Adding the $\mathcal{O}(\alpha_t^2)$ corrections to FeynHiggs

Thomas Hahn

Max-Planck-Institut für Physik
München

MSSM Higgs-mass corrections

- Higgs mass predicted in MSSM but receives substantial radiative corrections, e.g. $M_h \leqslant M_Z$ at tree level.
- FeynHiggs is a public tool for computing the Higgs mass corrections (and much more).
- Leading two-loop contributions: $\mathcal{O}(\alpha_s\alpha_t)$, $\mathcal{O}(\alpha_t^2)$.
- Available in rMSSM in FH for long (eff. potential method).

  Degrassi, Slavich, et al. 2001, 2002, 2003
- $\mathcal{O}(\alpha_s\alpha_t)$ in the cMSSM available (diagrammatic calc.).

  Rzehak et al. 2007

$\mathcal{O}(\alpha_t^2)$ corrections in FeynHiggs

- **Two-loop Higgs-mass corrections ($\approx$ 5 GeV) important for precise prediction in MSSM.**
- $\mathcal{O}(\alpha_t^2)$**: Diagrammatic 2L calculation in full cMSSM.**
  Specific approximations ($p^2 = 0$, gaugeless).
  Nontrivial renormalization.

Hollik, Paßehr 2014

$\mathcal{O}(\alpha_t^2)$ corrections in FeynHiggs

- **Two-loop Higgs-mass corrections ($\approx$ 5 GeV) important for precise prediction in MSSM.**
- $\mathcal{O}(\alpha_t^2)$: **Diagrammatic 2L calculation in full cMSSM.**
  Specific approximations ($p^2 = 0$, gaugeless).
  Nontrivial renormalization.

  Hollik, Paßehr 2014

This talk: Show 'How' (not 'What') of this calculation.

Work in collaboration with Sebastian Paßehr [arXiv:1508.00562].

Shopping List

Need to compute:

① **Unrenormalized 2L self-energies**
$\Sigma^{(2)}_{hh}$, $\Sigma^{(2)}_{hH}$, $\Sigma^{(2)}_{hA}$, $\Sigma^{(2)}_{HH}$, $\Sigma^{(2)}_{HA}$, $\Sigma^{(2)}_{AA}$, $\Sigma^{(2)}_{H^+H^-}$
at $p^2 = 0$ at $\mathcal{O}(\alpha_t^2)$.

② **1L diagrams with insertions of 1L counterterms.**

③ **2L counterterms** for ①.

④ **2L tadpoles** $T_h^{(2)}$, $T_H^{(2)}$, $T_A^{(2)}$ at $\mathcal{O}(\alpha_t^2)$ appearing in ③.

Calculational Setup

- For tree level and one-loop **many packages available.**
- Packages like Madloop, GoSam, etc. try to be **comprehensive** = do all parts of the calculation.
- This means **higher automation** (good!).
  For QCD often ok since e.g. renormalization simple.
- But: controlled by (e.g.) **parameter cards,** not easy to use beyond intended purpose.
- Calculations often have some ‘speciality’ that requires **extra programming** and/or **extra packages.**
- May want to switch to other packages for **cross-checks.**

Template for Calculations

- Starting point: several Mathematica 8 *(sic)* notebooks with parallel instructions poured all over, duplicate code (e.g. gaugeless limit implemented multiply), etc.
- Reorganization of entire procedure:
  - Break calculation into **several steps.**
  - Implement each step as **independent program** (invoked from command line).
  - In lieu of 'in vivo' debugging **keep detailed logs.**
  - Coordinate everything through a **makefile.**
- Outcome: **Template for 2L calculation** in **nontrivial model** with **nontrivial renormalization** with **optimized output.**

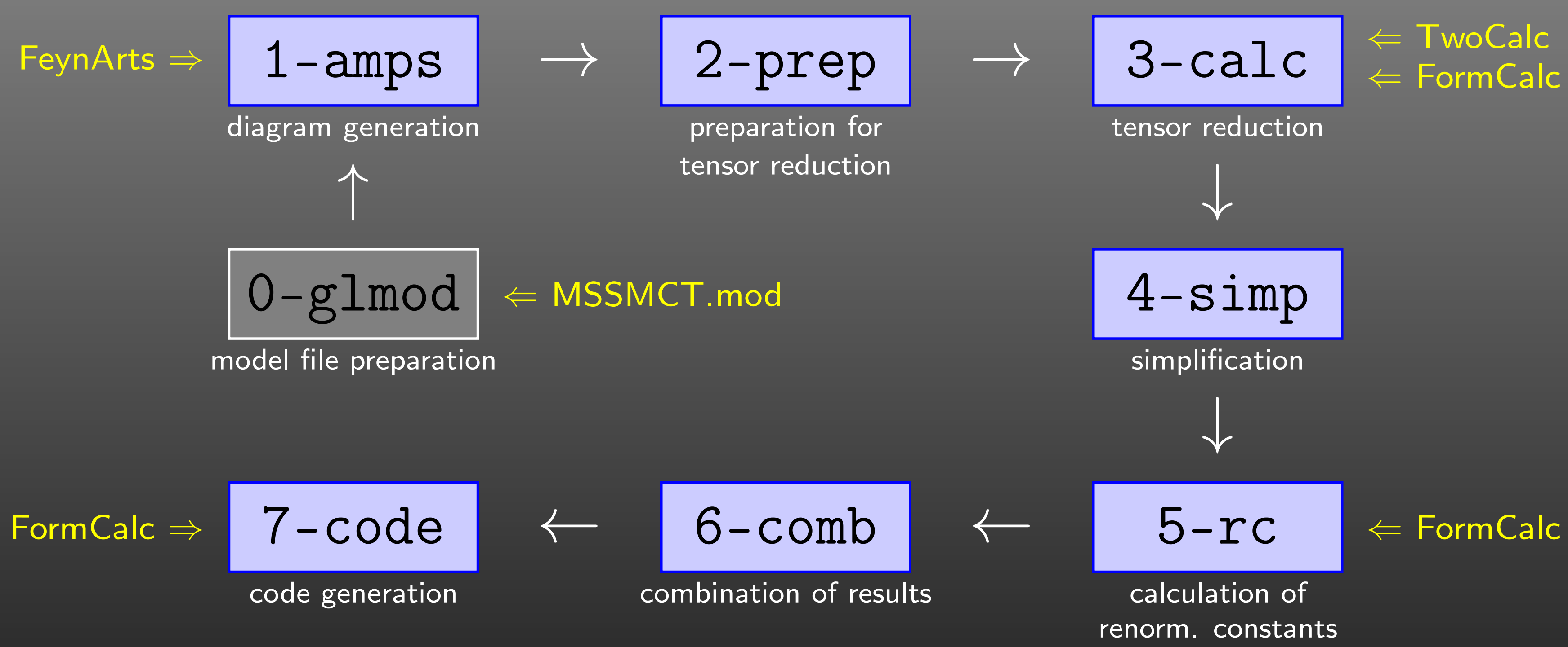
Wheels we don't reinvent

Will use external packages

- **FeynArts** (for diagram generation),
  Hahn 2001–2015
- **MSSMCT.mod** (MSSM model file with 1L counterterms),
  Schappacher et al. 2014
- **FormCalc** (for 1L tensor reduction, code generation),
  Hahn 1996–2015
- **TwoCalc** (for 2L tensor reduction).
  Weiglein et al. 1992, 1994

Steps of the Calculation

Calculation split into 7 (8) steps:



Script Structure

- Shell scripts (`/bin/sh`), run from command line as e.g.
  `./1-amps arg1 arg2`
- `arg1` = `h0h0`, `h0HH`, `h0A0`, `HHHH`, `HHA0`, `A0A0`, `HmHp` (self-energies),
  `h0`, `HH`, `A0` (tadpoles).
- `arg2` = `0` for virtual 2L diagrams,
  `1` for 1L diagrams with 1L counterterms.
- Inputs/outputs defined in first few lines, e.g.

```
in=m/$1/2-prep.$2
out=m/$1/3-calc.$2
```

- Symbolic output + log files go to '`m`' subdirectory.
  Log file = Output file + `.log.gz`
- Fortran code goes to '`f`' subdirectory.

Scripting Mathematica

Efficient batch processing with Mathematica:

Put everything into a script, using sh's Here documents:

```
#! /bin/sh ................. Shell Magic
math << \_EOF_ ............ start Here document (note the \)
  << FeynArts`
  << FormCalc`
  top = CreateTopologies[...];
  ...
_EOF_ ..................... end Here document
```

Everything between "<< *tag*" and "*tag*" goes to Mathematica as if it were typed from the keyboard.

Note the "\\" before *tag*, it makes the shell pass everything literally to Mathematica, without shell substitutions.

Scripting Mathematica

- Everything contained in **one compact shell script,** even if it involves several Mathematica sessions.
- Can combine with arbitrary shell programming, e.g. can use **command-line arguments** efficiently:

```
#! /bin/sh
math -run "arg1=$1" -run "arg2=$2" ... << \END
  ...
END
```

- Can easily be **run in the background,** or combined with utilities such as **make.**

Debugging hint: `-x` **flag** makes shell echo every statement,

```
#! /bin/sh -x
```

Step 0: Gaugeless Limit

Gaugeless approximation:

① Set gauge couplings $g, g' = 0 \Rightarrow M_W, M_Z = 0$.

② Keep finite weak mixing angle.

③ Keep $\frac{\delta M_W^2}{M_W^2}$ and $\frac{\delta M_Z^2}{M_Z^2}$ finite.

Must set $m_b = 0$ so that $\mathcal{O}(\alpha_t^2)$ corrections form supersymmetric and gauge-invariant subset.

Most efficient to modify Feynman rules (not ③, though):

- Load `MSSMCT.mod` model file.
- Modify couplings, remove zero ones.
- Write out `MSSMCTgl.mod` model file.

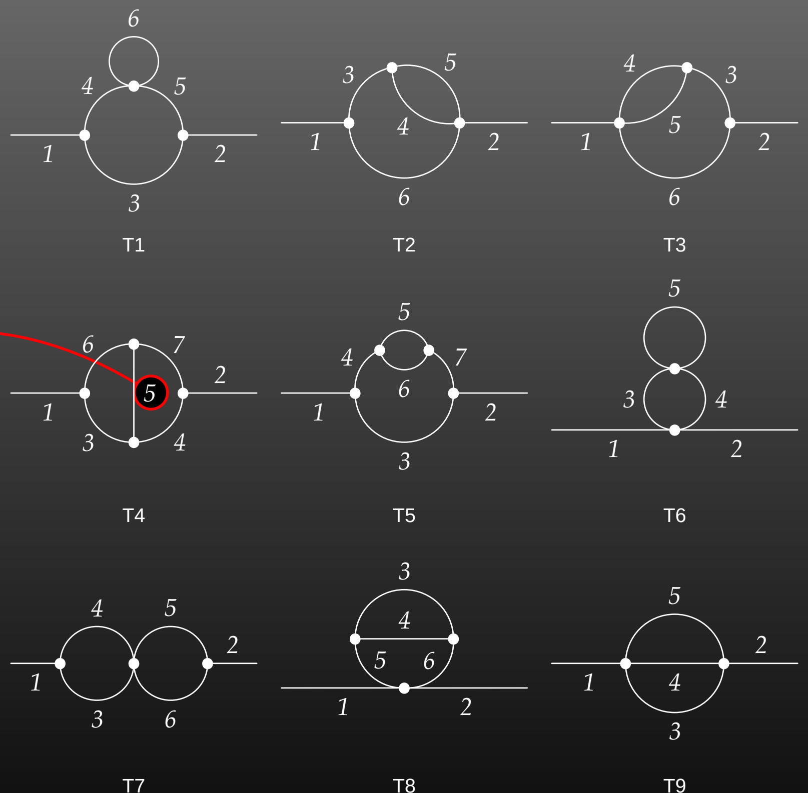
Step 1: Diagram Generation

- **Generate 2L virtual and 1L+counterterm diagrams using wrappers for FeynArts functions.**

Simple diagram selection functions, e.g.

```
sel[0][S[_] -> S[_]] = {
  t[3] && htb[6],
  t[3] && tb[6],
  t[3] && tb[6],
  t[3] && t[4] && htb[5],
  t[3] && htb[5|6],
  t[3] && htb[5],
  t[3] && t[5],
  t[5] && ht[3|4],
  t[3|4|5] && ht[3|4|5] }
```

one of $h_i, \tilde{\chi}, t, \tilde{t}, b, \tilde{b}$



Step 2: Preparation for Tensor Reduction

- Take $p^2 \to 0$ limit.
- Simplify ubiquitous sfermion mixing matrices U_{ij}, mostly by exploiting unitarity ($\sim$ 50% size reduction).

Efficiently Exploit Unitarity in Mathematica

Unitarity of 2 x 2 matrix: $UU^\dagger = U^\dagger U = \mathbb{1}$, **i.e.**

$$
\begin{aligned}
U_{11}U_{11}^* + U_{12}U_{12}^* &= 1, & U_{11}U_{21}^* + U_{12}U_{22}^* &= 0, \\
U_{21}U_{21}^* + U_{22}U_{22}^* &= 1, & U_{21}U_{11}^* + U_{22}U_{12}^* &= 0, \\
U_{11}U_{11}^* + U_{21}U_{21}^* &= 1, & U_{11}U_{12}^* + U_{21}U_{22}^* &= 0, \\
U_{12}U_{12}^* + U_{22}U_{22}^* &= 1, & U_{12}U_{11}^* + U_{22}U_{21}^* &= 0.
\end{aligned}
$$

Problem: `Simplify` **will rarely arrange the U's in just the way that these rules can be applied directly.**

Solution: Introduce auxiliary symbols which immediately deliver the r.h.s. once `Simplify` **considers the l.h.s., i.e. increase the 'incentive' for** `Simplify` **to use the r.h.s.**

But: Upvalues work only one level deep.

Efficiently Exploit Unitarity in Mathematica

Introduce

$\texttt{USf}[1, j]\ \texttt{USfC}[1, j] \to \texttt{UCSf}[1, j]$,

$\texttt{USf}[2, j]\ \texttt{USfC}[2, j] \to \texttt{UCSf}[2, j]$,

$\texttt{USf}[1, j]\ \texttt{USfC}[2, j] \to \texttt{UCSf}[3, j]$, **+ ditto for 1st index**

and formulate unitarity for the `UCSf`**:**

```
UCSf[2,1] = UCSf[1,2];    UCSf[3,2] = -UCSf[3,1];
UCSf[2,2] = UCSf[1,1];    UCSfC[3,2] = -UCSfC[3,1];
                          UCSf[2,3] = -UCSf[1,3];
...                       UCSfC[2,3] = -UCSfC[1,3];
```

Step 3: Tensor Reduction

- Relatively straightforward application of TwoCalc and FormCalc for tensor reduction.
- Observe: Need two Mathematica sessions since TwoCalc and FormCalc cannot be loaded into one session, easily accomodated in shell script.

Step 4: Simplification

- Tensor reduction traditionally increases # of terms most.
- Step 4 reduces size before combination of results.
- Empirical simplification recipe.
- **'DiagMark' trick** (D. Stöckinger):
  - Introduce `DiagMark[`m_i`]` where m_i = masses in loop in FeynArts output.
  - Few simplifications can be made between parts with different `DiagMark` $\Rightarrow$ Can apply simplification as

    ```
    Collect[amp, _DiagMark, simpfunc]
    ```

  - Much faster.

Step 5: Calculation of Renormalization Constants

- Compute 1L renormalization constants (RC) with FormCalc.
- Substitute explicit mass dependence in

$$\texttt{dMVsq1} \to \texttt{MV2}\,\texttt{dMVsq1MV2} \qquad (V = W, Z)$$

  such that gaugeless limit can be taken safely.
- Expand in ε, collect powers for easier handling later, e.g.

```
{ dMf1[3,3] -> RC[-1, dMf1[-1,3,3]] +
               RC[0, dMf1[0,3,3]],          - expansion
  {dMf1[-1,3,3] -> ...,                     - actual expressions for ε-coeffs
   dMf1[0,3,3] -> ...} }
```

Step 6: Combination of Results

- Expand amplitude in ε (similar as RC).
- Insert RCs.
- Add genuine 2L counterterms (hand-coded).
- Pick only ε^0 term (unless debug flag set).
- Perform final simplification.

Step 7: Code Generation

- Introduce abbreviations to shorten code.
- Write out Fortran code using FormCalc's code-generation functions.
- Add static code which computes e.g. the necessary parameters for the generated code.
- Total final code size: 350 kBytes.

Summary

As a by-product of the implementation of the $\mathcal{O}(\alpha_t^2)$ corrections in FeynHiggs we now have a

- Suite of scripts which can be used as a
- Template for similar calculations,
  - Two-loop,
  - Nontrivial model (MSSM),
  - Nontrivial renormalization,
  - Specific approximations (gaugeless, $p^2 = 0$),
  - Optimized output.

Code is included in public release of FeynHiggs 2.11+ in the `gen/tlsp` directory.

More details in arXiv:1508.00562.