LanHEP - a package for automatic generation of Feynman rules from the Lagrangian. *Updated version 2.0.*

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

May 9, 1999

Abstract

This short note describes new features of the version 1.5 of the LanHEP software package. 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.

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
http://wwwlapp.in2p3.fr/cpp/lanhep.html
```

In this paper we describe new features implemented into LanHEP package since the 1.3 version:

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

- using the superpotential formalism in MSSM and its extensions;
- splitting the vertices with 4 colored particles at multiplet level;
- checking the BRST invariance of the Lagrangian;
- constructing the ghost Lagrangian.

Also the existing functionality is improved:

- the function `anti` can be applied to the multiplets, rather than to the particles only;
- it is possible to use numeric indices at the matrices and multiplets, e.g. `eps^1^2^3`;
- Reducing the expressions containing elements of orthogonal matrices is improved. In the previous versions only the products of 2 elements were processed, i.e. $C_{ij}C^T_{jk} = \delta_{ik}$. Now products of 4 elements can be simplified as $C_{ij}C^T_{jk}C_{lm}C^T_{mn} = \delta_{ik}\delta_{ln}$;
- internal data representation is improved, so the speed and memory requirements significantly enhanced.

1 Reducing the 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 the 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 the user was obliged to simplify all expressions appearing in the output and declare them by `SetAngle` statement.

Now LanHEP 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) the 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 the 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 all generated substitution rule.

4 same as 1, prints all generated substitution rule and some intermediate expressions (debug mode).

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

2 Two-component notation for fermions

It is possible to write Lagrangian terms using two-component notation for fermions. The connection between two-component and four-component notations is summarized by the 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:

$$\begin{array}{lcl}
\xi & \to & \texttt{up(p)} \\
\bar\eta & \to & \texttt{down(p)} \\
\eta & \to & \texttt{up(cc(p))}\ \ or\ \ \texttt{up(P)} \\
\bar\xi & \to & \texttt{down(cc(p))}\ \ or\ \ \texttt{down(P)}
\end{array}$$

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 $P_{R,L} = (1 \pm \gamma^5)/2$):

$$\begin{array}{lrcl}
\eta_1\xi_2 = \bar\psi_1 P_L \psi_2 & \texttt{up(P1)*up(p2)} & \to & \texttt{P1*(1-gamma5)/2*p2} \\
\bar\eta_1\bar\xi_2 = \bar\psi_1 P_R \psi_2 & \texttt{down(P1)*down(p2)} & \to & \texttt{P1*(1+gamma5)/2*p2} \\
\xi_1\xi_2 = \bar\psi_1^c P_L \psi_2 & \texttt{up(p1)*up(p2)} & \to & \texttt{cc(p1)*(1-gamma5)/2*p2} \\
\bar\xi_1\bar\xi_2 = \bar\psi_1 P_R \psi_2^c & \texttt{down(P1)*down(P2)} & \to & \texttt{P1*(1+gamma5)/2*cc(P2)} \\
\xi_1\eta_2 = \bar\psi_1^c P_L \psi_2^c & \texttt{up(p1)*up(P2)} & \to & \texttt{cc(p1)*(1-gamma5)/2*cc(P2)} \\
\bar\xi_1\bar\eta_2 = \bar\psi_1^c P_R \psi_2^c & \texttt{down(p1)*down(P2)} & \to & \texttt{cc(p1)*(1+gamma5)/2*cc(P2)} \\
\eta_1\eta_2 = \bar\psi_1 P_L \psi_2^c & \texttt{up(P1)*up(P2)} & \to & \texttt{P1*(1-gamma5)/2*cc(P2)} \\
\bar\eta_1\bar\eta_2 = \bar\psi_1^c P_R \psi_2 & \texttt{down(p1)*down(p2)} & \to & \texttt{cc(p1)*(1+gamma5)/2*p2} \\
 & & & \\
\bar\xi_1\sigma^\mu\xi_2 = \bar\psi_1\gamma^\mu P_L \psi_2 & \texttt{down(P1)*sigma*up(p2)} & \to & \texttt{P1*gamma*(1-gamma5)/2*p2} \\
\bar\eta_1\sigma^\mu\eta_2 = \bar\psi_1^c\gamma^\mu P_L \psi_2^c & \texttt{down(p1)*sigma*up(P2)} & \to & \texttt{cc(p1)*gamma*(1-gamma5)/2*cc(P2)}
\end{array}$$

3 Hermitian conjugate terms

LanHEP is able to check the correctness of hermitian conjugate terms in the Lagrangian. To 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 the particle declaration, where particle mass and width (for Breit-Wigner's propagators) are provided. Thus, the 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 the 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 with t'Hooft-Feynman and unitary gauge-fixing. 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 word in the input file. There is another opportunity, 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, the value for this key in the keys statement is ignored.

6 Using the superpotential formalism in MSSM and its extensions

In the supersymmetric models one makes use of the superpotential — a polynomial W depending on scalar fields A_i (superpotential also can be defined in terms of superfields, we do not consider this case). Then, there is the contribution to the Lagrangian: Yukawa terms in the form

$$-\frac{1}{2}\left(\frac{\partial^2 W}{\partial A_i \partial A_j}\Psi_i\Psi_j + H.c.\right)$$

and $F_i^* F_i$ terms, where $F_i = \partial W/\partial A_i$ (for more details, see [3] and references therein).

To use this formalism in LanHEP, one should define first the multiplets of matter fields and then define superpotential as let-substitution. The example of MSSM with single generation may read:

```
keep_lets W.
let W=eps*(mu*H1*H2+ml*H1*L*R+md*H1*Q*D+mu*H2*Q*U).
```

Here symbols `H1, H2, L, R, Q, U, D` are defined somewhere else as doublets and singlets in terms of scalar particles.

Note that before the definition of `W` this symbol should appear in the `keep_lets` statement. It is necessary to notify LanHEP that let-substitutions (multiplets) at the definition of `W` should not be expanded. Without this statement, `W` will not contain symbols of multiplets but only the particles which were used at multiplets definition.

Since `W` was declared in the `keep_lets` statement and contains the symbols of multiplets, one can evaluate the variational derivative of `W` by one or two multiplets, e.g. `df(W,H1)` or `df(W,H1,L)`. Thus the Yukawa terms may be written:

```
lterm - df(W,H1,H2)*fH1*fH2 - ...  + AddHermConj.
```

Here `fH1, fH2` are fermionic partners of corresponding multiplets.

To introduce the $F_i^* F_i$ terms one needs to declare conjugated superpotential, e.g. `Wc`, and write:

```
lterm - df(W,H1)*df(Wc,H1c) - ....
```

The better way is to use the function `dfdfc(W,H1)` instead:

```
lterm - dfdfc(W,H1) - ....
```

The function evaluates the variational derivative, multiply it by conjugated expression and returns the result. Moreover, it can introduce auxiliary fields to split vertices with 4 color particles (in the case of CompHEP output); see the next section for more details.

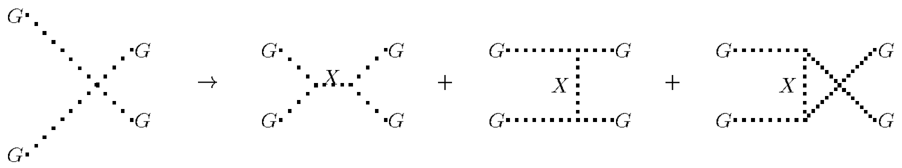
7 Splitting the vertices with 4 colored particles

CompHEP Lagrangian tables don't describe explicitly color structure of a vertex. If color particles present in the vertex, the following implicit convolutions are assumed (supposing p, q, r are color indices of particles in the vertex):

- δ_{pq} for two color particles A^1_p, A^2_q;
- λ^r_{pq} for three particles, which are color triplet, antitriplet and octet;
- f^{pqr} for three color octets.

Other color structures are forbidden in CompHEP.

So, to introduce the 4-gluon vertex $f^{pqr}G^q_\mu G^r_\nu f^{pst}G^s_\mu G^t_\nu$ one should split this 4-legs vertex into 3-legs vertices $f^{pqr}G^q_\mu G^r_\nu X^p_{\mu\nu}$:



Here the field $X^p_{\mu\nu}$ is Lorenz tensor and color octet, and this field also has constant propagator. If gluon name in CompHEP is `'G'`, the name `'G.t'` is used for this tensor particle; its indices denoted as `'m_'` and `'M_'` (`'_'` is the number of the particle in table item).

The described transformation is performed by LanHEP automatically and transparently for the user. Each vertex containing 4 color particles is splitted to 2 vertices which are joined by automatically generated auxiliary field.

The same technique is applied in MSSM where more vertices with 4 color particles appear: vertices with 2 gluons and 2 squarks and vertices with 4 squarks. However, the large amount of vertices with 4 squarks requires many auxiliary fields, which can easily break CompHEP limitations on the particles number. It is possible however to reduce significantly the number of vertices and auxiliary fields if one introduce auxiliary fields at the level of multiplets.

The vertices with 4 squarks come from DD and F^*F terms. For example, there is the term $\frac{1}{2}D^a_G D^a_G$ in the Lagrangian,

$$D^a_G = g_s(Q^*_i\lambda^a Q_i + D^*\lambda^a D + D^*\lambda^a D),$$

where Q, D, U are squarks multiplets, and λ is Gell-Mann matrix. Instead of evaluating this expression and that splitting all vertices independently, one can introduce one color octet auxiliary field ξ^a and write this Lagrangian term as $D^a_G\xi^a$.

Other DD terms contain both color and colorless particles. Thus, the term $D^i_A D^i_A$ with

$$D^i_A = g_1(Q^*T^iQ + L^*T^iL + H^*_1T^iH_1 + H^*_2T^iH_2),$$

can be represented as

$$g_1(Q^*T^iQ)\xi^i + g_1^2(Q^*T^iQ)(L^*T^iL + H^*_1T^iH_1 + H^*_2T^iH_2) + g_1^2(L^*T^iL + H^*_1T^iH_1 + H^*_2T^iH_2)^2$$

where ξ^i is the triplet of auxiliary fields. This terms can also be written in the another form:

$$g_1(Q^*T^iQ + L^*T^iL)\xi^i + g_1^2(Q^*T^iQ + L^*T^iL)(H_1^*T^iH_1 + H_2^*T^iH_2) + g_1^2(H_1^*T^iH_1 + H_2^*T^iH_2)^2,$$

where all vertices with 4 scalars (except vertices with Higgs particles) are splitted. Although the latter splitting is not obligatory, it can reduce significantly the amount of vertices.

The similar technique is applicable to the F^*F terms, with the transformation $F_i^*F_i \to F_i^*\xi_i^* + F_i\xi_i$.

Thus, we distinguish two types of vertices splitting: splitting at multiplet level and splitting at vertices level. Note that splitting the vertices with two gluons and two squarks must be done at vertices level after combining the similar terms, otherwise they would contain the elements of squark mixing matrices.

The vertices splitting at multiplet level is implemented in LanHEP mainly for MSSM needs. The first case refers to DD terms. The user should declare several let-substitutions and then put in `lterm` statement the squared sum:

```
let a1=g*Q*tau*q/2,
    a2=g*L*tau*l/2,
    a3=g*H1*tau*h1/2,
    a4=g*H2*tau*h2/2.
lterm - ( a1 + a2 + a3 + a4 ) ** 2 / 2.
```

In this case LanHEP looks for the square of the sum of several let-substitution symbols, each containing two color or merely scalar particles. If such expression is found, it is replaced as in the previous formulas. Higgs doublets are not taking into transformation, since they contain VEVs.

The vertices splitting in F^*F terms is performed by `dfdfc` function (see previous section). After taking the variational derivative the monomials with two color or scalar particles (except Higgs ones) are multiplied by auxiliary fields, thus mediating the vertices with 4 color (scalar) particles.

The multiplet level vertices splitting is controlled by the statement

`option SplitCol1=`N.

where N is a number:

-1 remove all vertices with 4 color particles from Lagrangian;

0 turn off multiplet level vertices splitting;

1 allows vertices splitting with 4 color multiplets;

2 allows vertices splitting with any 4 scalar multiplets (except Higgs ones).

The value of this option can be set to different values before executing different `lterm` statements.

The vertices level splitting is performed after combining similar terms of the Lagrangian. This splitting can be controlled by the statement

`option SplitCol2=`N.

where N is a number:

0 disable vertex level splitting;

1 enable vertex level splitting (only for vertices with 4 color particles).

For CompHEP output, the default value is 2 for `SplitCol1` and 1 for `SplitCol2`. For LaTeX output, default value is 0 for both options.

8 Checking BRST invariance and constructing the ghost Lagrangian

LanHEP can check the BRST invariance (see the reviews in [7, 8]) of the Lagrangian. First, the user should declare the BRST transformations for the fields in the model by means of `brst_transform` statement:

`brst_transform` *field* `->` *expression*.

For example, the BRST transformation for the photon $\delta A_\mu = \partial_\mu c^A + ie(W^+_\mu c^- - W^-_\mu c^+)$ can be prescribed by the statement:

```
brst_transform A -> deriv*'A.c'+i*EE*('W+'*'W-.c'-'W-'*'W+.c').
```

The file 'sm_brst.mdl' in LanHEP distribution contains the code for gauge and Higgs fields transformation corresponding to the CompHEP implementation of the Standard model.

Since the transformations are defined, the statement

```
CheckBRST.
```

enables the BRST transformation for Lagrangian terms (so it should be placed before the first `lterm` statement). The output is CompHEP or LaTeX file with resulting expression. Certainly, if the Lagrangian is correct, the output files are empty. However some expressions identical to zero could be not simplified and remain in the output.

The BRST transformations allow also to construct the ghost Lagrangian ([9]). The gauge-fixing Lagrangian reads as:

$$\mathcal{L}_{GF} = G^- G^+ + \frac{1}{2}|G^Z|^2 + \frac{1}{2}|G^\gamma|^2,$$

where G^i $(i = \pm, \gamma, Z)$ are gauge-fixing functions. The ghost Lagrangian ensures the BRST invariance of the entire Lagrangian and can be written as

$$\mathcal{L}_{Gh} = -\bar{c}^i \delta_{BRST} G^i + \delta_{BRST} \tilde{\mathcal{L}}_{Gh}.$$

where $\delta_{BRST}\tilde{\mathcal{L}}_{Gh}$ is an overall function, which is BRST-invariant.

So, for the photon and $G^\gamma = (\partial \cdot A)$, the LanHEP code for gauge-fixing and ghost terms read as:

```
let G_A = deriv*A.
lterm -G_A**2/2.
lterm -'A.C'*brst(G_A).
```

Here the `brst` function is used to get BRST-transformation of the specified expression.

The inverse BRST transformation can also be used. One can declare the transformations for the fields by means of `brsti_transform`. The function `brsti(`*expr*`)` can be used in `lterm` statements.

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.*

[7] T. Kugo, I. Ojima. Phys. Lett. 73B (1978) 459.

[8] L. Baulieu. Phys. Rep. 129 (1985) 1–74.

[9] F. Boudjema, *private communication.*