DESY 00–124
TTP00–14
hep-ph/0009029
July 2000

MATAD: a program package for the computation of MAssive TADpoles

Matthias Steinhauser

II. Institut für Theoretische Physik,
Universität Hamburg, D-22761 Hamburg, Germany

and

Institut für Theoretische Teilchenphysik,
Universität Karlsruhe, D-76128 Karlsruhe, Germany

Abstract

In the recent years there has been an enormous development in the evaluation of higher order quantum corrections. An essential ingredient in the practical calculations is provided by vacuum diagrams, i.e. integrals without external momenta. In this paper a program package is described which can deal with one-, two- and three-loop vacuum integrals with one non-zero mass parameter. The principle structure is introduced and the main parts of the package are described in detail. Explicit examples demonstrate the fields of application.

PROGRAM SUMMARY

Title of program: `MATAD`

Available from:
`http://www-ttp.physik.uni-karlsruhe.de/Progdata/MATAD/1/`

Computer for which the program is designed and others on which it is operable: Any work-station or PC where `FORM` is running.

Operating system or monitor under which the program has been tested: UNIX, `FORM` 2.3

No. of bytes in distributed program including test data etc.: 706000

Distribution format: ASCII

Keywords: three-loop computations, vacuum integrals, computer algebra, automation of computations

Nature of physical problem: Multi-loop integrals are needed for the evaluation of quantum corrections. An important class of loop diagrams is covered by so-called vacuum integrals which have no external momentum. `MATAD` can analytically compute those one-, two- and three-loop vacuum integrals where one mass scale is present.

Method of solution: The method of integration-by-parts is used in order to obtain recurrence relations which reduce complicated integrals to a small set of so-called master integrals. They have to be evaluated once and for all. In addition a user interface is provided which makes it easy to put in complicated diagrams in a rather compact way.

Restrictions on the complexity of the problem: The restrictions on the complexity are given by the hardware limitations of the computer and the limits on the size of the storage files inside `FORM`.

Typical running time: The runtime strongly depends on the complexity of the diagram under consideration. It may vary form a few seconds to the order of a few weeks.

LONG WRITE-UP

1 Introduction

The high experimental precision reached at the electron-positron machines LEP (CERN) and SLC (SLAC) and the hadron collider TEVATRON (FERMILAB) requires from the theoretical side the evaluation of higher order quantum corrections. In the cases where perturbative methods are applied the quantum corrections can be expressed through an expansion in the coupling constant of the underlying theory. The individual terms can in turn be expressed through so-called Feynman diagrams, which are often classified as multi-leg or multi-loop diagrams. Vacuum integrals, i.e. integrals without external momenta, constitute an important sub-class and often serve as building blocks in complex calculations.

In general the momentum integration of the loop integrals is divergent in four space-time dimensions. At present the most practical method to cope with this problem in higher loop orders is based on Dimensional Regularization [1]. There, the four space-time dimensions are replaced by $D = 4 - 2\varepsilon$ dimensions. Then the integrals are solved for a choice of ε that renders them finite. Finally an expansion for $\varepsilon \to 0$ is performed and the divergences manifest themselves as poles in ε.

Important progress in practical computations has been made roughly 20 years ago by establishing an algorithm for the evaluation of propagator-type diagrams up to three loops in the massless case [2]. They are important if there is only one external momentum which sets the mass scale for the problem. The formulae have been implemented on the computer in the `FORM` [3] package `MINCER` [4]. In 1995 for the first time three-loop diagrams in the opposite limit, i.e. zero external momentum but massive lines, were systematically examined [5]. Usually these are denoted as vacuum or tadpole diagrams. In [5] all integrals contributing to the photon propagator have been considered. The main characteristics of this class of diagrams is that the massive line forms a closed loop. These considerations have been extended to the W boson current correlators which led to one of the most prominent applications of three-loop vacuum integrals, namely the ρ parameter at $\mathcal{O}(\alpha\alpha_s^2)$ [6, 7]. The remaining cases have been considered in [8, 9, 10, 11]. Thus it is — at least in principle — possible to treat all problems where exactly one heavy mass is involved.

In this paper we want to present the program package `MATAD` which was designed for the computation of MAssive TADpoles at one-, two- and three-loop order as pictured in Fig. 1. Thereby each line may be massless or carry the mass M. In mathematical form the integrals to be solved by `MATAD` read

$$\int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{1}{(p^2 + M^2)^n} \,,$$

$$\int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{(p_1^2 + M_1^2)^{n_1} (p_2^2 + M_2^2)^{n_2} (p_3^2 + M_3^2)^{n_3}} \,,$$

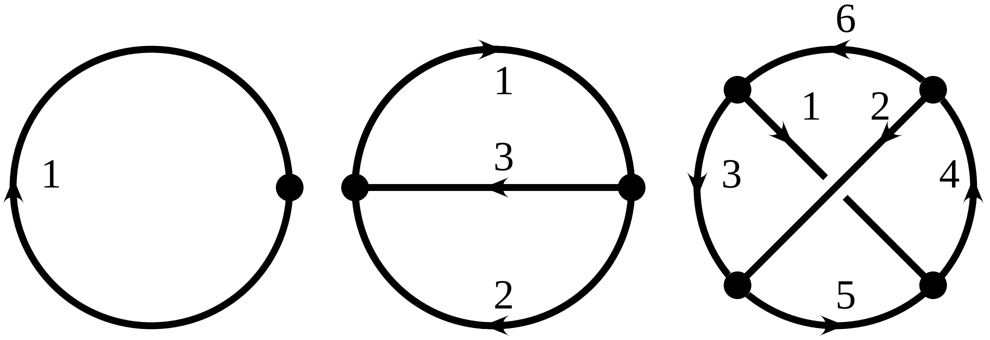

Figure 1: Prototype topologies for one-, two- and three-loop vacuum diagrams. The momentum p_i flows through the line i as indicated by the arrow. Each line may either be massless or carry mass M and may be raised to an arbitrary integer power.

$$\int \frac{\mathrm{d}^D p}{(2\pi)^D}\frac{\mathrm{d}^D k}{(2\pi)^D}\frac{\mathrm{d}^D l}{(2\pi)^D}\frac{1}{(p_1^2+M_1^2)^{n_1}(p_2^2+M_2^2)^{n_2}(p_3^2+M_3^2)^{n_3}} \times \\ \frac{1}{(p_4^2+M_4^2)^{n_4}(p_5^2+M_5^2)^{n_5}(p_6^2+M_6^2)^{n_6}}, \tag{1}$$

with $M_i = 0$ or $M_i = M$. These expressions correspond to the diagrams in Fig. 1 where the momentum p_i flows through the line i as indicated by the arrow. p_i can be expressed as a linear combination of the loop momenta. However, these relations are in our case not of interest. Note that the integrals in Eq. (1) are defined in Euclidean space.

The key idea for the computation of tadpole integrals is based on the integration-by-parts method [2] (see also Appendix A). It can be used for the derivation of recurrence relations which relate vacuum integrals with different denominator structures. The proper use of the recurrence relations allows the reduction of an arbitrary integral to simple ones, which can be solved by successively using one- and two-loop formulae, and a linear combination of a few so-called master integrals. Only for them a hard calculation is necessary. In the case of three-loop tadpole diagrams nine master integrals are needed.

At first sight the applications for vacuum integrals seem to be quite restricted. However, for diagrams involving several mass scales, which follow a certain hierarchy, it is very often advantageous to apply an asymptotic expansion [12] which allows for a systematic expansion in the inverse heavy scale. Then the multi-scale integrals are expressed as products of single scale ones. In [13] and [14] the rules for the so-called large-momentum and hard-mass procedure have been automated and computer programs, `LMP` [13] and `EXP` [14], have been developed. They generate for a given diagram all relevant sub-graphs together with the administrative files which govern the very calculation. `LMP` [13] is written in `PERL` and can be applied to problems where one large momentum is involved. `EXP` [14], written in `C++`, allows for a successive use of the large-momentum and hard-mass procedure and thus can deal with problems involving many scales. Both programs produce output which can be read into MATAD and MINCER. Thus the combination of both massive vacuum integrals and massless propagator-type diagrams is very powerful to attack problems involving several different mass scales. We want to mention that MATAD can be easily linked to a generator for Feynman diagrams. More details — in particular on the automation of the computation of Feynman diagrams — can be found in [15].

The outline of the paper is as follows: In Section 2 the structure of MATAD and the way it works is described. With the help of this section the reader should be able to use MATAD for his own problems. Deeper insight into some selected parts is provided in Section 3. In Section 4 explicit examples are discussed and hints for the convenient usage of MATAD are given. In Appendix A the ideas of the integration-by-parts method are reviewed. Appendix B lists all massive/massless combinations which are implemented into the topology files and in Appendix C the notation of the input and output is described. Furthermore the results for the master integrals are listed and the switches for the input-file are described. Appendix D contains a list of all files of MATAD and, finally, in Appendix E the complete output of one of the considered examples is listed.

2 Structure and mode of operation

As MATAD is completely written in FORM [3] its installation reduces to copying the individual files into the corresponding directories. In the main directory the following files appear:

```
form.set  inc/  matadform  prc/  problems/
```

The directories `inc` and `prc` contain the include-files and procedures, respectively. They are described in more detail in Section 3 and Appendix D. `matadform` is a shell script which calls FORM in such a way that files from sub-directories can be included. It has to be adjusted by the user by simply specifying the corresponding paths. The file `form.set` contains FORM-specific settings which have to be adjusted according to the underlying platform. For details concerning the different switches we refer to the FORM manual [3]. The user-specific files are all contained in the folder `problems`.

There are at least two files which should be provided by the user: `main<prb>` and `<prb>.dia` where `<prb>` stands for the name of the considered problem. The first one contains apart from some parameters essentially the information which diagram should be treated. Some explicit examples are given below. All the information about the diagrams, the projectors to be applied, etc. is contained in the file `<prb>.dia`. It is built up by FORM folds and splits into two parts so that its generic structure looks as follows

```
*--#[ TREAT0:
[...]
*--#] TREAT0:

*--#[ TREAT1:
[...]
```

```
*--#] TREAT1:

*--#[ TREAT2:
[...]
*--#] TREAT2:

*--#[ TREATMAIN:
[...]
*--#] TREATMAIN:

*
* in the following list each diagram is contained in a separate FORM fold
*

*--#[ d1l1:
[... diagram 1 ...]
#define TOPOLOGY "XY"
*--#] d1l1:

*--#[ d1l2:
[... diagram 2 ...]
#define TOPOLOGY "XY"
*--#] d1l2:

[...]
```

The first part consists of the first four folds — the so-called special treat files. They provide the possibility to interact at different stages and thus influence the computation. Whereas `TREAT0`, `TREAT1` and `TREAT2` are read before the recurrence relations are applied the content of `TREATMAIN` is read right before the results are stored to disk. The second part of `<prb>.dia` contains a list of all diagrams to be considered where each diagram is written in a separate fold. The name of these folds is arbitrary.

Once these two files are set up the calculation is simply initiated by calling the program `main<prb>` and the following steps are performed. They are also illustrated in Fig. 2.

1. Read global settings. They are partly contained in `inc/main.gen` and should not be modified. Others can be set by the user in the file `main<prb>`. They are described in Appendix C.4.

2. Read the input data for the diagram specified in `main<prb>` with the help of the variables `PRB`, `FOLDER` and `DIAGRAM`. The generic `FORM` command reads `#include problems/'PRB'/'FOLDER'.dia # 'DIAGRAM'`.

As a next step the file `treat.prc` is called and the following operations are performed.

3. Insert Feynman rules for functions appearing in the input. In a first step the fermions (encoded in the functions `S`, `SS`, . . . , cf. Appendix C.1) are resolved. It is important to do this before any contraction of indices is done. Then the propagators and vertices are treated.

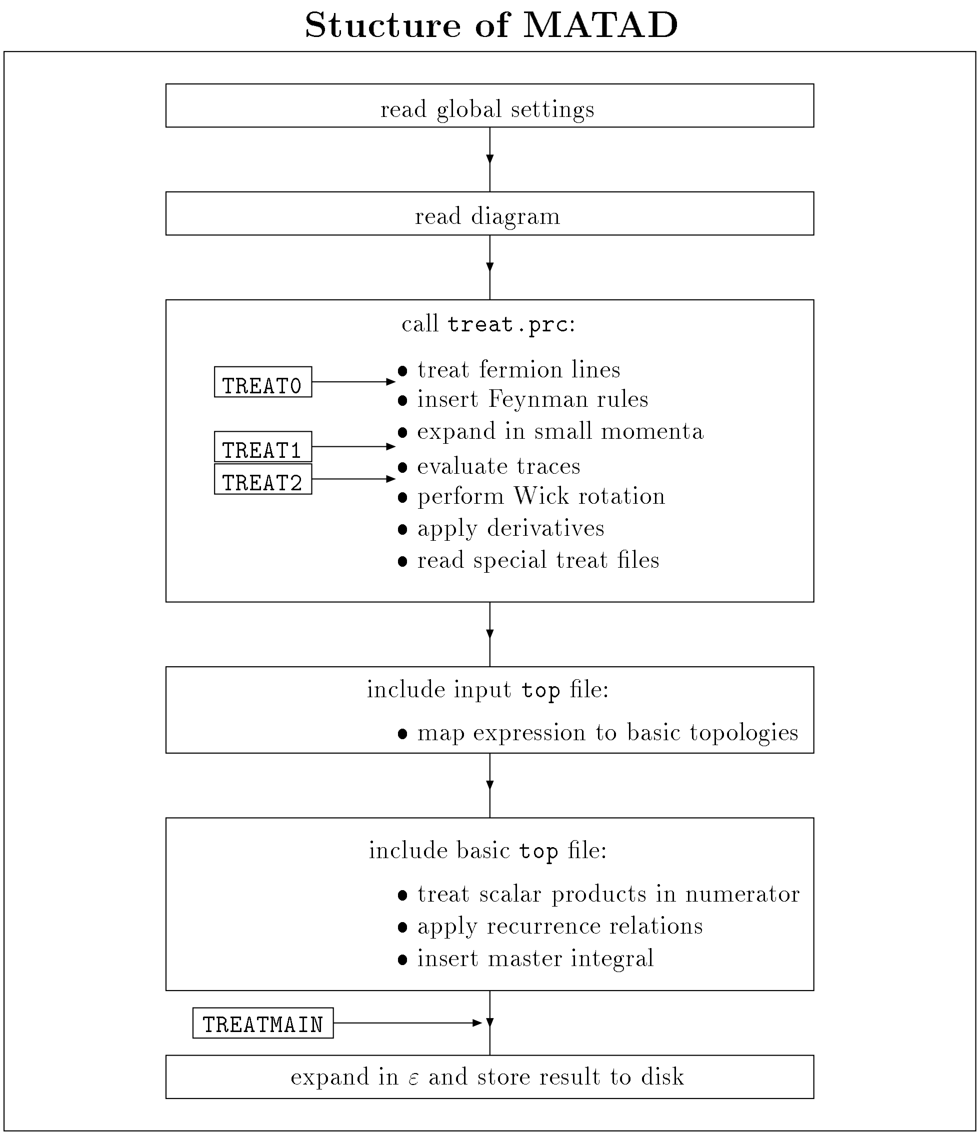

Figure 2: Flowchart illustrating the structure of `MATAD`.

In the current version the QCD Feynman rules are implemented (except the four-gluon vertex; see Appendix C.1). It is, however, straightforward to implement new vertices in the user-specific treat files.

4. Apply projector. This should be done in one of the special treat files. The optimal position depends on the integrals to be computed. From now on only scalar integrals without any free indices are present.

5. Expansion of the scalar denominators in the small quantities (mass and/or momen-

tum).

6. Perform traces.

7. Do Wick rotation. This is done by multiplying each momentum by the imaginary unit (see also Appendix C). From now on the expression is defined in Euclidean space.

8. Apply derivatives in order to factorize the external momentum. In this context see also the variables `DALA12` and `DALAQN` in Appendix C.4.

As there is the possibility to interact at three different places — after the fermions are treated (`TREAT0`) and before and after the traces are performed (`TREAT1`, respectively, `TREAT2`) the order of the commands may slightly be varied by the user.

At this stage the scalar products in the numerator of the integrals should be formed by either only loop momenta or only external momenta (which then constitutes a trivial prefactor). In the denominator the (scalar) propagators may be raised to arbitrary power.

The next steps constitute the main part of `MATAD`.

9. Express the scalar products of the numerator in terms of the denominators. This produces a "1" in the numerator of the integrals. It might be that this step is very time and memory consuming.

10. Apply recurrence relations to reduce the number of different integrals to simpler ones and to a small set of master integrals.

11. Expand the result in ε and store it in the directory `problems/'PRB'/results/'DIAGRAM'.res` under the name `'DIAGRAM'`.

An expansion in ε is also done at various intermediate steps. Although poles of at most third order may appear for a three-loop vacuum integral terms up to order ε^6 have to be kept in the expansion as artificial poles may appear during the application of the recurrence relations (cf. step 10).

Steps 9 and 10 constitute the central part of `MATAD`. They heavily depend on the loop-order and the topology which has to be specified apart from the very diagram in the folds `d1l1`, `d1l2`, … (see above). Thus let us elaborate on this point in the following. In principle it suffices to define one input topology at one-, two- and three-loop order, where the number of internal lines amount to one, three and six, respectively (see Fig. 1). If one allows each line to be massless and carry the mass M at the same time these three topologies are sufficient to cover all possible cases that can occur in the calculation of one-, two- and three-loop vacuum integrals. Note that a partial fractioning for terms like

$$\frac{1}{(p^2)^a\,(p^2+M^2)^b}\,, \qquad (2)$$

where a and b are positive integers, leads to the same topologies with the only difference that now each line is either massless of massive. It is, however, not at all practical to

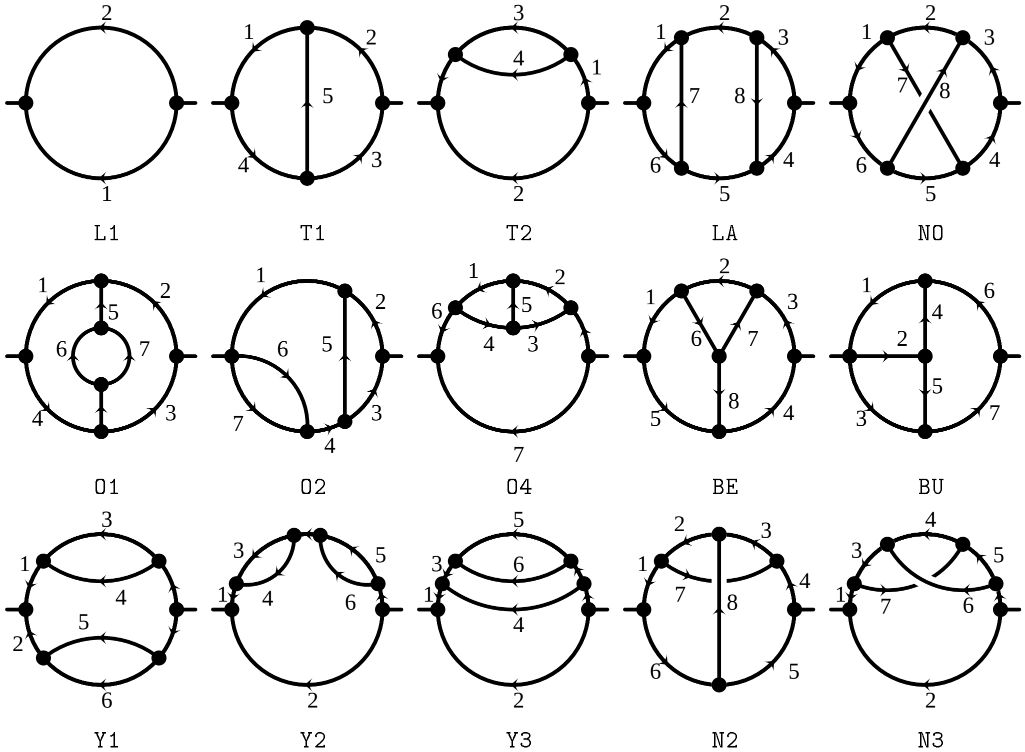

Figure 3: The input of the diagram to be computed can be mapped to one of these topologies. The implemented massive/massless combinations can be found in Appendix B. The momentum p_i flows through line i as indicated by the arrow.

rewrite the input to the notation of Fig. 1 before generating the file `dia.<prb>`. On the contrary it is advantageous to enlarge the input topologies. Currently the topologies shown in Fig. 3 are implemented in `MATAD`. The momentum p_i flowing through line i can be expressed as a linear combination of the loop momenta. For our purpose these relations are, however, not of interest. The choice of the topologies was guided by the package `MINCER` [4] and for convenience the same notation concerning the definition of the momenta p_i has been adopted. The implemented massive/massless combinations of each topology are listed in Appendix B. After the declaration of the diagram in the folds `d1l1`, `d1l2`, ...the corresponding topology is specified via

```
#define TOPOLOGY "XY"
```

where `XY` corresponds to one of the topologies of Fig. 3.

The notation concerning the momenta as introduced in Fig. 3 is quite convenient to be used for the input. However, the very recursion procedure is formulated for the three-loop topology of Fig. 1 where the lines are either massive or massless. This leads to 14

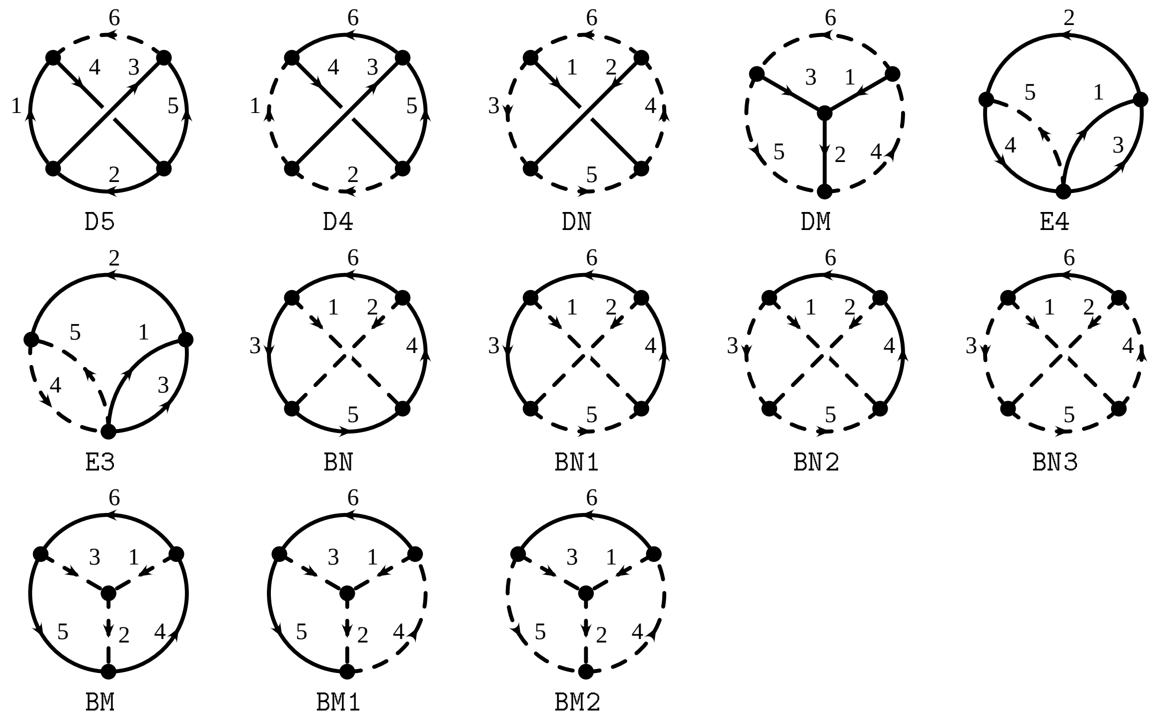

Figure 4: Basic three-loop topologies implemented into MATAD. They are reduced to either simple integrals or master integrals with the help of recurrence relations.

different cases which are classified in [8]. Thus before step 9 is performed the momenta are transformed from the notation of Fig. 3, which is used in the input, to the so-called basic topologies shown in Fig. 4. For them — after decomposing the scalar products in the numerator into parts of the denominator — the very recursion procedure is performed. Note that at this stage all propagators may be raised to an arbitrary integer power.

Actually some of the three-loop diagrams (e.g. BN3) can be computed by the successive use of one- and two-loop procedures for massless propagator type diagrams or vacuum integrals, respectively. In such cases some of the (one- and two-loop) routines from MINCER [4] are used for parts of the computation. The corresponding procedures are listed in Appendix D. For other cases (e.g. E4 or BN2) simple relations reduce one of the lines to zero and the resulting diagram can again be solved easily. Only for the cases D5, D4, DN, DM, E3, BN and BN1 the recursion procedure has to be applied until one arrives at master integrals. They coincide with the corresponding diagrams of Fig. 4 where all denominators are raised to power one. Only the one pictured in Fig. 5, which results from BN1, is needed in addition. From this diagram even the $\mathcal{O}(\varepsilon)$ part is required. The analytic expressions are given in Appendix C.3. It should be mentioned that the recurrence relations for BM are quite involved. However, for this topology no difficult mas-

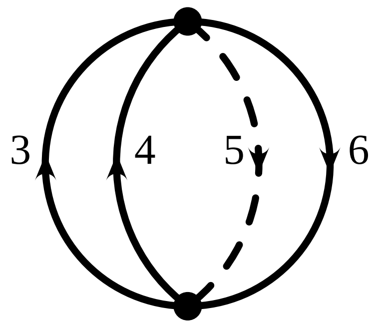

Figure 5: Master diagram resulting from topology BN1. Here all propagators are raised to power one. Its constant part contains the expression S2 and its $\mathcal{O}(\varepsilon)$ part OepS2 as listed in Appendix C.3.

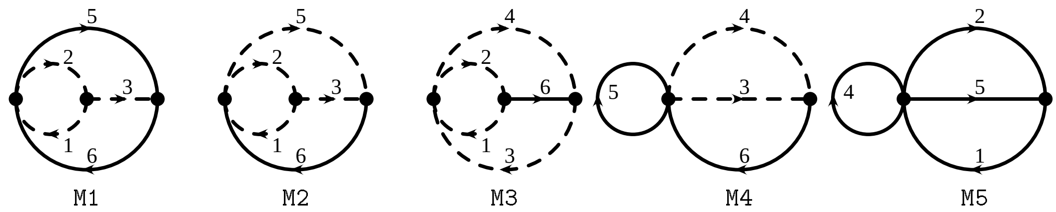

Figure 6: Simple integrals which occur as a result of the recurrence procedure next to the master integrals. All propagators may be raised to an arbitrary power.

ter integral is needed. Note that the basic topologies BM1 and BM2 actually coincide with BN2 and BN3, respectively. For convenience partly different codes had been written which actually turned out to be quite useful while debugging MATAD.

The recurrence relations leading to simple diagrams and master integrals for the three-loop topologies are derived in Refs. [5] and [8]. Except the case where all six lines are massive all of them are implemented into MATAD. The reason why this case is missing is simply that it was not yet needed for practical calculations. All master integrals are listed in Appendix C.3.

In Fig. 6 the simple integrals are listed which also result from the recursion procedure. They can all be expressed in terms of Γ functions by the successive use of results for massless one-loop two-point functions, $P_{ab}(Q)$, in combination with one- (V_a) and two-loop (V_{abc}) vacuum integrals. For convenience we want to list the explicit results for $P_{ab}(q)$, V_a and V_{abc} in Euclidean space:

$$P_{ab}(Q) \;=\; \int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{1}{p^{2a}\,(p+Q)^{2b}}$$

$$
\begin{aligned}
&= \frac{\left(Q^2\right)^{D/2-a-b}}{(4\pi)^{D/2}} \frac{\Gamma(a+b-D/2)\Gamma(D/2-a)\Gamma(D/2-b)}{\Gamma(a)\Gamma(b)\Gamma(D-a-b)}\,, \\
V_a &= \int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{1}{(p^2+M^2)^a} \;=\; \frac{\left(M^2\right)^{D/2-a}}{(4\pi)^{D/2}} \frac{\Gamma(a-D/2)}{\Gamma(a)}\,, \\
V_{abc} &= \int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{(p^2+M^2)^a (k^2+M^2)^b ((p+k)^2)^c} \\
&= \frac{\left(M^2\right)^{D-a-b-c}}{(4\pi)^D} \frac{\Gamma(a+b+c-D)\Gamma(a+c-D/2)\Gamma(b+c-D/2)\Gamma(D/2-c)}{\Gamma(a)\Gamma(b)\Gamma(a+b+2c-D)\Gamma(D/2)}\,.
\end{aligned}
\tag{3}
$$

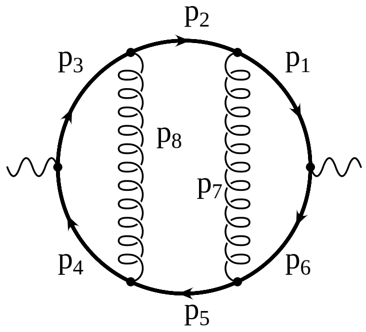

Figure 7: Three-loop diagram of type `LA`.

The vacuum integrals occurring at one- and two-loop level are quite simple. Actually most of them can be expressed in terms of Γ functions for arbitrary exponents of the propagators and no recursion relations are needed (cf. Eq. (3)). Only for the two-loop integral in Fig. 1 where all three lines carry the mass M it is useful to implement simple recurrence relations which reduce the integrals to one master integral where all exponents are raised to the first power only. For three-loop calculations the result of this integral is needed up to $\mathcal{O}(\varepsilon)$ (`T1ep`, see Appendix C.3).

For definiteness we want to consider an explicit example. Let us consider the three-loop diagram of Fig. 7 which we would like to expand up to fourth order in the external momentum, q_1. For simplicity we neglect the tensor structure and consider only the scalar integral obtained in the case when the full line represents a massive particle and the curly line a massless one. In the directory `problems/scalar/` one can find the following files

```
mainscalar  results/  scalar.dia
```

where `results/` is a directory to store the result of the diagram. The file `scalar.dia` looks as follows

```
*--#[ TREAT0:
*--#] TREAT0:

*--#[ TREAT1:
*--#] TREAT1:

*--#[ TREAT2:
*--#] TREAT2:

*--#[ TREATMAIN:
*--#] TREATMAIN:

*--#[ scalar:
        M^4
        *s1m
        *s2m
        *s3m
        *Dh(p4,q1)
        *Dh(p5,q1)
        *Dh(p6,q1)
        /p7.p7
        /p8.p8
        ;

        #define TOPOLOGY "LA"
*--#] scalar:
```

In this case no special treat file is needed. The very diagram can be found in the fold `scalar` where the multiplication with M^4 is done in order to end up with a dimensionless expression. The external momentum is routed through the lines 4, 5 and 6 as can be seen in the arguments of the function `Dh`. The other massive propagators are denoted by `s1m`, `s2m` and `s3m`. The massless lines translate into the factors `1/p7.p7` and `1/p8.p8`. For more details on the notation we refer to Appendix C.1. The file `mainscalar` looks as follows:

```
#define PRB "scalar"
#define DALAQN "q1"
#define GAUGE "0"
#define POWER "4"
#define CUT "0"
#define FOLDER "scalar"
#define DIAGRAM "scalar"
#-
#include main.gen
```

With the command

```
#define POWER "4"
```

we require an expansion up to fourth order in q_1 and

```
#define DALAQN "q1"
```

effectively factors out the terms $(q_1^2)^n$. The calculation is initiated with the command

```
> matadform problems/scalar/mainscalar
```

and after a few seconds the final result is displayed on the screen

```
   scalar =
       - 2 + 2*z3 - Q1.Q1*z3*M^-2 + 227/216*Q1.Q1*M^-2 + 1/2*Q1.Q1^2*z3*M^-4
          - 1876/3375*Q1.Q1^2*M^-4;
```

As expected it is finite and contains three terms in the expansion in q_1^2/M^2. The result is stored to the directory `problems/scalar/results/` and can be read with the `FORM` command `load`. Details on the notation of the output and the used conventions can be found in Appendix C.

3 Some details on the internal structure

Once the recurrence relations are implemented it is in principle possible to compute diagrams of arbitrary complexity. However, in practice one arrives quite soon at the limits set by the soft- or hardware. The algebra language `FORM` is designed to deal with large expressions. Still it happens quite easily that in internal steps the expressions exceed several Megabytes and even approach the order of a few Gigabyte. This significantly slows down the performance and it is advantageous to implement several tricks. Some of them are described in this section.

• During the recurrence procedure it happens very often that whole blocks of commands have to be executed until the recursion has reached an end. Within `FORM` there are the commands

```
  repeat;
    [...]
  endrepeat;
```

which in principle allows for such a construction. However, in practice it is not possible to use between `repeat` and `endrepeat` a command which forces `FORM` to combine identical terms like, e.g., `.sort`. For this reason a preprocessor variable, `NOR` (see also Appendix C.4), has been introduced which in combination with the construction

```
  #do i=1,'NOR'
    [...]
  #enddo
```

allows the use of `.sort` in intermediate steps of the recursion commands.

• At this point we should also mention that the procedure `ACCU` has been adopted from `MINCER` [4]. It collects in the argument of the function `acc()` the polynomials in ε and thus significantly reduces the number of terms. E.g., the expression

```
x1*x2 + 4*ep*x1*x2 + 12*ep^3*x1*x2
```

transforms after `#call ACCU{test}` to

```
acc(1 + 4*ep + 12*ep^3)*x1*x2
```

and instead of three only one term has to be treated in the following commands.

• Very often it happens that propagators of the type $1/(M^2 - (p+q)^2)^n$ have to be expanded in the momentum q and afterwards derivatives w.r.t. q are applied in order to factor out powers of q^2. It turns out that it is very useful to expand in a first step only the part $2pq$ and keep the factors q^2 unexpanded in the form $1/(M^2 - (p^2 + q^2))^n$. Thus less terms have to be considered while the derivatives are applied. Afterwards the expansion in q^2 is performed. A related discussion can also be found in [16].

• The expansion of the scalar denominators in a small momentum is also very time consuming — especially for high values of 'POWER'. In order to do this in an effective way the variable `poco`, which is an abbreviation for "power counting", is defined via

```
S poco(:'POWER');
```

after the declaration of the preprocessor variable `POWER`. This definition ensures that the terms involving $\texttt{poco}^n$ with $n >$`'POWER'` are automatically set to zero. Depending on the problem `poco` should also be considered in the special treat files.

• The application of the recurrence relations can lead to spurious $1/\varepsilon$ poles (cf. Appendix A) which are in general quite dangerous if an expansion in ε is performed in intermediate steps. In `MATAD` at most three $1/\varepsilon$ poles arise from the recurrence relations. Together with a possible $1/\varepsilon^3$ term from the three-loop integrals an expansion up to order ε^6 has to be done in intermediate steps in order to get the correct constant term.

• For quite a lot of applications the topology `BN` (cf. Fig 4) plays a crucial role. The original recurrence procedure for this toplogy to master integrals was proposed in [5]: In a first step the exponent of three out of the four massive denominators are reduced to one. Then the exponents of the massless lines are reduced to zero. Finally the remaining line is treated and one arrives (apart from simple integrals) at an integral consisting of four massive lines connecting two vertices. It is connected to the corresponding master integral of Fig. 4 through a simple relation.

The equation involved in the last recursion step is quite involved. It generates from each term more than ten terms at each call. Note that the exponent of the last massive denominator gets increased by the proceeding steps. This enormously slows down the calculation — in some cases it makes it even impossible. The idea to circumvent this problem is based on the observation that the massless exponents can be reduced to zero even if none of the massive ones is reduced to one. The corresponding recurrence relations are short and thus one arrives with only little effort at three-loop integrals with four massive lines, $B_N(0,0,n_3,n_4,n_5,n_6)$ (cf. Fig. 5 where all lines are massive and the exponents of the denominators are given by n_3, n_4, n_5 and n_6). These diagrams are now treated in the following way: Temporarily an external momentum is introduced which flows through one of the lines. We choose line 3 for definiteness. In a second step the operator $\Box_q = \partial/\partial q^\mu \partial/\partial q_\mu$ is applied and q is set to zero afterwards. This leads to an equation connecting $B_N(0,0,n_3,n_4,n_5,n_6)$, $B_N(0,0,n_3-1,n_4,n_5,n_6)$ and $B_N(0,0,n_3-2,n_4,n_5,n_6)$

$$
\begin{aligned}
B_N(0,0,n_3,n_4,n_5,n_6) &= -\frac{1}{4M^2(n_3-2)(n_3-1)}\Box_q B_N(0,0,n_3-2,n_4,n_5,n_6) \\
&\quad + \frac{-2D+4(n_3-1)}{4M^2(n_3-1)} B_N(0,0,n_3-1,n_4,n_5,n_6)\,, \qquad (4)
\end{aligned}
$$

which can be applied until $n_3 = 3$. As the other indices are not affected the same procedure can be applied to them as well and one ends up with the integrals $B_N(0,0,1,1,1,1)$, $B_N(0,0,2,1,1,1)$, $B_N(0,0,2,2,1,1)$, $B_N(0,0,2,2,2,1)$ and $B_N(0,0,2,2,2,2)$. Both their values for $q=0$ and the result for the application of $(\Box_q)^n$ has to be known where the index n depends on how often Eq. (4) had to be applied. The overall number of the integrals needed is still small and for most practical applications well below 100. They can be computed once and for all using the method of Ref. [5] and can be collected in a table. Currently the table contains all results up to $n=10$ and partly for $n=11$. In case the table is too small the original recurrence procedure [5] has to be used. This is done by defining the preprocessor variable `BNRECOLD` in the main file.

4 Examples

In this section we want to discuss some typical examples which can be treated with `MATAD`. In particular the content of the `TREAT` folds and the switches in `main.<prb>` shall be discussed in detail. In Section 4.1 two-loop corrections to the photon polarization function are considered and in Section 4.2 the calculation of three-loop vertex corrections contributing to the Higgs decay into gluons is discussed. As a last example we consider the computation of the fermion propagator in the limit of a small external momentum.

4.1 Photon polarization function

In this section only a calculation of two-loop diagrams is presented. However, we want to take this opportunity to show a concept which can also be used for the treatment of more complex problems. In particular it is shown how the complete calculation can be automated and one can ensure that all results are indeed up-to-date.

To be precise, we want to consider the two-loop diagrams induced by a massive quark to the photon polarization function. The problem shall be called `Pi`. Then the file `Pi.dia` looks as follows:

```
*
* problem: Pi
*

*--#[ TREAT0:
```

```
  #message project out transversal part
  multiply, (d_(mu1,mu2)-q1(mu1)*q1(mu2)/q1.q1)*deno(3,-2);
  .sort
*--#] TREAT0:

*--#[ TREAT1:
*--#] TREAT1:

*--#[ TREAT2:
*--#] TREAT2:

*--#[ TREATMAIN:
*--#] TREATMAIN:

*
* 2-loop diagrams for problem Pi
*

*--#[ d2l1:
((-1)
*Dg(nu1,nu2,p5)
*S(mu1,q1,p1m,nu1,q1,p2m,mu2,p3m,nu2,p4m)
*1);

#define TOPOLOGY "T1"
*--#] d2l1:

*--#[ d2l2:
((-1)
*Dg(nu1,nu2,p4)
*S(mu1,q1,-p2m,mu2,p1m,nu2,p3m,nu1,p1m)
*1);

#define TOPOLOGY "T2"
*--#] d2l2:

*--#[ d2l3:
((-1)
*Dg(nu1,nu2,p4)
*S(nu1,p3m,nu2,p1m,mu2,-q1,-p2m,mu1,p1m)
*1);

#define TOPOLOGY "T2"
*--#] d2l3:
```

The function `deno(x,y)` means $1/(x + y\varepsilon)$ and can be used for denominators. `TREAT0` contains the projector to the transversal part and the three contributing diagrams are named `d2l1`, `d2l2` and `d2l3`. The external momentum is denoted by `q1`. The mass of the quarks, which in the final result appears as `M`, is introduced through adding the symbol `m` to the corresponding momentum which is defined through the topologies `T1` and `T2` in Fig. 3. We want to assume that $q_1^2 \ll M^2$ and thus perform an expansion in q_1. This is achieved with the notation `S(...,q1,p1m,...)`. For more details concerning the notion we refer to Appendix C.

The file `mainPi` reads:

```
#define PRB "Pi"
#define PROBLEM0 "1"
#define DALAQN "q1"
#define GAUGE "xi"
#define POWER "4"
#define CUT "1"
#define FOLDER "Pi"
***#define DIAGRAM "d2l1"
#-
#include main.gen
```

For most of the specified variables we refer to Appendix C.4. We only want to mention that a general gauge parameter `xi` is chosen with the command `#define GAUGE "xi"`. This allows for an explicit check that the final result, i.e. the sum of the three diagrams is gauge parameter independent. The definition `#define POWER "4"` requests for an expansion up to fourth order in `q1`. The definition of the variable `DIAGRAM` is commented as it will be defined during the call of `mainPi` (see below).

The computation of each diagram could be started separately and the results could be summed at the end. Instead we want to take the opportunity to present a method which is unavoidable for problems where a large number of diagrams contribute. In the following we want to present two more files which can easily be adopted to other problems. The first one, `makePi`, is a so-called `GNU make` file and could look as follows:

```
SHELL = /bin/sh

DIA2 = \
      problems/Pi/results/d2l1.res\
      problems/Pi/results/d2l2.res\
      problems/Pi/results/d2l3.res

problems/Pi/results/Pi.2.res: $(DIA2)
        matadform problems/Pi/comPi > problems/Pi/log/comPi.log

ii = $(notdir $(basename $@))

$(DIA2): problems/Pi/mainPi
        if [ -f problems/Pi/results/$(ii).res ]; \
           then rm problems/Pi/results/$(ii).res; fi
        time matadform -d DIAGRAM=$(ii) \
          problems/Pi/mainPi > problems/Pi/log/$(ii).log;
        if [ -f problems/Pi/results/$(ii).res ]; \
           then rm problems/Pi/log/$(ii).log; fi
```

For details concerning the individual commands we refer to the literature [17]. For us it is only important that at the beginning the diagrams we want to compute are listed.

Furthermore, after the line '`problems/Pi/results/Pi.2.res: $(DIA2)`' the command is given which specifies what shall be done once the computation of the individual diagrams is finished: The diagrams are summed with the help of the program `comPi`

```
*
* comPi
*
#-
#include declare.matad
#define PRB "Pi"
.global

#do i=1,3
  load problems/'PRB'/results/d2l'i'.res;
#enddo

g res'PRB'2 =
#do i=1,3
  + d2l'i'
#enddo
;

b ep;
print;
.store

#+
save problems/'PRB'/results/res'PRB'.2.res res'PRB'2;
.end
```

To initiate the calculation one simply has to specify the command

```
> make -f problems/Pi/makePi
```

where `make` has to call the `GNU` version [17] of the `make` command. The final result is stored in the file `problems/Pi/results/resPi.2.res`. It reads

```
   resPi2 =
       + ep^-1 * (  - 6*Q1.Q1 + 8/5*Q1.Q1^2*M^-2 )

       + ep * (  - 35/6*Q1.Q1 - 6*Q1.Q1*z2 + 8/5*Q1.Q1^2*M^-2*z2 + 3116/1215*
         Q1.Q1^2*M^-2 )

       + 13/3*Q1.Q1 - 128/405*Q1.Q1^2*M^-2;
```

Indeed, as expected, there is no trace of the gauge parameter `xi`.

4.2 Higgs decay into two gluons

Party for convenience and partly for historical reasons the notation of the input topologies in Fig. 3 is closely connected to two-point functions. However, MATAD only deals with vacuum diagrams independent of the number of external legs. In this section we want to show that also problems involving at first sight three-point functions can be approached using MATAD.

Let us consider QCD corrections to the decay of the Standard Model Higgs boson into two gluons. It is convenient to construct an effective theory where the top quark is integrated out. Details on the theoretical background can be found in [18]. Here it shall only be mentioned that the coefficient function which contains the dependence on the mass of the top quark, M_t, can be computed from triangle diagrams as pictured in Fig. 8. According to their Lorentz structure the result can be written as follows

$$K(M_t)\ (q_1^\nu q_2^\mu - q_1 q_2 g^{\mu\nu})\ , \tag{5}$$

where q_1 and q_2 are the momenta of the gluons with polarization vectors $\epsilon^\mu(q_1)$ and $\epsilon^\nu(q_2)$. Thus the vertex diagrams have to be expanded up to linear order both in q_1 and q_2 and an appropriate projector has to be applied in order to get $K(M_t)$.

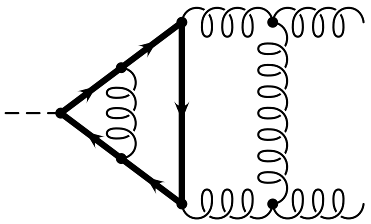

Figure 8: Sample diagram contributing to the decay of the Higgs boson. Solid and looped lines represent quarks and gluons, respectively.

The file containing the diagrams could look as follows

```
*
* problem: hgg
*

*--#[ TREAT0:
multiply, (
  a*deno(2,-2)*(q1.q2*d_(mu,nu)-q2(nu)*q1(mu)-q2(mu)*q1(nu))
 +b*deno(2,-2)*(-q1.q2*d_(mu,nu)+(3-2*ep)*q2(nu)*q1(mu)+q2(mu)*q1(nu))
);
.sort
*--#] TREAT0:

*--#[ TREAT1:
*--#] TREAT1:
```

```
*--#[ TREAT2:
*--#] TREAT2:

*--#[ TREATMAIN:
*--#] TREATMAIN:

*--#[ d31335:
        ((-1)
        *M
        *Dg(nu1,nu2,p1)
        *Dg(nu7,nu8,-p4)
        *Dg(nu3,nu4,q1,-p1)
        *Dg(nu5,nu6,q2,p1)
        *S(-q1,-p3m,nu7,-q1,p5m,nu4,-p2m,nu6,q2,p5m,nu8,q2,-p3m)
        *V3g(mu,q1,nu1,-p1,nu3,p1-q1)
        *V3g(nu,q2,nu2,p1,nu5,-p1-q2)
        *1);

        #define TOPOLOGY "O4"
*--#] d31335:
```

where the diagram `d31335` corresponds to the one shown in Fig. 8. The fold `TREAT0` contains (up to an overall factor $(q_1.q_2)^{-2}$) the projector on the coefficients in front of the structures $g^{\mu\nu}$ and $q_1^\nu q_2^\mu$ of Eq. (5). They are marked by the symbols `a` and `b`, respectively. Thus the transversality of Eq. (5) can be explicitly checked in the sum of all contributing diagrams (the result of a single diagram does in general not have a transverse structure).

The corresponding `main`-file is very similar to the one listed in Section 4.1. The only difference (apart from replacing `Pi` by `hgg`) is that the commands

```
#define DALAQN "q1"
#define GAUGE "xi"
#define POWER "4"
#define CUT "1"
```

should be replaced by

```
#define DALA12 "1"
#define GAUGE "0"
#define POWER "2"
#define CUT "0"
```

The third line ensures that an expansion of the integrand up to second order in the external momenta is performed and the first one sets q_1^2 and q_2^2 to zero and factors out the scalar product $q_1 q_2$. `#define CUT "0"` sets ε to zero in the final result as the terms of $\mathcal{O}(\varepsilon)$ are anyway not computed completely at three-loop order. In this example we choose Feynman gauge which is achieved with `#define GAUGE "0"`.

After calling `MATAD` it takes of the order of a minute to obtain the result:

```
d31335 =
    + ep^-2 * ( 40*Q1.Q2*M^2*a + 344/9*Q1.Q2^2*a - 232/9*Q1.Q2^2*b )

    + ep^-1 * (  - 308/3*Q1.Q2*M^2*a - 3530/27*Q1.Q2^2*a + 1786/27*Q1.Q2^2*
      b )

    + 60*Q1.Q2*M^2*z2*a + 734/3*Q1.Q2*M^2*a - 1936/9*Q1.Q2^2*z3*a + 1136/9*
      Q1.Q2^2*z3*b + 172/3*Q1.Q2^2*z2*a - 116/3*Q1.Q2^2*z2*b + 46817/81*
      Q1.Q2^2*a - 26239/81*Q1.Q2^2*b;
```

Note that the terms proportional to `Q1.Q2*M^2` cancel after adding all contributing diagrams.

4.3 Fermion propagator

In this example we compute the small-momentum expansion of a three-loop diagram which contributes to the fermion propagator. Recently these kind of diagrams have been considered in different kinematical regions in order to obtain the three-loop relation between the $\overline{\mathrm{MS}}$ and on-shell quark mass [19, 11]. This subsection contains all relevant input information whereas the complete output is given in Appendix E.

The fermion self energy, $\Sigma(q)$, can be decomposed into a scalar and vector part

$$\Sigma(q) \;=\; M\Sigma_S(q^2) + \not{q}\,\Sigma_V(q^2)\,, \tag{6}$$

where q is the external momentum and M is the mass of the quark. We are interested in the computation of the scalar functions Σ_S and Σ_V.

The file `mainfp` looks as follows

```
#define PRB "fp"
#define PROBLEMO "1"
#define DALAQN "q1"
#define GAUGE "0"
#define POWER "1"
#define CUT "0"
#define FOLDER "fp"
#define DIAGRAM "d3l79"
#-
#include main.gen
```

The variable `POWER` is defined in such a way that an expansion up to third order is performed. This leads to the constant and order q^2/M^2 terms for the functions Σ_S and Σ_V as $\Sigma(q)$ itself has mass dimension one. The corresponding projectors and the diagram to be computed look as follows (`fp.dia`):

```
*
* problem: fp
*
```

```
*--#[ TREAT0:
  multiply, 1/4*(1/M + a * g_(1,q1)/q1.q1);
*--#] TREAT0:

*--#[ TREAT1:
*--#] TREAT1:

*--#[ TREAT2:
*--#] TREAT2:

*--#[ TREATMAIN:
*--#] TREATMAIN:

*--#[ d3l79:
        ((1)
        *Dg(nu3,nu4,p5)
        *Dg(nu5,nu6,-p6)
        *Dg(nu1,nu2,-q1,-p1)
        *S(nu2,-p1m,nu5,-p4m,nu4,-p3m,nu6,-p2m,nu3,-p1m,nu1)
        *1);

        #define TOPOLOGY "O4"
*--#] d3l79:
```

Note that in the fold `TREAT0` the vector part gets multiplied by `a` in order to distinguish Σ_V from Σ_S in the final result. The input for the diagram corresponds to the one pictured in Fig. 9. The complete output appearing on the screen and the result can be found in the Appendix E.

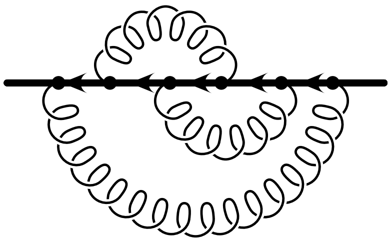

Figure 9: Sample diagram contributing to the fermion propagator. Solid and looped lines represent quarks and gluons, respectively.

Acknowledgments

It is a pleasure for me to thank K.G. Chetyrkin, R. Harlander, J.H. Kühn and T. Seidensticker for many discussions and useful advice within the recent years when `MATAD` was developed. This work was supported in part by DFG under Contract Ku 502/8-1 (*DFG-Forschergruppe "Quantenfeldtheorie, Computeralgebra und Monte-Carlo-Simulationen"*) and by SUN Microsystems through Academic Equipment Grant No. 14WU0148. I am grateful to the department of Theoretical Particle Physics of the University of Karlsruhe for the pleasant atmosphere during a visit when a major part of this project was carried out.

Appendix

A Integration-by-parts

The method of integration-by-parts plays a fundamental role in the computation of multi-loop diagrams [2]. For this reason we want to demonstrate its underlying idea by considering a typical example.

The integration-by-parts algorithm uses the fact that the D-dimensional integral over a total derivative is equal to zero:

$$\int \mathrm{d}^D p \frac{\partial}{\partial p^\mu} f(p, \ldots) \quad = \quad 0 \,. \tag{7}$$

By explicitly performing the differentiations one obtains recurrence relations connecting Feynman integrals of different complexity. The proper combination of different recurrence relations allows any Feynman integral (at least single-scale ones) to be reduced to a small set of so-called master integrals. The latter ones have to be evaluated only once and for all, either analytically or numerically.

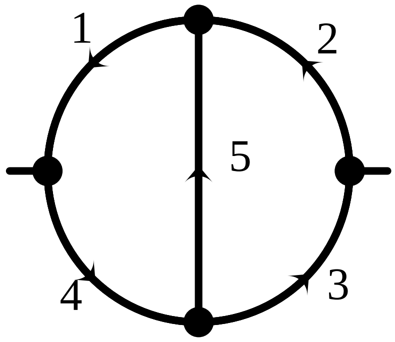

Figure 10: Two-loop master diagram. The arrows denote the direction of momentum flow.

Let us for definiteness consider the scalar two-loop diagram of Fig. 10. The corresponding Feynman integral shall be denoted by

$$I(n_1, \ldots, n_5) = \int \frac{\mathrm{d}^D p}{(2\pi)^D} \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{(p_1^2 + m_1^2)^{n_1} \cdots (p_5^2 + m_5^2)^{n_5}} \,, \tag{8}$$

where $p_1, \ldots, p_5$ are combinations of the loop momenta p, k and the external momentum q (we work in Euclidean space here). $n_1, \ldots, n_5$ are called the indices of the integral. Consider the sub-loop defined by the lines 2, 3 and 5, and take its loop momentum to be $p = p_5$. If we then apply the operator $(\partial/\partial p_5) \cdot p_5$ to the *integrand* of I, we obtain a relation of the form (7), where

$$f(p_5, \ldots) = \frac{p_5^\mu}{(p_5^2 + m_5^2)^{n_5}(p_2^2 + m_2^2)^{n_2}(p_3^2 + m_3^2)^{n_3}} \,. \tag{9}$$

Performing the differentiation and using momentum conservation at each vertex one derives the following equation:

$$\Big[- n_3 \mathbf{3}^+ \left(\mathbf{5}^- - \mathbf{4}^- + m_4^2 - m_5^2 - m_3^2\right) - n_2 \mathbf{2}^+ \left(\mathbf{5}^- - \mathbf{1}^- + m_1^2 - m_5^2 - m_2^2\right) \\ + D - 2n_5 - n_3 - n_2 + 2n_5 m_5^2 \mathbf{5}^+ \Big] \, I(n_1, \ldots, n_5) \;=\; 0 \,, \tag{10}$$

where the operators $\mathbf{1}^\pm, \mathbf{2}^\pm, \ldots$ are used in order to raise and lower the indices: $\mathbf{I}^\pm I(\ldots, n_i, \ldots) = I(\ldots, n_i \pm 1, \ldots)$. In Eq. (10), generally referred to as the triangle rule, it is understood that the operators to the left of $I(n_1, \ldots, n_5)$ are applied *before* integration. If the condition $m_5 = 0, m_3 = m_4$ and $m_1 = m_2$ holds, increasing one index always means to reduce another one. Therefore this recurrence relation may be used to shift the indices n_1, n_4 or n_5 to zero which leads to much simpler integrals.

The triangle rule constitutes an important building block for the general recurrence relations. The strategy is to combine several independent equations of the kind (10) in order to arrive at relations connecting one complicated integral to a set of simpler ones. For example, while the direct evaluation of even the completely massless case for the diagram in Fig. 10 is non-trivial, application of the triangle rule (10) leads to

$$I(n_1, \ldots, n_5) \;=\; \frac{1}{D - 2n_5 - n_2 - n_3} \Big[n_2 \mathbf{2}^+ \left(\mathbf{5}^- - \mathbf{1}^-\right) + n_3 \mathbf{3}^+ \left(\mathbf{5}^- - \mathbf{4}^-\right) \Big] \, I(n_1, \ldots, n_5) \,. \tag{11}$$

Repeated application of this equation reduces one of the indices n_1, n_4 or n_5 to zero. For example, for the simplest case ($n_1 = n_2 = \ldots = n_5 = 1$) one obtains the equation pictured in Fig. 11: The non-trivial diagram on the l.h.s. is expressed as a sum of two quite simple integrals which can be solved by applying the one-loop formula. This example also shows a possible trap of the integration-by-parts technique. In general its application introduces artificial $1/\varepsilon$ poles which cancel only after combining all terms. They require the expansion of the individual terms up to sufficiently high powers in ε in order to obtain, for example, the finite part of the original diagram. This point must carefully be respected in computer realizations of the integration-by-parts algorithm: One must not cut the series at too low powers because then the result goes wrong; keeping too many terms, on the other hand, may intolerably slow down the performance.

In our example, the l.h.s. in Fig. 11 is finite, each term on the r.h.s., however, develops $1/\varepsilon^2$ poles. The first three orders in the expansion for $\varepsilon \to 0$ cancel, and the $\mathcal{O}(\varepsilon)$ term of the square bracket, together with the $1/\varepsilon$ in front of it, leads to the well-known result (omitting factors $1/16\pi^2$): $I(1,1,1,1,1) = 6\zeta(3)/q^2$, where q is the external momentum.

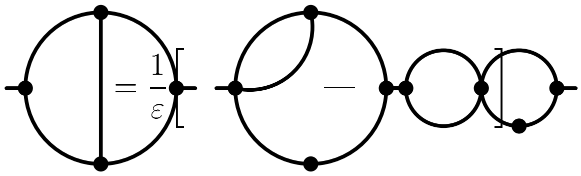

Figure 11: Symbolic equation resulting from Eq. (11) applied to the diagram $I(1,1,1,1,1)$. The dot indicates that the respective denominator appears twice.

In general, the successive application of recurrence relations generates a huge number of terms out of a single diagram. Therefore, a calculation carried out by hand becomes very tedious and the use of computer algebra is essential.

B The topology files

This part of the appendix provides a complete list of those massive/massless combinations which are implemented in the topology files `inc/TOPOLOGY/topXY`. In the following tables for all three-loop topologies of Fig. 3 the lines are listed which have to be massive. All other lines may be massless or absent. In some cases some of the lines have to be completely absent, i.e. it is even not allowed for them to be massless. This is explicitly specified in the columns “absent”.

`L1`							
massive	absent	massive	absent	massive	absent	massive	absent
1	—	2	—	1,2	—		

`T1`							
massive	absent	massive	absent	massive	absent	massive	absent
1,2	—	1,2,4	—	1,2,3,4	—	1,4	—
2,4	—	1	—	2	—	1,2,5	—

`T2`							
massive	absent	massive	absent	massive	absent	massive	absent
1,3	—	2	—	1,2,3	—	2,4	—
3,4	—	3	—	1,2	—		

`BE`							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,5	—	1,2,3	—	4,5	—		

`BU`							
massive	absent	massive	absent	massive	absent	massive	absent
1,3,6,7	—	4,5,6,7	—				

LA							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,5,6	—	1,3,4,6,7,8	—	1,6,7	—	3,4,8	—
1,2,3	—	1,3,7,8	4,5,6	1,7	4,5,6	3,8	4,5,6
N0							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,5,6	—	1,2,3	—				
N2							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,5,6	—	1,2,3,4	—	5,6	—	2,3	—
N3							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,5	—	2	—	1,3,4,5	—	3,4	—
3,4,5	—	4,5	—	1,3	—	1,5	—
O1							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,6,7	—	1,2,3,4	—	6,7	—	1,2,6,7	—
1,2	—	1,5,6	3,4	1,5,7	3,4	5,6	—
5	—	5,7	—	1,5,7	—	7	—
1,5	—	1	—	1,7	—	6	—
1,6,7	—	1,2,7	3,4				
O2							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,4,7	—	3,4,7	—	1,2	—	7	—

O4							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,6,7	—	1,2,6	—	7	—	1,3,5,6	7
1,3,5	—	3,5	—	1,5	—	5	—
1,3,5,7	—	3,5,7	—	1,2	—	1	—
2	—	1,6	—	2,6	—	1,3,7	6
3,7	6	2,6	—	1,3,5,7	6	1,5,7	6
1,2,7	6	2,3	7	1,2,3	7	3,4	7
3,4,5	7	1,3,4	7	1,2,5	7	1,3,4,5,6	7
1,4,5,6	7	1,3,4,5	7	1,4,5	7	1,4	7
3,4,6	7	1,2,3,4	7	2,3,4	7	1,2,3,4,5	7
1,2,4,5	7	1,3,4,6	7	1,2,3,5	7	2,3,5	7
3,4,5,6	7	4,5,6	7	2,3,4,5	7	2,4,5	7
2,4	7	1,2,4	7	1,5	7	1,3	7
3	7	2,5	7	1,2	7	4,5	7
4	7	1,2,4,5,6	7	2,4,5,6	7	6	7
4,6	7	3,5,6	7	3,6	7	1,5,6	7
2,5,6	7	1,3,6	7	1,2,5,6	7	1,2,4,6	7
2,4,6	7	5,6	7	1,4,6	7		

Y1							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,6	—	1,3	—	3	—	3,5,6	—

Y2							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,5	—	2	—	1,3,5	—	3	—
5	—	1,3	—	1,5	—		

Y3							
massive	absent	massive	absent	massive	absent	massive	absent
1,2,3,5	—	1,3,5	—	2	—	5	—
3,5	—	1,3	—				

C Some details on the notation

C.1 Notation of the input

In this subsection we describe the notation to be used for the input.

On one side of the following expressions the `FORM` notation is used whereas the other side displays the corresponding mathematical terms. It is always assumed that p_i $(i = 1, \ldots, 9)$ is a loop momentum and q_i $(i = 1, \ldots, 3)$ is a small momentum in which an expansion is performed.

If scalar integrals are to be computed with `MATAD` the following notation has to be used:

$$
\begin{array}{llcl}
\text{massless lines:} & \frac{1}{p_1^2}, \frac{1}{p_2^2}, \ldots & \longrightarrow & \texttt{1/p1.p1, 1/p2.p2,}\ldots \\
& \frac{1}{-(p_1+q_1)^2}, \ldots & \longrightarrow & \texttt{Dl(p1,q1),}\ldots \\
\text{massive lines:} & \frac{1}{M^2-p_1^2}, \frac{1}{M^2-p_2^2}, \ldots & \longrightarrow & \texttt{s1m, s2m,}\ldots \\
& \frac{1}{M^2-(p_1+q_1)^2}, \ldots & \longrightarrow & \texttt{Dh(p1,q1),}\ldots
\end{array}
\tag{12}
$$

Fermions are treated with the help of the (non-commutative) function `S` and the function `Dg` can be used for the gluon propagator:

$$
\begin{array}{lcllcl}
\texttt{S(mu)} & = & \gamma^\mu , & \texttt{S(p1)} & = & \frac{\not{p}_1}{-p_1^2} , \\
\texttt{S(p1m)} & = & \frac{M+\not{p}_1}{M^2-p_1^2} , & \texttt{S(q1,p1)} & = & \frac{\not{p}_1+\not{q}_1}{-(p_1+q_1)^2} , \\
\texttt{S(q1,p1m)} & = & \frac{M+\not{p}_1+\not{q}_1}{M^2-(p_1+q_1)^2} , & & & \\
\texttt{Dg(mu,nu,p1)} & = & \frac{-g^{\mu\nu}-\xi\frac{p_1^\mu p_1^\nu}{-p_1^2}}{-p_1^2} , & \texttt{Dg(mu,nu,q1,p1)} & = & \frac{-g^{\mu\nu}-\xi\frac{(p_1+q_1)^\mu (p_1+q_1)^\nu}{-(p_1+q_1)^2}}{-(p_1+q_1)^2} .
\end{array}
\tag{13}
$$

Instead of `S` also `SS`, `SSS` or `SSSS` may be used which is useful in case more than one fermion line is involved. The vertices between three gluons and gluon and ghosts are implemented through the following functions (where the colour factors are not taken into account):

$$
\begin{array}{lcl}
\texttt{V3g(i1,p1,i2,p2,i3,p3)} & = & (p_2-p_1)^{i_3}\, g^{i_1 i_2} + (p_3-p_2)^{i_1}\, g^{i_2 i_3} + (p_1-p_3)^{i_2}\, g^{i_3 i_1} , \\
\texttt{Vgh(i1,p1)} & = & -p_1^{i_1} .
\end{array}
\tag{14}
$$

If small momenta are present they can simply be added to p_1, p_2 or p_3. In analogy to the expansion in a small momentum it is possible to introduce functions where an expansion in small masses can be performed.

In the above expressions instead of `q1` also `q2` or `q3` may be chosen as momenta in which an expansion is done. For the loop momenta `p1`, …, `p9` is allowed.

For the Feynman rules given above the user has to check the consistency with own sign conventions. In particular all unnecessary factors like, e.g., the strong coupling constant are omitted. Also the imaginary unit, i, is suppressed. Furthermore the Feynman rules are chosen in such a way that the vertices are proportional to i and all propagators are proportional to $1/i$ which leads to convenient cancellations. Thus each diagram can be written as

$$
i\left(\frac{1}{i}\int\frac{\mathrm{d}^D k_j}{(2\pi)^D}\right)^l \times [\text{expression formed by the Feynman rules of (13) and (14)}\,] , \tag{15}
$$

where l is the number of loops. The factor $1/i$ in front of the integration momenta disappears after the Wick rotation. The overall factor i, which occurs for all diagrams and which is independent of the number of loops, is omitted.

C.2 Notation and conventions of the output

At this point some words about the notation of the output of `MATAD` are in order. As already mentioned, the final expressions are expanded in ε where — following general $\overline{\text{MS}}$ conventions — the factors γ_E and $\ln 4\pi$ have been dropped. Furthermore the factors $(1/16\pi^2)^l$ and $(1/M^{2\varepsilon})^l$ where l is the number of loops are omitted. Concerning the remaining symbols the translation is given in the following table:

ε	`ep`
Euclidean external momenta	`Q, Q1, Q2,` ...
$\zeta_2, \zeta_3, \ldots$ (Rieman zeta function)	`z2, z3,` ...
mass appearing in the integrals	`M`
parametrization of the finite parts of the master integrals	`D5, D4,` ... (see Appendix C.3)

Note that all momenta which occur in the output have to be interpreted in Euclidean space.

C.3 Master integrals

`MATAD` does not insert the complete finite parts of the master integrals automatically. Instead they are parameterized by the symbols `D5`, `D4`, `DM`, `DN`, `B4`, ... where the notation is adopted from the one introduced in Fig. 4. In the following list one can find the corresponding analytical expressions [5, 9, 10, 11, 20][1]

$$
\begin{aligned}
\mathtt{D6} &= 6\zeta_3 - 17\zeta_4 - 4\zeta_2\ln^2 2 + \frac{2}{3}\ln^4 2 + 16\mathrm{Li}_4\left(\frac{1}{2}\right) - 4\left[\mathrm{Cl}_2\left(\frac{\pi}{3}\right)\right]^2 ,\\
\mathtt{D5} &= 6\zeta_3 - \frac{469}{27}\zeta_4 + \frac{8}{3}\left[\mathrm{Cl}_2\left(\frac{\pi}{3}\right)\right]^2 - 16\sum_{m>n>0}\frac{(-1)^m\cos(2\pi n/3)}{m^3 n}\\
&\approx -8.2168598175087380629133983386010858249695 ,\\
\mathtt{D4} &= 6\zeta_3 - \frac{77}{12}\zeta_4 - 6\left[\mathrm{Cl}_2\left(\frac{\pi}{3}\right)\right]^2 ,\\
\mathtt{D3} &= 6\zeta_3 - \frac{15}{4}\zeta_4 - 6\left[\mathrm{Cl}_2\left(\frac{\pi}{3}\right)\right]^2 ,\\
\mathtt{DM} &= 6\zeta_3 - \frac{11}{2}\zeta_4 - 4\left[\mathrm{Cl}_2\left(\frac{\pi}{3}\right)\right]^2 ,\\
\mathtt{DN} &= 6\zeta_3 - 4\zeta_2\ln^2 2 + \frac{2}{3}\ln^4 2 - \frac{21}{2}\zeta_4 + 16\mathrm{Li}_4\left(\frac{1}{2}\right) ,\\
\mathtt{B4} &= -4\zeta_2\ln^2 2 + \frac{2}{3}\ln^4 2 - \frac{13}{2}\zeta_4 + 16\mathrm{Li}_4\left(\frac{1}{2}\right) ,\\
\mathtt{E3} &= -\frac{139}{3} - \frac{\pi\sqrt{3}\ln^2 3}{8} - \frac{17\pi^3\sqrt{3}}{72} - \frac{21}{2}\zeta_2 + \frac{1}{3}\zeta_3
\end{aligned}
$$

[1] Note that there is a misprint in Eq. (22) of Ref. [11]: all four appearing Clausen functions must be squared and the coefficient of ζ_4 in the second to last equation must read "-22" instead of "$-31/2$".

$$
\begin{aligned}
& + 10\sqrt{3}\mathrm{Cl}_2\left(\frac{\pi}{3}\right) - 6\sqrt{3}\mathrm{Im}\left[\mathrm{Li}_3\left(\frac{e^{-i\pi/6}}{\sqrt{3}}\right)\right] , \\
\texttt{S2} \quad = \quad & \frac{4}{9\sqrt{3}}\mathrm{Cl}_2\left(\frac{\pi}{3}\right) , \\
\texttt{OepS2} \quad = \quad & -\frac{763}{32} - \frac{9\pi\sqrt{3}\ln^2 3}{16} - \frac{35\pi^3\sqrt{3}}{48} + \frac{195}{16}\zeta_2 - \frac{15}{4}\zeta_3 + \frac{57}{16}\zeta_4 \\
& + \frac{45\sqrt{3}}{2}\mathrm{Cl}_2\left(\frac{\pi}{3}\right) - 27\sqrt{3}\mathrm{Im}\left[\mathrm{Li}_3\left(\frac{e^{-i\pi/6}}{\sqrt{3}}\right)\right] , \\
\texttt{T1ep} \quad = \quad & -\frac{45}{2} - \frac{\pi\sqrt{3}\ln^2 3}{8} \\
& - \frac{35\pi^3\sqrt{3}}{216} - \frac{9}{2}\zeta_2 + \zeta_3 + 6\sqrt{3}\mathrm{Cl}_2\left(\frac{\pi}{3}\right) - 6\sqrt{3}\mathrm{Im}\left[\mathrm{Li}_3\left(\frac{e^{-i\pi/6}}{\sqrt{3}}\right)\right] , \qquad (16)
\end{aligned}
$$

with $\mathrm{Cl}_2(x) = \mathrm{Im}[\mathrm{Li}_2(e^{ix})]$. Li_2, Li_3 and Li_4 are the Di-, Tri- and Quadrilogarithm, respectively. `S2` both appears in the diagram of Fig. 5 and the two-loop diagram of Fig. 1 where all three lines have the same mass. Their $\mathcal{O}(\varepsilon)$ parts, which can contribute to the finite part of the three-loop results, contain `OepS2` and `T1ep`, respectively. We should mention that those expressions of Eq. (16) which coincide with an existing topology (e.g. `D5` $\leftrightarrow$ `topD5`) indeed agree with the finite part of the corresponding master integral. On the other hand `B4`, e.g., comprises the complicated parts of the finite part of the master integral corresponding to `topBN`, however, not the complete one. `D6` is not yet implemented into `MATAD` and listed for completeness only.

C.4 Parameters and switches in `main.<prb>`

In the following a brief description of the switches in the file `main<prb>` is given. In all cases the `FORM` syntax reads `#define <VAR> "<VAL>"` where `<VAR>` is the variable to be defined and `<VAL>` the corresponding value.

It is required to define at least the following five variables:

`DIAGRAM`: name of the diagram to be computed.

`FOLDER`: the file containing the special treat files and the diagrams is called `'FOLDER'.dia`.

`GAUGE`: determines the choice for the gauge parameter. The variable ξ as defined in the function `Dg` in Eq. (13) is replaced by the value of `GAUGE`. In particular `#define GAUGE "0"` corresponds to Feynman gauge and with `#define GAUGE "xi"` the calculation is performed for general gauge parameter.

`POWER`: determines the depth of the expansion in the small quantities.

`PRB`: name of problem. `PRB` corresponds to the name of the fold where the problem-dependent files can be found.

The definition of the remaining variables is optional:

`BNRECOLD`: if this variable is defined the original recurrence procedure of Ref. [5] is used for the topology `BN`.

`CUT`: determines the depth of the expansion in ε of the final result. At one-, two- and three-loop order at most the terms of order ε^2, ε^1, respectively, ε^0 are reliable. The default value is "2".

`DALA12`: expansion in q_1q_2. If this variable is set q_1^2 and q_2^2 are set to zero and powers in q_1q_2 are factored out. Currently only the expansion up to order $(q_1q_2)^4$ is implemented. Note that only positive powers of q_1q_2 can be treated.

`DALAQN`: apply d'Alembert operator, $\Box_q = \partial/\partial q^\mu \partial/\partial q_\mu$, in order to factor out powers in q^2. In the argument of `DALAQN` the momentum is specified with respect to which the derivatives are performed.

`NOR`: is an abbreviation for Number Of Recursions. As the naive use of the `repeat`–`endrepeat` construction significantly slows down the performance the most complicated procedures are buffered by a `#do`–`#enddo` construction. `'NOR'` constitutes the upper bound of the do-loop. The default value is "10".

`PROBLEM0/1/2/MAIN`: If one of these variables is set a special treat file is read at the corresponding position. The value has to agree with the one defined in the file `<prb>.dia`.

`TIME`: If this variable is set the statistics is printed at various steps of the calculation.

There are more switches in `inc/main.gen`. However, they should not be modified as most of them are in an experimental stage and not sufficiently tested.

D List of files

In this appendix we provide a list of all files belonging to `MATAD`. The directory `inc` contains essentially the include-files. In particular some of the procedures are collected in files which are included at the very beginning. The tables for the topology `BN` are located in `inc/TABLEDAL` and `inc/TABLEREC`. The directories `inc/TREAT` and `inc/TOPOLOGY` essentially contain the files treating the individual (input and basic) topologies and in the folder `prc` some auxiliary procedures are collected.

```
form.set  inc/  matadform  prc/  problems/

inc:
TABLEDAL/  bnm2m         expandDr    nomBM         reduceBM_2  tblBN
TABLEREC/  declare.matad expepgam    nomBN         reduceBN
TOPOLOGY/  denoexp       expnomdeno  nomdecomBN    reduceBN1
```

```
TREAT/     expandBN       main.gen    recursion_2    reduceBN_2
bnbm2prc   expandBNM      matad.info  redcut         symmetryBM
bnbmprc    expandBNM_2    matminprc   redcutnomdeno  symmetryBN

inc/TABLEDAL:
BNd.tbl  BNd0.tbl

inc/TABLEREC:
BN.tbl  BNn1.tbl  BNn1n2.tbl  BNn2.tbl

inc/TOPOLOGY:
topBE    topBN    topBU  topE3  topM2  topN3  topO4.add  topY3
topBM    topBN1   topD4  topE4  topM3  topNO  topT1      topempty
topBM1   topBN2   topD5  topL1  topM4  topO1  topT2
topBM2   topBN3   topDM  topLA  topM5  topO2  topY1
topBM_2  topBN_2  topDN  topM1  topN2  topO4  topY2

inc/TREAT:
treat.dala12  treatbm2   treatbn2   treatd5  treate4  treatm4
treat.dalaav  treatbm_2  treatbn3   treatdm  treatm1  treatm5
treatbm       treatbn    treatbn_2  treatdn  treatm2  treatn1
treatbm1      treatbn1   treatd4    treate3  treatm3  treatt1

prc:
aver.prc   cutep.prc     dalaqn.prc  difvecsc.prc  solveS.prc  treat.prc
aver1.prc  dala12sc.prc  dalasc.prc  pochtabl.prc  tabBN.prc
```

Files taken over from MINCER

```
one.prc       simplify.prc  finish.prc     tabtwo.prc
accu.prc      dotwo.prc     newtwo.prc     two.prc
triangl2.prc  triangle.prc  pochtabl.prc
```

E Fermion propagator: output

In this appendix we present the complete output of the example discussed in Section 4.3. Calling MATAD

```
> matadform problems/fp/mainfp
```

leads to

```
FORM version 2.3 Apr 24 1997

*
* mainfp
*
#define PRB "fp"
#define PROBLEMO "1"
```

```
    #define DALAQN "q1"
    #define GAUGE "0"
    #define POWER "1"
    #define CUT "0"
    #define FOLDER "fp"
    #define DIAGRAM "d3l79"
    #-
*~~  MATAD -- computation of MAssive TADpoles
*~~  read generic main file
*~~  read diagram
    G dia=
    #include problems/'PRB'/'FOLDER'.dia # 'DIAGRAM'
            ((1)
            *Dg(nu3,nu4,p5)
            *Dg(nu5,nu6,-p6)
            *Dg(nu1,nu2,-q1,-p1)
            *S(nu2,-p1m,nu5,-p4m,nu4,-p3m,nu6,-p2m,nu3,-p1m,nu1)
            *1);

            #define TOPOLOGY "O4"
    *--#] d3l79:
    #-
*~~  Treat the traces
*~~  Include special treat-file 0
*~~  Feynman rules for vertices and propagators:
*~~    gluon-ghost-ghost-vertex
*~~    3-gluon-vertex
*~~    gluon propagator
*~~    ghost propagator
*~~  expand denominators
*~~  Dh
*~~  Dl
*~~  1
*~~  1
*~~  Change notation to p1,p2,...
*~~  Trace 1
*~~  Trace 2
*~~  Trace 3
*~~  Trace 4
*~~  treat DL(x)
*~~  Do Wick-rotation
*~~  Apply d Alembertian w.r.t. q1
*~~  average done
*~~  Expand Dr(p,q)
*~~  1
*~~  q_i -> Q_i
*~~  include TOPOLOGY-file
*~~  this is topO4
*~~  Recursion of type d5
*~~  this is topD5
*~~  numerator
*~~  do recursion
```

```
*~~  - done
*~~  Recursion of type d4
*~~  this is topD4
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type dm
*~~  this is topDM
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type dn
*~~  this is topDN
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type e4
*~~  this is topE4
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type e3
*~~  this is topE3
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type bn_2
*~~  this is topBN_2
*~~  numerator
*~~  do recursion
*~~  Use table for BN
*~~  - done
*~~  Recursion of type bn1
*~~  this is topBN1
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type bn2
*~~  this is topBN2
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type bn3
*~~  this is topBN2
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type bm_2
*~~  this is topBM_2
*~~  numerator
*~~  do recursion
*~~  - done
```

```
*~~  Recursion of type bm1
*~~  this is topBM1
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Recursion of type bm2
*~~  this is topBM2
*~~  numerator
*~~  do recursion
*~~  - done
*~~  Integration of the simple integrals
*~~  Recursion of type m1
*~~  this is topM1
*~~  Recursion of type m2
*~~  this is topM2
*~~  Recursion of type m3
*~~  this is topM3
*~~  Recursion of type m4
*~~  this is topM4
*~~  Recursion of type m5
*~~  this is topM5
*~~  perform integration
*~~  Recursion of type t1
*~~  this is treatn1
*~~  Recursion of type n1
*~~  this is treatn1
*~~  Simplify
*~~  Do the "rest"-integration

Time =       5.33 sec    Generated terms =         23
            d3l79        Terms in output =         23
                         Bytes used      =        410

   d3l79 =
       + ep^-3 * (  - 8/3 - 1/3*a )

       + ep^-2 * ( 56/3 - 20/3*a )

       + ep^-1 * ( 112/3 - 16*z3 + 19/2*z2*a - 20*z2 - 97/12*a )

       + 334/3 + 1215/2*S2*a - 1620*S2 + 16*D3*a - 40*D3 - 1141/3*z3*a + 2368/
         3*z3 + 144*z4*a - 288*z4 + 57*z2*a - 156*z2 - 32*a*B4 - 77/6*a + 64*
         B4;

    save problems/'PRB'/results/'DIAGRAM'.res 'DIAGRAM';
    .end
```

References

[1] G. 't Hooft and M. Veltman, *Nucl. Phys.* **B 44** (1972) 189;
C.G. Bollini and J.J. Giambiagi *Nuovo Cim.* **B 12** (1972) 20.

[2] F.V. Tkachov, *Phys. Lett.* **B 100** (1981) 65;
K.G. Chetyrkin and F.V. Tkachov, *Nucl. Phys.* **B 192** (1981) 159.

[3] J.A.M. Vermaseren, *Symbolic Manipulation with FORM*, (Computer Algebra Netherlands, Amsterdam, 1991).

[4] S.A. Larin, F.V. Tkachov, and J.A.M. Vermaseren, Rep. No. NIKHEF-H/91-18 (Amsterdam, 1991).

[5] D.J. Broadhurst, *Z. Phys.* **C 54** (1992) 599.

[6] L. Avdeev, J. Fleischer, S. Mikhailov, and O. Tarasov, *Phys. Lett.* **B 336** (1994) 560; (E) ibid. **B 349** (1995) 597.

[7] K.G. Chetyrkin, J.H. Kühn, and M. Steinhauser, *Phys. Lett.* **B 351** (1995) 331.

[8] L.V. Avdeev, *Comp. Phys. Commun.* **98** (1996) 15.

[9] D.J. Broadhurst, *Eur. Phys. J.* **C 8** (1999) 311.

[10] J. Fleischer and M.Yu. Kalmykov, *Phys. Lett.* **B 470** (1999) 168, hep-ph/9910223 (v2).

[11] K.G. Chetyrkin and M. Steinhauser, *Nucl. Phys.* **B 573** (2000) 617.

[12] V.A. Smirnov, *Mod. Phys. Lett.* **A 10** (1995) 1485;
for explicit examples see also [15].

[13] R. Harlander, Ph. D. thesis, University of Karlsruhe (Shaker Verlag, Aachen, 1998).

[14] Th. Seidensticker, Diploma thesis (University of Karlsruhe, 1998), unpublished.

[15] R. Harlander and M. Steinhauser, *Prog. Part. Nucl. Phys.* **43** (1999) 167.

[16] O.V. Tarasov, talk given at 4^{th} *International Workshop on Software Engineering and Artificial Intelligence for High Energy and Nuclear Physics* (AIHENP95), Pisa, Italy, 3-8 April 1995; Report Nos.: BI-TP-95/19 and hep-ph/9505277.

[17] see, e.g., R.M. Stallman and R. McGrath, *GNU Make, A Program for Directing Recompilation.*

[18] K.G. Chetyrkin, B.A. Kniehl, and M. Steinhauser, *Phys. Rev. Lett.* **79** (1997) 353.

[19] K.G. Chetyrkin and M. Steinhauser, *Phys. Rev. Lett.* **83** (1999) 4001.

[20] A.I. Davydychev and J.B. Tausk, *Nucl. Phys.* **B 397** (1993) 123;
A.I. Davydychev and J.B. Tausk, *Phys. Rev.* **D 53** (1996) 7381;
A.I. Davydychev *Phys. Rev.* **D 61** (2000) 087701.