Loop Calculations with
\textit{FeynArts, FormCalc, and LoopTools}

\textbf{THOMAS HAHN}

Institut für Theoretische Physik, Universität Karlsruhe
D–76128 Karlsruhe, Germany

Three programs are presented for automatically generating and calculating Feynman diagrams: the diagrams are generated with \textit{FeynArts}, algebraically simplified with \textit{FormCalc}, and finally evaluated numerically using the \textit{LoopTools} package. The calculations are performed analytically as far as possible, with results given in a form well suited for numerical evaluation. The latter is then straightforward using the implementations of the one-loop integrals in \textit{LoopTools}.

1. Introduction

With the increasing accuracy of experimental data, one-loop calculations have long since become indispensable. Doing such calculations by hand is arduous and error-prone and in some cases simply impossible. So for some time already, software packages have been developed to automate these calculations (\textit{e.g.} \cite{1, 2}). Yet one remaining obstacle is that these packages generally tackle only part of the problem, and there is still considerable work left in making them work together.

In this paper three packages, \textit{FeynArts, FormCalc, and LoopTools}, are presented which work hand in hand. The user has to supply only small driver programs whose main purpose is to specify the necessary input parameters. This makes the whole system very “open” in the sense that the results are returned as \textit{Mathematica} expressions which can easily be manipulated, \textit{e.g.} to select or modify terms.

\textit{FormCalc} can work either in dimensional regularization or in constrained differential renormalization \cite{3}, the latter of which is equivalent at the one-loop level to regularization by dimensional reduction \cite{4}. This makes \textit{FormCalc} suitable \textit{e.g.} for calculations in supersymmetric models.

Since one-loop calculations can range anywhere from a handful to several hundreds of diagrams (particularly so in models with many particles like
the MSSM), speed is an issue, too. FormCalc, the program which does the algebraic simplification, therefore uses FORM [5] for the time-consuming parts of the calculation. Owing to FORM’s speed, FormCalc can process, for example, the 1000-odd one-loop diagrams of W–W scattering in the Standard Model [6] in about 5 minutes on an ordinary Pentium PC.

The following table summarizes the steps in a one-loop calculation and the distribution of tasks among the programs FeynArts, FormCalc, and LoopTools:

| Diagram generation | FeynArts |
|--------------------|----------|
| • Create the topologies |
| • Insert fields |
| • Apply the Feynman rules |
| • Paint the diagrams |
| Algebraic simplification | FormCalc |
| • Contract indices |
| • Calculate traces |
| • Reduce tensor integrals |
| • Introduce abbreviations |
| Numerical evaluation | LoopTools |
| • Convert Mathematica output to Fortran code |
| • Supply a driver program |
| • Implementation of the integrals |

The following sections describe the main functions of each program. Furthermore, the FormCalc package contains two sample calculations in the electroweak Standard Model, $ZZ \rightarrow ZZ$ [7] and $e^+e^- \rightarrow \bar{t}t$ [8], which demonstrate how the programs are used together.

## 2. FeynArts

FeynArts is a Mathematica package for the generation and visualization of Feynman diagrams and amplitudes [9]. It works in the three basic steps sketched in Fig. 1.

The first step is to create all different topologies for a given number of loops and external legs. For example, to create all one-loop topologies for a $1 \rightarrow 2$ process, the following call to CreateTopologies is used:

```plaintext
top = CreateTopologies[1, 1 -> 2]
```

In the second step, the actual particles in the model have to be distributed over the topologies in all allowed ways. E.g. the diagrams for $Z \rightarrow bb$ are produced with

```plaintext
ins = InsertFields[top, V[2] -> {F[4,{3}], -F[4,{3}]}]]
```
Find all distinct ways of connecting incoming and outgoing lines (CreateTopologies)

Topologies

Determine all allowed combinations of fields (InsertFields)

Draw the results (Paint)

Diagrams

Apply the Feynman rules (CreateFeynAmp)

Amplitudes

Fig. 1. Flowchart for the generation of Feynman amplitudes with FeynArts.

where $F[4, \{3\}]$ is the $b$-quark, $-F[4, \{3\}]$ its antiparticle, and $V[2]$ the $Z$ boson. The fields and their couplings are defined in a special file, the model file, which the user can supply or modify. Model files are currently provided for QED, the electroweak Standard Model, and QCD; a MSSM model file is in preparation.

The diagrams can be drawn with Paint[ins], depending on the options either on screen, or in a PostScript or LATEX file. Finally, the analytic expressions for the diagrams are obtained by

$$amp = \text{CreateFeynAmp}[\text{ins}]$$

3. FormCalc

The evaluation of the FeynArts output proceeds in two steps:

1. The symbolic expressions for the diagrams are simplified algebraically with FormCalc which returns the results in a form well suited for numerical evaluation.
2. The Mathematica expressions then need to be translated into a Fortran program. (The numerical evaluation could, in principle, be done in Mathematica directly, but this becomes very slow for large amplitudes.) The translation is done by the program NumPrep which is part of the FormCalc package. For compiling the generated code one needs a driver program (also in FormCalc), and the numerical implementations of the one-loop integrals in LoopTools.

The structure of FormCalc is simple: it prepares the symbolic expressions of the diagrams in an input file for FORM, runs FORM, and retrieves the results. This interaction is transparent to the user. FormCalc combines the speed of FORM with the powerful instruction set of Mathematica and the latter greatly facilitates further processing of the results. The following diagram shows schematically how FormCalc interacts with FORM:

```
Mathematica
PRO: user friendly
CON: slow on large expressions

FORM
PRO: extremely fast on polynomial expressions,
CON: not so user friendly

user interface

input file

MathLink

ProcessFile
```

The main function in FormCalc is OneLoop (the name is not strictly correct since it works also with tree graphs). It is used like this:

```
<< FormCalc'
amps = << myamps.m (* load some amplitudes *)
result = OneLoop[amps]
```

where it is assumed that the file myamps.m contains amplitudes generated by FeynArts. OneLoop uses dimensional regularization by default. To calculate in constrained differential renormalization (≡ dimensional reduction at the one-loop level), one has to put $\text{Dimension} = 4$ before invoking OneLoop. Note that OneLoop needs no declarations of the kinematics of the underlying process; it uses the information FeynArts hands down.

Even more comprehensive than OneLoop, the function ProcessFile can process entire files. It collects the diagrams into blocks such that index summations (e.g. over fermion generations) can later be carried out easily, i.e. only diagrams which are summed over the same indices are put in one block. ProcessFile is invoked e.g. as

```
ProcessFile["vertex.amp", "results/vertex"]
```
which reads the *FeynArts* amplitudes from `vertex.amp` and produces files of the form `results/vertexid.m`, where *id* is an identifier for a particular block.

`OneLoop` and `ProcessFile` return expressions where spinor chains, dot products of vectors, and Levi-Civita tensors contracted with vectors have been collected and abbreviated. A term in such an expression may look like

\[
C_{0i}[cc1, MW2, S, MW2, MZ2, MW2, MW2] \times \\
(\text{AbbSum12}(-8a2\times MW2 + 4a2\times MW2\times S2 - 2a2\times CW^2\times MW2\times S2 + 16a2\times CW^2\times S\times S2 + 4a2\times C2\times MW2\times SW^2) + \\
\text{Abb47}(-32a2\times CW^2\times MW2\times S2 + 8a2\times CW^2\times S2\times T + \\
8a2\times CW^2\times S2\times U) - \\
\text{AbbSum13}(-64a2\times CW^2\times MW2\times S2 + 16a2\times CW^2\times S2\times T + \\
16a2\times CW^2\times S2\times U))
\]

Here, the tensor coefficient function \( C_{1}(M_{W}^{2}, s, M_{W}^{2}, M_{Z}^{2}, M_{W}^{2}, M_{W}^{2}) \) is multiplied with a linear combination of abbreviations like `Abb47` or `AbbSum12` with certain coefficients. These coefficients contain the Mandelstam variables \( S, T, \) and \( U \) and some short-hands for parameters of the Standard Model, e.g. \( a2 = \alpha^2 \).

The abbreviations like `Abb47` or `AbbSum12` are introduced automatically and can significantly reduce the size of an amplitude. The definitions of the abbreviations can be retrieved by `Abbreviations[]` which returns a list of rules such that `result //. Abbreviations[]` gives the full, unabbreviated expression.

### 4. LoopTools

*LoopTools* supplies the actual numerical implementations of the one-loop functions needed for programs made from the *FormCalc* output. It is based on the reliable package *FF* [10] and provides in addition to the scalar integrals of *FF* also the tensor coefficients in the conventions of [11]. *LoopTools* offers three interfaces: Fortran, C++, and *Mathematica*, so most programming tastes should be served.

Using *LoopTools* functions in Fortran and C++ is very similar. In Fortran it is necessary to include the two files `tools.F` and `tools.h`, the latter one in every function or subroutine. In C++, `ctools.h` must be included once. Before using any *LoopTools* function, `bcaini` must be called and at the end of the calculation `bcaexi` may be called to obtain a summary of errors. It is of course possible to change parameters like the scale \( \mu \) from dimensional regularization; this is described in detail in the manual [12].
A very simple program would for instance be

```fortran
#include "tools.F"

program simple
#include "tools.h"
call bcaini
print *, B0(1000D0,50D0,80D0)
call bcaexi
end
```

```c++
#include "ctools.h"

main()
{
    bcaini();
    cout << B0(1000.,50.,80.) << "\n";
    bcaexi();
}
```

The Mathematica interface is even simpler to use:

```mathematica
In[1]:= Install["bca"]
In[2]:= B0[1000, 50, 80]
Out[2]= -4.40593 + 2.70414 I
```

5. Requirements and Availability

All three packages require Mathematica 2.2 or above; FormCalc needs in addition FORM, preferably version 2 or above; LoopTools needs a Fortran compiler, gcc/g++, and GNU make.

The packages should compile and run without change on any Unix platform. They are specifically known to work under DEC Unix, HP-UX, Linux, Solaris, and AIX. All three packages are open source and stand under the GNU library general public license. They are available from

http://www-itp.physik.uni-karlsruhe.de/feynarts
http://www-itp.physik.uni-karlsruhe.de/formcalc
http://www-itp.physik.uni-karlsruhe.de/looptools

Acknowledgements

This work has been supported by DFG under contract Ku 502/8–1.

REFERENCES

[1] R. Mertig, M. Böhm, and A. Denner, *Comp. Phys. Commun.* **64** (1991) 345.
[2] L. Brücher, *Nucl. Instr. Meth.* **A389** (1997) 327.
[3] F. del Aguila, A. Culatti, R. Muñoz Tápia, and M. Pérez-Victoria, *Nucl. Phys. B**537** (1999) 561.
[4] T. Hahn and M. Pérez-Victoria, Comp. Phys. Commun. 118 (1999) 153.

[5] J.A.M. Vermaseren, Symbolic Manipulation with FORM, CAN, Amsterdam (1991).

[6] A. Denner and T. Hahn, Nucl. Phys. B525 (1998) 27.

[7] A. Denner, S. Dittmaier, and T. Hahn, Phys. Rev. D56 (1997) 117.

[8] W. Beenakker, S.C. van der Marck, and W. Hollik, Nucl. Phys. B386 (1991) 24.

[9] J. Kühlbeck, M. Böhm, and A. Denner, Comp. Phys. Commun. 60 (1991) 165; T. Hahn, FeynArts 2.2 user’s guide, available at http://www-itp.physik.uni-karlsruhe.de/feynarts.

[10] G.J. van Oldenborgh and J.A.M. Vermaseren, Z. Phys. C46 (1990) 425.

[11] A. Denner, Fortschr. Phys. 41 (1993) 307.

[12] T. Hahn, FormCalc and LoopTools user’s guide, available at http://www-itp.physik.uni-karlsruhe.de/formcalc.