TwoCalc:

self energies at two-loop level

Heidi Rzehak
Max-Planck-Institut für Physik, München

TwoCalc in general

TwoCalc:

- analytical evaluation of two-loop self energy diagrams (in terms of scalar integrals)
- Mathematica program
- uses FeynArts-Input (with modification)
- uses some FeynCalc-routines
- output: scalar parts of the self energies
- numerical evaluation: left as a problem for the user

What is TwoCalc good for?

Knowledge of two-loop self energies needed for:

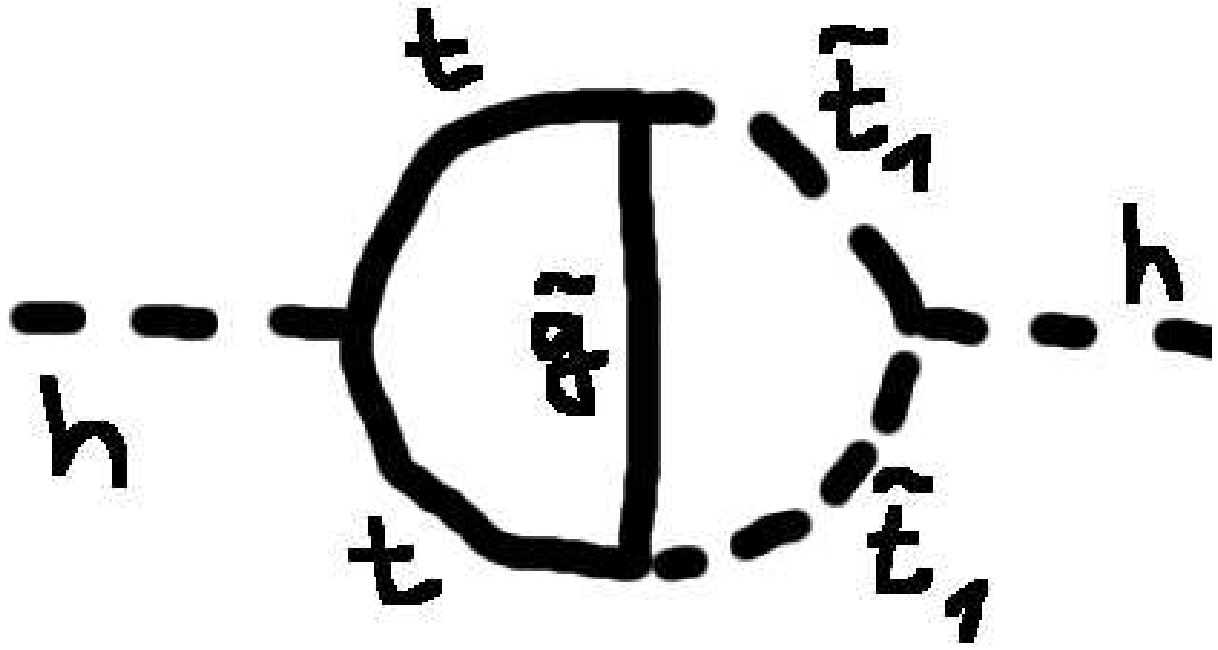
Precision calculations:

e.g. • W-boson mass, Z-boson mass, ρ parameter, Δr

• mass of the lightest Higgs boson in the MSSM

FeynArts-Input

Example: this diagram shall be evaluated:



FeynArts-Input

FeynArts program:

```
Get["FeynArts.m"];

SetOptions[InsertFields,
           Model -> MSSMQCD,
           InsertionLevel->{Particles},
           ExcludeParticles->{S[1],S[2],S[3],
            S[4],S[5],S[6],S[11,___],S[12,___],
            S[14,___],V[1],V[2],V[3],V[5],U[_],
            F[4,___],F[11,___],F[12,___]},
           Restrictions->{NoGeneration1,
                          NoGeneration2}];

SetOptions[Paint, PaintLevel -> {Particles},
           ColumnsXRows -> {4, 5}];
```

choice of the model file

level specification

exclusions

FeynArts-Input

two loops

self energy

```
tt = CreateTopologies[2, 1->1,
          ExcludeTopologies->{Tadpoles}];
```

choose process:

```
selfhh2loop = InsertFields[tt, S[1]->S[1]];
```

choose diagram:

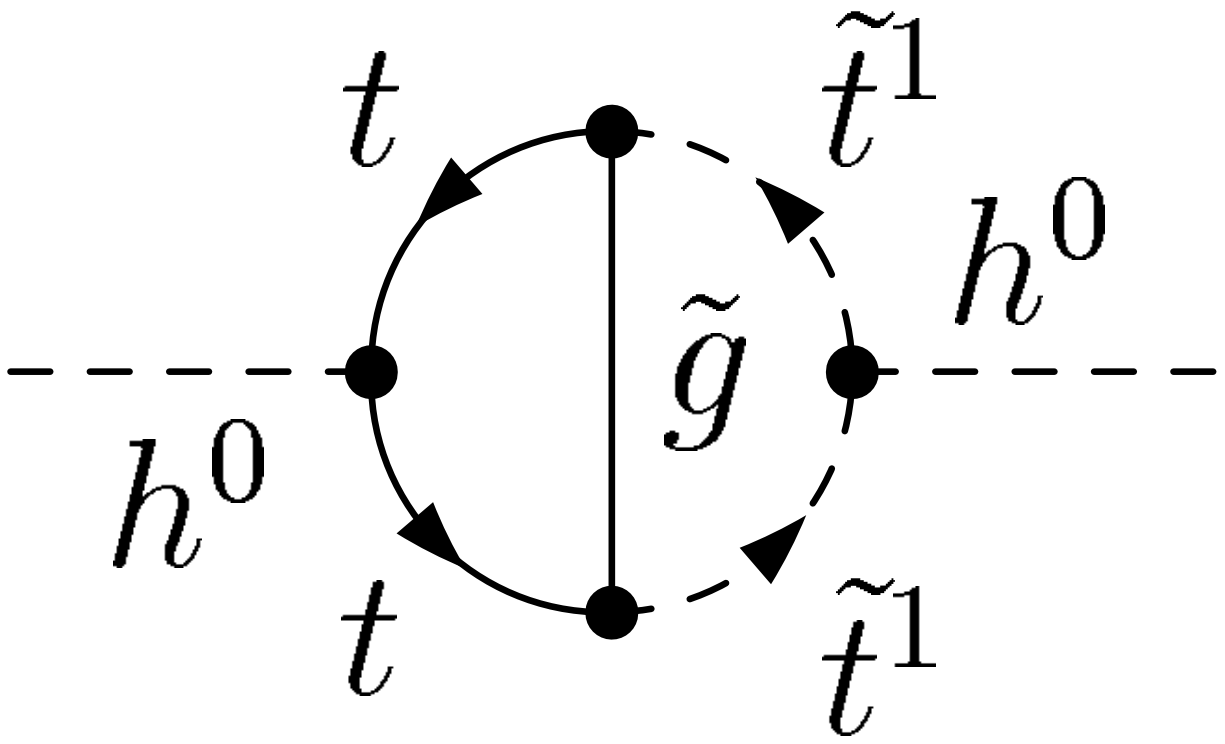
```
selfhh2loop = Discard[selfhh2loop, 1...32,
                                   34...88];
```

painting the chosen diagrams:

```
Paint[selfhh2loop,
      DisplayFunction ->
      (Display["pic/2ldiag.ps", #]&)];
```

FeynArts-Input

Just checking: It is the right diagram:



FeynArts-Input

Further FeynArts program:

Generation of the FeynArts amplitude:

```
ampself = CreateFeynAmp[selfhh2loop];
```

Output:

```
FeynAmpList[Model ->...,...,
 Process ->
  {{S[1], FourMomentum[Incoming, 1], Mh0}} ->
  {{S[1], FourMomentum[Outgoing, 1], Mh0}}]
 [FeynAmp[GraphID[...],
          Integral[FourMomentum[Internal, 1],
                   FourMomentum[Internal, 2]],
          FeynAmpDenominator[..]MatrixTrace[..]
          ...SumOver[Index[Colour, 2], 3]...]]
```

process

single diagram

summation over colour indices

FeynArts-Input

TwoCalc: no evaluation of sums over colour indices

SUNSimplify.m:

- takes as input: FeynArts amplitude
- evaluates sums over colour indices
- gives as output: FeynArts amplitude

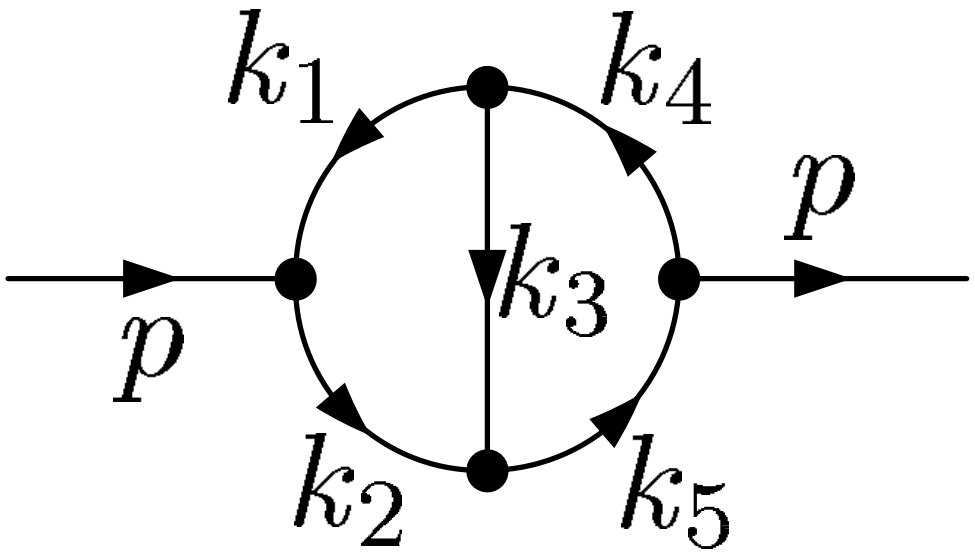
```
Get["SUNSimplify.m"];

ampself = SUNSimplify[ampself];
```

FeynArts-Input

TwoCalc: uses different conventions

simplesubst.m:

- takes as input: FeynArts amplitude
- redefines momenta:



$$k_1 = q_1$$
$$k_2 = q_1 + p$$
$$k_3 = q_2 - q_1$$
$$k_4 = q_2$$
$$k_5 = q_2 + p$$

p =FourMomentum[Incoming/Outgoing, 1]: external momenta

q_i =FourMomentum[Internal, i]: loop momenta

FeynArts-Input

TwoCalc: uses different conventions

simplesubst.m:

- takes as input: FeynArts amplitude
- redefines momenta
- gives as output: TwoCalc compatible amplitude

```
Get["simplesubst.m"];

ampself = SimpleSubst[ampself];
```

FeynArts-Input

FeynArts amplitude as input for TwoCalc:

```
FeynAmpList[Model ->...,...,
  Process -> {{S[1],p1,Mh0}} -> {{S[1],p1,Mh0}}],
  [FeynAmp[GraphID[...]
           k1,k2,k3,k4,k5,p,
           ...*(-1 + SUNN^2)*
           DiracTrace[(MT + DiracSlash[k2]). ...]*
           FeynAmpDenominator[
            PropagatorDenominator[k1, MT],
            PropagatorDenominator[k2, MT],
            PropagatorDenominator[k3, MGl],
            PropagatorDenominator[k4, MSf[1,3,3]],
            PropagatorDenominator[k5, MSf[1,3,3]]]
           *...]]
```

Running TwoCalc

First step:

```
Get["TwoLoop.m"];
```

Second step:

```
SetOptions[TwoLoop, Dimension->$D,...]
```

should be set
$D: number of dimensions

Further options for TwoLoop:

- choice of regularisation scheme:
  - dimensional regularisation (default)
  - dimensional reduction: `DimReduction -> True`
- choice of function for collecting scalar integrals
- specification of initial substitutions

Running TwoCalc

Third step:

Get the FeynArts amplitude:

```
amp = Get["amp/ampself.amp"];
```

Fourth step:

Do the calculation:

```
self=TwoLoopSum[amp,SelfEnergyPart->8,...];
```

further options:

- graph selection
- collection possibilities

Running `TwoCalc`

Meaning of `SelfEnergyPart`:

`TwoCalc` calculates only scalar parts of self energies:

self energies must be decomposed:

vector bosons: transversal and longitudinal parts

fermions: scalar, vector and axial vector parts

`SelfEnergyPart` specifies:

what part of which self energy should be calculated

Running TwoCalc

Third step:

Get the FeynArts amplitude:

```
amp = Get["amp/ampself.amp"];
```

Fourth step:

Do the calculation:

```
self=TwoLoopSum[amp,SelfEnergyPart->8,...];
```

(scalar) part of a
self energy of
scalar bosons

TwoCalc Output

Output:

```
T[Df[k1,MT]*Df[k2,MT]*Df[k3,MSf[1,3,3]]*
   Df[k4,MGl]]*...
+ B0[p2,MT,MT]*B0[p2,MSf[1,3,3],MSf[1,3,3]]*...
+ T[Df[k1,MSf[1,3,3]]*Df[k2,MSf[1,3,3]]*
   Df[k3,MT]*Df[k4,MGl]]*...
+ T[Df[k1,MSf[1,3,3]]*Df[k2,MSf[1,3,3]]*
   Df[k3,MGl]*Df[k4,MT]*Df[k5,MT]]*...
```

in terms of scalar integrals

TwoCalc Output

Scalar two-loop integrals:

```
T[Df[ki1,M1]*Df[ki2,M2]*...*Df[kin,Mn]]
```

$$= \langle\langle \frac{1}{(k_{i_1}^2 - m_1^2)(k_{i_2}^2 - m_2^2)\ldots(k_{i_n}^2 - m_n^2)} \rangle\rangle$$

$\langle\langle \ldots \rangle\rangle$ denotes the integration over the loop momenta

TwoCalc Output

In case of a more complicated output:

- more scalar two-loop integrals
- also scalar two-loop integrals of the form:

```
Y[Num[k1]*
  Den[Df[k2,M2]*Df[k3,M3]*Df[k4,M4]*Df[k5,M5]]]
```

$$= \langle\langle \frac{k_1^2}{(k_2^2 - m_2^2)(k_3^2 - m_3^2)(k_4^2 - m_4^2)(k_5^2 - m_5^2)} \rangle\rangle$$

TwoCalc Output

In case of a more complicated output:

further simplification is useful:

two connected routines available:

Simple.m: extension of the look-up table in TwoCalc:
decompositions of scalar two-loop integrals
into products of one-loop integrals and
simpler two-loop integrals

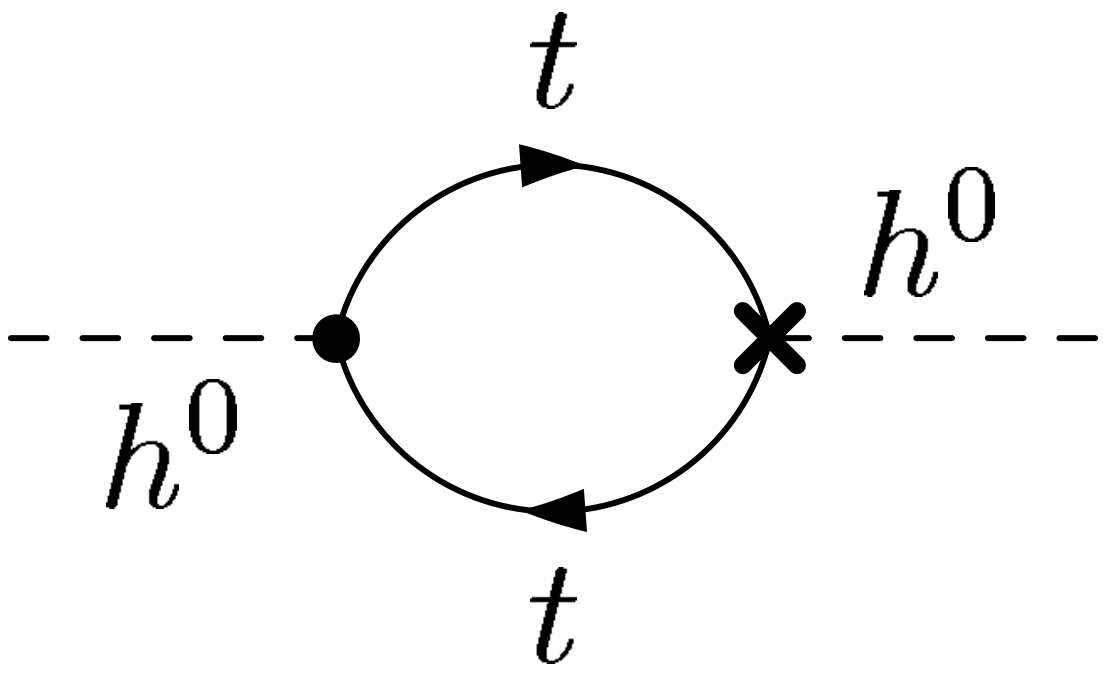
Symm.m: application of symmetry relations

Diagrams with counterterm insertions

Full calculation of a two-loop self energy:

(not only a single diagram)

Diagrams with counterterm insertions have to be included:



Diagrams with counterterm insertions

Procedure:

- create `FeynArts` amplitude:

  `CreateTopologies` → `CreateCTTopologies`

- evaluate sums over colour indices
- change conventions

  `FeynAmp[...,k1,k2,k3,k4,k5,p,...]`

  → `FeynAmp[...,k1,k2,p,...]`

- use `OneCalc` instead of `TwoCalc`

  `TwoLoop.m` → `OneCalc.m`

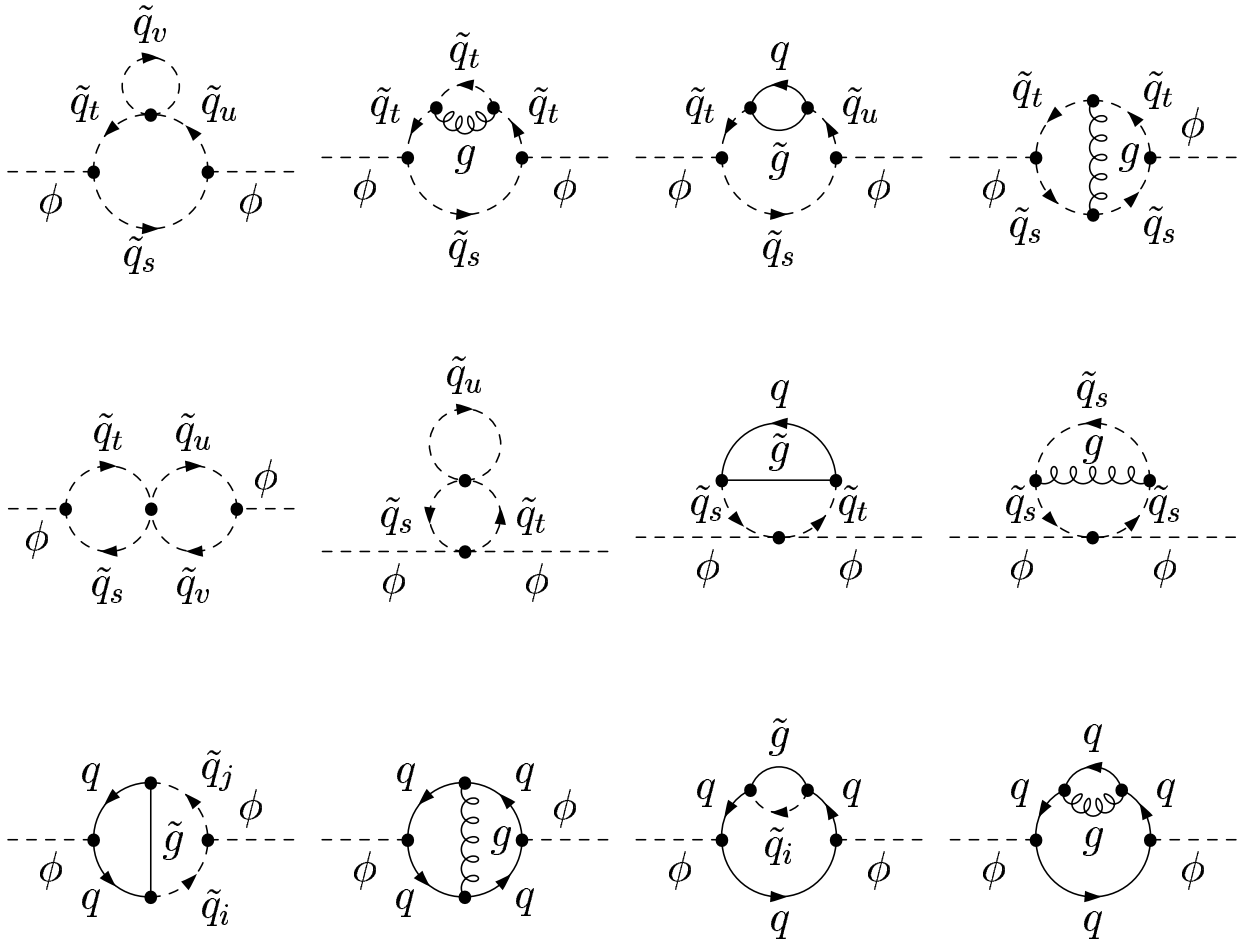
  `TwoLoop/TwoLoopSum` → `OneLoop/OneLoopSum`

Getting “reasonable” numbers

- Determine all self energy diagrams needed for the correction that shall be calculated
  (can be a subset of diagrams)

Getting “reasonable” numbers

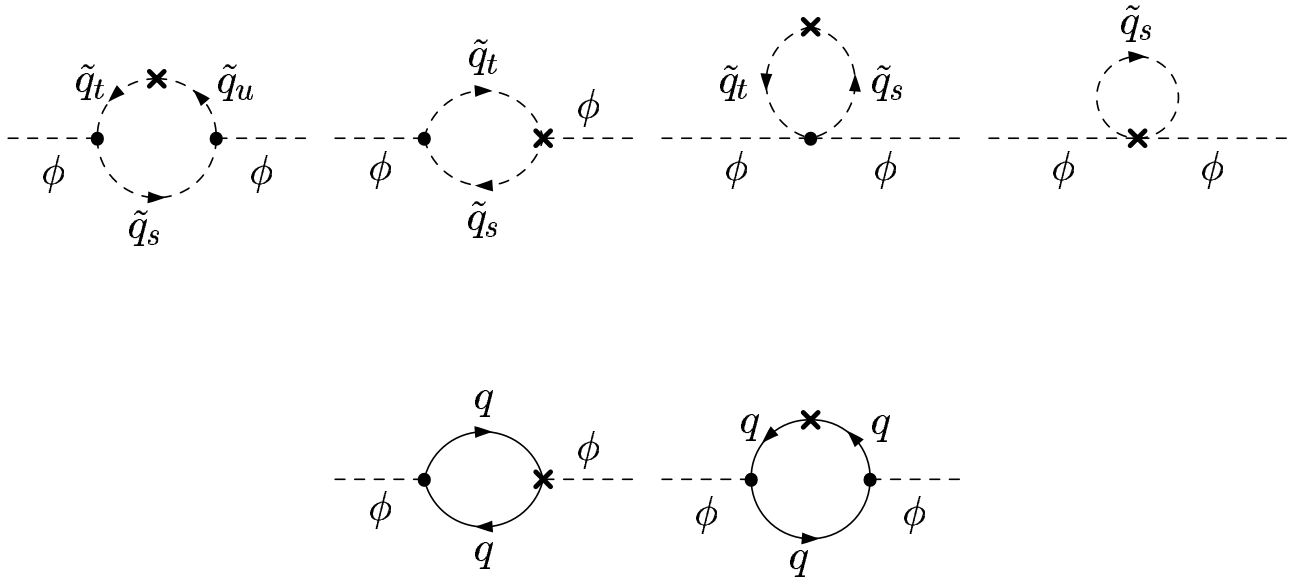
For corrections to the mass of the lightest Higgs boson $\mathcal{O}(\alpha_q \alpha_s)$:



Getting “reasonable” numbers

... and the corresponding diagrams

with counterterm insertions



Getting “reasonable” numbers

- Determine all self energy diagrams needed
  for the correction that shall be calculated
    (can be a subset of diagrams)
- Do not forget two-loop counterterms
- Use `FeynArts`, `SUNSimplify`, `simplesubst` to create the amplitudes
- Use `TwoCalc` and `OneCalc` for the evaluation of the amplitudes in terms of scalar integrals
- Expand in $\epsilon = 4 - D$

Getting “reasonable” numbers

For the expansion:

Divide scalar integrals in

- a divergent part $\sim \frac{1}{\epsilon}, \sim \frac{1}{\epsilon^2}$
- a finite part

In the expansion:

Divergent terms have to cancel

Last problem:

Numerical evaluation of the finite parts of the scalar integrals

Numerical Evaluation

Numerical evaluation of the finite parts of scalar integrals:

- some can be expressed analytically
- some need numerical integration

Existing package: S2L:

- C++ program for the numerical integrations
- can be used via a MathLink connection

Finally

Nobody knows as much about a program as the author:

- `FeynArts`: T. Hahn
- `TwoCalc`, `OneCalc`, `Simple.m`, `Symm.m`: G. Weiglein
- `simplesubst.m`: G. Weiglein, A. Freitas
- `SUNSimplify.m`: T. Hahn, H.R.
- `S2L`: S. Bauberger