Contents lists available at ScienceDirect

Computer Physics Communications

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

Computer Programs in Physics

FeynCalc 10: Do multiloop integrals dream of computer codes? ☆

Vladyslav Shtabovenko [a,b,*], Rolf Mertig [c], Frederik Orellana [d]

[a] Theoretische Physik 1, Center for Particle Physics Siegen, Universität Siegen, Walter-Flex-Str. 3, 57068 Siegen, Germany
[b] Institut für Theoretische Teilchenphysik (TTP), Karlsruhe Institute of Technology (KIT), Wolfgang-Gaede-Straße 1, 76131 Karlsruhe, Germany
[c] GluonVision GmbH, Bötzowstr. 10, 10407 Berlin, Germany
[d] Technical University of Denmark, Anker Engelundsvej 1, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Keywords:
High energy physics
Feynman diagrams
Loop integrals
Dimensional regularization
Renormalization
Dirac algebra
Passarino–Veltman
Cheng–Wu
Feynman integrals
Symanzik polynomials
Light-cone
Multiloop
FeynCalc

ABSTRACT

In this work we report on a new version of FEYNCALC, a MATHEMATICA package widely used in the particle physics community for manipulating quantum field theoretical expressions and calculating Feynman diagrams. Highlights of the new version include greatly improved capabilities for doing multiloop calculations, including topology identification and minimization, optimized tensor reduction, rewriting of scalar products in terms of inverse denominators, detection of equivalent or scaleless loop integrals, derivation of Symanzik polynomials, Feynman parametric as well as graph representation for master integrals and initial support for handling differential equations and iterated integrals. In addition to that, the new release also features completely rewritten routines for color algebra simplifications, inclusion of symmetry relations between arguments of Passarino–Veltman functions, tools for determining matching coefficients and quantifying the agreement between numerical results, improved export to LaTeX and first steps towards a better support of calculations involving light-cone vectors.

PROGRAM SUMMARY
Program Title: FeynCalc
CPC Library link to program files: https://doi.org/10.17632/cmpjr5ktmp.3
Developer's repository link: https://github.com/FeynCalc/feyncalc
Licensing provisions: GPLv3
Programming language: Wolfram Language
Supplementary material: Manual, example notebooks.
Journal reference of previous version: Comput. Phys. Commun. 256 (2020) 107478
Does the new version supersede the previous version?: Yes.
Reasons for the new version: Addition of new routines required for multiloop calculations.
Summary of revisions: FEYNCALC can be now used to calculate multiloop Feynman diagrams either standalone or as a part of a toolchain.
Nature of problem: Analytic calculations of higher-order corrections to particle physics processes using Feynman diagrammatic expansion.
Solution method: The required algorithms and algebraic identities are implemented in Wolfram MATHEMATICA.
Additional comments including restrictions and unusual features: Depending on the complexity of the problem, the number of terms might become so high that MATHEMATICA alone will not be sufficient to finish the calculation within a reasonable time frame or at all.

1. Introduction

Modern high energy physics heavily relies on Feynman's diagrammatic approach to the calculation of perturbative corrections in particle reactions. The vast majority of predictions required to match the expected experimental precision of the High luminosity LHC [1] as well as the proposed future colliders are still obtained by calculating multitudes of complicated Feynman diagrams, and it does not seem likely, that this situation is going to change in the near future.

The number of diagrams involved can easily go into thousands or even millions, which makes the usage of automation indispensable. The importance of efficient algorithms for such calculations has already been recognized decades ago (cf. e.g. refs. [2,3]) and the unceasing development of new calculational techniques is one of the main reasons why

☆ The review of this paper was arranged by Prof. Z. Was.
* Corresponding author.
E-mail addresses: shtabovenko@physik.uni-siegen.de (V. Shtabovenko), rolfm@gluonvision.com (R. Mertig), fror@dtu.dk (F. Orellana).

https://doi.org/10.1016/j.cpc.2024.109357
Received 10 January 2024; Received in revised form 15 August 2024; Accepted 26 August 2024
Available online 29 August 2024

calculations unthinkable now might nonetheless become feasible in the next years. A good overview over modern methods used in higher-order perturbative calculations can be found e.g. in refs. [4,5].

While efficient codes for automatic numerical calculations at tree-level and one-loop accuracy are available to the wide public since years, the treatment of multiloop diagrams remains challenging and requires considerable expertise.

Beyond one-loop the proliferation of diagrams becomes an issue, where processes involving thousands or tens of thousands diagrams are not uncommon. Their complexity increases as well, meaning that algebraic evaluation requires significantly more time and computational resources. At one loop all integrals with quadratic propagators can be conveniently reduced to a basis (e.g. that of Passarino–Veltman [6]) and then evaluated either analytically or numerically using existing libraries. However, already at two loops this approach becomes unfeasible. First, the complete basis of two-loop master integrals with arbitrary masses and multiple legs is not known yet. Second, the analytic reduction of multiloop integrals to some basis integrals is currently doable only when they involve few kinematic scales. For example, as of now a reduction of integrals appearing in the $2 \to 3$ process $q\bar{q} \to t\bar{t}H$ with five scales requires an intricate combination of state-of-the-art numerical and analytic techniques [7] and cannot be done by brute force. At the same time, at one-loop a diagram with five legs and five scales would not pose any major difficulties. Even when one manages to obtain a final set of master integrals, their evaluation often turns out to be a significant challenge. Analytic results are scarce [8] and are often not generic enough to cover all phenomenologically interesting (i.e. those with different masses and multiple legs) master integrals even at two loops. Numerical evaluation using sector decomposition [9–12] is always possible,[1] but achieving sufficient precision in problematic phase space regions can quickly become challenging. In this sense, performance issues arising in multiloop calculations are usually solved on a case-by-case basis, but for a general-purpose code one would need better algorithms that are efficient enough to cover all interesting processes, in particular those involving massive particles.

Limiting ourselves to multiloop calculations that are feasible with the current-day technology, the task of streamlining all the necessary calculational steps such, that they can be performed in a fully automatic fashion, is still very much work in progress. A recent review of the field can be found e.g. in ref. [13]. Some remarkable achievements towards multiloop automation are the release of CARAVEL [14], a framework for multiloop computations using numerical unitarity, as well as the recent version of PYSECDEC [15] capable of evaluating multiloop amplitudes numerically. Apart from that, there are also ongoing efforts to “upgrade” existing one-loop codes at least to two-loop accuracy (cf. e.g. [16–20]).

Irrespective of whether one wants to calculate a cross section, a matching coefficient or a renormalization constant, the vast majority of multiloop calculations usually require the completion of some fundamental steps that can be summarized as follows: (i) generation of Feynman diagrams for the given process, (ii) algebraic simplification of the corresponding amplitudes (including suitable expansions in small parameters), (iii) reduction of the occurring loop integrals to a smaller set of master integrals, using Integration-By-Parts (IBP) [21,22] techniques, (iv) evaluation of the master integrals. Fortunately, the first three steps in the above list can be completed in a highly automatized fashion by combining publicly available software with suitable self-written code. In most cases the problems one has to deal with happen to be of technical (e.g. bugs in the code, performance bottlenecks, lack of computing power) rather than conceptual nature. Analytic calculations of master integrals usually require more experience and creativity, unless the desired results are already available in the literature and can be directly plugged into the final result.

A lot of frameworks addressing steps (i)-(iii) happen to be some private codes developed by single researchers or whole research groups specializing on loop calculations. Sometimes they can be obtained upon request (albeit without much support or proper documentation) but most tools are still available only to collaborators. With the new generation of researchers embracing open source ideas and making their codes public (cf. e.g. refs. [23,25,24,26]) this situation started to change. The corresponding tools aim at connecting different steps behind a multiloop calculation to each other within a single framework and normally need to address the tasks of generating Feynman diagrams, inserting Feynman rules, identifying the occurring loop integral topologies, minimizing their number and supplying some templates for the subsequent amplitude evaluation in FORM [27,28].

One aim of the present work is to make analytic multiloop calculations accessible to a broader range of particle theorists. The method is to extend the well-established one-loop functionality of FEYNCALC [29–32] with the modern state-of-the art algorithms. The new version of the package presented here thus includes a large set of optimized routines for dealing with multiloop calculations.

This paper is the first in a series of three publications describing our take on creating a new framework for semi-automatic multiloop calculations. While this work revolves solely around FEYNCALC, the two subsequent papers will introduce a new version of FEYNHELPERS [33] — an add-on for connecting FEYNCALC to other software tools related to quantum field theory (QFT) and finally a FORM-based framework for symbolic evaluation of Feynman diagrams that makes use of FEYNCALC and MATHEMATICA during certain calculational steps.

The remainder of the present paper is organized as follows: In Section 2 we briefly introduce FEYNCALC and compare it to similar codes, while instructions on installing the package from GITHUB can be found in Section 3. New symbols and functions related to the multiloop capabilities of the package are presented in Section 4. Section 5 discusses some special routines that can be handy when calculating master integrals. Last but not least, new features and improvements that are not directly related to multiloop calculations can be found in Section 6. A description of example calculations utilizing new multiloop capabilities is offered in Section 7. Finally, in Section 8 we summarize our results and provide an outlook on the future of the package.

Contrary to previous FEYNCALC papers, the amount of MATHEMATICA code presented will be kept to a bare minimum. The reason is that FEYNCALC 10 features a comprehensive manual, covering every symbol introduced in the package (including examples) and containing a tutorial for learning the basics of the package. In addition to that, examples related to the functionality described in this paper can be found in the accompanying MATHEMATICA notebook.

2. Context and state of the art

Originally, FEYNCALC was designed to handle loop integrals using Passarino-Veltman [6] reduction, which effectively limited its applicability to one-loop calculations only. Subsequent iterations of the code [34] allowed for working with certain types of multiloop integrals (e.g. using the TARCER [35] add-on) but until version 10 it was neither very efficient nor suitable for general purpose multiloop calculations.

The number of actively developed tools for semi-automatic calculations with an emphasis on tree- and one-loop level similar to FEYNCALC has been steadily decreasing over the last years. Both HEPMATH [36] and PACKAGE-X [37,38] have unfortunately been abandoned, while there seems to be hardly new codes covering a wide range of applications as well as receiving regular updates and bug fixes. In this context it is also worth mentioning PHYSICS,[2] a package shipped with MAPLE computer algebra system, whose particle physics related capabilities have

[1] For some problematic integrals one may need to request help from the code developers, though.

[2] https://www.maplesoft.com/products/maple/features/physicsresearch.aspx.

been significantly extended in the last few years. Also, FORMTRACER [39] can be handy in some circumstances, although this tool is mostly limited to algebraic simplifications.

What we regard as a crucial feature of semi-automatic codes is the ability to access intermediate expressions at any stage of the calculation and to organize calculations in a flexible way by sequentially applying high-level functions to the original input. Depending on the quantities one wants to derive, this approach might be more efficient than using programs offering a higher level of automation and less flexibility. Of course, fully automated tools for tree-level and one-loop calculations such as MADGRAPH [40], GOSAM [41,42], HERWIG [43,44], HELAC-NLO [45], POWHEG-BOX [46–48], SHERPA [49,50], WHIZARD [51,52], CALCHEP [53], COMPHEP [54], GRACE [55,56] etc. are very useful when performing the tasks they were originally designed for, e.g. calculating cross-sections and decay rates.

Since version 10 FEYNCALC can be in principle used to perform real multiloop calculations or at least to derive multiloop amplitudes written as linear combinations of loop integrals belonging to a certain set of integral topologies. Of course, due to the performance limitations of MATHEMATICA as compared to FORM the number of multiloop diagrams one can completely evaluate with FEYNCALC and their complexity are rather limited. Thus, in this context, FEYNCALC should be seen as providing supplementary tools for structuring and reducing multiloop calculations and a supplementary way to check specific calculations of other multiloop frameworks. Our code has already been employed in this fashion in some research projects cf. e.g. [57–63] and we expect more publications to appear in the near future.

ALIBRARY[3] is one of the few tools that uses MATHEMATICA for gluing different parts of the computational setup together and implementing some convenience functions. Nevertheless, this code uses QGRAF [64] for generating Feynman diagrams (Feynman rules for QCD are already included) and FORM for evaluating them. The topology identification part is outsourced to a dedicated tool called FEYNSON [25] — developed by the author of ALIBRARY. The package also includes interfaces to GRAPHVIZ [65], LITERED [66,67], FIRE [68–70], KIRA [71–75], PYSECDEC [76–78] and other related programs. FORM source files or configuration files for FIRE and KIRA generated by ALIBRARY can be used independently — which is an approach similar to that of FEYNCALC. Tensor reduction and Dirac algebra simplifications are not included, since the code tacitly assumes the usage of projectors. However, those parts can be still added to the generated FORM code by hand at a later stage.

The original idea behind the PYTHON package TAPIR [23] was to create a modern replacement for Q2E [79,80], a C++ code for inserting Feynman rules into QGRAF output. Together with EXP and CALC, Q2E is a part of the so-called Karlsruhe tool chain[4] that has been used in many cutting-edge multiloop calculations. In the course of its development, TAPIR obtained numerous features that go far beyond the capabilities of Q2E. In particular, it can be also used for identifying and minimizing integral topologies, visualizing Feynman diagrams, performing partial fraction decomposition and generating amplitudes in FORM format. Furthermore, it understands Feynman rules in the Universal FeynRules Output (UFO) [81,82] format. The FORM code for evaluating the amplitudes is, however, not part of TAPIR. Just as in the case of ALIBRARY, TAPIR has no built-in capabilities to perform tensor reduction of loop integrals or to simplify chains of Dirac matrices that do not involve Dirac traces.

The program FEAMGEN.JL [26] is written in the JULIA language, has built-in support for UFO models and uses YAML run cards describing the process that needs to be calculated. Automatically generated FORM code takes care of Dirac and color algebra, while a built-in routine of FEAMGEN.JL minimizes the number of loop integral topologies. However, the evaluated amplitude still requires tensor reduction or a suitable projector, while the obtained topologies are not readily converted into configuration files for IBP-reduction tools such as FIRE or KIRA. On the other hand, since the output of FEAMGEN.JL is offered in form of JDL2 or plain text files, those additional steps can be also done by the user.

In the case of HEPLIB [24,83], the authors chose to employ C++ and in particular the GINAC [84] library as the means of connecting different calculational steps with each other. As such, this tool is usable both in C++ and PYTHON. Apart from the fact that HEPLIB uses QGRAF and FORM for the common tasks of generating and evaluating amplitudes, it also features tensor reduction, partial fraction decomposition and automatic creation of configuration files for FIRE and KIRA as well as a custom implementation of sector decomposition [9–12] for numerical evaluation of master integrals.

A comparison of some loop-related features present in FEYNCALC and other tools is presented in Table 1. It goes without saying that since all the above-mentioned codes employ FORM, they easily outperform MATHEMATICA (and hence FEYNCALC) when it comes to the evaluation of Feynman diagrams. FEYNCALC can identify the occurring topologies and minimize their number as well as directly simplify Dirac and color algebra and carry out tensor reduction of loop integrals. Feynman diagram generation is done using FEYNARTS, although an experimental interface to QGRAF is available in the development version of the yet unreleased FEYNHELPERS add-on. The same goes for automatic generation of run cards needed to perform an IBP-reduction of the master integrals: This feature is not part FEYNCALC but will be offered in near future via FEYNHELPERS.

3. Installation

The fastest and most convenient way to install FEYNCALC is to use the automatic installer by evaluating

```
In[1]:= Import@"https://raw.githubusercontent.com/
        FeynCalc/feyncalc/master/install.m"
        InstallFeynCalc[]
```

on a freshly started MATHEMATICA kernel. All versions of MATHEMATICA from 10.0 upwards are supported. The code above will install the stable version of the package. The development version — with potential bugs but also the newest features, can be obtained by setting the option `InstallFeynCalcDevelopmentVersion` of `InstallFeynCalc` to `True`. In the case of internet connection problems one can also install the package manually. For further instructions we refer to the section “Installation” of the package manual. The source code of FEYNCALC can be obtained from https://github.com/FeynCalc/feyncalc.

4. Topologies and loop integrals

At one loop, almost every calculation involving only integrals with quadratic propagators can be handled using the so-called Passarino–Veltman (PaVe) technique. By considering the most generic basis of tensor structures for the given rank made of metric tensors and external momenta, each occurring tensor integral can be reduced to a linear combination of scalar functions. These quantities are known as PaVe coefficient or scalar functions and can be straightforwardly evaluated analytically or numerically. Rewriting the amplitude in terms of these functions usually concludes the loop-related part of the calculation. PaVe-Reduction is implemented in numerous codes including FEYNCALC — where the relevant routines are called `TID` and `PaVeReduce`. The conceptual simplicity behind the PaVe technology and the availability of reliable numerical codes (cf. e.g. refs. [85–91]) make it the default choice for the majority of practitioners. Unfortunately, the complexity of multiscale multiloop integrals does not allow one to apply these methods beyond one-loop with the same ease and efficiency.

[3] https://magv.github.io/alibrary/.

[4] The tools Q2E, EXP and CALC are not public, but can be obtained upon request, cf. http://sfb-tr9.ttp.kit.edu/software/html/q2eexp.html

Table 1
Some differences between tools for automatizing multiloop amplitude evaluation.

Feature	FEYNCALC	ALIBRARY	TAPIR	FEAMGEN.JL	HEPLIB
Language	MATHEMATICA	MATHEMATICA	PYTHON	JULIA	C++
Diagram generation	FEYNARTS, QGRAF (FEYNHELPERS)	QGRAF	QGRAF	QGRAF	QGRAF
Diagram visualization	FEYNARTS, TIKZ-FEYNMAN (FEYNHELPERS)	TIKZ, GRAPHVIZ	TIKZ-FEYNMAN	TIKZ-FEYNMAN	TIKZ-FEYNMAN
Topology mappings	yes	FEYNSON	yes	yes	yes
Partial fractioning	yes	yes	yes	no	yes
Dirac algebra except traces	yes	no	no	yes	yes
Color algebra	yes	COLOR.H	COLOR.H	yes	yes
Tensor reduction	yes	no	no	no	yes
Uses FORM	no	yes	yes	yes	yes
Interface to IBP codes	FEYNHELPERS	yes	no	yes	yes
New models	FEYNRULES	by hand	UFO	UFO	by hand

Instead, it is customary to treat each integral family on its own, by first reducing all relevant integrals to a smaller set of master integrals and then calculating those using suitable analytic or numerical techniques.

To that aim it is necessary to have code(s) that can (i) introduce integral families from analyzing the propagators present in the amplitude, (ii) minimize the number of integral families by finding possible mappings between them, (iii) ensure that the set of propagators in each family forms a basis and, if necessary, (iv) perform tensor reduction or (v) partial fraction decomposition. Upon completing these steps, one should obtain a list of integral topologies present in the amplitude and the corresponding loop integrals belonging to these topologies. This information can be then passed to an IBP-reduction program such as FIRE, KIRA, LITERED, REDUZE [92,93], AZURITE [94] etc. that will minimize the number of integrals that need to be evaluated. In the following we would like to focus on explaining how these five steps can be performed with the aid of FEYNCALC.

4.1. Three main building blocks

The three main building blocks of FEYNCALC's new multiloop capabilities are the symbols `FCTopology` and `GLI` as well as the routine `FCFeynmanPrepare`. In this context `FCTopology` represents an integral family that consists of propagators forming a basis. The syntax reads

```
In[1]:= FCTopology[id, {propagators}, {loop momenta}, {
        external momenta}, {kinematics}, {extra}]
```

where the first argument denotes the name of the topology, the second argument enumerates the propagators, while the third and fourth lists contain names of loop and external momenta. `id` can be a string or a symbol, while `{propagators}` must be a list of `FAD`, `SFAD`, or `GFAD` propagator containers.

Kinematic constraints (e.g. specific values of scalar products made of external momenta) can be specified in the fifth argument, while the last argument may be used to incorporate some addition information (e.g. that this topology is a subtopology of a larger topology). A simple `FCTopology` example would be

```
In[1]:= FCTopology[topo1, {SFAD[p1], SFAD[p2], SFAD[q -
        p1 - p2], SFAD[q - p2], SFAD[q - p1]}, {p1, p2},
        {q}, {Hold[SPD][q] -> qq}, {}]
```

that describes an on-shell 2-loop propagator-type topology. Notice that the construction `Hold[SPD][q]` prevents the scalar product q^2 from being immediately evaluated in case it has already been assigned a value via `SPD[q] = xyz`.

Having defined an integral family, we can also introduce integrals belonging to it. To that aim FEYNCALC uses `GLI`s (a shortcut for "Generic Loop Integral") defined as

```
In[1]:= GLI[id, {powers}]
```

where the first argument denotes the family name (and must match the `id` of the corresponding `FCTopology`) and the second contains powers of the involved propagators. The propagator powers must integers, e.g.

```
In[1]:= GLI[topo1, {1,1,1,-2,2}]
```

but most routines can also deal with symbolic powers. Similar notation is used in many other software packages related to multiloop calculations, such as FIRE, LITERED, KIRA or PYSECDEC.

The function `FCFeynmanPrepare` is used to derive the Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$ for the given topology or set of master integrals. It can be invoked, not only on `FCTopology` or `GLI` objects, but also on integrals using explicit propagator representation via `FeynAmpDenominator`. The underlying algorithm is based on the code UF.M [95] that is used in FIRE and FIESTA [96,97]. The $\mathcal{U}$ and $\mathcal{F}$ polynomials encode numerous properties of the related topologies or master integrals (cf. ref. [98] for an extensive review) and can be used to derive one-to-one mappings between those objects.

`FCFeynmanPrepare` is not limited to the derivation of Symanzik polynomials. It can also calculate other useful quantities such as the matrix M with $\mathcal{U} = \det M$ or J and Q^μ as in $\mathcal{F} = \det M(QM^{-1}Q - J)$. Moreover, the routine is capable of dealing with both Minkowskian and Euclidean integrals. To avoid any confusion, let us stress that with "Euclidean" we explicitly mean integrals defined in the flat space with the Euclidean metric signature $g_E^{\mu\nu} = \text{diag}(1,1,1,1)$. To that aim one needs to set the option `"Euclidean"` to `True`.

4.2. Basic operations

The "old" loop-related FEYNCALC functions such as `TID`, `FDS` or `PaVeReduce` are designed to work with loop integrals in the propagator representation. Therefore, upon introducing the new `GLI`-representation it became necessary to add a large set of new routines that accept input containing `GLI`- and `FCTopology`-symbols. However, in some cases the existing routines were just modified to be able to deal with the new objects.

One of the simplest manipulations applicable to a `GLI` is the conversion into the propagator representation. This can be done using `FCLoopFromGLI`. This function requires two arguments, which are a `GLI` integral and the corresponding topology in the form of an `FCTopology`. Both arguments can be also lists, which is useful when processing multiple integrals in one go.

Since a topology has a rather involved syntax, it can be validated using `FCLoopValidTopologyQ`. This helps to avoid user errors when entering topologies by hand or converting them from the output of other tools. A list of all kinematic invariants present in a topology (or a list thereof) can be obtained with the aid of `FCLoopGetKinematicInvariants`.

A priori, the set of propagators contained in an `FCTopology` does not necessarily have to form a basis. However, since many loop-related manipulations make sense only when working with a proper propagator basis, FEYNCALC provides tools to verify this property. These are `FCLoopBasisOverdeterminedQ` and `FCLoopBasisIncompleteQ`, which can tell whether the given set of propagators is overdetermined or incomplete. In the latter case the basis can be automatically completed with suitable propagators using `FCLoopBasisFindCompletion`.

Table 2
Differences between different computer codes in the assumed $i\eta$-prescription for propagators. Expressions in green correspond to input yielding a correct imaginary part, while red terms will create inconsistencies.

Symbolic expression	Meaning in FIESTA	Meaning in FC / PYSECDEC
$[(p+q)^2]^{-1}$	$[(p+q)^2 - i\eta]^{-1}$	$[(p+q)^2 + i\eta]^{-1}$
$[-(p+q)^2]^{-1}$	$[-(p+q)^2 - i\eta]^{-1}$	$[-(p+q)^2 + i\eta]^{-1}$
$[(p+q)^2 - m^2]^{-1}$	$[(p+q)^2 - m^2 - i\eta]^{-1}$	$[(p+q)^2 - m^2 + i\eta]^{-1}$
$[-(p+q)^2 + m^2]^{-1}$	$[-(p+q)^2 + m^2 - i\eta]^{-1}$	$[-(p+q)^2 + m^2 + i\eta]^{-1}$

Integrals containing too many propagators must be subjected to partial fraction decomposition. Although FEYNCALC's `ApartFF` can now handle `GLI`s, in the context of loop calculations done with `FORM`, one would usually like to get explicit replacement rules that rewrite a product of overdetermined denominators into a linear combination of terms with fewer denominators. To cover that case we introduced a new routine called `FCLoopCreatePartialFractioningRules` that will generate such rules and return a list of new topologies appearing on the right-hand side of the replacement rule.

Sometimes one might be interested in selecting particular topologies from a large list under the condition that those appear in the given loop integrals. To this aim we can use `FCLoopSelectTopology`, which is also employed by numerous high-level FEYNCALC functions.

Differentiation of loop integrals with respect to vectors (similar to what can be achieved with `FourDivergence`) or scalars is implemented in `FCLoopGLIDifferentiate`. This can be used e.g. when deriving symbolic IBP relations, systems of differential equations or performing asymptotic expansions [99]. The routine is therefore very similar to LITERED's `Dinv`. Notice that to have a proper differential equation, the differentiated loop integrals still require an IBP reduction. In this sense the function does not give one the final equation right away but merely performs one important step in the derivation thereof. Equally, it does not implement any methods to solve the resulting equation.

When doing asymptotic expansions one normally would like to attach a particular scaling parameter (say λ) to specific masses or momenta in the topology and expand the loop integrals in λ to the given order. The former can be accomplished via `FCLoopAddScalingParameter` while for the latter one would use `FCLoopGLIExpand`.

An important point to keep in mind when working with loop integrals is the $i\eta$-prescription in the propagators. By default, FEYNCALC uses the standard convention, where a Minkowskian propagator is understood to be

$$[p^2 - m^2 + i\eta]^{-1} \tag{1}$$

However, an alternative prescription used e.g. in FIESTA is to pull out an overall minus sign, which leads to

$$[-p^2 + m^2 - i\eta]^{-1} \tag{2}$$

Notice that this propagator is still Minkowskian, just written in different way as compared to Eq. (1). For the sake of completeness, we list the relevant conventions for FEYNCALC, FIESTA and PYSECDEC in Table 2. Notice that when using `SFAD` and `GFAD` shortcuts to enter loop integral propagators, FEYNCALC will explicitly display $i\eta$, unless the global variable `$FCShowIEta` has been set to `$False`.

FEYNCALC can convert topologies to the convention of Eq. (2) via the function `FCLoopSwitchEtaSigns`. `FCLoopGetEtaSigns` is used internally to ensure the consistency of the $i\eta$-prescription among propagators present in integrals and topologies.

4.3. Topology identification

Given a multiloop amplitude expressed as a linear combination of scalar loop integrals, one usually wants to reduce these integrals to a basis of master integrals. This procedure, usually referred to as the IBP-reduction, is not a mere convenience, but a strict necessity. While the number of unreduced loop integrals can easily go into hundreds of thousands or even millions, the number of master integrals often lies between $\mathcal{O}(100)$ and $\mathcal{O}(1000)$ for two- and three-loop calculations that are feasible with modern techniques. In practice, the reduction often turns out to be one of the main bottlenecks in analytic calculations and it is imperative to organize it in the most efficient way. For example, reducing each loop integral separately would be a waste of computational resources that should be avoided.

A better approach is to organize integrals into families and then do the reduction for each family. An integral family or a topology is defined as a set of linear independent propagators plus additional kinematic constraints such as values of masses or external momenta squared. It goes without saying that the number of families should better be as small as possible, otherwise one would be wasting computer time and resources. A caveat, however, lies in the fact that dimensionally regularized loop integrals are invariant under shifts of loop momenta. Hence, two integrals that look very different might still belong to the same family. Also, two topologies that seem to be quite distinct could represent the same quantity modulo momentum shifts.

Such ambiguities can be avoided using a procedure called topology identification or minimization, where the set of all loop integral topologies present in the amplitude is mapped to a smaller set of topologies independent of each other. Most algorithms for solving this task either consider graph representations of Feynman diagrams and amplitudes or analyze the propagators present in the amplitude. In the former case the initial problem of finding mappings between different topologies is converted into the requirement to find subgraph isomorphisms. When working with symbolic propagators one is mainly interested in finding a representation that removes the invariance under momentum shifts and generates unique expressions that can be directly compared with each other. Of course, a brute-force enumeration of all possible momentum shifts is also possible, although mostly prohibitively expensive performance-wise. A purely graph-based algorithm is implemented e.g. in the C++ programs Q2E/EXP, while the MATHEMATICA package TOPOID [100] and the generator of optimized IBP identities NEATIBP [101] look only at propagators of the loop integrals. Hybrid approaches are realized e.g. in REDUZE, FEYNSON, TAPIR or PYSECDEC.

FEYNCALC follows a purely propagator-based approached by using the so-called Pak algorithm [102] — a special prescription for comparing topologies or integrals with each other invented by Alexey Pak. Our implementation heavily relies on the ideas and tricks that can be found in the doctoral thesis of Jens Hoff [103] and their realization in Hoff's program TOPOID.[5]

The starting point for Pak's algorithm is the naive observation that in the Feynman parametric representation of loop integrals (or topologies) the loop momenta are integrated out. Hence, the shift invariance seems to be gone. Unfortunately, there still remains a residual ambiguity, which is related to the relabeling of Feynman parameters $x_i \leftrightarrow x_j$. Pak's insight was to introduce a canonical way to label x_i for the given combination of $\mathcal{U}$ and $\mathcal{F}$ polynomials. Then, given two canonically ordered characteristic polynomials $\mathcal{P}_1 \equiv \mathcal{U}_1 \times \mathcal{F}_1$ and $\mathcal{P}_2 \equiv \mathcal{U}_2 \times \mathcal{F}_2$, it is guaranteed that for identical integrals or topologies we will find $\mathcal{P}_1 = \mathcal{P}_2$. This property is the corner stone for FEYNCALC's functionality of finding one-to-one mappings between topologies or master integrals.

When implementing this technique we introduced a number of auxiliary functions that return the necessary building blocks for applying Pak's algorithm. For example, `FCLoopToPakForm` can be used to generate the canonically ordered characteristic polynomial $\mathcal{P}$ from the given propagator representation, while `FCLoopPakOrder` can apply Pak ordering to any polynomial.

[5] https://github.com/thejensemann/TopoID.

A nice property of $\mathcal{P}$ is that it can be used to detect scaleless integrals that vanish in dimensional regularization. The description of the underlying algorithm can be found in Section 2.3 of ref. [100]. In FEYNCALC the scalefulness property can be checked using the functions `FCLoopPakScalelessQ` (for characteristic polynomials) and `FCLoopScalelessQ` (for loop integrals).

The workflow envisioned in FEYNCALC begins with applying `FCLoopFindTopologies` to the given amplitude. The function will return a list of the form `{amp, topos}`, with `amp` being the amplitude rewritten in a form suitable for further processing and `topos` constituting a list of all distinct sets of propagator denominators. In `amp` these sets are grouped into `GLI`s, while `topos` is made of `FCTopology` objects. In the next step one should get rid of tensor integrals, unless this has already been done by applying suitable projectors.

To this aim one can use the function `FCLoopTensorReduce` — which still uses `Tdec` as back-end, but is optimized for the representation of the amplitude generated by `FCLoopIdentifyTopologies`. Alternatively, one could also apply `FCMultiLoopTID` to the amplitude before running `FCLoopIdentifyTopologies`, but for performance reasons we do not recommend this. It is also worth mentioning that the old code used in `Tdec`[6] for recognizing symmetries between tensor reduction coefficients has been replaced with the algorithm described in ref. [102]. On selected examples the new symmetrizer can lead to systems of linear equations being much smaller as compared to FEYNCALC 9.3.1. Still, for higher rank tensor integrals MATHEMATICA's capabilities might be insufficient to solve the linear system in a reasonable amount of time. This issue can be worked around using the new version of FEYNHELPERS, which allows `Tdec` to use FERMAT [105] as a solver back-end.

The one-to-one mappings between topologies can be revealed by applying `FCLoopFindTopologyMappings` to the `topos` list. Every mapping rule between two topologies contains of a list of momentum shifts that convert the propagators of the first topology into those of the second topology and should be also applied to all numerators multiplying the first topology. It is also possible to map the given topologies onto a specific set of selected topologies (e.g. to facilitate comparisons to other calculations) using the option `PreferredTopologies`.

Due to the nature of Pak's algorithm `FCLoopFindTopologyMappings` can only find relations between topologies that contain exactly the same number of propagators. Since not all topologies appearing in the amplitude normally have the full set of propagators required to form a basis, this often leaves some room for mapping smaller topologies into larger ones. Here with "larger topologies" we mean both incomplete topologies with a larger number of propagators as well as the so-called supertopologies that have a complete propagator basis. In FEYNCALC one can deal with this situation by first identifying all nonvanishing subtopologies of the given topology via `FCLoopFindSubtopologies`. Then, one can try to find mappings between those subtopologies and the actual smaller topologies appearing in the amplitude. The subtopologies contain a special marker that relates them to the parent topology, so that `FCLoopFindTopologyMappings` knows how to generate correct mappings pointing to the original topology.

Once the final set of topologies has been sufficiently minimized, one can apply the generated mapping rules to the full amplitude with the aid of `FCLoopApplyTopologyMappings`. In fact, this routine will also rewrite all scalar products in terms of invert propagators and combine them with the existing propagator denominators, so that the resulting amplitude will appear as a linear combination of different `GLI`s. In the background this high-level function uses the auxiliary routines `FCLoopCreateRulesToGLI` and `FCLoopCreateRuleGLIToGLI`. The output of `FCLoopApplyTopologyMappings` is, in principle, suitable for the subsequent IBP reduction. The process of converting the occurring loop integrals and the list of final topologies into run cards for tools such as FIRE and KIRA can be easily automatized. The relevant code is already part of FEYNHELPERS that will be presented elsewhere.

5. Master integrals

Unless one is trying to perform a cutting-edge calculation[7] or lacks access to sufficient computational resources, the IBP reduction usually goes through without much additional effort. Even when some fine-tuning is needed, this usually amounts to playing with the configuration files of the respective programs, resubmitting jobs running on a cluster or possibly asking the developers for an advice. Once all reduction tasks have been completed, one is normally left with a list of master integrals from different integral topologies.

Depending on the tool that was used to perform the reduction, it may be necessary to check whether all of these integrals are indeed distinct. Given that identical masters can have rather different propagator representations, this task should be better performed in an algorithmic fashion. FEYNCALC can make use of the built-in Pak's method to reveal all one-to-one mappings between the master integrals. The corresponding function is called `FCLoopFindIntegralMappings` and has been modeled after the routine `FindRules` in FIRE. Using the option `PreferredIntegrals` one can choose a list of preferred master integrals to be mapped onto. This also works for factorizing integrals, which can be entered as products of `GLI`s.

Graphical representations of master integrals serve as an important tool to better understand the obtained results and relate them to the calculations that has already been done in the literature. The most common visualization method is to relate the propagators and the flow of loop momenta inside the integral to graphs made of directed edges. These edges can be styled to account for the types of propagators and their masses. For example, massless propagators are usually plotted as dashed or dotted lines, while massive propagators are shown as solid lines of different colors. If the propagator appears squared, this can be hinted using a dot or a cross.

While switching from a graph to a propagator representation for the given integral (or topology) is a trivial step, the converse is not true. The construction of graphs for arbitrary integrals can be tricky and requires both care and effort. Some publicly available tools such as AZURITE, PLANARITYTEST [106] or LITERED allow for automatizing this task to some extent. In FEYNCALC, the corresponding function is called `FCLoopIntegralToGraph`. In the case of a successful reconstruction, it returns a directed MATHEMATICA graph as well as the line momenta running through the edges and some additional information. Although the graph can be directly shown using the built-in MATHEMATICA function `GraphPlot`, one should better use FEYNCALC's `FCLoopGraphPlot` which takes options for styling the edges and making the output look more similar to what is usually seen in the literature. However, it should be noted that due to multiple problems that older MATHEMATICA versions have with the visualizations of graphs, `FCLoopGraphPlot` requires at least MATHEMATICA 11.0, while best results can be expected with the version 12.2 or newer. We would also like to stress that the graph obtained with `FCLoopIntegralToGraph` can also (upon some minimal adjustments) be plotted using other suitable software such as GRAPHVIZ.

One of the advantages of having master integrals shown as graphs is that one can readily assess possible cuts via visual examination. In this context we understand "cutting" as the process of sending the cut propagators on-shell so that pictorially the graph splits into two graphs. However, analyzing dozens or even hundreds of graphs by eye can still be tedious and prone to human errors. To streamline this process FEYNCALC offers `FCGraphCuttableQ`, which can decide whether the given graph can be cut without touching the specified lines. This is relevant

[6] `Tdec` is an auxiliary routine generating tensor decomposition formulas for generic multiloop integrals that can be used in FEYNCALC or FORM [104].

[7] Of course, if the given calculation involves enough masses and/or legs, even the reduction of two-loop amplitudes can quickly become unfeasible.

e.g. for heavy particles in the loops that kinematically cannot go on-shell. This way one can readily determine whether the given master integrals can develop an imaginary part or not. A more generic routine is offered under the name `FCGraphFindPath`. Its task is to determine, whether the given graph can be traversed by starting and finishing at one of the external edges. The internal edges can be assigned weights 1 or -1, with the latter meaning that this edge cannot be passed.

Having dealt with the problem of obtaining a set of unique master integrals and visualizing them, it is fair to ask whether FeynCalc can also be useful for evaluating the master integrals. As far as numerical evaluation is concerned, the FeynHelpers interface (to be presented in a future publication) makes it easy to generate ready-to-use FIESTA or pySecDec scripts for evaluating the given `GLI`s. Analytic results are of course much more difficult to obtain. Calculating loop integrals in this fashion often involves trying or even combining different techniques available on the market (cf. e.g. refs. [4,5]) in the hope that an integral, that is intractable using method A may turn out to be easy when attacked with method B. While FeynCalc obviously cannot deliver analytic solutions upon pressing a button, it nevertheless offers a set of handy routines that facilitate common steps required for some calculational methods.

As far as the derivation of the Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$ is concerned, the previously mentioned function `FCFeynmanPrepare` can readily generate those expressions. If one is interested in evaluating the master integral via a direct analytic integration of its Feynman parametric representation, the more useful routine would be `FCFeynmanParametrize`. Notice that this function supports both quadratic and eikonal propagators and can also deal with Euclidean or tensor integrals. Cartesian integrals living in $D-1$ dimensions are equally supported.

In general, it is very difficult to carry out all Feynman parametric integrations while keeping the full dependence on the dimensional regulator ε. For simpler integrals this can be often achieved by first joining specific subsets of propagators before combining the rest in the final integrand. This trick may often result in a greater freedom when exploiting the Cheng–Wu theorem [107] and trying to find a working sequence of integrations. To this aim FeynCalc offers `FCFeynmanParameterJoin`, which makes the unification of different propagators simple and straightforward. Its output can be then passed to `FCFeynmanParametrize`, thus obtaining the final integrand depending on the introduced sets of Feynman parameters x_i, y_i, z_i etc.

The applicability of the Cheng–Wu theorem can be readily checked via `FCFeynmanProjectiveQ`. If, for some reason, the integral turns out not to be projective, it can be rendered projective using `FCFeynmanProjectivize` by automatically performing a projective transformation.

If one insists on integrating Feynman parameters order by order in ε, one should keep in mind that naively expanding the integrand in the dimensional regulator may introduce divergences in the Feynman parameters, which is clearly undesirable. One possible solution to this problem involves the so-called analytic regularization that was developed in refs. [108–110] and implemented in the Maple package HyperInt [108]. In a nutshell, when expanding an integrand that has been treated using this technique, all ε poles become explicit so that the integrations in x_i remain finite. Unlike sector decomposition, analytic regularization is guaranteed not to spoil the linear reducibility property of the integrand. In FeynCalc analytic regularization is implemented using the functions `FCFeynmanFindDivergences` and `FCFeynmanRegularizeDivergence`, that were inspired by HyperInt's `findDivergence` and `dimregPartial`.

Nowadays, the method of differential equations [111–116] belongs to the most popular and efficient techniques for calculating large numbers of master integrals analytically or numerically. The discovery of the canonical form [117,118] and a rapid advance in the software for automatically finding such forms was very beneficial for the field of multiloop calculations. When using tools such as Fuchsia [119], CANONICA [120,121], Libra [122,123], epsilon [124], Initial [125,126] etc. one is often confronted with the necessity to perform a change of variables e.g. for rationalizing square roots appearing at intermediate stages.[8] To automatize this step FeynCalc offers a function called `FCDiffEqChangeVariables`. At the moment only differential equations of one variable are supported. Given the old variable x, the new variable $y(x)$ as well as the inverse relation $x(y)$, this routine eliminates x in favor of y in the given matrix. This can be $\mathcal{A}$ from the pre-canonical form of the differential equation $F' = \mathcal{A}F$, but also $\mathcal{B}$ from the canonical form $G' = \varepsilon\mathcal{B}G$ or just the transformation matrix $\mathcal{T}$ with $F = \mathcal{T}G$. Notice that in the case of $\mathcal{T}$ one should disable the inclusion of the prefactor $1/f'(y)$ to avoid incorrect results. This is done by setting the option `Prefactor` to `False`.

Having obtained a canonical form using one of the existing tools, one usually starts constructing the solution to the system order by order in ε. Here one can make use of `FCDiffEqSolve` that can quickly generate such expressions written in terms of `FCIteratedIntegral` objects. The latter can be regarded as a placeholder for expressions of the type

$$\int_a^b dx f(x), \tag{3}$$

where $f(x)$ can be an iterated integral itself. This way nested integrals can be represented in FeynCalc by wrapping new `FCIteratedIntegral` heads around the existing ones.

To facilitate the evaluation of such integrals, the rational functions involved are transformed into a special representation called `FCPartialFractionForm`. The main idea is to write expressions of the form

$$n + \frac{f_1}{[x-r_1]^{p_1}} + \frac{f_2}{[x-r_2]^{p_2}} + \ldots \tag{4}$$

as `FCPartialFractionForm[n,{{f1,x-r1,p1},{f2,x-r2,p2},...},x]`. From here one can easily rewrite `FCIteratedIntegral` objects in terms of Harmonic or Goncharov polylogarithms [128–130]. This is done using `FCIteratedIntegralEvaluate` with the result containing `FCGPL` symbols. The conversion of rational functions to this representation is handled by `ToFCPartialFractionForm`.

Notice that for the time being, `FCGPL`s are mere placeholders. More GPL-related routines are expected to be added in future versions of FeynCalc. Our goal is to have some minimal implementation of core symbolic properties of GPLs that can be used by the related FeynCalc functions (mainly for computing loop integrals) without the need to employ any external packages. This way we can ensure that there will not be any unwanted side effects that often arise when having multiple packages loaded on the same kernel or when using packages that have not yet been made 100% compatible with the most recent Mathematica version.[9]

Of course, for extensive manipulations of multiple polylogarithms the users should resort to special codes such as HPL [131,132], PolyLogTools [133] or MPL [134] that have been developed over the years and are well established in the field.

8 The transformations intended to remove such square roots can be automatically obtained using RationalizeRoots [127].

9 For example, PolyLogTools currently relies on the Combinatorica library that is being deprecated since Mathematica 10 and generates multiple warnings when loaded on version 13.3 or 14.0. The HPL package that is equally required by PolyLogTools is not being actively maintained since years and gets patched on the fly whenever it is loaded by PolyLogTools. Then, MPL is a Maple package and hence cannot be used directly with Mathematica.

6. Features and improvements unrelated to multiloop calculations

6.1. Improved color algebra simplifications

In the past, FeynCalc was often unable to simplify various SU(N) color algebraic expressions using `SUNSimplify` and `SUNTrace`. Sometimes chaining multiple instances of the two routines with different options would do the trick, but such workarounds were different to find and far from being obvious to users.

To improve on this situation, in FeynCalc 10 the function `SUNSimplify` was rewritten from scratch. The new version implements a much larger number of color algebraic relations, while the new code is easier to maintain and extend. Notice that the evaluation of the color trace is now handled in a manner similar to what is done in `DiracSimplify`. By default, an `SUNTrace` object remains unevaluated, unless the option `SUNTraceEvaluate` is set to `True`. However, the more convenient way to evaluate such expressions is to use `SUNSimplify`. The default value of the `SUNTraceEvaluate` option in `SUNSimplify` is set to `Automatic`. This means that if a trace can be simplified without naively rewriting everything in terms of structure constants, the function will do so. Setting this option to `False` will leave all traces untouched, while `True` means that the user explicitly wants to eliminate the traces in favor of `SUNF` and `SUND` symbols.

6.2. Passarino–Veltman functions

As a package, that was originally developed with one-loop calculations in mind, FeynCalc is of course equipped with symbols representing Passarino–Veltman functions and a set of routines for working with them. One particular shortcoming related to this functionality that became obvious in the past few years, was FeynCalc's ignorance of many symmetry relations between `PaVe` functions. This way some results looked longer and more complicated than they actually should have been and certain cancellations did not take place.

In FeynCalc 10 we tried to add all symmetries up to rank 10 for B-functions, rank 9 for C-functions, rank 8 for D-functions, rank 7 for E-functions and rank 6 for F-functions. The corresponding files are located inside the directory `Tables/PaVeSymmetries` and can be (if needed) extended to even higher ranks. Whenever the user enters `PaVe` functions, `PaVeOrder` will automatically reorder their arguments in a canonical way, unless the option `PaVeAutoOrder` has been explicitly set to `False`.

Also, the deprecated `OneLoop` routine offered a functionality that was tricky to reproduce using other functions: The ability to simplify IR-finite expressions involving `PaVe` functions by analyzing their UV-poles and expanding the D-dependent prefactors accordingly so that the expression becomes $\mathcal{O}(\varepsilon^0)$. To improve on this, we added `PaVeLimitTo4` which does exactly that. Notice that the absence of IR-poles is assumed but not explicitly checked, meaning that it is the user's duty to ensure this condition's validity.

6.3. Lagrangians and operators

In the course of our ongoing work to improve the usefulness of FeynCalc for nonrelativistic calculations, we extended the functionality of the package for manipulating Lagrangians to support Cartesian nabla operators. Supplementing the already existing symbols `LeftPartialD` ($\sim \overleftarrow{D}^\mu$), `RightPartialD` ($\sim \overrightarrow{D}^\mu$), `LeftRightPartialD` ($\sim \overleftrightarrow{D}^\mu$) and `LeftRightPartialD2` ($\sim \overleftrightarrow{D}^2$) we now also have `LeftNablaD` ($\sim \overleftarrow{\nabla}^i$), `RightNablaD` ($\sim \overrightarrow{\nabla}^\mu$), `LeftRightNablaD` ($\sim \overleftrightarrow{\nabla}^\mu$) and `LeftRightNablaD2` ($\sim \overleftrightarrow{\nabla}^2$).

Notice that although one still cannot use `FeynRule` to derive Feynman rules for nonrelativistic operators, other useful routines such as `ExpandPartialD` and `ExplicitPartialD` can now deal with nabla operators or gauge covariant derivatives with Cartesian indices (i.e. D^i from $D^\mu = (D^0, D^i)$).

Another useful addition to this part of FeynCalc's capabilities is `ShiftPartialD`, which allows the user to reshuffle derivatives in specific operators by applying integration by parts on the Lagrangian level. In this case the surface terms are always assumed to vanish.

Last but not least, in order to further facilitate the process of writing custom functions working with FeynCalc symbols (e.g. for deriving Feynman rules in the spirit of Appendix C from [135]), version 10 also features `FCTripleProduct` as a shortcut for vector products $(\vec{a} \times \vec{b}) \cdot \vec{c}$ as well as two routines for extracting all free or dummy indices in the given expression. They are called `FCGetFreeIndices` and `FCGetDummyIndices` respectively.

6.4. Dirac algebra

It is now possible to apply Gordon identities to suitable spinor chains by means of `GordonSimplify`. The function works both in 4 and D dimensions, while the option `Select` allows to choose whether one wants to trade the right-handed projector P_R for the left-handed P_L one or vice versa.

Furthermore, the calculation of Dirac traces in the Larin [136] scheme now proceeds according to the so-called Moch-Vermaseren-Vogt [137] formula, which greatly improves the computational efficiency as compared to the previous implementation.

The code for the evaluation of some special spinor chains such as $\bar{v}(p)\gamma^5 v(p)$, $\bar{u}(p)u(p)$ or $\bar{v}(p)v(p)$ was moved from `DiracSimplify` to a dedicated routine called `SpinorChainEvaluate`. Setting the same-named option of `DiracSimplify` to `False` will prevent FeynCalc from replacing such objects with their explicit values — which can be useful for certain types of calculations.

Unfortunately, FeynCalc still does not support the spinor-helicity formalism [138–144], which constitutes a much more efficient way to deal with fermions, especially in the massless case. This feature remains on our to-do list.

6.5. Convenience functions for research activities

Some of the functions introduced in FeynCalc 10 are not directly related to the evaluation of amplitudes or loop integrals but rather belong to the category of the so-called convenience routines. One of them is called `FCMatchSolve` and has been developed to automatize the determination of renormalization constants, matching coefficients and other parameters. To this aim, for a given expression (e.g. the difference of two amplitudes or the sum of some diagrams and the corresponding counterterms), one first needs to collect all unique structures (e.g. Dirac chains, color factors, ε, α_s etc.). Then, one can pass this expression to `FCMatchSolve` together with the list of symbols that should be regarded as fixed variables. In this case the function regards all other variables as free parameters and tries to choose them such, that the input expression vanishes. In practice, this approach turns out to be more efficient and robust than using `Collect` and `Solve`.

Another common task in particle phenomenology is the numerical evaluation of the final analytic expressions for cross sections, decay rates, matching coefficients and other experimentally accessible parameters. Comparing such quantities to the literature or to the results of peers can be nontrivial for several reasons: Firstly, contrary to symbolic expressions, the comparison will not be exact, but rather up to a given number of n significant digits. Second, when finding disagreement between two large expressions involving numbers of different origin, one would often want to identify terms that agree with less significant digits than required, rather than merely state the lack of numerical agreement. Using `FCCompareNumbers` one can streamline the task of comparing two numerical or semi-numerical expressions, while retaining full control over the number of significant digits required. Again, even though a similar result could be achieved using custom codes, FCCompareNumbers is an attempt at saving time by automating trivial operations and avoiding the most common pitfalls of manual evaluation.

Putting the often long and complicated analytic expressions obtained in a multiloop calculation into a proper form suitable for a publication, can be regarded as an art of its own. When using MATHEMATICA for organizing the expressions and converting them into LaTeX, one is often faced with the problem that sums of terms are not ordered in the way one would want them to. This is because MATHEMATICA's Times and Plus functions sort terms using an internal canonical ordering that does not necessarily agree with one's aesthetic preferences. To remedy this, FEYNCALC now comes with a function called FCToTeXReorder that first converts Times- and Plus-type expressions into nested lists of the form {a,b,...,Plus} and {a,b,...,Times} respectively. Terms inside those lists can then be grouped and ordered according to the user's preferences, using custom factoring and sorting functions. The intermediate result of such manipulations can be readily previewed with FCToTeXPreviewTermOrder. Once the expressions have been brought into a suitable form, one can directly apply the built-in TeXForm command to the output of FCToTeXPreviewTermOrder and then copy the generated LaTeX code into the source file of the publication.

6.6. Tensors with light-cone components

In many QFT calculations (especially those involving highly energetic particles) it is natural to decompose Lorentz tensors into components along two light-like reference vectors n and $\bar{n}$ satisfying

$$n^2 = \bar{n}^2 = 0, \quad n \cdot \bar{n} = 2. \tag{5}$$

For example, a four-vector can be then written as

$$p^\mu = \frac{\bar{n}^\mu}{2}(p \cdot n) + \frac{n^\mu}{2}(p \cdot \bar{n}) + p_\perp^\mu \equiv p_+^\mu + p_-^\mu + p_\perp^\mu, \tag{6}$$

with the perpendicular component being defined as the difference between the full vector and the sum of the plus and minus components

$$p_\perp^\mu \equiv p^\mu - p_+^\mu - p_\perp^\mu = p^\mu - \frac{\bar{n}^\mu}{2}(p \cdot n) - \frac{n^\mu}{2}(p \cdot \bar{n}). \tag{7}$$

To facilitate such calculations using FEYNCALC, version 10 of the package introduces special symbols for defining light-like reference vectors as well as additional quantities specifying the plus, minus and perpendicular components of Lorentz tensors. First of all, one has to tell FEYNCALC, which symbols represent n and $\bar{n}$ by assigning the corresponding values to $FCDefaultLightConeVectorN and $FCDefaultLightConeVectorNB. In addition to that, one should also implement the constraints from Eq. (5) e.g. as in

```
In[1]:= FCClearScalarProducts[]
        ScalarProduct[n,n] = 0; ScalarProduct[nb,nb] = 0;
              ScalarProduct[n,nb] = 2;
```

After these preliminary steps one can start using new shortcuts for the lightcone components such as FVPL[p,μ] for p_+^μ or SPLR[p,q], $(p \cdot q)_\perp$ etc. In the case of plus and minus components, FEYNCALC would insert explicit expressions constructed from the full tensor contracted with n and $\bar{n}$ vectors. Perpendicular components are represented using the symbol LightConePerpendicularComponent, which takes a LorentzIndex or Momentum as first argument and requires Momentum[n] and Momentum[nb] for the remaining two arguments.

FEYNCALC can work with expressions involving light-cone components of vectors, metric tensors and scalar products in 4 or D dimensions. Dirac matrices can be also defined on the light-cone — with both DiracSimplify and DiracTrace able to simplify the corresponding expressions.

6.7. Up-to-date documentation using continuous integration

With this release we also address shortcomings of the FEYNCALC documentation: The lack of a proper manual in the form of a PDF file, new functions introduced but not documented, the absence of a novice-friendly tutorial. Technical issues forced us to rethink the whole concept behind the documentation of the package. We decided to stop maintaining the documentation sources in form of MATHEMATICA notebooks in favor of switching to text-based .m and markdown files. Using a modified version of J. Podkalicki's MATHEMATICA to Markdown converter M2MD[10] together with the PANDOC [145] document converter, we created a workflow, where .m and .md (markdown) files containing the whole documentation can be semi-automatically converted to HTML (for the online documentation) or LaTeX (for the PDF manual). The LaTeX-form of the manual is kept in a separate repository[11] and every change in the source files triggers an update of the public PDF file that can be readily downloaded[12] by anyone. This way, it is easy to keep both the online and PDF versions of the manual up to date without the need to update or modify their content manually. Furthermore, we also took care to automatically synchronize the descriptions of FEYNCALC symbols and functions in the documentation to the texts shown when looking up their usage information (e.g. as in ?FV or ?TID). We hope and believe that these changes will significantly improve the user experience of FEYNCALC and make the program more accessible to new users.

7. Examples

In this section we would like to draw reader's attention to several examples included with the package that make use of the new multiloop-related routines. Unlike old FEYNCALC codes, where one-loop amplitudes were always expressed in terms of Passarino–Veltman functions, the calculations presented below are carried out in a different way, where the resulting amplitudes are written in terms of GLIs belonging to previously identified integral families in the FCTopology notation.

7.1. Electron self-energy in massless QED at 2 loops

We start by generating the 3 required 2-loop diagrams in QED using FEYNARTS. Upon setting the electron mass to zero, we apply DiracSimplify to the amplitudes and then continue to the topology identification stage. Using FCLoopFindTopologies we find two distinct sets of propagators that can be fitted into two trial topogies. However, upon identifying all nonvanishing subtopologies by means of FCLoopFindSubtopologies, we can apply FCLoopFindTopologyMappings and map everything into one single topology.

The next step is to carry out the tensor reduction (FCLoopTensorReduce) and then apply mappings between trial topologies to the diagrams. This can be conveniently handled by FCLoopApplyTopologyMappings. In addition to the above steps this routine will also rewrite the scalar products involving loop momenta in terms of inverse denominators (in the GLI-notation) and bring the amplitudes into a form where they are expressed as linear combinations of various GLI integrals.

We omit the technicalities related to the IBP reduction (which can be also automatized using FEYNHELPERS) and use an already available reduction table to reduce everything to 3 master integrals. Two of them are, however, identical — which can be revealed via FCLoopFindIntegralMappings. Thus, we obtain the final result for the 2-loop electron self-energy in massless QED expressed in terms of 2 master integrals. Comparing this to the result given in Eq. 5.51 of ref. [146] we find full agreement, as expected.

7.2. Photon self-energy in massless QED at 2 loops

This example is almost identical to the previous one apart from the obvious fact that we need to generate another set of diagrams. The calcu-

[10] https://github.com/kubaPod/M2MD.
[11] https://github.com/FeynCalc/feyncalc-manual.
[12] https://github.com/FeynCalc/feyncalc-manual/releases/tag/dev-manual.

lation is carried out with full gauge dependence, even though ξ cancels in the final result. The obtained result given in terms of 2 master integrals can be compared in Eq. 5.18 of ref. [146].

7.3. Gluon self-energy in massless QCD at 2 loops

The gluon self-energy calculation shares many similarities with the previous examples but also requires some adjustments. First of all, due to the complexity of this calculation (18 diagrams) we use Feynman gauge. Then, we employ Lorentz and color projectors to extract the scalar self-energy function $\Pi(p^2)$ directly. Apart from these technicalities, the main steps of the calculation closely follow those of the two previous examples. Here we choose to insert explicit expressions for the 2 master integrals and compare the so-obtained results at $\mathcal{O}(\varepsilon^0)$ with the sum of Eqs. 6.10-6.11 in ref. [147], finding complete agreement.

7.4. Topology identification for $B_c \to \eta_c$ form factors at 2 loops

This examples demonstrate the usage of FeynCalc for the sole purpose of minimizing a set of topologies obtained from elsewhere (e.g. in FORM calculation). We refer to refs. [148,149] for more details on the underlying physics and the technical aspects of this calculation. The given set of 251 2-loop topologies with 3 external momenta and 3 scales contains not only quadratic but also eikonal as well as mixed quadratic-eikonal propagators (cf. Appendix C). We first deal with the mixed propagators by completing the square using the routine `FCLoopReplaceQuadraticEikonalPropagators`. Then we address topologies containing overdetermined sets of propagators by generating the corresponding partial fraction decomposition rules via `FCLoopCreatePartialFractioningRules`. Having uncovered all suitable topology mappings with the aid of `FCLoopFindTopologyMappings`, we add missing propagators needed to have a complete basis in each topology (`FCLoopBasisFindCompletion`) and generate rules for rewriting scalar products containing loop momenta in terms of inverse propagators (`FCLoopCreateRuleGLIToGLI`) so that the full amplitude can be written solely in terms of `GLI`s. All these results can be readily converted into FORM `id`-statements and thus used in a FORM-based setup.

8. Summary

FeynCalc 10 is a big step towards the goal of bringing multiloop calculations closer to the broad audience of interested phenomenologists. The presented release of the package integrates numerous new functions designed to facilitate and streamline manipulations of loop integrals and topologies. Even though the underlying algorithms are well-known to the practitioners and have already been implemented in many publicly available software packages, having them all conveniently accessible via high-level functions within one framework significantly lowers the bar for using those techniques in daily research. Owing to FeynCalc's focus on flexibility, modularity and ease-of-use, users can casually employ these new functions whenever it is convenient for them, without the need to abandon their existing codes. The only condition is to convert the integral families appearing in the calculation into the `FCTopology`-notation, which normally can be done using just a few simple replacement rules.

Despite of all this progress, we would again like to stress that doing multiloop calculations with FeynCalc alone is not the goal we are aiming for. Our vision is to have a FORM-based calculational framework, where only certain steps (e.g. topology minimization) should be performed using FeynCalc. To this end it was necessary to equip FeynCalc with the functions and symbols described in the present work. The next steps are to release an improved interface (a new version of FeynHelpers) connecting FeynCalc to other popular tools used in multiloop calculations and to make the related FORM-based setup publicly available. These tasks are currently being worked on and we hope to complete them in the near future.

CRediT authorship contribution statement

Vladyslav Shtabovenko: Writing – review & editing, Writing – original draft, Software. **Rolf Mertig:** Writing – review & editing, Writing – original draft, Software. **Frederik Orellana:** Writing – review & editing, Writing – original draft, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

One of the authors (VS) would like to acknowledge Guido Bell, Simone Biondini, William Torres Bobadilla, David Broadhurst, Konstantin Chetyrkin, Joshua Davies, Florian Herren, Marvin Gerlach, Dennis Horstmann, Tobias Huber, Vitaly Magerya, Martin Lang, Fabian Lange, Ulrich Nierste, Erik Panzer, Kai Schönwald, Alexander Smirnov, Vladimir Smirnov and Matthias Steinhauser for useful discussions on different aspects of automatic perturbative calculations. The research of VS was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 396021762 — TRR 257 "Particle Physics Phenomenology after the Higgs Discovery". This paper has been assigned preprint numbers TTP23-056, P3H-23-089 and SI-HEP-2023-27.

Appendix A. Pak's algorithm

In this section we provide a brief description of Pak's algorithm including an illustrative example, similar to the one that was presented in ref. [100]. The starting point is always the derivation of the Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$ for the given integral or topology. Then we can construct a characteristic polynomial $\mathcal{P} \equiv \mathcal{U} + \mathcal{F}$[13] describing the given integral family. In the case of a loop integral we also need to save the power of each denominator.

The polynomial is $\mathcal{P}$ typically of the form

$$\mathcal{P} = \sum_{i=1}^{l} h_i \, x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}, \tag{A.1}$$

where h_i is a kinematics-dependent factor, n stands for the number of propagators or Feynman parameters x_i and l is the total number of terms. It is convenient to write $\mathcal{P}$ as an $l \times (n+1)$ matrix,

$$\mathcal{P} \to \begin{pmatrix} h_1 & x_1^{a_1} & x_2^{a_2} & \cdots & x_n^{a_n} \\ h_2 & x_1^{b_1} & x_2^{b_2} & \cdots & x_n^{b_n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ h_l & x_1^{z_1} & x_2^{z_2} & \cdots & x_n^{z_n} \end{pmatrix}, \tag{A.2}$$

where we now want to find a canonical way to rename the x_i. To that aim we set $i = 1$ and generate new matrices by switching the $(i+1)$-th column with each of the next columns. We can keep track of these permutations by giving the matrices suitable names containing the column numbers.

Looking only at the first $(i+1)$-columns of the new matrices we need to sort their rows. The exact nature of the sorting algorithm is irrelevant here, as long as we always use the same procedure for all matrices.

[13] The choice $\mathcal{U} \times \mathcal{F}$ is also possible, but will usually contain a larger number of terms, so for performance reasons we prefer the sum and not the product.

Computer algebra systems such as MATHEMATICA or MAPLE usually can sort lists out-of-the box. For other programming languages one could e.g. implement some lexicographic sorting algorithm.

Having obtained the sorted matrices we extract the i-th column from each of them, generating a list of vectors. Upon sorting this list we take the first vector and keep only matrices that contain the corresponding column while discarding the rest. Then we increase i by one unit and start another iteration of the cycle, where we start with the set of matrices obtained previously.

This procedure is repeated until we reach $i = n - 1$. Then we collect the final permutations σ of the remaining matrices, sort them and take the first permutation as our canonical way to name the Feynman parameters x_i. Notice that σ also provides us with a list of symmetries under x_i-renamings.

To illustrate this procedure let us consider the following characteristic polynomial

$$\mathcal{P} = c_2 x_2 x_3 + c_1 x_2^2 + c_2 x_1 x_3 + c_1 x_1^2 \Rightarrow \begin{pmatrix} c_2 & 0 & 1 & 1 \\ c_1 & 0 & 2 & 0 \\ c_2 & 1 & 0 & 1 \\ c_1 & 2 & 0 & 0 \end{pmatrix} \equiv M_0^{(123)} \tag{A.3}$$

In the first iteration ($i = 1$) we start with $\{M_0^{(123)}\}$ and permute the second column

$$\left\{ M_0^{(123)} = \begin{pmatrix} c_2 & 0 & 1 & 1 \\ c_1 & 0 & 2 & 0 \\ c_2 & 1 & 0 & 1 \\ c_1 & 2 & 0 & 0 \end{pmatrix}, M_0^{(213)} = \begin{pmatrix} c_2 & 1 & 0 & 1 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_1 & 0 & 2 & 0 \end{pmatrix}, \right.$$
$$\left. M_0^{(321)} = \begin{pmatrix} c_2 & 1 & 1 & 0 \\ c_1 & 0 & 2 & 0 \\ c_2 & 1 & 0 & 1 \\ c_1 & 0 & 0 & 2 \end{pmatrix} \right\}. \tag{A.4}$$

After sorting rows with respect to the first two columns we get

$$\left\{ \tilde{M}_0^{(123)} = \begin{pmatrix} \mathbf{c_1} & \mathbf{0} & 2 & 0 \\ \mathbf{c_1} & \mathbf{2} & 0 & 0 \\ \mathbf{c_2} & \mathbf{0} & 1 & 1 \\ \mathbf{c_2} & \mathbf{1} & 0 & 1 \end{pmatrix}, \tilde{M}_0^{(213)} = \begin{pmatrix} \mathbf{c_1} & \mathbf{0} & 2 & 0 \\ \mathbf{c_1} & \mathbf{2} & 0 & 0 \\ \mathbf{c_2} & \mathbf{0} & 1 & 1 \\ \mathbf{c_2} & \mathbf{1} & 0 & 1 \end{pmatrix}, \right.$$
$$\left. \tilde{M}_0^{(321)} = \begin{pmatrix} \mathbf{c_1} & \mathbf{0} & 2 & 0 \\ \mathbf{c_1} & \mathbf{0} & 0 & 2 \\ \mathbf{c_2} & \mathbf{1} & 1 & 0 \\ \mathbf{c_2} & \mathbf{1} & 0 & 1 \end{pmatrix} \right\}. \tag{A.5}$$

The maximal vector among the second columns of all matrices is $(0, 2, 0, 1)^T$, which means that we need to keep $\tilde{M}_0^{(123)}$ and $\tilde{M}_0^{(213)}$ while discarding $\tilde{M}_0^{(321)}$.

The next iteration ($i = 2$) starts with

$$\left\{ \tilde{M}_0^{(123)} = \begin{pmatrix} c_1 & 0 & 2 & 0 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 0 & 1 \end{pmatrix}, \tilde{M}_0^{(213)} = \begin{pmatrix} c_1 & 0 & 2 & 0 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 0 & 1 \end{pmatrix} \right\}. \tag{A.6}$$

Permuting the third column we get

$$\left\{ M_1^{(123)} = \begin{pmatrix} c_1 & 0 & 2 & 0 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 0 & 1 \end{pmatrix}, M_1^{(132)} = \begin{pmatrix} c_1 & 0 & 0 & 2 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 1 & 0 \end{pmatrix}, \right.$$
$$\left. M_1^{(213)} = \begin{pmatrix} c_1 & 0 & 2 & 0 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 0 & 1 \end{pmatrix}, M_1^{(231)} = \begin{pmatrix} c_1 & 0 & 0 & 2 \\ c_1 & 2 & 0 & 0 \\ c_2 & 0 & 1 & 1 \\ c_2 & 1 & 1 & 0 \end{pmatrix} \right\}. \tag{A.7}$$

Sorting rows with respect to the first three columns does not introduce any changes in the matrices

$$\{\tilde{M}_1^{(123)} = M_1^{(123)}, \tilde{M}_1^{(132)} = M_1^{(132)}, \tilde{M}_1^{(213)} = M_1^{(213)}, \tilde{M}_1^{(231)} = M_1^{(231)}\} \tag{A.8}$$

This time the maximal vector the third columns is $(2, 0, 1, 0)^T$ so that we keep only $\tilde{M}_1^{(123)}$ and $\tilde{M}_1^{(213)}$. Since $i = 3 = n - 1 = 3$ the algorithm terminates here.

The outcome of this procedure is the symmetries under the renamings of x_i

$$\sigma = \{(123), (213)\}, \tag{A.9}$$

meaning that

$$\mathcal{P}^{(123)} = c_2 x_2 x_3 + c_1 x_2^2 + c_2 x_1 x_3 + c_1 x_1^2, \tag{A.10}$$
$$\mathcal{P}^{(213)} = c_2 x_1 x_3 + c_1 x_1^2 + c_2 x_2 x_3 + c_1 x_2^2 \tag{A.11}$$

are equivalent. Here (123) is our canonical naming scheme. With $\mathcal{P}^{(123)}$ we have an expression that uniquely characterizes the corresponding set of propagators. Any other loop integral or integral topology that differs from the given one only by a finite set of loop momentum shifts should have the same characteristic polynomial. This is why by comparing $\mathcal{P}$'s (and propagator powers) we can identify one-to-one mappings between different integrals.

Appendix B. Mapping of smaller topologies into larger topologies

Pak algorithm can only find mappings between topologies that contain the same number of propagators. In practice, one often encounters cases were a topology with a smaller number of propagators happens to fit into a topology with a larger number of propagators. Such relations can be uncovered in the following way.

First, we need a list of parent topologies that contain enough propagators to allow fitting smaller topologies. Those parent topologies can have external origin or stem from the current calculation. Ideally, each parent topology should have a complete set of propagators forming a basis, so that it can be directly used for IBP reduction. In the next step, we can analyze each parent topology and determine all of its nonvanishing subtopologies. This means that we try removing one, two or more propagators from that topology and then check if the resulting topology becomes scaleless and hence vanishes.[14] This can be done using the routine `FCLoopFindSubtopologies`. Each nonvanishing subtopology receives a marker of the form `FCGV["SubtopologyOf"] -> topologyID` which is added to the 6th slot of the corresponding `FCTopology` object, where `topologyID` refers to the parent topology from which this subtopology has been obtained.

Then, we can run `FCLoopFindTopologyMappings` where the smaller topologies are given as the first argument of the function, while the list of subtopologies obtained from `FCLoopFindSubtopologies` is passed via the option `PreferredTopologies`. Mappings between small topologies and nonvanishing subtopologies with the same number of propagators can be now determined using the conventional Pak algorithm. Once we have the momentum shifts that allow us to map a small topology into a nonvanishing subtopology of a larger topology, we can immediately work out the mapping into the parent topology. Thanks to the `SubtopologyOf`-marker we already know the name of the parent topology, so we only need to apply the momentum shifts to it and pass both topologies to the auxiliary routine `FCLoopCreateRuleGLI-ToGLI`. This gives us the desired relation between a smaller and a larger topology.

One should remark, that the determination of nonvanishing subtopologies can be quite time-consuming and tends to generate large numbers

[14] The criterion for checking the scalefulness of an arbitrary loop integral was presented in ref. [150]. A good description of the algorithm can be found in Sec. 2.3.4 of ref. [103]. In FEYNCALC the corresponding routines are called `FCLoopScalelessQ` (for checking loop integrals or topologies) and `FCLoop-PakScalelessQ` (for checking characteristic polynomials $\mathcal{P}$).

of resulting topologies. Thus, given a big set of complicated topologies, the total number of their nonvanishing subtopologies can easily get several orders of magnitude larger. Processing all those topologies using `FCLoopFindTopologyMappings` can, therefore, take a lot of time. For this reason we do not recommend using this feature extensively, unless the number of topologies is small (e.g. $\mathcal{O}(100)$) or it is absolutely necessary to find some specific mappings. However, we aim to speed up this functionality in near future.

Appendix C. Limitations of the current approach to topology minimization

While Pak algorithm can tell us that two topologies are identical, finding a set of momentum shifts that realizes this mapping may not always be straightforward.

For topologies containing only standard quadratic propagators of the form[15] $l^2 \pm m^2$ the situation is very simple. The line momentum l, which is a linear combination of some loop momenta p_j and external momenta q_k, directly gives us the momentum flow through the corresponding propagator. The Pak algorithm orders the Feynman parameters of each topology in a canonical way and this ordering can be applied also to the propagators since each Feynman parameter x_i corresponds to the ith propagator of the integral family. Having two sets of ordered propagators $\{D_1, D_2, \ldots, D_n\}$ and $\{D'_1, D'_2, \ldots, D'_n\}$ with $D^{(\prime)}_i = l^{(\prime)2}_i \pm m^{(\prime)2}_i$ we can directly write down a system of equations[16] for the line momenta

$$l_1^2 = (l'_1)^2, \quad l_2^2 = (l'_2)^2, \quad \ldots, \quad l_n^2 = (l'_n)^2 \tag{C.1}$$

and solve it for p_k or p'_k. Since both topologies are identical, there must be at least one solution, which gives us the desired momentum shifts. Notice, that in general such systems turn out to be overdetermined since the number of loop momenta is usually much smaller than the total number of propagators.

Now let us consider topologies containing some eikonal propagators E_i. Pak algorithm still can recognize that they are identical, as eikonal propagators do not pose any additional complications when calculating the Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$. However, denominators of the form $q \cdot s \pm m^2$ do not allow us to recover the momentum flow through this propagator unambiguously. This complicates the process of determining the necessary momentum shifts. To avoid dealing with other than linear systems of equations, we choose a pragmatic approach, where we simply remove the eikonal propagators from both sets of ordered propagators.

Since every loop integral containing only eikonal propagators will be scaleless, there must be at least one quadratic propagator D_i for each loop momentum. Furthermore, as have been observed above, the system of equations we need to solve is usually overdetermined so that upon removing suitable equations it should still remain solvable. Although we recognize that there may be pathological cases where this approach will fail, as of now we are not aware of a better solution to this problem that does not involve massive performance penalties.

Last but not least, there also exists an interpolating case between quadratic and purely eikonal propagators, where the denominators are of the form $l^2 + c_1 l \cdot s + c_2$ with c_i being some constants. Such propagators can arise e.g. when doing asymptotic expansions at the integrand level. On the one hand, naively discarding them may lead to unsolvable systems of equations. On the other hand, automatic determinations of the momentum flow through such lines can be tricky. To this aim FEYNCALC contains a helper function `FCLoopReplaceQuadraticEikonalPropagators` that should be applied to a list of topologies containing mixed propagators. Given the loop momenta as well as some extra replacement rules[17] this routine should be able to rewrite mixed propagators in terms of purely quadratic ones.

[15] Here and below it is understood that m can be also zero without changing the presented discussion.

[16] We square both sides of the equation to allow for solutions that introduce sign changes of some loop momenta e.g. $p_j \to -p_j$.

[17] e.g. that $p_1^2 - 2p_1 \cdot p_2 + p_2^2$ combines to $(p_1 - p_2)^2$.

References

[1] G. Apollinari, O. Brüning, T. Nakamoto, L. Rossi, High luminosity large hadron collider HL-LHC, CERN Yellow Rep. 5 (2015) 1–19, https://doi.org/10.5170/CERN-2015-005.1, arXiv:1705.08830.

[2] M. Veltman, Algebraic techniques, Comput. Phys. Commun. 3 (1972) 75–78, https://doi.org/10.1016/0010-4655(72)90115-4.

[3] F.V. Tkachov, Algebraic algorithms for multiloop calculations. The first 15 years. What's next?, Nucl. Instrum. Methods A 389 (1997) 309–313, https://doi.org/10.1016/S0168-9002(97)00110-1, arXiv:hep-ph/9609429.

[4] V.A. Smirnov, Feynman Integral Calculus, Springer, Berlin, Heidelberg, 2006.

[5] S. Weinzierl, Feynman Integrals. A Comprehensive Treatment for Students and Researchers, UNITEXT for Physics, Springer, 2022, arXiv:2201.03593.

[6] G. Passarino, M.J.G. Veltman, One loop corrections for e+ e- annihilation into mu+ mu- in the Weinberg model, Nucl. Phys. B 160 (1979) 151–207, https://doi.org/10.1016/0550-3213(79)90234-7.

[7] B. Agarwal, G. Heinrich, S.P. Jones, M. Kerner, S.Y. Klein, J. Lang, V. Magerya, A. Olsson, Two-loop amplitudes for $t\bar{t}H$ production: the quark-initiated N_f-part, J. High Energy Phys. 05 (2024) 013, https://doi.org/10.1007/JHEP05(2024)013, Erratum: J. High Energy Phys. 06 (2024) 142, arXiv:2402.03301.

[8] C. Bogner, S. Borowka, T. Hahn, G. Heinrich, S.P. Jones, M. Kerner, A. von Manteuffel, M. Michel, E. Panzer, V. Papara, Loopedia, a database for loop integrals, Comput. Phys. Commun. 225 (2018) 1–9, https://doi.org/10.1016/j.cpc.2017.12.017, arXiv:1709.01266.

[9] K. Hepp, Proof of the Bogolyubov-Parasiuk theorem on renormalization, Commun. Math. Phys. 2 (1966) 301–326, https://doi.org/10.1007/BF01773358.

[10] E.R. Speer, Mass singularities of generic Feynman amplitudes, Ann. Inst. Henri Poincaré Phys. Théor. 26 (1977) 87–105.

[11] T. Binoth, G. Heinrich, An automatized algorithm to compute infrared divergent multiloop integrals, Nucl. Phys. B 585 (2000) 741–759, https://doi.org/10.1016/S0550-3213(00)00429-6, arXiv:hep-ph/0004013.

[12] G. Heinrich, Sector decomposition, Int. J. Mod. Phys. A 23 (2008) 1457–1486, https://doi.org/10.1142/S0217751X08040263, arXiv:0803.4177.

[13] J.M. Campbell, et al., Event generators for high-energy physics experiments, SciPost Phys. 16 (5) (2024) 130, https://doi.org/10.21468/SciPostPhys.16.5.130, arXiv:2203.11110.

[14] S. Abreu, J. Dormans, F. Febres Cordero, H. Ita, M. Kraus, B. Page, E. Pascual, M.S. Ruf, V. Sotnikov, Caravel: a C++ framework for the computation of multi-loop amplitudes with numerical unitarity, Comput. Phys. Commun. 267 (2021) 108069, https://doi.org/10.1016/j.cpc.2021.108069, arXiv:2009.11957.

[15] G. Heinrich, S.P. Jones, M. Kerner, V. Magerya, A. Olsson, J. Schlenk, Numerical scattering amplitudes with pySecDec, Comput. Phys. Commun. 295 (2024) 108956, https://doi.org/10.1016/j.cpc.2023.108956, arXiv:2305.19768.

[16] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones, M. Kerner, J. Schlenk, T. Zirke, Numerical multi-loop calculations: tools and applications, J. Phys. Conf. Ser. 762 (1) (2016) 012073, https://doi.org/10.1088/1742-6596/762/1/012073, arXiv:1604.00267.

[17] S. Borowka, N. Greiner, G. Heinrich, S.P. Jones, M. Kerner, J. Schlenk, U. Schubert, T. Zirke, Higgs boson pair production in gluon fusion at next-to-leading order with full top-quark mass dependence, Phys. Rev. Lett. 117 (1) (2016) 012001, https://doi.org/10.1103/PhysRevLett.117.079901, Erratum: Phys. Rev. Lett. 117 (2016) 079901, arXiv:1604.06447.

[18] S. Pozzorini, N. Schär, M.F. Zoller, Two-loop tensor integral coefficients in OpenLoops, J. High Energy Phys. 05 (2022) 161, https://doi.org/10.1007/JHEP05(2022)161, arXiv:2201.11615.

[19] M.F. Zoller, S. Pozzorini, N. Schaer, Towards two-loop automation in OpenLoops, PoS LL2022 (2022) 073, https://doi.org/10.22323/1.416.0073, arXiv:2207.07468.

[20] D. Canko, G. Bevilacqua, C. Papadopoulos, Two-loop amplitude reduction with HELAC, PoS RADCOR2023 (2024) 081, https://doi.org/10.22323/1.432.0081, arXiv:2309.14886.

[21] K.G. Chetyrkin, F.V. Tkachov, Integration by parts: the algorithm to calculate β-functions in 4 loops, Nucl. Phys. B 192 (1981) 159–204, https://doi.org/10.1016/0550-3213(81)90199-1.

[22] F.V. Tkachov, A theorem on analytical calculability of 4-loop renormalization group functions, Phys. Lett. B 100 (1981) 65–68, https://doi.org/10.1016/0370-2693(81)90288-4.

[23] M. Gerlach, F. Herren, M. Lang, tapir: a tool for topologies, amplitudes, partial fraction decomposition and input for reductions, Comput. Phys. Commun. 282 (2023) 108544, https://doi.org/10.1016/j.cpc.2022.108544, arXiv:2201.05618.

[24] F. Feng, Y.-F. Xie, Q.-C. Zhou, S.-R. Tang, HepLib: a C++ library for high energy physics, Comput. Phys. Commun. 265 (2021) 107982, https://doi.org/10.1016/j.cpc.2021.107982, arXiv:2103.08507.

[25] V. Maheria, Semi- and Fully-Inclusive Phase-Space Integrals at Four Loops, PhD. thesis, 2022.

[26] Q.-f. Wu, Z. Li, FeAmGen.jl: a Julia program for Feynman amplitude generation, Comput. Phys. Commun. 301 (2024) 109230, https://doi.org/10.1016/j.cpc.2024.109230, arXiv:2310.07634.

[27] J.A.M. Vermaseren, New features of FORM, arXiv:math-ph/0010025, 10 2000.

[28] J. Kuipers, T. Ueda, J.A.M. Vermaseren, J. Vollinga, FORM version 4.0, Comput. Phys. Commun. 184 (2013) 1453–1467, https://doi.org/10.1016/j.cpc.2012.12.028, arXiv:1203.6543.

[29] R. Mertig, M. Bohm, A. Denner, FEYN CALC: computer algebraic calculation of Feynman amplitudes, Comput. Phys. Commun. 64 (1991) 345–359, https://doi.org/10.1016/0010-4655(91)90130-D.

[30] V. Shtabovenko, R. Mertig, F. Orellana, New developments in FeynCalc 9.0, Comput. Phys. Commun. 207 (2016) 432–444, https://doi.org/10.1016/j.cpc.2016.06.008, arXiv:1601.01167.

[31] V. Shtabovenko, R. Mertig, F. Orellana, FeynCalc 9.3: new features and improvements, Comput. Phys. Commun. 256 (2020) 107478, https://doi.org/10.1016/j.cpc.2020.107478, arXiv:2001.04407.

[32] V. Shtabovenko, FeynCalc goes multiloop, J. Phys. Conf. Ser. 2438 (1) (2023) 012140, https://doi.org/10.1088/1742-6596/2438/1/012140, arXiv:2112.14132.

[33] V. Shtabovenko, FeynHelpers: connecting FeynCalc to FIRE and Package-X, Comput. Phys. Commun. 218 (2017) 48–65, https://doi.org/10.1016/j.cpc.2017.04.014, arXiv:1611.06793.

[34] R. Mertig, W.L. van Neerven, The calculation of the two loop spin splitting functions P(ij)(1)(x), Z. Phys. C 70 (1996) 637–654, https://doi.org/10.1007/s002880050138, arXiv:hep-ph/9506451.

[35] R. Mertig, R. Scharf, TARCER: a Mathematica program for the reduction of two loop propagator integrals, Comput. Phys. Commun. 111 (1998) 265–273, https://doi.org/10.1016/S0010-4655(98)00042-3, arXiv:hep-ph/9801383.

[36] M. Wiebusch, HEPMath 1.4: a mathematica package for semi-automatic computations in high energy physics, Comput. Phys. Commun. 195 (2015) 172–190, https://doi.org/10.1016/j.cpc.2015.04.022, arXiv:1412.6102.

[37] H.H. Patel, Package-X: a Mathematica package for the analytic calculation of one-loop integrals, Comput. Phys. Commun. 197 (2015) 276–290, https://doi.org/10.1016/j.cpc.2015.08.017, arXiv:1503.01469.

[38] H.H. Patel, Package-X 2.0: a Mathematica package for the analytic calculation of one-loop integrals, Comput. Phys. Commun. 218 (2017) 66–70, https://doi.org/10.1016/j.cpc.2017.04.015, arXiv:1612.00009.

[39] A.K. Cyrol, M. Mitter, N. Strodthoff, FormTracer - a mathematica tracing package using FORM, Comput. Phys. Commun. 219 (2017) 346–352, https://doi.org/10.1016/j.cpc.2017.05.024, arXiv:1610.09331.

[40] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.S. Shao, T. Stelzer, P. Torrielli, M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, J. High Energy Phys. 07 (2014) 079, https://doi.org/10.1007/JHEP07(2014)079, arXiv:1405.0301.

[41] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, G. Ossola, T. Reiter, F. Tramontano, Automated one-loop calculations with GoSam, Eur. Phys. J. C 72 (2012) 1889, https://doi.org/10.1140/epjc/s10052-012-1889-1, arXiv:1111.2034.

[42] G. Cullen, et al., G$OSAM$-2.0: a tool for automated one-loop calculations within the Standard Model and beyond, Eur. Phys. J. C 74 (8) (2014) 3001, https://doi.org/10.1140/epjc/s10052-014-3001-5, arXiv:1404.7096.

[43] M. Bahr, et al., Herwig++ physics and manual, Eur. Phys. J. C 58 (2008) 639–707, https://doi.org/10.1140/epjc/s10052-008-0798-9, arXiv:0803.0883.

[44] J. Bellm, et al., Herwig 7.0/Herwig++ 3.0 release note, Eur. Phys. J. C 76 (4) (2016) 196, https://doi.org/10.1140/epjc/s10052-016-4018-8, arXiv:1512.01178.

[45] G. Bevilacqua, M. Czakon, M.V. Garzelli, A. van Hameren, A. Kardos, C.G. Papadopoulos, R. Pittau, M. Worek, HELAC-NLO, Comput. Phys. Commun. 184 (2013) 986–997, https://doi.org/10.1016/j.cpc.2012.10.033, arXiv:1110.1499.

[46] P. Nason, A new method for combining NLO QCD with shower Monte Carlo algorithms, J. High Energy Phys. 11 (2004) 040, https://doi.org/10.1088/1126-6708/2004/11/040, arXiv:hep-ph/0409146.

[47] S. Frixione, P. Nason, C. Oleari, Matching NLO QCD computations with parton shower simulations: the POWHEG method, J. High Energy Phys. 11 (2007) 070, https://doi.org/10.1088/1126-6708/2007/11/070, arXiv:0709.2092.

[48] S. Alioli, P. Nason, C. Oleari, E. Re, A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX, J. High Energy Phys. 06 (2010) 043, https://doi.org/10.1007/JHEP06(2010)043, arXiv:1002.2581.

[49] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, J. Winter, Event generation with SHERPA 1.1, J. High Energy Phys. 02 (2009) 007, https://doi.org/10.1088/1126-6708/2009/02/007, arXiv:0811.4622.

[50] E. Bothmann, et al., Event generation with Sherpa 2.2, SciPost Phys. 7 (3) (2019) 034, https://doi.org/10.21468/SciPostPhys.7.3.034, arXiv:1905.09127.

[51] M. Moretti, T. Ohl, J. Reuter, O'Mega: an optimizing matrix element generator (2001) 1981–2009, arXiv:hep-ph/0102195.

[52] W. Kilian, T. Ohl, J. Reuter, WHIZARD: simulating multi-particle processes at LHC and ILC, Eur. Phys. J. C 71 (2011) 1742, https://doi.org/10.1140/epjc/s10052-011-1742-y, arXiv:0708.4233.

[53] A. Belyaev, N.D. Christensen, A. Pukhov, CalcHEP 3.4 for collider physics within and beyond the Standard Model, Comput. Phys. Commun. 184 (2013) 1729–1769, https://doi.org/10.1016/j.cpc.2013.01.014, arXiv:1207.6082.

[54] E. Boos, V. Bunichev, M. Dubinin, L. Dudko, V. Ilyin, A. Kryukov, V. Edneral, V. Savrin, A. Semenov, A. Sherstnev, CompHEP 4.4: automatic computations from Lagrangians to events, Nucl. Instrum. Methods A 534 (2004) 250–259, https://doi.org/10.1016/j.nima.2004.07.096, arXiv:hep-ph/0403113.

[55] F. Yuasa, et al., Automatic computation of cross-sections in HEP: status of GRACE system, Prog. Theor. Phys. Suppl. 138 (2000) 18–23, https://doi.org/10.1143/PTPS.138.18, arXiv:hep-ph/0007053.

[56] J. Fujimoto, et al., GRACE/SUSY automatic generation of tree amplitudes in the minimal supersymmetric standard model, Comput. Phys. Commun. 153 (2003) 106–134, https://doi.org/10.1016/S0010-4655(03)00159-0, arXiv:hep-ph/0208036.

[57] M. Gerlach, U. Nierste, V. Shtabovenko, M. Steinhauser, Width difference in the B-B^- system at next-to-next-to-leading order of QCD, Phys. Rev. Lett. 129 (10) (2022) 102001, https://doi.org/10.1103/PhysRevLett.129.102001, arXiv:2205.07907.

[58] D. Stöckinger, M. Weißwange, Full three-loop renormalisation of an abelian chiral gauge theory with non-anticommuting γ_5 in the BMHV scheme, J. High Energy Phys. 02 (2024) 139, https://doi.org/10.1007/JHEP02(2024)139, arXiv:2312.11291.

[59] T. Yang, Renormalization of twist-two operators and four-loop splitting functions in QCD, PoS RADCOR2023 (2024) 056, https://doi.org/10.22323/1.432.0056.

[60] T. Gehrmann, A. von Manteuffel, T.-Z. Yang, Renormalization of twist-two operators in covariant gauge to three loops in QCD, J. High Energy Phys. 04 (2023) 041, https://doi.org/10.1007/JHEP04(2023)041, arXiv:2302.00022.

[61] P. Kühler, D. Stöckinger, M. Weißwange, Advances at the γ_5-frontier, arXiv:2407.07247, 7 2024.

[62] P. Reeck, Update on technical aspects of B meson mixing at NNLO, in: Loops and Legs in Quantum Field Theory, 2024, arXiv:2406.16084.

[63] D. Akpinar, F. Febres Cordero, M. Kraus, M.S. Ruf, M. Zeng, Spinning black hole scattering at $\mathcal{O}(G^3 S^2)$: Casimir terms, radial action and hidden symmetry, arXiv:2407.19005, 7 2024.

[64] P. Nogueira, Automatic Feynman graph generation, J. Comput. Phys. 105 (1993) 279–289, https://doi.org/10.1006/jcph.1993.1074.

[65] https://www.graphviz.org/.

[66] R.N. Lee, Presenting LiteRed: a tool for the loop InTEgrals REDuction, arXiv:1212.2685, 12 2012.

[67] R.N. Lee, LiteRed 1.4: a powerful tool for reduction of multiloop integrals, J. Phys. Conf. Ser. 523 (2014) 012059, https://doi.org/10.1088/1742-6596/523/1/012059, arXiv:1310.1145.

[68] A.V. Smirnov, FIRE5: a C++ implementation of Feynman integral REduction, Comput. Phys. Commun. 189 (2015) 182–191, https://doi.org/10.1016/j.cpc.2014.11.024, arXiv:1408.2372.

[69] A.V. Smirnov, F.S. Chuharev, FIRE6: Feynman integral REduction with modular arithmetic, Comput. Phys. Commun. 247 (2020) 106877, https://doi.org/10.1016/j.cpc.2019.106877, arXiv:1901.07808.

[70] A.V. Smirnov, M. Zeng, FIRE 6.5: Feynman integral reduction with new simplification library, Comput. Phys. Commun. 302 (2024) 109261, https://doi.org/10.1016/j.cpc.2024.109261, arXiv:2311.02370.

[71] P. Maierhöfer, J. Usovitsch, P. Uwer, Kira—a Feynman integral reduction program, Comput. Phys. Commun. 230 (2018) 99–112, https://doi.org/10.1016/j.cpc.2018.04.012, arXiv:1705.05610.

[72] P. Maierhöfer, J. Usovitsch, Kira 1.2 release notes, arXiv:1812.01491, 12 2018.

[73] P. Maierhöfer, J. Usovitsch, Recent developments in Kira, CERN Yellow Rep.: Monogr. 3 (2020) 201–204, https://doi.org/10.23731/CYRM-2020-003.201.

[74] J. Klappert, F. Lange, P. Maierhöfer, J. Usovitsch, Integral reduction with Kira 2.0 and finite field methods, Comput. Phys. Commun. 266 (2021) 108024, https://doi.org/10.1016/j.cpc.2021.108024, arXiv:2008.06494.

[75] F. Lange, P. Maierhöfer, J. Usovitsch, Developments since Kira 2.0, SciPost Phys. Proc. 7 (2022) 017, https://doi.org/10.21468/SciPostPhysProc.7.017, arXiv:2111.01045.

[76] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones, M. Kerner, J. Schlenk, T. Zirke, pySecDec: a toolbox for the numerical evaluation of multi-scale integrals, Comput. Phys. Commun. 222 (2018) 313–326, https://doi.org/10.1016/j.cpc.2017.09.015, arXiv:1703.09692.

[77] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones, M. Kerner, J. Schlenk, A GPU compatible quasi-Monte Carlo integrator interfaced to pySecDec, Comput. Phys. Commun. 240 (2019) 120–137, https://doi.org/10.1016/j.cpc.2019.02.015, arXiv:1811.11720.

[78] G. Heinrich, S. Jahn, S.P. Jones, M. Kerner, F. Langer, V. Magerya, A. Pöldaru, J. Schlenk, E. Villa, Expansion by regions with pySecDec, Comput. Phys. Commun. 273 (2022) 108267, https://doi.org/10.1016/j.cpc.2021.108267, arXiv:2108.10807.

[79] R. Harlander, T. Seidensticker, M. Steinhauser, Complete corrections of order alpha alpha-s to the decay of the Z boson into bottom quarks, Phys. Lett. B 426 (1998) 125–132, https://doi.org/10.1016/S0370-2693(98)00220-2, arXiv:hep-ph/9712228.

[80] T. Seidensticker, Automatic application of successive asymptotic expansions of Feynman diagrams, in: 6th International Workshop on New Computing Techniques in Physics Research: Software Engineering, Artificial Intelligence Neural Nets, Genetic Algorithms, Symbolic Algebra, Automatic Calculation, 1999, arXiv:hep-ph/9905298.

[81] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter, UFO - the universal FeynRules output, Comput. Phys. Commun. 183 (2012) 1201–1214, https://doi.org/10.1016/j.cpc.2012.01.022, arXiv:1108.2040.

[82] L. Darmé, et al., UFO 2.0: the 'universal Feynman output' format, Eur. Phys. J. C 83 (7) (2023) 631, https://doi.org/10.1140/epjc/s10052-023-11780-9, arXiv:2304.09883.

[83] F. Feng, S.-R. Tang, Y.-D. Gao, HepLib: a C++ library for high energy physics (version 1.1), Comput. Phys. Commun. 285 (2023) 108631, https://doi.org/10.1016/j.cpc.2022.108631.

[84] C.W. Bauer, A. Frink, R. Kreckel, Introduction to the GiNaC framework for symbolic computation within the C++ programming language, J. Symb. Comput. 33 (2002) 1–12, https://doi.org/10.1006/jsco.2001.0494, arXiv:cs/0004015.

[85] T. Hahn, M. Perez-Victoria, Automatized one loop calculations in four-dimensions and D-dimensions, Comput. Phys. Commun. 118 (1999) 153–165, https://doi.org/10.1016/S0010-4655(98)00173-8, arXiv:hep-ph/9807565.

[86] A. Denner, S. Dittmaier, L. Hofer, Collier: a fortran-based complex one-loop LIbrary in extended regularizations, Comput. Phys. Commun. 212 (2017) 220–238, https://doi.org/10.1016/j.cpc.2016.10.013, arXiv:1604.06792.

[87] A. Denner, S. Dittmaier, Reduction of one loop tensor five point integrals, Nucl. Phys. B 658 (2003) 175–202, https://doi.org/10.1016/S0550-3213(03)00184-6, arXiv:hep-ph/0212259.

[88] A. Denner, S. Dittmaier, Reduction schemes for one-loop tensor integrals, Nucl. Phys. B 734 (2006) 62–115, https://doi.org/10.1016/j.nuclphysb.2005.11.007, arXiv:hep-ph/0509141.

[89] A. Denner, S. Dittmaier, Scalar one-loop 4-point integrals, Nucl. Phys. B 844 (2011) 199–242, https://doi.org/10.1016/j.nuclphysb.2010.11.002, arXiv:1005.2076.

[90] R.K. Ellis, G. Zanderighi, Scalar one-loop integrals for QCD, J. High Energy Phys. 02 (2008) 002, https://doi.org/10.1088/1126-6708/2008/02/002, arXiv:0712.1851.

[91] A. van Hameren, OneLOop: for the evaluation of one-loop scalar functions, Comput. Phys. Commun. 182 (2011) 2427–2438, https://doi.org/10.1016/j.cpc.2011.06.011, arXiv:1007.4716.

[92] C. Studerus, Reduze-Feynman integral reduction in C++, Comput. Phys. Commun. 181 (2010) 1293–1300, https://doi.org/10.1016/j.cpc.2010.03.012, arXiv:0912.2546.

[93] A. von Manteuffel, C. Studerus, Reduze 2 - distributed Feynman integral reduction, arXiv:1201.4330, 1 2012.

[94] A. Georgoudis, K.J. Larsen, Y. Zhang, Azurite: an algebraic geometry based package for finding bases of loop integrals, Comput. Phys. Commun. 221 (2017) 203–215, https://doi.org/10.1016/j.cpc.2017.08.013, arXiv:1612.04252.

[95] V.A. Smirnov, Analytic Tools for Feynman Integrals, vol. 250, Springer, Berlin, Heidelberg, 2012.

[96] A.V. Smirnov, FIESTA4: optimized Feynman integral calculations with GPU support, Comput. Phys. Commun. 204 (2016) 189–199, https://doi.org/10.1016/j.cpc.2016.03.013, arXiv:1511.03614.

[97] A.V. Smirnov, N.D. Shapurov, L.I. Vysotsky, FIESTA5: numerical high-performance Feynman integral evaluation, Comput. Phys. Commun. 277 (2022) 108386, https://doi.org/10.1016/j.cpc.2022.108386, arXiv:2110.11660.

[98] C. Bogner, S. Weinzierl, Feynman graph polynomials, Int. J. Mod. Phys. A 25 (2010) 2585–2618, https://doi.org/10.1142/S0217751X10049438, arXiv:1002.3458.

[99] M. Beneke, V.A. Smirnov, Asymptotic expansion of Feynman integrals near threshold, Nucl. Phys. B 522 (1998) 321–344, https://doi.org/10.1016/S0550-3213(98)00138-2, arXiv:hep-ph/9711391.

[100] J. Hoff, The Mathematica package TopoID and its application to the Higgs boson production cross section, J. Phys. Conf. Ser. 762 (1) (2016) 012061, https://doi.org/10.1088/1742-6596/762/1/012061, arXiv:1607.04465.

[101] Z. Wu, J. Boehm, R. Ma, H. Xu, Y. Zhang, NeatIBP 1.0, a package generating small-size integration-by-parts relations for Feynman integrals, Comput. Phys. Commun. 295 (2024) 108999, https://doi.org/10.1016/j.cpc.2023.108999, arXiv:2305.08783.

[102] A. Pak, The toolbox of modern multi-loop calculations: novel analytic and semi-analytic techniques, J. Phys. Conf. Ser. 368 (2012) 012049, https://doi.org/10.1088/1742-6596/368/1/012049, arXiv:1111.0868.

[103] J.S. Hoff, Methods for multiloop calculations and Higgs boson production at the LHC, Ph.D. thesis, KIT, Karlsruhe, 2015.

[104] P. Reeck, V. Shtabovenko, M. Steinhauser, B meson mixing at NNLO: technical aspects, J. High Energy Phys. 08 (2024) 002, https://doi.org/10.1007/JHEP08(2024)002, arXiv:2405.14698.

[105] R. Lewis, FERMAT, https://home.bway.net/lewis.

[106] K. Bielas, I. Dubovyk, J. Gluza, T. Riemann, Some remarks on non-planar Feynman diagrams, Acta Phys. Pol. B 44 (11) (2013) 2249–2255, https://doi.org/10.5506/APhysPolB.44.2249, arXiv:1312.5603.

[107] H. Cheng, T.T. Wu, Expanding Protons: Scattering at High-Energies, 1987.

[108] E. Panzer, Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals, Comput. Phys. Commun. 188 (2015) 148–166, https://doi.org/10.1016/j.cpc.2014.10.019, arXiv:1403.3385.

[109] E. Panzer, On hyperlogarithms and Feynman integrals with divergences and many scales, J. High Energy Phys. 03 (2014) 071, https://doi.org/10.1007/JHEP03(2014)071, arXiv:1401.4361.

[110] E. Panzer, Feynman integrals and hyperlogarithms, Ph.D. thesis, Humboldt U, 2015, arXiv:1506.07243.

[111] A.V. Kotikov, Differential equation method: the calculation of N point Feynman diagrams, Phys. Lett. B 267 (1991) 123–127, https://doi.org/10.1016/0370-2693(91)90536-Y, Erratum: Phys. Lett. B 295 (1992) 409.

[112] A.V. Kotikov, Differential equations method: new technique for massive Feynman diagrams calculation, Phys. Lett. B 254 (1991) 158–164, https://doi.org/10.1016/0370-2693(91)90413-K.

[113] A.V. Kotikov, Differential equations method: the calculation of vertex type Feynman diagrams, Phys. Lett. B 259 (1991) 314–322, https://doi.org/10.1016/0370-2693(91)90834-D.

[114] Z. Bern, L.J. Dixon, D.A. Kosower, Dimensionally regulated pentagon integrals, Nucl. Phys. B 412 (1994) 751–816, https://doi.org/10.1016/0550-3213(94)90398-0, arXiv:hep-ph/9306240.

[115] E. Remiddi, Differential equations for Feynman graph amplitudes, Nuovo Cimento A 110 (1997) 1435–1452, https://doi.org/10.1007/BF03185566, arXiv:hep-th/9711188.

[116] T. Gehrmann, E. Remiddi, Differential equations for two-loop four-point functions, Nucl. Phys. B 580 (2000) 485–518, https://doi.org/10.1016/S0550-3213(00)00223-6, arXiv:hep-ph/9912329.

[117] J.M. Henn, Multiloop integrals in dimensional regularization made simple, Phys. Rev. Lett. 110 (2013) 251601, https://doi.org/10.1103/PhysRevLett.110.251601, arXiv:1304.1806.

[118] J.M. Henn, Lectures on differential equations for Feynman integrals, J. Phys. A 48 (2015) 153001, https://doi.org/10.1088/1751-8113/48/15/153001, arXiv:1412.2296.

[119] O. Gituliar, V. Magerya, Fuchsia: a tool for reducing differential equations for Feynman master integrals to epsilon form, Comput. Phys. Commun. 219 (2017) 329–338, https://doi.org/10.1016/j.cpc.2017.05.004, arXiv:1701.04269.

[120] C. Meyer, Transforming differential equations of multi-loop Feynman integrals into canonical form, J. High Energy Phys. 04 (2017) 006, https://doi.org/10.1007/JHEP04(2017)006, arXiv:1611.01087.

[121] C. Meyer, Algorithmic transformation of multi-loop master integrals to a canonical basis with CANONICA, Comput. Phys. Commun. 222 (2018) 295–312, https://doi.org/10.1016/j.cpc.2017.09.014, arXiv:1705.06252.

[122] R.N. Lee, Reducing differential equations for multiloop master integrals, J. High Energy Phys. 04 (2015) 108, https://doi.org/10.1007/JHEP04(2015)108, arXiv:1411.0911.

[123] R.N. Lee, Libra: a package for transformation of differential systems for multiloop integrals, Comput. Phys. Commun. 267 (2021) 108058, https://doi.org/10.1016/j.cpc.2021.108058, arXiv:2012.00279.

[124] M. Prausa, Epsilon: a tool to find a canonical basis of master integrals, Comput. Phys. Commun. 219 (2017) 361–376, https://doi.org/10.1016/j.cpc.2017.05.026, arXiv:1701.00725.

[125] C. Dlapa, J. Henn, K. Yan, Deriving canonical differential equations for Feynman integrals from a single uniform weight integral, J. High Energy Phys. 05 (2020) 025, https://doi.org/10.1007/JHEP05(2020)025, arXiv:2002.02340.

[126] C. Dlapa, J.M. Henn, F.J. Wagner, An algorithmic approach to finding canonical differential equations for elliptic Feynman integrals, J. High Energy Phys. 08 (2023) 120, https://doi.org/10.1007/JHEP08(2023)120, arXiv:2211.16357.

[127] M. Besier, P. Wasser, S. Weinzierl, RationalizeRoots: software package for the rationalization of square roots, Comput. Phys. Commun. 253 (2020) 107197, https://doi.org/10.1016/j.cpc.2020.107197, arXiv:1910.13251.

[128] E. Remiddi, J.A.M. Vermaseren, Harmonic polylogarithms, Int. J. Mod. Phys. A 15 (2000) 725–754, https://doi.org/10.1142/S0217751X00000367, arXiv:hep-ph/9905237.

[129] A.B. Goncharov, Multiple polylogarithms, cyclotomy and modular complexes, Math. Res. Lett. 5 (1998) 497–516, https://doi.org/10.4310/MRL.1998.v5.n4.a7, arXiv:1105.2076.

[130] A.B. Goncharov, Multiple polylogarithms and mixed Tate motives, arXiv:math/0103059, 3 2001.

[131] D. Maitre, HPL, a mathematica implementation of the harmonic polylogarithms, Comput. Phys. Commun. 174 (2006) 222–240, https://doi.org/10.1016/j.cpc.2005.10.008, arXiv:hep-ph/0507152.

[132] D. Maitre, Extension of HPL to complex arguments, Comput. Phys. Commun. 183 (2012) 846, https://doi.org/10.1016/j.cpc.2011.11.015, arXiv:hep-ph/0703052.

[133] C. Duhr, F. Dulat, PolyLogTools — polylogs for the masses, J. High Energy Phys. 08 (2019) 135, https://doi.org/10.1007/JHEP08(2019)135, arXiv:1904.07279.

[134] C. Bogner, MPL—a program for computations with iterated integrals on moduli spaces of curves of genus zero, Comput. Phys. Commun. 203 (2016) 339–353, https://doi.org/10.1016/j.cpc.2016.02.033, arXiv:1510.04562.

[135] N. Brambilla, H.S. Chung, V. Shtabovenko, A. Vairo, FeynOnium: using FeynCalc for automatic calculations in nonrelativistic effective field theories, J. High Energy Phys. 11 (2020) 130, https://doi.org/10.1007/JHEP11(2020)130, arXiv:2006.15451.

[136] S.A. Larin, The renormalization of the axial anomaly in dimensional regularization, Phys. Lett. B 303 (1993) 113–118, https://doi.org/10.1016/0370-2693(93)90053-K, arXiv:hep-ph/9302240.

[137] S. Moch, J.A.M. Vermaseren, A. Vogt, On $\gamma 5$ in higher-order QCD calculations and the NNLO evolution of the polarized valence distribution, Phys. Lett. B 748 (2015) 432–438, https://doi.org/10.1016/j.physletb.2015.07.027, arXiv:1506.04517.

[138] F.A. Berends, R. Kleiss, P. De Causmaecker, R. Gastmans, T.T. Wu, Single Bremsstrahlung processes in gauge theories, Phys. Lett. B 103 (1981) 124–128, https://doi.org/10.1016/0370-2693(81)90685-7.

[139] M. Caffo, E. Remiddi, Evaluation of transition amplitudes between Dirac spinors, Helv. Phys. Acta 55 (1982) 339.

[140] P. De Causmaecker, R. Gastmans, W. Troost, T.T. Wu, Multiple Bremsstrahlung in gauge theories at high-energies. 1. General formalism for quantum electrodynamics, Nucl. Phys. B 206 (1982) 53–60, https://doi.org/10.1016/0550-3213(82)90488-6.

[141] Z. Xu, D.-H. Zhang, L. Chang, Helicity Amplitudes for Multiple Bremsstrahlung in Massless Nonabelian Gauge Theory. 1. New Definition of Polarization Vector and Formulation of Amplitudes in Grassmann Algebra, 12 1984.

[142] R. Kleiss, W.J. Stirling, Spinor techniques for calculating p anti-p —> W+- / Z0 + Jets, Nucl. Phys. B 262 (1985) 235–262, https://doi.org/10.1016/0550-3213(85)90285-8.

[143] J.F. Gunion, Z. Kunszt, Improved analytic techniques for tree graph calculations and the G g q anti-q lepton anti-lepton subprocess, Phys. Lett. B 161 (1985) 333, https://doi.org/10.1016/0370-2693(85)90774-9.

[144] Z. Xu, D.-H. Zhang, L. Chang, Helicity amplitudes for multiple Bremsstrahlung in massless nonabelian gauge theories, Nucl. Phys. B 291 (1987) 392–428, https://doi.org/10.1016/0550-3213(87)90479-2.

[145] https://pandoc.org/.

[146] A. Grozin, Lectures on QED and QCD, in: 3rd Dubna international advanced school of theoretical physics, arXiv:hep-ph/0508242, 2005.

[147] A.I. Davydychev, P. Osland, O.V. Tarasov, Two loop three gluon vertex in zero momentum limit, Phys. Rev. D 58 (1998) 036007, https://doi.org/10.1103/PhysRevD.58.036007, arXiv:hep-ph/9801380.

[148] P. Böer, G. Bell, T. Feldmann, D. Horstmann, V. Shtabovenko, Soft-overlap contribution to $\boldsymbol{Bc} \rightarrow \boldsymbol{\eta c}$ form factors: diagrammatic resummation of double logarithms, PoS RADCOR2023 (2024) 086, https://doi.org/10.22323/1.432.0086, arXiv:2309.08410.

[149] V. Shtabovenko, New multiloop capabilities of FeynCalc 10, in: Loops and Legs in Quantum Field Theory, 2024, arXiv:2407.01447.

[150] A. Pak, A. Smirnov, Geometric approach to asymptotic expansion of Feynman integrals, Eur. Phys. J. C 71 (2011) 1626, https://doi.org/10.1140/epjc/s10052-011-1626-1, arXiv:1011.4863.