

FeynMaster 2 manual

Duarte Fontes*, Jorge C. Romão†

*Departamento de Física and CFTP, Instituto Superior Técnico
Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal,*

(Dated: September 29, 2021)

We present the manual of FEYNMASTER 2, which contains significant improvements over the previous version of the program. It can be downloaded in <https://porthos.tecnico.ulisboa.pt/FeynMaster/>.

1 Introduction

FEYNMASTER [1] was recently introduced as a multi-tasking software for particle physics studies. Combining FEYNRULES [2,3], QGRAF [4] and FEYNCALC [5–7], FEYNMASTER is able to perform the totality of the following list of tasks:

- a) generation and drawing of Feynman rules;
- b) generation and drawing of Feynman diagrams;
- c) generation of amplitudes;
- d) loop calculations;
- e) algebraic calculations;
- f) renormalization.

It was presented not as competing with other existing software that perform some of the elements of the above list (e.g., refs. [2–21]), but rather as an alternative — with four main advantages. First, FEYNMASTER has a hybrid character concerning automatization: not only does it automatically generate the results, but it also allows the user to act upon them through `MATHEMATICA` notebooks. Second, the complete set of analytical expressions for the counterterms in the modified minimal subtraction ($\overline{\text{MS}}$) scheme can be automatically calculated. Third, FEYNMASTER includes a thorough interaction with numerical calculations through `FORTRAN`, using `LOOPTOOLS` [12] in one-loop calculations. Finally, all the printable outputs of FEYNMASTER — the complete set of Feynman rules (tree-level and counterterms), the Feynman diagrams, as well as a list containing both the expressions and computed counterterms — are automatically written in `LATEX` files.

FEYNMASTER 2 has significant improvements over the previous version; we highlight three of them. First of all, FEYNMASTER 2 is much more practical, since:

- a) a model is now specified by a single model file,
- b) the file controlling the FEYNMASTER run now contains just the data relevant for the run,
- c) the presence of QGRAF has virtually vanished.

The last point means that there is no longer such thing as a QGRAF style or convention, which in turn implies that there is a single set of conventions (the FEYNRULES ones) used throughout the whole program. A second major improvement in FEYNMASTER 2 is that the calculated results can now be automatically stored. That is, a certain process can be calculated once and for all, which considerably simplifies the manipulation of results and the use of FEYNMASTER as a whole. In the third place, the

*duartefontes@tecnico.ulisboa.pt

†jorge.romao@tecnico.ulisboa.pt

renormalization is significantly faster; for example, the generation and impression of the total set of Feynman rules (tree-level and counterterms) for the Standard Model is now completed in just 5 minutes. In addition, several functions were corrected, improved or simply created.

The general usage of FEYNMASTER can be summarized in a few lines: after the user has defined the relevant directories and the model, the FEYNMASTER run is controlled from a single file (`Control.m`). Here, the sequence of processes to study can be chosen, as well as many different options. FEYNMASTER is then ready to run. The run automatically generates and opens several PDF files — according to the options chosen in `Control.m` — and creates the above-mentioned MATHEMATICA notebooks.

The present paper is a complete and auto-sufficient manual of FEYNMASTER 2, and is organized as follows. In section 2, we explain how to download and install FEYNMASTER. Section 3 is devoted to the creation of models, and section 4 to the detailed usage of FEYNMASTER. Then, in section 5, we give some examples. Finally, in section 6 we present a quick first usage of FEYNMASTER.

2 Installation

FEYNMASTER can be downloaded in:

<https://porthos.tecnico.ulisboa.pt/FeynMaster/>.

FEYNRULES, QGRAF and FEYNCALC, essential to run FEYNMASTER, can be downloaded in <https://feynrules.irmp.ucl.ac.be/>, <http://cfif.ist.utl.pt/~paulo/qgraf.html> and <https://feyncalc.github.io/>, respectively.¹ After downloading FEYNMASTER, the downloaded file should be extracted and the resulting folder (named `FeynMaster`) can be placed in any directory of the user's choice. Then, the directories corresponding to FEYNRULES and QGRAF, as well as the one corresponding to the FEYNMASTER output (i.e., the directory where the outputs of FEYNMASTER shall be saved), should be defined. This must be done by writing the appropriate paths after `dirFR`, `dirQ` and `dirFMout`, respectively, in the beginning of the `RUN-FeynMaster` batch file lying inside the `FeynMaster` folder.²

To test FEYNMASTER, jump to section 6, where instructions for a quick first usage are given.

¹The user of FEYNMASTER is supposed to be familiar with both FEYNRULES (in order to define new models) and FEYNCALC (in order to manipulate the final results). There is no need to know QGRAF, since the non-trivial part of this program — the definition of the model — is automatically carried through by FEYNMASTER. We verified that FEYNMASTER runs properly with the latest public versions of FEYNRULES, QGRAF and FEYNCALC — namely, versions 2.3.36, 3.4.2 and 9.3.0, respectively. More instructions on how to download and install QGRAF can be found in <https://porthos.tecnico.ulisboa.pt/CTQFT/node9.html> and in <https://porthos.tecnico.ulisboa.pt/CTQFT/node33.html>. Finally, when using Linux or Mac, the executable QGRAF file should be named `qgraf`.

To run FEYNMASTER, it is also necessary to have PYTHON, MATHEMATICA and L^AT_EX installed; links to download are <https://www.python.org/downloads/>, <http://www.wolfram.com/mathematica/> and <https://www.latex-project.org/get/>, respectively. We tested FEYNMASTER using version 3.6 of PYTHON and version 12.0 of MATHEMATICA. (Note that FEYNMASTER will *not* run if a PYTHON version prior to 3 is used, and it is only guaranteed to properly work if a version of MATHEMATICA not older than 10.3 is used.) As for L^AT_EX, the user is required to update the package database; note also that, in the first run of FEYNMASTER, some packages (like `feynmp-auto` and `breqn`) may require authorization to be installed. Finally, for the numerical calculations, the needed files are written in the FORTRAN 77 format. A FORTRAN compiler is needed, as well as an installation of LOOPTOOLS. The programs were shown to work both with the gfortran and ifort compilers and with LoopTools-2.14 and above.

²These lines are commented in the batch file, and must continue to be so; besides, the beginning of these lines (with `dirFR`, `dirQ` and `dirFMout`) should *not* be erased. By default, the folder with the models for FEYNMASTER is inside the `FeynMaster` folder and is named `Models`; this can be changed by defining a variable `dirFRmod` in the beginning of `RUN-FeynMaster` and setting a path for it. In a similar way, the folder with the models for QGRAF (that is, the folder that shall contain the QGRAF models, which will be automatically generated by FEYNMASTER) is automatically generated and set inside the QGRAF folder with the default name `Models`; as before, this can be changed by defining a variable `dirQmod` in the beginning of `RUN-FeynMaster` and setting a path for it.

3 Creating a new model

FEYNMASTER is a model dependent program: it cannot run without the specification of a model. As already mentioned, one of the major simplifications of FEYNMASTER 2 lies in model building: while in the previous version the specification of each model required two files, in FEYNMASTER 2 only one file is required to fully define a model. This one file — which we now identify as *the* FEYNMASTER model file — is essentially a regular model file for FEYNRULES; as such, it has the termination `.fr`.³ Its name is identified in the following as the *internal name* of the model. This model file must be inside a folder with the same name — that is, with the internal name of the model —, which in turn must be inside the folder with the models for FEYNMASTER.

FEYNMASTER already comes with three models : QED, Scalar QED (SQED) and the Standard Model (SM).⁴ These serve as prototypes, and we highly recommend them as guiding tools in the creation of a new model. Since there are already models available, this section can be skipped in a first utilization of FEYNMASTER.

We now describe in detail the creation of a FEYNMASTER model file. As we just mentioned, this file is essentially a model for FEYNRULES. Yet, it is characterized by special features: while some of the definitions and attributes are common to FEYNRULES, some were specifically designed for FEYNMASTER, as we now show.

Lagrangian

The FEYNMASTER model file must include the Lagrangian of the theory. The Lagrangian must be separated in different parts — each of them corresponding to a different type of interaction — and each of those parts should have a specific name: see Table 1.⁵ The Lagrangian parts involving fermions

| Type of interactions | FEYNRULES name |
|--------------------------------|----------------|
| pure gauge | LGauge |
| fermion-gauge | LFermions |
| Yukawa | LYukawa |
| scalar-scalar and gauge-scalar | LHiggs |
| ghosts | LGhost |
| gauge fixing | LGF |

Table 1: Name of the different Lagrangian parts according to the type of interaction.

(LFermions and LYukawa) can be written either in terms of Dirac fermions or Weyl fermions; in both cases, the Feynman rules will be presented for Dirac fermions.

Parameters

Parameters are defined via the usual FEYNRULES variable `M$Parameters`. Different attributes can be associated to a certain parameter; their order is irrelevant. As in FEYNRULES, the parameters are

³It is assumed here that the user is familiar with FEYNRULES and knows how to create a FEYNRULES model. If this is not the case, we refer to the FEYNRULES website <https://feynrules.irmp.ucl.ac.be/>. Finally, it should be clear that the FEYNMASTER model file can always be used as a FEYNRULES model file.

⁴The SM model file is written in an arbitrary R_ξ gauge and with the η parameters of ref. [22], and it closely follows ref. [23] for renormalization.

⁵It is not mandatory to define all the 6 Lagrangian parts present in Table 1 (for example, if the model does not have a ghost sector, there is no need to define LGhost); only, no other name besides the 6 ones specified in the right column of Table 1 will be recognized by FEYNMASTER.

by default real; otherwise, one must include the attribute `ComplexParameter` and set it to `True` (i.e., `ComplexParameter -> True`). The L^AT_EX name of a parameter should be set with the attribute `TeXName`⁶ and written between double commas in L^AT_EX style (ex: `TeXName -> "\\lambda_2"`).⁷ Parameters can have indices, which are defined as in FEYNRULES (ex: `Indices -> {Index[Gen], Index[Gen]}`).⁸ A numerical value can be given to a parameter through the attribute `NumValue` (ex: `NumValue -> 0.0047`); in the case of parameters with indices, the set of values should be written as a MATHEMATICA list (ex: for a parameter with dimensions 2×2 , one can set `NumValue -> {0.5, 1, 1.5, 2}`, for the matrix entries 11, 12, 21, 22, respectively).⁹ The renormalization rule for a parameter can be defined through the attribute `Renormalization`, in such a way that the rule must be written inside braces (ex: `Renormalization -> {mf -> mf + dmf}`).¹⁰ Finally, parameters corresponding to counterterms should be flagged by setting the attribute `Counterterm` to `True`.¹¹

Particle classes

Particles are defined via the usual FEYNRULES variable `M$ClassesDescription`; again, the order of attributes associated to a certain parameter is irrelevant. The `ClassName` cannot correspond to FEYNMASTER internal indices, such as `J1`, `J2`, `J3`, `J4`, `p1`, `p2`, `q1`, `q2`. The attribute `SelfConjugate` should always be defined, either to `True` or to `False`. Unphysical particles (except Weyl fermions) should be flagged with the attribute `Unphysical` set to `True`. Every propagating particle must have a mass assignment, either to a variable or to 0 (ex: `Mass -> MH` or `Mass -> 0`). Propagating particles should also have a L^AT_EX name (ex: `TeXName -> "c_{W^-}"`); when a particle is not its own complex conjugate, a L^AT_EX name for the antiparticle should also be specified, through the attribute `TeXAntiName` (ex: `TeXAntiName -> "\\bar{c}_{W^-}"`). The decay width can be defined through `DecayWidth` (ex: `DecayWidth -> DWZ`).¹² Weyl components must be specified for Dirac fermions whenever the latter are composed of Weyl particles defined in the model (ex: `WeylComponents -> {uqL, uqR}`). Neutrinos should be flagged with the attribute `Neutrino` as `True`.¹³ Finally, renormalization is defined in a way similar to that of the parameters (ex: `Renormalization -> {H -> H + 1/2 dZH H}`).¹⁴

Extra (optional) information

Besides `M$Parameters` and `M$ClassesDescription`, FEYNMASTER allows extra optional quantities to be defined. These are presented in Table 2 and for each one we present a brief description, the default

⁶Which does not exist in FEYNRULES and should not be confused with the FEYNRULES attribute `TeX`.

⁷The backslash should always be doubled in `TeXName`, as well as in `TeXAntiName` (see below).

⁸In this example, the range of `Gen` must be defined in the model; for example, `IndexRange[Index[Gen]] = Range[3]`.

⁹The attribution of numerical values is useful only in case the user wants to exploit the numerical interface of FEYNMASTER — to be explained in section 4.4.2. The SM model file includes numerical values according to [24].

¹⁰It should be clear that it is not mandatory to define renormalization rules. That is, FEYNMASTER can be used even if no renormalization is performed. Finally, note that, if a parameter is complex, the renormalization of its complex conjugate is automatically considered.

¹¹There can be counterterms for both parameters and fields; since counterterms themselves are parameters, they are defined in the parameters sections `M$Parameters`.

¹²The decay width thus defined will show up in the denominator of the propagator of the particle at stake whenever such propagator mediates the s -channel of a $2 \rightarrow 2$ scattering process. This will also automatically happen for the corresponding Goldstone boson of the particle at stake (in the case it corresponds to a gauge boson) if, in the definition of the particle corresponding to the Goldstone, the attribute `Goldstone` is associated to the gauge boson at stake (ex: `Goldstone -> Z`).

¹³This is relevant only for the definition of number of polarizations in decay or scattering processes. In particular, the functions `DecayWidth` and `DiffXS` (cf. Table 8 below) use this information.

¹⁴In the case of gauge bosons, generic Lorentz indices should be included; ex: `A[mu_] -> A[mu] + 1/2 dZ3 A[mu]`. Renormalization of antiparticles is automatically included and needs not be specified by the user.

value and an example. A few of comments are in order.

| Quantity | Description | Default value | Example |
|------------------------------|---|------------------------|---------------------------------|
| <code>M\$ModelExtName</code> | model name to be printed in the FEYNMASTER final documents | <i>(internal name)</i> | "Standard Model" |
| <code>M\$FCeqs</code> | identities for FEYN CALC | <i>(empty)</i> | { $\xi_A \rightarrow 1$ } |
| <code>M\$FCsimp</code> | simplifications for FEYN CALC | <i>(empty)</i> | { $cw^2 + sw^2 \rightarrow 1$ } |
| <code>M\$PrMassFL</code> | when true, masses of propagators are extracted from the Lagrangean | True | False |
| <code>M\$GFreno</code> | when true, renormalization rules are applied to the Gauge Fixing Lagrangean | False | True |
| <code>M\$RestFile</code> | restrictions file for the FEYNRULES model | <i>(empty)</i> | "MyRest.rst" |
| <code>RenoPreRep</code> | function to rewrite the Lagrangean immediatly before renormalization | <i>(empty)</i> | <i>(cf. SM model file)</i> |

Table 2: Optional variables for the FEYNMASTER model file. See text for details.

First, `M$FCeqs` should be distinguished `M$FCsimp`: while both are defined in the FEYNMASTER model file as lists of replacement rules, the former ends up being converted into a list of equalities for the FEYN CALC, whereas the latter is only used as a set of replacement rules. So, considering the examples presented in Table 2, the FEYN CALC notebook will have `xiA` defined as 1 (i.e., `xiA:=1`), while it will use the rule `cw^2+sw^2->1` to simplify the calculations (more details on section 4.4.2).

Second, if `PrMassFL` is set to **True**, the poles of the propagators are extracted from the Lagrangean — i.e., they are defined as the bilinear terms of the field at stake in the Lagrangean — and the propagator is written in the most general form. If `PrMassFL` is set to **False**, the poles will match the variable corresponding to the mass of the propagator and the propagator will be written in the Feynman gauge. While setting `PrMassFL` to **True** is certainly the most faithful way to describe the propagator, this option may bring certain difficulties: on the one hand, it requires the definition of a Gauge Fixing term for gauge boson propagators; on the other hand, the bilinear terms can be very complicated expressions.

Finally, `RenoPreRep`, if used, must be defined as a `MATHEMATICA` function with a single argument. It is applied to the Lagrangean immediatly before the renormalization process, thus allowing to rewrite it (i.e., the Lagrangean) in a more convenient way. This may be useful to obtain simpler expressions for the Feynman rules of the counterterms.

4 Usage

Once the initial specifications are concluded (i.e., once the directories and the model are defined), FEYNMASTER is ready to be used. In this section, we explain in detail how to use it. We start by showing how to edit the file that controls the FEYNMASTER run. Then, after describing how to run FEYNMASTER,

we comment its outputs and explain how to use the notebooks we alluded to in the Introduction.

4.1 Control.m

As mentioned before, the FEYNMASTER run is uniquely controlled from the `Control.m` file (which lies inside the `FeynMaster` folder). In this section, we explain the different components of this file.

Model selection

First, the model must be chosen; this is done by editing the name after the variable `model` in the beginning of `Control.m`. This name must correspond to the internal name of the model (the same name given to the FEYNMASTER model file).

Process specification

Then, the user should start by specifying the desired process (or processes).¹⁵ One process is specified through the definition of the set of variables shown in Table 3.¹⁶ As before, for each variable, a brief

| Variable | Description | Default value | Example |
|---------------------------|------------------------|----------------|---|
| <code>inparticles</code> | incoming particles | <i>(empty)</i> | <code>e,ebar</code> |
| <code>outparticles</code> | outgoing particles | <i>(empty)</i> | <code>H,Z</code> |
| <code>loops</code> | number of loops | 0 | 1 |
| <code>parsel</code> | intermediate particles | <i>(empty)</i> | <code>{avoid,Z,1,3},{keep,e,1,1}</code> |
| <code>factor</code> | quantity to factor out | 1 | <code>16 Pi^2/I</code> |
| <code>options</code> | options | <i>(empty)</i> | <code>onepi</code> |

Table 3: Variables that specify one process. See text for details.

description, the default value and an example are presented. We now describe them in more detail.

`inparticles` and `outparticles`, corresponding to the incoming and outgoing particles of the process, should contain only particles defined in the FEYNMASTER model. Antiparticles are defined according to the FEYNRULES convention (with the suffix `bar`). Different particles should be separated by commas. Whenever both a particle and an antiparticle are considered, the particle should always be written first, and the antiparticle after it. Tadpoles are obtained by selecting a single incoming particle and no outgoing particles.

Concerning the `loops` variable, it should be clear that, whatever the number of loops, FEYNMASTER will always correctly generate the amplitudes for every diagram involved, although it is only prepared to properly draw and compute diagrams with number of loops inferior to 2. The one-loop calculations are performed with the FEYNMASTER function `OneLoopTID` (cf. Table 8 below).

`parsel` allows the specification of intermediate particles contributing to the process.¹⁷ It applies not only

¹⁵It is possible to define a sequence of processes — that is, a series of processes to be run in a single FEYNMASTER run. To define a sequence of processes, the user must copy the lines of the first process, paste them after it, and edit them to define the second process, and repeat the same procedure for more processes (an example of a sequence of processes will be given in section 5.1).

¹⁶As in Table 2, all variables in Table 3 can be completely omitted from the `Control.m` file, in which case the default values are applied.

¹⁷It is similar to (and actually based on) the `iprop` option in QGRAF.

to particles in loops, but to all intermediate particles. The selections must be written between braces and different selections should be separated by commas (see example in Table 3, where two specific selections are given). Each specific selection contains four arguments: the first should either be `avoid` or `keep`, the second should correspond to a particle of the model,¹⁸ and the last two should be non-negative integer numbers such that the second is not smaller than the first. We illustrate how it works by considering the example in Table 3: `{avoid,Z,1,3}` discards all the diagrams with number of Z propagators between 1 and 3, while `{keep,e,1,1}` keeps only diagrams with number of e propagators between 1 and 1 (i.e., exactly equal to 1).

`factor` is a number, written in `MATHEMATICA` style that is to be factored out in the final calculations (more details on section 4.4.2).

`options` allows all the QGRAF options, as well as a couple of specific option for FEYNMASTER—`ugauge`, which performs the calculation in the unitary gauge, and `noPVauto`, which employs the option `PaVeAutoReduce -> False` in TID.¹⁹ In Table 4, besides these, we present some of the QGRAF options; for each one, a brief description and the converse option are presented.

| Option | Description | Converse option |
|------------------------|---|------------------------|
| <code>ugauge</code> | unitary gauge | <i>(none)</i> |
| <code>noPVauto</code> | employs <code>PaVeAutoReduce -> False</code> in TID | <i>(none)</i> |
| <code>onepi</code> | 1-particle irreducible diagrams only* | <code>onepr</code> |
| <code>onshell</code> | no self-energy insertions on the external lines* | <code>offshell</code> |
| <code>nosigma</code> | no self-energy insertions (nowhere)* | <code>sigma</code> |
| <code>nosnail</code> | no snails (i.e., tadpoles or a collapsed tadpoles)* | <code>snail</code> |
| <code>notadpole</code> | no tadpole insertions, i.e. no 1-point insertions* | <code>tadpole</code> |
| <code>simple</code> | at most one propagator connecting any two different vertices, and no propagator starting and ending at the same vertex* | <code>notsimple</code> |

* Taken from the QGRAF manual; please consult it for more information.

Table 4: Options for `Control.m`.

FeynMaster logical variables

Once the model and the process (or sequence of processes) are specified, the user must select the logical value of the 9 variables present in the end of `Control.m`. As logical variables, they admit only the values `True` (or, alternatively, `T` or `t`) and `False` (or, alternatively, `F` or `f`). We describe them in Table 5. Some remarks are in order, concerning the effect of these variables when set to `True`.

`FRinterLogic`, by establishing an interface with `FEYNRULES`, performs several tasks. First, it runs `FEYNRULES` (for the model selected in the initial variable `model`), prints the complete tree-level Feynman

¹⁸Care should be taken not to select antiparticles, but only particles. This is because the propagator in FEYNMASTER is defined through the particle, and not the antiparticle.

¹⁹As shall be seen, loop integrals in FEYNMASTER are performed with the function `OneLoopTID`, which uses the `FEYNCALC` function `TID`. By default, `OneLoopTID` uses `TID` with the option `PaVeAutoReduce -> True`. This simplifies some special cases of Passarino-Veltman functions. In some cases, however, it may be relevant to perform loop integrals in a more expanded way, in which case the option `PaVeAutoReduce -> False` must be used by selecting the option `noPVauto` in `Control.m`.

| Variable | Effect (when chosen as <code>True</code>) |
|---------------------------|--|
| <code>FRinterLogic</code> | establish an interface with <code>FEYNRULES</code> |
| <code>RenoLogic</code> | perform renormalization |
| <code>Draw</code> | draw and print the Feynman diagrams |
| <code>Comp</code> | compute the final expressions |
| <code>FinLogic</code> | print the final result of each diagram |
| <code>DivLogic</code> | print the UV divergent part of each diagram |
| <code>SumLogic</code> | compute the sum of the expressions |
| <code>MoCoLogic</code> | apply momentum conservation |
| <code>LoSpinors</code> | include spinors |

Table 5: Logical variables of `Control.m`. See text for details.

rules of the model in a PDF file and opens this file.²⁰ Second, it generates the QGRAF model file, a crucial element in the generation of Feynman diagrams. Third, it generates the complete tree-level Feynman rules in `FEYN CALC` style, which will play a decisive role in all the calculations. Finally, it generates a `MATHEMATICA` notebook specifically designed to run `FEYNRULES` — hereafter dubbed the `FEYNRULES` notebook. This notebook is useful in case the user wants to have control over the generation of Feynman rules, and is the subject of section 4.4.1.

Note that, even if all logical variables are set to `False`, `FEYNMASTER` always performs some actions. First, it generates a `MATHEMATICA` notebook specifically designed to run `FEYN CALC` — hereafter dubbed the `FEYN CALC` notebook. This notebook is very useful should the user want to have control over calculations, and is the subject of section 4.4.2. Second, once there is a QGRAF model, `FEYNMASTER` always runs QGRAF, which writes in a symbolic form the total diagrams that contribute to the process at stake — the same process which was specified through the variables in Table 3. Finally, `FEYNMASTER` takes the QGRAF output and writes the amplitude for each diagram in a file that the `FEYN CALC` notebook shall have access to.

`RenoLogic` concerns the renormalization of the model. If `FRinterLogic` is set to `True`, `RenoLogic` prints the complete set of Feynman rules for the counterterms interactions in a PDF file and opens this file; moreover, it stores those interactions in a file which the `FEYN CALC` notebook shall have access to. A second important feature of `RenoLogic` is described below, in the context of the `Comp` variable.

`Draw` takes the QGRAF output, draws the Feynman diagrams in a `LATEX` file, prints them in a PDF file and opens this file. This operation is achieved with the help of `feynmf` [25], a `LATEX` package to draw Feynman diagrams. Since the diagrams are written in a `LATEX` file, they can not only be edited by the user, but also directly copied to the `LATEX` file of the user’s paper.²¹

²⁰The results are automatically written; this is especially challenging when it comes to (automatically) breaking the lines in a long equation. This challenge is in general surpassed with the `LATEX` `breqn` package, which `FEYNMASTER` employs. However, `breqn` is not able to break a line whenever the point where the line is to be broken is surrounded by three or more parentheses; in those cases, unfortunately, the lines in the `FEYNMASTER` PDF outputs simply go out of the screen. For documentation on the `breqn` package, cf. <https://www.ctan.org/pkg/breqn>.

²¹As already suggested, `Draw` is at present only guaranteed to properly draw the diagrams up to one-loop. Moreover, diagrams with more than two particles in the initial or final states, as well as some reducible diagrams, are also not warranted.

`Comp` computes the final expressions using `FEYN CALC` and stores them in a file.²²

`FinLogic` and `DivLogic` are nothing but options for `Comp`, so that their value is only relevant if `Comp` is set to `True`. If at least one of them is set to `True`, a PDF file is generated and opened: when `FinLogic` is selected, the file includes the (total) final analytical expression for each diagram; when `DivLogic` is selected, it includes the analytical expression for the UV divergent part of each diagram.²³

At this point, we should clarify the difference between UV divergences and infrared (IR) divergences. It is well known that, while the former are in general present in loop integrals, the latter can only show up when there is a massless particle running inside the loop (in which case the IR divergence comes from the integration region near $k^2 = 0$, with k the loop momentum). In the present version of `FEYNMASTER`, we restrict the treatment of divergences to the UV ones. Indeed, we assume that the IR divergences can be regulated by giving the massless particle a fake mass — which one shall eventually be able to set to zero in physical processes, after considering real emission graphs. With that assumption, IR divergences will never show up explicitly (only implicitly through the fake mass). In the following, unless in potentially dubious statements, we will stop writing UV explicitly: it is assumed that, whenever we mention divergences, we shall be referring to UV divergences.

`SumLogic` is relevant both when `Comp` is `True` and when it is `False`. In the first case, it calculates the sum of the analytical expressions (when the PDF file with the analytical expressions is generated, `SumLogic` prints their sum).²⁴ The situation where `SumLogic` is `True` and `Comp` is `False` shall be described in section 4.4.2.

We now explain the effect of `RenoLogic` when `Comp` is set to `True`. In case the user defined a single process in `Control.m`, `RenoLogic` causes `FEYNMASTER` to look for counterterms that might absorb the divergences of the process at stake, and to calculate those counterterms in $\overline{\text{MS}}$.²⁵ Such counterterms are then stored in a file (`CTfin.m`, described below) and, should the PDF file with the analytical expressions be generated, printed in such file. In case the user defined a sequence of processes, the subsequently computed counterterms are added to `CTfin.m`; however, what is particularly special about the sequence is that, for a certain process of the sequence, `FEYNMASTER` will compute the counterterms by making use of the counterterms already computed in the previous processes.²⁶ In the end of the run, `CTfin.m` contains all the counterterms that were computed (in $\overline{\text{MS}}$) to absorb the divergences of the processes of the sequence. In this way, and by choosing an appropriate sequence of processes, it is possible to automatically renormalize the whole model in $\overline{\text{MS}}$ with a single `FEYNMASTER` run.

As `SumLogic` described above, the last two logical variables of Table 5 — `MoCoLogic` and `LoSpinors` — are relevant both when `Comp` is `True` and when it is `False`. `LoSpinors` — only relevant with external fermions

²²By ‘final expressions’ we mean the analytical expressions for the diagrams; these are written in terms of Passarino–Veltman integrals in case the number of loops equals 1; more details on section 4.4.2. It is normal that the warning ‘*front end is not available*’ shows up when `Comp` is set to `True`.

²³The limitation we alluded to in note 20 applies here too.

²⁴More specifically: if `FinLogic` is `True`, `SumLogic` includes in the PDF file the sum of the total final expressions; if `DivLogic` is `True`, it includes the sum of the expressions for the divergent parts; if both are `True`, it includes both the sum of the total expressions and the sum of the expressions for the divergent parts.

²⁵That is, calculates them in such a way that the counterterms are precisely equal to the divergent part they absorb (except for the $\ln(4\pi)$ and the Euler-Mascheroni constant γ , which are also absorbed in the $\overline{\text{MS}}$ scheme). By ‘calculating’ we mean here writing the analytical expression.

²⁶This is, in fact, the major advantage of writing a series of processes in a single `FEYNMASTER` run (as opposed to one process per run); indeed, this feature is not possible with one process per run, since `FEYNMASTER` rewrites `CTfin.m` for the model at stake everytime `Comp` and `RenoLogic` are both set to `True`.

— is actually prior to any calculation, since it modifies the amplitudes themselves, including spinors in them. When `Comp` is `True`, `MoCoLogic` is such that the calculations use momentum conservation. The situation where `MoCoLogic` is `True` and `Comp` is `False` is also postponed to section 4.4.2.

Tips

When using FEYNMASTER, it can be convenient to have access to some tips in order to properly fill in `Control.m`. Several such tips are available: in the command line, going to the directory corresponding to the `FeynMaster` folder, one can run the `RUN-FEYNMASTER` batch file with an argument after it. If this argument is simply `help`, then several instructions about the use of `Control.m` are printed in the command line. If, instead, the argument corresponds to internal name of a `FeynMaster` model, all the particles of the model are printed.²⁷ If the argument is neither `help` nor the name of a `FeynMaster` model, all the particles of the model currently selected in `Control.m` are printed.

4.2 Run

After editing the `Control.m` file, everything is set. To run FEYNMASTER, just run the `RUN-FEYNMASTER` batch file inside the `FeynMaster` folder (with no arguments after it).²⁸

4.3 Outputs

Depending on the logical value of the variables of Table 5, FEYNMASTER can have different outputs. We now list the total set of outputs, assuming that all those variables are set to `True`.²⁹ First, in the directory where the FEYNMASTER model is, two files are generated: the FEYNRULES notebook (`Notebook.nb`) and an auxiliary file for it (`PreControl.m`). Second, the QGRAF model with the internal name of the model is created, and placed inside the folder with the models for QGRAF (cf. note 2); besides, a file named `last-output` (with the last output from QGRAF) is created inside the directory corresponding to QGRAF. Then, if it does not exist yet, a directory with the internal name of the model is created inside the directory for the FEYNMASTER output. Inside it, and if they do not exist yet, three directories are created, `Counterterms`, `FeynmanRules` and `Processes`, which we now describe.

`Counterterms` contains one folder, `TeXs-drawing`, and two files, `CTini.m` and `CTfin.m`. `TeXs-drawing` is where the PDF file with the complete set of Feynman rules for the counterterms interactions is stored, as well as the \LaTeX file that creates it. `CTini.m` is the file which the FEYNCALC notebook has access to and where the Feynman rules for the counterterms interactions are stored. `CTfin.m`, in turn, is the aforementioned file containing the counterterms that were computed (in $\overline{\text{MS}}$) to absorb the divergences of the processes of the sequence at stake.

`FeynmanRules`, besides several auxiliary files to be used in the FEYNCALC notebook, contains yet another `TeXs-drawing` folder, where the PDF file with the complete set of Feynman rules for the tree-level

²⁷Antiparticles are not shown, and are obtained simply by adding `bar` after the name of the corresponding particle.

²⁸Care should be taken not to run FEYNMASTER when the relevant notebooks are open. More specifically, if `FRinterLogic` is set to `True`, and if the FEYNRULES notebook created for the process at stake already exists, this notebook cannot be open during the run; in the same way, if `Comp` is set to `True`, and if the FEYNCALC notebook designed for the process at stake already exists, such notebook cannot be open during the run.

²⁹Actually, when `Comp` and `RenoLogic` are both `True` and there are external fermions, `LoSpinors` should be `False`. This is irrelevant for what follows, since `LoSpinors` has no influence on the outputs as a whole.

interactions is stored, as well as the \LaTeX file that creates it.

`Processes` contains a folder for each of the different processes studied. These folders are named with the index (in the sequence of processes) corresponding to the process at stake, as well as with a string containing the names of the incoming and the outgoing particles joined together. Inside each folder, there are three other folders, `Lists`, `TeXs-drawing` and `TeXs-expressions`, as well as three files, `Amplitudes.m`, `Helper.m` and the FEYNALC notebook, `Notebook.nb`. In order: `Lists` contains files where the analytical expressions for the process at staked are stored (more details below); `TeXs-drawing` contains the PDF file with the printed Feynman diagrams, as well as the \LaTeX file that creates it; `TeXs-expressions` contains the PDF file with the printed expressions, as well as the \LaTeX file that creates it; `Amplitudes.m` contains the amplitudes for the diagrams (written in FEYNALC style); `Helper.m` is an auxiliary file for the FEYNALC notebook.

Finally, recall that, in case there is already a QGRAF model, FEYNMASTER will run even if all variables of Table 5 are set to `False`. This is relevant since it generates not only the QGRAF output (`last-output`), but also the folder (or folders) for the specific process (or processes) selected, containing the files described above.³⁰

4.4 The notebooks

As previously mentioned, a major advantage of FEYNMASTER is its hybrid character concerning automatization. Indeed, not only does it automatically generate the results, but it also allows the user to handle them. This is realized due to the automatic creation of the FEYNRULES notebook and the FEYNALC notebook. We now describe them in detail.

4.4.1 The FeynRules notebook

We mentioned in section 4.1 that, when FEYNMASTER is run with the logical variable `FRinterLogic` set to `True`, the FEYNRULES notebook `Notebook.nb` is automatically created in the directory where the FEYNMASTER model is. By running it, the user can access the vertices for the different Lagrangean parts, according to Table 6.³¹ Besides the usual FEYNRULES instructions, two useful functions — `GetCT` and

| Lagrangean part | vertices |
|------------------------|----------------------------------|
| <code>LGauge</code> | <code>vertsGauge</code> |
| <code>LFermions</code> | <code>vertsFermionsFlavor</code> |
| <code>LYukawa</code> | <code>vertsYukawa</code> |
| <code>LHiggs</code> | <code>vertsHiggs</code> |
| <code>LGhost</code> | <code>vertsGhosts</code> |

Table 6: Names of the different vertices according to the Lagrangean part (compare with Table 1).

`MyTeXForm` — are available. `GetCT` is a function that, for a certain Lagrangean piece given as argument, yields the Feynman rules for the respective counterterms.³² `MyTeXForm` is FEYNMASTER’s version of

³⁰While the QGRAF output is not overwritten when QGRAF is run on its own, it is overwritten when QGRAF is run inside FEYNMASTER.

³¹The run will generate several FEYNMASTER internal files, among which is `built-model`, the QGRAF model file.

³²In order to clarify the particles involved in the Feynman rule at stake, each term includes a factor η_i for each particle i ; for example, a term in the output of `GetCT` including $\eta_{A^2} \eta_{WP} \eta_{WPbar}$ contributes to the Feynman rule for the counterterm of `A`, `A`, `WP`, `WPbar`.

`MATHEMATICA`'s `TeXForm`; it is a function that uses `PYTHON` (as well as inner `FEYNMASTER` information concerning the `LATEX` form of the parameters of the model) to write expressions in a proper `LATEX` form.³³

4.4.2 The FeynCalc notebook

Whenever `FEYNMASTER` is run, and independently of the logical values of the variables of Table 5, the `FEYNCALC` notebook is automatically created. This notebook, as already the `FEYNRULES` one just described, is totally ready-to-use: the user does not have to define directories, nor import files, nor change conventions. Just by running the notebook, there is immediate access to a whole set of results: not only to some basic elements — such as the total Feynman rules for the model and amplitudes for the diagrams —, but also to the totality of the results obtained should the `Comp` logical variable had been turned on. This last feature is made possible in `FEYNMASTER 2` due to the creation of lists with analytical expressions. Indeed, when `Comp` is set to true, `FEYNMASTER` now stores the analytical expressions inside the aforementioned folder `Lists`, in such a way that, after the run, the `FEYNCALC` notebook has immediate access to those expressions (more details below).

But this is just part of the flexibility involved in the `FEYNCALC` notebook. In fact, since all the referred results are written in a `MATHEMATICA` notebook, the user has great control over them, as he or she can operate algebraically on them, or select part of them, or print them into files, etc. Moreover, since the `FEYNCALC` package is loaded, and since all the results are written in a `FEYNCALC`-readable style, the control at stake is even greater, for the user can apply all the useful tools of that package: operate on the Dirac algebra, perform contractions, solve loop integrals, etc.³⁴

We now present some useful features introduced by `FEYNMASTER` in the `FEYNCALC` notebook. We start with the variables related to the analytical expressions for the Feynman diagrams: see Table 7. We

| Variable | Meaning |
|----------------------|---|
| <code>amp</code> | list with all the amplitudes |
| <code>ampi</code> | amplitude for diagram i |
| <code>res</code> | list with all the final expressions |
| <code>resi</code> | final expression for diagram i |
| <code>resD</code> | list with all the expressions for the divergent parts |
| <code>resDi</code> | expression for the divergent part of diagram i |
| <code>restot</code> | sum of all the final expressions |
| <code>resDtot</code> | sum of all the expressions for the divergent parts |

Table 7: Useful variables concerning expressions for the diagrams. See text for details.

should clarify the meaning of final expression, corresponding to the `res` list: for a certain index j , `resj` (or `res[[j]]`) takes the amplitude `ampj`, divides it by i , rewrites the loop integral in terms of Passarino–Veltman integrals (in case it is a one-loop process), writes it in 4 dimensions — including possible finite

³³`MyTeXForm` prints the `LATEX` form of the expression at stake not only on the screen, but also in an external file named `MyTeXForm-last-output.tex` in the directory where the notebook lies.

³⁴In the following, we assume the user to be familiar with `FEYNCALC`. For more informations, consult the `FEYNCALC` website: <https://feyncalc.github.io/>.

parts coming from this conversion³⁵ — and factorizes the previously selected `factor`. Note that, due to the division by i , the final expressions — that is, `res` — correspond to \mathcal{M} , and not to $i\mathcal{M}$. The divergent parts are written in terms of the variable `div`, defined as:³⁶

$$\text{div} = \frac{1}{2} \left(\frac{2}{\epsilon} - \gamma + \ln 4\pi \right).$$

Let us now consider the details related to the `Lists` folder. When `Comp` is set to `True`, the lists `res` and `resD` are calculated and stored in `Lists` in the files `res.in` and `resD.in`, respectively. If, in addition, `SumLogic` is set to `True`, `restot` and `resDtot` are also calculated and stored in `Lists` in the files `restot.in` and `resDtot.in`, respectively. The access of the FEYN CALC notebook to these files depends on the variable `compNwrite`, which is visibly defined in the middle of the notebook: when it is `False`, the notebook will load the files stored in `Lists` (should they exist); when it is `True`, it will calculate them anew. By default, `compNwrite` is `False`, which means that, after having been calculated with `Comp`, the analytical expressions are by default immediately accessible to the FEYN CALC notebook.

Next, we consider useful functions to manipulate the results: see Table 8. We now explain some of them in more detail (note that a description of a certain function `Func` shows up in the notebook by writing `?Func`).

Concerning `DecayWidth` and `DiffXS`, while the former is written solely in terms of masses, the latter is written also in terms of the center of momentum energy `S`, as well of the scattering angle `Theta`. Both functions take by default one simple argument, corresponding to the expression (say, \mathcal{E}) which the function at stake is to be applied to; the final results are then proportional to $|\mathcal{E}|^2$. `DecayWidth` also accepts an argument corresponding to a two-element list (e.g., `DecayWidth[{\mathcal{E}_0, \mathcal{E}_1}]`); in this case, the result is proportional to $2 \operatorname{Re} [\mathcal{E}_0 \mathcal{E}_1^*]$.

`GetDirac` and `RepDirac` are useful functions to handle expressions with external fermions. The former yields a list containing the different Dirac structures of the expression given as argument; the latter replaces the Dirac structures of the expression given as argument with non-matricial elements `ME[j]` — that is, the first Dirac structure is replaced by `ME[1]`, the second by `ME[2]`, and so on —, thus simplifying the manipulation of the expression. `RepDirac` admits a second (optional) argument, consisting of a list of Dirac structures (ex: `{GA[p1], GS[q1]}`); this can be relevant to establish a correspondance between `ME[j]` elements and Dirac structures. Indeed, such correspondance can obey one of two rules: either only one argument is given to `RepDirac`, in which case the `ME[j]` elements follow the order of the Dirac structures yielded by `GetDirac`, or a list of Dirac structures is given as a second argument of `RepDirac`, in which case the `ME[j]` elements follow the order of that list.

`GetDiv` yields the UV divergent part of an expression; care should be taken if IR divergences are not regulated via a fake mass, for in that case, although they can show up as poles of the Passarino–Veltman

³⁵As is well known, in the dimensional regularization scheme, the infinities are tamed by changing the dimensions of the integrals from 4 to d , in such a way that the divergences are regulated by the parameter $\epsilon = 4 - d$. When solving the integrals in terms of Passarino–Veltman integrals, the result will in general depend explicitly on the dimension d , as well as on the Passarino–Veltman integrals themselves — which usually diverge, with divergence proportional to $1/\epsilon$. But since $d = 4 - \epsilon$, there will in general be finite terms (order ϵ^0) coming from the product between d and the divergent parts in the Passarino–Veltman integrals. Hence, when converting the result back to 4 dimensions (since the final result is written in 4 dimensions), one cannot forget to include such terms. Finally, recall that IR divergences will never show up explicitly if the potentially IR divergent integrals are tamed by giving the massless particle a fake mass.

³⁶In the PDF file with the printed expressions, we change the name `div` to ω_ϵ .

| Function | Action |
|---------------------------|--|
| <code>ChangeTo4</code> | change to 4 dimensions |
| <code>DecayWidth</code> | calculate the decay width [*] |
| <code>DiffXS</code> | calculate the differential cross section [†] |
| <code>GetDirac</code> | yield the Dirac structures as a list |
| <code>GetDiv</code> | get the divergent part of an expression |
| <code>GetFinite</code> | get the finite part of an expression in dimensional regularization [‡] |
| <code>FacToDecay</code> | rewrite expressions (with form factors) for decays ^{*◊} |
| <code>FCtoFT</code> | convert expressions to <code>FORTRAN</code> |
| <code>MyTeXForm</code> | write expressions in a proper <code>L^AT_EX</code> form |
| <code>MyPaVeReduce</code> | apply <code>FEYNALC</code> 's <code>PaVeReduce</code> and convert to 4 dimensions |
| <code>OneLoopTID</code> | solve one-loop integrals with <code>FEYNALC</code> TID decomposition method [#] |
| <code>RepDirac</code> | replace the Dirac structures with elements <code>ME[j]</code> |
| <code>TakeReal</code> | applies the operator <code>Re</code> to the Passarino-Veltman functions |
| <code>TrG5</code> | calculate the trace for expressions with γ_5 |

^{*} Only applicable to processes with 1 incoming and 2 outgoing particles.

[†] Only applicable to processes with 2 incoming and 2 outgoing particles.

[‡] Cf. note 35.

[◊] Not yet available for three external gauge bosons.

[#] Note the arguments: `OneLoopTID(k, amp)`, with `k` the loop momentum and `amp` the amplitude. Cf. also note 19.

Table 8: Useful functions. See text for details.

functions, they will not be detected by `GetDiv`. Moreover, `GetDiv` only yields the UV divergent part of the Passarino–Veltman functions [26, 27] that `FEYNALC` and `FEYNMASTER` can handle — namely, integrals whose power of the loop momenta in the numerator is at most 3 (except for the D functions, where we extended `FEYNALC` up to the fourth power of the loop momenta).

`FacToDecay` simplifies expressions to render the calculation of the decay width easier. It yields a two-element list: the first element is the expression given as argument, but rewritten in terms of form factors; the second is a list of replacements, associating to each form factor the corresponding analytical expression (written using the kinematics of the process).³⁷

`FCtoFT` is the function that allows the numerical interface of `FEYNMASTER`; when applied to an expression `exp`, it generates a directory named `newfiles`, inside which there are at least five files: `MainFT.F`, `Myexp.F`, `MyVariables.h`, `MyParameters.h` and `Makefile-template`.³⁸ The first one, `MainFT.F`, is the beginning of a main `FORTRAN` program, which must be completed according to the user's will. `MainFT.F` calls the function `Myexp`, which is the `FORTRAN` version of the expression `FCtoFT` was applied to, and which is written in the file `Myexp.F`.³⁹ In turn, `MyVariables.h` and `MyParameters.h` respectively contain the

³⁷The form factors are defined as complex parameters in the options of the `FEYNALC` function `ComplexConjugate`. This will be relevant to calculate the `DecayWidth`.

³⁸Here, and in what follows, `exp` (as in `Myexp.F`) corresponds to the name of the expression.

³⁹According to the size of the expression at stake, and in order to render the compilation faster, one or several files named `MyAux_exp_i.F` (with `i` corresponding to the index of the file) may be generated; these files are consistently and automatically called by `Myexp.F`.

variables and the numerical values associated to the different parameters in the FEYNMASTER model file. Finally, `Makefile-template` consists of a skeleton of a makefile.⁴⁰ In the case of cross sections, the integration routine `IntGauss.f` is also generated and called in `MainFT.F`, which in this case is completed with a concrete example.⁴¹

`MyTeXForm` is the same function as the one described in section 4.4.1.

`MyPaVeReduce` is FEYNMASTER’s version of FEYNCALC’s `PaVeReduce`; it applies `PaVeReduce` and writes the result in 4 dimensions — again, not without including possible finite parts coming from this conversion.

Finally, we consider `TrG5`. Since FEYNMASTER is prepared to compute divergent integrals — and, more specifically, to compute them via dimensional regularization —, it defines Dirac and Lorentz structures (like $g^{\mu\nu}$ or γ^μ) in dimension d , not in dimension 4. However, the definition of γ_5 in dimension d is not trivial, as chiral fermions are a property of four dimensions. In fact, the treatment of γ_5 in dimensional regularization is still an open problem (see e.g. refs. [28–34]). By default, FEYNMASTER assumes the so-called naive dimensional regularization scheme [32], which takes the relation $\{\gamma_5, \gamma^\mu\} = 0$ to be valid in dimension d . This naive approach is applied both when the loop diagram has external fermions, and when it has only inner fermions (forming a closed loop). In the second case, in order to calculate the corresponding trace, FEYNMASTER uses `TrG5`. This function starts by separating the expression it applies to in two terms: one with γ_5 , another without γ_5 . It then computes the trace of the former in 4 dimensions, while keeping the dimension of the latter in its default value d .⁴²

Having clarified the functions in Table 8, we must consider a new table, Table 9, which contains some useful variables concerning renormalization. Two notes should be added. First, in the `CTprocess` variable, `process` corresponds to the names of the incoming and the outgoing particles joined together (for example,

| Function | Meaning |
|-------------------------|--|
| <code>CTprocess</code> | expression containing the total counterterm for the <code>process</code> at stake |
| <code>PreResReno</code> | sum of the total divergent part and <code>CTprocess</code> |
| <code>CTfinlist</code> | list with all the counterterms computed so far in $\overline{\text{MS}}$ |
| <code>ResReno</code> | the same as <code>PreResReno</code> , but using counterterms previously stored in <code>CTfinlist</code> |
| <code>PosResReno</code> | the same as <code>ResReno</code> , but using also the counterterms calculated for the process at stake |

Table 9: Useful variables concerning renormalization. See text for details.

⁴⁰Whenever `MyAux_exp_i.F` files are generated (see note 39), `Makefile-template` automatically includes those files, so that the compilation becomes easier.

⁴¹By default, the Passarino-Veltman functions will be converted as complex; this can be modified by defining a new variable in the FEYNCALC notebook `retil`, and define it as `True` (i.e. `retil=True`). `FctoFT` admits a second (optional) argument, as we now explain. Suppose that `exp` is a complicated expression, but such that it can be written in a simple way using three form factors, `F1`, `F2` and `F3`; that is, `exp = f(F1,F2,F3)`, where `f` is a simple function, while `F1`, `F2` and `F3` correspond to complicated expressions. In cases like this, it is convenient to write `FctoFT` with two arguments: the first one is the expression `exp`, but written in terms of auxiliary variables `F1aux`, `F2aux` and `F3aux` instead of the complicated expressions `F1`, `F2` and `F3` (that is, `exp = f(F1aux,F2aux,F3aux)`); the second argument is a list of the replacements between the auxiliary variables and their corresponding form factor (in this example, `{F1aux -> F1, F2aux -> F2, F3aux -> F3}`).

⁴²It is a matter of course that the calculation of the term with γ_5 in dimension 4 can only be an issue when the integral multiplying it is divergent. This is simply because one does not need to regularize an integral that is not divergent. In particular, there is no need to use dimensional regularization for a finite integral, so that all the calculations can be made in dimension 4. Note also that FEYNCALC already includes different schemes to handle γ_5 , and is expected to improve the treatment of γ_5 in dimensional regularization in future versions.

in the SM, for the process $h \rightarrow Z\gamma$, `CTprocess` is `CTHZA`). Second, `PosResReno` should always be zero, since in the $\overline{\text{MS}}$ scheme the divergent parts are exactly absorbed by the counterterms.

Some final comments on the FEYN CALC notebook. First, the indices of the particles are described in Table 10, and momentum conservation can be applied through the replacement rule `MomCons`.⁴³ Second, the

| | order in <code>Control.m</code> | QGRAF index | Lorentz index in FEYN CALC | Lorentz index in \LaTeX | Color index in \LaTeX | Momentum index in FEYN CALC | Momentum index in \LaTeX |
|-----------------------|------------------------------------|----------------|----------------------------------|---------------------------------|-------------------------------|-----------------------------------|----------------------------------|
| Incoming particles | 1 | -1 | -J1 | μ | a | p1 | p_1 |
| | 2 | -3 | -J3 | ρ | c | p2 | p_2 |
| Outgoing particles | 1 | -2 | -J2 | ν | b | q1 | q_1 |
| | 2 | -4 | -J4 | σ | d | q2 | q_2 |

Table 10: Particle indices for the FEYN CALC notebook.

`Helper.m` file contains, among other definitions, both the `factor` (in case it was defined in `Control.m`) as well as the FEYN CALC identities (in case they were defined as `M$FCeqs` in the FEYN MASTER model). Third, even if `Comp` is set to `False` in `Control.m`, setting `SumLogic` and `MoCoLogic` to `True` has consequences for the FEYN CALC notebook: when the notebook is run with `compNwrite` set to `True`, the total expressions will be calculated and momentum conservation will be applied, respectively. Finally, the replacement rule `FCsimp` contains the simplifications for FEYN CALC (in case they were defined as `M$FCsimp` in the FEYN MASTER model) and is applied in the calculation of `res`, `resD`, `restot` and `resDtot`.

5 Examples

5.1 Creation and complete automatic renormalization of a toy model

Here we exemplify how to create a model, on the one hand, and how to completely renormalize it, on the other. The model will be very simple: QED with an extra fermion. We first show how to create such a toy model.

Probably the simplest way to create any model whatsoever is to copy and modify an already existing model. Given the similarity between our toy model and QED, we go to the directory with models for FEYN MASTER and duplicate the directory `QED`, after which we name the duplicate `QED2`. We get inside `QED2` and change the name of the model file from `QED.fr` to `QED2.fr`. We then open `QED2.fr` and edit the model in three steps: first, we modify the parameter list to:

```
(**** Parameter list ****)
M$Parameters = {
  m1 == {TeXName -> "m_1", Renormalization -> {m1 -> m1 + dm1}},
  m2 == {TeXName -> "m_2", Renormalization -> {m2 -> m2 + dm2}},
  ee == {TeXName -> "e", Renormalization -> {ee -> ee + de ee}},
  xiA == {TeXName -> "\\xi_A"},

```

⁴³Whenever there is one incoming particle and two outgoing particles, `MomCons` replaces p1 by the remaining momenta; in all the other cases, `MomCons` replaces q1 by the remaining momenta.


```

de == {Counterterm -> True, TeXName -> "\\delta e"},
dZ3 == {Counterterm -> True, TeXName -> "\\delta Z_3"},
dZ1L == {Counterterm -> True, TeXName -> "\\delta Z_1^L"},
dZ1R == {Counterterm -> True, TeXName -> "\\delta Z_1^R"},
dZ2L == {Counterterm -> True, TeXName -> "\\delta Z_2^L"},
dZ2R == {Counterterm -> True, TeXName -> "\\delta Z_2^R"},
dm1 == {Counterterm -> True, TeXName -> "\\delta m_1"},
dm2 == {Counterterm -> True, TeXName -> "\\delta m_2"};

```

Then, in the particle classes list, we slightly modify what we had, and we add a second fermion:⁴⁴

```

(***** Particle classes list *****)
M$ClassesDescription = {
  W[1] == {
    ClassName -> psi1L,
    SelfConjugate -> False,
    QuantumNumbers -> {Q-> Q},
    Renormalization -> {psi1L -> psi1L + 1/2 dZ1L psi1L},
    Chirality -> Left},
  W[2] == {
    ClassName -> chi1R,
    SelfConjugate -> False,
    QuantumNumbers -> {Q-> Q},
    Renormalization -> {chi1R -> chi1R + 1/2 dZ1R chi1R},
    Chirality -> Right},
  W[3] == {
    ClassName -> psi2L,
    SelfConjugate -> False,
    QuantumNumbers -> {Q-> Q},
    Renormalization -> {psi2L -> psi2L + 1/2 dZ2L psi2L},
    Chirality -> Left},
  W[4] == {
    ClassName -> chi2R,
    SelfConjugate -> False,
    QuantumNumbers -> {Q-> Q},
    Renormalization -> {chi2R -> chi2R + 1/2 dZ2R chi2R},
    Chirality -> Right},
  F[1] == {
    ClassName -> f1,
    TeXName -> "f_1",
    TeXAntiName -> "\\bar{f}_1"},

```

⁴⁴The fermions are defined both in terms of Weyl spinors (the W variables) and Dirac spinors (the F variables). It is certainly true that, in models with no parity violation (like the present one), there is no need to define the fermions in terms of Weyl spinors. Nevertheless, we consider them for illustrative purposes.

```

SelfConjugate -> False,
QuantumNumbers -> {Q-> Q},
Mass -> m1,
WeylComponents -> {psi1L, chi1R}},
  F[2] == {
ClassName -> f2,
TeXName -> "f_2",
TeXAntiName -> "\\bar{f_2}",
SelfConjugate -> False,
QuantumNumbers -> {Q-> Q},
Mass -> m2,
WeylComponents -> {psi2L, chi2R}},
  V[1] == {
ClassName -> A,
TeXName -> "\\gamma",
Renormalization -> {A[mu_] -> A[mu] + 1/2 dZ3 A[mu]},
Mass -> 0,
SelfConjugate -> True}};

```

We modify the Lagrangean to include a second fermion:

```

LGauge := -1/4 FS[A, \[Mu], \[Nu]] FS[A, \[Mu], \[Nu]]
LFermions := I psi1Lbar.sibar[mu].del[psi1L, mu] + I chi1Rbar.si[mu].del[chi1R, mu] \
- m1 (psi1Lbar.chi1R + chi1Rbar.psi1L) \
+ ee psi1Lbar.sibar[mu].psi1L A[mu] + ee chi1Rbar.si[mu].chi1R A[mu] \
+ I psi2Lbar.sibar[mu].del[psi2L, mu] + I chi2Rbar.si[mu].del[chi2R, mu] \
- m2 (psi2Lbar.chi2R + chi2Rbar.psi2L) \
+ ee psi2Lbar.sibar[mu].psi2L A[mu] + ee chi2Rbar.si[mu].chi2R A[mu]
LGF := -1/2/xiA del[A[mu], mu] del[A[nu], nu]

```

This completes the model. Now, we want to proceed to its complete automatic renormalization — that is, to the determination of the analytical expressions for the complete set of counterterms (in the $\overline{\text{MS}}$ scheme). To do so, we open `Control.m`; we start by setting `model: QED2`. Then, we must choose a sequence of processes such that all the counterterms are computed. To do so, note that the total set of counterterms is:

$$\delta Z_3, \quad \delta Z_1^L, \delta Z_1^R, \delta m_1, \quad \delta Z_2^L, \delta Z_2^R, \delta m_2, \quad \delta e. \quad (1)$$

However, from the renormalization of QED, we know that the first one, δZ_3 , can be determined by the vacuum polarization of the photon; the following three, $\delta Z_1^L, \delta Z_1^R, \delta m_1$, can be determined by the self-energy of f_1 ; by the same token, $\delta Z_2^L, \delta Z_2^R, \delta m_2$ can be determined by the self-energy of f_2 ; finally, δe can be determined by one of the vertices (either $f_1 \bar{f}_1 \gamma$ or $f_2 \bar{f}_2 \gamma$) at 1 loop. Therefore, we write:

```

inparticles: A
outparticles: A
loops: 1

```

```

inparticles: f1
outparticles: f1
loops: 1
options: onepi

```

```

inparticles: f2
outparticles: f2
loops: 1
options: onepi

```

```

inparticles: A
outparticles: f1,f1bar
loops: 1
options: onepi

```

Finally, concerning the logical variables of `Control.m`, we set them all to `True`, except `LoSpinors`, which we set to `False`. This being done, everything is set. We then go to the `FeynMaster` folder and run batch the file `RUN-FEYNMASTER`. In total, 10 PDF files are automatically and subsequentially generated and opened: one for the tree-level Feynman rules, another one for the counterterms Feynman rules, and two files per process — one with the Feynman diagrams, another with the respective expressions. In the last file for the expressions, we read “*This completes the renormalization of the model*”, and the list of the full set of counterterms is presented.

5.2 $h \rightarrow \gamma\gamma$ in the Standard Model

In this example, we use the $h \rightarrow \gamma\gamma$ in the SM as an illustration of several features of `FEYNMASTER`. We use the SM model file that comes with `FEYNMASTER`. As for `Control.m`, we set it as:⁴⁵

```

model: SM

inparticles: H
outparticles: A,A
loops: 1

FRinterLogic: T
RenoLogic: T
Draw: T
Comp: F
FinLogic: F
DivLogic: F

```

⁴⁵ We are setting `FRinterLogic` to `True`, which only needs to be done in case it was not yet done before. Actually, generating all Feynman rules for both the tree-level interactions and the counterterms in the SM may take around 5 minutes. Therefore, if we have already performed that operation, we can save time by setting `FRinterLogic` and `RenoLogic` to `False`.

```

SumLogic: T
MoCoLogic: F
LoSpinors: F

```

We then run the batch file `RUN-FeynMaster`. In total, 3 PDF files will automatically be generated and opened: one for the tree-level Feynman rules, another one for the counterterms Feynman rules and a third one for the Feynman diagrams. We go to the directory `SM/Processes/1-HAA` (meanwhile generated inside the directory for the FEYNMASTER output) and open the FEYNCALC notebook `Notebook.nb`. We then run the `Notebook.nb`, after which we are ready to test some relevant features.

5.2.1 Notebook access to Feynman rules

First, we want to gain some intuition on how the notebook has access to the SM Feynman rules and to the amplitudes of the $h \rightarrow \gamma\gamma$ decay. We write

```
In[14] := amp1
```

which should yield the expression:

$$\frac{2 e^3 m_W g^{-J_2-J_4}}{s_w (k_1^2 - m_W^2) ((p_1 - k_1)^2 - m_W^2)} - \frac{2 D e^3 m_W g^{-J_2-J_4}}{s_w (k_1^2 - m_W^2) ((p_1 - k_1)^2 - m_W^2)}. \quad (2)$$

This is the amplitude for the first Feynman diagram, where D represents the dimension. Now, where does the notebook get this information from? To answer the question, we open `Amplitudes.m` inside `1-HAA`. If we check the first line, we realize that `amp1` is essentially a product of Feynman rules such as `propWP[...]` and `vrtxAAWPWPbar[...]`.⁴⁶ These rules are defined in the auxiliary files inside the directory `SM/FeynmanRules` (inside the directory for the FEYNMASTER output). Although they have been automatically generated, they can always be edited for particular purposes.

5.2.2 Finiteness and gauge invariance

Next, we use some of the features described in section 4.4.2 to test two important properties of $h \rightarrow \gamma\gamma$: finiteness and gauge invariance. We start with the former; by writing

```
In[15] := resD
```

we obtain the list with all the expressions for the divergents parts. It is a non-trivial list: although some of its elements are zero, most of them are not. However, when we sum the whole list, we find:

```
In[16] := resDtot
```

```
Out[16] = 0
```

so that the process as a whole is finite, as expected for this decay mode.

Let us now check gauge invariance. First of all, note that the total amplitude M for $h \rightarrow \gamma\gamma$ can be written as

$$M = \epsilon_1^\nu \epsilon_2^\sigma M_{\nu\sigma}, \quad (3)$$

where we are just factoring out the polarization vectors ϵ_1^ν and ϵ_2^σ of the two photons. Then, it is easy to show that gauge invariance forces $M^{\nu\sigma}$ to have the form

$$M^{\nu\sigma} = (g^{\nu\sigma} q_1 \cdot q_2 - q_1^\sigma q_2^\nu) F, \quad (4)$$

⁴⁶Amplitudes 13 to 22 have an extra factor 3, which corresponds to the color number. This factor is automatically added by FEYNMASTER for diagrams with closed loops of quarks.

where q_1 and q_2 are the 4-momenta of the two photons, and F is a scalar function of the momenta and the masses. In other words, it is a consequence of gauge invariance that, in the total process, the coefficient of $g^{\nu\sigma} q_1 \cdot q_2$ must be exactly opposite to that of $q_1^\sigma q_2^\nu$. To test this, we define some replacement rules:

```
(* momentum conservation in scalar products and four-vectors *)
In[17]:= dist = {SP[p1, x_] -> SP[q1, x] + SP[q2, x], FV[p1, x_] -> FV[q1, x] + FV[q2, x]};

(* external particles on-shell *)
In[18]:= onshell = {SP[q1, q1] -> 0, SP[q2, q2] -> 0, SP[p1, p1] -> mH^2};

(* kinematics *)
In[19]:= kin = {SP[q1, q2] -> MH^2/2, SP[p1, q1] -> MH^2/2, SP[p1, q2] -> MH^2/2};

(* transversality of the external photons polarizations *)
In[20]:= transv = {FV[q1, -J2] -> 0, FV[q2, -J4] -> 0};
```

which we use to define a new `res` list:

```
In[21]:= resnew = (res /. dist /. onshell /. kin /. transv) // Simplify;
```

Finally, we write the coefficients of $g^{\nu\sigma} q_1 \cdot q_2$ and $q_1^\sigma q_2^\nu$ as

```
In[22]:= resnewT = (Coefficient[resnew, MT[-J2, -J4]] // MyPaVeReduce)
           /(MH^2/2) // Simplify // FCE;
In[23]:= resnewL = (Coefficient[resnew, FV[q1, -J4]*FV[q2, -J2]] // MyPaVeReduce)
           // Simplify // FCE;
```

respectively, to conclude that

```
In[24]:= Total[resnewT] + Total[resnewL] // Simplify
Out[24]:= 0
```

in accordance with gauge invariance. For what follows, it is convenient to save the expressions for the total transverse and longitudinal part. We write

```
In[25]:= FT = Total[resnewT] // Simplify;
In[26]:= FL = Total[resnewL] // Simplify;
```

5.2.3 MyTeXForm

We now illustrate how to use `MyTeXForm` inside the `FEYNCalc` notebook. Suppose we want to write the sum of final results for the diagrams with quartic vertices (diagrams 1 to 6) in a $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ document. We define the variable `toprint1` as

```
In[27]:= toprint1 = Sum[res[[i]], {i, 1, 6}] // Simplify
```

after which we write

```
In[28]:= toprint1 // MyTeXForm
```

If we now copy the outcome as plain text and paste it in a \LaTeX document like the present one, we get:

$$\begin{aligned}
& - \left(e^3 \left((m_h^2 + 6 m_W^2) B_0(p_1^2, m_W^2, m_W^2) \right. \right. \\
& \quad \left. \left. + m_W^2 \left(-4 + B_0(q_1^2, m_W^2, m_W^2) + B_0(q_2^2, m_W^2, m_W^2) \right) \right) g^{\nu\sigma} \right) / (16 m_W \pi^2 s_w)
\end{aligned} \tag{5}$$

Note that we did not need to break the line manually in the \LaTeX equation. This is because we are using the `breqn` package, which automatically breaks lines in equations.⁴⁷

5.2.4 Fortran interface

We mentioned in the Introduction that FEYNMASTER includes a numerical interface with `FORTRAN`. We now show how it works in the context of $h \rightarrow \gamma\gamma$. Suppose we want to plot the decay width as a function of the Higgs mass; we could start by computing the total $h \rightarrow \gamma\gamma$ decay width:

```
In[27]:= X0 = restot // DecayWidth
```

However, although this works, it takes a long time and produces large expressions. It is simpler to exploit the generic form of eq. 4 and use a form factor; that is,

```
In[27]:= X0 = FTaux (MT[-J2, -J4] MH^2/2 - FV[q1, -J4] FV[q2, -J2]) // DecayWidth
```

where `FTaux` is an auxiliary variable that, in the end, must be replaced by the absolute value of the total transverse part `FT`, defined above. Then, we write

```
In[28]:= (X0 /. FTaux -> Abs[FT] // Simplify) // FCtoFT
```

where we performed the referred replacement.⁴⁸ As explained in section 4.4.2, the command `FCtoFT` generates five files: `MainFT.F`, `MyFunctionFT.F`, `MyVariables.h`, `MyParameters.h` and `Makefile-template`. We open `MainFT.F` and, immediately after the comments *Write now the rest of the program*, we write⁴⁹

```
xMH=38d0
do i=1,162
xMH=xMH+1d0
write(50,98)xMH,MyFunction(..., xMH, ...)
enddo
```

where 50 and 98 represent the output file and the impression format, respectively. We are varying the Higgs mass from 38 GeV to 200 GeV in steps of 1 GeV. The result is presented in Fig. 1.

5.2.5 Edition of Feynman diagrams

Finally, we briefly explain how to edit the Feynman diagrams. Recall that they were written in a \LaTeX file inside `SM/1-HAA/TeXs-drawing` folder. We open the file `diagrams.tex` and consider the first diagram; the original code produces the original diagram:

⁴⁷For documentation, cf. <https://www.ctan.org/pkg/breqn>. Recall that the line breaking does not work when the point where the line is to be broken is involved in three or more parentheses.

⁴⁸Alternatively, following the instructions in note 41, we could also have written `FCtoFT[X0, {FTaux -> Abs[FT]}]`.

⁴⁹The parameters loaded from the file `MyParameter.h` cannot be changed inside the `FORTRAN` program (`MainFT.F`). Hence, since we define the parameter `MH` as the Higgs boson mass, we name `xMH` the variable we are using to make the plot; in doing so, we must be careful to replace `MH` for `xMH` in the arguments of `MyFunction` inside the loop.

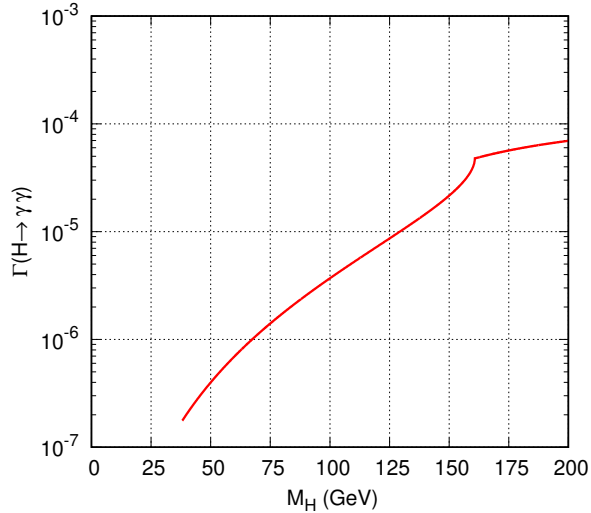
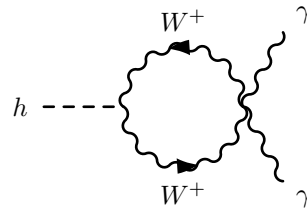


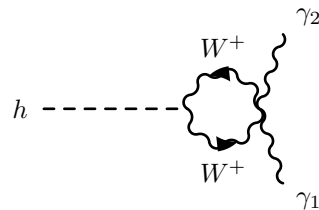
Figure 1: Width of the process $h \rightarrow \gamma\gamma$ as a function of the Higgs boson mass.

```
(...)
\fmflabel{\gamma}{...}
\fmflabel{\gamma}{...}
\fmf{dashes,tension=3}{...}
\fmf{photon,tension=3}{...}
\fmf{photon,tension=3}{...}
\fmf{photon,label=W^+,right=1}{...}
\fmf{phantom_arrow,tension=0,right=1}{...}
\fmf{photon,label=W^+,right=1}{...}
\fmf{phantom_arrow,tension=0,right=1}{...}
(...)
```



However, we can modify the code in order to change the aspect of the diagram. In particular, we can change the labels, the tensions and the curvatures.⁵⁰ For example:

```
(...)
\fmflabel{\gamma_1}{...}
\fmflabel{\gamma_2}{...}
\fmf{dashes,tension=1}{...}
\fmf{photon,tension=3}{...}
\fmf{photon,tension=3}{...}
\fmf{photon,label=W^-,right=1}{...}
\fmf{phantom_arrow,tension=0,right=1}{...}
\fmf{photon,label=W^-,right=1}{...}
\fmf{phantom_arrow,tension=0,right=1}{...}
(...)
```



⁵⁰The tensions represent the strength of the lines: the larger the tension, the tighter the line will be. The default tension is 1. The curvature is represented by the variable `right`. Note that tensions, labels and curvatures are just a few examples of variables that can be changed to generate a different diagram. For more informations, please consult the `feynmf` manual.

5.3 QED Ward identity

In the previous example, we showed how to use FEYNMASTER to manipulate the results of a single process. Here, we illustrate how it can also be used to combine information of several processes. For that purpose, we consider a simple task: prove the QED Ward identity.

It is easy to show that the Ward identity at one-loop order in QED can be written as:

$$p_1^\nu \Gamma_\nu(p_1, p_2, p_3) = e \left(\Sigma(p_2) - \Sigma(p_3) \right), \quad (6)$$

where

$$\Gamma_\nu(p_1, p_2, p_3) = \text{[Feynman diagram: vertex with incoming photon } p_1 \text{ and outgoing fermions } p_2, p_3 \text{]} , \quad \Sigma(p_i) = \text{[Feynman diagram: fermion self-energy loop with momentum } p_i \text{]} , \quad (7)$$

and where the momenta p_1 and p_3 are incoming, while p_2 is outgoing. In order to prove eq. 6 with FEYNMASTER, we need to consider the two processes depicted in eq. 7: the QED vertex and the fermion self-energy. Hence, we open and edit `Control.m` according to:

```

model: QED

inparticles: A
outparticles: f,fbar
loops: 1
options: onepi

inparticles: f
outparticles: f
loops: 1
options: onepi

FRinterLogic: T
RenoLogic: F
Draw: F
Comp: T
FinLogic: F
DivLogic: F
SumLogic: T
MoCoLogic: F
LoSpinors: F

```

We then run FEYNMASTER. After this, we go to the directory `QED/Processes/1-Affbar` (meanwhile generated inside the directory for the FEYNMASTER output), we copy the notebook lying there to a

different directory and we rename it `Notebook-Global.nb`. This is going to be the notebook where we shall combine the information of both processes. We open it, and delete most of the lines there: in a first phase, we only want to load the general files. So it must look like this:

```
In[1]:= << FeynCalc`
In[2]:= dirNuc = "...";
In[3]:= dirFey = "...";
In[4]:= dirCT = "...";
In[5]:= Get["Feynman-Rules-Main.m", Path -> {dirFey}]
In[6]:= Get["FunctionsFM.m", Path -> {dirNuc}]
In[7]:= compNwrite = False;
```

Now, we want to load the first process. To do so, we write:

```
In[8]:= dirHome = "...";
In[9]:= SetDirectory[dirHome];
In[10]:= << Helper.m;
In[11]:= Get["Definitions.m", Path -> {dirNuc}];
In[12]:= Get["Finals.m", Path -> {dirNuc}];
```

where `dirHome` should be set to the directory corresponding to `1-Affbar`. Next, we define new variables: `X0` as Γ of eq. 6, with the above-mentioned momentum definitions, and `X1` as the whole left-hand side of eq. 6.

```
In[13]:= X0 = res[[1]] /. {p1 -> p2 - p3, q1 -> p2, q2 -> -p3};
In[14]:= X1 = Contract[X0 FV[p2 - p3, -J1]];
```

We now load the second process:

```
In[15]:= dirHome = "...";
In[16]:= SetDirectory[dirHome];
In[17]:= << Helper.m ;
In[18]:= Get["Definitions.m", Path -> {dirNuc}];
In[19]:= Get["Finals.m", Path -> {dirNuc}];
```

where `dirHome` should now be set to the directory corresponding to `2-ff`. From this, and recalling that the default momentum of a self-energy is `p1` (cf. Table 10), we can obtain the right-hand side of eq. 6 by writing:

```
In[20]:= Y0a = res[[1]] /. p1 -> p2;
In[21]:= Y0b = res[[1]] /. p1 -> p3;
In[22]:= Y1 = (e (Y0a - Y0b) // DiracSimplify) // Simplify;
```

Finally, we prove the Ward identity by showing that both sides of eq. 6 are equal:

```
In[23]:= WI = (Y1 - X1) // Simplify;
In[24]:= CheckWI = MyPaVeReduce[WI] // Calc
```

which yields 0, thus completing the proof.

6 Quick first usage

For a quick first usage of FEYNMASTER, the user should follow this sequence of steps:

- 1) Make sure you have installed `PYTHON`, `MATHEMATICA` and `LATEX`, on the one hand, and `FEYNRULES`, `QGRAF` and `FEYNCALC`, on the other;
- 2) Download `FEYNMASTER` in <https://porthos.tecnico.ulisboa.pt/FeynMaster/>;
- 3) Extract the downloaded file and place the resulting folder in a directory at will;
- 4) Edit the file `RUN-FeynMaster` as explained in section 2;
- 5) Run `RUN-FeynMaster`.

This should generate and open 4 PDF files relative to QED: the Feynman rules for the tree-level interactions, the Feynman rules for the counterterms interactions, the Feynman diagram for the one-loop vacuum polarization, and a document containing not only the expressions for the vacuum polarization, but also the expression for the associated counterterm in $\overline{\text{MS}}$.

Acknowledgements

Both authors are very grateful to Ant3nio P. Lacerda, who kicked off the entire program. We also thank Vladyslav Shtabovenko, Paulo Nogueira and Augusto Barroso for useful discussions concerning `FEYN-CALC`, `QGRAF` and renormalization, respectively; Maximilian L3schner for bringing the `feynmf` package to our attention; Miguel P. Bento and Patrick Blackstone for testing the program; Darius Jur3iukonis for a careful reading of the manuscript; Jo3o P. Silva for the suggestion of the name ‘`FEYNMASTER`’, as well as for a careful reading of the manuscript. D.F. is also grateful to Isabel Fonseca for many useful suggestions concerning `PYTHON` and to Sofia Gomes for a suggestion regarding the printing of the Feynman rules. Both authors are supported by projects CFTP-FCT Unit 777 (UID/FIS/00777/2013 and UID/FIS/00777/2019), and PTDC/FIS-PAR/ 29436/2017, which are partially funded through POCTI (FEDER), COMPETE, QREN and EU. D.F. is also supported by the Portuguese *Funda33o para a Ci4ncia e Tecnologia* under the project SFRH/BD/135698/2018.

References

- [1] D. Fontes and J. C. Rom3o, *Comput. Phys. Commun.* **256** (2020), 107311 doi:10.1016/j.cpc.2020.107311 [arXiv:1909.05876 [hep-ph]].
- [2] N. D. Christensen and C. Duhr, “FeynRules - Feynman rules made easy,” *Comput. Phys. Commun.* **180** (2009) 1614 doi:10.1016/j.cpc.2009.02.018 [arXiv:0806.4194 [hep-ph]].
- [3] 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** (2014) 2250 doi:10.1016/j.cpc.2014.04.012 [arXiv:1310.1921 [hep-ph]].
- [4] P. Nogueira, “Automatic Feynman graph generation,” *J. Comput. Phys.* **105** (1993) 279. doi:10.1006/jcph.1993.1074
- [5] R. Mertig, M. Bohm and A. Denner, “FEYN CALC: Computer algebraic calculation of Feynman amplitudes,” *Comput. Phys. Commun.* **64** (1991) 345. doi:10.1016/0010-4655(91)90130-D
- [6] V. Shtabovenko, R. Mertig and F. Orellana, “New Developments in FEYN CALC 9.0,” *Comput. Phys. Commun.* **207** (2016) 432 doi:10.1016/j.cpc.2016.06.008 [arXiv:1601.01167 [hep-ph]].

- [7] V. Shtabovenko, R. Mertig and F. Orellana, *Comput. Phys. Commun.* **256** (2020), 107478 doi:10.1016/j.cpc.2020.107478 [arXiv:2001.04407 [hep-ph]].
- [8] G. Belanger, F. Boudjema, J. Fujimoto, T. Ishikawa, T. Kaneko, K. Kato and Y. Shimizu, “Automatic calculations in high energy physics and Grace at one-loop,” *Phys. Rept.* **430** (2006) 117 doi:10.1016/j.physrep.2006.02.001 [hep-ph/0308080].
- [9] A. Pukhov *et al.*, “CompHEP: A Package for evaluation of Feynman diagrams and integration over multiparticle phase space,” hep-ph/9908288.
- [10] J. Kublbeck, M. Bohm and A. Denner, “Feyn Arts: Computer Algebraic Generation of Feynman Graphs and Amplitudes,” *Comput. Phys. Commun.* **60** (1990) 165. doi:10.1016/0010-4655(90)90001-H
- [11] T. Hahn, “Generating Feynman diagrams and amplitudes with FeynArts 3,” *Comput. Phys. Commun.* **140** (2001) 418 doi:10.1016/S0010-4655(01)00290-9 [hep-ph/0012260].
- [12] T. Hahn and M. Perez-Victoria, “Automatized one loop calculations in four-dimensions and D-dimensions,” *Comput. Phys. Commun.* **118** (1999) 153 doi:10.1016/S0010-4655(98)00173-8 [hep-ph/9807565].
- [13] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, G. Ossola, T. Reiter and F. Tramontano, “Automated One-Loop Calculations with GoSam,” *Eur. Phys. J. C* **72** (2012) 1889 doi:10.1140/epjc/s10052-012-1889-1 [arXiv:1111.2034 [hep-ph]].
- [14] G. Cullen *et al.*, “GOSAM-2.0: a tool for automated one-loop calculations within the Standard Model and beyond,” *Eur. Phys. J. C* **74** (2014) no.8, 3001 doi:10.1140/epjc/s10052-014-3001-5 [arXiv:1404.7096 [hep-ph]].
- [15] A. Lorca and T. Riemann, *Comput. Phys. Commun.* **174** (2006) 71 doi:10.1016/j.cpc.2005.09.003 [hep-ph/0412047].
- [16] C. Degrande, “Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians: a proof-of-principle,” *Comput. Phys. Commun.* **197** (2015) 239 doi:10.1016/j.cpc.2015.08.015 [arXiv:1406.3030 [hep-ph]].
- [17] A. V. Semenov, “LanHEP: A Package for automatic generation of Feynman rules in gauge models,” hep-ph/9608488.
- [18] A. Semenov, “LanHEP: A package for automatic generation of Feynman rules from the Lagrangian,” *Comput. Phys. Commun.* **115** (1998) 124. doi:10.1016/S0010-4655(98)00143-X
- [19] J. X. Wang, “Progress in FDC project,” *Nucl. Instrum. Meth. A* **534** (2004) 241 doi:10.1016/j.nima.2004.07.094 [hep-ph/0407058].
- [20] M. Tentyukov and J. Fleischer, “A Feynman diagram analyzer DIANA,” *Comput. Phys. Commun.* **132** (2000) 124 doi:10.1016/S0010-4655(00)00147-8 [hep-ph/9904258].
- [21] J. Alwall *et al.*, *JHEP* **1407** (2014) 079 doi:10.1007/JHEP07(2014)079 [arXiv:1405.0301 [hep-ph]].
- [22] J. C. Romão and J. P. Silva, “A resource for signs and Feynman diagrams of the Standard Model,” *Int. J. Mod. Phys. A* **27** (2012) 1230025 doi:10.1142/S0217751X12300256 [arXiv:1209.6213 [hep-ph]].
- [23] A. Denner, “Techniques for calculation of electroweak radiative corrections at the one loop level and results for W physics at LEP-200,” *Fortsch. Phys.* **41** (1993) 307 doi:10.1002/prop.2190410402 [arXiv:0709.1075 [hep-ph]].
- [24] M. Tanabashi *et al.* [Particle Data Group], “Review of Particle Physics,” *Phys. Rev. D* **98** (2018) no.3, 030001. doi:10.1103/PhysRevD.98.030001
- [25] T. Ohl, “Drawing Feynman diagrams with Latex and Metafont,” *Comput. Phys. Commun.* **90** (1995) 340 doi:10.1016/0010-4655(95)90137-S [hep-ph/9505351].

- [26] J. C. Romão, *Advanced Quantum Field Theory* (IST, 2019), Available online at <http://porthos.tecnico.ulisboa.pt/Public/textos/tca.pdf>.
- [27] A. Denner and S. Dittmaier, *Nucl. Phys. B* **734** (2006) 62 doi:10.1016/j.nuclphysb.2005.11.007 [hep-ph/0509141].
- [28] D. A. Akyeampong and R. Delbourgo, *Nuovo Cim. A* **17** (1973) 578. doi:10.1007/BF02786835
- [29] M. S. Chanowitz, M. Furman and I. Hinchliffe, *Nucl. Phys. B* **159** (1979) 225. doi:10.1016/0550-3213(79)90333-X
- [30] A. Barroso, M. A. Doncheski, H. Grotch, J. G. Korner and K. Schilcher, *Phys. Lett. B* **261** (1991) 123. doi:10.1016/0370-2693(91)91336-T
- [31] D. Kreimer, hep-ph/9401354.
- [32] F. Jegerlehner, *Eur. Phys. J. C* **18** (2001) 673 doi:10.1007/s100520100573 [hep-th/0005255].
- [33] W. Greiner, S. Schramm and E. Stein, Berlin, Germany: Springer (2002), Appendix B
- [34] N. Zerf, *Phys. Rev. D* **101** (2020) no.3, 036002 doi:10.1103/PhysRevD.101.036002 [arXiv:1911.06345 [hep-ph]].