LanHEP - a package for automatic generation of Feynman rules from the Lagrangian. `Updated version 1.4.`

A. Semenov
email: `semenov@theory.npi.msu.su`

March 31, 2008

Introduction

The LanHEP program is aimed for Feynman rules generation in momentum representation. It reads the Lagrangian written in the compact form close to one used in publications, with summation over indices of broken symmetries and using special symbols for complicated expressions. The output is Feynman rules in terms of physical fields and independent parameters. Two formats are available for this output: CompHEP tables and tables in LaTeX code.

The format of the LanHEP input file is described in details in the LanHEP User's Manual [2]. The LanHEP program and related papers can be found in the WWW page:

`http://theory.npi.msu.su/~semenov/lanhep.html`

Here we describe new features implemented into LanHEP package:

- heuristics in simplification of large trigonometric expressions;
- verifying and generating hermitian conjugate terms of the Lagrangian;
- 2-component fermion notation;
- probing the kinetic and mass terms of the Lagrangian;
- conditional processing of model.

1 Reducing trigonometric expressions

Some physical models, such as Minimal Supersymmetric Standard Model [3] and Two Higgs Doublet Model [4] (Feynman rules generations for these models by means of LanHEP are described in [5, 6]) involve large expressions built of trigonometric functions. In particular, in the models mentioned above two angles, α and β, are involved, thus the Lagrangian may be written in LanHEP notation using the following definitions:

```
parameter sa=0.5:'sinus alpha',
          ca=Sqrt(1-sa**2):'cosine alpha'.
parameter sb=0.9:'sinus beta',
          cb=Sqrt(1-sb**2):'cosine beta'.
```

One can find in the output large expressions in terms of `sa, sb, ca, cb`. Usually these expressions can be simplified by using derivative values like $\sin 2\alpha$, $\sin(\alpha+\beta)$ etc. In the previous version 1.3 user was obliged to simplify all expressions appearing in the output and declare them by `SetAngle` statement.

LanHEP version 1.4 can apply several heuristic algorithms to simplify these expressions. For each angle α, user should declare parameters for $\sin\alpha$, $\cos\alpha$, $\sin 2\alpha$, $\cos 2\alpha$, $\tan\alpha$. Than user should use `angle` statement:

```
angle sin=p1, cos=p2, sin2=p3, cos2=p4, tan=p5, texname=name.
```

Here *pN* — parameter identifiers, *name* — LaTeX name for angle, it is used to generate automatically LaTeX names for trigonometric functions of this angle if these names are not set explicitly by `SetTexName` statement. This statement should immediately follow the declaration of the parameters for $\sin\alpha$ and $\cos\alpha$. Only the `sin` and `cos` options are obligatory. Other parameters (i.e. $\sin 2\alpha$, $\cos 2\alpha$, $\tan\alpha$) should be declared if these parameters are defined before $\sin\alpha$ and $\cos\alpha$. If the former parameters will be declared in terms of latter ones, they are recognized automatically and they need not appear in `angle` statements.

For example, the declaration for trigonometric functions of β angle in MSSM may read:

```
parameter tb=2.52:'Tangent beta'.
parameter sb=tb/Sqrt(1+tb**2):'Sinus beta'.
parameter cb=Sqrt(1-sb**2):'Cosine beta'.

angle sin=sb, cos=cb, tan=tb, texname='\\beta'.

parameter s2b=2*sb*cb:'Sinus 2 beta'.
parameter c2b=cb**2-sb**2:'Cosine 2 beta'.
```

Here the parameters `s2b` and `c2b` are recognized automatically by LanHEP as $\sin 2\alpha$ and $\cos 2\alpha$ since they are declared in terms of `sa, ca` parameters.

For a couple of angles (α, β in this example) user should declare the parameters for $\sin(\alpha+\beta)$, $\cos(\alpha+\beta)$, $\sin(\alpha-\beta)$, and $\cos(\alpha-\beta)$ to allow using all implemented heuristics:

```
parameter sapb=sa*cb+ca*sb :  'sin(a+b)'.
parameter samb=sa*cb-ca*sb :  'sin(a-b)'.
parameter capb=ca*cb-sa*sb :  'cos(a+b)'.
parameter camb=ca*cb+sa*sb :  'cos(a-b)'.
```

These parameters are recognized as trigonometric functions automatically, by analysis of right-side expressions.

It is possible to control the usage of heuristics by the statement:

```
option SmartAngleComb=N.
```

where N is a number:

0 heuristics are switched off;

1 heuristics are switched on (by default);

2 same as 1, prints the generated substitution rule if the simplified expressions is consists of more than 3 monomials;

3 same as 1, prints the generated substitution rule.

4 same as 3, prints some intermediate expressions.

The substitution rules are printed as `SetAngle` statement and can be used for manual improvement of expressions.

2 Two-component notation for fermions

Now it is possible to write Lagrangian terms using two-component notation for fermions[1]. The connection between two-component and four-component notations is summarized by following relations:

$$\psi = \begin{pmatrix} \xi \\ \bar\eta \end{pmatrix}, \qquad \psi^c = \begin{pmatrix} \eta \\ \bar\xi \end{pmatrix}, \qquad \bar\psi = \begin{pmatrix} \eta \\ \bar\xi \end{pmatrix}^T, \qquad \bar\psi^c = \begin{pmatrix} \xi \\ \bar\eta \end{pmatrix}^T.$$

If the user has declared a spinor particle `p` (with antiparticle `P`), the LanHEP notation for its components is:

ξ	$\rightarrow$	`up(p)`
$\bar\eta$	$\rightarrow$	`down(p)`
η	$\rightarrow$	`up(cc(p))` *or* `up(P)`
$\bar\xi$	$\rightarrow$	`down(cc(p))` *or* `down(P)`

To use the four-vector $\bar\sigma^\mu$ one should use the statement

```
special sigma:(spinor2,spinor2,vector).
```

Note that the `sigma` object is not defined by default, thus the above statement is required. It is possible also to use another name instead of `sigma` (this object can be recognized by LanHEP by the types of its indices).

LanHEP uses the following rules to convert the two-component fermions to four-component ones (we use $P_{R,L} = (1 \pm \gamma^5)/2$):

$\eta_1\xi_2 = \bar\psi_1 P_L \psi_2$	`up(P1)*up(p2)`	$\rightarrow$	`P1*(1-gamma5)/2*p2`
$\bar\eta_1\bar\xi_2 = \bar\psi_1 P_R \psi_2$	`down(P1)*down(p2)`	$\rightarrow$	`P1*(1+gamma5)/2*p2`
$\xi_1\xi_2 = \bar\psi_1^c P_L \psi_2$	`up(p1)*up(p2)`	$\rightarrow$	`cc(p1)*(1-gamma5)/2*p2`
$\bar\xi_1\bar\xi_2 = \bar\psi_1 P_R \psi_2^c$	`down(P1)*down(P2)`	$\rightarrow$	`P1*(1+gamma5)/2*cc(P2)`
$\xi_1\eta_2 = \bar\psi_1^c P_L \psi_2^c$	`up(p1)*up(P2)`	$\rightarrow$	`cc(p1)*(1-gamma5)/2*cc(P2)`
$\bar\xi_1\bar\eta_2 = \bar\psi_1^c P_R \psi_2^c$	`down(p1)*down(P2)`	$\rightarrow$	`cc(p1)*(1+gamma5)/2*cc(P2)`
$\eta_1\eta_2 = \bar\psi_1 P_L \psi_2^c$	`up(P1)*up(P2)`	$\rightarrow$	`P1*(1-gamma5)/2*cc(P2)`
$\bar\eta_1\bar\eta_2 = \bar\psi_1^c P_R \psi_2$	`down(p1)*down(p2)`	$\rightarrow$	`cc(p1)*(1+gamma5)/2*p2`
$\bar\xi_1\sigma^\mu\xi_2 = \bar\psi_1\gamma^\mu P_L \psi_2$	`down(P1)*sigma*up(p2)`	$\rightarrow$	`P1*gamma*(1-gamma5)/2*p2`
$\bar\eta_1\sigma^\mu\eta_2 = \bar\psi_1^c\gamma^\mu P_L \psi_2^c$	`down(p1)*sigma*up(P2)`	$\rightarrow$	`cc(p1)*gamma*(1-gamma5)/2*cc(P2)`

[1]Note, that this feature is under tests now.

3 Hermitian conjugate terms

Since the version 1.4, LanHEP is able to check the correctness of hermitian conjugate terms in the Lagrangian. Do do this, user should use the statement:

```
    CheckHerm.
```

For example, let us consider the following (physically meaningless) input file, where charged scalar field is declared and cubic terms are introduced:

```
    scalar h/H.
    parameter a,b,c.
    lterm a*(h*h*H+H*H*h)+b*h*h*H+c*H*H*h + h**3.
    CheckHerm.
```

The output of LanHEP is the following (here 2 is the symmetry factor):

```
CheckHerm: vertex (h, h, h): conjugate (H, H, H) not found.
CheckHerm: inconsistent conjugate vertices:
    (H, h, h)             (H, H, h)
      2*a          <->       2*a
      2*b          <->   (not found)
   (not found)     <->       2*c
```

If the conjugated vertex is not found, warning message is printed. If both vertices are present but they are inconsistent, more detailed output is provided. For each couple of the incorrect vertices LanHEP outputs 3 kinds of monomials: 1) those which are found in both vertices (these monomials are correctly conjugated), 2) the monomials found only in first vertex, and 3) the monomials found only in the second vertex.

It is possible to generate hermitian conjugate terms automatically by putting the symbol `AddHermConj` to `lterm` statement:

    lterm *expr* + AddHermConj.

Continuing the former example, one can write:

```
    lterm a*H*H*h + b*H**3 + AddHermConj.
```

Note, that the symbol `AddHermConj` adds hermitian conjugate to all terms in the `lterm` statement. It means in particular that in the statement

    lterm *expr1* + (*expr2* + AddHermConj).

the conjugate terms are added to both *expr1* and *expr2*. Thus, one should not place self-conjugate terms in the `lterm` statement where `AddHermConj` present (or one should supply these terms with 1/2 factor).

4 Probing the kinetic and mass terms of the Lagrangian

When LanHEP is used to generate CompHEP model files, the information about particles propagators is taken from particle declaration, where particle mass and width (for Breit-Wigner's propagators) are provided. Thus, user is not obliged to supply kinetic and mass terms in `lterm` statements. Even if these terms are written, they do not affect the CompHEP output. Version 1.4 of LanHEP allows user to examine whether the mass sector of the Lagrangian is consistent. To do this, use the statement

```
    CheckMasses.
```

This statement must be put after all `lterm` statements of the input file.

When the `CheckMasses` statement is used, LanHEP creates the file named `masses.chk` in the current directory. It contains warning messages if some inconsistencies are found:

- missing or incorrect kinetic term;
- mass terms with mass different from the value specified at particle declaration;
- off-diagonal mass terms.

Note that in the complicated models like MSSM the masses could depend on other parameters and LanHEP is not able to prove that expressions in actual mass matrix and in parameters declared to be the masses of particles are identical. Moreover, it is often impossible to express the masses as formulas written in terms of independent parameters. For this reason LanHEP evaluates the expressions appearing in mass sector numerically (basing on the parameters values specified by user).

It is typical for MSSM that some fields are rotated by unitary matrices to diagonalize mass terms. In some cases, values of mixing matrices elements can not be expressed by formulas and need to be evaluated numerically. LanHEP can recognize mixing matrix if the elements of this matrix were used in `OrthMatrix` statement. LanHEP restores and prints the mass original matrix before fields rotation by mixing matrix. Then LanHEP calculates numerical values of mixing matrix elements to diagonalize the physical mass matrix (CERNlib routine E202 is used).

5 Conditional processing of the model

Let us consider the LanHEP input files for Standard Model in t'Hooft-Feynman and unitary gauges. It is clear that these input files differ by several lines only - the declaration of gauge bosons and Higgs doublet. It is more convenient to have only one model definition file. In this case the conditional statements are necessary. LanHEP allows the user to define several *keys*, and use these keys to branch among several variants of the model. The keys have to be declared by the `keys` statement:

```
    keys name1=value1, name2=value2, ... .
```

Then one can use the conditional statements:

```
    do_if key==value1.
          actions1
    do_else_if key==value2.
          actions2
    do_else_if key==value3.
```

```
        ...
do_else.
        default actions
end_if.
```

The statements `do_else_if` and `do_else` are not obligatory. The value of the key is a number or symbolic string. An example of using these statement in the Standard model may read:

```
keys Gauge=unitary.

do_if Gauge==Feynman.
vector Z/Z:('Z-boson',mass MZ = 91.187, width wZ = 2.502, gauge).
do_else_if Gauge==unitary.
vector Z/Z:('Z-boson',mass MZ = 91.187, width wZ = 2.502).
do_else.
write('Error:  key Gauge must be either Feynman or unitary').
quit.
end_if.
```

Thus, to change the choice of gauge fixing in the generated Feynman rules it is enough to modify one line of input file. There is another possibility, to set the key value from the command line at LanHEP launch:

```
lhep -key Gauge=Feynman filename.
```

If the value of key is set from the command line at the program launch, it can not be modified by `keys` statement.

References

[1] E. Boos et al., INP MSU–94–36/358, SNUTP 94-116 (Seoul, 1994); hep-ph/9503280
P. Baikov et al., Physical Results by means of CompHEP, in Proc.of X Workshop on High Energy Physics and Quantum Field Theory (QFTHEP-95), ed.by B.Levtchenko, V.Savrin, Moscow, 1996, p.101 , hep-ph/9701412

[2] A. Semenov. *LanHEP — a package for automatic generation of Feynman rules. User's manual.* INP MSU 96–24/431, Moscow, 1996; hep-ph/9608488
A. Semenov. Nucl.Inst.&Meth. **A393** (1997) p. 293.
A. Semenov. LanHEP - a package for automatic generation of Feynman rules from the Lagrangian. Updated version 1.3. INP MSU Preprint 98–2/503.

[3] J. Rosiek. Phys. Rev. **D41** (1990) p. 3464.

[4] J.F. Gunion, H.E. Haber, G. Kane and S. Dawson. *The Higgs Hunter Guide,* Addison-Wesley, 1990.

[5] A. Belyaev, A. Gladyshev and A. Semenov. IFT–P–093–97, hep-ph/9712303.

[6] M. Dubinin and A. Semenov, *in preparation.*