FeynCalc 9.3: New features and improvements

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

[a]*Institut für Theoretische Teilchenphysik (TTP), Karlsruher Institut für Technologie (KIT), 76131 Karlsruhe, Germany*
[b]*Zhejiang Institute of Modern Physics, Department of Physics, Zhejiang University, Hangzhou, 310027, China*
[c]*Technische Universität München, Physik-Department T30f, James-Franck-Str. 1, 85747 Garching, Germany*
[d]*GluonVision GmbH, Bötzowstr. 10, 10407 Berlin, Germany*
[e]*Technical University of Denmark, Anker Engelundsvej 1, Building 101A, 2800 Kgs. Lyngby, Denmark*

Abstract

We present FEYNCALC 9.3, a new stable version of a powerful and versatile MATHEMATICA package for symbolic quantum field theory (QFT) calculations. Some interesting new features such as highly improved interoperability with other packages, automatic extraction of the ultraviolet divergent parts of 1-loop integrals, support for amplitudes with Majorana fermions and γ-matrices with explicit Dirac indices are explained in detail. Furthermore, we discuss some common problems and misunderstandings that may arise in the daily usage of the package, providing explanations and workarounds.

Keywords: High energy physics, Feynman diagrams, Loop integrals, Dimensional regularization, Renormalization, Dirac algebra, Passarino–Veltman, Majorana, FeynCalc

1. Introduction

Feynman diagrammatic expansion is one of the most common techniques to approach perturbative calculations in modern theoretical high energy physics (HEP). For the given theory each Feynman diagram can be converted to an algebraic expression (e. g. scattering amplitude) using a set of well-defined rules (Feynman rules). In principle, every single step of a typical HEP calculation from deriving Feynman rules for the given Lagrangian till obtaining explicit results for a particular observable can be carried out using computer codes. In practice, different tools are required to automatize different parts of the calculation and it is often far from trivial to interface and employ multiple codes in an efficient, correct and consistent way. Given the sheer multitude of HEP programs that are publicly available or can be obtained upon request by contacting the authors, it may not be easy to decide which tool fits best for one's project.

This is why we would like to attract the attention of the reader to FEYNCALC [1, 2], a tool that is particularly well suited for manipulating Feynman amplitudes in a specific way, where it is important to keep track of each calculational step. FEYNCALC is an open-source MATHEMATICA package that can be used standalone or conveniently integrated in a custom computational setup. It has been available to the HEP community for almost three decades and is still actively developed and widely used in various fields of particle physics, ranging from Standard Model (SM) and Beyond Standard Model (BSM) processes to Effective Field Theory (EFT) calculations.

Together with PACKAGE-X [3, 4] and HEPMATH [5], FEYNCALC belongs to the category of software tools for semi-automatic calculations. The main goal of such packages

E-mail address: v.shtabovenko@kit.edu
**E-mail address:* rolfm@gluonvision.com
E-mail address: fror@dtu.dk

is to provide the user with an extensive toolbox for symbolic QFT calculations (including, but not limited to, the evaluation of Feynman diagrams), automatizing such common operations as index contractions, matrix chain simplifications, manipulations of loop integrals or summations over polarizations of external particles. These operations can be performed in any order according to the best judgment of the user. Indeed, the practical usage of FEYNCALC is conceptually very similar to pen and paper calculations, with all the related merits and drawbacks. A skillful practitioner can employ the package in highly nontrivial ways, obtaining results that would be difficult to get using a more automatic but less flexible framework. Yet to avoid mistakes and wrong results it is absolutely necessary to possess a good understanding of technical subtleties and especially of the physics behind the particular calculation.

This design philosophy is very different from the behavior of more automatic publicly available frameworks such as MADGRAPH5_AMC@NLO [6], GOSAM [7, 8], HERWIG++ [9], SHERPA [10], WHIZARD/O'MEGA [11, 12], FEYNARTS/FORMCALC [13, 14], CALCHEP [15], COMPHEP [16] and many others. Although here the user may somewhat influence the course of the calculation via different options and switches specified in the input card file, all the technicalities are taken care of by the automatic algorithms under the hood. While this approach is certainly well justified from the point of view of creating fast and easy-to-use codes, it also makes it rather difficult to compute anything that was not envisaged by the package developers or to compute things in a different way.

Of course, the enormous flexibility of FEYNCALC due to is its reliance on MATHEMATICA for all algebraic manipulations comes at a price. The performance is admittedly much worse as compared to codes that use faster symbolic manipulation systems, notably FORM [17, 18]. While this may not be too problematic for smaller calculations that finish within minutes on a modern computer[1], the matter becomes much more pressing when dealing with the increasing number of diagrams, loops and legs. Bearing in mind the intrinsic limitations of MATHEMATICA when manipulating very large expressions with hundreds of thousands or even millions of terms, the only sensible way to address this issue is to adopt a hybrid approach, by offloading computationally heavy operations to FORM, while still employing FEYNCALC for more appropriate tasks. This strategy was successfully applied e. g. in the recent calculation of the Energy-Energy-Correlation function in gluon-initiated Higgs decays at NLO [20], where the most demanding parts of the computation (squaring of double-real matrix elements and reduction of 3-loop integrals) were done with the aid of FORM, COLOR [21] and FIRE [22], while FEYNCALC was used to work out linearly independent bases for loop integral topologies, recalculate the simpler amplitudes for real-virtual corrections and perform various cross-checks. In this sense we believe that it is perfectly possible to employ FEYNCALC in complicated multi-loop setups as long as one does not try to calculate *everything* relying solely on FEYNCALC and MATHEMATICA.

The aim of this paper is to report on the new FEYNCALC 9.3, the next major release since 2016. The installation of the package is covered in Sec. 2, while Sec. 3 describes new features introduced in this version. In Sec. 4 we provide best practices for using FEYNCALC in loop calculations. The current status of the project and an outlook towards future developments are summarized in Sec. 5.

---

[1]An interesting approach to such projects is the new FEYNMASTER [19] framework.

2. Installation

2.1. Automatic installation

The most convenient way to install the current stable version of FEYNCALC is to evaluate the following line

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

in a MATHEMATICA (version 8 or later is required) notebook that runs on a new kernel. The automatic installer essentially downloads the current snapshot of the package from the hotfix-stable branch of the main repository on GITHUB[2] and sets up the diagram generator FEYNARTS to work with FEYNCALC out-of-the-box. Since the stable version of FEYNCALC receives bug fixes until a new official release, it is useful to reinstall the package from time to time to ensure that one is running the most recent version. The updates to the stable version are visible as commits to the hotfix-stable branch[3] of the repository.

The automatic installer may not work with MATHEMATICA versions 8 and 9 due to their limitations when accessing URLs via HTTPS. In that case one can always download all the necessary files by hand and run the installer in the offline mode or proceed with a manual installation. The corresponding instructions including a collection of workarounds for older MATHEMATICA versions can be found in the project wiki[4].

Users who employ the FREE WOLFRAM ENGINE FOR DEVELOPERS and use JUPYTER as a front end should install FEYNCALC via the text-based interface[5]. Although FEYNCALC can be loaded in a JUPYTER NOTEBOOK, not all front end related features may work correctly due to the existing limitations of the JUPYTER interface.

It should be noted that in FEYNCALC 9.3 the internal mechanism to load FEYNARTS and TARCER [23] has been reorganized in the form of add-ons so that instead of $LoadFeynArts and $LoadTARCER one should use

```
In[2]:=  $LoadAddOns={"FeynArts"};
         <<FeynCalc`
```

and

```
In[3]:=  $LoadAddOns={"TARCER"};
         <<FeynCalc`
```

respectively. The old syntax is still supported for compatibility reasons. Since $LoadAddOns is a list of strings, one can equally specify multiple add-ons to be loaded together with FEYNCALC, e. g. $LoadAddOns={"AddOn1", "AddOn2", . . . }.

Apart from FEYNARTS there are also other external packages that can come handy when doing calculations with FEYNCALC. With FEYNRULES [24, 25] it is straightforward to implement new FEYNARTS models by providing a Lagrangian and deriving the Feynman rules automatically. As far as the analytic evaluation of 1-loop integrals is concerned, the FEYNHELPERS [26] add-on provides a convenient interface to the corresponding library of PACKAGE-X.

---

[2]https://github.com/FeynCalc/feyncalc
[3]https://github.com/FeynCalc/feyncalc/commits/hotfix-stable
[4]https://github.com/FeynCalc/feyncalc/wiki/Installation
[5]https://reference.wolfram.com/language/tutorial/UsingATextBasedInterface.html

2.2. Manual installation

FEYNCALC can be also installed manually by downloading the snapshot directly[6], unpacking the zip file and copying the `FeynCalc` directory to `Applications` inside `$UserBase-Directory`[7]. To make the documentation work, one should rename the folder `DocOutput` inside `FeynCalc` to `Documentation`. Then, to avoid warnings when opening FEYNCALC documentation pages on older MATHEMATICA versions, it is useful to disable the corresponding warning messages via

```
In[4]:= SetOptions[$FrontEnd, MessageOptions ->
        {"InsufficientVersionWarning" -> False}]
```

Many FEYNCALC symbols such as 4-vectors, Dirac and color matrices or propagators have extensive typesetting rules attached to them. However, the output can be typeset only when the default cell output format of the front end is set to `TraditionalForm` instead of the default `StandardForm`. It is possible to change this setting only for the current front end session via

```
In[5]:= CurrentValue[$FrontEndSession,
        {CommonDefaultFormatTypes, "Output"}] = TraditionalForm
```

without affecting the global MATHEMATICA settings. The same effect can be achieved by adding the line

```
$FCTraditionalFormOutput=True;
```

to the special configuration file `FCConfig.m` inside the FEYNCALC directory. The file should be created if missing. In general, `FCConfig.m` may contain arbitrary MATHEMATICA code that will be evaluated before loading FEYNCALC. This can be useful to customize the package to the user's requirements.

It is worth noting that FEYNCALC also offers two dedicated commands to switch between `StandardForm` and `TraditionalForm` output in the new cells on the fly. These are `FCEnableTraditionalFormOutput[]` and `FCDisableTraditionalFormOutput[]`.

2.3. Development version

One of our goals is to ensure that when installing FEYNCALC the users always obtain a stable version of the package ready for productive use. For this reason every new feature must be carefully tested and validated before it can be added to the stable branch of the repository. The playground where we can freely test different ideas and approaches without worrying about possible side effects is the `master` branch of the FEYNCALC repository, also known as the *development version* of FEYNCALC. When this version is considered sufficiently stable and ready for the productive use, its content gets merged into the `hotfix-stable` branch, which leads to the release of a new stable version of FEYNCALC.

Users that are interested in testing the development version or even employing it on a regular basis (as practiced by FEYNCALC enthusiasts) can install the code using the same automatic installer but with an additional option `InstallFeynCalcDevelopmentVersion` set to `True`

```
In[6]:= Import@"https://raw.githubusercontent.com/FeynCalc/feyncalc/master/install.m"
        InstallFeynCalc[InstallFeynCalcDevelopmentVersion -> True]
```

---

[6]https://github.com/FeynCalc/feyncalc/archive/hotfix-stable.zip

[7]`$UserBaseDirectory` is a global MATHEMATICA variable that contains the full path to the user base directory. The `Applications` subdirectory of this directory is the standard place where MATHEMATICA looks for packages when the user issues <<PackageName`.

Any feedback on the development version including inquiries about new features and bug reports is highly encouraged and can be submitted via the official mailing list[8] or GITHUB Issues[9].

3. New features in FEYNCALC 9.3

3.1. *Loading* FEYNCALC *together with other packages*

A common issue that often arises when loading several MATHEMATICA packages into the same kernel is known as "shadowing". Suppose that two distinct packages A and B both define a function Foo. Loading both packages sequentially into the same kernel makes Foo simultaneously appear in multiple contexts (A' and B'), which creates unwanted interference effects between A'Foo and B'Foo. For example, when the user enters Foo in the MATHEMATICA front end or command line interface, it will depend on the ordering of A' and B' in the global variable $ContextPath, whether the input will be interpreted as A'Foo or B'Foo.

In general, when creating new packages it is recommended to include some unique identifier (e.g. a shorthand of the package name) to the name of each symbol defined in the package (e.g. aFoo and bFoo), so that overlaps between different packages can be avoided. Unfortunately, MATHEMATICA in no way enforces such naming conventions and many package developers are reluctant to use prefixed symbol names. Thus, when several packages for similar purposes need to be employed in the same MATHEMATICA session, there is always a high chance of running into shadowing issues.

FEYNCALC and FEYNARTS constitute a good example of two packages that, for historical reasons, share a large amount of symbols with same names but different definitions. Here our solution was to patch the source code of FEYNARTS by changing the relevant symbol names and allowing FEYNCALC to work only with the modified version of the package. Another approach was taken in [26], were the relevant packages (PACKAGE-X or FIRE) are deliberately not added to the $ContextPath and the communication between them and the FEYNCALC session is handled using dedicated interfaces. Both workarounds are admittedly not particularly elegant and require permanent maintenance: New versions of the packages may introduce changes that necessitate further adjustments in the patching or interface codes. Given the vast number of existing MATHEMATICA packages for HEP calculations, one can never hope to make all of them compatible with FEYNCALC in a way this was done for FEYNARTS, PACKAGE-X or FIRE.

As a universal workaround, version 9.3 allows the user to temporarily rename the conflicting symbols of FEYNCALC. There are almost no limitations neither on the number of symbols that can be renamed nor on their new names. These changes are not persistent, i.e. they do not modify any files on the disk and apply only to the current FEYNCALC session. This also implies that all codes evaluated in that session must be compatible to the changes that were introduced when loading the modified FEYNCALC. For example, if the user chooses to rename Contract to FCContract, it is mandatory to use the latter name when writing FEYNCALC codes in that session, as Contract would have no definition in the FEYNCALC context.

The technical implementation of this method relies on the new loading mechanism implemented in FEYNCALC 9.3. When the user evaluates <<FeynCalc', only a very small portion (minimal loader) of the code is directly evaluated on the MATHEMATICA kernel. The loader will then import the remaining parts of the FEYNCALC code as strings and subsequently execute them via ToExpression. At this point it is possible to perform the

---

renaming of almost arbitrary FEYNCALC symbols in a very simple way: the code just needs to search for the corresponding name in the strings and replace it accordingly. The list of symbols to be renamed must be specified as a list of replacement rules assigned to the global variable $RenameFeynCalcObjects.

For example, suppose that we want to use FEYNCALC and LITERED [27, 28], a well-known toolkit for multiloop calculations, on the same kernel. Normally, this leads to the shadowing of Factor1, Factor2 and MetricTensor, which happen to be defined in both packages. Using the new renaming mechanism one can circumvent this issue by running

```
In[1]:= $RenameFeynCalcObjects={"Factor1"->"FCFactor1",
            "Factor2"->"FCFactor2", "MetricTensor" -> "FCMetricTensor"};
        <<FeynCalc`
        <<LiteRed`
```

Notice that the ordering of the packages does not matter here, in the sense that one can first load FEYNCALC and then LITERED or vice versa. Similarly one can also have FEYNCALC loaded together with the reduction package FIRE 6 [29]

```
In[2]:= $RenameFeynCalcObjects={"Contract"->"FCContract"};
        <<FIRE6`
        <<FeynCalc`
```

or the spinor-helicity formalism toolkit S@M [30]

```
In[3]:= $SpinorsPath = FileNameJoin[{$UserBaseDirectory, "Applications", "Spinors"}];
        $RenameFeynCalcObjects = {"Schouten" -> "fcSchouten", "Gamma1" -> "fcGamma1",
        "Gamma2" -> "fcGamma2", "Gamma3" -> "fcGamma3"};
        << Spinors`
        << FeynCalc`
```

While this method might cause some confusion, especially when sharing notebooks with codes that run on top of the "standard" or a "modified" version of FEYNCALC, the resulting benefits vastly outweigh any inconveniences. In particular, FEYNCALC 9.3 can be easily loaded together with any number of other MATHEMATICA packages without risking any shadowing issues, provided that these packages do not conflict among themselves. This enormously simplifies the task of integrating FEYNCALC into complicated computational setups relying on multiple packages and reduces the number of potential pitfalls for new users.

3.2. UV divergent parts of Passarino–Veltman functions

One of the most popular applications of FEYNCALC is the reduction of 1-loop tensor integrals into linear combinations of Passarino–Veltman functions [31], which can be then evaluated using publicly available codes such as PACKAGE-X, LOOPTOOLS [13], COLLIER [32–35], QCDLOOP [36], ONELOOP [37] etc.

Sometimes one is interested only in the ultraviolet (UV) divergent parts of the occurring Passarino–Veltman functions, e. g. when calculating renormalization constants using minimal subtraction or performing a quick check for the cancellation of the UV poles in a renormalized amplitude. For functions with up to 4 legs this functionality is already present in PACKAGE-X and can be accessed via the FEYNHELPERS routine PaXEvaluateUV-Part. FEYNCALC 9.3 offers an alternative implementation of this capability that does not rely on PACKAGE-X and works with an arbitrary number of external legs. The new routine PaVeUVPart is based on the algorithm[10] developed by Georg Sulyok [38], who kindly permitted us to integrate his code into FEYNCALC.

---

[10]The MATHEMATICA notebook with the code is contained in the source of the arXiv preprint for [38]. It is self-contained and does not require any external packages.

Since `PaVeUVPart` operates on `PaVe` functions, all 1-loop integrals in the input must be first converted to this notation either by using `ToPaVe` (for scalar integrals) or by employing `TID` with the options `UsePaVeBasis` and `ToPaVe` both set to `True` (for tensor integrals). For example, we can readily obtain the UV divergence of the tadpole $A_0(m^2)$ by issuing

In[1]:= **`PaVeUVPart[PaVe[0, {}, {m^2}]]`**

Out[1]= $-\dfrac{2\,\text{m}^2}{\text{D}-4}$

A less trivial application is the analysis of the fully massive 4-point coefficient function D_{000000} with full kinematic dependence, which immediately returns the correct answer (cf. e. g. [34])

In[2]:= **`PaVeUVPart[PaVe[0,0,0,0,0,0, {p11, p12, p23, p33, p22, p13},`**
      `{m0^2, m1^2, m2^2, m3^2}]]`

Out[2]= $\dfrac{-5\,\text{m0}^2-5\,\text{m1}^2-5\,\text{m2}^2-5\,\text{m3}^2+\text{p11}+\text{p12}+\text{p13}+\text{p22}+\text{p23}+\text{p33}}{480(\text{D}-4)}$

It is important to keep in mind that Passarino–Veltman functions may exhibit not only UV but also infrared (IR) divergences. Poles in $1/\varepsilon$ due to soft or collinear singularities[11] will mix with $1/\varepsilon$ from UV divergences, unless special care was taken to employ different regulators for UV and IR. This is also the reason why the scaleless 2-point function $B_0(0,0,0)$ can be put to zero, despite being proportional to $(1/\varepsilon_{\text{UV}} - 1/\varepsilon_{\text{IR}})$. Since `PaVeUVPart` only deals with UV singularities, it cannot tell anything about possible IR divergences of the occurring Passarino–Veltman functions. For example, a C_0 function is always UV finite, but may exhibit IR poles depending on the kinematic invariants in its arguments. Therefore, here `PaVeUVPart` will always return 0 (which is correct), but the UV finiteness in no way implies that C_0 is also IR finite.

In order to correctly extract UV divergences of log divergent scaleless loop integrals (such as $B_0(0,0,0)$ or $C_{00}(0,0,0,0,0,0)$), that would be otherwise discarded as scaleless, one should use the global switch `$KeepLogDivergentScalelessIntegrals`. This forces FEYN-CALC not to put such integrals to zero and allows the package to consistently disentangle UV and IR divergences at 1-loop, provided that no Integration-By-Parts [39, 40] reductions are employed[12]. For example, one can consider the scaleless tensor integral $\int d^D l\, l^\mu l^\nu / l^6$ and convince oneself that it indeed has a UV pole, that is normally canceled by an IR pole with an opposite sign, making the whole integral vanish

In[3]:= **`$KeepLogDivergentScalelessIntegrals = True;`**
      `TID[FVD[l, μ] FVD[l, ν] FAD[{l, 0, 3}], l, ToPaVe -> True] // PaVeUVPart`

Out[3]= $-\dfrac{2\,\text{i}\,\pi^2 \text{g}^{\mu\nu}}{(\text{D}-4)\,\text{D}}$

The ability to investigate such subtleties with very little effort makes `PaVeUVPart` (especially in conjunction with `$KeepLogDivergentScalelessIntegrals`) a very useful addition to the FEYNCALC toolkit.

3.3. Majorana spinors

Majorana fermions appear in the spectrum of numerous HEP models, ranging from Standard Model extensions to special theories that help to improve our understanding of

---

[11] The double poles $1/\varepsilon^2$ caused by simultaneous soft and collinear singularities are also possible.

[12]Existing IBP codes always mix UV and IR divergences, thus making it impossible to trace back their origin.

QCD. A prominent representative of the latter category is $\mathcal{N} = 4$ Supersymmetric Yang-Mills (SYM) [41, 42], where for suitable observables the principle of maximal transcendentality [43–45] is conjectured to link the highest transcendental weight pieces of pQCD results to those in $\mathcal{N} = 4$ SYM. Even in cases when the principle is not satisfied exactly, the $\mathcal{N} = 4$ SYM and pQCD results may still share the basis of master integrals and be composed of the same set of building block functions. This is why for computationally challenging observables a simpler calculation in $\mathcal{N} = 4$ SYM can be often regarded as the first step towards the full pQCD result. One of the recent examples for the application of this strategy is the analytic fixed-order result for the Energy-Energy correlations [46] in e^+e^- annihilation . The calculation of this event shape variable at NLO in $\mathcal{N} = 4$ SYM was presented in [47] and turned out to be very helpful in the task of tackling the more challenging pQCD case [48, 49]. This story could repeat itself also at NNLO, where the $\mathcal{N} = 4$ SYM result was made available recently [50].

From the technical point of view, evaluation of Feynman diagrams with Majorana particles involves several subtleties that do not arise in calculations with Dirac fermions. First of all, depending on the chosen prescription to handle such particles, one may end up with Feynman rules that contain explicit charge-conjugation matrices and transposed Dirac matrices, which introduces an additional layer of complexity. Second, the determination of the relative sign between diagrams is much less straightforward as in the Dirac case due to the possible violations of the fermion number flow.

A simple and unambiguous set of prescriptions for writing down amplitudes with Majorana spinors that circumvents the above-mentioned issues was presented in [51, 52][13] and subsequently implemented in FEYNARTS. This implies that the amplitudes with Majorana spinors generated by FEYNARTS contain only the usual Dirac matrices γ^μ and γ^5 as well as the four-spinors u, v, $\bar{u}$ and $\bar{v}$, while the relative signs between diagrams have already been fixed[14]. Furthermore, the output is free of transposed Dirac matrices and explicit charge-conjugation matrices.

However, even under these favorable conditions, special care is still required when squaring amplitudes with Majorana spinors. The simplest approach to calculate unpolarized matrix elements squared $|\mathcal{M}|^2$ in diagrams with spinor particles is to apply the standard spin sum formulas

$$\sum_s u(p,s)\bar{u}(p,s) = \slashed{p} + m, \quad \sum_s v(p,s)\bar{v}(p,s) = \slashed{p} - m, \tag{1}$$

thus converting products of closed spinor chains into Dirac traces. Unfortunately, in the presence of Majorana fermions the contributions from interference diagrams may involve terms which do not allow for a direct application of the formulas from Eq. (1). This issue can be resolved by transposing some of the spinor chains and introducing the charge conjugation matrix C with (cf. e. g. [54] for a pedagogical treatment)

$$v^T(p,s) = -\bar{u}(p,s)C, \quad \bar{u}^T(p,s) = C^{-1}v(p,s),$$
$$C\gamma^{\mu T}C^{-1} = -\gamma^\mu, \quad C\gamma^{5T}C^{-1} = \gamma^5. \tag{2}$$

In FEYNCALC 9.3 the default function for rewriting products of closed spinor chains as Dirac traces, FermionSpinSum, can automatically reorder the chains when necessary, as in the following example where formulas from Eq. (1) cannot be applied straightforwardly

$$\bar{u}(k_2)P_R v(k_1) \ \bar{v}(p_1)P_R u(p_2) \ \bar{u}(p_1)P_L u(k_2) \ \bar{u}(p_2)P_L u(k_1) \tag{3}$$

---

[13]cf. also [53] for recent developments in this direction.
[14] In the presence of vertices that contain more than two fermion fields the amplitudes need to be post-processed to build correct fermion chains and fix the signs.

In[4]:= **FermionSpinSum[SpinorUBar[k2, m2].GA[6].SpinorV[k1, m1] * SpinorVBar[p1, m1].**
GA[6].SpinorU[p2, m2] * SpinorUBar[p1, m1].GA[7].SpinorU[k2, m2] * SpinorUBar[
p2, m2].GA[7].SpinorU[k1, m1]]

Out[4]= $-\mathrm{tr}((\bar{\gamma}\cdot\bar{k1}+m1).\bar{\gamma}^6.(m2-\bar{\gamma}\cdot\bar{k2}).\bar{\gamma}^7.(\bar{\gamma}\cdot\bar{p1}-m1).\bar{\gamma}^6.(\bar{\gamma}\cdot\bar{p2}+m2).\bar{\gamma}^7)$

The automatic reordering can be turned off by setting the option SpinorChainTranspose to False. This can be used as a consistency check, since amplitudes involving only Dirac spinors should not require any reordering.

Furthermore, we added a dedicated function SpinorChainTranspose that transposes closed spinor chains as in

In[5]:= **SpinorChainTranspose[SpinorVBar[p1, m1].GA[6].SpinorU[p2, m2]]**

Out[5]= $-(\varphi(-\bar{p2},m2)).\bar{\gamma}^6.(\varphi(\bar{p1},m1))$

Since it is normally not useful to transpose every chain in the expression, the value of the option Select specifies types of the chains that should be processed. By default, these are all chains that start with a $\bar{v}$-spinor and end with a u- or v-spinor, where the spinors may depend on 4 or D-dimensional momenta.

In[6]:= **OptionValue[SpinorChainTranspose, Select] // StandardForm**

Out[6]= {{SpinorVBar[_,_],SpinorU[_,_]},
         {SpinorVBar[_,_],SpinorV[_,_]},
         {SpinorVBarD[_,_],SpinorUD[_,_]},
         {SpinorVBarD[_,_],SpinorVD[_,_]}}

Using patterns one can also provide more sophisticated selection criteria.

Finally, the operation of transposing a chain of Dirac matrices X and sandwiching it between C and C^{-1} is provided through the new function FCChargeConjugateTransposed (abbreviated with FCCCT). By default, the expression remains in the unevaluated form as CX^TC^{-1}

In[7]:= **FCCCT[GA[7].(GS[k2] + m2).GA[6]]**

Out[7]= $C(\bar{\gamma}^7.(\bar{\gamma}\cdot\bar{k2}+m2).\bar{\gamma}^6)^{\mathsf{T}}C^{-1}$

The explicit result can be obtained by setting the option Explicit to True or by applying the function Explicit to the expression

In[8]:= **FCCCT[GA[7].(GS[k2] + m2).GA[6]]//Explicit**

Out[8]= $\bar{\gamma}^6.(m2-\bar{\gamma}\cdot\bar{k2}).\bar{\gamma}^7$

Hence, while FEYNCALC 9.2 would fail to construct the trace for an expression similar to that in Eq. (3), the new version 9.3 not only bypasses these limitations in a transparent way, but also allows the user to explicitly reorder selected spinor chains or to apply the charge conjugation operation to arbitrary strings of Dirac matrices.

3.4. Dirac matrices and spinors with explicit indices

The internal FEYNCALC representation of Dirac algebra is built upon the symbols DiracGamma and Spinor. Both are defined as noncommutative objects and their products must be written using Dot (".", noncommutative multiplication operator) instead of Times ("*", commutative multiplication operator). The Dirac indices are always suppressed, which is sufficient as long as one works only with closed spinor chains and Dirac traces. A single open chain (e.g. $(\gamma^\mu(\not{p}+m)\gamma^\nu)_{ij}$ written as GAD[μ].(GSD[p]+m).GAD[ν]) is still unambiguous even without explicit Dirac indices. However, once the number of the un-contracted Dirac indices exceeds 2 (e.g. in $\gamma^\mu_{ij}\gamma^\nu_{kl}$) ambiguities become unavoidable.

In order to mitigate these shortcomings FEYNCALC 9.3 features an initial support for explicit Dirac indices via special heads `DiracIndex`, `DiracIndexDelta` and `DiracChain`. Then, $\gamma^\mu_{ij}\gamma^\nu_{kl}$ can be directly written as

In[9]:= **DCHN[GA[μ], i, j] DCHN[GA[ν], k, l]**

Out[9]= $(\bar{\gamma}^\mu)_{ij}(\bar{\gamma}^\nu)_{kl}$

where we used the built-in shortcut `DCHN` for `DiracChain`. Internally, FEYNCALC understands `DCHN[GA[$\mu$], i, j]` as

Out[12]//StandardForm=
            DiracChain[DiracGamma[LorentzIndex[μ]],DiracIndex[i],DiracIndex[j]]

The last two arguments of a three argument `DiracChain` may also contain spinors as in

In[13]:= **{DCHN[GS[q], SpinorUBar[p1], i], DCHN[GSD[q], j, SpinorV[p2]], DCHN[GS[q], SpinorUBar[p1], SpinorV[p2]]}**

Out[13]= $\{(\bar{u}(p1).\bar{\gamma}\cdot\bar{q})_i,(\gamma\cdot q.v(p2))_j,(\bar{u}(p1).\bar{\gamma}\cdot\bar{q}.v(p2))\}$

while a `DiracChain` with just two arguments is used to represent spinor bilinears or standalone spinors

In[14]:= **{DCHN[SpinorUBar[p1], i], DCHN[j, SpinorV[p2]], DCHN[SpinorUBar[p1], SpinorV[p2]]}**

Out[14]= $\{(\bar{u}(p1))_i,(v(p2))_j,(\bar{u}(p1).v(p2))\}$

As far as the Kronecker delta in the Dirac space δ_{ij} is concerned, it must be entered as `DIDelta[i, j]`. Of course, several routines have been added to manipulate the new indexed objects in a convenient way. `DiracChainJoin` can be used to contract dummy indices

In[15]:= **DCHN[GA[μ], i, j] DIDelta[j, k] DCHN[GA[ν], k, l]**
         DiracChainJoin[%]

Out[15]= $\delta_{jk}(\bar{\gamma}^\mu)_{ij}(\bar{\gamma}^\nu)_{kl}$

Out[16]= $(\bar{\gamma}^\mu.\bar{\gamma}^\nu)_{il}$

while `DiracChainFactor` is useful to pull non-Dirac symbols out of an indexed Dirac chain

In[17]:= **DCHN[2 x GS[q], i, j]**
         DiracChainFactor[%]

Out[17]= $(2 x \bar{\gamma}\cdot\bar{q})_{ij}$

Out[18]= $2 x (\bar{\gamma}\cdot\bar{q})_{ij}$

To expand sums inside an indexed Dirac chain one can employ `DiracChainExpand`

In[19]:= **DCHN[GS[q] + m, i, j]**
         DiracChainExpand[%]

Out[19]= $(\bar{\gamma}\cdot\bar{q}+m)_{ij}$

Out[20]= $(\bar{\gamma}\cdot\bar{q})_{ij}+m (1)_{ij}$

with `DiracChainCombine` being nearly the inverse of this operation

In[21]:= **DCHN[GA[μ].GS[q], i, j] + m DCHN[GA[μ], i, j]**
         DiracChainCombine[%]

Out[21]= $m (\bar{\gamma}^\mu)_{ij}+(\bar{\gamma}^\mu.(\bar{\gamma}\cdot\bar{q}))_{ij}$

10

Additional functions for working with indexed Dirac chains are planned for the future. Still, already the built-in functionality turns out to be surprisingly powerful when approaching tasks that require manipulations of explicit Dirac indices. For example, FEYN-CALC 9.3 is now able to determine relative signs of amplitudes with 4-fermion vertices generated from suitable FEYNARTS models. The related algorithm was adopted from FORMCALC and implemented in an auxiliary routine `FCFADiracChainJoin`[15].

When converting FEYNARTS output into a proper FEYNCALC expression, `FCFAConvert` will automatically call `FCFADiracChainJoin`, closing the chains accordingly and obtaining correct relative signs, so that no additional user interaction is required. As usual, this behavior can be disabled by setting the `FCFADiracChainJoin` option of `FCFAConvert` to `False`. Needless to say that this level of convenience became possible only after support for indexed Dirac chains became part of FEYNCALC.

3.5. Experimental support for new propagators and tensor types

It is worth mentioning that FEYNCALC 9.3 also contains many new features and capabilities that are still considered not sufficiently stable or even experimental. They are mainly related to the native support for Cartesian tensors, Pauli algebra and loop integrals with nonstandard propagators. The main goal of these efforts is to make FEYNCALC more useful for EFT calculations, with a special focus on EFTs of strong interactions and nonrelativistic EFTs (NREFTs). Although all the corresponding symbols and functions are properly documented and available to the user of the version 9.3, their syntax is still subject to change in the future. It will be finalized with the official release of the FEYNONIUM[16] add-on, that will be described elsewhere [57].

To provide a sneak preview of what is already possible with FEYNCALC 9.3 in this respect, let us give one example of a loop integral manipulation that was completely out of reach in the older versions of the package. Using the new representation of denominators `StandardFeynAmpDenominator` (abbreviated as `SFAD`) we can directly write down an integral with eikonal propagators and simplify it e. g. by applying partial fractioning

```
In[23]:= SFAD[{0, 2 p.q}}, p + q, q]
         ApartFF[%, {q}]
```

Out[23]= $\dfrac{1}{(2\,(p\cdot q) + i\eta))((p+q) + i\eta)^2 . (q^2 + i\eta)}$

Out[24]= $-\dfrac{1}{p^2(q^2+i\eta).((p+q)+i\eta)^2} + \dfrac{1}{p^2(2\,(p\cdot q)+i\eta).(q^2+i\eta)} - \dfrac{1}{p^2(2\,(p\cdot q)+i\eta).((p+q)^2+i\eta)}$

There also exists a Cartesian version of `SFAD` called `CFAD`. Tensor reductions of Cartesian integrals with usual propagators are possible using `TID`, while `FCMultiLoopTID` can process tensor eikonal integrals which often cannot be reduced to scalar integrals with unit numerators. Nonetheless, the treatment of such integrals within FEYNCALC is still very much work in progress and we hope to improve on that in the subsequent versions of the package.

---

[15]One of the authors (V. S.) would like to acknowledge David Straub for bringing his attention to this issue and the suggestion to determine relative signs in the same way as it is done in FORMCALC. Thomas Hahn is acknowledged for helpful explanations regarding the relevant FORMCALC code.

[16]The original idea for this project arose during the work on [55]. Development versions of FEYNCALC and FEYNONIUM were then employed in [56].

4. Efficient usage of FEYNCALC in loop calculations

In this section we would like to address some common mistakes, misconceptions and pitfalls that often affect FEYNCALC users and make it more challenging for them to use the package in real-life calculations. The topics have been chosen from our subjective experience of answering some recurring questions via the official mailing list or in private communication.

4.1. Faster 1-loop tensor reduction

Tensor reduction of 1-loop integrals in FEYNCALC is handled by TID. Unlike the old and currently deprecated OneLoop, the default working mode of TID is to reduce each integral to the scalar Passarino–Veltman functions A_0 (tadpole), B_0 (bubble), C_0 (triangle), D_0 (box) etc. On the one hand, this method is often desirable for analytic studies since it leads to a minimal number of master integrals. On the other hand, doing full reduction of tensor integrals that depend on many kinematic invariants inevitably generates page-long expressions with master integrals being multiplied by very large and complicated prefactors.

Contrary to that, the algorithm implemented in OneLoop performs only a partial reduction, which expresses the result in terms of Passarino-Veltman coefficient functions (e.g. A_{00}, B_1, C_{12} etc.). The coefficient functions can be reduced to the scalar functions by means of PaVeReduce, so that they do not represent the most compact basis of 1-loop integrals. However, their prefactors are just tensors made out of 4-momenta and the metric. There are no huge polynomials of kinematic invariants and the results written in terms of these functions appear very compact. Furthermore, the so-obtained results are equally suitable for numerical and symbolic evaluation.

It is obvious that the full reduction à la TID requires significantly more time than the partial one à la OneLoop, especially if the kinematics is completely generic[17]. As a consequence, we have often received complaints that while the old OneLoop was very fast, TID is very slow. The simple solution to this problem, is to change the working mode of TID, so that it will also perform a partial reduction into coefficient functions. The relevant options is called UsePaVeBasis. Setting it to True as in

In[25]:= **TID[amp, q, UsePaVeBasis->True]**

leads to a very fast tensor reduction and the output is similar to what one would have obtained with OneLoop.

4.2. Treatment of γ^5 in dimensional regularization

The precise treatment of γ^5 in dimensional regularization within FEYNCALC is a recurring question among FEYNCALC users. It is a well-known fact (cf. e.g. [58]) that the definition of γ^5 in 4 dimensions cannot be consistently extended to D dimensions without giving up either the anticommutativity property $\{\gamma^5, \gamma^\mu\} = 0$ or the cyclicity of the Dirac trace.

Over the years people developed numerous prescriptions (schemes) to circumvent these issues and arrive at physical results. Nonetheless, as of today there seems to be no universally accepted solution that can be readily applied to any model at any loop order in a fully automatic fashion. This is mainly related to the fact that calculations with γ^5 are not limited to the algebraic manipulations of Dirac matrices. In addition to that one is usually required to check that the final result preserves all the relevant symmetries, e.g.

---

[17]For performance reasons, it is crucial to specify as much kinematic invariants as possible, before performing the reduction. Doing it the other way around can be orders of magnitude slower.

generalized Ward identities and Bose symmetry. If some of those symmetries are violated due to the chosen γ^5 scheme, they must be restored by hand, e.g. by introducing finite counterterms. Unfortunately, an explicit determination of such counterterms for a given model is a nontrivial task, especially beyond 1-loop[18].

This is why we would like to make clear that this nonalgebraic part of a γ^5-calculation is not the task of FEYNCALC but the duty of the user. One should not naively expect that a mere tool for algebraic calculations such as FEYNCALC can replace physical insight into the problem at hand, especially with such a subtle issue as γ^5. Therefore, the answer to the question whether FEYNCALC could automatically handle γ^5 in such a way, that the user can simply read off the final result without any further thoughts about γ^5 is a clear "no".

The responsibility of FEYNCALC is to ensure that algebraic manipulations of Dirac matrices (including γ^5) are consistent within the chosen scheme. For that purpose FEYNCALC implements two[19] ways to handle γ^5 in D-dimensions.

The Naive or Conventional Dimensional Regularization (NDR or CDR respectively) [63] *assumes* that one can define a D-dimensional γ^5 that anticommutes with any other Dirac matrix and does not break the cyclicity of the trace. For FEYNCALC this means that in every string of Dirac matrices all γ^5 can be safely anticommuted to the right end of the string. In the course of this operation FEYNCALC can always apply $(\gamma^5)^2 = 1$. Consequently, all Dirac traces with an even number of γ^5 can be rewritten as traces that involve only the first four γ-matrices and evaluated directly. The problematic cases are γ^5-odd traces with an even number of other Dirac matrices, where the $\mathcal{O}(D-4)$ pieces of the result depend on the initial position of γ^5 in the string. Using the anticommutativity property they can be always rewritten as traces of a string of other Dirac matrices and one γ^5. If the number of the other Dirac matrices is odd, such a trace is put to zero i.e.

$$\mathrm{Tr}(\gamma^{\mu_1} \ldots \gamma^{\mu_{2n-1}} \gamma^5) = 0, \quad n \in \mathbb{N} \tag{4}$$

If the number is even, the trace

$$\mathrm{Tr}(\gamma^{\mu_1} \ldots \gamma^{\mu_{2n}} \gamma^5) \tag{5}$$

is returned unevaluated, since FEYNCALC does not know how to calculate it in a consistent way. A user who knows how these ambiguous objects should be treated in the particular calculation can still take care of the remaining traces by hand. This ensures that the output produced by FEYNCALC is algebraically consistent to the maximal extent possible in the NDR scheme without extra assumptions.

The other γ^5 scheme available in the package is the Breitenlohner-Maison implementation [64] of the t'Hooft-Veltman [65] prescription, often abbreviated as BMHV, HVBM, HV or BM scheme. In this approach γ^5 is treated as a purely 4-dimensional object, while D-dimensional Dirac matrices and 4-vectors are decomposed into 4- and $D-4$-dimensional components. Following [66] FEYNCALC typesets the former with a bar and the latter with a hat e.g.

$$\gamma^\mu = \bar{\gamma}^\mu + \hat{\gamma}^\mu, \quad p^\mu = \bar{p}^\mu + \hat{p}^\mu \tag{6}$$

The main advantage of the BMHV scheme is that the Dirac algebra (including traces) can be evaluated without any algebraic ambiguities. However, calculations involving tensors from three different spaces (D, 4 and $D-4$) often turn out to be rather cumbersome, even when using computer codes. Moreover, this prescription is known to artificially violate Ward identities in chiral theories, which is something that can be mostly avoided

---

[18]Interesting comments and further references can be found e.g. in Chapter D of [59]. For SM calculations cf. [60, 61].

[19]It is also possible to use Larin's scheme [62] but the underlying code is still poorly optimized and hence rather slow. A complete rewrite is scheduled for the future.

when using NDR. Within BMHV FEYNCALC can simplify arbitrary strings of Dirac matrices and calculate arbitrary traces out-of-the-box. The evaluation of γ^5-odd Dirac traces is performed using the West-formula from [67]. It is worth noting that $D-4$-dimensional components of external momenta are not set to zero by default, as it is conventionally done in the literature. If this is required, the user should evaluate `Momentum[`p_i`,D-4]=0` for each relevant momentum p_i. To remove such assignments one should use `FCClearScalar-Products[]`.

In FEYNCALC 9.3 the mechanism for switching between different schemes was reworked to be more consistent and user friendly. The new standard way to select a scheme is to use the function `FCSetDiracGammaScheme[]`[20]. Currently, it supports following arguments: `"NDR"`, `"NDR-Discard"`, `"BMHV"` and `"Larin"`. Here `"NDR-Discard"` constitutes a special variety of the `"NDR"` scheme where the remaining γ^5-odd traces are set to zero. The name of the current scheme can be requested using `FCGetDiracGammaScheme[]`. NDR still remains the default scheme used in FEYNCALC.

4.3. Taking the correct limit $D \to 4$

When doing calculations in dimensional regularization it is often necessary to promote 4-dimensional quantities (e. g. scalar products or Dirac matrices) to D-dimensional ones and vice versa. Another task that often arises in this context is the expansion of the obtained results around $D=4$ in the regularization parameter ε. Although FEYNCALC is equipped with special functions for automatizing these operations, sometimes the users do not succeed in employing them correctly. Therefore, it appears necessary to reiterate the basics of handling objects living in different dimensions when using the package.

In order to change the dimension of Dirac matrices, 4-vectors, scalar products and other suitable quantities, one should use `ChangeDimension`, where the first argument is the input expression and the second one is the dimension. It is important to stress that `ChangeDimension` does not operate on standalone D-symbols, which often appear in the prefactors of Dirac spinor chains or loop integrals. This can be seen e. g. when applying the function to a D-dimensional integral with a D-dependent prefactor and inspecting the result

```
In[26]:= ChangeDimension[(D-3)/(D-2) FAD[l, l+p] FVD[l, μ] FVD[l, ν], 4] // FCE //
        StandardForm
```

$$
\text{Out[26]}= \frac{(-3+D) \; \text{FAD[l,l+p,Dimension} \to 4] \; \text{FV[l,}\mu] \; \text{FV[l,}\nu]}{-2+D}
$$

where we see that the prefactor has not been changed. This behavior is intended, since FEYNCALC contains a dedicated function that covers the other case: `FCReplaceD` replaces all occurrences of D in the prefactors without altering the dimensions of matrices and tensors. Applying `FCReplaceD` to the expression from the previous example we can immediately observe the difference to the action of `ChangeDimension`

```
In[27]:= FCReplaceD[(D-3)/(D-2)  FAD[l, l+p] FVD[l,μ] FVD[l, ν], D->4-2 Epsilon] // FCE
        // StandardForm
```

$$
\text{Out[27]}= \frac{(1-2 \; \text{Epsilon}) \; \text{FAD[l,l+p]} \; \text{FVD[l,}\mu] \; \text{FVD[l,}\nu]}{2-2 \; \text{Epsilon}}
$$

The biggest advantage of using `ChangeDimension` and `FCReplaceD` as compared to naive replacement rules, is that only the former method allows one to distinguish between prefactors, matrices and tensors. In most cases doing something similar to `exp /. D->4` or `exp /. D->4-2 Epsilon` will not produce the desired result.

---

[20]The usage of old switches `$BreitMaison` and `$Larin` is still supported for compatibility reasons but is considered deprecated. The `$West` switch was removed, but the same effect can be achieved by setting the option `West` of `DiracTrace` to `True` (default) or `False`.

In fact, the naive application of the `D->4` rule to dimensionally regularized expressions appears to be one of the most common mistakes that we observe in the practice of using FEYNCALC in loop calculations. One should always keep in mind that the product of the $\mathcal{O}(\varepsilon)$ piece of the prefactor and the $1/\varepsilon$-pole of a divergent loop integral generates a finite $\mathcal{O}(\varepsilon^0)$ contribution to the final result which cannot be discarded. Schematically, at 1-loop one may often observe terms that look like

$$f(D)\mathcal{I}_D = (f_0 + f_1\varepsilon + \ldots)\left(\frac{c_{-1}}{\varepsilon} + c_0 + \ldots\right). \tag{7}$$

Replacing $f(D)$ with $f(4)$ in this expression is equivalent to dropping the contribution from $f_1 c_{-1}$, while keeping the other finite piece $f_0 c_0$, which is obviously inconsistent. Therefore, naive simplifications of prefactors by setting $D = 4$ instead of expanding the full expression *around $D = 4$* will most likely yield wrong results and should be avoided.

4.4. Debugging FEYNCALC functions

When a FEYNCALC function behaves in a strange way by generating error messages, aborting the evaluation or simply requiring too much time to finish, one might be interested to understand the cause for this behavior without contacting the developers[21]. This can be achieved by activating the debugging output, which is available for most high-level functions (e. g. `TID`, `DiracSimplify`, `Contract` etc.) via the special option `FCVerbose`. For example, evaluating

```
In[28]:=  DiracSimplify[GAD[μ].(GSD[p] + m).GAD[μ], FCVerbose -> 3]
```

will not only return the final result but also generate a large amount of text output related to different stages of the evaluation inside the main function.

In total, there are 3 levels of verbosity, that differ in the amount of information printed in the front end. Setting `FCVerbose` to 1 will merely show internal timings required to complete a particular operation. This can be helpful to localize potential bottlenecks in the code and possibly take countermeasures by changing the relevant options. For example, if some function can quickly obtain the result but gets stuck while collecting the terms with respect to some symbols, one may check if setting the option `Collecting` (when available) to `False` can help to avoid this issue. With `FCVerbose->3`[22] the functions will also output intermediate results of the internal evaluations. This can significantly reduce the performance, as the front end would need to format the (possibly very large) intermediate expressions accordingly. Therefore, this level of verbosity can be useful only when trying to fix a bug in the FEYNCALC code or to understand some anomalous behavior of the package.

4.5. Syntax checks

When FEYNCALC returns obviously wrong results, this may be caused not only by bugs in the code but also by syntax mistakes and inconsistencies in the input expressions. For example, the use of `Times` instead of `Dot` when writing down strings of Dirac or color matrices is a very common mistake among new users. Therefore, it may come as a surprise that in general FEYNCALC happily accepts obviously incorrect inputs such as

```
In[29]:=  DiracSimplify[GA[μ] GA[ν] GA[μ]]
```

or

---

[21]Of course, the developers always appreciate bug reports and interesting examples of performance bottlenecks.

[22]`FCVerbose->2` as an intermediate level between 1 and 3 exists as well, but as of now it is not widely used. Very few functions also support `FCVerbose->4`, which was introduced during hunts for some tough bugs in the past.

In[30]:= `Contract[MT[`μ`, `ν`] FV[`**p**`, `μ`] FV[`**p**`, `μ`]]`

In the past, we received many requests to implement thorough syntax checks that would detect such errors, abort the evaluation and warn the user. Unfortunately, from our experience, such checks are extremely expensive performance-wise and can never achieve a 100% success rate. Therefore, a syntax checker that by default analyzes the input of every high-level function is clearly not a viable solution.

Nonetheless, in order to address some of these issues in FEYNCALC 9.3 we added a new function `FCCheckSyntax` that attempts to validate the given input by searching for the most common input errors, e.g.

In[31]:= `FCCheckSyntax[GA[`μ`] GA[`ν`] GA[`μ`]]`

> FCCheckSyntax::failmsg: Error! FCCheckSyntax has found an inconsistency in your input expression and must abort the evaluation. The problem reads: Commutative products of DiracGamma in $\left(\overline{\gamma}^{\mu}\right)^2 \overline{\gamma}^{\nu}$

While `FCCheckSyntax` obviously cannot detect every possible inconsistency, we are convinced that it can be very helpful for beginners that are not yet fully familiar with the FEYNCALC syntax or have little prior experience with MATHEMATICA. When in doubt, one can always apply `FCCheckSyntax` to the given expression and check if an error message is generated.

5. Summary

In this article we presented a new version FEYNCALC, a well-known MATHEMATICA package for symbolic calculations in QFT and particle physics. Some new features of FEYNCALC 9.3 that should be interesting for a broad audience of practitioners include a practical solution to shadowing issues, native ability to determine the UV-divergent pieces of arbitrary Passarino-Veltman functions, support for calculations with Majorana spinors, introduction of Dirac matrices and spinors with explicit Dirac indices and automatic determination of relative signs for 4-fermion vertices in amplitudes generated with FEYNARTS. Furthermore, in order to promote good practices of employing FEYNCALC in loop calculations, we tried to address some questions that may arise in the daily use of the package and to warn the users against potential pitfalls.

In order to put the amount of new features and the time passed since the previous stable release into perspective, we would like to state that most changes in the code of FEYNCALC between versions 9.2 and 9.3 were related to the still experimental support for Cartesian tensors, Pauli matrices and nonstandard loop integrals that are crucial for making FEYNCALC applicable to nonrelativistic calculations. As the corresponding symbols and routines are not yet finalized, we restricted ourselves to briefly mentioning their existence and referring interested readers to the existing documentation.

To summarize, we are convinced that FEYNCALC can be a very handy tool in the software toolkit of every HEP phenomenologist, be it for professional or educational use. The enormous flexibility in choosing the way how to organize and carry out a particular calculation, many worked out examples and a large, friendly community make FEYNCALC stand out among similar software packages and contribute to its popularity in the particle physics community.

The developers of the package are strongly committed to continue working towards a faster and more feature-rich FEYNCALC, with a special emphasis on EFTs and multiloop calculations. The former should be addressed in the FEYNONIUM project, while for the latter we plan to add a native interface to QGRAF [68] and a set of routines for topology identification and better interoperability with FORM along the lines of the approaches used in [69] and [70].

We are also investigating possibilities to provide a completely web-based version of FEYNCALC that would run on a public and private WOLFRAM ENTERPRISE CLOUD. This could be useful not only as a showcase of FEYNCALC's abilities (e. g. by reproducing known results from the literature) but also for educational purposes.

References

[1] R. Mertig, M. Bohm, and A. Denner, *FEYN CALC: Computer algebraic calculation of Feynman amplitudes*, Comput. Phys. Commun. **64**, 345 (1991).

[2] V. Shtabovenko, R. Mertig, and F. Orellana, *New Developments in FeynCalc 9.0*, Comput. Phys. Commun. **207**, 432 (2016), `arXiv:1601.01167`.

[3] H. H. Patel, *Package-X: A Mathematica package for the analytic calculation of one-loop integrals*, Comput. Phys. Commun. **197**, 276 (2015), `arXiv:1503.01469`.

[4] H. H. Patel, *Package-X 2.0: A Mathematica package for the analytic calculation of one-loop integrals*, Comput. Phys. Commun. **218**, 66 (2017), `arXiv:1612.00009`.

[5] M. Wiebusch, *HEPMath 1.4: A mathematica package for semi-automatic computations in high energy physics*, Comput. Phys. Commun. **195**, 172 (2015), `arXiv:1412.6102`.

[6] J. Alwall *et al.*, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **07**, 079 (2014), arXiv:1405.0301.

[7] G. Cullen *et al.*, *Automated One-Loop Calculations with GoSam*, Eur. Phys. J. **C72**, 1889 (2012), arXiv:1111.2034.

[8] G. Cullen *et al.*, *GOSAM-2.0: a tool for automated one-loop calculations within the Standard Model and beyond*, Eur. Phys. J. **C74**, 3001 (2014), arXiv:1404.7096.

[9] M. Bahr *et al.*, *Herwig++ Physics and Manual*, Eur. Phys. J. **C58**, 639 (2008), arXiv:0803.0883.

[10] T. Gleisberg *et al.*, *Event generation with SHERPA 1.1*, JHEP **02**, 007 (2009), arXiv:0811.4622.

[11] M. Moretti, T. Ohl, and J. Reuter, *O'Mega: An Optimizing matrix element generator*, p. 1981 (2001), arXiv:hep-ph/0102195.

[12] W. Kilian, T. Ohl, and J. Reuter, *WHIZARD: Simulating Multi-Particle Processes at LHC and ILC*, Eur. Phys. J. **C71**, 1742 (2011), arXiv:0708.4233.

[13] T. Hahn and M. Perez-Victoria, *Automatized one loop calculations in four-dimensions and D-dimensions*, Comput. Phys. Commun. **118**, 153 (1999), arXiv:hep-ph/9807565.

[14] T. Hahn, *Generating Feynman diagrams and amplitudes with FeynArts 3*, Comput. Phys. Commun. **140**, 418 (2001), arXiv:hep-ph/0012260.

[15] A. Belyaev, N. D. Christensen, and A. Pukhov, *CalcHEP 3.4 for collider physics within and beyond the Standard Model*, Comput. Phys. Commun. **184**, 1729 (2013), arXiv:1207.6082.

[16] CompHEP, E. Boos *et al.*, *CompHEP 4.4: Automatic computations from Lagrangians to events*, Nucl. Instrum. Meth. **A534**, 250 (2004), arXiv:hep-ph/0403113.

[17] J. A. M. Vermaseren, *New features of FORM*, (2000), arXiv:math-ph/0010025.

[18] J. Kuipers, T. Ueda, J. A. M. Vermaseren, and J. Vollinga, *FORM version 4.0*, Comput. Phys. Commun. **184**, 1453 (2013), arXiv:1203.6543.

[19] D. Fontes and J. C. Romão, *FeynMaster: a plethora of Feynman tools*, (2019), arXiv:1909.05876.

[20] M.-X. Luo, V. Shtabovenko, T.-Z. Yang, and H. X. Zhu, *Analytic Next-To-Leading Order Calculation of Energy-Energy Correlation in Gluon-Initiated Higgs Decays*, JHEP **06**, 037 (2019), arXiv:1903.07277.

[21] T. van Ritbergen, A. N. Schellekens, and J. A. M. Vermaseren, *Group theory factors for Feynman diagrams*, Int. J. Mod. Phys. **A14**, 41 (1999), arXiv:hep-ph/9802376.

[22] A. V. Smirnov, *FIRE5: a C++ implementation of Feynman Integral REduction*, Comput. Phys. Commun. **189**, 182 (2015), arXiv:1408.2372.

[23] R. Mertig and R. Scharf, *TARCER: A Mathematica program for the reduction of two loop propagator integrals*, Comput. Phys. Commun. **111**, 265 (1998), arXiv:hep-ph/9801383.

[24] N. D. Christensen and C. Duhr, *FeynRules - Feynman rules made easy*, Comput. Phys. Commun. **180**, 1614 (2009), arXiv:0806.4194.

[25] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, and B. Fuks, *FeynRules 2.0 - A complete toolbox for tree-level phenomenology*, Comput. Phys. Commun. **185**, 2250 (2014), arXiv:1310.1921.

[26] V. Shtabovenko, *FeynHelpers: Connecting FeynCalc to FIRE and Package-X*, Comput. Phys. Commun. **218**, 48 (2017), arXiv:1611.06793.

[27] R. N. Lee, *Presenting LiteRed: a tool for the Loop InTEgrals REDuction*, (2012), arXiv:1212.2685.

[28] R. N. Lee, *LiteRed 1.4: a powerful tool for reduction of multiloop integrals*, J. Phys. Conf. Ser. **523**, 012059 (2014), arXiv:1310.1145.

[29] A. V. Smirnov and F. S. Chuharev, *FIRE6: Feynman Integral REduction with Modular Arithmetic*, (2019), arXiv:1901.07808.

[30] D. Maitre and P. Mastrolia, *S@M, a Mathematica Implementation of the Spinor-Helicity Formalism*, Comput. Phys. Commun. **179**, 501 (2008), arXiv:0710.5559.

[31] G. Passarino and M. J. G. Veltman, *One Loop Corrections for e+ e- Annihilation Into mu+ mu- in the Weinberg Model*, Nucl. Phys. **B160**, 151 (1979).

[32] A. Denner, S. Dittmaier, and L. Hofer, *Collier: a fortran-based Complex One-Loop LIbrary in Extended Regularizations*, Comput. Phys. Commun. **212**, 220 (2017), arXiv:1604.06792.

[33] A. Denner and S. Dittmaier, *Reduction of one loop tensor five point integrals*, Nucl. Phys. **B658**, 175 (2003), arXiv:hep-ph/0212259.

[34] A. Denner and S. Dittmaier, *Reduction schemes for one-loop tensor integrals*, Nucl. Phys. **B734**, 62 (2006), arXiv:hep-ph/0509141.

[35] A. Denner and S. Dittmaier, *Scalar one-loop 4-point integrals*, Nucl. Phys. **B844**, 199 (2011), arXiv:1005.2076.

[36] R. K. Ellis and G. Zanderighi, *Scalar one-loop integrals for QCD*, JHEP **02**, 002 (2008), arXiv:0712.1851.

[37] A. van Hameren, *OneLOop: For the evaluation of one-loop scalar functions*, Comput. Phys. Commun. **182**, 2427 (2011), arXiv:1007.4716.

[38] G. Sulyok, *A closed expression for the UV-divergent parts of one-loop tensor integrals in dimensional regularization*, Phys. Part. Nucl. Lett. **14**, 631 (2017), arXiv:hep-ph/0609282.

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

[40] F. V. Tkachov, *A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions*, Phys. Lett. **100B**, 65 (1981).

[41] L. Brink, J. H. Schwarz, and J. Scherk, *Supersymmetric Yang-Mills Theories*, Nucl. Phys. **B121**, 77 (1977).

[42] F. Gliozzi, J. Scherk, and D. I. Olive, *Supersymmetry, Supergravity Theories and the Dual Spinor Model*, Nucl. Phys. **B122**, 253 (1977).

[43] A. V. Kotikov and L. N. Lipatov, *DGLAP and BFKL evolution equations in the N=4 supersymmetric gauge theory*, in *35th Annual Winter School on Nuclear and Particle Physics Repino, Russia, February 19-25, 2001*, 2001, arXiv:hep-ph/0112346.

[44] A. V. Kotikov and L. N. Lipatov, *DGLAP and BFKL equations in the N = 4 supersymmetric gauge theory*, Nucl. Phys. **B661**, 19 (2003), `arXiv:hep-ph/0208220`, [Erratum: Nucl. Phys.B685,405(2004)].

[45] A. V. Kotikov, L. N. Lipatov, A. I. Onishchenko, and V. N. Velizhanin, *Three loop universal anomalous dimension of the Wilson operators in N = 4 SUSY Yang-Mills model*, Phys. Lett. **B595**, 521 (2004), `arXiv:hep-th/0404092`, [Erratum: Phys. Lett.B632,754(2006)].

[46] C. L. Basham, L. S. Brown, S. D. Ellis, and S. T. Love, *Energy Correlations in electron - Positron Annihilation: Testing QCD*, Phys. Rev. Lett. **41**, 1585 (1978).

[47] A. V. Belitsky, S. Hohenegger, G. P. Korchemsky, E. Sokatchev, and A. Zhiboedov, *Energy-Energy Correlations in N=4 Supersymmetric Yang-Mills Theory*, Phys. Rev. Lett. **112**, 071601 (2014), `arXiv:1311.6800`.

[48] L. J. Dixon, M.-X. Luo, V. Shtabovenko, T.-Z. Yang, and H. X. Zhu, *Analytical Computation of Energy-Energy Correlation at Next-to-Leading Order in QCD*, Phys. Rev. Lett. **120**, 102001 (2018), `arXiv:1801.03219`.

[49] O. Gituliar and S. Moch, *Fuchsia and Master Integrals for Energy-Energy Correlations at NLO in QCD*, Acta Phys. Polon. **B48**, 2355 (2017), `arXiv:1711.05549`, [PoSRAD-COR2017,038(2017)].

[50] J. M. Henn, E. Sokatchev, K. Yan, and A. Zhiboedov, *Energy-energy correlation in N=4 super Yang-Mills theory at next-to-next-to-leading order*, Phys. Rev. **D100**, 036010 (2019), `arXiv:1903.05314`.

[51] A. Denner, H. Eck, O. Hahn, and J. Küblbeck, *Feynman rules for fermion number violating interactions*, Nucl. Phys. **B387**, 467 (1992).

[52] A. Denner, H. Eck, O. Hahn, and J. Küblbeck, *Compact Feynman rules for Majorana fermions*, Phys. Lett. **B291**, 278 (1992).

[53] M. Paraskevas, *Dirac and Majorana Feynman Rules with four-fermions*, (2018), `arXiv:1802.02657`.

[54] P. B. Pal, *Dirac, Majorana and Weyl fermions*, Am. J. Phys. **79**, 485 (2011), `arXiv:1006.1718`.

[55] N. Brambilla, W. Chen, Y. Jia, V. Shtabovenko, and A. Vairo, *Relativistic corrections to exclusive $\chi_{cJ} + \gamma$ production from e^+e^- annihilation*, Phys. Rev. **D97**, 096001 (2018), `arXiv:1712.06165`.

[56] N. Brambilla, H. S. Chung, W. K. Lai, V. Shtabovenko, and A. Vairo, *Order v^4 corrections to Higgs boson decay into $J/\psi + \gamma$*, Phys. Rev. **D100**, 054038 (2019), `arXiv:1907.06473`.

[57] N. Brambilla, H. S. Chung, V. Shtabovenko, and A. Vairo, *FeynOnium: Using FeynCalc for automatic calculations in Nonrelativistic Effective Field Theories*, TTP19-021, TUM-EFT-92/17, in preparation .

[58] F. Jegerlehner, *Facts of life with gamma(5)*, Eur. Phys. J. **C18**, 673 (2001), `arXiv:hep-th/0005255`.

[59] A. Blondel *et al.*, *Standard model theory for the FCC-ee Tera-Z stage*, in *Mini Workshop on Precision EW and QCD Calculations for the FCC Studies : Methods and Techniques CERN, Geneva, Switzerland, January 12-13, 2018*, Geneva, 2019, CERN, CERN, `arXiv:1809.01830`.

[60] T. L. Trueman, *Spurious anomalies in dimensional renormalization*, Z. Phys. **C69**, 525 (1996), `arXiv:hep-ph/9504315`.

[61] A. Denner and S. Dittmaier, *Electroweak Radiative Corrections for Collider Physics*, (2019), `arXiv:1912.06823`.

[62] S. A. Larin, *The Renormalization of the axial anomaly in dimensional regularization*, Phys. Lett. **B303**, 113 (1993), `arXiv:hep-ph/9302240`.

[63] M. S. Chanowitz, M. Furman, and I. Hinchliffe, *The Axial Current in Dimensional Regularization*, Nucl. Phys. **B159**, 225 (1979).

[64] P. Breitenlohner and D. Maison, *Dimensional Renormalization and the Action Principle*, Commun. Math. Phys. **52**, 11 (1977).

[65] G. 't Hooft and M. J. G. Veltman, *Regularization and Renormalization of Gauge Fields*, Nucl. Phys. **B44**, 189 (1972).

[66] A. J. Buras and P. H. Weisz, *QCD Nonleading Corrections to Weak Decays in Dimensional Regularization and 't Hooft-Veltman Schemes*, Nucl. Phys. **B333**, 66 (1990).

[67] T. H. West, *FeynmanParameter and Trace: Programs for expressing Feynman amplitudes as integrals over Feynman parameters*, Comput. Phys. Commun. **77**, 286 (1993).

[68] P. Nogueira, *Automatic Feynman graph generation*, J. Comput. Phys. **105**, 279 (1993).

[69] F. Feng and R. Mertig, *FormLink/FeynCalcFormLink : Embedding FORM in Mathematica and FeynCalc*, (2012), `arXiv:1212.3522`.

[70] A. K. Cyrol, M. Mitter, and N. Strodthoff, *FormTracer - A Mathematica Tracing Package Using FORM*, Comput. Phys. Commun. **219**, 346 (2017), `arXiv:1610.09331`.

[71] A. Kachanovich, U. Nierste, and I. Nišandžić, *Revisiting evaluation of Higgs to lepton-lepton-gamma*, in preparation .

[72] A. Kachanovich, U. Nierste, and I. Nišandžić, *On the gauge independence of $b \to s$ scalar penguin*, in preparation .