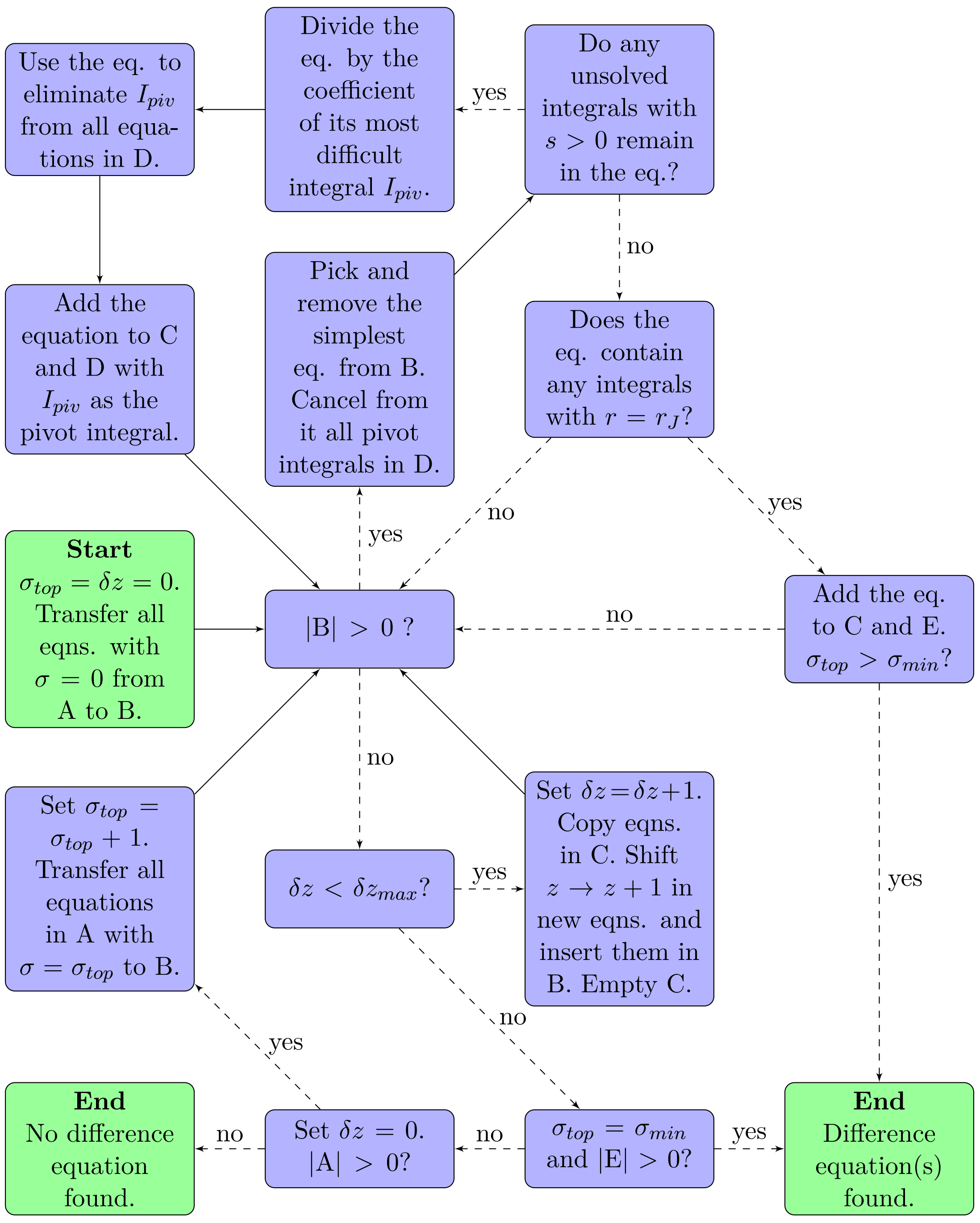
Use the eq. to eliminate I_{piv} from all equations in D.
Divide the eq. by the coefficient of its most difficult integral I_{piv}.
yes
Do any unsolved integrals with $s > 0$ remain in the eq.?
no
Add the equation to C and D with I_{piv} as the pivot integral.
Pick and remove the simplest eq. from B. Cancel from it all pivot integrals in D.
Does the eq. contain any integrals with $r = r_J$?
no
yes
yes
Start $\sigma_{top} = \delta z = 0$. Transfer all eqns. with $\sigma = 0$ from A to B.
$|B| > 0$?
no
Add the eq. to C and E. $\sigma_{top} > \sigma_{min}$?
no
Set $\sigma_{top} = \sigma_{top} + 1$. Transfer all equations in A with $\sigma = \sigma_{top}$ to B.
$\delta z < \delta z_{max}$?
yes
Set $\delta z = \delta z + 1$. Copy eqns. in C. Shift $z \to z + 1$ in new eqns. and insert them in B. Empty C.
yes
no
yes
End No difference equation found.
no
Set $\delta z = 0$. $|A| > 0$?
no
$\sigma_{top} = \sigma_{min}$ and $|E| > 0$?
yes
End Difference equation(s) found.

Figure 5.1: Flow chart of the variation of the Laporta algorithm in *SPADES*. σ_{top} signifies the σ parameter the equations currently in Pool B started out with, while δz marks how many times those equations have undergone the shift $z \to z + 1$.

speed up the process.

The user can choose to ignore the right hand side of Eq. (3.1) and only generate the left hand side of the difference equation. In this case, all integrals of types II, IV and V are set to zero right after the equations have been generated. For the Laporta algorithm this only makes a difference, if there are *unsolved integrals* from subsectors (type II), but almost all steps after the Laporta algorithm are omitted. To speed things up further, the user can choose to replace the space-time dimension d with a numeric value. This greatly simplifies the polynomial algebra and thus shortens the runtime considerably, but may also lead to incorrect results, if a coefficient should vanish only for that specific choice of d.

Substituting a number for d can however also help improve the runtime for exact calculations including subsectors. Since all steps of the Laporta algorithm are recorded anyway, *SPADES* can run it once with d as a number and identify the exact set of equations needed to generate the difference equation. A second Laporta algorithm is then started for just those equations. In this way only the minimal amount of steps necessary with the given system of equations is executed for a general d. For many of the more difficult systems of equations at the 5-loop level this approach has proven to be faster than only running the Laporta algorithm once with a symbolic d.

6 Application to massive vacuum integrals

I have applied the ideas of section 4 and the program described in section 5 to the master integrals of fully massive vacuum integrals with a single mass scale up to the 5-loop level. The (partial) difference equations found have been compared with those J. Möller derived in his dissertation [12] where the latter were available. In this section, I will discuss these results along with intermediate results such as number or distribution of syzygies, the impacts of the user-defined parameters, runtimes and memory requirements as well as the limitations of the current implementation in *SPADES*.

6.1 Topologies and master integrals

There are many possible choices one can make for the auxiliary topologies and bases of master integrals. For this thesis, I will adopt the choices J. Möller made in his dissertation [12] to simplify the process of comparing results. All propagators D_i of the auxiliary topologies A_1,..., A_5 will be of the form $D_i = d_i^2 + m^2$ using a single mass scale, where the d_i are given in table 6.1.

	A_1	A_2	A_3	A_4	A_5
1	k_1	k_1	k_1	k_1	k_1
2		k_2	k_2	k_2	k_2
3		$k_1 - k_2$	k_3	k_3	k_3
4			$k_1 - k_2$	k_4	k_4
5			$k_1 - k_3$	$k_1 - k_4$	k_5
6			$k_2 - k_3$	$k_2 - k_4$	$k_1 - k_3$
7				$k_3 - k_4$	$k_1 - k_4$
8				$k_1 - k_2$	$k_1 - k_5$
9				$k_1 - k_3$	$k_2 - k_3$
10				$k_1 - k_2 - k_3$	$k_2 - k_4$
11					$k_2 - k_5$
12					$k_3 - k_5$
13					$k_4 - k_5$
14					$k_1 + k_2 - k_4$
15					$k_3 - k_4$

Table 6.1: Choices for the momenta of the of the auxiliary topologies A_n for $n = 1, \dots, 5$ loops. Displayed are the d_i in $D_i = d_i^2 + m^2$.

Since the auxiliary topology for n loops consists of $\frac{n(n+1)}{2}$ propagators which can have either positive or non-positive exponents, the auxiliary topologies A_1,...,A_5 give rise to 2, 8, 64, 1024 and 32768 sectors in total, respectively. As described in section 2, they can be grouped into zero sectors, antisectors and physical sectors, of which only the latter need to be considered for difference equations. The number of physical sectors is typically greater than the number of topologies, which is why I will choose only one sector per topology as a representative and map all other sectors in the same topology to its representative using sector shifts. The diagrams corresponding to the topologies are listed in Figures A.1 and

A.2. Table 6.2 gives an overview of the numbers of topologies and different types of sectors. The lists of representative sectors for the topologies can be found in Tables A.1 and A.2 for 1-4 loops and 5 loops respectively. Included there are also the numbers of sectors in the topologies and possible sector symmetry transformations for the representatives.

Loops	Topologies	Physical sectors	Trivial zero sectors	Non-trivial zero sectors	Trivial antisectors	Non-trivial antisectors
1	1	1	1	0	0	0
2	2	4	4	0	0	0
3	5	38	22	4	0	0
4	16	680	176	105	1	62
5	67	22051	1941	3625	121	5030

Table 6.2: Numbers of topologies and different types of sectors for the auxiliary topologies $A_1, \ldots, A_5$.

Starting at the 2-loop level, one encounters so-called factorised topologies, whose corresponding graphs consist of two or more disconnected graphs with fewer loops. Integrals without negative exponents from these topologies factorise into a product of integrals with fewer loops n_i, where $\sum_i n_i = n$. The difference equations for the factorised integrals will thus have the same forms as the equivalent ones found at lower loop levels for the individual factors. The numbers of topologies given above are composed of

- *1-loop*: $1 = 1$ non-factorised topology + 0 factorised topologies
- *2-loop*: $2 = 1$ non-factorised topology + 1 factorised topology (1-loop squared) $\equiv 1\{2\} + 1\{1^2\}$
- *3-loop*: $5 = 3\{3\} + 1\{1^3\} + 1\{1 \cdot 2\}$
- *4-loop*: $16 = 10\{4\} + 1\{1^4\} + 1\{1^2 \cdot 2\} + 1\{2^2\} + 3\{1 \cdot 3\}$
- *5-loop*: $67 = 48\{5\} + 1\{1^5\} + 1\{1^3 \cdot 2\} + 1\{1 \cdot 2^2\} + 3\{1^2 \cdot 3\} + 3\{2 \cdot 3\} + 10\{1 \cdot 4\}$

In principle, it would be sufficient to generate via the Laporta algorithm difference equations only for master integrals of the (1 + 1 + 3 + 10 + 48) non-factorised topologies and extrapolate from them all remaining ones. I will however also include the difference equations for factorised topologies in the calculation, as this provides a simple means of cross-checking some of the results. There are a total of 1, 2, 5, 19 and 131 master integrals at 1-5 loops respectively [12], which are listed in Tables A.3, A.4 and A.5.

For 1-3 loops there is exactly one master integral per topology, which is the integral with $a_i = 1$ for all positive exponents of the representative sector of that topology and $a_i = 0$ elsewhere. In the following, I will refer to such integrals as *binary integrals*. At 4 loops, one also has one binary master integral for each topology, but 3 of the topologies (with representative sectors 841, 1009 and 1011) additionally require one master integral each with one exponent greater than 1. In general, different master integrals will have different difference equations, however, if two master integrals differ only in one exponent (>0), the two difference equations for that propagator position will be identical, since the different exponents will be replaced by the same symbolic exponent z. In practice, this means that after calculating the difference equations for all binary master integrals, one already has one (but not necessarily all) difference equation for each integral with at most one exponent greater than one. Since one difference equation is sufficient to evaluate the master integral numerically via factorial series, I will not generate the difference equations for the remaining positions x for those integrals. At 5 loops one has a total of 131 master integrals [12], 67 of which are binary and 64 master integrals with at least one $a_i > 1$. Of these 64, 50 only contain one exponent $a_i > 1$ and are therefore covered by the difference equations of the binary master integrals, but the remaining 14 differ in two exponents from the respective binary integrals. There are two possible ways of dealing with these integrals, the more obvious of which is simply generating the difference equations for them in addition to those of the binary master integrals. J. Möller pointed out [12] that it is also possible to instead choose a slightly larger master integral basis[3], in which these 14 integrals are replaced by 16 additional integrals, which contain only one exponent greater than one. In this way it is possible to cover the complete set of master integrals with only the difference equations of the 67 binary master integrals. In the following, I will therefore only consider the difference equations for the 1, 2, 5, 16 and 67 binary master integrals.

It was already mentioned in section 3 that for a given master integral the difference equations for the different propagator positions x are not all independent, since the sector symmetries may allow shifting the symbolic exponent from x to different positions. The difference equations for all positions that are connected in this way will have the same form and therefore I will only consider the one for the leftmost (lowest) position of each such set of positions. As a consequence the number of difference equations per master integral depends not only on the number of propagators t, but also on the number of sector symmetry transformations. The total numbers of independent difference equations for the binary master integrals are 1, 2, 7, 33 and 234 for 1-5 loops respectively.

[3]Strictly speaking the set of master integrals is at this point no longer a basis, since its elements are linearly dependent.

6.2 Results

The difference equations for fully massive vacuum integrals with a single mass scale up to 4 loops are already known in the literature (see e.g. [34] or [12]). Since I have written *SPADES* specifically for this thesis and it thus is completely untested otherwise, reproducing these results and cross-checking them with the literature is an important first test of the code's functionality. I have therefore implemented an automated comparison of the generated difference equations with those found by J. Möller [12], who used a very different approach based on Tarasov's space-time dimensional relations and a completely independent program for obtaining the equations. All 1+2+7+33 difference equations up to 4 loops were generated successfully and the results are in perfect agreement with those of J. Möller. The orders R of the equations are summarised in Table 6.3. One can see from the table that the simple 1-loop difference equation with order $R = 1$ derived in Eq. (3.3) seems to be an exception, as all other difference equations with order 1 occur only for factorised topologies where the 1-loop case is one of the factors. At 2 and 3 loops the maximum order is 2, while at 4 loops, R goes as high as 5 for the difference equation 993#1 (where the notation is [Sector ID]#[Position x]).

1-loop				
#	t	ID	Position x	Order R
1	1	1	1	1

2-loop				
#	t	ID	Position x	Order R
1	2	6*	1	1
2	3	7	1	2

3-loop				
#	t	ID	Position x	Order R
1	3	56*	1	1
2	4	60*	1,3	2,1
3	4	51	1	2
4	5	62	1,2	2,2
5	6	63	1	2

4-loop				
#	t	ID	Position x	Order R
1	4	960*	1	1
2	5	992*	1,2	2,1
3	5	961*	1,4	2,1
4	5	841	1	4
5	6	1008*	1,3,4	2,1,2
6	6	993	1,2,4	5,2,2
7	6	978*	1	2
8	6	952	1	2
9	7	1016	1,4	2,2
10	7	1012*	1,3	2,1
11	7	1010	1,2,5	2,2,2
12	7	1009	1,3,4	3,3,3
13	8	1020	1,3,4,8	2,2,3,2
14	8	1011	1,3	4,3
15	9	1022	1,2	3,2
16	9	511	2	3

Table 6.3: Orders R of the difference equations found for 1-4 loops. Sector IDs with a * denote factorised topologies.

The difference equations at 5 loops were first studied in the dissertation of J. Möller [12]. He generated 117 of the possible 234 difference equations and determined orders of further 61 equations by setting all integrals on the right hand side in Eq. (3.1) to zero. Using *SPADES* I was able to, for the first time, determine orders for all 234 difference equations at 5 loops, which are summarised in Table 6.4. It should be noted that the values in that table are to be understood as upper limits for the orders R, as it is possible that difference equations with lower orders will show up, if one chooses larger systems of equations. At the time of writing this, 57 difference equations have been fully determined including subsectors, 8 of which were previously unknown. These numbers could be higher if not for two limiting factors which will be discussed in section 6.3. Due to those same limiting factors, 4 of the 57 equations where the right hand side was determined have a higher order than obtained in the case where subsectors were set to zero. For the remaining 177 equations, the left hand sides of Eq. (3.1) have been generated, with the exception of 3 difference equations (30239#3, 30526#1, 30526#7). So far, *SPADES* was only able to determine the order of these equations by replacing the space-time dimension d with integer numbers. In each case, the calculation was repeated with several different numeric choices (11, 12, 13, 1987) for d, with difference equations of the same order emerging at the same points in the calculations. It is therefore reasonable to assume that these orders are not merely the result of some coefficient(s) vanishing by coincidence for a numeric d.

All results I obtained for the 117 difference equations J. Möller determined fully have been cross-checked with the latter and are in perfect agreement with the exception of difference equation 30858#1. For this equation he states an order $R = 5$, while *SPADES* generated an equation of order 4. The equations have been checked to be compatible in the sense that the lower order equation reduces the higher order one to $0 = 0$ if used repeatedly to remove the integral with the greatest symbolic exponent. This difference also affects the results for difference equations in which the corresponding integrals appear on the right hand side. Of the additional 61 orders Möller determined I was able to confirm 59. In one case (30862#1) I found a lower order difference equation than he did (3 instead of 4), while for equation 32745#3 *SPADES* only generated a difference equation of order 4, where he found $R = 3$. In both cases the equations are compatible. The fact that *SPADES* was able to confirm almost all of the orders found suggests that the amount of information lost due to using the GKK approach instead of the conventional IBP approach is very small. In fact, it is not clear that any information is lost at all, since it is entirely possible that *SPADES* could still find the difference equation of order 3 for 32745#3, if the set of seed integrals is expanded or a higher value for Δ is chosen.

In general, the orders of the difference equations at 5 loops are much higher than at lower loop levels, with the highest value being 20. One can observe a strong correlation between the order of difference equations and the number of master integrals in a sector. All sectors with a difference equation of order 8 or higher have at least 2 master integrals, for $R \geq 12$ they show a minimum of 5 master integrals.

#	t	ID	Position x	Order R
1	5	31744*	1	$\underline{1}$
2	6	32256*	1,2	$\underline{2}$,$\underline{1}$
3	6	31746*	1,3	$\underline{2}$,$\underline{1}$
4	6	29702*	1,3	$\underline{4}$,$\underline{1}$
5	6	28686	1	$\underline{4}$
6	7	32512*	1,2,3	$\underline{2}$,$\underline{1}$,$\underline{2}$
7	7	32288*	1,5	$\underline{2}$,$\underline{1}$
8	7	32258*	1,2,3,5	$\underline{5}$,$\underline{2}$,$\underline{2}$,$\underline{1}$
9	7	31754*	1,3	$\underline{2}$,$\underline{2}$,
10	7	30872*	1,4	$\underline{2}$,$\underline{1}$
11	7	30858	1,2	$\underline{4}$,$\underline{2}$
12	7	30214	1,2,3	$\underline{7}$,$\underline{4}$,$\underline{2}$
13	7	29703	1,3	$\underline{7}$,8($\underline{9}$)
14	8	32640*	1,2,3	$\underline{2}$,$\underline{1}$,$\underline{2}$
15	8	32576*	1,2,5,6	$\underline{2}$,$\underline{2}$,$\underline{1}$,$\underline{2}$
16	8	32528*	1,2,3	$\underline{2}$,$\underline{2}$,$\underline{2}$
17	8	32513*	1,2	$\underline{2}$,$\underline{1}$
18	8	32386	1,2,3	$\underline{5}$,$\underline{2}$,2
19	8	32274	1,3,4	$\underline{5}$,2($\underline{4}$),2($\underline{4}$)
20	8	32266	1,2,3,5,6	$\underline{5}$,2($\underline{4}$),2,2,2
21	8	32259*	1,2,3,5	$\underline{3}$,$\underline{3}$,$\underline{3}$,$\underline{1}$
22	8	31380	1,2,3,8	$\underline{7}$,2,2,$\underline{2}$
23	8	31246	1,2,4,6	11,9,8,$\underline{6}$
24	8	30876	1	$\underline{2}$
25	8	30862	1,2	$\underline{3}$,$\underline{3}$
26	8	30222	1,2,3	8,7,8
27	9	32704	1,2,3,4,6	2,2,2,2,2
28	9	32648*	1,2,3,4,12	3,1,2,2,2
29	9	32608*	1,5,6	4,1,3
30	9	32592	1,3,4,6	2,2,2,2
31	9	32529*	1,2	2,2
32	9	32518	1,2	2,2
33	9	32394	1,2,3,12	5,2,2,2
34	9	32390	1,2,3,4,5,8,13	7,3,2,3,3,3,3
35	9	32329	1,3	2,2
36	9	32278	1,2,3,5,14	8,3,2,3,3
37	9	32270	1,2,3,6,12	12,7,11,7,7
38	9	32267	1,2,3,5	3,3,4,2
39	9	31516	1,2,3,6,12	6,6,11,5,5
40	9	31388	1,2,6	3,4,3
41	9	30231	1,13	20, 11
42	10	32736	1,2,3,5,6,9	6,4,4,2,3,3
43	10	32712	1,2,5,6,8	3,2,2,2,2
44	10	32708	1,2,3,4,6,13	3,2,2,2,2,2
45	10	32674	1,2,3,4	3,2,2,2
46	10	32652*	1,2,3	3,1,2
47	10	32596	1,6	11,7
48	10	32562	1,3,4	3,2,2
49	10	32534	1,2,3,4,11,13	6,4,2,4,3,3
50	10	32398	1,2,3,4,5,6	8,3,2,3,6,2
51	10	32391	1,2,3,6	6,4,3,3
52	10	32279	1,3,4,6,13,14	16,11,18,9,9,6
53	10	31420	1,6	3,3
54	10	30563*	1,5	3,1
55	10	30239	1,3,14	19,18*,6
56	10	29550	1,6,12	10,10,7
57	11	32744	1,2,5,6,7,8,10	11,4,2,9,3,2,3
58	11	32737	1,2,3,5,6,9,15	3,2,3,2,2,2,3
59	11	32713	1,2,3,4,7	3,2,2,2,2
60	11	32682	1,2,3	2,2,2
61	11	31736	1,3,4,7	10,3,3,7
62	11	30691	1,2,3,5	6,3,3,2
63	11	30526	1,2,3,6,7	17*,16,11,9,16*
64	12	32745	1,2,3,7,8	3,2,4,2,2
65	12	31740	1	6
66	12	30699	1,2,5	7,3,2
67	12	30527	1,6	18,10

Table 6.4: Orders R of difference equations for 5 loops. Underlined orders signify that the complete difference equation including subsectors was generated. Orders with a * have only been determined using a numeric value for d, while sector IDs with a * denote factorised topologies. The notation $R(\underline{R'})$ indicates that a left hand side of a difference equation with order R is known, but the full difference equation including subsectors has only been generated with order R'.

In the following, I will discuss some details of the calculations that lead to the above results. Since the central point of the approach in this thesis are the syzygies that guarantee that certain exponents will not be raised, I have collected some numbers of syzygies found at different loop levels n, numbers of propagators t and degrees Δ in Table 6.5. The values found there are taken from the calculation of specific difference equations, but are typical for calculations with the same parameters n and t.

Loops	t	Difference equation	$\Delta = 0$	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$
2	2	6#1	2	3	0	0	0
2	3	7#1	0	6	0	0	0
3	3	56#1	3	12	0	0	0
3	4	51#1	0	18	0	0	0
3	6	63#1	0	3	22	13	0
4	4	960#1	4	30	0	0	0
4	5	841#1	0	40	0	0	0
4	7	1009#1	0	7	83	24	0
4	9	511#2	0	6	1	197	13
5	5	31744#1	5	60	0	0	0
5	6	28686#1	0	75	0	0	0
5	7	29703#3	0	29	230	6	0
5	8	30222#1	0	14	225	135	0
5	9	32390#4	0	11	160	132	29
5	10	29550#1	0	10	43	527	62
5	11	32737#5	0	25	0	138	613
5	12	30527#1	0	10	0	108	1193

Table 6.5: Typical numbers of syzygies for various loop levels, numbers of propagators t and degrees Δ.

As one can see, the only topologies with syzygies at degree $\Delta = 0$ are those which factorise into at least one 1-loop topology. In general, the number of syzygies increases with the number of loops n and different propagators t. Furthermore, the distribution is shifted to higher degrees for greater t. This is expected, as more propagators that must not be raised put more constraints on the syzygies, thus allowing less syzygies to be found at lower degrees. A general rule of thumb seems to be that the peak of the syzygy distribution starts at $\Delta = 1$ for $t = n$ and moves up one degree roughly every two steps of t. If this trend holds at the 6-loop level, one would expect the syzygy distributions for the most difficult topologies to peak at $\Delta = 5$, which would be beyond the current limits of *SPADES* as will be discussed in section 6.3. So far calculating syzygies up to the peak of the distribution was sufficient in each case to generate a difference equation. However, in some cases including syzygies of higher degrees leads to a difference equation of lower order. It is thus quite possible that some of the difference equations at 5 loops with very high R turn out to actually have a lower order and the difference equation of order 3 for 32745#3 is found once $\Delta = 5$ will be accessible. One last interesting fact to notice from Table 6.5 is that it is possible to have gaps in the syzygy distribution at degree Δ, with syzygies found at both $\Delta - 1$ and $\Delta + 1$. This shows that it is in general not advisable to use the non-existence of syzygies at a given degree as a criterion for terminating the syzygy finding algorithm prior to the user-defined Δ_{max}.

The inclusion of the extensions to the syzygy algorithm discussed in section 4.3 opens up possibilities of finding additional syzygies, which at first lead to raised propagators which are immediately cancelled by symmetry relations or the vanishing of the integral in dimensional regularisation. As expected, for many degrees Δ this leads to a higher number of syzygies being found. However, in all cases where this was studied, the additional syzygies seem to add no new information to the system of equations. This is a curious result, as it might suggest that the information of symmetry relations and dimensional regularisation is somehow already contained in the system of equations without being entered manually or the extensions being used, even though as shown in section 4 no reason is known why this should in general be the case for a finite set of seed integrals. Future investigation will have to determine whether this result is a coincidence, a property of vacuum diagrams or holds true in general. Even though the extensions do not seem to provide additional information, they might still be helpful in reducing the amount of redundant information. In many cases they actually reduce the overall number of syzygies found. This is due to the fact that the additional syzygies found at say $\Delta = 1$ are mapped to the higher degrees via subroutine 1 and may remove degrees of freedom that would have led to syzygies at that degree. Since every syzygy is mapped with every x_i allowed and can thus remove multiple degrees of freedom, the total number of syzygies can be reduced in this way. Some examples of this for the symmetry extension as well as cases where the extension raises the number of syzygies found are listed in Table 6.6.

Difference equation	Symmetry extension	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	Sum
30222#1	no	14	225	135	374
30222#1	yes	21	233	105	359
30876#1	no	26	127	73	226
30876#1	yes	52	29	73	154
30858#1	no	20	330	0	350
30858#1	yes	46	108	23	177
28686#1	no	75	0	0	75
28686#1	yes	98	0	0	98
29703#3	no	29	230	6	265
29703#3	yes	47	206	64	317

Table 6.6: Comparison of the numbers of syzygies found for selected 5-loop calculations with and without the symmetry extension enabled.

An interesting aspect of the results of the Laporta algorithm is the number of equations that are actually needed to generate the difference equations. Since *SPADES* already records the steps of the Laporta algorithm, the number of equations out of the whole system that were combined to eventually form the difference equation is easily obtained. Unfortunately, I have no such data available from calculations with conventional IBP approaches for comparison, but I expect that this number is considerably smaller in *SPADES* due the fact that all steps concerning raised exponents are already taken care of in the syzygy part of the program. For the non-factorisable 5-loop topologies the number of equations needed for the difference equation ranges from 10 to 500, with the majority of topologies between 50 and 250. This does not include equations for integrals of subsectors, which are assumed to have been generated in previous calculations. However, one usually wants to generate more than these 10-500 equations to solve as many integrals with negative exponents as possible for use in calculations where the current sector appears as a subsector itself. I have found that a parameter $s_{max} = 4$ for the set of seed integrals (see Eq. (5.6)) seems to be a good choice at 5 loops, which depending on the topology and number of syzygies typically generates between 10^4 and 10^5 equations in total, of which $70 - 95\%$ are completely redundant. Some ideas to reduce the number of redundant equations are given in section 7.

6.3 Runtimes and limitations

All calculations have been performed on machines with Intel® Xeon® X5660 processors (2.80 GHz, 12 MB Cache) and 48 GB of RAM. *SPADES* is currently limited to using a single CPU core. In this section, I will give an overview of runtimes and limitations of the different parts of the program, starting with the syzygy finding algorithm and then moving on to the generation of difference equations via Laporta's algorithm.

The implementation of the syzygy finding algorithm as described in section 5.1 currently has two bottlenecks, the row reduction of the B_Δ performed by the method `lSolveSparse` and the generation of the nullspace of the matrix Q in subroutine 2. All other parts of the algorithm usually account for only a negligible fraction of the overall runtime. Table 6.7 lists some runtimes for these two parts of the algorithm.

		Difference	$\Delta = 2$		$\Delta = 3$		$\Delta = 4$	
Loops	t	equation	B_Δ	Subr. 2	B_Δ	Subr. 2	B_Δ	Subr. 2
4	4	960#1	< 1s	< 1s	< 1s	< 1s	< 1s	< 1s
4	5	841#1	< 1s	< 1s	< 1s	< 1s	< 1s	1.9s
4	7	1009#1	< 1s	< 1s	3.1s	2.0s	138s	8.1s
4	9	511#2	< 1s	1.1s	< 1s	9.7s	214s	9.9s
5	5	31744#1	< 1s	< 1s	< 1s	1.7s	< 1s	24s
5	6	28686#1	< 1s	< 1s	< 1s	3.7s	8.3s	39s
5	7	29703#3	< 1s	1.3s	61s	6.2s	6430s	112s
5	8	30222#1	< 1s	1.6s	79s	22s	41300s	447s
5	9	32390#4	< 1s	1.5s	38s	21s	22200s	486s
5	10	29550#1	< 1s	2.4s	< 1s	96s	82900s	2250s
5	11	32737#5	< 1s	1.2s	< 1s	26s	31s	822s
5	12	30527#1	< 1s	2.8s	< 1s	248s	44s	23500s

Table 6.7: Runtimes for `lSolveSparse(`B_Δ`)` and subroutine 2 of the syzygy finding algorithm. The numbers of syzygies found in the calculations are listed in Table 6.5. All runtimes for $n \leq 3$ and/or $\Delta \leq 1$ are omitted due to being negligibly small.

As one can see, the runtime is heavily dependent on the number of loops and the degree Δ, since they determine the size of the matrices B_Δ and Q. Runtimes of `lSolveSparse` with typical matrix sizes were already given in Table 5.1. For large matrices the elements in the intermediate steps of the Gauss algorithms can become very large objects, which considerably slows down the progress. Furthermore, this leads to very high memory requirements during the calculations, with a single matrix at 5 loops and $\Delta = 4$ occupying in some cases as much as 30 GBs of RAM (for a matrix size of approximately $4 \cdot 10^4 \times 1.3 \cdot 10^5$). It is therefore obvious that $\Delta = 5$ at 5 loops is not within reach for the current implementation. Several ideas to improve the algorithm in the future are listed in section 7.

For the Laporta part of *SPADES* I will divide the runtimes into four pieces. The first time comprises all preparations for the Laporta algorithm. This includes generating the blueprint equations from the syzygies, inserting the values for the seed integrals, applying sector shifts and symmetries to all integrals and sorting their coefficients into the matrix for the Laporta algorithm. The second time given is the runtime of the Laporta algorithm for a numeric space-time dimension d. I have chosen $d = 12$ for all runtimes listed here, which only on one occasion out of all calculations done at 5 loops resulted in a coefficient which was 0 for the numeric, but not the general d. The third time refers to the same Laporta algorithm with a symbolic d, where all equations known to be redundant from the first Laporta algorithm are omitted from the beginning. The fourth and last value given is the time for determining the right hand side of the difference equation, which contains all integrals of subsectors of the current sector. This includes retracing and executing the steps from the Laporta algorithm for the subsector integrals as well as applying the previously generated difference equations of those sectors to reduce the number of integrals to a minimum. Several user-defined parameters which were introduced in section 5.2 can affect the runtime considerably: s_{max} is the maximum number of negative powers distributed in the seed integrals, δz_{max} is the number of times an equation can be shifted by 1 in the symbolic exponent and σ_{min} is the minimum number of negative powers in the sector of the master integral up to which equations have to be evaluated by the Laporta algorithm. Several runtimes for difference equations that have been generated including the subsectors are listed in Table 6.8.

n	t	Difference equation	s_{max}	δz_{max}	σ_{min}	# eqns.	Preparation	Laporta $d = 12$	Laporta symb. d	Subsectors
4	4	960#1	3	3	5	4688	3s	2.0m	9s	0s
4	6	993#1	4	5	4	5821	3.2m	20s	2s	5.2m
4	8	1020#4	3	5	3	1502	1.9m	2s	<1s	5s
4	9	511#2	3	5	3	1551	2.2m	3.3m	<1s	5.3m
5	5	31744#1	4	4	5	47861	1.8m	23m	18s	0s
5	6	28686#1	4	4	4	36498	30m	3.1m	50s	7.5h
5	7	29703#1	3	4	4	8300	6.6m	3.3m	18.3m	24m
5	7	30858#1	4	4	5	37660	47m	1.2h	2.7h	16h
5	8	32266#1	3	4	4	7528	13m	2.5m	26s	18m
5	8	31380#1	3	4	4	6400	18m	7.4m	3.6m	1.6d
5	8	31246#6	3	4	3	8210	29m	44s	19s	15h

Table 6.8: Several runtimes for the Laporta part of *SPADES*. s = seconds, m = minutes, h = hours, d = days.

From Table 6.8 one can learn several things:

- The runtimes for two topologies with similar parameters and sizes of the system of equations can be vastly different. This makes it very difficult to predict even the order of magnitude of the runtime.
- Some of the steps before the Laporta algorithm are still far from being optimised. This is especially true for the application of the sector symmetries.
- The calculations for the subsectors take up the bulk of the runtime in most cases where there are any subsectors ($t > n$). This gets progressively worse for higher t, since there are more subsectors to consider. The problem could be reduced by utilising multiple processors as discussed in section 7.
- The Laporta algorithm with a numeric d has the potential to greatly reduce the runtime of the actual Laporta algorithm one wants to perform by removing redundant equations. As an example, in the calculation for difference equation 32266#1 the first Laporta algorithm reduces the size of the system of equations from 7528 to 451 equations. As seen in Table 6.8 the combined runtime of both Laporta algorithms is approximately 3 minutes. If one skips the first Laporta algorithm and does the full calculation with a symbolic d, this value increases to 10 minutes.

There are currently two major limitations which prevent the generation of all difference equations at the 5-loop level. The first is the time it takes to calculate the subsector part of the equations, which as seen in Table 6.8 grows rapidly with the number of propagators t. Some suggestions that might reduce that time are given in section 7. The second problem is the current implementation of the sector symmetries. As described in section 5.2, a database is created which holds the symmetrised versions of all integrals that appear in the calculation. If the integral to be symmetrised has any negative exponents, the corresponding propagators might be shifted to linear combinations of propagators. For integrals with large s, the symmetrised expression can thus contain a sum of many integrals, all with potentially large coefficients. Additionally, the number of possible integrals grows with s. Currently the database is held in the RAM, but already at $t = 8$ for 5 loops its memory requirements grow too large, thus limiting the parameter s_{max} which governs the size of the set of seed integrals. This means that too few equations for integrals with negative exponents are generated for a given sector ID, which would be needed to replace these integrals in calculations for sectors that have ID as a subsector. This problem can however be avoided in the future by switching to an external database, which would additionally reduce the time for applying the sector symmetries, since integrals from subsectors would already have been symmetrised in previous calculations. The memory requirements of the Laporta algorithm itself have not posed a problem so far, since they remained below 1.5GB in almost all cases.

In Table 6.9 I list several runtimes for the difference equations with $t \geq 9$, where I set all integrals of subsectors to zero. In this way both of the above mentioned problems are avoided, since the calculation is completely finished after the Laporta algorithm and the expressions for the symmetrised integrals are much shorter.

Table 6.9 shows even wider gaps between the runtimes for different topologies than Table 6.8. I believe the extremely long runtime for the difference equations with very high

n	t	Difference equation	R	s_{max}	δz_{max}	σ_{min}	# eqns.	Prepa-ration	Laporta $d = 12$	Laporta symb. d
5	9	32704#1	2	2	4	1	183	6s	1s	<1s
5	9	32390#1	7	1	4	3	1344	6.0m	1.6m	17s
5	9	32270#1	12	2	8	4	4984	1.8m	2.9h	8.5h
5	9	30231#1	20	1	20	4	2898	13s	2.0h	31d
5	10	32712#2	2	1	4	1	108	2s	<1s	<1s
5	10	32279#6	9	1	4	5	1648	57s	54m	47m
5	10	30239#1	19	2	15	5	3240	13s	43m	8.2d
5	11	32737#1	3	2	8	4	1982	7s	11s	1s
5	11	30526#2	16	2	20	5	7054	1.6m	2.0h	13.3d
5	12	31740#1	6	1	5	5	617	35s	56s	23s
5	12	30527#1	18	2	15	5	2568	45s	6.8m	13d

Table 6.9: Several runtimes for the Laporta part of *SPADES* with subsector integrals set to zero. s = seconds, m = minutes, h = hours, d = days.

orders to be due to the current implementation of shifts in the symbolic exponent z. By shifting z by 1, evaluating the new equations, then shifting it again and repeating this process, in some constellations very large coefficients can appear in the intermediate steps of the Laporta algorithm. During the calculation of equation 30231#1, the *Fermat* executable needed roughly 600MB of memory for a single coefficient, where this value is normally around 10-50MB. The problem is most apparent for the difference equations with high orders, since they could only be found using higher values for δz_{max}. A different approach to z-shifts that would likely solve this problem is suggested in section 7. The above mentioned large coefficients caused the corresponding calculations to be the only ones so far where the matrix of the Laporta algorithm needed more than 1.5GB of RAM. While the memory requirements would probably be reduced by implementing better z-shift behaviour, they might still prove to be problematic once the full difference equation is considered, since e.g. already the left hand side of the difference equation 30231#1 results in a 63MB output file when saved as a string with no compression.

Apart from several specific problems which were identified in this section, the overall runtimes of *SPADES* are encouraging. In many cases, the Laporta algorithm at the 5-loop level only takes a few minutes or even seconds, which seems difficult to achieve without the use of syzygies [12]. Considering that *SPADES* is not yet optimised in many regards, I believe a full set of 5-loop difference equations for the massive vacuum integrals is within reach once the problems listed here are overcome and the performance is boosted by implementing some of the ideas presented in section 7.

7 Possible improvements

There are several ideas for improving performance and functionality of *SPADES* that I was not able to implement within the time constraints of a master's thesis, but hope to test in the future. One of the most obvious improvements, albeit non-trivial to implement, is the parallelisation of the code. Currently *SPADES* runs on a single processor, but many parts of the calculation are well suited to be distributed among multiple processors. This is especially true for the part which takes up the bulk of computation time for most difference equations, which is the determination of the right hand side of the equation after the Laporta algorithm. Since all the necessary steps are already determined and recorded in the Laporta algorithm, there is absolutely no interdependence between the manipulations of the coefficients of different integrals at this point, thus making it an ideal candidate for parallelisation. The prospect is especially appealing because the calculations which currently take the most time are those with many subsectors involved, which in turn allows more processors to be used efficiently in parallel due to the higher number of different integrals.

In the Laporta algorithm itself, I would like to change the way the shifts in the symbolic exponent z are handled. Currently, as described in section 5.2, a batch of equations is solved as much as possible via the Laporta algorithm, then the shift $z \to z+1$ is applied to the ones that still contain integrals of types I to III and this process is repeated δz_{max} times. While many results found with the current implementation rely on that shift, it also leads to numerous equations which have the same form as the ones they originated from, only with a shifted z, but are never actually used to cancel integrals from other equations. Furthermore, as discussed in section 6.3, this can also lead to very large coefficients in the intermediate steps, thus slowing down the Laporta algorithm. A more elegant solution would be to group all integrals which only differ in the symbolic exponent together and only allow one equation in the system to have an integral from that (pivot) group as its most difficult integral. The equation could then be treated in the same way as a difference equation, with the order R being the difference between the greatest and the lowest symbolic exponent in the pivot group of integrals. If a second equation with the same pivot group should appear, the equation with the lower order has to be repeatedly used (with the appropriate z-shift) to cancel the integral with the greatest symbolic exponent from the equation with the higher order, until only one equation with integrals in the pivot group remains. This implementation would reduce the number of redundant equations in the system as well as allow for unlimited z-shifts rather then having a maximum δz_{max}.

In the current implementation *SPADES* generates the system of equations by applying the syzygies first to a general seed integral with symbolic exponents a_i at all non-positive positions and only afterwards substituting numeric values for the remaining a_i to obtain the final equations. The intermediate equations containing the symbolic a_i can be seen as the blueprints for the whole system of equations. In principle, it should be possible to run a smaller, modified version of a Laporta algorithm already on this set of blueprint equations. A new ordering prescription for the difficulty of the integrals with the symbolic a_i would be needed, which upon substitution of (random) numeric values would have to resemble the prescription for the final integrals as closely as possible. Furthermore, the algorithm would have to be restricted to only divide or multiply by terms which are non-zero for any numeric choice for the exponents. This restriction would severely limit the amount of operations a Laporta algorithm could perform. Nevertheless, an attempt might be worthwhile, since any simplification of one of the blueprint equations translates into a simplification of many equations in the system upon substituting all distributions of numeric values for the exponents a_i.

One of the limiting factors of the current implementation for finding syzygies is the size of the matrices B_Δ for large Δ, if many syzygies have been found at previous degrees. Since the B_Δ are only needed to determine a set of pivot columns which in turn are used to prevent finding redundant syzygies, one might try to skip this step where B_Δ becomes too large for efficient calculations. Instead of mapping the full syzygies found at previous degrees to degree Δ, one could also map only the pivot column positions of $B_{\Delta-1}$ to the new degree. In doing so one risks ending up with less pivot positions at degree Δ than possible and thus finding redundant syzygies at that degree. While this approach completely eliminates the need to apply `lSolveSparse` to B_Δ, having less pivot positions would increase the number of elements in the matrix Q and thus the runtime of subroutine 2. This downside as well as the risk of redundant syzygies are good reasons not to abandon the complete matrices B_Δ altogether, but the approach might be used to extend the range of degrees Δ *SPADES* can cover where the B_Δ grow too large.

In section 3.1, I portrayed the idea of R.N. Lee [39] to reduce the number of redundant IBPs by taking advantage of the fact that the IBP operators $O_{ij} = \frac{\partial}{\partial k_i^\mu} q_j^\mu$ fulfil the commutator relation

$$[O_{ik}, O_{jl}] = d\left(\delta_{il} O_{jk} - \delta_{jk} O_{il}\right). \tag{7.1}$$

This allowed him to write a multiplicative basis for the IBP operators (see Eq. (3.23)) much smaller than the set of all O_{ij}. As described in section 3.1, the major problem with applying this to non-symbolic reductions is that some of the IBP operators could only be cast aside for seed integrals with (r,s) values up to $(r_{max}-n+1, s_{max}-n+1)$. However, a modified version of this approach applied to syzygy operators could avoid the better part of this problem. Given any (master) integral J, I will label a general syzygy operator for a syzygy $\alpha^{\Delta,k}$ of degree Δ for J as

$$U^{\Delta,k} \equiv \frac{\partial}{\partial k_i^\mu} q_j^\mu \alpha_{i,j}^{\Delta,k}, \quad U^{\Delta,k} \in U^\Delta, \tag{7.2}$$

where summation over repeated indices is implied and U^Δ is the set of all syzygy operators of degree Δ for J. The operators obey the relation

$$\begin{aligned} \left[U^{\Delta,k}, U^{\Delta',k'}\right] &= \frac{\partial}{\partial k_i^\mu} q_j^\mu \underbrace{\left\{\alpha_{a,j}^{\Delta,k}\left(\frac{\partial}{\partial k_a^\nu} q_b^\nu \alpha_{i,b}^{\Delta',k'}\right) - \alpha_{a,j}^{\Delta',k'}\left(\frac{\partial}{\partial k_a^\nu} q_b^\nu \alpha_{i,b}^{\Delta,k}\right)\right\}}_{\equiv \alpha_{i,j}^{\Delta+\Delta',k,k'}} \\ &\equiv U^{\Delta+\Delta',k,k'} \in U^{\Delta+\Delta'}. \end{aligned} \tag{7.3}$$

$U^{\Delta+\Delta',k,k'}$ is again a syzygy operator for the integral J because it has the correct form and also does not allow raised propagators because neither $U^{\Delta,k}$ nor $U^{\Delta',k'}$ do. As with the IBP operators, to ensure that the information gained by acting with $U^{\Delta+\Delta',k,k'}$ on an integral I is already in the system, one has to act with $U^{\Delta,k}$ on all integrals that result from acting on I with $U^{\Delta',k'}$ and vice versa. The important difference to the IBP case here is that $U^{\Delta,k}$ and $U^{\Delta',k'}$ do not raise any propagators[4]. If the integral I has r' and s values (r'_I, s_I), acting on it with $U^{\Delta,k}$ will thus only yield integrals with values ($r' \leq r'_I$, $s \leq s_I + \Delta + 1$). This means that seed integrals for syzygies with a lower degree Δ would have to allow for higher s-, but not r'-values than those for syzygies with higher degrees. This is a sensible step in any case, since it allows all syzygies to reach similar s-values of integrals in the final equations. This makes it possible to ensure that the information $U^{\Delta+\Delta',k,k'}$ would carry is already in the system of equations with very little additional effort. The syzygy algorithm could then avoid finding $U^{\Delta+\Delta',k,k'}$ by adding the appropriate row for $\alpha_{i,j}^{\Delta+\Delta',k,k'}$ to $B_{\Delta+\Delta'}$. This opens up a second way of "mapping up" syzygies to higher degrees in addition to the multiplication by the x_i performed by subroutine 1. However, unlike subroutine 1, the $\alpha_{i,j}^{\Delta+\Delta',k,k'}$ could potentially introduce the space-time dimension d as an additional variable into the B_Δ. The longer runtime for `lSolveSparse` this additional variable would cause can be avoided by substituting a numeric value for d (or in fact any variable) in the B_Δ. This does not produce wrong results for the syzygies, since one is only interested in the positions of the pivot columns of B_Δ, not the specific values of its elements. The only (negligible) drawback of substituting a numeric value for d is the chance that coefficients might vanish only for that value of d and thus reduce the number of pivot columns found, but for sensible numeric choices this chance should be very small.

One of the two time consuming parts of finding the syzygies is typically generating the basis of syzygy vectors that are orthogonal to P_Δ. Here each term in P_Δ contains exactly one of the dummy variables t_i which correspond to the propagators D_i whose exponents should not be raised for the sector ID. One could thus write P_Δ as

$$P_\Delta = \sum_{i \in C} t_i P_\Delta^{(i)} \tag{7.4}$$

and reformulate the task of subroutine 2 to find a basis for all syzygy vectors which are orthogonal to all $P_\Delta^{(i)}$. This task could be simplified by not starting the construction of a basis from scratch, but from the set of syzygy vectors for that degree already found for subsectors. Assuming a subsector ID_{sub} of ID is reached by removing the j-th propagator from the set of propagators with positive exponents, the syzygy vectors from that subsector are already orthogonal to all $P_\Delta^{(i)}$ for all $i \neq j$. If the subsector in question is additionally shifted to a different sector ID'_{sub} via a sector shift, the syzygy vectors would have been determined for ID'_{sub} and would need to be subjected to the inverse sector shift to fulfil this constraint. A part of the basis that subroutine 2 needs to find can then be constructed from those linear combinations of the syzygy vectors of the subsector which are orthogonal to $P_\Delta^{(j)}$ as well. This can be repeated for each $j \in C$, although it is likely that many of the additional syzygy vectors found in this way would be redundant due to linear dependence with those already found from the previously considered subsector(s). The procedure described here would not completely eliminate the need for subroutine 2 as currently implemented, since it is possible that the subroutine would find additional syzygies which

[4]Exceptions to this are the symbolic propagator and non-positive propagators. Neither case poses a problem since a raised symbolic propagator is covered by z-shifts and raised non-positive propagators can only lower s, but not raise r.

cannot be constructed from syzygies of subsectors of the same degree. While it is always possible to construct all syzygies of degree Δ for the sector ID from the syzygies of ID_{sub} if all syzygies of the latter are first mapped to degree Δ, this mapping could lead to factorisable syzygies for sector ID, which are unwanted. Subroutine 2 can thus find non-factorisable syzygies which are not constructible from syzygies of the same degree of subsectors. However, I believe that constructing syzygies from subsectors first can already remove many of the degrees of freedom in subroutine 2 and thus speed up calculation of the remaining syzygies considerably.

There are also several smaller improvements that have yet to be implemented, only two of which I will list here. The first is simply an additional requirement on subroutine 2 of the syzygy part to minimise the number of elements at the positions which correspond to the coefficients of the derivatives in the syzygy operators. Elements at these positions tend to lead to more integrals than the elements at the positions which correspond to the lower rows of the matrix E, due to the derivatives hitting the former but not the latter. This change could reduce the number of integrals in the blueprint equations and thus save steps in the Laporta algorithm. Another idea for a possible improvement stems from the fact that preliminary tests have shown that the current order of choosing equations in the Laporta algorithm might not be ideal. Instead of choosing the simplest remaining equation first, it can in some cases be beneficial to choose the equation with the smallest number of integrals. Future testing will have to find a good compromise between these two criteria.

8 Conclusion and outlook

In this thesis, I have studied the generation of so-called difference equations for fully massive vacuum Feynman integrals with a single mass scale. To this end, I employ the ideas of Gluza, Kajda and Kosower [1] to search for linear combinations of integration by parts operators that do not raise the exponents of a given integral they operate on. These operators can be expressed via vectors which consist of polynomials of scalar products of momenta and masses, the so-called syzygies. An algorithm for finding these syzygies, which is solely based on simple linear algebra, was proposed by Schabinger [6]. I have modified and extended this approach and implemented it in C++ as the first part of a program called *SPADES*. The second part of *SPADES* then translates the syzygies into a system of equations for the vacuum integrals which in turn is solved for a difference equation via a modified Laporta algorithm.

I have applied this setup to all master integrals of the fully massive vacuum integrals up to 5 loops. The difference equations up to 4 loops were already known in the literature and *SPADES* generated all of them correctly. At the 5-loop level I was able to, for the first time, determine a complete set of orders for all difference equations. Many of them I already generated fully and for almost all of them I could obtain the equation when I set all integrals of subsectors to zero. The results have been compared with those of J. Möller where the latter were available and are in agreement. The runtimes for obtaining most of these results are within reasonable limits, even though *SPADES* is not yet fully optimised and several problems with the current implementation were identified. This suggests that the approach taken in this thesis is a very viable choice over the conventional IBP method for generating difference equations. The advantage over the latter is the fact that a sizeable part of the work, removing integrals with raised propagators, is taken out of the Laporta algorithm and solved with the simple means of linear algebra for all integrals with arbitrary non-positive exponents at once.

In section 7, I listed several ideas to overcome the problems identified in section 6.3 and enhance both the performance and functionality of *SPADES*. I believe that with their implementation a complete set of difference equations for the 5-loop massive vacuum integrals is within reach. Once this is achieved, the equations will have to be used to numerically evaluate the master integrals via factorial series, which in itself will present a challenge due to the high orders of some of the difference equations found. Another logical step is to remove *SPADES*'s current limitation to vacuum integrals with a single mass scale. Both external momenta and multiple masses can be introduced via the parameters χ_i as described in section 4. This would open up a large number of integral classes, for many of which difference equations have yet to be found.

A Appendix

1-loop				
#	t	ID	Sectors	Symmetries
1	1	1	1	1

2-loop				
#	t	ID	Sectors	Symmetries
1	2	6*	3	4
2	3	7	1	6

3-loop				
#	t	ID	Sectors	Symmetries
1	3	56*	16	24
2	4	60*	12	12
3	4	51	3	24
4	5	62	6	8
5	6	63	1	24

4-loop				
#	t	ID	Sectors	Symmetries
1	4	960*	132	192
2	5	992*	141	48
3	5	961*	66	48
4	5	841	15	120
5	6	1008*	102	16
6	6	993	66	12
7	6	978*	12	144
8	6	952	12	48
9	7	1016	7	48
10	7	1012*	12	48
11	7	1010	48	8
12	7	1009	33	8
13	8	1020	15	8
14	8	1011	15	8
15	9	1022	3	12
16	9	511	1	72

Table A.1: Chosen representatives of the physical sectors for the auxiliary topologies A_1, A_2, A_3 and A_4, the number of sectors in the topology they belong to and the number of sector symmetry transformations (including the identity transformation) possible for the representatives. Sector IDs with a * denote factorised topologies.

#	t	ID	Sectors	Symmetries
1	5	31744*	1398	1920
2	6	32256*	2070	288
3	6	31746*	1141	192
4	6	29702*	430	240
5	6	28686	80	720
6	7	32512*	1885	64
7	7	32288*	448	288
8	7	32258*	1780	24
9	7	31754*	164	288
10	7	30872*	296	96
11	7	30858	212	72
12	7	30214	420	48
13	7	29703	228	48
14	8	32640*	234	96
15	8	32576*	1440	16
16	8	32528*	266	96
17	8	32513*	248	192
18	8	32386	162	48
19	8	32274	496	16
20	8	32266	680	12
21	8	32259*	766	16
22	8	31380	324	16
23	8	31246	428	8
24	8	30876	11	384
25	8	30862	94	32
26	8	30222	210	24
27	9	32704	256	16
28	9	32648*	550	16
29	9	32608*	336	16
30	9	32592	536	8
31	9	32529*	34	288
32	9	32518	84	48
33	9	32394	128	24
34	9	32390	544	4
35	9	32329	10	384
36	9	32278	272	8
37	9	32270	400	4
38	9	32267	140	16
39	9	31516	316	4
40	9	31388	54	32
41	9	30231	76	12
42	10	32736	236	4
43	10	32712	192	8
44	10	32708	196	8
45	10	32674	47	32
46	10	32652*	142	24
47	10	32596	76	10
48	10	32562	50	32
49	10	32534	232	4
50	10	32398	208	4
51	10	32391	88	8
52	10	32279	260	2
53	10	31420	21	32
54	10	30563*	10	144
55	10	30239	28	12
56	10	29550	24	16
57	11	32744	172	2
58	11	32737	144	4
59	11	32713	72	8
60	11	32682	17	32
61	11	31736	68	4
62	11	30691	16	16
63	11	30526	40	4
64	12	32745	50	4
65	12	31740	3	48
66	12	30699	8	12
67	12	30527	4	16

Table A.2: Chosen representatives of the physical sectors for the auxiliary topology A_5, the number of sectors in the topology they belong to and the number of sector symmetry transformations (including the identity transformation) possible for the representatives. Sector IDs with a * denote factorised topologies.

1-loop			
#	t	ID	Master Integral
1	1	1	J_1

2-loop			
#	t	ID	Master Integral
1	2	6	$J_{1,1,0}$
2	3	7	$J_{1,1,1}$

3-loop			
#	t	ID	Master Integral
1	3	56	$J_{1,1,1,0,0,0}$
2	4	60	$J_{1,1,1,1,0,0}$
3	4	51	$J_{1,1,0,0,1,1}$
4	5	62	$J_{1,1,1,1,1,0}$
5	6	63	$J_{1,1,1,1,1,1}$

4-loop			
#	t	ID	Master Integral
1	4	960	$J_{1,1,1,1,0,0,0,0,0,0}$
2	5	992	$J_{1,1,1,1,1,0,0,0,0,0}$
3	5	961	$J_{1,1,1,1,0,0,0,0,0,1}$
4	5	841	$J_{1,1,0,1,0,0,1,0,0,1}$
5	6	1008	$J_{1,1,1,1,1,1,0,0,0,0}$
6	6	993	$J_{1,1,1,1,1,0,0,0,0,1}$
7	6	978	$J_{1,1,1,1,0,1,0,0,1,0}$
8	6	952	$J_{1,1,1,0,1,1,1,0,0,0}$
9	7	1016	$J_{1,1,1,1,1,1,1,0,0,0}$
10	7	1012	$J_{1,1,1,1,1,1,0,1,0,0}$
11	7	1010	$J_{1,1,1,1,1,1,0,0,1,0}$
12	7	1009	$J_{1,1,1,1,1,1,0,0,0,1}$
13	8	1020	$J_{1,1,1,1,1,1,1,1,0,0}$
14	8	1011	$J_{1,1,1,1,1,1,0,0,1,1}$
15	9	1022	$J_{1,1,1,1,1,1,1,1,1,0}$
16	9	511	$J_{0,1,1,1,1,1,1,1,1,1}$
17	5	841	$J_{\mathbf{3},1,0,1,0,0,1,0,0,1}$
18	7	1009	$J_{\mathbf{2},1,1,1,1,1,0,0,0,1}$
19	8	1011	$J_{\mathbf{2},1,1,1,1,1,0,0,1,1}$

Table A.3: List of all master integrals for 1-4 loops [12]. The notation in this table is $J_{a_1,\ldots,a_m} \equiv I(a_1,\ldots,a_m)$. Note that at 4 loops there are 3 master integrals with $a_i > 1$.

#	t	ID	Master integral
1	5	31744	$J_{1,1,1,1,1,0,0,0,0,0,0,0,0,0}$
2	6	32256	$J_{1,1,1,1,1,1,0,0,0,0,0,0,0,0}$
3	6	31746	$J_{1,1,1,1,1,0,0,0,0,0,0,0,1,0}$
4	6	29702	$J_{1,1,1,0,1,0,0,0,0,0,0,1,1,0}$
5	6	28686	$J_{1,1,1,0,0,0,0,0,0,0,1,1,1,0}$
6	7	32512	$J_{1,1,1,1,1,1,1,0,0,0,0,0,0,0}$
7	7	32288	$J_{1,1,1,1,1,1,0,0,0,1,0,0,0,0}$
8	7	32258	$J_{1,1,1,1,1,1,0,0,0,0,0,0,1,0}$
9	7	31754	$J_{1,1,1,1,1,0,0,0,0,0,1,0,1,0}$
10	7	30872	$J_{1,1,1,1,0,0,0,1,0,0,1,1,0,0,0}$
11	7	30858	$J_{1,1,1,1,0,0,0,1,0,0,0,1,0,1,0}$
12	7	30214	$J_{1,1,1,0,1,1,0,0,0,0,0,1,1,0}$
13	7	29703	$J_{1,1,1,0,1,0,0,0,0,0,0,1,1,1}$
14	8	32640	$J_{1,1,1,1,1,1,1,1,0,0,0,0,0,0}$
15	8	32576	$J_{1,1,1,1,1,1,1,0,1,0,0,0,0,0}$
16	8	32528	$J_{1,1,1,1,1,1,1,0,0,0,1,0,0,0}$
17	8	32513	$J_{1,1,1,1,1,1,1,0,0,0,0,0,0,1}$
18	8	32386	$J_{1,1,1,1,1,1,0,1,0,0,0,0,1,0}$
19	8	32274	$J_{1,1,1,1,1,1,0,0,0,0,1,0,0,1,0}$
20	8	32266	$J_{1,1,1,1,1,1,0,0,0,0,0,1,0,1,0}$
21	8	32259	$J_{1,1,1,1,1,1,0,0,0,0,0,0,1,1}$
22	8	31380	$J_{1,1,1,1,0,1,0,1,0,0,1,0,1,0,0}$
23	8	31246	$J_{1,1,1,1,0,1,0,0,0,0,0,1,1,1,0}$
24	8	30876	$J_{1,1,1,1,0,0,0,1,0,0,1,1,1,0,0}$
25	8	30862	$J_{1,1,1,1,0,0,0,1,0,0,0,1,1,1,0}$
26	8	30222	$J_{1,1,1,0,1,1,0,0,0,0,0,1,1,1,0}$
27	9	32704	$J_{1,1,1,1,1,1,1,1,1,0,0,0,0,0,0}$
28	9	32648	$J_{1,1,1,1,1,1,1,1,0,0,0,1,0,0,0}$
29	9	32608	$J_{1,1,1,1,1,1,1,0,1,1,0,0,0,0,0}$
30	9	32592	$J_{1,1,1,1,1,1,1,0,1,0,1,0,0,0,0}$
31	9	32529	$J_{1,1,1,1,1,1,1,0,0,0,1,0,0,0,1}$
32	9	32518	$J_{1,1,1,1,1,1,1,0,0,0,0,0,1,1,0}$
33	9	32394	$J_{1,1,1,1,1,1,0,1,0,0,0,1,0,1,0}$
34	9	32390	$J_{1,1,1,1,1,1,0,1,0,0,0,0,1,1,0}$
35	9	32329	$J_{1,1,1,1,1,1,0,0,1,0,0,1,0,0,1}$
36	9	32278	$J_{1,1,1,1,1,1,0,0,0,0,1,0,1,1,0}$
37	9	32270	$J_{1,1,1,1,1,1,0,0,0,0,0,1,1,1,0}$
38	9	32267	$J_{1,1,1,1,1,1,0,0,0,0,0,1,0,1,1}$
39	9	31516	$J_{1,1,1,1,0,1,1,0,0,0,1,1,1,0,0}$
40	9	31388	$J_{1,1,1,1,0,1,0,1,0,0,1,1,1,0,0}$
41	9	30231	$J_{1,1,1,0,1,1,0,0,0,0,1,0,1,1,1}$
42	10	32736	$J_{1,1,1,1,1,1,1,1,1,1,0,0,0,0,0}$
43	10	32712	$J_{1,1,1,1,1,1,1,1,1,0,0,1,0,0,0}$
44	10	32708	$J_{1,1,1,1,1,1,1,1,1,0,0,0,1,0,0}$
45	10	32674	$J_{1,1,1,1,1,1,1,1,0,1,0,0,0,1,0}$
46	10	32652	$J_{1,1,1,1,1,1,1,1,0,0,0,1,1,0,0}$
47	10	32596	$J_{1,1,1,1,1,1,1,0,1,0,1,0,1,0,0}$
48	10	32562	$J_{1,1,1,1,1,1,1,0,0,1,1,0,0,1,0}$
49	10	32534	$J_{1,1,1,1,1,1,1,0,0,0,1,0,1,1,0}$
50	10	32398	$J_{1,1,1,1,1,1,0,1,0,0,0,1,1,1,0}$
51	10	32391	$J_{1,1,1,1,1,1,0,1,0,0,0,0,1,1,1}$
52	10	32279	$J_{1,1,1,1,1,1,0,0,0,0,1,0,1,1,1}$
53	10	31420	$J_{1,1,1,1,0,1,0,1,0,1,1,1,1,0,0}$
54	10	30563	$J_{1,1,1,0,1,1,1,0,1,1,0,0,0,1,1}$
55	10	30239	$J_{1,1,1,0,1,1,0,0,0,0,1,1,1,1,1}$
56	10	29550	$J_{1,1,1,0,0,1,1,0,1,1,0,1,1,1,0}$
57	11	32744	$J_{1,1,1,1,1,1,1,1,1,1,0,1,0,0,0}$
58	11	32737	$J_{1,1,1,1,1,1,1,1,1,1,0,0,0,0,1}$
59	11	32713	$J_{1,1,1,1,1,1,1,1,1,0,0,1,0,0,1}$
60	11	32682	$J_{1,1,1,1,1,1,1,1,0,1,0,1,0,1,0}$
61	11	31736	$J_{1,1,1,1,0,1,1,1,1,1,1,1,0,0,0}$
62	11	30691	$J_{1,1,1,0,1,1,1,1,1,1,0,0,0,1,1}$
63	11	30526	$J_{1,1,1,0,1,1,1,0,0,1,1,1,1,1,0}$
64	12	32745	$J_{1,1,1,1,1,1,1,1,1,1,0,1,0,0,1}$
65	12	31740	$J_{1,1,1,1,0,1,1,1,1,1,1,1,1,0,0}$
66	12	30699	$J_{1,1,1,0,1,1,1,1,1,1,0,1,0,1,1}$
67	12	30527	$J_{1,1,1,0,1,1,1,0,0,1,1,1,1,1,1}$

Table A.4: List of all master integrals with $a_i \leq 1\ \forall i$ at 5 loops [12]. The notation in this table is $J_{a_1,\ldots,a_m} \equiv I(a_1,\ldots,a_m)$. Master integrals with $a_i > 1$ are found in Table A.5.

#	t	ID	Master integral
68	6	29702	$J_{\mathbf{3},1,1,0,1,0,0,0,0,0,0,0,1,1,0}$
69	6	28686	$J_{\mathbf{3},1,1,0,0,0,0,0,0,0,0,1,1,1,0}$
70	7	30214	$J_{\mathbf{2},1,1,0,1,1,0,0,0,0,0,0,1,1,0}$
71	7	30214	$J_{\mathbf{3},1,1,0,1,1,0,0,0,0,0,0,1,1,0}$
72	7	29703	$J_{\mathbf{2},1,1,0,1,0,0,0,0,0,0,0,1,1,1}$
73	7	29703	$J_{\mathbf{2},\mathbf{2},1,0,1,0,0,0,0,0,0,0,1,1,1}$
74	7	29703	$J_{\mathbf{3},1,1,0,1,0,0,0,0,0,0,0,1,1,1}$
75	8	32259	$J_{\mathbf{2},1,1,1,1,1,0,0,0,0,0,0,0,1,1}$
76	8	31246	$J_{1,\mathbf{2},1,1,0,1,0,0,0,0,0,1,1,1,0}$
77	8	31246	$J_{\mathbf{2},1,1,1,0,1,0,0,0,0,0,1,1,1,0}$
78	8	31246	$J_{\mathbf{3},1,1,1,0,1,0,0,0,0,0,1,1,1,0}$
79	8	30862	$J_{\mathbf{2},1,1,1,0,0,0,1,0,0,0,1,1,1,0}$
80	8	30222	$J_{\mathbf{2},1,1,0,1,1,0,0,0,0,0,1,1,1,0}$
81	8	30222	$J_{\mathbf{2},\mathbf{2},1,0,1,1,0,0,0,0,0,1,1,1,0}$
82	8	30222	$J_{\mathbf{3},1,1,0,1,1,0,0,0,0,0,1,1,1,0}$
83	9	32608	$J_{\mathbf{2},1,1,1,1,1,1,0,1,1,0,0,0,0,0}$
84	9	32390	$J_{\mathbf{2},1,1,1,1,1,0,1,0,0,0,0,1,1,0}$
85	9	32278	$J_{\mathbf{2},1,1,1,1,1,0,0,0,0,1,0,1,1,0}$
86	9	32270	$J_{1,1,\mathbf{2},1,1,1,0,0,0,0,0,1,1,1,0}$
87	9	32270	$J_{1,\mathbf{2},1,1,1,1,0,0,0,0,0,1,1,1,0}$
88	9	32270	$J_{\mathbf{2},1,1,1,1,1,0,0,0,0,0,1,1,1,0}$
89	9	32270	$J_{\mathbf{2},\mathbf{2},1,1,1,1,0,0,0,0,0,1,1,1,0}$
90	9	32270	$J_{\mathbf{3},1,1,1,1,1,0,0,0,0,0,1,1,1,0}$
91	9	32267	$J_{\mathbf{2},1,1,1,1,1,0,0,0,0,0,1,0,1,1}$
92	9	31516	$J_{1,1,\mathbf{2},1,0,1,1,0,0,0,1,1,1,0,0}$
93	9	31516	$J_{1,\mathbf{2},1,1,0,1,1,0,0,0,1,1,1,0,0}$
94	9	31516	$J_{\mathbf{2},1,1,1,0,1,1,0,0,0,1,1,1,0,0}$
95	9	31388	$J_{\mathbf{2},1,1,1,0,1,0,1,0,0,1,1,1,0,0}$
96	9	30231	$J_{\mathbf{2},1,1,0,1,1,0,0,0,0,1,0,1,1,1}$
97	9	30231	$J_{\mathbf{2},1,1,0,1,1,0,0,0,0,\mathbf{2},0,1,1,1}$
98	9	30231	$J_{\mathbf{2},1,\mathbf{2},0,1,1,0,0,0,0,1,0,1,1,1}$
99	9	30231	$J_{\mathbf{2},\mathbf{2},1,0,1,1,0,0,0,0,1,0,1,1,1}$
100	9	30231	$J_{\mathbf{3},1,1,0,1,1,0,0,0,0,1,0,1,1,1}$
101	10	32736	$J_{\mathbf{2},1,1,1,1,1,1,1,1,1,0,0,0,0,0}$
102	10	32596	$J_{\mathbf{2},1,1,1,1,1,1,0,1,0,1,0,1,0,0}$
103	10	32596	$J_{\mathbf{2},1,\mathbf{2},1,1,1,1,0,1,0,1,0,1,0,0}$
104	10	32596	$J_{\mathbf{2},\mathbf{2},1,1,1,1,1,0,1,0,1,0,1,0,0}$
105	10	32596	$J_{\mathbf{3},1,1,1,1,1,1,0,1,0,1,0,1,0,0}$
106	10	32534	$J_{\mathbf{2},1,1,1,1,1,1,0,0,0,1,0,1,1,0}$
107	10	32398	$J_{\mathbf{2},1,1,1,1,1,0,1,0,0,0,1,1,1,0}$
108	10	32391	$J_{\mathbf{2},1,1,1,1,1,0,1,0,0,0,0,1,1,1}$
109	10	32279	$J_{1,1,1,\mathbf{2},1,1,0,0,0,0,1,0,1,1,1}$
110	10	32279	$J_{\mathbf{2},1,1,1,1,1,0,0,0,0,1,0,1,1,1}$
111	10	32279	$J_{\mathbf{2},\mathbf{2},1,1,1,1,0,0,0,0,1,0,1,1,1}$
112	10	32279	$J_{\mathbf{3},1,1,1,1,1,0,0,0,0,1,0,1,1,1}$
113	10	30239	$J_{\mathbf{2},1,1,0,1,1,0,0,0,0,1,1,1,1,1}$
114	10	30239	$J_{\mathbf{2},1,\mathbf{2},0,1,1,0,0,0,0,1,1,1,1,1}$
115	10	30239	$J_{\mathbf{2},\mathbf{2},1,0,1,1,0,0,0,0,1,1,1,1,1}$
116	10	30239	$J_{\mathbf{3},1,1,0,1,1,0,0,0,0,1,1,1,1,1}$
117	10	29550	$J_{1,1,1,0,0,\mathbf{2},1,0,1,1,0,1,1,1,0}$
118	10	29550	$J_{\mathbf{2},1,1,0,0,1,1,0,1,1,0,1,1,1,0}$
119	10	29550	$J_{\mathbf{3},1,1,0,0,1,1,0,1,1,0,1,1,1,0}$
120	11	32744	$J_{\mathbf{2},1,1,1,1,1,1,1,1,1,0,1,0,0,0}$
121	11	31736	$J_{\mathbf{2},1,1,1,0,1,1,1,1,1,1,1,0,0,0}$
122	11	30526	$J_{1,1,\mathbf{2},0,1,1,1,0,0,1,1,1,1,1,0}$
123	11	30526	$J_{1,\mathbf{2},1,0,1,1,1,0,0,1,1,1,1,1,0}$
124	11	30526	$J_{\mathbf{2},1,1,0,1,1,1,0,0,1,1,1,1,1,0}$
125	11	30526	$J_{\mathbf{2},\mathbf{2},1,0,1,1,1,0,0,1,1,1,1,1,0}$
126	11	30526	$J_{\mathbf{3},1,1,0,1,1,1,0,0,1,1,1,1,1,0}$
127	12	31740	$J_{\mathbf{3},1,1,1,0,1,1,1,1,1,1,1,1,0,0}$
128	12	30527	$J_{\mathbf{2},1,1,0,1,1,1,0,0,1,1,1,1,1,1}$
129	12	30527	$J_{\mathbf{2},1,\mathbf{2},0,1,1,1,0,0,1,1,1,1,1,1}$
130	12	30527	$J_{\mathbf{2},\mathbf{2},1,0,1,1,1,0,0,1,1,1,1,1,1}$
131	12	30527	$J_{\mathbf{3},1,1,0,1,1,1,0,0,1,1,1,1,1,1}$

Table A.5: List of all master integrals with at least one $a_i > 1$ at 5 loops [12]. The notation in this table is $J_{a_1,\ldots,a_m} \equiv I(a_1,\ldots,a_m)$. Master integrals with $a_i \leq 1\ \forall i$ are found in Table A.4.

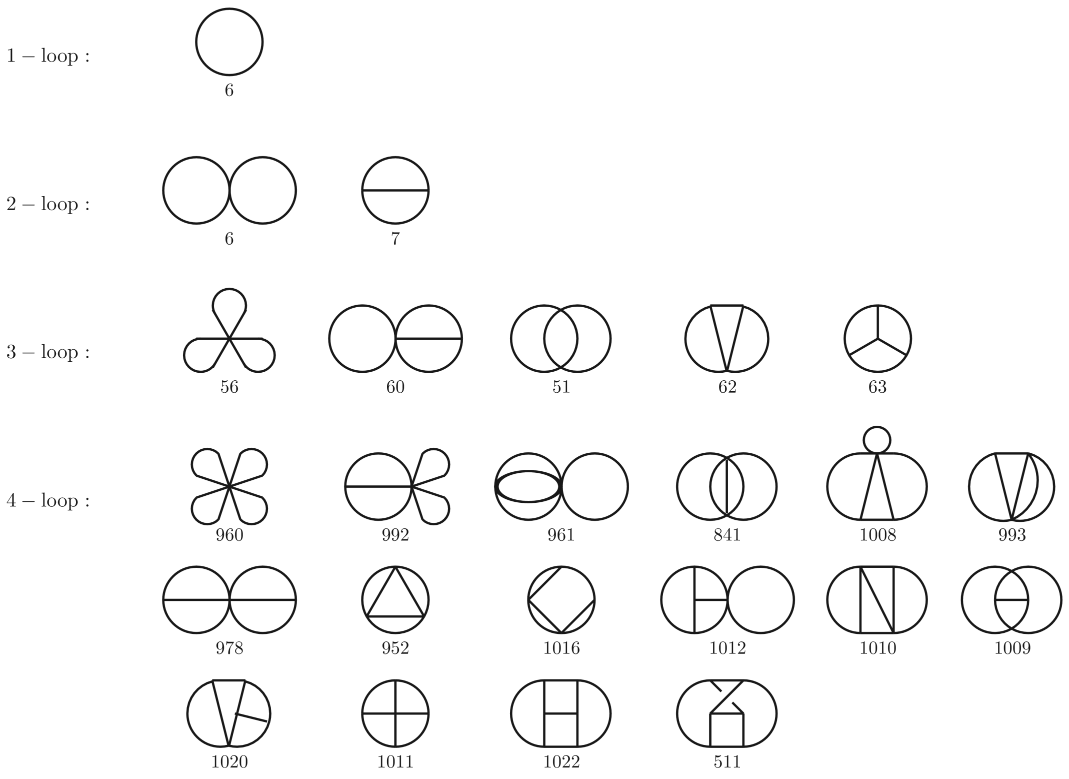

Figure A.1: All diagrams of the vacuum topologies up to 4 loops with the sector IDs of the respective representative sectors [12, 45, 46].

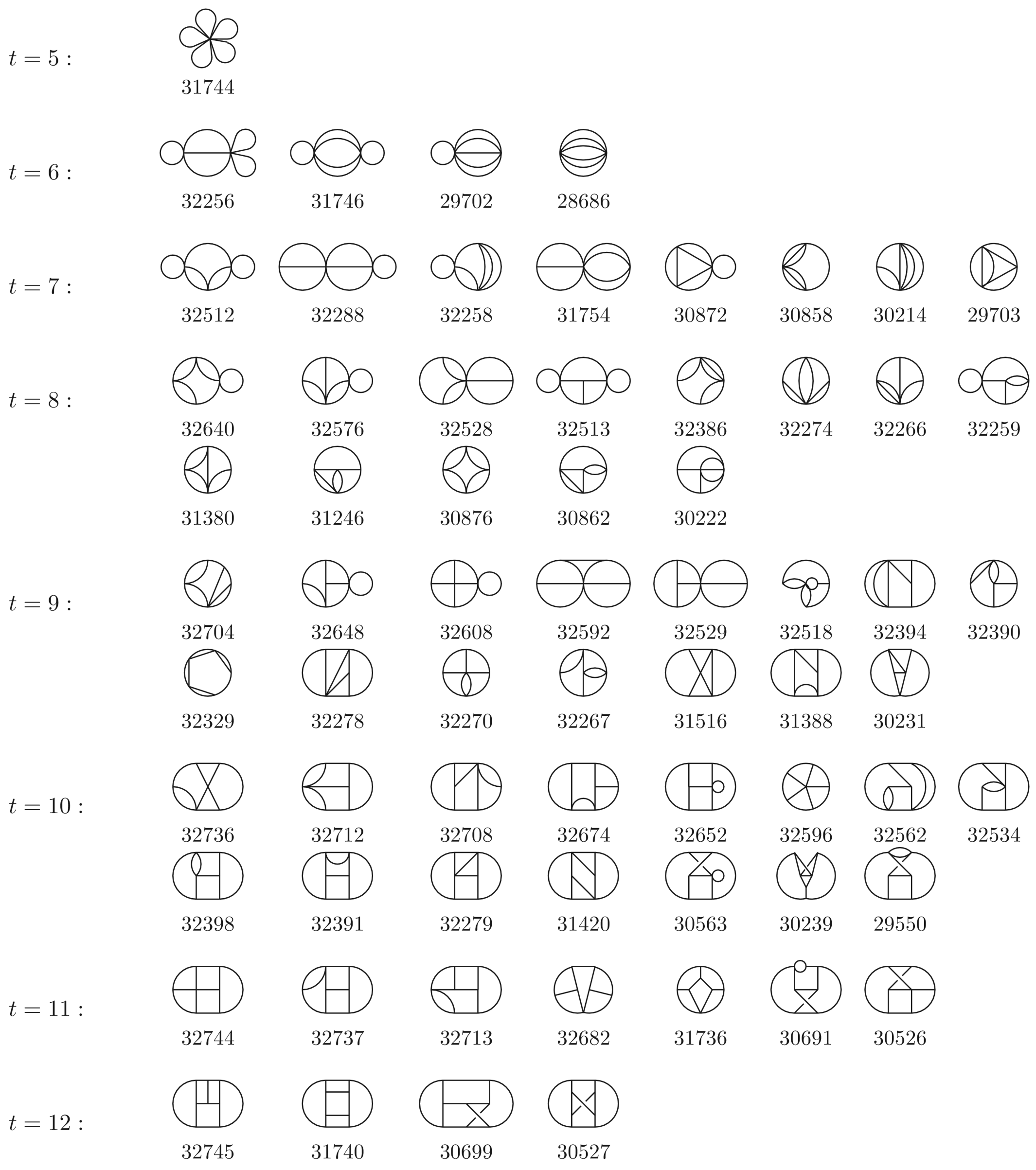

Figure A.2: All diagrams of the 67 vacuum 5-loop topologies with the sector IDs of the respective representative sectors [12, 45, 47].

References

[1] J. Gluza, K. Kajda and D. A. Kosower, *Towards a Basis for Planar Two-Loop Integrals*, Phys. Rev. D **83** (2011) 045012 [arXiv:1009.0472].

[2] C. Anastasiou and A. Lazopoulos, *Automatic integral reduction for higher order perturbative calculations*, JHEP **0407** (2004) 046 [hep-ph/0404258].

[3] A. V. Smirnov, *Algorithm FIRE – Feynman Integral REduction*, JHEP **0810** (2008) 107 [arXiv:0807.3243].

[4] C. Studerus, *Reduze-Feynman Integral Reduction in C++*, Comput. Phys. Commun. **181** (2010) 1293 [arXiv:0912.2546 [physics.comp-ph]]; A. von Manteuffel and C. Studerus, *Reduze 2 - Distributed Feynman Integral Reduction*, arXiv:1201.4330.

[5] S. Laporta, *High precision epsilon expansions of massive four loop vacuum bubbles*, Phys. Lett. B **549** (2002) 115 [hep-ph/0210336].

[6] R. M. Schabinger, *A New Algorithm For The Generation Of Unitarity-Compatible Integration By Parts Relations*, JHEP **1201** (2012) 077 [arXiv:1111.4220].

[7] K. G. Chetyrkin, M. Misiak and M. Munz, *Beta functions and anomalous dimensions up to three loops*, Nucl. Phys. B **518** (1998) 473 [hep-ph/9711266].

[8] Y. Schröder and M. Steinhauser, *Four-loop singlet contribution to the rho parameter*, Phys. Lett. B **622** (2005) 124 [hep-ph/0504055].

[9] R. Boughezal and M. Czakon, *Single scale tadpoles and O(G(F m(t)**2 alpha(s)**3)) corrections to the rho parameter*, Nucl. Phys. B **755** (2006) 221 [hep-ph/0606232].

[10] Y. Schröder and A. Vuorinen, *High precision evaluation of four loop vacuum bubbles in three-dimensions*, hep-ph/0311323.

[11] Y. Schröder and A. Vuorinen, *High-precision epsilon expansions of single-mass-scale four-loop vacuum bubbles*, JHEP **0506** (2005) 051 [hep-ph/0503209].

[12] Jan Möller, *Fully Massive Tadpoles at 5-loop: Reduction and Difference Equations*, Dissertation, University of Bielefeld, 2012.

[13] C. -N. Yang and R. L. Mills, *Conservation of Isotopic Spin and Isotopic Gauge Invariance*, Phys. Rev. **96** (1954) 191.

[14] L. D. Faddeev and V. N. Popov, *Feynman Diagrams for the Yang-Mills Field*, Phys. Lett. B **25** (1967) 29.

[15] G. 't Hooft, *Dimensional regularization and the renormalization group*, Nucl. Phys. B **61** (1973) 455.

[16] G. 't Hooft and M. J. G. Veltman, *Regularization and Renormalization of Gauge Fields*, Nucl. Phys. B **44** (1972) 189.

[17] D. J. Gross and F. Wilczek, *Ultraviolet Behavior of Nonabelian Gauge Theories*, Phys. Rev. Lett. **30** (1973) 1343.

[18] W. E. Caswell, *Asymptotic Behavior of Nonabelian Gauge Theories to Two Loop Order*, Phys. Rev. Lett. **33** (1974) 244.

[19] D. R. T. Jones, *Two Loop Diagrams in Yang-Mills Theory*, Nucl. Phys. B **75** (1974) 531.

[20] O. V. Tarasov, A. A. Vladimirov and A. Y. .Zharkov, *The Gell-Mann-Low Function of QCD in the Three Loop Approximation*, Phys. Lett. B **93** (1980) 429.

[21] S. A. Larin and J. A. M. Vermaseren, *The Three loop QCD Beta function and anomalous dimensions*, Phys. Lett. B **303** (1993) 334 [hep-ph/9302208].

[22] T. van Ritbergen, J. A. M. Vermaseren and S. A. Larin, *The Four loop beta function in quantum chromodynamics*, Phys. Lett. B **400** (1997) 379 [hep-ph/9701390].

[23] K. G. Chetyrkin, *Four-loop renormalization of QCD: Full set of renormalization constants and anomalous dimensions*, Nucl. Phys. B **710** (2005) 499 [hep-ph/0405193].

[24] K. G. Chetyrkin and V. A. Smirnov, *R* Operation Corrected*, Phys. Lett. B **144** (1984) 419.

[25] M. Misiak and M. Munz, *Two loop mixing of dimension five flavor changing operators*, Phys. Lett. B **344** (1995) 308 [hep-ph/9409454].

[26] J. C. Collins, *Normal Products in Dimensional Regularization*, Nucl. Phys. B **92** (1975) 477.

[27] T. Binoth, E. W. N. Glover, P. Marquard and J. J. van der Bij, *Two loop corrections to light by light scattering in supersymmetric QED*, JHEP **0205** (2002) 060 [hep-ph/0202266].

[28] Michael E. Peskin and Daniel V. Schroeder, *An Introduction to Quantum Field Theory*, Westview, 1995.

[29] M. Czakon, *Automatized analytic continuation of Mellin-Barnes integrals*, Comput. Phys. Commun. **175** (2006) 559 [hep-ph/0511200].

[30] A. V. Smirnov and M. N. Tentyukov, *Feynman Integral Evaluation by a Sector decompositTion Approach (FIESTA)*, Comput. Phys. Commun. **180** (2009) 735 [arXiv:0807.4129 [hep-ph]].

[31] A. V. Smirnov and A. V. Petukhov, *The Number of Master Integrals is Finite*, Lett. Math. Phys. **97** (2011) 37 [arXiv:1004.4199 [hep-th]].

[32] O. V. Tarasov, *Generalized recurrence relations for two loop propagator integrals with arbitrary masses*, Nucl. Phys. B **502** (1997) 455 [hep-ph/9703319].

[33] A. V. Smirnov, V. A. Smirnov and M. Steinhauser, *Applying Mellin-Barnes technique and Groebner bases to the three-loop static potential*, PoS RADCOR **2007** (2007) 024 [arXiv:0805.1871 [hep-ph]].

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

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

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

[37] K. G. Chetyrkin and F. V. Tkachov, *Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops*, Nucl. Phys. B **192** (1981) 159; F. V. Tkachov, *A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions*, Phys. Lett. B **100** (1981) 65.

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

[39] R. N. Lee, *Group structure of the integration-by-part identities and its application to the reduction of multiloop integrals*, JHEP **0807** (2008) 031 [arXiv:0804.3008 [hep-ph]].

[40] C. Bogner and S. Weinzierl, *Feynman graph polynomials*, Int. J. Mod. Phys. A **25** (2010) 2585 [arXiv:1002.3458 [hep-ph]].

[41] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalem Polynomideal*, http://www.risc.jku.at/Groebner-Bases-Bibliography/details.php?details_id=706, Dissertation, 1965.

[42] D. Cabarcas and J. Ding, *Linear Algebra to Compute Syzygies and Gröbner Bases*, ISSAC '11: Proceedings of the 36th international symposium on Symbolic and algebraic computation, 2011.

[43] C. W. Bauer, A. Frink and R. Kreckel, *Introduction to the GiNaC framework for symbolic computation within the C++ programming language*, cs/0004015 [cs-sc].

[44] R. H. Lewis, *Computer algebra system Fermat*, http://home.bway.net/lewis/.

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

[46] K. Kajantie, M. Laine and Y. Schroder, *A Simple way to generate high order vacuum graphs*, Phys. Rev. D **65** (2002) 045008 [hep-ph/0109100].

[47] Jannis Schücker, Templates for 5-loop Vacuum Diagrams in Axodraw, not published.

Acknowledgements

I would like to extend my gratitude to the following people:

- York Schröder, for his great supervision of this thesis, his patience with all my questions and for providing lists of sector shifts, symmetries and zero sectors.
- Jan Möller, for helpful discussions about difference equations and how to best find them, and for providing me with his results, which were invaluable for cross-checking my own.
- Lisa, Benedict and Michael for proofreading this thesis.
- The people of D6/E6, especially Anne, Eva, Benedict and Michael, for a great atmosphere to work in.
- Last but not least, my family and friends, for always being supportive and patient throughout my work on this thesis.

Declaration

I hereby affirm that this master's thesis represents my own work and has not been previously submitted to any examination office. All resources used have been referenced.

Bielefeld, November 19th, 2012

Thomas Luthe