2005 International Linear Collider Workshop - Stanford, U.S.A.

New Developments in FormCalc 4.1

Thomas Hahn
Max-Planck-Institut für Physik, Föhringer Ring 6, D–80805 Munich, Germany

FormCalc is a matrix-element generator that turns FeynArts amplitudes up to one loop into a Fortran code for computing the squared matrix element. The generated code can be run with FormCalc's own driver programs or used with other 'frontends', e.g. Monte Carlos. Major new or enhanced features in Version 4.1 are: treatment of external fermions, phase-space integration, code-generation functions, extensions for the MSSM, the HadCalc frontend.

arXiv:hep-ph/0506201v1 21 Jun 2005

1. Introduction

FormCalc [1] is a package for the calculation of Feynman amplitudes based on Mathematica and FORM [2]. Amplitudes generated by FeynArts [3] are first simplified analytically and then converted to a self-contained Fortran code for the computation of the squared matrix element. Currently, diagrams up to one loop can be simplified, and kinematics are supplied for $1 \to 2$, $2 \to 2$, and $2 \to 3$ processes.

The present article describes the features added or enhanced in version 4.1. They fall into several categories:

- Weyl–van der Waerden (WvdW) spinor formalism for external fermion lines,
- Multidimensional phase-space integration with Cuba 1.2,
- Public functions for writing out Mathematica expressions as optimized Fortran code,
- FeynHiggs interface,
- Non-minimal flavour violation (NMFV) for the MSSM,
- Various useful scripts,
- The HadCalc frontend, by M. Rauch, for hadronic calculations.

2. Weyl–van der Waerden spinor formalism

Amplitudes involving external fermions have the form $\mathcal{M} = \sum_{i=1}^{n} c_i F_i$, where the F_i are (products of) fermion chains. The textbook recipe is to compute probabilities, such as $|\mathcal{M}|^2 = \sum_{i,j=1}^{n} c_i^* c_j F_i^* F_j$, and evaluate the $F_i^* F_j$ by standard trace techniques: $|\langle u| \Gamma |v\rangle|^2 = \langle u| \Gamma |v\rangle \langle v| \bar{\Gamma} |u\rangle = \mathrm{Tr}\big(\Gamma |v\rangle \langle v| \bar{\Gamma} |u\rangle \langle u|\big)$.

The problem with this approach is that instead of n of the F_i one needs to compute n^2 of the $F_i^* F_j$. Since essentially $n \sim$ (number of vectors)!, this quickly becomes a limiting factor in problems involving many vectors, e.g. in multi-particle final states or polarization effects.

The solution is of course to compute the amplitude $\mathcal{M}$ directly and this is done most conveniently in the WvdW formalism [4]. The implementation of this technique in an automated program has been outlined in [5].

The `FermionChains` option of `CalcFeynAmp` determines how fermion chains are returned: `Weyl`, the default, selects Weyl chains. `Chiral` and `VA` select Dirac chains in the chiral (ω_+/ω_-) and vector/axial-vector ($1/\gamma_5$) decomposition, respectively. The Weyl chains do not need to be further evaluated with `HelicityME`, which applies the trace technique.

The WvdW method has other advantages, too: Polarization does not 'cost' extra in terms of CPU time, that is, one gets the spin physics for free. Whereas with the trace technique the formulas become significantly more bloated when polarization is taken into account, in the WvdW formalism one actually needs to sum up the polarized amplitudes to

get the unpolarized result. There is also better numerical stability because components of k^μ are arranged as 'large' and 'small' matrix entries, viz.

$$\sigma_\mu k^\mu = \begin{pmatrix} k_0 + k_3 & k_1 - \mathrm{i}k_2 \\ k_1 + \mathrm{i}k_2 & k_0 - k_3 \end{pmatrix}. \tag{1}$$

Cancellations of the form $k_0 - k = \sqrt{k^2 + m^2} - \sqrt{k^2}$ for $m \ll k$ can be avoided (e.g. by observing that $k_0 - k = m^2/(k_0 + k)$ for on-shell k) and hence mass effects are treated more accurately.

3. Multidimensional integration with Cuba

FormCalc uses the Cuba library [6] for numerical integration of multidimensional phase-spaces. Cuba provides four integration routines: Vegas, Suave, Divonne, and Cuhre. All four have a very similar invocation and can thus be interchanged easily, e.g. for comparison. The flexibility of a general-purpose method is particularly useful in the setting of automatically generated code. The following table gives an overview of the features of the Cuba routines; specific details on the implementation and the actual usage of the Cuba routines are provided in [6] and shall not be repeated here.

Routine	Basic method	Type	Variance reduction
Vegas	Sobol sample	quasi-Monte Carlo	importance sampling
	or Mersenne Twister sample	pseudo-Monte Carlo	
Suave	Sobol sample	quasi-Monte Carlo	globally adaptive subdivision
	or Mersenne Twister sample	pseudo-Monte Carlo	+ importance sampling
Divonne	Korobov sample	lattice method	stratified sampling,
	or Sobol sample	quasi-Monte Carlo	aided by methods from
	or Mersenne Twister sample	pseudo-Monte Carlo	numerical optimization
	or cubature rules	deterministic	
Cuhre	cubature rules	deterministic	globally adaptive subdivision

Apart from some important bug-fixes, Cuba Version 1.2 includes the following new features:

- Pseudo-random sampling using the Mersenne Twister generator has been added to all Monte Carlo algorithms. This is useful not only for comparison, but also because in high dimensions the numerator in the convergence rate for quasi-random samples, $\mathcal{O}(\log^{d-1} n/n)$, becomes noticeable.

- Vegas can memorize its internal grid for subsequent invocations, to speed up integration on similar integrands.

- Vegas can save its internal state in a file such that the calculation can be resumed e.g. after a crash.

- There exists a one-stop invocation for the Cuba routines,

  ```
  subroutine Cuba(ndim, integrand, result, error)
  ```

  with which the number of parameters that need to be passed is reduced to a minimum. (Note that this particularly simple form is special to FormCalc, and that Cuba's original definition is slightly more involved.)

- A partition viewer has been added. It allows to view the partitioning of the integration region performed by Suave, Divonne, and Cuhre (Vegas does not tessellate the integration region). To use the partition viewer, set the verbosity level to 3 and pipe the output of your program through the `partview` program, as in

  ```
  myprogram | partview 1 2 1 3
  ```

  Each pair of numbers on the command-line refers to a hyperplane to display, i.e. in the above example the 1–2 and 1–3 hyperplanes are shown. See Fig. 1 for a screenshot.

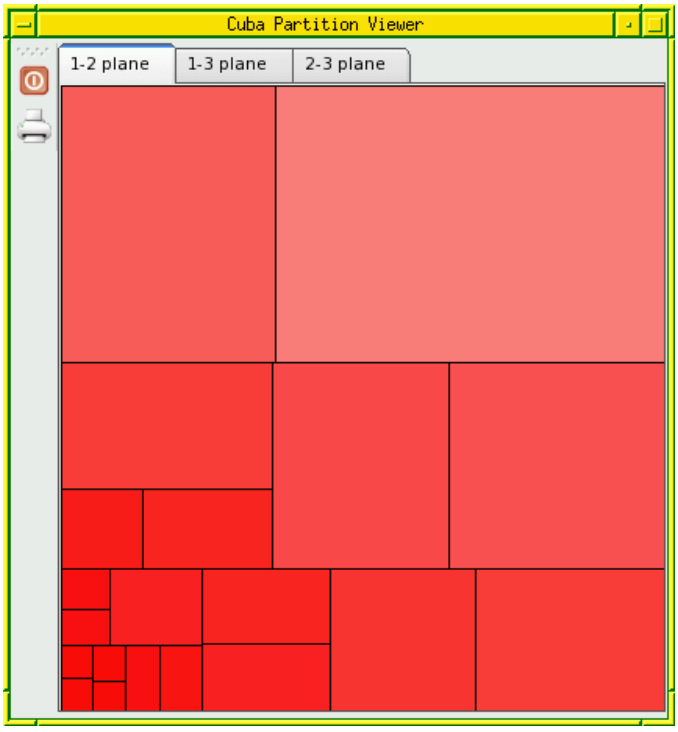
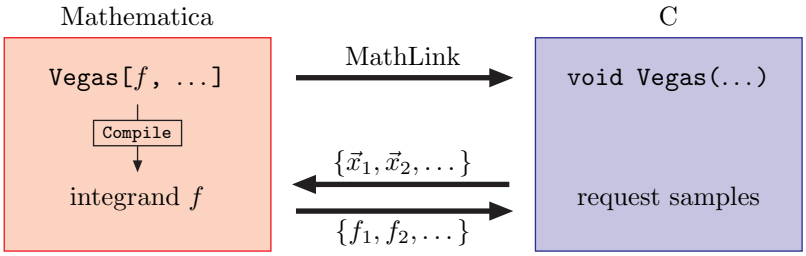


Figure 1: A screenshot of the Cuba Partition Viewer.

Cuba possesses also a Mathematica interface. The MathLink executables are loaded e.g. with `Install["Vegas"]` and make the Cuba routines available in a way almost like `NIntegrate`.



The integrand stays in Mathematica at all times (see figure above), which means that one can integrate functions which are not simple to implement in C or Fortran, e.g. `Cuhre[Zeta[x y], {x,.2,.3}, {y,.4,.5}]`.

4. Public code-generation functions

FormCalc's code-generation functions, hitherto used only internally, have been reorganized for greater user-friendliness and are now publicly available. They can be used to write out an arbitrary Mathematica expression as optimized Fortran code. The basic procedure is very simple:

1. *handle* `= OpenFortran["`*file.F*`"]`
   opens *file.F* as a Fortran file for writing,
2. `WriteExpr[`*handle*`, {`*var* `->` *expr*`, ...}]`
   writes out Fortran code to calculate *expr* and store the result in *var*,
3. `Close[`*handle*`]`
   closes the file again.

The code generation is fairly sophisticated and goes well beyond merely applying Mathematica's `FortranForm`. The generated code is optimized, e.g. common subexpressions are pulled out and computed in temporary variables. Expressions too large for Fortran are split into parts, as in

```
var = part1
var = var + part2
...
```

If the expression is too large even to be sensibly evaluated in one file, the `FileSplit` function can distribute it on several files and optionally write out a master subroutine which calls the individual parts.

To further automate the code generation, such that the resulting code needs few or no changes by hand, many ancillary functions are available, e.g. `CommonDecl` writes out common-block declarations for a given list of variables.

5. FeynHiggs interface and NMFV

FormCalc's initialization code for the MSSM, `mssm_ini.F`, now contains an interface to FeynHiggs [7, 8] for the computation of the Higgs masses. Since in particular the light Higgs-boson mass receives significant radiative corrections, its precise determination is phenomenologically important. From the user perspective, a preprocessor flag governs the choice of Higgs masses:

- `#define HIGGS_MASSES TREE`
  use the tree-level Higgs masses,
- `#define HIGGS_MASSES SIMPLE`
  use a simple one-loop formula,
- `#define HIGGS_MASSES FEYNHIGGS`
  invoke FeynHiggs 2.2 [8] – this is the most precise determination,
- `HIGGS_MASSES` undefined
  use the FeynHiggsFast approximation [9] – quite precise, but valid only for real SUSY parameters.

Also for the MSSM, non-minimal flavour violation (NMFV) can be enabled. In the usual, minimal, setup, there is no flavour-violation beyond the Standard Model's CKM effects, i.e. left–right mixing is independent for each squark flavour. In contrast, NMFV means full 6×6 mixing among both up-type squarks ($\tilde{u}_1, \tilde{u}_2, \tilde{c}_1, \tilde{c}_2, \tilde{t}_1, \tilde{t}_2$) and down-type squarks ($\tilde{d}_1, \tilde{d}_2, \tilde{s}_1, \tilde{s}_2, \tilde{b}_1, \tilde{b}_2$), see e.g. [10].

On the technical side, the diagrams have to be generated using the `FVMSSM.mod` model file in FeynArts, and in the numerical evaluation the preprocessor flag `FLAVOUR_VIOLATION` has to be defined. Once this flag is enabled, the 6×6 mass matrix is accessible as `LambdaSf` and in addition to the usual squark masses and mixing matrices, `MSf` and `USf`, there exist `MASf` and `UASf` containing the corresponding NMFV quantities.

6. Shell scripts

FormCalc 4.1 includes a few useful shell scripts:

- `sfx` packs all source files in the directory it is invoked in into a mail-safe self-extracting archive. The archived code in particular does not need FormCalc to compile or run.
- `turnoff` switches off (and on) the evaluation of certain parts of the amplitude, which is a handy thing for testing. For example, "`turnoff box`" turns off the boxes (actually, all parts of the amplitude with 'box' in their name). `turnoff` without arguments then restores all modules.
- `pnuglot` produces a high-quality plot in Encapsulated PostScript format from a data file in just one line.

- `submit` automatically distributes a parameter scan on a cluster. The available machines first have to be declared in a file `.submitrc`, e.g.

```
# Optional: 'nice' to start jobs with
nice 10
# Pentium 4
pcl301
pcl305
# Dual Xeon
pcl247b   2
pcl319a   2
...
```

  After that the command line, typically something like "`run uuuu 500,1000`", simply has to be prefixed by `submit`, i.e. "`submit run uuuu 500,1000`".

7. The HadCalc frontend

HadCalc is a new frontend for FormCalc, i.e. it uses the generated Fortran code with a custom set of driver programs. It was written by Michael Rauch who also maintains it independently from FormCalc.

HadCalc automates the calculation of hadronic cross-sections. It automatically performs the convolution with PDFs and allows various cuts to be applied. Cross-sections can be computed either fully integrated or differential in invariant mass, rapidity, or transverse momentum. HadCalc operates either in batch mode, like FormCalc, or interactively, which allows the user e.g. to play with parameters.

The program is not (yet) public. It can currently be obtained on request from ⟨mrauch@mppmu.mpg.de⟩.

8. Summary

FormCalc 4.1 is the current release of the FormCalc package. It has many new features enhancing performance and user-friendliness. The new public Fortran code-generation functions should be interesting also to people who do not use FormCalc to compute Feynman amplitudes. The package is available as open source and stands under the GNU Lesser General Public License (LGPL). It can be obtained from the Web site `http://www.feynarts.de/formcalc`.

References

[1] T. Hahn, M. Pérez-Victoria, *Comp. Phys. Commun.* **118** (1999) 153 [hep-ph/9807565].
[2] J.A.M. Vermaseren, math-ph/0010025. See also http://www.nikhef.nl/∼form.
[3] T. Hahn, *Comp. Phys. Commun.* **140** (2001) 418 [hep-ph/0012260].
[4] S. Dittmaier, *Phys. Rev.* **D59** (1999) 016007 [hep-ph/9805445].
[5] T. Hahn, *Nucl. Phys. Proc. Suppl.* **116** (2003) 636 [hep-ph/0210220].
[6] T. Hahn, *Comp. Phys. Commun.* **168** (2005) 78 [hep-ph/0404043].
[7] S. Heinemeyer, W. Hollik, G. Weiglein, *Comp. Phys. Commun.* **124** (2000) 76 [hep-ph/9812320].
The FeynHiggs Web site is at `http://www.feynhiggs.de`.
[8] T. Hahn, S. Heinemeyer, W. Hollik, G. Weiglein, these proceedings.
M. Frank, T. Hahn, S. Heinemeyer, W. Hollik, G. Weiglein, in preparation.
[9] S. Heinemeyer, W. Hollik, G. Weiglein, hep-ph/0002213.
[10] S. Heinemeyer, W. Hollik, S. Peñaranda, these proceedings [hep-ph/0506104].