CalcHEP- Calculator for High Energy Physics - a package for evaluation of Feynman diagrams, integration over multi-particle phase space, and event generation.

A.Pukhov*

User's manual for version 2.1
(May 16, 2002)

*pukhov@theory.sinp.msu.ru

Contents

1 Preface

1.1 Introduction

CalcHEP is a package for automatic calculations of elementary particle decay and collision properties in the lowest order of perturbation theory (the tree approximation). The main idea prescribed into CalcHEP is to make available passing on from the Lagrangian to the final distributions effectively with a high level of automation.

CalcHEP is a next development of the CompHEP[1] package which was created by the author together with his colleagues in Skobeltsyn Institute of Nuclear Physics. Other packages created to solve a similar problem are *FeynArts/FeynCalc*[3], *GRACE*[4], *HELAS*[5], *MADGRAPH*[6], PHELAS[7], O'MEGA[8]. See also the review [9].

CalcHEP is a menu-driven system with the context help. The notations used in CalcHEP are very similar to those used in particle physics.

The present version has 2 built-in physical models. They are the Standard Model and its minimal super-symmetry extension (MSSM). Both of them are presented in two gauges, the unitary and the t'Hooft - Feynman ones. The realization of MSSM was done by A. Semenov [2]. The user can change particle interaction and model parameters. It is also possible to create new models of particle interaction.

In the present version polarizations are not taken into account. Averaging over initial and summing over final polarizations are performed automatically.

The CalcHEP package consists of two parts: symbolic and numerical. The both of them are written in the *C* programming language. The symbolic part produces *C* codes for a squared matrix element, and they are used in the numerical calculation later on.

The symbolic part of CalcHEP lets the user:

- select a process by specifying incoming and outgoing particles for the decays of $1 \to 2, \ldots, 1 \to 5$ types and the collisions of $2 \to 2, \ldots, 2 \to 4$ types;
- generate Feynman diagrams, display them, and create the corresponding LaTeX output;
- exclude some diagrams;

- generate and display squared Feynman diagrams;
- calculate analytical expressions corresponding to squared diagrams by using the fast built-in symbolic calculator; There is a possibility to perform these calculations in the leading order of $1/N_c$ expansion[1].
- save symbolic results corresponding to the squared diagrams calculated in the *Reduce* and *Mathematica* codes for further symbolic manipulations;
- generate the optimized *C* codes for the squared matrix elements for further numerical calculations;
- launch the compilation of the generated codes and the corresponding numerical session.

The numerical part of CalcHEP offers to:

- convolute the squared matrix element with structure functions and beam spectra. The *CTEQ* parton distribution functions, the ISR and Beamstrahlung spectra of electrons, the laser photon spectrum, and the Weizsaecker-Williams photon structure functions are available;
- modify physical parameters (total energy, charges, masses etc.) involved in the process;
- select the scale parameter for evaluation of the QCD coupling constant and parton structure functions;
- apply various kinematic cuts.
- define the kinematic scheme (phase space parameterization) for effective Monte Carlo integration;
- introduce a phase space mapping in order to smooth sharp peaks of a squared matrix element and structure functions;
- perform a Monte Carlo phase space integration by *Vegas*;

[1]See section (3.2.5).

- generate events;
- display distributions in various kinematic variables;
- create the graphical and LaTeX outputs for histograms.

CalcHEP generates only partonic level events. The task of hadronization is passed to other programs like PYTHIA [32]. The interface with PYTHIA is described in [33].

1.2 Acknowledgments

The author thanks A.Semenov for the implementation of MSSM in terms of CalcHEP/CompHEP notations. The possibility to work with MSSM is a more attractive facility of the package.

The current version of CalcHEP contains about 10% of code written by my colleagues V. Edneral, V. Ilyin, A. Kryukov, and A. Semenov for the CompHEP package. Namely, the color factors evaluation (A. Kryukov), the database table manipulation (A. Kryukov), graphic and pseudo-graphic representation of diagrams (V. Edneral), implementation of X11 (A. Semenov), some auxiliary numerical routines (V. Ilyin). I express my gratitude to all of them for their kind permission to use their codes in the CalcHEP package.

The author acknowledges beneficial discussions with E. Boos and V. Ilyin which motivated the development of the package.

2 Installation procedure

2.1 CalcHEP Web page

CalcHEP codes and manual are disposed on the following Web site
`http://theory.sinp.msu.ru/~pukhov/calchep.html`
Also there is a mirror at DESY
`http://www.ifh.de/~pukhov/calchep.html`

2.2 License

Non-profit Use License Agreement

This Agreement is to be held between the Author of the CalcHEP program and a Party which acquires the program. On acquiring the program the Party agrees to be bound by terms of this Agreement.

1. This License entitles the Licensee (one person) and the Licensee's research group to obtain a copy of the source code of CalcHEP and to use the acquired program for academic research or other non-profit purposes within the research group; or, it entitles the Licensee (a company, organization or computing center) to install the program and allow an access to the executable code to the members of the Licensee for academic research or other non-profit use.

2. No user or site will re-distribute the source code or executable code to a third party in the modified form. Any re-distribution must be accompanied with the current license.

3. This License does not permit any commercial (profit-making or proprietary) use or re-licensing or re-distributions. Persons interested in a for-profit use should contact the Author.

4. The Author of CalcHEP do not guarantee that the program is free of errors or meets its specification and cannot be held responsible for loss or consequential damage as a result of using it.

2.3 How to get the code

If you agree with the license above, you may get CalcHEP code from the CalcHEP Web site. The name of received file should be

`calchep_2.1.tar.gz`

Unpack this file by

`gtar -xzf calchep_2.1.tar.gz` [2]
As a result a directory calchep_2.1 should be created. Below we shall refer to it as `$CALCHEP`[3]

This directory contains source code of the CalcHEP package for UNIX platforms. After compilation of this code the CalcHEP binary executable files appear in the same directory.

2.4 Compilation procedure

In order to compile the CalcHEP source code you need a *C* compiler with the X11 graphics library and X11 include files. The compilation is launched by the

`./mk_all`

command started from the `calchep_2.1` directory.

If the *C* compiler is detected and the *C* sources are compiled successfully you will see the message:

`"CalcHEP can be started."`

Otherwise the corresponding error message comes out. See the section (2.6) for a discussion of possible problems.

The files in the `calchep_2.1` directory are used only for reading and execution. It provides a possibility to have only one installed package for several users. The `mk_all` command sets the needed file properties.

[2]If `gtar` is absent, perform the following couple of commands:
```
gunzip calchep_2.1.tar.gz
tar -xf calchep_2.1.tar .
```
[3]In script commands of the package the environment variable `CALCHEP` contains the path to commands.

2.5 User installation and start of the CalcHEP session

Each user has to create a directory in his own disk space to work with CalcHEP. One user can create several working directories for various tasks. The user working directory is prepared by the

```
./mkUsrDir <dir.Name>
```

command started from the `$CALCHEP` directory. For example

```
./mkUsrDir ~/work ,
```

creates the `work` directory in your disk space and put into this directory all needed files. Namely,

```
models/    tmp/        results/
calchep    calchep.ini
```

The directory `models` is used for the files which describe models of particle interactions. The directory `tmp` is created for temporal files. The directory `results` is assigned for a CalcHEP output.

To launch CalcHEP you should issue a command

```
./calchep.
```

It starts the CalcHEP symbolic session. Symbolic session in its turn creates the file `results/n_calchep` which performs numerical calculations.

The `calchep.ini` file allows to choose an appropriate text font for the CalcHEP window, select the color or colorless mode and also switch on/off the sound signal. The syntax of this file is self-explanatory.

2.6 Problems in compilation

Due to its first instruction the `mk_all` command calls `makeCC` which in its turn checks the existence of the *CC* file in the `$CALCHEP` directory. If this file exists, the *C* compiler name and its options are read from this file. Otherwise, as one usually has at the first start, this file is created and the needed option of *C* compiler are written down into this file.

After that the *makeCC* program tests the compiler. For this goal it generates various programs with the same name *test.c* and tries to compile and link them. If compilation is not satisfactory, the *makeCC* command finishes with the corresponding error message and asks you to correct the file `CC` in order to fit the requirement. The current test file `test.c` is saved. So, in the case of such an error you could update the `CC` file and start `mk_all`

again. The options listed below must be supported by C compiler:

- the ANSI mode of function prototypes;
- the signed char type;
- IEEE standard for treatment of denormalized numbers, NaNs, infinities and the handling of error cases.
- accesses to the X11 include and library files.

An example of content of the *CC* file for Linux platform is presented below:

```
cc -fsigned-char -I/usr/X11R6/include -L/usr/X11R6/lib
```

If all tests are passed on successfully the message appears:

```
C compiler OK. Starting CalcHEP source code compilation.
```

If your UNIX platform is one of the listed: Linux, IRIX, IRIX64, HP-UX, AIX, OSF1, the necessary options are known from the beginning except of the path to *X11*.

The *X11* library should exist in your computer. Say, if you can open a window by `xterm` command, then it is available. The only problem to set the right path to it. But the *X11* include files can be absent. Sometime it happens in case of Linux installed on a home computer.

Anyway, in the case of some problem on this step you can send a request for help to the CalcHEP author.

2.7 Compiler tuning.

A correction of the *CC* file could be used to tune compiler options according to your UNIX platform. In this case you should create `CC` by starting `makeCC`, improve the `CC` file, and launch `mk_all` after that. For example, you could switch on an optimization flag.

Also, there are two *macro* definitions for the *C* code, which could be useful to tune:

The first one is `STRSIZ`. This is a maximum size of strings in the CalcHEP models. By default `STRSIZ=2048`. If you would like to embed in CalcHEP some new interaction with a very cumbersome vertex you have to increase this size by the

```
-DSTRSIZ=<new value>
```

option.

The second *macro* defines a type of integer numbers which are used in symbolic calculations by CalcHEP. By default CalcHEP uses the 'long' type.

-DNUM_LONG_LONG forces the compiler to use *'long long'*. This type is not an ANSI standard and, perhaps, it is not supported in your case. Different realizations of the *'long long'* type use different formats for reading and writing such numbers. The user may specify this format defining the *'NUM_STR'*. By default CalcHEP uses

```
#define NUM_STR "lld"
```

-DNUM_DOUBLE option forces CalcHEP to emulate integer numbers via *'double'* float ones.

The *makeCC* program inserts an option which guarantees 8-byte size integer calculation in CalcHEP. If the size of the standard 'long' type is not enough, then the 'double' one is used.

3 User guide

3.1 Elements of the user interface

There are the following elements of the user interface in the CalcHEP package: *On-line Help, Menu, Message, String Editor, Table Editor, Diagram Viewer* and *Plot Viewer*. You can control them using the *Arrows* keys, *Enter*, *Esc*[4], *Backspace*, *PgUp/PgDn*[5] keys and the mouse click. CalcHEP is sensitive to the left mouse button release.

You can toggle on/off colors and a sound for the CalcHEP session as well as choose the most appropriate font for the CalcHEP window. Just edit the *comphep.ini* file which appears as a result of user installation (Section 2.5). We hope the syntax is obvious.

1. On-line Help. At almost every point when CalcHEP is waiting for your input, you can press the *F1* key to get a context sensitive help information. If the screen height is not enough to display the full help message, you will see the *PgDn* mark in the right-bottom corner of the help window. To get the next page of the message press the *PgDn* key or click the mark. To close the help window press the *Esc* key or click the asterisk in the top-left corner of the help border.

2. Menu. The menu program displays a list of menu functions. One of them is highlighted. See a typical example of menu in Fig.1. Use the arrow keys or a mouse click to highlight a desired function. Press the *Enter* key or click on the highlighted function to activate it.

If the menu is too large you will see only a part of it. Use the *PgDn/PgUp* buttons or click on the corresponding marks in the menu corners to scroll.

In order to get back to the previous menu level press the *Esc* key or click the asterisk in the top-left corner of the menu border.

The menu program is also sensitive to the functional keys *F1, F2, ..,F9*. The list of active functional keys depends on the program point and is displayed on the bottom line of the screen. Generally the functional keys activate the following programs:

[4] Use the `'Ctrl ['` sequence if the *Esc* key is absent on your keyboard

[5] On some keyboards there are *Prev/Next* instead of *PgUp/PgDn*

F1- Help	: displays a help message about the highlighted menu function;
F2- Manual	: displays an information about service facilities.
F3- Models	: displays contents of the current model of particle interactions.
F4- Diagrams	: browses the generated Feynman diagrams.
F6- Results	: views and deletes CalcHEP output files.
F9- Quit	: quits the CalcHEP session.

To call one of these programs just press the functional key or click on the corresponding symbol on the bottom line of the screen. The digit keys act as the functional keys. For example, *'3'* acts as *F3*.

3. Message. CalcHEP writes informative and dialogue messages during the session. The informative messages finish with the *"Press any key"* string. You can continue your work either following this instruction or clicking the mouse on the message area. In the second case the message has a label *(Y/N?)*. You should press the *Y* or *N* key or just click on them in the message window to answer 'Yes' or 'No'.

4. String Editor. If you would like to enter a new process or change a parameter value, CalcHEP calls the String Editor. As a rule, previous information about this item is available and the kept string is displayed. If you would like to edit the original string, use the *left/right arrow* keys or the mouse click to put the cursor on the desired position. Otherwise, if the fist input character is a printing symbol, the original string will be deleted. The *Delete* key works as the *Backspace* key and removes a character left to the cursor. To terminate the input you can press *Enter* to accept the resulting string or the *Esc* key to cancel it.

5. Table Editor. CalcHEP uses *tables* to store the information about parameters, particles, vertices, cuts and distributions. For all these cases any unit (displayed as a table line) consists of several fields (table columns). The program Table Editor is invented to provide the user a possibility to view and change the table contents. In some program points the Table Editor is used to browse a table contents without a permission to change data.

Table is displayed on the screen as it follows (see Fig.2). The top line of the window contains a title of the table. Below there are table columns surrounded by a frame box. The columns are separated by vertical lines. The first horizontal line contains column names. One cell (a line - column intersection) is highlighted. If the table is open for changes, the highlighted cell contains the cursor. The current line number is displayed in the top-right corner of the window.

To change position of the cursor and the highlighted cell one can use the arrow keys, the *Tab* key and the mouse click. If one types any printing symbol it will be inserted into the table at the cursor position. The *PgUp*, *PgDn* keys are used to scroll the table. The *F1* and *F2* functional keys provide information about the meaning of table fields and about facilities of the Table Editor. To exit the table one has to press the *Esc* key.

There are some auxiliary commands which help the user to operate the tables. These commands can be realized by means of *Control* symbols or by mouse click on the command label displayed on the table border:

Top (^ T) moves the cursor (highlighted cell) to the top line of the table.

Bottom (^ B) moves the cursor (highlighted cell) to the bottom line of the table.

GoTo (^ G) moves the cursor to the line directed by the user.

Find (^ F) searches the string directed by the user.

Zoom (^ Z) key switches on the zoom mode to view/edit contents of the highlighted cell. In this case CalcHEP opens a special window and the field text wraps this window. To terminate the *Zoom* mode one has either to press *Enter* to accept changes or *Esc* to forget them.

ErrMess (^ E) redisplays an error message concerning the contents of one of the tables which has been previously generated by CalcHEP.

The above commands are available in both modes of the Table Editor. The labels of these commands are disposed on the bottom border of the table. The following commands are available only if the table is open for changes:

Clr (^ C) clears the contents of the current field right to the cursor position.

Rest (^ R) restores the contents of the current field which existed before last entering the corresponding cell.

Del (^ D) cuts the current line from the table and put it into the buffer.

New (^ N) creates a new line and fill it with the buffer contents. Also you can press the *Enter* key to create a new line.

Size (^ S) allows the user to change the width of current field. This command is active only if the cursor is disposed in the column whose name is surrounded by the '>','<' symbols.

The labels of these commands are disposed on the top border of the window.

6. Diagram Viewer. This program was designed to display several Feynman diagrams on the screen. The Viewer splits the screen into rectangle cells and puts the diagram images in these cells one by one. One cell is marked by surrounding box frame. The total number of diagrams and the ordinal number of the marked one are displayed in the right-top corner of the screen. See an example in Fig.3.

The number of diagrams which can be displayed simultaneously depends on the window size. If you would like to see more diagrams on one screen, increase the window using the window manager.

You can use the *PgDn/PgUp* keys to scroll diagram set. The *Home/End* keys display the beginning/end of the set. To display a diagram with some ordinal number you should press the *'#'* key and after that the needed number. You can move the position of the surrounding box by the *Arrows* keys or by the mouse click. To finish work with the Diagram Viewer press the *Esc* key.

The labels for the above commands are displayed on the bottom border of the window and you may use the mouse click to activate one of them.

The Diagram Viewer may have some optional functions which depend on the context. The labels for these functions are shown on the top border of the window. One of them generates a file with graphical diagram image in the LaTeX format. Press the *F1* key to get an information about these commands. You may use the mouse click on the label or its first symbol to activate the function.

7. Plot Viewer. This program is designed to display smooth curves and histograms. Examples are presented in Fig.9 and Fig.6.

Being launched the *Plot Viewer* displays a picture and waits for the keyboard signal as it is shown in Fig.9. The program ends with pressing the *Esc* key. If some other key is pressed then the user gets a menu which you can see in Fig.6. This menu allows to change the limits of vertical axis and

its scale. Note that the logarithmic scale is available only if the lower limit is positive and the ratio of upper and lower limits is more then ten. To re-display the plot choose the *'Redraw plot'* option.

The menu also provides the user with a possibility to save a graphical plot image as a LaTeX file and as a numerical table. The name of the corresponding file is displayed on the screen just after the writing. The numerical table created in this way can be displayed on the screen later on by the

```
tab_view < table_file
```

command disposed in the `$CALCHEP` directory. This `tab_view` is just the same *Plot Viewer* but compiled as a stand-along program.

The *'Exit Plot'* menu function completes the *Plot Viewer* session.

3.2 Menu system for symbolic calculation

The scheme of menus for the symbolic calculation session is presented in Fig.4.

3.2.1 Choice of the model and work with it

Menu 1. This menu contains a list of available models. It offers you an option to select a model of elementary particle interaction for subsequent work.

CalcHEP contains the Standard Model and Minimal Super Symmetry Model (MSSM). MSSM written in CalcHEP format by A.Semenov [2]. Both models are presented in two gauges: the t'Hooft-Feynman one and the unitary one. We recommend to choose the t'Hooft-Feynman gauge for calculations because the ultra-violet cancellations between diagrams caused by gauge invariance are absent in this case. See the discussion in Section B. The Standard Model in the unitary gauge is used for verification of gauge invariance.

The bottom menu function provides you with a possibility to include a new model into the CalcHEP list. New model is created as a copy of one of existing models. On the next menu level you can change this copy. If you choose the "New Model" menu function you will be prompted for a new model name and a template source. CalcHEP adds the underscore symbol '_' in front of the name of new model. It serves to distinguish user's models from the built-in ones. To choose a template the list of existing models appears.

There is a possibility of automatic conversion of the Lagrangian written in a compact form in the coordinate space into the CalcHEP table format. It can be performed by means of the *LanHEP* program written by A. Semenov [11]. The input is expressed in terms of complex objects, such as a covariant derivative and a gauge field tensor.

Menu 2. The first function of this menu lets you enter the physical process which you wish to deal with. A format of process specification is explained below. You can also use the context help facility pressing the *F1* key on any step of the input.

Before entering a process you may also edit the model contents by means of the *Edit Model* menu. Later on you will be able only to browse the model contents (by pressing *F3*), but not to change it.

If the currently used model is a user-created one, the menu function *Delete model* removes this model and CalcHEP returns to *Menu 1*. In the case of a built-in model *Delete model* restores the default version of model instead. Before the deletion or restoration the corresponding warning appears and you can cancel the operation.

Menu 3. Information about a model is stored in four tables. They are text files which are disposed in user's *models* directory and can be corrected by an ordinary text editor. But we strongly recommend to use CalcHEP facilities to edit these files because in this case CalcHEP can control possible mistakes of the user input.

CalcHEP displays a menu of model tables. By choosing a position of this menu you can edit the corresponding part of the model.

The *Parameters*, *Constraints*, *Particles* and *Vertices* menu functions let you browse and edit correspondingly:

1. independent parameters of the chosen model;
2. parameters depending on the basic ones;
3. list of particles and their properties;
4. vertices of interaction.

See Section 4.1 for the format of these tables and also Section 3.1 for the explanation of facilities of the table editor.

CalcHEP verifies the model when you try to leave this menu after some changes made in one of the tables. If some error is detected the corresponding message appears and no exit from the menu occurs. This message contains the diagnostics, the table name, and the number of line where the error has been detected. You can recall this message later on within the table editor by pressing the *Ctrl E* key.

The check stops when the first error is detected. You can fix the error and try to leave the *"Edit Model"* menu once more.

When you enter the *"Edit Model"* menu the current version of the model is saved and you have a possibility to return to this version forgetting your corrections. Just answer *N* the question

Save correction ?
Y/N?

which appears every time when you try to leave this menu after correcting the model.

There is a sequence of points which are being checked:

- correctness of identifiers and numbers;
- declaration of an identifier before its use in the expression;
- declaration of a particle before its use in the vertex;
- correctness of algebraic expressions;
- compatibility of Lorentz indices;
- existence of the conjugated vertex.

See Section 4.1 for a full list of requirements on the model.

3.2.2 Input of the process

After activating the *"Enter process"* function of *Menu 3* the list of particles together with their notation conventions is displayed. The notation of anti-particle is shown in parentheses after that of particle. In the case of the Standard Model the corresponding screen is shown in Fig.5. If the list is too long one may use the *PgUp* and *PgDn* buttons to scroll it.

In the bottom part of the screen the prompt '*Enter process:*' appears. The syntax for the input is:

`P1[,P2] -> P3,P4 [,...,[N*x]]`

These 'P1'..'P4' are particle names, N is a number of inclusive particles. The sets of *in*- and *out*-particles are separated by the arrow `'->'` formed of two printing symbols. The particles inside of each set are separated by commas. The total number of particles should not exceed 6.

For example, the input u, U `->` G, G denotes the process of annihilation of the u-quark and $\bar{u}$-quark into two gluons. The input

`u,U -> G,G,2*x`

is a request to construct all processes of annihilation of the u-quark and $\bar{u}$-quark into two gluons accompanied by two arbitrary particles.

If the program finds an unknown name among the in-particles or out-particles it will try to consider it as a name of composite particle and will ask you about its parton contents. For instance, after the input

`e,p -> 3*x`

the question appears:

`Is 'p' a composite particle Y/N ?`

If you choose *'Y'*, you will be prompted to specify the parton structure of *'p'*. A possible input is

`'p' consists of:` u, U, d, D, G

If your input contains x-particles, then on the next step you can restrict these particles. The input should be

`Exclude X-particles :` $P1 > n1$ `[`$, P2 > n2$ `...]`

where $P1, P2, ...$ are particle names, $n1, n2, ..$ are the quantity limits. Such input means that diagrams where the number of particles of sort P_i among X-particles exceeds n_i will not be constructed. The input "$P > 0$" may be shortened to the "P". It forbids the appearance of addition P particles.

On the next step of input you are prompted to exclude diagrams with specified virtual particles. The input should be

`Exclude diagrams with :` $P1 > n1$ `[`$, P2 > n2$ `...]`

where $P1, P2, ...$ are particle names, $n1, n2, ..$ are the quantity limits. Such an input means that diagrams where the number of virtual particles P_i is more than n_i will not be constructed. For example:

`Exclude diagrams with :` $W+ > 1$

means that only those diagrams will be generated which contain less than two virtual W-bosons. If one would like to forbid the appearance of some

virtual particle P at all, the input "$P > 0$" may be shortened to the "P".

This option may be used to exclude diagrams which are suppressed due to a large virtual particle mass, or a small coupling constant, or for some other reasons. Use the empty input to get a full set of diagrams.

Use the *Esc* key to return to the previous level of input and the *F1* key to get the online help.

After the input is completed CalcHEP starts the Feynman diagram generation. If the number of generated diagrams is zero the corresponding warning appears and you return to the beginning of process input, otherwise the next menu appears.

3.2.3 Squaring of diagrams and symbolic calculation

Menu 5. This menu appears on the screen just after construction of Feynman diagrams and together with the information about numbers of diagrams and subprocesses generated.

The *"View diagrams"* function gives you a tool to view a graphic representation of generated Feynman diagrams, to remove some diagrams before the squaring and to create the LaTeX output for undeleted diagrams.

If a few subprocesses have been generated then the subprocess menu appears after an activation of the *"View diagrams"* menu function. There is a possibility to remove all diagrams in the highlighted subprocess by pressing the *F7* button. In its turn the *F8* key restores all diagrams of the highlighted subprocess deleted before.

When you choose a subprocess the diagram graphic viewer is launched. See Section 3.1 for details or use the $F1$ and $F2$ functional keys for online help.

Here we would like to note some peculiarities of constructed diagrams.

- Incoming particles are drawn on the left side of diagrams, whereas the outgoing ones are shown on the right.
- Particles with spins $0, \frac{1}{2}, 1$ are represented by the dotted, solid, and dashed line correspondingly.
- Charged particles are represented by arrow lines. The arrow indicates the direction of particle (not anti-particle) propagation.

- Incoming and outgoing particles are labeled by relevant names in the end of the line. Virtual particles are labeled by their names at the mid-line. If a particle is not neutral, the particle's name is used for the labeling (not the anti-particle's one).

- In the case of collision process a first colliding particle is disposed in the top part of picture, whereas the second one is drawn in the bottom part.

- CalcHEP produces only one representative of a set of diagrams which can be transformed one to another by replacing the identical outgoing particles. For example, CalcHEP creates only one diagram for the $e1, E1$ -> A, A process, whereas all textbooks present two diagrams in this case. In other words, momenta are not assigned yet to the outgoing particles on the step of diagrams generation.

- CalcHEP does not generate diagrams with Faddeev-Popov and Goldstone ghosts[6]. Such diagrams are restored on the step of squared diagram evaluation. Any ghost has a real particle as a prototype (see Section 4.1.4). As a rule, a diagram with ghosts has got the parent diagram where ghosts are replaced by real particles. During the calculation of these parent diagrams the contributions of the corresponding child diagrams are also calculated and added to the contribution of the parent one (see Section B). There exist some exceptions from this rule. For example, the Standard Model contains the vertex with four Goldstone bosons associated with the Z-boson, however the Z^4 vertex is absent in the theory. To provide a possibility to take into account the child diagrams with the four-Goldstone vertex CalcHEP generates false diagrams with Z^4 interaction.

- Vertices with a complicated color structure, for example, the four-gluon one, are implemented by means of nonphysical tensor field. This field is treated by CalcHEP as a special gluon ghost and does not appear in the constructed diagrams. A contribution of the four-gluon vertex is restored when squared diagrams are evaluated, in the same manner as the contribution of Faddeev-Popov and Goldstone ghosts. See Sections 4.1.4, 4.1.5, 4.2 for further explanations.

[6]In the following text we use *Goldstone ghost* instead of commonly used *Goldstone boson*. For our convenience we name both of these two kinds of fields and the auxiliary tensor field as *ghosts*.

The second function of this menu (*Squaring*) is the instruction for CalcHEP to create squared diagrams. CalcHEP uses the squared diagram technique for evaluation of squared matrix elements. See Section B for details.

Menu 6. The *"View squared diagrams"* function is similar to the *"View diagram"* one of the previous menu but is applied to the set of squared diagrams. Each squared diagram is a graphic representation of AB^* contribution in the squared matrix element, where A and B are some amplitudes corresponding to Feynman diagrams.

Let us summarize some features of CalcHEP squared diagrams generation:

- CalcHEP never constructs both AB^* and BA^* diagrams but only one of them. Instead, for simplicity, CalcHEP calculates $2 * Re(AB^*)$ on the symbolic and numerical levels.

- CalcHEP constructs only one representative of a set of diagrams which can be transformed one to another by permutation of identical outgoing particles. The needed symmetrization for such particles is performed later on, at the step of numerical or symbolic summation.

- Each CalcHEP diagram represents a set of squared diagrams where some physical particles are replaced by their ghosts in all possible ways according to the existing vertices of the model. To see the ghost squared diagrams for each displayed one has to press the *'G'* key.

If you browse the squared diagrams after usage of the *"Symbolic calculation"* function you will see that each of the squared diagrams is marked by one of the following labels *CALC, ZERO, Out of memory, Del.* They mean that the diagram has been successfully calculated; gives a zero contribution; cannot be calculated; or has been deleted, correspondingly.

The *"Symbolic calculation"* function starts symbolic evaluation of the generated squared diagrams. This evaluation is performed by the built-in symbolic calculator.

During the calculation the information about a number of calculated diagrams and about a step of evaluation of the current diagram is shown.

The *"Reduce program"* function creates a code of symbolic evaluation in the format of *Reduce* language [24] separately for each squared diagram. These codes are not used for the further CalcHEP processing, but they can be useful for cross-checking the CalcHEP software. On one hand, one can investigate the *Reduce* code to get conviction that it correctly calculates a contribution of the squared diagram. On the other hand, you may compare the result of the *Reduce* evaluation of diagram code with the result of built-in symbolic calculator. There are some tools created for this purpose. See Section A for details.

General purpose of a package like CalcHEP is to create the corresponding *C* source code for further numerical processing and compile this code using the corresponding system facilities. User's control is not necessary for this step as well as for the step of symbolic calculations. CalcHEP provides the user with a possibility to perform the above steps in a non-interactive mode. To start the non-interactive session one could activate the *"Make n_calchep"* menu function. Then the current interactive session ends and a new batch process starts. See Section 3.4 for details of the non-interactive calculations in CalcHEP.

As the outcome, the executable file *n_calchep* is created for the further numerical processing. You may find it in the *results* directory. For the time of batch calculation the *LOCK* file appears in the user working directory to prevent a double start of the CalcHEP package within the same place of the file system.

As a rule the System Administrator requires to start a large job in a special batch regime. The execution of one of the *"Make n_calchep"* functions may require large computer resources for some processes with large number of diagrams. In this case you could correct your *calchep* command file according to the System Administrator rules. The corresponding calls are disposed after labels 24) and 25) in this file.

3.2.4 Output of results and launching of the numerical calculation

Menu 7. The *"View squared diagrams"* function is identical to the same function of the previous menu. Following a logic of the program you can reach this menu only if all squared diagrams are marked either as deleted or as calculated. If you undelete some diagrams then you will be returned to the previous menu.

The *"Write results"* function lets you save the results of symbolic calculation in terms of one of computer languages for further numerical or symbolic evaluation. See *Menu 8* for details. The files are saved in the user's *results* sub-directory.

The next menu function realizes numerical calculations in CalcHEP, particularly Monte Carlo sessions. It calls the corresponding compiler to create and launch an executable file *n_calchep* in the directory *results*. It is assumed that the corresponding source code was created earlier using the *"Write results"* menu function.

Menu 8. This menu provides the user a possibility to save the obtained symbolic results in different formats. The *Reduce* and *Mathematica* outputs may be used for further symbolic manipulation. For example, you can get symbolic representation of a sum of all diagrams, perform some substitutions or expansions, evaluate the symbolic expression for the total cross-section or width of the process. As a rule such manipulations are possible and reasonable for the processes with a small number of final particles. The structure of the symbolic output and some programs for manipulation with the *Reduce* output are described in Section 5.

Otherwise the *C* output is used to get numerical results. They may be compiled and linked with the library of CalcHEP routines for phase space integration. The produced executable program allows one to obtain numerical values of the total cross-section or width, distributions and an event flow, taking into account a variety of cuts. In order to create an executable file for numerical calculations based on the C output use functions of *Menu 6* or launch the `$CALCHEP/make__n_calchep` program as it has been explained in Section 3.5.

3.2.5 Some options

There are some switches which influence on the work of symbolic calculator and C-output.

- Conservation low *ON/OFF*. If the switch is "ON" then energy-momentum conservation low for incoming and outgoing particles is substituted at the level of symbolic calculation. Usually, but not always, it leads to more compact expressions in comparing with the "OFF" setting. In the time of numerical calculation the conservation low is satisfied anyway.

- Nc = *3/inf*. This switch has two positions, "3" and "inf". "3" corresponds to normal SU(3) chromodynamic. If the switch is in the "inf" position then symbolic calculation occur for $SU(N_c)$ color group, only the leading term of $1/N_c$ expansion is saved, and $N_c = 3$ is substituted into the final answer. This trick allows to remove a lot of interference diagram, which are zero in the leading term of $1/N_c$ expansion.

- color chains *ON/OFF*. CalcHEP has event generator which generates events and color flows corresponding to color particles. This option allows to switch off generation of the color flows.

- diagram in C-code *ON/OFF*. CalcHEP attaches to the C-code of diagram its pseudo-graphic image. In order to decrease the size of the code the uses can switch off this option. Usually the size of diagram image is small in comparing with the code size.

The default position of a switch is the first presented one. Press the *F3* functional key to obtain the menu which allows to change the switch positions.

3.3 Numerical calculation by CalcHEP

3.3.1 Sketch of the menu system

A general scheme of CalcHEP numerical calculations is presented in Fig.7. The CalcHEP screen with the *Menu 1* is presented in Figs. 1. Two functions of this menu launch integration programs. One of them is Monte Carlo program *Vegas* [22, 23]. The other one is the ordinary *Simpson* integration routine which can operate only for the 2 -> 2 type of reactions. Also there is a function which launches event generation.

Other menu items are needed to set an environment for integration and event generation. Below we briefly describe all of them. More detailed explanation is given in the next sections.

By means of the *Subprocess* function the user may select one subprocess for a further processing. This function is active only if several subprocesses have been evaluated symbolically before. The name of the current subprocess is displayed on the screen (Fig.1). Subprocess summation is not implemented in the current version.

The *In_state* function gives the user an option to choose energy distributions for incoming particles and change values for their momenta.

The *"Model parameters"* function allows the user to assign a new value to any independent parameter involved in the evaluation. After an activation of this function the corresponding menu of parameters with information about their current values is displayed. As a result of changing the independent parameters some constrained parameter can get non-evaluated as a result of a negative square root argument or division by zero in the relevant expressions. In this case an error message appears on the screen and the user is required to choose an admissible set of parameter values.

Although the user can assign some value to the QCD strong coupling parameter *GG* it does not influence on results of calculation. This parameter will be reset according to the rules defined by the *"QCD coupling"* menu function.

There is an alternative way to change values of parameters. It is realized via *F5* functional key. Here the user is asked about the name of parameter he would like to change and after that he is prompted to enter a new value[7].

CalcHEP substitutes a Breit-Wigner exact propagator for virtual particles. This trick is needed to avoid the integral divergence near the mass pole and is motivated by high order corrections. The manner of implementation of the Breit-Wigner propagator is driven by the *"Breit-Wigner"* menu function. See Section (3.3.4) for details.

The *Cuts* menu function provides the user with a possibility to cut a phase space volume. A set of physical variables which could be used to construct a cut function is described in Section 3.3.5. Details of the *Cuts* menu processing are given in Section 3.3.7.

The *Kinematics* and *Regularization* menu functions are used to construct an appropriate mapping of phase space onto the *Vegas* integration volume. This is important for successful Monte Carlo integration and event generation.

3.3.2 Numeration of Monte Carlo sessions

CalcHEP numerical calculations are organized as a sequence of Monte Carlo integration sessions. Each session has its ordinal number. The current session number is displayed. The session number is increased automatically if one changes subprocess, momenta of incoming particles, distribution functions,

[7]This option is implemented for the purposes of batch calculation. See section (3.4).

QCD coupling, cuts.

All results of calculations are written into the *results* directory. To distinguish files produced in different sessions we include a session number N in the output file names. The results of integration and all settings are stored in the protocol file *prt_*`N`. Events generated by *n_comphep* are stacked in the file *events_*`N`.

The latest settings are saved in the file *session.dat* when *n_comphep* is quitted. On a next launch this file is scanned to restore the last settings.

3.3.3 QCD coupling

CalcHEP substitutes the running strong coupling $\alpha(Q)$ into the matrix element . Q in its turn can depend on the total energy of reaction or phase space point. The scale Q should be defined as some algebraic expression which can contain float point numbers, phase space functions listed in the (3.3.5) section, standard C float point functions like *sqrt(x)* and *fabs(x)*. The same Q scale is used as an argument of parton distribution functions.

The $\alpha(Q)$ function is calculated in the NLLO approximation and depends on a number of quark flavors (*nf*) with masses less than Q and the corresponding $\Lambda^{(nf)}$ parameters (see Eq.9.5a of [27], p.85).

$\Lambda^{(5)}$ is defined by the user. Relevant $\Lambda^{(4)}$ and $\Lambda^{(6)}$ are recalculated from $\Lambda^{(5)}$ according to the matching equation (Eq.9.7 of [27], p.86).

If one of the active particle distribution functions contains its own definition of $\alpha(Q)$, then $\Lambda^{(5)}$ is ignored. The corresponding message appears in the first menu line. See section (F) for further explanation.

If the user is going to change a numerical value of $\Lambda^{(5)}$ or the Q scale definition he should call the necessary function of *Menu 4*. The $\alpha(Q)$ plot for a user-defined region of s is also available by means of this menu.

3.3.4 Breit-Wigner propagator

The propagator of virtual particle has a pole at $p^2 = m^2$:

$$\frac{1}{p^2 - m^2} \quad .$$

If the pole is situated inside the phase space volume it leads to a non-integrable singularity. The general solution of this problem is an account

of special set of high order corrections [34]. They transform the propagator to the Breit-Wigner form

$$\frac{1}{p^2 - m^2 - i \cdot \Gamma(p^2) \cdot m} \quad ,$$

where the value $\Gamma(m^2)$ is the particle width (reversed mean life time).

First problem which appears in the way of implementation of this expression is a choice of $\Gamma(p^2)$ dependence. The Γ value is essential near the pole point $p^2 = m^2$. Thus, for the first approximation we can put $\Gamma(p^2) = \Gamma(m^2) = const$. It corresponds to the position *OFF* of the *S dependence"* switch. In some papers it is declared that $\Gamma(p^2) = \Gamma(m^2) \cdot \sqrt{p^2}/m$ describes the pole shape better. This choice corresponds to the position *ON* of the *"S dependence"* switch.

The second and even more important problem is a gauge symmetry breaking. Generally we have this symmetry in any order of perturbation theory but the intervention of a part of higher order terms to the lowest order expression via the Breit-Wigner propagator can break it.

The gauge symmetry is responsible for some cancellation of diagram contributions (see Section B.5), and its violation, in turn, prevents the cancellations and can lead to a completely wrong result. The user could solve this problem by setting the *"Gauge invariance"* menu switch to the position ON. In this case the contribution of a diagram which does not contain the Breit-Wigner propagator is multiplied by factor [28, 29]

$$\frac{(p^2 - m^2)^2}{(p^2 - m^2)^2 + (width(p^2) \cdot m)^2} \quad . \tag{1}$$

This trick corresponds to the symbolic summation of all diagram contributions at a common denominator expression and to a subsequent substitution of the width term into the factored denominator. The trick allows to keep all gauge-motivated cancellations. As a defect of the trick it must be mentioned that the factor (1) kills a contribution of non-resonant diagrams in the resonance point [30]. If the particle width is very small such an approximation is reasonable, but in some cases it can also lead to an error.

As a rule it is enough to substitute widths only to s-channel propagators. There is a menu option *"t-channel widths"* which should be switched OFF to prevent implementation of Breit-Wigner formula to t-channel propagators.

3.3.5 Phase space functions

There is a set of phase space functions which can be used to construct cuts and distributions in the framework of CalcHEP. The general notation looks like

(Key Character)(momentum set).

A set of momenta is represented as a set of digits. Any digit corresponds to a momentum number. For example, *C13* means a cosine of the angle between momenta $\mathbf{p_1}$ and $\mathbf{p_3}$. The momenta are assigned to particles according to their sequence in the process name.

Below we list the available key characters and explain the meaning of the corresponding physical functions.

A - angle in degree units.

C - cosine of angle.

J - jet cone angle. The jet cone angle J_{ij} is defined as $\sqrt{\Delta y^2 + \Delta\varphi^2}$, where Δy is the pseudo-rapidity difference and $\Delta\varphi$ is the azimuth angle difference for momenta $\mathbf{p_i}$ and $\mathbf{p_j}$.

E - energy of the particle set.

M - mass of the particle set.

P - cosine in the rest frame of pair. *Pij* is defined as follows: we imply a boost in the direction of $\mathbf{p_i} + \mathbf{p_j}$ to get the rest frame of the pair. Then P is a cosine of angle between the transformed $\mathbf{p_i}$ and the direction of boost.

T - transverse momentum of the particle set.

S - squared invariant mass of the particle set.

Y - rapidity of the particle set.

N - pseudo-rapidity of the particle set.

U - user's implemented function. The character string following U is passed on to the function `usrfun(char * str)` which, as it is assumed, calculates a corresponding value.

See next section for further explanations.

Zero components of all momenta are positive, and the conservation law means that the sum of momenta of incoming particles is equal to that of outgoing particles. Relative momenta signs for the S function are substituted automatically.

There are some natural requirements on the momentum set, namely: a) the momentum sets following *A, C, J, P* must consist of two elements; b) the momenta following *M, P, T* must belong to outgoing particles only; c) any momentum can appear only once in one set.

3.3.6 User functions

The user has a possibility to attach his own functions to the set of phase space functions. If some cut or histogram uses a phase space function which is started from the character 'U' then all characters following 'U' are passed to the "`usrfun`" program to evaluate the corresponding value. Say "Ua1b2" leads to the call `usrfun("a1b2")`.

The patch version of `usrfun()` is presented in the CalcHEP code. But if the user writes his version of `usrfun()` and puts the corresponding object file *usrfun.o* on his WORK directory then the linker attaches the user's version to the *n_calchep* code.

The user phase space function has a prototype "`double usrfun(char*str)`". Particle momenta which are needed for evaluation are stored in

```
extern double pvect[];
```

The m^{th} component of k^{th} momentum occupies the $m + 4 * (k - 1)$ position in `pvect`. It is assumed that 'm' varies from 0 to 3 and the 'k' momentum counter starts from 1. The 0^{th} components of momenta are positive, and sum of incoming 4-vectors is equal to sum of outgoing 4-vectors.

The number of incoming and outgoing particles and the correspondence between the particle name and momenta can be determined by means of service functions described in Section 5.4.

3.3.7 Cuts

The *Cuts* menu function opens a table for cuts definition. This table consists of three columns. The phase space function in notations of Section 3.3.5

must be written in the first column. The second and the third columns define the minimal and maximal limits for this function. If one of them is empty the corresponding limit is not applied. The limit may be presented by an algebraic formulas which contain numbers and identifiers enumerated in the *"Model parameters"* and *Constrains* menus. Parentheses "()" and operations "+,-,/,*,^ ,sqrt()" also are permitted.

It is very important to note that only the cuts on masses and squared momenta (the *S* and *M* cases) are realized via a reduction of the phase space. Other ones are realized by multiplication of squared matrix element by a step function.

3.3.8 Kinematics

The *Kinematics* function allows to display and change a general scheme of the phase space parameterization. A kinematics scheme in CalcHEP is defined by a set of subsequent 1 -> 2 decays of the incoming state into the outgoing particle sets.

After activation of this menu function the current decay scheme will be displayed and a dialogue message will propose the user to accept it or put in another one.

During the input CalcHEP consequently asks the user to define a first outgoing cluster of virtual decay. The user input should be a sequence of particle (momentum) numbers without any separating symbols. It is assumed that particles are numerated in the same order as they are written down in the (sub)process name.

An example of kinematics definition for 2 -> 4 process is given by the following lines:

12 -> **3**,456
456 -> **56**,3
56 -> 5,6

Here the user input is indicated in bold characters.

In the framework of such a decay scheme the multi-particle phase space is parameterized by masses of sub-clusters and by two-dimensional spherical angles of 1 -> 2 decays [39, 31]. A choice of kinematics influences on the phase space mapping and by this way it influences on convergence of Monte Carlo integration. See Section G.1.

3.3.9 Regularization

In general case a squared matrix element is too singular for direct Monte Carlo integration. Singularities of the matrix element are caused by poles of virtual particle propagators and can have one of the following forms

$$1/(p^2 - m^2) \tag{2}$$
$$1/(p^2 - m^2)^2 \tag{3}$$
$$1/((p^2 - m^2)^2 + (m \cdot \Gamma)^2) \tag{4}$$

where m, Γ, and p are the mass, width, and momentum of virtual particle.

The *Regularization* menu function allows the user to point out dangerous denominators for automatic smoothing the sharp peaks of the squared matrix element. The regularization table contains four fields: *momenta, mass, width* and *power*.

Momentum of virtual particle is a sum of momenta of incoming and outgoing particles. Just type the ordering numbers of these momenta in the *Momentum* field. The sign is substituted automatically. For example, for a collision process **12** is treated as $(p1+p2)$ and **134** is treated as $(p1-p3-p4)$.

Mass and *Width* describe a position of the pole. The user can write down some algebraic expressions which contain the identifiers enumerated in the *"Model parameters"* and *Constrains* menus.

The *Power* field defines an exponent of the propagator. Acceptable values are 1 and 2. Of course, in a squared matrix element any propagator appears to the power of 2. But sometimes as a result of gauge cancellations the exponent can be efficiently decreased to 1. If the *Width* field is not equal to 0 such a cancellation is not expected and CalcHEP will use value 2 for the exponent ignoring the user input.

The work of regularization program is sensitive to the *S* and *M* types of cuts (see Section 3.3.7) and is not sensitive to other ones. Consequently, if you would like to smooth some singularity due to the pole inside the phase space, you should apply the *S* or *M* cut to exclude the pole point from the consideration.

The algorithm of regularization is explained in Section G.1.

3.3.10 VEGAS menu

We use Vegas for Monte Carlo integration of the squared matrix element. See Section G.2 or [22, 23] for explanation of the *Vegas* algorithm. The work of the *Vegas* program is driven by *Menu 6* of Fig.7. Vegas is launched by the *"Start integration"* menu function. It performs *nSteps* runs with subsequent adaptation of the Monte Carlo weight function to the integrand. For each run *Vegas* calls the integrand *nCalls* times. The user can set the *nSteps* and *nCalls* parameters applying the first two items of *Menu 6*. The results of calculations are displayed on the screen and copied to the protocol file *results/prt_#*, where # is a session number.

The user has a possibility to fill in some histograms during the Monte Carlo integration. In this case he needs to activate the *"Set Distribution"* function. This menu function opens a table of distributions which can be filled in by the user. The first column defines a phase space function which the user would like to scan. The set of available functions and the input format are defined in Section 3.3.5. The next two columns define a region of variation of the corresponding phase space function. They can be algebraic expressions containing numbers and the identifiers listed in the *"Model parameters"* and *Constrains* menus. After the following *"Start integration"* the user can browse the distributions using the *"Display distributions"* menu function. This menu function asks the user to choose one distribution and a number of bins. After that the distribution appears.

If the required precision of calculation is not reached, the user can call *"Start integration"* once more to increase the Monte Carlo statistics. In general, all statistics for the integral and histograms and also the improved Vegas grid are saved and updated by next Vegas session. Even if the user quits *n_calchep*, this information is stored in the *session.dat* file and will be available on the next *n_calchep* session.

All statistics are cleaned automatically if the user changes the *"In state"*, one of model parameter, the *QCD* settings or the *Breit-Wigner* switches. By the other words, when the physical problem is changed. The corresponding message appears on the screen. Also one can use the *"Clear statistics"* menu function to start Monte Carlo from the beginning. It can be useful, because the results obtained during the first iterations, when the grid does not corresponds to the integrand, can be wrong.

CalcHEP automatically clears the grid if the user changes particle distribution functions, *Cuts* or *Kinematics*. Also the user can do it explicitly by means of the *"Clear grid"* menu function.

3.3.11 Event generation

The procedure of event generation consists of two steps. The first step is a preparation (initialization) of generator (*Menu 7*). The second step is itself a generation of events (*Menu 8*). Note that the efficiency of generator should be better if the user preliminary launches VEGAS to construct an appropriate grid on the phase space. CalcHEP generates the partonic level events. The task of hadronization is passed to other programs like PYTHIA [32].

See sections (G.3, D) for explanations of the used algorithms and the corresponding references. The interface with PYTHIA is described in [33].

Preparation of generator. On this step we divide the integration volume on several sub-cubes and find the maximum of the distribution function in each sub-cube. The user has to define the number of *Sub-cubes*, numbers of the calls of the function for the *Random search* of maximum and number of steps for the search by the *Simplex* method. After that the procedure of preparation of generator can be started.

The larger number of *Sub-cubes*, the more efficiency of the generator, but the more time for preparation of generator is needed. The preliminary Vegas run provides the user an estimation of the needed time.

The preparation of generator is finished by the message which gives estimation of efficiency of the generator prepared.

Work of generator. Before launching the generator the user has to specify the needed *Number of events*. In order to reduce the number of multiple events one can multiply the detected maxima by some factor. It should be done by means the second function of *Menu 8*. If in spite of this factor CalcHEP meets a point x where the distribution function exceeds the estimation for the maximum, them the multiple event is generated but the estimation for the maximum is increased for the future generation. In this point the program can look for a new maximum by means of the simplex method starting from the point x. The number of steps is defined by the third menu function. One can set this number zero to prevent this search.

When the needed number of events are generated, CalcHEP writes on the screen the message which informs the user about the efficiency of the generator and the number of multiple events. The user can accept the generated set of events or refuse them. We recommend in the beginning to test the generator specifying a request for generation of a small number of events.

The generator improves itself during each run by means of the improving the maxima estimation in sub-cubes. If nevertheless generator works badly, the user has to return to *Menu 7*.

The generated events are written in the text format into the file *results/events#.txt* where # means the session number. Format of this file is described in the section (5.5).

3.3.12 Integration by means of the Simpson method

In the case of `2->2` type of process we provide the user with a possibility to calculate the cross-section using the adaptive Simpson algorithm. It allows to get the results instantly and with a high precision.

Note that some user's settings are ignored when you choose this option. Namely, the structure functions are switched off, the center-of-mass rapidity is assigned to zero, regularizations are also ignored and only angle and cosine cuts for incoming and outgoing particles are taken into account.

When the *Simpson* menu function is activated CalcHEP tries to evaluate the cross-section. Sometimes it is not possible, for example, if the corresponding value is infinite, as we have in the case of $e^+, e^+ \to e^+, e^+$ reaction. When program detects a pole inside of the integration region it produces an error message like

Division by zero Press any key

In the above case you should remove the region of small angle scattering by applying the corresponding cuts and after that activate the *Simpson* function once more.

If the cross-section is successfully integrated the result is displayed on the screen. *Menu 6* which is displayed on the screen in the same time (see Fig.7) has the following options:

a) to set relative precision of integral evaluation. A default value is 10^{-4};

b) to display dependence of the differential cross-section on the cosine of the scattering angle;

c) to calculate and display on the screen the total cross-section as a function of P_{cms} or any other parameter involved in the calculation.

3.3.13 Process of two particle decay

In the case of `1->2` decay process the menu scheme described above is not used because in this simple case the phase space integration is not needed. CalcHEP writes on the screen the calculated particle width. If a multichannel process like `Z->2*x` has been introduced then branchings for all decay modes are displayed (Fig.8). The menu on the screen contains only one function which provides the user with a possibility to calculate the total particle width as a function of one of the model parameters.

3.4 Non-interactive sessions

3.4.1 General concept

CalcHEP was created for calculation in the interactive regime. But long-time calculations should be done in non-interactive, batch, mode. If one launches `s(n)_calchep` with an option "-blind", say

`s_calchep -blind STRING`

then `s(n)_calchep` does not open the X11 window and reads the control characters from *STRING*. To imitate the functional keys the following control characters are defined:

'{': Escape key
']': Right Arrow
'[': Left Arrow key
'}': Enter key

Note, that CalcHEP menus are sensitive to the *Right Arrow* and *Left Arrow* keys in the same manner as to *Up Arrow* and *Down Arrow* keys. So, the user has a possibility to concatenate into *STRING* a list of commands which simulate the keyboard and execute `s(n)_calchep` in the non-interactive mode.

The work in the blind mode is simplified a little. It automatically passes by all warnings and messages answering 'Y' (Yes) any question itself. The s_calchep title screen is missed in this mode and the simulation of keyboard hit, which is used to leave the first screen, is not needed.

An example of the blind session is realized in your *calchep* command file. When you activate the *"Make n_calchep_c"* function of *Menu 6*, s_calchep ends with an exit code 24 and

```
s_calchep -blind "]}}]]]}9"
```

is launched. The above command performs symbolic calculations and writes down the C-code for the squared matrix element. Note that the last symbol '9' simulates the work of the *F9* functional key which quits the session (See Section 3.1).

Such approach allows to create various batch tasks without an intervention into the C code. From the other side, it is difficult to code the correct sequence of commands in *STRING* because debugging tools are absent. Directory calchep_2.1 contains several examples of implementation of batch tasks. They should cover the main part of user requests and can be modified to satisfy the rest. See below.

3.4.2 Batches for symbolic calculations

The command

s_blind *Nmodel Process Noutput*

performs symbolic calculation of process *Process* in framework of the model *Nmodel* and writes results according to parameter *Noutput*. Here *Nmodel* is the ordering number of model in the CalcHEP list (Menu 1). The *Noutput* parameter should corresponds to Menu 8. *Process* must be written without spaces and be enclosed into quotation marks. For example,

```
s_blind 1 "e,E->m,M" 1
```

performs symbolic calculation in the framework of the Standard Model and writes results in Reduce notations. s_blind should be launched from user *WORK* directory.

3.4.3 Batches for numerical calculations

These scripts should be launched from the *WORK/results* directory. They call *./n_calchep* with the *-blind* option.

- set_momenta
  changes momenta of incoming particles. This function needs two arguments, the values of incoming momenta.

- set_param
  changes numerical values of variables. There are two forms of its usage. The first one is

  change_par *file_name*

  The file has to contain two columns, namely, *name* and *value*. Here *name* is a name of parameter and *value* is a value to be assigned. Another form is

  set_param *name value [... name value]*

  If some *name* is wrong or the constrained parameters can not be evaluated after the assignments then change_par return non-zero exit code[8].

- set_vegas *nSess nCalls*
  Sets parameters of the VEGAS session. *nSess* is the number of iterations, *nCalls* is the number of calls for one iteration. See *Menu 6* in Fig.(7).

- run_vegas *[x]*
  launches the VEGAS session. If this program is started with some parameter, then it cleans Monte Carlo statistics of previous sessions.

- prep_gen *sub-cubes Random_search Simplex_search*
  prepares event generator. The parameters correspond to menu 7 Fig.7. See session (3.3).

- gen_events *N_events new_max*
  generates *N* events. Result is written in *events_#* file. The parameters correspond to menu 8 Fig.7. See session (3.3).

- pcm_cycle *pcm0 step N*
  organizes a cycle for calculation of total cross section for different values of particles momenta in the center-of-mass frame. *pcm0* is the initial

[8] Unix exit code is stored in the $? variable

value of momentum, *step* is a step of scanning and N is a number of steps. Result of calculation is stored in *pcm_tab_#1_#2* file where #1 and #2 are the ordering numbers of the first and the last numerical sessions.

- `name_cycle` *name val0 step N*
  organizes a cycle for calculation of total cross section for different values of the *name* parameter. *val0* is the initial value of parameter, *step* is a step of scanning and N is a number of steps. Result of calculation is stored in *name_tab_#1_#2* file where #1 and #2 are the ordering numbers of the first and the last numerical sessions.

3.5 CalcHEP files and commands

3.5.1 Files and commands in the working directory

We shall use an alias name *WORK* for the directory which has been created by the user during the CalcHEP installation (Section 2.5). It contains the following sub-directories and files:

```
models/ results/ tmp/
calchep calchep.ini
```

The *models* directory contains files which describe physical models. When you modify a model or create a new one all changes are kept in this directory.

The directory *results* keeps a CalcHEP output which is produced as a result of symbolic or numerical calculations.

The directory *tmp* is used for temporary files. The file *tmp/safe* keeps user settings between the sessions. If this file exists the symbolic session starts its work with reading this file to restore the user settings of the previous session. A bug presenting in this file could be a reason for a fatal error in the beginning of the CalcHEP session. In the case of such an error we recommend to remove the *tmp/safe* file.

All these directories are necessary for CalcHEP work. Absence of one of them will lead to a fatal error.

You can toggle on/off the color and the sound for a CalcHEP session as well as choose the most appropriate font for the CalcHEP window [9]. To do

[9] The size of window depends on the size of font.

this just edit the *calchep.ini* file. We hope the syntax there is self-explanatory enough.

The command

```
./calchep
```

starts a CalcHEP session. This is a shell script which calls commands disposed in the `$CALCHEP` directory.

3.5.2 Scheme of calls in the CalcHEP session

The command *./calchep* being issued from within the *WORK* directory launches the

```
s_calchep
```

command. The latter is the main CalcHEP program. Prefix *'s_'* denotes symbolic. It performs symbolic calculations and generates the C codes of calculated Feynman diagrams.

Below we describe commands for compilation and numerical module launching. The general scheme of calls can be presented by the following diagram

$$\text{s\_calchep} \rightarrow \text{launch\_n\_calchep} \rightarrow \left| \begin{array}{l} \text{make\_\_n\_calchep} \rightarrow \left| \begin{array}{l} \text{nCompil} \rightarrow 100 \\ \text{ld\_n} \end{array} \right. \\ \text{n\_calchep} \end{array} \right.$$

The binary executable *n_calchep* is created in the *WORK/results* sub-directory as a result of compilation. Other commands are disposed in the `$CALCHEP` directory. They are shell scripts except of binary *s_calchep*. It is assumed that *s_calchep* and *launch_n_calchep* are started from the *WORK* directory, whereas other commands are started from within the *WORK/results* sub-directory.

The programs *nCompil* and *100* compiles the source code produced prior by *s_calchep*. After the compilation a library of functions which evaluate matrix element is created and then the source code and the object files are removed.

Another command, *ld_n*, calls a linker to create the executable file *n_calchep* for numerical job.

The command *make__n_calchep* is used to start *nCompil* and *ld_n* subsequently.

In its turn, *make__n_calchep* is started by *launch_n_calchep* in a special window. In the case of successful compilation the window is closed and just created *n_calchep* is started in another window.

The *s_calchep* command starts *launch_n_calchep* via the functions *C-compiler* of *Menu 6* (Fig.4) and thus the chain of above calls is realized.

If you have prepared the *C* codes you could compile them outside of the symbolic session by the *make__n_calchep* command started from within the *results* directory.

If you would like to modify the compilation procedure, copy *make__n_calchep* into your *WORK/* directory and modify it as you like.. *launch_n_calchep* calls for the user program if it exists.

3.5.3 LOCK files

WORK/calchep creates a file *LOCK* in the *WORK/* directory. Presence of the *LOCK* file prevents a double launching of *s_calchep* from within the same place of the file system. *LOCK* is automatically removed in the end of the *s_calchep* session. In the same manner *launch_n_calchep* creates a file *LOCK* in the *WORK/results* sub-directory.

If your session has been canceled abnormally you have to remove these *LOCK, results/LOCK* files manually before launching the next session.

4 Implementation of models of particle interactions

4.1 Definition of a model in CalcHEP

Description of particle interaction model in CalcHEP consists of four parts. They are *parameters*, *constrains*, *particles*, and *vertices*.

4.1.1 Independent parameters of the model

The Table *Parameters* consists of three fields:

1. *Name* for an identifier of the parameter. It may contain up to 6 characters. The first character must be a letter, others may be either letters or digits;

   CalcHEP identifiers are sensitive to the case of characters. Also names of different identifiers must be different after rewriting them in the low case. For example, the identifier "MZ" is used for the Z-boson mass, thus the forms "Mz", "mZ","mz" are forbidden both for new definitions and for identification of the Z-boson mass.

   There are some reserved names which cannot be used here:

   - `i` is reserved for imaginary unity;
   - `Sqrt2` is reserved for $\sqrt{2}$;
   - `p1,p2,p3,...` are reserved for particle momenta;
   - `m1,...,M1,...` are reserved for Lorentz indices of particles;
   - **G5** is used for the γ^5 Dirac matrix;

2. *Value* for a numerical value of parameter in some power of GeV units;

3. *Comment* for brief description of a parameter.

An asterisk written just before the parameter name forces the appearance of this parameter in numerical sessions even it is not involved in the evaluation [10].

[10] This trick allows to implement some artificial parameters in numerical session. It can be used, for example, to investigate dependence of cross section on a cut.

4.1.2 Constraints between the parameters

The Table *Constraints* consists of three fields:

1. *Name* for the constrained parameter. The requirements for this field are the same as for the name of independent parameter (see above);

2. *Expression* which must be an algebraic formula composed of:

   - integer and float point numbers,
   - identifiers enumerated in the *Parameters* Table,
   - identifiers defined above in this Table,
   - parentheses (), arithmetic operators +, -, /, *, ^ . For raising to a power (^) the second operand must be an integer;
   - standard functions of C mathematic library, namely, $sqrt(x)$, $sin(x)$, $cos(x)$, $tan(x)$, $asin(x)$, $acos(x)$, $atan(x)$, $exp(x)$, $log(x)$, $fabs(x)$, $atan2(x,y)$.
   - function $if(x,y,z)$ which returns y if $x > 0$ and z otherwise.
   - any other function which should be written by the user. These functions must return *double* and have one or several *double* arguments. In order to create the executable file `n_calchep` CalcHEP looks for these functions in library *WORK/modellib.a* and after that in `$CALCHEP`*/modellib.a.*

3. *Comment* for brief description of parameters.

An asterisk written just before the name of constraint forces the appearance of this parameter in numerical sessions even it is not involved in the evaluation [11].

4.1.3 Description of particles

Each row in the *Particles* Table describes a particle – anti-particle pair. The rows consist of 10 fields:

[11]This trick allows to control some physical parameters, say Higg mass, during the numerical session.

1. *Full name* for a full name of particle. Just for clear orientation, not processed anywhere;

2&3. *"A"* and *"Ac"* containing designations of the particle and anti-particle, respectively. Any character is allowed. The name may contain one symbol, two symbols or three symbols started from '~'. For a completely neutral particle the *"A"* and "Ac" fields must be identical.

The symbols presented in these columns designate not only the particles, but also quantum fields which appear in Lagrangian. In the boson case the ant-particle symbol "Ac" corresponds to the Hermitian conjugated field

$$Ac = A^+$$

In the fermion case it corresponds to the C-conjugated field

$$A^c_i = C_{ij}\bar{A}^j \tag{5}$$

where $\bar{A}$ denotes the Dirac conjugated field.

Following to the standard agreement, the field operator acting on the vacuum, creates the corresponding *anti*-particle state.

4. *2*Spin* for a doubled particle spin: *0* for scalar, *1* for spinor and *2* for vector particles. Neutral spinor particle is teated as a Majorana one.

5. *Mass* for a mass identifier or symbol '*0*'. In the first case its value must be defined in the *Parameters* or *Constraints* table. If this field contains zero, then CalcHEP considers this particle as massless;

6. *Width* for a particle decay width. It must contain an identifier defined in the first two tables or '*0*';

7. *Color* for a dimension of the color SU(3) group representation. You have to choose among *1*, *3*, *8*. *Unity* corresponds to a colorless particle. *Three* corresponds to a color triplet (fundamental representation). In this case the anti-particle *"Ac"* is transformed by conjugated $\bar{3}$ representation. *Eight* corresponds to a color octet (adjoint representation);

8. *Aux* for an auxiliary field which allows to modify particle propagators. If the *Aux* field is empty the standard expressions for propagators are substituted:

(a) spin 0 case:

$$< 0|T[A(p_1), A^+(p_2)]|0 >= \Delta_c(p_1, p_2, M) = \frac{\delta(p_1+p_2)}{(2\pi)^4 i(M^2 - p_1^2)} \quad ;$$

(b) spin 1/2 case:

$$< 0|T[A(p_1), \bar{A}(p_2)]|0 >= (\not{p}_1 + M)\, \Delta_c(p_1, p_2, M) \quad ,$$

where

$$\not{p} = p^\mu \gamma_\mu \quad .$$

(c) spin 1 case:

$$\begin{aligned} < 0|T[A^{m_1}(p_1),\ (A^{m_2})^+(p_2)]0 > &= -(g^{m_1 m_2} + p_1^{m_1}\ p_2^{m_2}/M^2) \\ &\times \ \Delta_c(p_1, p_2, M) \quad . \end{aligned} \tag{6}$$

Zero mass vector particle must be marked as a *gauge* bosons. See below.

Possible objects for the *Aux* field are:

'l','L' is permitted for massless fermions (2*spin = 1) only. The propagator is changed to

$$\frac{\not{p}_1(1+\gamma_5)}{2}\, \Delta_c(p_1, p_2, M) \quad .$$

This is a way to introduce a left-handed fermion as a neutrino or a massless polarized fermion. When CalcHEP performs the averaging over incoming particle polarizations it takes into account that there is only one polarization state;

'r','R' is permitted for massless fermion (2*spin=1) particle only. They change the propagator for

$$\frac{\not{p}_1(1-\gamma_5)}{2}\, \Delta_c(p_1, p_2, M) \quad .$$

This is a way to introduce right-handed fermions;

''* is permitted for massive particle only. In this case $\Delta_c(p_1, p_2, M)$ is replaced to

$$\frac{\delta(p_1+p_2)}{(2\pi)^4 i\ M^2} \ .$$

In the case of vector particle we also remove $p_1^{m1} p_2^{m2}$ term in the propagator numerator (6).

Such particles cannot appear as incoming or outgoing ones. They are used to describe a point-like interactions, for example, the 4-fermion interaction in old fashioned electroweak model;

'G','g' is permitted for vector (2*spin=2) particle. In this case the propagator of vector particle accepts the Feynman form

$$-g^{m_1 m_2}\, \Delta_c(p_1, p_2, M) \quad ;$$

Note, a massless vector particle must be defined as a gauge one.

9&10. *LaTeX(A)* and *LaTeX(A+)* for particle and anti-particle designations in the LaTeX format. They are substituted in the LaTeX image of Feynman diagrams generated by CalcHEP. The names are used in the mathematical mode.

4.1.4 Ghost fields in CalcHEP

CalcHEP constructs a list of quantum fields according to the Table of particles described above. Besides of names enumerated in this table CalcHEP generates auxiliary fields, for example, the Faddeev-Popov ghosts [34]. We use the name *ghost* for all of them.

The ghost fields do not correspond to physical degrees of freedom, but each of them has some real particle as a parent. The names of ghost fields are constructed by CalcHEP from the parent particle name followed by a suffix which specify a type of ghost. The particle name and the suffix are separated by the dot symbol. It is assumed that the ghost fields explicitly appear in the vertices of interactions together with real particle fields and thus contribute to the particle interaction. Below we list CalcHEP ghost fields.

Faddeev-Popov ghost and anti-ghost. They are generated for any gauge vector particle, in other words, for those particles which have the mark *'g'* in the *'Aux'* column of the *Particles* Table described in the previous section. The names of Faddeev-Popov ghosts and anti-ghosts are constructed by means of suffixes *'c'* and *'C'* respectively. For example, *G.c, G.C* are gluon ghosts and *W+.c, W+.C, W-.c, W-.C* are W-boson ghosts.

The operation of Hermitian conjugation transforms a Faddeev-Popov ghost to itself whereas an anti-ghost is transformed to itself with the opposite sign. Thus the rules of Hermitian conjugation of gluon and W-boson ghosts are

$$\begin{aligned}(G.c)^+ &= G.c\\ (G.C)^+ &= -G.C\\ (\mathrm{W+.c})^+ &= \mathrm{W\text{-}.c}\\ (\mathrm{W+.C})^+ &= -\mathrm{W\text{-}.C}\end{aligned}$$

Faddeev-Popov (anti)ghosts are scalar, anti-commutative fields [12]. The nonzero propagators for these fields are:

$$<0|T[\mathrm{A+.c}(p_1),\ A.C(p_2)]|0> = <0|T[\mathrm{A+.C}(p_1),\ A.c(p_2)]|0> = \Delta_c(p_1,p_2,M),$$

where M is a mass of the parent particle. This equality of masses is a consequence of the choice of the t'Hooft-Feynman gauge.

The appearance of the Faddeev-Popov ghosts in gauge theories can be explained in the following way. In the case of the t'Hooft-Feynman gauge the quantum field of massless gauge particle has two unphysical components, because the four-dimensional vector field describes a particle with two polarization states. The contributions of Faddeev-Popov ghost and anti-ghost compensate the contribution of these unphysical polarizations. In the case of massive gauge boson there are three physical polarizations and the compensation by means of Faddeev-Popov ghosts looks wrong from the viewpoint of naive arguments based on counting degrees of freedom. Indeed, in this case one additional *Goldstone* ghost appears [34].

Goldstone ghost. It is generated for any massive gauge vector particles. The name of this field is constructed by means of suffix *'f'*. For example, *W+.f, W-.f* are the W-boson Goldstone ghosts.

[12] The well-known spin-statistic relation is not valid for unphysical fields.

This ghost is scalar, commutative, and satisfies the same conjugation rule as the parent particle. For example, $(W+.f)^+ = W-.f$. The nonzero propagators for these fields are:

$$T[\mathrm{A+.f}(p_1),\ A.f(p_2)] = \Delta_c(p_1,p_2,M),$$

where M is a mass of the parent particle.

Tensor ghost. Whereas the Faddeev-Popov and Goldstone ghosts are standard elements of modern quantum field theory, the tensor ghost is an original CalcHEP invention. This is an auxiliary field with a point-like propagator which is used to construct vertices with complicated color structure, for example, the four-gluon vertex.

The tensor ghost is generated automatically for any vector particle with a non-trivial SU(3) color group representation. Its name is constructed by means of suffix *'t'*. This ghost is commutative, and satisfies the same conjugation rule as the parent particles. It is Lorentz-transformed like a tensor field. The propagator is

$$<0|T[\mathrm{A+.t}^{m_1M_1}(p_1),\ A.t^{m_2M_2}(p_2)]|0> = \frac{1}{(2\pi)^4 i}\,\delta(p_1+p_2)\,g^{m_1m_2}\,g^{M_1M_2}\,. \tag{7}$$

4.1.5 Interaction vertices

The Table *Vertices* contains interaction vertices. The first four fields $A1$, $A2$, $A3$, $A4$ include the names of the interacting particles. These fields must contain particle names in CalcHEP notation. $A4$ may be empty. The last two fields *'Factor'* and *'LorentzPart'* define a vertex itself. Let S be the action, then a functional derivative of S over fields is represented as

$$\frac{\delta S}{\delta A1_{[m1]}(p1)\,\delta A2_{[m2]}(p2)\,\delta A3_{[m3]}(p3)\,[\delta A4_{[m4]}(p4)]} =$$
$$(2\pi)^4\delta(p1+p2+p3\,[+p4])\,[C^{-1^T}]\,ColorStructure\cdot Factor\cdot LorentzPart\,. \tag{8}$$

Here p and m denote 4-momenta and Lorentz indices. The brackets [ ] are used to mark the optional parts of expression. So, $A4$, $p4$, and $m4$ appear only in the case of four particle vertex. The Fourier transformation is defined by

$$A(x) = \int \exp(-ikx) A(k)\, d^4k \ . \tag{9}$$

'Factor' must be a rational monomial constructed of the model identifiers, integer numbers and imaginary unity. One may remove a vertex from the consideration multiplying the *'Factor'* by zero.

'LorentzPart' must be a tensor or Dirac γ-matrix expression. Coefficients of this expression are polynomials of the model identifiers and scalar products of momenta. The division '/' operator is forbidden in *'LorentzPart'*. It must be transferred to the *'Factor'* field.

Similar to the *Reduce* notation, in order to construct scalar products of momenta, momentum components, and metric tensors we use the dot symbol, for example,

$$\begin{aligned} p1.p2 \quad & is \quad g_{\mu\nu} p_1^\mu p_2^\nu \ ; \\ p1.m2 \quad & is \quad p_1^{m_2} \ ; \\ m1.m2 \quad & is \quad g_{m_1 m_2} \ . \end{aligned}$$

To implement the Dirac γ-matrix with index $'m'$ we use a symbol $G(m)$, whereas $G(p)$ denotes $p_\mu \gamma^\mu$. Anti-commutation relations for γ matrices is

$$G(v1)\, G(v2) + G(v2)\, G(v1) = 2\ v1.v2, \tag{10}$$

where $v1, v2$ are momenta or indices.

The γ_5 matrix is denoted by $G5$. It is defined by equation

$$\gamma_5 = i\ \gamma_0 \gamma_1 \gamma_2 \gamma_3$$

In the case of anti-commuting fields the right-side derivatives are assumed. The number of fermion fields in one vertex must be two or zero. If you would like to implement a four-fermion interaction, use an auxiliary unphysical field which may be constructed by means of the '*' symbol in the 'Aux' column of the particle table (see Section 4.1.3).

CalcHEP interprets the anti-particle spinor field as a C-conjugated particle field, rather than the Dirac conjugated one. The definition of the C-conjugation matrix (5) yields

$$\frac{\delta}{\delta \psi^c} = C^{-1^T} \frac{\delta}{\delta \bar{\psi}} \tag{11}$$

This is a reason of appearance of the C^{-1^T} matrix in (8). The particle and anti-particle fields can appear in vertices in any order. Vertex also can contain two fields corresponding to particles or to two fields corresponding to some anti-particles. By other words, vertices with the fermion number violation are allowed.

Any fermion vertex can be presented in two form depending on the order of fermion fields. After permutation of fermion fields the *LorentzPart* is transformed according to

$$G(v_1)\, G(v_2) \ldots [G5] \ldots G(v_n) \to (-G(v_n)) \ldots [G5] \ldots (-G(v_2))\, (-G(v_1)) \ . \tag{12}$$

Note that structures like $m1.m2$ and $p1.m2$ are forbidden for the vertex with fermions. In order to implement these structures use the equation (10).

There is a possibility to construct a vertex with the Levi-Chivita tensor. In case of a boson vertex it can be done by means of `eps(v1,v2,v3,v4)` function, where *vi* is either some momentum or a Lorentz index. In the case of fermion vertex one has to realize the Levi-Chivita tensor in terms of γ-matrices following to the formula

```
eps(v1,v2,v3,v4) = -i*G5*[G(v1)*G(v2)*G(v3)*G(v4)]
```

Here `"[..]"` denotes the anti-symmetrization. It's a cumbersome expression with 24 terms.

Let us note that by definition (8) the *LorentzPart* has the corresponding symmetry property in the case when identical particles appear in one vertex. This symmetry is not checked by CalcHEP, and its absence will lead to wrong results. The equation 12 can be used to check this symmetry in the case of two identical Majorana fields in one vertex.

ColorStructure is substituted by CalcHEP automatically. For a colorless particle vertex it is equal to 1. For $(3 \times \bar{3})$ and for (8×8) vertices the unity tensor is substituted. If CalcHEP meets a vertex with three particles in the adjoint representation $(8 \times 8 \times 8)$, it substitutes

$$-\, i\, f(a1, a2, a3),$$

where $f^{\alpha_1}_{\alpha_2 \alpha_3}$ are the structure constants of SU(3). Color indices $a1, a2, a3$ are taken in the same order as they appear in the particle columns. For the

$(3 \times \bar{3} \times 8)$ vertex CalcHEP substitutes

$$\frac{1}{2}\lambda(\bar{i}, i, a),$$

where $\lambda(\bar{i}, i, a)$ are the Gell-Mann matrices. More complicated color structures are not implemented yet, but it is possible to construct them by means of unphysical particles (*Aux*='*') or tensor ghosts (Section 4.1.4). In the case of tensor auxiliary field use the capital *'M'* for designation of the second Lorentz index of this field as it is shown in equation (7).

4.2 Examples

4.2.1 Implementation of QCD Lagrangian

3-gluon vertex. Lagrangian (45) contains the following 3-gluon vertex:

$$S_{3G} = -\frac{g}{2}\int(\partial_\mu G^\alpha_\nu - \partial_\nu G^\alpha_\mu) f^\alpha_{\beta\gamma} G^\beta_\mu G^\gamma_\nu d^4x = -g\int \partial^{\mu_2} G^{\alpha_1}_{\mu_1} g^{\mu_1\mu_3} f_{\alpha_1\alpha_2\alpha_3} G^{\alpha_2}_{\mu_2} G^{\alpha_3}_{\mu_3} d^4x \ ,$$

where G^α_μ is the gluon field, g is the strong coupling constant. Applying the Fourier transformation (9) we get

$$S_{3G} = (2\pi)^4 g \int \delta(p_1+p_2+p_3)\ i\ f_{\alpha_1\alpha_2\alpha_3} p_1^{\mu_2} g^{\mu_1\mu_3} G^{\alpha_1}_{\mu_1}(p_1) G^{\alpha_2}_{\mu_2}(p_2) G^{\alpha_3}_{\mu_3}(p_3) d^4p_1 d^4p_2 d^4p_3 \ .$$

This vertex contains three identical fields, so the calculation of functional derivatives gives us six terms:

$$\frac{\delta S_{3g}}{\delta G^{\alpha_1}_{\mu_1}(p_1)\delta G^{\alpha_2}_{\mu_2}(p_2)\delta G^{\alpha_3}_{\mu_3}(p_3)} = (2\pi)^4\delta(p_1+p_2+p_3)\ i\ f_{\alpha_1\alpha_2\alpha_3}$$
$$g(p_1^{\mu_2} g^{\mu_1\mu_3} - p_3^{\mu_2} g^{\mu_1\mu_3} + p_2^{\mu_3} g^{\mu_1\mu_2} - p_1^{\mu_3} g^{\mu_1\mu_2} + p_3^{\mu_1} g^{\mu_2\mu_3} - p_2^{\mu_1} g^{\mu_2\mu_3}) \ .$$

Comparing this expression with the CalcHEP vertex representation (8) where *ColorFactor* is $(-i\, f_{\alpha_1\alpha_2\alpha_3})$ we get

$$Factor \cdot LorentzPart = -g\left((p_1^{\mu_2} - p_3^{\mu_2}) g^{\mu_1\mu_3} + (p_2^{\mu_3} - p_1^{\mu_3}) g^{\mu_1\mu_2} + (p_3^{\mu_1} - p_2^{\mu_1}) g^{\mu_2\mu_3}\right) \ .$$

CalcHEP uses the notation GG for the strong coupling constant g. So for the 3-gluon vertex in the CalcHEP format we finally get

A1	A2	A3	A4	Factor	Lorentz part
G	G	G		GG	m1.m2*(p1-p2).m3+m2.m3*(p2-p3).m1+m3.m1*(p3-p1).m2

Quark-gluon interaction. The interaction of a gluon with a quark is described by the following term of Lagrangian (45):

$$S_{QqG} = g\int G^\alpha_\mu(x)\bar{q}(x)\gamma^\mu \hat{t}_\alpha q(x)\ d^4x \ .$$

Applying the Fourier transformation and substituting $\bar{q} = q^+\gamma^0$ we get

$$S_{QqG} = g(2\pi)^4 \int \delta(p_1+p_2+p_3) G^\alpha_\mu(p_3)\bar{q}(p_1)\gamma^\mu \hat{t}_\alpha q(p_2)\ d^4p_1 d^4p_2 d^4p_3 \ ;$$

$$\frac{\delta S_{QqG}}{\delta\bar{q}_i(p_1)\delta q^j(p_2)\delta G^\alpha_\mu(p_3)} = g(2\pi)^4\delta(p_1+p_2+p_3)(\hat{t}_\alpha)^i_j\gamma^{\mu_3} \ .$$

The factor $(2\pi)^4\delta(p_1+p_2+p_3)(\hat{t}_\alpha)^i_j$ is substituted by CalcHEP automatically. The factor C^{-1^T} appears in (8) according to (11) and also is substituted by CalcHEP. Thus, the quark-gluon interaction is implemented in the CalcHEP *Vertex* table as the following record:

A1	A2	A3	A4	Factor	Lorentz part
Q	q	G		GG	G(m3)

where q and Q are designations for a quark and a corresponding antiquark.

Interaction of ghosts with gluon. This interaction is described by the non-linear term of (47), namely

$$S_{\bar{c}cG} = -g\int \bar{c}_\alpha(x)\partial^\mu(f^\alpha_{\beta\gamma} G^\beta_\mu(x) c^\gamma(x)) d^4x \ ;$$

Fourier transformation and subsequent evaluation of functional derivatives gives us

$$S_{\bar{c}cG} = -g(2\pi)^4\int \delta(p_1+p_2+p_3)\bar{c}_\alpha(p_1)(-i\,p_2 - i\,p_3)^{\mu_3} f^\alpha_{\beta\gamma} G^\beta_{\mu_3}(p_3) c^\gamma(p_2) d^4p_1 d^4p_2 d^4p_3$$

$$\frac{\delta S_{\bar{c}cG}}{\delta\bar{c}_{\alpha_1}(p_1)\delta c^{\alpha_2}(p_2)\delta G^{\alpha_3}_{\mu_3}(p_3)} = g(2\pi)^4\delta(p_1+p_2+p_3) p_1^{\mu_3} i\, f^{\alpha_1}_{\alpha_2\alpha_3} \ .$$

The factor $(2\pi)^4\delta(p_1+p_2+p_3)(-i\, f_{\alpha_1\alpha_2\alpha_3})$ is substituted by CalcHEP. Thus, this interaction may be implemented as the following record in the

Vertex table.

A1	A2	A3	A4	Factor	Lorentz part
G.C	G.c	G		-GG	p1.m3

where $G.C$ and $G.c$ are the CalcHEP notations for the Faddeev-Popov ghosts $\bar{c}$ and c respectively.

4-gluon interaction. Besides the 3-gluon interaction term the Lagrangian (45) contains also the term of 4-gluon interaction:

$$S_{4G} = -\frac{g^2}{4} g^{\mu\mu'} g^{\nu\nu'} \delta_{\alpha\alpha'} \int f^{\alpha}_{\beta\gamma} G^{\beta}_{\mu}(x) G^{\gamma}_{\nu}(x) f^{\alpha'}_{\beta'\gamma'} G^{\beta'}_{\mu'}(x) G^{\gamma'}_{\nu'}(x) d^4x \ .$$

The Fourier transformation and functional differentiation lead us to the expression which contains three different $SU(3)$ color structures:

$$\frac{\delta S_{4G}}{\delta G^{\alpha_1}_{\mu_1}(p_1)\delta G^{\alpha_2}_{\mu_2}(p_2)\delta G^{\alpha_3}_{\mu_3}(p_3)\delta G^{\alpha_4}_{\mu_4}(p_4)} = -g^2(2\pi)^4\delta(p_1+p_2+p_3+p_4)\delta_{\epsilon\epsilon'} \times$$
$$\Big(f^{\epsilon}_{\alpha_1\alpha_2} f^{\epsilon'}_{\alpha_3\alpha_4}(g^{\mu_1\mu_3}g^{\mu_2\mu_4} - g^{\mu_1\mu_4}g^{\mu_2\mu_3}) + f^{\epsilon}_{\alpha_1\alpha_3} f^{\epsilon'}_{\alpha_2\alpha_4}(g^{\mu_1\mu_2}g^{\mu_3\mu_4} - g^{\mu_1\mu_4}g^{\mu_2\mu_3})$$
$$+ f^{\epsilon}_{\alpha_1\alpha_4} f^{\epsilon'}_{\alpha_2\alpha_3}(g^{\mu_1\mu_2}g^{\mu_3\mu_4} - g^{\mu_1\mu_3}g^{\mu_2\mu_4})\Big) \ . \tag{13}$$

The complicated color structure of this vertex cannot be directly written down in the CalcHEP format. To implement this vertex we use the following trick. We introduce an auxiliary tensor field $t^{\alpha}_{\mu\nu}(x)$ and the Lagrangian of its interaction with the gluon field:

$$S_{aux} = \int \left(\frac{i\,g}{\sqrt{2}} f^{\alpha}_{\beta\gamma} t_{\alpha}{}^{\mu\nu}(x) G^{\beta}_{\mu}(x) G^{\gamma}_{\nu}(x) - \frac{1}{2} t^{\alpha}{}_{\mu\nu}(x) t_{\alpha}{}^{\mu\nu}(x) \right) d^4x \ .$$

It is easy to notice that the functional integration over the auxiliary field $t^{\alpha}_{\mu\nu}(x)$ reproduces the term of 4-gluon interaction in the partition function:

$$e^{i\,S_{4G}(G)} = \int e^{i\,S_{aux}(G,t)} \prod_{x,\alpha,\mu,\nu} d\,t^{\alpha}_{\mu\nu}(x) \ .$$

For each colored vector particle CalcHEP adds a tensor field with the same color to the internal list of quantum fields. The propagator of this

field (7) corresponds to the Lagrangian $(-\frac{1}{2}t^{\alpha}{}_{\mu\nu}(x)t_{\alpha}{}^{\mu\nu}(x))$. Consequently, in order to realize the 4-gluon interaction we must introduce a vertex for the interaction of the gluon with this tensor field

$$S_{tGG} = \frac{i\,g}{\sqrt{2}} \int f^{\alpha}_{\beta\gamma} t_{\alpha}{}^{\mu\nu}(x) G^{\beta}_{\mu}(x) G^{\gamma}_{\nu}(x) d^4x \ ;$$
$$\frac{\delta S_{tGG}}{G^{\alpha_1}_{\mu_1}(p1) G^{\alpha_2}_{\mu_2}(p2) \delta t^{\alpha_3}_{\mu_3\mu_3'}(p_3)} = (2\pi)^4\delta(p_1+p_2+p_3)(-i\, f_{\alpha_1\alpha_2\alpha_3}) \times \frac{g}{\sqrt{2}}(g^{\mu_2\mu_3}g^{\mu_1\mu_3'} - g^{\mu_1\mu_3}g^{\mu_2\mu_3'}) \ .$$

In CalcHEP notations this vertex looks like the following

A1	A2	A3	A4	Factor	Lorentz part
G	G	G.t		GG/Sqrt2	m2.m3*m1.M3 -m1.m3*m2.M3

where $G.t$ is a CalcHEP notation for the auxiliary tensor field $t^{\alpha}_{\mu\nu}$ associated with the vector field G. Capital M denotes the second Lorentz index of the tensor field.

From the viewpoint of the Feynman diagram technique such realization of the 4-gluon interaction means that instead of one 4-gluon vertex we substitute three sub-diagrams presented in Fig.12. Contribution of each of these diagrams corresponds to one of the terms of expression (13).

4.2.2 Neutrino as a Majorana fermion

It is well known that in the Standard Model only the left component of massless neutrino takes part in interactions. So one can describe neutrino by a Majorana field which has the same number of degrees of freedom as a left Dirac one. To realize such a particle in the framework of CalcHEP one should add the following record to the table of particles:

Full name	A	A+	2*spin	mass	width	color	aux
neutrino	MN	MN	1	0	0	1	

In terms of Dirac field Ψ_{ν} a neutrino appears in the Standard Model

Lagrangian in the following way (60)[13]:

$$
\begin{aligned}
L_\nu &= \frac{i}{2}(\bar\Psi_\nu\gamma_\mu\partial^\mu\Psi_\nu - (\partial^\mu\bar\Psi_\nu)\gamma_\mu\Psi_\nu) + \frac{e}{4\sin\Theta_w\cos\Theta_w}Z_\mu\bar\Psi_\nu\gamma^\mu(1-\gamma^5)\Psi_\nu \\
&+\frac{e}{2\sqrt{2}\sin\Theta_w}\left(W^-_\mu\bar\Psi_e\gamma^\mu(1-\gamma^5)\Psi_\nu + W^+_\mu\bar\Psi_\nu\gamma^\mu(1-\gamma^5)\Psi_e\right)\ , \qquad (14)
\end{aligned}
$$

where Ψ_e is the electron field. To rewrite it in terms of a Majorana fermion let us perform the substitution

$$\Psi_\nu = \frac{1}{2}(1-\gamma^5)\psi_l + \frac{1}{2}(1+\gamma^5)\psi_r\ ,$$

where ψ_l and ψ_r are Majorana fermions. Omitting Lagrangian for ψ_r and applying the following identities for Majorana fermions

$$\frac{i}{4}(\bar\psi\gamma_\mu\gamma^5\partial^\mu\psi - (\partial^\mu\bar\psi)\gamma_\mu\gamma^5\psi) = \bar\psi\gamma^\mu\psi \quad = \quad 0$$

which can be obtained by means of (12), we get

$$
\begin{aligned}
L_\nu &= \frac{i}{4}(\bar\psi_l\gamma_\mu\partial^\mu\psi_l - (\partial^\mu\bar\psi_l)\gamma_\mu\psi_l) - \frac{e}{4\sin\Theta_w\cos\Theta_w}Z_\mu\bar\psi_l\gamma^\mu\gamma^5\psi_l \\
&+ \frac{e}{2\sqrt{2}\sin\Theta_w}\left(W^-_\mu\bar\Psi_e\gamma^\mu(1-\gamma^5)\psi_l + W^+_\mu\bar\psi_l\gamma^\mu(1-\gamma^5)\Psi_e\right)\ .
\end{aligned}
$$

Here the first term is the free Lagrangian for a massless Majorana fermion. Other terms define the interaction. Using the definition (8) we can rewrite them in the CalcHEP notations:

A1	A2	A3	A4	Factor	Lorentz part
MN	MN	Z		-EE/(2*SW*CW)	G(m3)*G5
E	MN	W-		EE/(2*Sqrt2*SW)	G(m3)*(1-G5)
MN	e	W+		EE/(2*Sqrt2*SW)	G(m3)*(1-G5)

Let us emphasize that there are two identical neutrino fields in that term of the Lagrangian which describes the interaction of neutrinos with a Z-boson. It leads to the additional factor 2 and to the symmetry property

[13] where $Y=-1$, $\Psi_1=\Psi_\nu$, $\Psi_2=\Psi_e$, $g_2=e/\sin\Theta_w$, $g_1=e/\cos\Theta_w$

of the corresponding vertex. One of the typical mistakes in realization of such a vertex is an introduction of the G(m3)*(1-G5) term which breaks the symmetry property. Correct evaluation of the functional derivative (8) with the help of the identity (12) never produces such a term.

4.2.3 Leptoquarks

In this section we present an example of Lagrangian which contains the matrix of C-conjugation. Such matrix appears in interactions violating the fermion number conservation. Let Ψ_e and Ψ_u be the fermion fields of electron and u-quark, respectively, interacting with the scalar complex leptoquark field F

$$L = \lambda\bar\Psi^c_u(1+\gamma^5)\Psi_e F + \lambda\bar\Psi_e(1+\gamma^5)\Psi^c_u F^+$$

where $\bar\Psi^c_u = \Psi^T_u C$ and $\Psi^c_e = C\bar\Psi^T_e$.

In the Majorana basis which is used in CalcHEP the charge conjugation operator $C=-\gamma^0$. So the Lagrangian can be presented in the form

$$
\begin{aligned}
L &= -\lambda\Psi^T_u\gamma^0(1+\gamma^5)\Psi_e F - \lambda\Psi^+_e\gamma_0(1+\gamma^5)\gamma^0(\gamma^0)^T\Psi^+_u F^+ \\
&= -\lambda\Psi^T_u\gamma^0(1+\gamma^5)\Psi_e F + \lambda\Psi^+_u\gamma^0(1+\gamma^5)\Psi^+_e F^+\ .
\end{aligned}
$$

Direct implementation of the definition (8) gives us the *Vertex* table

A1	A2	A3	A4	Factor	Lorentz part
u	e	F		-lambda	(1+G5)
E	U	F+		lambda	(1+G5)

By means of equation (12) we can rewrite this table in the equivalent form:

A1	A2	A3	A4	Factor	Lorentz part
e	u	F		-lambda	(1+G5)
U	E	F+		lambda	(1+G5)

5 CalcHEP output files

5.1 LaTeX output

CalcHEP uses the *Axodraw* package by J.A.M. Vermaseren [37] to write diagrams and plots in LaTeX format. To use this package the *Axodraw* style should be included in the *documentstyle* statement. An example would be

`\documentstyle[axodraw]{article}`

Under kind permission of the author we put a copy of *axodraw.sty* file in the `$CALCHEP` directory. One bug in the axis drawing routine has been corrected.

The *Axodraw* syntax is very easy and you can alter the CalcHEP output to get some local correction in picture if any. The user can also change line width, scale of picture, and size of characters. The corresponding LaTeX and *Axodraw* instructions are presented in the beginning of the output. For example,

```
{  \small              % letter  size control
   \SetWidth{0.7}      % line  width control
   \SetScale{1.0}      % line scale control
   \unitlength=1.0 pt  % text position control
   ............
}
```

Note that the *SetScale* instruction influences on positions of lines, whereas the *unitlength* variable is responsible for position if texts. Consequently, if the user would like to change the scale of picture, he must improve these instructions in the same manner. For instance, to increase the picture by two times use

```
   \SetScale{2.0}      % picture size control
   \unitlength=2.0 pt  % picture size control
```

In the case of Feynman diagram output CalcHEP substitutes LaTeX names of particles as they are defined in the *Particle* table (see Section 4.1.3).

5.2 Symbolic answer in *Reduce* and *Mathematica* formats

5.2.1 General structure

The CalcHEP symbolic output can be used for further manipulation with an answer which has been obtained by means of the built-in symbolic calculator. It might be a summation of all diagrams to a common denominator expression, a symbolic integration of answer, a representation of answer as a function of special set of variables and so on. We have tried to present results in the form which can be easily used for different purposes.

All diagram contributions for one subprocess are stored in one file. The subprocess ordering number is attached to the file name. For example, the *symb1.red* and *symb1.m* files are generated to present the symbolic answer of the first subprocess in the *Reduce* [24] and *Mathematica* [25] format correspondingly.

The structure of the output file can be described by the following scheme:

Initial declarations;
$initSum()$;
Answer for the first diagram;
$addToSum()$;
Answer for the second diagram;
$addToSum()$;
....
....
$finishSum()$;

'Initial declarations' includes the declaration of vector variables for momenta and the conservation law relations for them, the declaration of independent parameters involved in calculation and numerical values of them, the declaration of constrained parameters and substitution rules for them, and, at last, the declaration of the process name. The momenta are named $p1, p2, p3, ...$ They are assigned to particles according to a particle sequence in the process name. The signs of momenta are defined in such a way that the sum of momenta of incoming particles is equal to the sum of momenta of outgoing particles. The list of substitutions of numerical values for independent parameters is written down in variable $parameters$. The list of substitutions for the constrained parameters is stored in variable $substitutions$. The lists

of incoming and outgoing particles are stored in variables $inParticles$ and $outParticles$ respectively.

CalcHEP writes down subsequently expressions for diagram contributions and after any record it calls a summation procedure $addToSum()$. Just before and after summation the procedures $initSum()$ and $finishSum()$ are called. These three procedures must be written by the user and loaded in advance. Such a construction of the output allows one to use it for different purposes creating appropriate procedures.

Now we shall explain the structure of diagram contribution. It is started from a pseudo-graphic diagram image like in Fig.10. After that assignments for $totFactor$, $numerator$, $denominator$ variables follow:

$totFactor$ is a ration function depending on model parameters;

$numerator$ is a polynomial of model parameters and momenta scalar products;

$denominator$ is presented as a product of propagator denominators

$$propDen(P, Mass, Width) \ ,$$

where P, $Mass$, and $Width$ are the momentum, mass, and width of the corresponding virtual particle. In the case $Width = 0$ *propDen* must be defined as $(Mass^2 - P^2)$. The treatment of the $Width$ argument can be arranged by the user as he likes.

In these terms the diagram contribution to the squared matrix element may be expressed in the following way:

$$totFactor\frac{numerator}{denominator} \ .$$

As it was mentioned above (Section 3.2.3) the result obtained by summation of all diagrams must be symmetrized in the case of identical outgoing particles. This work may be done by the $finishSum()$ procedure.

5.2.2 Example: Summation of diagrams and symbolic integration by means of the *Reduce* and Matematica packages

We have prepared some summation programs for working with *Reduce* output. They are:

`sum_cd.red` which presents the squared matrix element as a common denominator expression;

`sum_22.red` which presents the squared matrix element as a sum of pole terms;

`sum_int.red` which presents a symbolical expression for the total cross section.

The last two options are available only for `2->2` processes. These files are stored in the *utile* directory of CalcHEP package.

Let you prepare the symbolic output *symb1.red* for the $\gamma\mu$ scattering (*A,m* `->` *A,m*) [14]. Copy the `sum_cd.red`, `sum_22.red`, and `sum_int.red` files in your *WORK* directory and launch the *Reduce* system from within the *result* directory. The possible *Reduce* sessions are:

```
%1
  in"../sum_cd.red"$                    % to load   summation package
  in"symb1.red"$                        % to  read contributions of diagrams
  sum;                                  % to write the answer

(32*ee**4*(2*p1.p2**4-4*p1.p2**3*p1.p3+3*p1.p2**2*p1.p3**2-2*p1.p2**2*p1.p3
*mm**2-p1.p2*p1.p3**3 + 2*p1.p2*p1.p3**2*mm**2 + p1.p3**2*mm**4))
/(propden(p1+p2,mm,0)**2*propden(p2-p3,mm,0)**2)$

%2
  in"../sum_22.red"$                 % to load  summation package
  in"symb1.red"$                     % to read contributions of diagrams
  sum;                               % to write the answer
2*ee**4*(4*sp(mm)**2*mm**4 + 8*sp(mm)*up(mm)*mm**4 + 4*sp(mm)*mm**2 +
sp(mm)*t+ 4*up(mm)**2*mm**4 + 5*up(mm)*mm**2 - up(mm)*s + 1)$
```

Here $s = (p1 + p2)^2$; $t = (p1 - p3)^2$; the functions sp, tp, up are defined by the following way

$$sp(x) = 1/(s - x^2);$$
$$tp(x) = 1/(t - x^2);$$
$$up(x) = 1/(u - x^2) \ \ where \ \ u = (p1 - p4)^2$$

[14] In the CalcHEP models the electron is massless. It leads to divergence in the total cross section of Compton scattering. Thus we use μ instead of electron.

```
%3
  in"../sum_int.red"$            % to load summation package
  in"symb1.red"$                 % to read contributions of diagrams
  sum;                           % to  write  the answer for total cross section
(ee**4*(2*s**4*log(s/mm**2) + s**4 - 12*s**3*log(s/mm**2)*mm**2 + 14*s**3*mm**2
- 6*s**2*log(s/mm**2)*mm**4 - 16*s**2*mm**4 + 2*s*mm**6 - mm**8))/(16*s
**2*pi*(s**3 - 3*s**2*mm**2 + 3*s*mm**4 - mm**6))$
```

Sometimes the expression for total cross-section includes a cumbersome square root of kinematic variables which appears as a result of evaluation of integrand limits. In this case the integration routine introduces a new variable `be_` for this square root to present the total cross-section in a more compact form. The substitution for the `be_**2` is generated by the integration routine.

There are similar packages `sum_cd.m`, `sum_22.m`, and `sum_int.m` for operation with the *Mathematica* output. See below examples of Mathematica version 3.0 sessions. The sum of diagrams in the form of common denominaror expression. `SC[pi, pj]` denotes the scalar product of momenta.

```
In[1]:= <<"../sum_cd.m"
In[2]:= <<"symb1.m"
In[3]:= sum

                4                 4                 3                 4            2
Out[3]= (32 EE   (2 SC[p1, p2]  - 4 SC[p1, p2]  SC[p1, p3] + Mm  SC[p1, p3]  +

                            2                           2
>         SC[p1, p2] (2 Mm  - SC[p1, p3]) SC[p1, p3]  +

                      2                    2
>         SC[p1, p2]  SC[p1, p3] (-2 Mm  + 3 SC[p1, p3]))) /

                                 2                            2
>     (propDen[-p1 - p2, Mm, 0]  propDen[p2 - p3, Mm, 0] )
```

The sum in terms of `s` and `t` kinematic variables


```
In[1]:=  <<"../sum_22.m"
In[2]:= <<"symb1.m"
In[3]:= sum

             4    2     2        4              4   4
         2 EE  (Mm  + s)    2 EE  t      8 EE  Mm
Out[3]= ---------------- - ------- + --------------- +
              2    2          2            2          2
          (Mm  - s)        Mm  - s   (-Mm  + s + t)

             4   4       4   2       4  2
     2 (3 EE  Mm  + 6 EE  Mm  s - EE  s )
>    ----------------------------------
             2          2
          (Mm  - s) (-Mm  + s + t)
```

The total cross section may be obtained by

```
In[2]:= <<"../sum_int.m"
In[3]:= <<"symb1.m"
In[4]:= sum

           4     8       6           4  2        2  3    4
Out[4]= (EE  (-Mm  + 2 Mm  s - 16 Mm  s  + 14 Mm  s  + s  -

                                                2
             2       4       2      2       Mm                2     2     3
>       2 s  (-3 Mm  - 6 Mm  s + s ) Log[---])) / (16 pi s  (-Mm  + s) )
                                            s
```

5.3 *Reduce* program

The *Reduce* program was the first CalcHEP output which opened the possibility to produce some physical results by means of this package. Later on, when the CalcHEP built-in symbolic calculator was created, the *Re-*

duce output became unnecessary. But we still keep it in the package for testing.

The CalcHEP symbolic calculator looks like a black box. On the contrary the *Reduce* program is written in terms which can be understood and checked. The comparison of result produced by the built-in symbolic calculator with that of *Reduce* evaluation of the generated code is a good check of CalcHEP software (see Section A).

Below we describe the structure of *Reduce* program. The set of Feynman rules implemented in CalcHEP is presented in the section (C.4).

CalcHEP generates a separate file for each squared diagram. The files are named as *pNNN-MMM.red* where *NNN* is the subprocess number and *MMM* is the diagram ordering number.

The file begins with from the declaration of momenta and Lorentz indices. For example,

```
% ----------- VARIABLES ------------------
 vector  A,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,ZERO_;
 vector  m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16;
%
%--------- Mass shell declarations -----------
 MASS  P1 = 0$  MSHELL P1$
 MASS  P2 = 0$  MSHELL P2$
 MASS  P3 = Mm$  MSHELL P3$
 MASS  P4 = Mm$  MSHELL P4$

%-------- Momentum substitutions --------
 Let  p4 = +p1+p2-p3$
 Let  p5 = +p1+p2$
 Let  p6 = -p1-p2$
```

Vector A is used by the *Reduce* package to construct the γ^5 matrix, $\gamma^5 = G(ln, A)$. Vectors whose names begin with `'p'` are used for designation of momenta. Vectors whose names begin with `'m'` are reserved for Lorentz indices. We see then the mass-shell declarations for incoming and outgoing particles. After that the file contains the substitutions of momenta of on-shell and virtual particles which we have according to the conservation law.

Then the diagram total factor with self explanatory comments is written down. For example, in the case of $e^+e^- \to \mu^+\mu^-$ process we have

```
%---------- Factors ---------------
 SymmFact:=1/1$    % Diagram symmetry factor
 AverFact:=1/4$       % Normalization factor of polarization average
 FermFact:=1$      % (-1)**(number of in-fermion particles)
 ColorFact:=1/1$    %  QCD color weight of diagram
%
totFactor_:=EE**4$
totFactor_:=totFactor_*SymmFact*AverFact*FermFact*ColorFact$
```

These declarations are self-explanatory except, maybe, of the $SymmFact$ variable. Generally $SymmFact = N/D$ where N is equal to 1 in the case of left-right squared diagram symmetry. Otherwise the factor equals 2. D is a factorial connected with presence of identical outgoing particles and partially reduced by a number of various possibilities to assign the momenta of outgoing particles to the corresponding diagram lines.

The program for evaluation of one squared diagram actually includes the codes for a set of diagrams which appear after replacing of some physical particles by their ghosts according to the existing interaction vertices. We shall call all diagrams of the set as *ghost* diagrams. Note that all these diagrams have the same denominator. The evaluation is started from the initialization of variable for the sum of numerators of the set:

`numerator_:=0$`

The program for evaluating each diagram of the set is advanced by the pseudo-graphical image of diagram. (See, for example, Fig.10). The name of particle and corresponding momentum are written down near the line. The Lorentz index is written just on the line.

The diagram code is started from the fermion loops evaluation. The program moves along the fermion line and multiplies the vertex and propagator terms. The *nospur* instruction is declared before each loop evaluation to prevent the default trace evaluation in the end of any instruction.

If the result of multiplication contains Lorentz indices which can be contracted, the program declares the corresponding vectors as indices by means of the *index* instruction.

As a rule the fermion vertices are multiplied in the order which corresponds to moving in the direction opposite to the direction of fermion arrows. But if the diagram contains a vertex with the *C-conjugate* operator or Majorana particles the order of multiplication is chosen at random, because the propagator lines have different orientations or have no them at all. If the orientation of fermion line is chosen, the vertices are divided into two groups, normal and reverse. For the normal vertex the incoming fermion line is attached to the second fermion, whereas for the reverse vertex it is attached to the first one. CalcHEP transforms the reverse vertices to normal form by means of the rule (12) and writes down the corresponding comment.

After the last multiplication the state of the *spur* switch is restored and the γ-matrix trace is evaluated with simultaneous multiplication by factor of '-4'. The '4' is needed because the *Reduce* trace evaluation omits this factor and the minus appears from the Feynman rules.

The next step is the multiplication of vertices and contraction of Lorentz indices . The indices for contraction are declared by the *Index* instruction before the multiplication. The fragment of the corresponding code looks like

```
Index m2$
Vrt_3:=Vrt_3*Vrt_5$
RemInd  m2$
```

This code means that the vertex number 3 is multiplied by the vertex number 5 with contraction of *index* $m2$. The result of multiplication is considered as a generalized vertex number 3.

Here we would like to note that there is a bug in the *Reduce* package which forbids to convolute several indices in one operation. To bypass this bug we declare the 'm' vectors as indices step by step and recalculate the expression on each step.

If two vertices are connected by a propagator of massive vector particle treated in the physical gauge (6) the code are organized in the following way. CalcHEP writes the code for the vertices product with contracted indices and the code for the product of the same vertices with the prior multiplication by relevant momentum. After that these contributions are summarized. Below we present an example of such a code:

```
Index m1$
Vrt_0:=Vrt_1*Vrt_2$
```



```
RemInd  m1$
Vrt_L:=Vrt_1$   Vrt_R:=Vrt_2$
Vrt_L:=(Vrt_L where m1=>(+P2+P3)/MZ)$
Vrt_R:=(Vrt_R where m1=>(+(-P2)+(-P3))/MZ)$
Vrt_0:=Vrt_0 + Vrt_L*Vrt_R$
```

The code for any diagram evaluation is terminated by instruction

$$numerator\_ := numerator\_ + DiagramFactor * GhostFact * Vrt\_1$$

which summarizes the ghost diagram contributions. Here *GhostFact* equals $(-1)^{l+v}$ where the l is a number of loops of Faddeev-Popov ghosts and v is a number of vector field lines in the diagram. The $(-1)^v$ factor appears because the evaluations described above correspond to substitution of the $(g_{\mu\nu} - k_\mu k_\nu/M^2)$ factor for the propagator and the density matrix of vector field whereas the correct expression has the opposite sign.

The last step of the program is the assignment of variable *denominator_*. It is expressed as a production of $propDen(P, Mass, Width)$ functions as it is explained in Section 5.2.

As a result the symbolic answer for the evaluated diagram may be presented in the form

$$totFactor\_ \frac{numerator\_}{denominator\_}\ .$$

5.4 *C* output files

CalcHEP *C*-output is used to compile the squared matrix element for subsequent evaluation of cross sections and distributions. The explanation below needs only if you would like to use the CalcHEP phase space integration facilities for a matrix element produced outside of CalcHEP, or, vice versa, to use the CalcHEP matrix element into some other program.

All global functions and variables of C-output have extension *_ext*. Thus, if one would like to collect into one executable program several matrix elements, he may change this extension and by this way resolve conflict of names [15].

[15] It can be done by means of `sed` command
sed s/_ext/*your_extension*/g old_file > new_file

Now we shall list the functions and the global constant parameters defined in this output and called by the program *n_calchep_c*. All prototypes of functions described below are stored in *include/sqme_out.h*.

Parameter section

`const int nvar_ext` is a number of independent physical parameters involved in the evaluation of squared matrix element;

`const int nfunc_ext` is a number of constrained parameters involved in the evaluation of squared matrix element;

`char * varName_ext[]` is an array which contains names of paremetres;

`double va_ext[]` is an array which contains numerical values of parameters.

The index of theses arrays runs from 1 to `nvar_ext+nfunc_ext`. The zero index is used for the center of mass momentum;

`int calcFunc_ext(void)` calculates all constrained parameters for current values of independent ones. It returns 0 in the case of success, otherwise the it returns the parameter number which can not be evaluated. The `calcFunc_ext` function should be called before the first call of `sqme_ext` or when some parameters where changed.

Process section

`const int nin_ext` is a number of incoming particles

`const int nout_ext` is a number of outgoing particles

`const int nprc_ext` is a total number of subprocesses

`char * pinf_(int nsub, int nprtcl, double * mass)` returns the name of the particle involved in interaction and numerical value of its mass. Here $0 < nsub <= nprc\_ext$ is the number of subprocess, $0 < nprtcl <= nin\_ext + nout\_ext$ is the ordering number if particle for the subprocess. First nin_ext numbers are reserved for incoming particles, while next $nout\_out$ are assigned to outgoing ones.

One can substitute the *NULL* constant instead of *mass* parameter if the corresponding information is not needed.

The squared matrix element

`double sqme_ext(int nsub,double * momenta, int * err)` returns the numerical value of squared matrix element after summation over out-particle polarizations and averaging over in-particle polarizations. Here *nsub* is a subprocess number, *momenta* is an array of particle momenta. The i^{th} Lorentz component of the particle number N corresponds to *momenta[4*N+i]*, **err* is a return parameter which does not equal zero if *sqme_ext* cannot be evaluated.

We assume that the zero Lorentz components of momenta are positive and the sum of incoming particle momenta is equal to the sum of outgoing particle momenta.

It is assumed that the same units (for example, powers of GeV) are used for momenta and for physical variables. *sqme_ext* produces an answer in the

$$[P]^{(4-2*(nin\_ext+nout\_ext))}$$

units, where `[P]` is a unit of momentum. Standard normalization for *sqme* is used. See Eqs. (34.10) and (34.24) in [27] p.211,212. When *n_calchep_c* calls the *sqme_ext* function it substitutes momenta in the GeV units.

Widths implementation parameters

`int rwidth_ext, gwidth_ext, twidth_ext` are the variables which define a particle width treatment for calculations of the squared matrix element. They mean *running width* and *gauge invariant width*, and *t-channel width* correspondingly . See Section 3.3.4 for details. The value of this variables can be changed by means of the *n_comphep_c* menu.

Color basis

`void build_cb_ext(int nsub)` constructs *color basis* for given subprocess *nsub*. In result of this procedure the following parameters get their values:

`int cb_pow_ext` - the power of the color basis;

`int cb_nc_ext` - number of color chains for the subprocess;

`int * cb_chains_ext`- $2 \cdot cb\_pow\_ext \cdot cb\_nc\_ext$ numbers which describe color basis elements in terms of color chains. The k^{th} basis element ($0 <=$

$k < cb\_pow\_ext$) is described by cb_nc_ext color chains. The i^{th} chain ($0 <= i < cb\_nc\_ext$) is started on particle

$$cb\_chains\_ext[2 \cdot k \cdot cb\_nc\_ext + 2 \cdot i]$$

and finished on particle

$$cb\_chains\_ext[2 \cdot k \cdot cb\_nc\_ext + 2 \cdot i + 1]$$

Here particles are numerated according to the `pinf_ext` function. The first end of chain corresponds to $\bar{3}$ color, the color of the second end is 3 `double * cb_coeff_ext[i]` ($0 <= i < cb\_pow\_ext$) presents probabilities of different color basis configurations. They are evaluated by `sqme_ext` if the color basis was preliminary initiated by means of `build_cb_ext`;

`void destroy_cb_ext(void)` destroys the color basis.

Example

Subdirectory `$CALCHEP\utilie` contains an example of main function *main_22.c* which calculates $d\sigma/d\cos(\phi)$ Note, in order to compile the program except of *main_22.c* and c-code produced by CalcHEP one needs the files *num_out.h, num_in.h,* and *sqme.inc* disposed in sub-directory `$CALCHEP\include` and the file `$CALCHEP/c_sources/sqme_aux/sqme_aux.c`

5.5 Format of events

In general all needed comments are attached to the file. See below an example of header of such file

```
#CalcHEP version 2.0
#PROCESS   e    E   -> b    B
#Initial_state
  SQRT(S) 2.000000E+02
  Rapidity(c.m.s) 0.000000E+00
  StrFun1: ISR(200 Beamstr.: OFF)
  StrFun2: ISR(200 Beamstr.: OFF)
#MASSES  0.0000000000E+00 0.0000000000E+00 4.6200000000E+00 4.6200000000E+00
#Cross_section(Width) 1.695041E+01
#Number_of_events        1000
```


After that the table of events is written. The first column presents weights of events. Normal value for $weight$ is 1. The reason of appearance $weight \neq 1$ is explained in the section (G.3). After that the columns which specify the momenta of particles are presented. The first line of the table contains titles for the columns. Say `P3_2` means the second component of momentum of the third particle. The zero, energy, components are not presented because they can be calculated using the information about the particle masses. For incoming particles only the third momentum component is presented because other ones are zero.

After description of momentum the event record contains information about color flows. Each color flow is presented by a couple of particle numbers enclosed into brackets. It corresponds to propagation of color 3 from the first particle to the second one. See section (D) for algorithm of color flows generation.

Information about color flows is need to PYTHIA [32] to generate the correct fragmentation of colored quarks and gluons.

Appendix

A Self-check of the CalcHEP package

The CalcHEP authors have invented some tools for testing the program. The positive result of these checks allows us to justify that CalcHEP works correctly. Our tests relate to the symbolic result level and have been realized with the help of the *Reduce* [24] symbolic manipulation system. All test routines are stored in the `$CALCHEP/utile` directory. If the user is going to repeat them he has to copy the corresponding files into his working directory. All check commands must be started from the same directory.

A.1 Check of the built-in symbolic calculator

The first check is a comparison of results produced by the CalcHEP symbolic calculator (Section 5.2) with those by *Reduce* evaluation of the corresponding code (Section 5.3). The positive result of this comparison means that our built-in symbolic calculator works correctly. Note that the *Reduce* code may be viewed through by the user and the CalcHEP algorithms for evaluation of the squared matrix element can be verified in this way.

The check is realized by means of the program *check.red* which must be started from within the *Reduce* session by instruction

```
in"check.red";
```

It is assumed that the *Reduce* code for diagrams and the corresponding expressions evaluated by CalcHEP are stored in the *results* directory in advance. The results of this check are saved in the *message* file. It consists of a list of diagram numbers accompanied by the labels *OK* or *Error* depending on the result of comparison.

A.2 Comparison of results produced in two different gauges

The comparison of results produced in the unitary gauge and in the t'Hooft-Feynman gauge is a very impressive test of the package. In this way we check not only the CalcHEP code for symbolic evaluation but also the correctness of model implementation.

To perform this check we evaluate the symbolic sum of diagrams in different versions of the Standard Model and compare results. The non-zero difference is a signal of mistake. Symbolic summation is performed by *Reduce*. This summation is the most difficult step of comparison because the sum of diagrams can be extremely cumbersome.

The *Reduce* program *cmp.red* carries out a summation of symbolic answers written down in *results/symb1.red* as well as in *results_/symb1.red* and a comparison of the sums. In the case of zero difference *cmp.red* puts down *OK* into the *message* file, otherwise the word *Error* appears.

A.3 Automatic check for a set of processes

We have created the Unix script commands `cycle_check` and `cycle_cmp` which perform the above checks automatically for some set of processes. The `cycle_check` needs one numerical parameter which indicates the number of model which is tested. The `cycle_cmp` needs two numerical parameters which indicate the numbers of models which are compared.

In both cases the list of processes is read from the standard input and the generated *message* files are directed to the standard output device.

To simplify these tests we open a possibility to restrict the generated diagrams by excluding particles of the third generation. To realize this possibility just pass one addition parameter to `cycle_check` and `cycle_cmp`. The value of this parameter is unessential.

We have created various lists of Standard Model processes for testing. They are

22_0gen.prc	*Higgs and vector particles sector* *for 2->2* processes	11 processes,
22_2gen.prc	*2->2 for two generations*	142 processes,
22_3gen.prc	*2->2 for three generations*	294 processes,
23_2gen.prc	*2->3 for two generations*	455 processes.

Each list contains only one representative of the cross-symmetrical processes.

The following tests have been carried out for the current version:

```
cycle_check 4      < 22_3gen.prc
cycle_check 3      < 22_3gen.prc
```

```
cycle_check 4   2g < 23_2gen.prc
cycle_cmp   3 4    < 22_3gen.prc
```

B Ghost fields and the squared diagram technique for the t'Hooft-Feynman gauge

B.1 The problem

Every time when we are trying to create a model containing a massive vector particle we meet a problem caused by bad asymptotic of its propagator

$$\frac{i}{(2\pi)^4}\frac{g_{\mu\nu}-k_\mu k_\nu/m^2}{m^2-k^2}\ . \tag{15}$$

The $(g_{\mu\nu}-k_\mu k_\nu/m^2)$ factor gives a projection on physical degrees of freedom in the polarization space. This term appears because a 4-component vector field is used to describe a particle with 3 degrees of freedom. The $k_\mu k_\nu/m^2$ term leads to a fast growth of amplitudes at high energies, what breaks unitarity and is not compatible with the renormalizability of theory.

The problem mentioned above is solved in the framework of gauge field theories where the gauge symmetry is responsible for mutual cancellation of rapidly growing contributions of separate diagrams [34]. Let our model of particle interaction be based on a gauge theory. Then on a step of numerical evaluation we expect a cancellation of contributions which come from various Feynman diagrams. Consequently, finite precision numerical calculations may lead to wrong results. Accompanying problem is a cumbersome expression for each diagram as a result of appearance of mutually canceling terms.

At the same time there is a freedom in formulation of Feynman rules for gauge theories caused by an ambiguity of gauge fixing terms [34]. These terms modify the quadratic part of the Lagrangian and consequently may improve the vector particle propagator. Indeed, in the case of t'Hooft-Feynman gauge the propagator of vector particle takes the form

$$\frac{i}{(2\pi)^4}\frac{g_{\mu\nu}}{m^2-k^2}\ . \tag{16}$$

that provides a formulation of the theory where the problem of vector particle propagator is solved explicitly.

A price for this solution is an appearance of three additional unphysical particles in the model. They are a couple of Faddeev-Popov ghosts and one Goldstone ghost. All of them have scalar type propagators with the same mass m. Opposite to (15) the propagator (16) does not vanish when it projected onto the temporal polarization state

$$e^0 = k/m\ \ , \tag{17}$$

that also leads to the appearance of additional unphysical state. The main principles of gauge invariance guarantee [34] that an expression for the amplitude should be the same for any gauge if only physical incoming and outgoing states are considered.

Generally the t'Hooft-Feynman gauge solves the problem of cancellations. But while calculating processes with incoming or outgoing massive vector particles, we meet a similar problem. Indeed, we need to multiply the diagram contributions by polarization vectors. The polarization vectors (e^1,e^2,e^3) constitute an orthonormal basis in the sub-space orthogonal to a momentum k. Due to the relation

$$e^1_\mu e^1_\nu + e^2_\mu e^2_\nu + e^3_\mu e^3_\nu = k_\mu k_\nu/m^2 - g_{\mu\nu} \tag{18}$$

at least one of the polarization vectors has large (of order of k) components for any choice of polarization basis. Let vector k have the components

$$k = (\sqrt{m^2+p^2}, 0, 0, p)\ . \tag{19}$$

Then the polarization vectors can be chosen as

$$e^1 = (0,1,0,0)\ ; \tag{20}$$
$$e^2 = (0,0,1,0)\ ; \tag{21}$$
$$e^3 = (p/m, 0, 0, \sqrt{1+p^2/m^2})\ . \tag{22}$$

The first two vectors correspond to transversal polarizations and the third one corresponds to a longitudinal one. We see that the longitudinal vector has large (of order of k) components. It may imply a fast increase of cross-sections of processes with the longitudinal polarizations at high energies or an appearance of cancellations between various diagrams. Indeed the second case is realized and, hence, we have got a problem with diagram cancellations. When evaluating squared diagrams we get cumbersome expressions again as a result of substitution of the projector (18).

B.2 Incoming and outgoing ghosts

A solution of the above problem looks as a chess sacrifice. The main idea is to include incoming and outgoing unphysical states into the consideration. In the t'Hooft-Feynman gauge the masses of ghost partners equal the mass of parent vector particle. Let us consider ghost states as unphysical polarizations alike the temporal one (17). Note that the Faddeev-Popov ghost states and the temporal state have a negative norm, whereas the Goldstone state has a positive norm [34]. So the unphysical polarizations can give a positive as well as a negative contribution to the polarization sum.

The main statement is that a contribution of all unphysical polarizations to the sum of squared matrix element over polarizations equals zero [36]. As a result

$$\sum_{i\in S_{phys}} A_i A_i^* = \sum_{i\in S_{all}} \sigma(i) A_i A_i^* \,, \tag{23}$$

where i is a multi-index for polarization states; A_i is an amplitude of the process; S_{phys} is a set of physical polarization states; S_{all} is a full set of polarizations including unphysical ones; $\sigma(i) = \pm 1$ depending on a signature of the Hilbert space norm of the polarization state i.

A drawback due to enlarging a set of polarization states is clear: we have much more matrix element terms for evaluation and subsequent summation. To see an advantage of this trick let us sum the temporal (17) and longitudinal (22) polarization contributions. Note that both of them have components of order of k, but for calculation of the corresponding sum we can alter the basis of polarization states

$$e^0_\mu e^0_\nu - e^3_\mu e^3_\nu = e'^0_\mu e'^0_\nu - e'^3_\mu e'^3_\nu \,, \tag{24}$$

and in such a way to have new basis vectors of order of unity in spite of possible large value of the momentum k

$$e'^0 = (1,0,0,0) \;;$$
$$e'^3 = (0,0,0,1) \;.$$

In other words, the inclusion of the temporal polarization replaces the normalization condition (18) by

$$e^0_\mu e^0_\nu - e^1_\mu e^1_\nu - e^2_\mu e^2_\nu - e^3_\mu e^3_\nu = g_{\mu\nu} \;. \tag{25}$$

which does not lead to the fatal requirement on a value of polarization vectors. In the case of unpolarized calculation by the squared diagram technique we substitute $g_{\mu\nu}$ for the polarization sum according to (25). But now we have to add incoming and outgoing ghost states to the polarization sum in order to compensate the contribution of temporal polarization state.

B.3 Massless vector-particle case

We have discussed above the massive vector particle case. The Lagrangian of free massless vector field has got a gauge symmetry. So we need anyway to ensure a gauge symmetry for the interaction model to support a compatibility with the free field model. In this case the Feynman gauge leads to the propagator (16) with $m = 0$ and to the appearance of Faddeev-Popov ghosts [34]. However a Goldstone ghost does not appear and the longitudinal polarization becomes unphysical like temporal one. Summation over physical polarization states can be replaced by that over an extended set of polarizations like (23). See the corresponding proof in [35].

In the framework of amplitude technique there is no reason to include the incoming and outgoing ghost partners of massless vector particle into consideration. Longitudinal polarization becomes unphysical, but the transversal polarization vectors may be chosen of the order of unity. On the contrary, in the case of squared diagram technique the extension of polarization states is very useful and has been used in numerous calculations.

If only physical polarization states are taken into account, then for the squared diagram evaluation we must convolute free Lorentz indices, which appear after evaluation of the left and right parts of squared diagram, with the projector on physical sub-space. In the massless case this projector equals [35]

$$g_{\mu\nu} - \frac{k_\mu \eta_\nu + k_\nu \eta_\mu}{(k.\eta)} \,,$$

where η is an auxiliary vector with the zero Lorentz norm. Due to the gauge invariance the sum over all diagrams does not depend on this η, but each squared diagram contribution contains it. This leads to cumbersome expressions of squared diagram contributions in comparison with the case when the unphysical polarizations are included into the sum.

B.4 Summation of ghost diagrams in CalcHEP

CalcHEP uses the squared diagram technique with summation over polarizations. Basically one squared diagram corresponds to the

$$\sum_{i \in S_{all}} A_i^k A_i^{*l} \tag{26}$$

contribution in a squared matrix element, where A^k and A^l are the amplitudes related to some Feynman diagrams.

If we follow an idea of the previous section and take into account the ghost incoming particles, a number of squared diagrams increases significantly. For example, in the simplest case of $e^-, \gamma \to n_e, W^-$ process one squared diagram of Fig.11(a) is accompanied by three ghost diagrams with the similar topology.

Let us note that all diagrams of Fig.11 have the same denominators. The numerators of these diagrams are polynomials of scalar products of momenta. The powers of these polynomials are the same for all diagrams, so one might expect that the symbolic sum of all these diagrams has approximately the same size as one term.

Following this note CalcHEP calculates the symbolic sum of all sets of diagrams which become identical after replacing the ghost particles by their parents.

B.5 Gauge symmetry and cancellations

Cancellation of diagram contributions is an essential point both for symbolic and numerical processing, because a relatively small variation of one diagram contribution may lead to a significant error. Such variation can be caused either by finite precision of floating operations or by correction of Feynman rules, for instance, by including particle widths into consideration, or by removal of some diagram subset. We would like to stress again these obstacles to warn the user.

There are two well known examples of gauge cancellations. The first one is the ultraviolet cancellation of terms originating from the propagators of massive vector particles. This problem could be resolved by the calculation of squared matrix element in the t'Hooft - Feynman gauge.

The second example is the cancellation of double pole (t^{-2}) terms of t-channel photon propagator. There is a wide class of processes where the incoming electron goes out in the forward direction emitting a virtual photon like in Fig.13. The corresponding diagrams have got the $1/t$ pole, where t is the squared momentum of the virtual photon. For the above kinematics the photon appears very close to its mass shell ($t \approx 0$), hence this configuration gives a large contribution to the cross section.

For the squared matrix element we expect the $1/t^2$ pole, but it appears to be reduced up to $1/t$ pole [18] in the zero-electron-mass limit. This fact is caused by electro-magnetic U(1) gauge invariance. If diagrams of Fig.13 type contribute to your process, we strongly recommend to to set the *'Gauge invariance'* switch *ON* (see Section 3.3.4) to prevent the gauge symmetry breaking by width terms. Another way to solve this problem is the usage of the Weizsaecker-Williams approximation (see Section E.2).

C Feynman rules in CalcHEP

C.1 Lorentz part of diagram

Fermion loop calculation. Our algorithm of evaluation of fermion loops takes into account a possible appearance of vertices with fermion number violation. CalcHEP chooses arbitrary a direction of multiplication of vertices on the fermion line. Vertices and propagators are multiplied according to this direction to evaluate the trace. Those vertices which have the fermion line coming in the second fermion are taken in the form they are presented in the model table (section 4.1.5). Otherwise we preliminary rewrite them according to the equation (12).

There are two kinds of propagators. $<\psi(p1)\bar{\psi}(p2)>$ and $<\psi^c(p1)\bar{\psi}^c(p2)>$. In the case of Dirac or Majorana fermions both of them are equal to

$$<\psi(p1)\bar{\psi}(p2)>=<\psi^c(p1)\bar{\psi}^c(p2)>= (\not{p}_1 + M)\Delta_c(p_1, p_2, M)$$

were Δ_c is a scalar part of the propagator. Thus the propagators should provide the $(\not{p}+M)$ factors to the fermion line. The sign of p corresponds to the line direction.

This rule has an exception in case of massless left/right- fermions.

$$< \psi_{l/r}(p1)\bar{\psi}_{l/r}(p2) >= \frac{\not{p}_1(1\mp\gamma_5)}{2}\Delta_c(p_1,p_2,0)$$

whereas

$$< \psi^c_{l/r}(p1)\bar{\psi}^c_{l/r}(p2) >= \frac{\not{p}_1(1\pm\gamma_5)}{2}\Delta_c(p_1,p_2,0)$$

The result of trace evaluation is multiplied by (-1).

One remark about the case when some fermion vertex contains two identical fermions. In this case Wick contractions can be done by two ways. We have to remind (see section (4.1.5)) that expressions presented in the model table correspond to the functional derivative of the Lagrangian. Thus they have the symmetry properties and the factor 2 in comparing with the Lagrangian. Consequently, a special treatment of such vertices dose not need.

Contraction of Lorentz indices. The vertices presented in the Lagrangian table are defined as functional derivatives of the Lagrangian. Thus they should contain the needed combinatorial coefficients in case of identical particles.

A loop of Faddeev-Popov ghosts gives an extra factor (-1). For continence of indices contraction CalcHEP uses propagator of vector particle with wrong sing. The total sing of diagram is corrected in the end of evaluation. Namely the diagram is multiplied by the `GhostFact`$=(-1)^{n_g+n_v}$, where n_g is a number of Faddeev-Popov loops, n_v is a number of vector particles.

C.2 Color factor

Here we explain the technique [40] of evaluation of color factor of diagram in case of $SU(N_c)$ group. We shall use designation *gluon* (g), *quark* (q), and *anti-quark* $(\bar{q})$ for colored particles which belong to adjoint, fundamental and conjugated presentations respectively. Following to the section (4.1.5), the ggg color vertex presented by the group structure constants $-i\cdot f_{abc}$, whereas $gq\bar{q}$ vertex presented by the matrices $\hat{\tau}_a$ of the fundamental representation[16].

[16] $\hat{\tau}_a = \hat{\lambda}_a/2$, were λ_a are the Gell-Mann matrices.

By means of the basic relations

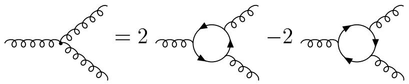
$$\hat{\tau}_a\hat{\tau}_b - \hat{\tau}_b\hat{\tau}_a = i\cdot f_{abc}\hat{\tau}_c$$

$$tr(\hat{\tau}_a\hat{\tau}_b) = \frac{1}{2}\delta_{ab}$$

one can express ggg vertex in terms of $gq\bar{q}$ ones

$$-i\cdot f_{abc} = 2\cdot tr(\hat{\tau}_b\hat{\tau}_a\hat{\tau}_c) - 2\cdot tr(\hat{\tau}_a\hat{\tau}_b\hat{\tau}_c) \quad (27)$$

This relation can be presented by the following diagram substitution which means that evaluation of diagram with $3g$ vertex can me reduced to evaluation of two diagrams where $3g$ vertex is replaced on quark loop.

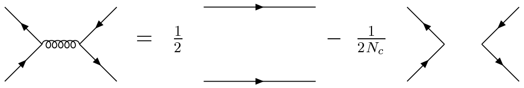


(27a)

Also there is a Fiertz identity

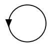
$$(\tau_a)^i_j(\tau_a)^k_l = \frac{1}{2}\delta^i_l\delta^k_j - \frac{1}{2N_c}\delta^i_j\delta^k_l \quad (28)$$

On the diagram level it can be presented as a substitution which removes gluon lines connecting quark loop(s):



(28a)

In result we have separated closed lines of quark color flows which are easy evaluated:

 $= tr(\hat{1}) = N_c$ (29)

There is an ambiguity in separate determination of color and Lorentz parts of 3-gluon vertex. This vertex by definition has a symmetry corresponding to permutation of legs. But it is a production of anti-symmetry

Lorentz part and anti-symmetry color part. Thus in case of separated calculation of Lorentz and color parts one has to take a care about identical orientation of legs in color and Lorentz diagrams.

In the $N_c \to \infty$ limit the tree level squared matrix element has the asymptotic

$$N_c^{(n_q+n_{\bar{q}}+n_g)/2}(C + O(1/N_c)) \tag{30}$$

where n_q, $n_{\bar{q}}$, n_q are the numbers of quarks and gluons involved into reaction. CalcHEP provides the user with a possibility to calculate matrix element in the $N_c = \infty$ limit where only the main term of equation (30) is taken into account. After the extraction of the leading term we substitute $N_c = 3$. A noticeable part of interference diagram is zero in this limit. Thus this approach a simplifies the calculations. From the other side, the precision of this approach is not so bad, specially in the case of QCD reaction where loop corrections has the same order. See section (3.2.5) about implementation of this option.

C.3 i factor and denominators.

Pro

C.4 Common factors.

The contribution of diagram is multiplied on some factors. They are[17].

- SymmFact=n/d. $n = 1$ if the left part of the squared diagram equals to the right part. Otherwise $n = 2$. The denominator d equals to the number of symmetric permutations of identical outgoing particles.
- AverFact=$1/N_a$, were Na is a number of polarization and color states of the incoming particles.
- FermFact= $(-1)^f$, were f is a total number of incoming fermions and anti-fermions.

[17] In notations of section (5.3).

D Color string basis.

CalcHEP performs averaging/summation over color states of incoming/outgoing quarks and gluons. But in order to describe the hadronization of outgoing particles one needs to specify the color states more detail.

CalcHEP passes the problem of hadronization to other programs like PYTHIA [32]. It its turn PITHYA performs hadronization in the framework of the color string model. According to this model pairs of outgoing partons with opposite color state are jointed into the colorless objects, strings. Partons itself are disposed in the ends on the string and, generally, run in different directions. When distance between parton increases, the color string breaks creating two strings with smaller energies. This process continues till the energy of strings is high enough. The low energy strings are treated as mesons.

Now we would like to explain the relation of this semi-phenomenological picture to the color structure of QCD matrix element. Let us consider some QCD amplitude diagrams without external gluons. Using the rules (27a) -(28a) we can transform the corresponding color diagram to the simple ones where separated quark lines connect $q\bar{q}$ pairs. It exactly corresponds to the color string picture described above.

It is easy to note that these color states are orthogonal only in $N_c \to \infty$ limit. The orthogonality is needed to treat the squared basis coefficients as mutually independent probabilities of production of the corresponding color strings. Thus the color string model should be considered in the framework of $1/N_c$ approximation. In the same approximation one can consider the gluon color state as a $q\bar{q}$ one (28a). Thus gluon is a particle where one color string is finished and another on is stated.

During event sampling CalcHEP generates a phase-space point according to $N_c = 3$ matrix element. In the same time it calculates the coefficients of expansion of $N_c = \infty$ matrix element over the color flow basis. The generated phase space point is accompanied with a color flows with a probability proportional to the squared bases coefficient.

E Built-in distribution functions and beam spectra

E.1 Backscattered photon spectrum

This function describes the spectrum of photons scattered backward from the interaction of laser light with the high energy electron beam:

$$f(x) = \begin{cases} 0, & \text{for } x > x_{max} \\ N(1-x+1/(1-x)\,(1-4x/x_0\,(1-x/(x_0(1-x))))), & \text{for } 0 < x < x_{max} \end{cases}$$

where $x_0 = 4.82$, $x_{max} = x_0/(1+x_0)$, N is a normalization factor.

The above spectrum corresponds to the special initial condition when unpolarized photons are created. See [17] for more details.

E.2 Weizsaecker-Williams approximation

Weizsaecker-Williams approximation is used to describe processes of electro-production in the case of small angle of charged particle scattering. In this case the virtual photon emitted by the scattering particle appears near to the mass shell (see Fig.13). It gives a possibility to reduce the process of electro-production to the photo-production one with an appropriate photon spectrum:

$$f(x) = (q^2\,\alpha/(2\pi))(\log((1-x)/(x^2\delta))(1+(1-x)^2)/x - 2(1-x-\delta\,x^2)/x),$$

where α is the fine structure constant, q is a charge of incoming particle, m is its mass, $\delta = (m/Q_{max})^2$. Q_{max} sets out the region of photon virtuality ($P^2 > -Q^2_{max}$) which contributes to the process. It is assumed that region of large virtuality can be taken into account by direct calculation of electro-production. As a rule this contribution is small enough.

Parameters q, m, and Q_{max} are defined by the user. See [18] for the further explanations. In the case of CalcHEP the Weizsaecker-Williams photon spectrum is available for charged leptons only.

E.3 ISR and Beamstrahlung

ISR (Initial State Radiation) is a process of photon radiation by the incoming electron due to its interaction with other collision particle. The resulting spectrum of electron has been calculated by Kuraev and Fadin [19]. In CalcHEP we realize the similar expression by Jadach, Skrzypek, and Ward [20]:

$$\begin{aligned} F(x) = \;& \exp(\beta(3/4 - Euler))\beta(1-x)^{\beta-1} \\ & ((1+x^2) - \beta((1+3x^2)\ln(x)/2 + (1-x)^2)/2)/(2\Gamma(1+\beta)), \end{aligned}$$

where

α	=	1/137.0359895 is the fine structure constant;
β	=	$\alpha(2\ln(SCALE/m)-1)/pi$
m	=	0.00051099906 is the electron mass;
$Euler$	=	0.5772156649 is the Euler constant;
$\Gamma()$	is	the gamma function;
$SCALE$	is	the energy scale of reaction.

In the Kuraev and Fadin article the parameter SCALE equals to the total energy of the process because they considered the process of direct e^+e^- annihilation. In order to apply this structure function to another processes we provide the user with a possibility to define this parameter.

Beamstrahlung is a process of energy loss by the incoming electron due to its interaction with the electron (positron) bunch moving in the opposite direction. The effective energy spectrum of electron can be described by the following function [21]

$$F(x) = \frac{1}{N_{cl}}[(1-E^{-N_{cl}})\,\delta(1-x) + \frac{\exp(-\eta(x))}{1-x}\,h(\eta(x)^{1/3}, N_{cl})] \quad ,$$

where

$$\begin{aligned} \eta(x) &= 2/(3\Upsilon)\,(1/x-1) \quad , \\ h(z, N_{cl}) &= \sum_{n\geq 1} \frac{z^n}{n!\Gamma(n/3)}\gamma(n+1, N_{cl}) \quad , \end{aligned}$$

and γ is the incomplete gamma function.

Function $F(x)$ depends on two parameters, N_{cl} and Υ, which in their turn are determined by a bunch design:

$$\begin{aligned} N_{cl} &= \frac{25\,\alpha^2 N}{12\,m(\sigma_x+\sigma_y)} \quad , \\ \Upsilon &= \frac{5\,\alpha\,N\,E}{6\,m^3\sigma_z(\sigma_x+\sigma_y)} \,, \end{aligned}$$

where

N	is number particles in the bunch,
$\sigma_x, \sigma_y, \sigma_z$	are sizes of bunch,
E	is a center-of-mass momentum.

The Beamstrahlung spectrum cannot be integrated by the current CalcHEP version because it contains a δ-function. Instead of it we provide the user with a possibility to integrate the squared matrix element with a convolution of Beamstrahlung and ISR spectra.

F External distribution functions and beam spectra

F.1 Structure of pdf-file in CalcHEP

In the process of development of CalcHEP package we met a challenge of providing the user with a possibility to operate with various distribution functions. After some experience of direct embedding of ailing codes into CalcHEP software, it was recognized that the best way should be a development of one standard for date files and realization of unique program that performs the interpolation of data stored into a file.

The structure of the pdf files in CalcHEP is motivated by CTEQ [15] and MRS [16] representations of parton distribution functions. CalcHEP format is described below.

Generally distribution function depends on two arguments. They are the Feynman parameter x and the energy scale Q. x is unitless and run in [0,1] interval. $Q > 0$ and traditionally is presented in GeV unit. Thus one has to describe grids for x and Q variables and to present the table of pdf-function corresponding to these grids.

CalcHEP pdf file contains several items of information, Each item is started by a keyword. All keywords are started from the '#' symbol. Numerical data following a keyword must be separated by white-space characters. The appearance of new keywords means the end of the previous item.

The keywords are

- `#x-grid`
  After this keyword the file has to contain a sequence of numbers which specify the x variable grid. The number of points has exceed 2. Data must increase from point to point.

- `#q-grid`
  Specifies the beginning of Q grid. The data must be positive. Data must increase from point to point. This item is optional. Generally we can consider distribution functions which do not depend on the Q parameter. An artificial Q grid is not needed in this case. For interpolation the $log(Q)$ grid is used indeed.

- `#1-parton, #2-parton ......`
  These keywords mark the beginnings of data for distribution functions. They can not precede the `#x-grid` and `#q-grid` items because the power of data for these items must be the production of the x-grid and q-grid powers. Distribution function for x corresponding to the i_x position of x-grid and Q corresponding to the i_q position of q-grid is disposed on the $(i_x + (i_q - 1)n_x)^{th}$ place of parton distribution.

  Such item can optionally be finished by word 'powers' after that one or two numbers (say r0 and r1) are follow. It means that after interpolation of the data the result will be multiplied on factor $x^{r0} * (1-x)^{r1}$ to get the real parton distribution. If only one number is presented then it corresponds to r0 and r1 is zero.

- `#distribution`
  After this word the title name of distribution function and the list of parton names is expected. The title name has been surrounded by the quotation marks (") symbols. The name may contain space symbols inside. Parton names may be simple and composed. A simple name should corresponds to particle names in CalcHEP designations. A composite name describes a group of partons who have the same distribution function. Say (say, b and $\bar{b}$ quarks into the proton). Partons of one group should be closed into brackets ( ) and separated by the commas.

  One file can contain several (usually two) the `#distribution` items. It allows to use one file both for the particle and the anti-particle. For example:
  `#distribution "CTEQ5M(proton)" (b,B) (c,C) (s,S) D U G u d`

```
#distribution "CTEQ5M(anti-proton)" (b,B) (c,C) (s,S) d u G U D
```

- `#x-min, #x-max, #q-min, #q-max`
  These keywords are used to specify the minimum and maximum boundaries of 'x' and 'q' intervals, where results of interpolation should be correct. An appearance of one number after each of these keywords is expected.

  These keywords are optional and do not influent on the work of the program. Only a warning is written any time when one of limit is broken.

- `#alpha`
  This keywords starts data for alpha(Q) for the Q-grid points. The item is optional and should not precede the #q-grid one.

  The correct usage of QCD parton distributions should be accompanied by the corresponding formula of strong coupling constant alpha(Q), Although the alpha(Q) dependence is presented by simple analytic formula, we include data for this function into the the file as a table. It allows to realize some general approach to mutual evaluation of distribution function and alpha(Q).

- `#mass`
  This is an optional keyword allows to introduce the mass of the composite particle. It is expected that corresponding numerical value follows this keyword. Default value for mass parameter is 1 GeV.

- `#lExtrapolation`
  The presence of this keyword means that linear interpolation has been used. If it doesn't written then quadratic interpolation is used.

  ./n_calchep checks the current (./), the up (../), and the `$CALCHEP` directories. All files with extension "pdf" disposed in these directories are considered as date tables for particle distribution functions.

F.2 *CTEQ5m* distribution functions

Calchep *utile* directory contains the *tbl_cteq_pdf.c* code for program which transforms CTEQ distribution tables to CalcHEP format. This program can

be compiled by
`cc tbl_cteq_pdf.c -lm`
The *a.out* file, which appears in the result of the compilation, operates by the following manner
`a.out <  cteq_file.tbl > calchep_file.pdf`
The file *steq5m.pdf* presented in the `$CALCHEP` directory is created by this way. See further explanations of CTEQ distribution function in the original paper [15].

G Monte Carlo phase space integration

G.1 Parameterization of multi-particle phase space

G.1.1 Parameterization via decay scheme

The element of phase space volume for a n-particle state is equal to [27]

$$d\Gamma_n(q) = (2\pi)^4\delta^4(q - p_1 - p_2 - p_3 - \ldots - p_n)\prod_{i=1}^{n}\frac{\delta(p_i^2 - m_i^2)}{(2\pi)^3}d^4p_i \ . \qquad (31)$$

The same expression is valid for both the decay of unstable particle with momentum q and the interaction of two particles with momenta q_1 and q_2 such that $q_1 + q_2 = q$. For further discussion we need a designation for a phase space volume of some subset S of the full n-particle set. According to (31)

$$d\Gamma(q, S) = (2\pi)^4\delta^4(q - \sum_{i\in S} p_i)\prod_{i\in S}\frac{\delta(p_i^2 - m_i^2)}{(2\pi)^3}d^4p_i \ . \qquad (32)$$

Let S_1 and S_2 be two disjoint particle subsets, then

$$\begin{aligned} & d\Gamma(q, S_1 \cup S_2) = \\ & \quad \int ds_1 ds_2\Big((2\pi)^4\delta^4(q - q_1 - q_2)\frac{\delta(q_1^2 - s_1)}{(2\pi)^3}d^4q_1\frac{\delta(q_2^2 - s_2)}{(2\pi)^3}d^4q_2\Big) \\ & \quad \times\frac{d\Gamma(q, S_1)}{2\pi}\times\frac{d\Gamma(q, S_2)}{2\pi} \ . \end{aligned} \qquad (33)$$

The above formula expresses a multi-particle volume in terms of two-particle one, the volumes $d\Gamma(q_1, S_1)$ and $d\Gamma(q_2, S_2)$ with a reduced number of particles, and the virtual squared masses $s_1,\ s_2$ of clusters $S_1,\ S_2$.

Recursive application of this formula allows one to express the multi-particle phase space in terms of two-particle phase space. In its turn the two-particle phase space is explicitly described by spherical angle Ω of motion of the first decaying particle in the rest frame of initial state [27].

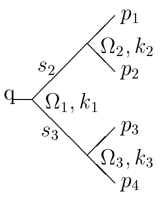
$$\frac{d\Gamma(q,[1,2])}{2\pi} = \frac{kd\Omega}{4(2\pi)^3\sqrt{q^2}} \, , \tag{34}$$

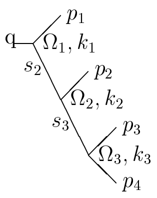
where k is the absolute value of three-dimensional momentum of outgoing particles in the rest frame. Thus, applying recursively (33) and (34) to (31) we obtain an explicit expression for the phase space volume in terms of the squared masses s_j of virtual clusters and the two-dimensional spherical angles Ω_j, where j is an ordinal number of decay:

$$d\Gamma_n(q) = \frac{k_1 d^2\Omega_1}{4(2\pi)^2\sqrt{q^2}} \prod_{j=2}^{n-1} \frac{k_j d^2\Omega_j}{4(2\pi)^3\sqrt{s_j}} \cdot \prod_{j=2}^{n-1} ds_j \tag{35}$$

Here k_j is a momentum of outgoing clusters produced by decay of the j^{th} cluster in its center-of-mass.

The expression (35) means some sequential `1->2` decay scheme which starts from incoming state and finishes with outgoing particles of the process. For example, the integration domain for s_j parameters depends on this scheme. Below we present two such schemes for a process with four outgoing particles:





In the case of CalcHEP project such decay scheme is defined by the user via the 'Kinematics' menu (see Section 3.3.8).

G.1.2 Polar vectors

To complete phase space parameterization we must fix a polar coordinate system choosing the polar and the azimuthal angles for each of decays

$$d^2\Omega_j = d\cos\Theta_j d\Phi_j \tag{36}$$

We have an ambiguity in the choice of polar coordinate. Let us remind that our goal is not only parameterization of phase space but also regularization of the squared matrix element in the phase space manifold. The main idea of such regularization is a cancellation of integrand sharp peaks by the phase space measure. Originally the phase space measure (35) has no cancellation factors, but we can create them by means of a Jacobian of transformed variables. To get an appropriate Jacobian we need to have the initial phase space variables related to poles of the squared matrix element.

In their turn the poles of squared matrix element are caused by virtual particle propagators and generally have one of the forms (2), (3) or (4) (Section 3.3.9) depending on a squared sum of momenta. Variables s_j in (35) are also equal to squared sums of momenta. So, the parameterization (35) allows us to smooth some peaks of the matrix element.

It appears to be that the polar coordinates can be chosen in such a way that all $cos\Theta_j$ have simple linear relations to the squared sums of momenta [14, 10]. The polar angle Θ_j can be unambiguously fixed by the *polar vector* $Pole_j$ whose space components in the rest frame of decay correspond to the $\Theta_j = 0$ direction. Let q_{j1} and q_{j2} be the momenta of the first and the second clusters produced by the j^{th} decay. Then

$$(Pole_j + q_{j1})^2 = (Pole_j^0 + q_{j1}^0)^2 - \mid \overline{Pole}_j \mid^2 - \mid \overline{q}_{j1} \mid^2 - 2cos\Theta_j \mid \overline{Pole}_j \mid\mid \overline{q}_{j1} \mid$$
$$(Pole_j + q_{j2})^2 = (Pole_j^0 + q_{j2}^0)^2 - \mid \overline{Pole}_j \mid^2 - \mid \overline{q}_{j2} \mid^2 + 2cos\Theta_j \mid \overline{Pole}_j \mid\mid \overline{q}_{j2} \mid$$

Thus, in order to get $cos\Theta_j$ related to a squared sum of some particle momenta we may construct the polar vector as a sum of particle momenta [14, 10].

For the non-contradictory construction we need to set the decays in some order with a natural requirement that the sub-decays of clusters produced by the j^{th} decay have the ordinal numbers larger than j. In giving such ordering we can construct a polar vector for each decay based on the incoming momenta and on those of particles produced by decays possessing smaller ordinal numbers.

The following statements can be proved. *In the framework of any ordered scheme of decays and for any sum P of particle momenta one can find the decay number j such that either $P^2 = s_j$ or P might be represented as $Pole_j + q_j$, where q_j is the momentum of one of the clusters in the j^{th} decay and $Pole_j$ is a polar vector constructed according to the above rule.* In other words, any of poles (2), (3), (4) can be expressed either in terms of s_j parameters or in terms some of $cos\Theta_j$ for an appropriate choice of the polar vector [14, 10].

In CalcHEP the ordering is arranged automatically, so that all subdecays of the first cluster have smaller numbers than those of the second cluster. Polar vectors are also constructed automatically according to the list of peaks prepared by the user.

G.1.3 Smoothing

The general idea of the integrand smoothing is trivial. Let us need to evaluate

$$\int_a^b F(x)dx \quad , \tag{37}$$

and let $F(x)$ have a peak like $f(x)$, where $f(x)$ is a simple symbolically integrable function in contrast to $F(x)$:

$$g(x) = \int_a^x f(x')dx' \; . \tag{38}$$

Now we may represent the integral (37) as

$$\int_a^b F(x)dx = \int_0^{g(b)} dy \frac{F(g^{-1}(y))}{f(g^{-1}(y))}, \tag{39}$$

where $g^{-1}(y)$ is the inverse function for $g(x)$. The integrand is a smooth function now.

We face very often squared matrix elements which have several poles in one of variables. For example, the $\gamma \to b, \bar{b}$, $Z \to b, \bar{b}$ and $H \to b, \bar{b}$ virtual subprocesses may contribute just to the same amplitude. Although in this case we can evaluate the integral function $g(x)$ symbolically, the inverse function $g^{-1}(y)$ can be computed only as a numerical solution of the corresponding equation. To bypass the calculation of inverse function CalcHEP uses the multi-channel Monte Carlo (branching) method to smooth a sum of peaks.

The idea of the branching method is the following. Let $F(x)$ have two peaks, one is similar to $f_1(x)$ and another to $f_2(x)$. $f_1(x)$ and $f_2(x)$ are singular but elementary functions. Then, instead of one integration (37), we could perform two ones:

$$\int F(x)dx = \int \frac{F(x)f_1(x)}{f_1(x)+f_2(x)}dx + \int \frac{F(x)f_2(x)}{f_1(x)+f_2(x)}dx \; , \tag{40}$$

but now each integration has only a single peak! It is easy to extend this method for an arbitrary number of peaks.

The branching method was used in [38] to separate peaks which came from various diagrams. In that paper there was also proposed to use the expression (40) where $f_i(x)$ is replaced by $\alpha_i f_i(x)$ with a subsequent search for optimal coefficients α_i. CalcHEP passes on this weight optimization to *Vegas*, combining two integrals in one *Vegas* hypercube.

As was mentioned above, CalcHEP automatically searches for a polar vector for each angle integration in order to reach a linear relation between $cos\Theta$ and one of the squared sum of momenta which is responsible for the peak. It could happen that various peaks need different polar vectors for the same decay. In this case CalcHEP uses the branching method again, but now for the whole two-dimension sphere integration. In other words, we use the branching equation (40) where x is the two dimensional sphere angle [14, 10].

G.2 Adaptive Monte Carlo integration package *Vegas*

This section contains a short description of the adaptive Monte Carlo program VEGAS. See for details [22, 23].

The Monte Carlo method reduces a task of integral evaluation to the task of mean value calculation. Let $g(x)$ is a density function satisfying

$$\int g(x)\,dx = 1,$$

then

$$\int f(x)\,dx = \int f(x)/g(x)\;g(x)\,dx = <f/g> = \lim_{N\to\infty} \sum (f(x_i)/g(x_i))/N,$$

where points x_i are sampled with the probability density $g(x)\,dx$.

The uncertainty σ_N of $<f/g>$ estimation by N sample points is proportional to square root of function's variance divided over N:

$$\sigma_N = \sqrt{(<(f/g)^2> - <f/g>^2)/N}\ .$$

VEGAS uses two techniques which allow to decrease the uncertainty of Monte Carlo calculation, namely the *importance sampling* and the *stratified sampling*.

G.2.1 Importance sampling

The idea of importance sampling technique is based on diminution of variance by a proper choice of the density function g(x). The general solution of this problem could be in choosing

$$g(x) = |f(x)| \,/ \int |f(x)|\, dx.$$

However this solution is useless because it returns us to the problem of evaluation of $f(x)$ integral and requires a generation of sampling points for complicated density function.

To bypass these problems VEGAS seeks this function in the factored form

$$g(x_1, x_2, \ldots, x_n) = g_1(x_1)\, g_2(x_2) \ldots g_n(x_n).$$

The optimal functions $g_i(x)$ could be easily evaluated in terms of $f(x)$ [22, 23]. VEGAS is an adaptive program. For the first iteration it puts $g_i(x) = 1$. The information about $f(x)$ which VEGAS gets during the iteration is used to refine the density function. Generally VEGAS performs several iterations improving the density function after each of them.

The following parameters manage VEGAS work:

1. *Itmx* is a number of iterations;

2. *Ncall* is a number of integrand calls for one iteration.

G.2.2 Stratified sampling

The idea of stratified sampling method is to divide a volume of integration into a large number of sub-volumes and calculate integrals separately in each

sub-volume. This method produces a smaller uncertainty comparing with the direct Monte Carlo method because here the uncertainty is caused only by a function variance in the sub-volumes, while the integrand variation from one sub-volume to another does not contribute to the uncertainty.

The stratified sampling method is used to estimate the integral for any VEGAS iteration. The larger number Ncall is chosen, the smaller size of sub-volume becomes available and, consequently, the more successfully the stratified sampling works.

G.3 Generation of events

CalcHEP generates events according to the Von Neumann algorithm. See [27], p.202. Let the probability density $f(x)$ is smaller that an easily generated density $F(x)$[18]. Then one can generate x according to distribution $F(x)$ and accept this event with probability $f(x)/F(x)$. This procedure is repeated in cycle until the needed number of events is generated.

To built $F(x)$ CalcHEP divides the space volume on large number of sub-cubes and in each sub-cube sets $F(x)$ a constant which equals to $\max f(x)$. CalcHEP has two strategies of detecting the corresponding maxima. First one is a random search. The program generates random points in each sub-cube and tests $f(x)$ in these points. The second one is a search by the simplex method [23]. Here the program analyzes function in vertices of some simplex and tries to shift one vertex of this simplex to increase the function. This method leads to fast converges to local maximum, but one has to take into account that the distribution function can have several local maxima on the cub-cube boundary. Thus, preliminary random search needs to define a good start point for the search by the simplex method. The number of calls for random search and the number of steps for simplex search are defined by the user.

In general, the detected maxima are lower than the true ones. To satisfy the inequality

$$f(x) \le F(x)$$

the function $F(x)$ based on the detected maxima may be multiplied by some factor, say 2. Of course, it decreases the efficiency of the generator just on the same factor. Nevertheless, in some sub-cubes were the variance of

[18] We assume $f(x)$ and $F(x)$ are not normalized.

the function is large this factor may be not enough. If CalcHEP finds a point x where $f(x) > F(x)$ it accompany point with an integer weight w. This weight is the integral part of $f(x)/F(x)$ plus one with the probability equal to the fraction part of $f(x)/F(x)$. From view point of calculation of various distributions one event with integer weight w should be treated as w independent events with identical parameters. But for the evaluation of statistical uncertainties a more careful treatment is needed.

H Lagrangian of the Standard Model

H.1 Gauge theories

Group. Group is defined by its structure constants $f^{\alpha}_{\beta\gamma}$ which are anti-symmetric ($f^{\alpha}_{\beta\gamma} = -f^{\alpha}_{\gamma\beta}$) and obey the Jacoby identity:

$$f^{\delta}_{\alpha\epsilon} f^{\epsilon}_{\beta\gamma} + f^{\delta}_{\gamma\epsilon} f^{\epsilon}_{\alpha\beta} + f^{\delta}_{\beta\epsilon} f^{\epsilon}_{\gamma\alpha} = 0 \; .$$

Group generators are Hermitian matrices $\hat{\tau}_{\alpha}$ which satisfy the commutation relations:

$$\hat{\tau}_{\alpha}\hat{\tau}_{\beta} - \hat{\tau}_{\beta}\hat{\tau}_{\alpha} = i\, f^{\gamma}_{\alpha\beta}\hat{\tau}_{\gamma} \; .$$

In particular the generators in the adjoint representation are

$$(\hat{\tau}_{\alpha})^{i}_{j} = -i\, f^{i}_{j\alpha} \; .$$

Group transformation may be represented with the help of group generators as

$$\hat{g}(w) = exp(i\hat{\tau}_{\alpha}\omega^{\alpha}) \; .$$

We assume that the Killing metric is orthonormal:

$$-\frac{1}{2} f^{\epsilon}_{\gamma\alpha} f^{\gamma}_{\epsilon\beta} = \delta_{\alpha\beta} \; .$$

This metric allows one to raise and lower the group indices. In the case of orthonormal Killing metric the structure constants are fully antisymmetric under interchange of any pair of indices.

Local gauge invariance. It is an invariance of Lagrangian under group transformations $g(x)$ which depend on a point in the space-time manifold. Fields in such theories are divided into two classes: matter fields and gauge fields. The gauge fields $A^{\alpha}_{\mu}(x)$ are vector ones. The number of such fields is equal to the number of group generators. The matter fields $\psi^{i}(x)$ can have an arbitrary Lorentz structure. Their internal components are transformed according to some representation of the group. Let $D(\omega)$ be an operator which performs an infinitesimal transformation of fields under the local gauge transformation. For the matter fields we have

$$\hat{D}(\omega)\psi(x) = i\,\omega^{\alpha}(x)\hat{\tau}_{\alpha}\psi(x) \; .$$

For gauge fields the local gauge transformations are defined by

$$(\hat{D}(\omega)A_{\mu})^{\alpha}(x) = f^{\alpha}_{\beta\gamma}A^{\beta}_{\mu}(x)\omega^{\gamma}(x) + \partial_{\mu}\omega^{\alpha}(x) \; .$$

The following expressions, namely a covariant derivative and a gauge field tension, are used to construct a local invariant Lagrangian:

$$\nabla_{\mu}\psi(x) = \partial_{\mu}\psi(x) - \hat{D}(A_{\mu})\psi(x) = \partial_{\mu}\psi(x) - iA^{\alpha}_{\mu}(x)\hat{\tau}_{\alpha}\psi(x) \; ;$$

$$F^{\alpha}_{\mu\nu}(x) = \partial_{\mu}A^{\alpha}_{\nu}(x) - \partial_{\nu}A^{\alpha}_{\mu}(x) + f^{\alpha}_{\beta\gamma}A^{\beta}_{\mu}(x)A^{\gamma}_{\nu}(x) \; .$$

It can be proved that

$$\hat{D}(\omega)[\nabla_{\mu}\psi(x)] = \hat{\tau}_{\alpha}\omega^{\alpha}(x)\nabla_{\mu}\psi(x) \; ;$$

$$(\hat{D}(\omega)F_{\mu\nu})^{\alpha}(x) = f^{\alpha}_{\beta\gamma}F^{\beta}_{\mu\nu}(x)\omega^{\gamma}(x) \; .$$

In these terms the Lagrangian of gauge theory is defined by the following expression [34]

$$L = -\frac{1}{4g^2}F^{\alpha}{}_{\mu\nu}F_{\alpha}{}^{\mu\nu} + L_m(\nabla_{\mu}\psi, \psi) \qquad (41)$$

where g is the coupling constant and $L_m(\partial_{\mu}\psi, \psi)$ is some Lagrangian of the matter fields which is invariant under the global gauge transformations.

Gauge fixing terms. In order to quantize the gauge theory one must add to (41) gauge fixing term and the corresponding Faddeev-Popov term. The first term breaks the gauge symmetry and in this way removes the divergence of the functional integral. The second term improves the integration measure to provide correct predictions for gauge invariant observable.

The general form of the gauge fixing term is

$$L_{FG}(x) = -\frac{1}{2}\sum_{\alpha}(\Phi^{\alpha}(x))^2 \ . \tag{42}$$

The corresponding Faddeev-Popov term is

$$L_{FP} = -\bar{c}_{\alpha}(x)(D(c)[\Phi^{\alpha}])(x) \ , \tag{43}$$

where $c^{\alpha}(x)$ and $\bar{c}_{\alpha}(x)$ are the auxiliary anti-commutative fields. They are called the Faddeev-Popov ghosts. Note that we may multiply (43) by an arbitrary factor which can be hidden in the definition of ghost fields. As a rule it is chosen in such a way to provide a convenient form of the ghost propagator.

The well-known choice of the gauge fixing terms, the Feynman-like gauge, is

$$L_{GF} = -\frac{1}{2g^2}\sum_{\alpha}(\partial^{\mu}A^{\alpha}_{\mu} + ...)^2 \ .$$

In this case the quadratic part of gauge field Lagrangian takes the simplest form

$$-\frac{1}{2g^2}\partial_{\mu}A_{\nu}\partial^{\mu}A^{\nu} \ . \tag{44}$$

The corresponding Faddeev-Popov Lagrangian is

$$L_{FP} = -\bar{c}_{\alpha}(x)(\Box c^{\alpha}(x) + \partial^{\mu}(f^{\alpha}_{\beta\gamma}A^{\beta}_{\mu}(x)c^{\gamma}(x)) + ...) \ .$$

The normalization of L_{FP} is chosen to have the Faddeev-Popov ghost propagator equal to the propagator of scalar particle:

$$T(\bar{c}_{\alpha}(p), c^{\beta}(q)) = \delta^{\beta}_{\alpha}\delta(p+q)\frac{1}{i(2\pi)^4}\,\frac{1}{-p^2} \ .$$

Normalization. For the purposes of applying the perturbation theory the gauge field is rescaled by substitution

$$A \to A/g \ .$$

In this way the coupling constant g leaves the quadratic part of the Lagrangian (41) and appears in the interaction terms.

H.2 QCD Lagrangian

QCD is a gauge theory based on the $SU(3)$ group. The corresponding gauge field $G^{\alpha}_{\mu}(x)$ is called a *gluon*. The matter fields $q_k(x)$ are *quarks*. They are triplets and are transformed according to the fundamental representation. Index k enumerates a sort of quarks.

QCD Lagrangian in the Feynman gauge is written down following the rules (41), (42), (43).

$$L_{phys} = -\frac{1}{4}F^{\alpha}{}_{\mu\nu}(x)F_{\alpha}{}^{\mu\nu}(x) + \sum_k \frac{i}{2}(\bar{q}_k(x)\gamma^{\mu}\nabla_{\mu}q_k(x) - \nabla_{\mu}\bar{q}_k(x)\gamma^{\mu}q_k(x)) \ . \tag{45}$$

where

$$\nabla_{\mu}q(x) = \partial_{\mu}q(x) - i\,g\,G^{\alpha}_{\mu}(x)\hat{t}_{\alpha}q(x) \ ;$$

$$F^{\alpha}_{\mu\nu}(x) = \partial_{\mu}G^{\alpha}_{\nu}(x) - \partial_{\nu}G^{\alpha}_{\mu}(x) + g\,f^{\alpha}_{\beta\gamma}G^{\beta}_{\mu}(x)G^{\gamma}_{\nu}(x) \ ;$$

$$L_{GF} = -\frac{1}{2}\sum_{\alpha}(\partial^{\mu}G^{\alpha}_{\mu})^2 \ ; \tag{46}$$

$$L_{FP} = -\bar{c}_{\alpha}(x)(\Box c^{\alpha}(x) + g\,\partial^{\mu}(f^{\alpha}_{\beta\gamma}G^{\beta}_{\mu}(x)c^{\gamma}(x))) \ , \tag{47}$$

$c^{\alpha}(x)$ are Faddeev-Popov ghosts, g is coupling constant, $f^{\alpha}_{\beta\gamma}$ are the $SU(3)$ structure constants, t_{α} are the generators in the fundamental representation[19].

[19] t_{α} are equal to $\lambda_{\alpha}/2$ where λ_{α} are the Gell-Mann matrices.

H.3 Lagrangian of electroweak interactions

H.3.1 Vector bosons

Gauge theory of electroweak interactions is based on the $SU(2)\times U(1)$ group. So, we have a triplet of $SU(2)$ vector fields $W^\alpha_\mu(x)$ and a single vector field $B_\mu(x)$. The $SU(2)$ structure constants are presented by the absolutely antisymmetric tensor $\epsilon^{\alpha\beta\gamma}$. The Lagrangian of gauge fields is written down according to (41):

$$L_{GF} = -\frac{1}{4}FW^\alpha{}_{\mu\nu}(x)FW_\alpha{}^{\mu\nu}(x) - \frac{1}{4}FB_{\mu\nu}(x)FB^{\mu\nu}(x)\ , \tag{48}$$

where

$$\begin{aligned} FW^\alpha_{\mu\nu}(x) &= \partial_\mu W^\alpha_\nu(x) - \partial_\nu W^\alpha_\mu(x) + g_2\epsilon^{\alpha\beta\gamma}W^\beta_\mu(x)W^\gamma_\nu(x)\ ; \\ FB_{\mu\nu}(x) &= \partial_\mu B_\nu(x) - \partial_\nu B_\mu(x)\ ; \end{aligned}$$

g_2 is a coupling constant for the $SU(2)$ gauge interaction.

Infinitesimal local gauge transformations are defined as follows

$$\begin{aligned} (\hat{D}(w,b)W_\mu)^\alpha(x) &= g_2\epsilon^{\alpha\beta\gamma}W^\beta_\mu(x)w^\gamma(x) + \partial_\mu w^\alpha(x)\ ; \\ D(w,b)B_\mu(x) &= \partial_\mu b(x)\ . \end{aligned} \tag{49}$$

Let us express W^1_μ and W^2_μ in terms of a mutually conjugated couple W^+_μ and W^-_μ

$$\begin{aligned} W^1_\mu &= (W^-_\mu + W^+_\mu)/\sqrt{2}\ ; \\ W^2_\mu &= (W^-_\mu - W^+_\mu)/(i\sqrt{2})\ . \end{aligned}$$

Thus, the Lagrangian of self-interaction for the $SU(2)$ gauge fields in term of W^+ and W^- has the form:

$$\begin{aligned} & i\,g_2\Big(\partial_\mu W^3_\nu(W^+_\mu W^-_\nu - W^-_\mu W^+_\nu) + \partial_\mu W^+_\nu(W^-_\mu W^3_\nu - W^3_\mu W^-_\nu) \\ & \qquad\qquad -\partial_\mu W^-_\nu(W^+_\mu W^3_\nu - W^3_\mu W^+_\nu)\Big) \\ & +g_2^2\Big(W^3_\mu W^+_\nu(W^-_\mu W^3_\nu - W^3_\mu W^-_\nu) + \frac{1}{2}W^+_\mu W^-_\nu(W^+_\mu W^-_\nu - W^-_\mu W^+_\nu)\Big)\ . \end{aligned} \tag{50}$$

All matter fields in the electroweak theory are either $SU(2)$ invariant singlets or belong to its fundamental representation. In the latter case they

form doublets. Generators for these doublets are expressed via the Pauli σ-matrices

$$\hat{\tau}_\alpha = \hat{\sigma}_\alpha/2\ .$$

Thus the infinitesimal local gauge transformations for doublets take a form:

$$D(w,b)\psi(x) = \frac{i\,g_2}{2}w^\alpha(x)\hat{\sigma}_\alpha\psi(x) + \frac{i\,g_1}{2}Y\,b(x)\psi(x)\ .$$

Here g_1 is the coupling constant of $U(1)$ gauge interaction. The constant Y depends on a type of the doublet. It is called a *hypercharge*.

In the gauge theory of electroweak interaction the gauge fields interact with a scalar (*Higgs*) doublet which has a nonzero vacuum state. Without loss of generality one can put $Y = 1$ for the Higgs doublet. By means of the gauge transformation the vacuum state of this field may be presented in the form:

$$\Phi_\Omega = \begin{pmatrix} 0 \\ \phi_0/\sqrt{2} \end{pmatrix}\ ,$$

where ϕ_0 is a real constant.

As a result of spontaneous symmetry breaking the W^α_μ and B_μ fields do not correspond to physical particles. Physical particles in this model are the photon (A_μ), W-bosons (W^+_μ, W^-_μ) and Z-boson (Z_μ). The photon field A_μ is a combination of gauge fields responsible for the local gauge transformations which save the Higgs vacuum Φ_Ω:

$$A_\mu = B_\mu\cos\Theta_w + W^3_\mu\sin\Theta_w\ ,$$

where the mixing angle $\Theta_w = \arctan(g_2/g_1)$. To complete W^+, W^-, and A up to the orthonormal basis of gauge fields we introduce

$$Z_\mu = -B_\mu\sin\Theta_w + W^3_\mu\cos\Theta_w\ .$$

Let $w^+(x)$, $w^-(x)$, $a(x)$, and $z(x)$ be parameters of the local gauge transformation corresponding to the fields W^+_μ, W^-_μ, A_μ, and Z_μ. Then for a matter doublet with a hypercharge Y the gauge transformation is given by the following expression:

$$\begin{aligned} & \hat{D}(w^+,w^-,a,z)\Psi = \\ & \quad \frac{i\,g_2}{2}\Bigg[\sqrt{2}\begin{pmatrix} 0 & w^+ \\ w^- & 0 \end{pmatrix} + \sin\Theta_w\,a\begin{pmatrix} 1+Y & 0 \\ 0 & Y-1 \end{pmatrix} \end{aligned}$$

$$+ \quad \cos\Theta_w \, z \begin{pmatrix} 1 - Y\tan^2\Theta_w & 0 \\ 0 & -1 - Y\tan^2\Theta_w \end{pmatrix} \Bigg] \Psi \ . \qquad (51)$$

H.3.2 Lagrangian of Higgs field

In the framework of renormalizable field theory the general expression for the gauge invariant Higgs Lagrangian is:

$$L_{Higgs} = (\nabla_\mu\Phi)^+(\nabla^\mu\Phi) - \frac{\lambda}{2}(\Phi^+\Phi - \frac{1}{2}\phi_0^2)^2 \ ,$$

where λ is a new coupling constant.

Some perturbations of the Higgs vacuum can be realized by means of the local gauge transformation which do not correspond to physical degrees of freedom. To separate physical and gauge degrees of freedom for a small perturbation of the Higgs vacuum we present Φ in the following form:

$$\Phi = \begin{pmatrix} i\,W_f^+ \\ (\phi_0 + H - i\,Z_f)/\sqrt{2} \end{pmatrix} \ . \qquad (52)$$

Here W_f^+ and Z_f are the unphysical *Goldstone* fields corresponding to variation of the Higgs vacuum caused by the $D(w^+,0,0,0)$ and $D(0,0,0,z)$ gauge transformations. The real field H corresponds to a physical degree of freedom which is associated with the Higgs particle.

The term $(-\frac{\lambda}{2}(\Phi^+\Phi - \frac{1}{2}\phi_0^2)^2)$ contains the mass term for the H field:

$$-\frac{\lambda}{2}(\phi_0)^2\,H^2 \ ,$$

and the following terms of self-interaction for H and the Goldstone fields:

$$-\frac{\lambda}{2}\Big((W_f^+W_f^- + (H^2+Z_f^2)/2)^2 + 2\phi_0 H(W_f^+W_f^- + (H^2+Z_f^2)/2)\Big) \ . \qquad (53)$$

Usually λ is expressed via the mass of H-boson M_H:

$$\lambda = \left(\frac{M_H}{\phi_0}\right)^2 \ .$$

Let us remind that

$$\nabla_\mu\Phi = \partial_\mu\Phi - \hat{D}(W+_\mu, W-_\mu, A_\mu, Z_\mu)\Phi \ \ ,$$

where due to (51)

$$\hat{D}(W+_\mu, W-_\mu, A_\mu, Z_\mu)\Phi =$$
$$\frac{i\,g_2}{2}\begin{pmatrix} W_\mu^+(\phi_0 + H - i\,Z_f) + i\,(2\sin\Theta_w A_\mu + \cos\Theta_w(1-\tan^2\Theta_w)Z_\mu)\,W_f^+ \\ (2iW_\mu^-W_f^+ - Z_\mu(\phi_0 + H - i\,Z_f)/\cos\Theta_w)/\sqrt{2} \end{pmatrix} \ . \qquad (54)$$

The term $(\hat{D}_\mu\Phi)^+(\hat{D}^\mu\Phi)$ gives us the mass terms for the W and Z-bosons:

$$\left(\frac{g_2\phi_0}{2}\right)^2 W_\mu^+W^{-\mu} + \frac{1}{2}\left(\frac{g_2\phi_0}{2\cos\Theta_w}\right)^2 Z_\mu Z^\mu \ , \qquad (55)$$

and the following terms describing the interaction of a couple of vector bosons with Higgs and Goldstones:

$$\begin{aligned} \frac{g_2^2}{4}(\quad & W_\mu^+W^{-\mu}(H^2 + Z_f^2 + 2\phi_0 H + 2W_f^+W_f^-) \\ & +4A_\mu A^\mu \sin^2\Theta_w W_f^+W_f^- \\ & +\frac{Z_\mu Z^\mu}{\cos^2\Theta_w}((H^2+Z_f^2)/2 + \phi_0 H + (1-2\sin^2\Theta_w)^2 W_f^+W_f^-) \\ & +4A_\mu Z^\mu \tan\Theta_w(1-2\sin^2\Theta_w)W_f^+W_f^- \\ & -2i\tan\Theta_w W_\mu^+(\cos\Theta_w A^\mu - \sin\Theta_w Z^\mu)(\phi_0 + H - i\,Z_f)W_f^- \\ & +2i\tan\Theta_w W_\mu^-(\cos\Theta_w A^\mu - \sin\Theta_w Z^\mu)(\phi_0 + H + i\,Z_f)W_f^+ \\ ) \ . & \end{aligned} \qquad (56)$$

The constant ϕ_0 now can be expressed in terms of the W-boson mass M_W:

$$\phi_0 = 2\,M_W/g_2 \ .$$

The Z-boson mass is related to the W-boson mass by means of constraint:

$$M_Z = M_W/\cos\Theta_w \ .$$

The term $(-(\hat{D}_\mu\Phi)\partial^\mu\Phi^+ - (\hat{D}_\mu\Phi)^+\partial^\mu\Phi)$ gives us off-diagonal quadratic terms:

$$-g_2\phi_0\left(\frac{1}{2}(W_\mu^+\partial^\mu W_f^- + W_\mu^-\partial^\mu W_f^+) + \frac{1}{\cos\Theta_w}Z_\mu\partial^\mu Z_f\right) \qquad (57)$$

and the following terms of interaction:

$$\begin{aligned}\frac{g_2}{2}(\quad & -H(W_\mu^-\partial^\mu W_f^+ + W_\mu^+\partial^\mu W_f^-) + (\partial^\mu H)(W_\mu^- W_f^+ + W_\mu^+ W_f^-)\\ & -i(Z_f(W_\mu^-\partial^\mu W_f^+ - W_\mu^+\partial^\mu W_f^-) + (\partial^\mu Z_f)(W_\mu^+ W_f^- - W_\mu^- W_f^+))\\ & +i(2\sin\Theta_w A_\mu + \cos\Theta_w(1-\tan^2\Theta_w)Z_\mu)(W_f^-\partial^\mu W_f^+ - W_f^+\partial^\mu W_f^-)\\ & +(\partial^\mu H)Z_\mu Z_f/\cos\Theta_w - (\partial^\mu Z_f)Z_\mu H/cos\Theta_w\\ ) \ . & \qquad (58)\end{aligned}$$

The off-diagonal quadratic terms are canceled by the gauge fixing terms. See below.

H.3.3 Gauge fixing and ghost terms for the t'Hooft-Feynman gauge

In the case of t'Hooft-Feynman gauge the gauge fixing terms are

$$-\frac{1}{2}(\partial^\mu A_\mu)^2 - \frac{1}{2}(\partial^\mu Z_\mu + M_Z\, Z_f)^2 - |\partial^\mu W_\mu^+ + M_W\, W_f^+|^2 \ .$$

The squared divergences of fields transform the quadratic part of Lagrangian for vector bosons to a diagonal form like (44). The squared Goldstone field terms give a mass to the Goldstone particle equal to the mass of the corresponding vector boson field. The off-diagonal quadratic terms, which follow from the gauge fixing Lagrangian, cancel the off-diagonal terms (57) up to complete divergence terms.

According to the general rule (43) the Faddeev-Popov Lagrangian is

$$\begin{aligned}& -A_{\bar c} D(W^+c, W_c^-, A_c, Z_c)(\partial^\mu A_\mu)\\ & -Z_{\bar c} D(W^+c, W_c^-, A_c, Z_c)(\partial^\mu Z_\mu + M_Z\, Z_f)\\ & -W_{\bar c}^- D(W^+c, W_c^-, A_c Z_c)(\partial^\mu W_\mu^+ + M_W\, W_f^+)\\ & -W_{\bar c}^+ D(W^+c, W_c^-, A_c, Z_c)(\partial^\mu W_\mu^- + M_W\, W_f^-) \ .\end{aligned}$$

Note that due to (49)

$$\begin{aligned}\hat D W_\mu^+ &= i\, g_2\, (W_\mu^+(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^+(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu)) + \partial_\mu W_c^+ \ ;\\ \hat D A_\mu &= -i\, g_2 \sin\Theta_w (W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu A_c \ ;\\ \hat D Z_\mu &= -i\, g_2 \cos\Theta_w (W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu Z_c \ ,\end{aligned}$$

and according to (54)

$$\begin{aligned}D W_f^+ &= W_c^+ M_W + \frac{g_2}{2}(W_c^+(H - i\, Z_f) + i\,(2\sin\Theta_w A_c + \cos\Theta_w(1-\tan^2\Theta_w)Z_c)\, W_f^+) \ ;\\ D Z_f &= M_Z Z_c + \frac{g_2}{2}(\frac{Z_c H}{\cos\Theta_w} - i(W_c^- W_f^+ - W_c^+ W_f^-)) \ .\end{aligned}$$

After substitution of these derivatives to the Faddeev-Popov Lagrangian we see that it contains the quadratic part:

$$-A_{\bar c}\Box A_c - Z_{\bar c}\Box Z_c - W_{\bar c}^-(\Box W_c^+ + M_W W_c^+) - W_{\bar c}^+(\Box W_c^- + M_W W_c^-) \ ,$$

and the following vertices of interaction:

$$\begin{aligned}-g_2(\quad & i\,\sin\Theta_w(\partial^\mu A_{\bar c})(W_\mu^+ W_c^- - W_\mu^- W_c^+)\\ & +i\,\cos\Theta_w(\partial^\mu Z_{\bar c})(W_\mu^+ W_c^- - W_\mu^- W_c^+)\\ & +\frac{M_Z Z_{\bar c}}{2}\left(\frac{Z_c H}{\cos\Theta_w} - i(W_c^- W_f^+ - W_c^+ W_f^-)\right)\\ & -i(\partial^\mu W_{\bar c}^-)(W_\mu^+(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^+(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu))\\ & +\frac{M_W W_{\bar c}^-}{2}(W_c^+(H - i\, Z_f) + i\,(2\sin\Theta_w A_c + \cos\Theta_w(1-\tan^2\Theta_w)Z_c)\, W_f^+)))\\ & +i(\partial^\mu W_{\bar c}^+)(W_\mu^-(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^-(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu))\\ & +\frac{M_W W_{\bar c}^+}{2}(W_c^-(H + i\, Z_f) - i\,(2\sin\Theta_w A_c + \cos\Theta_w(1-\tan^2\Theta_w)Z_c)\, W_f^-)))\\ ) \ . & \qquad (59)\end{aligned}$$

H.3.4 Unitary gauge

The *unitary* gauge may be considered as a limit of $\xi \to \infty$ of the following gauge fixing Lagrangian:

$$-\frac{1}{2}(\partial^\mu A_\mu)^2 - \frac{M_Z^2}{2\xi}(Z_f)^2 - \frac{M_W^2}{\xi}(W_f^+ W_f^-) \ .$$

In this limit one gets $Z_f = 0$ and $W_f = 0$, what decreases a number of vertices. Under these constraints the Faddeev-Popov Lagrangian (43) takes the form:

$$
\begin{aligned}
&-A_{\bar{c}}\partial^\mu(-i\, g_2 \sin\Theta_w (W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu A_c) - Z_{\bar{c}} M_Z \, (M_Z + \frac{g_2 H}{2\cos\Theta_w}) Z_c \\
&\qquad -W_{\bar{c}}^- M_W (M_W + \frac{g_2}{2} H) W_c^+ - W_{\bar{c}}^+ M_W (M_W + \frac{g_2}{2} H) W_c^- \; .
\end{aligned}
$$

Integration over A_c and $A_{\bar{c}}$ can be performed explicitly and gives a result which does not depend on other fields. So A_c and $A_{\bar{c}}$ ghosts may be omitted.

In the unitary gauge only physical polarization states of the incoming and outgoing $W^\pm$ and Z bosons are considered. So $W_c^\pm$ and Z_c are not needed in the external lines of Feynman diagrams and may be omitted also in tree level calculations. Consequently, in the unitary gauge all ghost and Goldstone fields may be omitted.

H.3.5 Summary of vertices for the boson sector

In the case of t'Hooft-Feynman gauge the full set of vertices is described by expressions (50), (53), (56), (58), and (59), where

$$
\begin{aligned}
W_\mu^3 &= \sin\Theta_w A_\mu + \cos\Theta_w Z_\mu \; ; \\
\lambda &= \left(\frac{g_2 M_H}{2 M_w}\right)^2 \; .
\end{aligned}
$$

The coupling constants g_2 and g_1 may be expressed in terms of the electromagnetic coupling constant: $g_2 = e \sin\Theta_w$ and $g_1 = e\cos\Theta_w$.

In the case of unitary gauge the interaction is defined by a subset of vertices which appears after removing the Faddeev-Popov ghosts and the Goldstone fields.

H.3.6 Interaction of vector bosons with fermions

Experiments in particle physics show that the W-bosons interact with left-handed components of fermions

$$
\Psi^L = \frac{1-\gamma^5}{2}\Psi \; .
$$

Thus, the $SU(2)$ group must transform only the left-handed fermion components. The initial Lagrangian of fermion field has the corresponding global

symmetry only if all fermions are massless. Indeed, the Lagrangian of free massless fermion field splits into two independent parts:

$$
\begin{aligned}
L &= \frac{i}{2}(\bar{\Psi}\gamma_\mu \partial^\mu \Psi - (\partial^\mu \bar{\Psi})\gamma_\mu \Psi) \\
&= \frac{i}{2}(\bar{\Psi}^L \gamma_\mu \partial^\mu \Psi^L - (\partial^\mu \bar{\Psi}^L)\gamma_\mu \Psi^L) + \frac{i}{2}(\bar{\Psi}^R \gamma_\mu \partial^\mu \Psi - (\partial^\mu \bar{\Psi}^R)\gamma_\mu \Psi^R) \; ,
\end{aligned}
$$

which allows to apply the $SU(2)$ gauge transformations to the left-handed components of fermion doublets. In the same time the mass term contains a product of left-handed and right-handed fermion components:

$$
m\bar{\Psi}\Psi = m(\bar{\Psi}^L \Psi^R + \bar{\Psi}^R \Psi^L) \; ,
$$

what forbids an appearance of such terms in the invariant Lagrangian. Later on we shall show how such fermion particles will acquire masses in result of the gauge invariant interaction of formerly massless fermion fields with the Higgs doublet.

Whereas $SU(2)$ transforms only the left-handed components of doublets, $U(1)$ interacts with both. Left-handed and right-handed components of fermions must have the same electric charge. It allows to find the hypercharge of right-handed components if a hypercharge of left-handed doublet is known. The $U(1)$ gauge field B_μ is equal to $\cos\Theta_w A_\mu - \sin\Theta_w Z_\mu$. So, the electromagnetic coupling constant for right-handed fermion with a hypercharge Y^R is $(g_1 Y^R \cos\Theta_w)/2$. Comparing it with the expression (51) we see that the hypercharges of right-handed components of doublet are

$$
\begin{aligned}
Y_1^R &= 1 + Y \; , \\
Y_2^R &= Y - 1 \; .
\end{aligned}
$$

Thus, we may unambiguously write down the vertices of interactions associated with the covariant derivative:

$$
\begin{aligned}
&\frac{g_2}{2}\begin{pmatrix} \bar{\Psi}_1 & \bar{\Psi}_2 \end{pmatrix} \left[\sqrt{2}\begin{pmatrix} 0 & W_\mu^+ \\ W_\mu^- & 0 \end{pmatrix} + \sin\Theta_w A_\mu \begin{pmatrix} 1+Y & 0 \\ 0 & Y-1 \end{pmatrix} \right. \\
+ &\left. \cos\Theta_w Z_\mu \begin{pmatrix} 1 - Y\tan^2\Theta_w & 0 \\ 0 & -1 - Y\tan^2\Theta_w \end{pmatrix} \right] \begin{pmatrix} \gamma^\mu \Psi_1^L \\ \gamma^\mu \Psi_2^L \end{pmatrix} \\
+ &\frac{g_1}{2}(\cos\Theta_w A_\mu - \sin\Theta_w Z_\mu) \begin{pmatrix} \bar{\Psi}_1 & \bar{\Psi}_2 \end{pmatrix} \begin{pmatrix} Y+1 & 0 \\ 0 & Y-1 \end{pmatrix} \begin{pmatrix} \gamma^\mu \Psi_1^R \\ \gamma^\mu \Psi_2^R \end{pmatrix} \; .
\end{aligned}
$$

After matrix multiplication we obtain:

$$
\begin{aligned}
& \frac{g_2}{2\sqrt{2}} \Big(W_\mu^- \bar{\Psi}_2 \gamma^\mu (1-\gamma^5) \Psi_1 + W_\mu^+ \bar{\Psi}_1 \gamma^\mu (1-\gamma^5) \Psi_2 \Big) \\
+ \quad & \frac{g_1}{4 \sin \Theta_w} Z_\mu \Big(\bar{\Psi}_1 \gamma^\mu (1 - \gamma^5 - 2(Y+1)\sin^2 \Theta_w) \Psi_1 \\
& \qquad - \bar{\Psi}_2 \gamma^\mu (1 - \gamma^5 + 2(Y-1)\sin^2 \Theta_w) \Psi_2 \Big) \\
+ \quad & \frac{g_1}{2} \cos \Theta_w A_\mu \Big((Y+1) \bar{\Psi}_1 \gamma^\mu \Psi_1 + (Y-1) \bar{\Psi}_2 \gamma^\mu \Psi_2 \Big) \ . \qquad (60)
\end{aligned}
$$

H.3.7 Interaction of the Higgs doublet with fermions and generation of fermion masses

The mass terms for fermions in the electroweak theory are generated via their $SU(2) \times U(1)$ invariant Yukawa interaction with Higgs doublet. Namely, the Yukawa Lagrangian

$$
-\frac{m_1\sqrt{2}}{\phi_0} \left(\bar{\Psi}_i^L \epsilon^{ij} \overset{*}{\Phi}{}^j \Psi_1^R - \bar{\Psi}_1^R \Phi^i \epsilon^{ij} \Psi_j^L \right) - \frac{m_2\sqrt{2}}{\phi_0} \left(\bar{\Psi}_i^L \Phi^i \Psi_2^R + \bar{\Psi}_2^R \overset{*}{\Phi}{}^i \Psi_i^L \right)
$$

produces the mass terms for doublet components

$$
-m_1 \bar{\Psi}_1 \Psi_1 - m_2 \bar{\Psi}_2 \Psi_2 \ ,
$$

which are accompanied by vertices of interactions of fermions with Goldstone fields:

$$
\begin{aligned}
- \quad & \frac{m_1 \, g_2}{2M_W} \left(\frac{i}{\sqrt{2}} (W_f^- \bar{\Psi}_2 (1+\gamma^5) \Psi_1 - W_f^+ \bar{\Psi}_1 (1-\gamma^5) \Psi_2) + H \bar{\Psi}_1 \Psi_1 + i Z_f \bar{\Psi}_1 \gamma^5 \Psi_1 \right) \\
- \quad & \frac{m_2 \, g_2}{2M_W} \left(\frac{i}{\sqrt{2}} (W_f^+ \bar{\Psi}_1 (1+\gamma^5) \Psi_2 - W_f^- \bar{\Psi}_2 (1-\gamma^5) \Psi_1) + H \bar{\Psi}_2 \Psi_2 - i Z_f \bar{\Psi}_2 \gamma^5 \Psi_2 \right) \ . \qquad (61)
\end{aligned}
$$

If there are several doublets with the same hypercharge, then a general form of Yukawa Lagrangian contains a product of terms from different doublets. Such terms form two mass matrices: one for upper and another for lower fermions. Each of these matrices can be diagonalized by means of the unitary transformation of doublets, but this cannot be done for both of them at the same time. In this case the basis of doublets is chosen in such a way to

present one of these matrices, for example, for upper fermions, in the diagonal form. Then the physical particles correspond to linear combinations of lower doublet fields realized by some unitary matrix which is called a *mixing matrix*.

Generally the Lagrangian is written down in terms of fermion fields which directly correspond to particles. Interactions of such fields with A, Z, H, and Z_f are the same as defined by (60) and (61), whereas interactions with $W^\pm$ and $W_f^\pm$ contain elements of the mixing matrix.

H.3.8 Quarks and leptons

The Standard Model contains three doublets of leptons with a hypercharge $Y = -1$:

$$
\begin{pmatrix} \nu_e \\ e^- \end{pmatrix} \begin{pmatrix} \nu_\mu \\ \mu^- \end{pmatrix} \begin{pmatrix} \nu_\tau \\ \tau^- \end{pmatrix} \ ,
$$

and three doublets of quarks with a hypercharge $Y = \frac{1}{3}$:

$$
\begin{pmatrix} u \\ d' \end{pmatrix} \begin{pmatrix} c \\ s' \end{pmatrix} \begin{pmatrix} t \\ b' \end{pmatrix} \ .
$$

Neutrinos, the upper components of lepton doublets, are so far massless as a result of experimental measurements. So the mixing matrix for leptons is not needed. Meantime both the upper and down components of quark doublets have nonzero masses. The corresponding mixing matrix is named the Cabibbo-Kobayashi-Maskawa matrix. It expresses the lower components of quark doublets in terms of the real quark fields:

$$
\begin{pmatrix} d' \\ s' \\ b' \end{pmatrix} = \begin{pmatrix} Kobayashi \\ Moskawa \\ matrix \end{pmatrix} \begin{pmatrix} d \\ s \\ b \end{pmatrix} \ .
$$

References

[1] E. Boos et al. in: "New Computing Techniques in Physics Research", ed. by D. Perret-Gallix and W. Wojcik, Editions du CNRS, 1990, p. 573

A. Pukhov et al. CompHEP - a package for evaluation of Feynman diagrams and integration over multi-particle phase space, Preprint INP MSU 98-41/542 (hep-ph/9908288)

[2] A. Semenov, preprint LAPTH-884/01, 2001.

[3] J. Küblbeck, M. Böhm, A. Denner, Comp. Phys. Comm. 60 (1990) 165 R. Mertig, M. Böhm, A. Denner, Comp. Phys. Comm. 60 (1990) 345

[4] T. Tanaka, T. Kaneko, Y. Shimizu, Comp. Phys. Comm. 64 (1991) 149

T. Ishikawa, T. Kaneko, K. Kato, S. Kawabata, Y. Shimizu, T. Tanaka, *GRACE manual, version 1.0*, KEK Report 92-19, 1993

[5] H. Murayama, I. Watanabe, K. Hagiwara, *HELAS: HELicity Amplitude Subroutine for Feynman Diagram Evaluation*, KEK Report 91-11, 1992

[6] T. Stelzer, W. F. Long, Comp. Phys. Comm. 81 (1994) 357

[7] C. G. Papadopolous, hep-ph/0007335, (2000)

[8] M. Moretti, T. Ohl, J. Reuter, hep-ph/0102195 (2001)

[9] R. Harlander, M. Steinhauser, Automatic computation of Feynman diagrams, hep-ph/9812357.

[10] D. Kovalenko, A. Pukhov, Nucl. Instr. and Meth. A389 (1997) 299

[11] A. Semenov, *LanHEP — a package for automatic generation of Feynman rules. User's manual*, preprint INP MSU 96–24/431, Moscow, 1996 (hep-ph/9608488)

A. Semenov. Nucl. Instr. and Meth. A393 (1997) 293

A. Semenov. *LanHEP - a package for automatic generation of Feynman rules from the Lagrangian. Updated version 1.3*, preprint INP MSU 98–2/503, Moscow, 1998

[12] A. Belyaev, A. Gladyshev, A. Semenov, preprint IFT–P–093–97 (hep-ph/9712303).

[13] M. Dubinin, A. Semenov, hep-ph/9812246

[14] V.A. Ilyin, D.N. Kovalenko, A.E. Pukhov, *Recursive algorithm for the generation of relativistic kinematics for collisions and decays with regularizations of sharp peaks*, Int. J. Mod. Phys. C7 (1996) 761 (hep-ph/9612479)

[15] H. L. Lai *et al.* [CTEQ Collaboration], "Global QCD analysis of parton structure of the nucleon: CTEQ5 parton distributions", Eur. Phys. J. C **12** (2000) 375; http://cteq.org

[16] A.D. Martin, R.G. Roberts and W.J. Stirling, Phys. Lett. B354 (1995) 155

[17] I.F. Ginzburg, G.L. Kotkin, V.G. Serbo, V.I. Telnov, Nucl. Instr. and Meth. 205 (183) 47

[18] V.M. Budnev et al., Phys. Rep. 15C (1975) 181

S. Frixione et al., Phys. Lett. B319 (1993) 339

[19] E. Kuraev, V. Fadin, Sov. J. Nucl. Phys. 41 (1985) 466

[20] S. Jadach, B. Ward Comp. Phys. Commun. 56 (1990) 351

S. Jadach, M. Skrzypek, Z.Phys. C 49,(1991) 577

[21] P. Chen, Phys. Rev. D46 (1992) 1186

[22] G.P. Lepage, Journal of Comput. Phys. 27 (1978) 192

[23] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992

[24] A. Hearn, J. Fitch, *REDUCE User's Manual*, v.3.6, RAND Publication CP78 (Rev. 7/95)

[25] S. Wolfram, *MATHEMATICA*, Addison-Wesley, 1991

[26] J.A.M. Vermaseren, *Symbolic manipulation with FORM, version 2: Tutorial and Reference Manual*, CAN, 1992

[27] Review of Particle Physics, Eur. Phys. J. C 15, 1-878(2000)

[28] U. Baur, J. Vermaseren, D. Zeppenfeld, Nucl. Phys. B375 (1992) 2

[29] Y. Kurihara, D. Perret-Gallix, Y. Shimizu, Phys. Lett. B349 (1995) 367

[30] E. Boos, M. Dubinin, L. Dudko, Int. J. Mod. Phys. A11 (1996) 5015

[31] V. Ilyin, D. Kovalenko, A. Pukhov, Int. J. Mod. Phys. C7 (1996) 761

D. Kovalenko, A. Pukhov, Nucl. Instr. and Meth. A389 (1997) 299

[32] T. Sjostrand, Comp. Phys. Comm. 82 (1994) 74

[33] A. S. Belyaev *et al.*, arXiv:hep-ph/0101232.

[34] J.B. Bjorken, S.D. Drell, *Relativistic quantum fields*, McGraw-Hill, New York, 1965

C. Itzykson, J. Zuber. *Quantum Field Theory*, McGraw-Hill, New York, 1980

[35] T.-P. Cheng and L.-F. Li, *Gauge theory of elementary particle physics*, Oxford Univ. Press, 1984

[36] L. Baulieu, Phys. Rep. 129 (1985) 1

[37] J.A.M. Vermaseren, Comp. Phys. Comm. 83 (1994) 45

[38] F.A. Berends, R. Pittau, R. Kleiss, Comp. Phys. Comm. 85 (1995) 437

[39] E. Byckling, K. Kajantie, Particle Kinematics, John Wiley & Sons, 1973

[40] A. P. Kryukov and A. Y. Rodionov, Nonabelian Gauge Theories," Comput. Phys. Commun. **48** (1988) 327.

Cvitanovic, Phys.Rev. D14 (1976) 1536

Pictures and figures

Figure 1: Example of the CalcHEP menu

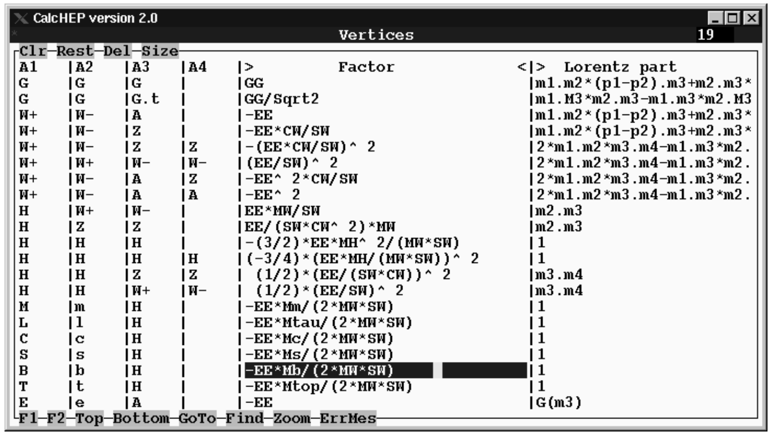


Figure 2: Example of the CalcHEP tables

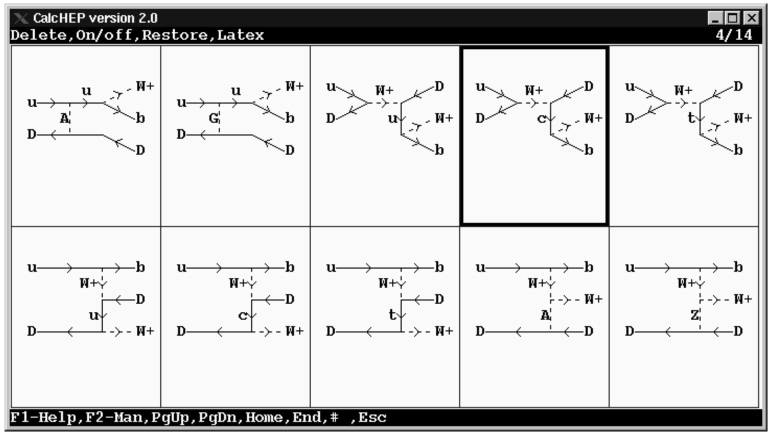


Figure 3: Example of the CalcHEP diagram images

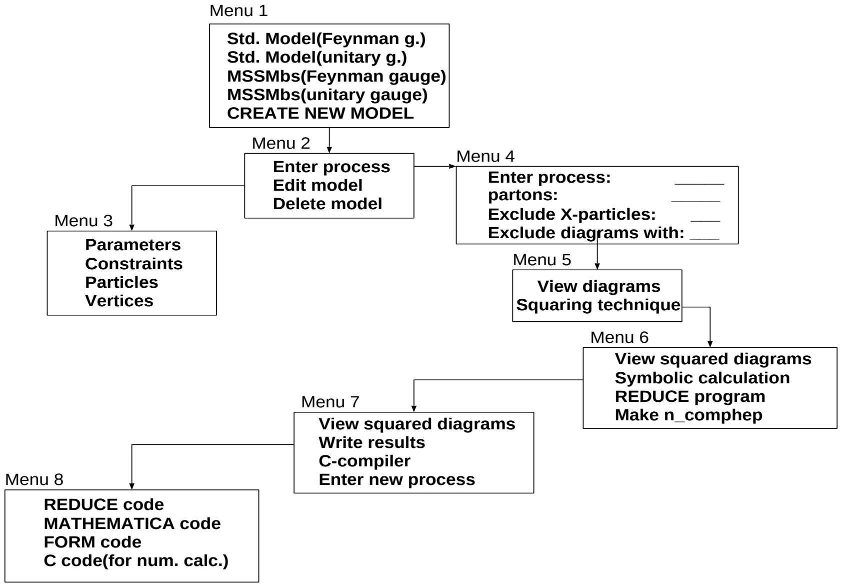


Figure 4: Menu scheme for the symbolic session

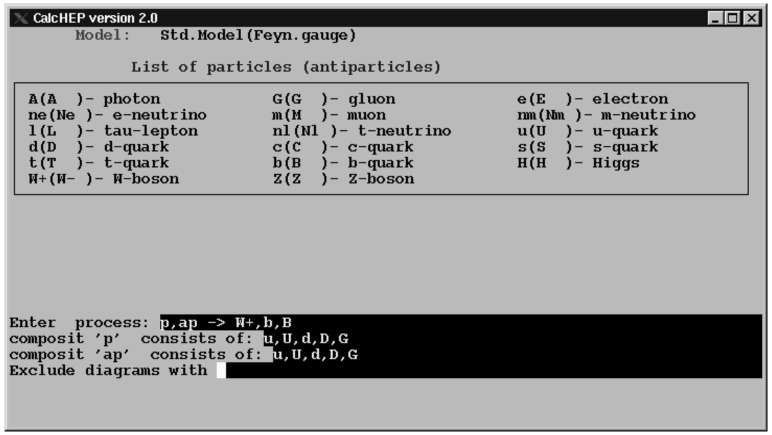


Figure 5: Example of the process input

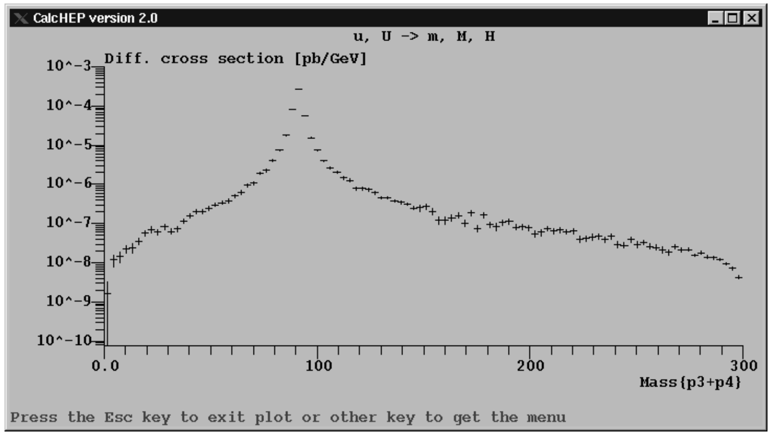


Figure 6: Example of the plot image

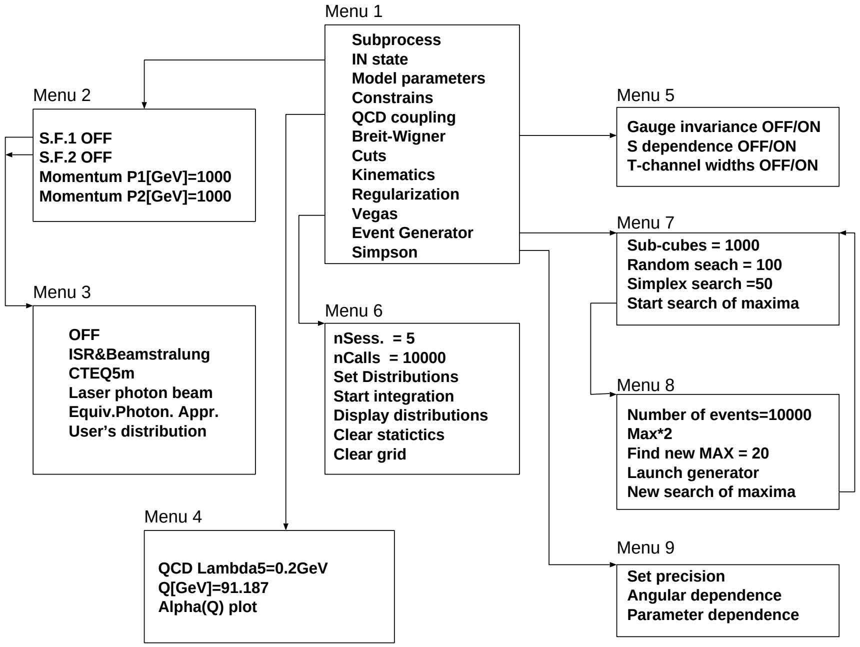


Figure 7: Menu scheme for the numerical session

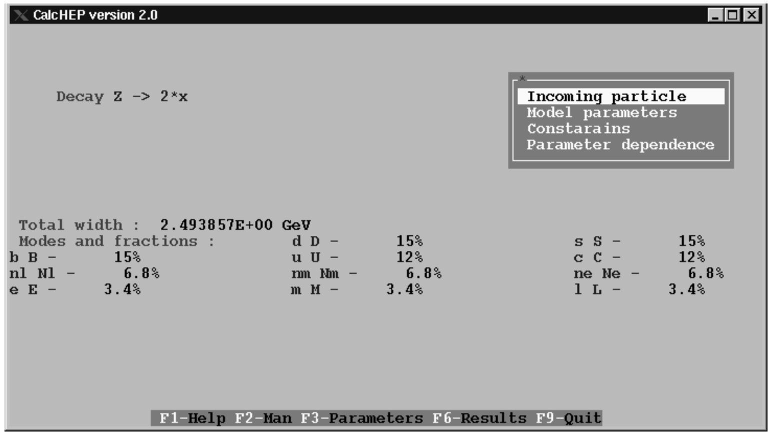


Figure 8: Representation of results for 1->2 processes

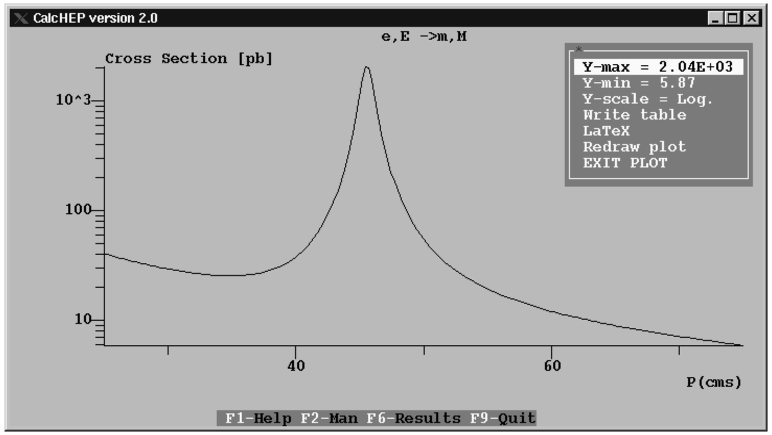


Figure 9: Plot for a 2->2 process


```
  e           M    !  M           e
==>==\     /==<====!==<==\     /==>==
  P1 |     |  P4   !  P4 |     |  P1
     |     |       !     |     |
  E  |  A  |  m    !  m  |  A  |  E
==<==@-1---@==>====!==>==@---2-@==<==
  P2    P5    P3   !  P3   -P6    P2
```

Figure 10: Example of pseudo-graphic diagram image

```
  e     ne   !  ne    e
==>==@==>====!==>===@==>==
     |       !      |
   W+|       !    W+|
     |       !      |
-----@--<----!--<---@-----
A       W-      W-       A
          a)
```

```
  e     ne   !  ne    e
==>==@==>====!==>===@==>=
     |       !      |
   W+|       !    W+|
     |       !      |
-----@--<----!--<---@-----
A      W-.f    W-.f      A
              b)
```

```
   e     ne   !  ne    e
 ==>==@==>====!==>===@==>==
      |       !      |
    W+|       !    W+|
      |       !      |
 --<--@--<----!--<---@--<--
A.C     W-.C    W-.C     A.C
             c)
```

```
   e     ne   !  ne    e
 ==>==@==>====!==>===@==>==
      |       !      |
    W+|       !    W+|
      |       !      |
 -->--@--<----!--<---@-->--
A.c     W-.c    W-.c     A.c
             d)
```

Figure 11: Ghost diagrams

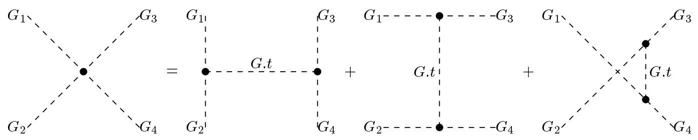


Figure 12: Splitting of four-gluon vertex

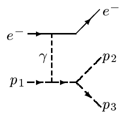


Figure 13: Example of process with the $1/t^2$ pole cancellation.