# CompHEP- a package for evaluation of Feynman diagrams and integration over multi-particle phase space

A.Pukhov* E.Boos M.Dubinin V.Edneral
V.Ilyin D.Kovalenko A.Kryukov
V.Savrin
*Skobeltsyn Institute of Nuclear Physics, Moscow State University*

S.Shichanin
*Institute for High Energy Physics, Protvino, Russia*

A.Semenov
*Joint Institute for Nuclear Research, Dubna, Russia*

**User's manual for version 33**

Preprint INP-MSU 98-41/542

*contact persons: Alexander Pukhov: pukhov@theory.npi.msu.su and Viacheslav Ilyin: ilyin@theory.npi.msu.su

# Contents

# 1 Preface

## 1.1 Introduction

CompHEP 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 the CompHEP is to make available passing on from the Lagrangian to the final distributions effectively with a high level of automation. Other packages created to solve a similar problem are *FeynArts/FeynCalc*[1], *GRACE*[2], *HELAS*[3], *MADGRAPH*[4]. See also the review [5].

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

The present version has 4 built-in physical models. Two of them are the versions of the Standard Model (SU(3)xSU(2)xU(1)) in the unitary and t'Hooft - Feynman gauges. 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 CompHEP package consists of two parts: symbolic and numerical. The symbolic part is written in the *C* programming language. It produces *Fortran* and *C* codes for a squared matrix element, and they are used in the numerical calculation later on. There are two versions of the numerical part: one is written in *Fortran* and another one is done in *C*. The facilities of both versions are almost equal. The *C* version has more comfortable interface but it does not possess an option to generate events and does not perform calculations with a quadruple precision.

The symbolic part of CompHEP 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;
- save symbolic results corresponding to the squared diagrams calculated in the *Reduce* and *Mathematica* codes for further symbolic manipulations;
- generate the optimized *Fortran* and *C* codes for the squared matrix elements for further numerical calculations;
- launch the built-in interpreter for numerical calculations;

The numerical part of CompHEP offers to:

- convolute the squared matrix element with structure functions and beam spectra. *CTEQ* and *MRS* 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;
- introduce 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*;
- store values of the calculated matrix element in a file for subsequent event generation;

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

See the review [6] about physical results produced by means of CompHEP.

## 1.2 History and contributions

The CompHEP project was founded in 1989 by the group of physicists and programmers of D.V. Skobeltsyn Institute of Nuclear Physics of Moscow State University. The project was initiated by Edward Boos, Viacheslav Ilyin and Victor Savrin. The primary formulation of physical problems for the project was done by E. Boos, Mikhail Dubinin, and Dmitri Slavnov. The first software working group was organized and managed by V. Ilyin.

The main author of the CompHEP software is Alexander Pukhov. He has developed almost all algorithms and data-representation structures of the package. Namely, the structure of physical model database, the algorithm for generation of Feynman diagrams, the algorithm for evaluation of squared matrix elements, the structure of output codes for different programming languages, the algorithm for optimization of numerical codes, the algorithm for phase-space integration with smoothing of propagator peaks. He also has created the specialized symbolic manipulation package for CompHEP.

The first version of the package appeared in 1989 [8, 9]. It was written in the *Turbo Pascal* programming language for the *MS-DOS* operation system. The program produced a code for calculation of squared diagrams, written in the *Reduce* symbolic manipulation language. Routines for evaluation of the color factors were written by Alexander Kryukov. The *Reduce* code-generation routines were written by Alexander Taranov and A. Pukhov.

The authors of CompHEP were being lead by an idea to create a user friendly software. So they paid a special attention to the interface and data-representation facilities. The general part of graphical interface was designed by A. Pukhov. The routines for graphic representation of diagrams were written by Victor Edneral. The context-sensitive help facility was designed by

Sergey Shichanin. The program for the CompHEP database table manipulation was done by A. Kryukov. Later on the plot drawing and LaTeX output were designed by V. Edneral, A. Pukhov and S. Shichanin.

The CompHEP symbolic answers were compared by Mikhail Dubinin with a large number of known symbolic expressions for differential and integral cross sections [20]. Starting from this point the physicists of CompHEP group E. Boos, M. Dubinin, and V. Ilyin applied CompHEP for studying new physics signals and relevant backgrounds.

In 1991 CompHEP got the built-in symbolic manipulation package created by A. Pukhov and the *Fortran* code output written by A. Pukhov and S. Shichanin [10].

In 1992 Andrei Davydychev proposed to use the t'Hooft-Feynman gauge for evaluation of squared diagrams. This idea was realized by A. Pukhov. It has opened the real possibility to calculate any 2 `->` 4 process with the help of the CompHEP package. In the same year V. Ilyin and S. Shichanin designed a *Fortran* program of phase-space integration for 2 `->` 3 processes and A. Pukhov developed the numerical interpreter for 2 `->` 2 processes. These achievements were announced in [10].

The numerical results by CompHEP for a large set of processes were compared with the *GRACE* package [21]. E. Boos, M .Dubinin, V .Ilyin and S .Shichanin performed some other cross-checks of the CompHEP package. Later on A. Belyaev, E. Boos and L. Dudko compared CompHEP with the FNAL program *VECBOS* [22] and extensive cross-checks of numerical results for $e^+, e^- \rightarrow 4 \quad fermions$ set of processes have been performed by M. Dubinin [23].

In 1993 the symbolic part of CompHEP was rewritten in the C programming language by V. Edneral [14]. The realization of the *Turbo Pascal* graphic routines by means of X11 tools was done by Andrey Semenov [15]. It has opened the way to create the version for UNIX workstations [14, 11, 7].

The main problem remaining in this version was the phase-space integration. CompHEP created the *Fortran* code for squared matrix elements with a high level of automation. Generally the matrix elements have a lot of singularities caused by the propagators of virtual particles. In order to succeed in the Monte Carlo phase-space integration of singular matrix elements the user was forced every time to modify the program of phase-space parameterization.

For automation of this operation step A. Pukhov proposed a general approach to the generation of multi-particle kinematics and to the regularization of matrix elements. In 1995-96 this scheme was realized by A. Pukhov, Dmitri Kovalenko and V. Ilyin [12, 13, 19]. At the same time V. Ilyin wrote new *Fortran* interface programs like *menus* and together with A. Kryukov embedded the parton structure functions in CompHEP. Later on the structure-function package was improved by A. Pukhov to include the regularization of integration over Feynman parameters. As a result we get a version which provides the user with a possibility of automatic evaluation starting from the input of Lagrangian and finishing with distributions in physical parameters. The corresponding service for a histogram filling also was done by A. Pukhov. The list of needed distributions was compiled by E. Boos.

During this work it was realized that the *Fortran* programming language is not convenient for the future development of the numerical part of CompHEP. In 1997 the *C* code output for the numerical calculation was designed and the *Fortran* program for the numerical evaluation was rewritten in *C* as well. This work was done by A. Pukhov. In the same year A. Kryukov imported CompHEP onto the MS-Windows95/NT platform.

The development of CompHEP was being under a continuous pressure of physicists' requests. The works of E. Boos, M. Dubinin, V. Ilyin, V. Savrin and S. Shichanin, who first used CompHEP for physical calculations, at the same time were defining a direction of the package development. They also contribute to and are responsible for debugging the package.

It must be especially noted the role of E. Boos and V. Ilyin in the popularization of CompHEP for the scientific community.

The manual was written by A. Pukhov, V. Savrin, and S. Shichanin.

## 1.3 Acknowledgments

The CompHEP project was supported by

Russian State Program on High Energy Physics;
RFBR (93-02-14428, 96-02-19773a, 96-02-18635, 98-02-17699);
ISF (M9B000, M9B300);
INTAS (1010-CT93-0024, 93-1180, 93-1180ext,96-0842);
Japanese Society for the Promotion of Science;
Japanese companies KASUMI Co, Ltd. and SECOM Co, Ltd.;
Royal Society of London (UK);

Organizing Committee of the AIHENP series of International Workshops.

During a long time the CompHEP group had not got a possibility to develop the project on UNIX workstations in Russia. The adaptation of the package for different UNIX platforms was done during the visits of the group members to various universities and scientific centers of the world. We very much appreciate this support and are grateful for cooperation to our foreign colleagues Y. Shimizu, (KEK, Japan), H.S. Song (Seoul National University, Korea), O. Eboli (University of Sao Paulo, Brazil), H.-J. Schreiber (DESY, Germany).

We acknowledge beneficial discussions with F. Cuypers, I. Ginzburg, F. Gutbrod, B. Mele, M. Sachwitz, W. von Schlippe, P. Osland, and members of Minami-Tateya group (KEK), and their benevolent attitude to our project during many years. In this context we are especially grateful to D. Perret-Gallix.

We also express our gratitude to our colleagues A. Taranov, P. Baikov, H. Eck, L. Gladilin, P. Silaev, S. Ostapchenko who contributed to the development of CompHEP software as well as to A. Davydychev, A. Rodionov and D. Slavnov for some helpful ideas.

We thank S. Ambrosanio and A. Belyaev for their numerous reports on CompHEP bugs.

# 2 Installation procedure

## 2.1 CompHEP Web page

CompHEP codes and manual are disposed on the following Web site

`http://theory.npi.msu.su/~comphep`

Also there is a mirror in DESY

`http://www.ifh.de/~pukhov`

## 2.2 License

**Non-profit Use License Agreement**

This Agreement is to be held between the Authors of the CompHEP 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 or executable code of CompHEP 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. Publications which result from using the program will contain references to the articles describing CompHEP. See the necessary references on the CompHEP Web page.

4. 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 Authors.

5. The Authors of CompHEP 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 codes

If you agree with the license above, you may get CompHEP code for the version 33 on the CompHEP Web page. The name of received file should be

```
comphep_33.#.tar.gz
```

where # denotes a number of release. Unpack this file by

```
gzip -d comphep_33.#.tar.gz
```

```
tar -xf comphep_33.#.tar
```

As a result a directory comphep_33.# should be created.

This directory contains source codes of the CompHEP package for UNIX platforms. After compilation of these codes the CompHEP binary executable files appear in the same directory. We shall refer to it below as a CompHEP root directory.

## 2.4 Compilation procedure

In order to compile the CompHEP source code you need a *C* compiler with the X11 graphics library. If you would like to use *Fortran* for numerical evaluation you need also a *Fortran* compiler. The compilation of the CompHEP *C* source code is launched by the

```
./create_c
```

command started from the CompHEP root directory.

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

```
"C-part of code has been successfully compiled and linked."
"CompHEP could be started."
```

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

If the *Fortran* compiler is available and you would like to use *Fortran* for numerical calculation launch the

```
./create_f
```

command. If the *Fortran* compiler is detected and the *Fortran* sources are also compiled successfully, the message appears:

```
"Fortran code has been successfully compiled."
```

See the next section in the case of problem with compilation.

The files in the CompHEP root directory will be used only for reading and execution during the user session. It provides us with a possibility to have one CompHEP root directory for several users. To set the appropriate files modes start the

```
./set_mod
```

command.

Each user has to create his own directory to work with CompHEP. See an instruction for the user installation in the next section. However for an express check of compiled version you could go to the *./test* directory and launch *./comphep* there.

### Possible problems

Due to its first instruction the *create_c* command tests the existence of the *CC* file in the CompHEP root directory. If this file exists, the C compiler name and its options are read from this file. Otherwise, as we usually have at the first start of *create_c*, this file is created and contains default parameters.

After that the *create_c* program tests the necessary compiler options. 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 *create_c* command finishes with the corresponding error message and asks you to rewrite the command 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 your *CC* file and start *create_c* again. The options listed below must be supported by *C* compiler:

- the ANSI mode of function prototypes;
- the signed char type;
- an access to the X11 include and library files.

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

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

The work of *create_f* is similar to that of *create_c* described above. The *Fortran* compiler must possess long character variables (e.g. CHARACTER*5000). The name of *Fortran* compiler and options of compilation are stored in the *F77* file.

If all tests are passed on successfully the message appears:

```
C (Fortran) compiler OK. Starting CompHEP  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*. Anyway, in the case of some problem on this step you can send a request for help to the CompHEP authors.

A correction of the *CC* and *F77* files could be used to tune compiler options according to your UNIX platform. In this case you should create *CC* or *F77* by starting *makeCC* or *makeF77*, perform tunings in the *CC* or *F77* files and launch *create_c* or *create_f* after that. For example, you could switch on an optimization flag.

### *C* compiler tuning

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 CompHEP models. By default *STRSIZ=2048*. If you would like to embed in CompHEP 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 CompHEP. By default CompHEP uses the 'long' type.

*-DNUM_LONG_LONG* forces the compiler to use *'long long'*. This type is not the 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 CompHEP uses

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

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

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

## 2.5 User installation and start of the CompHEP session

The CompHEP root directory is not intended to start a session. For this purpose the user has to prepare a special working directory. A few working directories may be created for various tasks and by different users.

In order to provide the programs with an access to the CompHEP files and commands the environment variable COMPHEP should contain the corresponding path. We recommend to define the COMPHEP variable in the user startup file. The name of the startup file and the syntax of the assignment instruction depends on the command interpreter. For example, in the case of *tcsh* the name of startup file is *~/.tcshrc* or *~/.cshrc* and the assignment is realized by the instruction

```
setenv COMPHEP <Path to CompHEP directory>
```

Let you have defined the COMPHEP variable and create a directory, say, *WORK* for the CompHEP session. To prepare this directory for processing call the program *install* from within this directory:

```
$COMPHEP/install
```

As a result the following sub-directories and files should appear in your *WORK* directory:

```
models/    tmp/        results/
comphep    comphep.ini
```

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

To start a CompHEP session you should issue a command

```
./comphep
```

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

### 2.6 Installation under MS Windows9x/NT

The CompHEP Windows9x/NT version is distributed in a compiled form due to the absence of standard C/Fortran compiler under the Windows9x/NT. Thus the program does not provide the user with compiling and linking options within the working session. The user should use the "Numerical interpreter" option to get numerical results.

The installation file `bNNi.zip` (where `NN` denotes a number of the release) is available to copy from the CompHEP Web-page. The installation procedure is the following:

1) create an installation directory and copy the distributive `bNNi.zip` to this directory;

2) unpack the distributive file `bNNi.zip` by the command

   `unzip bNNi.zip`

   As a result a set of files and subdirectories should be created. This set corresponds to the user working directory in the UNIX release (see section 2.5). Executable and other files corresponding to those in the CompHEP root directory of the UNIX version (see section 2.3) are stored in the subdirectory `bin`.

To start a CompHEP session the user should launch the command

`comphep.pif`

in the installation directory.

Contact person for the Windows9x/NT version of CompHEP is A.P. Kryukov (e-mail: kryukov@theory.npi.msu.su).

# 3 User guide

## 3.1 Elements of the user interface

CompHEP software is written in *C* and *Fortran*. The symbolic part is done in *C*. The numerical one was written in *Fortran* but later on was converted to *C*. However both versions of numerical part are now available. The *C* programs use a graphical window interface based on the X11 or MS-Windows facilities, whereas the *Fortran* ones use only a standard input/output. So in the latter case the service is more ugly.

### 3.1.1 Graphical interface

There are the following elements of the user interface in the CompHEP 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*[1], *Backspace*, *PgUp/PgDn*[2] keys and the mouse click. CompHEP is sensitive to the left mouse button release.

You can toggle on/off colors and a sound for the CompHEP session as well as choose the most appropriate font for the CompHEP window. Just edit *comphep.ini* 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 CompHEP 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.

[1] Use the `'Ctrl ['` sequence if the *Esc* key is absent on your keyboard
[2] On some keyboards there are *Prev/Next* instead of *PgUp/PgDn*

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:

| | | |
|---|---|---|
| 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 CompHEP output files. |
| F9- | Quit | : quits the CompHEP 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.** CompHEP 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, CompHEP 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.** CompHEP 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.3). 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 ordering number of the corresponding line 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.

*Find again (^ A)* repeats previous command *Find*.

*Zoom (^ Z)* key switches on the zoom mode to view/edit contents of the highlighted cell. In this case CompHEP 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 CompHEP.

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 (ˆ C)* clears the contents of the current field right to the cursor position.

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

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

*New (ˆ 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 (ˆ 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.4.

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.10 and Fig.7.

Being launched the *Plot Viewer* displays a picture and waits for the keyboard signal as it is shown in Fig.10. 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.7. 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 writing the file. The numerical table created in this way can be displayed on the screen later on by the command

```
$COMPHEP/tab_view < table_file
```

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.

### 3.1.2 Interface for a text-screen mode

Only the standard input/output *Fortran* facilities are used for this part of CompHEP. So the service is primitive. The user input here is a string terminated by the *Enter* key.

**1. Menu.** When you work with the menu the numerated list of menu functions is displayed. Possible inputs are:

`H<number>` - to get help information about the `<number>`$^{th}$ menu function;

`<number>` - to execute the corresponding menu function;

`X` - to quit the current menu.

The lower case characters $h$ and $x$ may be used also as key characters of the above commands. You can see in Fig.2 how this menu looks like on the display.

**2. Tables.** Tables are used to enter and change information about physical cuts and a phase space regularization. When you use this program a numerated list of elements is displayed. Elements which have a number older than

9 are numerated by letters (A,B,C,D ...). Under this list you see a prompt for commands:

`N - new, D<set> - delete, C<n> - change, X - quit.`

`N` is used to insert a new table line. The subsequent input depends on the table.

`D<set>` is used to delete records enumerated in `<set>`. For example, `D13A` will delete the first, third and tenth records.

`C<n>` allows you to change contents of the `<n>`$^{th}$ item.

`X` is used to quit.

The lower case characters *n, d, c* and *x* may be used also as key characters of the above commands.

**3. Viewer.** When you browse the help information or some file contents the following commands may be used:

to look at the next page type the *Spacebar* key;

to look at the preceding page type 'b';

to exit from the viewer type 'x'.

## 3.2 Menu system for symbolic calculation

The scheme of menus for the symbolic calculation session is presented in Fig.5.

### 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.

There are four models built in CompHEP: quantum electrodynamics, the model of electroweak four-fermion interaction, and two versions of the Standard Model. The QED model is included as an example of realization of the simplest particle interaction scheme in CompHEP. The four-fermion interaction model gives an example of realization of the four-fermion interaction in the CompHEP notations. The Standard Model is presented in two gauges: the unitary one and the t'Hooft-Feynman one. We recommend to choose the latter 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 CompHEP 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. CompHEP 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.

**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 CompHEP 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. Generally they are text files which are disposed in user's *models* directory and may be corrected by an ordinary text editor. But we strongly recommend to use CompHEP facilities to edit these files because in this case CompHEP can control possible mistakes of the user input.

CompHEP 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 3.1.1 for the explanation of facilities of the table editor.

CompHEP 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 ?<br>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.6. 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.

One can also construct inclusive processes. For example, 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 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 one enters a collision process, the information on total energy of colliding particles in the center-of-mass system is demanded:

`Enter Sqrt(S) in GeV:` *300*

CompHEP generates only those channels where the total mass of incoming particles and the total mass of outgoing particles are smaller than *Sqrt(S)*.

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.

Several restrictions separated by commas are allowed. If one has a restriction for a particle, the restriction for the corresponding anti-particle is not needed. 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 CompHEP 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 first function of this menu (*Squaring*) is the instruction for CompHEP to create squared diagrams. CompHEP uses the squared diagram technique for evaluation of squared matrix elements. See Section B for details.

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 3.1.1 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 self-conjugated, 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.
- CompHEP produces only one representative of a set of diagrams which can be transformed one to another by replacing the identical outgoing particles. For example, CompHEP 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 to the outgoing particles on the step of diagrams generation.
- CompHEP does not generate diagrams with Faddeev-Popov and Goldstone ghosts[3]. Such diagrams are restored on the step of squared diagram evaluation. Any ghost has a real particle as a prototype (see

[3] 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*.

Section 6). 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 ghost 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 ghost diagrams with a four-Goldstone vertex CompHEP generates false diagrams with $Z^4$ interaction.

- Vertices with a complicated color structure, for example, the four-gluon one, are implemented by means of unphysical tensor field. This field is treated by CompHEP 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 6, 4.1.5, 4.2 for further explanations.

**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 CompHEP squared diagrams generation:

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

- CompHEP 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 CompHEP 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 just 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 created specially in the framework of the CompHEP project.

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 [33] separately for each squared diagram. These codes are not used for the further CompHEP processing, but they can be useful for cross-checking the CompHEP 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 CompHEP is to create the corresponding *C* or *Fortran* 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. CompHEP 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_comphep_c* or *Make n_comphep_f* menu function. Then the current interactive session ends and a new batch process starts. See Section 3.2.5 for details of the non-interactive calculations in CompHEP.

As the outcome, the executable file *n_comphep_c* or *n_comphep_f* is created for the further numerical precessing. 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 CompHEP 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_comphep_* functions may require large computer resources for some processes with large number of diagrams. In this case you could correct your *comphep* 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 three menu functions realize numerical calculations in CompHEP, particularly Monte Carlo sessions. The first one calls the built-in numerical interpreter of the obtained symbolic expression. This program has almost the same possibilities as the numerical code produced by the *C*-compiler, but works slower than the compiled option.

The menu function *C-compiler/Fortran compiler* calls the corresponding compiler to create and launch an executable file *n_comphep_c/n_comphep_f* 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* and *Fortran* outputs are used to get numerical results. They may be compiled and linked with the library of CompHEP 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/Fortran* output use functions of *Menu 6* or launch the *$COMPHEP/make__n_comphep* program as it has been explained in Section 3.5.

### 3.2.5 Non-interactive session

CompHEP was created as a program for calculation in the interactive regime. But in practice long-time calculations are typical. Due to this circumstance we have implemented the batch (or, say, 'blind') calculation mode into CompHEP.

If one launches *s_comphep* with an option "-blind", e.g.

`$COMPHEP/s_comphep -blind STRING`

then CompHEP does not open the X11 window and reads the control characters from *STRING*. For imitating functional keys the following control characters are defined:

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

So, the user has a possibility to concatenate into *STRING* a list of commands which simulate the keyboard and execute *s_comphep* in the non-interactive mode.

The work of *s_comphep* in the blind mode is simplified a little. It automatically passes by all warnings and messages answering 'Y' (Yes) any question itself. The CompHEP 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 *comphep* command file. When you activate the *Make n_comphep_c* function of *Menu 6* then *s_comphep* ends with an exit code 24 and

`$COMPHEP/s_comphep -blind "]}}]]]}9"`

is launched. The above command performs symbolical calculations and writes down the C-code for the squared matrix element into the directory *results*. Note that the last symbol '9' simulates the work of the *F9* functional key which in its turn completes the session (See Section 3.1.1).

Another utilization of the *blind* mode is a check of CompHEP for some huge set of processes (Section A).

## 3.3 Numerical calculation by CompHEP

In the framework of CompHEP package there are four ways to realize numerical calculations, namely using:

1) built-in interpreter

2) compiled *C* output

3) compiled *double precision* (REAL*8) *Fortran* output

4) compiled *quadruple precision* (REAL*16) *Fortran* output.

For the first two items calculations are performed with a *double* precision. The interpreter works slower compared with the stand-alone compiled executables, but it does not need such steps as the storage and compilation of symbolic output. So it is reasonable to use the interpreter either in the case of simple process or if you have no compiler at all.

The general functionality scheme is the same for all four cases. The main differences are:

- for the first two items the user interface is more comfortable because it is based on graphic terminal facilities;

- results of calculation produced by the interpreter are displayed on the screen only, whereas in other cases they are stored onto a disk either;

- *batch* calculations (set of evaluations launched by one job task) are not available for the interpreter;

- for the current version only *Fortran* programs provide the user with a possibility to generate an event flow.

The interpreter is launched from within the symbolic session. Compiled programs can be created and started under the symbolic session (see *Menu 7* in Fig.5) or created by means of Section 3.5 commands and launched as an independent process.

### 3.3.1 Sketch of the menu system

A general scheme of CompHEP numerical calculations is presented in Fig.8. The CompHEP screen with the menu number 1 is presented in Figs. 1 and 2. Three functions of this menu launch integration programs. One of them is Monte Carlo program *Vegas* [31, 32]. The other one is the ordinary *Simpson* integration routine which can operate only for the simplest 2 -> 2 type of reactions. The *Batch* menu function is used to organize a set of *Vegas* evaluations in the non-interactive mode (see Section 3.3.10). Other menu items are needed to set an environment for integration. Below we briefly describe all of them. More detailed explanation is given in the following 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 current subprocess is displayed on the screen (Fig.1 and Fig.2). Subprocess summation is not implemented in the current version.

The *In_state* function gives the user an option to choose structure functions for incoming particles and assign some values for their momenta. The *GeV* units are assumed for the momenta. This menu function is available for collision processes only.

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 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 may 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 scale* menu function.

CompHEP substitutes a Breit-Wigner exact propagator for the s-channel virtual particles. This trick is needed to avoid the integral divergence near the mass pole and is motivated by high order corrections. The way 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 3.3.6.

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.

### 3.3.2 Numeration of Monte Carlo sessions

CompHEP numerical calculations are organized as a sequence of Monte Carlo integration sessions. Each session has its ordinal number. The current session number is displayed.

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. To distinguish files produced by *n_comphep_f* from those produced by *n_comphep_c* we use prefixes *'c_'* and *'f_'* as required. The results of integration and all settings are stored in the protocol file *f_prt*`_N` (*c_prt*`_N`). Events generated by *n_comphep_f* are stacked in the file *events*`_N`. A current version of the *n_comphep_c* program does not save events.

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

### 3.3.3 QCD scale

CompHEP substitutes the QCD strong coupling constant as a function of the $\Lambda^{(6)}$ parameter and the $Q^2$ scale value. It is calculated in the next-to-next-to-leading order (NLLO) and depends on a number of quark flavors (*nf*) with masses less than Q (see Eq.9.5a of [36]). Relevant constants $\Lambda^{(nf)}$ are recalculated from $\Lambda^{(6)}$ according to the matching equation (9.7) of [36].

If the user is going to change a numerical value of $\Lambda^{(6)}$ or the $Q^2$ scale definition he should call the necessary function of *Menu 4*. For the $Q^2$ choice he will be prompted what kind of scale he would like to implement, namely, the *constant* scale or the *running* one. In the first case a subsequent input of number in $GeV$ units is expected. In the second case CompHEP provides the user with a possibility to define the $Q^2$ scale as a squared sum of particle momenta. Digits on input will be transformed into ordinal momentum numbers. For example: the input '13' will be treated as $Q^2 = |(p1 - p3)^2|$.

If $Q^2$ is defined as a constant, the corresponding value of $\alpha = GG^2/(4\pi)$ is displayed on the screen. The $\alpha(s)$ plot for a user-defined region of $s$ is available also by means of the corresponding function of *Menu 4*. The same $Q^2$ scale is used as an argument of parton distribution functions.

### 3.3.4 Breit-Wigner propagator

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

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

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 [42]. They transform the propagator to the Breit-Wigner form

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

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

First problem which appears in a way of implementation of this expression is a choice of $\Gamma(p^2)$ dependence. The $\Gamma$ 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 *'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 *'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 [37, 38]

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

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 [39]. If the particle width is very small such an approximation is reasonable, but in some cases it can also lead to an error.

### 3.3.5 Phase space functions

There is a special set of phase space functions which may be used to construct cuts and distributions in the framework of CompHEP. 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 $\Delta y$ 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.

U - user's implemented function. The character string following $U$ is passed on to the user *Fortran/C* function *usrfun(str)* which, as it is assumed, calculates a corresponding value. See Section 3.6 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 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.

Input formats for limits are different for the *Fortran* and *C* realizations. In the case of *n_comphep_c* and *interpreter* sessions the user may type in the corresponding fields any algebraic formula which contains numbers and identifiers enumerated in the *Model parameters* menu. Parentheses "()" and operations "$+, -, /, *, **, sqrt()$" also are permitted.

In the case of *n_comphep_f* session only numbers are expected in the corresponding columns. To escape an input in $GeV^2$ units in the case of squared momenta function

$$V_{min} < S(momenta\ set) < V_{max},$$

the *n_comphep_f* program asks the user to put $V_{min}/\sqrt{|V_{min}|}$ and $V_{max}/\sqrt{|V_{max}|}$ in the relevant cells.

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.

### 3.3.7 Kinematics

The *Kinematics* function allows to display and change a general scheme of the phase space parameterization. A kinematics scheme in CompHEP 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 CompHEP consequently asks the user to define a first outgoing cluster of virtual decay. The user input must 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 [47, 40]. A choice of kinematics influences on the phase space mapping and by this way it influences on convergence of Monte Carlo integration. See Section D.2.

### 3.3.8 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

$$1/(p^2 - m^2) \tag{2}$$

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

where $m$, $\Gamma$, 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. In the case of *Fortran* program the corresponding numerical values must be written down in these fields. In the case of *C* program the user can write down some algebraic expressions which contain the identifiers enumerated in the *Model parameters* menu. For t-channel propagators (both incoming and outgoing momenta contribute to $P$) only a zero value in the *Width* field is permitted because CompHEP ignores particle width term for such propagators.

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 CompHEP 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 3.3.6) 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 D.2.

### 3.3.9 VEGAS menu

See Section D.1 or [31, 32] for explanation of the *Vegas* algorithm. The work of *Vegas* program is driven by *Menu 6* of Fig.8.

The first position of this menu starts the process of Monte Carlo integration. It consists of *Itmx* runs with subsequent adaptation of the Monte Carlo weight function to the integrand. For each run *Vegas* calls the integrand about *Ncall* times. The user may set the *Itmx* and *Ncall* parameters applying the next two items of the menu. The results of calculations are displayed on the screen and written to the protocol file.

If the required precision of calculation is not reached the user may call the *Start integration* menu function once more to increase the Monte Carlo statistic. But if the *Clear statistic* menu function is called, then the next Monte Carlo session will be started from the beginning. The Monte Carlo weight function is saved between runs anyway.

In the case of *Fortran* *n_comphep* program the user can write weighted events generated by *Vegas* into a file. The maximum and average values of event weights are displayed on the screen and written into the protocol file. The user must take into account that any new call of the *Start integration* menu function will rewrite this file.

In the case of *C* version of the *n_comphep* program 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. There could be either a number or an algebraic expression containing the identifiers listed in the *Model parameters* menu.

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. A call for *Clear statistic* menu function clears the stored distributions.

### 3.3.10 Batch calculations

CompHEP was created as a program for calculation in an interactive mode. But in practice long-time calculations are typical and it is reasonable to perform them in a batch queue regime. According to this requirement we have implemented the batch (or, say, 'blind') calculation mode in CompHEP.

During the work with the *n_comphep_f(c)* program in the interactive mode you have a possibility to store all current settings and task parameters in the *f(c)_batch.dat* file. The function *Batch* of the main menu is used to define such a task and save it in a file. It opens the *Menu 8* (see Fig.8). Its first function appends parameters of the current session to the *f(c)_batch.dat* file. It is possible to join several tasks in one *f(c)_batch.dat* file.

For the batch mode you are provided with a possibility to perform two sessions of *Vegas* calculation with automatic call for the *Clear statistic* function between the sessions. These sessions are driven by the *Itmx1, nCall1, Itmx2, nCall2* parameters which in their turn may be defined by means of functions of *Menu 8*. The meanings of these parameters are similar to the same name parameters of the *Vegas* menu. If you put one of these parameters equal to zero then only one session will be launched.

If you set the *Generate events* switch `ON`, then the squared matrix elements calculated during the last *Vegas* session will be saved in a file for further processing (see Section 3.4).

There is a possibility to organize cycle calculation within one task. The *Table size* function defines a number of steps for the cycle calculation. The user may choose a parameter to scan by means of the *Table param* function. The last two menu functions allow to set the extreme values for the scanning parameter.

Every time when you start *n_comphep_f* it checks a presence of the *f_batch.dat* file. If it exists you will be prompted:

```
Would you like to use the BATCH mode?
```

If you answer *'y'* then *n_comphep* reads the tasks from the *f_batch.dat* file and performs requested calculations.

So, if the *f_batch.dat* file exists the batch calculations can be launched by the pipe command:

```
echo y |n_comphep_f
```

The batch calculation in the case of *n_comphep_c* is started by

```
n_comphep_c -batch
```

CompHEP has an internal counter for Monte Carlo sessions which accrues after each Monte Carlo session. All output files have this session number as a part of their names. Hence, outputs from different tasks are written down in different files and all results are safely remembered after the *n_comphep* run. In the case of cycle calculation the additional joint file *f(c)_table*`_N_M` with results of calculation will be created. Here `N` and `M` are the first and the last number of session for the current cycle.

### 3.3.11 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 CompHEP 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<br>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 also displayed on the screen (see Fig.8) 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 or the forward-backward asymmetry as a function of any parameter involved in the calculation.

### 3.3.12 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. CompHEP writes on the screen the calculated particle width. If a multi-channel process like `Z->2*x` has been introduced then branchings for all decay modes are displayed (Fig.9). 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 CompHEP event generator

### 3.4.1 Concept of the event generator

The procedure of event generation consists of two steps. During the evaluation of cross section by *Vegas* CompHEP writes down the calculated squared matrix elements into a binary file *events_*N, where N is the session number. These files are compact owing to occupation of approximately four bytes per one event. All CompHEP settings and a current state of the random number generator are stored there too.

On the next step stand-along command *genEvents* restores the event flow based on this file. It repeats the *Vegas* session, but instead of the real matrix element evaluation it reads the corresponding values from the file. This trick allows one to repeat with a high speed the calculation of particle momenta and event weights produced earlier during the original Monte Carlo integration.

Note that the CompHEP generates the partonic level events. CompHEP passes the task of fragmentation to other programs like PYTHIA [41].

### 3.4.2 The *genEvents* command

To restore events from the *events_*N file the user should use the *$COPMHEP/genEvents* command. In being launched it is waiting for user commands. These instructions are read from the standard input and the outcome is directed to the standard output. Any instruction should be terminated by the end-of-line symbol (the *Enter* key).

The first instruction should contain an event file name, say, *event_1*. If such a file exists and has an appropriate format, *genEvents* writes the process name, e.g.

```
PROCESS: e1,E1 -> e2,E2
```

Then the line(s) with a physical cut definition could follow. The format of these lines is:

```
[function identifier] [min.limit] [max.limit]
```

See Section 3.3.5 for a format of the function identifier. The limits are just numbers. For the case of the $S$ function instead of the values of squared momentum *Val* the user has to put in the quantity $Val/\sqrt{|Val|}$, similar to the case of cut definition in Section 3.3.6.

The last line of input may be empty or contain a desired distribution specification. For the first case the weighted event flow will be created. Otherwise the corresponding distribution will be produced. The format of input which specifies a distribution is the same as that for cuts, but contains an addition field which defines the number of bins:
`[function identifier] [min.limit] [max.limit] [number of bins]`
The number of bins has not to exceed 300.

The *genEvents* command verifies the input. But the user-friendly interface is not implemented here. For the case of wrong input the error message appears and the program terminates. So, in order to escape the repetition of typing we recommend to store the input in a file and pass it to *genEvents* by

```
$COMPHEP/genEvents < user_input_file
```

or

```
cat user_input_file | $COMPHEP/genEvents
```

Let us give an example of *user_input_file*:

```
event_1
A13 5 175
A15 5 175
M35 1 100 50
```

It is an instruction to build a 50 bin distribution for the invariant mass of $3^{rd}$ and $5^{th}$ particles under condition of a 5 degree cut for the angle between the momenta of these particles and the collision axis.

To get a matching of particles with their momenta see the process string in the corresponding protocol file *f_prt_*N.

The output for the distribution has got the following formats.

1. The process string:

```
PROCESS: e1 , E1 -> e2 , E2
```

2. The *name* of the X-axis in double quotas, limits for the axis and a number of bins:

`"Cosine(p1,p3)" from` $-1.0$ `to` $1.00$ $N\_bins = 10$

3. The name of the Y-axis:

```
Diff.  cross section [pb]
```

4. The sequence of *N_bins* lines for the averaged differential cross section for each bin and its statistical uncertainty separated by "+/-":

```
..............
8.0978E+02 +/- 9.2786E+00
...............
```

The following normalization of data is assumed: a sum of all values of the first column multiplied by the bin size is equal to the cross section.

The graphical image of the distribution may be displayed by the $COMPHEP/tab_view command. One of the ways to use it is the long pipe:

```
cat user_input_file | $COMPHEP/genEvents |$COMPHEP/tab_view
```

The options of the *tab_view* program are described in Section 3.1.1. See Fig.7 as an example of plot representation. One more possibility is to save the intermediate result and then use it:

```
cat user_input_file | $COMPHEP/genEvents > hist_file
$COMPHEP/view_tab < hist_file.
```

To generate the event flow the user has to enter an empty line instead of the distribution definition. After that the *genEvents* program writes down the weighted events in the standard output. The first six lines of output contain general information about the process, namely the process name, masses of particles, structure function records, the QCD scale, and values of center-of-mass energy and rapidity. For example,

```
PROCESS: e1 , E1    ->  e2 , E2
MASSES:   0.000000E+00  0.000000E+00  1.057000E-01  1.057000E-01
StructFun1: OFF
StructFun2: OFF
QCD constant scale: 9.12E+01, Lambda6=1.18E-01 GeV
SQRT(S):   9.000E+01
Rapidity:   0.000E+00
```

The sign of rapidity is defined in such a way that its change towards a positive value corresponds to an acceleration of the first incoming particle.

The seventh line contains the column titles. The following lines contain columns of numerical information about events. In the first column the event weights are written down. They are normalized so that the sum of all numbers of the first column gives the calculated cross section or decay width.

The other columns contain the components of three-momenta of particles. The corresponding momentum and component numbers are described by the seventh (title) line of the output. For example, *P2_3* means the third component of the second momentum. Note that the $3^{rd}$ component is assigned to the collision axis. The first incoming particle has this component positive while the second one has it negative. There are no columns for the momenta of incoming particles which are equal to zero by definition, for example, for the $1^{st}$ and $2^{nd}$ components of momenta of colliding particles.

The event flow output is normally very large, so we recommend to redirect the output to the subsequent program for some processing

```
cat user_input_file | $COMPHEP/genEvents | userProcessing
```

A few such programs were created in the framework of CompHEP project. They are disposed in the $COMPHEP directory.

I. `unweight`. It transforms weighted events into unweighted ones. It needs a parameter $maxw$ which specifies the maximum value of weights. Indeed this program transforms the floating number weight into the integer number one according to the following expression

$$floor(\frac{weight}{maxw}) + p\left(\frac{weight}{maxw} - floor(\frac{weight}{maxw})\right) \quad ,$$

where $floor(x)$ function rounds $x$ downwards to the nearest integer and $p(x)$ is the random function which equals 1 with probability $x$ and zero otherwise. If the integer weight does not equal zero then this event is written down into the output in the same format as the unweighted event flow. The true maximum weight may be found in the protocol file.

II. `filterD2`. This program generates decays of one of outgoing particles into two others. It needs 6 arguments:
1) name of disintegrated particle;
2) name of the first decay particle;
3) numerical value of the mass of the first particle;
4) name of the second decay particle;
5) numerical value of the mass of the second particle;
6) branching fraction for this decay mode. For example,

```
filterD2 Z e2 0.1 E2 0.1 0.034
```

will generate decays of the Z boson into muons. This program reads event flow from the standard input and writes down generated events into the standard output. It can be used just after `genEvents` and after `unweight`.

III. `filterDN` This function can generate more complicated sub-decays, for example, `1->3` ones. It is assumed that before its usage the events of corresponding decay have been generated by CompHEP, transformed into the unweighted format, and stored in some file. The name of this file must be passed to `filterDN` as the first argument. Numerical value of the corresponding branching fraction must be passed as the second argument.

You can obtain a wrong result using some part of the whole event sample because *Vegas* generates strongly correlated events according to the *Stratified Sampling* algorithm (see Section D.1). If events are written down in some file they may be mixed up by means of the `$COMPHEP/randomize` program. The first argument of this function is a name of the file where events are stored. The randomized event flow is directed to the standard output.

The programs mentioned above, namely `unweight, filterD2, filterDN, randomize`, can have three optional arguments. They must be some integers which are used to re-initialize the starting point of the random number generator [4].

The generated event flow can be transformed to the table of distribution of some physical variable by means of the `mk_tab` program. This program needs four arguments, namely, a name of variable in format of Section 3.3.5, minimum and maximum limits of this variable and a number of bins, which cannot exceed 300. The `mk_tab` program reads the event flow in the standard input and writes the generated table down to the standard output. This table can be transformed into a plot by means of the `tab_view` utility.

## 3.5 CompHEP 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 CompHEP installation (Section 2.5). It contains the following sub-directories and files:

```
models/ results/ tmp/
comphep comphep.ini
```

[4] We use standard random number generator *drand48*.

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 CompHEP 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 CompHEP session. In the case of such an error we recommend to remove the *tmp/safe* file.

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

You can toggle on/off the color and the sound for a CompHEP session as well as choose the most appropriate font for your X-terminal. To do this just edit the *comphep.ini* file. We hope the syntax there is self-explanatory enough.

The command

```
./comphep
```

starts a CompHEP session. This is a shell script which calls commands disposed in the CompHEP root directory. It is assumed that the *$COMPHEP* environment variable (see Section 2.5) points to this directory.

### 3.5.2 Scheme of calls in the CompHEP session

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

```
$COMPHEP/s_comphep
```

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

Below we would like to describe commands for compilation and numerical module launching. The command names in the *C* and *Fortran* cases are distinguished by suffixes *'c'* and *'f'* correspondingly. We shall use the '*' symbol as a common notation for these *'c'* and *'f'*. The general scheme of calls can be presented by the following diagram

```
                                          | make__n_comphep * → | nCompil_*
s_comphep → launch_n_comphep * →          |                     | ld_*
                                          | n_comphep_*
```

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

The program nCompil_* compiles the source code produced prior by *s_comphep*. After successful compilation it creates a library of object files and then removes the source and the object files.

Another command ld_* calls a linker to create the executable file n_comphep_* for numerical job.

The command *make__n_comphep* with parameter 'c' or 'f' is used to start nCompil_* and ld_* subsequently.

In its turn, *make__n_comphep* is started by *launch_n_comphep* in a special window. In the case of successful compilation the window is closed and just created n_comphep_* is started in another window.

If you copy *$COMPHEP/make__n_comphep* into your *WORK/* directory, then *launch_n_comphep* will call for your own version. It gives you a possibility to modify a compilation procedure if necessary.

The *s_comphep* command starts *launch_n_comphep* via the functions

```
.............
 C-compiler
Fortran compiler
```

of *Menu 6* (Fig.5) and thus the chain of above calls is realized.

If you have prepared the *C* or *Fortran* codes you could compile them outside of the symbolic session by the *make__n_comphep* command with the *c/f* argument started from within the *results* directory.

Let us say some words about other programs stored in the *$COMPHEP* directory. The program *genEvents* is used to prepare an event flow and histograms using the information saved by Monte Carlo session. The programs *unweight*, *randomize*, *filter2D*, *filterND*, and *mk_tab* transform the event flow generated by *genEvents*. See Section 3.4 for details. The program *tab_view* provides the user with a possibility of visual presentation of distributions. See Section 3.1.

### 3.5.3 LOCK files

*WORK/comphep* creates a file *LOCK* in the *WORK/* directory. Presence of the *LOCK* file prevents a double launching of *WORK/comphep* from within the same place of the file system. *LOCK* is automatically removed in the end of the *WORK/comphep* session. In the same manner *$COMPHEP/launch_n_comphep* creates a file *c_LOCK* or *f_LOCK* in the *WORK/results* sub-directory.

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

## 3.6 User programs in CompHEP

### 3.6.1 Concept of user program implementation

We provide the user with a possibility to attach his own codes to the *n_comphep_f(c)* and *genEvents* programs. In this way you are able to
1) expand the set of phase space functions for cuts and histograms
2) implement new structure functions.

CompHEP archive file *num_f(c).a* contains some patch programs. When you create your version of these programs and pass the names of its object files to the linker, they will be embedded into the executable file.

In the case of *n_comphep_f(c)* command the following instruction does this job:

```
$COMPHEP/make_n_comphep f(c) userObjectFiles.o
```

It is assumed that the above command is started from within the user's directory *results* which contains sources or their archives.

The *genEvents* file is disposed in the *$COMPHEP* directory and generally cannot be modified by the user. But one could create his own version of this program. In this case use

```
f77 -o userEvents $COMPHEP/events.o userObjFile.o $COMPHEP/num_f.a
```

### 3.6.2 *Fortran* case[5]

The user phase space function is evaluated by

```
REAL*8 FUNCTION USRFUN(CHARACTER *9 TEXT)
```

The patch is disposed in the file *$COMPHEP/f_source/usr/u_var.f.*

This function is called by *n_comphep_f* and *$COMPHEP/genEvents* programs if the phase space function whose name begins with the 'U' character is used (see Section 3.3.5). The characters following 'U' are passed to the *USRFUN* routine as its argument.

To create this function one needs to know the particle momenta. They are stored in

```
COMMON/PVECT/ REAL*8 P(0:3,100)
```

The first argument of array $P$ is a Lorentz momentum component. $P(0,N)$ is the energy of the $N^{th}$ particle. We have $P(0,N) > 0$ as for incoming as for outgoing particles. In the case of collision process $P(3,N)$ is a projection of the particle space momentum onto the collision axis. The direction of this axis is chosen so that $P(3,1) > 0$ and $P(3,2) < 0$.

The second argument of this array is a momentum ordering number. The first momentum numbers are assigned to incoming particles and the subsequent ones are assigned to outgoing particles. The numbers of incoming and outgoing particles and a correspondence between the particle name and the momentum number may be determined by means of service functions described in Section 5.4.

The work of user structure function is controlled by a set of routines. The patches for them are disposed in *$COMPHEP/f_source/usr/sf_prv.f.* All of them should be realized by the user:

1. *LOGICAL FUNCTION P_PRV(CHARACTER*6 P_NAME)*

returns *TRUE* if the user structure function can be applied to the *P_NAME* particle. It is used to create the menu of possible structure functions for the particle *P_NAME.*

2. *SUBROUTINE M_PRV(I)*

is called for the user input of the structure function parameters just after the user structure function has been chosen. It is assumed that this routine saves values of the entered parameters in some COMMON for subsequent usage in

[5] Below we insert parameter type definitions into headers of the *Fortran* routines. It is done for brevity only and contradicts to the *Fortran* style

N_PRV described below. The argument *'I'* here and in the following means an incoming parton number. It can be 1 or 2.

3. *CHARACTER*60 FUNCTION N_PRV(I)*
returns the structure function name. If the structure function has a few tuning parameters then the values of these parameters must be included in the returned name. CompHEP uses the structure function name to keep information about parameters.

4. *LOGICAL FUNCTION R_PRV(I,CHARACTER*60 NAME, REAL*8 CMASS, REAL*8 BE)*
checks *NAME*. If *NAME* has the same format as the one produced by *N_PRV()*, *R_PRV()* should read the parameter values, store them in some global variables for subsequent usage, and return TRUE. Otherwise FALSE is returned. *REAL*8 CMASS* and *REAL*8 BE* are outgoing parameters. The first one must be equal to the mass on the composite particle which constituents are described by user's structure function. The second one informs CompHEP that the structure function is singular as
$(1-X)^{(BE-1)}$, where $0 < BE \leq 1$.
In the nonsingular case $BE = 1$ must be returned.

5. *REAL*8 FUNCTION C_PRV(I,REAL*8 X)*
returns the value of structure function divided by $BE*(1-X)^{(BE-1)}$. Here $X$ is the Feynman scaling variable. It is assumed that possible structure function parameters having been stored by *R_PRV* are used for evaluation. The structure function is normalized so that

$$be(1-x)^{be-1}c\_prv(i,x)dx$$

is a probability to find the $i^{th}$ parton with fractions $[x, x+dx]$ of initial momentum.

In the simplest case, if the user would like to implement a new structure function without singularity and extra parameters, he can just take a copy of the *$COMPHEP/f_source/usr/sf_prv.f* file, improve *P_PRV* so that it returns $'.TRUE.'$ anyway, and rewrite originally trivial *C_PRV*.

### 3.6.3 *C* case

The user phase space function has a prototype

```
double usrfun(char * name)
```

The patch is disposed in the file *$COMPHEP/c_source/num/userFun.c*. This function is called by *n_comphep_c* programs if the phase space function whose name is started from the 'U' character is used (see Section 3.3.5). The characters following 'U' are passed to the *usrfun* routine as its argument.

Particle momenta which are needed to evaluate *usrfun* are stored in the global variable

```
double pvect[400]
```

The $m^{th}$ component of $k^{th}$ momentum occupies the $m + 4 * (k - 1)$ position in *pvect*. We assume that 'm' varies from 0 to 3 and the 'k' momentum counter starts from 1. The agreement about momentum signs is the same as in the *Fortran* case. The number of incoming and outgoing particles and the correspondence between the particle name and momentum number can be determined by means of service functions described in Section 5.5.

The work of user *structure function* is driven by a set of routines similar to those of the *Fortran* case. All of them have to be realized by the user if he would like to implement a particular structure function. Patches for these routines are disposed in

*$COMPHEP/c_source/num/strfun/sf_prv.c.*

This file contains:

1) *int p_prv(char *particleName)*

returns *1* if the user structure function can be implemented to the *particleName* particle. Otherwise 0 should be returned. It is used to create a menu of possible structure functions;

2) *void m_prv(int i)*

is called for the user input of the structure function parameters just after the particular structure function is chosen. It is assumed that the input is saved in some static variables. The argument *'i'* here and below means an incoming parton number. It could be 1 or 2;

3) *void n_prv(int i, char * funcName )*

returns the structure function name. If the structure function has a few tuning parameters then values of these parameters have to be included in *funcName*. CompHEP uses the structure function name to keep information about parameters;

4) *int r_prv(int i, char *funcName)*

checks *funcName*. If *funcName* has an appropriate format, *r_prv* must read the value of parameters, store them in some global variables and returns 1. Otherwise 0 is returned;

5) *int mass_prv(int i)*

returns the mass of the composite particle which constituents are described by this structure function.

6) *double be_prv(int i)*
performs needed initializations before structure function evaluation. It returns parameter *be* which informs CompHEP that the structure function is singular as

$$(1-x)^{(be-1)};$$

7) *double c_prv(int i, double x)*
returns the value of structure function divided by $be(1-x)^{(be-1)}$, where $x$ is the Feynman scaling variable. It is assumed that possible turning parameters have been stored by *r_prv* in some static variables.

In the simplest case, if the user would like to implement a new structure function without singularity and extra parameters, he may take a copy of the *$COMPHEP/c_source/num/strfun/sf_prv.c* file, improve *p_prv* so that it returns 1 in any case, and rewrite the originally trivial *c_prv* function.

# 4 Implementation of models of particle interactions

## 4.1 Definition of a model in CompHEP

Description of particle interaction model in CompHEP 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;

   CompHEP 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, if the identifier "Me" is used in the table for the electron mass, then the forms "ME", "mE","me" are forbidden as for new definitions as for identification of the electron 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 $\gamma^5$ Dirac matrix;

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

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

### 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 parameres (see above);

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

   - integer numbers,
   - identifiers enumerated in the *Parameters* Table,
   - identifiers defined above in this Table,
   - parentheses (), arithmetic operators +, -, /, *, **, and the *sqrt*() function.

   For raising to a power the second operand must be an integer;

3. *Comment* for brief description of parameters.

#### 4.1.3 Description of particles

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

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

2&3. *"A"* and *"A+"* 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 *"A+"* fields must be identical;

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 CompHEP 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 *"A+"* 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:

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

   (b) spin 1/2 case:

   $$< 0|T[A(p_1), A^+(p_2)\ \gamma_0]|0 >= (\not{p}_1 + M)\, ScPr(p_1, p_2, M) \quad ,$$

   where
   $$\not{p} = p^\mu \gamma_\mu \quad .$$
   Here "$A^+$" is the Hermitian conjugation of "$A$"; thus the Dirac $\gamma_0$ matrix is needed to make the Dirac conjugate field;

   (c) spin 1 case:

   $$\begin{aligned} < 0|T[A^{m_1}(p_1),\ (A^{m_2})^+(p_2)]0 > &= -(g^{m_1 m_2} + p_1^{m_1}\ p_2^{m_2}/M^2) \\ &\times\ ScPr(p_1, p_2, M) \ . \end{aligned} \tag{5}$$

   Zero mass vector particle must be marked as a *gauge* one using the *Aux* field (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

$$\frac{\not p_1(1+\gamma_5)}{2}\, ScPr(p_1,p_2,M) \ \ .$$

This is a way to introduce a left-handed fermion as a neutrino or a massless polarized fermion. When CompHEP 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

$$\frac{\not p_1(1-\gamma_5)}{2}\, ScPr(p_1,p_2,M) \ \ .$$

This is a way to introduce right-handed fermions;

'*' is permitted for massive particle only. In this case $ScPr(p_1,p_2,M)$ 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 (5).

Such particles cannot appear as incoming or outgoing ones. They are used to describe a point-like interaction as one has in the electroweak 4-fermion interaction model;

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

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

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 CompHEP. The names are used in the mathematical mode.

### 4.1.4 Ghost fields in CompHEP

CompHEP constructs a list of quantum fields according to the Table of particles described above. Besides of names enumerated in this table CompHEP generates auxiliary fields, for example, the Faddeev-Popov ghosts [42]. 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 prototype. The names of ghost fields are constructed by CompHEP from the prototype 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 CompHEP 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 [6]. The nonzero propagators for these fields are:

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

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

where $M$ is a mass of the prototype 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 [42].

**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.

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

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

where $M$ is a mass of the prototype particle.

**Tensor ghost.** Whereas the Faddeev-Popov and Goldstone ghosts are standard elements of modern quantum field theory, the tensor ghost is an original CompHEP 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 prototype particles. It is Lorentz-transformed like a tensor field. The propagator is

$$< 0|T[\text{A+.t}^{m_1 M_1}(p_1),\ A.t^{m_2 M_2}(p_2)]|0 >= \frac{1}{(2\pi)^4 i}\,\delta(p_1+p_2)\,g^{m_1 m_2}\,g^{M_1 M_2}\ . \tag{6}$$

#### 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 CompHEP 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

$$\frac{\delta S}{\delta A1_{[m1]}(p1)\,\delta A2_{[m2]}(p2)\,\delta A3_{[m3]}(p3)\,[\delta A4_{[m4]}(p4)]} =$$
$$(2\pi)^4\delta(p1+p2+p3\,[+p4])\,[\gamma_0]\,ColorStructure\cdot Factor\cdot LorentzPart\;. \qquad (7)$$

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

$$A(x) = \int \exp(-ikx)A(k)\,d^4k\;. \qquad (8)$$

*'Factor'* must be a rational monomial constructed of the model identifiers, integer numbers and imaginary unity.

*'LorentzPart'* must be a tensor or Dirac $\gamma$-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 $\gamma$-matrix with index $'m'$ we use a symbol $G(m)$, whereas $G(p)$ denotes $p_\mu\gamma^\mu$. Anti-commutation relations for $\gamma$ matrices should be written as

$$G(v1)\,G(v2) + G(v2)\,G(v1) = 2\;v1.v2, \qquad (9)$$

where $v1, v2$ are momenta or indices.

The $\gamma_5$ matrix is denoted by $G5$. It is defined by equation

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

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).

CompHEP interprets the anti-particle spinor field as the Hermitian conjugated particle field, rather than the Dirac conjugated one. Also it is assumed that all spinor fields are written in the Majorana basis and the matrix of C-conjugation is equal to $(-\gamma_0)$ [7]. After substitution $C \to -\gamma_0$ all possible vertices could be written as

$$A1(p1)\,\gamma_0\,G(v1)\,G(v2)\ldots G(vn)\,A2(p2)$$

where $A1$ and $A2$ are some spinor fields, each of them corresponding to particle or anti-particle. This form of Lagrangian is assumed in the vertex expression (7). $\gamma_0$ is substituted by CompHEP automatically in the case of a spinor particle vertex and is not expected in *LorentzPart* of (7).

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 (9).

Let us note that by definition (7) the *LorentzPart* has the corresponding symmetry property in the case when identical particles appear in one vertex. This symmetry is not checked by CompHEP, but its absence will lead to wrong results. The following equation may be used to check the symmetry in the case of fermion vertex:

$$\begin{aligned} A1(p1)\,\gamma_0\,G(v1)\,G(v2)\ldots[G5]\ldots G(vn)\,A2(p2) = \\ A2(p2)\,\gamma_0\,(-G(vn))\ldots[G5]\ldots(-G(v2))\,(-G(v1))\,A1(p1)\,. \end{aligned} \tag{10}$$

It may be useful also to check the Lagrangian self-conjugation property. Note that the anticommutation of *A1* and *A2* is already taken into account in (10).

[7] Above the Majorana spinor was defined as a Hermitian self-conjugated one: $\psi = \psi^+$. In the same time it must be $C$-self-conjugated too: $\psi = C\bar{\psi}^T$. Following these requests we get the phase of $C$ which is different from one chosen in the textbook [42]

*ColorStructure* is substituted by CompHEP automatically. For a colorless particle vertex it is equal to 1. For $(3\times\bar{3})$ and for $(8\times 8)$ vertices the unity tensor is substituted. If CompHEP 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 CompHEP 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 6). 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 (6).

## 4.2 Examples

### 4.2.1 Implementation of QCD Lagrangian

**3-gluon vertex.** Lagrangian (40) 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 $G^\alpha_\mu$ is the gluon field, $g$ is the strong coupling constant. Applying the Fourier transformation (8) we get

$$S_{3G}=(2\pi)^4g\int\delta(p_1+p_2+p_3)\ i\ f_{\alpha_1\alpha_2\alpha_3}p_1^{\mu_2}g^{\mu_1\mu_3}G^{\alpha_1}_{\mu_1}(p_1)G^{\alpha_2}_{\mu_2}(p_2)G^{\alpha_3}_{\mu_3}(p_3)d^4p_1d^4p_2d^4p_3\ .$$

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 CompHEP vertex representation (7) where *ColorFactor* is $(-i\, f_{\alpha_1\alpha_2\alpha_3})$ we get

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

CompHEP uses the notation $GG$ for the strong coupling constant $g$. So for the 3-gluon vertex in the CompHEP 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 (40):

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

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

$$S_{QqG} = g(2\pi)^4\int \delta(p_1+p_2+p_3)G_\mu^\alpha(p_3)q^+(p_1)\gamma^0\gamma^\mu\hat{t}_\alpha q(p_2)\, d^4p_1 d^4p_2 d^4p_3 ;$$

$$\frac{\delta S_{QqG}}{\delta q_i^+(p_1)\delta q^j(p_2)\delta G_\mu^\alpha(p_3)} = g(2\pi)^4\delta(p_1+p_2+p_3)(\hat{t}_\alpha)^i_j\gamma^0\gamma^{\mu_3} .$$

The factor $(2\pi)^4\delta(p_1+p_2+p_3)(\hat{t}_\alpha)^i_j\gamma^0$ is substituted by CompHEP automatically. Thus, the quark-gluon interaction is implemented in the CompHEP *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 (42), 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 CompHEP. 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 CompHEP notations for the Faddeev-Popov ghosts $\bar{c}$ and $c$ respectively.

**4-gluon interaction.** Besides the 3-gluon interaction term the Lagrangian (40) 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:

$$\frac{\delta S_{4G}}{\delta G^{\alpha_1}_{\mu_1}(p_1)\delta G^{\alpha_2}_{\mu_2}(p_2)\delta G^{\alpha_3}_{\mu_3}(p_3)\delta G^{\alpha_4}_{\mu_4}(p_4)} = -g^2(2\pi)^4\delta(p_1+p_2+p_3+p_4)\delta_{\epsilon\epsilon'}\times$$
$$\Big(f^\epsilon_{\alpha_1\alpha_2} f^{\epsilon'}_{\alpha_3\alpha_4}(g^{\mu_1\mu_3}g^{\mu_2\mu_4} - g^{\mu_1\mu_4}g^{\mu_2\mu_3}) + f^\epsilon_{\alpha_1\alpha_3} f^{\epsilon'}_{\alpha_2\alpha_4}(g^{\mu_1\mu_2}g^{\mu_3\mu_4} - g^{\mu_1\mu_4}g^{\mu_2\mu_3})$$
$$+ f^\epsilon_{\alpha_1\alpha_4} f^{\epsilon'}_{\alpha_2\alpha_3}(g^{\mu_1\mu_2}g^{\mu_3\mu_4} - g^{\mu_1\mu_3}g^{\mu_2\mu_4})\Big)\ . \qquad (11)$$

The complicated color structure of this vertex cannot be directly written down in the CompHEP 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 CompHEP adds a tensor field with the same color to the internal list of quantum fields. The propagator of this field (6) 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

$$\begin{aligned}
S_{tGG} &= \frac{i\,g}{\sqrt{2}}\int f^{\alpha}_{\beta\gamma}t_{\alpha}{}^{\mu\nu}(x)G^{\beta}_{\mu}(x)G^{\gamma}_{\nu}(x)d^4x\;;\\
\frac{\delta S_{tGG}}{G^{\alpha_1}_{\mu_1}(p1)G^{\alpha_2}_{\mu_2}(p2)\delta t^{\alpha_3}_{\mu_3\mu_3\prime}(p_3)} &= (2\pi)^4\delta(p_1+p_2+p_3)(-\,i\,f_{\alpha_1\alpha_2\alpha_3})\\
&\times\;\frac{g}{\sqrt{2}}(g^{\mu_2\mu_3}g^{\mu_1\mu_3\prime}-g^{\mu_1\mu_3}g^{\mu_2\mu_3\prime})\;.
\end{aligned}$$

In CompHEP 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 CompHEP 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.13. Contribution of each of these diagrams corresponds to one of the terms of expression (11).

### 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 CompHEPone 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 $\Psi_\nu$ a neutrino appears in the Standard Model Lagrangian in the following way (55)[8]:

$$\begin{aligned} L_\nu &= \frac{i}{2}(\bar{\Psi}_\nu\gamma_\mu\partial^\mu\Psi_\nu - (\partial^\mu\bar{\Psi}_\nu)\gamma_\mu\Psi_\nu) + \frac{e}{4\sin\Theta_w\cos\Theta_w}Z_\mu\bar{\Psi}_\nu\gamma^\mu(1-\gamma^5)\Psi_\nu \\ &+\frac{e}{2\sqrt{2}\sin\Theta_w}\left(W_\mu^-\bar{\Psi}_e\gamma^\mu(1-\gamma^5)\Psi_\nu + W_\mu^+\bar{\Psi}_\nu\gamma^\mu(1-\gamma^5)\Psi_e\right) \;, \end{aligned} \tag{12}$$

where $\Psi_e$ 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 $\psi_l$ and $\psi_r$ are Majorana fermions. Omitting Lagrangian for $\psi_r$ 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 (10), 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 (7) we can rewrite

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

them in the CompHEP notations:

| A1 | A2 | A3 | A4 | Factor | Lorentz part |
|---|---|---|---|---|---|
| MN | MN | Z | | -EE/(2*SW*CW) | G(m3)*G5 |
| E1 | MN | W- | | EE/(2*Sqrt2*SW) | G(m3)*(1-G5) |
| MN | e1 | 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 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 (7) with the help of the identity (10) 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 $\Psi_e$ and $\Psi_u$ 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 CompHEP 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 (7) gives us the *Vertex* table

| A1 | A2 | A3 | A4 | Factor | Lorentz part |
|---|---|---|---|---|---|
| u | e1 | F | | -lambda | (1+G5) |
| E1 | U | F+ | | lambda | (1+G5) |

By means of equation (10) we can rewrite this table in the equivalent form:

| A1 | A2 | A3 | A4 | Factor | Lorentz part |
|---|---|---|---|---|---|
| e1 | u | F | | -lambda | (1+G5) |
| U | E1 | F+ | | lambda | (1+G5) |

### 4.3 LanHEP and SUSY models

There is a possibility of automatic conversion of the Lagrangian written in the compact form in the coordinate space into the CompHEP table format. It can be performed by means of the *LanHEP* program written by A. Semenov [16]. The input is expressed in terms of complex objects, such as a covariant derivative and a gauge field tensor.

The *LanHEP* program was used [17] to generate CompHEP model files for the Minimal Supersymmetric Model. Another result produced by *LanHEP* is the realization the general two-Higgs-doublet model [18] The CompHEP WWW page contains references to the LanHEP code and to the MSSM Lagrangian model files.

# 5 CompHEP output files

## 5.1 LaTeX output

CompHEP uses the *Axodraw* package by J.A.M. Vermaseren [45] 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 $COMPHEP/ directory. One bug in the axis drawing routine has been corrected.

The *Axodraw* syntax is very easy and you can alter the CompHEP 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 CompHEP 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 CompHEP 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* [33] and *Mathematica* [34] 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 $p1, p2, p3, \ldots$. 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.

CompHEP 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.11. 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* package

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

*sum.red* which presents the squared matrix element as a common denominator expression;

*sum2pole.red* which presents the squared matrix element as a sum of pole terms;

*sum2tot.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 *COMPHEP/test* directory.

Let you prepare the symbolic output *symb1.red* for the Compton scattering *A,e1* `->` *A,e1* in the framework of QED model [9]. Launch the *Reduce* system from within the *result* directory. The possible *Reduce* sessions are:

```
%1
  in"$COMPHEP/test/sum.red"$      % loading the summation package
  in"symb1.red"$                  % reading contributions of diagrams
  sum;                            % writing 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
*me**2-p1.p2*p1.p3**3 + 2*p1.p2*p1.p3**2*me**2 + p1.p3**2*me**4))
/(propden(-p1-p2,me,0)**2*propden(p2-p3,me,0)**2)$

%2
  in"$COMPHEP/test/sum2pole.red"$  % loading the summation package
  in"symb1.red"$                   % reading contributions of diagrams
  sum;                             % writing the answer
2*ee**4*(4*sp(me)**2*me**4 + 8*sp(me)*up(me)*me**4 + 4*sp(me)*me**2 +
sp(me)*t+ 4*up(me)**2*me**4 + 5*up(me)*me**2 - up(me)*s + 1)$
```

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

$$sp(x) = 1/(s - x^2);$$

$$tp(x) = 1/(t - x^2);$$

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

```
%3
  in"$COMPHEP/test/sum2tot.red"$ % loading the summation package
```

[9] In other CompHEP models the electron is massless.

```
  in"symb1.red"$                  % reading contributions of diagrams
  sum;                            % writing answer for total cross section
(ee**4*(2*s**4*log(s/me**2) + s**4 - 12*s**3*log(s/me**2)*me**2 + 14*s**3*me**2
- 6*s**2*log(s/me**2)*me**4 - 16*s**2*me**4 + 2*s*me**6 - me**8))/(16*s
**2*pi*(s**3 - 3*s**2*me**2 + 3*s*me**4 - me**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 express 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.m*, *sum2pole.m*, and *sum2tot.m* for operation with the *Mathematica* output.

## 5.3 *Reduce* program

The *Reduce* program was the first CompHEP output which opened the possibility to produce some physical results by means of this package. Later on, when the CompHEP built-in symbolic calculator was created, the *Reduce* output became unnecessary. But we still keep it in the package for testing.

The CompHEP 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 CompHEP software (see Section A). Below we shall describe the structure of *Reduce* program.

CompHEP 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 $\gamma^5$ 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.11). 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. CompHEP transforms the reverse vertices to normal form by means of the rule (10) and writes down the corresponding comment.

After the last multiplication the state of the *spur* switch is restored and the $\gamma$-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 (5) the code are organized in the following way. CompHEP 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 relevent 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 discribed 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 whearas 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 *Fortran* output files

CompHEP *Fortran* output is used to compile the squared matrix element for subsequent evaluation of cross sections and distributions. The explanation below is necessary only if you would like to use the CompHEP *Fortran* output for other phase space integration program or, vice versa, to use the CompHEP routines to integrate a phase space function produced in some other manner outside CompHEP.

Below we insert the declaration of parameter type into the header of *Fortran* routine. It is done for brevity only and contradicts to the *Fortran* style. We use also the *Real*X* designation for declaring the parameter of floating point type which is *Real*8* or *Real*16* depending on the type of output.

`FUNCTION LENR()`
returns 8 in the case when CompHEP produces a code with Real*8 (DOUBLE PRECISION) floating arithmetics or 16 in the case when CompHEP produces a code with Real*16 (QUADRUPLE PRECISION) floating arithmetics.

`FUNCTION NIN()`
returns a number of incoming particles.

`FUNCTION NOUT()`
returns a number of outgoing particles.

`FUNCTION NPRC()`
returns a total number of subprocesses.

`CHARACTER*6 FUNCTION PINF(NSUB,NPRTCL)`
returns a particle name for the subprocess `NSUB`. `NPRTCL` is the particle ordering number in the subprocess. The first `NIN()` numbers numerate the incoming particles.

`SUBROUTINE PMAS(NSUB,NPRTCL,REAL*X VAL)`
returns the particle mass. Incoming parameters are: `NSUB` - the subprocess number and `NPRTCL` - the ordering number of particle. Returned parameter is VAL. In the case when LENR()=8 VAL must be a REAL*8 variable,

otherwise it must be described as a REAL*16 one.

`FUNCTION NVAR()`

returns a number of physical parameters involved in the calculation.

`SUBROUTINE VINF(NUMVAR,CHARACTER *6 NAME,REAL*8 VAL)`

provides the information about physical parameters involved into evaluations. It returns the physical parameter name NAME and its value VAL. The incoming `NUMVAR` is the parameter ordering number which must be smaller than or equal to `NVAR()`.

`SUBROUTINE ASGN(NUMVAR,REAL*8 VALNEW)`

assigns to the parameter `NUMVAR` a new value `VALNEW`.

`SUBROUTINE VINI`

makes the initialization of all physical parameters.

`SUBROUTINE CPTH(CHARACTER*60 PATH, CHARACTER*1 F_SLASH)`

assigns new values to all its parameters. *PATH* is the path to the CompHEP root directory (see Section 2.5). `F_SLASH` is a symbol which separates file name in the path. It is "/" or "\" in the *UNIX* and *MS-Windows* cases correspondingly.

`REAL*8 FUNCTION SQME(NSUB)`

returns a numerical value of squared matrix element for the subprocess *NSUB*. Summation over out-particle polarization states and an averaging over in-particle polarization states are carried out. Information about scalar products of momenta must be passed on to the *SQME* through

$$COMMON/SCLR/REAL*X\ \ PP(15) \tag{13}$$

The assignments of scalar product values to elements of the *PP* array, like

$$PP(INDX(k,l)) = p_k{\cdot}p_l \quad ,$$

must be done before the *SQME* call. We assume that all four-momenta $p_i$ of particles have positive energy components ($p_i^0 > 0$). *PP* must be of the *REAL*8* type in the case of double precision calculation and of the *REAL*16* type for the quadruple precision one.

*SQME* is sensitive to the states of *LOGICAL* flags *GWIDTH* and *RWIDTH* from the *COMMON/WDTH/GWIDTH,RWIDTH*. They define the treatment of particle finite width. The *GWIDTH* flag indicates that the widths do not break gauge invariance. *RWIDTH* states that the running widths are substituted. See the discussion in Section 3.3.4.

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

$$[P]^{(2*(NIN()+NOUT()-4))}$$

units, where `[P]` is a unit of momentum. Standard normalization for SQME is used. See Eqs. (23.12) and (23.26) in [36]. When *n_comphep_f* calls the *SQME* function it substitutes momenta in the *GeV* units.

`FUNCTION INDX(K,L)`
returns the position of scalar product $p_k.p_l$ in the *PP* array (13).

## 5.5 *C* output files

CompHEP *C*-output is quite similar to the *Fortran* one. Units and normalizations are also the same. Now we shall list the functions and the global constant parameters defined in this output and called by the program *n_comphep_c*.

### Parameter section

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

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

`int calcFunc(void)` calculates all constrained parameters for current values of independent ones. It returns 0 in the case of success, otherwise 1.

`int vinf_(int numvar, char *name, double *val)` provides the information about parameters. Here the incoming parameter *numvar* is a parameter number. The outgoing parameters **name* and **val* are the parameter name and value, correspondingly. If *name or **val* is *NULL* no assignment for the corresponding parameter will be done.

The parameter numbers in the range from 1 till *nvar_* are reserved for independent parameters and the next *nfunc_* numbers are used for constraints. Equal-to-zero *numvar* is associated with the center of mass energy. If the `numvar` value is out of this range, `vinf_(..)` returns 1.

`int asgn_(int numvar, double valnew);` assigns a new value *valnew* to the *numvar* parameter. It returns 0 if $0 <= numvar <= nvar\_$, or 1 otherwise.

### Process section

`const int nin_` is a number of incoming particles

`const int nout_` is a number of outgoing particles

`const int nprc_` is a total number of subprocesses

`char processch[]` contains the process name. It is needed to be only displayed on the screen during the *n_comphep_c* session.

`int pinf_(int nsub, int nprtcl, char *name, double * mass)` returns the *name* and the *mass* of particle with the *nprtcl* ordering number for the subprocess *nsub*. First *nin_* numbers are reserved for incoming particles, while next *nout_* are assigned to outgoing ones. One can substitute the *NULL* constant instead of any outgoing parameter if the corresponding information is not needed. The function returns 0 if $nsub <= nprc\_$ and $nprtcl <= nin\_ + nout\_$, otherwise 1 is returned.

### The squared matrix element

`double sqme_(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_* 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.

### Widths implementation parameters

`int rwidth, gwidth` are the logical variables which define a particle width treatment for calculations of the squared matrix element. They mean *running width* and *gauge invariant 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.

# Appendix

# A Self-check of the CompHEP package

The CompHEP authors have invented some tools for testing the program. The positive result of these checks allows us to justify that CompHEP works correctly. Our tests relate to the symbolic result level and have been realized with the help of the *Reduce* [33] symbolic manipulation system. All test routines are stored in the *$COMPHEP/test* 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 CompHEP 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 CompHEP 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 CompHEP 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 CompHEP 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 asymptotics of its propagator

$$\frac{i}{(2\pi)^4}\frac{g_{\mu\nu}-k_\mu k_\nu/m^2}{m^2-k^2}\,. \tag{14}$$

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 [42]. 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 [42]. 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

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

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 (14) the propagator (15) does not vanish when it projected onto the temporal polarization state

$$e^0 = k/m \ , \tag{16}$$

that also leads to the appearance of additional unphysical state. The main principles of gauge invariance guarantee [42] 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

$$e^1_\mu e^1_\nu + e^2_\mu e^2_\nu + e^3_\mu e^3_\nu = k_\mu k_\nu / m^2 - g_{\mu\nu} \tag{17}$$

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

$$k = (\sqrt{m^2 + p^2}, 0, 0, p) \ . \tag{18}$$

Then the polarization vectors can be chosen as

$$e^1 = (0, 1, 0, 0) \ ; \tag{19}$$
$$e^2 = (0, 0, 1, 0) \ ; \tag{20}$$
$$e^3 = (p/m, 0, 0, \sqrt{1 + p^2/m^2}) \ . \tag{21}$$

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 (17).

## 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 (16). Note that the Faddeev-Popov ghost states and the temporal state have a negative norm, whereas the Goldstone state has a positive norm [42]. 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 [44]. As a result

$$\sum_{i \in S_{phys}} A_i A_i^* = \sum_{i \in S_{all}} \sigma(i) A_i A_i^* \ , \tag{22}$$

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 (16) and longitudinal (21) 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

$$e^0_\mu e^0_\nu - e^3_\mu e^3_\nu = e'^0_\mu e'^0_\nu - e'^3_\mu e'^3_\nu \ , \tag{23}$$

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 (17) by

$$e^0_\mu e^0_\nu - e^1_\mu e^1_\nu - e^2_\mu e^2_\nu - e^3_\mu e^3_\nu = g_{\mu\nu} \; . \tag{24}$$

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 (24). 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 (15) with $m = 0$ and to the appearance of Faddeev-Popov ghosts [42]. 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 (22). See the corresponding proof in [43].

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 [43]

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

where $\eta$ is an auxiliary vector with the zero Lorentz norm. Due to the gauge invariance the sum over all diagrams does not depend on this $\eta$, 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 CompHEP

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

$$\sum_{i\in S_{all}} A_i^k A_i^{*l} \tag{25}$$

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.12(a) is accompanied by three ghost diagrams with the similar topology.

Let us note that all diagrams of Fig.12 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 CompHEP 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 cancelation 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.14. 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 [27] in the zero-electron-mass limit. This fact is caused by electro-magnetic U(1) gauge invariance. If diagrams of Fig.14 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 C.4).

# C Distribution functions and beam spectra

## C.1 *CTEQ4m* distribution functions

There are several sets of disribution functions presented by CTEQ collaboration which describe the densities of quarks and gluons in (anti)proton. In CompHEP we have implemented the *CTEQ4m*[24] set. Apart from the Feynman variable $x$ they depends on the QCD-scale parameter $Q$. The available range is

$$\begin{array}{lll} 10^{-5} & < x < & 1; \\ 1.6 & < Q < & 10000 \text{ GeV}. \end{array}$$

For points outside these ranges the extrapolation is used and the corresponding warning is directed to the standard output file.

The *CTEQ4m* set of functions is produced in the framework of the next-to-leading calculations in the standard $\overline{MS}$ scheme with $\alpha_s(M_Z) = 0.116$. See further explanations in [24].

## C.2 *MRS* structure functions

The *MRS* group also presents several sets of proton structure functions. Two of them are implemented in CompHEP. They are *MRS*($A'$) and *MRS*($G$) [25].

In the $A'$ case it is supposed that the sea quark and gluon densities are parameterized at small $x$ by the $x^{-\lambda}$ function where $\lambda$ is the same for the gluon and for the sea quarks. This suggestion is motivated by the QCD theory. The G set uses two different $\lambda$ parameters for the sea quarks and for the gluon densities. A fit of experimental data for the G set is better.

The next-to-leading order evaluation and the $\overline{MS}$ factorization scheme are used for evaluation of the $Q^2$ dependence of structure function. $\alpha_s(M_Z) = 0.113$ [0.114] in the $A'$ [$G$] case, correspondingly.

The available range is

$$\begin{array}{lcl} 10^{-5} & < x < & 1; \\ 5 & < Q^2 < & 1310720 \text{ GeV}^2. \end{array}$$

See complete explanation in [25].

## C.3 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 [26] for more details.

## C.4 Weizsaecker-Williams approximation

Weizsaecker-Williams approximation is used to describe processes of electroproduction 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.14). 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 $\alpha$ 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 [27] for the further explanations. In the case of CompHEP the Weizsaecker-Williams photon spectrum is available for charged leptons only.

## C.5 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 [28]. In CompHEP we realize the similar expression by Jadach, Skrzypek, and Ward [29]:

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

where

| | | | |
|---|---|---|---|
| $\alpha$ | = | $1/137.0359895$ | is the fine structure constant; |
| $\beta$ | = | $\alpha(2\ln(SCALE/m)-1)/pi$ | |
| $m$ | = | 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 [30]

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

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 $\gamma$ is the incomplete gamma function.

Function $F(x)$ depends on two parameters, $N_{cl}$ and $\Upsilon$, 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 CompHEP version because it contains a $\delta$-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.

# D Monte Carlo phase space integration

## D.1 Adaptive Monte Carlo integration package *Vegas*

This section contains a short description of the adaptive Monte Carlo program VEGAS. See for details [31, 32].

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 $\sigma_N$ 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*.

### D.1.1 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 $f(x)$ [31, 32]. 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.

### D.1.2 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.

## D.2 Parameterization of multi-particle phase space

### D.2.1 Parameterization via decay scheme

The element of phase space volume for a $n$-particle state is equal to [36]

$$d\Gamma_n(q) = (2\pi)^4\delta^4(q - p_1 - p_2 - p_3 - ... - p_n)\prod_{i=1}^{n}\frac{\delta(p_i^2 - m_i^2)}{(2\pi)^3}d^4p_i\ . \qquad (26)$$

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 (26)

$$d\Gamma(q, S) = (2\pi)^4\delta^4(q - \sum_{i\in S} p_i)\prod_{i\in S}\frac{\delta(p_i^2 - m_i^2)}{(2\pi)^3}d^4p_i\ . \qquad (27)$$

Let $S_1$ and $S_2$ be two disjoint particle subsets, then

$$d\Gamma(q, S_1 \cup S_2) =$$

$$\int ds_1 ds_2 \Big( (2\pi)^4 \delta^4 (q - q_1 - q_2) \frac{\delta(q_1^2 - s_1)}{(2\pi)^3} d^4 q_1 \frac{\delta(q_2^2 - s_2)}{(2\pi)^3} d^4 q_2 \Big)$$
$$\times \frac{d\Gamma(q, S_1)}{2\pi} \times \frac{d\Gamma(q, S_2)}{2\pi} \,. \tag{28}$$

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 $\Omega$ of motion of the first decaying particle in the rest frame of initial state [36].

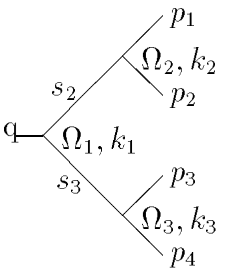
$$\frac{d\Gamma(q, [1,2])}{2\pi} = \frac{k d\Omega}{4(2\pi)^3 \sqrt{q^2}} \,, \tag{29}$$

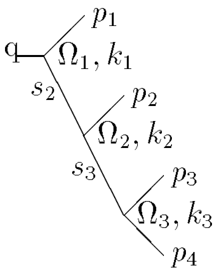
where $k$ is the absolute value of three-dimensional momentum of outgoing particles in the rest frame. Thus, applying recursively (28) and (29) to (26) we obtain an explicit expression for the phase space volume in terms of the squared masses $s_j$ of virtual clusters and the two-dimensional spherical angles $\Omega_j$, where $j$ is an ordinal number of decay:

$$d\Gamma_n(q) = \frac{k_1 d^2\Omega_1}{4(2\pi)^2 \sqrt{q^2}} \prod_{j=2}^{n-1} \frac{k_j d^2 \Omega_j}{4(2\pi)^3 \sqrt{s_j}} \cdot \prod_{j=2}^{n-1} ds_j \tag{30}$$

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

The expression (30) 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 CompHEP project such decay scheme is defined by the user via the 'Kinematics' menu (see Section 3.3.7).

### D.2.2 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

$$d^2\Omega_j = d\cos\Theta_j d\Phi_j \tag{31}$$

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 (30) 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 3.3.8) depending on a squared sum of momenta. Variables $s_j$ in (30) are also equal to squared sums of momenta. So, the parameterization (30) 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 [19, 13]. The polar angle $\Theta_j$ 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 [19, 13].

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 [19, 13].

In CompHEP the ordering is arranged automatically, so that all sub-decays 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.

### D.2.3 Smoothing

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

$$\int_a^b F(x)dx \quad , \tag{32}$$

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

$$g(x) = \int_a^x f(x')dx' \ . \tag{33}$$

Now we may represent the integral (32) as

$$\int_a^b F(x)dx = \int_0^{g(b)} dy \frac{F(g^{-1}(y))}{f(g^{-1}(y))}, \tag{34}$$

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 CompHEP 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 (32), we could perform two ones:

$$\int F(x)dx = \int \frac{F(x)f_1(x)}{f_1(x)+f_2(x)}dx + \int \frac{F(x)f_2(x)}{f_1(x)+f_2(x)}dx \ , \tag{35}$$

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 [46] to separate peaks which came from various diagrams. In that paper there was also proposed to use the expression (35) where $f_i(x)$ is replaced by $\alpha_i f_i(x)$ with a subsequent search for optimal coefficients $\alpha_i$. CompHEP passes on this weight optimization to *Vegas*, combining two integrals in one *Vegas* hypercube.

As was mentioned above, CompHEP 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 CompHEP uses the branching method again, but now for the whole two-dimension sphere integration. In other words, we use the branching equation (35) where $x$ is the two dimensional sphere angle [19, 13].

# E Lagrangian of the Standard Model

## E.1 Gauge theories

**Group.** Group is defined by its structure constants $f^{\alpha}_{\beta\gamma}$ which are antisymmetric ($f^{\alpha}_{\beta\gamma} = -f^{\alpha}_{\gamma\beta}$) 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 [42]

$$L = -\frac{1}{4g^2}F^\alpha{}_{\mu\nu}F_\alpha{}^{\mu\nu} + L_m(\nabla_\mu\psi, \psi) \tag{36}$$

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 (36) 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 observables.

The general form of the gauge fixing term is

$$L_{FG}(x) = -\frac{1}{2}\sum_\alpha(\Phi^\alpha(x))^2 \ . \tag{37}$$

The corresponding Faddeev-Popov term is

$$L_{FP} = -\bar{c}_\alpha(x)(D(c)[\Phi^\alpha])(x) \ , \tag{38}$$

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 (38) 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

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

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

$$-\frac{1}{2g^2}\partial_\mu A_\nu \partial^\mu A^\nu \, . \tag{39}$$

The corresponding Faddeev-Popov Lagrangian is

$$L_{FP} = -\bar{c}_\alpha(x)(\Box c^\alpha(x) + \partial^\mu(f^\alpha_{\beta\gamma}A^\beta_\mu(x)c^\gamma(x)) + \ldots) \, .$$

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 (36) and appears in the interaction terms.

## E.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 (36), (37), (38).

$$L_{phys} = -\frac{1}{4}F^{\alpha}{}_{\mu\nu}(x)F_{\alpha}{}^{\mu\nu}(x) + \sum_k \frac{i}{2}(\bar{q}_k(x)\gamma^{\mu}\nabla_{\mu}q_k(x) - \nabla_{\mu}\bar{q}_k(x)\gamma^{\mu}q_k(x)) \; . \tag{40}$$

where

$$\nabla_{\mu}q(x) = \partial_{\mu}q(x) - i\,g\,G^{\alpha}_{\mu}(x)\hat{t}_{\alpha}q(x) \; ;$$

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

$$L_{GF} = -\frac{1}{2}\sum_{\alpha}(\partial^{\mu}G^{\alpha}_{\mu})^2 \; ; \tag{41}$$

$$L_{FP} = -\bar{c}_{\alpha}(x)(\Box c^{\alpha}(x) + g\,\partial^{\mu}(f^{\alpha}_{\beta\gamma}G^{\beta}_{\mu}(x)c^{\gamma}(x))) \; , \tag{42}$$

$c^{\alpha}(x)$ are Faddeev-Popov ghosts, $g$ is coupling constant, $f^{\alpha}_{\beta\gamma}$ are the $SU(3)$ structure constants, $t_{\alpha}$ are the generators in the fundamental representation[10].

## E.3 Lagrangian of electroweak interactions

### E.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 (36):

$$L_{GF} = -\frac{1}{4}FW^{\alpha}{}_{\mu\nu}(x)FW_{\alpha}{}^{\mu\nu}(x) - \frac{1}{4}FB_{\mu\nu}(x)FB^{\mu\nu}(x) \; , \tag{43}$$

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.

[10] $t_{\alpha}$ are equal to $\lambda_{\alpha}/2$ where $\lambda_{\alpha}$ are the Gell-Mann matrices.

Infinitesimal local gauge transformations are defined as follows

$$\begin{aligned}(\hat{D}(w,b)W_\mu)^\alpha(x) &= g_2\epsilon^{\alpha\beta\gamma}W_\mu^\beta(x)w^\gamma(x)+\partial_\mu w^\alpha(x)\ ;\\ D(w,b)B_\mu(x) &= \partial_\mu b(x)\ .\end{aligned} \tag{44}$$

Let us express $W_\mu^1$ and $W_\mu^2$ in terms of a mutually conjugated couple $W_\mu^+$ and $W_\mu^-$

$$\begin{aligned}W_\mu^1 &= (W_\mu^- + W_\mu^+)/\sqrt{2}\ ;\\ W_\mu^2 &= (W_\mu^- - W_\mu^+)/(i\sqrt{2})\ .\end{aligned}$$

Thus, the Lagrangian of self-interaction for the $SU(2)$ gauge fields in term of $W^+$ and $W^-$ has the form:

$$\begin{aligned}&i\,g_2\Big(\partial_\mu W_\nu^3(W_\mu^+W_\nu^- - W_\mu^-W_\nu^+)+\partial_\mu W_\nu^+(W_\mu^-W_\nu^3 - W_\mu^3W_\nu^-)\\ &\qquad\qquad -\partial_\mu W_\nu^-(W_\mu^+W_\nu^3 - W_\mu^3W_\nu^+)\Big)\\ &+g_2^2\left(W_\mu^3W_\nu^+(W_\mu^-W_\nu^3 - W_\mu^3W_\nu^-)+\frac{1}{2}W_\mu^+W_\nu^-(W_\mu^+W_\nu^- - W_\mu^-W_\nu^+)\right)\ .\end{aligned} \tag{45}$$

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 $\sigma$-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 $\phi_0$ is a real constant.

As a result of spontaneous symmetry breaking the $W^\alpha_\mu$ and $B_\mu$ fields do not correspond to physical particles. Physical particles in this model are the photon ($A_\mu$), W-bosons ($W^+_\mu$, $W^-_\mu$) and Z-boson ($Z_\mu$). The photon field $A_\mu$ is a combination of gauge fields responsible for the local gauge transformations which save the Higgs vacuum $\Phi_\Omega$:

$$A_\mu = B_\mu \cos\Theta_w + W^3_\mu \sin\Theta_w \ ,$$

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 $W^+_\mu$, $W^-_\mu$, $A_\mu$, and $Z_\mu$. Then for a matter doublet with a hypercharge $Y$ the gauge transformation is given by the following expression:

$$\begin{aligned} &\hat{D}(w^+, w^-, a, z)\Psi = \\ &\quad \frac{i\,g_2}{2}\left[\sqrt{2}\begin{pmatrix} 0 & w^+ \\ w^- & 0 \end{pmatrix} + \sin\Theta_w\, a \begin{pmatrix} 1+Y & 0 \\ 0 & Y-1 \end{pmatrix}\right. \\ &+ \quad \left.\cos\Theta_w\, z \begin{pmatrix} 1 - Y\tan^2\Theta_w & 0 \\ 0 & -1 - Y\tan^2\Theta_w \end{pmatrix}\right]\Psi \ . \end{aligned} \tag{46}$$

### E.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 $\lambda$ 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 $\Phi$ in the following form:

$$\Phi = \begin{pmatrix} i\,W^+_f \\ (\phi_0 + H - i\,Z_f)/\sqrt{2} \end{pmatrix} \ . \tag{47}$$

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 assoshated with the Higgs patricle.

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:

$$-\;\frac{\lambda}{2}\left((W_f^+W_f^- + (H^2+Z_f^2)/2)^2 + 2\phi_0 H(W_f^+W_f^- + (H^2+Z_f^2)/2)\right)\;. \quad (48)$$

Usually $\lambda$ is expressed via the mass of $H$-boson $M_H$:

$$\lambda = \left(\frac{M_H}{\phi_0}\right)^2 \;.$$

Let us remind that

$$\nabla_\mu \Phi = \partial_\mu \Phi - \hat{D}(W+_\mu, W-_\mu, A_\mu, Z_\mu)\Phi \quad ,$$

where due to (46)

$$\hat{D}(W+_\mu, W-_\mu, A_\mu, Z_\mu)\Phi =$$
$$\frac{i\,g_2}{2}\begin{pmatrix} W_\mu^+(\phi_0 + H - i\,Z_f) + i\,(2\sin\Theta_w A_\mu + \cos\Theta_w(1-\tan^2\Theta_w)Z_\mu)\,W_f^+ \\ (2iW_\mu^- W_f^+ - Z_\mu(\phi_0 + H - i\,Z_f)/\cos\Theta_w)/\sqrt{2} \end{pmatrix}\;. \quad (49)$$

The term $(\hat{D}_\mu\Phi)^+(\hat{D}^\mu\Phi)$ gives us the mass terms for the $W$ and $Z$-bosons:

$$\left(\frac{g_2\phi_0}{2}\right)^2 W_\mu^+ W^{-\mu} + \frac{1}{2}\left(\frac{g_2\phi_0}{2\cos\Theta_w}\right)^2 Z_\mu Z^\mu\;, \quad (50)$$

and the following terms describing the interaction of a couple of vector bosons with Higgs and Goldstones:

$$\frac{g_2^2}{4}(\qquad W_\mu^+ W^{-\mu}(H^2 + Z_f^2 + 2\phi_0 H + 2W_f^+W_f^-)$$

$$
\begin{aligned}
&+4A_\mu A^\mu \sin^2\Theta_w W_f^+ W_f^- \\
&+\frac{Z_\mu Z^\mu}{\cos^2\Theta_w}((H^2+Z_f^2)/2+\phi_0 H+(1-2\sin^2\Theta_w)^2 W_f^+ W_f^-) \\
&+4A_\mu Z^\mu \tan\Theta_w(1-2\sin^2\Theta_w)W_f^+ W_f^- \\
&-2i\tan\Theta_w W_\mu^+(\cos\Theta_w A^\mu-\sin\Theta_w Z^\mu)(\phi_0+H-i\,Z_f)W_f^- \\
&+2i\tan\Theta_w W_\mu^-(\cos\Theta_w A^\mu-\sin\Theta_w Z^\mu)(\phi_0+H+i\,Z_f)W_f^+ \\
)\,.
\end{aligned}
\tag{51}
$$

The constant $\phi_0$ 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:

$$-\,g_2\phi_0\left(\frac{1}{2}(W_\mu^+\partial^\mu W_f^- + W_\mu^-\partial^\mu W_f^+) + \frac{1}{\cos\Theta_w}Z_\mu\partial^\mu Z_f\right) \tag{52}$$

and the following terms of interaction:

$$
\begin{aligned}
\frac{g_2}{2}(\quad &-H(W_\mu^-\partial^\mu W_f^+ + W_\mu^+\partial^\mu W_f^-)+(\partial^\mu H)(W_\mu^- W_f^+ + W_\mu^+ W_f^-) \\
&-i(Z_f(W_\mu^-\partial^\mu W_f^+ - W_\mu^+\partial^\mu W_f^-)+(\partial^\mu Z_f)(W_\mu^+ W_f^- - W_\mu^- W_f^+)) \\
&+i(2\sin\Theta_w A_\mu+\cos\Theta_w(1-\tan^2\Theta_w)Z_\mu)(W_f^-\partial^\mu W_f^+ - W_f^+\partial^\mu W_f^-) \\
&+(\partial^\mu H)Z_\mu Z_f/\cos\Theta_w-(\partial^\mu Z_f)Z_\mu H/cos\Theta_w \\
)\ .
\end{aligned}
\tag{53}
$$

The off-diagonal quadratic terms are canceled by the gauge fixing terms. See below.

### E.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 (39). 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 (52) up to complete divergency terms.

According to the general rule (38) 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 (44)

$$
\begin{aligned}
\hat D W_\mu^+ &= i\, g_2\, (W_\mu^+(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^+(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu)) + \partial_\mu W_c^+\ ;\\
\hat D A_\mu &= -i\, g_2 \sin\Theta_w (W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu A_c\ ;\\
\hat D Z_\mu &= -i\, g_2 \cos\Theta_w (W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu Z_c\ ,
\end{aligned}
$$

and according to (49)

$$
\begin{aligned}
D W_f^+ &= W_c^+ M_W + \frac{g_2}{2}(W_c^+(H - i\, Z_f) + i\,(2\sin\Theta_w A_c + \cos\Theta_w (1-\tan^2\Theta_w) Z_c)\, W_f^+)\ ;\\
D Z_f &= M_Z Z_c + \frac{g_2}{2}(\frac{Z_c H}{\cos\Theta_w} - i(W_c^- W_f^+ - W_c^+ W_f^-))\ .
\end{aligned}
$$

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:

$$
\begin{aligned}
-\, g_2( \quad & i\,\sin\Theta_w (\partial^\mu A_{\bar c})(W_\mu^+ W_c^- - W_\mu^- W_c^+)\\
&+ i\,\cos\Theta_w (\partial^\mu Z_{\bar c})(W_\mu^+ W_c^- - W_\mu^- W_c^+)\\
&+ \frac{M_Z Z_{\bar c}}{2}\left(\frac{Z_c H}{\cos\Theta_w} - i(W_c^- W_f^+ - W_c^+ W_f^-)\right)\\
&- i(\partial^\mu W_{\bar c}^-)(W_\mu^+(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^+(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu))
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{M_W W_{\bar{c}}^-}{2}(W_c^+(H - i\,Z_f) + i\,(2\sin\Theta_w A_c + \cos\Theta_w(1-\tan^2\Theta_w)Z_c)\,W_f^+))) \\
&+i(\partial^\mu W_{\bar{c}}^+)(W_\mu^-(\sin\Theta_w A_c + \cos\Theta_w Z_c) - W_c^-(\sin\Theta_w A_\mu + \cos\Theta_w Z_\mu)) \\
&+\frac{M_W W_{\bar{c}}^+}{2}(W_c^-(H + i\,Z_f) - i\,(2\sin\Theta_w A_c + \cos\Theta_w(1-\tan^2\Theta_w)Z_c)\,W_f^-))) \\
&) \,.
\end{aligned}
\tag{54}
$$

### E.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 (38) takes the form:

$$
\begin{aligned}
-A_{\bar{c}}\partial^\mu(-i\,g_2\sin\Theta_w(W_\mu^+ W_c^- - W_\mu^- W_c^+) + \partial_\mu A_c) - Z_{\bar{c}} M_Z\,(M_Z + \frac{g_2 H}{2\cos\Theta_w})Z_c \\
-W_{\bar{c}}^- M_W(M_W + \frac{g_2}{2}H)W_c^+ - W_{\bar{c}}^+ M_W(M_W + \frac{g_2}{2}H)W_c^-\,.
\end{aligned}
$$

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.

### E.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 (45), (48), (51), (53), and (54), where

$$
\begin{aligned}
W_\mu^3 &= \sin\Theta_w A_\mu + \cos\Theta_w Z_\mu\;; \\
\lambda &= \left(\frac{g_2 M_H}{2M_w}\right)^2\,.
\end{aligned}
$$

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.

### E.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 $B_\mu$ 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 (46) 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:

$$\begin{aligned} & \frac{g_2}{2\sqrt{2}}\left(W_\mu^-\bar\Psi_2\gamma^\mu(1-\gamma^5)\Psi_1 + W_\mu^+\bar\Psi_1\gamma^\mu(1-\gamma^5)\Psi_2\right) \\ + & \frac{g_1}{4\sin\Theta_w} Z_\mu \Big( \bar\Psi_1\gamma^\mu(1-\gamma^5 - 2(Y+1)\sin^2\Theta_w)\Psi_1 \\ & \qquad - \bar\Psi_2\gamma^\mu(1-\gamma^5 + 2(Y-1)\sin^2\Theta_w)\Psi_2 \Big) \\ + & \frac{g_1}{2}\cos\Theta_w A_\mu \left((Y+1)\bar\Psi_1\gamma^\mu\Psi_1 + (Y-1)\bar\Psi_2\gamma^\mu\Psi_2\right)\ . \end{aligned} \tag{55}$$

### E.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

$$-\frac{m_1\sqrt{2}}{\phi_0}\left(\bar\Psi_i^L\epsilon^{ij}\stackrel{*}{\Phi}{}^j\ \Psi_1^R - \bar\Psi_1^R\Phi^i\epsilon^{ij}\Psi_j^L\right) - \frac{m_2\sqrt{2}}{\phi_0}\left(\bar\Psi_i^L\Phi^i\Psi_2^R + \bar\Psi_2^R\stackrel{*}{\Phi}{}^i\ \Psi_i^L\right)$$

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:

$$
\begin{aligned}
&- \frac{m_1 g_2}{2M_W}\left(\frac{i}{\sqrt{2}}(W_f^- \bar{\Psi}_2(1+\gamma^5)\Psi_1 - W_f^+ \bar{\Psi}_1(1-\gamma^5)\Psi_2) + H\bar{\Psi}_1\Psi_1 + iZ_f\bar{\Psi}_1\gamma^5\Psi_1\right) \\
&- \frac{m_2 g_2}{2M_W}\left(\frac{i}{\sqrt{2}}(W_f^+ \bar{\Psi}_1(1+\gamma^5)\Psi_2 - W_f^- \bar{\Psi}_2(1-\gamma^5)\Psi_1) + H\bar{\Psi}_2\Psi_2 - iZ_f\bar{\Psi}_2\gamma^5\Psi_2\right) .
\end{aligned}
\tag{56}
$$

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 (55) and (56), whereas interactions with $W^\pm$ and $W_f^\pm$ contain elements of the mixing matrix.

### E.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] J. Küblbeck, M. Böhm, A. Denner, Comp. Phys. Comm. 60 (1990) 165
R. Mertig, M. Böhm, A. Denner, Comp. Phys. Comm. 60 (1990) 345

[2] 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

[3] H. Murayama, I. Watanabe, K. Hagiwara, *HELAS: HELicity Amplitude Subroutine for Feynman Diagram Evaluation*, KEK Report 91-11, 1992

[4] T. Stelzer, W. F. Long, Comp. Phys. Comm. 81 (1998) 357

[5] R. Harlander, M. Steinhauser, Automatic computation of Feynman diagrams, hep-ph/9812357.

[6] P.A. Baikov et al., *Physical results by means of CompHEP*, in: Proc. of X Workshop on High Energy Physics and Quantum Field Theory (QFTHEP-95), ed. by B.Levtchenko, V.Savrin, Moscow, 1996, p.101 (hep-ph/9701412)

[7] E.E. Boos, M.N. Dubinin, V.A. Ilyin, A.E. Pukhov, V.I. Savrin, *CompHEP: specialized package for automatic calculations of elementary particle decays and collisions,* SNUTP report 94-116, Seoul, 1994 (hep-ph/9503280)

[8] E. Boos, M. Dubinin, V. Edneral, V. Ilyin, A. Kryukov, A. Pukhov, A. Rodionov, V. Savrin, D. Slavnov, A. Taranov, *CompHEP - computer system for calculation of particles collision characteristics at high energies*, preprint INP MSU 89-63/140, Moscow, 1989

[9] E. Boos, M. Dubinin, V. Edneral, V. Ilyin, A. Kryukov, A. Pukhov, V. Savrin, S. Shichanin, A. Taranov, in: "New Computing Techniques in Physics Research", Proc. of Int. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AI'90), ed. by D. Perret-Gallix and W. Wojcik, Editions du CNRS, 1990, p. 573

[10] E. Boos, M. Dubinin, V. Edneral, V. Ilyin, A. Kryukov, A. Pukhov, S. Shichanin, in: "New Computing Techniques in Physics Research II", Proc. of Int. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AI'91), ed. ed.by D. Perret-Gallix, World Scientific, Singapore, 1992, p. 665.

[11] A. Pukhov, in: "New Computing Techniques in Physics Research III", Proc. of Int. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AI'92), ed. ed. by K.-H. Becks and D. Perret-Gallix, World Scientific, Singapore, 1993, p. 473

[12] V.A. Ilyin et al., in "New Computing Techniques in Physics Research IV", Proc. of Int. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AI'94), ed. by B. Denby and D. Perret-Gallix, World Scientific, Singapore, 1995, p.295

[13] D. Kovalenko, A. Pukhov, Nucl. Instr. and Meth. A389 (1997) 299

[14] V. Edneral, A. Semenov, *The Translation of Computer Programs from Environment Turbo PASCAL/DOS to C/UNIX*, preprint INP MSU 93-26/318, Moscow, 1993 (in Russian)

[15] V. Edneral, A. Semenov *The Emulation of Graphic Library of BORLAND Turbo C/PASCAL at the Environment of OS UNIX*, preprint INP MSU 93-27/319, 1993 (in Russian)

[16] 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

[17] A. Belyaev, A. Gladyshev, A. Semenov, preprint IFT–P–093–97 (hep-ph/9712303).

[18] M. Dubinin, A. Semenov, hep-ph/9812246

[19] 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)

[20] E.E. Boos, M.N. Dubinin, V.F. Edneral, V.A. Ilyin, A.P. Kryukov, A.E. Pukhov, S.A. Shichanin, *Applications of CompHEP system to particle processes calculation*, NPI MSU report 91-9-213, Moscow, 1991

[21] E.E. Boos, M.N. Dubinin, V.A. Ilyin, A.E. Pukhov, S.A. Shichanin, T. Kaneko, S. Kawabata, Y. Kurihara, Y. Shimizu, H. Tanaka, *Automatic calculation in high-energy physics by GRACE/CHANEL and CompHEP*, Int. J. Mod. Phys. C5 (1994) 615

[22] A. Belyaev, E. Boos, L. Dudko, A. Pukhov, *W + 2 jets production at Tevatron: VECBOS and CompHEP comparison*, D0 note 2784, 1995 (hep-ph/9511306)

[23] *Event generators for WW physics* (conveners D. Bardin, R. Kleiss), in "Physics at LEP2", ed.by G. Altarelli, T. Sjostrand, F. Zwirner, CERN report 96-01, vol.II, Geneva, 1996, p.3

*Standard Model Processes* (conveners F. Boudjema, B. Mele), in "Physics at LEP2", ed.by G. Altarelli, T. Sjostrand, F. Zwirner, CERN report 96-01, vol.I, Geneva, 1996, p.207

[24] H.L. Lai, J. Huston, S. Kuhlmann, F. Olness, J. Owens, D. Soper W.K. Tung, H. Weerts, Phys. Rev. D55 (1997) 1280

[25] A.D. Martin, R.G. Roberts and W.J. Stirling, Phys. Lett. B354 (1995) 155

[26] I.F. Ginzburg, G.L. Kotkin, V.G. Serbo, V.I. Telnov, Nucl. Instr. and Meth. 205 (183) 47

[27] V.M. Budnev et al., Phys. Rep. 15C (1975) 181

S. Frixione et al., Phys. Lett. B319 (1993) 339

[28] E. Kuraev, V. Fadin, Sov. J. Nucl. Phys. 41 (1985) 466

[29] S. Jadach, B. Ward Comp. Phys. Commun. 56 (1990) 351
S. Jadach, M. Skrzypek, Z.Phys. C 49,(1991) 577

[30] P. Chen, Phys. Rev. D46 (1992) 1186

[31] G.P. Lepage, Journal of Comput. Phys. 27 (1978) 192

[32] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in Fortran*, Cambridge University Press, 1992

[33] A. Hearn, J. Fitch, *REDUCE User's Manual*, v.3.6, RAND Publication CP78 (Rev. 7/95)

[34] S. Wolfram, *MATHEMATICA*, Addison-Wesley, 1991

[35] J.A.M. Vermaseren, *Symbolic manipulation with FORM, version 2: Tutorial and Reference Manual*, CAN, 1992

[36] Review of Particles Properties, Phys.Rev. D54, 1996

[37] U. Baur, J. Vermaseren, D. Zeppenfeld, Nucl. Phys. B375 (1992) 2

[38] Y. Kurihara, D. Perret-Gallix, Y. Shimizu, Phys. Lett. B349 (1995) 367

[39] E. Boos, M. Dubinin, L. Dudko, Int. J. Mod. Phys. A11 (1996) 5015

[40] V. Ilyin, D. Kovalenko, A. Pukhov, Int. J. Mod. Phys. C7 (1996) 761
D. Kovalenko, A. Pukhov, Nucl. Instr. and Meth. A389 (1997) 299

[41] T. Sjostrand, Comp. Phys. Comm. 82 (1994) 74

[42] 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

[43] T.-P. Cheng and L.-F. Li, *Gauge theory of elementary particle physics*, Oxford Univ. Press, 1984

[44] L. Baulieu, Phys. Rep. 129 (1985) 1

[45] J.A.M. Vermaseren, Comp. Phys. Comm. 83 (1994) 45

[46] F.A. Berends, R. Pittau, R. Kleiss, Comp. Phys. Comm. 85 (1995) 437

[47] E. Byckling, K. Kajantie, Particle Kinematics, John Wiley & Sons, 1973

# Pictures and figures

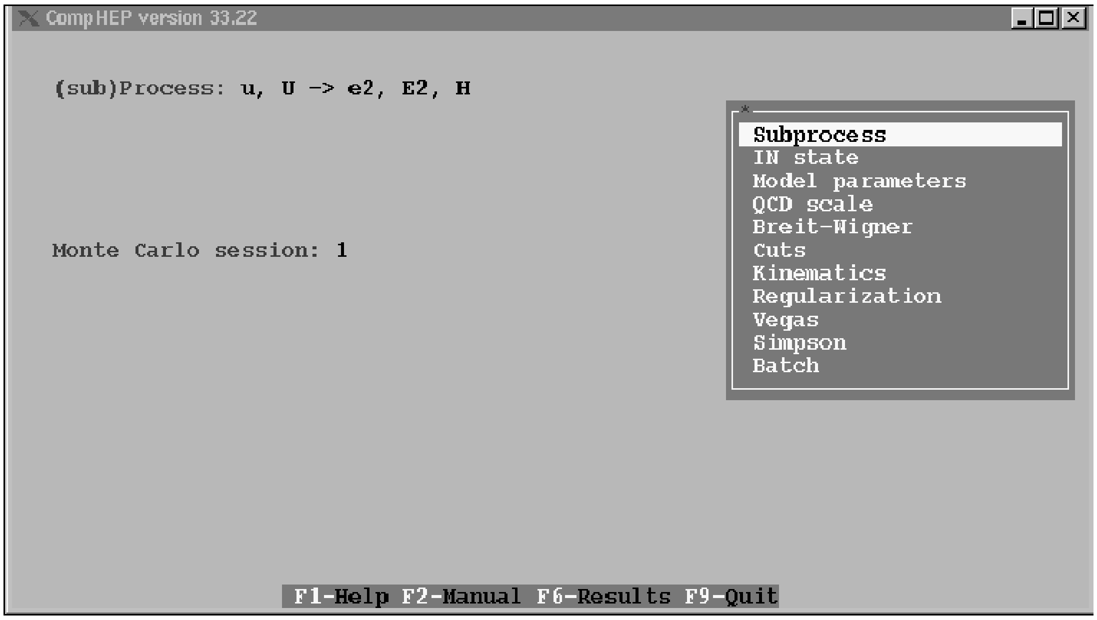
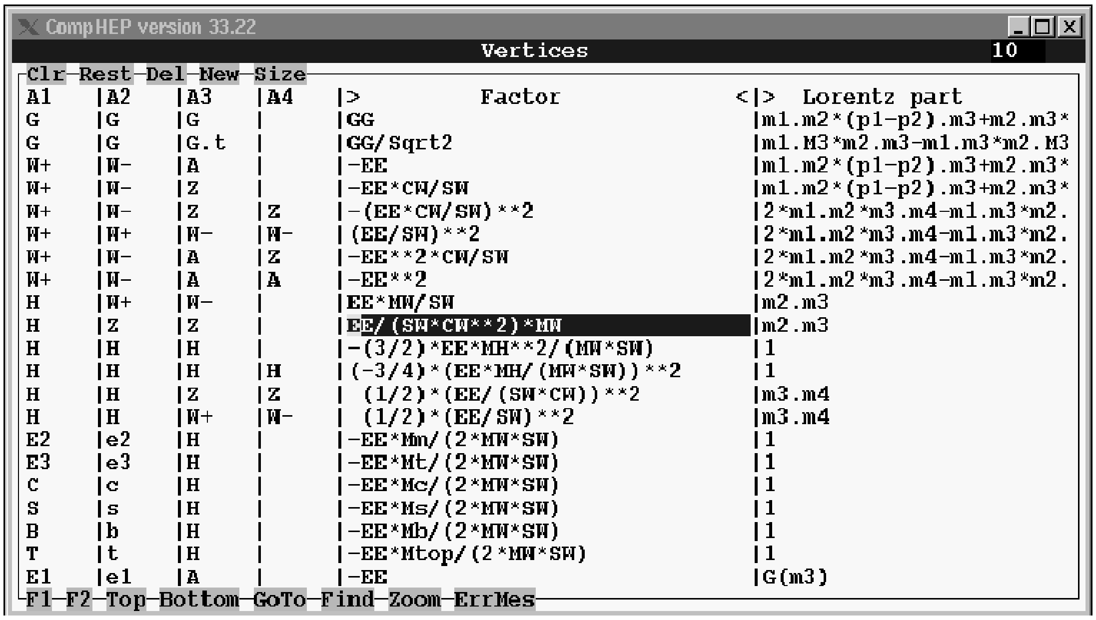



Figure 1: Example of the CompHEP menu

```
*****************************************************************
*                            M A I N  M E N U                   *
*    CompHEP numerical module    u  ,D   -> b  ,B  ,W+          *
*****************************************************************
*     x: Exit   hN: Help (N-menu position)  m: MAIN menu        *
*                                                               *
*    1: Subprocess                      2: IN state             *
*    3: Model parameters                4: QCD SCALE            *
*    5: Breit-Wigner                    6: Cuts                 *
*    7: Kinematics                      8: Regularization       *
*    9: Vegas                          10: Simpson              *
*   11: Batch                          12: View result          *
 ****************************************************************
   Type number of menu position and press ENTER:_
```

Figure 2: Example of the *Fortran* menu



CompHEP version 33.22

Vertices 10

Clr Rest Del New Size

| A1 | A2 | A3 | A4 | > Factor | <\|> Lorentz part |
|---|---|---|---|---|---|
| G | G | G | | GG | m1.m2*(p1-p2).m3+m2.m3* |
| G | G | G.t | | GG/Sqrt2 | m1.M3*m2.m3-m1.m3*m2.M3 |
| W+ | W- | A | | -EE | m1.m2*(p1-p2).m3+m2.m3* |
| W+ | W- | Z | | -EE*CW/SW | m1.m2*(p1-p2).m3+m2.m3* |
| W+ | W- | Z | Z | -(EE*CW/SW)**2 | 2*m1.m2*m3.m4-m1.m3*m2. |
| W+ | W+ | W- | W- | (EE/SW)**2 | 2*m1.m2*m3.m4-m1.m3*m2. |
| W+ | W- | A | Z | -EE**2*CW/SW | 2*m1.m2*m3.m4-m1.m3*m2. |
| W+ | W- | A | A | -EE**2 | 2*m1.m2*m3.m4-m1.m3*m2. |
| H | W+ | W- | | EE*MW/SW | m2.m3 |
| H | Z | Z | | EE/(SW*CW**2)*MW | m2.m3 |
| H | H | H | | -(3/2)*EE*MH**2/(MW*SW) | 1 |
| H | H | H | H | (-3/4)*(EE*MH/(MW*SW))**2 | 1 |
| H | H | Z | Z | (1/2)*(EE/(SW*CW))**2 | m3.m4 |
| H | H | W+ | W- | (1/2)*(EE/SW)**2 | m3.m4 |
| E2 | e2 | H | | -EE*Mm/(2*MW*SW) | 1 |
| E3 | e3 | H | | -EE*Mt/(2*MW*SW) | 1 |
| C | c | H | | -EE*Mc/(2*MW*SW) | 1 |
| S | s | H | | -EE*Ms/(2*MW*SW) | 1 |
| B | b | H | | -EE*Mb/(2*MW*SW) | 1 |
| T | t | H | | -EE*Mtop/(2*MW*SW) | 1 |
| E1 | e1 | A | | -EE | G(m3) |

F1 F2 Top Bottom GoTo Find Zoom ErrMes

Figure 3: Example of the CompHEP tables

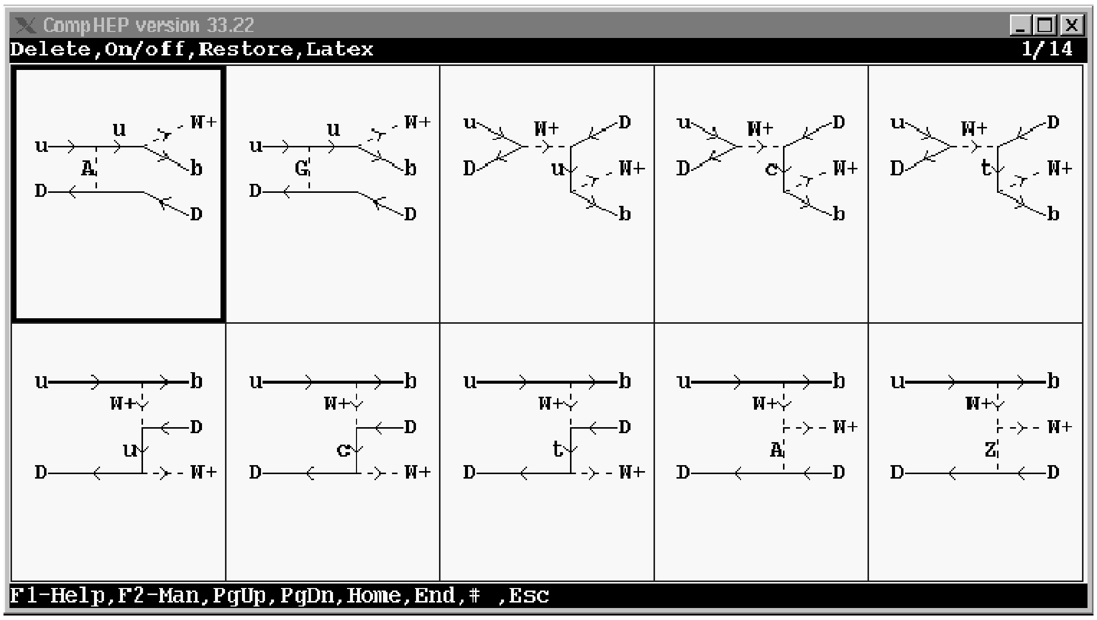



Figure 4: Example of the CompHEP diagram images

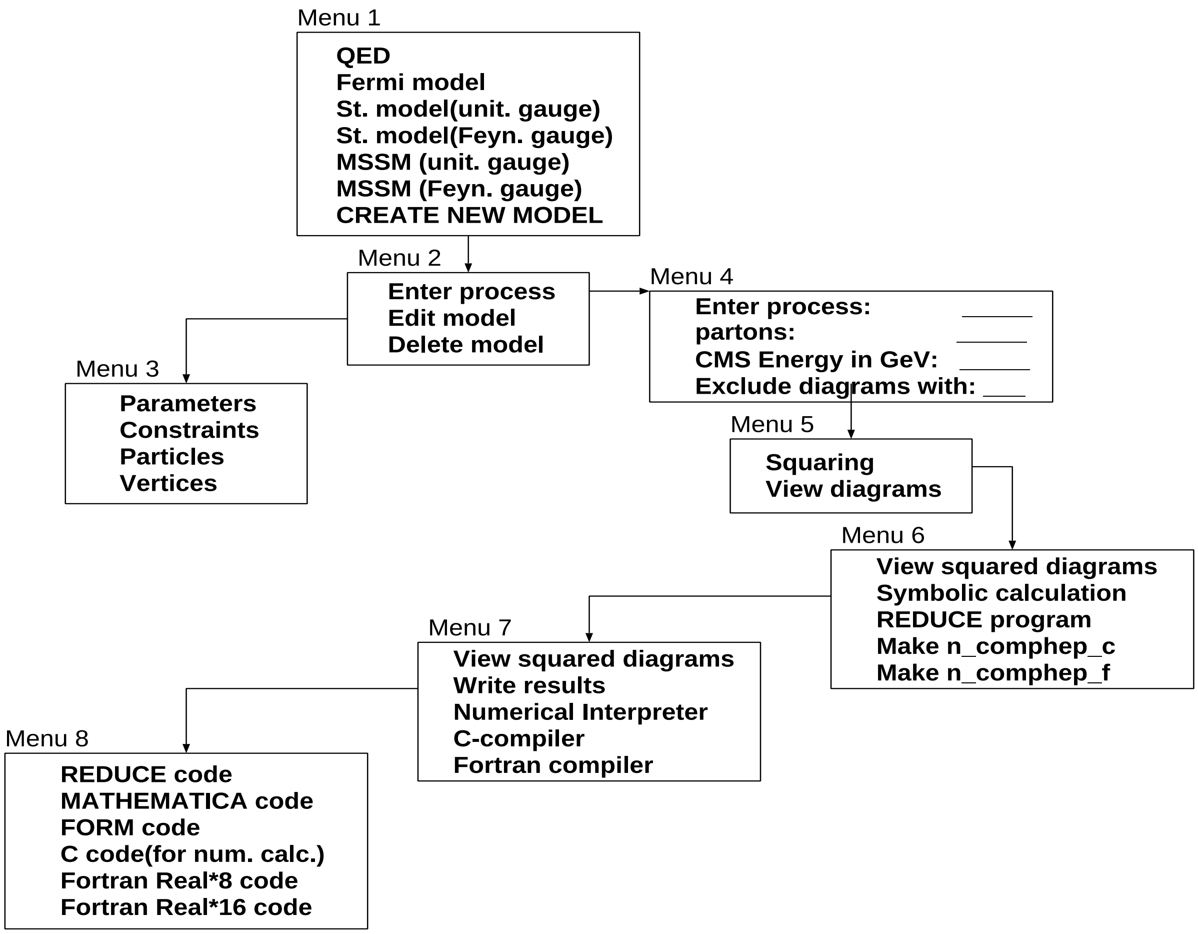



Figure 5: Menu scheme for the symbolic session

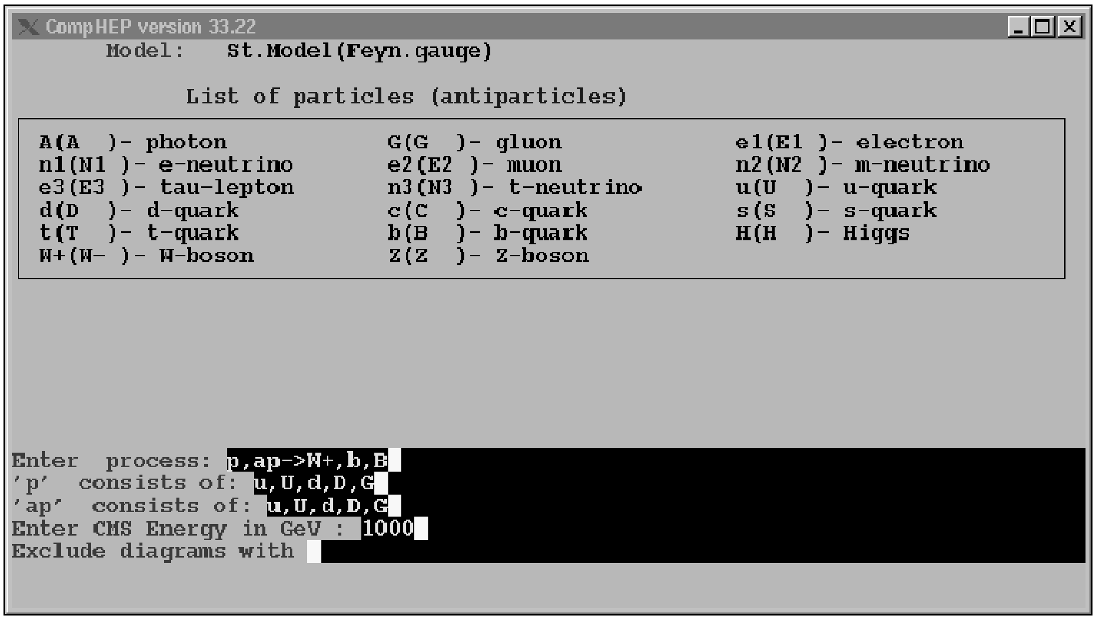



Figure 6: Example of the process input

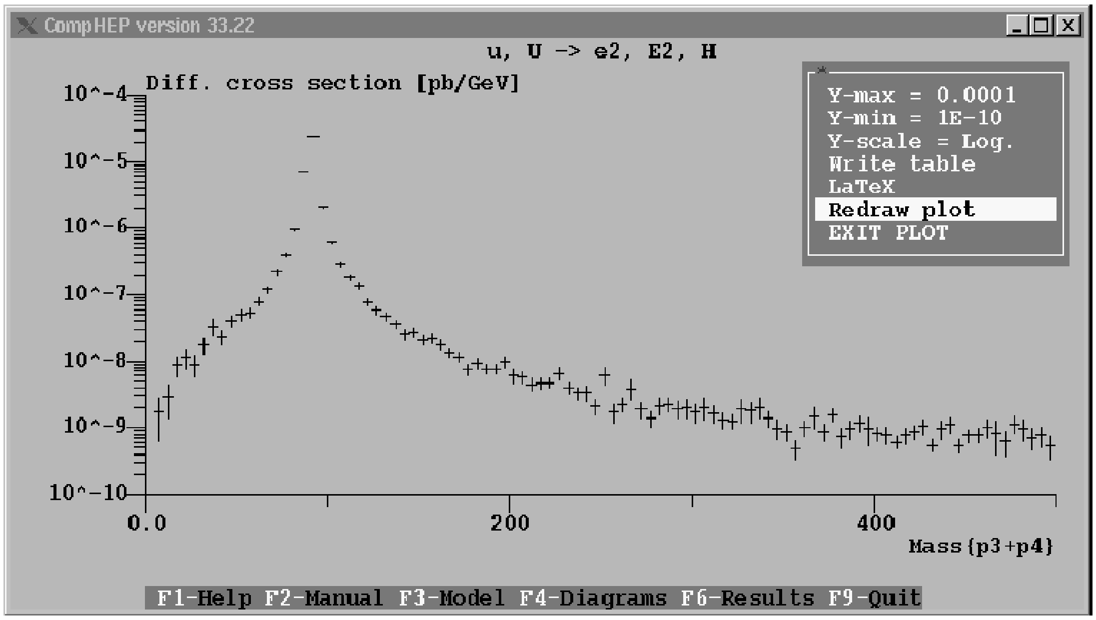



Figure 7: Example of the plot image

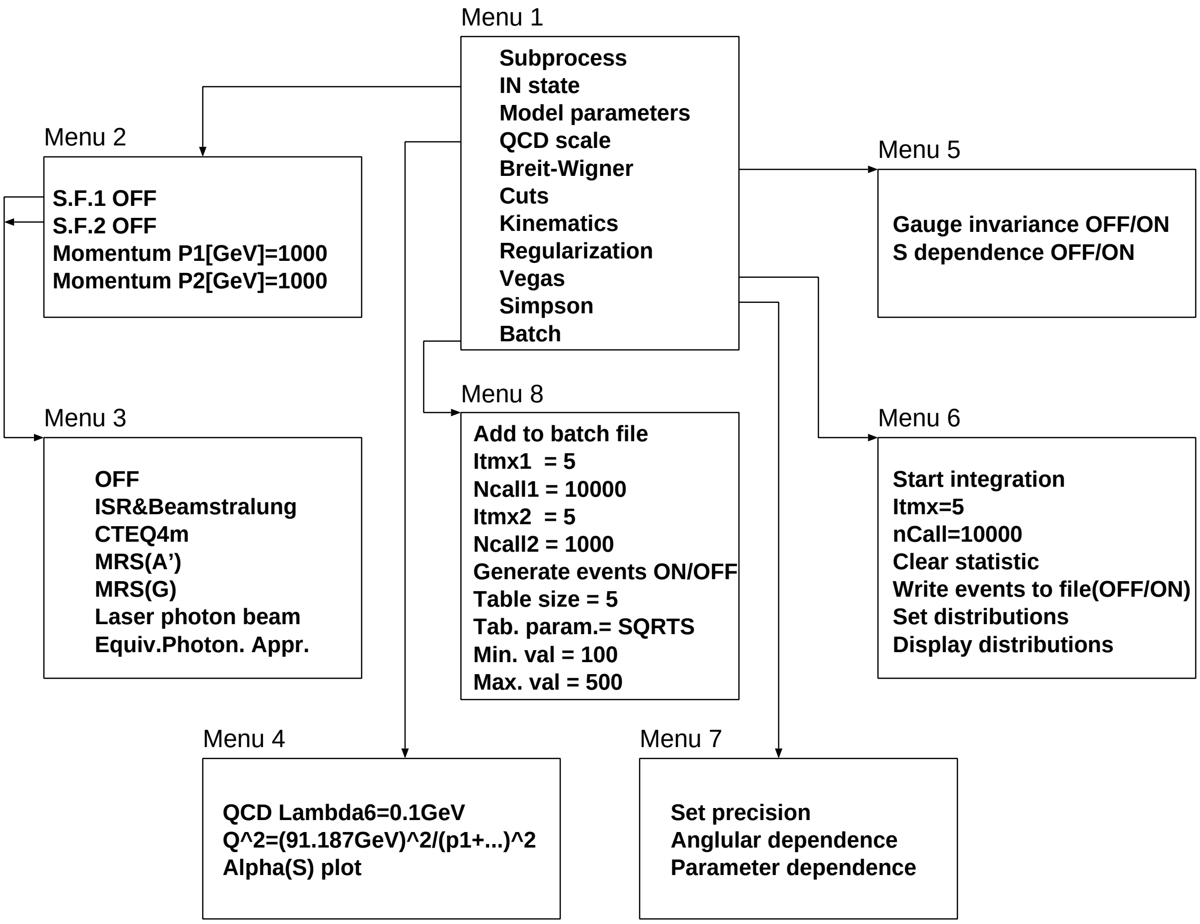



Figure 8: Menu scheme for the numerical session

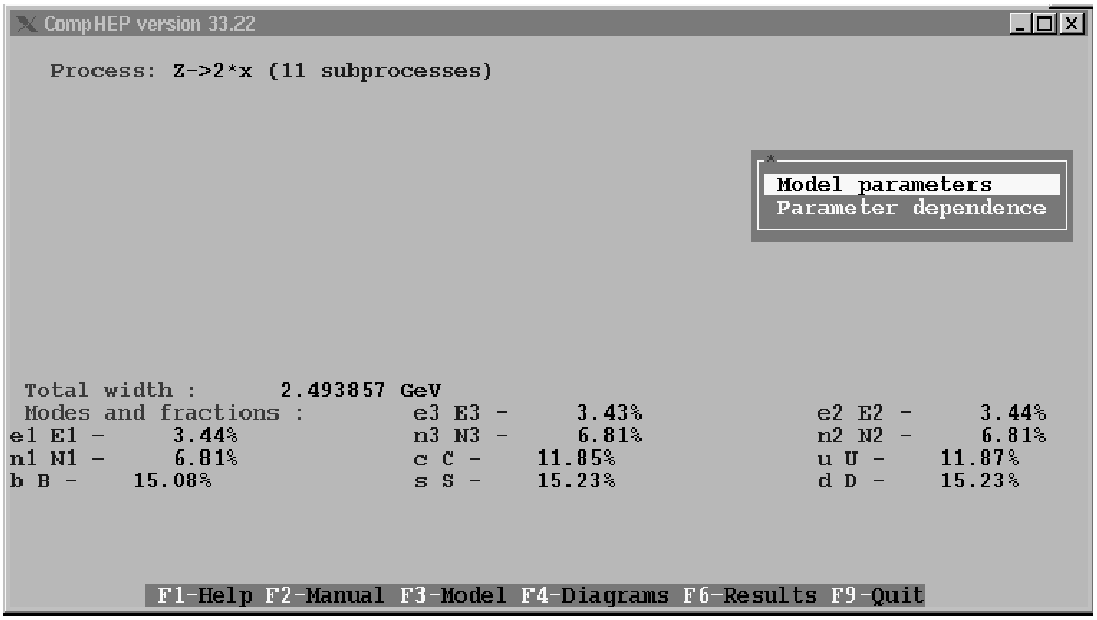



Figure 9: Representation of results for 1->2 processes

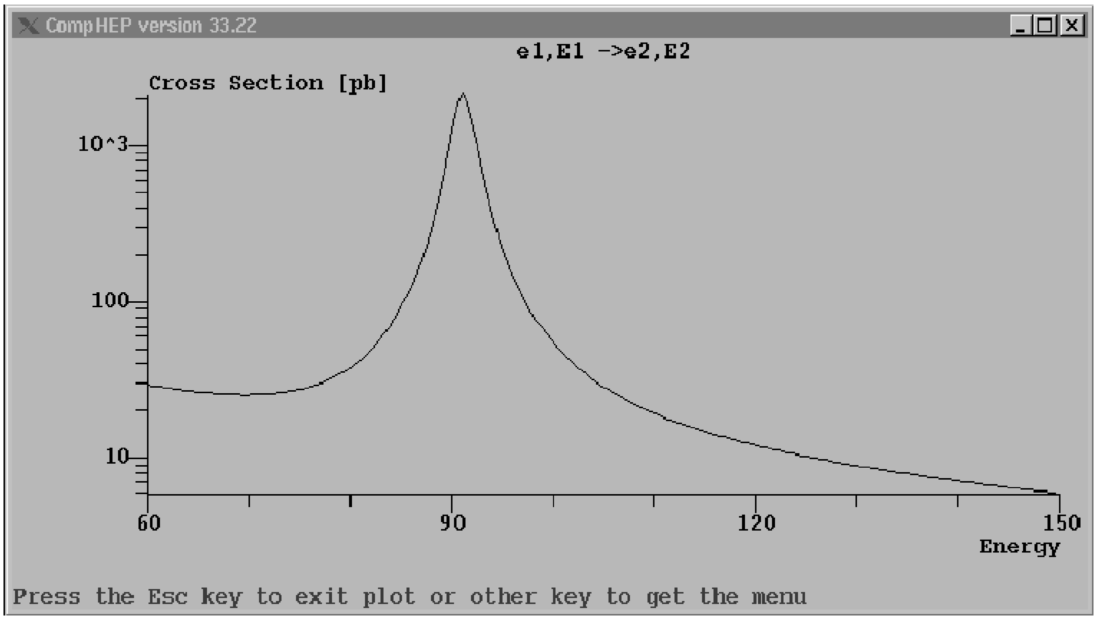



Figure 10: Plot for a 2->2 process

```
  e1         E2   !  E2          e1
==>==\     /==<====!==<==\     /==>==
  P1 |     |  P4   !  P4 |     |  P1
     |     |       !     |     |
  E1 |  A  |  e2   !  e2 |  A  |  E1
==<==@-1---@==>====!==>==@---2-@==<==
  P2    P5    P3   !  P3   -P6    P2
```

Figure 11: Example of pseudo-graphic diagram image

```
  e1    n1   !  n1    e1
==>==@==>====!==>===@==>==
     |       !      |
   W+|       !    W+|
     |       !      |
-----@--<----!--<---@-----
A       W-      W-       A
         a)
```

```
  e1    n1   !  n1    e1
==>==@==>====!==>===@==>=
     |       !      |
   W+|       !    W+|
     |       !      |
-----@--<----!--<---@-----
A      W-.f    W-.f      A
            b)
```

```
   e1    n1   !  n1    e1
 ==>==@==>====!==>===@==>==
      |       !      |
    W+|       !    W+|
      |       !      |
 --<--@--<----!--<---@--<--
A.C     W-.C    W-.C     A.C
            c)
```

```
   e1    n1   !  n1    e1
 ==>==@==>====!==>===@==>==
      |       !      |
    W+|       !    W+|
      |       !      |
 -->--@--<----!--<---@-->--
A.c     W-.c    W-.c     A.c
             d)
```

Figure 12: Ghost diagrams

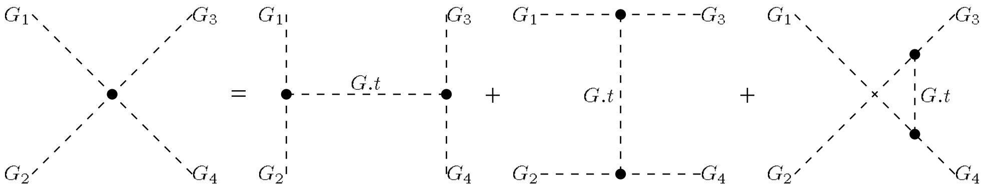



Figure 13: Splitting of four-gluon vertex

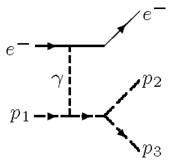



Figure 14: Example of process with the $1/t^2$ pole cancellation.